

Analyzing Dyadic Sequence Data

Comparing Different Statistical Models with Respect
to their Applicability and Interpretation

Peter Fuchs

A thesis presented for the degree of
Doctor of Natural Sciences

Department of Psychology
Bielefeld University

Germany

May 2, 2018

Contents

Preface	12
1. Introduction to Dyadic Sequence Data	17
1.1. Sequences	17
1.1.1. Definition of Sequences and States	17
1.1.2. Transitions	19
1.1.3. Metric of Time	20
1.1.4. Representation of Sequences and States	23
1.1.5. Transition-Plots and Properties of States	24
1.2. Dyadic Data	27
1.2.1. Classification of Dyads	27
1.2.2. Interdependency	29
1.2.3. Within and Between Variables	30
1.2.4. Conceptual Models for Dyadic Data	31
1.3. Combining Dyadic and Sequence Data	32
1.3.1. Conceptual Models for Dyadic Sequence Data	34
1.3.2. Dyadic Sequence Data in Psychology and Related Fields	35
2. Example Data Sets	38
2.1. Couples-Cope	38
2.1.1. The Couples-Cope sample	38
2.1.2. Stress Communication and Dyadic Coping	39
2.2. Give-Some	41
2.2.1. The Give-Some Sample	41
2.2.2. Cooperation in a Give-Some-Dilemma	42
2.2.3. The Give-Some Experiment	43
2.2.4. The Pre-Programmed Player	44

3. Research Questions and Corresponding Data Analyses	46
3.1. Getting an Overview (Visualization and Descriptives)	46
3.2. Duration of Behavior (Time-to-Event)	47
3.3. Assuming a Latent Dyadic Process (Common Fate)	48
3.4. Analyzing Promptness of Interaction (APIM)	49
3.5. Latent Groups or Clusters (Unobserved Heterogeneity)	51
4. First Steps in Analyzing Dyadic Sequence Data	55
4.1. Graphical Analysis	55
4.2. Association Between Behaviors	59
4.3. Inspecting Individual Cases	60
4.4. Finding Typical Subsequences	61
4.5. A Contrasting Example (Give-Some)	64
5. Dyadic Sequences in Time-to-Event Analysis	67
5.1. Time-to-Event Data	68
5.1.1. Transforming Sequences into Time-to-Event Data	70
5.1.2. Survival, Hazard, and Cumulative Hazard	72
5.2. Analyzing the Influence of Covariates	74
5.2.1. Time-Independent Covariates	74
5.2.2. Second Sequence as a Time-Dependent Covariate	76
5.3. Shared Frailty Model: A Bivariate Survival Model	77
5.4. Assumptions, Needed Sample Sizes, Practical Issues	79
6. Analyzing Dyadic Sequences Using Aggregated Logit Models (APIM)	82
6.1. State-Transition Tables	83
6.2. A Single Logit-Model for Single Case Analysis	84
6.3. Aggregating Results for Group Analysis	85
6.4. Actor-Partner-Interaction Model	87
6.5. Alternative Example (Give-Some)	87
6.6. Assumptions, Needed Sample Sizes, Practical Issues	89
7. Modelling Dyadic Sequences Using Multilevel-Models (APIM)	91
7.1. The General Idea of Multilevel-Modeling	91
7.2. Repeated Measures as Multilevel-Model	95
7.3. Generalized Multilevel Models	96
7.4. A Multilevel Model APIM	97

7.5.	Alternative Example (Give-some)	101
7.6.	Assumptions, Practical Issues, and Required Sample Size	102
7.7.	Alternative Multilevel Modeling Strategies	104
8.	Modelling Dyadic Sequences Using Markov Models	105
8.1.	Basic Markov Models	107
8.1.1.	Transition Matrix	109
8.2.	APIM as a basic MarkovModel	111
8.2.1.	Fitting a basic Markov model on the couples-cope data	111
8.2.2.	Converting Transitions into Actor and Partner Effects	112
8.3.	Hidden Markov Model	118
8.3.1.	Latent Hazard Model (Restricted Hidden Markov Model)	119
8.3.2.	Unrestricted HMM (Common Fate Like)	121
8.3.3.	Multi-Channel Approach (Pure Common Fate)	123
8.3.4.	Using the same Number of Latent States and Indicators	124
8.4.	Latent Groups: Mixture Markov Models	125
8.5.	Mixture Hidden Markov Models	129
8.6.	Which Markov Model is the Best?	131
8.7.	Practical Issues and Required Sample Size	133
9.	A Cluster Analytical Approach: OM-Distances	138
9.1.	OM-Distances for Sequence Data	140
9.2.	Algorithm	143
9.2.1.	k-Means Algorithm	143
9.2.2.	Ward’s Method (agglomerative hierarchical clustering)	145
9.2.3.	How to Choose a Cluster Solution	146
9.3.	Applying OM-Distances on the Couples-Cope DataSet	149
9.4.	Comparing Groups or Clusters of Sequences	150
9.4.1.	An ANOVA-Like Method for Sequence-Data	153
9.5.	Practical Issues and Required Sample Sizes	156
10.	Simulation studies	158
10.1.	Cox Regression	159
10.2.	Shared Frailty	163
10.3.	Aggregated Logit APIM	166
10.4.	Multilevel APIM	171
10.5.	Basic Markov APIM	175

10.6. Restricted Hidden Markov (Latent Hazard)	178
10.7. Unrestricted Hidden Markov	184
10.8. Mixture Markov	191
10.9. OM-Distances	197
10.10. ANOVA-like Approach for Comparing Subgroups	208
11. Summary and Concluding Comments	210
11.1. Findings on Dyadic Coping	211
11.2. Findings on Approaching Dyadic Sequence Data	213
11.3. Limitations and Outlook	217
A. Vignettes for R	219
A.1. Prerequisite Steps	219
A.2. Getting and Visualizing the Data	219
A.3. Analyzing Duration	221
A.3.1. Cox-Regression with Time-Independent Covariate	222
A.3.2. Cox-Regression with Time-Dependent Covariate	223
A.3.3. Shared Frailty Model	225
A.4. Aggregated Logit Model	226
A.5. Multilevel Model APIM	226
A.6. Markov Models	230
A.6.1. Basic Markov Model (MM)	230
A.6.2. Restricted Hidden Markov Model	231
A.6.3. Unrestricted HMM (common fate)	232
A.6.4. Multi-Channel Approach ("Pure Common Fate")	234
A.7. Mixture Markov	235
A.8. OM-Distances	235
B. R-Code for Simulation Studies	238
B.1. Cox-Regression:PowerAnalysis	238
B.2. Frailty Model	241
B.3. Power Simulation for Aggregated Logit Models	243
B.4. Interaction Delta and L on Type-IError	246
B.5. Multilevel APIM	246
B.6. Basic Markov Model APIM	248
B.7. Latent Hazard Model(Restricted Hidden Markov)	253
B.8. Hidden Markov (Correct Number of Latent States)	256

Contents

B.9. Mixture Markov (Correct Number of Latent States)	264
B.10. OM-Distances (Correct Number of Latent States)	278
C. The R-Package 'DySeq'	298

Formulary

Formel 5.1. Survival Function	72
Formel 5.2. Hazard Function	73
Formel 5.3. Cox-Regression in the Logit Notation	75
Formel 5.4. Cox-Regression in the Hazard-Ratio Notation	75
Formel 5.5. Cox-Regression for Time-Dependent Variables	77
Formel 5.6. Shared Frailty Model for Paired Observations	78
Formel 6.1. Predicted Logits of Showing DC	84
Formel 6.2. Predicted Odds of Showing DC	84
Formel 7.1. Generic Two-Level Multilevel Model	94
Formel 7.2. Multilevel Baseline Model (ICC)	95
Formel 7.3. Generic Logistic Multilevel Model	96
Formel 7.4. APIM as Logistic Multilevel Model	98
Formel 7.5. Conditional Level-1 Equations for MLM-APIM	98
Formel 8.1. Basic Markov Model	110
Formel 8.2. Basic Markov Model to APIM Conversion	117
Formel 8.3. Hidden Markov Model	118
Formel 8.4. Mixture Markov Model	126
Formel 8.5. Mixture Hidden Markov Model	131
Formel 9.1. TRATE-Formula	141
Formel 9.2. Silhouette Coefficient	149
Formel 9.3. Sum of Squares for ANOVA-like Group Comparison	155
Formel 10.1. Cohens Kappa	196

List of Tables

3.1.	Overview Research Questions and Related Models (Part A)	53
3.2.	Overview Research Questions and Related Models (Part B)	54
4.1.	State-Expand for SC and DC	56
6.1.	State-Transition Table for Couple ID 129	83
6.2.	Results of the Logit-Model for Couple ID 129	85
6.3.	Predicted Logits, Odds, and Probabilities for couple ID 129	86
6.4.	Averaged Logit Parameters Over all 64 Couples	86
6.5.	Averaged Logit Parameters for the Give-Some Example	88
7.1.	Fixed Effects for MLM-APIM (Couples-Cope Example)	100
7.2.	Random Effects for MLM-APIM (Couples-Cope Example)	100
7.3.	Multilevel Model Comparisons (Give-Some Example)	102
7.4.	Fixed Effects for MLM-APIM (Give-Some Example)	102
8.1.	Transition Matrix for the modified Give-Some Example	109
8.2.	Transition Matrix for the Give-Some Example	111
8.3.	Transition Matrix for the Couples-Cope Example	113
8.4.	Predicted Logits, Odds and Probabilities for DC by the Markov model	115
8.5.	Actor and Partner Effects for Markov-APIM (Couples-Cope)	115
8.6.	Hidden Markov Model for Modelling Latent Hazard	119
8.7.	Hidden Markov Model With 3 Latent States	123
8.8.	Two-Channel HMM With 3 Latent States	125
8.9.	Mixture Markov Model with 3 Latent Classes (MMM)	128
8.10.	Mixture Hidden Markov Model	132
9.1.	Distances Between German Cities	139
9.2.	Substitution-Cost Matrix for the Couples-Cope Data	142
9.3.	Averaged Logit Parameters for Showing DC by Clusters	152

9.4.	Transition Matrices for OM-Clusters	153
10.1.	Results for the Simulated Cox-Regressions (Part A)	161
10.2.	Results for the Simulated Cox-Regressions (Part B)	162
10.3.	Results for the Simulated Cox-Regressions (Part C)	164
10.4.	Simulated Frailty Models	165
10.5.	Aggregated Logit: Type-I Error Rates for Actor Effect	167
10.6.	MLM-APIM: Power-Analysis for Actor und Partner	174
10.7.	Basic Markov APIM: Power-Analysis for Actor und Partner	177
10.8.	Hidden Markov: Bias and SE for High Emissions	180
10.9.	Hidden Markov: Bias and SE for Low Emissions	181
10.10.	True Basic Markov Model	184
10.11.	True Hidden Markov Model with Two Latent States	186
10.12.	True Hidden Markov Model with Three Latent States	187
10.13.	Model Selection for Hidden Markov Simulation	189
10.14.	True Mixture Markov Model with 2 Latent Classes	198
10.15.	True Mixture Markov Model with 3 Latent Classes	199
10.16.	Model Selection for Mixture Markov Simulation	200
10.17.	Precisions of Estimates for Mixture Markov Simulation	201
10.18.	Post-Hoc Simulations for 3 laten classes	202
10.19.	Results for Simulation of OM-Distances (Part A)	205
10.20.	Results for Simulation of OM-Distances (Part B)	206
11.1.	Summary of Sample Size and Sequence Length Recommendations . .	216

List of Figures

1.1.	Examples for transition plots	25
1.2.	Comparison of APIM vs common fate model	33
1.3.	Example for interval sampling	34
4.1.	Entropy and State-Transitions for Couples-Cope Dataset	56
4.2.	State Distribution Plot of the Couples-Cope Example Data	57
4.3.	Multi-Channel State Distribution Plot (Couples-Cope)	58
4.4.	Entropy and State-Transitions for Couples-Cope	59
4.5.	Scatterplot Frequencies of DC vs SC (Couples-Cope)	60
4.6.	Individual Sequence Plots	62
4.7.	Most Common Subsequences for the Couples-Cope Dataset	63
4.8.	Visualization of Couples-Cope vs. Give-Some (A)	65
4.9.	Visualization of Couples-Cope vs. Give-Some (B)	66
5.1.	Transforming Sequences into Time-to-Event Variables	70
5.2.	Survival, Hazard, and Cumulated Hazard (Couples-Cope)	73
5.3.	Predicted Hazard for Different Values on a Covariate	76
5.4.	Frailty Model for Paired Observations	79
5.5.	Simulated Power and Type-I-Error for the Cox Model	80
6.1.	APIM Estimated From Aggregated Logit-Models	87
7.1.	Example of a Nested Data Structure	92
8.1.	Comparison of MM, HMM, MHMM and MMM.	106
8.2.	Basic Markov Model for the Modified Give-Some Example.	108
8.3.	Basic Markov Model for the Couples-Cope Example.	111
8.4.	Transition plot for the three states hidden Markov model	122
8.5.	State Distribution for Three Latent Classes	129
9.1.	Illustration of k-means algorithm	144

List of Figures

9.2.	Illustration of Ward’s method	146
9.3.	Dendrogram for German Cities	147
9.4.	Scree Plot for German Cities	148
9.5.	Scree Plot for the Couples-Cope Dataset	150
9.6.	State-Distribution Plot for the two Couples-Cope Clusters	151
10.1.	Bias and Type-I Error (Aggregated Logit Model)	168
10.2.	Bias for Actor Effect (Aggregated Logit Model)	170
10.3.	Power Simulation for Aggregated Logit Model	171
10.4.	Bias for Hazard and Emissions (Latent Hazard Model)	182
10.5.	Distribution of latent Hazard-Estimates	183
10.6.	Biases for the 2-latent-state Model	192
10.7.	Biases for the 3-latent-state Model	193
10.8.	Detailed Bias for Emissions (3 Latent States)	194
10.9.	Simulated Scree Plots	207
10.10.	Power Analysis for ANOVA-like Group Comparison	209

Preface

The investigation of social dynamics is one of the most fascinating aspects of empirical psychological research. The smallest unit, in which such social interactions can occur, is a *group* of two persons, a so-called dyad. This monograph explores and compares several statistical modeling techniques for dyadic interactions in which the dependent variable is categorical. This is done by linking each presented model to prototypical research question that might arise in psychological research, applying it to an example datasets, and then translating the results back into a psychological context. Finally, simulations studies are conducted so that recommendations on sample size can be given.

Dyadic social dynamics can be found in almost any aspect of life; merchant and customer must interact with each other to make a deal; coworker may need to interact with each other for problem-solving; or parents must interact with their children so that they feel loved and learn about the world. Another example which is used as the main example in this monograph is, how romantically involved couples react toward each other after one of them encountered a stressful event.

For this example, an empirical dataset is provided that describes whether and when the stressed partner communicated his or her stress, and also whether and when the other partner reacts with a dyadic coping response. That is, the unstressed partner responds with either positive or negative coping strategies in an attempt to alleviate his or her partner's feelings of stress.

Several behavior-patterns are conceivable: the stressed partner might communicate stress directly after stress was induced or only after some time. Each behavior might have different consequences, e.g., the former might be more likely to trigger a quick and continued response by the other partner. Which again might encourage the stressed partner to talk about the stressful event. However, if the partner shows delayed stress communication, it might be misinterpreted as an indicator that the stress is not very strong or important. Therefore, the partner may show dyadic coping only intermittently, which again discourages the stressed partner from talking about the event.

As this example shows, social dynamics are not static but rather evolve over time. Using such data allows for an understanding of the evolution of a certain behavior across time. Because of that, statistical models for analyzing longitudinal dyadic data are needed for investigating dyadic dynamics. Kenny, Kashy and Cook (2006) provide an extensive overview for models in which the dependent variable is at least interval scaled. However, the case for a categorically dependent variable, such as “was stress communication shown or not at a certain time interval?” is covered only briefly in their monograph. Screening through methodological papers and textbooks revealed an absence of such models, at least in the field of psychology.

Therefore, this monograph aims to close this gap by providing an overview for researchers who encounter data about dyadic interactions with categorically dependent variables – so-called dyadic sequence data. To this end, an introduction to sequences, dyads, and dyadic sequences (see Chapter 1) are provided. The main part of this monograph, however, presents and compares different statistical models which can be applied to dyadic sequence data. Some of which stem from within but the majority stems from outside of the traditional field of psychology. For example, OM-Distances are mainly used in sociology, but a few studies in the field of psychology were found that use OM-Distances (some of the few are Wuerker, 1996a and Wuerker, 1996b). Another example is Hidden Markov Models which are not often used in psychology but for which many textbooks in the fields of biostatistics and econometrics exist. Thus, Hidden Markov Models are often presented in the context and language of those fields. Because psychology researchers are often unfamiliar with those fields, they might overlook whether those models fit their research questions, and stick to other, more familiar but also less adequate models. Hence, the monograph focuses on showing how the presented models might be integrated into psychological research. This is done by linking a prototypical research question to each presented model, and by then translating the results for the example datasets back into a psychological context.

For some research questions, more than one model can be applied adequately to answer it, and researchers might want to know which one to choose for their research question. The answer to that question depends on the sample size, the number of observed time intervals, and the expected effect size. Yet, one obstacle for comparing the models is that models come in different notations and produce different kinds of estimates as output. For example, aggregated Logit-Models estimate beta-coefficients, whereas Markov Models produce transition probabilities. Because of that, it is difficult to compare those models based only on existing simulation studies. Hence, models will be presented in a unified notation, which also allows one to introduce a conversion

formula for those estimates. Moreover, simulations are done for all presented models so that recommendation for sample size and number of time intervals are comparable across models that are used for answering the same research questions.

All R-code for using the presented models in practice and for reproducing the simulations is provided in the Appendix. Yet some models were not implemented in R before this monograph was written, so an R-Package, called DySeq, was created and is now provided via CRAN. The original code of the package, as it was used in this monograph, is also included in the appendix as well.

Due to the pace of development and breadth of research, a truly comprehensive collection and comparison of all models, which can be applied to dyadic sequence data, is certainly beyond the scope of this monograph. For example, hundreds of time-to-event models exist, and most of them can be adapted to dyadic sequence data, but this monograph illustrates only two of them. The selection might be influenced by the authors' personal research interests and expertise to some degree, but tally far more, so those models were chosen which seem – to the author's best knowledge and judgment – most useful for psychology research and at the same time most prototypical for their type of application.

An additional limitation is that for most parts of this monograph only examples with dichotomous dependent variables are used, yet it is indicated whether and how a certain model can be generalized for dependent variables with more than two levels.

This monograph does not introduce new models, but rather re-frames or modifies existing models in a new way so that they can be used for investigating dyadic sequence data. For example, hidden Markov Models have existed for a relatively long time. But restricting them in a certain way allows one to interpret them as a time-to-event model for dyadic sequence data. Hence, the novelty of this monograph lies not in the presented models themselves, but rather how they are put into context: this monograph translates statistical models in research questions, allowing a researcher who engages dyadic sequence data to select potential statistical models guided by her or his research questions. Another novelty is that some of the models are compared for the first time. For example, Markov-Models and aggregated Logit-Models have existed for a relatively long time now, and both can be used for answering the same research question. However, to this monograph's author's knowledge, those two models have never been compared before. By introducing a conversion formula (see Equation 8.2, it is also possible to translate those models' estimates into each other, which makes recommendations for statistical power comparable.

Finally, the author hopes that this monograph will encourage other researchers in the field of psychology to engage research questions about social dynamics more often and to not shy away from using statistical models from outside of their primary domain.

Overview

This monograph consists mainly of two parts. The first part provides the background for the rest of this monograph. That includes Chapter 1, which provides an introduction to sequences, dyadic data, and dyadic sequence data. Chapter 2 shows and discusses two example datasets of dyadic sequence data. These example datasets are then used in Chapter 3 for deriving research questions that often arise in the analysis of dyadic interactions and for linking those to the corresponding statistical models. Chapter 3 also includes a table that shows which model can be used for which research question, and where to find the description, the simulations, and the R-code for that model.

The second part of this monograph describes statistical models and routines. Chapter 4 provides basic descriptive statistics and an overview of data-visualization methods for dyadic sequence data. The approaches shown in this chapter are especially useful for getting a first inside look into a dataset or for exploratory research. In the following sections statistical models are presented according to each chapters name. Each model is introduced in light of the research questions of Chapter 3. After that, the models are applied on the example data for showing how to interpret the results in a given research context. Finally, recommendations on sample size and the number of observation intervals are given. Those recommendations are based on simulation studies, which can be found at Chapter 10.

The R-code for reproducing the simulations can be found at Appendix B. An R-Package, called *DySeq*, exists as part of this monograph (see Appendix C), which includes the example data and the implementations of all presented models that were not implemented before this monograph was started. For all models that were already implemented, Vignettes can be found in Appendix A.

1. Introduction to Dyadic Sequence Data

Many research questions in psychology concern social interactions, which take place between two or more persons. If data is collected about the interaction of two individuals, the data is said to be dyadic data. Most social interactions cannot be observed at a single point in time, but rather develop over a given period of time. Hence, observations of social interaction often produce longitudinal or so-called time-series data. Moreover, if the observed variable is also categorical, the data is often referred to as sequence data. *Dyadic sequence data* is the combination of all three of the above, and therefore describes categorically coded social interactions of pairs over time. This chapter provides definitions, special terms, and examples of sequence data, dyadic data, and the combination of both.

1.1. Sequences

Generally speaking, the term *sequence* can refer to any arrangement of items that follows a certain order. For example, a series of scenes in a movie is often called a sequence because there is a number of items (scenes) that are arranged in a certain order (e.g., chronologically). However, this contribution focuses on applications to social sciences, in which the majority of sequences describe *transitions* between *states* over time. To this end, the general definition has to be narrowed down correspondingly.

1.1.1. Definition of Sequences and States

Throughout this monograph, the following definition by Helske and Helske (2016) will be used: "By the term sequence we refer to an ordered set of categorical states" (p.3). The relevant terms of their definition are *states*, *ordered set*, and *categorical*, all of which will be elaborated on the following.

A *state* is a "particular condition that someone or something is in at a specific time" (State, 2016), which implies that states are temporary. For example, a person can be in a state of being ill, yet the state might change in the future, and the person will

hopefully get well again. The term *ordered set* implies that these states are arranged in a certain order (e.g., chronologically). This order determines the interpretation of so-called *transitions*, which are discussed in the following chapter.

Furthermore, the definition by Helske and Helske (2016) states that the measurement level is *categorical*. Agresti (2002) defines a categorical variable by the following:

"A categorical variable has a measurement scale consisting of a set of categories. For instance, political philosophy is often measured as liberal, moderate, or conservative. Diagnoses regarding breast cancer based on a mammogram use the categories normal, benign, probably benign, suspicious, and malignant." (p.1)

Moreover, Agresti (2002) distinguishes between nominal categorical variables (variables without natural ordering), such as religious affiliation (Catholic, Protestant, Jewish, Muslim, other), favorite type of music (e.g., classical, country, folk, jazz, rock), and ordinal categorical variables (variables with natural ordering) such as social class (upper, middle, lower) or patient political philosophy (liberal, moderate, conservative).

All models presented in this monograph can handle both nominal and ordinal variables as long as the number of categories is small. A negative example would be recording subjects' employment with hundreds of possible categories (e.g., cook, janitor, mechanic, and the like). By contrast, a positive example would be a type of employment measured by the following categories: unemployment, employee, contractor, or self-employed. The line between an appropriate number and too many categories may depend on the actual statistical model. Therefore, each introduction to a statistical model in this monograph will include a recommendation on this matter.

A special case that is often used in psychological research is the use of single rating items with only a few categories, such as *strongly disagree*, *disagree*, *neither agree nor disagree*, *agree*, or *strongly agree*. Such items are often treated as metric variables in social sciences even though they are clearly categorical according to Agresti's definition. The reason for this is that these items can be used as both categorical and scaled (metric) variables if the distance between each category value is the same (equidistance assumption), and if categories are symmetrically ordered around a neutral value (symmetry assumption). Items that meet these assumptions are called *Likert* items (Burns and Bush, 2007). However, even if those preconditions cannot be assumed, the induced bias will be small as long as the variables' distribution is at least symmetrical and five or more categories are used (Johnson and Creech, 1983; Bollen and Barb, 1981). Because of that, rating items are often treated solely as metric variables. Yet it

is important to remember that these items share the characteristics of both scaled and categorical items, and therefore can be used as states in a sequence.

If states are measured on an interval scale or higher, sequence analysis is not the recommended analytical strategy, and other methods might be more appropriate. For example, Cockerill et al. (1991) used multiple regression for modeling states on a continuous scale; Steyer et al. (1999) used structural equation modeling for decomposing continuous scores' variances into variances of states, trait, and measurement error.

1.1.2. Transitions

Transitions are considered crucial in sequence analysis. Kenny et al. (2006) define them as the following:

"A transition is an observation of two temporally separated observations in a system over time. In time-sampling designs, it is usually the observation from one epoch to the next." (p. 386)

For clarification, *Epoch* is a synonym for a time interval. For instance, if time is measured in 10-second intervals, the observation from the first 10-second interval to the second interval is a transition. The important part of this definition is "observation [...] over time". Thus, sequences should represent states *and* time. Bringing this together with the former definition of sequences as an "ordered set of [...] states" shows that the order of a sequence should represent time.

In the following example, health states of a person are coded once a week. The person is healthy in the first week, then transitions to a state of being ill in the second, and after that, transitions into being healthy again in the third week. Ordering the states chronologically results in the following sequence *healthy-ill-healthy*, which describes a total of two transitions. The order represents time because each element of the sequence refers to one particular time interval, and transitions describe whether and when states have changed over time. Hence, two sequences with equal states but different order may describe different processes. For example, the sequence *healthy-healthy-ill* indicates that someone is healthy at the beginning and gets ill at the end, whereas the sequence *ill-healthy-healthy* indicates that someone is ill in the beginning and becomes well again in the second week.

Chronological ordering is very common in social sciences; thus, sequence data is often referred to as *longitudinal categorical data* or *categorical time-series*. Both terms are often used synonymously. However, the former is more frequently used in the case of

many short sequences, and the latter is more often used in the case of few, but long sequences (Bartolucci et al., 2012). This differentiation will be adapted throughout this monograph because it is useful for model selection. One and the same model might require different estimation approaches, and therefore different R-Packages, depending on whether the sequences represent longitudinal categorical data or categorical time-series data. The former may consist of any number of time intervals greater than one, but needs many observations, and the latter may consist of any number of observational units (including one), but needs many time intervals.

In contrast to the previous definition by Kenny et al. (2006), states can be ordered by principles other than time. For example, genes within a gene-sequence are ordered by locus. Or, as investigated by Levitt and Nass (1989), subjects in textbooks are ordered by chapters. Because chronological order is the most common - and for the sake of simplicity - this monograph assumes that all sequences are ordered by time unless noted otherwise.

1.1.3. Metric of Time

Time can be conceptualized in different ways, and most textbooks (Hosmer and Lemeshow, 1999; Mills, 2011; Aalen et al., 2008) distinguish between *continuous time* and *discrete time*. Some statistical models in this monograph assume the former (Cox-Regression; Clayton-Oak Model), most assume the latter (Optimal Matching; Markov Model; Hidden Markov Model; Mixture Markov Models). Yet the distinction between those two metrics is not always easy, and sometimes arbitrary.

A common form of collecting observational data in psychology is interval sampling (Kenny et al., 2006), which measures time in a discrete form. This means that the observation is sliced into time intervals of equal lengths, and for each interval, it is coded if a person shows a certain behavior. This kind of sampling naturally results in sequences with chronological order. Each state refers to a certain interval of time in ascending order: the first state describes whether the behavior was shown in the first time interval, the second state describes the same for the second time interval and so on. This type of time metric is often considered to be discrete because each of a sequence's elements refers to an interval rather than continuously recorded points in time.

The problem about measuring time continuously is that even the smallest time unit represents a time interval. And even if we knew the exact milisecond, one milisecond again would be a time interval of 1000 microseconds. If something happens in the

3rd second of an observation, we only know that the event happened somewhere between the 2001th millisecond and the 3000th millisecond. And even if we would know the exact millisecond, one millisecond would again be a time interval of 1000 microseconds. This can be continued endlessly because time units can be split into smaller units infinitely. Singer and Willett (2003) provided a definition that takes this fact into account, and defined the metric of time as follows:

"We distinguish between data recorded in thin precise units and those recorded in thicker intervals by calling the former *continuous time* and the latter *discrete time*." (p.313)

Therefore, whether a sequence represents a continuous or a discrete metric of time depends on the interval's length and the context. Consider the following example: treated alcoholics are being observed, and it is coded whether someone is in a state of craving alcohol, in a state without craving, or in a state of an actual relapse. Coding the state for every second would fit Singer and Willett's (2003) definition of continuous time. However, coding it only once a month would be considered discrete, whereas coding every day might be considered a continuous measurement by most but not all researchers. Because of that, the decision whether a certain sequence represents continuous time or discrete time is often a subjective one. Hence, the metric of time itself should be seen as a continuum: a certain sequence might represent time as more discrete or as more continuous, or as something in-between, but it is never either completely discrete or continuous.

Most statistical models that are presented in this monograph assume either discrete or continuous time. However, most of them can deal with both in practice. Hence, the main question should not be whether time is measured discretely or continuously, but whether it is measured appropriately. Regarding the previous example, it would be impractical to code the alcoholics' state every second; conversely, if their state was coded only once a month, it would be very likely for the researcher to miss important information. This is particularly the case if she or he is interested in finding covariates that can predict craving and relapse. Too many things can happen and change multiple times within one month that may affect relapse and craving. Because of that, inference would be impossible. In this case, a daily or weekly coding might be more appropriate. But again, this depends on the study's purpose: if, for example, a researcher just wants to compare long-term craving-relapse patterns, monthly coding might be the right choice.

Some attention must be paid to the fact that if time is measured in intervals, these intervals can be *regular* or *irregular* (Crowder, 2012). The previous example described regular time intervals, which means that at some point in time the coding starts and each following time interval has a fixed length (e.g., every day at a certain time the actual state will be coded). An example of irregular intervals are trials in a visual attention experiment, in which correct answers and different types of errors are coded as states. Typically, the length of one trial is determined by the participant's reaction time. For example, participant A's second time interval may refer to seconds 3 to 7 because he or she needed three seconds for the first trial and four seconds for the second trial. However, participant B's second time interval may refer to seconds 10 to 12 because he or she needed ten seconds for the first trial and two for the second. The trials are still ordered in a chronological way because the first trial will take place before the second and so on. Yet, contrary to regular intervals, trials may refer to time intervals of different lengths via starting and ending points. If time intervals are irregular, interpretation of statistical analysis should reflect this fact. For example, if a trend can be found in this example, it should not be interpreted as a function of time, but as a function of trials.

Another requirement is that the sequence's indices should be *synchronous*. That is, the first element of a sequence should describe the same ordering unit (e.g., time unit, trials) for all observational units. For example, if time units are trials, the first element of each sequence should refer to the first trial, the second element to the second trial, and so on. Otherwise, interpretation might be compromised. This requirement seems obvious at first glance, yet it becomes important when missing data is involved. If for any reason the first time interval or trial was not observed, the first sequence element should be marked as missing. The same applies for all following intervals. Hence, the length of sequences remains the same no matter whether observations are missing or not. Let us consider health states as an example again: at the beginning of the study, one person does not show up in week one. In week two, the person is healthy, and after that, in week three, the person is healthy again. The resulting sequence should be *Missing Observation - Healthy - Healthy*. It still contains three elements, even though one observation is missing. The majority of models in this monograph can be applied to sequences without a synchronous index, but results will often not be interpretable. Thus, the focus of this monograph is on sequences with regular intervals and a synchronous index. As a consequence, all example datasets that are used throughout this contribution have a synchronous index.

1.1.4. Representation of Sequences and States

According to Gabadinho et al. (2009), the most common and intuitive representation is the *state-sequence* format. That is, states are hyphenated, and each hyphen symbolizes a transition. Take the following example 1-2-2-1: an observational unit starts in a state of 1, then transits into a state of 2, stays in a state of 2 for one unit of the ordering variable (e.g., time interval), and then transits back to a state of 1. This representation will be used for illustrating examples in this contribution. Alternatively, sequences can be represented by a vector e.g., $(1, 2, 2, 1)$ allowing for representing multiple sequences as one data matrix. The latter representation will be used for introducing model equations.

Throughout this monograph, the state variable is denoted S when used as a random variable. Specific singular states and their realizations are usually denoted as s . Their order is indexed by t . If states are ordered by time, t denotes the actual time interval (realization of time), and T is the corresponding random variable. Indexing starts with $t = 1$ for the first observed time interval. Thus, $t = 0$ refers to the immediate time interval before the observation started. The maximum number of observations is denoted m . Consequently, the former sequence $(1, 2, 2, 1)$ can be rewritten as $s_1 = 1, s_2 = 2, s_3 = 2, s_4 = 1$. The general notation for a single sequence is $(s_t)_{t=1}^m = (s_1, s_2, s_3, \dots, s_m)$. If all sequences have the same length, this length will be denoted L .

The transition from one state to another state is indicated by an arrow. The following denotes the transition that takes place by increasing the ordering unit by one (e.g., the transition from one-time interval to another time interval): $s_t \rightarrow s_{t+1}$. For example, the probability of transitioning from a state of 1 to a state of 2 between two time intervals is denoted $P(1_t \rightarrow 2_{t+1})$.

In psychology research, typically more than one observational unit is observed; hence, an additional index i is used that refers to observational units. In the case of several observational units, sequences are denoted $(s_{it})_{t=1}^m$. Thus, referring to the first observational unit can be denoted as $(s_{11}, s_{12}, s_{13}, \dots, s_{1m})$. The number of observational units is denoted with N .

Furthermore, it is possible that several sequences are nested within one observational unit. This is sometimes referred to as *multi channel sequences*. In this case, the sequences within an observational unit are indexed with j . For example, $(s_{ijt})_{t=1}^m = (s_{211}, s_{212}, s_{213}, \dots, s_{21m})$ denotes the first sequence ($j = 1$) within the second observational unit ($i = 2$). The number of sequences nested in one observational unit is denoted k .

Hence, in a dataset with an equal number of sequences nested per observational unit $N * k$ is the number of sequences in that dataset.

Nested sequences occur in particular when more than one variable of interest is coded per observational unit. For example, 1 indicates the state of being single and 2 of being in a relationship. At the same time, the same person might be in a state of having a job (state A) or not having a job (state B). Thus, the sequence for having a job could be A-A-B-B and the sequence for the relationship could be 1-2-2-1. These are two different sequences because they describe different properties. But they are nested together because they belong to the same observational unit, and they share the same time index.

Such sequences, which contain variables for the same observational unit and follow the same order, but differ regarding the states of interest, can be combined into one by a procedure referred to as state-expand (Vermunt, 1997). State-expand means that for every possible combination of states, a new state is defined. If the possible states for the relationship sequence are 1 and 2, and for the employment sequence are A and B, the four possible combinations are 1A, 2A, 1B, 2B. For example, 1A represents a state of being single and having a job, whereas 2B describes a state of being in a relationship and having no job. Thus, the combined sequence for the two sequences A-A-B-B and 1-2-2-1 is 1A-2A-2B-1B. Combining sequences is especially useful for plotting nested sequence data in one plot, or for applying statistical models or procedures that do not assume a nested data structure.

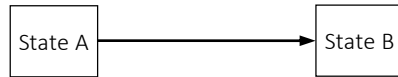
Further representations include graphical representations, which range from using colored blocks instead of hyphenated strings over state-distribution plots to transition-plots. The first two are especially useful for screening sequence data, and therefore will be discussed further in Chapter 4. Transition-plots will be explained in detail later in Chapter 8 because they are often used for plotting the results of a fitted Markov model.

However, transition-plots can also be used to visualize which transitions are in general possible and which are not. Depending on that, states can have different properties. Because of that, simplified versions of transition-plots will be used for demonstrating different properties of states in the following chapter.

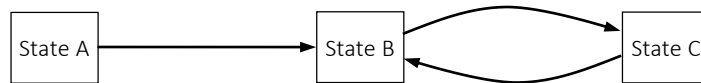
1.1.5. Transition-Plots and Properties of States

Figure 1.1 shows five example models which feature states with different properties. Every square represents one distinct state. An arrow from one state A to another state B indicates that transitions from A to B are possible. If the probability for a

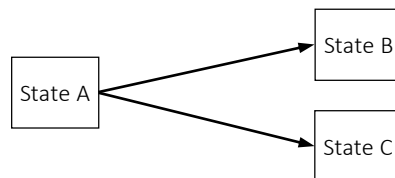
a) Model with an Absorbing State B



b) Model with a Closed Communicating Class



c) Model with two Competing States



d) Model with a Single Communicating Class



e) Model with a Communicating Class and an Absorbing State

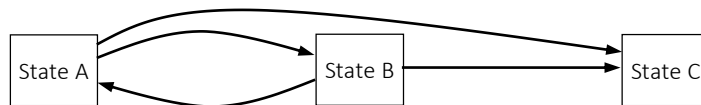


Figure 1.1.: Examples for transition plots featuring states and sequences with different properties

transition is known (or estimated), it is often attached to the corresponding arrow in the same way as in path analysis or structural equation modeling. However, for this chapter, an arrow simply means that the probability is greater than zero. Thus, if a transition from state B to state A is not possible, as seen in 1.1.a, no arrow is drawn from B to A. In other terms, an absent arrow means that the corresponding transition probability is zero. For example, it is possible to change from a state of *being alive* to a state of *being dead*, but not vice versa. Thus, an arrow should be drawn from *being alive* to *being dead* but not from *being dead* to *being alive*.

A state B is said to be *accessible* from another state A if it is possible to transition from state A to state B directly or indirectly. In 1.1.b, the state C is indirectly accessible from state A: an observational unit cannot transit within one time interval from A to C; yet it can transition from A to B first, and then transition to C in the next step. 1.1.a is an example where the State B is directly accessible from state A, but State A is not accessible from state B at all.

For example, an observational unit such as a human can be alive and then die. If this happens, it is said that the observational unit leaves the state of being alive and transitions to a state of being dead. Thus, the state of being dead is accessible from a state of being alive. However, if a human dies, he or she cannot become alive again. Therefore the state of being alive is not accessible from a state of being dead.

In fact, being dead is an *absorbing state*, meaning that once an observational unit enters this state, it cannot leave it. Therefore, the event of a transition from one state to an absorbing state can only happen once per observational unit (e.g., a person can die only once). Because of that, sequences with absorbing states can easily be transformed into time-to-event data, also known as survival data, by coding whether and when an observational unit has transitioned to the absorbing state (see 5.1).

1.1.c also describes sequences that can be transformed into time-to-event data, but in this case, into another subtype of so-called *competing events* (or competing risk). That is, more than one possible event can happen, but only one of them can happen per observational unit. For example, patients with a terminal illness may be alive (state A), but two different causes of death exist. For instance, a person can die because of the illness (state B) or because of liver failure due to the medication (state C). Thus, the competing events would be "death from disease" and "death from liver failure".

1.1.d shows a model in which state A is accessible from state B, and vice versa. If a set of states can access each other, it is called a *communicating class*. If a model consists only of one communicating class, it is called irreducible. In event-to-time analysis,

such sequences are often referred to as *recurrent events* (Mills, 2011). For example, someone can be healthy (state A), then ill (state B), and then get well again (state A).

1.1.b shows a model where it is possible to start in a state A, but once someone transitions to state B, it is not possible to leave the communicating class consisting of state B and C. That is, it is possible to change from state B to state C, and vice versa, but it is not possible to get back to A. A communicating class that cannot be left once entered is called a *closed communicating class*. An example of this could be the following: the Alcoholics Anonymous believe that someone can be in a state of "being not an alcoholic" (state A). But, once a person becomes an "alcoholic" (state B), a cure (transition to A) is impossible (Alcoholics Anonymous, 2002, pp. 30-33). However, it is possible to become an *abstinent alcoholic* (state C), which is a person that does not drink alcohol, but is always in danger of relapsing. The relapse, in this case, would be a transition from C to B (the alcoholic stops being abstinent).

1.1.e shows a communicating class that is not closed. In this example, it is possible to transition from A to B, and then back to A. Therefore, they form a communicating class. The difference regarding the previous example is that it is possible to leave this communicating class by transitioning either from state A or B to state C. An example for such a model in real life could be that a person is healthy (A) and might become ill (transition to B). An ill person can get well again (transition to A) or die (transition to C). If she or he dies, she or he cannot become healthy or ill again. Less likely, but also possible is that a person is in a healthy state (A) and dies from one time interval to the next (transit to C).

1.2. Dyadic Data

In psychology, many research questions and theories concern the psychological mechanisms or dynamics in social interactions. The smallest unit in which social interactions are observable is a dyad. A dyad is "[s]omething that consists of two elements or parts." (State, 2016). For instance, a married couple is a dyad because it consists of two partners. Other examples are siblings, best friends, roommates, two trading partners, or opponents in a game of chess.

1.2.1. Classification of Dyads

Dyads can be categorized by *type*, *distinguishability*, and *linkage*. The specification of the type of dyad depends on the roles of the two dyad members (e.g., mother-child dyads,

heterosexual or homosexual couples). Thus, this classification's purpose is solely a descriptive one.

Moreover, dyads can be considered distinguishable or indistinguishable. In distinguishable dyads, there is at least one relevant quality (role) of the two members, which allows for a clear distinction between the two (e.g., mothers and daughters). In indistinguishable dyads, there is no such quality (role) that may differentiate between the two members (e.g., monozygotic twins). The choice of the statistical model for the analysis of dyadic data strongly depends on the distinguishability of the two partners (for an overview, see Kenny et al., 2006). In this contribution, we focus on distinguishable dyads (e.g., heterosexual couples).

The distinction between distinguishable and indistinguishable dyads is not always easy. A dyad of two mutual best friends can be "made" distinguishable if any variable can be found that allows for an unambiguous distinction. For example, mutual best friends can be distinguished by "the older one" or "the younger one". Or else in the case of coworkers, the variable can be "the one with longer working experience" and "the one with shorter working experience". But as a result, the focus of possible interpretations will be shifted. Hence, a researcher should refrain from such practices unless they serve the intended research question (e.g., "does work experience account for differences within coworker-dyads?").

Kenny et al. (2006) point out that dyads can be distinguished even further by their type of linkage. Whereas this classification is often irrelevant for statistical model selection, it is useful for their interpretation. Linkage describes how two individuals originally became dyads. Thus, it gives the first indication of why members of a dyad are similar (or dissimilar). Four types of linkage exist: *voluntary*, *kinship*, *experimental*, and *yoke*.

Voluntary linkage means that both members of the dyad freely chose to be paired. For instance, mutual best friends, married couples (at least in most cultures), or a language tandem. Typically, members of such dyads share traits or interests because something must have brought them together. For example, mutual best friends often share similar hobbies, personality traits, or moral concepts even before they knew each other; or for instance, members of a language tandem share the interest in learning or improving a language.

Linkage by kinship means dyad members are paired by lineage (e.g., siblings). Members of such dyads may be similar due to genetic factors and shared environment as well. However, it is also possible to assume the opposite in some cases. Especially in

adolescence children often oppose their parents' beliefs and habits, and try to become *different*.

Experimental linkage refers to dyads that are paired artificially for the sake of research. This is often done by randomization. However, a researcher could also be interested in knowing whether similar persons cooperate more than dissimilar, and choose to manipulate similarity between dyad members (e.g., putting only very similar or very dissimilar persons together). This type of linkage gives the researcher a lot of experimental control. It is also the most artificial one because dyads are analyzed that would possibly never have existed otherwise. Therefore, generalization on *real world dyads* should be done only with caution.

Finally, *yoked linkage* means that two observational units are linked to each other in a way that they receive the same stimuli, but do not interact with each other. probably the most prominent example for yoked linkage stems from Seligman's early research on learned helplessness (Seligman, 1972). Dogs of one group were given electric shocks at random times. The shock could be avoided by pressing a lever. Each dog of this group was paired with another dog from a second group that was given a shock at the same time. Intensity and duration were also matched. However, the second dog's lever did not affect the shock at all. Thus, both dogs of one pair received the same environmental stimuli, but neither interacted with each other. The only difference between two linked dogs was that one dog could escape the shock by his own actions, while the other dog could only wait until the shock ended.

1.2.2. Interdependency

The most important characteristic of dyads in social sciences is interdependency. For example, a mother and her child are clearly interdependent: if the mood of the child changes, it will affect the mood of the mother and vice versa. Because of that, and akin effects, members of a dyad are often more similar toward each other than other persons. Or, although not as often as the previous case, they are more dissimilar than other people (see the previous example on linkage by kinship from Chapter 1.2.1). However, most standard statistical tests lean heavily on the assumption of independence. Ignoring the interdependency between dyad members may lead to a loss of statistical power and increased Type-I-error rates. Hence, "spurious significant" results become more likely (Kenny and Judd, 1986).

A frequently used method for avoiding that problem is to collect data only from one individual in the dyad (Gonzalez and Griffin, 2012). However, by doing so, that part of

the data which is of key interest for dyadic research is lost. This is particularly the case for all kinds of social interactions. Consider the following example: a researcher might be interested in mother-son interactions, promptly after the mother has told her son that she ate all his Halloween candy. The specific interaction will depend on the son's characteristics, the mother's characteristics, and their unique relationship and history. The boy might be angry at first, and insulting his mother, but he might start laughing the moment he realizes she is joking. However, it is also possible that the mother becomes angry over her son's insults. Then starts scolding him, after which he reacts with truculence. Collecting only the boys' data would only reveal that some boys laugh about it, and other will become defiant. The most important part - the mother-son interaction which causes the outcome - is completely ignored. Information is lost about how the boy's initial reaction influences the mother's reaction, which again influences the boy's final reaction.

The better option is choosing statistical analyses that account for interdependence. For example, the analysis of variance (ANOVA) allows to model factors as either between (independent) or within (dependent) factors for scaled (metric) dependent variables. Furthermore, ordinary regression can be extended to multilevel models, which can also handle interdependency (Hox et al., 2010). Both models assume that the dependent variable is measured on an interval or higher scale. However, the multilevel approach can be combined with generalized linear modeling (Gill, 2000) for approaching the analysis of categorical data, as shown in Chapter 7. Another approach is to analyze each dyad separately and then to summarize the results (see Chapter 6). Moreover, Markov models, which are genuinely made for sequence data, can be applied to dyadic data by different approaches (see Chapter 8).

1.2.3. Within and Between Variables

Often in dyadic data analysis both, dyads and individuals, are treated as observational units. Therefore, it is important to differentiate between *within-dyad* variables and *between-dyad* variables. A *between* variable has always the same value for both members of a dyad but might vary across dyads. A *within* variable varies within dyads.

For example, a researcher might be interested in knowing whether couples' satisfaction with their partnership depends on their income. Using only *between* variables means taking the total partnership satisfaction and total income of each couple and, for instance, running a regression. So, the total income of a couple might explain why some couples are more satisfied than others. For instance, a higher income might re-

duce money related problems. And because of that, the total partnership satisfaction might increase. By contrast, *within* variables would be the individual partnership satisfaction and income of each member. So, income of one partner may account for his or her own satisfaction (a so-called *actor effect*, see Chapter 1.2.4) and for his or her partner's satisfaction (a so-called *partner effect*).

This example shows that some variables can be both. Income can be a between variable if the total income of a dyad is calculated, and income can be a within variable if the individual income of the members is used. However, other variables are exclusively within or between variables. For example, the number of years in marriage of a couple is always the same for both members, and therefore is considered to be a between variable. An exclusive within variable is, for example, eye color, because each member has his or her own color.

1.2.4. Conceptual Models for Dyadic Data

Kenny (1996) introduced three common conceptual models that describe how members of a dyad affect each other: the *Actor-Partner Interdependence Model* (APIM), the *mutual influence model*, and the *common fate model*. These three models are originally designed for scaled cross-sectional data, yet two of which can be adapted for sequence data: the APIM and the common fate model. Therefore, only these two will be described in this chapter. Their longitudinal adaptations are discussed in 1.3.1.

The APIM, in its cross-sectional form (see Figure 1.2A), assumes that characteristics of one member of the dyad can affect either the self (*actor effect*) or the partner (*partner effect*). Taking a heterosexual couple, for example, men's stress coping ability might affect the amount of which he is willing to communicate his stress (*actor effect of the man*). The same might also be true for the woman (*actor effect of the woman*). However, it is also possible that men's willingness to communicate stress also depends on the woman's coping ability (*partner effect of the man*). The opposite might also be true; the woman's willingness might depend on man's coping ability (*partner effect of the woman*). If the dependent variables are scaled, residuals can be allowed to correlate (not shown in Figure 1.2A). Positively correlated residuals might indicate that they have a common cause. For example, one couple has higher than average values on both of their stress coping ability. If all coefficients are positive, the model would assume that their willingness to communicate their stress is also above average for both. However, maybe the couple had an argument right before the measurement and thus both rates lower on the scale. Hence, the residuals for both would be negative.

The opposite might be true if the couple is in an exceptionally good mood, which results in positive residuals. Therefore, it is possible to think of correlated residuals as a latent variable that describes a common influence. The APIM, according to Cook and Kenny (2005), is the most favored model by dyadic researchers.

According to Ledermann and Kenny (2012) the common fate model (see Figure 1.2.B) is a theoretically important but underutilized model. The model assumes that for each variable, a construct exists at the level of the dyad rather than at the individual level. The corresponding within variables are considered to be indicators of this particular construct. For example, a husband and wife might have scored different values of dyadic coping ability on a partnership test. According to the common fate model, these scores are an expression of their combined coping ability. The same applies for stress communication abilities. Moreover, the model suggests that correlations between individual coping and stress communication abilities are produced solely through a correlation on the latent (common) level.

1.3. Combining Dyadic and Sequence Data

As in many other fields of psychology, research on dyadic interactions relies mainly on self- and partner reports. Typically, these reports describe an overall evaluation of a psychological mechanism (e.g., evaluation of the joint efforts to cope with stress), or they describe typical patterns of behaviors, which the two dyad members experience when together (e.g., how they jointly deal with the stress of one partner). However, self (and partner) reports potentially suffer from different biases: they may integrate an evaluative perspective about past behaviors, but also social comparisons, with other couples, which may be top-down biased by overarching constructs, such as relationship satisfaction, for example. They may also be biased due to self-deception, exaggeration, social desirability, mood dependency, or oblivion (Lucas and Baird, 2006).

Hence, in many contributions authors call for multimethod measurements (e.g., Eid and Diener, 2006) including behavioral coding and the analysis of behavioral interactions. According to Kenny et al. (2006) interval sampling (sequential coding) is the preferred method when different behaviors can be observed in interaction sequences.

Interval sampling in the context of dyadic observations implies that the observation period is divided into time intervals of the same length (e.g., 8 minutes may be divided into 48 intervals of ten seconds each). For each interval, whether a particular behavior occurred (0 = no; 1 = yes) is coded. Hence, if one behavior is coded for each dyad member, two sequences are recorded per dyad.

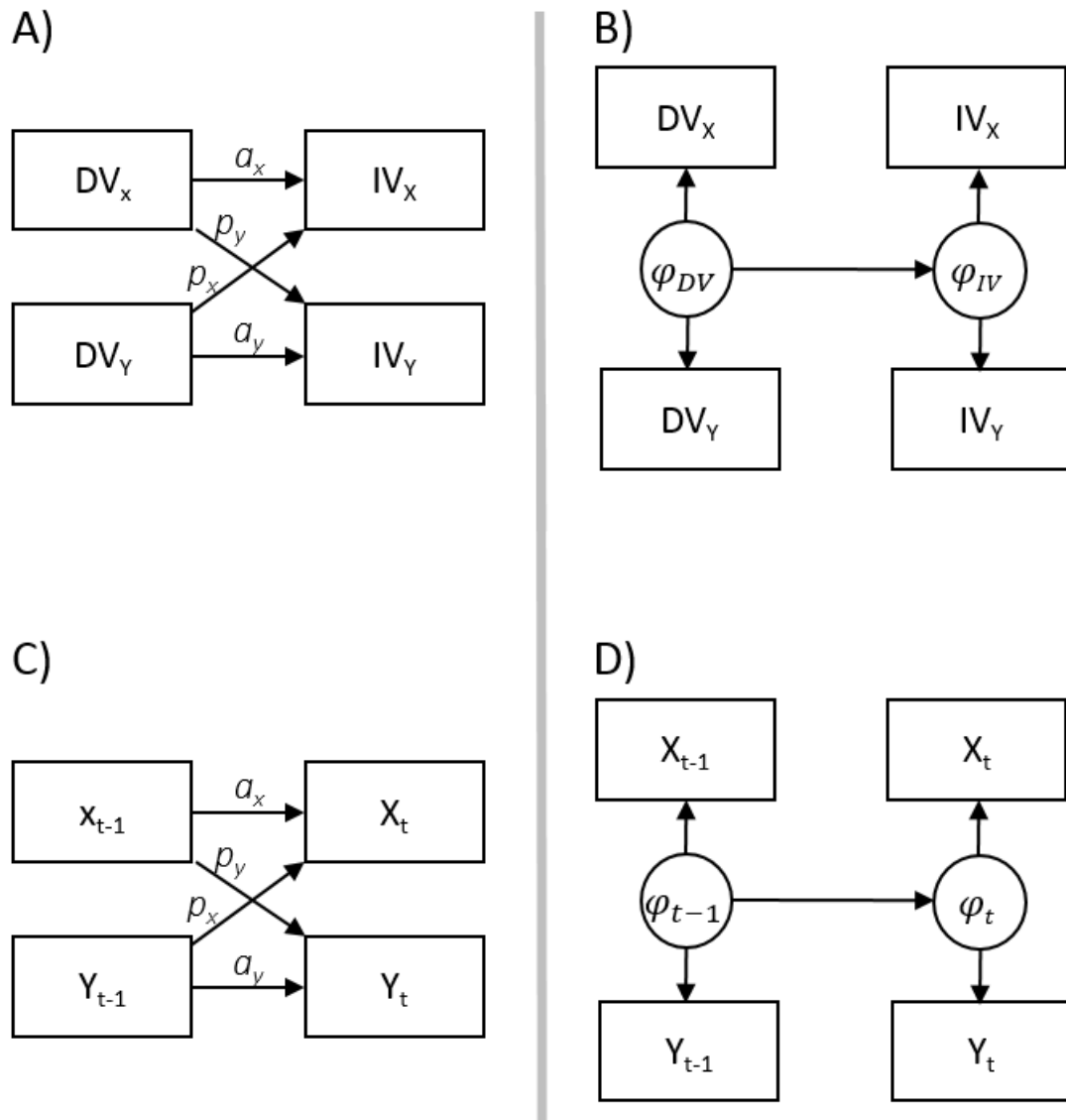


Figure 1.2.: A) shows the conceptual Actor-Partner-Interdependence Model (APIM), B) the longitudinal version both stem from Cook and Kenny (2005). DV and IV in the cross-sectional model typically represent different constructs, and X and Y represent different partners, whereas in the longitudinal version, the same variable is measured for each partner at two times (t replacing the DV and $t - 1$ replacing the IV). a_x is the actor effect for variable x , a_y is the actor effect for variable y , p_x is the partner effect for variable x , p_y y is the partner effect for variable y . B) shows the common fate model adapted from Kenny et al. (2006), correlations between DV and IV and between partners are solely explained by a latent common variable ϕ . Model D) shows the longitudinal adaptation presented in this monograph: auto-correlations and correlations between partners are explained solely by a latent common variable.

Figure 1.3 depicts an example of interval sampling for two behaviors: stress communication (SC) by one partner and coping reaction (dyadic coping: DC) by the other partner. The entries in the data matrix indicate whether the behavior occurred within a given sequence (interval); 1-1-0 for stress communication (SC), for example, indicates that the first partner communicated her or his stress in the first interval, did so in the second, but did not communicate her or his stress in the 3rd interval. By contrast, a sequence of 0-1-1 for dyadic coping (DC) indicates that the 2nd partner showed dyadic coping in the second and the third interval, but not in the first. The sequences are dyadic because they allow for studying interdependence. For example, stress communication (SC) might trigger dyadic coping (DC) responses, or vice versa, as indicated by the arrows. However, it is also possible that both behaviors are only indicators of an underlying dyadic coping process.

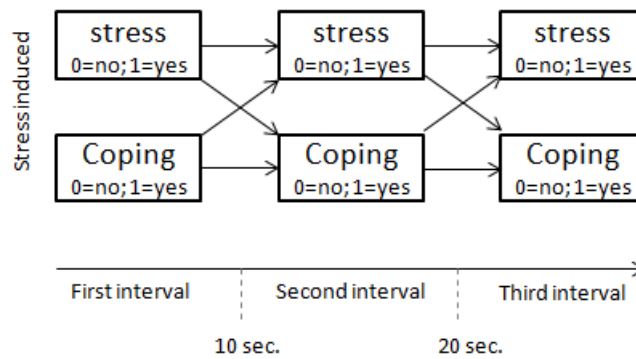


Figure 1.3.: Interval sampling for the example dataset. Displayed are the first three ten-seconds intervals of observation after stress was induced. Horizontal arrows between the boxes refer to actor effects. Crossed arrows refer to partner effects. SC: did stress communication occur? DC: did dyadic coping occur?

1.3.1. Conceptual Models for Dyadic Sequence Data

The APIM has been adapted for the analysis of longitudinal metric data (Cook and Kenny, 2005) and sequence data (Kenny et al., 2006). In the adapted version for metric data (see also Figure 1.2.C), the same constructs are repeatedly measured over time (e.g., her SC and his DC). Effects between two time intervals within one partner (female partners' SC at time $t - 1$ to SC at time t) are called actor effects, which correspond to

autoregressive effects in time-series analysis. Effects from one partner at $t - 1$ to the other partner at t are called partner effects (female partners' SC at time $t - 1$ to male partners' DC at time t). These effects correspond to cross-lagged effects in time-series analysis. The adaptation of the APIM for binary categorical sequence data is depicted in Figure 1.3. In this adaptation, the occurrence of male partners' behavior (here DC) at time t is predicted by their immediate previous behavior at $t - 1$ (men's actor effect), and by the behavior of their partners at $t - 1$ (men's partner effect). The occurrence of female partners' behavior (here SC) at time t is predicted by their behavior at $t - 1$ (women's actor effect), and by the behavior of their partners at $t - 1$ (women's partner effect).

In the common fate model, it is assumed that there is a property of the couple which influences both partners' behaviors. Consider a conflict between partners, where the conflict describes the couple as a whole and will likely lead to stress communication. In the same vein, female partners' stress may be seen as a property of the couple. Her stress will likely lead to stress communication by her and coping reactions by him. In sequential data and according to the common fate model, her stress communication and his coping reaction may be indicators of a latent status (female partners' stress), which can change over time (see Figure 1.2.D). Hence, changes in the two behaviors are modeled as indicators of one latent variable. Depending on the research question and the underlying assumptions, researchers may choose between the models. In the remainder of this contribution, we will outline possible adaptations, applicability, and interpretation of the different models relying on prototypical research questions.

1.3.2. Dyadic Sequence Data in Psychology and Related Fields

Dyadic sequence data can occur in all fields of psychological research when behavior of dyads is coded over time. This is the case in developmental psychology when parent-child dyads are observed, or in couples research. The same is true for investigating the interaction between a therapist and her or his client. However, there are two psychological research branches that commonly produce dyadic sequence data. The first one is research about *cooperation* in social dilemmas, and the second one is *joint action*.

Dyadic sequences often occur in studies about *cooperation*, especially if social dilemma games are used for measuring the amount of cooperation. Dawes (1980) characterizes social dilemmas by the two properties "(a) the social payoff to each individual for defecting behavior is higher than the payoff for cooperative behavior, regardless of what

the other society members do, yet (b) all individuals in the society receive a lower payoff if all defect than if all cooperate."

Several dilemmas have been established for simulating social dilemmas in experimental settings, most of which consist of several rounds. Each turn all players, usually two players, have to decide whether they want to cooperate or to deceive the other. Thus, time is measured as a discrete variable (game turns) and a player transitions through states of cooperation or deception (which is categorical). Furthermore, previous decisions of both players may affect actual decisions (interdependency). Thus, dyadic sequences are obtained.

Outside of psychology research, this kind of data is often analyzed by models made for sequence data (see Rapoport, 1963; Luce and Raiffa, 1957; Shubik, 1964). In mathematics (or gaming theory), the focus is often on research questions, such as 'What is the best strategy in this game?', whereas in the context of psychology, it is more common to aggregate the data and report means of cooperation (for a meta-analysis see Sally, 1995). Therefore, psychological studies focus more on research questions such as: 'Does the psychological construct A increase/decrease the amount of cooperation?' Using statistical models that can be applied directly to sequence data would enable new research question, such as: 'Does the psychological construct A affect cooperation patterns?' or 'How does cooperation develop over time and does this development depend on other psychological constructs?'

Another field where dyadic sequence data is common is *joint action*. Sebanz et al. (2006) defined joint action as the following: "[joint action] can be regarded as any form of social interaction whereby two or more individuals coordinate their actions in space and time to bring about a change in the environment." Researchers investigating joint actions often link two observational units together (*experimental linkage*) and code the behavior of those two. Examples are, two people must carry a table from one room into another, and their communication is coded; or two people have to react the same time in a visual attention experiment. If behavior is coded categorically and time is measured in time intervals, the resulting data would also be in the form of dyadic sequences.

This kind of research finds many real-world applications in the field of daily work and work security. For example, two air-traffic controllers often have to do identical jobs, such as observing the same area on a radar. This is called the redundancy principle. The idea is that each air-traffic controller has a small probability of making a mistake, yet the probability that both controllers are making the same mistake at the same time is extremely small. For instance, if the probability to make a mistake was

.001 for one controller in one time interval, then the probability that both will make the same mistake in the same interval would be .000001 (.001*.001). However, this holds true only as long as both probabilities are independent. Yet air-traffic controllers have to communicate with each other, and that may influence their error rates. Therefore, a researcher might be interested in screening for partner effects. For example, a panic reaction by one controller might increase the probability for the second to produce an error.

In neighboring disciplines the analysis of sequences is more common. Abbott (1995) propagated sequence analysis in quantitative sociology, and since then the number of applications has increased rapidly (Abbott and Tsay, 2000). For example, Casper and Wilson (2015) analyzed prominent actors' behavior in a national crisis via sequences. To give another example, Fuller and Stecy-Hildebrandt (2015) analyzed career pathways for temporary workers.

2. Example Data Sets

Typical statistical models will be exemplified throughout this contribution by using two different example datasets: *Couple-Cope*, which stems from a study of Bodenmann et al. (2015), serves as the primary example, and *Give-Some*, which originated from the bachelor thesis of Halstenberg (2016), as a supplementary example, in which the behavior of one member is pre-programmed.

2.1. Couples-Cope

This empirical dataset will be used for introducing typical sequence-data structures and for showing how to apply the presented statistical models of this contribution to them. The dataset is included in the R-Package "DySeq" (see Appendix C). The R code in Appendix A describes how to retrieve the example data and how to run the presented statistical procedures and models on it.

2.1.1. The Couples-Cope sample

The sample contains voluntarily linked dyads (see 1.2.1), which stem from a study of Bodenmann et al. (2015). In this sample, 198 heterosexual couples living in Switzerland participated. The couples had to have been in their current romantic relationship for at least a year and use the German language as their primary language. About 56% of the women and 40% of the men were students, and their age ranged from 20 to 45 years. During the study, either the woman, the man, or both partners were stressed using the Trier Social Stress Test (TSST; Kirschbaum et al., 1993).

Directly after the stress induction, both partners joined each other again, and the couple was left alone for eight minutes without any further instruction. During this period, which was introduced as time the experimenters would need for some adjustment of the experimental installations (a "fake" waiting condition), the two partners were filmed. In fact, the eight minutes of the waiting situation were the core interest of the study because this situation was supposed to reveal how partners interact after

one or both of them have been stressed. In the remainder of this monograph, this waiting condition will be called the interaction sequence.

For the sake of simplicity, only those 64 couples are considered where only the female partner was stressed. Thus, the analyses are restricted to her stress communication (sequence 1) and the male partner's support reaction (sequence 2; supportive dyadic coping). Stress communication (SC) includes all verbal and non-verbal behaviors signaling stress. Supportive dyadic coping (DC) includes all verbal and non-verbal behaviors aiming to support the partner's coping efforts in a positive way. Both behaviors were coded relying on the SEDC (Bodenmann, 1995) in 48 intervals, ten seconds in length. The data structure hence consists of two interdependent sequences with 48 entries. The sampling method was already shown in Figure 1.3. The presented statistical models throughout this monograph can be applied to any comparable data situation.

2.1.2. Stress Communication and Dyadic Coping

Whereas other conceptualizations and definitions exist for both stress and coping, this contribution focuses on Bodenmann's (2005) approach and the taxonomy, which can be seen as an expansion of the transactional stress model by Lazarus (Lazarus, 1966; Lazarus and Launier, 1978). Lazarus describes stress as a person-environment transaction. The person's appraisal mediates the impact of an external stressor. In a first step, a person evaluates the situation regarding its ambiguity, relevance, and controllability (primary appraisal). In a second step, the person evaluates his or her coping ability and resources. Depending on the outcome of both steps, a stress reaction might be triggered, and the person tries to overcome the stress by applying coping strategies. That step is followed by the phase of reappraisal, in which the person reevaluates the situation and resources, adapts his or her coping reactions if necessary, and finally ends the coping reaction if the stress was reduced successfully.

Bodenmann (2005) defines *dyadic stress* as the following: "Dyadic stress represents a distinct form of social stress. It involves common concerns, emotional intimacy between two people, and the continuity of a social system (i.e., the maintenance of the marriage)." In dyads, stress communication becomes important because it allows the unstressed partner to perceive the stress of the other partner. He or she might then appraise the stress and then initiate coping responses, too. Because one partner's well-being and satisfaction depend on the other's welfare and satisfaction, both make

efforts to restore a state without stress. The actual form of this coping might depend on the couple's characteristics, but also on the type of stress.

Furthermore, Bodenmann (2005) classifies *dyadic stress* by the following: *the origin of stress*, whether it is *direct or indirect*, and whether it is *simultaneous or sequential*. The origin of stress describes whether the stress originated from inside or from outside the couple. Examples of an internal stressor would be different goals or marital conflict. Whereas an example for an external stressor would be that one partner is a victim of mobbing at his or her workplace. Direct dyadic stress means that a couple encounters the same stressor, whereas indirect implies that one partner faces stress and then "brings the stress home." That is, the stressed partner did not cope successfully with a stressful event before he or she met his or her partner. That might be due to a lack of coping abilities, the impact of the stressor, or simply a lack of time. Simultaneous stress means that both partners experience the stress at the same time; for example, political unrest affects both partners simultaneously. Sequential means that one partner encounters stress first and then, after some time, the other partner is introduced to the stressor, too.

Applying this classification to the sample data shows that the induced dyadic stress is external, indirect, and sequential: the female is stressed initially by the experimenters (outside source), after which the couple comes together (sequential), and then the man is introduced to her stress via her verbal or non-verbal stress communication (indirect). According to Bodenmann (2005), stress communication can be problem-focused, which is typically verbally expressed, or emotion-focused, which is communicated verbally or non-verbally. In the present study, all these forms were coded simply as stress communication (SC). However, future research might also investigate patterns of different forms of stress communication with the presented model.

Bodenmann (2005) also distinguishes different forms of dyadic coping activity, such as *positive dyadic coping*, which includes all forms of coping efforts that aim to reduce common stress. Exemplary forms of positive dyadic coping are: helping the partner to reframe the situation, expressing solidarity, engaging in joint problem solving, or taking over responsibilities of the other. *Negative dyadic coping* includes all forms of coping efforts that are performed in a negative way, such as mocking, sarcasm, or the like.

Both positive and negative dyadic coping can be classified further. In the present dataset, it was coded whether *supportive dyadic coping* was shown. This is a form of positive dyadic coping that is characterized by one partner's efforts to support the other partner's coping efforts (emotionally or problem-focused). The other forms are

common dyadic coping, which means that both partners engage stress in a symmetrical way, and *delegated dyadic coping*, which means that the partner steps into the breach. For example, one partner is stressed because of an important meeting and the other partner takes over.

2.2. Give-Some

As mentioned before (see 1.3.2), social dilemma games that are used for measuring cooperation produce dyadic sequential data. Several dilemma games have been established for simulating social dilemmas in experimental settings. Whereas the previous dataset was about the interaction of two humans, this dataset is about the interaction of a human and a pre-programmed algorithm. The computer in this setting is programmed to ignore the humans behavior; thus, patterns should become less coherent than those of the last example. Therefore, this example will be used as a contrasting example.

2.2.1. The Give-Some Sample

The dataset contains sequences of 42 subjects (28 females, 14 males) that engaged in a so-called four-coin dilemma. All participants were students at Bielefeld University and received course credit plus the chance to win one of three 10€ vouchers. The age of participants ranged from 18 to 38 years, with a mean age of 21.91 years ($SD = 3.78$). Two persons were aware that the other player was pre-programmed, and one person was later identified as an outlier. As a consequence, these three were excluded from the original dataset.

A brief outline of the history of social dilemma games and the specific setup of this experiment will be presented in the following two sections. Up to this point, it is sufficient to know that participants had to exchange coins with a pre-programmed algorithm within each of the 32 game turns. The number of given coins indicates cooperative behavior. Thus, the more coins were given, the more cooperative behavior was shown. On average, participants gave less coins ($M = 1.83$, $SD = 0.37$) than the algorithm ($M = 2.00$, $SD = .71$). Giving more coins than did the opponent in the last turn was classified as cooperation. Overall, human players cooperated in 63.98% of all turns, whereas the computer cooperated in 70.35% of all turns.

2.2.2. Cooperation in a Give-Some-Dilemma

The core idea of dilemma games can be explained by the prisoner's dilemma, which is probably the best-known dilemma paradigm in psychology. In its original form (Poundstone, 2011), two accomplices of one crime are arrested. The problem is that the prosecutors can charge both culprits only with minor offenses. Thus, if both criminals stick together (cooperate with each other), both will receive a prison sentence of only one year. Because of that, the prosecutors offer a deal: if one of the criminals testifies against the other, the defector will be set free and the other criminal will be sentenced to three years in prison. The clue to this dilemma is that if both betray each other, both will be in prison for two years.

Examining the dilemma on the group level reveals the following: if both kept silent (cooperate), the combined time of imprisonment would be two years; if only one cooperates but the other defects, the combined sentence will be three years; and if both betray each other, the combined punishment will be four years. Thus, the best course of action for minimizing the combined sentence is to cooperate.

However, on an individual level, the best course is always to betray the other. If the other cooperates, the betrayer will be set free. If the other betrays too, the betrayer has only to serve two years instead of three. According to classical gaming theory, the criminals or, in terms of gaming theory, players, will behave rationally, on an individual level. That is, both will betray each other (Milovsky, 2016). Despite that prediction, Messick and McClintock (1968) found that people differ in their tendency to cooperate.

The original prisoner's dilemma was modified many times, and the give-some dilemma is one descendant of this process. The first extension, the *iterated prisoner's dilemma*, introduced the idea that players can engage the dilemma repeatedly. Because of that, they remember previous actions of their opponent and might change their strategy accordingly. This development led to the *take-some dilemma*, in which players could grade their betrayal. In this version, each person starts with an amount of money and each turn the players can choose how strongly they want to blacken the other player. The stronger the betrayal, the higher the penalty for the other. The clue is that the betrayer could keep half of the opponent's fine for him or herself. Again, it would be best to cooperate on the group level, because to total on the group level will be highest if they do not betray each other at all. However, on the individual level, it is best to defect because the individual will always get more money the more she or he betrays the other. Finally, the give-some dilemma emerged and changed the

perspective from grading the betrayal to grading the cooperation. Players start each round with a certain number of coins. Each turn, both players decide secretly and simultaneously on how many coins they want to exchange with each other. Each coin that a player keeps for himself is worth one victory point (VP). Yet every coin which was received by the other player is worth two VPs. Each VP is worth real money (e.g., both players get 1€ per VP). In this scenario, the group wealth will be maximized if both players give all their coins to the other player (the total money would be doubled). However, on an individual level, it would be best to keep all coins for oneself (for an actual example with four coins, see Chapter 2.2.3).

Axelrod and Hamilton (1981) found that algorithms that used only an individually rational strategy performed poorly in simulations. Instead, altruistic strategies worked better in the long run. The best-performing, yet most simple, was the tit-for-tat algorithm. The idea of the tit-for-tat strategy is to copy the opponent's behavior from the last round. Therefore, if the other cooperated during the last round, the algorithm would do so, too. In other words, the algorithm rewards cooperation and punishes betrayal.

Messick and McClintock (1968) found that human players use three different types of strategies: 1) individualistic, 2) competitive and 3) pro-social. The individualistic strategy focuses only on maximizing their outcomes: persons who follow this plan will always betray the other person in the prisoner's dilemma game. If there is more than one round, these individuals will only cooperate if it benefits their individual outcome. On the other hand, people who use a competitive strategy will maximize the difference between both players. Competitive players tend to defect, yet they might cooperate a little for "luring" the other player into cooperation. They aim for being better than the other player, even if that will decrease their individual outcome. The last strategy, pro-social, tries to maximize the outcome of both players. Players of this style will try to cooperate most of the time, yet might deploy a tit-for-tat strategy if the other player defects. They aim for increasing the group outcome.

2.2.3. The Give-Some Experiment

In the bachelor thesis of Halstenberg (2016), which was overseen by the author of this monograph, students participated in a four-coin give-some dilemma: each player starts with four coins that are worth one point for oneself and two points for the opponent. Both players have to submit zero to four of them to the other player. The decision is made secretly and simultaneously.

For example, player A submits zero coins and receives one coin in turn one, meaning that Player B keeps three coins and submits one. Player A gains a total of six points (4 own coins * 1 = 4 VP; 1 coin from the other player * 2 = 2 VP; adding up 4 + 2 = 6). Player B, on the other hand, gains a total of three points (3 own coins * 1 = 3 VP; zero coins from the other player * 2 = 0 VP; adding up 3 + 0 = 3). Clearly, it would be best for both if they always submitted all their coins so that they would always receive eight victory points (0 * 1 + 4 * 2 = 8). However, on an individual level, it is always best to keep all coins because giving them away would not result in victory points for oneself.

The game consisted of 32 rounds. Yet the experimenters did not tell the players how long the game was because, otherwise, it is always most profitable to give zero coins during the last turn (the opponent cannot react anymore after that). It was also concealed that they would play against an algorithm (a pre-programmed player). Instead, they assumed that they would play against a person in the adjacent room. In fact, this person was only an assistant of the experimenter and did not participate in the experiment. Nevertheless, during the entire experimental session, the confidant was treated like a participant in order to keep up the deception.

For both the human player and the artificial player, how many coins they gave each turn resulting in two time-series was recorded. These time-series were transformed into two dichotomous sequences by the following rule: if a player gives more or equal coins than the other did in the previous turn, it is coded as *cooperation*. Otherwise it is coded as *defection*. Because this transformation is not applicable to the first turn of the dilemma, the resulting sequence consists of 31 entries. For one demonstration in this monograph, a slightly different transformation was used. In Chapter 8, the data is used for demonstrating Markov models with more than two states. Therefore, three states were created by the following transformation: if a player gives more coins than the other in the previous turn, it is coded as *cooperation*. If the player gives fewer coins than the other did in the previous turn, it is *giving less*, for an equal amount is coded *giving equal*, and if more, it is coded *giving more*.

2.2.4. The Pre-Programmed Player

The algorithm was not programmed to react or adapt to the human players actions; instead, it followed a semi-random program. The 32 rounds were divided into eight blocks. Within each block, the computer gave 1x1, 2x2, and 1x3 coins in randomized order. The only exception was the very first turn, in which the algorithm always gave

two coins followed by one, two and three coins in randomized order. The algorithm was originally designed that way for several reasons: on average, the algorithm gave two coins, which is also the center of the event space. Moreover, it was always possible for the human player to give more or fewer coins than the algorithm did before. Finally, a random behavior should not induce a certain type of behavior for the human player. Hence, a human player would be more likely to show his or her *natural* behavior in a dilemma-type-situation. However, the algorithm has another benefit for its use in this monograph, the random behavior of the algorithm gives an example for patterns that are extremely erratic. Therefore, it is a good contrast to the *couples-cope* example, which will show a more consistent pattern.

3. Research Questions and Corresponding Data Analyses

Several potential research questions may motivate researchers to collect sequential behavioral data. These may range from more general issues about the associations between observed behaviors of partners to more detailed questions about the processes during social interactions or interactions between partners. In this chapter, general research questions will be introduced and exemplified using the example datasets. Furthermore, recommendations are given for choosing the statistical model that fits a certain research question best. Some research questions can be answered by more than one statistical model. For these models, advantages and disadvantages are briefly sketched in this chapter, yet will be explained in more detail in the corresponding chapters. Tables 3.1 and 3.2 give an overview of the research question, the approaches that can be used for answering these questions, the corresponding chapter, and provides further notes.

3.1. Getting an Overview About Dyadic Sequences (Visualization and Descriptives)

For scaled (metric) and categorical data, a consensus exists on how a researcher can give an overview. Typically, when the outcomes of a study are scaled (metric) variables, then means and standard deviations are reported. Furthermore, data is often visualized via histograms or density plots. If the data is categorical, frequencies are reported. Additionally, data can be visualized via (clustered) bar plots. However, most researchers in psychology are not used to sequence data. Thus, there is no consensus on how to present or to visualize sequence data. Descriptive questions about sequence data might include: "How often do states change?", "Are there regions of time in which states change more often?", or "How to visualize the distribution of states over time?" Moreover, "Are there distinct subpatterns of behavior that occur more often?" Chapter 4 will answer to these fundamental questions. Finally, a preliminary analysis could

also ask, "Is there an association between the states of one sequence and the other sequence?" Linking this to couples-cope example, "Do men show dyadic coping (DC) more frequently if their partners communicate their stress (SC) more frequently?" (See Chapter 4.2).

3.2. Duration of Behavior (Time-to-Event)

Another question is: what is the typical duration of a particular behavior? That is, how long does it take until the response was coded the last time (cavalierly presuming that this was really the last time the behavior was shown and was not repeated after the observation period). Taking the couples-cope data for example, how long does stress communication last? Or, whether and when does stress communication stop? These kind of questions can generally be answered by time-to-event analysis.

Yet what might be even more interesting to know is "What influences those durations?" Or, more statistically speaking, which covariates influence the time to an event. For example, the goal of dyadic coping is to support the partner's efforts in dealing with stress. Therefore, it is reasonable to assume that a better dyadic coping ability of one partner should shorten the other partner's duration of stress communication. Thus, the natural questions arise as to "how long does SC last? And does it depend on covariates, as men's self-assessed dyadic coping ability for example?" Time-to-event analysis can answer these questions (see Chapter 5) by estimating a so-called survival rate and a hazard curve. The former shows how the probability that a behavior is still shown up to a time interval is distributed over time, and the latter shows how the probability that a behavior ends within a time interval is distributed.

Furthermore, a researcher might be interested in investigating whether two behaviors of dyads end in temporal proximity. For example, if stress communication and dyadic coping responses of the partner are solely indicators of an underlying dyadic coping process, both behaviors should end right after the couple successfully solved the coping process. The Clayton-Oak model (see Chapter 5.3) can be used for investigating how close the temporal proximity is: the model is similar to a multi-level model, yet built for time-to-event variables. By building one cluster per couple, it can be used for investigating whether time-to-event variables of partners are highly correlated (close temporal proximity) or not, and whether the correlation is significantly greater than zero.

Finally, if dyadic sequences are believed to be best explained by a common fate model, another question could be "How long does it take until the latent process ends?"

or "How long does it take until couples solve the stress?" These kind of questions can be tackled by restricting hidden-Markov models (see 8.) Because this approach combines time-to-event modeling with the common-fate model, it is explained at the end of the following chapter (Common Fate).

3.3. Assuming a Latent Dyadic Process (Common Fate)

As shown in Chapter 1.3.1, the common fate model assumes that the coded behaviors of two dyad members are only indicators of a latent process. Thus for the couples-cope dataset, the assumption is that there is a dyadic coping process and that SC and DC are indicators of this process. By contrast, it is safe to assume that there is no common latent process for the give-some example because the algorithm was not programmed in that way. However, there might be a latent process for only one of the dyad members. For example, at first, the human player might be in a state in which he or she believes that it is possible to cooperate with the other player. However, over time the player might register that the other player acts more or less randomly. Thus, the human player might change his intentions and accordingly adapt his or her strategy. These kind of questions can be analyzed by hidden Markov models (see Chapter 8).

The core idea of hidden Markov models is that the latent process is modeled as a Markov chain. That is, the probability of the dyad transitioning from or staying in a latent state at t depends only on the previous state at $t - 1$. The corresponding states in the observed sequences are only seen as indicators of the latent state. For example, a couple which is in a state of active stress solving might have a higher likelihood of showing stress communication (SC) and dyadic responses (DC), whereas in a stress-free state, the occurrence might be less likely. These probabilities are called emissions and can be interpreted as a measurement error or reliability of the indicators: If the woman's SC were a perfect indicator for the couple being in a state of stress, it should be 1.00. Meaning, if the woman shows SC, the couple is in a state of stress. However, an emission of 0.50 would mean that the woman will show SC only in half of the intervals in which she and her partner are in a common state of stress.

Several questions can be asked for such models: How many latent states exist? What are the transition probabilities? And how can the latent states be described using the associated emission probabilities?

Moreover, Markov models can be restricted so that they estimate a latent hazard for the latent process. This answers the question of "how long does it take until the latent process ends?" This approach can be useful when the latent state is assumed to

be finite. For example, stress is induced, and at some point in time the stress should end. Thus, two states can be assumed: *stress* and *no stress*. Furthermore, because stress is induced initially, all dyads should start in a state with *stress*. Finally, the model is restricted so that *no stress* is an absorbing state (dyads can only leave the state of stress, but not enter it again, see Chapter 1.1.5). Thus, the only transition-probability that is estimated is the latent-hazard, which is the probability for the latent process to end from one time interval to the next (see Chapter 8). The hazard can also be used for calculating the estimated survival time. This approach combines the previous research question about duration with the common-fate model.

3.4. Analyzing Promptness of Interaction (APIM)

Does the behavior of one member trigger an immediate reaction by the other, and vice versa? And furthermore, how stable is a behavior? Or in other words, what is the probability that a behavior at time interval $t - 1$ is immediately followed by the same behavior at t ? These questions are equivalent to partner and actor effects of the APIM presented in Chapter 1.3.1.

For the couples-cope data, this questions would be translated into: "Does SC by one partner evoke a prompt DC reaction by the other and vice versa?" (partner effects); "What is the stability of DC/SC?" (actor effects). Or for the give-some example, the question can be translated into: "Does cooperation, which only occurs by chance, trigger cooperation of the human player in the following turn?" For that example, interest lies only at the partner effect for the algorithm because we already know that there is no effect from the human behavior toward the algorithm.

This kind of research question is especially important because it gives an indication regarding causality. For example, if stress communication is promptly followed by dyadic coping responses, but not vice versa, it could be because stress communication causes dyadic coping responses. One partner perceives the stress and helps the stressed partner to solve the stress. Or, maybe both behaviors trigger a response from each other. That could be because one partner reacts to the induced stress by communicating stress and the other partner responds to it by applying dyadic coping behavior, which might encourage the other to keep the stress communication up, thus the other will also keep up the dyadic coping behavior. Even though statistical models exist which explore the promptness of these reactions, causality should never be based on statistical models alone. For example, a common-fate model might produce the same behavior patterns as an APIM with bidirectional effects. Thus, causation

should always be based on deduction, yet statistical models can add support for those kind of theses. Or at least show that one important pre-requirement for causation, the temporal precedence, is fulfilled.

This monograph provides three strategies for applying APIMs on dyadic sequence data: **(A)** The aggregated logit models approach by Bakeman Gottman (1997; see Chapter 6), **(B)** multi-level logistic regression (see Chapter 7), and **(C)** Basic Markov Models (see Chapter 8).

(A) Analyzes each dyad (e.g., couple) separately for each dependent variable using logit models. After that, the estimates are aggregated and post-hoc tests can be conducted to test whether estimates are significantly different from zero or differ between groups. The main advantages of this approach is that it is computationally fast and that it can be used for single case (single dyads) analysis. Moreover, effects can be interpreted directly as actor or as partner effects. Thus, questions such as "is there a significant partner effect of dyadic coping on stress communication?" can be answered directly.

(B) The multi-level approach is used to model the dependencies between partners: dyads are handled as level-2-units, where the individuals (level-1-units) are nested within couples. The behavior at t serves as the dependent variable and the behavior at $t - 1$ as the independent variable. Interaction-terms are used to represent the fact that variables might represent behavior of one partner or the other. The categorical nature of the dependent variable is accounted for by using the logistic function for dichotomous sequences. In theory, other link functions, such as the multi-logit function, could be used for sequences with more than two categories. This model provides estimates similar to (A), but also provides variances for the estimates (random effects) as an indicator of how much these estimates differ between couples. Again estimates can be interpreted as actor and partner effects. Standard errors are provided so that inferential statistics are available for each effect. Thus, the same questions as in (A), plus the question of whether couples are similar (low random effects) or heterogeneous (high random effects) regarding their promptness, can be answered. One limitation of this model is that it cannot be used for single case analysis.

(C) The APIM is directly translated into a Markov chain, a model assuming that the probability that a dyad is in a certain state at t depends only on the previous state at $t - 1$. The output of the Markov model comes as transition probabilities, whereas the previous two models give results as beta-coefficients in a logit-metric. Therefore, the output can be interpreted relatively intuitively. For example, the following question can be addressed directly by the model without transforming the output: "what is

the probability of a couple that shows stress communication (SC) and dyadic coping (DC) at one time interval ($t - 1$) to show none of those behaviors at the next time interval (t)?" or "what is the probability that stress communication is shown after an interval in which no stress communication or dyadic coping was shown?". However, one drawback is that these probabilities cannot be interpreted directly as actor or a partner effects. Instead, the output must be transformed into beta-coefficients that are comparable to the actor and partner effects of the previous models. Hence, this approach is to choose when transition probabilities, rather than actor and partner effects, are at interest. Yet for comparability to other studies, transition probabilities can be transformed after that, and p -values can be simulated using a Monte-Carlo approach.

3.5. Latent Groups or Clusters (Unobserved Heterogeneity)

Are the mechanisms producing the behavioral patterns the same for all couples, or is there unobserved heterogeneity such that there are different typical response patterns? Applying those question to the couples-cope dataset: Are there different types of couples regarding their response patterns to the induction of stress? For example, does the experience of stress lead to very prompt and adequate SC and DC behaviors in all couples, resulting in a quick solution to the problem, or are there also couples struggling with the stressor for a long time? Or, for the give-some dilemma: Are there typical joint patterns of human-computer interaction?

Detecting these subgroups accounts for so-called unobserved heterogeneity in the population. Dealing with sequence data, the notion of unobserved heterogeneity implies that subgroups differ regarding their specific transition matrices. This monograph provides two strategies for tackling this research question: (A) Mixture Markov models can be used to address this question (see Chapter 8); (B) clustering via the OM-procedure (see Chapter 9).

(A) Mixture Markov models assume that all dyads follow a basic Markov model (APIM), yet the transition probabilities differ between latent classes. For example, some couples (latent class 1) might show low probabilities of change from non-stress-related behavior to stress-related behavior, whereas other couples (latent class 2) bounce between time intervals in which they show stress-related behavior and those in which they do not. Thus, a more specific question could be how many latent classes exist? How can they be described using the transition probabilities?

The approach can also be applied to hidden Markov models (Common-Fate), in which case the emissions and transition probabilities may vary between different classes. Therefore, emissions must be taken into account for interpreting different classes, too. For example, it is possible to assume a case in which two classes have the exact same transitions probabilities, yet one class contains *silent* couples, who are less likely to communicate their stress, and *talkative* couples, who are very likely to communicate their stress. Assuming that both classes communicate stress only if they are in a state of common stress would result in low emissions for the former class and high emissions for the latter.

Using Mixture Markov models is useful in that several latent classes might exist, especially when the researcher has a strong assumption of whether the sequences are produced by an APIM or a common-fate model. Another feature of this model is that membership to an latent class is probabilistic. Therefore, each sequence has one probability for each class that expresses how likely it is that the sequence belongs to that particular class. These so-called posterior probabilities can be used for assessing uncertainty. For example, a sequence might belong with a probability of .51 to latent class one and is therefore assigned to that class. However, it is uncertain whether this assignment is correct. Whereas, a posterior probability of .98 suggest that the amount of uncertainty is very small.

(B) The OM-procedure clusters dyads according to their similarity. Sequences of dyads are considered similar if the number of computational steps to transform one sequence into the other are small. The main question of the OM-procedure is "How many clusters exist?" and "How can they be interpreted?" For the latter, clusters are analyzed separately using visualization, descriptive statistics, and other models such as Markov models. Then the results are compared between the clusters and used to characterize the clusters. Cluster analysis does not claim that these clusters represent real latent classes. A cluster is just a collection of dyads that are relatively similar to each other. Thus, a researcher should be careful to interpret clusters found by the OM-procedure as differences caused by real existing types of dyads. This approach is useful for data reduction or in early research as explorative analysis. Moreover, it can be an alternative for the mixture Markov models if sample sizes are small.

Table 3.1.: Overview Research Questions and Related Models (Part A)

Research Question	Approach	Chapter	Further Notes
Getting an Overview (Visualization and Descriptives)	Mean time	4.1	mean time spend in states.
	State-Distribution-Plot	4.1	frequencies over time.
	Entropy and Transitions	4.1	measures for variability.
	Correlations	4.1 - 4.2	across time in 4.1, across dyads in 4.2.
	Subsequence-Analysis	4.4	frequencies of subsequences; most frequent subsequences.
	Sequence-Plot	4.3	single-case analysis; outliers.
Duration of Behavior (Time-to-Event)	Cox-Regression	5.1 - 5.2.1	for non-dyadic sequences.
	Cox + time-dependent cov.	5.2.2	for dyadic sequences.
	Shared Frailty Model	5.3	latent common variable.
	Latent-Hazard Model	8.3.1	time-to-event as a latent process; latent process leads into an absorbing state.
Assuming a Latent Dyadic Process (Common Fate)	Hidden-Markov	8.3	behavior is caused by a latent process, indicators might influence each other.
	Multi-Channel Hidden-Markov	8.3.3	behavior is only caused by a latent process, indicators are independent of each other.
	Latent-Hazard	8.3.1	time-to-event as a latent process; latent process leads into an absorbing state.

Table 3.2.: Overview Research Questions and Related Models (Part B)

Research Question	Approach	Chapter	Further Notes
Analyzing Promptness of Interaction (APIM)	Aggregated Logit Model	6	computational fast; single case analysis possible
	Multilevel Model	7	estimates and tests random effects; no single case analysis possible
	Basic Markov Model	8	single case analysis possible; focus on transition probabilities; only bootstrapped p -values
Latent Groups or Clusters (Unobserved Heterogeneity)	Mixture Markov Model	8.4	assumes separate Markov chains for each latent group, a sequence belong to a latent group with a certain probability.
	OM-Clustering	9	Sequences are groups using OM-distances, so that distances within a cluster are minimized.
	Comparing Groups or Cluster	9.4	Clusters or Groups are compared using data visualization, applying different models, and by an ANOVA-like approach using OM-distances.

4. First Steps in Analyzing Dyadic Sequence Data

Before estimating statistical models, analysts should principally plot their data to gain initial insight into their data. Despite the fact that graphical analysis is always subjective to some degree, Anscombe (1973) demonstrated that visualization is also essential to a good statistical analysis. He presented four datasets, which seemed identical when only summary statistics (e.g., Pearson correlation; p -value) were inspected, yet revealed completely different forms of statistical association when plotted (e.g., linear relationship, squared relationship, perfect linear relationship with one outlier). Following his advice, this chapter will discuss several methods for plotting sequence data, all of which were originally introduced by Gabadinho et al. (2009). However, this chapter discusses them in light of dyadic sequence data. Additionally, some summary statistics will be shown which cover the research questions from Chapter 3.1. However, these statistics should be complemented by their corresponding graphics (e.g., scatter plot, histogram).

4.1. Graphical Analysis

Figure 4.1 depicts the mean time spent in the four states aggregated across all couples. The plot clearly shows that the state of showing SC and DC is the most dominant, followed by a state of showing no stress related behavior at all. States of showing only one stress-related behavior ("SC only" or "DC only") are rare. However, we do not know how these states developed over time, or how different the trajectories are.

The distribution of states over time is seen in Figure 4.2 and allows for a first graphical examination of the paired sequences. To this end, the two sequences have been joined via the state expand procedure (Vermunt, 1997). For each time interval, the joint occurrence / non-occurrence of the two behaviors (see Table 4.1) is depicted, which results in four possible states per time interval for the couples-cope example.

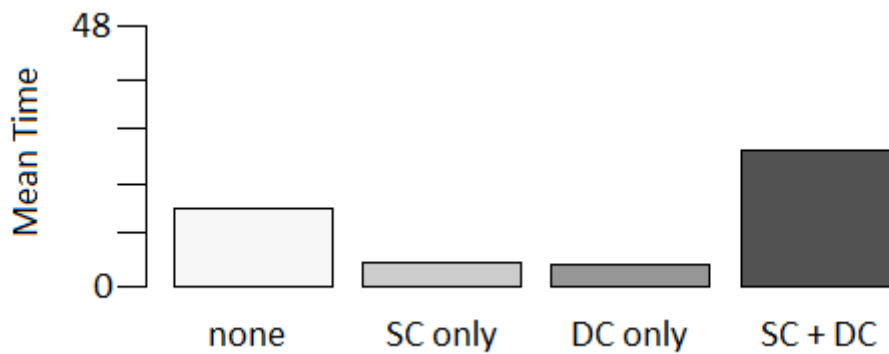


Figure 4.1.: Mean time spend in the four states.

The distribution plot shows that in the beginning, the simultaneous display of stress communication (SC) by women and dyadic coping (DC) reaction by men (SC+DC) is the most frequently displayed behavior (almost 100%). After roughly ten intervals (1 minute and 40 seconds), frequencies for the combination of no SC and no DC reaction (no reaction), a stress communication but no DC response (SC only), and no SC but a DC reaction (DC only) increase. In the following minutes, the frequencies of SC only and DC only remain rather stable, but frequencies for no reaction increase further whereas frequencies for both reactions decrease. One possible explanation could be that couples intensively discuss the stressful event in the beginning, and thereon some of the couples manage to be less stressed. Hence, no SC nor DC reaction is necessary for them, whereas other couples still discuss the stressful event until the end of the interaction sequence.

Figure 4.3 shows the distribution of both sequences separately. The disadvantage of this graphic is that we cannot see whether states occur together or not. However, the advantage is that we can see that occurrence of both SC and DC declines almost in par-

Table 4.1.: State-Expand for SC and DC

		DC (Dyadic coping)	
		No	Yes
SC (Stress communication)	No	none	DC only
	Yes	SC only	SC+DC

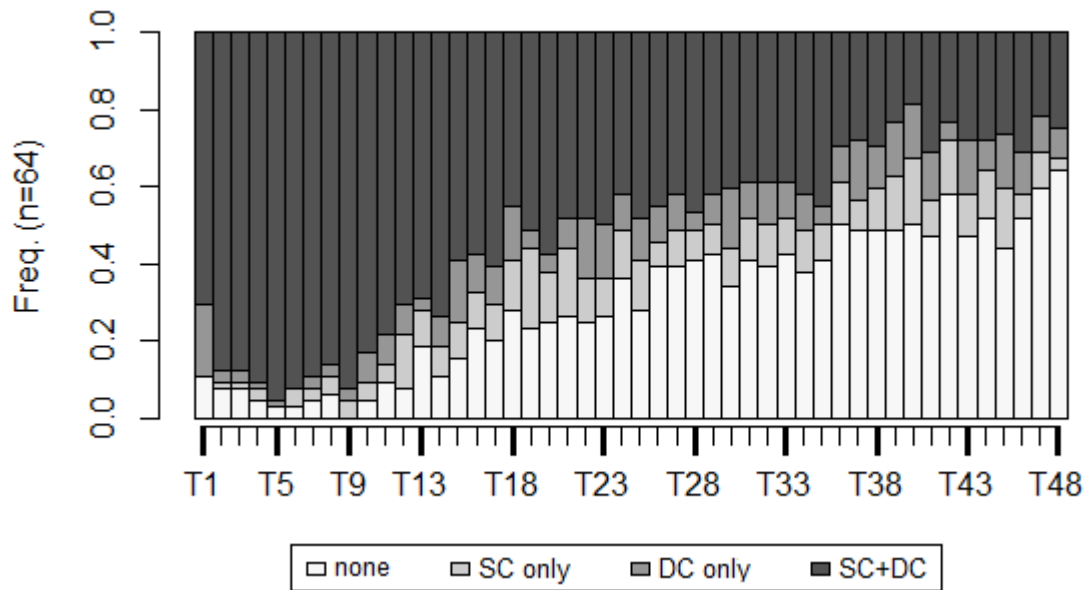


Figure 4.2.: State distribution plot of the couples-cope dataset ($N = 64$); Y-axis: relative frequency of shown behavior; X-Axis: observation period sliced into 48 time intervals; SC: stress communication; DC: dyadic coping; none: No SC or DC was shown; SC+DC: SC and DC was shown.

allel. The frequencies of both time-series reveals a strong and significant relationship ($r = .97$; $p < .001$). Therefore, we might assume that there is an association between these behaviors. However, as Granger and Newbold (1974) pointed out, one should be careful interpreting Pearson correlation for time-series. The correlation might be biased due to autoregressive effects or the existence of a third time dependent variable.

The Shannon entropy (Shannon, 2001) as a measure of dispersion (Figure 4.4.A) depicts that couples show very homogeneous behavioral patterns in the first sequences (simultaneous display of SC and DC) and that the couples become more dissimilar at sequence 20. That is, they show all different combinations of SC and DC behavior.

An additional initial insight can be gained by inspecting the number of state-transitions as a measure of stability. The number of state-transitions (Figure 4.4.B) depicts how often couples change from one state to another. A high number indicates frequent changes in a couple's behavior. Hence, the number of state-transitions allows for differentiating between volatile couples who frequently change their behavior (high scores) from those who tend not to change their behaviors often (low scores). The histogram shows that the majority of couples change their behavior about 10 to 20 times

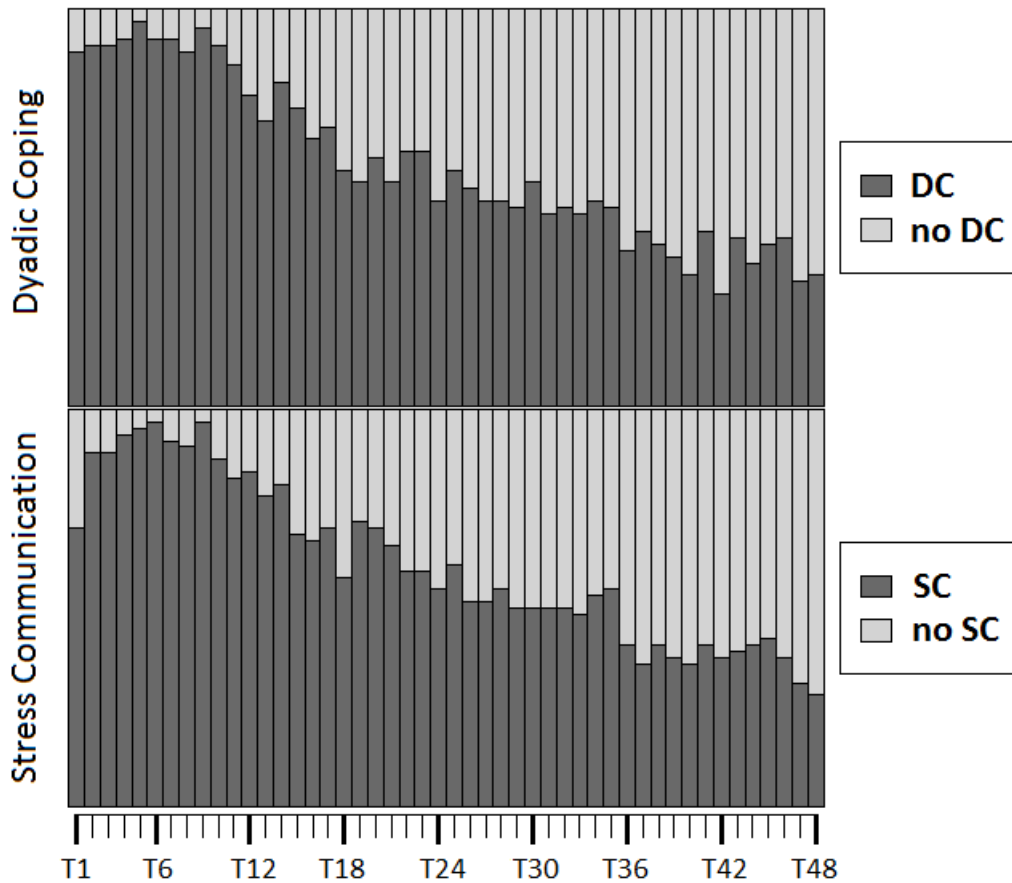


Figure 4.3.: Multi-channel state distribution plot of the couples-cope example data ($N = 64$); Y-axis: relative frequency of shown behavior; X-Axis: observation period sliced into 48 time intervals; SC: stress communication; DC: dyadic coping; none: No SC or DC was shown; SC+DC: SC and DC was shown.

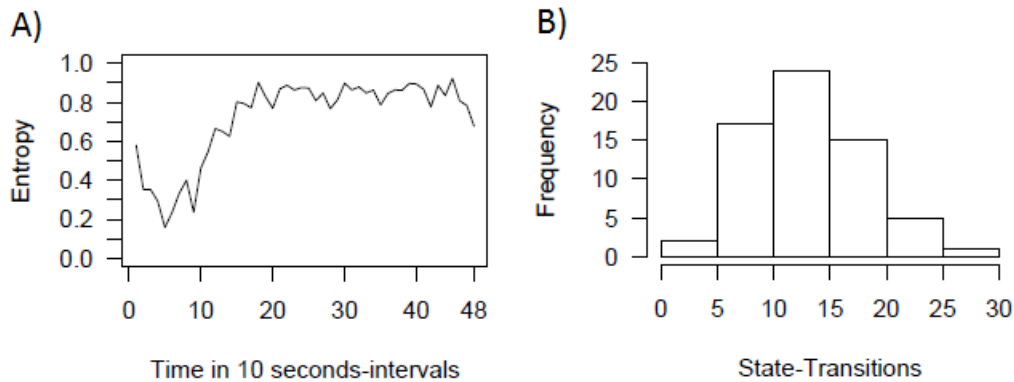


Figure 4.4.: (A) Shannon entropy plot and (B) histogram of state-transitions for the example data.

out of 47 possible changes during the 48 intervals, which is roughly about one to two times within a minute.

4.2. Association Between Behaviors

Knowing that the number of transitions varies across couples, one might also ask if DC and SC not only correlate over time, but also across observational units. Or, more specifically asking, do men show DC more frequently if their partners show SC more often? This question can be answered by calculating the Pearson correlation between the frequencies of the behavior of interest shown by the first partner and the frequencies of the interesting response by the other partner. For the sample dataset, we find a high Pearson correlation of $r = .86$ ($p < .001$) between the frequencies of stress communication and DC responses. Which, overall, implies that in couples where women show high rates of SC, men tend to show more DC reactions. And vice versa, if women show low rates of SC, men show low rates of DC as well. However, due to the aggregation of the data, information about the direction and the contingency (i.e., promptness) of this association are lost. Hence, it remains an open question whether SC leads to prompt DC responses, if DC leads to prompt SC reactions, or if the association is bidirectional. The last question can be answered using aggregated logit models (see Chapter 6), multilevel models (see Chapter 7), or Markov models (see Chapter 8).

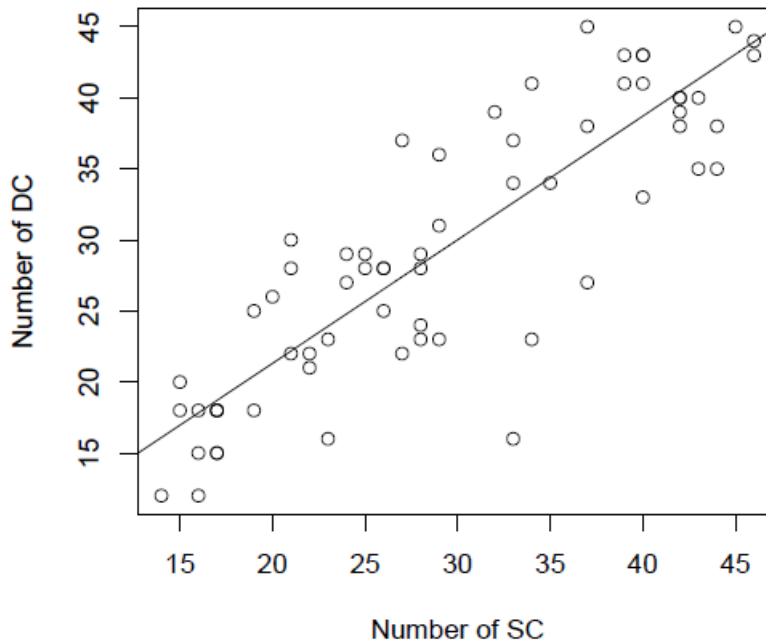


Figure 4.5.: Frequency of SC is plotted against DC; each point represents one couple.

4.3. Inspecting Individual Cases

Most research questions in psychology concern individual differences. Those differences are often investigated by comparing groups or by drawing samples and testing whether individual differences in one variable can explain differences in another variable. Although single case analysis is less often conducted, single case analysis is very important for sequence data for a number of reasons. The first one is that sequences that are completely different than the majority of the sample might influence future analysis (outliers). The second argument for single case analysis is that outliers can provide further insight. For example, imagine: 98% of the couple cope sample had shown a reaction to the initial stress induction, whereas 2% had shown no stress reaction at all. In that case, one could conclude that the stress induction failed and, thus, exclude those samples from future analysis. Instead, it could be promising to investigate why the stress induction failed: Maybe the couples saw through the experiment. Asking them how they did that could be useful for designing future experiments. Or perhaps the couples are especially stress resistant. Analyzing the traits of these couples could give insight for research about stress resilience. Finally, single case analysis could be used to get a picture of typical state patterns that exist in the data.

Figure 4.6.A shows a visual representation of individual states. Each row depicts the patterns of one dyad from the couples-cope example, and their corresponding row number in the dataset are shown on the y-axis. The x-axis shows the time intervals. The shading distinguishes between the different states. Figure 4.6B presents a close-up of the sequences first (ID = 1), thirty-third (ID = 102), and the forty-seventh (ID = 144).

The second couple shows a state of SC and DC for at least three and a half minutes. Then the stress communication stops, and one-time interval later, the partner stops his dyadic coping efforts as well. There is a short flickering of SC later on, yet the partner does not react, and no further stress communication is shown. One possible interpretation is that this couple successfully coped with the stress.

The second couple shows signs of stress-related behavior, yet there are time gap, in which no such behavior is shown. The first gap is only one time interval (10 seconds). This can be caused by nearly anything. However, the second gap is more interesting because SC ends one time interval before DC stopped. This constellation looks nearly the same as the previous example. However, roughly one minute later, the female starts to communicate her stress again, yet in this case, the male responds with dyadic coping. After that, a new phase of combined SC and DC is triggered which lasts for about one and a half minute.

Inspecting 4.6.A again suggests that almost all sequences show these prototypical patterns. Few sequences exist that show a delayed response to the stress induction, but besides that, no digressive sequences exist. Virtually all sequences begin with a long phase of stress communication and dyadic coping, which ends at some point. Some couples show stress-related behavior up until the end of the observation period, and some couples take some *breakes* (gaps without stress-related behavior). However, the duration, until the last stress response is shown, differs a lot. Thus, future analysis, such as shown in Chapter 5, might investigate whether covariates can explain these differences or if couples can be grouped according to their fastness (see Chapter 8.4). For example, are there *fast copers*, who quickly reduce the perceived stress, but also stress-prone couples (i.e., *slow copers*), who do not find a way to reduce the stress?

4.4. Finding Typical Subsequences

A researcher might also be interested in knowing what the most common subsequences are. That is, which patterns occur in most sequences. For example, Figure 4.7 shows the seven most common subsequences. The first bar (SC + DC > SC only) shows that 88% of couples experience a transition from a state of SC and DC to a state

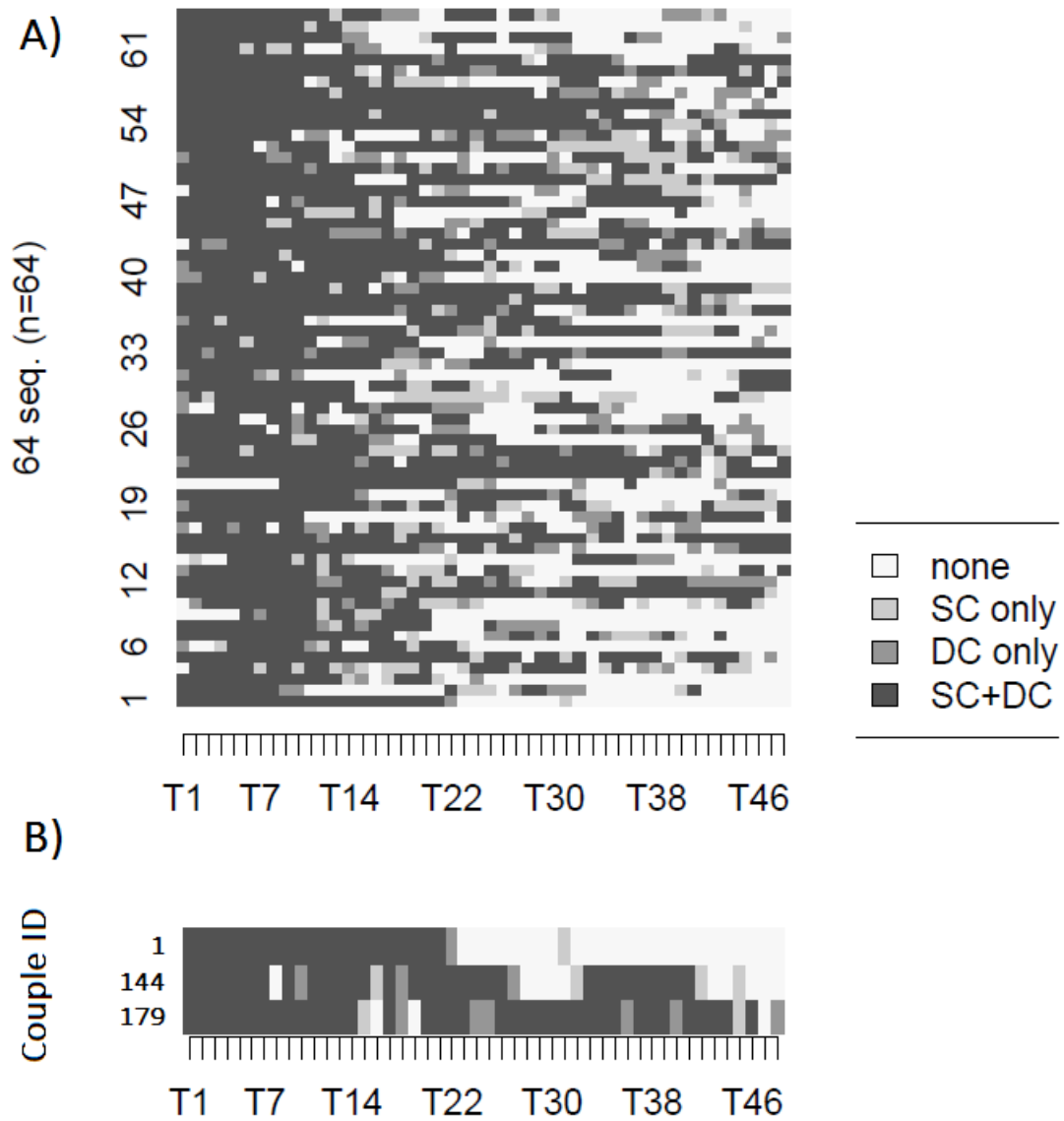


Figure 4.6.: A) Shows all individual sequences in one plot, each row represents one sequence, each colored block is one state. B) compares only three individual sequences (couples with the ID 1, 144, and 179 from the couples-cope example, represented by row numbers 1, 47, 59 from A).

of SC only at least once. The second bar (SC only > SC + DC) depicts that 81% of dyads transition from a state of SC only to a state of showing both at least once. The next bars follow the same principle. For example, the state of showing no SC and no DC is most likely preceded by a state of showing DC only, or by demonstrating DC and SC. Thus, one might infer that dyadic coping is often shown until the couple stops showing any stress-related behavior at all. Finally, the seventh bar shows a more complex, yet frequent, sequence. The subpattern (SC + DC > SC only)-(SC only > SC + DC) means that a couple shows a state of combined SC and DC at some point in time, then transitions to a state of SC only. The hyphen indicates that the couple stays in that particular state for some time until it transitions back to a state of showing both SC and DC. At least 75% of couples experience this subpattern at least once. Therefore, it seems quite common that dyadic coping accompanies stress communication. However, the men stop their dyadic coping for some time, whereas the stress communication of the woman goes on.

The analysis of subsequences can be used to compare different groups of dyads. For example, if there are *slow* and *fast* copers, as suggested in the previous chapter, it could be interesting to compare those clusters. That will be shown in Chapter 9.4.

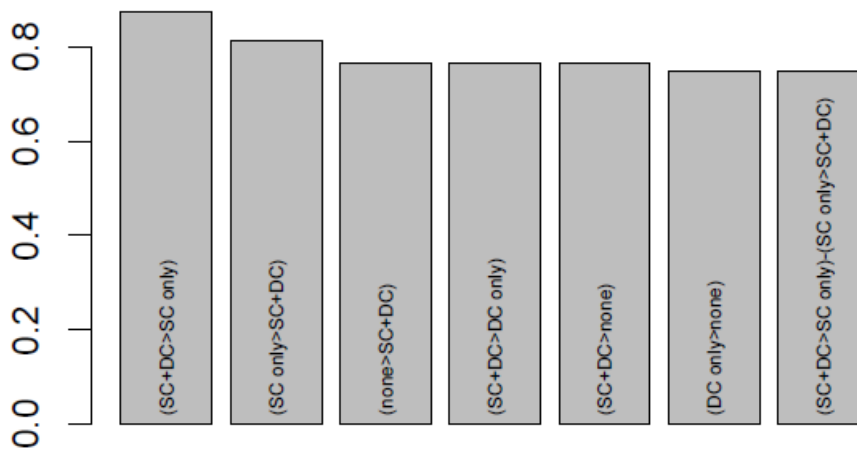


Figure 4.7.: The seven most common subsequences for the Couples-Cope dataset. SC: stress communication, DC: dyadic coping.

4.5. A Contrasting Example (Give-Some)

Figures 4.8 and 4.9 show most of the previously introduced visualizations for the couples-cope example side-by-side with the corresponding plot for the give-some dilemma. By contrast, there is no clear trend in the state distribution (see Figure 4.8.A2) for the give-some example. Moreover, Figure 4.8.B2 shows no distinct phases of particular behavior. Instead, dyads change quickly from one state to another. Both together result in high values of the Shannon entropy (see Figure 4.8.C2), meaning there are no game turns, in which certain states are dominant. The next figure (4.9.D2) shows again that there are many more transitions for the give-some dataset than for the couples-cope dataset (4.9.D1), even though the maximum number of possible transitions is less (31 for the give-some example as opposed to 47 in the couples-cope dataset).

This plot reveals that there is a sequence with only a few transitions. The sequence is marked in Figure 4.8.B2 by an arrow. Inspecting this sequence reveals that the algorithm shows cooperative behavior during almost all of the game turns (except one), whereas the human player does only cooperate in three game turns. Therefore, this sequence can be seen as an outlier because it behaves completely differently than the rest of the sequences, which change quickly from one state into another. Finally, the scatter plot for the give-some example (see Figure 4.9.E2) reveals a negative linear relationship ($r = .73$, $p < .001$), as opposed to the positive relationship in the couples-cope example (see Figure 4.9.E1). The negative correlation can be explained by the following: If a human player tends to give less coins, the likelihood of the algorithm for cooperation increases because it always gives 2 coins on average. Vice versa, if a human player tends to give more coins on average, the probability for the algorithm to cooperate by chance decreases.

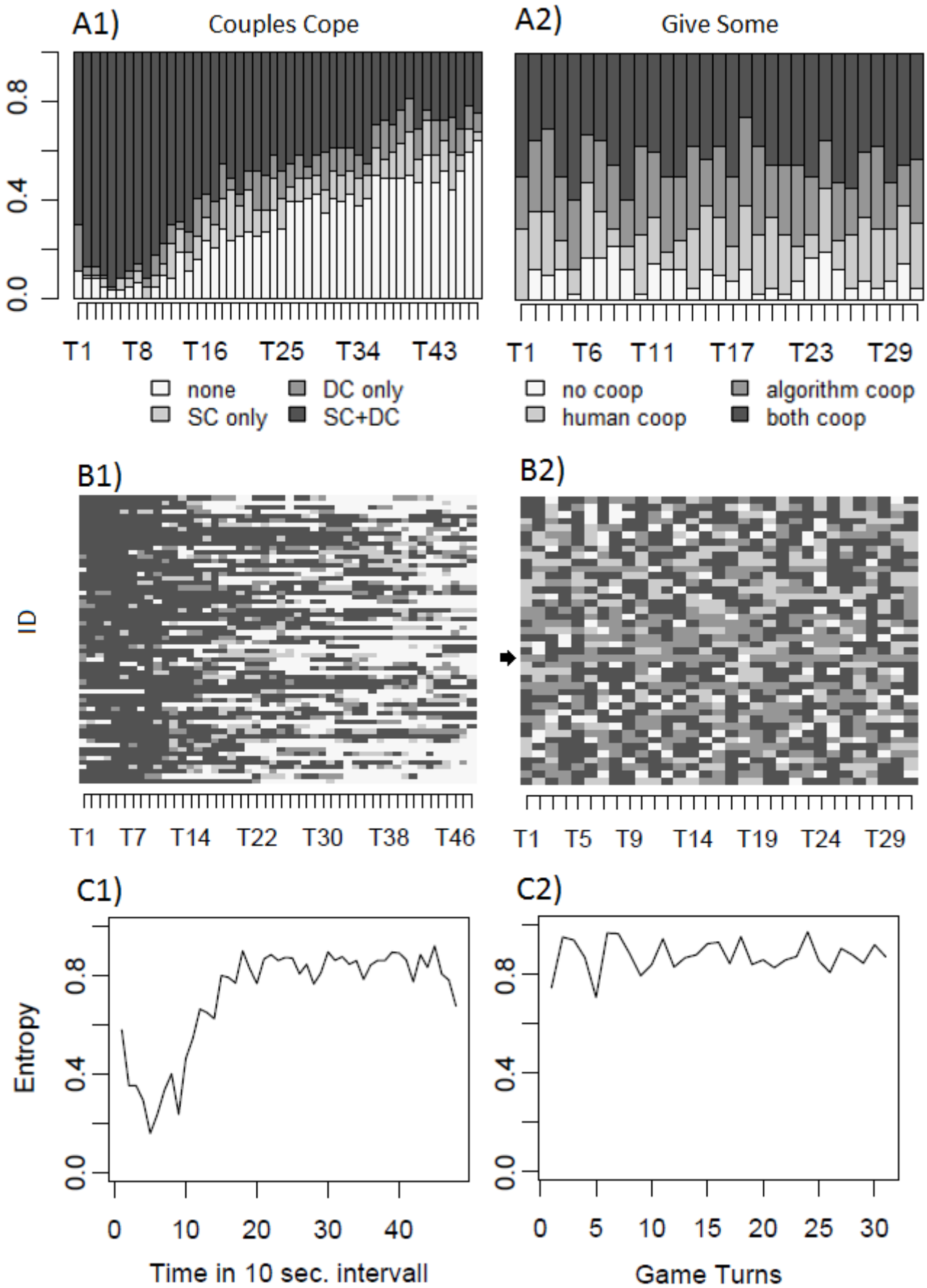


Figure 4.8.: Visualization of the couples-cope (1) example vs. the give-some example(2): A) shows the state distribution for couples-cope (A1) and give-aome (A2); B) shows individual sequencesfor couples-cope (B1) and give-some (B2), the arrow marks an outlier; C) Shannon entropy for couples-cope (C1) and give-gome (C2)

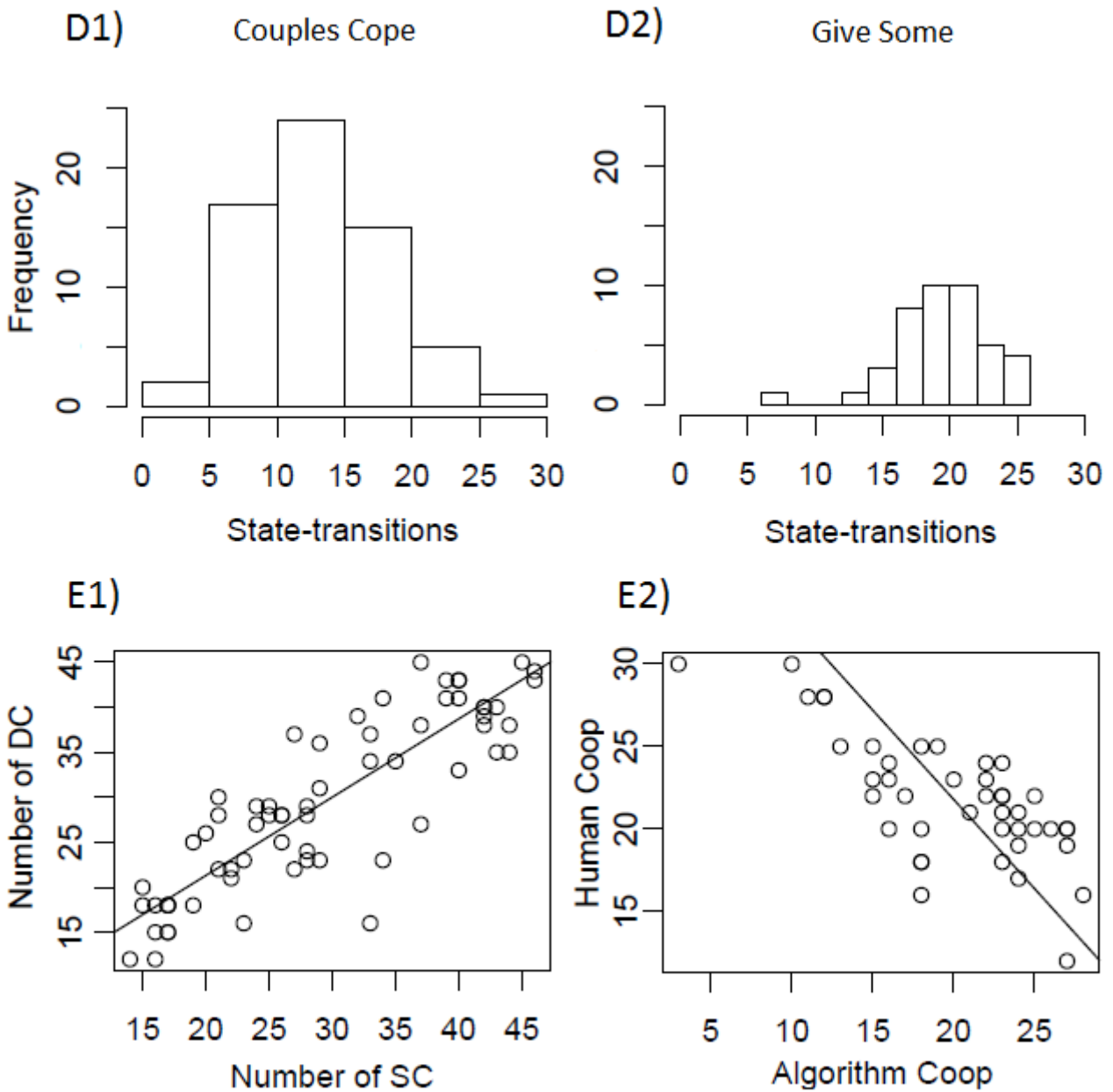


Figure 4.9.: Visualization of the couples-cope (1) example vs. the give-some example(2): D) number of transitions for couples-cope (D1) and give-some (D2); E) linear relationship between SC and DC (E1) and between human cooperation and cooperation by the algorithm (E2)

5. Dyadic Sequences in Time-to-Event Analysis

In principle, time-to-event analysis is used to answer the general questions of "whether and when" an event happens for a single observation unit, and if its occurrence depends on covariates (Singer and Willett, 2003). Hence, it directly relates to the research question of Chapter 3.2 (*Duration of Behavior*). Sequence data with only one final absorbing state can be transformed into time-to-event data without loss of information (see 1.1.5). Sequences are replaced by two variables: the first variables represent whether an observational unit entered the absorbing state within the observation period or not (e.g., 0 = *no*; 1 = *yes*); the second variables is the time interval t , at which the observational unit entered the absorbing state. If the observational unit did not enter the absorbing state, the second variable is the last observed time interval.

Additionally, other types of sequences can be transformed in a similar way (see 5.1.1) by asking whether and when a particular state is shown for the first/last time. Considering the couples-cope example (see 2.1), the question can refer to whether and when stress communication was shown for the last time. Under the assumption that the end of stress communication also indicates the end of stress, that question can be translated into "What is the typical duration of stress? Furthermore, does it depend on time-independent covariates?" In this chapter, men's self-assessed dyadic coping ability (mDCa) will serve as such a time-independent covariate; that is, it varies between observational units, but it is constant over time. The same question can be asked for the dyadic coping (DC) reactions by the men. However, the dyadic coping responses are time-dependent. At some points, DC is shown, and at others, it is not. Both questions will be answered by applying the Cox-regression (see 5.2).

Moreover, one might be interested in knowing if the duration of stress communication and dyadic coping responses might be caused by an underlying latent variable similar to the common fate model (see 1.2). The latter can be modeled by so-called frailty models; one of which, the shared frailty model, will be used in this chapter as an illustrating example (see 5.3).

An alternative approach for modeling an underlying latent process is to use hidden Markov chains. However, even though similar research questions can be asked to those models, they belong to another class of statistical model. Therefore, they are covered in Chapter 8.3.1.

5.1. Time-to-Event Data

According to Singer and Willett (2003), the most important aspects of time-to-event analysis include the definition of an *event*, *time*, and *the beginning of time*.

In biology, one of the most common *events* that is analyzed is death. For example, researchers investigate the questions: "How long does it take until caterpillars die?" and "Does it depend on covariates, such as the caterpillars' nutrition?" The good thing about death as an event is that it is an unambiguous event: a caterpillar is either dead or not. By contrast, events in the social sciences are harder to define because they often have more than one definition.

For example, in psychology, a possible event might be alcohol relapse after withdrawal therapy. Fuller (1997) provides a summary of the most frequently used definition. It ranges from the one-time consumption of any amount of alcohol, over more than two or three alcoholic drinks per day, to cut-off values of alcohol breakdown residuals in blood tests. Other definitions treat one-time consumption of alcohol as a slip and define actual relapse as a change in behavioral patterns (e.g., loss of control). Thus, time-to-event data about alcohol-relapse might reflect different aspects of relapse, depending on the definition.

Moreover, the definition of an event can affect the measured time of occurrence. For instance, alcohol relapse is often defined as addictive behavior patterns operationalized by drinking several days in a row (e.g., five days). The problem with that is that it is unclear whether the actual relapse started on the first day of drinking because that was the moment the relapse was initiated, or on the last day (e.g., the fifth) because that was the moment the pattern was completed. There is no general answer to that question because it depends on the actual research question. However, it is safe to say that events should be defined as accurately as possible before collecting data, in order to avoid ambiguity.

Singer and Willett's (2003) second aspect is the definition of *time*: As discussed in Chapter 1.1.3, time can be defined as continuous or discrete. Data that measures time in thin intervals is regarded as continuous, whereas data with wide intervals is regarded as discrete. The time-to-event analyses that are shown in this monograph

(Cox-regression and shared frailty model) assume a continuous definition of time. The problem with applying these models to discrete time data is that so-called ties can occur, which can induce a bias in the model's estimates. A tie means that two or more events happen within the same time interval.

Cox-regression is used in this chapter, among other reasons, because several corrections for handling ties in Cox-regression exist. The most prominent ones are the Breslow correction (Breslow,1974), the Kalbfleisch-Prentice correction (Kalbfleisch and Prentice,1973), and the Efron approximation (Efron,1977). Hertz-Picciotto and Rockhill (1997) conducted a simulation in which a Cox-regression was fitted to discrete time data. The Efron-approximation worked best under all conditions. The bias in point estimates and standard errors were small. The only exception was in the most extreme condition ($n = 25$ and 10 ties per interval), which means that about 40% of the sample had the event at the exact same time interval. However, for the second most extreme conditions ($n = 25$, and 5 ties per interval, as well as $n = 50$ and 10 ties per interval), the bias was below 1%. Therefore, as a rule of thumb, one can say that the number of ties per interval should be less than 20% of the sample size if the Efron-approximation is used.




Singer and Willett's (2003) third aspect is the definition of the *beginning of time*, which means one has to determine when the observation starts. In the relapse example, that could be right after subjects commit to abstinence, after the last day of detoxication, or after subjects are released from withdrawal therapy. Thus, the date at which the observation starts might be different for each participant. Alternatively, the observation can start for all persons at the same time. For instance, a researcher could measure whether and when employees become ill for the first time in a year. Hence, the observation starts for all employees on the first of January. Finally, a less obvious example for the beginning of time can be birth; that is, asking whether and when something happened for the first time in life. This definition is especially interesting for fields such as epidemiology ("Whether and when does an illness strike for the first time in a humans life?") or developmental psychology ("Whether and when does a child acquire a skill, such as walking only on foot, for the first time?").

If the event is not observed, it is said to be *censored*. Three main forms of censoring exist: right, left, and interval censored. The first one is the most common one and means a data point is above a certain value (or time), but it is unknown by how much. This type often occurs when the event has not happened until the end of the observation period (also known as type-I censoring). Therefore, it is not known whether and when the event will occur after the observation. Left censored means that

it is only known that the event occurred before a certain time but not exactly when. Finally, interval censored means that the event occurred somewhere between two time points.

5.1.1. Transforming Sequences into Time-to-Event Data

The transformation of sequence data into time-to-event data is straight forward. Both *sequence data* and *time-to-event data* share the same definition for the *beginning of time*. Thus, the beginning of time is whatever the first entry of sequence refers to. In the couples cope example, the beginning of time is the moment right after the stress induction and right before the fake-waiting condition started.

ID	Sequence	$t_{Event} = \max(t_{(s=X)})$	$t_{Event} = \min(t_{(s=X)})$
A		6	3
B		8	6
C			8

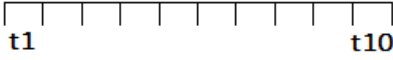


Figure 5.1.: The state of interest are the black colored states in the graphical representation of the sequences A, B, and C, and is denoted as X in the formulas. A and B are sequences with recurrent states, and their transformations are time-to-event data. The first transformation answers the question "How long until the black state is shown for the last time?", and the second the question "How long until the black state is entered for the first time?"; C shows a sequence with one absorbing state. The first transformation is not applicable because of an absorbing state (e.g., death) does not end.

The event can be defined as the following: "When is a state shown for the last time?" or "When is a state shown for the first time since the beginning?" Both transformations are depicted in Figure 5.1 and can both be applied to sequences with recurrent states, but information might be lost by the transformation because whole sequences are reduced to two pieces of information: first, whether the event occurred within the observation period, and second, when it occurred.

Applying the second transformation ("When is a state shown for the first time since the beginning?") on sequences with only two states will not result in a loss of information if one of those states is an absorbing state. An observational unit will enter the absorbing state or not (first information). Furthermore, a unit can only enter the absorbing state at one time interval (second information).

Therefore, the second transformation is especially useful for research questions with absorbing states (e.g., "How long until a caterpillar dies?"), but also, for research about relapses (e.g., "Whether and when does the first relapse after therapy occur?"). The first transformation is useful if the duration of behavior patterns is investigated, especially if these patterns consist of sporadic behavior. For example, a researcher might be interested in knowing what the typical duration of relapse is ("How long till the relapse ends?"). In that case, the researcher might define the beginning of time as the first day of relapse, and record when the patient drinks alcohol after that.

Defining the first day without drinking as the event is not a good strategy because the alcoholic might show isolated times of abstinence. However, designating the last day of drinking is a better way to determine the end of relapse, yet there is still one problem with defining the event in that way. Near the end of the observation period, an alcoholic might show drinking behavior for the last time only because the observation period has ended. Thus, if the behavior of interest is shown until the last time interval, it should be treated as right-censored (meaning, the event did not occur within the observation period (see Chapter 5.1.2). Additionally, a definition of the event may also include that the person must be sober for at least a number of days before the observation period ends.

Regarding the couples-cope data, the beginning of time is defined as the first time interval of the observation right after the stress induction. The event is defined as the last coded communication of stress. For the sake of demonstration, it is presumed that this is the time when the stress reaction stops. However, it is undebatable whether any other reason, for example, an uncooperative partner, may also lead to the last stress communication. About 28.13% of the data was right censored because stress communication did not stop until the 48th interval.

Time was measured discretely in sequences, yet most time-to-event analyses assume continuous time. As demonstrated in Chapter 1.1.3 and the previous chapter, applying these models on discrete time data is relatively safe as long as there are not too many ties. Ties can be avoided by measuring time with a large number of thin-graded intervals. For the couples-cope example, time was measured by intervals of ten seconds, and the observation period was 8 minutes, so each sequence is 48 intervals long.

Because there are only 64 couples, ties should be relatively rare. Analyzing the frequencies reveals only a small number of ties: The maximum number of ties is in the 47th interval, which has six ties. That is still below 20% of the sample size; and thus bias should be negligible (see Chapter 5.1).

5.1.2. Survival, Hazard, and Cumulative Hazard

Common functions that describe event occurrence over time are the survival function (S), the hazard function (H), and the cumulated hazard function. Changes in all three functions can be quite small, especially when the first events occur, and thereby, rounding can conceal small differences between intervals and/or functions. Because of that, and for demonstration purposes, example results of the first intervals with actual events will be presented with four digits in the following.

The survival function (see Equation 5.1) is the probability that an event has not happened to an individual until the end of a given time interval. It starts at time zero, presuming no observation unit has already experienced the event before the start of the observation. Therefore, by definition, the survival for time zero must be one. The survival indicates how many *survivors* (women who still communicate their stress) remain in the sample until a particular time interval:

$$\text{Survival for interval}_j = \frac{n \text{ who have not experienced the event by the end to the interval}}{n \text{ in the data}} \quad (5.1)$$

The survival for the sample data can be seen in Figure 5.2.A: It starts with a value of 1 and holds steady until interval 19, indicating that 100% of couples maintain their SC until the beginning of this interval. Then it drops by 0.0156 (1/64); the first one out of 64 couples ends SC before the end of time interval 19. In interval 20, it drops again by 0.0156 (another couple ends SC). Still, 96.88% ($62/64 * 100$) of couples remain in the sample at the end of the 20th interval. Next, the decrease accelerates over time ($S_{t=25} = .91$; $S_{t=30} = .89$; $S_{t=35} = .83$; $S_{t=40} = .69$), and at interval 45, the number of couples with ongoing SC has halved (median lifetime, ML). At the last interval, it drops to .30, indicating that 30% of the sample has not ended their SC until the end of the observation period.

The hazard function is the conditional probability that an individual (or in this case, a couple) will experience the event (last SC) in a specified time interval; given that it has not experienced it in any preceding period. The hazard for a given time interval

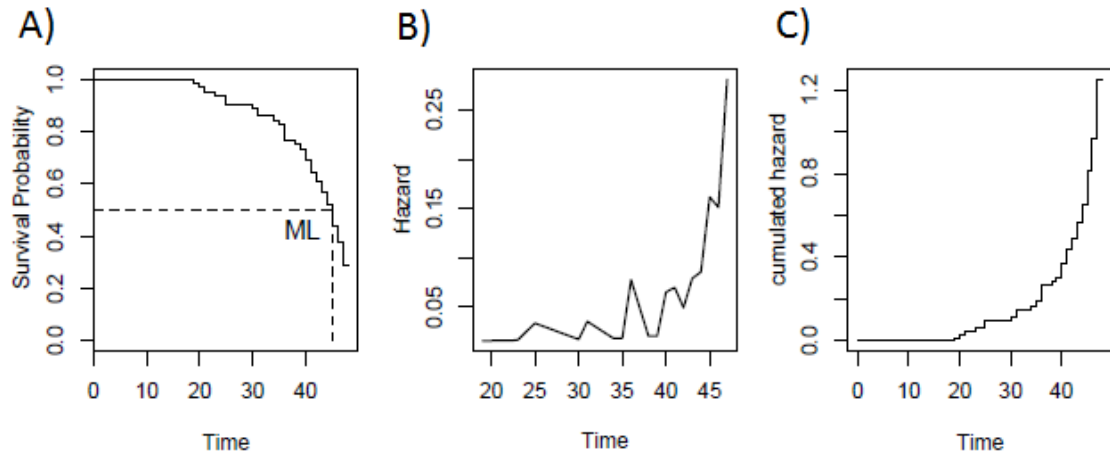


Figure 5.2.: A) shows the survival probability, B) the hazard, C) cumulated hazard for the couples-cope example; the event is defined as showing stress communication for the last time; the dotted line in A) indicates the median life time (ML)

(see Equation 5.2) can be estimated by dividing the number of female partners who communicate their stress (SC) for the last time (n events) by the number of female partners who did not end their SC until that particular interval. Thus, the denominator includes those female partners who communicate their stress for the last time and those who continue communicating their stress (n at risk). If couples cannot be observed for any reason, they are not at risk and are referred to as censored data. Censoring can occur because observational units leave the experiment unexpectedly or because they did not show the event until the last time interval.

$$\text{Hazard for interval } j = \frac{n \text{ events in time interval } j}{n \text{ at risk in interval } j} \quad (5.2)$$

The hazard for the sample data can be seen in Figure 5.2.B: the hazard is 0 at the beginning and stays this way until interval 19, indicating that none of the couples have ended their SC before. At interval 19, the first couple ends their SC; therefore, the hazard rises from zero to .0156 (1 event, 64 at risk). At interval 20, the hazard slightly increases to .0159 (1 event, 63 at risk: one out of the 64 couples has experienced the event before!). Over time, intervals with a hazard greater than zero become more frequent, yet the hazard itself is still low ($h_{t=21}=.02$; $h_{t=23}=.02$; $h_{t=25}=.03$; $h_{t=30}=.02$;

$h_{t=31}=.04$; $h_{t=34}=.02$; $h_{t=35}=.02$; $h_{t=36}=.08$). Starting from interval 38, the hazard is constantly present (at every interval, at least one couple ends their SC) and increases dramatically over time ($h_{t=38}=.02$; $h_{t=39}=.02$; $h_{t=40}=.06$; $h_{t=41}=.07$; $h_{t=42}=.05$; $h_{t=43}=.08$; $h_{t=44}=.08$; $h_{t=45}=.15$; $h_{t=46}=.14$); it peaks at time interval 47, with a value of .25 (6 events, 24 at risk). Plotting the hazard shows if and how it depends on time. In this example, it seems that a minimum amount of time is needed to end SC (no events before interval 19). Furthermore, the longer SC has been maintained, the more likely it will end.

The cumulated hazard function is the sum of all interval-specific hazards preceding a particular interval. It is shown in Figure 5.2.C. The cumulated hazard is zero until interval 19 because it is also zero until that interval. At interval 19, the hazard is .0156, and the cumulated hazard also becomes .0156 ($0 + .0156$). At interval 20, the hazard is .0159 and the cumulated hazard becomes .0315 ($0 + .0156 + .0169$). At the last interval, it reaches 1.19, which is the sum of all hazards over all intervals. The cumulated hazard cannot be easily interpreted, and its boundaries are zero and positive infinity. However, most statistical software, such as SPSS or Stata, use the cumulated hazard alongside the survival function for visualization of time-to-event data. Hence, it is shown in this chapter.

5.2. Analyzing the Influence of Covariates

Predicting interindividual differences (or differences between couples) in hazard can be done by using Cox's proportional hazard model (Cox regression; Cox, 1972). In Cox regression, the time-specific hazard of an observational unit (i.e., couple) is compared to the time-specific baseline hazard. The time-specific baseline hazard is defined as the hazard for all units, with all covariates equal to zero. The baseline hazard can vary freely between all intervals and is not restricted in any way. Covariates can be time-independent (e.g., sex or traits) or time-dependent (e.g., states).

5.2.1. Time-Independent Covariates

If time-independent covariates are mean centered, the time-specific baseline hazard becomes the mean hazard, as shown in Figure 5.2.B for the couples-cope data. For the following analyses, all covariates are assumed to be mean centered. The different parameterizations of the Cox regression are shown in Equations 5.3 (logit notation) and 5.4 (hazard-ratio notation).

$$\ln \left(\frac{h_i(t)}{h_0(t)} \right) = \beta_1 x_{1i} + \dots + \beta_k x_{ki} \quad (5.3)$$

$$\Leftrightarrow \frac{h_i(t)}{h_0(t)} = \exp(\beta_1)^{x_{1i}} * \dots * \exp(\beta_k)^{x_{ki}} \quad (5.4)$$

For the couples-cope example, men's self-assessed ability of dyadic coping is added as a mean centered covariate. Fitting the model on the data results in a positive, but non-significant effect ($\exp(\beta_1) = 1.16$, $p = .216$); therefore, inferring that a true effect exists is not allowed. Nevertheless, the $\exp(\beta_1)$ can be interpreted descriptively for exemplification purposes: for an individual with a value of zero on the covariate (0 = average self-assessed ability), the predicted hazard ratio becomes one; meaning that the individual hazard is equal the baseline hazard shown in Figure 5.2.B. However, if a person rates her- or himself one point higher than the average, the hazard ratio becomes 1.16 ($\exp(\beta_1)^1$) indicating that the predicted hazard for this individual is 1.16 times higher than the mean hazard for all time intervals. Conversely, if an individual rates him- or herself one point lower than the average mean, the predicted hazard is 0.86 ($\exp(\beta_1)^{-1}$) times the mean hazard. Because of that relationship between hazard and $\exp(\beta_1)$, it is sometimes referred to as a hazard ratio (HR).

The higher the hazard, the more likely it is that stress communication will not be shown after that time interval. Thus, the higher the men's self-assessed ability of dyadic coping, the more likely it is that a couple ends their SC at any given time interval, resulting in a shorter duration of SC. Hence, the direction of the effect points in the expected direction. Dyadic coping should support the partner's efforts in dealing with stress. Therefore, a better dyadic coping ability of one partner should shorten the duration of stress, and thereby the duration of stress communication as well. However, the effect is not significant, and therefore, inference on a wider population is not allowed.

Figure 5.3 shows the predicted hazards for the example data. The straight line shows the baseline hazard, the hyphenated line shows the hazard for couples with a value of five points above average on the covariate ($HR = 1.16^5 = 2.07$), and the dotted line shows the hazard for couples with a value of five points below the average on the covariate ($HR = 1.16^{(-5)} = 0.48$). The values of plus (+) and minus (-) 5 were chosen solely for demonstration purposes so that the HRs became close to 2 and to 0.5. Hence, for couples in which the men's mean centered self-assessed ability of dyadic coping is 5, the hazard is doubled across all time intervals. If the men's mean centered

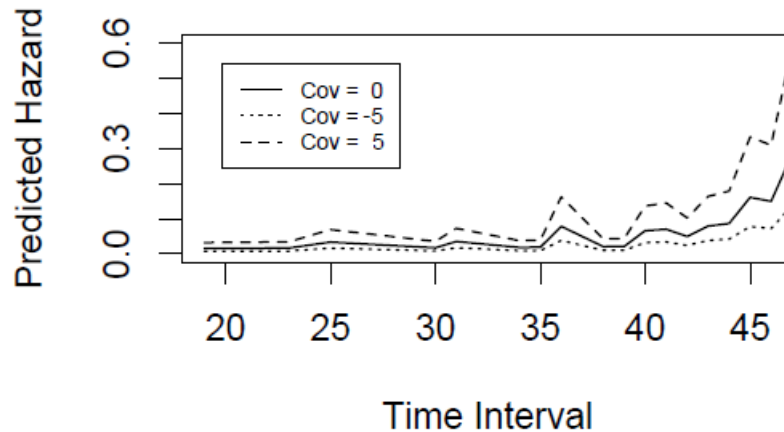


Figure 5.3.: The y-axis shows the predicted hazard for the couples-cope data. The straight line depicts the baseline hazard; the dotted line shows the hazard for couples with a value of 5 on the covariate, and the hyphenated line presents the hazard for couples with a value of minus 5 on the covariate.

self-assessed ability of dyadic coping is minus 5, the hazard is halved. Therefore, the hazard function form does not change due to the covariate. That is important to note because it reflects the only parametric assumption of the Cox regression, the so-called *proportional hazards* assumption.

5.2.2. Second Sequence as a Time-Dependent Covariate

The previous chapter showed how to transform one sequence into time-to-event data and how to add a time-independent covariate. However, time-dependent covariates can also be added, and categories can be added via dummy- or effect-coding. Therefore, it is possible to apply time-to-event analysis on dyadic sequence data. However, one limitation is that the researcher has to choose one sequence as the dependent variable and the other sequence as the independent variable.

Equation 5.5 shows the Cox regression for time-dependent variables. Covariates are now indexed with t as well. However, the effect of the variable (β) is not indexed with t . That means that the covariate might change over time, yet the effect of the covariate on the hazard is constant. The effect of dyadic coping, for example, is assumed to be the same at all time intervals (assumption of *constant covariate effects*). The covariate itself - for example, whether dyadic coping was shown or not - changes over time.

$$\frac{h_i(t)}{h_0(t)} = \exp(\beta_1)^{x_{1ti}} * \dots * \exp(\beta_k)^{x_{kti}} \quad (5.5)$$

In order to estimate the model, time is sliced into chunks in which the covariate is constant. Considering the following sequence where "S" stands for Stress, "C" for coping, and "0" for no behavior: "S0 – S0 – SC – SC – 0C – 0C," if coping is the covariate, the sequence is sliced into the following two: "S0 – S0" and "SC – SC – 0C – 0C." These sequences are now transformed into the SPELL format; that is, each line of the data matrix corresponds to the start and the end times of the covariate. Additionally, one column provides the information regarding whether the event occurred, and another provides the ID of the original sequence. Thus, the primary challenge of applying this model is not in specifying it, but rather structuring the data carefully (see Appendix A.3).

For the couples-cope example, a significant effect is estimated for dyadic coping (DC) on stress communication (SC) $\exp(\beta) = 1.72$, $p = .003$. Therefore, if DC is present, the *risk* for stopping SC is 1.72 times higher than without DC. This indicates that DC is beneficial for ultimately ending SC, probably because dyadic coping reduces or even solves stress.

5.3. Shared Frailty Model: A Bivariate Survival Model

Frailty models address the unobserved heterogeneity of multivariate time-to-event data (Crowder, 2012). They can be used for modeling paired observations of time-to-event variables. Hence, a researcher does not need to specify one sequence as dependent and the other as an independent variable. Instead both serve as dependent variables, and hence both members of a dyad can be analyzed simultaneously. Additionally, a frailty coefficient is estimated that stands for temporal proximity of the time-to-event variables.

The shared frailty model, a special case of frailty models, will be presented in this chapter because it is a direct extension to Cox-regression. Thus, the core idea of frailty models can be illustrated as an extension to the models already shown. Another feature is that as the Cox-regression so does the shared frailty model not assume a certain distribution for the hazard.

Equation 5.6 shows the model equation for a log-normal shared frailty model (Rondeau et al., 2012). The original notation was adapted for this monograph to fit the

previous models for time-to-event analysis. Hazards are now specific for the individuals (i), which are nested in dyads (j). Hazards are supposed to be similar within dyads, but dissimilar between dyads. This dissimilarity is expressed by η , the so-called frailty. That is a latent variable following a normal distribution. The variance of η , σ^2 can be estimated and tested. High and significant values indicate that differences between hazards can be explained by dyad membership rather than the type of behavior that is observed (e.g., whether the hazard of SC or DC is analyzed). By contrast, low variances indicate that a dyad membership does not explain differences between hazards very well. In other words, high and significant values of σ^2 indicate that there is a latent variable that is shared by members of one dyad.

$$\frac{h_{ij}(t)}{h_0(t)} = \exp(\eta_i) * (\beta_1)^{x_{1tij}} * \dots * \exp(\beta_k)^{x_{ktij}} \quad (5.6)$$

$$\eta_i \sim N(0, \sigma^2)$$

For the interpretation of σ , it is useful to note that η_i has the same metric as a hazard ratio. For example, if η_i is zero, the hazard is taken by one ($\exp(0) = 1$). Thus, the hazard stays the same as the baseline hazard. If η_i is 1, for example, for an observational unit, then both hazards (SC and DC hazard) would be 2.72 times as high as the baseline hazard. Keeping that in mind and combining it with the fact that the latent variable is normally distributed allows for a rough interpretation. For example, $\sigma = 1$ means that about 16% of the sample will score 1 or higher. Therefore, about 16% will have at least 2.72 times the hazard as a dyad with $\eta_i = 0$. The same goes for the lower 16% that will have taken their hazard by at least 1/2.72 (or an even smaller value). By contrast, $\sigma = 3$ would mean that the hazard for the upper 16% is taken by 20 ($\exp(3) = 20.09$) and for the lower by 1/20. Thus, the higher σ , the more heterogeneous the dyads.

The model was fitted to the couples-cope data. Events were defined as showing SC and DC for the last time, and men's self-assessed ability of dyadic coping was used as an additional time-independent covariate. The estimated frailty was relatively high and significant $\sigma^2 = 12.69$, $p < .001$, indicating that the hazard of the upper 16% of the sample is 35.25 times bigger than the baseline hazard ($\exp(\sqrt{12.69})$). The effect of men's self-assessed ability is still positive and becomes significant for this model $\beta = 1.3713$, $p < .001$, indicating that the higher the self-assessed ability, the higher the hazard, and thus, the shorter the duration of SC and DC. One possible explanation for the significant effect of the dyadic coping ability is that power was increased because

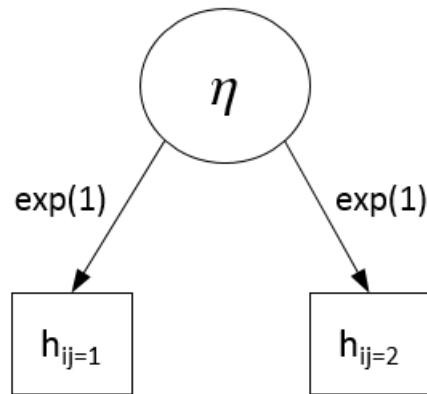


Figure 5.4.: depicts a frailty model for paired observations. Each time-to-event variable is interpreted as an indicator for an underlying latent variable η .

the number of observations was increased. By adding the time-to-event variables for both SC and DC, the number of observations was doubled.

5.4. Assumptions, Needed Sample Sizes, Practical Issues

The Cox regression is commonly referred to as Cox's proportional hazard model because it is based on the assumption of proportional hazards, as depicted in Figure 5.3. Otherwise, the Cox regression comes without any further parametric assumptions; hence, it is often said to be a semi-parametric model. It does not assume any distribution for the baseline hazard. One further assumption which was mentioned earlier in this chapter is that no ties exist. Nevertheless, Cox regression can be applied to data with ties when the Efron approximation (Efron, 1977) is used, and the sample size is greater than 50, or the maximum number of ties per time interval does not exceed 20% of the sample size (Hertz-Picciotto and Rockhill, 1997). The final assumption is that the effect is constant at all time intervals. These assumptions apply to all three of the above in the case of time-independent or time-dependent variables. The frailty is also assumed to have a constant effect on the hazard (across time and observational units).

A rule of thumb for the needed sample size for the Cox regression is that at least ten events per covariate are needed (Peduzzi et al., 1995), or else estimates might become unstable. In some cases, that rule can be relaxed down to 5-9 observations per covariate (Vittinghoff and McCulloch, 2007). However, minimum sample size recommendations regarding Type-II errors are rare. Because of that, a simulations study was conducted

(see 10.1), investigating how the true β affects the needed sample size. Figure 5.5 shows the main results of the study. For example, if a researcher aims for a power of .80 and believes that the true effect is relatively strong ($\beta = 1$ or $\beta = 2$), small sample sizes, such as $N = 20$, seem to be sufficient. For $\beta = 0.80$, the sample size should be $N = 30$, for $\beta = 0.60$ it is $N = 40$, for $\beta = .40$ it is $N = 80$, and finally for small effects, such as $\beta = .20$, it should be between $N=200$ and $N=500$ (outside of Figure 5.5).

The recommended length of the observation period is 30 time intervals if N is at least 20 and if the baseline hazard is at least .05 or higher. However, biases can still occur if the true effect size is extremely high (see 10.1). In general, the observation period needs to be longer the lower the baseline hazard and the bigger the effect size.

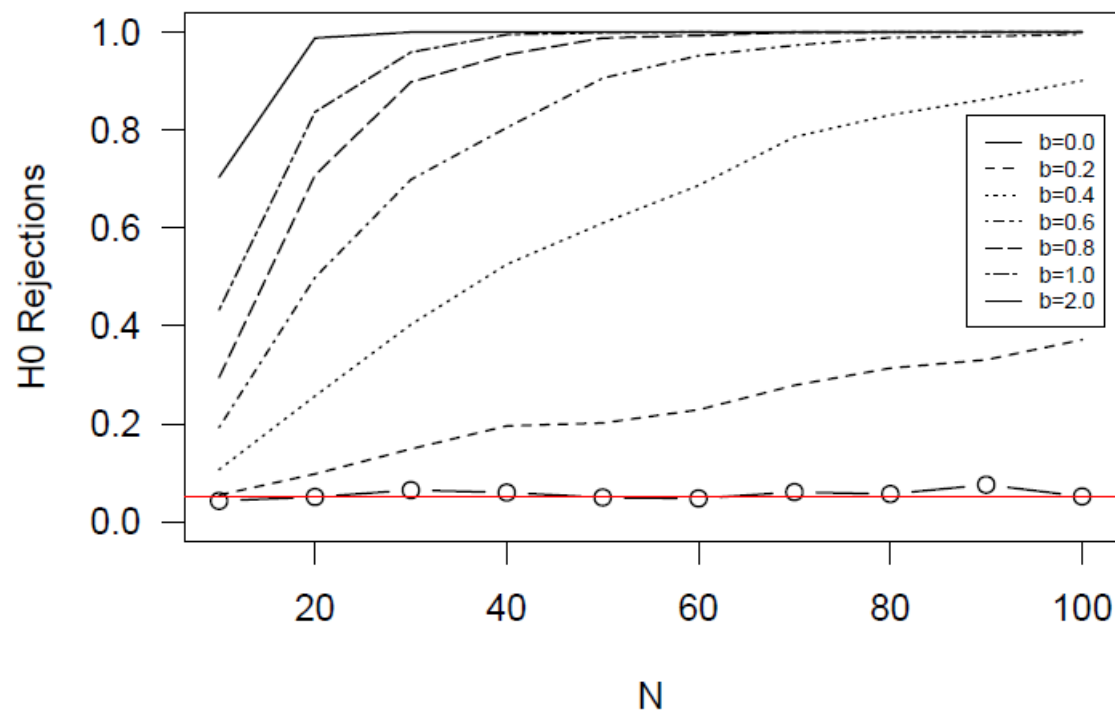


Figure 5.5.: The red line shows the nominal Type-I error rate for different sample sizes; the line interrupted by dots ($\beta = 0.0$) show the simulated Type-I error rate; the other lines are depicting statistical power dependent on sample and effect sizes.

As for the frailty model, the simulation revealed a strong bias for small samples sizes regarding the frailty coefficient, which also leads to a highly inflated Type-I error. In all cases with true zero frailty, the inferential tests became significant (see 10.2). However,

the idea that the hazard of nested data depends on a latent variable perfectly fits the needs of dyadic sequence analysis. Therefore, future research might investigate why the model performed badly in the simulation and improve its estimation.

6. Analyzing Dyadic Sequences Using Aggregated Logit Models (APIM)

In Chapter 3.4, several research questions that can arise in light of dyadic sequence data. One of these questions was "Does a particular behavior by one partner evoke a specific, prompt reaction by the other?", another one was "What is the stability of DC/SC?" (actor effects)?" Bakeman and Gottman (1997) used logit models to answer these questions. Logit models can be seen as an ANOVA for dichotomous dependent variables (Eid et al., 2010). That is, one or more factors (independent variables) influence the frequencies of the dependent variable. Hence, the data to be analyzed takes the form of frequency tables.

Therefore, in a first step, the sequences have to be transformed into so-called *state – transition tables*. The behavior of interest (e.g., dyadic coping) in interval t is mapped against the combination of the observed behaviors in the preceding interval $t - 1$ (e.g., stress communication by women and dyadic coping by men). This is done separately for every pair of sequences (e.g., there are as many state-transition tables as couples.) In the second step, logit models are estimated separately for each of the couples (single case analysis). The odds of showing the behavior of interest versus not showing the behavior are predicted by the preceding states of both sequences (e.g., behaviors of the two partners). In a final step, the parameters of a single case analysis are aggregated across all couples and tested to be unequal to zero.

Additionally, it is possible to compare the aggregated parameters of certain groups. Thereby, it is possible to test questions such as "Is the probability that a partner will respond with dyadic coping promptly if his or her partner shows stress communication higher in the group in which women were stressed compared to the group in which men were stressed?"

6.1. State-Transition Tables

The data, as mentioned above, must be transformed into state-transition tables before logit models can be applied. Therefore, one sequence must be declared to be the dependent variable. Ideally, the sequence describes the states that are assumed to be triggered by certain states of the other sequence. Taking the *Couples-Cope* dataset as an example, it is possible to consider that stress communication by one partner could trigger dyadic coping responses by the other partner.

An example of a corresponding state-transition table is shown in Table 6.1 for the couple with the ID number 129. On the right side, the frequencies of the dependent variable (dyadic coping at time t) can be seen. The table displays that the male partner of couple 129 showed dyadic coping responses in 30 time intervals ($23 + 1 + 3 + 3$), whereas he did not show any dyadic coping response in 17 time intervals ($4 + 1 + 1 + 11$). The first row shows that 27 ($23 + 4$) time intervals exist in which the female partner showed stress communication (SC) and the male partner showed dyadic coping reactions (DC). In 23 of those time intervals, the DC was kept up until the next interval (first column) and not in four (second column). The second row shows that intervals with SC, but without DC, are rare for couple 129. Such intervals occurred only two times: one time followed by DC and one time not. The third row shows that time intervals without SC, but with DC are also relatively rare. They occurred only four times, three times followed by DC and one time without being followed by DC. The last row shows that intervals without SC or DC occurred 14 times, three of which were followed by DC and 11 which were not.

Overall, the state-transition table for couple 129 displays that dyadic coping reactions clearly depend on the previous behavior. It becomes more likely when it is preceded by stress-related behavior (SC and DC) than if it is not preceded by SC or DC.

Table 6.1.: State-Transition Table for Couple ID 129

Prior behavior ($t-1$)		Dyadic coping (t)	
SC	DC	Yes	No
yes	yes	23	4
yes	no	1	1
no	yes	3	1
no	no	3	11

Notes: SC: Stress Communication; DC: Dyadic Coping

6.2. A Single Logit-Model for Single Case Analysis

Equation 6.1 shows a logit model for our example data. The left side (dependent variable; criterion) of the equation shows the logarithmized odds of showing DC against not showing DC for the male partner in a given interval (t), the so-called *logit*. It is predicted by the SC of the female partner in the preceding interval ($t - 1$), the DC behavior by the male partner in the preceding interval ($t - 1$) and the interaction of the SC and DC in the previous interval.

$$\ln \left(\frac{P(DC_t)}{1 - P(DC_t)} \right) = \beta_0 + \beta_1 * SC_{t-1} + \beta_2 * DC_{t-1} + \beta_3 * SC_{t-1} * DC_{t-1} \quad (6.1)$$

This notation has numerous advantages, for example, the computation of predicted values is exactly the same as in a two-way ANOVA and the estimates for the betas are approximately t -distributed (Kenny et al., 2006). However, the metric of the logits makes the direct interpretation of the coefficients difficult. Therefore, the exponential function $exp()$ is typically applied to both sides of the equation, resulting in Equation 6.2, and the following interpretation for the exponented β s: $exp(\beta_0)$ represents the base odds of the probability for showing the behavior of interest against the probability for not showing the behavior (grand mean). This base rate is influenced by the preceding SC ($exp(\beta_1)$), the preceding DC ($exp(\beta_2)$), and their interaction ($exp(\beta_3)$).

If a certain behavior, for example, stress communication, was shown (SC = 1), the exponent of $exp(\beta_1)$ becomes one, and the base rate is taken times $exp(\beta_1)$. Conversely, if stress communication was not shown (SC = -1) the exponent of $exp(\beta_1)$ becomes minus one, and the base rate is taken times $\frac{1}{exp(\beta_1)}$. It is the same for DC and $exp(\beta_2)$.

However, interpretation is more complex for the interaction term. The latter is the product of SC and DC; and therefore becomes one if SC and DC are both shown or if both are not shown; but becomes minus one if only one of the two is shown. Therefore, if DC and SC are congruent, the base rate is taken by $exp(\beta_3)$, and if not, it is taken by the inverse of $exp(\beta_3)$.

$$\frac{P(DC_t)}{1 - P(DC_t)} = exp(\beta_0) * exp(\beta_1)^{SC_{t-1}} * exp(\beta_2)^{DC_{t-1}} * exp(\beta_3)^{SC_{t-1} * DC_{t-1}} \quad (6.2)$$

For the couples-cope example, the estimated β s and $exp(\beta)$ s of the logit model of the couple with the ID 129 are shown in Table 6.2. The grand mean (β_0) is 0.33, suggesting that dyadic coping occurs more often than no dyadic coping within this couple; to be precise, 1.39 times more often ($exp(\beta_0)$). The main effect for previous SC is significant and positive ($\beta_1 = .92$), indicating that if DC was shown in the previous time interval, the chance that it will be shown again is 2.52 ($exp(\beta_1)$) times higher than the grand mean suggests (*Partner Effect*; Kenny et al., 2006). The main effect of previous DC is also significant and positive ($\beta_2 = .50$), where the chance that DC is shown is 1.65 ($exp(\beta_2)$) times higher when DC was also shown in the previous interval (*Actor Effect*).

Table 6.2.: Results of the Logit-Model for Couple ID 129

	β	$exp(\beta)$
Grand mean	0.33	1.39
SC at t-1 (Actor Effect)	0.92***	2.52
DC at t-1 (Partner Effect)	0.50***	1.65
Interaction Effect	-0.10	0.91

Notes: * $p < .05$; ** $p < .01$; *** $p < .001$

Finally, the interaction effect shows a negative estimate: if one effect is present, it weakens the other effect. Even though the interaction is not statistically significant, it is still part of the equation for predicting the odds. For example, the predicted odds for couple 129 showing DC if SC and DC were shown in the previous interval are 5.26 ($\approx 1.39 * 2.52 * 1.65 * 0.91 = exp(\beta_0) * exp(\beta_1) * exp(\beta_2) * exp(\beta_3)$). Or, for example, if SC, but not, DC were not shown in the previous interval, the odds would be 2.33 ($\approx 1.39 * 2.52 * \frac{1}{1.65} * \frac{1}{0.91} = exp(\beta_0) * exp(\beta_1) * \frac{1}{exp(\beta_2)} * \frac{1}{exp(\beta_3)}$).

An overview for all predicted odds is displayed in Table 6.3. The table also shows predicted logits and probabilities. The transformation from logits into odds is $odds = exp(logit)$ and from odds into logit is $logit = ln(odds)$. The transformation from odds into probabilities is $P = Odds / (Odds + 1)$ and from probabilities into odds is $Odds = P / (1 - P)$.

6.3. Aggregating Results for Group Analysis

In the second step, the N logit models (one for each couple) are aggregated to examine the overall pattern in the sample. Since the estimates for the β -parameters are t-distributed over the 64 tables (Kenny et al., 2006), they may be aggregated by taking

Table 6.3.: Predicted Logits, Odds, and Probabilities for couple ID 129

		Logit		Odds		Probabilities	
		DC_{t-1}		DC_{t-1}		DC_{t-1}	
		Yes	No	Yes	No	Yes	No
SC_{t-1}	Yes	1.65	0.85	5.26	2.33	.84	.70
	No	< 0.01	-1.19	1.01	0.30	.50	.23

Notes: SC_{t-1} : Stress Communication at t-1; DC_{t-1} : Dyadic Coping at t-1

their mean value and allowing for standard hypothesis testing with N (couples) as a number of observations.

Table 6.4 (first part: "DC as Dependent Variable") depicts the aggregated estimates for the 64 couples of the sample data. Overall, the baseline parameter, the partner effect that SC at t-1 leads to DC at t, and the actor effect that DC at t-1 leads to DC at t are positive and significant. Therefore, it is principally more probable to observe DC than no DC, and the probability increases with preceding stress-related behaviors (either SC and/or DC), yet there is no significant interaction effect.

Table 6.4.: Averaged Logit Parameters Over all 64 Couples

	β	$exp(\beta)$
DC as Dependent Variable		
grand mean	0.25**	1.28
SC at t-1 (actor effect)	0.79***	2.21
DC at t-1 (partner effect)	0.70***	2.01
interaction effect	0.04	1.04
SC as Dependent Variable		
grand mean	0.28**	1.32
actor effect	1.05***	2.85
partner effect	0.52***	1.68
interaction effect	0.10	1.11

Notes: * $p < .05$; ** $p < .01$; *** $p < .001$; SC is stress communication shown at $t - 1$ (-1="no"; 1="yes"); DC is dyadic coping shown at $t - 1$ (-1="no"; 1="yes")

6.4. Actor-Partner-Interaction Model

In dyadic data analysis, the Actor-Partner-Interdependence model (APIM; Kenny et al., 2006, see Chapter 1.3.1) plays a major role. Estimating a second aggregated logistic model with women's SC as the dependent variable would lead to an analogous APIM analysis regarding aggregated logit models. The estimates for that model can be seen in the second part of Table 6.4 ("SC as Dependent Variable").

Figure 6.1 presents the estimates as a graphical representation of the APIM. The actor effect of DC is smaller than the actor effect of SC, indicating that stress communication is less volatile than dyadic coping responses. In addition to the results for DC, the model shows that a preceding SC leads to the following SC, and that preceding DC also leads to SC. That indicates a bidirectional relationship. However, the partner effect from SC to DC is bigger than the partner effect from DC to SC. Overall, the APIM shows that stress communication seems to be the driving force of the dyadic coping process.

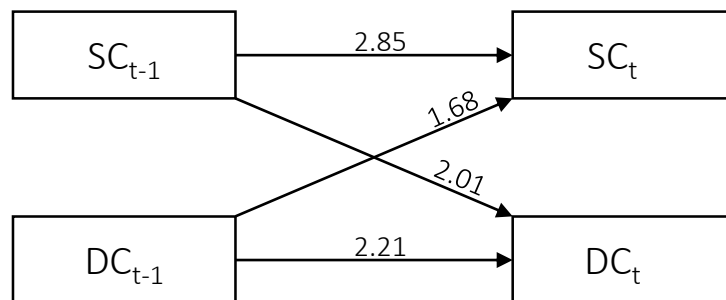


Figure 6.1.: Estimates are odds ratios for showing SC/DC predicted by previous SC/DC at $t-1$. The odds ratios from SC and DC at $t-1$ to DC at t are the same odds ratios as in table 6.4. The other odds ratios are obtained by the same procedure, but with SC as the dependent variable instead of DC.

6.5. Alternative Example (Give-Some)

Table 6.5 displays the results for the give-some dataset. The grand mean is bigger for the cooperation of the algorithm than for the cooperation by the human player, reflecting the fact that the algorithm cooperated more often than the human player (see 2.2).

There is a strong, negative partner effect for the algorithm. Therefore, if the human player cooperated at $t - 1$, it is more likely that the algorithm will not cooperate in the next turn. Furthermore, there is also a negative partner effect for the human player; thus, if the algorithm did cooperate the last turn, cooperation by the human player becomes less likely.

Both negative partner effects can be explained best by the following: the algorithm always gives the same amount. Humans might learn that the algorithm does not change its behavior whatever they do. Still, players differ regarding their mean number of given coins. Some give more, some give less than the algorithm. Thus, if a human tends to give more coins than the algorithm, he or she will cooperate, which is followed by uncooperative behavior most of the time (the algorithm gives less). And, vice versa, if the human tends to give less, she or he will produce uncooperative behavior, which is most likely followed by cooperative behavior because the algorithm tends to give more coins.

Furthermore, a positive actor effect for the algorithm is revealed. The effect is, however, relatively small. Surprisingly, there is only a very weak and not significant actor effect by the human player, indicating that previous cooperative behavior is not predictive for future cooperation in that specific setting. Moreover, there are no interaction effects for both models. Thus, actor and partner effects are independent of each other.

Table 6.5.: Averaged Logit Parameters for the Give-Some Example

	β	$exp(\beta)$
DV: cooperation by human player		
grand mean	0.67***	1.96
actor effect	-0.07	0.93
partner effect	-0.24***	0.79
interaction effect	0.02	0.98
DV: cooperation by algorithm		
grand mean	0.98***	2.66
actor effect	0.11*	1.12
partner effect	-0.64***	0.53
interaction effect	-0.02	0.98

Notes: * $p < .05$; ** $p < .01$; *** $p < .001$; actor effect: was cooperation shown at $t - 1$ by the same player (-1="no", 1="yes"); partner effect: was cooperation shown at $t - 1$ by the other player (-1="no", 1="yes").

6.6. Assumptions, Needed Sample Sizes, Practical Issues

The aggregated logit model assumes a time-homogeneous autoregressive process. That is, the probability of showing a certain behavior in one time interval depends only on the previous time interval. Thus, for all possible value of t , the following should be true: $P(s_t)|s_{t-1}$. However, according to Helske and Helske (2016), such models can still be useful for describing data, even if stationary cannot be assumed. Alternatively, semi-Markov models can be used (an extension of Markov models) because they do not rely on this assumption (Yu, 2010).

Moreover, some behavior may be shown rarely or never. However, one assumption of the logit models is that all state-transition tables show no zero frequencies. One way of dealing with that is to exclude all observational units that show zero frequencies. The problem with that approach is that if zero frequencies occur systematically, it may lead to biased estimations. An alternative is adding a constant, typically 0.5, to all cells (Everitt, 1992), to make the model estimable at the cost of a weak bias toward lower effects. It is, therefore, a conservative approach for handling zero frequencies.

Furthermore, it is advised that the predicted cell frequency in a logit model should be at least five for every cell. That is especially important for single case analysis. In this case, Hope's (1968) Monte Carlo test should be used for statistical inferences. The accompanying R-Package 'DySeq' provides two functions (EstFreq and EstTime) that address the issue of low and zero frequencies. EstFreq simulates the estimated number of cells with low or zero frequencies, depending on the expected mean state-transition-table. The other function, EstTime, computes the minimum number of time intervals that result in as many cases with low or zero frequencies as considered tolerable by the researcher. Both functions are only implemented for cases with two dichotomous coded behaviors of interest.

Regarding the sample size and optimal sequence length, a simulation study was conducted (see Chapter 10.3). There is a small bias in estimated actor effects that decreases with longer sequences. For example, the mean bias on β is -.08 for sequences with ten time intervals, and β is -.02 for sequences with a hundred time intervals. Thus, longer sequences should be preferred. Moreover, the combination of very small sequences and very big samples sizes will result in increased Type-I errors. Thus, this combination should be avoided.

All other effects behave normally, however only one effect was tested per time. Using the downward biased actor effect as a lower bound approximation, and aiming for a power of at least .80, reveals that effects with low effects sizes ($\beta = 0.2$; OR= 1.22)

should at least include 50 or more time intervals with $N = 30$. If sample sizes are bigger (80 or more), 30 time intervals are sufficient. For medium effect sizes ($\beta = 0.4$; OR = 1.49), even very small samples ($N = 10$) can be estimated as long as the sequences include at least 70 intervals. Vice versa, extremely short sequences can be estimated so long as the sample size is big ($N = 80$ or more). However, as stated above, that particular combination should be avoided due to the increased Type-I error for actor effects. For bigger effect sizes ($\beta = 0.6$ and 0.8 ; OR = 22.3 and 1.82), thirty observational units with thirty observations each are sufficient.

These results are limited to cases in which positive actor effects are assumed. That is, it is assumed that behavior is more or less stable. For instance, SC and DC are supposed to shown over consecutive time intervals. The opposite would be that behavior oscillates, yet most behavior in psychological research is not assumed to oscillate.

Please note that the categorization of *small*, *medium*, and *big* effect sizes are based solely on the results of the simulation study and reflect their impact on the power. An evaluation of effect sizes in terms of practical significance, however, should always depend on the subject.

7. Modelling Dyadic Sequences Using Multilevel-Models (APIM)

Another way for answering the research questions "Does a particular behavior by one partner evoke a specific, prompt reaction by the other?" and "What is the stability of DC/SC?" (actor effects)?" (the research questions from Chapter 3.4 is to specify an APIM via multilevel models. Instead of running multiple logit models per dyad and dependent variable, a single multilevel model can be used for estimating an APIM. To this end, the multilevel model has to address three challenges: the longitudinal aspect of sequence data, the fact that sequences are categorical, and the dyadic data structure. The longitudinal aspect can be addressed by the multilevel modeling approach itself because it accounts for dependencies within nested observation. Repeated measures can be seen as multiple observations that are nested within individuals (see Chapter 7.1). Multilevel models can be extended to generalized multilevel models (Hox et al., 2010), allowing for categorical dependent variables (see Section 7.3). Finally, the dyadic data structure can be addressed by incorporating a dummy coded moderator variable (e.g., sex), which distinguishes between the two members of a dyad (see Section 7.4).

7.1. The General Idea of Multilevel-Modeling

Multilevel models (also referred to as hierarchical linear models, mixed-effects models, random-effects models, random coefficient regression models, and covariance components models) deal with the analysis of hierarchical data structures; that is, observations are nested within groups. An often used example is educational research, where students are grouped in classes, classes are grouped in schools, and schools are grouped into districts. Therefore, variables describe individuals, but the individuals may be grouped into larger or so-called higher-order units. However, in the case of repeated measurement data, the individual is the higher order unit (replaces the group), and multiple measurements (observations of the individual) are nested within the individual.

Before multilevel models were common, linear regression models were used for the analysis of hierarchical data. Several strategies were used to handle the nested data structure: A) Ignoring the structure, B) aggregation of data, C) running multiple within-group analyses, and D) modeling level membership by interaction terms. Each strategy comes with certain problems.

(A) Inference statistics of linear regression are based on the assumptions of linearity, normality, homoscedasticity, and independence. If characteristics of higher levels (e.g., class and school) are attributed to the individuals (e.g., students), the assumption of independence is violated (Aitkin and Longford, 1986). That can lead to biased standard errors and an inflated Type-I error rate. Moreover, contextual effects of upper levels can overshadow individual effects as demonstrated in Figure 8.1. In that example, all groups show small positive correlations when regression is fitted separately, yet correlation becomes negative when one regression is fitted for the whole sample.

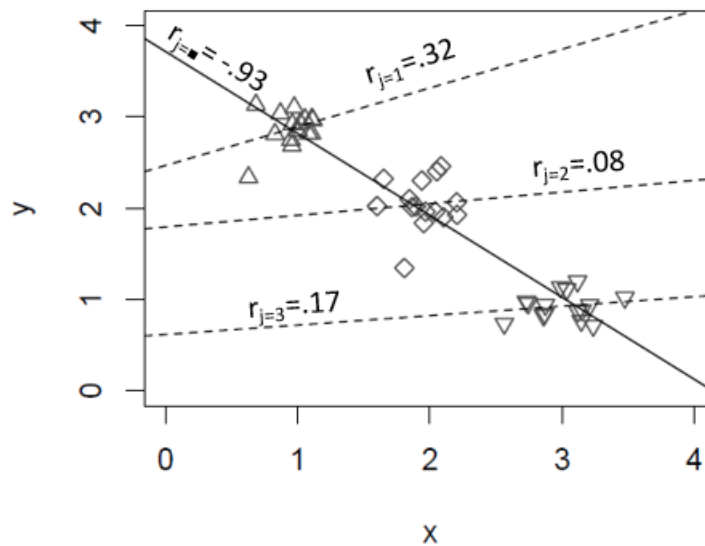


Figure 7.1.: Example of a nested data structure; hyphenated lines are regression lines estimated separately for all three groups ($j = 1$, upwards triangles; $j = 2$, diamonds; $j = 3$, downward triangles.) The straight line shows the regression line that is estimated for all individual data points, ignoring the nested data structure.

(B) Aggregating data and ignoring individual differences leads to a loss of information. Moreover, analysis results reflect only associations on their particular aggregation

level. Interpretation on a lower level is forbidden and may lead to an ecological fallacy (Ess and Sudweeks, 2001). Consider the following example: the aggressive behavior of students and parental income of students are aggregated (e.g., averaged) over schools. The analysis reveals that aggressive behavior is more frequent in lower income schools. We do not know whether or not low-income students are more aggressive than other students (loss of information), yet one might infer so nevertheless. That interpretation of the results leads to an ecological fallacy because the association can be different on an individual level (e.g., schools with students with a low average parental income may be located more often in urban areas, whereas those with high average parental income might more often live in the countryside). Aggressive behavior might differ between schools from different areas, which alone might account for the correlation on the school level. However, correlation of parental income and aggressive behavior of students within schools can still be correlated in any way including zero correlations or even positive ones.

(C) Running multiple within group analyses is not practical if dozens of schools are analyzed. Using this approach will result in as many correlations as there are higher level units. Moreover, if sample sizes are small within higher level units (e.g., only three students per class), power will be low. Therefore, the majority of regressions might show only non-significant results, even if true effects are present and even if the overall survey includes hundreds of students. Thus, that approach is not practical.

(D) Using interaction terms is the way to go when the number of higher-level units is small. For example, if data is gathered from students of three different classes, the class can be used as a moderator. Therefore, association coefficients may vary between classes, and all data is used at once (higher power). However, using this approach will still result in as many group-specific correlations (simple slopes/effects) as there are higher-level units. Thus, if the number of higher level units is big, multilevel models should be used instead.

Aitkin and Longford (1986) not only demonstrated the need for statistical analysis that can account for nested data, but they also propagated the use of multilevel models for this purpose. Multilevel equations are extended linear regression equations. In the first line, Equation 7.1 shows the level-1 equation for a generic two-level multilevel model. y_{ij} is the dependent variable (criterion), x_{ij} is the independent variable (predictor). Both are indexed with i for level-1, and j for level-2. Thus, if observational units at level-1 are students, i refers to a certain student and j may refer to her or his class. β_{0j} is the intercept, which is the value that is predicted if x is zero. β_{1j} is the slope for the predictor x , which is interpreted as "per unit increase of x , the predicted value of

y is increased by β_{1j} units of y ." That is the same as in ordinary regression, yet both β s can vary over j . Hence, even though they both have a value of zero on x , the predicted value of y_{ij} might differ between two students because the students belong to different classes with different intercepts. Moreover, the increase in y per unit increase of x can be different between classes.

Level – 1

$$y_{ij} = \beta_{0j} + \beta_{1j} * x_{ij} + e_{ij}$$

Level – 2

$$\beta_{0j} = \gamma_{00} + \gamma_{01} * z_j + u_{0j}$$

$$\beta_{1j} = \gamma_{10} + \gamma_{11} * z_j + u_{1j}$$

(7.1)

The second line of the equation depicts that the group-specific intercept depends on the grand intercept (γ_{00}), a random deviation (u_{0j} ; level-2 intercept residual), and moreover can further depend on a level-2 predictor (z_j). A level-2 predictor is the same for all individuals (i) that belong to the same group (j). For example, the experience of the class teacher in years is the same for all her or his students. Thus, more years of experience might increase the values for all students in y (e.g., test-score).

The third line refers to the slope of the level-1 equation. Similar to the second line, the slope is predicted by the grand mean slope (γ_{10}), a random deviation (u_{1j} ; level-2 residual for the slope), and can again depend on the previous level-2 predictor z_j . Therefore, higher values of z_j can strengthen or weaken the effect of the level-1 predictor the same way a moderator variable would do. That type of moderation is called *cross-level interaction*; z_j is a level-2 variable that influences a level-1 effect, and hence the name.

In multilevel analysis, only the *fixed effects* and the *variances of the random effects* are estimated and interpreted. Fixed effects include all parameters that are not allowed to vary between observational units of any level. Random effects are the residuals of lower order effects. High variances of random effects indicate that effects are very dissimilar between groups, whereas little variances mean that they are very similar.

The multilevel model presented in Equation 7.1 is a minimal example for introducing the core ideas of multilevel modeling. By practice, more predictors can be included, and more than two levels can be modeled. In contrast, not all multilevel

models must include cross-level interactions or predictors on all levels. In fact, one particular model without any predictors, the intercept-only model, is often estimated to decide whether a multilevel model is useful or not.

The intercept-only model (Hox et al., 2010) can be seen in Equation 7.2. It includes no predictors at all. The value of y_{ij} for an individual i is only predicted by group mean β_{0j} . Therefore, the variance of the individual residual e_{ij} is the variance within groups. The level-2 part of the equations shows that group means are predicted by the grand mean γ_{00} . Thus, the variance of u_{0j} is the between-group variance. The sum of both variances is equal the total variance of y_{ij} . Thereby, the proportion of variance that stems from the nested data structure, the intra-class correlation (ICC), can be calculated by the following: $var(u_{0j}) / [var(u_{0j}) + var(e_{ij})]$ (Hox et al., 2010).

$$\begin{aligned}
 & \text{Level} - 1 \\
 & y_{ij} = \beta_{0j} + e_{ij} \\
 & \\
 & \text{Level} - 2 \\
 & \beta_{0j} = \gamma_{00} + u_{0j}
 \end{aligned}
 \tag{7.2}$$

Multilevel models are difficult to estimate by ordinary least squares (OLS) because more than one type of residual has to be minimized at the same time. Instead, iterative numerical procedures are often applied to obtain the estimates. Standard estimation methods in most statistical software (SPSS, Stata, lme4) are full maximum likelihood (Longford, 1987) and restricted maximum likelihood estimations (Searle et al., 2009). Maximum likelihood estimation bears the benefit that nested models can be compared via a likelihood ratio test. Restricted maximum likelihood can be used to this end, too, as long as the fixed effect model is the same for all compared models. Alternative procedures include Bayes estimation (Browne et al., 2006) and iteratively reweighted generalized least squares (Goldstein, 1989).

7.2. Repeated Measures as Multilevel-Model

In the case of repeated measurement, the individual serves as the group j , with multiple measurements (i) nested within the individual. That leads to a 2-level model. The benefit of using multilevel models for repeated measure compared to other methods,

such as repeated measure ANOVA, is that the number of measurements must not be the same for all persons. Most important for the application shown in Chapter 7.4, multilevel models allow the inclusion of time-dependent covariates quite easily.

In Equation 7.1, if i represents a single data point measured over time, x_{ij} can also vary over time. Thus, time-dependent covariates can be included by adding them to the level-1 equation. By contrast, z_j cannot vary over time but only between individuals (j). Therefore, covariates that are included on level-2 are time-independent.

Another merit of longitudinal multilevel models is that time itself can be added as a level-1 predictor for modeling trends (e.g., linear, quadratic). If these trends are allowed to vary between persons (j), the multilevel model becomes a growth model (Hox and Stoel, 2005).

7.3. Generalized Multilevel Models

Multilevel models assume scaled (metric) dependent variables. However, generalized multilevel models can handle binary outcomes, ordered categorical outcomes, and multi-category, nominal scale outcomes. The topic is vast and is discussed by Gill (2000) in detail. This chapter, however, focuses only on those aspects that are important for Chapter 7.4, which demonstrates generalized multilevel models for dichotomous sequence data. Therefore, the adaptation of conventional multilevel models for binary outcomes (logistic multilevel) will be introduced briefly in this chapter.

$$\begin{aligned}
 & \text{Level} - 1 \\
 & L_{ij} = \ln \left(\frac{P(y_{ij} = 1)}{1 - P(y_{ij} = 1)} \right) \\
 & L_{ij} = \beta_{0j} + \beta_{1j} * x_{ij} + e_{ij}
 \end{aligned} \tag{7.3}$$

$$\begin{aligned}
 & \text{Level} - 2 \\
 & \beta_{0j} = \gamma_{00} + \gamma_{01} * z_j + u_{0j} \\
 & \beta_{1j} = \gamma_{10} + \gamma_{11} * z_j + u_{1j}
 \end{aligned}$$

Equation 7.3 depicts a logistic, multilevel model. Instead of predicting the outcome (y_{ij}) directly, a link variable (L_{ij}) is predicted by the multilevel model. On the right side of the equation, everything functions in the same way as in ordinary multilevel

modeling. There is one exception and that is the error distribution. Because L_{ij} is not assumed to follow a normal distribution anymore, the error terms have to be adapted. In the case of logistic regression, the error distribution is assumed to follow a binomial distribution. Finally, L_{ij} is linked via a logit transformation with the probability that the outcome becomes one (first line). The last principle is exactly the same as in Chapter 6.

7.4. A Multilevel Model APIM

As mentioned in this chapter's introduction, an APIM multilevel model for sequence data has to address three challenges: the longitudinal aspect of sequence data, the fact that sequences are categorical, and the dyadic data structure.

The longitudinal aspect can be addressed by modeling observed behavior as repeated measures on level-1 that are nested within couples at level-2. Each behavior should be predicted by the previous behavior at $t - 1$. Therefore, all but the first interval can be included as a level-1 observation. Thus, the example dataset provides 94 observations (two variables each measured at 47 time intervals) for each dyad.

The dependent variable for each interval is the occurrence of the behavior at time interval t (occurrence = 1; non-occurrence = 0), which can be handled by generalized multilevel models, as explained. The independent variables are effect coded; thus, the occurrence of the same behavior at $t - 1$ (actor effect) is coded with $AE=1$ for occurrence and with $AE=-1$ for non-occurrence. The same is true for partner effects, which are coded $PE = 1$ when the other behavior occurred at $t - 1$, and $PE = -1$ otherwise.

Finally, the dyadic data structure can be addressed by incorporating a dummy coded moderator variable (e.g., sex coded with *male* = 0 and *female* = 1), which distinguishes between the two members of a dyad. At first glance, it may seem inconsistent that the behavior is effect coded and that the moderator is dummy coded. However, that combination of coding strategies leads to relatively easy interpretations for an otherwise complex model. Equation 7.4 presents the corresponding model equations in logit parameterization:

Level – 1

$$L_{ij} = \ln \left(\frac{P(DV = 1)}{1 - P(DV = 1)} \right)$$

$$L_{ij} = \beta_{0j} + \beta_{1j} * AE_{ij} + \beta_{2j} * PE_{ij} + \beta_{3j} * AE_{ij} * PE_{ij}$$

$$+ \beta_{4j} * Sex + \beta_{5j} * AE_{ij} * Sex + \beta_{6j} * PE_{ij} * Sex$$

$$+ \beta_{7j} * AE_{ij} * PE_{ij} * Sex + e_{ij} \quad (7.4)$$

Level – 2

$$\beta_{0j} = \gamma_{00} + u_{0j}; \quad \beta_{1j} = \gamma_{10} + u_{1j} \quad ; \dots ; \quad \beta_{7j} = \gamma_{70} + u_{7j}$$

Conditional Level – 1 Equations for the Male Partners (*sex = 0*) :

$$L_{ij} = \ln \left(\frac{P(DC = 1)}{1 - P(DC = 1)} \right)$$

$$L_{ij} = \beta_{0j} + \beta_{1j} * AE_{ij} + \beta_{2j} * PE_{ij} + \beta_{3j} * AE_{ij} * PE_{ij} + e_{ij}$$

(7.5)

Conditional Level – 1 Equations for the Female Partners (*sex = 1*) :

$$L_{ij} = \ln \left(\frac{P(SC = 1)}{1 - P(SC = 1)} \right)$$

$$L_{ij} = (\beta_{0j} + \beta_{5j}) + (\beta_{1j} + \beta_{6j}) * AE_{ij} + (\beta_{2j} + \beta_{6j}) * PE_{ij}$$

$$+ (\beta_{3j} + \beta_{7j}) * AE_{ij} * PE_{ij} + e_{ij}$$

In equations 7.4 and 7.5, the dependent variable (DV) changes from male to female partners. If the males' behavior is to be predicted, DV represents dyadic coping (DC). If the females' behavior is to be predicted, DV represent stress communication (SC). In the same vein, AE represents male DC and PE represents female SC at $t - 1$ if the occurrence of male DC is predicted; AE represents female SC and PE male DC at the previous interval if the occurrence of female SC is predicted.

Equation 7.5 shows the conditional equations for men and women if the variable $sex = 0$ (upper conditional equation for male partners) or if $sex = 1$ (lower equation

for female partners). Hence, predicting DC behavior by male partners, the upper equation has to be interpreted; predicting female SC, the lower equation has to be interpreted. The partner with sex = 0 (male partners) represents the reference category. The corresponding average logit, actor, partner, and interaction effects can be obtained directly from the equation. For the non-reference category (sex = 1; female partners), the average logit, actor, partner, and interaction effects differ from the ones for the reference category by the logit parameters associated with the variable sex.

For example, β_{1j} is the actor effect for the reference category (male partners), and β_{5j} shows if the actor effect for the non-reference group (female partners) is larger (positive value) or smaller (negative value). The actor effect for female partners is, hence, depicted by (β_{1j} and β_{5j}). The same is true for partner effects (β_{2j} and β_{6j}) and the actor *partner interaction (β_{3j} and β_{7j}) considering female partners. The intercept (β_{0j}) represents the average logit for the reference group (male partners), and the main effect of sex (β_{4j}) shows the difference of female partners' intercept from the male partners' intercept.

Each level-1 β may differ between couples (j). The level-2 equations show that each couple's regression parameters depend on the average effect over all couples ($\beta_{\bullet j}$; fixed effects) and a couple-specific residual ($u_{\bullet j}$; random effect). The fixed effects at level-2 represent the average effects across all couples and conceptually correspond to the average parameters of the aggregated means approach. Additionally, the variances of the random effects indicate how much the regression effects differ between couples. For example, the variance of the level-2 random component associated with the intercept ($var(u_{0j})$) describes differences in the mean logits (for the reference group) between couples.

Moreover, correlations between random effects can be investigated. For example, a positive $cor(u_{1j}, u_{2j})$ indicates that larger (couple-specific) actor effects are associated with larger partner effects in the reference group. In another example, $cor(u_{5j}, u_{6j})$ is the correlation of the level-2 random effects associated with the differences between the female regression parameters and the corresponding male regression parameters. Also, a positive estimate of $cor(u_{5j}, u_{6j})$ indicates that differences in partner effects are larger for couples with larger than average differences in female actor effects. Thus, couples with higher female actor effects tend to produce higher female partner effects, and vice versa.

Models with different subsets of random effects can be tested against each other for finding a parsimonious model (Hox et al., 2010). For the example data, the comparative fit index BIC (Schwarz et al., 1978) was used. In the best fitting model, in-

7. Modelling Dyadic Sequences Using Multilevel-Models (APIM)

intercept $var(e_{0j})$, actor $var(e_{1j})$, and partner effects $var(e_{2j})$ were specified as random variables. Table 7.1 provides the fixed effects estimates of the generalized multilevel model. Overall, the fixed effects are very similar to the results of the aggregated logit approach. Direction and level of significance are the same for both methods. The multilevel approach tends to estimate slightly greater actor and partner effects, yet they are practically equivalent. The biggest difference is the actor effect for DC (.70 for aggregated logit models vs. .97 for multilevel approach), the smallest difference is between the partner effects for SC (.52 for the aggregated models vs. .49 for the multilevel model).

Table 7.1.: Fixed Effects for MLM-APIM (Couples-Cope Example)

	β for SC	β for DC	SC - DC
Mean Logit	0.25**	0.22*	0.03
SC at t-1 (Actor Effect)	1.26***	0.97***	-0.29***
DC at t-1 (Partner Effect)	0.49***	0.69***	0.21*
Actor*Partner Interaction	0.03	0.01	-0.02

Notes: * $p < .05$; ** $p < .01$; *** $p < .001$

Table 7.2 shows the random effects of the generalized multilevel model. The variances of the random effects can be found on the main diagonal. More interestingly, we find that the random intercept correlates negatively with the random parts of the actor and partner effects ($r = -.71$ and $r = -.66$), indicating that in couples with high base rates of SC and DC behaviors, the occurrence of these behaviors is less strongly associated with prior behavior than for couples with lower base rates. Random components of actor and partner effects correlate positively ($r = .77$), indicating that in couples with larger influences from SC at $t - 1$ on DC (or DC at $t - 1$ on SC), we also find larger influences from DC at $t - 1$ on DC (or SC at $t - 1$ on SC).

Table 7.2.: Random Effects for MLM-APIM (Couples-Cope Example)

	Mean Logit	Actor Effect	Partner Effect
Mean Logit	0.27		
Actor Effect	-.71	0.06	
Partner Effect	-.66	.77	0.17

Notes: * $p < .05$; ** $p < .01$; *** $p < .001$; Variances of level-2 residuals are on the principal diagonal; their correlations are shown in the other cells.

7.5. Alternative Example (Give-some)

Table 7.3 displays the model comparisons for the give-some example. The variable "Mod" serves the same purpose as sex did in the last chapter. Thus, a random effect of Mod means that differences between humans' and the algorithm's mean cooperation varies between players. Two models achieved the lowest AIC, both including a random intercept and a random effect of Mod. One of the two models also includes a random partner effect.

The model without the random partner effect was chosen for the following reasons: 1) If several models explain data equally, the parsimony principle requires to choose the simplest one. 2) BIC is smaller for the model with only a random intercept and the random effect of Mod. 3) plus, the model was the only one that converged beside the most basic multilevel model. Further models were tested, which included all kinds of interactions pairings, yet all those models failed to converge, too.

Regarding the random effects of the final model, variance of the intercept is relatively small ($var(u_{0j}) = .06$), whereas the variance for Mod is comparatively big ($var(u_{Mod,j}) = .89$), indicating that differences between human player's and the algorithm's cooperative reaction vary a lot between pairings. Both random effects show a strong negative correlation ($r = -.83$). Therefore, the higher the human's baseline probability that cooperation is shown, the more the difference between both baseline probabilities decreases. That makes perfect sense because the mean cooperation of the algorithm was higher (70.35%) for the algorithm than for the human players (63.98%). However, participants also showed variance in their mean number of given coins ($SD = .37$); therefore, some players might cooperate more often than others, and by doing so, decrease the difference between themselves and the algorithm.

The results of the fixed effects are presented in Table 7.4. Overall, the results are very similar to the results obtained by the aggregated logit models. Estimated effects of the multilevel model are all bigger than those of the aggregated logit models. However, all algebraic signs are the same except for the actor effect of the algorithm that was estimated to be positive and significant previously, but now is estimated to be negative and not significant.

Table 7.3.: Multilevel Model Comparisons (Give-Some Example)

Random Effects	AIC	BIC	Failed to converge?
Interc., AE, PE, AE*PE., Mod.	2957	3213	yes
Interc., AE, PE, and AE*PE	2970	3075	yes
Intercept, AE, PE, and Mod.	2935	3040	yes
Intercept, AE, and PE	2969	3050	yes
Intercept, AE, Mod.	2933	3015	yes
Intercept, PE, Mod.	2928	3010	yes
Intercept, Mod.	2928	2992	no
Intercept	2975	3027	no

Notes: Intercept is abbreviated as interc.; AE = actor effect; PE = partner effect; Mod. (moderator): is the dependent variable coded as 0 = *human player's cooperation* and 1 = *algorithm's cooperation*; boldface: lowest values of AIC and BIC, lower values indicate better model fit.

Table 7.4.: Fixed Effects for MLM-APIM (Give-Some Example)

	β (human)	β (algorithm)	human - algorithm
Mean Logit	0.81***	1.39***	0.58**
Actor Effect	-0.17*	-0.09	-0.08*
Partner Effect	-0.34***	-1.03***	-0.69***
Actor*Partner	0.06	0.17	0.11

Notes: * $p < .05$; ** $p < .01$; *** $p < .001$; DV is cooperation either for the human player or the algorithm

7.6. Assumptions, Practical Issues, and Required Sample Size

The multilevel adaptation assumes a time-homogeneous autoregressive process. Thus, for all possible values of t , the following should be true: $P(s_t | s_{t-1})$. Moreover, the error distribution must be specified correctly. If further (scaled) variables are added as covariates, linearity and homoscedasticity of logits are added. Linearity of logits means that an increase by one unit should increase the predicted logits by the same unit regardless of the actual value of the predictor. That assumption is broken, for example, when quadratic effects are added. Homoscedasticity of logits means that the accuracy of prediction should be independent of the predictors value. That assumption is bro-

ken if, for instance, the variance of predicted logits is higher for bigger values of the predictor.

The generalized multilevel model bears the advantage that one single model obtains all estimates. Furthermore, random effects and their correlations can be modeled, tested and interpreted. Additionally, the model tests whether the effects differ between the depended variables. Moreover, time-dependent and time-independent variables can be added.

However, the generalized multilevel model bears the disadvantage that the data set has to be prepared in an unusual way with one entry representing one observation of only one behavior, the two preceding behaviors of both partners and the dummy coded variable. That procedure results in twice as many entries as there are intervals minus 2 (e.g. $2 * 47$ entries for one couple).

Furthermore, multilevel-models require large sample sizes for estimating the coefficient variances accurately. A simulation study conducted by Maas and Hox (2005) showed that standard errors at level-2 are biased downward if level-2 sample sizes (e.g. couples) are comparably small (less than 50). Unbiased standard errors could be found for a sample size of 100 at level-2. Furthermore, if random effect variances are small, the estimation of model parameters can become erroneous with negative variances, for example.

However, the simulation study of Maas and Hox (2005) investigated multilevel models in a very general way; therefore, a simulation study was conducted for exploring type-I error rates, power issues and potential biases for the presented multilevel adaptation.

The detailed results can be found at Chapter 10.4. There seems to be a very small bias for small samples ($N=10$) with short sequences ($length=10$). The effect size was underestimated for $\beta = 0.2$ and $\beta = 0.4$ and overestimated for $\beta = 1.00$. The Bias ranged from -0.04 for $\beta = 0.2$ to $.11$ for $\beta = 1.00$. However, no noticeable bias was observed for samples with an least $N = 50$ or $length = 50$. Moreover, the bias affected only the effects sizes, whereas Type-I-error rates were nominal for all conditions.

The multilevel APIM performed better regarding power than the aggregated logit-model. For small effects ($\beta = 0.2$; odds ratio of 1.22) and medium effects ($\beta = 0.4$; odds ratio of 1.49) a good power (above .80) can be achieved even with small sample sizes ($N = 10$) if the sequences are very long ($length > 100$). Vice versa short sequences ($length = 10$) achieve the same power if sample sizes are big ($N > 100$). Same goes for medium sized samples ($N = 50$) with sequences of $length = 50$. If effects are really big

($\beta = 0.8$; odds ratio of 2.23) even small sample sizes ($N = 10$) with small sequences ($length = 10$) will achieve a high power ($> .95$).

7.7. Alternative Multilevel Modeling Strategies

A disadvantage of the presented multilevel adaptation is that the model cannot estimate random effects for both members of a dyad separately. It can, however, estimate random effects for the differences between dyad members. Kenny et al. (2006) proposed an alternative approach using the double entry strategy from (Raudenbush et al., 1995) in order to obtain separate random effects for both members. However, common R packages cannot handle this kind of analysis, whereas the presented adaptation in this monograph can be estimated by common R packages for multilevel modeling, such as lme4 (Bates et al., 2014).

8. Modelling Dyadic Sequences Using Markov Models

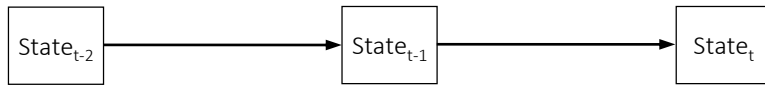
In principle, basic Markov Models (MMs; Figure 8.1a) can be used to address the same research questions as aggregated logit-models, yet there is no need to define a dependent variable. Instead, the focus is on transition probabilities, which are the probabilities for changing from one state into another state if time advances by one unit (see 1.1.5). Another feature is that MMs allow for simultaneous estimation of mean transition rates for all observation units at once. Furthermore, MMs can be extended to hidden (or latent) Markov models (HMMs; Figure 8.1b).

HMMs treat observed states as indicators of latent states, which explain transition rates of observed variables by an underlying, not directly observable process. For example, stress communication (SC) and dyadic coping reaction (DC) may be seen as observable indicators of a couple's coping process, and thereby modeled as a common fate model (see 1.2.4). As this process is going on, a couple might transition through different latent states, such as *being stressed* or *not being stressed*. A state of being stressed could be characterized by a higher probability of observing SC and/or DC than when not in a state of stress. Therefore, observed states are regarded as probabilistic functions of hidden states, resulting in so-called emission probabilities. Because HMMs take latent states into account, research questions such as "How many latent states are there? And how do they affect the observable states?" can be asked.

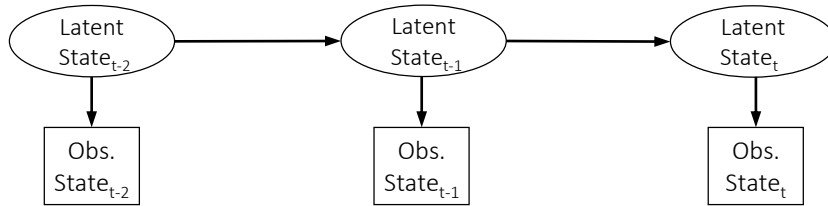
MMs can be extended also to mixture Markov models (MMMs; Figure 8.1d). MMMS allow explaining unobserved heterogeneity by a categorical, time-independent latent variable, often called the latent class. In other words, MMMs assume that groups (latent classes) exist and that observation units are similar within and dissimilar between these groups. Similarity in this context means two things: first, that observation units of one group share similar transition-probabilities, and second that latent states affect observed states in a similar way.

A special feature of mixture models is that class membership is probabilistic in nature. That means each observational unit belongs to a latent class with a certain

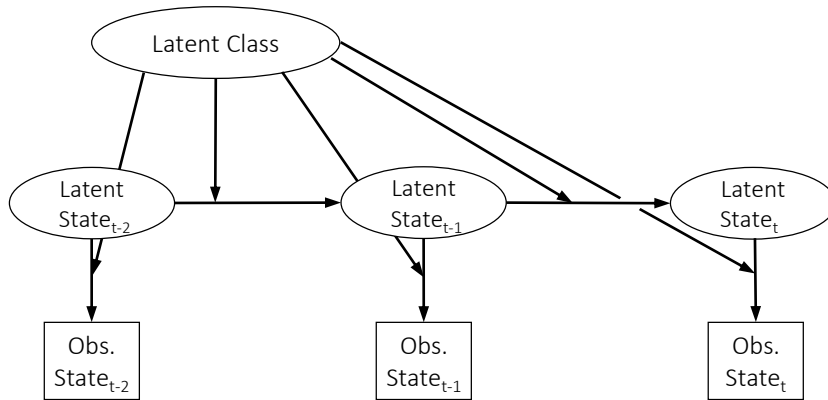
a) Basic Markov Model (MM)



b) Hidden Markov Model (HMM)



c) Mixture Hidden Markov Model (MHMM)



d) Mixture Markov Model (MMM)

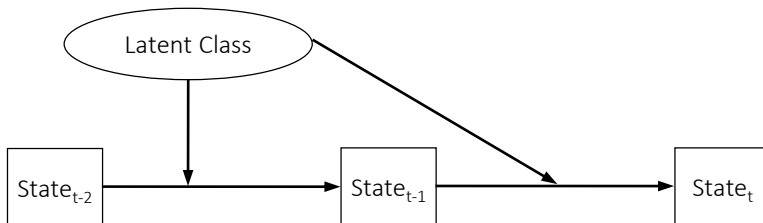


Figure 8.1.: Comparison of MM, HMM, MHMM and MMM denoted as path diagrams.

probability. Possible research questions that can be tackled with MMMs are: How many latent classes exist? Do they differ regarding their number of latent states? Are there similar latent states across the latent classes? If so, do transition rates of the latent states differ between latent classes?

Finally, MMMs can be extended even further to mixture hidden Markov models (MHMMs), which are Markov models assuming latent states and latent classes (MHMMs; Figure 8.1.c). Possible research questions are: How many latent classes exist? DO their latent classes differ regarding the emission probabilities? And what are the differences regarding the transition rates of observed states?

8.1. Basic Markov Models

A modified version of the give-some data (see Chapter 2.2) will be used for illustrating the core idea of basic Markov models (MMs). The give-some data contains 42 sequences of participants' behavior in a social dilemma game in which they had to divide coins between themselves and an artificial opponent for 32 turns. In the previously shown example was coded whether cooperation was shown or not. However, more than two categories are needed for introducing Markov Models properly. Thus, for this application, it is coded whether a person gave *less*, *equal*, or *more* coins than his or her opponent in the last turn. Thus, three states exist. For the sake of simplicity, only the behavior of the human player is used for now, yet the original example (see Chapter 2.2) will be used later for demonstration of dyadic sequences. Converting a basic Markov model into an APIM will be shown using the couples-cope dataset.

A basic Markov model assumes that all sequences follow the form of one Markov chain. That is, the probability for being in a certain state at time t only depends on the immediate previous state at $t - 1$, but not on the states before $t - 1$. This is called the Markov property and can be denoted as $P(s_t | s_{t-1})$. Hence, for the give-some example, the assumption is that the probability of a person being in a state of giving less, equal, or more depends only on the state of the immediate previous turn. For example, a participant was in a state of giving less in time interval one. The probabilities of giving less, equal, or more coins in time interval three depend only on the fact that she or he gave less in time interval two, whereas her or his behavior at the first time interval does not affect the probabilities at all. This assumption is often referred to by saying that MMs describe a Type 1 autoregressive processes (AR1). Moreover, most MMs assume that the process is time-homogeneous, that is, transition probabilities are time-independent. Thus, $P(s_t | s_{t-1})$ should be the same for all values of t . Therefore,

MMs assume exactly the same as the previous modeling strategies for APIMs, namely, the multilevel model and the aggregated logit models.

Time homogeneous transition probabilities can be plotted in the form of a transition diagram; Figure 8.2 displays the best fitting MM for the modified give-some example. Each of the three boxes represents one of the following states, e.g., giving less, equal, or more coins than the opponent. Circled arrows show the probability of staying in the attached state. Hence, in Figure 8.2 the probability that a participant gives less coins than the opponent is .40 if he or she had also given less coins in the previous turn (State 1), .48 for giving equal again (State 2), and .22 for giving more again (State 3). High probabilities indicate that the behavior is stable over time, and thus, it is relatively likely that someone gives the same amount of coins in consecutive turns. Whereas low probabilities indicate that a behavior is not stable, and thus, it is very unlikely that a player gives more coins than the other consecutively.

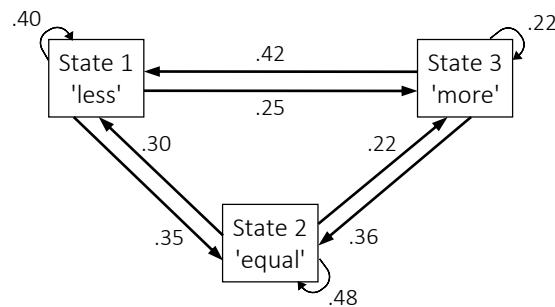


Figure 8.2.: Basic Markov Model for the Modified Give-Some Example.

The straight arrows describe how likely it is to transit from one state, from which the arrows start, into another state, to which the arrows point. For example, the probability that a person changes from a state of giving fewer coins to a state of giving more coins is .25, whereas the probability is .42 that a participant that gave more in the last turn gives fewer coins in the actual turn.

Overall, Figure 8.2 shows that the state of giving an equal number of coins is the most stable behavior, followed by giving fewer coins. Giving more coins is not only the least stable state, but it is also the least likely state to enter from any other state. Thus, participants are repeatedly switching between giving less or equal coins, while giving more coins happens only occasionally.

8.1.1. Transition Matrix

The transition matrix contains the same information as Figure 8.2 and is shown in Table 8.1. Cells display the probability for transitioning from the states displayed in the rows ($t - 1$) to the states shown in the columns (t). Hence, the main diagonal's probabilities correspond to the circled arrows (stability) from Figure 8.2, whereas the other cells are equivalent to the straight arrows (probabilities for changing a state).

For example, the first cell (row: Less; column: Less) shows that the probability of a state of giving less is followed by a state of also giving less is .40 ($P(\text{State}_t = \text{Less} | \text{State}_{t-1} = \text{Less}) = .40$), whereas the second cell in the same row shows that the transition from a state of giving less to a state of giving equal has a probability of .35 ($P(\text{State}_t = \text{Equal} | \text{State}_{t-1} = \text{Less}) = .35$). The final cell in that row shows that the probability of transitioning from less to more is .25 ($P(\text{State}_t = \text{More} | \text{State}_{t-1} = \text{Less}) = .25$). The same logic applies to the other two rows.

Table 8.1.: Transition Matrix for the modified Give-Some Example

	→ Less	→ Equal	→ More
Less →	.40	.35	.25
Equal →	.30	.48	.22
More →	.42	.36	.22

Notes: $X \rightarrow$ transition from X; $\rightarrow X$ transition to X; cells display transition probabilities; less = giving less; equal = giving equal; more = giving more.

The transition matrix can be used to calculate how the distribution of states will change from one time interval to another. For example, suppose at the beginning ($t = 0$) of the give-some example, all persons were in a state of giving more coins. Therefore, the state distribution would be (0/0/1), that is: 0% are in a state of giving less, 0% are in a state of giving an equal number of coins, and 100% are in a state of giving more. The expectation is that 42% will change over to a state of giving less (first column, third row), 36% will change over to a state of giving equal (second column, third row), and 22% will stay in the state of giving more. That would result in $p(\text{less})_{t=1} = .42$; $p(\text{equal})_{t=1} = .36$; $p(\text{more})_{t=1} = .22$ for the state distribution at $t = 1$. For $t = 2$, of the 22% percent that still give more than they received, only 22% (third row, third column) will stay in that state, which is 4.84% of the total. However, 25% (first row, third column) of that 42%, who gave less at $t = 1$, will transit to a state of giving more (10.5% of total). Same goes for 22% (second row, third column) of that 36% who gave an equal amount of coins (7.92% of total). Thus, the total portion of persons

who give more at $t = 2$ are 23.26% (4.84% + 10.5% + 7.92%). Same applies to the other states. Thus, the expected state distribution for $p(\text{less})_{t=2} = .3684$; $p(\text{equal})_{t=2} = .399$; $p(\text{more})_{t=2} = .2326$.

The calculation can be written in a very compact form when it is denoted in matrix algebra (see Equation 8.1). $S_{t=0}$ is a vector with as many entries as there are states containing the relative frequencies of each state for a given interval t . The transition matrix M is a square matrix with as many rows and columns as there are states. Multiplying $S_{t=0}$ with M once for every time unit (t) that passed since $t = 0$ results in the predicted state distribution for that particular time interval.

$$S_t = S_{t=0} * M^t \tag{8.1}$$

t : time interval

S_t : state distribution at time interval t

M : transition matrix

Dyadic sequences have to be combined via the state expand procedure (see Chapter 4.1) before applying a basic Markov model. The results for the couples-cope example are presented in Table 8.2. Cooperation by the algorithm has the highest stability of all states, yet the probability of staying in this state is only .43. There is also a high probability of transitioning from that state to a state in which both players cooperate. The transition rates are similar from a state in which only the human player cooperates. Switching to a state where both cooperate is the most probable transition. However, this state is also not very stable, the combined probability to fall back into a state where only the human or the algorithm cooperates is .50.

This transition matrix shows why the individual sequence plot of Chapter 4 (Figure 4.8) showed very volatile behavior. Players are frequently transitioning between the three states with cooperation. Because of that, the plot shows a wild mix of black (both cooperate), darkgrey (algorithm cooperates), and lightgrey (human cooperates) blocks. Moreover, white spots seem to be rare, which can be explained by the transition matrix again. Transition rates toward a state of no cooperation are very low.

Table 8.2.: Transition Matrix for the Give-Some Example

Cooperation by	→ None	→ Human	→ Algorithm	→ Both
None →	.02	.05	.22	.71
Human →	.12	.33	.11	.43
Algorithm →	.03	.07	.43	.47
Both →	.15	.25	.25	.36

Notes: Cooperation means giving a larger or equal number of coins than the other did at $t - 1$; $X \rightarrow$ transition from X ; $\rightarrow X$ transition to X .

8.2. APIM as a basic MarkovModel

As shown above, a Markov chain describes a process over discrete time (Briggs and Sculpher, 1998), where the state at time t (e.g., showing DC) depends only on the previous state at $t - 1$. Thus, the assumption is the same as for the aggregated logit and the multilevel adaptation of the APIM. Dyadic sequences can be combined via the state expand procedure (see Chapter 4.1). After that, a basic Markov model (MM) can be fitted. However, the MM will not provide single values for each actor and partner effect directly. Instead transitions probabilities are estimated. Results for the couples-cope data are displayed in Table 8.3 and Figure 8.3. For the latter, arrows brightness is used to indicate how likely transition are: the more likely the transition the darker the arrow.

8.2.1. Fitting a basic Markov model on the couples-cope data

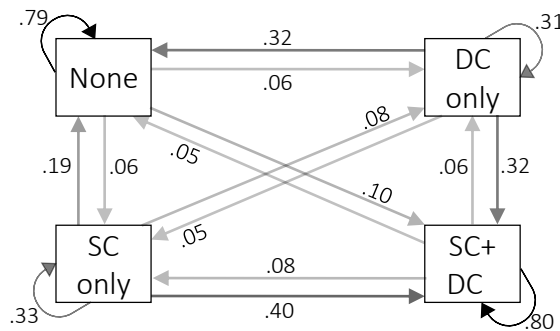


Figure 8.3.: Basic Markov Model for the Couples-Cope Example.

In Table 8.3, the first cell shows that a couple showing no SC or DC at a given interval ($t - 1$) will very likely (with a probability of .79) show no SC or DC at the next interval (t). Keeping in mind that at the beginning of the interaction sequence almost all couples showed SC and DC, we can derive that once a couple attained the status of no stress reaction and no support over the course of time, they most likely finished their coping process. The remainder of the 1st row shows that it is very unlikely that a state without SC or DC will be followed by a state of only SC ($p(SC_t|none_{t-1}) = .06$), or only DC ($p(DC_t|none_{t-1}) = .06$). However, a small but substantial probability of $p(SC_t + DC_t|none_{t-1}) = .10$ depicts that a state with SC and DC will occur after a state without any of the two behaviors.

Furthermore, showing only SC or only DC at a time is not a very stable state. Couples that are in a state of showing only SC or DC are very likely to switch back into a state of showing SC and DC at the same time (.40 for $SC \rightarrow SC+DC$ and .32 for $DC \rightarrow SC+DC$). However, those two states have the highest probability for transitioning into a state of showing no SC or DC, whereas switching from SC+DC into no stress related behavior at all has a very low probability of only .05. Therefore, it is more likely to leave a state of SC+DC by transitioning to SC only or DC only ($p(SC_t|SC + DC_{t-1}) + p(DC_t|SC + DC_{t-1}) = .14$ vs. $p(none|SC + DC_{t-1}) = .05$). Yet, these states have relatively high probabilities for transitioning back to SC+DC. Therefore, it is very likely for couples to change forth and back between SC+DC and a state of SC or DC a couple of times before they enter a state without SC or DC. It is also very likely to even return from a state without SC and DC to a state with SC+DC $p(SC + DC_t|none_{t-1}) = .10$ or showing SC or DC only $p(SC_t|none_{t-1}) + p(DC_t|none_{t-1}) = .12$. This indicates that entering a state without stress related communication does not necessarily mean that the stress itself ended. Couples might take a short break from stress communication or communicate for a short period of time in a way that it is not observable. Especially the last reason can be seen as a form of measurement error that can be estimated by hidden Markov models (see Chapter 8.3). However, before tackling this issue, transition probabilities must be transformed into actor and partner effects for so that it can be interpreted as an APIM.

8.2.2. Converting Transitions into Actor and Partner Effects

The interpretation of transition probabilities as actor and partner effects is straightforward, yet the interaction effect is directly incorporated into the transition probabilities.

Table 8.3.: Transition Matrix for the Couples-Cope Example

	→ None	→ SC	→ DC	→ SC+DC
None →	.79	.06	.06	.10
SC →	.19	.33	.08	.40
DC →	.32	.05	.31	.32
SC+DC →	.05	.08	.06	.80

Notes: Cells show transition probabilities between states; None: no stress communication and no dyadic coping; SC: only stress communication, but no dyadic coping; DC: only dyadic coping, but not stress communication; SC+DC: stress communication and dyadic coping; $X \rightarrow$ transition from X; $\rightarrow X$ transition to X.

Thus, there are two conditional actor effects and two conditional partner effects for both dyad members each.

The female actor effects is the probability that SC is shown at t after SC was shown at $t - 1$ regardless of whether DC is also shown at t . One of the two possible actor effects has the condition that DC is not shown at $t - 1$. It can be denoted as $(p((SC_t \cup SC_t + DC_t) | SC_{t-1}))$. Because all probabilities within a row of the transition matrix are mutually exclusive, it is simply the sum of the transition probabilities from the state with SC and no DC (SC→) to any states with SC (→ SC and → SC+DC). For the example data, this effect is $p(SC_t | SC_{t-1}) + p(SC_t + DC_t | SC_{t-1}) = .33 + .40 = .73$ (2nd; 2nd and 4th column). The second conditional female actor effect is the actor effect when DC was shown at $t - 1$, and hence, is calculated as $p(SC_t | SC_{t-1} + DC_{t-1}) + p(SC_t + DC_t | SC_{t-1} + DC_{t-1}) = .08 + .80 = .88$ (4th row; 2nd and 4th column).

Two partner effects exist for the women: one without additional display of SC at $t - 1$ and one with additional display. The former is .37, which is the sum of $p(SC_t | DC_{t-1})$ and $p(SC_t + DC_t | DC_{t-1})$. The second partner effect is always equivalent to the conditional actor effect with additional display of DC: $p(SC_t | SC_{t-1} + DC_{t-1}) + p(SC_t + DC_t | SC_{t-1} + DC_{t-1})$. Thus, one way to interpret these conditional effects is, that the actor effect without additional display of DC is the *pure* actor effect for women. The same is true for the partner effects without SC. Hence, the third probability can be interpreted as the combined effect (probability to show SC when SC and DC are shown in $t - 1$). Deviation from the expected combined effect based on the pure actor and pure partner hints that an interaction effect might exist (inference on that will be shown at the end of this chapter).

Actor and partner effects for men are calculated in the same way. However, men's behavior is showing DC rather than SC, and thus, probabilities for transitioning to

a state of DC or SC+DC are added up. The role of conditions is also switched: the pure actor effect for men is now $p(DC_t|DC_{t-1}) + p(SC_t + DC_t|DC_{t-1})$, the pure partner effect is $p(DC_t|SC_{t-1}) + p(SC_t + DC_t|SC_{t-1})$, and the combined effect is $p(DC_t|SC_{t-1} + DC_{t-1}) + p(SC_t + DC_t|SC_{t-1} + DC_{t-1})$. The results are .63 for the pure actor effect (31+.32; 3rd row; 3rd and 4th column), .48 for the pure partner effect (.08+.40; second row; 3rd and 4th column), and the combined effect is .86 (.06+.80; 4th row; 3rd and 4th column).

Effects can be transformed into actor and partner effects that are more comparable to the β effects of the aggregated logit and the multilevel models. The core idea is that a fourfold tables can be produced for every partner that shows the same predicted probabilities as a logit model would do. Then the probabilities are transformed into logits. Finally, the logits are put into the logit model's formula and the formula is solved for the β s.

Table 8.4 shows the probability to show DC regardless of whether SC is shown at the same time ($p(DC_t \cup DC + SC_t)$), and therefore, shows the previously calculated effects for the men. Thus, within the "Probabilities" subtable, the first cell refers to the combined effect (1st row; 1st column), the second cell to the partner effect (1st row; 2nd column), the third cell to the actor effect (2nd row; 1st column), and the last cell refers to the probability for showing DC if no SC and no DC was shown at $t - 1$ (2nd row; 2nd column). It can be seen as a baseline probability for showing DC and is calculated as $p(DC_t|none_{t-1}) + p(SC_t + DC_t|none_{t-1})$. The predicted probabilities can be transformed into odds [$p/(1 - p)$] and logits [$logit = \ln(p/(1 - p))$]. Table 8.4 shows DC as a dependent variable as shown in the middle (Odds) and the left-side (Logit) of Table 8.4.

Estimates for the β s can be calculated by using the logits: intercept is the mean of all logits ($0.17 = (1.88 + 0.54 - 0.09 - 1.67)/4$); actor effect is the mean of the logits for showing the same behavior at $t - 1$ minus the intercept ($1.05 = (0.54 + 1.88)/2 - 0.17$); partner effect is the mean of the logits for the partner's acting minus the intercept ($0.73 = ((-0.9) + 1.88)/2 - 0.17$); because effects are additive in their logit form, the actor*partner interaction is the logit of the combined effect minus the sum of intercept, actor, and partner effect ($-0.07 = 1.88 - (0.17 + 0.73 + 1.05)$). The results are displayed in Table 8.5, but differ slightly from the previously calculated values (by .01) because results were computed in R without intermediate rounding, and therefore are less prone to rounding errors.

Equation 8.2 shows the generic calculation for the β s. However, the equation assumes a certain structure for the transition matrix: two partners are assumed (A and

B), the first state refers to a state where both partners show no reaction, the second state is that A shows a reaction but not B, third one is that B shows a reaction and A does not, and the fourth is that both show a reaction. Thus, the equation is limited to a maximum of two partners and to cases in which only one behavior is coded per partner.

Table 8.4.: Predicted Logits, Odds and Probabilities for DC by the Markov model

		Logit		Odds		Probabilities	
		DC_{t-1}		DC_{t-1}		DC_{t-1}	
		Yes	No	Yes	No	Yes	No
SC_{t-1}	Yes	1.88	-0.09	6.58	0.92	.87	.48
	No	0.54	-1.67	1.71	0.19	.63	.16

Notes: SC_{t-1} : Stress Communication at $t - 1$; DC_{t-1} : Dyadic Coping at $t - 1$

The estimates are consistent with the previous analysis of aggregated logit and multilevel models. The actor effects are stronger than partner effects. However, actor effects are bigger for women than for men, whereas men show stronger partner effects. Interaction effects are near zero. All the above were also true for the aggregated logit and the multilevel model; moreover, differences between the male and the female effects are nearly identical to the estimates of the multilevel approach.

Table 8.5.: Actor and Partner Effects for Marcov-APIM (Couples-Cope)

	β for SC	β for DC	SC - DC
Mean Logit	0.19**	0.17**	0.02
Actor Effect	1.31***	1.05***	0.26
Partner Effect	0.55***	0.73***	0.21
Actor*Partner Interaction	-0.04	-0.06*	0.02

Notes: * $p < .05$; ** $p < .01$; *** $p < .001$; p -values are based on non-parametric bootstrapping with 10.000 bootstrap samples.

One advantage of basic Markov models as an APIM is that effects can easily be interpreted in terms of probabilities. This is especially useful for partner*actor interactions effects because they are directly displayed as conditional probabilities.

A disadvantage of basic Markov models is that they come without p -values for significance testing. However, it is always possible to bootstrap confidence intervals for

all kind of statistics (Efron and Tibshirani, 1994) and confidence intervals can be used for approximating p -values (Davison and Hinkley, 1997). Non-parametric bootstrapping (ordinary non-parametric; Canty and Ripley, 2012), based on 10,000 samples, was used for approximating the p -values shown in Table 8.5. The inference statistic seems, overall, to be consistent with the previous approaches, but seems to be more progressive (smaller p -values). For example, the interaction effect for DC was never significant when the previous approaches were applied.

Another practical disadvantage is that results must be converted into β s if they should be compared directly with other models, such as multilevel APIM or aggregated logit APIM. Fortunately, the R-package DySeq provides a function that transforms transition matrices into β -coefficients (*TransToAPIM*) and a second function that transforms β -coefficients into a transition matrix (*APIMtoTrans*).

A practical merit of the Markov model is that most software packages for Markov modeling, such as the R-Packages TraMineR (Gabadinho et al., 2009) or seqHMM (Helske and Helske, 2016), can read the sequence data directly without the need for transforming the data into another format. Therefore, the data pre-processing for Markov models is easier than for the other models, which also decreases the risk of spoiling an analysis by improper data preparation.

$$\begin{aligned}
 \text{Intercept}_A &= \left(\ln\left(\frac{M_{1,2} + M_{1,4}}{1 - M_{1,2} + M_{1,4}}\right) + \ln\left(\frac{M_{2,2} + M_{2,4}}{1 - M_{2,2} + M_{2,4}}\right) \right. \\
 &\quad \left. + \ln\left(\frac{M_{3,2} + M_{3,4}}{1 - M_{3,2} + M_{3,4}}\right) + \ln\left(\frac{M_{4,2} + M_{4,4}}{1 - M_{4,2} + M_{4,4}}\right) \right) / 4 \quad (8.2) \\
 \text{Actor}_A &= \ln\left(\frac{M_{2,2} + M_{2,4}}{1 - M_{2,2} + M_{2,4}}\right) + \ln\left(\frac{M_{4,2} + M_{4,4}}{1 - M_{4,2} + M_{4,4}}\right) - \text{Intercept}_A \\
 \text{Partner}_A &= \ln\left(\frac{M_{3,2} + M_{3,4}}{1 - M_{3,2} + M_{3,4}}\right) + \ln\left(\frac{M_{4,2} + M_{4,4}}{1 - M_{4,2} + M_{4,4}}\right) - \text{Intercept}_A \\
 \text{Interaction}_A &= \ln\left(\frac{M_{4,2} + M_{4,4}}{1 - M_{4,2} + M_{4,4}}\right) - (\text{Intercept}_A + \text{Actor}_A + \text{Partner}_A) \\
 \\
 \text{Intercept}_B &= \left(\ln\left(\frac{M_{1,3} + M_{1,4}}{1 - M_{1,3} + M_{1,4}}\right) + \ln\left(\frac{M_{2,3} + M_{2,4}}{1 - M_{2,3} + M_{2,4}}\right) \right. \\
 &\quad \left. + \ln\left(\frac{M_{3,3} + M_{3,4}}{1 - M_{3,3} + M_{3,4}}\right) + \ln\left(\frac{M_{4,3} + M_{4,4}}{1 - M_{4,3} + M_{4,4}}\right) \right) / 4 \\
 \text{Actor}_B &= \ln\left(\frac{M_{3,3} + M_{3,4}}{1 - M_{3,3} + M_{3,4}}\right) + \ln\left(\frac{M_{4,3} + M_{4,4}}{1 - M_{4,3} + M_{4,4}}\right) - \text{Intercept}_B \\
 \text{Partner}_B &= \ln\left(\frac{M_{2,3} + M_{2,4}}{1 - M_{2,3} + M_{2,4}}\right) + \ln\left(\frac{M_{4,3} + M_{4,4}}{1 - M_{4,3} + M_{4,4}}\right) - \text{Intercept}_B \\
 \text{Interaction}_B &= \ln\left(\frac{M_{4,3} + M_{4,4}}{1 - M_{4,3} + M_{4,4}}\right) - (\text{Intercept}_B + \text{Actor}_B + \text{Partner}_B)
 \end{aligned}$$

- $M_{n,m}$: transition matrix
 n : transition from state s to another state
 m : transition from state s to another state
 A : Sequence A
 B : Sequence B
 s_1 : transition from a state where A and B are 0
 s_2 : transition from a state of A is 1 and B is 0
 s_3 : transition from a state of A is 0 and B is 1
 s_4 : transition from a state of A and B are 1

8.3. Hidden Markov Model

HMMs treat observed states as indicators for latent states. The probability that a particular state is observed depends on the latent state. For example, the probability for showing states of stress-related behavior might be bigger when the latent state is *stress*, whereas a couple that is in a latent state of *no stress* might show that kind of behavior also, but less likely and thereby less frequent. These probabilities are called emissions.

The transitions between the latent states are assumed to follow a Markov chain, which explains transition-rates of observed variables by an underlying, not directly observable process. Therefore, two matrices are estimated, one transition matrix (M) and an emission matrix (E) containing all emissions. Additionally, the initial state distribution must be estimated, too, because it is also not observed directly, and the Markov chain needs a starting point.

Equation 8.3 shows the hidden Markov model. The state distribution (S) at t is now predicted by the corresponding latent state distribution Z times the emission matrix (E). The emission matrix has the same number of rows, yet the number of columns is equal the number of observed states. The latent state distribution is supposed to follow a Markov chain and is predicted by the latent initial state distribution times the transition matrix, which now has as many rows and columns as latent states exist.

$$\begin{aligned} Z_t &= Z_{t=0} * M^t \\ S_t &= Z_t * E \end{aligned} \tag{8.3}$$

t : time interval

Z_t : hidden state distribution at time interval t

S_t : observed state distribution at time interval t

M : transition matrix

E : emission matrix

In the following, three HMMs are shown a latent hazard model featuring absorbing states, a common fate like model with implicit interactions, and a *pure* common fate model.

8.3.1. Latent Hazard Model (Restricted Hidden Markov Model)

The Markov chain can be restricted to model theoretical assumptions. For example, one may assume that two latent states exist: one that corresponds to a state on which couples tries to solve the stress (*stress*) and a second one of having successfully coped with it (*solved stress*). A plausible assumption would be that couples who left a state of stress will not enter it again. In terms of Markov modeling, such a state, which cannot be left once entered, is called an absorbing state. That restriction lets the latent process become similar to a hazard for time-to-event analysis (see Chapter 5.1.2). Another assumption of time-to-event analysis is that at $t = 0$, all observational units are alive, which translate into Markov models that they are not in the absorbing state. Thus the initial state probability for the absorbing state is set to zero. Applying those restrictions creates a model that assumes a latent hazard and treats the observed sequences only as indicators for an underlying process. The only difference is that the hazard is constant in this model, whereas it can vary freely in a Cox-regression.

Table 8.6.: Hidden Markov Model for Modelling Latent Hazard

Initial State	State 1	State 2			
Probabilities	1.00	.00			
Transitions	→State 1	→State 2			
State1 →	.97	.03			
State2 →	.00	1.00			
Emissions	None	SC	DC	SC+DC	
State1 →	.07	.09	.07	.77	
State2 →	.65	.09	.12	.14	

Notes: None: no stress communication and no dyadic coping; SC: only stress communication, but no dyadic coping; DC: only dyadic coping, but not stress communication; SC+DC: stress communication and dyadic coping; $X \rightarrow$ transition from X; $\rightarrow X$ transition to X; State2 was modeled as an absorbing state.

The results for the latent states model are presented in Table 8.6. The emissions show that state 1 can be interpreted as being in stress because it goes with high proba-

bilities of showing stress-related behavior: $p(\text{SC+DC} \mid \text{State1}) = .77$; $p(\text{DC} \mid \text{State1}) = .09$; $p(\text{SC} \mid \text{State1}) = .07$. Whereas state 2 can be described as a state of solved or reduced stress because it goes with a relatively high probability of showing no stress-related behavior ($p(\text{none} \mid \text{State2}) = .66$). The initial state probabilities show that all couples are initially in a state of stress and the transitions matrix shows that the probabilities for leaving this state (solving the stress) is .03. Returning to a state of being in stress is not possible due to the model restriction. So over time, more and more couples will cope with the stress and enter state 2.

The latent state assumes that stress is a latent variable that will end at some point. There is a certain probability for each time interval that the state of stress will end, which is the hazard, except that the hazard affects a latent variable. Thus, the hazard for latent stress is .03 in this particular case. However, this model is limited by the fact that the hazard is assumed to be constant over time. This may not be a realistic assumption for all cases, yet it makes the calculation of the survival rate relatively easy: $Survial = 1 * (1 - h)^t$. The median lifetime is between the 22nd ($S_{t=22} = .5117$) and the 23rd interval ($S_{t=23} = .4963$), which gives an answer to the research question about duration (see Chapter 4.2). However, this approach addresses the question "What is the typical duration of being in stress?" rather than "What is the typical duration of stress communication?" It is worth mentioning that according to this model, stress communication, or at least behavior that is perceived as such, can occur even after the state of stress has ended.

Expected frequencies for observed variables can be derived by first calculating the expected frequency for each hidden state given a certain value for t , and then multiply it with the emissions, and finally summing the frequencies across the latent states. For example, the model would estimate that at the end of the observation period $t = 48$ 23% of the sample are still in a state of stress ($Survial = 1 * (1 - .03)^{48}$). Those 23% of the have a 7% probability for showing no stress related behavior (emission: state 1 \rightarrow None). 77% left that state of stress and have a probability of 65% for showing no stress related behavior (emission: state 2 \rightarrow None). Thus, 51.66% of couples are expected to show no SC, DC, or SC+DC at $t = 48$ ($.23 * .07 + .77 * .65 * 100\%$). In fact, 64.06% of couples showed no stress related behavior at $t = 48$. Therefore, at least regarding this category and time interval, the model seems not to fit very well on the observed data. Another way for assessing the model is by comparing the model with a suited baseline model, such as a basic Markov model. This approach will be shown in the following chapter for the unrestricted hidden Markov model.

The disadvantages of this model are the assumption of a constant hazard, and the fact that it is only applicable to cases in which a latent process is supposed to end at some point in time. However, the merit of the model is that it does not need to define a minimum number of intervals that have to pass after the last shown behavior, which created a problem in the previous time-to-event analysis: dyads that showed stress until time interval 47, but not in the last interval, may have stopped there, but they might also just have taken a brake from communicating stress. Thus, it was unclear whether and when their stress really stopped (see Chapter 5.2).

8.3.2. Unrestricted HMM (Common Fate Like)

First step for estimating an unrestricted HMM is to determine the number of hidden states. This can be derived from theoretical assumption or by model comparisons. The latter can be done by comparing the AIC (Aitkin and Longford, 1986) or the BIC (Schwarz et al., 1978) between models with different latent structures. The basic Markov model without any latent states can be seen as a special case of latent Markov models. If E in Equation 8.3 is an identity matrix, the equation becomes identical with Equation 8.1. Therefore, the basic Markov model can be used as a baseline model.

For the couples-cope, the MM had an AIC of 5048 and a BIC of 5138, whereas the model with two latent states had an AIC of 5694 and a BIC of 5742. The model with three latent states had an AIC of 5035 and a BIC of 5137. Finally, the model with four latent states did not fit better than the model with three latent states (AIC = 5062; BIC = 5207).

The results for the model with three latent states is presented in Table 8.7 and supplemented by Figure 8.4. Emissions can be displayed in a transition plot by replacing each state by a subplot, Figure 8.4 shows a transition plot where the states are replaced by pie charts, showing the state dependent distribution for the hidden Markov model.

State 2 can be identified as a state of combined stress reactions, and it is also the most prominent state in the beginning. Furthermore, it is nearly impossible ($P(\text{State } 2 \rightarrow \text{State } 3) = 0.00018$) to transition directly from state 2 to state 3, which shows very low emissions for stress-related behavior. State 1 seems to be the connection between those two. It is relatively accessible by the other two states and it is relatively likely to transition from State 1 to State 3 ($P(\text{State } 1 \rightarrow \text{State } 3) = 0.16$). Therefore state 1 seems to be the *in-between* state, which is reflected by its emissions: it is the only state in which showing only SC or DC is relatively likely.

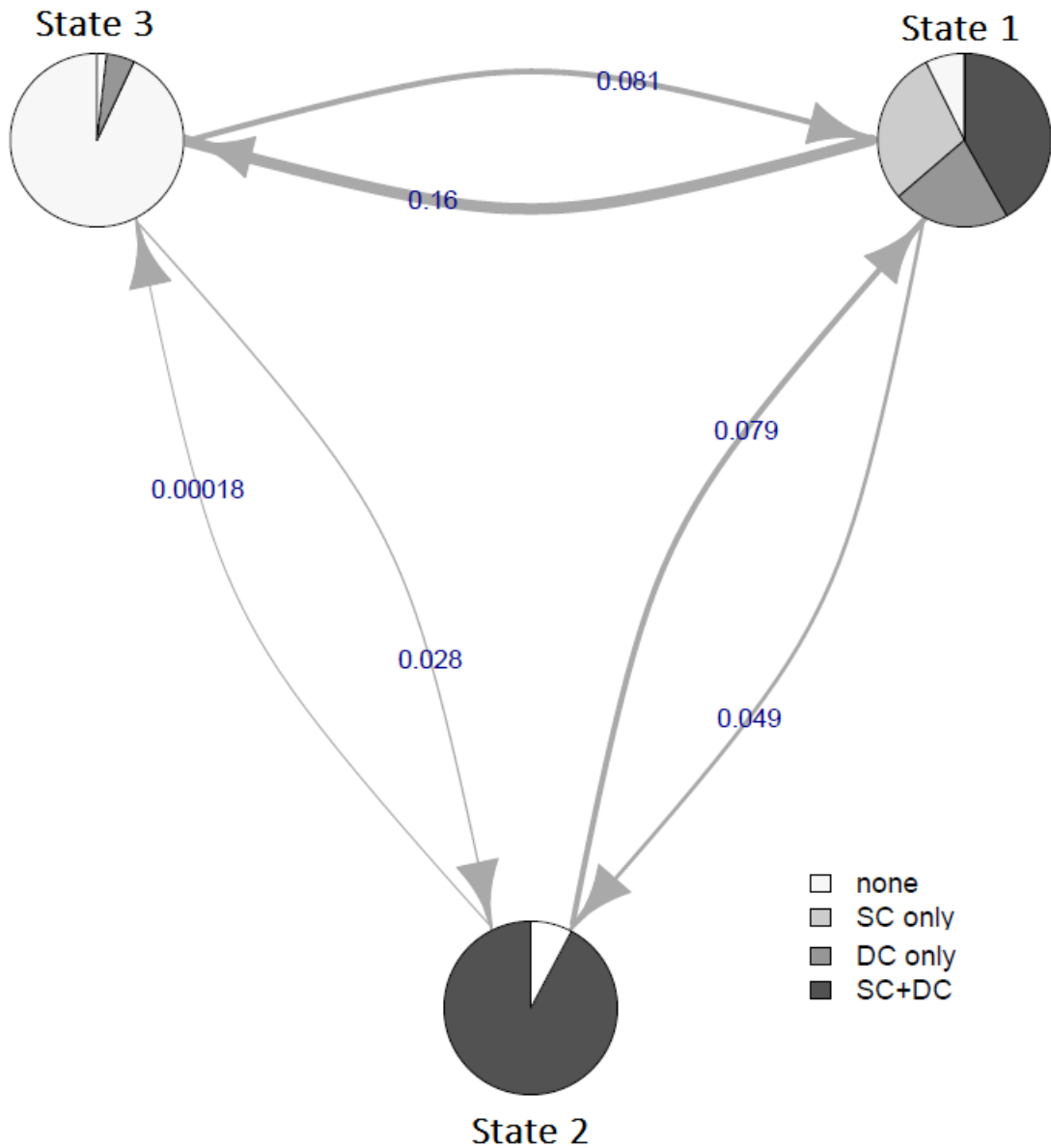


Figure 8.4.: Transition plot for the three states hidden Markov model created with the seqHMM (Helske and Helske, 2016) package. Pie charts show the emissions as the distribution of observed states for a hidden state, whereas arrows indicate transition probabilities. Thicker arrows indicate a stronger transition. Circled arrows (stability) are omitted to avoid crowding.

One interpretation is that state 2 is a state of highly frequent SC and DC interactions. Therefore, both appear often in the same time interval. Over time, the interaction might slow down a bit (state 1 is entered). Therefore, SC and DC are still shown together in .42 of the time intervals, yet intervals with only SC or DC are more often observed. Finally, the stress is solved completely (state 3 is entered). Relapsing one stage (from 3 to 1 or from 1 to 2) of the stress solving process is always possible ($P(\text{State } 3 \rightarrow \text{State } 1) = .081$; $P(\text{State } 1 \rightarrow \text{State } 2) = .079$), however falling back two stages (from 3 to 2) is very unlikely ($P(\text{State } 3 \rightarrow \text{State } 1) = .028$).

Overall, the results are similar to the latent hazard Markov model. Both show that most couples start in a state associated with stress-related behavior, which is left over time. The difference is that the latent hazard was restricted that way by theoretical assumptions, yet the unrestricted model shows a similar structure. Further differences are that an in-between state seems to exist and low probabilities exist for regressing into a previous state.

Table 8.7.: Hidden Markov Model With 3 Latent States

Initial State	State 1	State 2	State 3
Probabilities	0	.90	.10

Transitions	→State 1	→State 2	→ State 3
State 1 →	.79	.05	.16
State 2 →	.08	.92	.00
State 3 →	.08	.03	.89

Emissions	None	SC	DC	SC+DC
State 1 →	.07	.29	.22	.42
State 2 →	.02	.03	.04	.92
State 3 →	.93	.01	.05	.00

*Notes:*None: no stress communication and no dyadic coping; SC: only stress communication, but no dyadic coping; DC: only dyadic coping, but not stress communication; SC+DC: stress communication and dyadic coping; $X \rightarrow$ transition from X; $\rightarrow X$ transition to X; State2 was modeled as an absorbing state.

8.3.3. Multi-Channel Approach (Pure Common Fate)

In the previous model, SC and DC still interact or at least correlate on an implicit level with each other. For example, in state 1, SC had a probability .29 for occurring

without DC, yet the probability was .42 when DC was also present a time t . The multi-channel approach allows treating both sequences as independent indicators. That means probabilities for showing SC or DC depend only on the latent state, but not on the other sequence.

The results for a model with three latent states that treat both sequences as independent indicators is presented in Table 8.8. It seems very similar to the previous model. However, the state's assignment seems to have changed (the assignment is done more or less randomly in the estimation process): State 1 is now the most dominant at the beginning and has the highest probabilities of showing SC (.96) and DC (.95). State 2 is now the *in-between* state, with lower probabilities of showing SC (.62) and DC (.69). State 3 is a state with high probabilities of showing no SC (.95) and no DC (.99). However, the model's AIC is 5102, and the BIC is 5174. Therefore, it does not fit better than a model that allows implicit interactions. According to the AIC, it does not even fit better than a model without latent process at all. Hence, the three latent states model with implicit interactions is the most preferable.

One merit of the multi-channel approach is that it is more parsimonious than a normal hidden Markov model. Each row of the latter's emission matrix must sum up to 1. Therefore, three time three probabilities could vary freely for the last example's emissions matrix. Whereas for the multi-channel model the emissions for SC and DC must each sum up to 1. Hence, only two probabilities can vary freely per row of the emission matrix, making it two times three parameters.

Another advantage arises when missings are present on paired sequences. Combining sequences into one will result in missings whenever there is one missing in either sequence. For example, the sequences "0-0-0-NA" and "NA-0-NA-NA" will result in the combined sequence "NA-0-NA-NA". Thus, information for the first two states in the first sequence exist, yet they are not used. Therefore, handling the sequences as independent allows one to use the information of both sequences more efficiently because sequences do not have to be combined into one sequence anymore.

8.3.4. Using the same Number of Latent States and Indicators

If the number of latent states and observed states are equal, and if all latent states can be matched with one observed state, the emissions are often interpreted as a measurement error (Bartolucci et al., 2012). The idea is that observation (manifest states) might simply misinterpret the true (latent) states, which causes erroneous data. However, applying the hidden Markov models with four states to both datasets did not

Table 8.8.: Two-Channel HMM With 3 Latent States

Initial State Probabilities	State 1	State 2	State 3	
	1.00	.00	.00	
Transitions	→State 1	→State 2	→ State 3	
State 1 →	.91	.08	.01	
State 2 →	.04	.82	.13	
State 3 →	.03	.08	.90	
Emissions	SC=1	SC=0	DC=1	DC=0
State 1 →	.96	.04	.95	.05
State 2 →	.62	.38	.69	.31
State 3 →	.05	.95	.01	.99

*Notes:*None: no stress communication and no dyadic coping; SC = 1: stress communication, SC = 0: no stress communication; DC = 1: dyadic coping, DC = 0: no dyadic coping; X→ transition from X; →X transition to X; State2 was modeled as an absorbing state.

reveal such a relationship. Yet this type of application might be useful for sequences in which each recorded state serves clearly as surrogate for one not directly observable state.

8.4. Latent Groups: Mixture Markov Models

As described in the introduction of this chapter, mixture Markov models (MMMs) can be used to identify latent classes, and thereby answer the research question about unobserved heterogeneity (see Chapter 4.5). The question "can dyads be grouped according to their typical pattern of displaying SC and DC behaviors?" can be translated into MMMs by asking "can dyads be grouped according to their transition matrix?"

Equation 8.4 displays the model. Both the initial state distribution and the transition matrix depend on the latent class; thereby each latent class is defined by its Markov chain. Thus, the observed state distribution for the whole sample is the weighted sum of all class dependent state distributions. A feature of this model is that membership to an latent class is probabilistic. Therefore, each sequence has one probability for each class that expresses how likely it is that the sequence belongs to that particular class. These so-called posterior probabilities can be used for assessing uncertainty. For

example, a sequence might belong with a probability of .51 to latent class one and is therefore assigned to that class. However, it is uncertain whether this assignment is correct. Whereas, a posterior probability of .98 suggest that the amount of uncertainty is very small.

$$S_t|c_j = (S_{t=0}|c_j) * (M|c_j)^t \quad (8.4)$$

$$S_t = \sum_{j=1}^J [(S_t|c_j) * P(c_j)]$$

c_j : the latent class j

J : number of latent classes

$P(c_j)$: probability for belonging to the latent class j

t : time interval

S_t : observed state distribution at time interval t

M : transition matrix

t : time interval

As before, model comparisons can be used to decide on the number of latent classes. AIC Aitkin and Longford (1986) and BIC (Schwarz et al., 1978) were used for model comparison. The basic Markov model (MM; AIC=5048; BIC=5138) can be used again as the baseline model because it is a special case of a mixture Markov model with only one latent class. Fitting a model with more than three latent classes resulted in convergence problems. Therefore, only models with one class (the basic Markov model), two, or three classes were fitted. A model with two latent classes fit better than the MM, according to the AIC (5018), but not to the BIC (5205). Adding a third class improved the AIC further (5006), but the BIC got even worse (5289).

BIC typically weights parsimony higher than the AIC. The latter has a penalty of 2 for every parameter estimated, whereas the BIC uses the natural logarithm of the sample size increases as penalty. Thus, as long as the sample size is bigger than 9, the BIC weights parsimony much higher than the AIC. According to Tofghi and Enders (2008), the BIC is consistent, meaning that it tends to select the correct model more frequently as sample size increases, whereas the AIC does not. Deciding in favor of model simplicity and estimator consistency results in a model without any latent

classes. However, for the sake of demonstration a model with three classes is estimated and presented in Table 8.9. The dependent state distributions are shown in Figure 8.5.

Class 1 is the biggest one, an estimated 54% of dyads belong to that class. Only 72% of this class start in a state of showing SC+DC. The most stable are SC+DC and show no stress-related behavior. This class shows the highest transition rates toward a state of showing no stress-related behavior. A Figure 8.5 shows, couples within this class solve stress relatively fast.

An estimated 26% of the dyads belong to the second class, which is quite the opposite of the first one. Nearly all dyads start in a state of DC+SC. SC+DC is less stable than in the previous class, meaning couples leave this state more quickly. However, probabilities for returning to this state are much higher than in all other classes. Also, the state of showing no stress-related behavior is not very stable (.46) and on top of that, the transition rates for entering this state are also the lowest of all classes. However, stabilities for states of showing either SC or DC are higher than in the other groups. That might indicate that those couples alternate between stress communication and dyadic coping. Figure 8.5 shows that most couples in this class do not leave the stress-related states within the observations duration.

The rest of the dyads, 21%, belong to the third class, which has a low stability for DC and low transition rate toward DC, yet relatively high rates for entering and sustaining a state of showing only SC. Therefore, DC is rarely shown alone. However, transition rates toward a state without stress-related behavior are the second highest, and the stability for this state is also somewhere in the middle. Figure 8.5 shows that this group seems to solve the stress faster than the second group but not as fast as the first one.

Table 8.9.: Mixture Markov Model with 3 Latent Classes (MMM)

Class	Class 1	Class 2	Class 3	
Probabilities	.54	.26	.21	
Initial State Probabilities	None	SC	DC	SC+DC
Class 1	.12	.00	.17	.72
Class 2	.00	.00	.00	.94
Class 3	.15	.00	.47	.38
Transitions: Class 1	→ None	→ SC	→ DC	→ SC+DC
None →	.86	.03	.04	.06
SC →	.28	.22	.10	.40
DC →	.43	.06	.33	.18
SC+DC →	.04	.07	.05	.84
Transitions: Class 2	→ None	→ SC	→ DC	→ SC+DC
None →	.46	.03	.20	.30
SC →	.04	.41	.03	.52
DC →	.22	.02	.40	.36
SC+DC →	.05	.08	.08	.79
Transitions: Class 3	→ None	→ SC	→ DC	→ SC+DC
None →	.64	.17	.06	.13
SC →	.21	.39	.10	.31
DC →	.25	.09	.11	.55
SC+DC →	.08	.10	.08	.74

Notes: None: no stress communication and no dyadic coping; SC=1: stress communication, SC=0: no stress communication; DC=1: dyadic coping, DC=0: no dyadic coping; X→ transition from X; →X transition to X

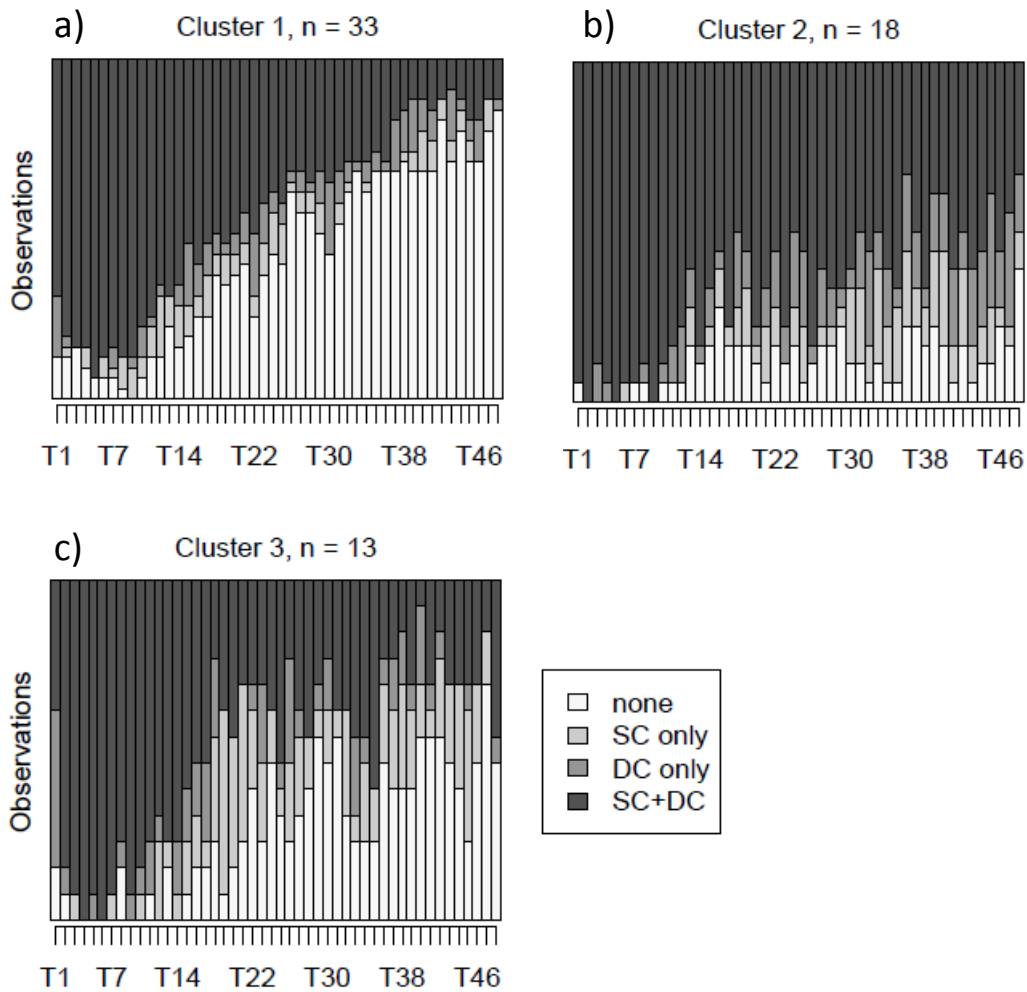


Figure 8.5.: State Distribution for Three Latent Classes (here labeled Clusters); SC: only stress communication is shown; DC: only dyadic coping is shown; SC+DC both are shown; none: none are shown.

8.5. Mixture Hidden Markov Models

As described in the introduction of this chapter, mixture Markov model (MMMs) and hidden Markov models (HMMs) can be combined into mixture hidden Markov models (see Equation 8.5). These models assume that the observed state distribution (S_t) depends on latent states (z_t) that follow a Markov chain ($Z_{t=0} * M^t$). Latent states

and observed states are linked again with an emission matrix (E). However, initial distribution, transition rates, and emission matrix depend on the latent class (c).

Therefore, MHMMs have a lot of free parameters to estimate. For each initial state distribution ($S_{t=0}$), the number of probabilities that can vary freely equals $\#Z - 1$, where $\#Z$ is the number of latent states. For each transition matrix $\#Z * (\#Z + 1)/2$ parameters can vary freely. Finally, for each emission matrix the number is $Z * (\#S - 1)$, where $\#S$ is the number of observed states. The latter might be reduced if the multi-channel approach is used (see Chapter 8.3.3). All this applies to the hidden Markov model as well; however, the mixture model has as many initial distributions, transitions, and emission matrices as there are classes. Therefore, mixture Markov models can become very difficult to estimate, especially with small sample sizes.

Combining the previous two models into one did not work with the couples-cope data at all, yet a much smaller model with only two latent states and two latent classes was possible. Fit indices were bad (AIC = 5224, BIC = 5339). Therefore, the model is only presented for the sake of demonstration, and interpretation should be made with care.

Both groups show balanced transition probabilities. That is, leaving or switching the states has the same probability for both latent states. The chance of transitioning from state one to two is slightly bigger in class one. Thus, over time, the proportion of couples with that state should increase. That is not very intuitive because the emissions show that state one is associated with stress-related behavior, whereas the second state is associated with showing no SC or DC. Hence, couples in the first class should show combined states of SC and DC more often in the end. However, such couples could not be identified via visual inspection (see Figure 4.7).

The second class shows transition probabilities that match the expectations better: the chance of transitioning from state two to one is slightly bigger in class one. Moreover, transition probabilities are lower than in class 1 meaning that couples will stay longer in one of the two latent states and transition only slowly. The emissions are also different. State one is still associated with stress-related behavior, yet the second one is different. Showing no stress-related behavior at all is most probable in this state, too, yet observing any other behavior is still more likely. Inspection the individual sequence plot (see Figure 4.7), reveals that such sequences exist: couples show initially SC+DC and at some time interval they start to transition between all for states often.

Overall, the mixture hidden Markov model fits bad. The second class seems to describe some of the data well, but the majority of the data (67%) is supposed to be in class 1, which does not describe the data well. This was already indicated by the initial

model comparisons. Thus, the mixture hidden Markov model should not be used for the couples-cope data.

$$S_t|c_j = (Z_{t=0}|c_j) * (M|c_j)^t * (E|c_j) \quad (8.5)$$

$$S_t = \sum_{j=1}^J [(S_t|c_j) * P(c_j)]$$

c_j : the latent class j

J : number of latent states

$P(c_j)$: probability for belonging to the latent class j

t : time interval

S_t : observed state distribution at time interval t

Z_t : hidden state distribution at time interval t

M : transition matrix

E : emission matrix

t : time interval

8.6. Which Markov Model is the Best?

When deciding for the best model, three aspects should be considered: the research question, the interpretability, and the model fit.

Overall, the results of the MHMM are difficult to interpret, and it did not fit well with the data. Therefore, it should not be used for the example data. However, it might be well suited for other research topics and datasets. For the couples-cope data, the best fitting model regarding the BIC was the HMM, with three latent states. Moreover, the results indicate that stress coping is a process with at least three consecutive states, each of which can easily be interpreted by the emissions. One shows a state of stress that goes with a high likelihood of showing stress-related behavior; one state is quite the opposite and therefore can be interpreted as *solved stress*; and finally, another state exists that is a *in-between* state in which stress-related behavior is less likely than in a state of stress, but more likely than in a state of *solved stress*. Interestingly, it is quite possible to relapse into a previous stage of stress solving, yet is very unlikely

Table 8.10.: Mixture Hidden Markov Model

Class 1 (P=.67)				
	State 1	State 2		
Initial States	.89	.11		
Transitions	→State 1	→State 2		
State1 →	.939	.071		
State2 →	.074	.926		
Emissions	None	SC	DC	SC+DC
State1 →	.03	.11	.10	.78
State2 →	.87	.03	.10	.01
Class 2 (P=.33)				
	State 1	State 2		
Initial States	1.00	.00		
Transitions	→State 1	→State 2		
State1 →	.956	.044		
State2 →	.040	.960		
Emissions	None	SC	DC	SC+DC
State1 →	.01	.04	.03	.91
State2 →	.39	.28	.15	.17

*Notes:*None: no stress communication and no dyadic coping; SC: only stress communication, but no dyadic coping; DC: only dyadic coping, but not stress communication; SC+DC: stress communication and dyadic coping; X→ transition from X; →X transition to X; State2 was modeled as an absorbing state; transitions are displayed with three digits because two digits would conceal differences.

to fall back two stages at once. The model is interpretable and fits also well with the data. However, even this model will not be a good model if it does not answer the researcher's questions. For example, if one wants to assume a latent process with only two states and wants to know when one of them ends completely, the restricted HMM might be better suited. Moreover, if different types of couples should be identified a mixture Markov model should be used.

Therefore, the research question should be considered as the most important aspect for initial model selection. Interpretability can often be improved by using visualizations such as those presented in Chapter 4.1. Moreover, comparing model results with those visualizations also shows whether the model makes "sense" or not. Comparing models using the model fit is a good way for deciding on the number of latent states or latent classes. However, it is always recommended to test always against a reasonable baseline model. In this monograph a basic Markov model was used because it assumes no latent structure at all, while allowing the state distribution to change over time.

8.7. Practical Issues and Required Sample Size

Hidden Markov models are designed for sequences with many time intervals, yet the number of observational units can be very small. Even single case analysis (sample size of $N = 1$) is possible with hidden Markov models (for implementation, see Visser et al., 2010). Alternatively, Bartolucci et al. (2012) provided implementations for latent Markov models, which can be applied to short sequences (typically three to five time intervals) but need big sample sizes. The sample size of example studies varies between $N = 237$ (Vermunt and Hagnaars, 2004) and $N = 11,400$ (Bartolucci et al., 2012).

Because of that, simulation studies were conducted to give recommendations on the basic Markov APIM, the latent hazard, the hidden Markov, and the mixture Markov model. The simulation studies can be found in Chapters 10.8 to 10.5, yet the most important results are discussed here.

Simulating the optimal sample size and sequence length for the APIM is difficult regarding the Type-I error or statistical power because Markov models usually do not provide p -values at all. However as stated above, Chapter 8.5 provides approximated p -values via bootstrapping. Thus, the same simulations were conducted for the Markov model APIM as for the multilevel model. The transition probabilities were translated into logits so that the results can be compared. P -values were obtained via

non-parametric bootstrapping with 1000 bootstrap samples. In practice, more bootstrap samples should be used (10.000 to 100.000), yet the computation time for the simulation increases exponentially the more bootstrap samples were used. Therefore, the precision of estimates might be slightly higher in a non-simulation application.

For the simulation, 1000 samples were drawn per simulation condition. First of all, no bias for the Type-I error was detected. However, effect sizes were slightly underestimated when samples sizes were small, or sequences were relatively short ($N = 10$; $L = 10$). Also, stronger effect sizes were less affected by the bias than small effect sizes. Regarding the power, the basic Markov APIM performs similarly as the multilevel APIM. For small effects ($\beta = 0.2$; odds ratio of 1.22) and medium effects ($\beta = 0.4$; odds ratio of 1.49), a good power (above .80) can be achieved even with small sample sizes ($N = 10$) if the sequences are very long ($L > 100$). Vice versa short sequences ($L = 10$) achieve the same power if sample sizes are big ($N > 100$). The same goes for medium sized samples ($N = 50$) with sequences of $L = 50$. If effects are really big ($\beta = 0.8$; odds ratio of 2.23) even small sample sizes ($N = 10$) with small sequences ($L = 10$) will achieve a very high power ($> .95$).

For the restricted hidden Markov model, (*latent hazard*) 1000 simulation samples were used per condition. Only the bias of estimates and the simulated standard errors are provided in Chapter 10.6 because rather than testing whether a hazard or an emission is significant from zero it is more important to know how precise the estimation is. The recommendation on sample size and sequence length were based on the rationale that the bias for both hazard and mean absolute bias across the emissions should be less than .01. Based on that, a sample size of $N = 15$ requires a length of at least 40 time intervals; a sample size of $N = 20$ requires 25 time intervals per sequence. If the sample size is at least $N = 30$, a length of $L = 20$ is sufficient, and for $N > 45$ even a length of $L = 15$ gives good results. Standard error was also below .01 for these conditions.

However, these recommendations are only accurate for high emissions, meaning that good indicators must be chosen. If only weak indicators are used, the sample size or the sequence length should be increased. Also, the lowest hazard that was simulated was .01. A lower value for the true hazard might need longer sequences because it takes longer for observational units for transitioning into the absorbing state. However, if a higher hazard is expected, shorter sequences might be sufficient. Because of the above, these recommendations should only be treated as an orientation.

For the hidden Markov 500 simulation samples were used per condition. For determining the number of latent states in an unrestricted hidden Markov model, the AIC

can be recommended as a fit index if sample sizes small ($N = 10$) and sequences are relatively long ($L = 50$). The BIC, on the other hand, performs well if sample sizes are high and sequences are long enough: The correct classification rate got near 100% if the sample size was at least 30 and sequence length was 50, or vice versa, if the sample size was 50 and sequence length was 30. As a minimum recommendation should be $N \geq 30$ and $L \geq 30$, which gives 91% correct classification rate if three or less latent states are expected. This finding is limited by the fact that very distinct latent states were simulated. That is, each latent state had one or two exclusively corresponding indicators with relatively high emissions (the sum of the corresponding emissions were .90). These indicators had low emissions for the other latent states (below .10). Therefore, latent states were easy to distinguish, which was necessary for estimating the bias of transition probabilities. However, it also limits the recommended sample sizes to cases in which the good and unambiguous indicators are used.

The simulation results are not very clear for the estimates of the unrestricted hidden Markov model. If only two latent states exist, *bias* and *SE* for transition rates and emissions are below .01 for $N > 10$ and $L > 10$. However, for the model with three latent states showed a heavy bias, even with $N = 100$; $L = 100$. However, *bias* and *SE* converged at $N > 50$; $L > 30$, and hence, precision will not improve with bigger sample sizes. The bias was towards less extreme probabilities. Therefore, it was still possible to interpret latent states by their emissions. Hence, if the exact probabilities are not the focus of the research, the bias is not problematic. One finding was that different optimizers had a huge impact on the model performance. Therefore, it might be possible avoiding this bias with better optimizers in the future.

For the mixture Markov mode, it was simulated whether AIC or BIC finds the correct number of latent classes and if it depends on the sample size and sequence length. Additionally, the inter-rater agreement between true class membership and most probable membership was used as a measure of goodness of classification. 500 simulation samples were used per condition. The precision of transition rates was evaluated via mean absolute *bias* across all transition matrices and mean *SE*. If only one class or two classes exist, correct model selection works very well with small samples and short sequences. AIC had above 90% correct model selections for $N > 10$ with $L > 10$, BIC needed at least $N > 30$ with $L > 10$ or $N = 10$ with $L > 30$. However, with bigger sample sizes and longer sequences, BIC performed better and showed more consistent results across the simulated conditions. If three latent classes exist, worked well for between $N \geq 30$ and $N \leq 100$ with sequence length between $L \geq 30$ and $L \leq 30$ and could achieve correct classification rates close to or above 90%. BIC could reach

80% correct selection rate for $N = 50$ with $L = 50$ and $N = 100$ with $L = 30$ or $L = 50$. Therefore, AIC seemed to work overall better.

The correct classification of sequences worked well with $N = 10$ and $L = 10$ when only two latent classes existed (Cohen's $\kappa = .98$), if three latent classes exist $N \geq 30$ with $L \geq 30$ is recommended (Cohen's κ showed $>.90$ in most of those conditions) or $N \geq 10$ with $L \geq 50$ is recommended (Cohen's κ showed $>.90$ in most of those conditions).

Bias was good for $N \geq 30$ with $L = 10$ or $L \geq 30$ with $N = 10$ if only one or two classes existed. A Markov model with three latent classes did not perform well regarding the bias of transition probabilities. It was also revealed during this simulation that the Markov model has problems handling absorbing states and transition probabilities that are close to 1 or 0. Transition rates are biased toward less extreme values. The bias is stronger the closer the true transition probabilities are to 1 or 0. The mean absolute bias across all transition probabilities was between .33 ($N = 10$ and $L = 10$) and .21 ($N = 100$ and $L = 100$). However, excluding bias of the absorbing states from the calculation showed a weaker than before still noticeable (.08 for $N = 10$ with $L = 10$; .03 $N = 100$ with $L = 100$). However, a cautious recommendation can be given, but only by making a very strong assumption. The bias of .03 for $N = 100$ with $L = 100$ might be caused by the model adjusting the remaining transition rates to compensate for the bias of the transition rates from the absorbing state. An indication that this might be true is the fact that all conditions high sample size and long sequences showed the same mean absolute bias of .03. Hence, sample sizes and sequence lengths that resulted in a bias of .03 should be sufficient when no extreme transition probabilities exist. Applying this logic shows that sequences should have a length of at least 50. $N = 10$ with $L = 30$ and $N = 30$ with $L = 10$ showed a value of .04 and might also be sufficient.

Overall, all models performed well as long as the latent structure has only two states or classes. Even small sample size of $N = 10$ performed well as long as sequences were $L \geq 30$, and vice versa, short sequences such as $L = 10$ worked overall well if N was ≥ 30 . For the model with three latent classes or states, model selection seems still to work, but estimates for transition matrices and emissions become quite biased. However, as long as the point estimates are only seen as tendencies for the interpretation, the bias should not result in wrong interpretations. For example, the true indicator with the highest emissions for a latent state has a very probability for showing a high emission in the results of the fitted model. Another finding was that optimizers have a very high impact on how well the models performed. A lot

of R packages use only an EM-algorithm for fitting Markov models on sequences. However, the EM-algorithm alone performed very bad (except for the basic Markov model). Therefore, a researcher should choose packages that make use of advanced optimizers such as the seqHMM package ((Helske and Helske, 2016)).

9. A Cluster Analytical Approach: OM-Distances

Cluster analyses have been around in psychological research for a relatively long time, formerly known under the term profile similarity in the 1950's (see Cronbach and Gleser, 1953), they became prominent in the behavioral sciences in the 1970's (see Cooley and Lohnes, 1971).

Cluster analysis is not a single modeling technique rather than a vast collection of classification algorithms. All these algorithms have in common that they divide data into homogeneous subsets. The most common use in psychological research is forming homogeneous groups of persons regarding a set of variables (e.g., dividing persons into progressive, conservative, or liberal thinking based on several items about their political opinions). However, cluster analysis can be used on many other targets, such as items, ratings, or behavior patterns, among other things. Outside of the psychological context, cluster analysis is often used for finding groups of similar customers or products.

Most clustering algorithms follow a three-step procedure: in a first step, a distance matrix is created that describes the (dis-)similarity of the observational units; in a second step, the algorithm creates one or more suggestion of how the data might be split into homogeneous subgroups; if more than one suggestion is generated, a third step is needed for choosing the best cluster solution.

Steps two and three are independent of the data-type, which is used. However, step one differs by what type of data is used and by what exactly is considered (dis-)similarity in a certain application case. As a very basic-example for interval-scaled data, consider the following case: German cities should be clustered by their location. To keep it simple, only four cities are considered with the following coordinates: Essen ($51^{\circ}27'N$ $7^{\circ}0'E$), Bochum ($51^{\circ}28'N$ $07^{\circ}12'E$), Berlin ($52^{\circ}31'N$ $13^{\circ}23'E$), and Potsdam ($52^{\circ}24'N$ $13^{\circ}4'E$). Calculating the distance between all four cities results in the following: Berlin-Potsdam (27.74 km), Berlin-Essen (453.95 km), Berlin-Bochum

(439.37 km), Essen-Bochum (14.69 km), Potsdam-Bochum (413.51 km), and Potsdam-Essen (428.13). The corresponding distance matrix is shown in Table 9.1.

Summarizing the results, Essen and Bochum seem to be in close proximity to each other and therefore might be in the same region (cluster). However, both are distant from Berlin and Potsdam, and thus might not be in the same cluster as they are. Inspecting Berlin and Potsdam shows that they are close to each other so they might belong together, and therefore form a second cluster. This intuition works in this example because only two variables (longitude and latitude) were used, so that the problem was two-dimensional. However, if more variables are used, as is often done in psychological research, the problems become high dimensional. For example, clustering persons regarding their scores on 9 items from a questionnaire, results in a 9 dimensional problem. Thus, it cannot be solved by human intuition anymore; instead, an algorithm is used that imitates this intuition for high dimensional data.

If sequence data is used, another problem occurs: Euclidean distance is only meaningful for metric or interval data. Moreover, it is not even clear how to calculate Euclidean distance for sequence data. What is the Euclidean distance between A-B-B-C and B-C-A-A? Distance measure for categorical data exist, yet they only check whether categories are equal or not. This is not useful because, for a sequence with length 48 and 4 states, such as in the couples-cope data, the number of possible levels is 5,308,416. Therefore, it is very unlikely that any sequences will match. Moreover, treating each time interval as a separate variable might solve that problem, but still result in a distance measure that is not meaningful. Therefore, a special distance measure must be used.

Table 9.1.: Distances Between German Cities

	Essen	Bochum	Berlin	Potsdam
Essen	0.00	14.69	453.95	428.13
Bochum	14.69	0.00	439.37	413.51
Berlin	453.95	439.37	0.00	27.74
Potsdam	428.13	413.51	27.74	0.00

Notes: Euclidean distance in km (kilometer).

9.1. OM-Distances for Sequence Data

Abbott and Tsay (2000) propagated the use of Optimal Matching Distances (OM-Distances) as a distance measure for sequence data. The metric of (dis-)similarity between sequences is defined by Levenshtein-distances (Levenshtein, 1966): two sequences are similar if they show essentially the same pattern of behavior. They differ in the extent to which some elements of one sequence have to be changed in order to perfectly match the other sequence. This is called the *cost*.

Consider a first case, where the sequence of two couples are compared. Couple 1 perfectly fits the sequence of couple 2, but with a shift of one interval (that is, the couples show exactly the same behavioral pattern, yet couple 1 is one time interval *behind*). For example: the first sequence is "0-1-1-1" and second sequence is "1-1-1". In this case, the two sequences can be made identical by removing the first element in the first sequence and shifting the remainder of the sequence to the left (deletion), or by copying the first element of couple 1 and paste it at the beginning of the second sequence (insertion). Therefore, the minimal cost for transforming both sequences into each other is one operation.

Consider a second case with two totally identical sequences, which only differ at the element in the fourth interval. For example, the first sequence is "0-0-0-1" and the second is "0-0-0-0". The two sequences can be made identical by substituting the fourth interval in the first sequence with 0, or by substituting the fourth interval in the second sequence with 1. Again only one operation is needed. However, substitution is weighted differently than insertion or deletion, and the cost for this transformation would be 1 times a weight (weighting).

The weighting depends on the two elements that should be substituted and represents the dissimilarity between these two. A higher weighting stands for more dissimilarity. For exemplification, the previous example is extended by a third sequence: "0-0-0-3". Once again, it would take only one operation to transform it into one of the other two.

However, the similarity between these three sequences may still differ, because an entry of 3 may be very dissimilar to 1, yet similar to 0. For example, 0 might stand for a behavior in which both partners interact non-verbally in positive way, 1 stands for a behavior in which both partners interact verbally in a negative way, and 3 might refer to a behavior in which interact verbally in a positive way. Thus, 0 and 3 show positive behavior and for that are considered more similar toward each other than 3 and 1.

This could be expressed by differently weighting a substitution of 3 with 1, for example, by 1.5. Thus, the cost for obtaining the first sequence would be 1.5 (1 operation times 1.5). Whereas the weighting for substituting 3 by 0 might be 0.5, resulting in a cost of $1 \cdot 0.5$ for obtaining the second sequence. The boundaries are 0 (both entries are completely exchangeable) and 2 (they are completely different; deleting one of the entries for a cost of 1 and inserting the other entry for a cost of 1 would result in the same cost).

The specific weighting is often derived from theoretical assumptions. However, the weighting can also be derived by applying the “TRATE”-formula of Gabadinho et al. (2009), seen in Equation 9.1. It exploits the fact that the sequences represent longitudinal data, and it assumes that states are more similar the more often they follow each other promptly. This is done by subtracting the transition rates of two entries/states from 2: transition rates are the probability that one state will be observed at time $t+1$ given that the other state has been observed at time t . Every pair of entries/states has two transition probabilities, for example, A can follow B, or B can follow A. The “TRATE”-formula’s boundaries are zero and two. Zero if A is always followed by B and vice versa, and two if A is never followed by B and vice versa. Two is the natural upper bound because circumventing one substitution by deleting one entry and inserting another would also have a combined cost of two.

$$\begin{aligned} \text{Cost for } i \neq j : & 2 - p(i|j) - p(j|i) \\ \text{for } i = j : & 0 \end{aligned} \tag{9.1}$$

$i = \text{state observed at time interval } t$

$j = \text{state observed at } t + 1$

Table 8.3 shows the transitions-matrix from which the substitution-cost-matrix is computed. The same technique as in Chapter 8.2 for combining the sequences of both dyads is used here. Cells display the probability that the state depicted in the row is directly followed by the state in the column. The main diagonal can be interpreted as stability (probability to stay in a certain state). For example, the first cell shows that a state without SC nor DC is most likely to be followed by the same state

($p(\text{none}|\text{none}) = .79$), indicating that once this state is attained, it will remain rather stable. The second cell in that row shows that it is very unlikely that a state without SC nor DC will be followed by a state of only SC ($p(\text{SC}|\text{none}) = .06$), or only DC ($p(\text{DC}|\text{none}) = .06$), yet a small but substantial probability of $p(\text{SC} + \text{DC}|\text{none}) = .10$ exists that SC and DC will be shown simultaneously after a state without both responses. The fourth row shows that the opposite transition, both responses stopping within the same interval, is even more unlikely ($p(\text{none}|\text{SC} + \text{DC}) = .05$).

Applying the TRADE-Formula on the states SC+DC and none results in a weighting of 1.85 (=2-.10-.05) for substituting the states SC+DC and none into each other; that is, replacing none with SC+DC, or replacing SC+DC with none. The complete weighting for the sample analysis is stored in the so-called substitution-cost matrix, which is presented in Table 9.2. Rows represent the entry/state that should be substituted and the columns represent the entry/state to be replaced with. For example, the first row shows that substituting a state without SC nor DC (none) with a state of SC is weighted with 1.75, and substituting it with DC is weighted by 1.62. Yet, substituting it with a state/entry that shows both responses (SC+DC) at the interval is weighted with 1.85. Therefore, according to the TRADE-formula, a state with only DC is most similar to a state with no response at all, SC is more dissimilar, and showing both responses at the same time is the most dissimilar state to no response.

Table 9.2.: Substitution-Cost Matrix for the Couples-Cope Data

	→ None	→ SC	→ DC	→ SC+DC
None →	0.00	1.75	1.62	1.85
SC →	1.75	0.00	1.87	1.52
DC →	1.62	1.87	0.00	1.61
SC+DC →	1.85	1.52	1.61	0.00

Notes: None: no stress communication and no dyadic coping; SC: only stress communication, but no dyadic coping; DC: only dyadic coping, but not stress communication; SC+DC: stress communication and dyadic coping; X→ transition from X; →X transition to X.

For every two observation units (couples), the minimum cost is computed for transforming their sequences into each other. The results are stored in a distances matrix with as many rows and columns as number of observations, and cells representing the minimal cost between the associated observation units. This dissimilarity matrix corresponds to other distance measures, such as the Euclidian distance in the previous German city example, except that it assumes sequence data rather than metric data.

9.2. Algorithm

In a second step, clusters of similar sequences can be identified via clustering algorithms. To date, the two most prominent methods seem to be k-means and agglomerative hierarchical clustering, with k-means being far more leading. This observation is backed up by Google Trends (Google Trends, 2018). An analysis of the last 5 years showed that only those two methods occurred in the 25 related queries and the mean interest for the topic "k-mean clustering" was 27%, whereas for "hierarchical clustering" it was 5% (100% interest was defined in this analysis as the peak interest in the topic "Cluster Analysis" within the last 5 years).

9.2.1. k-Means Algorithm

What the algorithm does is representing the observational units as objects in a M-dimensional space, where M is the number of clustering variables. The researcher specifies a certain number of cluster (K) she or he wants to obtain. After that, in an initial step, the algorithm creates one seed per cluster. Those seeds are created randomly, so they represent nothing else than a random location. In a second step, the euclidean (in some implementations the squared euclidean) distance is calculated between each seed and each object. And each object is allocated to the seed with the smallest distance. All objects that are allocated to one seed built one cluster. In a third step, centroids are calculated for each cluster. Centroids represent a cluster's center and replace the initials seeds. In a fourth step, the (squared) euclidean distance between each object and each centroid is calculated. Each object is then assigned to its nearest centroid (smallest distance). Thus cluster memberships of objects may change. In the fifth step, the centroids are updated as the new mean/center based on the new cluster memberships. After that, step four and five are repeated until objects do not change their belonging cluster anymore. After that, centroids will represent their cluster's center.

The k-means algorithm is illustrated based on the German city example in Figure 9.1. The German cities are clustered by longitude and latitude, thus the space is two-dimensional and can be imagined as a map of Germany. Each city is one point on this map. (A) If K is equal to two, then one would drop two different colored pins, for example, grey and black, on this map randomly. By accident the black pin will fall somewhere between Berlin, Potsdam, and Bochum and the grey pin will fall in the most western part of Germany. (B) Berlin, Potsdam, and Bochum are nearer to the black pin than to the grey pin. Thus they are allocated to the black cluster. Essen

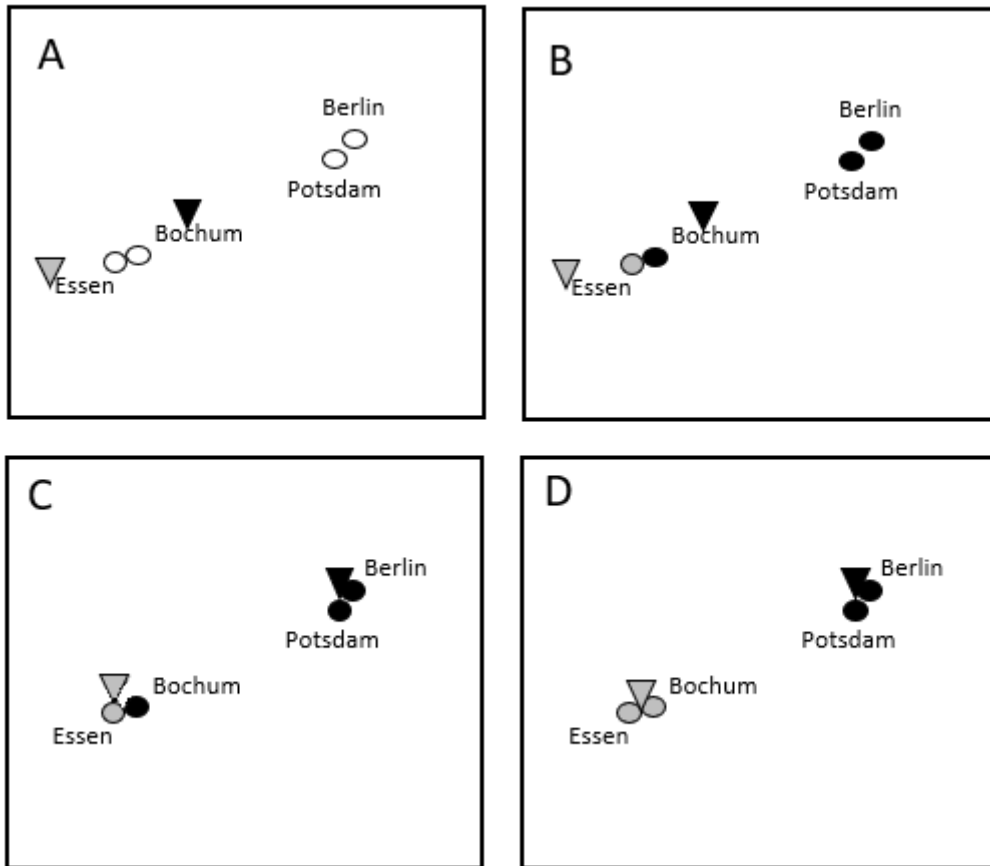


Figure 9.1.: Illustration of k-means algorithm. Dots represent the location of German cities Berlin, Potsdam, Bochum, Essen. Triangles in A represent initial seeds, and in B to D cluster centroids.

is closest to the grey pin. (C) After that, the black pin would be put in the center between Berlin, Potsdam, and Bochum, whereas the grey pin will be located directly in Essen. (D) Now the grey pin is still closest to Essen, but it is also now closer to Bochum, than is the black pin. Thus, Bochum is re-allocated to the grey cluster now. The black pin is still closer to Berlin and Potsdam than the grey pin. Hence, they still belong to the black cluster. After that, the black pin is moved into a new center, which is now between Berlin and Potsdam. The same goes for the grey pin, which is placed between Essen and Bochum. There are no further re-allocations needed, and a final cluster solution is found.

This algorithm works efficiently for high dimensional data, and it is fast and well-proven. However, drawbacks are that different starting points for the centroids (seeds) can lead to different outcomes (Bradley and Fayyad, 1998). Moreover, the algorithm generally provides solutions that are only locally optimal for a given dataset (Steinley, 2003). Furthermore, the observational units must be represented directly as a euclidean vector so that means/centers can actually be calculated. The problem with that is, that sequence data cannot be represented as an directly euclidean vector. One might argue that any distance matrix can be transformed into euclidean distances via multidimensional scaling (MDS). However, applying MDS will result in a loss of information (loss of variance). Applying MDS to the example dataset resulted in a loss of .52% of variance if the distances were projected onto two dimensions. Using a higher number of dimensions such as 10, still resulted in a loss of 29% of the original variance. For these reasons, k-means is not considered as a useful clustering algorithm for sequence data.

9.2.2. Ward's Method (agglomerative hierarchical clustering)

A valid alternative is agglomerative hierarchical clustering. The most prominent version is Ward's method (Ward, 1963). It is commonly used (Willett, 1988), it yields a unique and exact hierarchy of cluster solutions (i.e., for every specified number of clusters only one solution exists), and it is comparable to most methods for identifying the number of clusters. Yet the main advantage for sequence data is that it can be applied directly on a distance matrix, and thus no information about variance is lost.

Ward's method does not re-allocate objects to clusters; instead it only joins two clusters with each iteration, producing a new solution with one less cluster than the previous solution. At every iteration, the sum of errors (S_e) is computed and those two clusters are joined, which increases the SE the least. Depending on the implementation, the SS_e is based on the euclidean (S_e) or the squared euclidean distance to the cluster's centroid (SS_e), yet it can be replaced with any other function. The SE for each solution can be investigated by the researcher for choosing the best solution.

The Ward's method is illustrated based on the German cities example in Figure 9.2. (A) All cities start as their own cluster; thus there is no variance in each cluster and therefore the SS_e becomes zero. (B) Bochum and Essen are closest to each other, so fusing them into one cluster would increase the S_e the least. The S_e becomes 14.69, which is the distance between Essen and Bochum. (C) The next pair that creates the least increase of SS_e is Berlin and Potsdam, which increases the SS_e further by 27.74

to 42.43. (D) Finally, only two clusters are left which can be joined into one, which results in an S_e of 613.36.

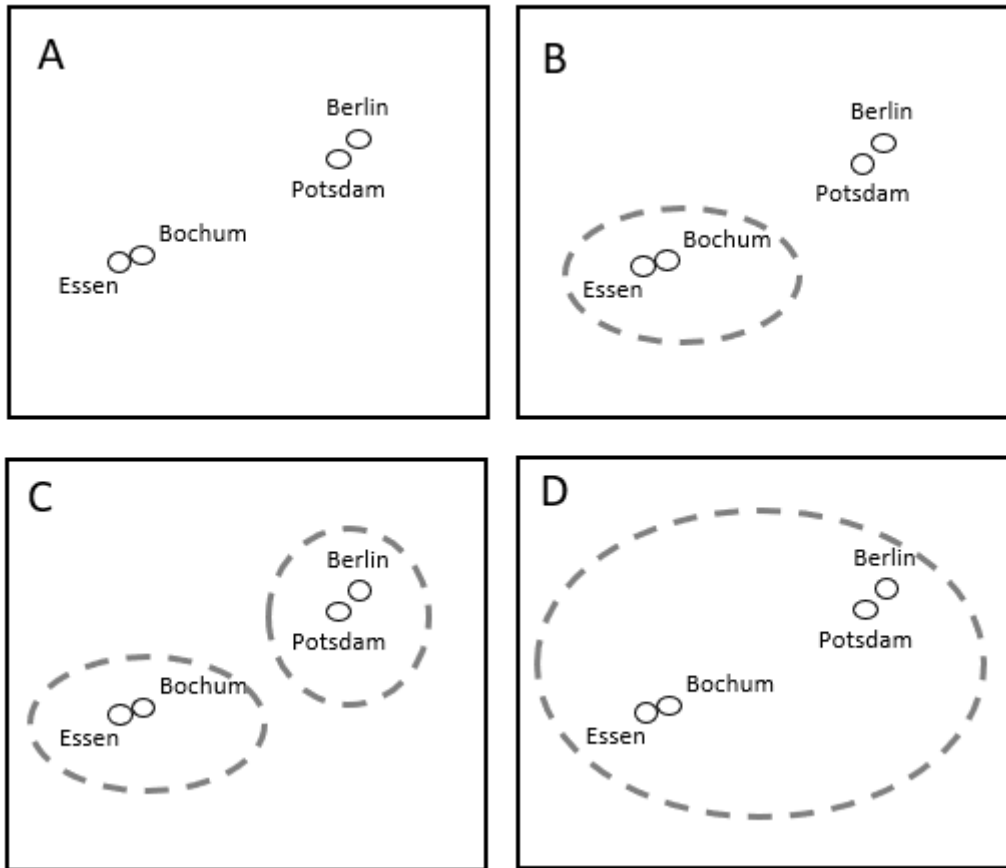


Figure 9.2.: Illustration of Ward's method. Dots represent the location of German cities Berlin, Potsdam, Bochum, and Essen. Dotted circles represent clusters. Cities without cluster are treated as a cluster with only one object.

9.2.3. How to Choose a Cluster Solution

As described in the previous chapter, Ward's method generates multiple cluster solution, from which a Researcher has to choose. The three most common methods will be presented: The dendrogram, the scree plot and the silhouette test.

Figure 9.3 shows the dendrogram for the German cities example. The graphic must be read upwards. The objects are shown at the bottom. Each line represents one cluster with only one object in it. Then objects are merged into one cluster, and the

lines connect. Therefore, all vertical lines show the point at which clusters were joined. The corresponding point on the y-axis shows the corresponding S_e . Big jumps in the S_e , as here from the second to the third join are bad. Therefore, the last solution before that jump should be used. In this case, after the second joined cluster, so the two-cluster solution should be used.

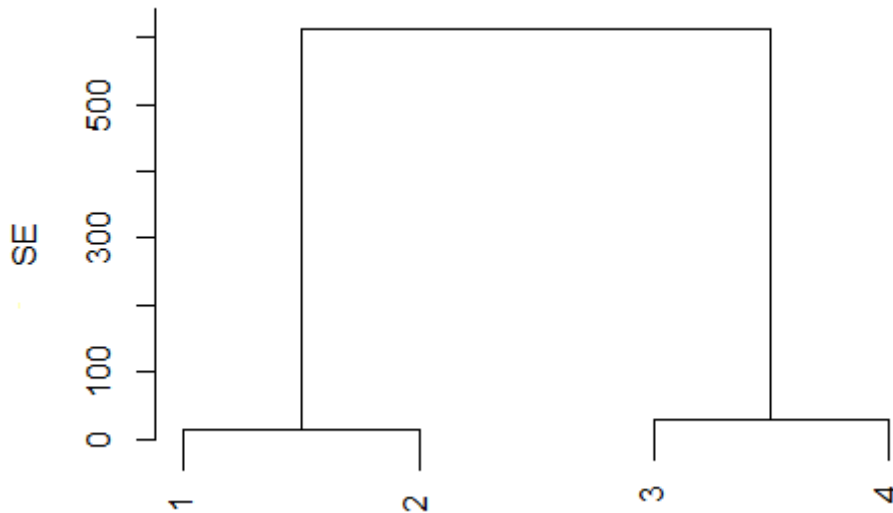


Figure 9.3.: Shows the dendrogram for the German cities example. Y-axis represents the S_e ; Observations correspond as the following: 1 = Bochum; 2 = Essen; 3 = Berlin; 4 = Potsdam

Figure 9.4 shows the scree plot for the German cities example. This graphic shows basically less information than the dendrogram. Instead of observations, only the solutions identified by their number of clusters are plotted on the x-axis. However, even if the plot provides less information, it is easier to see if there are big jumps in the S_e . The big increase in S_e is seen between the one-cluster solution and the two-cluster solution, and the solution before the increase (right-side) should be used. Therefore, the two-cluster solution fits our example best. In fact, dendrogram and scree plot will always indicate the same solution.

The silhouette test by Rousseeuw (1987) is based on the tightness and separation of clusters. If objects within a cluster are tight, the silhouette coefficient will be higher than for loose clusters. The coefficient is bound between minus and plus one. As a rule of thumb, values greater .75 indicate a strong structure, if it is at least greater

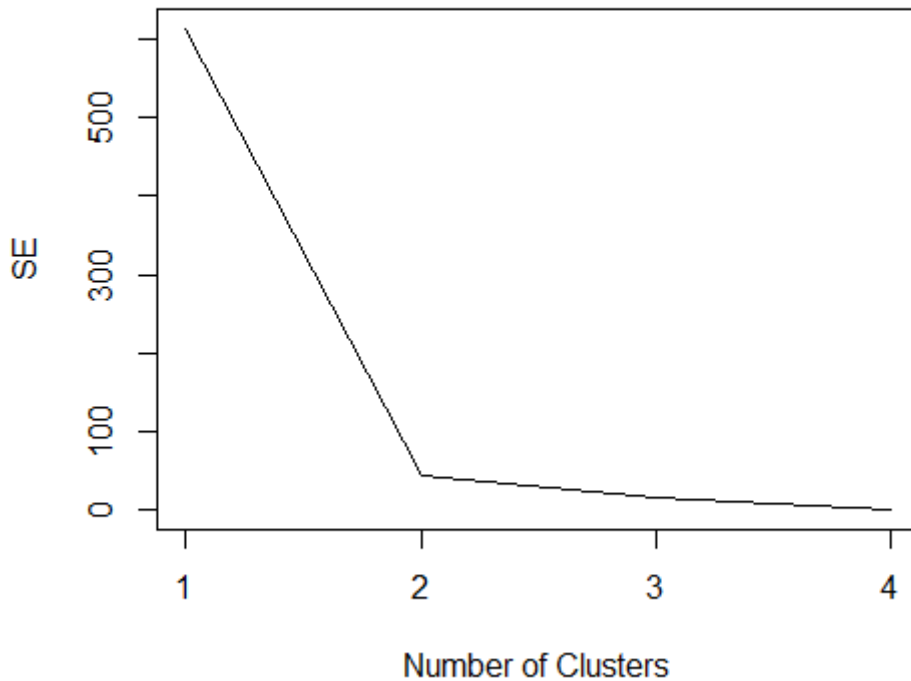


Figure 9.4.: Shows the screeplot for the German Cities example. The y-axis represents the within error sum (S_e). The x-axis shows the cluster solution, identified by its number of clusters.

.50 a medium structure can be assumed, at least greater .25 indicates only a weak structure and below that no structure should be assumed (Struyf et al., 1997; Rousseeuw and Kaufman, 1990).

The silhouette coefficient can be calculated for each cluster separately, yet an overall coefficient for the whole cluster solution is available which is simply the mean of all silhouette coefficients of all clusters. The calculation for each cluster is the following: First the average dissimilarity between all objects within a cluster are calculated. Dissimilarity is operationalized as the euclidean distance, yet any other function can be used instead. This within-dissimilarity is denoted $a(i)$ by Rousseeuw (1987), where i is the cluster for which the coefficient should be calculated. After that, the dissimilarity between the cluster to all objects within another cluster are calculated as the mean

distance. This is denoted $d(i, C)$, where C is the other cluster. There are as many values for $d(i, C)$ as there are other clusters. However, only the $d(i, C)$ with the smallest values, which is in most cases the nearest cluster, will be used and denoted $b(i)$. If $a(i)$ is bigger than $b(i)$, the silhouette coefficient $b(i)/a(i) - 1$; if $b(i)$ is bigger than $a(i)$, the silhouette coefficient is $1 - a(i)/b(i)$; if $a(i)$ equals $b(i)$, the silhouette coefficient is 0. The complete formula is depicted in Equation 9.2.

Overall, the silhouette coefficient for the German city example is .95 for the two-cluster solution, and .48 for the three-cluster solution. It cannot be computed for cases in which all objects are in one cluster or in which every object is its own cluster. The best coefficient was achieved for the two-cluster solution. Hence, all three tests indicate that the two-factor solution fits best.

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \quad (9.2)$$

where $b(i) = \min(d(i, C))$

$s(i)$: silhouette coefficient for target cluster

$a(i)$: within – dissimilarity of target cluster

$d(i, C)$: mean dissimilarity within another cluster C toward the target cluster.

9.3. Applying OM-Distances on the Couples-Cope DataSet

Using the Ward method on the couples-cope dataset yielded less conclusive results for the silhouette coefficient. The best value was achieved for a two-cluster solution. Yet, the average coefficient was only 0.5. Hence, a scree plot was used in addition for evaluating the optimal number of clusters. The result is shown in Figure 9.5 and clearly indicates a two-cluster solution.

The last step is to interpret or to describe the clusters, which can be achieved by comparing the sequences of each cluster. That includes plotting the clusters for comparing the state-distributions; using the cluster membership as a dummy variable, and testing whether it is correlating with other variables, which might explain differences

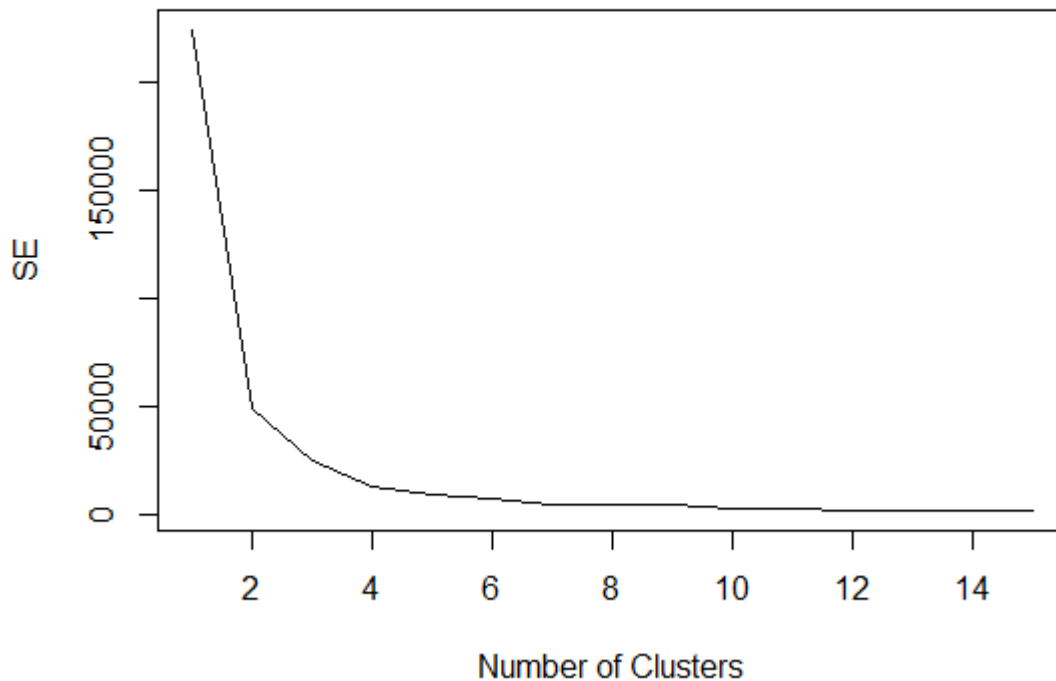


Figure 9.5.: Shows the scree plot for the couples-cope dataset. The y-axis represents the within error sum (S_e). The x-axis shows the cluster solution, identified by its number of clusters.

in the state distributions; applying models for dyadic sequence data separately on each cluster. All of which is shown in the next chapter.

9.4. Comparing Groups or Clusters of Sequences

Comparing clusters of sequences is essential for interpreting results of cluster analysis. What exactly makes them different? Is it possible to give the cluster meaningful names? The same questions may also arise with groups, which were known before the OM-procedure was applied. Therefore, several ways of comparing groups or clusters are shown in the following, using the identified clusters of the previous Chapter 9.3 for comparison.

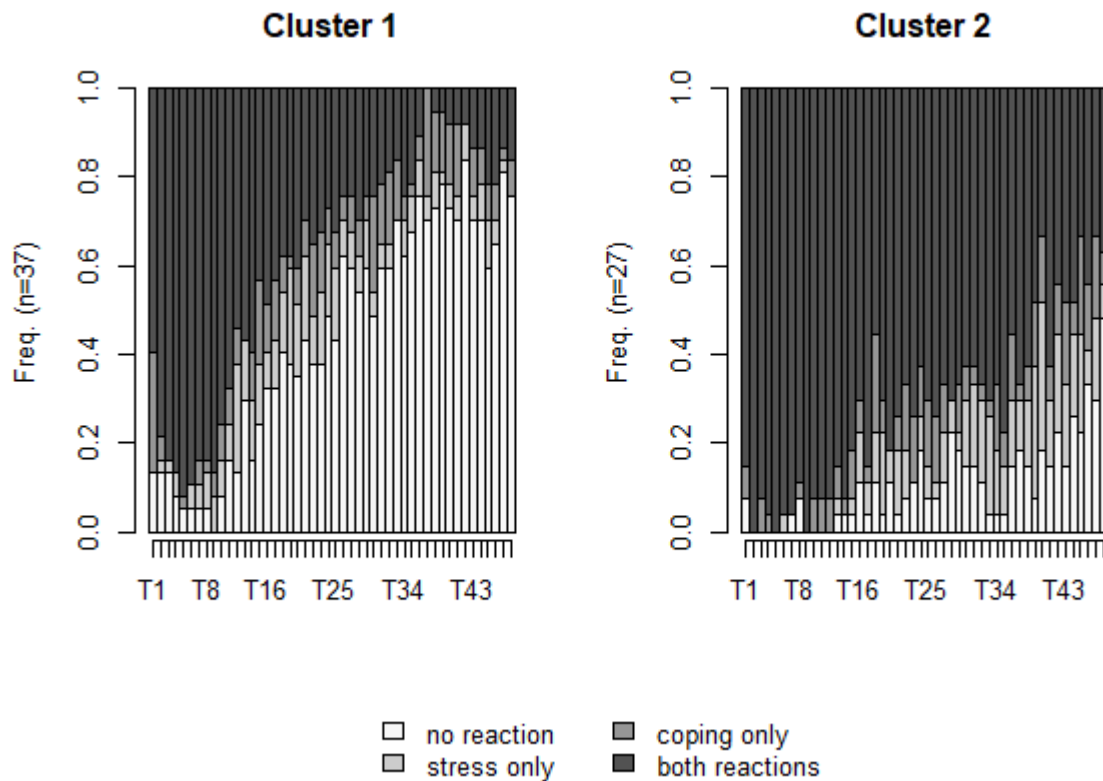


Figure 9.6.: State-distribution plot for the two couples-cope clusters

One way for comparing groups or clusters is to plot them. Figure 9.6 shows the state-distribution plots (see Chapter 4.1) for both clusters. The obvious difference is that couples in Cluster 1 leave the state of stress communication and dyadic coping faster than those in Cluster 2.

Alternatively, the cluster membership can be used as a dummy variable for post-hoc correlation analysis or used as a covariate for one of the previous approaches. For example, a researcher might want to test whether the difference in leaving a state of stress communication is also statistically significant between both clusters.

That finding can be confirmed by using the cluster membership as a dummy coded second covariate for the Cox regression from Chapter 5.2. The analysis reveals a negative effect on the hazard ($\exp(B) = 0.39, p = .004$), meaning that the duration of

stress communication is longer in the second cluster. Therefore, the first cluster will be referred to as the *quick coper* and the second as the *slow coper*.

Using the cluster membership for a post-hoc correlation analysis reveals a correlation between the cluster membership, and men's self-assessed dyadic coping ability reveals a weak negative correlation ($r = -.21$), indicating that the men in the slow coper cluster tend to evaluate their dyadic coping ability lower than men in the fast coper cluster – this association, however, is not significant ($p_{two-tailed} = .095$).

Table 9.3.: Averaged Logit Parameters for Showing DC by Clusters

	Cluster 1	Cluster 2	p^a
grand mean	-0.07	0.67***	<.001
SC at t-1 (actor effect)	0.84***	0.73***	.347
DC at t-1 (partner effect)	0.83***	0.52***	.023
interaction effect	-0.11*	0.24*	.001

Notes: p^a p-value for difference between clusters; p-values for difference from zero are indicated via asterisks * $p < .05$; ** $p < .01$; *** $p < .001$; SC is stress communication shown at $t - 1$ (-1 = no; 1 = yes); DC is dyadic coping shown at $t - 1$ (-1 = no; 1 = yes); dependent variable is showing DC at t .

Another way for comparing the groups is to compare their actor and partner effects. In this example, the aggregated logit model is used to compare the actor and partner effects on stress communication. Table 9.3 provides the results as mean logits for both clusters. Because the individual logits of the couples are t -distributed, a simple t -test can be used for comparing the effects between groups. The results show that the actor effect is significantly lower within the slow coper group, indicating that stress communication might be less continuous in this group (e.g., the female partners may be more often interrupted while communicating stress). The partner effect is also lower in the slow coper group, and although not significant, this finding indicates that for this group, the DC response is less likely (or not as prompt). Another finding is that the interaction effect was not significant in the overall sample, yet considering the two clusters, it shows that for the quick coper, the interaction effect is negative: If stress communication is accompanied by dyadic coping, the probability that the stress reaction will be kept up is less than expected by the main effects. For the slow coper, it is the opposite: If stress communication is accompanied by dyadic coping, it is more likely that it will be kept up. These findings indicate that at least two separate styles of dyadic coping might exist. Identifying the exact nature of these separate styles might be the subject of further research. However, possible explanations could

be that, for instance, the slow coper encourages their partners to communicate their stress through active listening, whereas fast copers tend to appease their partners. Or, another possibility, stressed partners within the slow coping couples like to be comforted and keep their SC up so that their partners will keep up their DC.

Cluster 1 resembles latent class one of the mixture Markov model with three latent classes that was applied in Chapter 10.8). Moreover, the number of sequences that are assigned to this cluster is also close to the estimated size of the first latent class (58% vs 54% for the mixture Markov model.) Therefore, the cluster analysis seems to identify the same latent group. Cluster 2 seems to be a mix of the latent classes two and three. All transition rates for the first two rows are between the transition rates of those two latent classes. Row four shows a higher stability for SC+DC though.

Table 9.4.: Transition Matrices for OM-Clusters

Cluster	Cluster 1	Cluster 2		
Probabilities	.58	.42		
Transitions: Cluster 1	→ None	→ SC	→ DC	→ SC+DC
None →	.83	.05	.05	.07
SC →	.26	.30	.10	.35
DC →	.39	.05	.34	.22
SC+DC →	.08	.09	.06	.76
Transitions: Cluster 2	→ None	→ SC	→ DC	→ SC+DC
None →	.55	.11	.10	.24
SC →	.11	.37	.06	.46
DC →	.22	.05	.27	.47
SC+DC →	.03	.07	.07	.83

*Notes:*None: no stress communication and no dyadic coping; SC=1: stress communication, SC=0: no stress communication; DC=1: dyadic coping, DC=0: no dyadic coping; X→ transition from X; →X transition to X

9.4.1. An ANOVA-Like Method for Sequence-Data

The ANOVA-like method for comparing groups of sequences is especially useful for known groups, if a researcher wants simply to know if two or more sets of sequences

differ significantly. However, for the sake of demonstration, the previously found clusters will be used as groups for the comparison.

Studer et al. (2011) proposed an ANOVA-like method for sequence data. The core idea is to analyze how much dissimilarities can be explained by the groups. In classical ANOVA, sum of squares (SS) are calculated within each group (within-SS) and between groups (between-SS). Within-SS is produced by taking the deviation from each observation from its group mean, squaring it, and summing it up. The between-SS is calculated by taking each group's mean minus the grand mean (mean over all groups), squaring it, weighting it by the group's size, and summing it up.

Therefore, in order to construct an ANOVA-like method for sequence data one needs something equivalent to a mean and to a deviation. The deviation of an univariate observation from its mean is the same as the distance from that mean. Hence, OM-Distances can be used for creating a distance matrix. The problem is that the distance matrix shows only the distance or deviation between observational units. This problem can be solved after choosing a surrogate for the mean.

For a distance matrix created out of metric variables, the centroid is the most equivalent to the mean, however, according to Studer et al. (2011), it is not suitable for complex non-numeric objects such as sequences. Instead the medoid is used, which is the most central sequence. Or, in other words, out of a set of data points in space, the medoid is the data point with the lowest average dissimilarity to all other the other points. The most important part is that the medoid is an actual data point, whereas the centroid can be any point outside of the dataset.

Just for making the difference more clear, consider cities on a map again: calculating the medoid answers the question: "What is the most central city?". Doing this for Germanies top 100 largest cities might result in "Hannover", because this city has the closest average distance to all other cities. Whereas calculating the centroid answers the question "What is the most central location between those cities?". Therefore, the answer might be "longitude:51.9064911, latitude: 9.9212109", which refers to an empty field.

Summarizing the above, the medoid it is very similar to a median rather than a mean. However, the main advantage is that the medoid is actually a data point. That means, it is always possible to identify one observation as the mediod of a dataset or a subgroup.

Because of that, the distance between a groups's medoid and the sequences that belong to the same group can be derived directly from the distance matrix. The same goes for distances between the medoid for the whole dataset and the medoids of sep-

arate groups. Thus the within-SS is simply the sum of the squared distances between all sequences and the medoid-sequence of their group. The between-SS is the sum of the squared distance between the group-medoid sequences and the overall medoid-sequence, weighed by their group size.

According to Studer et al. (2011), using this method, the Equation 9.3 of classic ANOVA holds true. Therefore, the percentage of dissimilarity that is explained by the groups can be calculated as $SS_{between} / (SS_{between} + SS_{within})$. Furthermore, they suggest a pseudo-F statistics that is created the same way as Fisher's F in classic ANOVA: $F_{pseudo} = SS_{between} / (m - 1) / SS_{within} / (W - m)$, where W is the number of groups and m is the number of sequences. It is important to note that this F-statistic does not follow Fisher's F-distribution. Hence, permutation tests are used instead (Chernick et al., 2011).

$$SS_{total} = SS_{between} + SS_{within} \tag{9.3}$$

SS_{total} : The total amount of dissimilarity

$SS_{between}$: The amount of dissimilarity which
can be explained by the groups

SS_{within} : The amount of dissimilarity which
cannot be explained by the groups

Using this ANOVA-like test for sequences on the couples-cope data results in a Pseudo R^2 of .22, indicating that 22% of dissimilarities can be explained by the clusters. The F_{pseudo} was 17.13 with a corresponding $p_{permut} < .001$, indicating a significant difference between both clusters. However, the clusters were found by explorative analysis, and post-hoc significance tests should therefore always be validated by a cross-validation. Nevertheless, this ANOVA-like test for sequences can be used without cross-validation if it is used with known groups, and if the hypotheses are made a-priori.

9.5. Practical Issues and Required Sample Sizes

The clustering with OM-distances has three advantages: 1) it is computationally fast, which might be a crucial factor if sequences should be grouped in real-time. However, within the field of science and research, the slower computation of mixture Markov model should not be a major drawback for the latter. 2) As discussed below, simulation studies indicate that OM-clustering seems to be more robust than mixture Markov models. 3) OM-clustering can be applied to datasets that contain sequences with different lengths. Differences between longer and shorter sequences can be weighted by adjusting the insertion and deletion cost.

For OM-distances, no power analysis or rules of thumb exist, but OM-distances have been applied successfully in studies with even relatively small sample sizes ($N < 50$; Wuerker, 1996a and 1996b; Blair-Loy, 1999). However, studies with many expected clusters often show bigger sample sizes, e.g., $N = 578$ for 9 clusters in Aassve et al. (2007).

Two simulation studies were conducted for recommending sample sizes and sequence length: one for the OM-clustering (see Chapter 10.9) and a second one for (see Chapter 10.10), each with 1000 samples per condition. For the clustering: one, two and three cluster solutions were tested on the same true models that were used for the mixture Markov simulations. Several methods for determining the correct number of clusters were used. The silhouette coefficient performed best. However, the coefficient can only distinguish between cluster solutions with more than one cluster. Therefore, if no cluster solution could achieve a silhouette coefficient of above .40, a one cluster solution was selected (cut-off values of .25 and .50 were tried, but .40 performed best). Visual selection criteria, at least, the scree plot cannot be recommended.

Clustering using OM-distances performed worse than the mixture Markov model for small samples sizes and short sequences. However, it performed better than the mixture Markov model, when big sample sizes and long sequences were simulated. Especially the length of sequences had a huge effect on the bias. Model selection worked well ($> 80\%$ correct model selection) for $N \geq 30$ with $L \geq 100$ or $N \geq 100$ with $L \geq 50$, when the silhouette coefficient was used. A small bias was detected, yet for the recommended sample sizes and sequence lengths it was only .01 and *S.E.* was .03 which seems acceptable.

Using OM-distances for group comparisons achieves a power of above .80 for two-group comparisons and three-group comparisons if the sample size is $N = 10$ and sequence length is $L \geq 20$. If L is 10, N should be 20 for two-group comparisons

and 30 three-group comparisons. Hence, a general rule might be 10 per group. If a power of above .95 is targeted L should be ≥ 30 or N should be ≥ 40 for two or three-group comparisons.

10. Simulation studies

Simulation studies were conducted for the Cox regression, the frailty model, aggregated logit APIM, multilevel APIM, basic Markov APIM, restricted hidden Markov model, mixture Markov model, OM-clustering and the ANOVA-like approach from Chapter 9.4.1. The goal of each simulation was to get recommendations on sample size and sequence lengths. While each simulation might vary regarding sample sizes, sequence lengths, effects sizes, and other aspects, the core principle that was used is the same for all simulations.

Different methods for conducting simulation studies exist, in this case, distribution-based simulations were used. The general principle will be exemplified using the simulations for the Cox regression: The first step was to determine which parameters, which sample sizes, and which lengths should be manipulated. These are referred to as the simulation conditions analogously to experimental conditions.

In the second step, one true model was specified from which sequences can be generated. Parameters that should be manipulated in the simulation were set to be variable between each simulation condition. All other parameters were set to a constant value. For example, for the Cox regression, the initial state for the non-absorbing state was set to 1.00, the baseline hazard was set to a constant hazard of .05 and the length of sequences was set to $L = 30$. The covariate was drawn from a standard normal distribution. The effect of the covariate on the hazard was linked the same way as it is linked in the Cox regression. The true value of β was set to be variable between simulation conditions because effect size should influence the needed sample size.

The third step was to loop through all combination of simulation conditions and to generate a certain number of samples for each combination. These samples are referred to as simulation samples. For example, for the condition $\beta = 0.2$ with $N = 10$, ten sequences were generated by the model described above with $\beta = 0.2$. This was repeated 1000 times resulting in 1000 samples for this condition.

The fourth step, still part of the loop, was to fit a Cox regression on each sample. After that, each sample was dropped (but they are still replicable when the SEED is known; the SEEDs are noted in Appendix B).

The fifth step was to aggregate all results of interest, in this case, the β s of each model that was fitted on the simulation samples: standard deviation of the β s is the simulated standard error and typically decreases with higher sample sizes, the mean of the β s can be compared with the true β . Differences between the mean of the estimated β s and the true β indicate a systematic bias. However, such differences can also occur by chance, yet differences that are caused by chance should become less likely or at least smaller the more simulation samples are drawn.

The distribution-based approach was used for increasing the processing speed of the simulations. Processing power is important because many of the presented models need a long time for estimation. Therefore, higher processing power enables to draw more simulation samples, test more conditions, and finally conduct post-hoc analysis faster. Performance of different strategies depends on technical factors such as what programming language was used.

Alternative approaches, such as population-based simulations that simulate a whole population from which samples are drawn, perform slower when R is used. R can be a very fast programming language as long as certain bottlenecks are avoided. R is slow when it comes to indexing, drawing subsamples from a dataset becomes very slow when the dataset is very big because this process needs a lot of indexing. So drawing subsamples from a population with millions of rows would be one bottleneck. Moreover, R works in memory only. Therefore, if the memory size of the workspace exceeds the RAM, chunks of data must be swapped between RAM and disk drive, slowing down the process even more. Finally, R is prone to RAM fragmentation, which can be avoided by freeing RAM as fast as possible. The latter is the reason for dropping the datasets in step four. Overall, the distribution-based approach was chosen due to technical factor, but also with the intention to generate more simulation samples, and thereby, increasing the precision of the simulations.

10.1. Cox Regression

A simulation study was conducted for determining the optimal sample size for Cox regression dependent on the expected effect size. However, planned simulations revealed also biases in some cases. Follow-up simulations were conducted for investigating the effect of the observation period's length on these biases.

The first simulation assumed a constant *hazard* of .05 and an observation period of 30 time intervals (discrete time). A total of 84 conditions were simulated, each a thousand times. The following sample sizes were iterated: $N = 10$, $N = 20$, $N = 20$,

$N = 30, N = 40, N = 50, N = 60, N = 70, N = 80, N = 90, N = 100, N = 200,$ and $N = 400$. The first ten values were chosen because small sample sizes are common in psychology research. The last two conditions investigate the power develops with relatively big sample sizes. The following effect sizes were simulated for each sample size condition: $\beta=0, \beta=0.2, \beta=0.4, \beta=0.6, \beta=0.8, \beta=1,$ and $\beta=2$. Values for the covariate were drawn from a standard normal distribution. Thus, β 's can be translated as "per standard deviation above the mean, the hazard is increased by $(1 - \exp(b)) * 100\%$ ". Applying that to the selected β 's results in 0%, 22%, 45%, 82%, 123%, 172%, and 739% increment above the baseline hazard per standard deviation.

The results are presented in Table 10.1 to 10.3. The Type-I error (H_0 rejections when $\beta=0$) varies closely around the expected value of .05 indicating that Cox-regression is robust even if time is measured discrete. The power behaves as expected: the power goes up when N increases. For a true β of 0.2 (hazard increase of 22%) the samples need to be very big for achieving a good power (somewhere between $N = 200$ and $N = 400$), whereas for a true β of .4 (hazard increase of 45%), the needed sample sizes become more realistic for psychology research ($N = 80$). However, if the hazard is expected to be doubled per SD increase in the covariate, even small sample sizes (Between $N = 30$ and $N = 40$) will be sufficient.

For the majority of cases, bias is relatively small and conservative. Only very small sample sizes tend to induce a progressive bias. Both biases go up with increasing true values of beta. These findings are partially consistent with the study of Hertz-Picciotto and Rockhill (1997) that showed the same effect for the sample size. The effect of the true beta on the bias of point estimation is new. Because of that, follow-up simulations were conducted to investigate if this might be caused by an interaction between true effect size and baseline hazard, the observation period, or both.

The lower part of Table 10.3 shows the results of the follow-up simulations and reveals that bias depends on hazard and length of the observation period ($\max(t)$). That makes sense actually because the lower the hazard, the more likely it is that units will not show the event within the observational period. Thus, if the baseline hazard is small, only units with extreme values on the covariate will show the event. For example, the probability of not showing the event within ten time intervals for the average person is 0.90 $((1 - 0.01)^{10})$. If the true β is two, the probability for not showing the event, for a person that is one sd above average, is .46 $((1 - 0.01 * \exp(2)^1)^{10})$ and for one SD under the average it is .99 $((1 - 0.01 * \exp(2)^{-1})^{10})$. Thus, units with lower values on the covariate have a larger probability to become censored. The problem can be tackled by increasing the observation period and by this reducing

Table 10.1.: Results for the Simulated Cox-Regressions (Part A)

Simulation conditions	Mean Bias	S.E.	H_0 Rejections
N=10; $\beta=0$	0.02	0.56	.043
N=20; $\beta=0$	-0.00	0.30	.051
N=30; $\beta=0$	-0.00	0.24	.065
N=40; $\beta=0$	-0.00	0.20	.060
N=50; $\beta=0$	-0.00	0.16	.050
N=60; $\beta=0$	+0.00	0.15	.048
N=70; $\beta=0$	+0.00	0.15	.061
N=80; $\beta=0$	+0.00	0.13	.057
N=90; $\beta=0$	+0.00	0.13	.076
N=100; $\beta=0$	-0.01	0.11	.052
N=200; $\beta=0$	-0.00	0.08	.055
N=400; $\beta=0$	-0.00	0.05	.047
N=10; $\beta=0.2$	0.02	0.58	.055
N=20; $\beta=0.2$	+0.00	0.30	.098
N=30; $\beta=0.2$	-0.00	0.25	.149
N=40; $\beta=0.2$	0.01	0.20	.196
N=50; $\beta=0.2$	-0.01	0.17	.202
N=60; $\beta=0.2$	-0.01	0.15	.229
N=70; $\beta=0.2$	-0.00	0.14	.279
N=80; $\beta=0.2$	-0.01	0.13	.314
N=90; $\beta=0.2$	-0.00	0.12	.331
N=100; $\beta=0.2$	-0.01	0.11	.372
N=200; $\beta=0.2$	-0.01	0.08	.645
N=400; $\beta=0.2$	-0.01	0.05	.963
N=10; $\beta=0.4$	0.06	0.57	.107
N=20; $\beta=0.4$	-0.01	0.33	.257
N=30; $\beta=0.4$	-0.01	0.25	.403
N=40; $\beta=0.4$	-0.01	0.21	.526
N=50; $\beta=0.4$	-0.02	0.18	.610
N=60; $\beta=0.4$	-0.01	0.17	.687
N=70; $\beta=0.4$	-0.01	0.15	.768
N=80; $\beta=0.4$	-0.01	0.14	.831
N=90; $\beta=0.4$	-0.02	0.13	.863
N=100; $\beta=0.4$	-0.02	0.12	.901
N=200; $\beta=0.4$	-0.02	0.08	.993
N=400; $\beta=0.4$	-0.02	0.05	1.000

Notes: Mean bias is calculated as the mean deviation from the estimates ($\hat{\beta}$) from true β ; S.E.: is the simulated standard error (standard deviation of estimates); H_0 Rejections represent Type-I error rates in all conditions with $\beta = 0$ and power (1 - Type-II error rate) for the other conditions; for each condition 1000 samples were drawn; baseline hazard $h_0 = .05$.

Table 10.2.: Results for the Simulated Cox-Regressions (Part B)

Simulation conditions	Mean Bias	S.E.	H_0 Rejections
N=10; b=0.6	0.10	.70	.193
N=20; b=0.6	0.01	.33	.499
N=30; b=0.6	-0.02	.26	.699
N=40; b=0.6	-0.02	.22	.805
N=50; b=0.6	-0.02	.19	.906
N=60; b=0.6	-0.02	.18	.952
N=70; b=0.6	-0.02	.16	.973
N=80; b=0.6	-0.02	.15	.989
N=90; b=0.6	-0.03	.14	.991
N=100; b=0.6	-0.03	.13	.996
N=200; b=0.6	-0.03	.10	1.000
N=400; b=0.6	-0.03	.06	1.000
N=10; b=0.8	0.19	3.45	.295
N=20; b=0.8	0.02	.37	.708
N=30; b=0.8	-0.00	.28	.898
N=40; b=0.8	-0.01	.23	.954
N=50; b=0.8	-0.00	.21	.988
N=60; b=0.8	-0.02	.22	.993
N=70; b=0.8	-0.03	.17	1.000
N=80; b=0.8	-0.04	.16	1.000
N=90; b=0.8	-0.03	.15	1.000
N=100; b=0.8	-0.04	.13	1.000
N=200; b=0.8	-0.03	.10	1.000
N=400; b=0.8	-0.04	.06	1.000
N=10; b=1	0.17	.78	.433
N=20; b=1	0.01	.42	.837
N=30; b=1	-0.02	.30	.959
N=40; b=1	-0.01	.25	.995
N=50; b=1	-0.02	.22	.999
N=60; b=1	-0.03	.19	1.000
N=70; b=1	-0.04	.18	.999
N=80; b=1	-0.04	.16	1.000
N=90; b=1	-0.03	.16	1.000
N=100; b=1	-0.04	.15	1.000
N=200; b=1	-0.03	.10	1.000
N=400; b=1	-0.04	.06	1.000

Notes: Mean bias is calculated as the mean deviation from the estimates ($\hat{\beta}$) from true β ; S.E.: is the simulated standard error (standard deviation of estimates); H_0 Rejections represent Type-I error rates in all conditions with $\beta = 0$ and power (1 - Type-II error rate) for the other conditions; for each condition 1000 samples were drawn; baseline hazard $h_0 = .05$; zero values with - indicate that the actual value is below zero, but so small that it is rounded to zero.

the overall probability for censoring. The effect of that can also be seen in Table 10.3; the bias decreases with longer observation periods even under extreme conditions (very small N and extremely high hazard). Overall, a length of 30 intervals seemed to sufficient when N was at least 20, when β was one or lower, and when baseline hazard was at least .05.

The interpretation of this simulation is limited to cases with constant hazards and the presented conditions, yet the script (see Appendix B.1) for the simulation can easily be modified for conducting similar simulations studies. If the baseline hazard should not be constant, the function `simSeq()` has to be replaced with another function that assumes another baseline hazard.

10.2. Shared Frailty

Simulation-based power analysis was conducted for the shared frailty model. A total of 25 (5*5) conditions were simulated. The simulated frailty parameters were 0, 0.25, 0.50, 0.75, and 1. The sample size was also varied starting from 20 to 100 in steps of twenty. For each condition, only 100 samples were drawn because computation for one frailty model took very long.

The results are displayed in Table 10.4. The model performs badly in this simulation. First of all, the Type-I-error is incredible high; in 100% cases, in which the model converged, all models were significant. Same is true for all condition that includes a frailty parameter greater zero. Thus, the p -value is not trustworthy at all.

Additionally, the estimated $\hat{\sigma}$ are way off the expectable deviation from the true score. On top of that, it increases only slightly with the true σ . Strangely, the estimated $\hat{\sigma}$ seems to become greater the bigger the sample size. Sample size and effect size, however, should be independent.

Many things can cause the results. The code for the simulation was doubled checked because the results seemed odd, yet no error was found. The implementation might be erogenous. Future research could test that by running the same simulation using another package or software. However, the package is in its 85th release; therefore, it is very likely that the program works correctly. Finally, some parts of the simulation might violate implicit assumptions of the model. For example, the model or the implementation might not expect ties, which occur because the times are estimated in discrete space. Future research might test these by re-running the simulation (see Appendix B.2) with ties and without ties. Furthermore, estimating the baseline haz-

Table 10.3.: Results for the Simulated Cox-Regressions (Part C)

Simulation conditions	Mean Bias	S.E.	H_0 Rejections
N=10; b=2 ^a	1.19	28.29	.704
N=20; b=2	-0.24	0.57	.988
N=30; b=2	-0.28	0.44	1.000
N=40; b=2	-0.30	0.34	1.000
N=50; b=2	-0.32	0.32	1.000
N=60; b=2	-0.31	0.30	1.000
N=70; b=2	-0.32	0.27	1.000
N=80; b=2	-0.33	0.25	1.000
N=90; b=2	-0.33	0.23	1.000
N=100; b=2	-0.35	0.21	1.000
N=200; b=2	-0.37	0.16	1.000
N=400; b=2	-0.37	0.11	1.000
Follow-Up Simulations ^b	Mean Bias	S.E.	H_0 Rejections
N=10; b=2; $h_0=0.01$; max(t)=10	14.33	58.13	- ^c
N=10; b=2; $h_0=0.01$; max(t)=50	2.73	19.35	- ^c
N=10; b=2; $h_0=0.01$; max(t)=100	1.25	2.39	.407
N=10; b=2; $h_0=0.05$; max(t)=10	3.36	20.90	- ^c
N=10; b=2; $h_0=0.05$; max(t)=50	1.97	4.33	.703
N=10; b=2; $h_0=0.05$; max(t)=100	1.85	1.16	.711
N=10; b=2; $h_0=0.10$; max(t)=10	1.70	3.12	.688
N=10; b=2; $h_0=0.10$; max(t)=50	1.63	0.88	.692
N=10; b=2; $h_0=0.10$; max(t)=100	1.64	0.86	.735

Notes: Mean bias is calculated as the mean deviation from the estimates ($\hat{\beta}$) from true β ; S.E.: is the simulated standard error (standard deviation of estimates); H_0 Rejections represent Type-I error rates in all conditions with $\beta = 0$ and power (1 - Type-II error rate) for the other conditions; for each condition 1000 samples were drawn; ^a not all models did converge in this simulation; ^b the number of simulations was increased to 10.000 per condition because of the extreme high S.E.; ^c p -value could not be calculated for all samples because no events were observed, and thus relative frequency of H_0 rejections would be misleading.

ard non-parametrically might give the optimizer too much freedom for estimating $\hat{\sigma}$. Future research might test these by rerunning the simulation and running parametric models.

Table 10.4.: Simulated Frailty Models

Simulation conditions	$\hat{\sigma}$	non-convergence	H_0 Rejections	corrected ^a
N=20; $\sigma=0.00$	2.72	.02	1.000	.980
N=20; $\sigma=0.25$	2.71	.03	.959	.930
N=20; $\sigma=0.50$	2.76	.00	.970	
N=20; $\sigma=0.75$	2.75	.00	.980	
N=20; $\sigma=1.00$	2.81	.00	.950	
N=40; $\sigma=0.00$	2.87	.00	1.000	
N=40; $\sigma=0.25$	2.87	.00	1.000	
N=40; $\sigma=0.50$	2.86	.00	1.000	
N=40; $\sigma=0.75$	2.86	.00	1.000	
N=40; $\sigma=1.00$	2.90	.00	1.000	
N=60; $\sigma=0.00$	2.89	.00	1.000	
N=60; $\sigma=0.25$	2.91	.00	1.000	
N=60; $\sigma=0.50$	2.97	.00	1.000	
N=60; $\sigma=0.75$	2.96	.00	1.000	
N=60; $\sigma=1.00$	3.01	.00	1.000	
N=80; $\sigma=0.00$	2.91	.00	1.000	
N=80; $\sigma=0.25$	2.95	.00	1.000	
N=80; $\sigma=0.50$	2.98	.00	1.000	
N=80; $\sigma=0.75$	2.93	.00	1.000	
N=80; $\sigma=1.00$	3.02	.00	1.000	
N=100; $\sigma=0.00$	2.96	.00	1.000	
N=100; $\sigma=0.25$	2.98	.00	1.000	
N=100; $\sigma=0.50$	2.96	.00	1.000	
N=100; $\sigma=0.75$	2.97	.00	1.000	
N=100; $\sigma=1.00$	2.98	.00	1.000	

Notes: $\hat{\sigma}$ is the estimated value of σ ; non-convergence is the relative frequency of models that did not converge; H_0 Rejections represent Type-I error rates in all conditions with $\sigma=0$ and power (1 - Type-II error rate) for the other conditions; ^a H_0 rejections were based only on models that converged, the column "corrected" shows the frequency of rejected H_0 based on all samples; for each condition 100 samples were drawn.

10.3. Aggregated Logit APIM

Preliminary simulations showed that the error was always higher for the actor effect. Even in simulations without true effect, the Type-I error rate was often higher than expected for the actor effect. Moreover, the actor effect was negatively biased. Therefore, a systematic simulation was conducted for investigating if the bias depends on the sample size or the length of the sequences, see Table 10.5. Results for each simulated condition are based on 1000 samples. The initial states were equally distributed.

The table shows clearly that the bias for the actor effect depends on the length of sequences. The shorter the sequence, the more negative the bias. At the same time, while the sample size is increased, the Type-I error increases. That can be explained by the increased power, which makes it more likely that the bias is mistaken as a true effect.

Interestingly, the effect can be regulated by delta, the constant which is added to every cell (0.05 per default; see Figure 10.1). Increasing delta suppresses the bias. The most extreme condition ($\beta = 0$; $N = 100$, $L = 10$) was simulated with $\Delta = 1$. Both biases were reduced to -.05 (from originally -.08), and Type-I error was reduced to .220 and .280 (previously .333, and .336). A delta of 5 reduced the bias to near zero and Type-I was reduced further to .090 and .100. However, increasing delta will always lead to lower power for all effects because the higher the constant, the more equal the relative frequencies of the cells become.

Preliminary power analyses showed that the power is always the lowest for the actor effect. One explanation is that the bias decreases the estimated β even though a true effect exists. Figure 10.2 shows that the bias for the actor effect is independent of the sample size but tends to increase when true effect sizes are big, and sequences are short. Therefore, the sequence length was included in the simulation as a condition. Moreover, the true actor effect size starting at $\beta = 0.2$ was increased by 0.2 until a maximum of 1 was reached. For each condition, 1000 samples were drawn. The initial state distributions were randomly drawn from a uniform distribution. The results are presented in Figure 10.3. The graphs show that the effect of sample size on the power depends on the length of the sequences. Longer sequences produce higher power, and thereby increase the chance to reveal true effects.

Table 10.5.: Aggregated Logit: Type-I Error Rates for Actor Effect

Simulation conditions	Actor1		Actor2	
	$\hat{\beta}$	Type-I error	$\hat{\beta}$	Type-I error
$\beta=0$; N=10; L=10	-0.08	.062	-0.08	.061
$\beta=0$; N=10; L=30	-0.06	.080	-0.06	.073
$\beta=0$; N=10; L=50	-0.04	.058	-0.04	.063
$\beta=0$; N=10; L=70	-0.03	.060	-0.03	.059
$\beta=0$; N=10; L=100	-0.02	.060	-0.02	.060
$\beta=0$; N=30; L=10	-0.08	.142	-0.08	.124
$\beta=0$; N=30; L=30	-0.06	.126	-0.06	.141
$\beta=0$; N=30; L=50	-0.04	.114	-0.04	.099
$\beta=0$; N=30; L=70	-0.03	.103	-0.03	.088
$\beta=0$; N=30; L=100	-0.02	.068	-0.02	.082
$\beta=0$; N=50; L=10	-0.08	.192	-0.08	.178
$\beta=0$; N=50; L=30	-0.06	.204	-0.07	.223
$\beta=0$; N=50; L=50	-0.04	.169	-0.04	.160
$\beta=0$; N=50; L=70	-0.03	.139	-0.03	.146
$\beta=0$; N=50; L=100	-0.02	.106	-0.02	.089
$\beta=0$; N=70; L=10	-0.08	.233	-0.08	.242
$\beta=0$; N=70; L=30	-0.07	.302	-0.06	.256
$\beta=0$; N=70; L=50	-0.04	.212	-0.04	.198
$\beta=0$; N=70; L=70	-0.03	.155	-0.30	.151
$\beta=0$; N=70; L=100	-0.02	.146	-0.20	.131
$\beta=0$; N=100; L=10	-0.08	.333	-0.08	.336
$\beta=0$; N=100; L=30	-0.07	.415	-0.06	.369
$\beta=0$; N=100; L=50	-0.04	.279	-0.04	.276
$\beta=0$; N=100; L=70	-0.03	.236	-0.03	.220
$\beta=0$; N=100; L=100	-0.02	.147	-0.02	.165

Notes: $\hat{\beta}$ is estimated mean effect; β is true effect; N is sample size; L is length of sequence; 1000 samples were drawn per simulation condition. H_0 rejections are the power (1 - Type-II error).

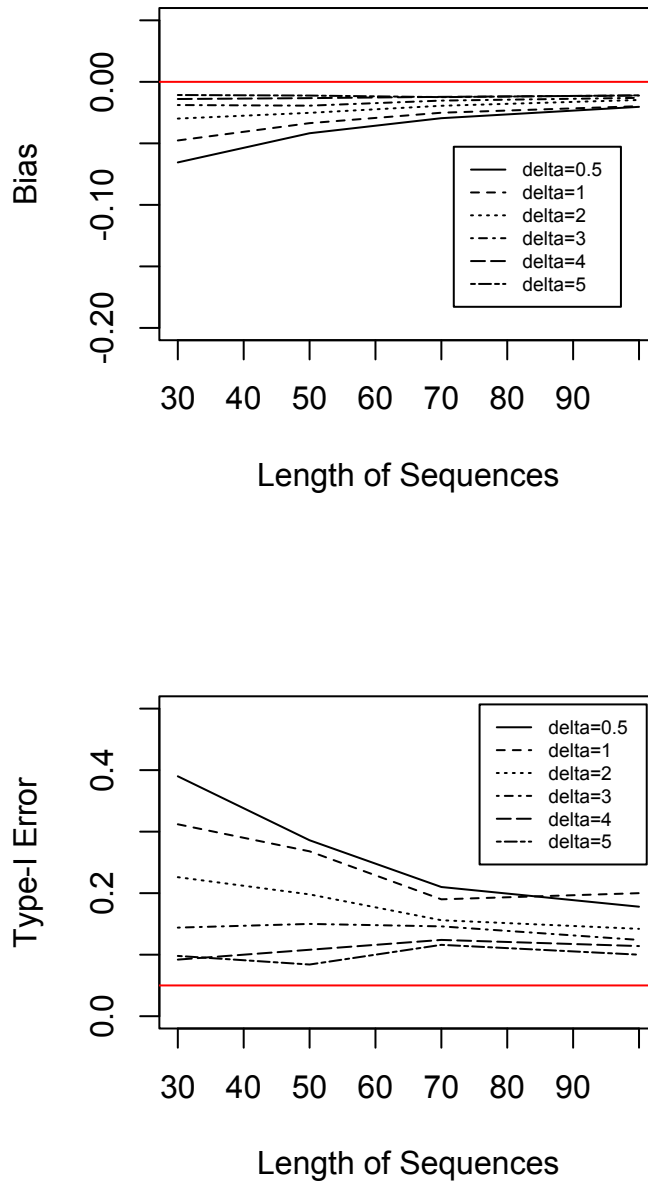


Figure 10.1.: Bias and Type-I Error in Relation to Length and Delta

For small effects (Figure 10.3.a; $\beta = 0.2$; odds ratio of 1.22), the sequences length is more important. For example, sequences with 30 intervals need about 80 observational units to achieve a power of .80, whereas sequences with 50 intervals need only about 30 units to achieve the same power. For medium sizes such as (Figure 10.3.b; $\beta = 0.4$; odds ratio of 1.49) sequences of with a length of 50 will be sufficient even with small sample sizes. If effects are really big (Figure 10.3.d; $\beta = 0.8$; odds ratio of 2.23) even small sample sizes with small sequences will be sufficient. Please note that to the knowledge of this monograph's author, no benchmarks for estimates of logit models exist concerning what is *small*, *medium*, and *big*. The categorization here was chosen as a rough orientation for the power analysis. In practice, the evaluation of effect size should depend on the subject.

The power simulation is limited in three ways. The first limitation is that only the actor effect was increased. However, the actor effect was the only effect that showed a bias in the preliminary simulations; therefore, this simulation serves as a lower bound estimation for the other effects as well. The second limitation is that the actor effect was only increased, but the bias would increase the power if negative actor effect is analyzed. Moreover, the bias would overestimate the negative effect. A negative actor effect means that showing a behavior at $t - 1$ decreases the probability at t . That would lead to an oscillating behavior. However, for all cases, in which the actor effect is expected to be more or less stable (positive values), the results can be applied. Finally, the third limitation is that combinations of actor, partner and especially interaction effects might lead to further biases which are not revealed yet.

One feature of the aggregated logit model is its computation time. For example, running one model with $N = 1000$ and $L = 1000$ on an Intel(R) Core(TM) i7-3770 CPU with one 3.40Ghz core reserved only for R took 1 min. and 9 sec. (recoding inclusive).

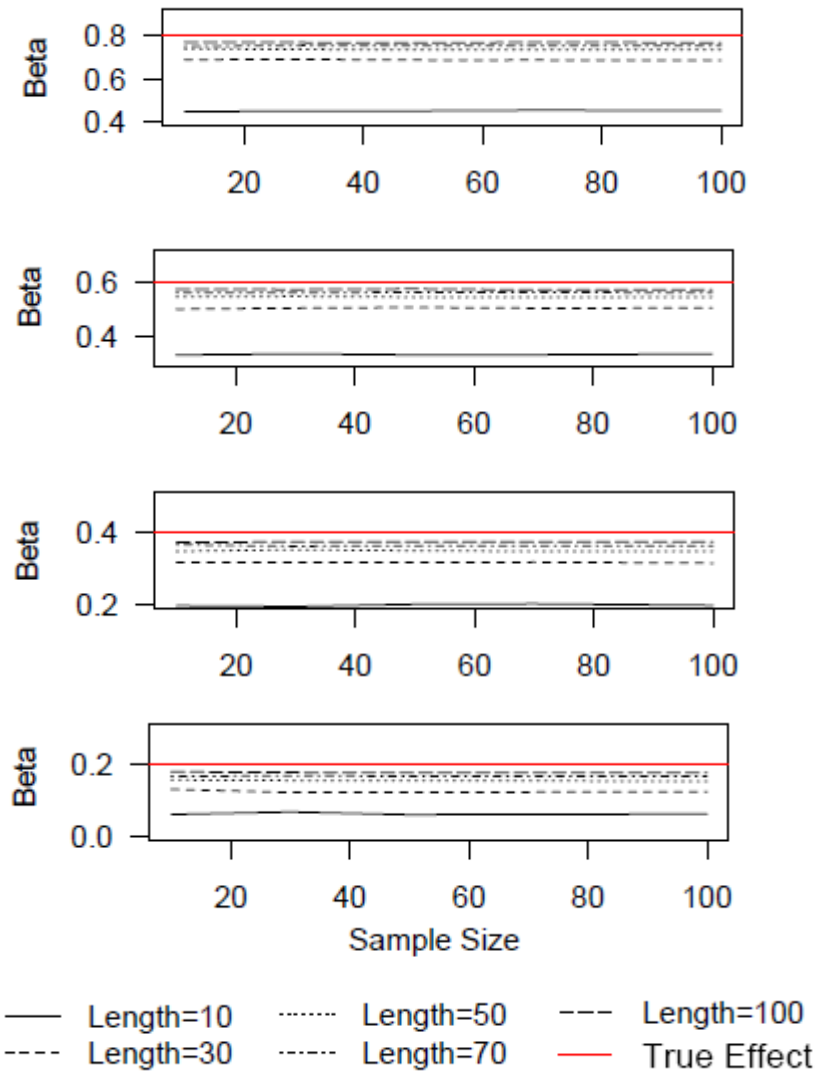


Figure 10.2.: Bias for Actor Effect

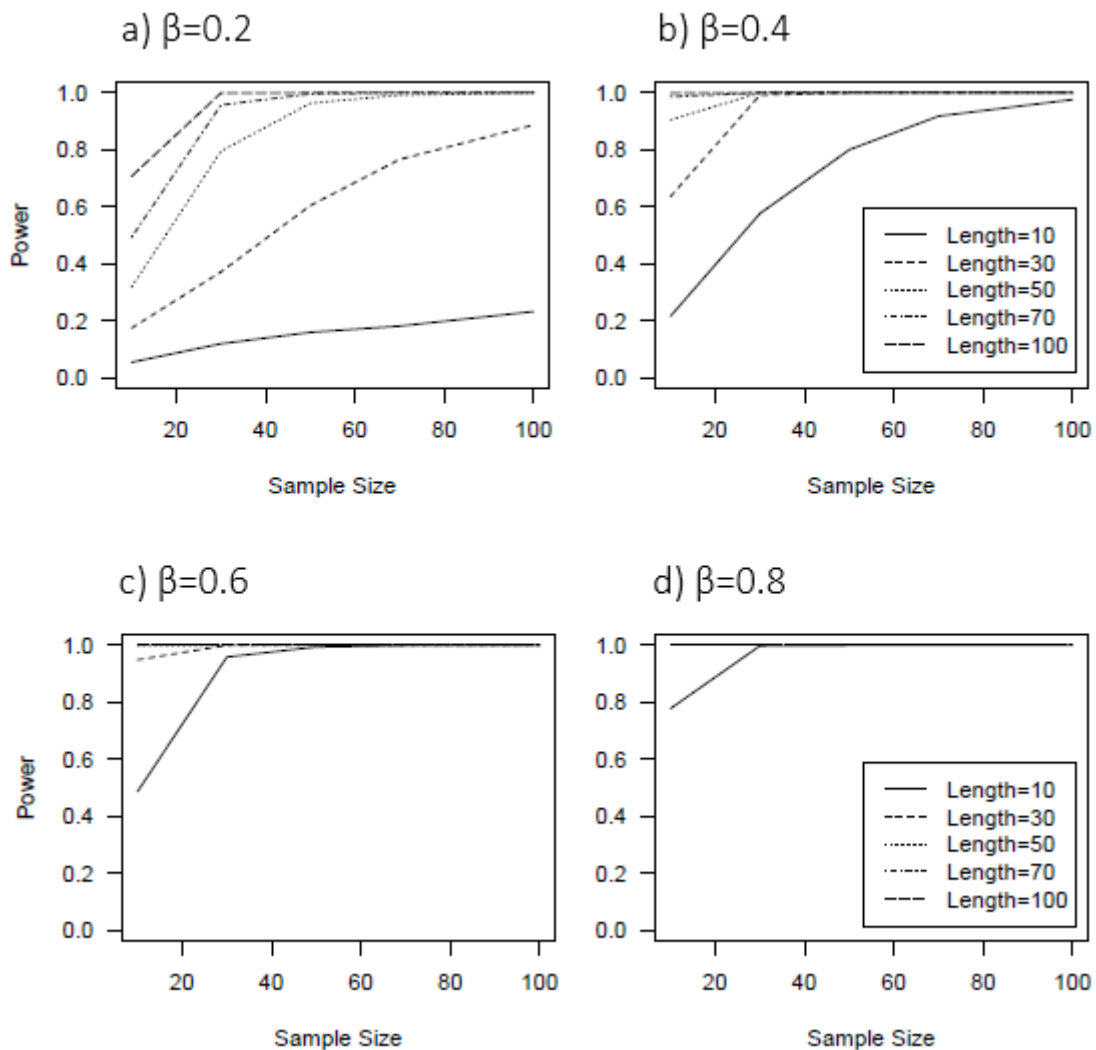


Figure 10.3.: Power Simulation for Aggregated Logit Model

10.4. Multilevel APIM

Estimation of generalized multilevel models takes a lot of computation time especially when many random effects are modeled. Thus, for the first simulation, only a random intercept model was used as a minimal example. However, computation takes still a lot of time using lme4 (several minutes per iteration). Because of that, the default optimizer (a combination of Nelder-Mead and bobyqa; Nelder and Mead, 1965) was replaced with the "bobyqa" optimizer (Powell, 2009), which computes faster. However,

in some cases the optimizer may stop prematurely giving suboptimal results. Nine combinations of sample size and sequence length were simulated ($N = 10, N = 50, N = 100; L = 10, L = 50, L = 100$). For each combination, the effect size was either $b = 0, b = 0.2, b = 0.4, \text{ or } b = 1.00$. Finally, it was varied whether the effect was an actor effect or a partner effect except for the zero effect condition. Therefore, a total of 63 conditions were simulated, each based on 500 simulated samples. The initial state distributions were chosen to be randomly drawn from a uniform distribution.

Estimates for Type-I error and bias varied only slightly across the simulation conditions. Hence, they are reported in aggregated form. The nominal error rate was .05; the mean estimated Type-I error for both intercepts was .0478, the bias was -0.0003; for the actor effects the error was .0522 and the *bias* -0.008; for the partner effect it was error .0522 and *bias* -.0056; finally partner*actor interaction showed an error of .0527. Overall, deviation of the true values was very small. Moreover, no correlations were found between the conditions and error-rate or bias.

For the power simulations, increasing the actor or partner effect above zero did not affect the Type-I error (they were always close to .05) and did not induce bias on the other effect (estimates varied closely around zero). Therefore, only the power and the bias of the effect that was increased are displayed in Table 10.6.

The multilevel approach performs overall well. Actor and partner effects are estimated unbiased. The multilevel model performs also better in terms of power than the aggregated logit model did. For example, the condition $N = 10, b = 0.2$ with $L = 100$ achieved a power of .858 for the multilevel approach, while the aggregated logit model achieved only a power of .701. If sequences are short but sample size is big, the difference becomes smaller .842 for the multilevel model vs. .800 for the aggregated logit model. Comparing the conditions closely reveals that the multilevel model's power seems to depend only on the number of observed transition regardless if they stem from long sequences or a bigger sample size. For example, comparing the condition $N = 100$ with $L = 10$ with the condition $N = 10$ with $L = 100$ results in very similar power estimates. Moreover, the multilevel model had only few problems with zero frequencies which occurred by chance. One exception though is condition $\beta = 1; N = 10; L = 10$. The big effect size combined with few small sequences resulted in resulted in many empty cells. This was also the only condition in which the optimizer produced many warning (for over 90% of the samples).

For small effects ($\beta = 0.2$; odds ratio of 1.22) and medium effects ($\beta = 0.4$; odds ratio of 1.49) a good power (above .80) can be achieved even with small sample sizes ($N = 10$) if the sequences are very long ($L > 100$). Vice versa short sequences ($L = 10$)

achieve the same power if sample sizes are big ($N > 100$). The same goes for medium sized samples ($N = 50$) with sequences of $L = 50$. If effects are big ($\beta = 0.8$; odds ratio of 2.23) even small sample sizes ($N = 10$) with small sequences ($L = 10$) will achieve a very high power ($> .95$).

The biggest disadvantage with the multilevel model is the computation time, which increases exponentially with increasing sample size and sequences length. For example, running one multilevel model with $N=1000$ and $L=1000$ on an Intel(R) Core(TM) i7-3770 CPU with one 3.40Ghz core reserved only for R took 43 min. and 30 sec. (with bobyqa optimizer, recoding inclusive).

Table 10.6.: MLM-APIM: Power-Analysis for Actor und Partner

Simulation conditions	Actor		Partner	
	$\hat{\beta}$	H_0 rejections	$\hat{\beta}$	H_0 rejections
$\beta=0.2$; N=10; L=10	0.16	.088	0.19	.094
$\beta=0.2$; N=10; L=50	0.20	.574	0.20	.598
$\beta=0.2$; N=10; L=100	0.20	.858	0.20	.870
$\beta=0.2$; N=50; L=10	0.19	.480	0.20	.512
$\beta=0.2$; N=50; L=50	0.20	1.000	0.20	1.000
$\beta=0.2$; N=50; L=100	0.20	1.000	0.20	1.000
$\beta=0.2$; N=100; L=10	0.20	.842	0.20	.860
$\beta=0.2$; N=100; L=50	0.20	1.000	0.20	1.000
$\beta=0.2$; N=100; L=100	0.20	1.000	0.20	1.000
$\beta=0.4$; N=10; L=10	0.37	.350	0.40	.402
$\beta=0.4$; N=10; L=50	0.40	.988	0.40	.988
$\beta=0.4$; N=10; L=100	0.40	1.000	0.41	1.000
$\beta=0.4$; N=50; L=10	0.39	.982	0.40	.986
$\beta=0.4$; N=50; L=50	0.40	1.000	0.40	1.000
$\beta=0.4$; N=50; L=100	0.40	1.000	0.40	1.000
$\beta=0.4$; N=100; L=10	0.39	1.000	0.40	1.000
$\beta=0.4$; N=100; L=50	0.40	1.000	0.40	1.000
$\beta=0.4$; N=100; L=100	0.40	1.000	0.40	1.000
$\beta=1$; N=10; L=10*	1.05	.978	1.11	.988
$\beta=1$; N=10; L=50	1.00	1.000	1.01	1.000
$\beta=1$; N=10; L=100	1.00	1.000	1.00	1.000
$\beta=1$; N=50; L=10	1.00	1.000	1.00	1.000
$\beta=1$; N=50; L=50	1.00	1.000	1.00	1.000
$\beta=1$; N=50; L=100	1.00	1.000	1.00	1.000
$\beta=1$; N=100; L=10	1.00	1.000	1.00	1.000
$\beta=1$; N=100; L=50	0.99	1.000	1.00	1.000
$\beta=1$; N=100; L=100	1.00	1.000	1.00	1.000

Notes: $\hat{\beta}$ is estimated effect; β is true effect; N is sample size; L is length of sequence;
 * over 90% of models had thrown optimizer warning, which might be caused by the fact that the big effect size combined with small sample size and short sequences resulted in samples in which many sequences had only a constant state; 500 samples were drawn per simulation condition; H_0 rejections are the power (1 - Type-II error).

10.5. Basic Markov APIM

The same combinations of sample size and sequence length were simulated as in Chapter 10.4 ($N = 10, N = 50, N = 100; L = 10, L = 50, L = 100$). For each combination, the effect size was either $b = 0, b = 0.2, b = 0.4,$ and $b = 1.00$. Finally, it was manipulated whether the effect was an actor effect or a partner effect except for the zero effect condition. Therefore, a total of 63 conditions were simulated. The initial state distributions were randomly drawn from a uniform distribution. Simulations for determining Type-I error rate were based on 1000 sample, while each other condition in this simulation is based only on 100 simulated samples.

The reason for that is the following: because of bootstrapping, the function took quite long for simulation. With $K = 1000$ bootstrap-samples a condition with many observations like $N = 100$ and $L = 100$ took several weeks. The reason for that is that bootstrapping itself is a simulation technique. So if 1.000 bootstrapping samples are drawn per bootstrapping, and 1.000 simulations-samples are created, the total number of samples becomes 1.000.000. Therefore, K was set to 100 for all 54 power-simulations, leading to a less precise simulation. However, no one would run a bootstrap with less than 1000 bootstrap-samples nowadays, and thus, creating more simulation samples with less bootstrap samples would create less realistic results. Hence, precision was treated for a more realistic simulation. Because Type-I error is often considered more important than Type-II, the 9 Type-I error simulations were run with 1000 simulation-samples.

The simulation was conducted via Amazon Web Services (cloud computing) because the simulations were very time demanding. However, the simulation server crashed several times, caused by the bootstrap function. Logs were inconclusive, and timing was not predictable. Temporal files were created so that a simulation condition could be restarted at the point where the server crashed. Rerunning the bootstrap at exact the same point where the server crashed previously did not provoke another immediate crash. Therefore, results were obtained for every simulation condition. This might affect the simulated results. The crashes occurred more often for small samples with short sequences yet high β s. However, the maximum number of crashes were two in one simulation condition. There is one exception which is the condition with $\beta = 1, N = 10,$ and $L = 10$. Simulations within that conditions crashed nearly every time, those that did not crash return NaNs (not a number) and Inf (Infinity) often. This condition also had a lot of cells with zero frequencies which might be one explanation.

Estimates for Type-I error and bias varied only slightly across all conditions. Hence, they are reported in aggregated form. Nominal error rate was .05; the mean estimated Type-I error for both intercepts was .051, the bias was .00002; for the actor effects the error was .048 and the bias -0.002 ; for the partner effect it was error .064 and bias -0.003 ; finally partner*actor interaction showed an error of .046 and bias was -0.001 . Overall, deviation of the true values was very small, and the Type-I error rate seems to be even closer to the nominal error rate than the previous models. Moreover, no correlations were found between the conditions and error-rate or bias. Hence, ordinary non-parametric bootstrap with 1000 bootstrapping-samples is a valid method for generating p -values for actor- and partner-effects that were computed from transition probabilities. That is, at least regarding the Type-I-error.

The basic Markov APIM performs similarly as the multilevel APIM regarding power. For small effects ($\beta = 0.2$; odds ratio of 1.22) and medium effects ($\beta = 0.4$; odds ratio of 1.49) a good power (above .80) can be achieved even with small sample sizes ($N = 10$) if the sequences are very long ($L > 100$). Vice versa short sequences ($L = 10$) achieve the same power if sample sizes are big ($N > 100$). Same goes for medium sized samples ($N = 50$) with sequences of length=50. If effects are really big ($\beta = 0.8$; odds ratio of 2.23) even small sample sizes ($N=10$) with small sequences ($L = 10$) will achieve a very high power ($>.95$).

As with the multilevel model, computation can take a long time. For example, running one multilevel model with $N=1000$ and $L=1000$ with 1000 bootstrap-samples on an Intel(R) Core(TM) i7-3770 CPU with one 3.40Ghz core reserved only for R took 34 min. and 26 sec. Overall, there is not much of a difference between the multilevel APIM and the basic Markov APIM. Both take very long for extreme sample sizes and long sequences; power estimates are similar, the same goes for Type-I error. The multilevel APIM seems to be more robust regarding extreme conditions that cause many empty cells. However, the basic Markov APIM is nearly as fast as the aggregated logit model, when bootstrapping is not performed. Therefore, in cases in which p -values are not needed, and empty cells are not likely, the basic Markov model should be superior to aggregated logit models, yet faster than the multilevel model.

Table 10.7.: Basic Markov APIM: Power-Analysis for Actor und Partner

Simulation conditions	Actor		Partner	
	$\hat{\beta}$	H_0 rejections	$\hat{\beta}$	H_0 rejections
$\beta=0.2$; N=10; L=10	0.19	.06	0.21	.03
$\beta=0.2$; N=10; L=50	0.18	.55	0.19	.61
$\beta=0.2$; N=10; L=100	0.19	.85	0.19	.86
$\beta=0.2$; N=50; L=10	0.19	.49	0.19	.50
$\beta=0.2$; N=50; L=50	0.20	1.00	0.20	1.00
$\beta=0.2$; N=50; L=100	0.20	1.00	0.20	1.00
$\beta=0.2$; N=100; L=10	0.21	.90	0.21	.90
$\beta=0.2$; N=100; L=50	0.20	1.00	0.20	1.00
$\beta=0.2$; N=100; L=100	0.20	1.00	0.20	1.00
$\beta=0.4$; N=10; L=10	0.40	.21	0.41	.01
$\beta=0.4$; N=10; L=50	0.39	.99	0.39	.99
$\beta=0.4$; N=10; L=100	0.39	1.00	0.41	1.00
$\beta=0.4$; N=50; L=10	0.39	.99	0.39	.99
$\beta=0.4$; N=50; L=50	0.40	1.00	0.40	1.00
$\beta=0.4$; N=50; L=100	0.40	1.00	0.40	1.00
$\beta=0.4$; N=100; L=10	0.40	1.00	0.42	1.00
$\beta=0.4$; N=100; L=50	0.40	1.00	0.40	1.00
$\beta=0.4$; N=100; L=100	0.40	1.00	0.40	1.00
$\beta=1$; N=10; L=10	*	*	*	*
$\beta=1$; N=10; L=50	1.00	1.00	1.02	1.00
$\beta=1$; N=10; L=100	1.00	1.00	1.00	1.00
$\beta=1$; N=50; L=10	0.98	1.00	1.02	1.00
$\beta=1$; N=50; L=50	1.00	1.00	1.00	1.00
$\beta=1$; N=50; L=100	1.00	1.00	1.00	1.00
$\beta=1$; N=100; L=10	1.00	1.00	1.00	1.00
$\beta=1$; N=100; L=50	0.99	1.00	1.00	1.00
$\beta=1$; N=100; L=100	1.00	1.00	1.00	1.00

Notes: $\hat{\beta}$ is estimated mean effect; β is true effect; N is sample size; L is length of sequence; 1000 samples were drawn per simulation condition.; * more than 90% of estimates resulted in infinite values or "Not a Number", which might be caused by the big effect size combined with the small sample size and short sequences. Many sequences had only a constant state; H_0 rejections are the power (1 - Type-II error).

10.6. Restricted Hidden Markov (Latent Hazard)

Four simulation studies were conducted for the latent Markov Model. All four studies investigated bias and simulated standard error for hazard and emissions. The first two studies differed only regarding their emissions (high vs. low), and both used the same conditions for hazard ($hazard = .01$, $hazard = .05$, $hazard = .10$), sample size ($N = 10$, $N = 50$, $N = 100$) and sequence length (number of time intervals; $L = 10$, $L = 50$, $L = 100$). The third and fourth simulations are post-hoc simulations that were conducted for investigating the results of the former simulation further. The fourth simulation investigated whether the bias induced on small samples by a very low hazard is the same for very high hazards. The fourth simulation investigated the effect of sample size and length in smaller steps, using a small hazard ($hazard = .01$ and high emissions).

High emissions were operationalized as the following: emissions are high when if states are strongly connected to one of the observed states. Therefore emissions in the first simulation were constructed as the following: If the hidden state was the initial state the probability for showing state 1 was .90, for showing state 2 it was .03, for the third state .03, and for the fourth .04. As for the absorbing state the emissions were .03, .03, .04 and .90. Thus, the first observed state is a strong indicator for being in the initial state, and the fourth state is a strong indicator of being in the absorbing state.

For low emissions the probabilities were the following: If the hidden state was the initial state the probability for showing state 1 was .70, and for each of the other states it was .10. For the absorbing state, the probability of showing the fourth observable state was .70 and .10 for the other states.

Table 10.8 displays the results for the high emissions, while Table 10.9 shows the results for low emissions. The overall finding is that the latent hazard model performs well if sequences are longer than 50 time intervals or if the sample size is bigger than 50. The bias is only big if the sample size is 10 and sequence length is 10 and if the hazard is very small ($h = .01$). Low emissions seem to add further bias.

The simulation also shows that a higher hazard will result in a lower bias. Three follow-up simulations were conducted to see if this is also true for extremely high values for the hazard. The conditions were $N = 10$, $L = 10$. Hazard was manipulated as being .90, .95 and .99. The bias for the hazard of .90 was essentially the same as for .10, for .95 the same as for .05 and for .99 the same as for .01. The biases differed only slightly on the second digit, which indicates that the bias is especially high the closer the true hazard is to either one or zero.

Another finding was that the estimates with low values for N and L tend to be not normally distributed. As shown in Figure 10.5 the distribution becomes normal if the samples have a size of $N = 40$ and $L = 40$ or greater. Therefore, the simulated standard errors of the simulations should not be used for estimating confidence intervals if sample sizes and sequence length is small. If confidence intervals are created via resampling-techniques such as bootstrapping or permutation tests, one should use non-parametric estimation methods that do not assume normally distributed estimates.

The fourth simulation investigated the effect of sample size and length in smaller steps, using a small hazard ($hazard = .01$) and high emissions. N started at 15 and was increased by 5 observational units per simulations step, and L started at 5 and was also increased per 5 intervals. 500 samples were drawn per iteration. The results can be seen in Figure 10.4. The plot shows that the bias for smaller samples sizes can be reduced if longer sequences are used. So for a sample size of $N = 15$, a sequence length of at least 40 time intervals is recommended. If the sample size is $N = 20$, a length of $L = 25$ is sufficient, and if the sample size is at least $N = 30$, a length of $L = 20$ is sufficient, and for $N > 45$ even a length of $L = 15$ seems fine.

However, these recommendations are only true for high emissions, meaning that good indicators must be chosen. If only weak indicators are used, the sample size or the sequence length should be increased. On the other hand, a low hazard of $h = .01$ was chosen which should induce a bigger bias as for example $h = .05$. Hence, if a higher hazard is expected, lower samples sizes and shorter sequences might be sufficient. Because of the above, these recommendations should only be treated as an orientation.

The corresponding R-script (see Appendix B.7) was written in a way that it could be adapted easily for any other constellation of emissions and hazard. Therefore, it can be used to get the optimal sample size and sequence length tailored to a specific application. Similar as it is done with a power-analysis before sampling.

Table 10.8.: Hidden Markov: Bias and SE for High Emissions

Simulation conditions	Hazard		Emissions		
	bias	SE	bias	max bias	SE
hazard=.01; N=10; L=10	.22	.18	.22	.44	.04
hazard=.01; N=10; L=50	<.01	.01	<.01	<.01	<.01
hazard=.01; N=10; L=100	<.01	<.01	<.01	<.01	<.01
hazard=.01; N=50; L=10	.02	.01	.03	.06	.01
hazard=.01; N=50; L=50	<.01	<.01	<.01	<.01	<.01
hazard=.01; N=50; L=100	<.01	<.01	<.01	<.01	<.01
hazard=.01; N=100; L=10	<.01	<.01	<.01	.01	<.01
hazard=.01; N=100; L=50	<.01	<.01	<.01	<.01	<.01
hazard=.01; N=100; L=100	<.01	<.01	<.01	<.01	<.01
hazard=.05; N=10; L=10	0.01	.01	.02	.05	.01
hazard=.05; N=10; L=50	<.01	<.01	.01	.01	<.01
hazard=.05; N=10; L=100	0.01	<.01	0.01	<.01	<.01
hazard=.05; N=50; L=10	<.01	<.01	<.01	.01	<.01
hazard=.05; N=50; L=50	<.01	<.01	<.01	<.01	<.01
hazard=.05; N=50; L=100	<.01	<.01	<.01	<.01	<.01
hazard=.05; N=100; L=10	<.01	<.01	<.01	<.01	<.01
hazard=.05; N=100; L=50	<.01	<.01	<.01	<.01	<.01
hazard=.05; N=100; L=100	<.01	<.01	<.01	<.01	<.01
hazard=.10; N=10; L=10	.01	<.01	.01	.03	<.01
hazard=.10; N=10; L=50	.01	<.01	.01	.03	<.01
hazard=.10; N=10; L=100	.01	<.01	.01	.02	<.01
hazard=.10; N=50; L=10	<.01	<.01	.01	<.01	<.01
hazard=.10; N=50; L=50	<.01	<.01	<.01	<.01	<.01
hazard=.10; N=50; L=100	<.01	<.01	<.01	<.01	<.01
hazard=.10; N=100; L=10	<.01	<.01	<.01	<.01	<.01
hazard=.10; N=100; L=50	<.01	<.01	<.01	<.01	<.01
hazard=.10; N=100; L=100	<.01	<.01	<.01	<.01	<.01

Notes: hazard is latent hazard; N is sample size; L is length of sequence; bias is the mean of absolute deviations from the true hazard; SE is simulated standard error; mean and SE of emissions are absolute meas over all 8 emissions; max bias is the highest mean absolute bias of the 8 emissions; 1000 samples were drawn per simulation condition.; Emissions were .90/.03/.03/.04 for the first state and .03/.03/.04/.90 for the absorbing state.

Table 10.9.: Hidden Markov: Bias and SE for Low Emissions

Simulation conditions	Hazard		Emissions		
	bias	SE	bias	max bias	SE
hazard=.01; N=10; L=10	.32	.14	.10	.38	.03
hazard=.01; N=10; L=50	.01	<.01	<.01	.01	<.01
hazard=.01; N=10; L=100	<.01	<.01	<.01	<.01	<.01
hazard=.01; N=50; L=10	.05	.03	.02	.10	.02
hazard=.01; N=50; L=50	<.01	<.01	<.01	<.01	<.01
hazard=.01; N=50; L=100	<.01	<.01	<.01	<.01	<.01
hazard=.01; N=100; L=10	.01	.01	.01	.02	.01
hazard=.01; N=100; L=50	<.01	<.01	<.01	<.01	<.01
hazard=.01; N=100; L=100	<.01	<.01	<.01	<.01	<.01
hazard=.05; N=10; L=10	0.06	.03	.02	.08	.02
hazard=.05; N=10; L=50	<.01	<.01	<.01	.01	<.01
hazard=.05; N=10; L=100	0.01	<.01	<.01	.01	<.01
hazard=.05; N=50; L=10	<.01	<.01	<.01	.01	<.01
hazard=.05; N=50; L=50	<.01	<.01	<.01	<.01	<.01
hazard=.05; N=50; L=100	<.01	<.01	<.01	<.01	<.01
hazard=.05; N=100; L=10	<.01	<.01	<.01	<.01	<.01
hazard=.05; N=100; L=50	<.01	<.01	<.01	<.01	<.01
hazard=.05; N=100; L=100	<.01	<.01	<.01	<.01	<.01
hazard=.10; N=10; L=10	.03	.01	.01	.03	.01
hazard=.10; N=10; L=50	.01	<.01	<.01	.01	<.01
hazard=.10; N=10; L=100	.01	<.01	<.01	.01	<.01
hazard=.10; N=50; L=10	<.01	<.01	<.01	<.01	<.01
hazard=.10; N=50; L=50	<.01	<.01	<.01	<.01	<.01
hazard=.10; N=50; L=100	<.01	<.01	<.01	<.01	<.01
hazard=.10; N=100; L=10	<.01	<.01	<.01	<.01	<.01
hazard=.10; N=100; L=50	<.01	<.01	<.01	<.01	<.01
hazard=.10; N=100; L=100	<.01	<.01	<.01	<.01	<.01

Notes: hazard is latent hazard; N is sample size; L is length of sequence; bias is the mean of absolute deviations from the true hazard; SE is simulated standard error; mean and SE of emissions are absolute meas over all 8 emissions; max bias is the highest mean absolute bias of the 8 emissions; 1000 samples were drawn per simulation condition.; Emissions were .70/.10/.10/.10 for the first state and .10/.10/.10/.70 for the absorbing state.

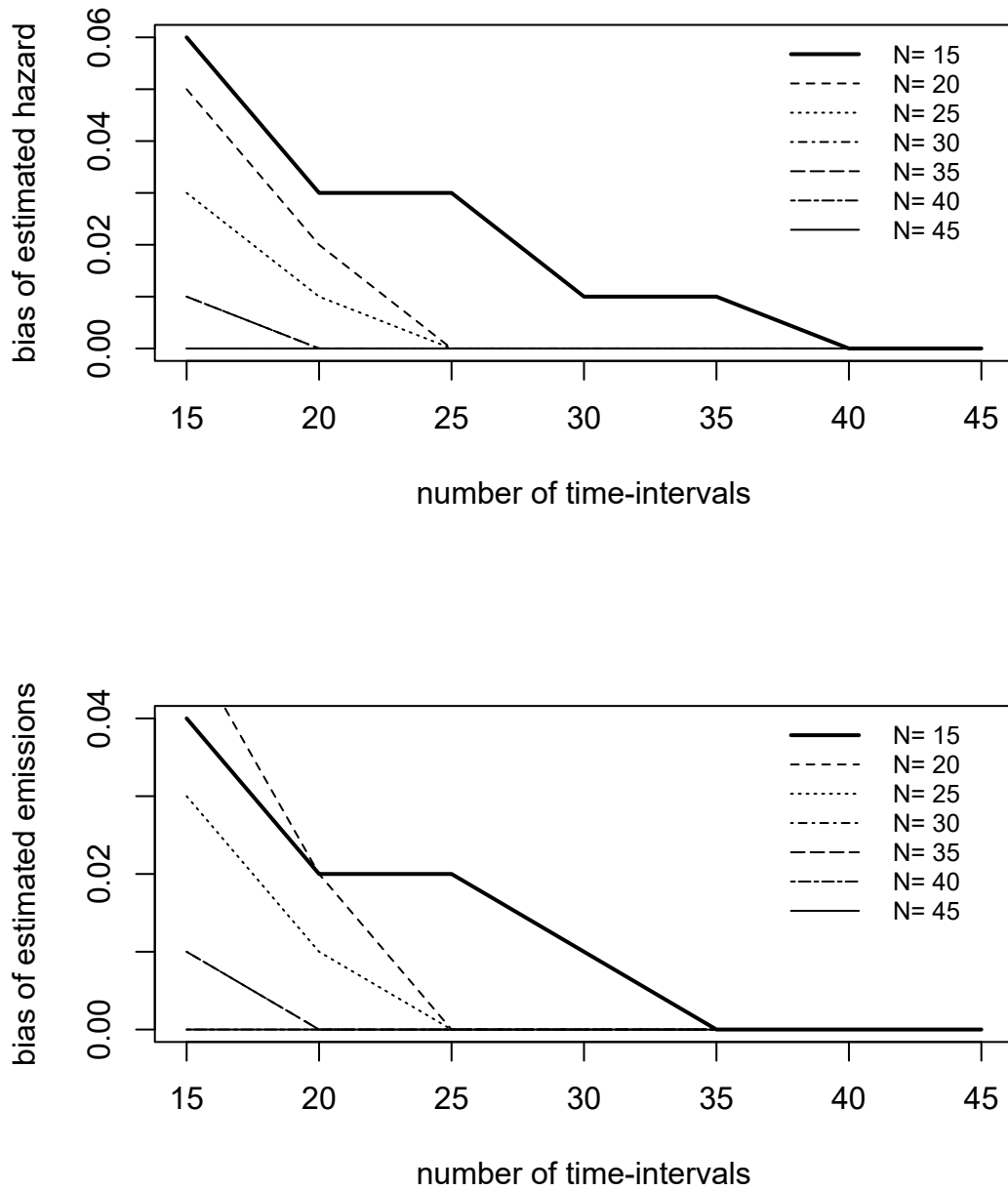


Figure 10.4.: Bias for Latent Hazard (top) and its Emissions (bottom)

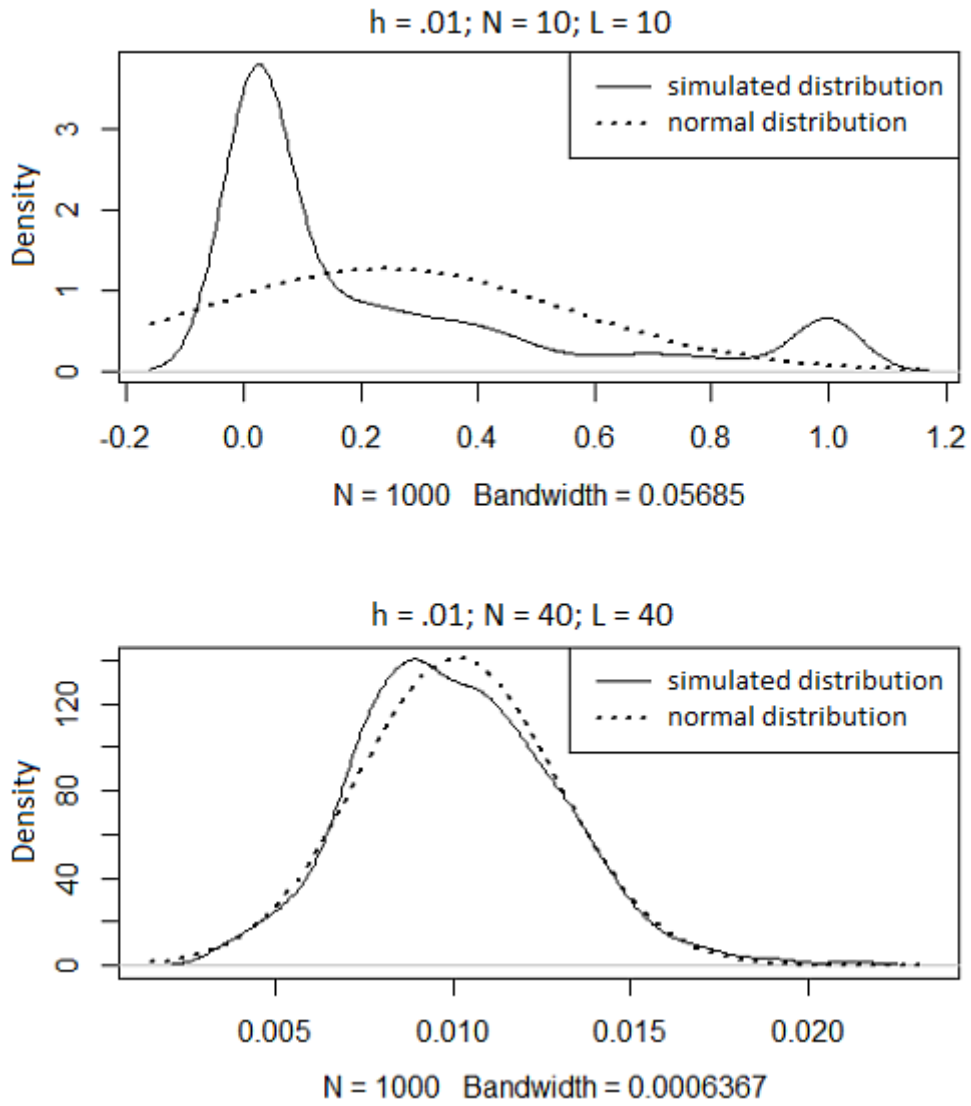


Figure 10.5.: Distribution of Hazard-Estimates

10.7. Unrestricted Hidden Markov

Several simulation studies were conducted for the hidden Markov model. A first simulation was conducted for investigating the effect of sample size and sequence length on AIC and BIC, and how this might lead to a correct or incorrect number of latent states. Three models were simulated: one basic Markov model (serves as comparison model without latent structure; see Table 10.10), a hidden Markov model with two latent states (see Table 10.11), and one with three latent states (see Table 10.12). Each model was simulated to have four indicators. Emissions, transitions and initial state distribution were chosen arbitrarily in the first simulation. However, all three models had to be changed for creating meaningful results; the tables show the models that were used in the last run of simulations. The results are shown in Tables 10.13 (correct Model selection) and Figures 10.6 (biases for estimates of the two latent state model) and 10.7 (biases for estimates of the three latent state model) and a closer look on bias patterns for the model with three latent states (Figure 10.8). The basic Markov model was also used in the following Chapter 10.8, and biases for transition probabilities can be found there in more detail (see Chapter 10.17).

Table 10.10.: True Basic Markov Model

Initial State Probabilities	State 1	State 2	State 3	State 4
	1.00	0.00	0.00	0.00
Transitions	→State 1	→State 2	→ State 3	→ State 4
State 1 →	.80	.10	.10	.00
State 2 →	.10	.70	.10	.10
State 3 →	.10	.10	.70	.10
State 4 →	.00	.10	.10	.80

Notes: States refer to observed states. No hidden states are assumed for the basic Markov model. left hand of → transition from; right hand of → transition to.

The first test runs showed that identifying which of the estimated hidden states corresponded to which true hidden state was too difficult with initial simulation values, at least for an automated process. Therefore, emission probabilities were modified in a way that each hidden state should show a distinct emission pattern. Taking the model with three latent classes as an example, hidden state 1 goes 90% of the time with observed state 1, whereas hidden state 2 goes 90% with observed state 2 or 3, and

hidden state 3 goes 80% with observed state 4. Therefore, emission patterns should be distinguishable even with statistical noise.

The next problem was that the first batch of simulations always favored a model with two latent classes regardless of sample size and sequence length. The reason for that was similar to a phenomenon called *the problem of equivalent models*. The original problem of equivalent models was formalized by Stelzl (1986). She pointed out that different statistical models can be fitted on the same data resulting in the same model fit, even if they imply completely different causation models. For example, the role of dependent and independent variables in regression can be switched, but R^2 will always be the same.

The problem in this simulation was similar: taking the basic Markov model, for example, applying all three models on the data generated by the basic Markov model resulted in equally good fit (concerning likelihood). Hence, the data can be explained by all three models equally good, even though the basic Markov model is the true model. The parsimony principle demands that the least complex model should be chosen in such a situation. Because of that, AIC and BIC penalize models for their number of free parameters (also referred to as degrees of freedom, df). Thus, if three models have a similar likelihood, they will always point to the model with the lowest number of free parameters. In this case, the model with the lowest number of free parameters is actually the model with two latent states ($df = 9$), followed by the basic Markov model ($df = 15$), and the most complex model is the hidden Markov model with three latent states ($df = 17$).

The model with two latent states has 9 parameters that can vary freely: two latent states exist, and thus, two the initial states have had to be estimated. However, the initial states have to sum up to one, and therefore, only one can vary freely. Each row of the transition table has to sum up to one, too, and thus two parameters can vary freely; for the emissions, each row has to sum up to one, too. The emissions matrix has two rows and four columns, and thus, six free parameters. Whereas the basic Markov has 15 parameters that can vary freely: There are four states, and thus three free parameters for the initial state distribution; and three free parameters per row of the transition matrix, so for four rows a total of 12 free parameters must be estimated. There are no free parameters for the emissions. Finally, the model with three latent states has 17 free parameters: Two for the initial state distribution, six for the transition probability and nine for the emissions.

However, all three model generated sequences with very similar distributions. The distribution plots became somewhat dissimilar after adjusting the emissions for the

Table 10.11.: True Hidden Markov Model with Two Latent States

Initial State	State 1	State 2		
Probabilities	1.00	.00		
Transitions	→State 1	→State 2		
State 1 →	.90	.10		
State 2 →	.05	.95		
Emissions	X=1	X=2	X=3	X=4
State 1 →	.90	.03	.03	.04
State 2 →	.03	.03	.04	.90

Notes: State refers to the latent state, while X refers to the observes states; left hand of → transition from; right hand of → transition to.

reasons mentioned above, but the initial stability rates were so low that observations would switch very fast from one state to another. Hence, the stability was increased so observations would stay for a longer period within a certain state. Finally, the first initial state was set to one and transitions were set so that observations would move slowly from state one to state two, from state two to state three (if it exists), and - for the basic Markov model - from state 3 to state 4. This was done because sequences entered their equilibrium states very fast if a uniform distribution was used. However, especially in Experimental Psychology, it is more plausible that observational units are manipulated to be in the same states at the beginning of the experiment and then transition into other states while the experiment is running. However, this setup for the simulation increases the importance of sequence length, because observational units are very similar at the beginning of all three models and need several time intervals for forming distinct patterns.

Table 10.13 shows the results of the final simulation regarding model selection. A basic Markov model, a model with two latent states, and a model with three latent states, were fitted on each dataset. AIC and BIC were used to determine the correct model. Correct classification is written in boldface.

The AIC performs better than the BIC, especially for small sample sizes. For example, the AIC correctly classified 71% of basic Markov models correctly in the "basic Markov; $N = 10$; $L = 10$ " condition, whereas the BIC classified only 5% correctly. There is one exception to this trend: the BIC performs better for the simulations of the

Table 10.12.: True Hidden Markov Model with Three Latent States

Initial State Probabilities	State 1	State 2	State 3	
	1.00	0.00	0.00	
Transitions	→ State 1	→ State 2	→ State 3	
State 1 →	.70	.20	.10	
State 2 →	.10	.70	.20	
State 3 →	.05	.05	.90	
Emissions	X=1	X=2	X=3	X=4
State 1 →	.90	.03	.03	.04
State 2 →	.05	.45	.45	.05
State 3 →	.05	.05	.10	.80

Notes: State refers to the latent state, while X refers to the observes states; left hand of → transition from; right hand of → transition to.

hidden Markov model with two latent states. However, inspecting the misclassification rate of BIC for the other simulations reveals that it seems to have a bias towards the most simple model, in this case, the hidden Markov model with two latent states. That might explain why the BIC performs so well even with a small number of observations for that particular model.

However, the behavior of the AIC for the two latent states model is different than for the other models as well. The rate of correct classifications increases with increasing number of observations and longer sequences for the basic Markov model and the three latent states model, whereas it decreases for the model with two latent states. A single simulation with $N = 100$, $L = 100$, and 500 samples confirmed that this trend continued (down to .89 for that simulation). This indicates that AIC might not be a good choice for a model with low degrees of freedom mixed with extreme sample sizes or sequence lengths. However, for the typical sample sizes used in Psychology, the correct classification rate is still good, and the AIC can be recommended as a fit index if sample sizes small ($N = 10$) and sequences are relatively long ($L = 50$). The BIC, on the other hand, performs well if the sample sizes are high and sequences are long enough: The correct classification rate got near 100% if the sample size was at least 30 and sequence length was 50, or vice versa, if the sample size was 50 and sequence length was 30. As a minimum recommendation should be $N \geq 30$ and

$L \geq 30$, which gives 91% correct classification rate if three or less latent states are expected.

However, this view covers only the correct model selection, but the estimates were also investigated regarding potential biases. Figures 10.6 and 10.7 show the mean absolute bias for the transitions and the estimation of both hidden Markov models. There is a very small bias for the transition probabilities for the 2 latent states model. However, the bias is so small that even in the $N = 10; L = 10$ condition it rounds only up to .01. There seems to be an anomaly for the transition probabilities when the sample size is 30. This might be caused by random fluctuation due to the very small values.

The model with three latent models shows a bigger, yet not concerning bias for the transition rates. The bias seems to converge against a minimum of .03 when $N \geq 50$ and $L \geq 30$. Inspecting the bias for each transition rate separately revealed that the bias is stronger for extreme transition rates. For example, the transition rates from state 1 were stronger affected as the transition rates from state 2 for the two latent state model. The bias is also towards a uniform distribution. Hence, in case of the two latent states mode, towards transition probabilities of .50.

The bias for the emissions of the three latent state model is huge. They seem to converge at .28 if $N \geq 50$ and $L \geq 30$. The same phenomenon occurred before for the transition probabilities but stronger. This is shown in Figure 10.8 for the condition $N = 10, L = 10; N = 10, L = 100; N = 100, L = 10; \text{ and } N = 100, L = 100$. Grey bars show the estimated emissions for each observed state stacked by hidden states. The black reference lines indicate the true emissions, and thus, are the same for each of the for conditions. All emissions were heavily biased towards a uniform distribution (towards .25). However, the pattern of emissions is overall the same. Therefore, hidden states can still be interpreted via the emissions with care. Emissions can still be compared in terms like "relatively strong compared to the other" vs. "relatively weak to the other". However, the order of magnitude can change. For example, the $N = 100, L = 10$ shows that the true emission from the hidden state 3 to the observed state 4 is highest, followed by the third observed state, and the observed states 1 and 2 share the same rank. Hence the Order is $4 > 3 > (2 \ \& \ 1)$ for the true emission. However, inspecting the estimated emissions shows that the observed state 4 has still the highest emission, yet observed state 1 follows in the second place. The complete order is $4 > 1 > 3 > 2$. Therefore, ranking emissions should not be used for interpretation of hidden states, especially when differences are small.

Table 10.13.: Model Selection for Hidden Markov Simulation

Simulation conditions	AIC			BIC		
	B	2S	3S	B	2S	3S
basic Markov; N=10; L=10	.71	.25	.04	.05	.95	.00
basic Markov; N=10; L=30	.95	.01	.04	.42	.58	.00
basic Markov; N=10; L=50	.99	.00	.01	.81	.19	.00
basic Markov; N=30; L=10	.97	.01	.02	.55	.45	.00
basic Markov; N=30; L=30	1.00	.00	.00	.99	.01	.00
basic Markov; N=30; L=50	1.00	.00	.00	1.00	.00	.00
basic Markov; N=50; L=10	.99	.00	.01	.90	.10	.00
basic Markov; N=50; L=30	1.00	.00	.00	1.00	.00	.00
basic Markov; N=50; L=50	1.00	.00	.00	1.00	.00	.00
2 latent states; N=10; L=10	.05	.94	.01	.00	1.00	.00
2 latent states; N=10; L=30	.00	.97	.03	.00	1.00	.00
2 latent states; N=10; L=50	.00	.98	.02	.00	1.00	.00
2 latent states; N=30; L=10	.00	.96	.04	.00	1.00	.00
2 latent states; N=30; L=30	.00	.94	.06	.00	1.00	.00
2 latent states; N=30; L=50	.00	.93	.07	.00	1.00	.00
2 latent states; N=50; L=10	.00	.95	.05	.00	1.00	.00
2 latent states; N=50; L=30	.00	.94	.06	.00	1.00	.00
2 latent states; N=50; L=50	.00	.92	.08	.00	1.00	.00
3 latent states; N=10; L=10	.19	.60	.21	.01	.99	.00
3 latent states; N=10; L=30	.10	.07	.83	.04	.89	.07
3 latent states; N=10; L=50	.01	.00	.99	.04	.62	.34
3 latent states; N=30; L=10	.14	.04	.82	.10	.83	.07
3 latent states; N=30; L=30	.00	.00	1.00	.01	.08	.91
3 latent states; N=30; L=50	.00	.00	1.00	.00	.00	1.00
3 latent states; N=50; L=10	.06	.00	.94	.15	.44	.41
3 latent states; N=50; L=30	.00	.00	1.00	.00	.00	1.00
3 latent states; N=50; L=50	.00	.00	1.00	.00	.00	1.00

Notes: Simulation conditions show the true model, basic Markov model (Abbr. B; see Table 10.10), 2 latent states (Abbr. S2; see Table 10.11) or 3 latent states (Abbr. S3; see Table 10.12), N = samples size, L = length of sequences; 500 samples were drawn per simulation condition; Columns show classification based on AIC (Akaike information criterion) or BIC (Bayesian information criterion) for B (basic Markov Model).

This simulation is limited by the following: 1) it covers only the case of very distinctive latent states; 2) it covers only the case in which all observational units start in the same latent state; 3) it covers only models with four indicators; 4) it covers only a maximum of three latent states; 5) even after running several post-hoc simulations with alternative initial states, transition matrices, and emissions, it remains unclear why the bias is relatively small for the model with two latent states, but relatively big for three latent states; 6) optimizer settings had a huge impact on model selection and overall performance. Other optimizers and settings might result in better estimates.

Some practical recommendations: Initial states, transition matrices, and emissions can be easily changed for simulating alternative scenarios using the R-code in Chapter B.8. This might be useful for estimating the needed sample size and sequence length for cases that are not covered in this chapter. As mentioned before, the optimizer and its settings had a huge impact on the overall performance. Initially, only the EM-algorithm was used, which converged fast, but it performed not good for the hidden Markov models. As recommended by Helske and Helske (2016), MSLS-LDS (Kucherenko and Sytsko, 2005) was used as a global optimizer, and low-storage BFGS (Liu and Nocedal, 1989) was used as a local optimizer. This resulted in better model selections and reduced bias. The price for that is longer computation time. Running the EM-algorithm with 1000 different starting values needed only a few seconds for estimating a three latent class model with $N = 100$ and $L = 100$, whereas the combination of MSLS-LDS and low-storage BFGS took several minutes. However, it is highly recommended to use these or other sophisticated algorithms rather than the EM-algorithm.

The following recommendation for sample size and sequence length can be made: A hidden Markov model with only two latent states can be estimated with small sample sizes such as 10, the potential bias is very small. However, if the number of latent states is not known, but assumed to be three or less, N should be at least 30 if sequences are short ($L = 10$), or vice versa, if the sample size is small ($N = 10$) the sequences should have a length of at least 30. Moreover, sequences should always be long enough so that all possible latent states can be reached by many of the observational units. Hence, if hidden states are believed to be more stable than in this simulation, sequences should be longer. Moreover, if more than three latent states are assumed, sample sizes and sequence length should be increased. The same goes for the case that hidden states should be detected that are relatively similar opposed to the very distinct states that were used in these simulations.

Overall, AIC and BIC can be recommended for model selection. AIC might a good choice if the true model has a high number of degrees of freedom and the sample sizes are relatively small. BIC performs overall good as long as sample size and sequence length are high. Therefore, the best strategy might be to strive for the highest possible number of observations with a reasonable sequence length and then to use the BIC for model selection.

10.8. Mixture Markov

Several simulation studies were conducted for the mixture Markov model. A basic Markov model, a mixture Markov model with 2 latent classes, and a mixture Markov model with 3 latent classes were used as true models. The true values for the simulations can be found in the following tables: Simulated Models: Basic Markov Model is seen in Table 10.10, two latent classes in Table 10.14, three latent classes in Table 10.15.

As an additional condition, sample size and the lengths of sequences were iterated across the values 10, 20, 30, 40, 50. Therefore, a total of 75 combinations were simulated (5 conditions for N , 5 conditions for L , and 3 different true models: $5 * 5 * 3 = 75$). 500 samples were drawn per simulation.

For each condition of the simulation, a Basic Markov Model, a mixture Model with 2 latent classes and one with 3 latent classes was fitted on the dataset. AIC and BIC were separately used for model selection: that model was selected which had the lowest AIC or BIC. The results of this selection can be seen in Table 10.16. The fraction of correct model selections is written in boldface. Not all conditions are shown, yet the displayed results should show a clear trend.

BIC performs well for the true basic Markov model and the model with two latent classes. For all conditions with none or 2 latent classes, the number of correct model selections is 1.00, except for $N = 10$; $L = 10$. If 3 latent classes should be detected, the BIC needs a higher sample size. The correct selection rate is zero for all $N = 10$ conditions, and all originally planned $L = 10$ conditions. Even with $N = 50$ and $L = 50$, the correct selection rate was only .88. Therefore, four additional simulations were conducted with $N = 100$ and $L = 10$; 30; 50; 100. The increased sample size lifted the correct selection rate to .36 for the $L = 10$ condition. Therefore, the sample size seems to compensate for a low L . However, the correct selection rate for longer sequences did not improve further and even dropped for the $N = 100$ and $L = 100$ condition. At first glance, this might indicate that there is an upper limit to the selection rate.

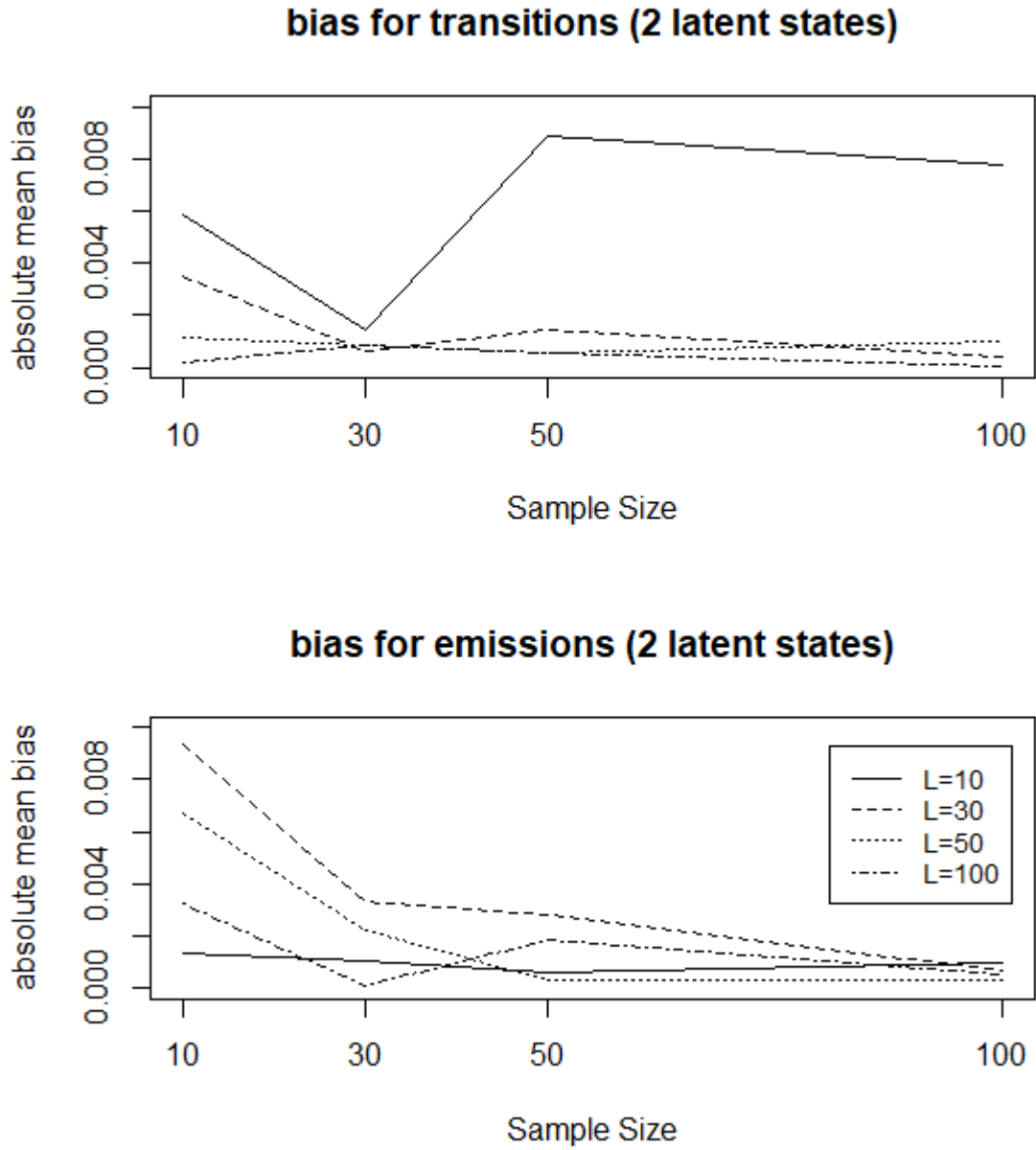
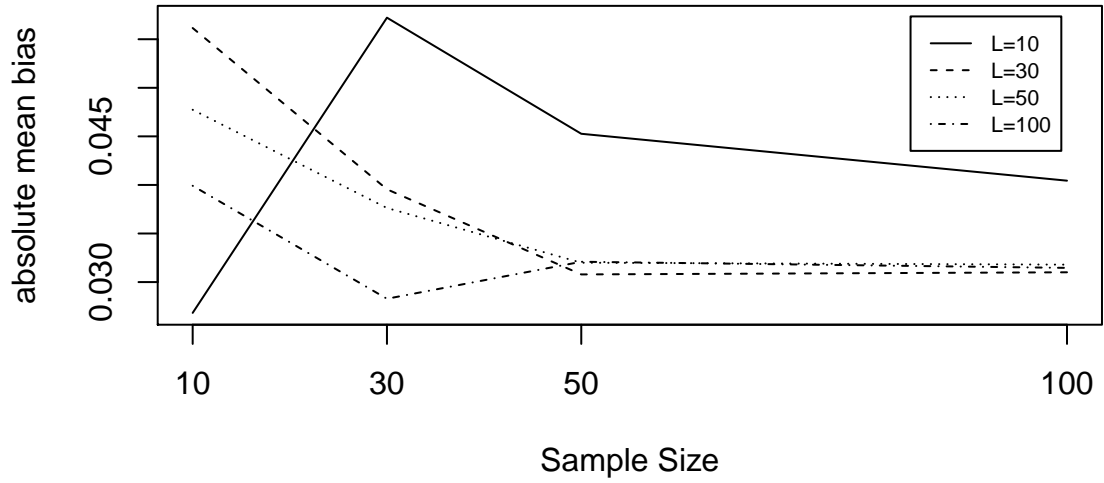


Figure 10.6.: Biases for the 2-latent-state Model

bias for transition probabilities (3-latent-states model)



bias for emissions (3-latent-states model)

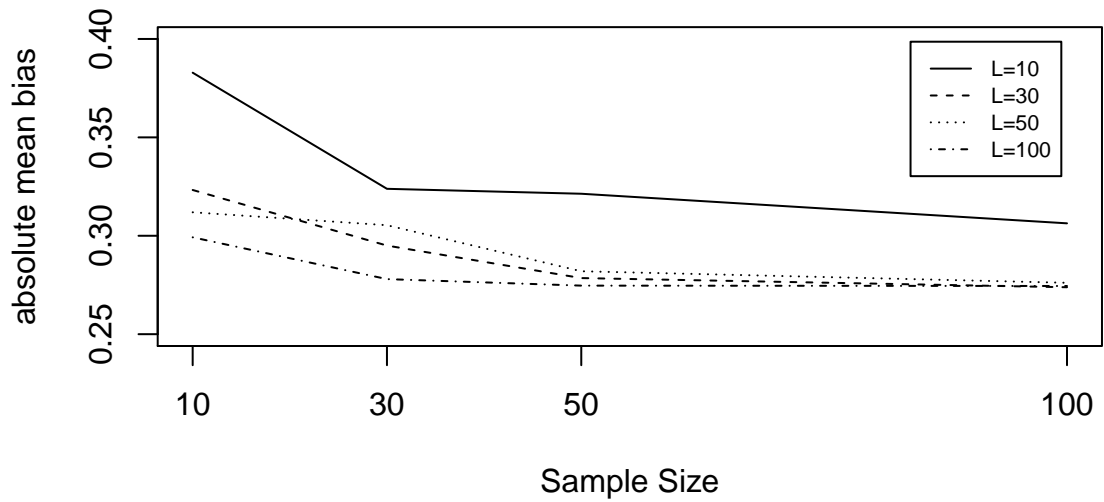


Figure 10.7.: Biases for the 3-latent-state Model

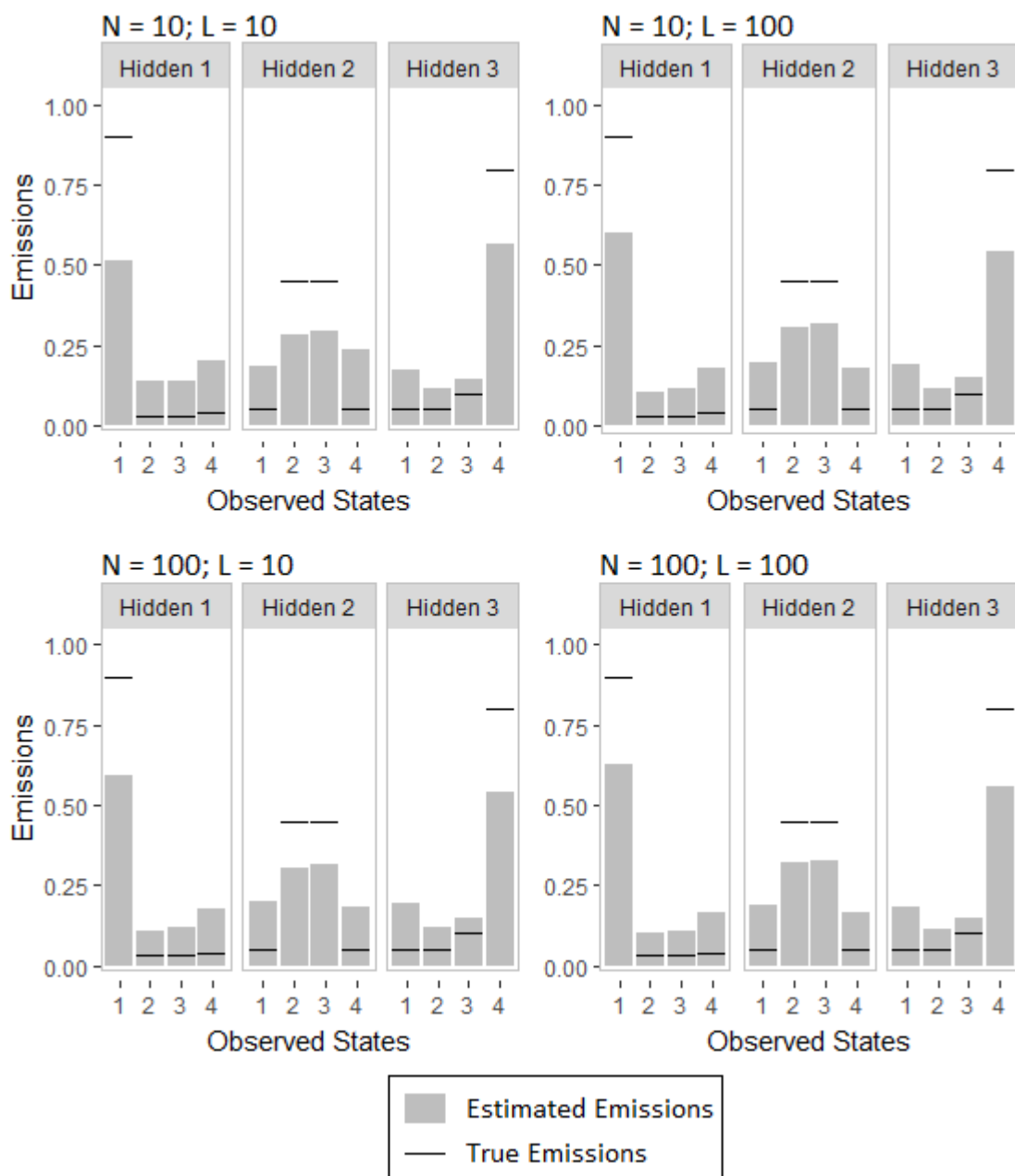


Figure 10.8.: Detailed Bias for Emissions (3 Latent States).

The modification to the transition probabilities of those post-hoc simulation were the following: for class 1 the row *State 1*→ was changed to .85, .05, .05, .05; for class 2 the row *State 3*→ was changed to .05, .05, .85, .05; and for class 1 the row *State 4*→ was changed to .05, .05, .05, .85.

The AIC behaves strangely for the basic Markov model and the 2 latent states model: on the one hand, the correct selection rate is relatively high with a rate of equal or above .95 for all conditions except for the basic Markov model with $N = 50$ and $L = 10$. On the other hand, the correct selection rate tends to decrease with higher values of N . This trend was the same for the post-hoc simulation, while the selection did not improve for conditions with low sample size or short sequences. Hence, BIC is recommended over AIC.

The Table 10.17 shows *bias*, *SE* for the transition probabilities in the first two columns for all simulation conditions. A modified calculation for avoiding bias induced by absorbing states was used for the model with 3 latent classes in column four and five. Cohen's κ was used for assessing how well observational units are assigned to their true class.

Because the order in which the latent classes are estimated is arbitrary, true latent classes and estimated classes had to be matched. This was done by using squared Euclidean distance. For example, for the model with two latent states, the Euclidean distance between the transition matrix of the first estimated class and the true first was calculated, then between the second estimated and the first true class. The estimated transition matrix with the lower distance was then matched with the first true class. As for the three latent class model, the procedure was similar: the distance between the transition matrix of the true first class was compared with all three estimated transition matrices regarding the squared Euclidean distance. The one with the lowest distance was matched with the first true class. The remaining two matrices were compared the matrix of the second true class, again the one with the closest distance was matched with the second true class. The last remaining class was matched with the true third class.

After that *Bias* and *SE* were computed across all transition probabilities assuming that the correct model was always selected. However, a close inspection of the biases revealed that the bias for the 3 latent class model was systematic in the way that all rows that belonged to an absorbing state had a stronger bias than the other. Therefore, those rows were excluded in the second calculation of *bias* and *SE* found in columns four and five. Cohen's κ is a measurement for rater agreement and, in this case, used for assessing how well sequences are classified regarding their true latent class.

Cohen's κ is basically the correct classification rate but corrected for the probability of correct classification by chance (see Equation 10.1). The benefit over the correct classification rate is that magnitude guidelines for Cohen's Kappa exist: according to (Landis and Koch, 1977), a κ of 0.41 – 0.60 can be seen as moderate, 0.61 - 0.80 as substantial, and 0.81 – 1 as almost perfect agreement. Sequences were assigned to the latent class with the highest posterior probability. Then κ was calculated between this posterior grouping and the true latent class.

$$\kappa = \left(\frac{P(DC_t)}{1 - P(DC_t)} \right) \quad (10.1)$$

The *SE* for the basic Markov seems to be relatively constant at about .08, dropping slightly with increasing sample size and sequence length to .07. There is a relatively strong bias for $N = 10$, which decreases with longer sequences. The bias gets acceptable small with $N = 50$ and $L = 30$ or with $N = 30$ and $L = 50$. Cohen's κ cannot be calculated because all sequences belong to the same class.

The model with 2 latent classes has a very similar *SE* as the basic Markov model, and the bias is even smaller across all conditions. Therefore, if only two latent classes are expected, sample sizes if $N = 30$ with $L = 10$ or $N = 10$ with $L = 30$ might be sufficient. Cohen's κ is very high. Thus, differing sequences between the two latent classes works very well.

The model with 3 latent classes has very high *bias* and *SE*. However, after excluding all transition probabilities with a true value of 0 or 1, the *bias* and *SE* was heavily reduced, indicating that the bias is mainly caused by the absorbing states. Aiming for a κ of at least .80 shows that, overall, $N = 30$ with $L = 30$ or more should be sufficient. However, for $N = 30$ with $L = 50$, κ is only .68. Therefore, $N > 50$ with $L > 50$ might be a safer recommendation.

The overall recommendation is to have at least $N = 50$ with $L = 50$ or greater and to use the BIC for model selection when three or less latent classes are expected. Moreover, advanced optimizers should be used instead of using only the EM-algorithm. Also, a researcher should keep in mind that transition rates might be biased if absorbing states exist.

However, this recommendation is limited by the following facts: 1) the simulation covers only the case of one sample of transition matrices per model. If transition matrices between latent classes vary only slightly, bigger sample sizes or longer sequences might be needed. 2) Only a maximum of three latent classes was tested. If more than

three latent classes are assumed, the sample size should be bigger. Each latent class splits the sequences into subgroups, which makes the estimation of transition probabilities for each latent class more difficult. Therefore, having at least 30 sequences in each latent class might be a good starting point for conducting further simulation studies on that topic. 3) A similar limitation is that only balanced datasets were tested. If the data is unbalanced, e.g., .75 belong to the latent class 1 and .25 to class 2, transition matrices are more difficult to estimate for the smaller group. Therefore, the sample size should be increased. For the given example, the sample size might be doubled so that the frequency in the smallest group fits the frequency that would be expected if the data was balanced. 4) All three simulated models have exactly four states. A higher number of states might need a bigger sample size and longer sequences because the more states exist, the less likely it might become to observe all possible transitions in a given dataset. 5) Performance of the models depended heavily on the chosen optimizers. Using other optimizers or other settings might increase performance further. In this simulation, optimizers were restricted in a way that they would not take longer than 5 Minutes for fitting a model. Preliminary simulations revealed that the EM-algorithm alone is a bad choice for an optimizer. Therefore, the same optimizers were used as in the hidden Markov simulation, which increased the percentage of correct model selection. Therefore these optimizers were used. However, the MSLS-LDS optimizer stopped quite often because it reached its time limit (60 seconds per default). The simulations took about a week per condition, and therefore, the second run with a higher time limit was not conducted.

10.9. OM-Distances

One, two and three cluster solutions were tested on the same true models that were used for the mixture Markov simulations, respectively the basic Markov model (see Table 10.10), the mixture Markov model with two latent classes (see Table 10.14), and the mixture Markov model with three latent classes (see Table 10.15).

Different strategies were tested for choosing the correct number of clusters. The standard strategy is to use the silhouette coefficient and choose the cluster solution with the best coefficient. However, if only one class exists, no silhouette coefficient will be computed. If a cluster achieves only a silhouette coefficient of below .25, it is often considered to be no *real* cluster (Struyf et al., 1997). Therefore, the following two rules were initially used: If the two- and the three-cluster solution achieved only

Table 10.14.: True Mixture Markov Model with 2 Latent Classes

Cluster	Class 1	Class 2			
Probabilities	.50	.50			
Initial State					
Probabilities	State 1	State 2	State 3	State 4	
Class 1	.25	.25	.25	.25	
Class 2	.25	.25	.25	.25	
Transitions: Class 1	→ State 1	→ State 2	→ State 3	→ State 4	
State 1 →	.05	.05	.05	.85	
State 2 →	.05	.05	.85	.05	
State 3 →	.05	.85	.05	.05	
State 4 →	.85	.05	.05	.05	
Transitions: Class 2	→ State 1	→ State 2	→ State 3	→ State 4	
State 1 →	.70	.20	.05	.05	
State 2 →	.05	.70	.20	.05	
State 3 →	.05	.05	.70	.20	
State 4 →	.01	.01	.01	.97	

Notes: left hand of → transition from; right hand of → transition to.

Table 10.15.: True Mixture Markov Model with 3 Latent Classes

Cluster	Class 1	Class 2	Class 3		
Probabilities	.34	.33	.33		
Initial State					
Probabilities	State 1	State 2	State 3	State 4	
Class 1	.25	.25	.25	.25	
Class 2	.25	.25	.25	.25	
Class 3	.25	.25	.25	.25	
Transitions: Class 1	→ State 1	→ State 2	→ State 3	→ State 4	
State 1 →	1.00	.00	.00	.00	
State 2 →	.20	.70	.05	.05	
State 3 →	.05	.25	.65	.05	
State 4 →	.05	.05	.25	.65	
Transitions: Class 2	→ State 1	→ State 2	→ State 3	→ State 4	
State 1 →	.10	.05	.05	.80	
State 2 →	.10	.05	.05	.80	
State 3 →	.00	.00	1.00	.00	
State 4 →	.10	.05	.05	.80	
Transitions: Class 3	→ State 1	→ State 2	→ State 3	→ State 4	
State 1 →	.70	.10	.10	.10	
State 2 →	.70	.00	.30	.00	
State 3 →	.70	.30	.00	.00	
State 4 →	.00	.00	.00	1.00	

Notes: left hand of → transition from; right hand of → transition to.

Table 10.16.: Model Selection for Mixture Markov Simulation

Simulation conditions	AIC			BIC		
	B	2S	3S	B	2S	3S
basic Markov; N=10; L=10	.99	.01	.00	1.00	.00	.00
basic Markov; N=10; L=30	.98	.02	.00	1.00	.00	.00
basic Markov; N=10; L=50	.97	.03	.00	1.00	.00	.00
basic Markov; N=30; L=10	.97	.03	.00	1.00	.00	.00
basic Markov; N=30; L=30	.96	.04	.00	1.00	.00	.00
basic Markov; N=30; L=50	.98	.02	.00	1.00	.00	.00
basic Markov; N=50; L=10	.91	.09	.00	1.00	.00	.00
basic Markov; N=50; L=30	.95	.05	.00	1.00	.00	.00
basic Markov; N=50; L=50	.98	.02	.00	1.00	.00	.00
2 latent classes; N=10; L=10	.00	1.00	.00	.83	.17	.00
2 latent classes; N=10; L=30	.00	.99	.01	.00	1.00	.00
2 latent classes; N=10; L=50	.00	.99	.01	.00	1.00	.00
2 latent classes; N=30; L=10	.00	1.00	.00	.00	1.00	.00
2 latent classes; N=30; L=30	.00	.97	.03	.00	1.00	.00
2 latent classes; N=30; L=50	.00	.99	.01	.00	1.00	.00
2 latent classes; N=50; L=10	.00	.98	.02	.00	1.00	.00
2 latent classes; N=50; L=30	.00	.97	.03	.00	1.00	.00
2 latent classes; N=50; L=50	.00	.98	.02	.00	1.00	.00
3 latent classes; N=10; L=10	.54	.46	.00	1.00	.00	.00
3 latent classes; N=10; L=30	.01	.59	.40	.90	.10	.00
3 latent classes; N=10; L=50	.00	.40	.59	.64	.36	.00
3 latent classes; N=30; L=10	.00	.79	.21	.94	.06	.00
3 latent classes; N=30; L=30	.00	.12	.88	.01	.67	.32
3 latent classes; N=30; L=50	.00	.08	.92	.00	.40	.60
3 latent classes; N=50; L=10	.00	.37	.63	.50	.50	.00
3 latent classes; N=50; L=30	.00	.07	.93	.00	.21	.79
3 latent classes; N=50; L=50	.00	.06	.94	.00	.12	.88
3 latent classes; N=100; L=10	.00	.11	.89	.00	.64	.36
3 latent classes; N=100; L=30	.00	.07	.93	.00	.14	.86
3 latent classes; N=100; L=50	.00	.12	.88	.00	.17	.83
3 latent classes; N=100; L=100	.00	.20	.80	.00	.27	.73

Notes: Simulation conditions show the true model, basic Markov model (Abbr. B; see Table 10.10), 2 latent classes (Abbr. S2; see Table 10.11) or 3 latent classes (Abbr. S3; see Table 10.12), N = samples size, L = length of sequences; 500 samples were drawn per simulation condition; Columns show classification based on AIC (Akaike information criterion) or BIC (Bayesian information criterion); a single dot indicates that this statistic was not computed.

Table 10.17.: Precisions of Estimates for Mixture Markov Simulation

Simulation conditions	incl. all States			excl. absorbing States	
	<i>Bias</i>	<i>SE</i>	κ	<i>Bias</i>	<i>SE</i>
basic Markov; N=10; L=10	.11	.08	.	.	.
basic Markov; N=10; L=30	.06	.08	.	.	.
basic Markov; N=10; L=50	.04	.08	.	.	.
basic Markov; N=30; L=10	.03	.08	.	.	.
basic Markov; N=30; L=30	.02	.08	.	.	.
basic Markov; N=30; L=50	.01	.07	.	.	.
basic Markov; N=50; L=10	.02	.08	.	.	.
basic Markov; N=50; L=30	.01	.07	.	.	.
basic Markov; N=50; L=50	.01	.07	.	.	.
2 latent classes; N=10; L=10	.02	.10	.98	.	.
2 latent classes; N=10; L=30	.00	.08	>.99	.	.
2 latent classes; N=10; L=50	.00	.08	>.99	.	.
2 latent classes; N=30; L=10	.00	.08	>.99	.	.
2 latent classes; N=30; L=30	.00	.08	>.99	.	.
2 latent classes; N=30; L=50	.00	.08	>.99	.	.
2 latent classes; N=50; L=10	.00	.08	>.99	.	.
2 latent classes; N=50; L=30	.00	.08	>.99	.	.
2 latent classes; N=50; L=50	.00	.08	>.99	.	.
3 latent classes; N=10; L=10	.33	.14	.43	.08	.12
3 latent classes; N=10; L=30	.24	.11	.82	.04	.09
3 latent classes; N=10; L=50	.22	.10	.89	.03	.09
3 latent classes; N=30; L=10	.28	.10	.58	.04	.09
3 latent classes; N=30; L=30	.26	.09	.90	.03	.07
3 latent classes; N=30; L=50	.23	.07	.93	.03	.47
3 latent classes; N=50; L=10	.25	.08	.92	.03	.07
3 latent classes; N=50; L=30	.31	.09	.68	.04	.08
3 latent classes; N=50; L=50	.25	.08	.92	.03	.07
3 latent classes; N=100; L=10	.30	.09	.79	.04	.08
3 latent classes; N=100; L=30	.25	.07	.93	.03	.06
3 latent classes; N=100; L=50	.17	.06	.91	.03	.05
3 latent classes; N=100; L=100	.21	.07	.86	.03	.06

Notes: Simulation conditions show the true model, basic Markov model (Abbr. B; see Table 10.10), 2 latent classes (Abbr. S2; see Table 10.11) or 3 latent classes (Abbr. S3; see Table 10.12), N=samples size, L=length of sequences; 500 samples were drawn per simulation condition; Columns show classification based on AIC (Akaike information criterion) or BIC (Bayesian information criterion).

Table 10.18.: Post-Hoc Simulations for 3 latent classes

Simulation conditions	Precision			Correct Model Selection	
	<i>Bias</i>	<i>SE</i>	κ	<i>AIC</i>	<i>BIC</i>
3 latent classes; N=10; L=30	.11	.10	.70	.48	.00
3 latent classes; N=10; L=50	.10	.09	.76	.62	.00
3 latent classes; N=10; L=100	.07	.09	.77	.57	.52
3 latent classes; N=30; L=30	.08	.09	.75	.59	.31
3 latent classes; N=30; L=50	.09	.09	.68	.42	.38
3 latent classes; N=30; L=100	.11	.09	.65	.34	.30
3 latent classes; N=50; L=10	.08	.09	.75	.59	.31
3 latent classes; N=50; L=30	.09	.09	.72	.46	.44
3 latent classes; N=50; L=50	.11	.09	.66	.38	.35
3 latent classes; N=100; L=30	.06	.09	.81	.71	.62
3 latent classes; N=100; L=50	.08	.09	.72	.53	.46
3 latent classes; N=100; L=100	.11	.09	.57	.50	.37
3 latent classes; N=300; L=30	.07	.10	.80	.75	.66
3 latent classes; N=300; L=50	.08	.09	.69	.57	.51
3 latent classes; N=300; L=100	.14	.09	.51	.50	.42

Notes: ; 100 samples were drawn per simulation condition; Columns show classification based on AIC (Akaike information criterion) or BIC (Bayesian information criterion).

an average silhouette coefficient of below .25. the one-cluster solution was chosen. Otherwise, the higher average silhouette coefficient was used. This rule performed badly when only one cluster exists (above .50 misclassification rate even with sample size = 100 and sequence length = 100).

Alternatively, the following rule was tested: choose the highest number of clusters where the minimum silhouette coefficient is above .25. This rules performed even worse. The next approach was to stick to the first rule but to increase the cut-off. The next benchmark value provided by Struyf et al. (1997) is .50 for a medium structure. Thus, the cut-off was set to .50. If the silhouette coefficient is below that, the cluster structure is considered to be weak. So the third rule was: If two and three cluster solution achieved only an average silhouette coefficient of below .50. the one-cluster solution was chosen. Otherwise, the higher average silhouette coefficient was used. This rule resulted in a heavy bias for the one-cluster solution. Finally, a cut-off of .40 was used, which resulted in overall consistent results and performed well for both cases in which only one true cluster existed. As for the conditions with two or three clusters, it performed still well, but only with increased sample size. The results can be seen in Table 10.19.

Dendrogram and scree plot must be interpreted by humans, and therefore, could not be tested directly. However, the author of this monograph tried to program an algorithm that mimics his intuition about scree plots. The algorithm performed badly. However, inspecting 500 random sample scree plot manually resulted also in only 15% correct classifications. Plotting all 1000 scree plots into an overlay plot (see Figure 10.9) revealed that the scree plot should not be used for model selection of sequence clusters. Small sample sizes ($N = 10$) with short sequences ($L = 10$) seem to indicate two classes most of the time when only one latent class exist, and vice versa, indicate a one-cluster solution when two latent classes exist. The scree plot seems to work better with big sample sizes ($N = 100$) and long sequences ($L = 100$). However, the scree plots that were generated by three true classes show a very ambiguous picture. There is a big drop from a one-class solution to a second class solution, yet only a small drop to a solution with three clusters. Hence, many researchers would choose two clusters; some might take three. The scree plot is not recommended as model selection tool based on this observation.

Overall, clustering using OM-distances performed worse than the mixture Markov model when samples sizes are small, and sequences are short. However, it performed better than the mixture Markov model when sample sizes are big, and sequences are long. Especially the length of sequences had a huge effect on the bias. Model selection

worked well (>80% correct model selection) for $N \geq 30$ with $L \geq 100$ or $N \geq 100$ with $L \geq 50$, when the silhouette coefficient was used. Mean absolute *bias* for these conditions was .01, and the *SE* was around .02 to .03. Cohen's κ was about .98 for these conditions. Unlike the mixture Markov model, no instabilities were detected for a model with three latent groups, even though the same true models were used for the simulation. Therefore, OM-clustering seems to be more robust than the mixture Markov model. Overall it is recommended to use OM-clustering when sample size and sequence length is $N \geq 30$ with $L \geq 100$ or $N \geq 100$ with $L \geq 50$. If a Cohen's κ of around .90, a bias of .03, and bigger *SE* of .04 is acceptable, OM-clustering can also be used for sample sizes of $N \geq 30$ with sequence lengths of $L \geq 50$.

The clustering with OM-distances has three advantages: 1) it is computationally fast, which might be a crucial factor if sequences should be grouped in real-time. However, within the field of science and research, the slower computation of mixture Markov model should not be a major drawback for the latter. 2) As discussed above, simulation studies indicate that OM-clustering seems to be more robust than mixture Markov models if sample sizes are big and sequences are long. 3) OM-clustering can be applied to datasets that contain sequences with different lengths. Differences between longer and shorter sequences can be weighted by adjusting the insertion and deletion cost.

Table 10.19.: Results for Simulation of OM-Distances (Part A)

Simulation conditions	Model Selection			Precision		
	B	2S	3S	<i>Bias</i>	<i>SE</i>	κ
basic Markov; N=10; L=10	.50	.40	.10	.10	.01	1.00
basic Markov; N=10; L=30	.79	.20	.01	.05	.00	1.00
basic Markov; N=10; L=50	.89	.10	.00	.03	.00	1.00
basic Markov; N=10; L=100	.97	.03	.00	.02	.00	1.00
basic Markov; N=30; L=10	.35	.62	.02	.03	.00	1.00
basic Markov; N=30; L=30	.74	.25	.01	.01	.00	1.00
basic Markov; N=30; L=50	.88	.12	.00	.01	.00	1.00
basic Markov; N=30; L=100	.95	.05	.00	.01	.00	1.00
basic Markov; N=50; L=10	.27	.72	.01	.02	.00	1.00
basic Markov; N=50; L=30	.80	.19	.01	.01	.00	1.00
basic Markov; N=50; L=50	.91	.09	.00	.01	.00	1.00
basic Markov; N=50; L=100	.97	.03	.00	.00	.00	1.00
basic Markov; N=100; L=10	.20	.80	.20	.01	.00	1.00
basic Markov; N=100; L=30	.80	.20	.01	.00	.00	1.00
basic Markov; N=100; L=50	.94	.06	.00	.00	.00	1.00
basic Markov; N=100; L=100	.98	.02	.00	.00	.00	1.00
2 latent classes; N=10; L=10	.46	.47	.07	.15	.10	.23
2 latent classes; N=10; L=30	.59	.37	.05	.09	.04	.57
2 latent classes; N=10; L=50	.58	.38	.05	.04	.02	.77
2 latent classes; N=10; L=100	.28	.70	.01	.01	.00	.95
2 latent classes; N=30; L=10	.20	.76	.04	.20	.08	.16
2 latent classes; N=30; L=30	.50	.48	.02	.11	.03	.55
2 latent classes; N=30; L=50	.37	.62	.01	.05	.01	.79
2 latent classes; N=30; L=100	.02	.98	.00	.01	.00	.96
2 latent classes; N=50; L=10	.12	.86	.01	.21	.08	.16
2 latent classes; N=50; L=30	.49	.50	.01	.11	.03	.55
2 latent classes; N=50; L=50	.28	.72	.00	.05	.01	.81
2 latent classes; N=50; L=100	.00	1.00	.00	.01	.00	.97
2 latent classes; N=100; L=10	.03	.97	.00	.23	.07	.15
2 latent classes; N=100; L=30	.42	.58	.00	.11	.03	.56
2 latent classes; N=100; L=50	.18	.82	.00	.05	.01	.85
2 latent classes; N=100; L=100	.00	1.00	.00	.01	.00	.98

Notes: Simulation conditions show the true model, basic Markov model (Abbr. B; see Table 10.10), 2 latent classes (Abbr. S2; see Table 10.11) or 3 latent classes (Abbr. S3; see Table 10.12), N=samples size, L=length of sequences; 500 samples were drawn per simulation condition; Columns show classification based on AIC (Akaike information criterion) or BIC (Bayesian information criterion).

Table 10.20.: Results for Simulation of OM-Distances (Part B)

Simulation conditions	Model Selection			Precision		
	B	2S	3S	<i>Bias</i>	<i>SE</i>	κ
3 latent classes; N=10; L=10	.40	.23	.37	.56	.16	.32
3 latent classes; N=10; L=30	.13	.18	.69	.16	.09	.65
3 latent classes; N=10; L=50	.05	.07	.88	.06	.07	.84
3 latent classes; N=10; L=100	.00	.00	.99	.01	.06	.96
3 latent classes; N=30; L=10	.12	.07	.82	.53	.13	.27
3 latent classes; N=30; L=30	.00	.01	.99	.13	.06	.66
3 latent classes; N=30; L=50	.00	.00	1.00	.03	.04	.89
3 latent classes; N=30; L=100	.00	.00	1.00	.00	.03	.98
3 latent classes; N=50; L=10	.07	.03	.90	.52	.12	.25
3 latent classes; N=50; L=30	.00	.00	1.00	.07	.05	.72
3 latent classes; N=50; L=50	.00	.00	1.00	.02	.03	.89
3 latent classes; N=50; L=100	.00	.00	1.00	.00	.03	.98
3 latent classes; N=100; L=10	.02	.00	.99	.48	.11	.25
3 latent classes; N=100; L=30	.00	.00	1.00	.06	.04	.74
3 latent classes; N=100; L=50	.00	.00	1.00	.01	.03	.89
3 latent classes; N=100; L=100	.00	.00	1.00	.00	.02	.98

Notes: Simulation conditions show the true model, basic Markov model (Abbr. B; see Table 10.10), 2 latent classes (Abbr. S2; see Table 10.11) or 3 latent classes (Abbr. S3; see Table 10.12), N = samples size, L = length of sequences; 500 samples were drawn per simulation condition; Columns show classification based on AIC (Akaike information criterion) or BIC (Bayesian information criterion).

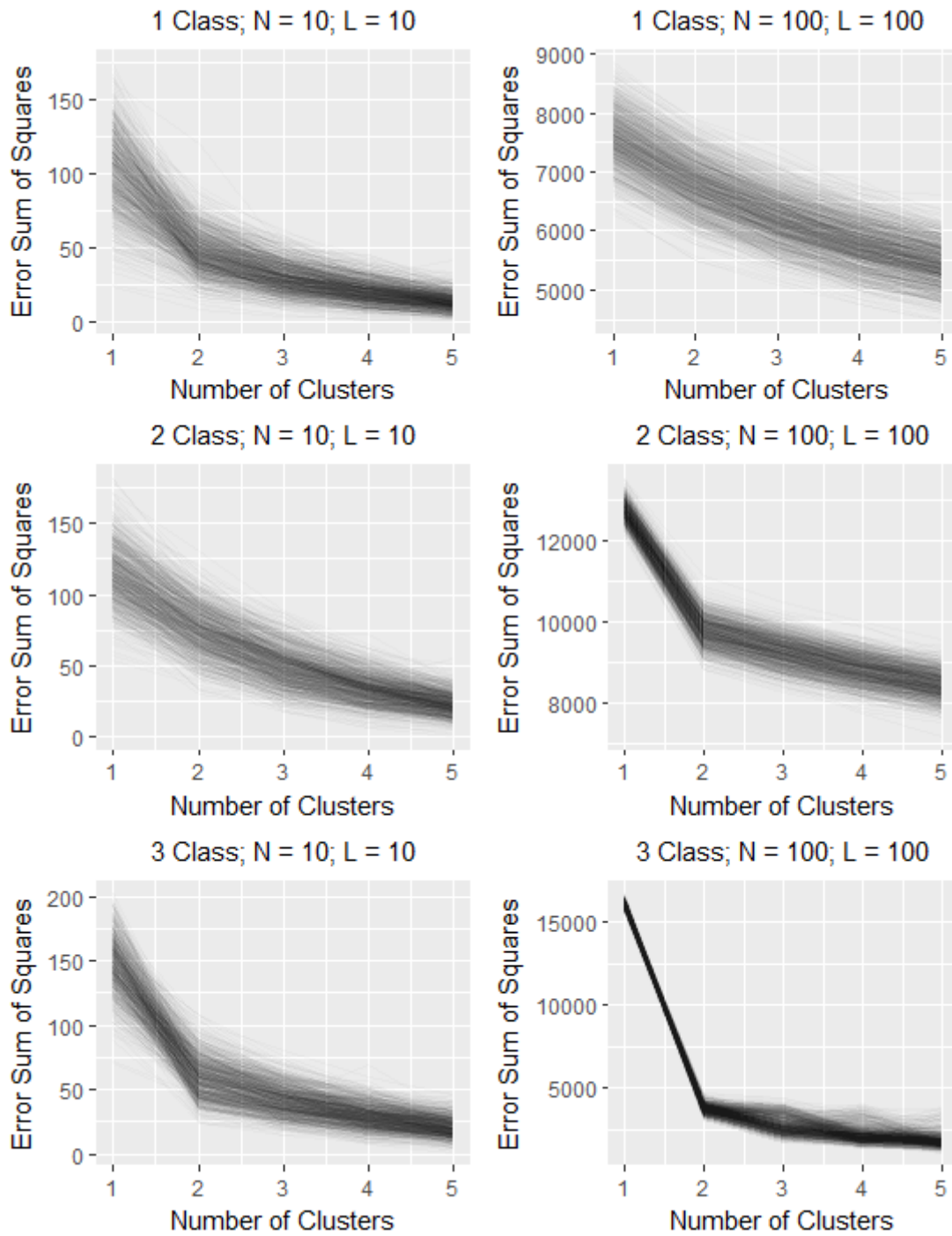


Figure 10.9.: Simulated scree plots. Each scree plot is one line, thus 1000 lines per simulation condition are plotted. Each line has a transparency of 98%, and thus, darker areas show higher concentrations scree plot lines.

10.10. ANOVA-like Approach for Comparing Subgroups

A simulation-based power analysis was conducted for the ANOVA-like approach using OM-distances for model comparison. Two and three groups were generated based on the same true models that were used for the mixture Markov simulations and the OM-clustering simulations (see Table 10.14) for two groups and Table 10.15 for three groups.

The following conditions were manipulated: N was iterated to be 10, 20, 30, 40, or 50; L was initially iterated to be 10, 20, 30, 40, or 50. However, after inspecting the results, $L = 5$ was added because power was already above .99 for $N = 10$ and $L = 40$, and thus the condition of $L = 50$ would not add further information. The results are displayed in Figure 10.10. The upper graph shows the power analysis for two groups, and the lower chart the results for three groups.

Using OM-distances for group comparisons achieves a power of above .80 for two-group comparisons and three-group comparisons if the sample size is $N = 10$ and sequence length is $L \geq 20$. If L is 10, N should be 20 for two-group comparisons and 30 three-group comparisons. Hence, a general rule might be 10 per group. If a power of above .95 is targeted L should be ≥ 30 or N should be ≥ 40 for two or three-group comparisons.

The results are limited by the fact that each group described fairly different transition probabilities. Smaller differences might be more difficult to detect. Therefore, future simulation studies should investigate how different measures of dissimilarity between transition matrices affect the power of this approach. However, a practical solution to this problem might be to run the simulation from Chapter B with other transition probabilities. Therefore, a researcher can try different expectations for the transition probabilities that match her or his research topic. Thereby, a rough estimation of the needed sample size can be obtained.

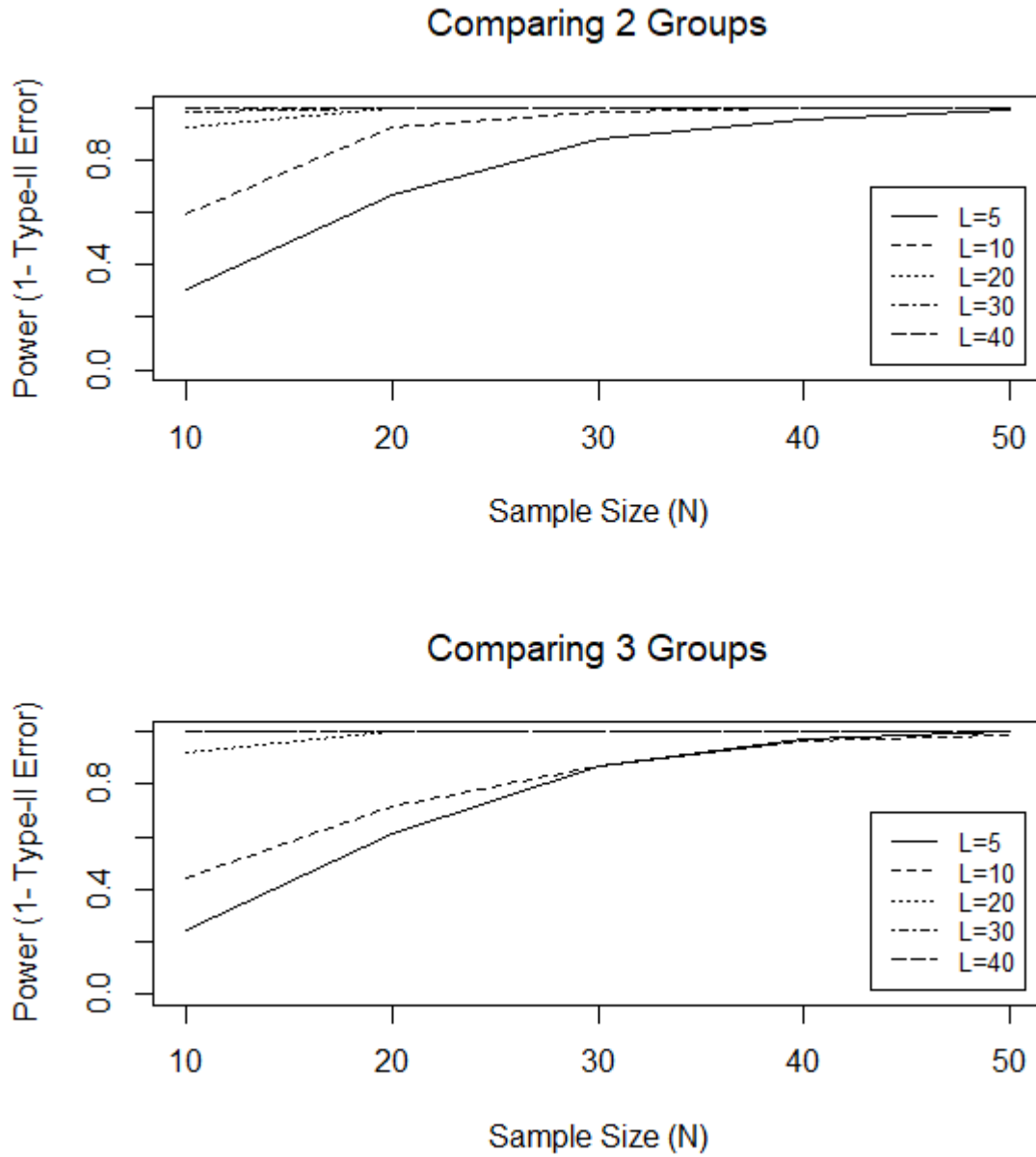


Figure 10.10.: Power Analysis for ANOVA-like Group Comparison Using OM-Distances.

11. Summary and Concluding Comments

This thesis aimed to explore possible statistical models and approaches that can answer typical research questions that might arise in psychological research. To this end, several statistical models and approaches have been presented for answering the research question of Chapter 3, some of which were not exclusive to dyadic sequence data alone. Chapter 4.1, for example, introduced the state-expand procedure that transforms two sequences into one and thereby allows for visualizations that are originally made for non-dyadic sequences. Using data visualization techniques and a diverse set of descriptive statistics allows getting an overview of a dataset of dyadic sequences, which answers the research question from Chapter 3.1 (*Getting an Overview About Dyadic Sequences*).

Moreover, it was shown that dyadic sequences could also be transformed into time-to-event data, especially if an absorbing state exists. By doing so, all kind of time-to-event models can be applied to sequence data (see Chapter 5.1). Time-to-event models ask the question "whether and when" does something happen (Singer and Willett, 2003)? Hence, they can give answers to the research question of Chapter 3.2 (*Duration of Behavior*). A special type of time-to-event models, the so-called frailty models (see Chapter 5.3), can also answer the questions of Chapter 3.3: *Is there a latent dyadic process?* Or, in other words, can the observed sequences of both couple members be explained by an unobserved latent process, a so-called common fate. However, they still estimate whether and when an event happens, or a behavior ends. Hence, they are limited to analyzing the latent process regarding the duration of behavior.

Alternatively, hidden Markov models can be used for modeling a latent process (see Chapter 8.3). However, hidden Markov models are more flexible: they can be restricted for estimating the duration of a latent state or behavior from observed indicators as was demonstrated in Chapter 8.3.1. However, other kinds of processes and dynamics are possible. Hidden Markov models can be applied to dyadic sequence data via the state-expand procedure which allows for correlation between indicators (observed states), or via the multi-channel approach (see Chapter 8.3.3).

Markov models that do not assume a latent process are referred to as basic Markov models and can be used for modeling an APIM, which answers the research question of Chapter 3.4 (*Analyzing Promptness of Interaction (APIM)*). An APIM is typically described in terms of actor and partner effects as shown in Chapter 1.2.4. However, Markov models do only provide transition probabilities. A conversion formula for transforming these probabilities into actor and partner effects was provided in Chapter 8.2.2.

Other models that were presented estimate actor and partner effects directly, namely the aggregated logit model (see Chapter 6)) and a multilevel model (see Chapter 7). Establishing the conversion formula allowed for comparing all three models with each other. Each model showed its own advantages and disadvantages. The aggregated logit model computes fast, the multilevel model produces very precise estimates, the basic Markov model also performs well, yet its estimates must be transformed, and *p*-values must be bootstrapped.

However, one advantage of the basic Markov model is that it can be extended to a mixture Markov model (see Chapter 8.4), which accounts for unobserved heterogeneity and thereby answers the final research question from Chapter 3.5 (*Are there latent groups or clusters?*). An alternative procedure for identifying latent groups, OM-clustering, was also introduced (see Chapter 9), which performed worse for smaller samples, but better for samples with long sequences and big datasets. Finally, an ANOVA-like approach was introduced for testing groups of sequences against each other (see Chapter 9.4.1).

Applying all these models and techniques showed how to apply and to interpret these models. Moreover, it gave inside into the *couples-cope* dataset. Additional simulation studies were conducted, which revealed strengths and weaknesses of the presented models (see Chapter 10). The code for running those simulations with other parameters is provided in Appendix B, and R vignettes for all presented applications are provided in Appendix A. All models that were not previously implemented in R (aggregated logit Model; multilevel APIM), and the conversion formula between basic Markov model and APIM is provided via the R package *DySeq*.

11.1. Findings on Dyadic Coping

Applying all the models mentioned above on *couples-cope* example revealed many insides. In the beginning, nearly all couples begin in a state of showing frequent stress communication and dyadic coping. This indicates that the experimental stress induc-

tion via the Trier Social Stress Test (TSST; Kirschbaum et al., 1993) was successful. Therefore, couples behave very similarly at the beginning of the observation period. However, over time, couples become more dissimilar, which might be caused by different dyadic coping skills or methods. Thus, sequence length and a long observational period seem to be important for detecting such differences. Unfortunately, 28.13% of the couples showed stress-related behavior until the end, and therefore a longer observation period might have given additional insights. Nevertheless, it was long enough to get other insights about the promptness of reactions, and possible latent structures. All three APIM models (aggregated logit, multilevel, and basic Markov) show that partner effects for dyadic coping responses are higher than for stress communication. This indicates that dyadic coping is a reaction to stress communication. The actor effect for stress communication is higher than for dyadic coping. Therefore, stress communication is a more stable behavior, than dyadic coping. That the induced stress leads to continued stress communication, which in response triggers dyadic coping responses. However, when stress communication ends, dyadic coping will also end. These findings are consistent with the works of Bodenmann (2005).

However, the best fitting Markov model was a Markov model featuring three latent states. The latent states could be interpreted as three stages of stress solving based on their emissions. The first stage, in which nearly all couples begin, show combined states of stress communication and dyadic coping responses. This might be caused by frequent stress communication and dyadic coping responses, therefore, they occur together within one time interval. The second stage is a state in which the communication slows down, so that time intervals with only stress communication or dyadic coping occur more often than in the first stage. Finally, they enter a state in which stress communication and dyadic coping do occur only sporadically. However, couples might relapse into a previous state of stress solving, yet according to the model's estimates, it is nearly impossible to relapse from the final stage back to the stage of high frequent stress communication and dyadic coping. This model performed a lot better without using the multi-channel approach. This indicates that occurrence of stress communication and dyadic coping cannot be explained solely by the latent process alone, rather than that, stress communication and dyadic coping correlate with each other even after controlling for a latent process. This correlation can be explained by the finding of the APIMs. Hence, the hidden Markov model does not discount the previous finding. Instead, it adds another component to the existing findings.

According to the Markov mixture model, couples can be separated into three different latent classes (slow, medium and fast copers). However, combining this mixture

model with the three latent states model was not possible. Therefore, this finding should be interpreted with cause. OM-Clustering does not assume to find *real* latent groups, even though it can be used to this end. However, it just groups sequences into groups of similar sequences. Thereby, it can always be used to simplify data without assuming true latent groups. It indicates only two clusters (*slow coper* and *fast coper*). Testing those two clusters against each other using the ANONA-like approach reveals that those clusters are significantly different from each other.

Separating the clusters shows an partner actor interaction effect with opposite signs for each cluster: If dyadic coping accompanies stress communication, the probability that the stress reaction will be kept up is less than expected by the main effects when the couples belong to the fast coper. For the slow coper, it is the opposite: If dyadic coping accompanies stress communication, it is more likely that it will be kept up. These findings indicate that at least two separate styles of dyadic coping might exist. Identifying the exact nature of these separate styles might be the subject of further research. However, possible explanations could be that, for instance, the slow coper encourages their partners to communicate their stress through active listening, whereas fast copers tend to appease their partners. Or, another possibility, stressed partners within the slow coping couples like to be comforted and keep their stress communication up so that their partners will keep up their dyadic coping.

11.2. Findings on Approaching Dyadic Sequence Data

What model should be used? - That depends on the research question, but for most research questions more than one model is possible. Therefore, one criterion can be the requirements for the data. Bigger samples, longer observation periods, and thinner time intervals will cost time and many. Therefore, a researcher might want to stick to models with low requirements. On the other hand, if a big dataset with long sequences already exists, the model that can utilize this data best should be used. Table X shows the research questions from Chapter 3, which models can be used to answering these questions, and the recommendations regarding sample size and length of sequences.

However, requirements on sample size and sequence length are not the only aspects that should be considered. Of all presented models, Markov models are the most versatile: The basic Markov model can be used to estimate an APIM. Therefore, it is an alternative to aggregated logit APIM and the multilevel APIM. However, a Markov chain's transition matrix can also be restricted in a way that it assumes an absorbing state. By doing so, it can be used to analyze event-to-time data, and thereby is an

alternative for the Cox regression. Moreover, using hidden Markov models with a restricted transition matrix analyze the duration of unobserved states. Thus, it is an alternative to frailty models. Moreover, unrestricted Markov models can be used for estimating common-fate models. Finally, mixture Markov models allow to account for unobserved latent groups, so it is an alternative for OM-clustering.

However, this flexibility comes at a price: estimation of complex Markov models can become challenging. Hidden Markov models with three latent states or three latent classes provided only unstable results in the simulation studies. Moreover, absorbing states as long as they are not anticipated as a restriction, seem to induce a heavy bias. Other optimizers might solve this problem. However, the most common optimizer, the EM-algorithm, provided even worse results. Hence, Markov models might be the most flexible model, but the biggest challenge is to find the correct specification for the model that should be fitted and to choose the correct optimizer. The model that provided the best "overall" default optimizers was the *seqHMM* package (Helske and Helske, 2016). However, it is recommended to give the global estimation step more time than the default 60 seconds.

The benefit of most other model is that they work *out of the box*: a researcher can run a Cox regression without worrying whether he is choosing the correct optimizer. Other model such as the aggregated logit APIM and the multilevel APIM needed a lot of data preparation, but this can now be handled by the *DySeq* Package.

There are more advantages to the specialized models. The Cox regression, for example, does not assume a certain distribution of the baseline hazard, whereas the latent hazard model (restricted hidden Markov) assumes a constant baseline model. Or another example, OM-clustering computes way faster than mixture Markov models are estimated, and the results for big datasets are more precise and more consistent than whose of the mixture Markov model. Therefore, OM-clustering is recommended over mixture Markov modeling, when sample sizes are big ($N > 100$ and $L > 100$), when more than two latent groups are expected, or when the aim is just data reduction.

Regarding the APIMs, the multilevel APIM and the basic Markov model performed similarly, yet both better than the aggregated logit model. The multilevel APIM allows to specify and to test random effects. Moreover, actor and partner effects are directly obtainable with *SE* and *p*-values. Therefore, it is recommended over the basic Markov model. However, if transition probabilities are of interest instead of actor and partner effects, the basic Markov APIM is recommended in such cases. The same goes for cases in which the APIM model should be compared to a common-fate like model, or other Markov models. The aggregated logit model may still be useful for really

big datasets, where the bias for this model is small, or in real-time analytics where processing speed might be more important than absolute bias.

However, one model was presented that performed so bad in the simulations that it is not recommended at all. This does not mean that frailty model is not useful overall, or that the used implementation of the shared frailty model is not useful. However, it should not be used, at least not in its current form, for dyadic sequence data. Other implementations or other frailty models might better with dyadic sequence data.

Finally, it is recommended to always plot dyadic sequence data before starting any analysis. Getting a general overview of the data first allows for plausibility-checks while interpreting later model results. For example, the mixture hidden Markov model was discarded because its results did not match what the data visualizations showed. This finding was supported by the fact that this model had a very bad model fit. However, data visualization adds a lot more to an analysis: It shows general trends like that stress-related communication often occurs in the beginning but occurs less often at the end of the observation period. It also showed for the couple-cope dataset that this trend might have kept on after the observation period. Moreover, anomalies can be detected such as the human player of the give-some dataset, who defected in all but three game turns. Finally, data visualization is a great way for communicating insights derived from the data analysis outside of the scientific community.

Table 11.1.: Summary of Sample Size and Sequence Length Recommendations

Research Question	Approach	Recommendations
Duration of Behavior	Cox regression	$N = 80$ for $\beta = 0.4$; $N = 80$ for $\beta = 0.6$; $N = 20$ for $\beta = 1$; $L > 30$ if <i>hazard</i> < 0.05 ; L must be long enough that the majority can transition into the absorbing state.
Common Fate	Hidden-Markov	For 2-3 hidden states: $N > 30$ for $L = 10$; $L > 30$ for $N = 10$; advanced optimizers, e.g., MSLS-LDS + BFGS, should be used.
Common Fate + Duration	Shared Frailty Model Latent-Hazard Model	not recommended. $N > 10$ & $L > 10$ for <i>hazard</i> = .05; $L > 50$ for <i>hazard</i> = .01 .
Analyzing Promptness of Interaction (APIM)	Aggregated Logit Model	$N > 100$ & $L > 100$; Delta must be adjusted if sequences are short (see Chapter 10.3).
	Multilevel Model	for $\beta = .02$ to 0.4: $L > 100$ for $N = 10$; $N > 100$ for $L = 10$; $N > 50$ if also $N > 50$; for $\beta > .08$: $N = 10$ for $L = 10$.
	Basic Markov Model	same as for multilevel model.
Latent Groups or Clusters (Unobserved Heterogeneity)	Mixture Markov Model	For 2 latent classes: $N > 30$ for $L = 10$; $L > 30$ for $N = 10$; For 2 latent classes: $N > 50$ and $L > 50$; advanced optimizers recommended; >2 classes might result in unstable estimates.
	OM-Clustering	For two latent classes: $N > 100$ for $L = 50$; $L > 100$ for $N = 30$; very robust; Scree Plot is not recommended.
	ANOVA-like Approach	long sequences ($L > 20$): $N = 10$ is sufficient for 2-3 groups; short sequences ($L = 10$): $n = 10$ per group.

11.3. Limitations and Outlook

A truly comprehensive collection and comparison of all models, which can be applied to dyadic sequence data, is beyond the scope of this monograph. This monograph tried to incorporate at least one model per class of models that can be applied to dyadic sequence data. The selection of these models was based on how useful this model is for psychology research and at the same time how prototypical it is for their type model class. One example is the shared frailty model that perfectly fitted one of the initial research questions and was a prototypical frailty model. However, it performed badly in the simulations, but the shared frailty model is only one frailty model out of hundreds, which might perform better. The same goes for multilevel models, that can be "tricked" into modeling dyadic sequence data in numerous ways. The shown modeling approach was used, because the recommended strategy by Kenny et al. (2006) could not be applied using the standard R packages for multilevel modeling, such as *lme4* (Bates et al., 2014). Hence, the possibility cannot be ruled out that other models or approaches might perform better.

Furthermore, it did not show how to add covariates to one of the APIMs or the Markov models, and how this might influence recommendations on sample size and the optimal length for sequences. Additionally, simulations studies were limited to a few selection simulation models with a maximum of three latent states or three latent classes. Therefore, recommendations for more complex latent structures with more classes, states or the combination of both, cannot be made. Also, many simulation samples were small due to limited processing power. However, advances in technology might increase future processing speed. Therefore, the R code in Appendix B might be used in the future for more precise simulations.

In the experience of this monographs author, the performance of optimizers is not often discussed in the community of psychology research. However, the simulations revealed a big impact on the performance of Markov models as soon as latent structure were introduced (mixture models or hidden states). This might be specific for Markov models, but future investigations of complex model's optimizers such as multilevel models or structural equation models might increase the ability of these models to find better solutions. Asparouhov and Muthen (2007) showed that this might be especially true for categorical variables. Helske and Helske (2016) added the feature to estimate Markov models in log-space, which should provide greater numerical stability at the cost of additional computation time. However, this features was not tested in this

monograph, yet future simulation studies might find that this approach might lead to better estimations.

As mentioned above, not all possible models for dyadic sequence data were presented, and it is impossible to show all. However, some models that were not covered are worth to mention because they allow for special applications. Latent Markov models (Bartolucci et al., 2012) are basically hidden Markov models that were designed for short sequences with only three to five time intervals, yet they need big sample sizes. Semi-Markov models are models that are similar to Markov model, yet their transition matrix depends on the amount of that that has passed by since the entry into the current state. Gabadinho et al. (2009) provided alternative distance measures for sequences, such as longest common prefix (LCP) or longest common subsequence (LCS) that can be used instead of OM-distances. Studer et al. (2011) also presented a way for running decision trees on sequence data using the ANOVA-like approach for group comparison via OM-distances.

A. Vignettes for R

This R-vignette provides a hands-on-tutorial for all analyses covered in this monograph. Please make sure to install all required packages, including the "DySeq"-Package, which provides the sample data. (Note: A growing collection of vignettes for DySeq is also available at Github: https://github.com/PeFox/DySeq_script.)

A.1. Prerequisite Steps

```
# The following packages are needed:
# "TraMineR", "RColorBrewer", "gmodels", "MASS", "survival", "fpc",
# "cluster"

# Install DySeq from CRAN:
# install.packages("DySeq")

# For an Overview of DySeq's abilities:
help(DySeq)
```

A.2. Getting and Visualizing the Data

```
# get the example data:
mydata<-CouplesCope # GiveSome for the GiveSome Example

# Variable labels and coding for the example data:
help(CouplesCope)

#####
## 5. GRAPHICAL ANALYSIS ##
#####
```

```
#####
# state-distribution plot (using TraMineR) #
#####

couple.labels <-c("time_interval", "SC_only", "DC_only", "SC+DC")

# create labels for plot
couple.seq <- seqdef(my.expand, labels = couple.labels)

# create a sequence object (the way TraMineR represents sequences)
seqdplot(couple.seq)

# Alternatively a grey version (using RColorBrewer)
attr(couple.seq, "cpal") <- brewer.pal(4, "Greys")
seqdplot(couple.seq, cex.legend=0.8, withlegend="right")

#####
# Entropy-plot and Histogramm of "Number of transitions" #
#####
# preparing the graphic device
# to show two graphics in a row
par (mfrow = c(1,2))
{
  # Entropy-plot:
  Entropy <- seqstatd(couple.seq)$Entropy
  plot(Entropy,
       main= "Entropy",
       col="black",
       xlab = "Time_in_10_sec._intervall",
       type = "l")

  # Histogramm of "Number of transitions":
  SeqL<-seqtransn(couple.seq)
  hist(SeqL,
       main="Number_of_transitions",
       xlab="State-transitions")
}
# setting the graphic device back:
par (mfrow = c(1,1))
```

```
# Number of transitions:
summary(SeqL)
```

A.3. Analyzing Duration

```
#####
## 6. ANALYSING DURATION ##
#####

## Objects from previous sections needed:
# - mydata

# Calculating the last occurrence of SC (last.stress)
last.stress<-LastOccur(mydata[,2:49],y=1)

#####
# Hazard, survival and cumhazard #
#####

# First a variable is needed that indicates if the event
# was shown (1; observed) or not (0; censored)

# Converting into a more time-to-event "friendly" format
event<-c(rep(1, length(last.stress)))
event[last.stress >=48]<-0

# number of ties per time interval:
table(last.stress[event==1])

# relative frequency of censored cases:
1-mean(event)

# The duration and the event variable are combined
# in a Surv-object
stress.surv<-Surv(last.stress, event)

# fitting the Cox regression
fit1 <- coxph(stress.surv~1, ties="efron")

# Preparing R's graphic device for three graphics
```

```

par (mfrow = c(1,3))

# Survival and cumhazard are already contained
# in the object "fit1"!
plot(survfit(fit1), conf.int="none", xlab="Time",
      ylab="Survival_Probability", xlim=c(0,48))

# Optional: if Median Lifetime should be added,
# run the following 9 lines:
# x <- 45; y<-seq(0, 0.5, by=0.01)
# x<-rep(x,length(y)); lines(x,y, lty=2)
# x<-seq(0, 45, by=0.1); y<-rep(0.5, length(x))
# lines(x,y, lty=2); x<-locator(1)
# After this line:
# click on the point within the Graphic where
# the ML-Label should be displayed
# text(x$x, x$y, "ML", cex=1.2)

plot(survfit(fit1), conf.int="none", xlab="Time",
      ylab="cumulated_hazard", fun="cumhaz")

# the DySeq package provides a function to compute
# the non-cumulated hazard:
NonCumHaz(survfit(fit1), plot=T) # Figure 5

# setting the graphics device back
par (mfrow = c(1,1))

```

A.3.1. Cox-Regression with Time-Independent Covariate

```

#####
## Cox regression: ##
## time-independent covariate ##
#####

# mean-centering EDCm (men's self-assessed
# dyadic coping ability)
EDCm.cent<-scale(mydata$EDCm, TRUE, FALSE)

```

```

# Fit the cox regression with
# EDCm.cent used as covariate
fit2 <- coxph(stress.surv~EDCm.cent, ties="efron")
summary(fit2)

# obtaining the exp.b for post-hoc calculations
exp.b<-unclass(summary(fit2))$coefficients[2]

# Plotting the Coxregression with simple hazards

# run:
plot(survfit(fit2)$time, NonCumHaz(survfit(fit2)),
      xlim=c(19,46), ylim=c(0,0.6), type="l",
      xlab="Time_Interval", ylab="Predicted_Hazard")

lines(survfit(fit2)$time, lty=2,
       NonCumHaz(survfit(fit2))*exp.b^5)

lines(survfit(fit2)$time, lty=3,
       NonCumHaz(survfit(fit2))*exp.b^-5)

# click into the graphics device to choose
# the location of the legend:
x<-locator(1)

legend(x$x, x$y, c("Cov_=_0", "Cov_=_+5", "Cov_=_-5"),
       lty=c(1,4,2), cex=0.7)

```

A.3.2. Cox-Regression with Time-Dependent Covariate

```

#####
## Cox-regression: ##
## time-dependent covariate ##
#####

```

Make sure the "TraMineRextras"-package is installed!

```
# install.packages("TraMineRextras")
```

```
library(TraMineRextras)
```

```
# The following objects are needed from the previous subsection:
```

```

# mydata
# last.stress

DC.seq <- seqdef(mydata[,50:97], labels = c("no_DC", "DC"))
DC.spell<- STS_to_SPELL(DC.seq, birthdate=rep(1, 64))
rowN<-length(DC.spell[,1])
event<-rep(0, rowN)
DC.spell<-cbind(DC.spell, event)

# Bringing dyadic coping responses into the SPELL-Format
DC.seq <- seqdef(mydata[,50:97], labels = c("no_DC", "DC"))
DC.spell<- STS_to_SPELL(DC.seq, birthdate=rep(1, 64))
rowN<-length(DC.spell[,1])
event<-rep(0, rowN)
DC.spell<-cbind(DC.spell, event)

#####
# New format is also SPELL-Format #
#####

for(i in 1:64){
  #i<-1

  pos<-which(DC.spell$id==i
             & DC.spell$begin<=last.stress[i]
             & DC.spell$end>=last.stress[i])

  x<-DC.spell[pos,]
  if(x$begin==x$end){
    x$event<-1
  } else {
    y<-x
    x$end<-last.stress[i]-1
    x$event<-0
    y$begin<-last.stress[i]
    y$event<-1
    DC.spell<-rbind(DC.spell, y)
  }
  DC.spell[pos,]<-x
}

```



```
pos<-which(DC.spell$id==i
           & DC.spell$begin>last.stress[i])
pos

if(length(pos)!=0) DC.spell[pos,]$event<-rep(1, length(pos))
}
DC.spell$end<-DC.spell$end+0.99

### Finally, the Cox-regression can be fitted:
fit_tp<-coxph(Surv(begin,end, event)~states, DC.spell)

## printing the summary:
summary(fit_tp)
```

A.3.3. Shared Frailty Model

```
# make sure that the frailtypack is installed:
# install.packages("frailtypack")
library(frailtypack)

# Transform SC and DC from sequences
# into time-to-event variables:
last.SC<-LastOccur(mydata[,2:49],y=1)
last.DC<-LastOccur(mydata[,50:97],y=1)

# putting them together into one vector:
My.time<-c(last.SC, last.DC)

# Censoring:
event<-c(rep(1, length(My.time))); event[My.time>=48]<-0

# creating ID-Variable
id<-rep(1:64, length(last.SC))

# Combining the time, the event and
# the id variables into one data.frame:
myFdata<-data.frame(My.time, event, id)

# Fitting the Frailty model
```

```
fitF0<-frailtyPenal(Surv(My.time,event)~cluster(id)+1,
                   data=myFdata, n.knots=24,
                   kappa=10000, RandDist="LogN")
```

A.4. Aggregated Logit Model

```
# The following packages are required:
# install.packages("DySeq")

# Transforming the sequences into combined sequences
#(state expand procedure)
my.states<-StateExpand(CouplesCope, # the data
                       2:49,        # first sequence
                       50:97)       # second sequence

# Create state-transition tables:
my.trans<-StateTrans(my.states, FALSE)

# Applies the Bakeman and Gottman approach
my.logseq<-LogSeq(my.trans)

# Getting the results
my.logseq

# Interaction plot
plot(my.logseq) # interaction can be plotted

# for single case analysis (41 refers to the row
# of the original data set! Not to the ID variable!)
single.LogSeq(my.logseq, 41)
```

A.5. Multilevel Model APIM

```
library("DySeq")
library("lme4")
library("lmerTest")

#####
# Data preparation #
#####
```

```

# Transforms sequences into multilevel data:
ML_data<-ML_Trans(CouplesCope, 2:49, 50:97)

# Transforms transitions into lagged actor and
# partner effects:
MLAP_data<-MLAP_Trans(ML_data)

# Multilevel models' output can become confusing quite fast.
# Therefore, labels should be used:
names(MLAP_data)[1]<-"sex"
MLAP_data$sex<-as.factor(MLAP_data$sex)
levels(MLAP_data$sex)<-c("female", "male")

# All variables are still dummy-coded.
# However, actor and partner effects should be
# effect coded for an easier interpretation.
MLAP_data$Partner[MLAP_data$Partner==0]<-(-1)
MLAP_data$Actor[MLAP_data$Actor==0]<-(-1)

#####
# Comparing Models #
#####

# The most complex model
set.seed(1234);
glmer(DV~1+sex+Actor+Partner+Actor*Partner+sex*Actor+sex*Partner
      +sex*Actor*Partner+(1+sex+Actor+Partner+Actor*Partner+
      sex*Actor+sex*Partner+sex*Actor*Partner|ID),
      data=MLAP_data, family=binomial)
# AIC 5256.744; BIC 5551.640
# Warning occurred: estimates are too close to boundaries!

# The most simple model (Random intercept only)
set.seed(1234) # AIC 5299.386; BIC 5359.706
glmer(DV~1+sex+Actor+Partner+Actor*Partner+sex*Actor+sex*Partner
      +sex*Actor*Partner+(1|ID), data=MLAP_data, family=binomial)

```

```

# Random effect for sex
set.seed(1234)
glmer(DV~1+sex+Actor+Partner+Actor*Partner+sex*Actor+sex*Partner
      +sex*Actor*Partner+(1+sex|ID), data=MLAP_data, family=binomial)
# AIC 5299.386; BIC 5375.723

# Random actor und partner effects
set.seed(1234)
glmer(DV~1+sex+Actor+Partner+Actor*Partner+sex*Actor+sex*Partner
      +sex*Actor*Partner+(1+Actor+Partner|ID), data=MLAP_data,
      family=binomial)
# AIC 5222.063; BIC 5315.893

# Random actor, partner, actor*partner effects
set.seed(1234)
glmer(DV~1+sex+Actor+Partner+Actor*Partner+sex*Actor+sex*Partner
      +sex*Actor*Partner+(1+Actor+Partner+Actor*Partner|ID),
      data=MLAP_data, family=binomial)
# AIC 5225.129; BIC 5345.768; plus tolerance warning

# Random actor, partner, actor*sex, partner*sex effects
set.seed(1234)
glmer(DV~1+sex+Actor+Partner+Actor*Partner+sex*Actor+sex*Partner
      +sex*Actor*Partner+(1+Actor+Partner+Actor*sex+Partner*sex|ID),
      data=MLAP_data, family=binomial)
# AIC 5237.065; BIC 5431.428; plus tolerance warning

# Best model fit is Random intercept + actor and partner effects
set.seed(1234)
fit<-glmer(DV~1+sex+Actor+Partner+Actor*Partner+
           sex*Actor+sex*Partner+sex*Actor*Partner+
           (1+Actor+Partner|ID),
           data=MLAP_data,
           family=binomial)
# AIC 5222.063
# BIC 5315.893

```

```

#Get estimates rounded:
round(fixef(fit)[1],2) #SC mean
round(fixef(fit)[3],2) #SC Actor
round(fixef(fit)[4],2) #SC Partner
round(fixef(fit)[5],2) #SC Actor*Partner
round(fixef(fit)[1]+fixef(fit)[2],2) #DC mean
round(fixef(fit)[3]+fixef(fit)[6],2) #DC Actor
round(fixef(fit)[4]+fixef(fit)[7],2) #DC Partner
round(fixef(fit)[5]+fixef(fit)[8],2) #DC Actor*Partner

summary(fit)

#####
##### DC as Reference-Category ###
#####

# Switching reference category:
MLAP_data2<-MLAP_data
contrasts(MLAP_data2$sex)[1]<-1
contrasts(MLAP_data2$sex)[2]<-0

set.seed(1234)
fit<-glmer(DV~1+sex+Actor+Partner+Actor*Partner+
           sex*Actor+sex*Partner+sex*Actor*Partner+
           (1+Actor+Partner|ID),
           data=MLAP_data2,
           family=binomial)
# AIC 5222.063
# BIC 5315.893

round(fixef(fit)[1],2);round(fixef(fit)[3],2)
round(fixef(fit)[4],2); round(fixef(fit)[5],2)
round(fixef(fit)[1]+fixef(fit)[2],2) #DC mean
round(fixef(fit)[3]+fixef(fit)[6],2) #DC Actor
round(fixef(fit)[4]+fixef(fit)[7],2) #DC Partner
round(fixef(fit)[5]+fixef(fit)[8],2) #DC Actor*Partner
summary(fit)

```

A.6. Markov Models

A.6.1. Basic Markov Model (MM)

```
library(depmixS4)
library(DySeq)
library(seqHMM)
library(TraMineR)

# Get the Example Data
mydata<-CouplesCope

# State Expand:
my.expand<-StateExpand(CouplesCope, 2:49, 50:97)

# Transform the Sequences into a stslst-object
my_seq<-seqdef(my.expand[,1:48],
               start = 1, labels = c("no_SC/DC",
                                     "SC_only", "DC_only", "SC+DC"))

# The fast way:
seqtrate(my_seq)

# As hidden Markov model:
# Start values
sc_init <- c(.25,.25,.25,.25) # initial state distribution

sc_trans <- matrix(
  c(.25),
  nrow = 4, ncol = 4, byrow = TRUE) # transition matrix.

sc_emiss <- matrix(c(1,0,0,0,
                    0,1,0,0,
                    0,0,1,0,
                    0,0,0,1), nrow = 4, ncol = 4) # emission matrix
```

```
# Now we put everything together into one model:
sc_initmod <- build_hmm(observations = my_seq,
                        initial_probs = sc_init,
                        transition_probs = sc_trans,
                        emission_probs = sc_emiss)
```

```
# Fit the Model
sc_fit <- fit_model(sc_initmod, global_step=TRUE,
                   local_step=TRUE)
sc_fit$model # Inspect the Model
```

```
# Get AIC and BIC
BIC(sc_fit$model); AIC(sc_fit$model)
```

A.6.2. Restricted Hidden Markov Model

```
# 2 Latent States
```

```
sc_init <- c(1,0) # Restriction: all couples in state 1
```

```
sc_trans <- matrix(
  c(0.50, 0.50, # The probability to stay or leave stress can vary
    0, 1), # The second state is absorbing (restricted to 1)
  nrow = 2, ncol = 2, byrow = TRUE)
```

```
# The transition Matrix is a 2*4 Matrix now,
sc_emiss <- matrix(0.25, nrow = 2, ncol = 4)
```

```
sc_initmod <- build_hmm(observations = my_seq, initial_probs = sc_init,
                        transition_probs = sc_trans,
                        emission_probs = sc_emiss)
```

```
# Use EM-Algorithm 1000 times for optimal starting values, then
# global and local optimizer are used:
```

```
sc_fit <- fit_model(sc_initmod,
                   global_step=TRUE, local_step=TRUE,
                   control_em=list(restart = list(times=1000)))
AIC(sc_fit$model); BIC(sc_fit$model)
```

A.6.3. Unrestricted HMM (common fate)

```
#####  
# 2 Latent States #  
#####  
  
sc_init <- c(.5, .5)  
  
sc_trans <- matrix(  
  c(0.50, 0.50,  
    0.50, 0.50),  
  nrow = 2, ncol = 2, byrow = TRUE)  
  
sc_emiss <- matrix(0.25, nrow = 2, ncol = 4)  
  
sc_initmod <- build_hmm(observations=my_seq, initial_probs=sc_init,  
  transition_probs = sc_trans,  
  emission_probs = sc_emiss)  
  
sc_fit <- fit_model(sc_initmod,  
  global_step=TRUE, local_step=TRUE,  
  control_em=list(restart = list(times=1000)))  
  
AIC(sc_fit$model); BIC(sc_fit$model)  
  
#####  
# 3 Latent States #  
#####  
  
sc_init <- c(.33, .34, .33)  
  
sc_trans <- matrix(  
  c(0.33, 0.33, 0.34,  
    0.33, 0.33, 0.34,  
    0.33, 0.33, 0.34),  
  nrow = 3, ncol = 3, byrow = TRUE)  
  
sc_emiss <- matrix(0.25, nrow = 3, ncol = 4)
```



```

sc_initmod <- build_hmm(observations=my_seq, initial_probs=sc_init,
                        transition_probs = sc_trans,
                        emission_probs = sc_emiss)

sc_fit <- fit_model(sc_initmod,
                   global_step=TRUE, local_step=TRUE,
                   control_em=list(restart = list(times=1000)))

AIC(sc_fit$model); BIC(sc_fit$model)

#####
# 4 Latent States #
#####

sc_init <- c(.25, .25,.25,.25)

sc_trans <- matrix(
  c(0.25, 0.25, 0.25, 0.25,
    0.25, 0.25, 0.25, 0.25,
    0.25, 0.25, 0.25, 0.25,
    0.25, 0.25, 0.25, 0.25),
  nrow = 4, ncol = 4, byrow = TRUE)

sc_emiss <- matrix(0.25, nrow = 4, ncol = 4)

sc_initmod <- build_hmm(observations = my_seq,
                        initial_probs = sc_init,
                        transition_probs = sc_trans,
                        emission_probs = sc_emiss)

sc_fit <- fit_model(sc_initmod,
                   global_step=TRUE, local_step=TRUE
                   control_em=list(restart = list(times=1000)))

sc_fit$logLik

BIC(sc_fit$model); AIC(sc_fit$model)

```

A.6.4. Multi-Channel Approach ("Pure Common Fate")

```
#####
# 3 Latent States (Multi-Channel #
#####
my_seq<-couple.seq
sc_init <- c(.33, .34, .33)

sc_trans <- matrix(
  c(0.33, 0.33, 0.34,
    0.33, 0.33, 0.34,
    0.33, 0.33, 0.34),
  nrow = 3, ncol = 3, byrow = TRUE)

sc_emiss1 <- matrix(0.5, nrow = 3, ncol = 2)
sc_emiss2 <- matrix(0.5, nrow = 3, ncol = 2)
sc_emiss <- list(sc_emiss1, sc_emiss2)

DC_seq <- seqdef(CouplesCope[,2:49],
  start = 1, labels = c("no_DC", "DC"))

SC_seq <- seqdef(CouplesCope[,50:97],
  start = 1, labels = c("no_SC", "SC"))

sc_obs <- list(SC_seq, DC_seq)

sc_initmod <- build_hmm(observations = sc_obs,
  initial_probs = sc_init,
  transition_probs = sc_trans,
  emission_probs = sc_emiss,
  channel_names = c("SC", "DC"))

sc_fit <- fit_model(sc_initmod,
  global_step=TRUE,
  local_step=TRUE,
  control_em=list(restart = list(times=10)))
AIC(sc_fit$model);BIC(sc_fit$model)
```

A.7. Mixture Markov

```

# Specify starting values for transition matrices of
# both latent groups
mytrans1<-matrix(c(.25), 4,4); mytrans2<-matrix(c(.25), 4,4)

# The transition matrices must be placed into a list ,
# before building the model:
mymixtrans<-list(mytrans1, mytrans2)

# Starting values for the initial probabilities , one for each chain:
myinit1<-c(.25, .25, .25, .25); myinit2<-c(.25, .25, .25, .25)
# again both must be placed inside a list:
mymixinit<-list(myinit1, myinit2)

my_mmodel<-build_mmm(couple.seq, # the data
                    mymixtrans, # starting values: transitions
                    mymixinit) # starting values: initial states

fit3<-fit_model(my_mmodel, global_step=TRUE, local_step=TRUE,
               control_em=list(restart = list(times=10)))
AIC(fit3);BIC(AIC)

```

A.8. OM-Distances

```

#####
## OM-Distances ##
#####

# Substitution-cost-matrix (Gabadinhos TRATE-Formula)
submat <- seqsubm(couple.seq, method = "TRATE")

# distance-matrix:
dist.oml <- seqdist(couple.seq, method = "OM", sm = submat)

#####
## Determine optimal Number of clusters ##
#####

```

```

# optimal number of clusters:
plot(pam(dist.oml, pamk(dist.oml)$nc), which.plot=1)

# Screeplot: (Indicates 1 or 2 clusters)
wss <- (nrow(dist.oml)-1)*sum(apply(mydata,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(mydata, centers=i)$withinss)
plot(1:15, wss, type="b",
     xlab="Number_of_Clusters",
     ylab="Within_groups_sum_of_squares")

# Dendrogram: (Indicates 2, maybe 3, Clustersolution)
plot(agnes (dist.oml, diss=TRUE, method = "ward"), which.plots=2)

#####
## Ward-algorithm ##
#####

clusterward1 <- agnes (dist.oml, diss=TRUE, method = "ward")

# 2 Clustersolution
cluster2 <- cutree (clusterward1, k=2)
cluster2fac <- factor (cluster2, labels = c("cluster_1;
.....fast_coper", "cluster_2;_slow_coper"))

#####
### comparing clusters and further analyses ###
#####

## separate state-distribution plots
seqdplot (couple.seq, group = cluster2fac)

## correlation between the cluster membership
# and men's self-assessed dyadic coping ability
clust2.dummy<-as.numeric(cluster2fac)
clust2.dummy[clust2.dummy==1]<-0
clust2.dummy[clust2.dummy==2]<-1
cor.test(mydata$EDCm, clust2.dummy)

LogSeq(my.trans, delta=0.5, subgroups=cluster2)

```

```
LogSeq(my.trans.SC, delta=0.5, subgroups=cluster2)

# ANOVA-like approach
da <- dissassoc(dist.oml, group = cluster2, R = 5000)
print(da$stat)
```

B. R-Code for Simulation Studies

All scripts are compressed: Replace ";" with linebreaks for better readability. Each Script ends in a loop, indicated by hashbox with the title "Simulation". Conditions have the same name as in Chapter 10.

B.1. Cox-Regression:PowerAnalysis

```
# Simulate Sequence
simSeq<-function(MyHaz,k){
MySeq<-0
W100<-seq(0,1,by=0.01)
for(i in 1:k){
if(MySeq[i]==0){
if(sample(W100,1)<=MyHaz){
MySeq<-c(MySeq,1)
} else {
MySeq<-c(MySeq,0)
}
} else {
MySeq<-c(MySeq,1)
}
}
return(MySeq)
}

# Simulating datasets
SimCoxDat<-function(N,Basehaz,B,k){
MyB<-B;cov0<-rnorm(N,0,1)
IndHaz<-Basehaz*exp(MyB)^cov0
MyDat<-data.frame()
for(i in IndHaz){
MyDat<-rbind(MyDat,simSeq(i,k))
}
colnames(MyDat)<-paste("T",0:k,sep="")
out<-cbind(IndHaz,cov0,MyDat);return(out)
}
```

```

}
#Transforming into event to time
library(DySeq); library(survival)
K<-1000#number of simulations; k<-30#sequences length
N<-30#sample size; Basehaz<-0.05#baseline hazard
B<-0#effect

SimCox<-function(N, Basehaz, B, k, K, plot=FALSE,
  digits=3, raw=FALSE){
  require(DySeq); require(survival)
  out<-list(); pb<-txtProgressBar(min=0, max=K, initial=0,
  char="=", width=40, title=,
  label, style=3, file="")
  for(i in 1:K){
    setTxtProgressBar(pb, i, title=NULL, label=NULL)
    x<-SimCoxDat(N, Basehaz, B, k)#####
    E<-LastOccur(x[, 3:length(x[, 1])], y=0)
    Event<-rep(1, N); Event[E==k+1]<-0
    out[[i]]<-coxph(Surv(E, Event)~cov0, data=x)
  }
  myCoeff<-c()
  for(i in 1:length(out)){
    myCoeff[i]<-summary(out[[i]])$coefficients[1]
  }
  bias<-mean(myCoeff); se<-sd(myCoeff)
  if(plot==TRUE){ hist(myCoeff)}
  myP<-c()
  for(i in 1:length(out)){
    myP[i]<-summary(out[[i]])$coefficients[5]
  }
  p.value<-mean(myP<.05)
  if(raw==TRUE){
    out2<-out
  } else {
    out2<-c(bias, se, p.value)
    names(out2)<-c("bias", "se", "sign")
    round(out2, digits)
  }
}

```

```
#####
#Simulationconditions#
#####

#SimuliereN10,20,30,40,50,60,70,80,90,100,200
#FuerB=0.2,B=0.4,B=0.6,B=0.8,B=1,B=2
#PrepareListforcompleteSimulation
simT<-list()
#Preparedata.frameforPoweranalysis-results
results<-data.frame();count<-1;out<-c()
K<-1000;k<-30;Basehaz=0.05

#SEEDis643952
set.seed(643952)

for(N in c(seq(10,100,by=10),200,500)){
out<-c()
for(B in c(seq(0,1,by=0.2),2)){

MZ<-SimCox(N,Basehaz,B,k,K,plot=FALSE,
digits=3,raw=FALSE)

simT[[count]]<-MZ;out<-c(out,MZ[3])

cat(paste("\nSimulation",count,"outof84:\n"))
count<-count+1
}
results<-rbind(results,out);print(results)
}

colnames(results)<-paste("B=",
c(seq(0,1,by=0.2),2),sep="")

rownames(results)<-paste("N=",c(seq(10,100,by=10),
,200,500),sep="");simT[seq(7,84,by=7)]

#####
##Follow-UpSimulations##
#####
```



```
#SEEDis281217
set.seed(281217)

MyLSim<-list();count2<-1
for(BASE in c(0.01,0.1,0.25)){
  for(MyL in c(10,50,100)){
    MyLSim[[count2]]<-SimCox(N=10,
    Basehaz=BASE,B=2,k=MyL,K=10000,
    plot=FALSE,digits=3,raw=FALSE)
    print(count2);count2<-count2+1
  }
}

SimCox(N=10,Basehaz=BASE,B=,k=MyL,K=10000,
plot=FALSE,digits=3,raw=FALSE)
```

B.2. Frailty Model

```
library(DySeq);library(frailtypack)
FRAIL<-c();Ps<-c()

#simulatingfrailty
set.seed(132870)

###Conditions:
SIM<-100;Est<-0;NonCov<-0;Rej<-0
Fresult<-data.frame(Est,NonCov,Rej)
counter<-1
for(N in c(20,40,60,80,100)){
  for(SIG in c(0,0.25,0.5,0.75,1)){
    for(k in 1:SIM){
      print(c(N,SIG,k))
      if(SIG==0){
        MyFrail<-rep(0,N)
      }else{
        MyFrail<-rnorm(N,0,SIG)
      }
      FHaz<-0.05*exp(MyFrail)
      #Preparingdata.frames
      myDat1<-data.frame()
      myDat1<-rbind(myDat1,simSeq(FHaz[1],30))
    }
  }
}
```

```

myDat2<-data.frame()
myDat2<-rbind(myDat2,simSeq(FHaz[1],30))
for(i in FHaz[2:N]){#Generating Samples
myDat1<-rbind(myDat2,simSeq(i,30))
myDat2<-rbind(myDat2,simSeq(i,30))
}
#Transforming intotime-to-event variables
last1<-LastOccur(myDat1,y=0)
last2<-LastOccur(myDat2,y=0)
My.time<-c(last1,last2)

event<-c(rep(1,length(My.time)))
event[My.time>=31]<-0

id<-rep(1:N,times=2)

myFdata<-data.frame(My.time,event,id)

fitF0<-NULL
#Fitting frailty Model
fitF0<-tryCatch(frailtyPenal(Surv(My.time,event)~
cluster(id)+1,data=myFdata,
n.knots=6,kappa=1000,
RandDist="LogN"),error=function(e)NULL)

#Calculating frailty parameter and p-value
if(!is.null(fitF0)){seH<-sqrt(fitF0$varTheta[1])
frail<-fitF0$sigma2
MyPValue<-signif(1-pnorm(frail/seH))
FRAIL[k]<-frail;Ps[k]<-MyPValue
}
}
Fresult[counter,]<-c(mean(FRAIL^0.5,na.rm=TRUE),
mean(is.na(FRAIL)),
mean(Ps<.05,na.rm=TRUE))
counter<-counter+1;FRAIL<-c();Ps<-c()
}
}
round(Fresult,3)

```

B.3. Power Simulation for Aggregated Logit Models

```
#####
##CreatingSequences##
#####
#Functionforgeneratingsequences
simSeq<-function(trans , initial , length){
  init<-sample(c(1,2,3,4),1 , prob=initial)
  for(i in 2:length){
    if (init[i-1]==1){
      init[i]<-sample(c(1,2,3,4),1 , prob=trans [1 ,])
    } else if (init[i-1]==2){
      init[i]<-sample(c(1,2,3,4),1 , prob=trans [2 ,])
    } else if (init[i-1]==3){
      init[i]<-sample(c(1,2,3,4),1 , prob=trans [3 ,])
    } else {
      init[i]<-sample(c(1,2,3,4),1 , prob=trans [4 ,])
    }
  }
  return (init)
}
#####
#Simulateacompletesampleofdyadicsequences#
#####
simSeqSample<-function(trans , initial , length ,N){
  comb<-simSeq(trans , initial , length)
  seq1<-c (); seq1 [comb==1 | comb==3]<-0
  seq1 [comb==2 | comb==4]<-1; seq2<-c ()
  seq2 [comb==1 | comb==2]<-0; seq2 [comb==3 | comb==4]<-1
  out<-c(seq1 , seq2); for (i in 2:N){
    comb<-simSeq(trans , initial , length)
    seq1 [comb==1 | comb==3]<-0; seq1 [comb==2 | comb==4]<-1
    seq2 [comb==1 | comb==2]<-0; seq2 [comb==3 | comb==4]<-1
    out<-rbind(out , c(seq1 , seq2))
  }; return (out);}
#####
#TransformAPIMLogitsintoTransitionMatrix#
#####
```

```

APIMtoTrans<-function(B0_1,AE_1,PE_1,Int_1,
B0_2,AE_2,PE_2,Int_2){
myTrans<-matrix(NA,4,4)
odds1<-exp(B0_1)*exp(AE_2)^(-1)*exp(PE_2)^(-1)*exp(Int_2)^(1)
prob1<-odds1/(odds1+1)
odds2<-exp(B0_2)*exp(AE_1)^(-1)*exp(PE_1)^(-1)*exp(Int_1)^(1)
prob2<-odds2/(odds2+1)
myTrans[1,1]<-(1-prob1)*(1-prob2)
myTrans[1,2]<-(1-prob1)*prob2
myTrans[1,3]<-prob1*(1-prob2);myTrans[1,4]<-prob1*prob2
odds1<-exp(B0_1)*exp(AE_2)^(-1)*exp(PE_2)^(1)*exp(Int_2)^(-1)
prob1<-odds1/(odds1+1)
odds2<-exp(B0_2)*exp(AE_1)^(1)*exp(PE_1)^(-1)*exp(Int_1)^(-1)
prob2<-odds2/(odds2+1)
myTrans[2,1]<-(1-prob1)*(1-prob2);myTrans[2,2]<-(1-prob1)*prob2
myTrans[2,3]<-prob1*(1-prob2);myTrans[2,4]<-prob1*prob2
odds1<-exp(B0_1)*exp(AE_2)^(1)*exp(PE_2)^(-1)*exp(Int_2)^(-1)
prob1<-odds1/(odds1+1)
odds2<-exp(B0_2)*exp(AE_1)^(-1)*exp(PE_1)^(1)*exp(Int_1)^(-1)
prob2<-odds2/(odds2+1)
myTrans[3,1]<-(1-prob1)*(1-prob2);myTrans[3,2]<-(1-prob1)*prob2
myTrans[3,3]<-prob1*(1-prob2);myTrans[3,4]<-prob1*prob2
odds1<-exp(B0_1)*exp(AE_2)^(1)*exp(PE_2)^(1)*exp(Int_2)^(1)
prob1<-odds1/(odds1+1)
odds2<-exp(B0_2)*exp(AE_1)^(1)*exp(PE_1)^(1)*exp(Int_1)^(1)
prob2<-odds2/(odds2+1)
myTrans[4,1]<-(1-prob1)*(1-prob2);myTrans[4,2]<-(1-prob1)*prob2
myTrans[4,3]<-prob1*(1-prob2);myTrans[4,4]<-prob1*prob2
return(myTrans)
}
#####
## Simulation ##
#####
# Seedis1803
SimAggLogit<-function(N=30,L=30,K=100,
B0_1=0,AE_1=0,PE_1=0,Int_1=0,
B0_2=0,AE_2=0,PE_2=0,Int_2=0,
init=c(0.25,0.25,0.25,0.25),
delta=0.05){

```

```

require(DySeq)
Trans1<-APIMtoTrans(B0_1,AE_1,PE_1,Int_1,
B0_2,AE_2,PE_2,Int_2)
intercept1<-c(); actor1<-c(); partner1<-c()
interac1<-c(); intercept2<-c(); actor2<-c()
partner2<-c(); interac2<-c(); Lintercept1<-c();
Lactor1<-c(); Lpartner1<-c(); Linterac1<-c()
Lintercept2<-c(); Lactor2<-c(); Lpartner2<-c()
Linterac2<-c()
for(count in 1:K){
Msample<-simSeqSample(Trans1,init,L,N)
my.states<-StateExpand(Msample,1:L,L+1:L)
my.trans<-StateTrans(my.states,TRUE)
my.logseq<-LogSeq(my.trans,delta=delta)
lambdas<-my.logseq[[1]]
intercept1[count]<-stats::t.test(lambdas[,1])$p.value
actor1[count]<-stats::t.test(lambdas[,2])$p.value
partner1[count]<-stats::t.test(lambdas[,3])$p.value
interac1[count]<-stats::t.test(lambdas[,4])$p.value
Lintercept1[count]<-mean(lambdas[,1])
Lactor1[count]<-mean(lambdas[,2])
Lpartner1[count]<-mean(lambdas[,3])
Linterac1[count]<-mean(lambdas[,4])
my.states<-StateExpand(Msample,1:L,L+1:L)
my.trans<-StateTrans(my.states,FALSE)
my.logseq<-LogSeq(my.trans,delta=delta)
lambdas<-my.logseq[[1]]
intercept2[count]<-stats::t.test(lambdas[,1])$p.value
actor2[count]<-stats::t.test(lambdas[,2])$p.value
partner2[count]<-stats::t.test(lambdas[,3])$p.value
interac2[count]<-stats::t.test(lambdas[,4])$p.value
Lintercept2[count]<-mean(lambdas[,1])
Lactor2[count]<-mean(lambdas[,2])
Lpartner2[count]<-mean(lambdas[,3])
Linterac2[count]<-mean(lambdas[,4])
}out<-c(); out<-c(out,mean(intercept1 <.05))
out<-c(out,mean(actor1 <.05)); out<-c(out,mean(partner1 <.05))
out<-c(out,mean(interac1 <.05)); out<-c(out,mean(intercept2 <.05))
out<-c(out,mean(partner2 <.05)); out<-c(out,mean(actor2 <.05))
out<-c(out,mean(interac2 <.05)); out<-c(out,mean(Lintercept1))

```

```

out<-c(out,mean(Lactor1));out<-c(out,mean(Lpartner1))
out<-c(out,mean(Linterac1));out<-c(out,mean(Lintercept2))
out<-c(out,mean(Lpartner2));out<-c(out,mean(Lactor2))
out<-c(out,mean(Linterac2));out<-t(out)
colnames(out)<-c("Rej.Inter1","Rej.AE1","Rej.PE1","Rej.AE*PE1",
"Rej.Inter2","Rej.Actor2","Rej.Partner2","Rej.AE*PE2",
"EstInter1","EstAE1","EstPE1","EstAE*PE1",
"EstInter2","EstActor2","EstPartner2","EstAE*PE2")
out<-t(out)
return(out)
}

```

B.4. Interaction Delta and L on Type-IError

```

MyNames<-c();out<-c();K<-1000
count<-0
for(EFF in c(0,0.2,0.4,0.6,0.8,1)){
for(N in c(10,30,50,70,100)){
for(L in c(10,30,50,70,100)){
SimOut<-SimAggLogit(N,L,K,
B0_1=0,AE_1=EFF,PE_1=0,Int_1=0,
B0_2=0,AE_2=0,PE_2=0,Int_2=0,
init=c(0.25,0.25,0.25,0.25),delta=.05)

out<-cbind(out,SimOut);count<-count+1
print(paste(count,"von",5*5*6))
MyNames<-c(MyNames,paste("b=",EFF,"N=",N,"L=",L,sep=""))
}
}
}
colnames(out)<-MyNames

```

B.5. Multilevel APIM

```

SimMLM<-function(N=30,L=30,K=100,B0_1=0,AE_1=0,PE_1=0,
Int_1=0,B0_2=0,AE_2=0,PE_2=0,
Int_2=0,init=c(0.25,0.25,0.25,0.25)){

require(DySeq);require(lme4);require(lmerTest)
#Bedingung:
Trans1<-APIMtoTrans(B0_1,AE_1,PE_1,Int_1,

```

B0_2,AE_2,PE_2,Int_2)

```

pb<-txtProgressBar(min=0,max=K,initial=0,char=" ",
width=NA,title,label,style=3,file="")
intercept1<-c();actor1<-c();partner1<-c();interac1<-c()
intercept2<-c();actor2<-c();partner2<-c();interac2<-c()
Lintercept1<-c();Lactor1<-c();Lpartner1<-c()
Linterac1<-c();Lintercept2<-c();Lactor2<-c()
Lpartner2<-c();Linterac2<-c()

for(count in 1:K){setTxtProgressBar(pb,count)
Msample<-simSeqSample(Trans1,init,L,N)
ML_data<-ML_Trans(Msample,1:L,L+1:L)
MLAP_data<-MLAP_Trans(ML_data)
names(MLAP_data)[1]<-"sex"
MLAP_data$sex<-as.factor(MLAP_data$sex)
levels(MLAP_data$sex)<-c("female","male")
MLAP_data$Partner[MLAP_data$Partner==0]<-(-1)
MLAP_data$Actor[MLAP_data$Actor==0]<-(-1)
fit<-glmer(DV~1+sex+Actor+Partner+Actor*Partner+
sex*Actor+sex*Partner+sex*Actor*Partner+
(1|ID),data=MLAP_data,
family=binomial,
control=glmerControl(optimizer="bobyqa",
optCtrl=list(maxfun=5e6)))
res<-summary(fit)$coefficients
intercept1[count]<-res[1,4];actor1[count]<-res[3,4]
partner1[count]<-res[4,4];interac1[count]<-res[5,4]
Lintercept1[count]<-res[1,1];Lactor1[count]<-res[3,1]
Lpartner1[count]<-res[4,1];Linterac1[count]<-res[5,1]
contrasts(MLAP_data$sex)<-c(1,0)
fit2<-glmer(DV~1+sex+Actor+Partner+Actor*Partner+
sex*Actor+sex*Partner+sex*Actor*Partner+
(1|ID),data=MLAP_data,
family=binomial,control=glmerControl(
optimizer="bobyqa",optCtrl=list(maxfun=5e6)))
res2<-summary(fit2)$coefficients;intercept2[count]<-res2[1,4]
actor2[count]<-res2[3,4];partner2[count]<-res2[4,4]
interac2[count]<-res2[5,4];Lintercept2[count]<-res2[1,1]
Lactor2[count]<-res2[3,1];Lpartner2[count]<-res2[4,1]

```

```

Linterac2[count]<-res2[5,1]};out<-c()
out<-c(out,mean(intercept1 <.05));out<-c(out,mean(actor1 <.05))
out<-c(out,mean(partner1 <.05));out<-c(out,mean(interac1 <.05))
out<-c(out,mean(intercept2 <.05));out<-c(out,mean(actor2 <.05))
out<-c(out,mean(partner2 <.05));out<-c(out,mean(interac2 <.05))
out<-c(out,mean(Lintercept1));out<-c(out,mean(Lactor1))
out<-c(out,mean(Lpartner1));out<-c(out,mean(Linterac1))
out<-c(out,mean(Lintercept2));out<-c(out,mean(Lpartner2))
out<-c(out,mean(Lactor2));out<-c(out,mean(Linterac2))
out<-t(out);colnames(out)<-c("Rej. Inter1", "Rej. AE1", "Rej. PE1",
"Rej. AE*PE1", "Rej. Inter2", "Rej. Actor2", "Rej. Partner2",
"Rej. AE*PE2", "EstInter1", "EstAE1", "EstPE1",
"EstAE*PE1", "EstInter2", "EstActor2", "EstPartner2",
"EstAE*PE2");out<-t(out);return(out)}

```

```

#####
#EffectoflengthonPower#
#####
#Seedis2860
set.seed(2860)

```

```

MyNames<-c();out<-c();K<-500;count<-0
for(EFF in c(0,0.2,0.4,1)){
for(N in c(10,50,100)){
for(L in c(10,50,100)){
SimOut<-SimMLM(N,L,K,
B0_1=0,AE_1=EFF,PE_1=0,Int_1=0,
B0_2=0,AE_2=0,PE_2=0,Int_2=0,
init=c(0.25,0.25,0.25,0.25))
out<-cbind(out,SimOut);count<-count+1
print(paste(count, "von",6*5*5))
MyNames<-c(MyNames, paste("b=",EFF, "N=",N, "L=",L, sep=""))}}
colnames(out)<-MyNames

```

B.6. Basic Markov Model APIM

```

library(DySeq); library(seqHMM); library(TraMineR); library(parallel)

```

```

#####
##CreatingSequences##
#####

```



```

simSeq<-function(trans , initial , length){

init<-sample(c(1,2,3,4),1 ,prob=initial)

for(i in 2:length){ if (init[i-1]==1){init[i]<-sample(
c(1,2,3,4),1 ,prob=trans [1 ,])} else if (init[i-1]==2){
init[i]<-sample(c(1,2,3,4),1 ,prob=trans [2 ,])} else
if (init[i-1]==3){init[i]<-sample(c(1,2,3,4),1 ,
prob=trans [3 ,])} else {init[i]<-sample(c(1,2,3,4),1 ,
prob=trans [4 ,])}}
return (init)}

#####
#Simulateacompletesampleofdyadicsequences#
#####
simSeqSample<-function(trans , initial , length ,N){
comb<-simSeq(trans , initial , length)
seq1<-c (); seq1 [comb==1 | comb==3]<-0
seq1 [comb==2 | comb==4]<-1; seq2<-c ()
seq2 [comb==1 | comb==2]<-0; seq2 [comb==3 | comb==4]<-1
out<-c (seq1 , seq2); for (i in 2:N){
comb<-simSeq(trans , initial , length)
seq1 [comb==1 | comb==3]<-0; seq1 [comb==2 | comb==4]<-1
seq2 [comb==1 | comb==2]<-0; seq2 [comb==3 | comb==4]<-1
out<-rbind (out , c (seq1 , seq2))
} return (out)}

#####
#TransformAPIMLogitsintoTransitionMatrix#
#####
APIMtoTrans<-function (B0_1 ,AE_1 ,PE_1 ,Int_1 ,
B0_2 ,AE_2 ,PE_2 ,Int_2){
myTrans<-matrix (NA,4 ,4)
odds1<-exp (B0_1)*exp (AE_2)^(-1)*exp (PE_2)^(-1)*exp (Int_2)^(1)
odds2<-exp (B0_2)*exp (AE_1)^(-1)*exp (PE_1)^(-1)*exp (Int_1)^(1)
prob1<-odds1/(odds1+1); prob2<-odds2/(odds2+1)
myTrans [1 ,1]<-(1-prob1)*(1-prob2); myTrans [1 ,2]<-(1-prob1)*prob2
myTrans [1 ,3]<-prob1*(1-prob2); myTrans [1 ,4]<-prob1*prob2
odds1<-exp (B0_1)*exp (AE_2)^(-1)*exp (PE_2)^(1)*exp (Int_2)^(-1)
odds2<-exp (B0_2)*exp (AE_1)^(1)*exp (PE_1)^(-1)*exp (Int_1)^(-1)
prob1<-odds1/(odds1+1); prob2<-odds2/(odds2+1)

```

```

myTrans[2,1]<-(1-prob1)*(1-prob2); myTrans[2,2]<-(1-prob1)*prob2
myTrans[2,3]<-prob1*(1-prob2); myTrans[2,4]<-prob1*prob2
odds1<-exp(B0_1)*exp(AE_2)^(1)*exp(PE_2)^(-1)*exp(Int_2)^(-1)
odds2<-exp(B0_2)*exp(AE_1)^(-1)*exp(PE_1)^(1)*exp(Int_1)^(-1)
prob2<-odds2/(odds2+1); prob1<-odds1/(odds1+1)
myTrans[3,1]<-(1-prob1)*(1-prob2); myTrans[3,2]<-(1-prob1)*prob2
myTrans[3,3]<-prob1*(1-prob2); myTrans[3,4]<-prob1*prob2
odds1<-exp(B0_1)*exp(AE_2)^(1)*exp(PE_2)^(1)*exp(Int_2)^(1)
odds2<-exp(B0_2)*exp(AE_1)^(1)*exp(PE_1)^(1)*exp(Int_1)^(1)
prob1<-odds1/(odds1+1); prob2<-odds2/(odds2+1)
myTrans[4,1]<-(1-prob1)*(1-prob2); myTrans[4,2]<-(1-prob1)*prob2
myTrans[4,3]<-prob1*(1-prob2); myTrans[4,4]<-prob1*prob2
return(myTrans)
}

```

```

#####
##TransitionintoAPIM#
#####
TransToAPIM<-function(M){
DC_none_L<-log(DC_none/(1-DC_none))
DC_SC_L<-log(DC_SC/(1-DC_SC))
DC_DC_L<-log(DC_DC/(1-DC_DC))
DC_SC_DC_L<-log(DC_SC_DC/(1-DC_SC_DC))
DCb0<-sum(DC_none_L,DC_SC_L,DC_DC_L,DC_SC_DC_L)/4
DCPart<-(DC_SC_L+DC_SC_DC_L)/2-DCb0
DCAct<-(DC_DC_L+DC_SC_DC_L)/2-DCb0
DCint<-DC_SC_DC_L-(DCb0+DCAct+DCPart)
SC_none<-sum(M[1,c(2,4)]); SC_SC<-sum(M[2,c(2,4)])
SC_DC<-sum(M[3,c(2,4)]); SC_SC_DC<-sum(M[4,c(2,4)])
SC_none_L<-log(SC_none/(1-SC_none))
SC_SC_L<-log(SC_SC/(1-SC_SC))
SC_DC_L<-log(SC_DC/(1-SC_DC))
SC_SC_DC_L<-log(SC_SC_DC/(1-SC_SC_DC))
SCb0<-sum(SC_none_L,SC_SC_L,SC_DC_L,SC_SC_DC_L)/4
SCAct<-(SC_SC_L+SC_SC_DC_L)/2-SCb0
SCPart<-(SC_DC_L+SC_SC_DC_L)/2-SCb0
SCint<-SC_SC_DC_L-(SCb0+SCAct+SCPart)
results<-c(DCb0,DCAct,DCPart,DCint,
SCb0,SCAct,SCPart,SCint)
names(results)<-c("DCIntercept","DCActor","DCPartner",

```

```

"DCInteraction", "SCIntercept", "SCActor", "SCPartner",
"SCInteraction"); return(results)}
#####
#MarkovasAPIM#
#####
trans1<-APIMtoTrans(B0_1=0,AE_1=0,PE_1=0,Int_1=0,
B0_2=0,AE_2=0,PE_2=0,Int_2=0)
x<-simSeqSample(trans=trans1, initial=rep(.25,4), length=L,N=30)
MasAPIM<-function(x, first, second, boot=1000, SimOut=FALSE, CPU=1,
sim="ordinary", parallel="multicore"){
out<-c(); require(boot)
MyBetas<-function(data, indices){
a<-StateExpand(data[indices,], first, second)
b<-suppressWarnings(seqdef(a[, first],
start=1,
labels=c("0-0", "1-0", "0-1", "1-1")))
z<-seqtrate(b); return(TransToAPIM(z))
} results<-boot(data=x, statistic=MyBetas,
R=boot, ncpus=CPU, sim=sim)
out[1]<-results$t0[1];
DC_b0_H0_Dist<-results$t[,1]-mean(results$t[,1])
out[9]<-mean(DC_b0_H0_Dist>abs(results$t0[1]))|
DC_b0_H0_Dist<(-abs(results$t0[1]))
out[2]<-results$t0[2];DC_Act_H0_Dist<-results$t[,2]-mean(results$t[,2])
out[10]<-mean(DC_Act_H0_Dist>abs(results$t0[2]))
|DC_Act_H0_Dist<(-abs(results$t0[2]))
out[3]<-results$t0[3];DC_Par_H0_Dist<-results$t[,3]-mean(results$t[,3])
out[11]<-mean(DC_Par_H0_Dist>abs(results$t0[3]))
|DC_Par_H0_Dist<(-abs(results$t0[3]))
out[4]<-results$t0[4];DC_Int_H0_Dist<-results$t[,4]-mean(results$t[,4])
out[12]<-mean(DC_Int_H0_Dist>abs(results$t0[4]))
|DC_Int_H0_Dist<(-abs(results$t0[4]))
out[5]<-results$t0[5];SC_b0_H0_Dist<-results$t[,5]-mean(results$t[,5])
out[13]<-mean(SC_b0_H0_Dist>abs(results$t0[5]))
|SC_b0_H0_Dist<(-abs(results$t0[5]))
out[6]<-results$t0[6];SC_Act_H0_Dist<-results$t[,6]-mean(results$t[,6])
out[14]<-mean(SC_Act_H0_Dist>abs(results$t0[6]))
|SC_Act_H0_Dist<(-abs(results$t0[6]))
out[7]<-results$t0[7];SC_Par_H0_Dist<-results$t[,7]-mean(results$t[,7])
out[15]<-mean(SC_Par_H0_Dist>abs(results$t0[7]))

```

```

|SC_Par_H0_Dist<(-abs(results$t0[7]))
out[8]<-results$t0[8];SC_Int_H0_Dist<-results$t[,8]-mean(results$t[,8])
out[16]<-mean(SC_Int_H0_Dist>abs(results$t0[8]))
|SC_Int_H0_Dist<(-abs(results$t0[8]))
if(SimOut){
names(out)<-c("DC_b0", "DC_Actor", "DC_Partner", "DC_Inter",
"SC_b0", "SC_Actor", "SC_Partner", "SC_Inter",
"P_DC_b0", "P_DC_Actor", "P_DC_Partner", "P_DC_Inter",
"P_SC_b0", "P_SC_Actor", "P_SC_Partner", "P_SC_Inter")
return(out)} else {out2<-data.frame(rep(NA,8), rep(NA,8))
rownames(out2)<-c("FirstIntercept", "FirstActor", "FirstPartner",
"FirstInteraction", "SecondIntercept",
"SecondActor", "SecondPartner", "SecondInteraction")
colnames(out2)<-c("Estimate", "P_Value")
out2[1:4,1]<-out[1:4]; out2[5:8,1]<-out[5:8]
out2[1:4,2]<-out[9:12]; out2[5:8,2]<-out[13:16]; return(out2)}}
#####
## Simulation ##
#####
set.seed(7505)
SimAggLogit<-function(N=30,L=30,K=10,
B0_1=0,AE_1=0,PE_1=0,Int_1=0,
B0_2=0,AE_2=0,PE_2=0,Int_2=0,
init=c(0.25,0.25,0.25,0.25),
boot=10,CPU=1,sim="ordinary"){
count<-1;trans1<-APIMtoTrans(B0_1,AE_1,PE_1,Int_1,
B0_2,AE_2,PE_2,Int_2)
intercept1<-c(); actor1<-c(); partner1<-c(); interac1<-c()
intercept2<-c(); actor2<-c(); partner2<-c(); interac2<-c()
Pintercept1<-c(); Pactor1<-c(); Ppartner1<-c(); Pinterac1<-c()
Pintercept2<-c(); Pactor2<-c(); Ppartner2<-c(); Pinterac2<-c()
for(i in 1:K){x<-simSeqSample(trans=trans1, initial=init, length=L,
N=N); SimM<-MasAPIM(x, 1:L, L+1:L, boot, TRUE)
intercept1<-c(intercept1, SimM[1]); actor1<-c(actor1, SimM[2])
partner1<-c(partner1, SimM[3]); interac1<-c(interac1, SimM[4])
intercept2<-c(intercept2, SimM[5]); actor2<-c(actor2, SimM[6])
partner2<-c(partner2, SimM[7]); interac2<-c(interac2, SimM[8])
Pintercept1<-c(Pintercept1, SimM[9]); Pactor1<-c(Pactor1, SimM[10])
Ppartner1<-c(Ppartner1, SimM[11]); Pinterac1<-c(Pinterac1, SimM[12])
Pintercept2<-c(Pintercept2, SimM[13]); Pactor2<-c(Pactor2, SimM[14])

```

```

Ppartner2<-c(Ppartner2,SimM[15]);Pinterac2<-c(Pinterac2,SimM[16])
cat(paste("\n",count,"\n",count,"\n",count,
"\n",count,"\n",count,"\n",count,
"\n",count,"\n",count,"\n",count,
"\n",count,"\n",count,"\n",count))
save(intercept1,actor1,partner1,interac1,intercept2,actor2,
partner2,interac2,Pintercept1,Pactor1,Ppartner1,Pinterac1,
Pintercept2,Pactor2,Ppartner2,Pinterac2,
file=paste("~/temp/Sim",count,".R"))
count<-count+1}res<-c();res<-c(res,mean(Pintercept1<.05))
res<-c(res,mean(Pactor1<.05));res<-c(res,mean(Ppartner1<.05))
res<-c(res,mean(Pinterac1<.05));res<-c(res,mean(Pintercept2<.05))
res<-c(res,mean(Pactor2<.05));res<-c(res,mean(Ppartner2<.05))
res<-c(res,mean(Pinterac2<.05));res<-c(res,mean(intercept1))
res<-c(res,mean(actor1));res<-c(res,mean(partner1))
res<-c(res,mean(interac1));res<-c(res,mean(intercept2))
res<-c(res,mean(partner2));res<-c(res,mean(actor2))
res<-c(res,mean(interac2));names(res)<-c("Rej.Inter1",
"Rej.AE1","Rej.PE1","Rej.AE*PE1","Rej.Inter2",
"Rej.Actor2","Rej.Partner2","Rej.AE*PE2",
"EstInter1","EstAE1","EstPE1","EstAE*PE1",
"EstInter2","EstActor2","EstPartner2",
"EstAE*PE2");return(res)}
Sim1<-SimAggLogit(N=30,L=30,K=100,
B0_1=0,AE_1=0,PE_1=0,Int_1=0,
B0_2=0,AE_2=0,PE_2=0,Int_2=0,
init=c(0.25,0.25,0.25,0.25),
boot=1000,CPU=8,sim="ordinary")
save(Sim1,file="Sim1.R")
for(i in 1:1000){file.remove(paste("temp/Sim",i,".R"))}

```

B.7. Latent Hazard Model(Restricted Hidden Markov)

```

library(DySeq);library(seqHMM);library(TraMineR);library(parallel)
#####
##CreatingSequences##
#####
simSeq<-function(trans,initial,len){
init<-sample(c(0,1),1,prob=initial)
for(i in 2:len){if(init[i-1]==0){init[i]<-sample(c(0,1),
1,prob=trans[1,])}else{init[i]<-sample(c(0,1),1,

```

```

prob=trans [2 ,])}} return (init))
#####
# SimulateEmissions#
#####
genEmis<-function(MySeq,MyEmis_A,MyEmis_B){
out<-numeric(length(MySeq)) for(i in 1:length(MySeq))
{ if(MySeq[i]==0){out[i]<-sample(c(0,1,2,3),1,
prob=c(MyEmis_A))} else {out[i]<-sample(c(0,1,2,3),1,
prob=c(MyEmis_B))}} out}
#####
### SimulateSample###
#####
Gen_LatentHaz_Sample<-function(trans,initial,len,Emission_A,
Emission_B,size){out<-matrix(NA,size,len) for(i in 1:size){
out[i,]<-genEmis(simSeq(trans=trans,initial=initial,len=len),
Emission_A,Emission_B)} out[size,1:4]<-c(0,1,2,3);out}
#####
# Simulating single condition#
#####
sim_Latent_Hazard<-function(trans=matrix(c(0.8,0.2,
0,1),2,2,byrow=T),
initial=c(1,0),len=50,
Emission_A=c(0.90,0.03,0.03,0.04),
Emission_B=c(0.03,0.03,0.04,0.90),
size=100,MySamples=100){
require(seqHMM); require(TraMineR); require(DySeq)
pb<-txtProgressBar(min=0,max=MySamples,initial=0,char="=",
width=NA,title,label,style=3,file="")
myHazard<-numeric(MySamples);Em_State1_0<-numeric(MySamples)
Em_State1_1<-numeric(MySamples);Em_State1_2<-numeric(MySamples)
Em_State1_3<-numeric(MySamples);Em_State2_0<-numeric(MySamples)
Em_State2_1<-numeric(MySamples);Em_State2_2<-numeric(MySamples)
Em_State2_3<-numeric(MySamples);for(k in 1:MySamples){
x<-Gen_LatentHaz_Sample(trans=trans,initial=initial,len=len,
Emission_A=Emission_A,Emission_B=Emission_B,size=size)
suppressMessages(my_seq<-seqdef(x,start=1,labels
=c("noSC/DC","SOnly","DOnly","SC+DC")))
sc_init<-c(1,0);sc_trans<-matrix(c(0.50,0.50,0,1),
nrow=2,ncol=2,byrow=TRUE)
sc_emiss<-matrix(0.25,nrow=2,ncol=4)

```

```

sc_initmod<-build_hmm(observations=my_seq,
initial_probs=sc_init,
transition_probs=sc_trans,
emission_probs=sc_emiss)
sc_fit<-fit_model(sc_initmod,global_step=TRUE,
local_step=TRUE,control_em=list(restart=list(times=10)))
sc_fit$model$transition_probs;sc_fit$model$emission_probs[1,2]
myHazard[k]<-sc_fit$model$transition_probs[1,2]
Em_State1_0[k]<-sc_fit$model$emission_probs[1,1]
Em_State1_1[k]<-sc_fit$model$emission_probs[1,2]
Em_State1_2[k]<-sc_fit$model$emission_probs[1,3]
Em_State1_3[k]<-sc_fit$model$emission_probs[1,4]
Em_State2_0[k]<-sc_fit$model$emission_probs[2,1]
Em_State2_1[k]<-sc_fit$model$emission_probs[2,2]
Em_State2_2[k]<-sc_fit$model$emission_probs[2,3]
Em_State2_3[k]<-sc_fit$model$emission_probs[2,4]
setTxtProgressBar(pb,k,title=NULL,label=NULL)}
out<-data.frame(myHazard,Em_State1_0,Em_State1_1,Em_State1_2,
Em_State1_3,Em_State2_0,Em_State2_1,Em_State2_2,Em_State2_3)}
#####
## Simulation ##
#####
set.seed(8840)
Emission_A<-c(0.90,0.03,0.03,0.04)
Emission_B<-c(0.03,0.03,0.04,0.90)
Emi<-"High"
trans<-matrix(c(0.9,0.1,
0,1),2,2,byrow=T)
Haz<-"H10"
for(i in c(10,20,30,40,50,100)){
for(k in c(10,20,30,40,50,100)){
Sim1<-sim_Latent_Hazard(trans=trans,
initial=c(1,0),
len=k,
Emission_A=Emission_A,
Emission_B=Emission_B,
size=i,
MySamples=1000)
save(Sim1,file=paste(Emi,"_",Haz,"_N",i,"_t",k,"_Sim.R",sep=""))}}

```

B.8. Hidden Markov (Correct Number of Latent States)

```

sim_Latent_Markov2<-function(trans=matrix(c(0.8,0.2,
0,1),2,2,byrow=T),
initial=c(1,0),
len=50,
Emission_A=c(0.90,0.03,0.03,0.04),
Emission_B=c(0.03,0.03,0.04,0.90),
size=10,
MySamples=10){
  require(seqHMM);require(TraMineR);require(DySeq)
  pb<-txtProgressBar(min=0,max=MySamples,initial=0,char=" ",
width=NA,title,label,style=3,file="")
  df1<-numeric(MySamples);AIC1<-numeric(MySamples)
  BIC1<-numeric(MySamples);df2<-numeric(MySamples)
  AIC2<-numeric(MySamples);BIC2<-numeric(MySamples)
  df3<-numeric(MySamples);AIC3<-numeric(MySamples)
  BIC3<-numeric(MySamples);Trans2_AA<-numeric(MySamples)
  Trans2_AB<-numeric(MySamples);Trans2_BA<-numeric(MySamples)
  Trans2_BB<-numeric(MySamples);Em_State1_0<-numeric(MySamples)
  Em_State1_1<-numeric(MySamples);Em_State1_2<-numeric(MySamples)
  Em_State1_3<-numeric(MySamples);Em_State2_0<-numeric(MySamples)
  Em_State2_1<-numeric(MySamples);Em_State2_2<-numeric(MySamples)
  Em_State2_3<-numeric(MySamples)
  for(k in 1:MySamples){
    x<-Gen_LatentHaz_Sample(trans=trans,initial=initial,
len=len,Emission_A=Emission_A,
Emission_B=Emission_B,
size=size)
    suppressMessages(my_seq<-seqdef(x,start=1,
labels=c("noSC/DC","SOnly","DOnly","SC+DC")))
    sc_init1<-c(.25,.25,.25,.25)
    sc_trans1<-matrix(c(.25),nrow=4,ncol=4,byrow=TRUE)
    sc_emiss1<-matrix(c(1,0,0,0,
0,1,0,0,
0,0,1,0,
0,0,0,1),nrow=4,ncol=4)
    sc_initmod1<-build_hmm(observations=my_seq,
initial_probs=sc_init1,
transition_probs=sc_trans1,

```



```

emission_probs=sc_emiss1)
sc_fit1<-fit_model(sc_initmod1,global_step=TRUE,
local_step=TRUE,control_em=list(restart=list(times=10)))
df1[k]<-attr(sc_fit1$model,"df");AIC1[k]<-AIC(sc_fit1$model)
BIC1[k]<-BIC(sc_fit1$model);sc_init2<-c(.50,.50)
sc_trans2<-matrix(c(0.50,0.50,
0.50,.50),
nrow=2,ncol=2,byrow=TRUE)
sc_emiss2<-matrix(0.25,nrow=2,ncol=4)
sc_initmod2<-build_hmm(observations=my_seq,
initial_probs=sc_init2,
transition_probs=sc_trans2,
emission_probs=sc_emiss2)
sc_fit2<-fit_model(sc_initmod2,global_step=TRUE,
local_step=TRUE,control_em=list(restart=list(times=10)))
df2[k]<-attr(sc_fit2$model,"df")
AIC2[k]<-AIC(sc_fit2$model)BIC2[k]<-BIC(sc_fit2$model)
trans2_A<-sc_fit2$model$transition_probs
trans2_B<-sc_fit2$model$transition_probs
trans2_B[2,]<-sc_fit2$model$transition_probs[1,]
trans2_B[1,]<-sc_fit2$model$transition_probs[2,]
trans2_B[,2]<-trans2_B[,1];trans2_B[,1]<-trans2_B[,2]
A<-sum(abs(matrix(c(Emission_A,Emission_B),nrow=2,
ncol=4,byrow=T)-sc_fit2$model$emission_probs))
B<-sum(abs(matrix(c(Emission_B,Emission_A),nrow=2,
ncol=4,byrow=T)-sc_fit2$model$emission_probs))
if(B>A){est_trans2<-as.numeric(trans2_A)
Em_State1_0[k]<-sc_fit2$model$emission_probs[1,1]
Em_State1_1[k]<-sc_fit2$model$emission_probs[1,2]
Em_State1_2[k]<-sc_fit2$model$emission_probs[1,3]
Em_State1_3[k]<-sc_fit2$model$emission_probs[1,4]
Em_State2_0[k]<-sc_fit2$model$emission_probs[2,1]
Em_State2_1[k]<-sc_fit2$model$emission_probs[2,2]
Em_State2_2[k]<-sc_fit2$model$emission_probs[2,3]
Em_State2_3[k]<-sc_fit2$model$emission_probs[2,4]
} else {est_trans2<-as.numeric(trans2_B)
Em_State1_0[k]<-sc_fit2$model$emission_probs[2,1]
Em_State1_1[k]<-sc_fit2$model$emission_probs[2,2]
Em_State1_2[k]<-sc_fit2$model$emission_probs[2,3]
Em_State1_3[k]<-sc_fit2$model$emission_probs[2,4]

```

```

Em_State2_0[k]<-sc_fit2$model$emission_probs[1,1]
Em_State2_1[k]<-sc_fit2$model$emission_probs[1,2]
Em_State2_2[k]<-sc_fit2$model$emission_probs[1,3]
Em_State2_3[k]<-sc_fit2$model$emission_probs[1,4]}
Trans2_AA[k]<-est_trans2[1];Trans2_AB[k]<-est_trans2[2]
Trans2_BA[k]<-est_trans2[3];Trans2_BB[k]<-est_trans2[4]
sc_init3<-c(.33,.33,.34);sc_trans3<-matrix(c(0.33,
0.33,0.34,0.33,.33,.34,0.33,.33,0.34),
nrow=3,ncol=3,byrow=TRUE)
sc_emiss3<-matrix(c(.25),nrow=3,ncol=4,byrow=T)
sc_initmod3<-build_hmm(observations=my_seq,
initial_probs=sc_init3,
transition_probs=sc_trans3,
emission_probs=sc_emiss3)
sc_fit3<-fit_model(sc_initmod3,global_step=TRUE,
local_step=TRUE,control_em=list(restart=list(times=10)))
df3[k]<-attr(sc_fit3$model,"df");AIC3[k]<-AIC(sc_fit3$model)
BIC3[k]<-BIC(sc_fit3$model);setTxtProgressBar(pb,k,title=NULL,label=NULL)}
out<-data.frame(df1,AIC1,BIC1,df2,AIC2,BIC2,df3,AIC3,BIC3,Trans2_AA,
Trans2_AB,Trans2_BA,Trans2_BB,Em_State1_0,Em_State1_1,Em_State1_2,
Em_State1_3,Em_State2_0,Em_State2_1,Em_State2_2,Em_State2_3)}
#####
##Simulation##
#####
set.seed(4341)
Emission_A<-c(0.90,0.03,0.03,0.04)
Emission_B<-c(0.03,0.03,0.04,0.90)
Emi<-"Scenario_A"
trans<-matrix(c(0.90,0.1,
0.05,.95),2,2,byrow=T)
Class<-"2Classes"
for(i in c(10,20,30,40,50,100)){
for(k in c(10,20,30,40,50,100)){
myTitle<-paste(Emi,"_",Class,"_N",i,"_t",k,"_Sim.R",sep="")
cat(paste("\n\n"))
print(myTitle)
Sim1<-sim_Latent_Markov2(trans=trans,
initial=c(1,0),
len=k,
Emission_A=Emission_A,

```

```

Emission_B=Emission_B,
size=i ,
MySamples=500)
save(Sim1 , file=paste(Emi, "_", Class , "_N" , i , "_t" , k , "_Sim.R" , sep=""))}}
#####
###SIM3HiddenStates###
#####
library(DySeq); library(seqHMM); library(TraMineR); library(parallel)

#####
#Simulating single condition#
#####
sim_Latent_Markov3<-function(trans=
matrix(c(0.8,0.2,0,
0.2,0.2,0.6,
0,0.5,0.5),3,3,byrow=T),
initial=c(.33,.33,.34),len=50,
          Emission_A=c(0.90,0.03,0.03,0.04),
Emission_B=c(0.03,0.03,0.04,0.90),
Emission_C=c(0.03,0.03,0.14,0.80),
size=10,MySamples=10){require(seqHMM)
require(TraMineR);require(DySeq)
pb<-txtProgressBar(min=0,max=MySamples,initial=0,char="=",
width=NA,title,label,style=3,file="")
df1<-numeric(MySamples);AIC1<-numeric(MySamples)
BIC1<-numeric(MySamples);df2<-numeric(MySamples)
AIC2<-numeric(MySamples);BIC2<-numeric(MySamples)
df3<-numeric(MySamples);AIC3<-numeric(MySamples)
BIC3<-numeric(MySamples);Trans3_AA<-numeric(MySamples)
Trans3_AB<-numeric(MySamples);Trans3_AC<-numeric(MySamples)
Trans3_BA<-numeric(MySamples);Trans3_BB<-numeric(MySamples)
Trans3_BC<-numeric(MySamples);Trans3_CA<-numeric(MySamples)
Trans3_CB<-numeric(MySamples);Trans3_CC<-numeric(MySamples)
Em_State1_0<-numeric(MySamples);Em_State1_1<-numeric(MySamples)
Em_State1_2<-numeric(MySamples);Em_State1_3<-numeric(MySamples)
Em_State2_0<-numeric(MySamples);Em_State2_1<-numeric(MySamples)
Em_State2_2<-numeric(MySamples);Em_State2_3<-numeric(MySamples)
Em_State3_0<-numeric(MySamples);Em_State3_1<-numeric(MySamples)
Em_State3_2<-numeric(MySamples);Em_State3_3<-numeric(MySamples)
for(k in 1:MySamples){

```

```

x<-Gen_Sample(trans=trans , initial=initial , len=len ,
Emission_A=Emission_A, Emission_B=Emission_B,
Emission_C=Emission_C, size=size)
suppressMessages(my_seq<-seqdef(x, start=1, labels=
c("noSC/DC" , "SOnly" , "DOnly" , "SC+DC" )))
sc_init1<-c(.25 ,.25 ,.25 ,.25)
sc_trans1<-matrix(c(.25) ,nrow=4,ncol=4,byrow=TRUE)
sc_emiss1<-matrix(c(1,0,0,0,
0,1,0,0,
0,0,1,0,
0,0,0,1) ,nrow=4,ncol=4)
sc_initmod1<-build_hmm(observations=my_seq ,
initial_probs=sc_init1 ,
transition_probs=sc_trans1 ,
emission_probs=sc_emiss1)
sc_fit1<-fit_model(sc_initmod1 , global_step=TRUE, local_step=TRUE,
control_em=list(restart=list(times=10)))
df1[k]<-attr(sc_fit1$model, "df"); AIC1[k]<-AIC(sc_fit1$model)
BIC1[k]<-BIC(sc_fit1$model); sc_init2<-c(.50 ,.50)
sc_trans2<-matrix(c(0.50 ,0.50 ,
0.50 ,.50) ,
nrow=2,ncol=2,byrow=TRUE)
sc_emiss2<-matrix(0.25 ,nrow=2,ncol=4)
sc_initmod2<-build_hmm(observations=my_seq ,
initial_probs=sc_init2 ,
transition_probs=sc_trans2 ,
emission_probs=sc_emiss2)
sc_fit2<-fit_model(sc_initmod2 , global_step=TRUE, local_step=TRUE
, control_em=list(restart=list(times=10)))
df2[k]<-attr(sc_fit2$model, "df"); AIC2[k]<-AIC(sc_fit2$model)
BIC2[k]<-BIC(sc_fit2$model); sc_init3<-c(.33 ,.33 ,.34)
sc_trans3<-matrix(c(0.33 ,0.33 ,0.34 ,
0.33 ,.33 ,.34 ,
0.33 ,.33 ,0.34) ,
nrow=3,ncol=3,byrow=TRUE)
sc_emiss3<-matrix(c(.25) ,nrow=3,ncol=4,byrow=T)
sc_initmod3<-build_hmm(observations=my_seq ,
initial_probs=sc_init3 ,
transition_probs=sc_trans3 ,
emission_probs=sc_emiss3)

```

```

sc_fit3<-fit_model(sc_initmod3, global_step=TRUE, local_step=TRUE,
                  control_em=list(restart=list(times=10)))
df3[k]<-attr(sc_fit3$model, "df"); AIC3[k]<-AIC(sc_fit3$model)
BIC3[k]<-BIC(sc_fit3$model)
setTxtProgressBar(pb,k, title=NULL, label=NULL)
trans3_A<-sc_fit3$model$transition_probs
ABC<-sum(abs(matrix(c(Emission_A, Emission_B, Emission_C),
                    nrow=3, ncol=4, byrow=T)-sc_fit3$model$emission_probs))
ACB<-sum(abs(matrix(c(Emission_A, Emission_C, Emission_B),
                    nrow=3, ncol=4, byrow=T)-sc_fit3$model$emission_probs))
BAC<-sum(abs(matrix(c(Emission_B, Emission_A, Emission_C),
                    nrow=3, ncol=4, byrow=T)-sc_fit3$model$emission_probs))
BCA<-sum(abs(matrix(c(Emission_B, Emission_C, Emission_A),
                    nrow=3, ncol=4, byrow=T)-sc_fit3$model$emission_probs))
CAB<-sum(abs(matrix(c(Emission_C, Emission_A, Emission_B),
                    nrow=3, ncol=4, byrow=T)-sc_fit3$model$emission_probs))
CBA<-sum(abs(matrix(c(Emission_C, Emission_B, Emission_A),
                    nrow=3, ncol=4, byrow=T)-sc_fit3$model$emission_probs))
fall<-c(ABC, ACB, BAC, BCA, CAB, CBA)
myorder<-which(fall==min(fall))
if(myorder==1){est_trans3<-as.numeric(trans3_A)
Em_State1_0[k]<-sc_fit3$model$emission_probs[1,1]
Em_State1_1[k]<-sc_fit3$model$emission_probs[1,2]
Em_State1_2[k]<-sc_fit3$model$emission_probs[1,3]
Em_State1_3[k]<-sc_fit3$model$emission_probs[1,4]
Em_State2_0[k]<-sc_fit3$model$emission_probs[2,1]
Em_State2_1[k]<-sc_fit3$model$emission_probs[2,2]
Em_State2_2[k]<-sc_fit3$model$emission_probs[2,3]
Em_State2_3[k]<-sc_fit3$model$emission_probs[2,4]
Em_State3_0[k]<-sc_fit3$model$emission_probs[3,1]
Em_State3_1[k]<-sc_fit3$model$emission_probs[3,2]
Em_State3_2[k]<-sc_fit3$model$emission_probs[3,3]
Em_State3_3[k]<-sc_fit3$model$emission_probs[3,4]
} if(myorder==2)
  {est_trans3<-as.numeric(trans3_A)[c(1,3,2,7,9,8,4,6,5)]
Em_State1_0[k]<-sc_fit3$model$emission_probs[1,1]
Em_State1_1[k]<-sc_fit3$model$emission_probs[1,2]
Em_State1_2[k]<-sc_fit3$model$emission_probs[1,3]
Em_State1_3[k]<-sc_fit3$model$emission_probs[1,4]
Em_State2_0[k]<-sc_fit3$model$emission_probs[3,1]

```

```

Em_State2_1[k]<-sc_fit3$model$emission_probs[3,2]
Em_State2_2[k]<-sc_fit3$model$emission_probs[3,3]
Em_State2_3[k]<-sc_fit3$model$emission_probs[3,4]
Em_State3_0[k]<-sc_fit3$model$emission_probs[2,1]
Em_State3_1[k]<-sc_fit3$model$emission_probs[2,2]
Em_State3_2[k]<-sc_fit3$model$emission_probs[2,3]
Em_State3_3[k]<-sc_fit3$model$emission_probs[2,4]}
  if(myorder==3){
    est_trans3<-as.numeric(trans3_A)[c(5,4,6,2,1,3,8,7,9)]
    Em_State1_0[k]<-sc_fit3$model$emission_probs[2,1]
    Em_State1_1[k]<-sc_fit3$model$emission_probs[2,2]
    Em_State1_2[k]<-sc_fit3$model$emission_probs[2,3]
    Em_State1_3[k]<-sc_fit3$model$emission_probs[2,4]
    Em_State2_0[k]<-sc_fit3$model$emission_probs[1,1]
    Em_State2_1[k]<-sc_fit3$model$emission_probs[1,2]
    Em_State2_2[k]<-sc_fit3$model$emission_probs[1,3]
    Em_State2_3[k]<-sc_fit3$model$emission_probs[1,4]
    Em_State3_0[k]<-sc_fit3$model$emission_probs[3,1]
    Em_State3_1[k]<-sc_fit3$model$emission_probs[3,2]
    Em_State3_2[k]<-sc_fit3$model$emission_probs[3,3]
    Em_State3_3[k]<-sc_fit3$model$emission_probs[3,4]}
  if(myorder==4){
    est_trans3<-as.numeric(trans3_A)[c(5,6,4,8,9,7,2,3,1)]
    Em_State1_0[k]<-sc_fit3$model$emission_probs[2,1]
    Em_State1_1[k]<-sc_fit3$model$emission_probs[2,2]
    Em_State1_2[k]<-sc_fit3$model$emission_probs[2,3]
    Em_State1_3[k]<-sc_fit3$model$emission_probs[2,4]
    Em_State2_0[k]<-sc_fit3$model$emission_probs[3,1]
    Em_State2_1[k]<-sc_fit3$model$emission_probs[3,2]
    Em_State2_2[k]<-sc_fit3$model$emission_probs[3,3]
    Em_State2_3[k]<-sc_fit3$model$emission_probs[3,4]
    Em_State3_0[k]<-sc_fit3$model$emission_probs[1,1]
    Em_State3_1[k]<-sc_fit3$model$emission_probs[1,2]
    Em_State3_2[k]<-sc_fit3$model$emission_probs[1,3]
    Em_State3_3[k]<-sc_fit3$model$emission_probs[1,4]}
  if(myorder==5){
    est_trans3<-as.numeric(trans3_A)[c(9,7,8,3,1,2,6,4,5)]
    Em_State1_0[k]<-sc_fit3$model$emission_probs[3,1]
    Em_State1_1[k]<-sc_fit3$model$emission_probs[3,2]
    Em_State1_2[k]<-sc_fit3$model$emission_probs[3,3]

```

```

Em_State1_3[k]<-sc_fit3$model$emission_probs[3,4]
Em_State2_0[k]<-sc_fit3$model$emission_probs[1,1]
Em_State2_1[k]<-sc_fit3$model$emission_probs[1,2]
Em_State2_2[k]<-sc_fit3$model$emission_probs[1,3]
Em_State2_3[k]<-sc_fit3$model$emission_probs[1,4]
Em_State3_0[k]<-sc_fit3$model$emission_probs[2,1]
Em_State3_1[k]<-sc_fit3$model$emission_probs[2,2]
Em_State3_2[k]<-sc_fit3$model$emission_probs[2,3]
Em_State3_3[k]<-sc_fit3$model$emission_probs[2,4]}
if(myorder==6){
  est_trans3<-as.numeric(trans3_A)[c(9,8,7,6,5,4,3,2,1)]
  Em_State1_0[k]<-sc_fit3$model$emission_probs[3,1]
  Em_State1_1[k]<-sc_fit3$model$emission_probs[3,2]
  Em_State1_2[k]<-sc_fit3$model$emission_probs[3,3]
  Em_State1_3[k]<-sc_fit3$model$emission_probs[3,4]
  Em_State2_0[k]<-sc_fit3$model$emission_probs[2,1]
  Em_State2_1[k]<-sc_fit3$model$emission_probs[2,2]
  Em_State2_2[k]<-sc_fit3$model$emission_probs[2,3]
  Em_State2_3[k]<-sc_fit3$model$emission_probs[2,4]
  Em_State3_0[k]<-sc_fit3$model$emission_probs[1,1]
  Em_State3_1[k]<-sc_fit3$model$emission_probs[1,2]
  Em_State3_2[k]<-sc_fit3$model$emission_probs[1,3]
  Em_State3_3[k]<-sc_fit3$model$emission_probs[1,4]}
  Trans3_AA[k]<-est_trans3[1];Trans3_AB[k]<-est_trans3[4]
  Trans3_AC[k]<-est_trans3[7];Trans3_BA[k]<-est_trans3[2]
  Trans3_BB[k]<-est_trans3[5];Trans3_BC[k]<-est_trans3[8]
  Trans3_CA[k]<-est_trans3[3];Trans3_CB[k]<-est_trans3[6]
  Trans3_CC[k]<-est_trans3[9]}
out<-data.frame(df1,AIC1,BIC1,df2,AIC2,BIC2,df3,AIC3,BIC3,
  Trans3_AA,Trans3_AB,Trans3_AC,Trans3_BA,
  Trans3_BB,Trans3_BC,Trans3_CA,Trans3_CB,
  Trans3_CC,Em_State1_0,Em_State1_1,
  Em_State1_2,Em_State1_3,Em_State2_0,
  Em_State2_1,Em_State2_2,Em_State2_3,
  Em_State3_0,Em_State3_1,Em_State3_2,
  Em_State3_3)}
#####
##Simulation##
#####
#Drawrandomseed

```

```

sample(1:10000,1)
set.seed(7166)
Emission_A←c(0.90,0.03,0.03,0.04)
Emission_B←c(0.05,0.45,0.45,0.05)
Emission_C←c(0.05,0.05,0.10,0.80)
Emi←"Scenario_A"
trans←matrix(c(0.7,0.2,0.1,
0.1,.70,0.2,
0.05,0.05,0.90),3,3,byrow=T)
Class←"3Classes"
for(i in c(10,20,30,40,50,100)){
for(k in c(10,20,30,40,50,100)){
myTitle←paste(Emi,"_",Class,"_N",i,"_t",k,"_Sim.R",sep="")
cat(paste("\\n\\n"))
print(myTitle)
Sim1←sim_Latent_Markov3(trans=trans,
initial=c(0.33,0.33,0.34),
len=k,
Emission_A=Emission_A,
Emission_B=Emission_B,
Emission_C=Emission_C,
size=i,
MySamples=500)
save(Sim1,file=paste(Emi,"_",Class,"_N",i,"_t",k,"_Sim.R",sep=""))}
mean(Sim1$AIC3<Sim1$AIC1&Sim1$AIC3<Sim1$AIC2)

```

B.9. Mixture Markov (Correct Number of Latent States)

```

#####
##CreatingSequences##
#####
simSeq←function(trans,initial,length){
init←sample(c(1,2,3,4),1,prob=initial)
for(i in 2:length){ if(init[i-1]==1){init[i]←sample(c(1,2,3,4),1,
prob=trans[1,])} else if(init[i-1]==2){init[i]←sample(c(1,2,3,4),1,
prob=trans[2,])} else if(init[i-1]==3){
init[i]←sample(c(1,2,3,4),1,prob=trans[3,])
} else{init[i]←sample(c(1,2,3,4),1,prob=trans[4,])} return(init)}
#####
#SimFunction#

```



```
#####
sim_Mixture_Markov2<-function(trans1=matrix(c(0.5,0.2,0.2,0.1,
0.8,0.05,0.05,0.1,0.5,0.1,0.2,0.2,0.1,0.1,0.1,0.7),4,4,byrow=TRUE),
initial1=c(.25,.25,.25,.25),trans2=matrix(c(0.1,0.1,0.1,0.7,
0.25,0.25,0.25,0.25,0.25,0.25,0.25,0.25,0.7,0.1,0.1,0.1),4,4,byrow=TRUE),
initial2=c(.25,.25,.25,.25),len=30,size=30,MySamples=10){
  options(warn=-1);require(seqHMM);require(TraMineR)
  require(DySeq);require(psych)
  pb<-txtProgressBar(min=0,max=MySamples,initial=0,char=" ",
width=NA,title,label,style=3,file="")
  df1<-numeric(MySamples);AIC1<-numeric(MySamples)
  BIC1<-numeric(MySamples);df2<-numeric(MySamples)
  AIC2<-numeric(MySamples);BIC2<-numeric(MySamples)
  df3<-numeric(MySamples);AIC3<-numeric(MySamples)
  BIC3<-numeric(MySamples);Trans1_11<-numeric(MySamples)
  Trans1_12<-numeric(MySamples);Trans1_13<-numeric(MySamples)
  Trans1_14<-numeric(MySamples);Trans1_21<-numeric(MySamples)
  Trans1_22<-numeric(MySamples);Trans1_23<-numeric(MySamples)
  Trans1_24<-numeric(MySamples);Trans1_31<-numeric(MySamples)
  Trans1_32<-numeric(MySamples);Trans1_33<-numeric(MySamples)
  Trans1_34<-numeric(MySamples);Trans1_41<-numeric(MySamples)
  Trans1_42<-numeric(MySamples);Trans1_43<-numeric(MySamples)
  Trans1_44<-numeric(MySamples);Trans2_11<-numeric(MySamples)
  Trans2_12<-numeric(MySamples);Trans2_13<-numeric(MySamples)
  Trans2_14<-numeric(MySamples);Trans2_21<-numeric(MySamples)
  Trans2_22<-numeric(MySamples);Trans2_23<-numeric(MySamples)
  Trans2_24<-numeric(MySamples);Trans2_31<-numeric(MySamples)
  Trans2_32<-numeric(MySamples);Trans2_33<-numeric(MySamples)
  Trans2_34<-numeric(MySamples);Trans2_41<-numeric(MySamples)
  Trans2_42<-numeric(MySamples);Trans2_43<-numeric(MySamples)
  Trans2_44<-numeric(MySamples);Kappa<-numeric(MySamples)
  correct_class<-numeric(MySamples);cluster_order<-numeric(MySamples)
  true_trans1<-trans1;true_trans2<-trans2
  for(k in 1:MySamples){x<-simSeqSample(true_trans1,
initial=initial1,len,N=size/2);x[1,1]<-1
x[2,1]<-2;x[3,1]<-3;x[4,1]<-4
y<-simSeqSample(true_trans2,initial=initial2,len,N=size/2)
x<-rbind(x,y)
suppressMessages(my_seq<-seqdef(x,start=1,alphabet=c(1,2,3,4),
labels=c("noSC/DC","SOnly","DConly","SC+DC")))
```

```

sc_init1<-c(.25,.25,.25,.25)
sc_trans1<-matrix(c(.25),nrow=4,ncol=4,byrow=TRUE)
sc_emiss1<-matrix(c(1,0,0,0,
0,1,0,0,
0,0,1,0,
0,0,0,1),nrow=4,ncol=4)
sc_initmod1<-build_hmm(observations=my_seq,
initial_probs=sc_init1,transition_probs=sc_trans1,
emission_probs=sc_emiss1);sc_fit1<-suppressMessages(fit_model(sc_initmod1,
em_step=TRUE,global_step=FALSE,local_step=FALSE,
control_em=list(print_level=0,restart=list(times=0,print_level=0)),
control_global=list(algorithm="NLOPT_GD_MLSL_LDS",maxtime=10,
,print_level=0,
local_opts=list(algorithm="NLOPT_LD_LBFGS",ftol_rel=1e-6,
xtol_rel=1e-4)),
control_local=list(algorithm="NLOPT_LD_LBFGS",maxtime=10,print_level=0),
log_space=FALSE,threads=1));df1[k]<-attr(sc_fit1$model,"df")
AIC1[k]<-AIC(sc_fit1$model);BIC1[k]<-BIC(sc_fit1$model)
mytrans1<-matrix(c(.25),4,4);mytrans2<-matrix(c(.25),4,4)
mymixtrans<-list(mytrans1,mytrans2);myinit1<-c(.25,.25,.25,.25)
myinit2<-c(.25,.25,.25,.25);mymixinit<-list(myinit1,myinit2)
my_mmodel<-build_mmm(my_seq,
n_clusters=2,mymixtrans,mymixinit)
sc_fit2<-suppressMessages(fit_model(my_mmodel,em_step=FALSE,
global_step=TRUE,local_step=TRUE,control_em=list(print_level=0,
restart=list(times=60,print_level=0)),
control_global=list(algorithm="NLOPT_GD_MLSL_LDS",maxtime=10,
print_level=0,local_opts=list(algorithm="NLOPT_LD_LBFGS",
ftol_rel=1e-6,xtol_rel=1e-4)),
control_local=list(algorithm="NLOPT_LD_LBFGS",maxtime=10,
print_level=0),
log_space=FALSE,threads=1))
trans1<-sc_fit2$model$transition_probs$'Cluster1'
trans2<-sc_fit2$model$transition_probs$'Cluster2'
if(sum((trans1-true_trans1)^2)<=sum((trans2-true_trans1)^2))
{cluster_order[k]<-"1-2"}
else{cluster_order[k]<-"2-1"}
if(sum((trans1-true_trans1)^2)<=sum((trans2-true_trans1)^2)){
Trans1_11[k]<-trans1[1,1];Trans1_12[k]<-trans1[1,2];Trans1_13[k]<-trans1[1,3];
Trans1_14[k]<-trans1[1,4];Trans1_21[k]<-trans1[2,1];Trans1_22[k]<-trans1[2,2]

```

```

Trans1_23[k]<-trans1 [2 ,3]; Trans1_24[k]<-trans1 [2 ,4]; Trans1_31[k]<-trans1 [3 ,1]
Trans1_32[k]<-trans1 [3 ,2]; Trans1_33[k]<-trans1 [3 ,3]; Trans1_34[k]<-trans1 [3 ,4]
Trans1_41[k]<-trans1 [4 ,1]; Trans1_42[k]<-trans1 [4 ,2]; Trans1_43[k]<-trans1 [4 ,3]
Trans1_44[k]<-trans1 [4 ,4]; Trans2_11[k]<-trans2 [1 ,1]; Trans2_12[k]<-trans2 [1 ,2]
Trans2_13[k]<-trans2 [1 ,3]; Trans2_14[k]<-trans2 [1 ,4]; Trans2_21[k]<-trans2 [2 ,1]
Trans2_22[k]<-trans2 [2 ,2]; Trans2_23[k]<-trans2 [2 ,3]; Trans2_24[k]<-trans2 [2 ,4]
Trans2_31[k]<-trans2 [3 ,1]; Trans2_32[k]<-trans2 [3 ,2]; Trans2_33[k]<-trans2 [3 ,3]
Trans2_34[k]<-trans2 [3 ,4]; Trans2_41[k]<-trans2 [4 ,1]; Trans2_42[k]<-trans2 [4 ,2]
Trans2_43[k]<-trans2 [4 ,3]; Trans2_44[k]<-trans2 [4 ,4]
post_probs<-summary(sc_fit2$model)$posterior_cluster_probabilities
classi<-post_probs[,1]>post_probs [, 2]; classi [ classi==0]<-2
true_class<-c(rep(1 , size/2) , rep(2 , size/2))
Kappa[k]<-suppressMessages (round (cohen.kappa (data.frame (classi ,
true_class ))$confid [3] , 2))
correct_class [k]<-mean (classi==true_class) } else {
Trans1_11[k]<-trans2 [1 ,1]; Trans1_12[k]<-trans2 [1 ,2]; Trans1_13[k]<-trans2 [1 ,3]
Trans1_14[k]<-trans2 [1 ,4]; Trans1_21[k]<-trans2 [2 ,1]; Trans1_22[k]<-trans2 [2 ,2]
Trans1_23[k]<-trans2 [2 ,3]; Trans1_24[k]<-trans2 [2 ,4]; Trans1_31[k]<-trans2 [3 ,1]
Trans1_32[k]<-trans2 [3 ,2]; Trans1_33[k]<-trans2 [3 ,3]; Trans1_34[k]<-trans2 [3 ,4]
Trans1_41[k]<-trans2 [4 ,1]; Trans1_42[k]<-trans2 [4 ,2]; Trans1_43[k]<-trans2 [4 ,3]
Trans1_44[k]<-trans2 [4 ,4]; Trans2_11[k]<-trans1 [1 ,1]; Trans2_12[k]<-trans1 [1 ,2]
Trans2_13[k]<-trans1 [1 ,3]; Trans2_14[k]<-trans1 [1 ,4]; Trans2_21[k]<-trans1 [2 ,1]
Trans2_22[k]<-trans1 [2 ,2]; Trans2_23[k]<-trans1 [2 ,3]; Trans2_24[k]<-trans1 [2 ,4]
Trans2_31[k]<-trans1 [3 ,1]; Trans2_32[k]<-trans1 [3 ,2]; Trans2_33[k]<-trans1 [3 ,3]
Trans2_34[k]<-trans1 [3 ,4]; Trans2_41[k]<-trans1 [4 ,1]; Trans2_42[k]<-trans1 [4 ,2]
Trans2_43[k]<-trans1 [4 ,3]; Trans2_44[k]<-trans1 [4 ,4];
post_probs<-summary(sc_fit2$model)$posterior_cluster_probabilities
classi<-post_probs[,1]<post_probs [, 2]; classi [ classi==0]<-2
true_class<-c(rep(1 , size/2) , rep(2 , size/2))
Kappa[k]<-suppressMessages (round (cohen.kappa (data.frame (classi ,
true_class ))$confid [3] , 2))
correct_class [k]<-mean (classi==true_class)
} df2[k]<-attr (sc_fit2$model , "df"); AIC2[k]<-AIC (sc_fit2$model)
BIC2[k]<-BIC (sc_fit2$model); mytrans1<-matrix (c (.25) , 4 , 4)
mytrans2<-matrix (c (.25) , 4 , 4); mytrans3<-matrix (c (.25) , 4 , 4)
mymixtrans<-list (mytrans1 , mytrans2 , mytrans3)
myinit1<-c (.25 , .25 , .25 , .25); myinit2<-c (.25 , .25 , .25 , .25)
myinit3<-c (.25 , .25 , .25 , .25); mymixinit<-list (myinit1 , myinit2 , myinit3)
my_mmodel<-build_mmm (my_seq , n_clusters=3 , mymixtrans , mymixinit)
sc_fit3<-suppressMessages (fit_model (my_mmodel , em_step=FALSE,

```

```

global_step=TRUE, local_step=TRUE, control_em=list(print_level=0,
restart=list(times=10, print_level=0)),
control_global=list(algorithm="NLOPT_GD_MLSL_LDS", maxtime=10,
print_level=0, local_opts=
list(algorithm="NLOPT_LD_LBFGS", ftol_rel=1e-6, xtol_rel=1e-4)),
control_local=list(algorithm="NLOPT_LD_LBFGS", maxtime=10
, print_level=0),
log_space=FALSE, threads=1)); df3[k]<-attr(sc_fit3$model, "df")
AIC3[k]<-AIC(sc_fit3$model); BIC3[k]<-BIC(sc_fit3$model)
print(cat(paste("\nlen", len))); print(cat(paste("size", size)))
setTxtProgressBar(pb, k, title=NULL, label=NULL)}
options(warn=0)
out<-data.frame(df1, AIC1, BIC1, df2, AIC2, BIC2, df3, AIC3
, BIC3, Trans1_11, Trans1_12,
Trans1_13, Trans1_14, Trans1_21, Trans1_22, Trans1_23,
Trans1_24, Trans1_31, Trans1_32, Trans1_33, Trans1_34,
Trans1_41, Trans1_42, Trans1_43, Trans1_44,
Trans2_11, Trans2_12, Trans2_13, Trans2_14, Trans2_21,
Trans2_22, Trans2_23, Trans2_24, Trans2_31, Trans2_32,
Trans2_33, Trans2_34, Trans2_41, Trans2_42, Trans2_43,
Trans2_44, Kappa, correct_class, cluster_order)}
#####
## Simulation ##
#####
set.seed(3899)
# Scenario A
Class<-"TwoClass_Scenario_A"
trans1<-matrix(c(0.05, 0.05, 0.05, 0.85,
0.05, 0.05, 0.85, 0.05,
0.05, 0.85, 0.05, 0.05,
0.85, 0.05, 0.05, 0.05), 4, 4, byrow=TRUE)
trans2<-matrix(c(0.70, 0.2, 0.05, 0.05,
0.05, 0.7, 0.20, 0.05,
0.05, 0.05, 0.7, 0.20,
0.01, 0.01, 0.01, 0.97), 4, 4, byrow=TRUE)
for(i in c(10, 20, 30, 50, 100)){
for(k in c(10, 20, 30, 40, 50, 100)){
myTitle<-paste(Class, "_N", i, "_t", k, "_Sim.R", sep="")
cat(paste("\n\n"))
print(myTitle)

```

```

Sim1<-sim_Mixture_Markov2( trans1=trans1 ,
trans2=trans2 ,
initial1=c(.25 ,.25 ,.25 ,.25) ,
initial2=c(.25 ,.25 ,.25 ,.25) ,
len=k,
size=i ,
MySamples=500)
save(Sim1, file=paste(Class , "_N" , i , "_t" , k , "_Sim.R" , sep=""))}}
#####
###3 latent classes###
#####
diff3<-function(size){ sizef<-floor(size/3)
if(size%%3==0){a<-sizef;b<-sizef;c<-sizef
} else if(size%%3==1){a<-sizef+1
b<-sizef;c<-sizef;} else {
a<-sizef+1;b<-sizef+1;c<-sizef
} out<-c(a,b,c)}
sim_Mixture_Markov3<-function(trans1=matrix(c(0.5,0.2,0.2,0.1,
0.8,0.05,0.05,0.1,0.5,0.1,0.2,0.2,0.1,0.1,0.1,0.7),4,4,byrow=TRUE),
initial1=c(.25,.25,.25,.25),trans2=matrix(c(0.1,0.1,0.1,0.7,
0.25,0.25,0.25,0.25,0.25,0.25,0.25,0.25,0.25,
0.7,0.1,0.1,0.1),4,4,byrow=TRUE),initial2=c(.25,.25,.25,.25),
trans3=matrix(c(0.1,0.1,0.1,0.7,0.25,0.25,0.25,0.25,0.1,0,0,
0.7,0.1,0.1,0.1),4,4,byrow=TRUE),initial3=c(.25,.25,.25,.25),
len=30,size=30,MySamples=10){
options(warn=-1);require(seqHMM);require(TraMineR)
require(DySeq);require(psych)
pb<-txtProgressBar(min=0,max=MySamples,initial=0,char="=",
width=NA,title,label,style=3,file="")
df1<-rep(NA,MySamples);AIC1<-rep(NA,MySamples)
BIC1<-rep(NA,MySamples);df2<-rep(NA,MySamples)
AIC2<-rep(NA,MySamples);BIC2<-rep(NA,MySamples)
df3<-rep(NA,MySamples);AIC3<-rep(NA,MySamples)
BIC3<-rep(NA,MySamples);Trans1_11<-rep(NA,MySamples)
Trans1_12<-rep(NA,MySamples);Trans1_13<-rep(NA,MySamples)
Trans1_14<-rep(NA,MySamples);Trans1_21<-rep(NA,MySamples)
Trans1_22<-rep(NA,MySamples);Trans1_23<-rep(NA,MySamples)
Trans1_24<-rep(NA,MySamples);Trans1_31<-rep(NA,MySamples)
Trans1_32<-rep(NA,MySamples);Trans1_33<-rep(NA,MySamples)
Trans1_34<-rep(NA,MySamples);Trans1_41<-rep(NA,MySamples)

```

```

Trans1_42<-rep (NA, MySamples) ; Trans1_43<-rep (NA, MySamples)
Trans1_44<-rep (NA, MySamples) ; Trans2_11<-rep (NA, MySamples)
Trans2_12<-rep (NA, MySamples) ; Trans2_13<-rep (NA, MySamples)
Trans2_14<-rep (NA, MySamples) ; Trans2_21<-rep (NA, MySamples)
Trans2_22<-rep (NA, MySamples) ; Trans2_23<-rep (NA, MySamples)
Trans2_24<-rep (NA, MySamples) ; Trans2_31<-rep (NA, MySamples)
Trans2_32<-rep (NA, MySamples) ; Trans2_33<-rep (NA, MySamples)
Trans2_34<-rep (NA, MySamples) ; Trans2_41<-rep (NA, MySamples)
Trans2_42<-rep (NA, MySamples) ; Trans2_43<-rep (NA, MySamples)
Trans2_44<-rep (NA, MySamples) ; Trans3_11<-rep (NA, MySamples)
Trans3_12<-rep (NA, MySamples) ; Trans3_13<-rep (NA, MySamples)
Trans3_14<-rep (NA, MySamples) ; Trans3_21<-rep (NA, MySamples)
Trans3_22<-rep (NA, MySamples) ; Trans3_23<-rep (NA, MySamples)
Trans3_24<-rep (NA, MySamples) ; Trans3_31<-rep (NA, MySamples)
Trans3_32<-rep (NA, MySamples) ; Trans3_33<-rep (NA, MySamples)
Trans3_34<-rep (NA, MySamples) ; Trans3_41<-rep (NA, MySamples)
Trans3_42<-rep (NA, MySamples) ; Trans3_43<-rep (NA, MySamples)
Trans3_44<-rep (NA, MySamples) ; cluster_order<-numeric (MySamples)
Kappa<-numeric (MySamples) ; correct_class<-numeric (MySamples)
ErrorCatch1<-character (MySamples) ; ErrorCatch2<-character (MySamples)
ErrorCatch3<-character (MySamples) ; true_trans1<-trans1
true_trans2<-trans2 ; true_trans3<-trans3
size3<-diff3 (size) ; for (k in 1:MySamples) {
x<-simSeqSample (true_trans1 , initial=initial , len ,N=size3 [1])
x [1,1]<-1 ; x [2,1]<-2 ; x [3,1]<-3 ; x [4,1]<-4
y<-simSeqSample (true_trans2 , initial=initial2 , len ,N=size3 [2])
w<-simSeqSample (true_trans3 , initial=initial3 , len ,N=size3 [3])
x<-rbind (x, y, w) ; suppressMessages (my_seq<-seqdef (x,
start=1, alphabet=c (1, 2, 3, 4), labels=c ("noSC/DC" , "SConly" ,
"DConly" , "SC+DC" )))
sc_init1<-c (.25 ,.25 ,.25 ,.25)
sc_trans1<-matrix (c (.25) , nrow=4, ncol=4, byrow=TRUE)
sc_emiss1<-matrix (c (1, 0, 0, 0,
0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1) , nrow=4, ncol=4)
sc_initmod1<-build_hmm (observations=my_seq ,
initial_probs=sc_init1 , transition_probs=sc_trans1 ,
emission_probs=sc_emiss1) ; tryCatch ({ suppressMessages (sc_fit1<-
fit_model (sc_initmod1 , em_step=TRUE, global_step=FALSE,
local_step=FALSE, control_em=list (print_level=0,
restart=list (times=0, print_level=0)) ,

```

```

control_global=list(algorithm="NLOPT_GD_MLSL_LDS",
maxtime=10,print_level=0,
local_opts=list(algorithm="NLOPT_LD_LBFGS",
ftol_rel=1e-6,xtol_rel=1e-4)),
control_local=list(algorithm="NLOPT_LD_LBFGS",maxtime=10,
print_level=0),log_space=FALSE,threads=1)),error=function(e)
{ErrorCatch1[k]<-e})if(nchar(ErrorCatch1[k])==0)
{df1[k]<-attr(sc_fit1$model,"df")
AIC1[k]<-AIC(sc_fit1$model);BIC1[k]<-BIC(sc_fit1$model)}
mytrans1<-matrix(c(.25),4,4);mytrans2<-matrix(c(.25),4,4)
mymixtrans<-list(mytrans1,mytrans2);myinit1<-c(.25,.25,.25,.25)
myinit2<-c(.25,.25,.25,.25);mymixinit<-list(myinit1,myinit2)
my_mmodel<-build_mmm(my_seq,n_clusters=2,
mymixtrans,mymixinit)
tryCatch({suppressMessages(sc_fit2<-fit_model(my_mmodel,
em_step=FALSE,global_step=TRUE,
local_step=TRUE,control_em=
list(print_level=0,restart=list(times=10,
print_level=0)),control_global=list(algorithm=
"NLOPT_GD_MLSL_LDS",maxtime=10,print_level=0,
local_opts=list(algorithm="NLOPT_LD_LBFGS",
ftol_rel=1e-6,xtol_rel=1e-4)),
control_local=list(algorithm="NLOPT_LD_LBFGS",maxtime=10,
print_level=0),log_space=FALSE,threads=1))),
error=function(e){ErrorCatch2[k]<-e})
if(nchar(ErrorCatch2[k])==0){df2[k]<-attr(sc_fit2$model,"df")
AIC2[k]<-AIC(sc_fit2$model);BIC2[k]<-BIC(sc_fit2$model)}
mytrans1<-matrix(c(.25),4,4);mytrans2<-matrix(c(.25),4,4)
mytrans3<-matrix(c(.25),4,4);mymixtrans<-list(mytrans1,mytrans2,mytrans3)
myinit1<-c(.25,.25,.25,.25);myinit2<-c(.25,.25,.25,.25)
myinit3<-c(.25,.25,.25,.25);mymixinit<-list(myinit1,myinit2,myinit3)
my_mmodel<-build_mmm(my_seq,n_clusters=3,mymixtrans,mymixinit)
tryCatch({suppressMessages(sc_fit3<-fit_model(my_mmodel,
em_step=FALSE,global_step=TRUE,local_step=TRUE,
control_em=list(print_level=0,restart=list(times=10,print_level=0)),
control_global=list(algorithm="NLOPT_GD_MLSL_LDS",maxtime=10,print_level=0,
local_opts=list(algorithm="NLOPT_LD_LBFGS",ftol_rel=1e-6,xtol_rel=1e-4)),
control_local=list(algorithm="NLOPT_LD_LBFGS",maxtime=10,print_level=0),
log_space=FALSE,threads=1))),error=function(e){ErrorCatch2[k]<-e})
if(nchar(ErrorCatch3[k])==0){

```

```

df3[k]<-attr(sc_fit3$model, "df");AIC3[k]<-AIC(sc_fit3$model)
BIC3[k]<-BIC(sc_fit3$model)
trans1<-sc_fit3$model$transition_probs$'Cluster1'
trans2<-sc_fit3$model$transition_probs$'Cluster2'
trans3<-sc_fit3$model$transition_probs$'Cluster3'
boundmatrix<-rbind(as.vector(trans1),as.vector(trans2),
as.vector(trans3))
ABC<-sum((rbind(as.vector(true_trans1),as.vector(true_trans2),
as.vector(true_trans3))-boundmatrix)^2)
ACB<-sum((rbind(as.vector(true_trans1),as.vector(true_trans3),
as.vector(true_trans2))-boundmatrix)^2)
BAC<-sum((rbind(as.vector(true_trans2),as.vector(true_trans1),
as.vector(true_trans3))-boundmatrix)^2)
BCA<-sum((rbind(as.vector(true_trans2),as.vector(true_trans3),
as.vector(true_trans1))-boundmatrix)^2)
CAB<-sum((rbind(as.vector(true_trans3),as.vector(true_trans1),
as.vector(true_trans2))-boundmatrix)^2)
CBA<-sum((rbind(as.vector(true_trans3),as.vector(true_trans2),
as.vector(true_trans1))-boundmatrix)^2)
fall<-c(ABC,ACB,BAC,BCA,CAB,CBA);myorder<-which(fall==min(fall))
cluster_order[k]<-myorder
post_probs<-summary(sc_fit3$model)$posterior_cluster_probabilities
true_class<-c(rep(1,size3[1]),rep(2,size3[2]),rep(3,size3[3]))
if(myorder==1){
#Order:ABC
Trans1_11[k]<-trans1[1,1];Trans1_12[k]<-trans1[1,2];Trans1_13[k]<-trans1[1,3]
Trans1_14[k]<-trans1[1,4];Trans1_21[k]<-trans1[2,1];Trans1_22[k]<-trans1[2,2]
Trans1_23[k]<-trans1[2,3];Trans1_24[k]<-trans1[2,4];Trans1_31[k]<-trans1[3,1]
Trans1_32[k]<-trans1[3,2];Trans1_33[k]<-trans1[3,3];Trans1_34[k]<-trans1[3,4]
Trans1_41[k]<-trans1[4,1];Trans1_42[k]<-trans1[4,2];Trans1_43[k]<-trans1[4,3]
Trans1_44[k]<-trans1[4,4];Trans2_11[k]<-trans2[1,1];Trans2_12[k]<-trans2[1,2]
Trans2_13[k]<-trans2[1,3];Trans2_14[k]<-trans2[1,4];Trans2_21[k]<-trans2[2,1]
Trans2_22[k]<-trans2[2,2];Trans2_23[k]<-trans2[2,3];Trans2_24[k]<-trans2[2,4]
Trans2_31[k]<-trans2[3,1];Trans2_32[k]<-trans2[3,2];Trans2_33[k]<-trans2[3,3]
Trans2_34[k]<-trans2[3,4];Trans2_41[k]<-trans2[4,1];Trans2_42[k]<-trans2[4,2]
Trans2_43[k]<-trans2[4,3];Trans2_44[k]<-trans2[4,4];Trans3_11[k]<-trans3[1,1]
Trans3_12[k]<-trans3[1,2];Trans3_13[k]<-trans3[1,3];Trans3_14[k]<-trans3[1,4]
Trans3_21[k]<-trans3[2,1];Trans3_22[k]<-trans3[2,2];Trans3_23[k]<-trans3[2,3]
Trans3_24[k]<-trans3[2,4];Trans3_31[k]<-trans3[3,1];Trans3_32[k]<-trans3[3,2]
Trans3_33[k]<-trans3[3,3];Trans3_34[k]<-trans3[3,4];Trans3_41[k]<-trans3[4,1]

```



```

Trans3_42[k]<-trans3 [4 ,2]; Trans3_43[k]<-trans3 [4 ,3]; Trans3_44[k]<-trans3 [4 ,4]
classi<-numeric( size )
cluster1<-post_probs[,1]>post_probs [,2]&post_probs[,1]>post_probs [,3]
classi [ cluster1==1]<-1
cluster2<-post_probs[,2]>post_probs [,1]&post_probs[,2]>post_probs [,3]
classi [ cluster2==1]<-2
cluster3<-post_probs[,3]>post_probs [,2]&post_probs[,3]>post_probs [,1]
classi [ cluster3==1]<-3
Kappa[k]<-suppressMessages (round( cohen . kappa( data . frame( classi
, true_class ))$confid [3] ,2)); correct_class [k]<-mean( classi==true_class )
} else if (myorder==2){
Trans1_11[k]<-trans1 [1 ,1]; Trans1_12[k]<-trans1 [1 ,2]; Trans1_13[k]<-trans1 [1 ,3]
Trans1_14[k]<-trans1 [1 ,4]; Trans1_21[k]<-trans1 [2 ,1]; Trans1_22[k]<-trans1 [2 ,2]
Trans1_23[k]<-trans1 [2 ,3]; Trans1_24[k]<-trans1 [2 ,4]; Trans1_31[k]<-trans1 [3 ,1]
Trans1_32[k]<-trans1 [3 ,2]; Trans1_33[k]<-trans1 [3 ,3]; Trans1_34[k]<-trans1 [3 ,4]
Trans1_41[k]<-trans1 [4 ,1]; Trans1_42[k]<-trans1 [4 ,2]; Trans1_43[k]<-trans1 [4 ,3]
Trans1_44[k]<-trans1 [4 ,4]; Trans2_11[k]<-trans3 [1 ,1]; Trans2_12[k]<-trans3 [1 ,2]
Trans2_13[k]<-trans3 [1 ,3]; Trans2_14[k]<-trans3 [1 ,4]; Trans2_21[k]<-trans3 [2 ,1]
Trans2_22[k]<-trans3 [2 ,2]; Trans2_23[k]<-trans3 [2 ,3]; Trans2_24[k]<-trans3 [2 ,4]
Trans2_31[k]<-trans3 [3 ,1]; Trans2_32[k]<-trans3 [3 ,2]; Trans2_33[k]<-trans3 [3 ,3]
Trans2_34[k]<-trans3 [3 ,4]; Trans2_41[k]<-trans3 [4 ,1]; Trans2_42[k]<-trans3 [4 ,2]
Trans2_43[k]<-trans3 [4 ,3]; Trans2_44[k]<-trans3 [4 ,4]; Trans3_11[k]<-trans2 [1 ,1]
Trans3_12[k]<-trans2 [1 ,2]; Trans3_13[k]<-trans2 [1 ,3]; Trans3_14[k]<-trans2 [1 ,4]
Trans3_21[k]<-trans2 [2 ,1]; Trans3_22[k]<-trans2 [2 ,2]; Trans3_23[k]<-trans2 [2 ,3]
Trans3_24[k]<-trans2 [2 ,4]; Trans3_31[k]<-trans2 [3 ,1]; Trans3_32[k]<-trans2 [3 ,2]
Trans3_33[k]<-trans2 [3 ,3]; Trans3_34[k]<-trans2 [3 ,4]; Trans3_41[k]<-trans2 [4 ,1]
Trans3_42[k]<-trans2 [4 ,2]; Trans3_43[k]<-trans2 [4 ,3]; Trans3_44[k]<-trans2 [4 ,4]
classi<-numeric( size )
cluster1<-post_probs[,1]>post_probs [,2]&post_probs[,1]>post_probs [,3]
cluster3<-post_probs[,2]>post_probs [,1]&post_probs[,2]>post_probs [,3]
cluster2<-post_probs[,3]>post_probs [,2]&post_probs[,3]>post_probs [,1]
classi [ cluster1==1]<- 1; classi [ cluster2==1]<- 2; classi [ cluster3==1]<-3
Kappa[k]<-suppressMessages (round( cohen . kappa( data . frame( classi ,
true_class ))$confid [3] ,2)); correct_class [k]<-mean( classi==true_class )
} else if (myorder==3){
Trans2_11[k]<-trans1 [1 ,1]; Trans2_12[k]<-trans1 [1 ,2]; Trans2_13[k]<-trans1 [1 ,3]
Trans2_14[k]<-trans1 [1 ,4]; Trans2_21[k]<-trans1 [2 ,1]; Trans2_22[k]<-trans1 [2 ,2]
Trans2_23[k]<-trans1 [2 ,3]; Trans2_24[k]<-trans1 [2 ,4]; Trans2_31[k]<-trans1 [3 ,1]
Trans2_32[k]<-trans1 [3 ,2]; Trans2_33[k]<-trans1 [3 ,3]; Trans2_34[k]<-trans1 [3 ,4]
Trans2_41[k]<-trans1 [4 ,1]; Trans2_42[k]<-trans1 [4 ,2]; Trans2_43[k]<-trans1 [4 ,3]

```

```

Trans2_44[k]<-trans1 [4 ,4]; Trans1_11[k]<-trans2 [1 ,1]; Trans1_12[k]<-trans2 [1 ,2]
Trans1_13[k]<-trans2 [1 ,3]; Trans1_14[k]<-trans2 [1 ,4]; Trans1_21[k]<-trans2 [2 ,1]
Trans1_22[k]<-trans2 [2 ,2]; Trans1_23[k]<-trans2 [2 ,3]; Trans1_24[k]<-trans2 [2 ,4]
Trans1_31[k]<-trans2 [3 ,1]; Trans1_32[k]<-trans2 [3 ,2]; Trans1_33[k]<-trans2 [3 ,3]
Trans1_34[k]<-trans2 [3 ,4]; Trans1_41[k]<-trans2 [4 ,1]; Trans1_42[k]<-trans2 [4 ,2]
Trans1_43[k]<-trans2 [4 ,3]; Trans1_44[k]<-trans2 [4 ,4]; Trans3_11[k]<-trans3 [1 ,1]
Trans3_12[k]<-trans3 [1 ,2]; Trans3_13[k]<-trans3 [1 ,3]; Trans3_14[k]<-trans3 [1 ,4]
Trans3_21[k]<-trans3 [2 ,1]; Trans3_22[k]<-trans3 [2 ,2]; Trans3_23[k]<-trans3 [2 ,3]
Trans3_24[k]<-trans3 [2 ,4]; Trans3_31[k]<-trans3 [3 ,1]; Trans3_32[k]<-trans3 [3 ,2]
Trans3_33[k]<-trans3 [3 ,3]; Trans3_34[k]<-trans3 [3 ,4]; Trans3_41[k]<-trans3 [4 ,1]
Trans3_42[k]<-trans3 [4 ,2]; Trans3_43[k]<-trans3 [4 ,3]; Trans3_44[k]<-trans3 [4 ,4]
classi<-numeric( size )
cluster2<-post_probs [,1]>post_probs [,2]&post_probs [,1]>post_probs [,3]
cluster1<-post_probs [,2]>post_probs [,1]&post_probs [,2]>post_probs [,3]
cluster3<-post_probs [,3]>post_probs [,2]&post_probs [,3]>post_probs [,1]
classi [ cluster1==1]<-1; classi [ cluster2==1]<-2; classi [ cluster3==1]<-3
Kappa[k]<-suppressMessages (round (cohen . kappa ( data . frame ( classi ,
true_class ))$confid [3] ,2)); correct_class [k]<-mean ( classi==true_class )
} else if (myorder==4){
Trans1_11[k]<-trans2 [1 ,1]; Trans1_12[k]<-trans2 [1 ,2]; Trans1_13[k]<-trans2 [1 ,3]
Trans1_14[k]<-trans2 [1 ,4]; Trans1_21[k]<-trans2 [2 ,1]; Trans1_22[k]<-trans2 [2 ,2]
Trans1_23[k]<-trans2 [2 ,3]; Trans1_24[k]<-trans2 [2 ,4]; Trans1_31[k]<-trans2 [3 ,1]
Trans1_32[k]<-trans2 [3 ,2]; Trans1_33[k]<-trans2 [3 ,3]; Trans1_34[k]<-trans2 [3 ,4]
Trans1_41[k]<-trans2 [4 ,1]; Trans1_42[k]<-trans2 [4 ,2]; Trans1_43[k]<-trans2 [4 ,3]
Trans1_44[k]<-trans2 [4 ,4]; Trans2_11[k]<-trans3 [1 ,1]; Trans2_12[k]<-trans3 [1 ,2]
Trans2_13[k]<-trans3 [1 ,3]; Trans2_14[k]<-trans3 [1 ,4]; Trans2_21[k]<-trans3 [2 ,1]
Trans2_22[k]<-trans3 [2 ,2]; Trans2_23[k]<-trans3 [2 ,3]; Trans2_24[k]<-trans3 [2 ,4]
Trans2_31[k]<-trans3 [3 ,1]; Trans2_32[k]<-trans3 [3 ,2]; Trans2_33[k]<-trans3 [3 ,3]
Trans2_34[k]<-trans3 [3 ,4]; Trans2_41[k]<-trans3 [4 ,1]; Trans2_42[k]<-trans3 [4 ,2]
Trans2_43[k]<-trans3 [4 ,3]; Trans2_44[k]<-trans3 [4 ,4]; Trans3_11[k]<-trans1 [1 ,1]
Trans3_12[k]<-trans1 [1 ,2]; Trans3_13[k]<-trans1 [1 ,3]; Trans3_14[k]<-trans1 [1 ,4]
Trans3_21[k]<-trans1 [2 ,1]; Trans3_22[k]<-trans1 [2 ,2]; Trans3_23[k]<-trans1 [2 ,3]
Trans3_24[k]<-trans1 [2 ,4]; Trans3_31[k]<-trans1 [3 ,1]; Trans3_32[k]<-trans1 [3 ,2]
Trans3_33[k]<-trans1 [3 ,3]; Trans3_34[k]<-trans1 [3 ,4]; Trans3_41[k]<-trans1 [4 ,1]
Trans3_42[k]<-trans1 [4 ,2]; Trans3_43[k]<-trans1 [4 ,3]; Trans3_44[k]<-trans1 [4 ,4]

classi<-numeric( size )
cluster2<-post_probs [,1]>post_probs [,2]&post_probs [,1]>post_probs [,3]
cluster3<-post_probs [,2]>post_probs [,1]&post_probs [,2]>post_probs [,3]
cluster1<-post_probs [,3]>post_probs [,2]&post_probs [,3]>post_probs [,1]

```

```

classi [ cluster1==1]<-1; classi [ cluster2==1]<-2; classi [ cluster3==1]<-3
Kappa[k]<-suppressMessages( round( cohen.kappa( data.frame( classi ,
true_class ))$confid [3] ,2))
correct_class [k]<-mean( classi==true_class )
} else if(myorder==5){
Trans1_11[k]<-trans3 [1 ,1]; Trans1_12[k]<-trans3 [1 ,2]; Trans1_13[k]<-trans3 [1 ,3]
Trans1_14[k]<-trans3 [1 ,4]; Trans1_21[k]<-trans3 [2 ,1]; Trans1_22[k]<-trans3 [2 ,2]
Trans1_23[k]<-trans3 [2 ,3]; Trans1_24[k]<-trans3 [2 ,4]; Trans1_31[k]<-trans3 [3 ,1]
Trans1_32[k]<-trans3 [3 ,2]; Trans1_33[k]<-trans3 [3 ,3]; Trans1_34[k]<-trans3 [3 ,4]
Trans1_41[k]<-trans3 [4 ,1]; Trans1_42[k]<-trans3 [4 ,2]; Trans1_43[k]<-trans3 [4 ,3]
Trans1_44[k]<-trans3 [4 ,4]; Trans2_11[k]<-trans1 [1 ,1]; Trans2_12[k]<-trans1 [1 ,2]
Trans2_13[k]<-trans1 [1 ,3]; Trans2_14[k]<-trans1 [1 ,4]; Trans2_21[k]<-trans1 [2 ,1]
Trans2_22[k]<-trans1 [2 ,2]; Trans2_23[k]<-trans1 [2 ,3]; Trans2_24[k]<-trans1 [2 ,4]
Trans2_31[k]<-trans1 [3 ,1]; Trans2_32[k]<-trans1 [3 ,2]; Trans2_33[k]<-trans1 [3 ,3]
Trans2_34[k]<-trans1 [3 ,4]; Trans2_41[k]<-trans1 [4 ,1]; Trans2_42[k]<-trans1 [4 ,2]
Trans3_12[k]<-trans2 [1 ,2]; Trans3_13[k]<-trans2 [1 ,3]; Trans3_14[k]<-trans2 [1 ,4]
Trans3_21[k]<-trans2 [2 ,1]; Trans3_22[k]<-trans2 [2 ,2]; Trans3_23[k]<-trans2 [2 ,3]
Trans3_24[k]<-trans2 [2 ,4]; Trans3_31[k]<-trans2 [3 ,1]; Trans3_32[k]<-trans2 [3 ,2]
Trans3_33[k]<-trans2 [3 ,3]; Trans3_34[k]<-trans2 [3 ,4]; Trans3_41[k]<-trans2 [4 ,1]
Trans3_42[k]<-trans2 [4 ,2]; Trans3_43[k]<-trans2 [4 ,3]; Trans3_44[k]<-trans2 [4 ,4]
classi<-numeric( size )
cluster3<-post_probs [ ,1] > post_probs [ ,2] & post_probs [ ,1] > post_probs [ ,3]
cluster1<-post_probs [ ,2] > post_probs [ ,1] & post_probs [ ,2] > post_probs [ ,3]
cluster2<-post_probs [ ,3] > post_probs [ ,2] & post_probs [ ,3] > post_probs [ ,1]
classi [ cluster1==1]<-1; classi [ cluster2==1]<-2; classi [ cluster3==1]<-3
Kappa[k]<-suppressMessages( round( cohen.kappa( data.frame( classi ,
true_class ))$confid [3] ,2)) correct_class [k]<-mean( classi==true_class )
} else {
Trans1_11[k]<-trans3 [1 ,1]; Trans1_12[k]<-trans3 [1 ,2]; Trans1_13[k]<-trans3 [1 ,3]
Trans1_14[k]<-trans3 [1 ,4]; Trans1_21[k]<-trans3 [2 ,1]; Trans1_22[k]<-trans3 [2 ,2]
Trans1_23[k]<-trans3 [2 ,3]; Trans1_24[k]<-trans3 [2 ,4]; Trans1_31[k]<-trans3 [3 ,1]
Trans1_32[k]<-trans3 [3 ,2]; Trans1_33[k]<-trans3 [3 ,3]; Trans1_34[k]<-trans3 [3 ,4]
Trans1_41[k]<-trans3 [4 ,1]; Trans1_42[k]<-trans3 [4 ,2]; Trans1_43[k]<-trans3 [4 ,3]
Trans1_44[k]<-trans3 [4 ,4]; Trans2_11[k]<-trans2 [1 ,1]; Trans2_12[k]<-trans2 [1 ,2]
Trans2_13[k]<-trans2 [1 ,3]; Trans2_14[k]<-trans2 [1 ,4]; Trans2_21[k]<-trans2 [2 ,1]
Trans2_22[k]<-trans2 [2 ,2]; Trans2_23[k]<-trans2 [2 ,3]; Trans2_24[k]<-trans2 [2 ,4]
Trans2_31[k]<-trans2 [3 ,1]; Trans2_32[k]<-trans2 [3 ,2]; Trans2_33[k]<-trans2 [3 ,3]
Trans2_34[k]<-trans2 [3 ,4]; Trans2_41[k]<-trans2 [4 ,1]; Trans2_42[k]<-trans2 [4 ,2]
Trans2_43[k]<-trans2 [4 ,3]; Trans2_44[k]<-trans2 [4 ,4]; Trans3_11[k]<-trans1 [1 ,1]
Trans3_12[k]<-trans1 [1 ,2]; Trans3_13[k]<-trans1 [1 ,3]; Trans3_14[k]<-trans1 [1 ,4]

```

```

Trans3_21[k]<-trans1 [2 ,1];Trans3_22[k]<-trans1 [2 ,2];Trans3_23[k]<-trans1 [2 ,3]
Trans3_24[k]<-trans1 [2 ,4];Trans3_31[k]<-trans1 [3 ,1];Trans3_32[k]<-trans1 [3 ,2]
Trans3_33[k]<-trans1 [3 ,3];Trans3_34[k]<-trans1 [3 ,4];Trans3_41[k]<-trans1 [4 ,1]
Trans3_42[k]<-trans1 [4 ,2];Trans3_43[k]<-trans1 [4 ,3];Trans3_44[k]<-trans1 [4 ,4]
classi<-numeric( size )
cluster3<-post_probs[,1]>post_probs [,2]&post_probs[,1]>post_probs [,3]
cluster2<-post_probs[,2]>post_probs [,1]&post_probs[,2]>post_probs [,3]
cluster1<-post_probs[,3]>post_probs [,2]&post_probs[,3]>post_probs [,1]
classi [ cluster1==1]<-1;classi [ cluster2==1]<-2;classi [ cluster3==1]<-3

Kappa[k]<-suppressMessages (round( cohen . kappa( data . frame( classi ,
      true_class ))$confid [3] ,2))
correct_class [k]<-mean( classi==true_class )
}
}
setTxtProgressBar (pb,k , title=NULL, label=NULL)
} options (warn=0);out<-data . frame( df1 ,AIC1 ,BIC1 ,df2 ,AIC2 ,BIC2 ,df3 ,
AIC3 ,BIC3 ,Trans1_11,Trans1_12,Trans1_13,Trans1_14,Trans1_21 ,
Trans1_22,Trans1_23,Trans1_24,Trans1_31,Trans1_32,Trans1_33 ,
Trans1_34,Trans1_41,Trans1_42,Trans1_43,Trans1_44,Trans2_11,Trans2_12 ,
Trans2_13,Trans2_14,Trans2_21,Trans2_22,Trans2_23,Trans2_24,Trans2_31 ,
Trans2_32,Trans2_33,Trans2_34,Trans2_41,Trans2_42,Trans2_43,Trans2_44 ,
Trans3_11,Trans3_12,Trans3_13,Trans3_14,Trans3_21,Trans3_22,Trans3_23 ,
Trans3_24,Trans3_31,Trans3_32,Trans3_33,Trans3_34,Trans3_41,Trans3_42 ,
Trans3_43,Trans3_44 ,
Kappa , correct_class , cluster_order
)
}

#####
#TestderSingelConditionSimulation#
#####

test_out<-sim_Mixture_Markov3(MySamples=2,len=5,size=20)
test_out

#####
##Simulation##
#####

```

```
#Drawrandomseed
sample(1:10000,1)

set.seed(8200)

#ScenarioA

Class<-"ThreeClass_Scenario_A"

trans1<-matrix(c(1.00,0.00,0.00,0.00,
0.20,0.70,0.05,0.05,
0.05,0.25,0.65,0.05,
0.05,0.05,0.25,0.65),4,4,byrow=TRUE)

trans2<-matrix(c(0.10,0.05,0.05,0.80,
0.10,0.05,0.05,0.80,
0.00,0.00,1.00,0.00,
0.10,0.05,0.05,0.80),4,4,byrow=TRUE)

trans3=matrix(c(0.7,0.1,0.1,0.1,
0.7,0,0.3,0,
0.7,0.3,0,0,
0,0,0,1),4,4,byrow=TRUE)

for(i in c(10,20,30,40,50,100)){
for(k in c(10,20,30,40,50,100)){

myTitle<-paste(Class,"_N",i,"_t",k,"_Sim.R",sep="")
cat(paste("\n\n"))
print(myTitle)

if(i==10&k==10){
} else {

Sim1<-sim_Mixture_Markov3(trans1=trans1,
trans2=trans2,
trans3=trans3,
initial1=c(.25,.25,.25,.25),
initial2=c(.25,.25,.25,.25),
initial3=c(.25,.25,.25,.25),
```

```
len=k,
size=i,
MySamples=500)
```

```
save(Sim1, file=paste(Class, "_N", i, "_t", k, "_Sim.R", sep=""))}}
```

B.10. OM-Distances (Correct Number of Latent States)

```
sim_OM2<-function(trans1=matrix(c(0.05,0.05,0.05,0.85,
0.05,0.05,0.85,0.05,
0.05,0.85,0.05,0.05,
0.85,0.05,0.05,0.05),4,4,byrow=TRUE),
initial1=c(.25,.25,.25,.25),
trans2=matrix(c(0.70,0.2,0.05,0.05,
0.05,0.7,0.20,0.05,
0.05,0.05,0.7,0.20,
0.01,0.01,0.01,0.97),4,4,byrow=TRUE),
initial2=c(.25,.25,.25,.25),
len=30,
size=30,
MySamples=10){

  options(warn=-1)

  require(seqHMM)
  require(TraMineR)
  require(DySeq)
  require(psych)
  require(cluster)
  require(fpc)

  pb<-txtProgressBar(min=0,max=MySamples,initial=0,char=" ",
width=NA,title,label,style=3,file="")

  Sil2<-numeric(MySamples); Sil3<-numeric(MySamples)
  Sil2_min<-numeric(MySamples); Sil3_min<-numeric(MySamples)
  wss1<-numeric(MySamples); wss2<-numeric(MySamples)
  wss3<-numeric(MySamples); wss4<-numeric(MySamples)
  wss5<-numeric(MySamples); Trans1_11<-numeric(MySamples)
  Trans1_12<-numeric(MySamples); Trans1_13<-numeric(MySamples)
```

```
Trans1_14<-numeric(MySamples); Trans1_21<-numeric(MySamples)
Trans1_22<-numeric(MySamples); Trans1_23<-numeric(MySamples)
Trans1_24<-numeric(MySamples); Trans1_31<-numeric(MySamples)
Trans1_32<-numeric(MySamples); Trans1_33<-numeric(MySamples)
Trans1_34<-numeric(MySamples); Trans1_41<-numeric(MySamples)
Trans1_42<-numeric(MySamples); Trans1_43<-numeric(MySamples)
Trans1_44<-numeric(MySamples); Trans2_11<-numeric(MySamples)
Trans2_12<-numeric(MySamples); Trans2_13<-numeric(MySamples)
Trans2_14<-numeric(MySamples); Trans2_21<-numeric(MySamples)
Trans2_22<-numeric(MySamples); Trans2_23<-numeric(MySamples)
Trans2_24<-numeric(MySamples); Trans2_31<-numeric(MySamples)
Trans2_32<-numeric(MySamples); Trans2_33<-numeric(MySamples)
Trans2_34<-numeric(MySamples); Trans2_41<-numeric(MySamples)
Trans2_42<-numeric(MySamples); Trans2_43<-numeric(MySamples)
Trans2_44<-numeric(MySamples); Kappa<-numeric(MySamples)
correct_class<-numeric(MySamples)
cluster_order<-numeric(MySamples)
true_trans1<-trans1; true_trans2<-trans2

for(k in 1:MySamples){
x<-simSeqSample(true_trans1, initial=initial, len, N=size/2)

x[1,1]<-1;x[2,1]<-2;x[3,1]<-3;x[4,1]<-4

y<-simSeqSample(true_trans2, initial=initial2, len, N=size/2)

x<-rbind(x,y)

suppressMessages(my_seq<-seqdef(x, start=1, alphabet=c(1,2,3,4),
labels=c("noSC/DC", "SOnly", "DConly", "SC+DC")))

####NumberofClusters####

submat<-suppressMessages(seqsubm(my_seq, method="TRATE"))

dist.oml<-suppressMessages(seqdist(my_seq, method="OM", sm=submat))

wss<-(nrow(dist.oml)-1)*sum(apply(x, 2, var))
```

```

for(i in 2:5)wss[i]<-sum(kmeans(x,centers=i)$withinss)

clusterward<-agnes(dist.oml,diss=TRUE,method="ward")#thealgorithm
cluster<-cutree(clusterward,k=2)

#wss#

wss1[k]<-wss[1]
wss2[k]<-wss[2]
wss3[k]<-wss[3]
wss4[k]<-wss[4]
wss5[k]<-wss[5]

####2 Classes: Sil#####

Sil2[k]<-pam(dist.oml,2)$silinfo$avg.width
Sil2_min[k]<-min(pam(dist.oml,2)$silinfo$clus.avg.width)

####3 Classes: Sil#####

Sil3[k]<-pam(dist.oml,3)$silinfo$avg.width
Sil3_min[k]<-min(pam(dist.oml,3)$silinfo$clus.avg.width)

####2 Classes#####

trans1<-suppressMessages(seqrate(my_seq[cluster==1,]))
trans2<-suppressMessages(seqrate(my_seq[cluster==2,]))

if(sum((trans1-true_trans1)^2)<=sum((trans2-true_trans1)^2))
{cluster_order[k]<-"1-2"}else{cluster_order[k]<-"2-1"}

if(sum((trans1-true_trans1)^2)<=sum((trans2-true_trans1)^2)){

Trans1_11[k]<-trans1[1,1];Trans1_12[k]<-trans1[1,2];Trans1_13[k]<-trans1[1,3]
Trans1_14[k]<-trans1[1,4];Trans1_21[k]<-trans1[2,1];Trans1_22[k]<-trans1[2,2]
Trans1_23[k]<-trans1[2,3];Trans1_24[k]<-trans1[2,4];Trans1_31[k]<-trans1[3,1]

```



```
Trans1_32[k]<-trans1 [3 ,2]; Trans1_33[k]<-trans1 [3 ,3]; Trans1_34[k]<-trans1 [3 ,4]
Trans1_41[k]<-trans1 [4 ,1]; Trans1_42[k]<-trans1 [4 ,2]; Trans1_43[k]<-trans1 [4 ,3]
Trans1_44[k]<-trans1 [4 ,4]; Trans2_11[k]<-trans2 [1 ,1]; Trans2_12[k]<-trans2 [1 ,2]
Trans2_13[k]<-trans2 [1 ,3]; Trans2_14[k]<-trans2 [1 ,4]; Trans2_21[k]<-trans2 [2 ,1]
Trans2_22[k]<-trans2 [2 ,2]; Trans2_23[k]<-trans2 [2 ,3]; Trans2_24[k]<-trans2 [2 ,4]
Trans2_31[k]<-trans2 [3 ,1]; Trans2_32[k]<-trans2 [3 ,2]; Trans2_33[k]<-trans2 [3 ,3]
Trans2_34[k]<-trans2 [3 ,4]; Trans2_41[k]<-trans2 [4 ,1]; Trans2_42[k]<-trans2 [4 ,2]
Trans2_43[k]<-trans2 [4 ,3]; Trans2_44[k]<-trans2 [4 ,4]
```

```
classi<-cluster
true_class<-c(rep(1 , size/2) , rep(2 , size/2))
```

```
Kappa[k]<-suppressMessages (round (cohen . kappa ( data . frame ( classi ,
true_class )) $confid [3] ,2))
correct_class [k]<-mean (classi==true_class)
```

```
} else {
```

```
Trans1_11[k]<-trans2 [1 ,1]; Trans1_12[k]<-trans2 [1 ,2]; Trans1_13[k]<-trans2 [1 ,3]
Trans1_14[k]<-trans2 [1 ,4]; Trans1_21[k]<-trans2 [2 ,1]; Trans1_22[k]<-trans2 [2 ,2]
Trans1_23[k]<-trans2 [2 ,3]; Trans1_24[k]<-trans2 [2 ,4]; Trans1_31[k]<-trans2 [3 ,1]
Trans1_32[k]<-trans2 [3 ,2]; Trans1_33[k]<-trans2 [3 ,3]; Trans1_34[k]<-trans2 [3 ,4]
Trans1_41[k]<-trans2 [4 ,1]; Trans1_42[k]<-trans2 [4 ,2]; Trans1_43[k]<-trans2 [4 ,3]
Trans1_44[k]<-trans2 [4 ,4]; Trans2_11[k]<-trans1 [1 ,1]; Trans2_12[k]<-trans1 [1 ,2]
Trans2_13[k]<-trans1 [1 ,3]; Trans2_14[k]<-trans1 [1 ,4]; Trans2_21[k]<-trans1 [2 ,1]
Trans2_22[k]<-trans1 [2 ,2]; Trans2_23[k]<-trans1 [2 ,3]; Trans2_24[k]<-trans1 [2 ,4]
Trans2_31[k]<-trans1 [3 ,1]; Trans2_32[k]<-trans1 [3 ,2]; Trans2_33[k]<-trans1 [3 ,3]
Trans2_34[k]<-trans1 [3 ,4]; Trans2_41[k]<-trans1 [4 ,1]; Trans2_42[k]<-trans1 [4 ,2]
Trans2_43[k]<-trans1 [4 ,3]; Trans2_44[k]<-trans1 [4 ,4]
```

```
classi<-c()
classi [cluster==1]<-2
classi [cluster==2]<-1
true_class<-c(rep(1 , size/2) , rep(2 , size/2))
```

```
Kappa[k]<-suppressMessages (round (cohen . kappa ( data . frame ( classi ,
true_class )) $confid [3] ,2))
correct_class [k]<-mean (classi==true_class)
```

```
}

setTxtProgressBar(pb,k, title=NULL, label=NULL)

}

options(warn=0)

out<-data.frame(Sil2 , Sil3 , Sil2_min, Sil3_min, wss1 , wss2 ,
wss3, wss4, wss5, Trans1_11, Trans1_12, Trans1_13, Trans1_14,
Trans1_21, Trans1_22, Trans1_23, Trans1_24, Trans1_31, Trans1_32,
Trans1_33, Trans1_34, Trans1_41, Trans1_42, Trans1_43, Trans1_44,
Trans2_11, Trans2_12, Trans2_13, Trans2_14, Trans2_21, Trans2_22,
Trans2_23, Trans2_24, Trans2_31, Trans2_32, Trans2_33, Trans2_34,
Trans2_41, Trans2_42, Trans2_43, Trans2_44, Kappa, correct_class ,
cluster_order)}

#####
##Simulation##
#####

#Drawrandomseed
sample(1:10000,1)
#Seedwas6906
set.seed(6906)

#ScenarioA
Class<-"TwoClass_Scenario_OM_A"

trans1<-matrix(c(0.05,0.05,0.05,0.85,
0.05,0.05,0.85,0.05,
0.05,0.85,0.05,0.05,
0.85,0.05,0.05,0.05),4,4,byrow=TRUE)
```

```

trans2<-matrix(c(0.70,0.2,0.05,0.05,
0.05,0.7,0.20,0.05,
0.05,0.05,0.7,0.20,
0.01,0.01,0.01,0.97),4,4,byrow=TRUE)

#Distancebetweentwomtransitionmatrices
sum(abs(trans1-trans2))
sum((trans1-trans2)^2)

for(i in c(10,20,30,40,50,100)){
for(k in c(10,20,30,40,50,100)){

myTitle<-paste(Class,"_N",i,"_t",k,"_Sim.R",sep="")
cat(paste("\n\n"))
print(myTitle)

Sim1<-sim_OM2(trans1=trans1,
trans2=trans2,
initial1=c(.25,.25,.25,.25),
initial2=c(.25,.25,.25,.25),
len=k,
size=i,
MySamples=1000)

save(Sim1, file=paste(Class,"_N",i,"_t",k,"_Sim.R",sep=""))
}
}

#####
##Threelatentgroups##
#####

#####

```

```

#getsizesifsizecannotbedevidedbythree#
#####

diff3<-function(size){
sizef<-floor(size/3)
if(size%%3==0){
a<-sizef
b<-sizef
c<-sizef
}else if(size%%3==1){
a<-sizef+1
b<-sizef
c<-sizef
}else{
a<-sizef+1
b<-sizef+1
c<-sizef
}
out<-c(a,b,c)
}

sim_OM3<-function(trans1=matrix(c(1.00,0.00,0.00,0.00,
0.20,0.70,0.05,0.05,
0.05,0.25,0.65,0.05,
0.05,0.05,0.25,0.65),4,4,byrow=TRUE),
initial1=c(.25,.25,.25,.25),
trans2=matrix(c(0.10,0.05,0.05,0.80,
0.10,0.05,0.05,0.80,
0.00,0.00,1.00,0.00,
0.10,0.05,0.05,0.80),4,4,byrow=TRUE),
initial2=c(.25,.25,.25,.25),
trans3=matrix(c(0.7,0.1,0.1,0.1,
0.7,0,0.3,0,
0.7,0.3,0,0,
0,0,0,1),4,4,byrow=TRUE),
initial3=c(.25,.25,.25,.25),
len=30,
size=30,
MySamples=10){

```

```
options (warn=-1)
```

```
require (seqHMM)  
require (TraMineR)  
require (DySeq)  
require (psych)  
require (cluster)  
require (fpc)
```

```
pb<-txtProgressBar (min=0,max=MySamples,initial=0,char=" ",  
width=NA, title , label , style =3, file = "")
```

```
Sil2<-numeric (MySamples); Sil3<-numeric (MySamples)  
Sil2_min<-numeric (MySamples); Sil3_min<-numeric (MySamples)  
wss1<-numeric (MySamples); wss2<-numeric (MySamples)  
wss3<-numeric (MySamples); wss4<-numeric (MySamples)  
wss5<-numeric (MySamples); Trans1_11<-numeric (MySamples)  
Trans1_12<-numeric (MySamples); Trans1_13<-numeric (MySamples)  
Trans1_14<-numeric (MySamples); Trans1_21<-numeric (MySamples)  
Trans1_22<-numeric (MySamples); Trans1_23<-numeric (MySamples)  
Trans1_24<-numeric (MySamples); Trans1_31<-numeric (MySamples)  
Trans1_32<-numeric (MySamples); Trans1_33<-numeric (MySamples)  
Trans1_34<-numeric (MySamples); Trans1_41<-numeric (MySamples)  
Trans1_42<-numeric (MySamples); Trans1_43<-numeric (MySamples)  
Trans1_44<-numeric (MySamples); Trans2_11<-numeric (MySamples)  
Trans2_12<-numeric (MySamples); Trans2_13<-numeric (MySamples)  
Trans2_14<-numeric (MySamples); Trans2_21<-numeric (MySamples)  
Trans2_22<-numeric (MySamples); Trans2_23<-numeric (MySamples)  
Trans2_24<-numeric (MySamples); Trans2_31<-numeric (MySamples)  
Trans2_32<-numeric (MySamples); Trans2_33<-numeric (MySamples)  
Trans2_34<-numeric (MySamples); Trans2_41<-numeric (MySamples)  
Trans2_42<-numeric (MySamples); Trans2_43<-numeric (MySamples)  
Trans2_44<-numeric (MySamples); Trans3_11<-numeric (MySamples)  
Trans3_12<-numeric (MySamples); Trans3_13<-numeric (MySamples)  
Trans3_14<-numeric (MySamples); Trans3_21<-numeric (MySamples)  
Trans3_22<-numeric (MySamples); Trans3_23<-numeric (MySamples)  
Trans3_24<-numeric (MySamples); Trans3_31<-numeric (MySamples)  
Trans3_32<-numeric (MySamples); Trans3_33<-numeric (MySamples)
```

```

Trans3_34<-numeric(MySamples); Trans3_41<-numeric(MySamples)
Trans3_42<-numeric(MySamples); Trans3_43<-numeric(MySamples)
Trans3_44<-numeric(MySamples); cluster_order<-numeric(MySamples)
Kappa<-numeric(MySamples)
correct_class<-numeric(MySamples)
true_trans1<-trans1; true_trans2<-trans2
true_trans3<-trans3; size3<-diff3(size)
for(k in 1:MySamples)
){
x<-simSeqSample(true_trans1, initial=initial, len, N=size3[1])
x[1,1]<-1; x[2,1]<-2
x[3,1]<-3; x[4,1]<-4
y<-simSeqSample(true_trans2, initial=initial2, len, N=size3[2])
w<-simSeqSample(true_trans3, initial=initial3, len, N=size3[3])
x<-rbind(x, y, w)
suppressMessages(my_seq<-seqdef(x, start=1,
alphabet=c(1,2,3,4), labels=c("noSC/DC", "SOnly", "DOnly", "SC+DC")))
####NumberofClusters####
submat<-suppressMessages(seqsubm(my_seq, method="TRATE"))
dist.oml<-suppressMessages(seqdist(my_seq, method="CM", sm=submat))
wss<-(nrow(dist.oml)-1)*sum(apply(x, 2, var))
for(i in 2:5) wss[i]<-sum(kmeans(x, centers=i)$withinss)
clusterward<-agnes(dist.oml, diss=TRUE, method="ward")#thealgorithm
cluster<-cutree(clusterward, k=3)

#wss#

wss1[k]<-wss[1]; wss2[k]<-wss[2]
wss3[k]<-wss[3]; wss4[k]<-wss[4]
wss5[k]<-wss[5]

####2 Classes: Sil####

Sil2[k]<-pam(dist.oml, 2)$silinfo$avg.width
Sil2_min[k]<-min(pam(dist.oml, 2)$silinfo$clus.avg.width)

####3 Classes: Sil####

Sil3[k]<-pam(dist.oml, 3)$silinfo$avg.width
Sil3_min[k]<-min(pam(dist.oml, 3)$silinfo$clus.avg.width)

```

```
#####3 Classes#####

trans1<-suppressMessages( seqtrate(my_seq[ cluster ==1 ,]))
trans2<-suppressMessages( seqtrate(my_seq[ cluster ==2 ,]))
trans3<-suppressMessages( seqtrate(my_seq[ cluster ==3 ,]))

boundmatrix<-rbind( as . vector( trans1 ) , as . vector( trans2 ) , as . vector( trans3 ))

ABC<-sum(( rbind( as . vector( true_trans1 ) , as . vector( true_trans2 ) ,
as . vector( true_trans3 )) - boundmatrix)^2)
ACB<-sum(( rbind( as . vector( true_trans1 ) , as . vector( true_trans3 ) ,
as . vector( true_trans2 )) - boundmatrix)^2)
BAC<-sum(( rbind( as . vector( true_trans2 ) , as . vector( true_trans1 ) ,
as . vector( true_trans3 )) - boundmatrix)^2)
BCA<-sum(( rbind( as . vector( true_trans2 ) , as . vector( true_trans3 ) ,
as . vector( true_trans1 )) - boundmatrix)^2)
CAB<-sum(( rbind( as . vector( true_trans3 ) , as . vector( true_trans1 ) ,
as . vector( true_trans2 )) - boundmatrix)^2)
CBA<-sum(( rbind( as . vector( true_trans3 ) , as . vector( true_trans2 ) ,
as . vector( true_trans1 )) - boundmatrix)^2)

fall<-c(ABC,ACB,BAC,BCA,CAB,CBA)
myorder<-which( fall ==min( fall ))

cluster_order[k]<-myorder

post_class<-cluster
true_class<-c(rep(1 , size3 [1]) , rep(2 , size3 [2]) , rep(3 , size3 [3]))

if(myorder==1){
#Order:ABC
Trans1_11[k]<-trans1 [1 , 1]; Trans1_12[k]<-trans1 [1 , 2]; Trans1_13[k]<-trans1 [1 , 3]
Trans1_14[k]<-trans1 [1 , 4]
Trans1_21[k]<-trans1 [2 , 1]; Trans1_22[k]<-trans1 [2 , 2]; Trans1_23[k]<-trans1 [2 , 3]
Trans1_24[k]<-trans1 [2 , 4]
Trans1_31[k]<-trans1 [3 , 1]; Trans1_32[k]<-trans1 [3 , 2]; Trans1_33[k]<-trans1 [3 , 3]
Trans1_34[k]<-trans1 [3 , 4]
Trans1_41[k]<-trans1 [4 , 1]; Trans1_42[k]<-trans1 [4 , 2]; Trans1_43[k]<-trans1 [4 , 3]
Trans1_44[k]<-trans1 [4 , 4]
```

```

Trans2_11[k]<-trans2 [ 1 , 1]; Trans2_12[k]<-trans2 [ 1 , 2]; Trans2_13[k]<-trans2 [ 1 , 3]
Trans2_14[k]<-trans2 [ 1 , 4]
Trans2_21[k]<-trans2 [ 2 , 1]; Trans2_22[k]<-trans2 [ 2 , 2]; Trans2_23[k]<-trans2 [ 2 , 3]
Trans2_24[k]<-trans2 [ 2 , 4]
Trans2_31[k]<-trans2 [ 3 , 1]; Trans2_32[k]<-trans2 [ 3 , 2]; Trans2_33[k]<-trans2 [ 3 , 3]
Trans2_34[k]<-trans2 [ 3 , 4]; Trans2_41[k]<-trans2 [ 4 , 1]; Trans2_42[k]<-trans2 [ 4 , 2]
Trans2_43[k]<-trans2 [ 4 , 3]; Trans2_44[k]<-trans2 [ 4 , 4]; Trans3_11[k]<-trans3 [ 1 , 1]
Trans3_12[k]<-trans3 [ 1 , 2]; Trans3_13[k]<-trans3 [ 1 , 3]
Trans3_14[k]<-trans3 [ 1 , 4]; Trans3_24[k]<-trans3 [ 2 , 4]
Trans3_21[k]<-trans3 [ 2 , 1]; Trans3_22[k]<-trans3 [ 2 , 2]; Trans3_23[k]<-trans3 [ 2 , 3]
Trans3_24[k]<-trans3 [ 2 , 4]
Trans3_31[k]<-trans3 [ 3 , 1]; Trans3_32[k]<-trans3 [ 3 , 2]; Trans3_33[k]<-trans3 [ 3 , 3]
Trans3_34[k]<-trans3 [ 3 , 4]; Trans3_44[k]<-trans3 [ 4 , 4]; Trans3_24[k]<-trans3 [ 2 , 4]
Trans3_41[k]<-trans3 [ 4 , 1]; Trans3_42[k]<-trans3 [ 4 , 2]; Trans3_43[k]<-trans3 [ 4 , 3]
Kappa[k]<-suppressMessages (round (cohen . kappa ( data . frame ( cluster ,
true_ class )) $confid [ 3 ] , 2))
correct_ class [ k]<-mean ( cluster == true_ class )

} else if (myorder == 2) {
#Order : ACB
Trans1_11[k]<-trans1 [ 1 , 1]; Trans1_12[k]<-trans1 [ 1 , 2]; Trans1_13[k]<-trans1 [ 1 , 3]
Trans1_14[k]<-trans1 [ 1 , 4]; Trans1_21[k]<-trans1 [ 2 , 1]; Trans1_22[k]<-trans1 [ 2 , 2]
Trans1_23[k]<-trans1 [ 2 , 3]; Trans1_24[k]<-trans1 [ 2 , 4]; Trans1_31[k]<-trans1 [ 3 , 1]
Trans1_32[k]<-trans1 [ 3 , 2]; Trans1_33[k]<-trans1 [ 3 , 3]; Trans1_34[k]<-trans1 [ 3 , 4]
Trans1_41[k]<-trans1 [ 4 , 1]; Trans1_42[k]<-trans1 [ 4 , 2]; Trans1_43[k]<-trans1 [ 4 , 3]
Trans1_44[k]<-trans1 [ 4 , 4]; Trans2_11[k]<-trans3 [ 1 , 1]; Trans2_12[k]<-trans3 [ 1 , 2]
Trans2_13[k]<-trans3 [ 1 , 3]; Trans2_14[k]<-trans3 [ 1 , 4]; Trans2_21[k]<-trans3 [ 2 , 1]
Trans2_22[k]<-trans3 [ 2 , 2]; Trans2_23[k]<-trans3 [ 2 , 3]; Trans2_24[k]<-trans3 [ 2 , 4]
Trans2_31[k]<-trans3 [ 3 , 1]; Trans2_32[k]<-trans3 [ 3 , 2]; Trans2_33[k]<-trans3 [ 3 , 3]
Trans2_34[k]<-trans3 [ 3 , 4]; Trans2_41[k]<-trans3 [ 4 , 1]; Trans2_42[k]<-trans3 [ 4 , 2]
Trans2_43[k]<-trans3 [ 4 , 3]; Trans2_44[k]<-trans3 [ 4 , 4]; Trans3_11[k]<-trans2 [ 1 , 1]
Trans3_12[k]<-trans2 [ 1 , 2]; Trans3_13[k]<-trans2 [ 1 , 3]; Trans3_14[k]<-trans2 [ 1 , 4]
Trans3_21[k]<-trans2 [ 2 , 1]; Trans3_22[k]<-trans2 [ 2 , 2]; Trans3_23[k]<-trans2 [ 2 , 3]
Trans3_24[k]<-trans2 [ 2 , 4]; Trans3_31[k]<-trans2 [ 3 , 1]; Trans3_32[k]<-trans2 [ 3 , 2]
Trans3_33[k]<-trans2 [ 3 , 3]; Trans3_34[k]<-trans2 [ 3 , 4]; Trans3_41[k]<-trans2 [ 4 , 1]
Trans3_42[k]<-trans2 [ 4 , 2]; Trans3_43[k]<-trans2 [ 4 , 3]; Trans3_44[k]<-trans2 [ 4 , 4]

classi<-numeric ( size )
classi [ cluster == 1 ]<-1

```



```
classi [ cluster == 2 ] <- 3
classi [ cluster == 3 ] <- 2
```

```
Kappa[k] <- suppressMessages ( round ( cohen . kappa ( data . frame ( classi ,
true_class )) $ confid [ 3 ] , 2 ) )
correct_class [ k ] <- mean ( classi == true_class )
```

```
} else if ( myorder == 3 ) {
```

```
#Order: BAC
```

```
Trans2_11[k] <- trans1 [ 1 , 1 ] ; Trans2_12[k] <- trans1 [ 1 , 2 ] ; Trans2_13[k] <- trans1 [ 1 , 3 ]
Trans2_14[k] <- trans1 [ 1 , 4 ] ; Trans2_21[k] <- trans1 [ 2 , 1 ] ; Trans2_22[k] <- trans1 [ 2 , 2 ]
Trans2_23[k] <- trans1 [ 2 , 3 ] ; Trans2_24[k] <- trans1 [ 2 , 4 ] ; Trans2_31[k] <- trans1 [ 3 , 1 ]
Trans2_32[k] <- trans1 [ 3 , 2 ] ; Trans2_33[k] <- trans1 [ 3 , 3 ] ; Trans2_34[k] <- trans1 [ 3 , 4 ]
Trans2_41[k] <- trans1 [ 4 , 1 ] ; Trans2_42[k] <- trans1 [ 4 , 2 ] ; Trans2_43[k] <- trans1 [ 4 , 3 ]
Trans2_44[k] <- trans1 [ 4 , 4 ] ; Trans1_11[k] <- trans2 [ 1 , 1 ] ; Trans1_12[k] <- trans2 [ 1 , 2 ]
Trans1_13[k] <- trans2 [ 1 , 3 ] ; Trans1_14[k] <- trans2 [ 1 , 4 ] ; Trans1_21[k] <- trans2 [ 2 , 1 ]
Trans1_22[k] <- trans2 [ 2 , 2 ] ; Trans1_23[k] <- trans2 [ 2 , 3 ] ; Trans1_24[k] <- trans2 [ 2 , 4 ]
Trans1_31[k] <- trans2 [ 3 , 1 ] ; Trans1_32[k] <- trans2 [ 3 , 2 ] ; Trans1_33[k] <- trans2 [ 3 , 3 ]
Trans1_34[k] <- trans2 [ 3 , 4 ] ; Trans1_41[k] <- trans2 [ 4 , 1 ] ; Trans1_42[k] <- trans2 [ 4 , 2 ]
Trans1_43[k] <- trans2 [ 4 , 3 ] ; Trans1_44[k] <- trans2 [ 4 , 4 ] ; Trans3_11[k] <- trans3 [ 1 , 1 ]
Trans3_12[k] <- trans3 [ 1 , 2 ] ; Trans3_13[k] <- trans3 [ 1 , 3 ] ; Trans3_14[k] <- trans3 [ 1 , 4 ]
Trans3_21[k] <- trans3 [ 2 , 1 ] ; Trans3_22[k] <- trans3 [ 2 , 2 ] ; Trans3_23[k] <- trans3 [ 2 , 3 ]
Trans3_24[k] <- trans3 [ 2 , 4 ] ; Trans3_31[k] <- trans3 [ 3 , 1 ] ; Trans3_32[k] <- trans3 [ 3 , 2 ]
Trans3_33[k] <- trans3 [ 3 , 3 ] ; Trans3_34[k] <- trans3 [ 3 , 4 ] ; Trans3_41[k] <- trans3 [ 4 , 1 ]
Trans3_42[k] <- trans3 [ 4 , 2 ] ; Trans3_43[k] <- trans3 [ 4 , 3 ] ; Trans3_44[k] <- trans3 [ 4 , 4 ]
```

```
classi <- numeric ( size )
classi [ cluster == 1 ] <- 2
classi [ cluster == 2 ] <- 1
classi [ cluster == 3 ] <- 3
```

```
Kappa[k] <- suppressMessages ( round ( cohen . kappa ( data . frame ( classi ,
true_class )) $ confid [ 3 ] , 2 ) )
correct_class [ k ] <- mean ( classi == true_class )
```

```
} else if ( myorder == 4 ) {
```

```
#Order: BCA
```

```
Trans1_11[k] <- trans2 [ 1 , 1 ] ; Trans1_12[k] <- trans2 [ 1 , 2 ] ; Trans1_13[k] <- trans2 [ 1 , 3 ]
```

```

Trans1_14[k]<-trans2 [1 ,4]; Trans1_21[k]<-trans2 [2 ,1]; Trans1_22[k]<-trans2 [2 ,2]
Trans1_23[k]<-trans2 [2 ,3]; Trans1_24[k]<-trans2 [2 ,4]; Trans1_31[k]<-trans2 [3 ,1]
Trans1_32[k]<-trans2 [3 ,2]; Trans1_33[k]<-trans2 [3 ,3]; Trans1_34[k]<-trans2 [3 ,4]
Trans1_41[k]<-trans2 [4 ,1]; Trans1_42[k]<-trans2 [4 ,2]; Trans1_43[k]<-trans2 [4 ,3]
Trans1_44[k]<-trans2 [4 ,4]; Trans2_11[k]<-trans3 [1 ,1]; Trans2_12[k]<-trans3 [1 ,2]
Trans2_13[k]<-trans3 [1 ,3]; Trans2_14[k]<-trans3 [1 ,4]; Trans2_21[k]<-trans3 [2 ,1]
Trans2_22[k]<-trans3 [2 ,2]; Trans2_23[k]<-trans3 [2 ,3]; Trans2_24[k]<-trans3 [2 ,4]
Trans2_31[k]<-trans3 [3 ,1]; Trans2_32[k]<-trans3 [3 ,2]; Trans2_33[k]<-trans3 [3 ,3]
Trans2_34[k]<-trans3 [3 ,4]; Trans2_41[k]<-trans3 [4 ,1]; Trans2_42[k]<-trans3 [4 ,2]
Trans2_43[k]<-trans3 [4 ,3]; Trans2_44[k]<-trans3 [4 ,4]; Trans3_11[k]<-trans1 [1 ,1]
Trans3_12[k]<-trans1 [1 ,2]; Trans3_13[k]<-trans1 [1 ,3]; Trans3_14[k]<-trans1 [1 ,4]
Trans3_21[k]<-trans1 [2 ,1]; Trans3_22[k]<-trans1 [2 ,2]; Trans3_23[k]<-trans1 [2 ,3]
Trans3_24[k]<-trans1 [2 ,4]; Trans3_31[k]<-trans1 [3 ,1]; Trans3_32[k]<-trans1 [3 ,2]
Trans3_33[k]<-trans1 [3 ,3]; Trans3_34[k]<-trans1 [3 ,4]; Trans3_41[k]<-trans1 [4 ,1]
Trans3_42[k]<-trans1 [4 ,2]; Trans3_43[k]<-trans1 [4 ,3]; Trans3_44[k]<-trans1 [4 ,4]
classi<-numeric( size )
classi [ cluster ==1] <-2
classi [ cluster ==2] <-3
classi [ cluster ==3] <-1

Kappa[k]<-suppressMessages( round( cohen . kappa( data . frame( classi
, true_class )) $confid [3] ,2))
correct_class [k]<-mean( classi ==true_class )

} else if (myorder ==5){
#Order : CAB
Trans1_11[k]<-trans3 [1 ,1]; Trans1_12[k]<-trans3 [1 ,2]; Trans1_13[k]<-trans3 [1 ,3]
Trans1_14[k]<-trans3 [1 ,4]; Trans1_21[k]<-trans3 [2 ,1]; Trans1_22[k]<-trans3 [2 ,2]
Trans1_23[k]<-trans3 [2 ,3]; Trans1_24[k]<-trans3 [2 ,4]; Trans1_31[k]<-trans3 [3 ,1]
Trans1_32[k]<-trans3 [3 ,2]; Trans1_33[k]<-trans3 [3 ,3]; Trans1_34[k]<-trans3 [3 ,4]
Trans1_41[k]<-trans3 [4 ,1]; Trans1_42[k]<-trans3 [4 ,2]; Trans1_43[k]<-trans3 [4 ,3]
Trans1_44[k]<-trans3 [4 ,4]; Trans2_11[k]<-trans1 [1 ,1]; Trans2_12[k]<-trans1 [1 ,2]
Trans2_13[k]<-trans1 [1 ,3]; Trans2_14[k]<-trans1 [1 ,4]; Trans2_21[k]<-trans1 [2 ,1]
Trans2_22[k]<-trans1 [2 ,2]; Trans2_23[k]<-trans1 [2 ,3]; Trans2_24[k]<-trans1 [2 ,4]
Trans2_31[k]<-trans1 [3 ,1]; Trans2_32[k]<-trans1 [3 ,2]; Trans2_33[k]<-trans1 [3 ,3]
Trans2_34[k]<-trans1 [3 ,4]; Trans2_41[k]<-trans1 [4 ,1]; Trans2_42[k]<-trans1 [4 ,2]
Trans2_43[k]<-trans1 [4 ,3]; Trans2_44[k]<-trans1 [4 ,4]; Trans3_11[k]<-trans2 [1 ,1]
Trans3_12[k]<-trans2 [1 ,2]; Trans3_13[k]<-trans2 [1 ,3]; Trans3_14[k]<-trans2 [1 ,4]
Trans3_21[k]<-trans2 [2 ,1]; Trans3_22[k]<-trans2 [2 ,2]; Trans3_23[k]<-trans2 [2 ,3]
Trans3_24[k]<-trans2 [2 ,4]; Trans3_31[k]<-trans2 [3 ,1]; Trans3_32[k]<-trans2 [3 ,2]

```

```

Trans3_33[k]<-trans2 [3 ,3];Trans3_34[k]<-trans2 [3 ,4];Trans3_41[k]<-trans2 [4 ,1]
Trans3_42[k]<-trans2 [4 ,2];Trans3_43[k]<-trans2 [4 ,3];Trans3_44[k]<-trans2 [4 ,4]
classi<-numeric( size )
classi [ cluster ==1]<-3
classi [ cluster ==2]<-1
classi [ cluster ==3]<-2

```

```

Kappa[k]<-suppressMessages (round( cohen . kappa( data . frame( classi ,
true_class ))$confid [3] ,2))
correct_class [k]<-mean( classi==true_class )

```

```

} else {
#Order :CBA

```

```

Trans1_11[k]<-trans3 [1 ,1];Trans1_12[k]<-trans3 [1 ,2];Trans1_13[k]<-trans3 [1 ,3]
Trans1_14[k]<-trans3 [1 ,4];Trans1_21[k]<-trans3 [2 ,1];Trans1_22[k]<-trans3 [2 ,2]
Trans1_23[k]<-trans3 [2 ,3];Trans1_24[k]<-trans3 [2 ,4];Trans1_31[k]<-trans3 [3 ,1]
Trans1_32[k]<-trans3 [3 ,2];Trans1_33[k]<-trans3 [3 ,3];Trans1_34[k]<-trans3 [3 ,4]
Trans1_41[k]<-trans3 [4 ,1];Trans1_42[k]<-trans3 [4 ,2];Trans1_43[k]<-trans3 [4 ,3]
Trans1_44[k]<-trans3 [4 ,4];Trans2_11[k]<-trans2 [1 ,1];Trans2_12[k]<-trans2 [1 ,2]
Trans2_13[k]<-trans2 [1 ,3];Trans2_14[k]<-trans2 [1 ,4];Trans2_21[k]<-trans2 [2 ,1]
Trans2_22[k]<-trans2 [2 ,2];Trans2_23[k]<-trans2 [2 ,3];Trans2_24[k]<-trans2 [2 ,4]
Trans2_31[k]<-trans2 [3 ,1];Trans2_32[k]<-trans2 [3 ,2];Trans2_33[k]<-trans2 [3 ,3]
Trans2_34[k]<-trans2 [3 ,4];Trans2_41[k]<-trans2 [4 ,1];Trans2_42[k]<-trans2 [4 ,2]
Trans2_43[k]<-trans2 [4 ,3];Trans2_44[k]<-trans2 [4 ,4];Trans3_11[k]<-trans1 [1 ,1]
Trans3_12[k]<-trans1 [1 ,2];Trans3_13[k]<-trans1 [1 ,3];Trans3_14[k]<-trans1 [1 ,4]
Trans3_21[k]<-trans1 [2 ,1];Trans3_22[k]<-trans1 [2 ,2];Trans3_23[k]<-trans1 [2 ,3]
Trans3_24[k]<-trans1 [2 ,4];Trans3_31[k]<-trans1 [3 ,1];Trans3_32[k]<-trans1 [3 ,2]
Trans3_33[k]<-trans1 [3 ,3];Trans3_34[k]<-trans1 [3 ,4];Trans3_41[k]<-trans1 [4 ,1]
Trans3_42[k]<-trans1 [4 ,2];Trans3_43[k]<-trans1 [4 ,3];Trans3_44[k]<-trans1 [4 ,4]

```

```

classi<-numeric( size )
classi [ cluster ==1]<-3
classi [ cluster ==2]<-2
classi [ cluster ==3]<-1

```

```

Kappa[k]<-suppressMessages (round( cohen . kappa( data . frame( classi ,
true_class ))$confid [3] ,2))

```

```

correct_class[k]<-mean(classi==true_class)

}

setTxtProgressBar(pb,k,title=NULL,label=NULL)

}

options(warn=0)
out<-data.frame(Sil2, Sil3, Sil2_min, Sil3_min,
wss1, wss2, wss3, wss4, wss5,
Trans1_11, Trans1_12, Trans1_13, Trans1_14,
Trans1_21, Trans1_22, Trans1_23, Trans1_24,
Trans1_31, Trans1_32, Trans1_33, Trans1_34,
Trans1_41, Trans1_42, Trans1_43, Trans1_44,
Trans2_11, Trans2_12, Trans2_13, Trans2_14,
Trans2_21, Trans2_22, Trans2_23, Trans2_24,
Trans2_31, Trans2_32, Trans2_33, Trans2_34,
Trans2_41, Trans2_42, Trans2_43, Trans2_44,
Trans3_11, Trans3_12, Trans3_13, Trans3_14,
Trans3_21, Trans3_22, Trans3_23, Trans3_24,
Trans3_31, Trans3_32, Trans3_33, Trans3_34,
Trans3_41, Trans3_42, Trans3_43, Trans3_44,
Kappa, correct_class, cluster_order)}

#####
#TestderSingelConditionSimulation#
#####

test_out<-sim_Mixture_Markov3(MySamples=10,len=100,size=100)
test_out

```

```
#####  
## Simulation ##  
#####  
  
#Drawrandomseed  
sample(1:10000,1)  
#Seedwas4331  
set.seed(9282)  
  
#ScenarioA  
  
Class<-"ThreeClass_Scenario_OM_A"  
  
trans1<-matrix(c(1.00,0.00,0.00,0.00,  
0.20,0.70,0.05,0.05,  
0.05,0.25,0.65,0.05,  
0.05,0.05,0.25,0.65),4,4,byrow=TRUE)  
  
trans2<-matrix(c(0.10,0.05,0.05,0.80,  
0.10,0.05,0.05,0.80,  
0.00,0.00,1.00,0.00,  
0.10,0.05,0.05,0.80),4,4,byrow=TRUE)  
  
trans3=matrix(c(0.7,0.1,0.1,0.1,  
0.7,0,0.3,0,  
0.7,0.3,0,0,  
0,0,0,1),4,4,byrow=TRUE)  
  
#Distancebetweentwomtransitionmatrices  
sum(abs(trans1-trans2))  
sum((trans1-trans2)^2)
```

```

for(i in c(10,20,30,40,50,100)){
for(k in c(10,20,30,40,50,100)){

myTitle<-paste(Class,"_N",i,"_t",k,"_Sim.R",sep="")
cat(paste("\\n\\n"))
print(myTitle)

Sim1<-sim_OM3(trans1=trans1 ,
trans2=trans2 ,
trans3=trans3 ,
initial1=c(.25 ,.25 ,.25 ,.25) ,
initial2=c(.25 ,.25 ,.25 ,.25) ,
initial3=c(.25 ,.25 ,.25 ,.25) ,
len=k ,
size=i ,
MySamples=1000)

save(Sim1 , file=paste(Class,"_N",i,"_t",k,"_Sim.R",sep=""))
}
}

#####
##SIMANOVA-like##
#####

Sim_Seq_ANOVA<-function(N,len , samples , initial1 ,
initial2 , initial3 , trans1 , trans2 , trans3){

pb<-txtProgressBar(min=0,max=samples , initial=0,char="=" ,
width=NA, title , label , style=3, file="")

diff3<-function(size){
sizef<-floor(size/3)
if(size%%3==0){
a<-sizef

```

```

b<-sizef
c<-sizef
} else if (size%%3==1){
a<-sizef+1
b<-sizef
c<-sizef
} else {
a<-sizef+1
b<-sizef+1
c<-sizef
}
out<-c(a,b,c)
}

if (!(is.null(trans3))){ size3<-diff3(N)}
else {size3<-c(); size3[1]<-ceiling(N/2); size3[2]<-floor(size3[1])}

out<-rep(NA,samples)

for(i in 1:samples){

x<-simSeqSample(trans1,initial=initial,len=len,N=size3[1])
y<-simSeqSample(trans2,initial=initial,len=len,N=size3[2])
if (!(is.null(trans3))){w<-simSeqSample(trans3,
initial=initial,len=len,N=size3[3])}

if (!(is.null(trans3))){x<-rbind(x,y,w)}
else {x<-rbind(x,y)}

mvad.alphabet<-c(1,2,3,4)
mvad.labels<-c("1","2","3","4")
mvad.scodes<-c("1","2","3","4")
suppressMessages(mvad.seq<-seqdef(x,1:len,
alphabet=mvad.alphabet,states=mvad.scodes,
labels=mvad.labels,xtstep=6))

```

```

suppressMessages(submat<-seqsubm(mvad.seq,method="TRATE"))
suppressMessages(dist.om1<-seqdist(mvad.seq,method="CM",indel=1,
sm=submat))

if(!is.null(trans3)){clus<-c(rep(1,size3[1]),rep(2,size3[2]),
rep(3,size3[3]))
} else {clus<-c(rep(1,size3[1]),rep(2,size3[2]))}

#rsq<-function(clus,data,indices){
#d<-data[indices,]
#da<-dissassoc(d,group=clus,R=1)
#out<-da$stat$t0[1]
#return(out)
#}

#results<-boot(data=dist.om1,statistic=rsq,
#R=1000,clus=clus)

#print(boot.ci(results,type="bca"))
da<-dissassoc(dist.om1,group=clus,R=1000)

out[i]<-da$stat$sp.value[1]
setTxtProgressBar(pb,i,title=NULL,label=NULL)
}
out
}

trans1<-matrix(c(.85,0.05,0.05,0.05,
0.20,0.70,0.05,0.05,
0.05,0.25,0.65,0.05,
0.05,0.05,0.25,0.65),4,4,byrow=TRUE)

trans2<-matrix(c(0.10,0.05,0.05,0.80,
0.10,0.05,0.05,0.80,
0.05,0.05,.85,0.05,

```



```
0.10,0.05,0.05,0.80),4,4,byrow=TRUE)
```

```
trans3=matrix(c(0.7,0.1,0.1,0.1,
0.7,0,0.3,0,
0.7,0.3,0,0,
0.05,0.05,0.05,.85),4,4,byrow=TRUE)
```

```
initial<-c(.25,.25,.25,.25)
```

```
len<-30
```

```
N<-30
```

```
samples<-100
```

```
Aout<-c()
```

```
A<-c()
```

```
z2<-0
```

```
for(k in c(4,5,6,7,8,9,10)){
```

```
for(z in c(10,20,30,40,50)){
```

```
z2<-z2+1
```

```
print(paste("N",z,"len",k))
```

```
A<-Sim_Seq_ANOVA(N=z,
```

```
len=k,
```

```
samples=1000,
```

```
initial1=initial,
```

```
initial2=initial,
```

```
initial3=initial,
```

```
trans1=trans1,
```

```
trans2=trans2,
```

```
trans3=NULL)
```

```
Aout[z2]<-mean(A<0.05)
```

```
}
```

```
}
```

```
Aout
```

C. The R-Package 'DySeq'

The R-Package DySeq can be obtained from CRAN by using the command `install.packages("DySeq")`. However, this will always give the newest version of DySeq. Future development of the DySeq-Package might alter results. The original code that was used for this monograph can be found on the following pages.

```
APIMtoTrans<-function(B0_1, AE_1, PE_1, Int_1,
                      B0_2, AE_2, PE_2, Int_2){

  # Coefficients of the first partner
  # B0_1 Intercept
  # AE_1 Actor Effect
  # PE_1 Partner Effect
  # Int_1 Interaction

  # Coefficients of the second partner
  # B0_2 Intercept
  # AE_2 Actor Effect
  # PE_2 Partner Effect
  # Int_2 Interaction

  # Preparing empty matrix object
  myTrans<-matrix(NA, 4,4)

  # State 1: -1 on both variables at t-1
  # State 2: -1 on both the first partners variable at t-1
  # State 3: -1 on both the second partners variable at t-1
  # State 4: +1 on both variables at t-1

  # First row:
  odds1<-exp(B0_2)*exp(AE_2)^(-1)*exp(PE_2)^(-1)*exp(Int_2)^(1)
  prob1<-odds1/(odds1+1)
  prob1 # probability to go from state 1 to either state 3 or 4
  1-prob1 # probability to go from state 1 to either 1 or 2
```

```

odds2<-exp(B0_1)*exp(AE_1)^(-1)*exp(PE_1)^(-1)*exp(Int_1)^(1)
prob2<-odds2/(odds2+1)
prob2 # probability to go from state 1 to either state 2 or 4
1-prob2 # probability to go from state 1 to either 3 or 4

myTrans[1,1]<-(1-prob1)*(1-prob2)
myTrans[1,2]<-(1-prob1)*prob2
myTrans[1,3]<-prob1*(1-prob2)
myTrans[1,4]<-prob1*prob2

# Second row:
odds1<-exp(B0_2)*exp(AE_2)^(-1)*exp(PE_2)^(1)*exp(Int_2)^(-1)
prob1<-odds1/(odds1+1)

odds2<-exp(B0_1)*exp(AE_1)^(1)*exp(PE_1)^(-1)*exp(Int_1)^(-1)
prob2<-odds2/(odds2+1)

myTrans[2,1]<-(1-prob1)*(1-prob2)
myTrans[2,2]<-(1-prob1)*prob2
myTrans[2,3]<-prob1*(1-prob2)
myTrans[2,4]<-prob1*prob2

# Third row:
odds1<-exp(B0_2)*exp(AE_2)^(1)*exp(PE_2)^(-1)*exp(Int_2)^(-1)
prob1<-odds1/(odds1+1)

odds2<-exp(B0_1)*exp(AE_1)^(-1)*exp(PE_1)^(1)*exp(Int_1)^(-1)
prob2<-odds2/(odds2+1)

myTrans[3,1]<-(1-prob1)*(1-prob2)
myTrans[3,2]<-(1-prob1)*prob2
myTrans[3,3]<-prob1*(1-prob2)
myTrans[3,4]<-prob1*prob2

# Fourth row:
odds1<-exp(B0_2)*exp(AE_2)^(1)*exp(PE_2)^(1)*exp(Int_2)^(1)
prob1<-odds1/(odds1+1)

odds2<-exp(B0_1)*exp(AE_1)^(1)*exp(PE_1)^(1)*exp(Int_1)^(1)
prob2<-odds2/(odds2+1)

```

```

myTrans[4,1]<-(1-prob1)*(1-prob2)
myTrans[4,2]<-(1-prob1)*prob2
myTrans[4,3]<-prob1*(1-prob2)
myTrans[4,4]<-prob1*prob2

return(myTrans)
}

Basic_Markov_as_APIM<-function(x, first, second, boot=1000,
SimOut=FALSE, CPU=1, sim="ordinary", parallel = "no"){
  out<-c()

  MyBetas<-function(data, indices){
    a<-StateExpand(data[indices,], first, second)
    b<-suppressMessages(TraMineR::seqdef(a[, first],
                                     start = 1,
                                     labels = c("0-0", "1-0", "0-1", "1-1")))
    z<-suppressMessages(TraMineR::seqrate(b))
    return(TransToAPIM(z))
  }

  results<-boot::boot(data=x,
                     statistic=MyBetas,
                     R=boot,
                     ncpus=CPU,
                     sim=sim)

  # Approximating the p-values for coefficient + saving estimate

  # DC
  # Intercept
  out[1]<-results$t0[1]
  DC_b0_H0_Dist<-results$t[,1]-mean(results$t[,1]) #H0 Distribution
  out[9]<-mean(DC_b0_H0_Dist>abs(results$t0[1]))
  DC_b0_H0_Dist<-abs(results$t0[1]))

```

```

# Actor
out[2]<-results$t0[2]
DC_Act_H0_Dist<-results$t[,2]-mean(results$t[,2]) # H0 Distribution
out[10]<-mean(DC_Act_H0_Dist>abs(results$t0[2]))|
DC_Act_H0_Dist<(-abs(results$t0[2]))

# Partner
out[3]<-results$t0[3]
DC_Par_H0_Dist<-results$t[,3]-mean(results$t[,3]) # H0 Distribution
out[11]<-mean(DC_Par_H0_Dist>abs(results$t0[3]))|
DC_Par_H0_Dist<(-abs(results$t0[3]))

# Interaction
out[4]<-results$t0[4]
DC_Int_H0_Dist<-results$t[,4]-mean(results$t[,4]) # H0 Distribution
out[12]<-mean(DC_Int_H0_Dist>abs(results$t0[4]))|
DC_Int_H0_Dist<(-abs(results$t0[4]))

# SC
# Intercept
# Intercept
out[5]<-results$t0[5]
SC_b0_H0_Dist<-results$t[,5]-mean(results$t[,5]) #H0 Distribution
out[13]<-mean(SC_b0_H0_Dist>abs(results$t0[5]))|
SC_b0_H0_Dist<(-abs(results$t0[5]))

# Actor
out[6]<-results$t0[6]
SC_Act_H0_Dist<-results$t[,6]-mean(results$t[,6]) # H0 Distribution
out[14]<-mean(SC_Act_H0_Dist>abs(results$t0[6]))|
SC_Act_H0_Dist<(-abs(results$t0[6]))

# Partner
out[7]<-results$t0[7]
SC_Par_H0_Dist<-results$t[,7]-mean(results$t[,7]) # H0 Distribution
out[15]<-mean(SC_Par_H0_Dist>abs(results$t0[7]))|
SC_Par_H0_Dist<(-abs(results$t0[7]))

# Interaction

```

```

out[8]<-results$t0[8]
SC_Int_H0_Dist<-results$t[,8]-mean(results$t[,8]) # H0 Distribution
out[16]<-mean(SC_Int_H0_Dist>abs(results$t0[8]))|
SC_Int_H0_Dist<(-abs(results$t0[8]))

if (SimOut){

  names(out)<-c("DC_b0",
              "DC_Actor",
              "DC_Partner",
              "DC_Inter",
              "SC_b0",
              "SC_Actor",
              "SC_Partner",
              "SC_Inter",
              "P_DC_b0",
              "P_DC_Actor",
              "P_DC_Partner",
              "P_DC_Inter",
              "P_SC_b0",
              "P_SC_Actor",
              "P_SC_Partner",
              "P_SC_Inter")

  output<-out

} else {

  out2<-data.frame(rep(NA, 8), rep(NA, 8))
  rownames(out2)<-c("First_Intercept",
                  "First_Actor",
                  "First_Partner",
                  "First_Interaction",
                  "Second_Intercept",
                  "Second_Actor",
                  "Second_Partner",
                  "Second_Interaction")
  colnames(out2)<-c("Estimate", "P_Value")
  out2[1:4,1]<-out[1:4]

```

```

out2[5:8,1]<-out[5:8]
out2[1:4,2]<-out[9:12]
out2[5:8,2]<-out[13:16]
As_APIM<-out2
a<-StateExpand(x, first, second)
b<-suppressMessages(TraMineR::seqdef(a[, first],
                                start = 1,
                                labels = c("0-0", "1-0", "0-1", "1-1")))
Transition_Matrix<-suppressMessages(TraMineR::seqtrate(b))

rownames(Transition_Matrix)<-c("[0:0_>]", "[1:0_>]",
"[0:1_>]", "[1:1_>]")
colnames(Transition_Matrix)<-c("[->_0:0]", "[->_1:0]",
"[->_0:1]", "[->_1:1]")

output<-list(Transition_Matrix, As_APIM)
names(output)<-c("Transition_Matrix",
"Transitions_converted_as_APIM")
}

return(output)
}

LastOccur<-function(x,y){
  if(!(is.matrix(x)|is.data.frame(x)))
    warning("x_must_be_a_matrix_or_dataframe!")

  output<-numeric(length(x[,1]))

  for(k in 1:length(x[,1])){
    for(i in 1:length(x[1,])){
      if(x[k,i]==y) {output[k]<-i}
    }
  }
  return(output)
}

LogSeq<-function(x, delta=0.5, subgroups=NA, single.case=FALSE){

```

```

if (class(x)[2]!="state.trans")
warning("x_should_be_a_state.trans_object._See_Help(StateTrans)!")

lambdas<-matrix(NA, length(x),4) # Empty table for lambdas

p.values<-matrix(NA, length(x),4)

for(i in 1:length(x)){

  x.long<-c(x[[i]][1:4,1],x[[i]][1:4,2])

  casearray<-array(x.long, c(2,2,2),
list(c("seq2_1", "seq2_0"),c("seq1_1", "seq2_0"), c("dep1", "dep0")))

  caselog<-MASS::loglm(~1+2+3+1*2*3, data=(casearray+delta), fit=F)

  if (single.case==TRUE){
    caselog.b1<-MASS::loglm(~1+2+3+2*3,1*2,
data=(casearray+delta), fit=F)
    caselog.b2<-MASS::loglm(~1+2+3+1*3+2*3,
data=(casearray+delta), fit=F)
    caselog.b3<-MASS::loglm(~1+2+3+2*3+1*3+1*2,
data=(casearray+delta), fit=F)

    p1<-stats::anova(caselog.b1, caselog.b3)[2,5]
    p2<-stats::anova(caselog.b2, caselog.b3)[2,5]
    p3<-unclass(summary(caselog.b3))$tests[2,3]

    p.values[i,1]<-NA
    p.values[i,3]<-unlist(p1)# Ist Partner daher Col==3
    p.values[i,2]<-unlist(p2)# Ist Actor daher Col==2
    p.values[i,4]<-unlist(p3)
  }

  b0<-stats::coef(caselog)$"3"[1]-
stats::coef(caselog)$"3"[2] #mean
  b1<-stats::coef(caselog)$"1.3"[1,1]-
stats::coef(caselog)$"1.3"[1,2] #partner

```



```

b2<-stats::coef(caselog)$"2.3"[1,1]-
stats::coef(caselog)$"2.3"[1,2] #actor
b3<-stats::coef(caselog)$"1.2.3"[1,1,1]-
stats::coef(caselog)$"1.2.3"[1,1,2] #interaction

lambdas[i,1]<-unlist(b0)
lambdas[i,3]<-unlist(b1) # Ist Partner daher Col==3
lambdas[i,2]<-unlist(b2) # Ist Actor daher Col==2
lambdas[i,4]<-unlist(b3)
}

output<-list()

output[[1]]<-lambdas
output[[2]]<-x
output[[3]]<-subgroups
output[[4]]<-p.values

attr(output, "firstSeq")<-attr(x, "firstSeq")

class(output)[1]<-"LogSeq"

return(output)
}

MLAP_Trans<-function(x) {

part1<-x[x$which.Dep==1,]

names(part1)<-c("which.seq",
              "DV",
              "Actor",
              "Partner",
              "ID")

part2<-x[x$which.Dep==2,c(1,2,4,3,5)]

```

```
names(part2)<-c("which.seq",
               "DV",
               "Actor",
               "Partner",
               "ID")

out<-rbind(part1, part2)
}

ML_Trans<-function(data, first, second){

  my.s<-StateExpand(data, first, second)

  myTrans1<-StateTrans(my.s)
  myTrans2<-StateTrans(my.s, first=FALSE)

  mylist<-list()

  for(k in 1:length(myTrans1)){

    which.Dep<-c(0)
    Dep<-c(0)
    V1<-c(0)
    V2<-c(0)

    ms<-data.frame(which.Dep, Dep, V1, V2)

    names(ms)<-c("which.Dep", "Dep", "Var1_at_t-1", "Var2_at_t-1")

    ms

    x<-as.numeric(myTrans1[[k]])

    if(x[1]!=0){
      for(i in 1:x[1]){
        ms<-rbind(ms, c(1,1,1,1))
      }
    }
  }
}
```

```
if(x[2]!=0){
  for(i in 1:x[2]){
    ms<-rbind(ms, c(1,1,1,0))
  }
}
```

```
if(x[3]!=0){
  for(i in 1:x[3]){
    ms<-rbind(ms, c(1,1,0,1))
  }
}
```

```
if(x[4]!=0){
  for(i in 1:x[4]){
    ms<-rbind(ms, c(1,1,0,0))
  }
}
```

```
if(x[5]!=0){
  for(i in 1:x[5]){
    ms<-rbind(ms, c(1,0,1,1))
  }
}
```

```
if(x[6]!=0){
  for(i in 1:x[6]){
    ms<-rbind(ms, c(1,0,1,0))
  }
}
```

```
if(x[7]!=0){
  for(i in 1:x[7]){
    ms<-rbind(ms, c(1,0,0,1))
  }
}
```

```
if(x[8]!=0){
  for(i in 1:x[8]){
    ms<-rbind(ms, c(1,0,0,0))
  }
}

ms<-ms[-1,]

# Zweite Variable
x<-as.numeric(myTrans2[[k]])

if(x[1]!=0){
  for(i in 1:x[1]){
    ms<-rbind(ms, c(2,1,1,1))
  }
}

if(x[2]!=0){
  for(i in 1:x[2]){
    ms<-rbind(ms, c(2,1,1,0))
  }
}

if(x[3]!=0){
  for(i in 1:x[3]){
    ms<-rbind(ms, c(2,1,0,1))
  }
}

if(x[4]!=0){
  for(i in 1:x[4]){
    ms<-rbind(ms, c(2,1,0,0))
  }
}

if(x[5]!=0){
  for(i in 1:x[5]){
    ms<-rbind(ms, c(2,0,1,1))
  }
}
```

```
    }  
  }  
  
  if(x[6]!=0){  
    for(i in 1:x[6]){  
      ms<-rbind(ms, c(2,0,1,0))  
    }  
  }  
  
  if(x[7]!=0){  
    for(i in 1:x[7]){  
      ms<-rbind(ms, c(2,0,0,1))  
    }  
  }  
  
  if(x[8]!=0){  
    for(i in 1:x[8]){  
      ms<-rbind(ms, c(2,0,0,0))  
    }  
  }  
  
  rownames(ms)<-as.character(1:length(ms[,1]))  
  
  mylist[[k]]<-ms  
}  
  
which.Dep<-c(0)  
Dep<-c(0)  
V1<-c(0)  
V2<-c(0)  
  
ms<-data.frame(which.Dep, Dep, V1, V2)  
  
names(ms)<-c("which.Dep", "Dep", "Var1_at_t-1", "Var2_at_t-1")  
  
for(i in 1:length(mylist)){  
  ms<-rbind(ms, mylist[[i]])  
}
```

```

}
ms<-ms[-1,]

ID<-c()
for(i in 1:length(mylist)){
  x<-rep(i, length(mylist[[i]][,1]))
  ID<-c(ID,x)
}
out<-cbind(ms,ID)
rownames(out)<-as.character(1:length(out[,1]))
out
}

NonCumHaz<-function(x, t=NA, plot=FALSE){
  if(!(any(class(x))=="survfit")||is.numeric(x))
    {warning("x_needs_to_be_a_numeric_vector_or_a_survfit-object!")}

  if(is.na(t) && plot && (!any(class(x))=="survfit"))
    {warning("plot_can_not_be_produced,
    _____because_time_reference_is_missing!")}

  if(is.numeric(x)){
    output<-numeric(length(x))
    for (i in 1:(length(x)-1)) output[i] <- x[1+i]-x[i]

    if(plot) {o1<-plot(output[1:(length(output)-1)]~
    t[1:(length(output)-1)],
    xlab="Time", ylab="Hazard", type="l")}
  }

  if(any(class(x))=="survfit"){
    my.cumhaz<-x$cumhaz
    my.cumhaz<-c(0,my.cumhaz)
    my.hazard<-c()
    for (i in 1:(length(my.cumhaz)-1)) my.hazard[i] <-
    my.cumhaz[1+i]-my.cumhaz[i]

    if(plot) {o1<-plot(my.hazard[1:(length(my.hazard)-1)]~

```

```

    x$time[1:(length(my.hazard)-1)], xlab="Time"
    , ylab="Hazard", type="l")
  output<-my.hazard
}

return(output)
}

NumbOccur<-function(x,y,t=NA, prop=TRUE){
  if(!(is.matrix(x)||is.data.frame(x)))
    warning("x_must_be_a_matrix_or_dataframe!")
  if(!(is.numeric(t)||is.na(t)))
    warning("t_must_be_NA_or_a_vector_with
    _number_elements_equal_number_sequences")
  if(!is.logical(prop)) warning("Argument_prob_must_be_logical")

  output<-numeric(length(x[,1]))
  index<-numeric(length(x[,1]))

  if(is.na(t[1])){
    index<-rep(length(x[,1]), length(x[,1]))
    for(k in 1:length(x[,1])){
      output[k] <- sum(x[k,])
    }
  }

  if(is.numeric(t)){ index<-t }

  for(k in 1:length(x[,1])){
    output[k]<-sum(as.numeric(x[k,1:index[k]]), na.rm=TRUE)
  }
  if(prop) output<-output/index

  return(output)
}

StateExpand<-function(x, pos1, pos2, replace.na=FALSE){
  l1<-length(pos1)
  l2<-length(pos2)

```

```

if(l1 != l2) warning("Both_sequences_must_have_the_same_length!")
x1<-as.matrix(x[,pos1])
x2<-as.matrix(x[,pos2])
output<-matrix(data=NA, nrow=length(x[,1]), ncol=l1)
output[x1==0 & x2==0]<-0
output[x1==1 & x2==0]<-1
output[x1==0 & x2==1]<-2
output[x1==1 & x2==1]<-3
if(is.numeric(replace.na)) output[is.na(output)]<-replace.na
class(output)[2]<-"state.expand"
return(output)
}

```

```

StateTrans<-function(x, first=TRUE, dep.lab=c("1","0"),
indep.lab=c("1-1","1-0","0-1","0-0")) {

```

```

  output<-list()
  comb<-x

```

```

  for(case in 1:length(x[,1])){

```

```

    y<-matrix(rep(0,8),4,2)
    colnames(y)<-dep.lab
    rownames(y)<-indep.lab

```

```

    if(first){

```

```

      for(i in 2:length(comb[case,])){
        if(comb[case,(i-1)]==0 &
          (comb[case,i]==1 | comb[case,i]==3)) y[4,1]<-y[4,1]+1
        if(comb[case,(i-1)]==1 &
          (comb[case,i]==1 | comb[case,i]==3)) y[2,1]<-y[2,1]+1
        if(comb[case,(i-1)]==2 &
          (comb[case,i]==1 | comb[case,i]==3)) y[3,1]<-y[3,1]+1
        if(comb[case,(i-1)]==3 &
          (comb[case,i]==1 | comb[case,i]==3)) y[1,1]<-y[1,1]+1
        if(comb[case,(i-1)]==0 &
          (comb[case,i]==0 | comb[case,i]==2)) y[4,2]<-y[4,2]+1
        if(comb[case,(i-1)]==1 &
          (comb[case,i]==0 | comb[case,i]==2)) y[2,2]<-y[2,2]+1

```



```

    if (comb[ case , (i - 1)] == 2 &
        (comb[ case , i] == 0 | comb[ case , i] == 2)) y[3,2] <- -y[3,2] + 1
    if (comb[ case , (i - 1)] == 3 &
        (comb[ case , i] == 0 | comb[ case , i] == 2)) y[1,2] <- -y[1,2] + 1}}
if (!first){
  for(i in 2:length(comb[ case , ])){
    if (comb[ case , (i - 1)] == 0 &
        (comb[ case , i] == 2 | comb[ case , i] == 3)) y[4,1] <- -y[4,1] + 1
    if (comb[ case , (i - 1)] == 1 &
        (comb[ case , i] == 2 | comb[ case , i] == 3)) y[2,1] <- -y[2,1] + 1
    if (comb[ case , (i - 1)] == 2 &
        (comb[ case , i] == 2 | comb[ case , i] == 3)) y[3,1] <- -y[3,1] + 1
    if (comb[ case , (i - 1)] == 3 &
        (comb[ case , i] == 2 | comb[ case , i] == 3)) y[1,1] <- -y[1,1] + 1
    if (comb[ case , (i - 1)] == 0 &
        (comb[ case , i] == 0 | comb[ case , i] == 1)) y[4,2] <- -y[4,2] + 1
    if (comb[ case , (i - 1)] == 1 &
        (comb[ case , i] == 0 | comb[ case , i] == 1)) y[2,2] <- -y[2,2] + 1
    if (comb[ case , (i - 1)] == 2 &
        (comb[ case , i] == 0 | comb[ case , i] == 1)) y[3,2] <- -y[3,2] + 1
    if (comb[ case , (i - 1)] == 3 &
        (comb[ case , i] == 0 | comb[ case , i] == 1)) y[1,2] <- -y[1,2] + 1}}
  output[[ case ]] <- y
  class(output)[2] <- "state.trans"
  attr(output, "firstSeq") <- first }
return(output)}

```

```

StateTrans <- function(x, first = TRUE, dep.lab = c("1", "0"),
  indep.lab = c("1-1", "1-0", "0-1", "0-0")) {
  output <- list()
  comb <- x
  for(case in 1:length(x[,1])){
    y <- matrix(rep(0, 8), 4, 2)
    colnames(y) <- dep.lab
    rownames(y) <- indep.lab

    if (first){
      for(i in 2:length(comb[ case , ])){
        if (comb[ case , (i - 1)] == 0 &
            (comb[ case , i] == 1 | comb[ case , i] == 3)) y[4,1] <- -y[4,1] + 1

```

```

if (comb[ case , (i - 1)]==1 &
      (comb[ case , i]==1 | comb[ case , i]==3)) y[2,1]<-y[2,1]+1
if (comb[ case , (i - 1)]==2 &
      (comb[ case , i]==1 | comb[ case , i]==3)) y[3,1]<-y[3,1]+1
if (comb[ case , (i - 1)]==3 &
      (comb[ case , i]==1 | comb[ case , i]==3)) y[1,1]<-y[1,1]+1

if (comb[ case , (i - 1)]==0 &
      (comb[ case , i]==0 | comb[ case , i]==2)) y[4,2]<-y[4,2]+1
if (comb[ case , (i - 1)]==1 &
      (comb[ case , i]==0 | comb[ case , i]==2)) y[2,2]<-y[2,2]+1
if (comb[ case , (i - 1)]==2 &
      (comb[ case , i]==0 | comb[ case , i]==2)) y[3,2]<-y[3,2]+1
if (comb[ case , (i - 1)]==3 &
      (comb[ case , i]==0 | comb[ case , i]==2)) y[1,2]<-y[1,2]+1
  }
}

if (!first){
  for(i in 2:length(comb[ case , ])){
    if (comb[ case , (i - 1)]==0 &
          (comb[ case , i]==2 | comb[ case , i]==3)) y[4,1]<-y[4,1]+1
    if (comb[ case , (i - 1)]==1 &
          (comb[ case , i]==2 | comb[ case , i]==3)) y[2,1]<-y[2,1]+1
    if (comb[ case , (i - 1)]==2 &
          (comb[ case , i]==2 | comb[ case , i]==3)) y[3,1]<-y[3,1]+1
    if (comb[ case , (i - 1)]==3 &
          (comb[ case , i]==2 | comb[ case , i]==3)) y[1,1]<-y[1,1]+1

    if (comb[ case , (i - 1)]==0 &
          (comb[ case , i]==0 | comb[ case , i]==1)) y[4,2]<-y[4,2]+1
    if (comb[ case , (i - 1)]==1 &
          (comb[ case , i]==0 | comb[ case , i]==1)) y[2,2]<-y[2,2]+1
    if (comb[ case , (i - 1)]==2 &
          (comb[ case , i]==0 | comb[ case , i]==1)) y[3,2]<-y[3,2]+1
    if (comb[ case , (i - 1)]==3 &
          (comb[ case , i]==0 | comb[ case , i]==1)) y[1,2]<-y[1,2]+1}}

  output[[ case ]]<-y

```

```

class(output)[2]<-"state.trans"

attr(output, "firstSeq")<-first

}
return(output)
}

```

```

TransToAPIM<-function(M){

  # transforming to actor & partner for the men
  DC_none<-sum(M[1,3:4]) # P(DC|none)
  DC_SC<-sum(M[2,3:4]) # P(DC|SC)
  DC_DC<-sum(M[3,3:4]) # P(DC|DC)
  DC_SC_DC<-sum(M[4,3:4]) # P(DC|SC+DC)

  # Transforming probabilities into logits
  DC_none_L<-log(DC_none/(1-DC_none))
  DC_SC_L<-log(DC_SC/(1-DC_SC))
  DC_DC_L<-log(DC_DC/(1-DC_DC))
  DC_SC_DC_L<-log(DC_SC_DC/(1-DC_SC_DC))

  # Transforming logits into betas
  DCb0<-sum(DC_none_L, DC_SC_L, DC_DC_L, DC_SC_DC_L)/4
  DCPart<-(DC_SC_L+DC_SC_DC_L)/2-DCb0
  DCAct<-(DC_DC_L+DC_SC_DC_L)/2-DCb0
  DCint<-DC_SC_DC_L-(DCb0+DCAct+DCPart)

  # For DC:
  B0_2<-DCb0 # Mean logit
  PE_2<-DCPart # Partner effect
  AE_2<-DCAct # Actor effect
  Int_2<-DCint # Interaction effect

  # transforming to actor & partner for the womens

  SC_none<-sum(M[1,c(2,4)]) # P(SC|none)
  SC_SC<-sum(M[2,c(2,4)]) # P(SC|SC)

```

```

SC_DC<-sum(M[3,c(2,4)]) # P(SC|DC)
SC_SC_DC<-sum(M[4,c(2,4)]) # P(SC|SC+DC)

SC_none_L<-log(SC_none/(1-SC_none))
SC_SC_L<-log(SC_SC/(1-SC_SC))
SC_DC_L<-log(SC_DC/(1-SC_DC))
SC_SC_DC_L<-log(SC_SC_DC/(1-SC_SC_DC))

SCb0<-sum(SC_none_L, SC_SC_L, SC_DC_L, SC_SC_DC_L)/4
SCAct<-(SC_SC_L+SC_SC_DC_L)/2-SCb0
SCPart<-(SC_DC_L+SC_SC_DC_L)/2-SCb0
SCint<-SC_SC_DC_L-(SCb0+SCAct+SCPart)

# For DC:
B0_1<-SCb0 # Mean logit
PE_1<-SCPart # Partner effect
AE_1<-SCAct # Actor effect
Int_1<-SCint # Interaction effect

results<-c(B0_1, AE_1, PE_1, Int_1,
           B0_2, AE_2, PE_2, Int_2)

names(results)<-c("Seq1_Intercept", "Seq1_Actor",
"Seq1_Partner", "Seq1_Interaction",
                 "Seq2_Intercept", "Seq2_Actor",
                 "Seq2_Partner", "Seq2_Interaction")

return(results)
}

simSeq<-function(trans, initial, length){

  init<-sample(c(1,2,3,4),1, prob=initial)

  for(i in 2:length){
    if(init[i-1]==1){
      init[i]<-sample(c(1,2,3,4),1, prob=trans[1,])
    } else if(init[i-1]==2){
      init[i]<-sample(c(1,2,3,4),1, prob=trans[2,])
    }
  }
}

```

```

    } else if (init[i-1]==3){
      init[i]<-sample(c(1,2,3,4),1, prob=trans[3,])
    } else {
      init[i]<-sample(c(1,2,3,4),1, prob=trans[4,])
    }
  }
  return(init)
}

```

```

simSeqSample<-function(trans, initial, length, N){
  comb<-simSeq(trans, initial, length)
  #split
  seq1<-c()
  seq1[comb==1|comb==3]<-0
  seq1[comb==2|comb==4]<-1
  seq2<-c()
  seq2[comb==1|comb==2]<-0
  seq2[comb==3|comb==4]<-1
  out<-c(seq1, seq2)
  for(i in 2:N){
    comb<-simSeq(trans, initial, length)
    seq1[comb==1|comb==3]<-0
    seq1[comb==2|comb==4]<-1
    seq2[comb==1|comb==2]<-0
    seq2[comb==3|comb==4]<-1
    out<-rbind(out, c(seq1, seq2))
  }
  return(out)
}

```

Bibliography

- Aalen, O., Borgan, O., and Gjessing, H. (2008). *Survival and event history analysis: a process point of view*. Springer Science & Business Media.
- Aassve, A., Billari, F. C., and Piccarreta, R. (2007). Strings of adulthood: A sequence analysis of young british women's work-family trajectories. *European Journal of Population/Revue européenne de Démographie*, 23(3-4):369–388.
- Abbott, A. (1995). Sequence analysis: new methods for old ideas. *Annual review of sociology*, pages 93–113.
- Abbott, A. and Tsay, A. (2000). Sequence analysis and optimal matching methods in sociology review and prospect. *Sociological methods & research*, 29(1):3–33.
- Agresti (2002). *Categorical Data Analysis, Second Edition*. Wiley Online Library.
- Aitkin, M. and Longford, N. (1986). Statistical modelling issues in school effectiveness studies. *Journal of the Royal Statistical Society. Series A (General)*, pages 1–43.
- Alcoholics Anonymous (2002). *Alcoholics Anonymous*. Hazelden Publishing.
- Anscombe, F. J. (1973). Graphs in statistical analysis. *The American Statistician*, 27(1):17–21.
- Asparouhov, T. and Muthen, B. (2007). Computationally efficient estimation of multilevel high-dimensional latent variable models. In *proceedings of the 2007 JSM meeting in Salt Lake City, Utah, Section on Statistics in Epidemiology*, pages 2531–2535.
- Axelrod, R. and Hamilton, W. D. (1981). The evolution of cooperation. *science*, 211(4489):1390–1396.
- Bakeman, R. and Gottman, J. M. (1997). *Observing interaction: An introduction to sequential analysis*. Cambridge university press.
- Bartolucci, F., Farcomeni, A., and Pennoni, F. (2012). *Latent Markov models for longitudinal data*. CRC Press.
- Bates, D., Maechler, M., Bolker, B., Walker, S., et al. (2014). lme4: Linear mixed-effects models using eigen and s4. *R package version*, 1(7).

- Blair-Loy, M. (1999). Career patterns of executive women in finance: An optimal matching analysis. *American journal of sociology*, 104(5):1346–1397.
- Bodenmann, G. (2005). Dyadic coping and its significance for marital functioning. *Couples coping with stress: Emerging perspectives on dyadic coping*, pages 33–50.
- Bodenmann, G., Meuwly, N., Germann, J., Nussbeck, F. W., Heinrichs, M., and Bradbury, T. N. (2015). Effects of stress on the social support provided by men and women in intimate relationships. *Psychological science*, page 0956797615594616.
- Bollen, K. A. and Barb, K. H. (1981). Pearson's r and coarsely categorized measures. *American Sociological Review*, pages 232–239.
- Bradley, P. S. and Fayyad, U. M. (1998). Refining initial points for k-means clustering. In *ICML*, volume 98, pages 91–99.
- Breslow, N. (1974). Covariance analysis of censored survival data. *Biometrics*, pages 89–99.
- Briggs, M. A. and Sculpher, M. (1998). An introduction to markov modelling for economic evaluation. *Pharmacoeconomics*, 13(4):397–409.
- Browne, W. J., Draper, D., et al. (2006). A comparison of bayesian and likelihood-based methods for fitting multilevel models. *Bayesian analysis*, 1(3):473–514.
- Burns, A. C. and Bush, R. F. (2007). *Basic marketing research using Microsoft Excel data analysis*. Prentice Hall Press.
- Canty, A. and Ripley, B. (2012). boot: Bootstrap r (s-plus) functions. *R package version*, 1(7).
- Casper, G. and Wilson, M. (2015). Using sequences to model crises. *Political Science Research and Methods*, 3(02):381–397.
- Chernick, M. R., González-Manteiga, W., Crujeiras, R. M., and Barrios, E. B. (2011). Bootstrap methods. In *International Encyclopedia of Statistical Science*, pages 169–174. Springer.
- Cockerill, I. M., Nevill, A. M., and Lyons, N. (1991). Modelling mood states in athletic performance. *Journal of Sports Sciences*, 9(2):205–212.
- Cook, W. L. and Kenny, D. A. (2005). The actor–partner interdependence model: A model of bidirectional effects in developmental studies. *International Journal of Behavioral Development*, 29(2):101–109.
- Cooley, W. W. and Lohnes, P. R. (1971). *Multivariate data analysis*. J. Wiley.
- Cox, D. R. (1972). Vregression models and life tables. *V Journal of the Royal Statistical Society, Series B*, 34(2):187.

- Cronbach, L. J. and Gleser, G. C. (1953). Assessing similarity between profiles. *Psychological bulletin*, 50(6):456.
- Crowder, M. J. (2012). *Multivariate survival analysis and competing risks*. CRC Press.
- Davison, A. C. and Hinkley, D. V. (1997). Bootstrap methods and their applications, Cambridge series in statistical and probabilistic mathematics. *Cambridge University Press*, 32:10013–2473.
- Dawes, R. M. (1980). Social dilemmas. *Annual review of psychology*, 31(1):169–193.
- Efron, B. (1977). The efficiency of cox's likelihood function for censored data. *Journal of the American statistical Association*, 72(359):557–565.
- Efron, B. and Tibshirani, R. J. (1994). *An introduction to the bootstrap*. CRC press.
- Eid, M., Gollwitzer, M., and Schmitt, M. (2010). *Statistik und Forschungsmethoden*.
- Eid, M. E. and Diener, E. E. (2006). *Handbook of multimethod measurement in psychology*. APA.
- Ess, C. and Sudweeks, F. (2001). *Culture, technology, communication: Towards an intercultural global village*. Suny Press.
- Everitt, B. S. (1992). *The analysis of contingency tables*. CRC Press.
- Fuller, R. K. (1997). Definition and diagnosis of relapse to drinking. *Liver Transplantation and surgery*, 3(3):258–262.
- Fuller, S. and Stecy-Hildebrandt, N. (2015). Career pathways for temporary workers: Exploring heterogeneous mobility dynamics with sequence analysis. *Social science research*, 50:76–99.
- Gabardinho, A., Ritschard, G., Studer, M., and Müller, N. S. (2009). Mining sequence data in R with the traminer package: A users guide for version 1.2. *Geneva: University of Geneva*.
- Gill, J. (2000). *Generalized linear models: a unified approach*, volume 134. Sage Publications.
- Goldstein, H. (1989). Restricted unbiased iterative generalized least-squares estimation. *Biometrika*, 76(3):622–623.
- Gonzalez, R. and Griffin, D. (2012). Dyadic data analysis. *APA handbook of research methods in psychology*, 3:439–450.
- Google Trends (2018). Data source: Google trends (www.google.com/trends). Compare: hierarchical clustering (topic); k-means clustering (topic); cluster analysis (topic); time range 5 years; retrieved at 2018-01-06T13:30:00.000+0100.
- Granger, C. W. and Newbold, P. (1974). Spurious regressions in econometrics. *Journal of econometrics*, 2(2):111–120.

- Halstenberg, E. (2016). *B.Sc. Thesis: The effect of Social Value Orientation on cooperation in a Four-Coin Dilemma: a quasi-replication study using the SVO Slider Measure, University Bielefeld, 2016.*
- Helske, S. and Helske, J. (2016). Mixture hidden markov models for sequence data: the seqhmm package in r.
- Hertz-Picciotto, I. and Rockhill, B. (1997). Validity and efficiency of approximation methods for tied survival times in cox regression. *Biometrics*, pages 1151–1156.
- Hope, A. C. (1968). A simplified monte carlo significance test procedure. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 582–598.
- Hosmer, D. W. and Lemeshow, S. (1999). *Applied survival analysis: Regression modelling of time to event data (1999).*
- Hox, J. and Stoel, R. D. (2005). Multilevel and sem approaches to growth curve modeling. *Wiley StatsRef: Statistics Reference Online.*
- Hox, J. J., Moerbeek, M., and van de Schoot, R. (2010). *Multilevel analysis: Techniques and applications.* Routledge.
- Johnson, D. R. and Creech, J. C. (1983). Ordinal measures in multiple indicator models: A simulation study of categorization error. *American Sociological Review*, pages 398–407.
- Kalbfleisch, J. D. and Prentice, R. L. (1973). Marginal likelihoods based on cox's regression and life model. *Biometrika*, 60(2):267–278.
- Kenny, D., Kashy, D., and Cook, W. (2006). *The analysis of dyadic data.* New York: Guilford.
- Kenny, D. A. (1996). Models of non-independence in dyadic research. *Journal of Social and Personal Relationships*, 13(2):279–294.
- Kenny, D. A. and Judd, C. M. (1986). Consequences of violating the independence assumption in analysis of variance. *Psychological Bulletin*, 99(3):422.
- Kirschbaum, C., Pirke, K.-M., and Hellhammer, D. H. (1993). The 'trier social stress test'—a tool for investigating psychobiological stress responses in a laboratory setting. *Neuropsychobiology*, 28(1-2):76–81.
- Kucherenko, S. and Sytsko, Y. (2005). Application of deterministic low-discrepancy sequences in global optimization. *Computational Optimization and Applications*, 30(3):297–318.
- Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *biometrics*, 33(1):159–174.
- Lazarus, R. S. (1966). Psychological stress and the coping process.

- Lazarus, R. S. and Launier, R. (1978). Stress-related transactions between person and environment. In *Perspectives in interactional psychology*, pages 287–327. Springer.
- Ledermann, T. and Kenny, D. A. (2012). The common fate model for dyadic data: variations of a theoretically important but underutilized model. *Journal of Family Psychology*, 26(1):140.
- Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707.
- Levitt, B. and Nass, C. (1989). The lid on the garbage can: Institutional constraints on decision making in the technical core of college-text publishers. *Administrative Science Quarterly*, pages 190–207.
- Liu, D. C. and Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.
- Longford, N. T. (1987). A fast scoring algorithm for maximum likelihood estimation in unbalanced mixed models with nested random effects. *Biometrika*, 74(4):817–827.
- Lucas, R. E. and Baird, B. M. (2006). Global self-assessment. In Diener, M. E. . E., editor, *Handbook of Multimethod Measurement in Psychology*, pages 29–42. Washington DC: APA.
- Luce, R. D. and Raiffa, H. (1957). Games and decisions. *New York, John Wiley Sons*.
- Maas, C. J. and Hox, J. J. (2005). Sufficient sample sizes for multilevel modeling. *Methodology*, 1(3):86–92.
- Messick, D. M. and McClintock, C. G. (1968). Motivational bases of choice in experimental games. *Journal of experimental social psychology*, 4(1):1–25.
- Mills, M. (2011). *Introducing survival and event history analysis*. Sage Publications.
- Milovsky, N. (2016). *The Basics of Game Theory and Associated Games*. Retrieved September 7, 2016, from https://issuu.com/johnsonnick895/docs/game_theory_paper.
- Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The computer journal*, 7(4):308–313.
- Peduzzi, P., Concato, J., Feinstein, A. R., and Holford, T. R. (1995). Importance of events per independent variable in proportional hazards regression analysis ii. accuracy and precision of regression estimates. *Journal of clinical epidemiology*, 48(12):1503–1510.
- Poundstone, W. (2011). *Prisoner's dilemma*. Anchor.
- Powell, M. J. (2009). The bobyqa algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*.

- Rapoport, A. (1963). Mathematical models of social interaction.
- Raudenbush, S. W., Brennan, R. T., and Barnett, R. C. (1995). A multivariate hierarchical model for studying psychological change within married couples. *Journal of Family Psychology*, 9(2):161.
- Rondeau, V., Mazroui, Y., and Gonzalez, J. R. (2012). frailtypack: an r package for the analysis of correlated survival data with frailty models using penalized likelihood estimation or parametrical estimation. *Journal of Statistical Software*, 47(4):1–28.
- Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.
- Rousseeuw, P. J. and Kaufman, L. (1990). *Finding Groups in Data*. Wiley Online Library.
- Sally, D. (1995). Conversation and cooperation in social dilemmas a meta-analysis of experiments from 1958 to 1992. *Rationality and society*, 7(1):58–92.
- Schwarz, G. et al. (1978). Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464.
- Searle, S. R., Casella, G., and McCulloch, C. E. (2009). *Variance components*, volume 391. John Wiley & Sons.
- Sebanz, N., Bekkering, H., and Knoblich, G. (2006). Joint action: bodies and minds moving together. *Trends in cognitive sciences*, 10(2):70–76.
- Seligman, M. E. (1972). Learned helplessness. *Annual review of medicine*, 23(1):407–412.
- Shubik, M. (1964). *Game Theory: And Related Approaches to Social Behavior*. John Wiley & Sons.
- Singer, J. D. and Willett, J. B. (2003). *Applied longitudinal data analysis: Modeling change and event occurrence*. Oxford university press.
- State (2016). *Oxford Dictionaries*. Retrieved September 7, 2016, from <http://www.oxforddictionaries.com/definition/english/state>.
- Steinley, D. (2003). Local optima in k-means clustering: What you don't know may hurt you. *Psychological Methods*, 8(3):294 – 304.
- Stelzl, I. (1986). Changing a causal hypothesis without changing the fit: Some rules for generating equivalent path models. *Multivariate Behavioral Research*, 21(3):309–331.
- Steyer, R., Schmitt, M., and Eid, M. (1999). Latent state–trait theory and research in personality and individual differences. *European Journal of Personality*.

- Struyf, A., Hubert, M., Rousseeuw, P., et al. (1997). Clustering in an object-oriented environment. *Journal of Statistical Software*, 1(4):1–30.
- Studer, M., Ritschard, G., Gabadinho, A., and Müller, N. S. (2011). Discrepancy analysis of state sequences. *Sociological Methods and Research*, 40(3):471–510.
- Tofighi, D. and Enders, C. K. (2008). Identifying the correct number of classes in growth mixture models. *Advances in latent variable mixture models*, (Information Age Publishing, Inc):317–341.
- Vermunt, J. K. (1997). Lem: A general program for the analysis of categorical data. *Department of Methodology and Statistics, Tilburg University*.
- Vermunt, J. K. and Hagnaars, J. A. (2004). 15 ordinal longitudinal data analysis. *Methods in human growth research*, 39:374.
- Visser, I., Speekenbrink, M., et al. (2010). depmixs4: An r-package for hidden markov models. *Journal of Statistical Software*, 36(7):1–21.
- Vittinghoff, E. and McCulloch, C. E. (2007). Relaxing the rule of ten events per variable in logistic and cox regression. *American journal of epidemiology*, 165(6):710–718.
- Wuerker, A. K. (1996a). The changing careers of patients with chronic mental illness: a study of sequential patterns in mental health service utilization. *The Journal of Behavioral Health Services and Research*, 23(4):458–470.
- Wuerker, A. M. (1996b). Communication patterns and expressed emotion in families of persons with mental disorders. *Schizophrenia bulletin*, 22(4):671.
- Yu, S.-Z. (2010). Hidden semi-markov models. *Artificial Intelligence*, 174(2):215–243.