Computational Statistics Quarterly 4,325-353(1991)
© Physica-Verlag 1991

How to Make the Teaching of Statistics Roar

Some Thoughts on Computer Based Experiments

P. Naeve
Department of Economics, University of Bielefeld, P. O. Box 8640, D-4800 Bielefeld 1

D. Trenkler
Department of Economics, University of Osnabrück, P. O. Box 4469, D-4500 Osnabrück

H. P. Wolf
Department of Economics, University of Bielefeld, P. O. Box 8640, D-4800 Bielefeld 1

SUMMARY
The application of computers in teaching statistics offers new possibilities. Two kinds of experiments are introduced. Firstly, experiments to demonstrate statistical concepts by exploiting the graphical capabilities of computers. Secondly, experiments which may help teach data analysis strategies. Examples for both kinds are given. First steps towards of experiment construction are developed. The usefulness of *Literate APL* is demonstrated too.

KEYWORDS: Teaching of statistics, APL, computer in teaching, statistical experiments.

1 The Teaching of Statistics.

"What is mathematics?" is the title of a famous book by Courant and Robbins [3]. As far as we know a book entitled "What is statistics?" has not been written as yet. Nevertheless the question has to be answered. It seems to be a convincing statement to say that statistics is what is taught. Really? It is just a shift in the problem space. "How to teach statistics?" is an ever returning theme in many a paper written by statisticians. See for instance the long row of presidential addresses to the Royal Statistical Society. To give one example let us listen to Professor Daniels [4].

Statistics is not mathematics, not even applied mathematics. It concerns the acquisition, interpretation and exploitation of data, and this may include some shrewd non-mathematical guesswork.

To make a long story short statistics is more than a collection of methods, procedures etc. based on theory. Statistics is doing statistics.

How can this be achieved? There are a number of proposals. We would like to add (a new?) one. Let us exploit the possibilities offered by the computers. This does not sound very new for computers are on the market and in statistical laboratories and institutes for quite a long time. But leaving some research activities aside the situation is best characterised as very poor use of computers so far. In 1976 the first author gave the opening speach at the 2. Compstat symposium in Berlin. His paper was centered around the following statements [10].

Firstly : *Statistics is taught as if we were living in the precomputer age.*
Secondly : *Computers enter the field of teaching statistics twofold: as a tool and as a medium.*
Thirdly : *One need not be an expert in computer science to take computers into consideration when teaching statistics.*

Sorry to say, not much has been changed since then. The time is ripe to do something about this so we want to propose two kinds of experiments in this paper which can be done on the computer. We think that the teaching of statistics can be improved considerably by this line of approach.

An experiment of type 1 is meant for learning statistical concepts. Understanding concepts seems to be a difficult and therefore disliked task. Many students are eager to accept certain assumptions and hurry to apply formulas which they hope to be appropriate. This attitude is well documented by the following quotation of one of our students:

To pass the examination is it necessary to understand *the concepts or is it sufficient to know how to* apply *them?*

Experiments we are thinking of can be used by the teacher for demonstrations and by the student to get a better understanding of the underlying concepts by self-paced experimentation because the experiments will allow for different parameter settings. Here the computer acts mostly as a medium.

An experiment of type 2 is intended for learning statistical strategies — or to say it more simply — for doing statistics with real data. Here the computer is mainly a tool providing statistical methods in a handy way, taking over the burden of housekeeping and preparing nice output especially in a graphical form.

It is one thing to make some proposals and yet another to show that it can be achieved. We dislike people who never reach the second state. Therefore, we will demonstrate our ideas at work — at least to such an extent as it can be done without a computer at hand. But hopefully our paper will convey the flavour of a real computer session well enough so that the reader is convinced that we can do it and that it is worth doing. The many opportunities of modern hard- and software technology offer a great potential for improving statistical education. The programming language *APL* and graphical facilities are indespensable means for our work. Exploiting them on a larger scale may eventually make the teaching of statistics more roaring[1]. The outline of the rest of our paper is as follows.

- Examples of type 1 experiments
- Proof versus experiment
- An example of an type 2 experiment
- A theory of experiment construction
- How it was done
- Writing a paper not a program

2 Examples of Type 1 Experiments.

Let us take the problem of random number generation as a first example. This is not a hard problem for a theoretical statistician for he is aware of the following theorem.

Theorem: Assume that U is uniformly distributed on (0,1).

(i) Let X be a discrete random variable with distribution function F_X and mass points x_1, x_2, x_3,... Define a random variable Y by $(Y = x_i) \Leftrightarrow U \in (F_X(x_{i-1}), F_X(x_i)]$. Then X and Y are equally distributed.

(ii) Let X be a continuous random variable with strictly increasing distribution function F_X. Then X and $F_X^{-1}(U)$ are identically distributed.

This theorem gives room for the following procedure. The random numbers are produced in two steps:

1. Draw a random sample of size n from a set of uniformly distributed random variables $u_1, \ldots, u_n$.
2. Deliver $x_1, \ldots, x_n$, with $x_i = F^{-1}(u_i)$.

$x_1, \ldots, x_n$ can be regarded as a random sample from a population whose cumulative distribution function is F.

Even though this formulation is correct it is too terse and only a minority of students if any will comprehend its implications. Usually the beginner does not *see* this procedure and, consequently, he is reluctant to apply it. But an eyecatching picture can help. The

[1]"Roaring" means not only absence of boredom but power.

following experiment is designed to demonstrate graphically the procedure contained in the theorem mentioned above.

The idea is to divide the procedure into single steps in order to gain a dynamic computer experiment. We will demonstrate this approach by considering the task of generating exponentially distributed random numbers. Just as a reminder we will give the following Definition: *X is called exponentially distributed, if the distribution function can be written as $F(x;\lambda) = 1 - e^{-\lambda x}$. Its inverse function is given by $F^{-1}(u) = -\ln(1-u)/\lambda$.*

We consider four ingredients for our experiment:

1. The drawing of uniform random numbers.
2. The inversion process.
3. The piling up of exponential random numbers.
4. The target distribution.

Let us split the screen into four parts in which the respective aspects will be demonstrated.

N-W-corner	**N-E-corner**
S-W-corner	**S-E-corner**

These parts are used as follows.

N-W: Here the histogram from the sample $u_1, \ldots, u_n$ is created. Every draw adds a small piece to a pillar of the histogram.

N-E: In this part the inverting process is represented, that is $x = F^{-1}(u)$. Every time a u is drawn, there appears from u of the y-axis a line which runs to the permanent visible graph of the function F. From the point of intersection there appears a line which runs vertically towards the x-axis and touches it in x. The elements of the random sample slides along this lines from the uniform into the exponential distribution.

S-E: In this part, the histogram that results from the transformed u_i is built up synchronously in time with the histogram from the random sample of the uniform distribution. The graph of the density function of the exponential distribution is added after the drawings are done.

S-W: In this place the graph of the density of the aspired exponential distribution is displayed to see our goal.

We will present three hardcopies taken during a run of this experiment. We will draw a sample of size $n = 50$ from the exponential distribution with $\lambda = 1$, one from the beginning, one from the middle and one from the end of the experiment.

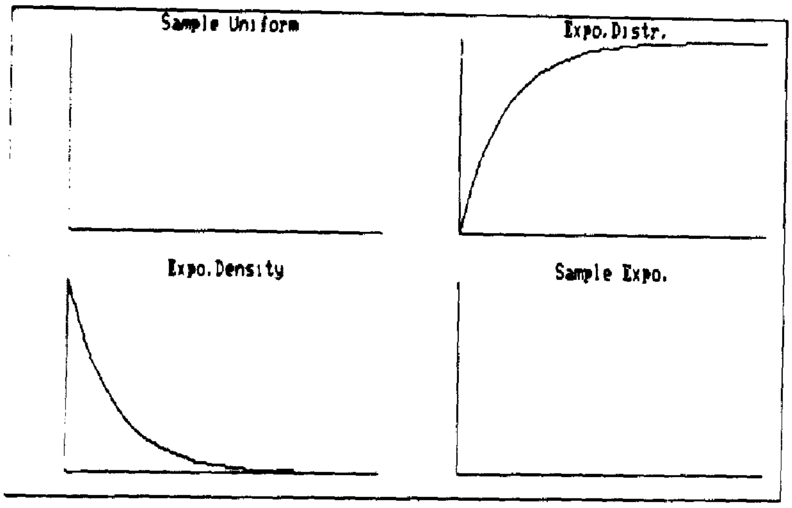


Figure 1: Display at the start of the experiment.

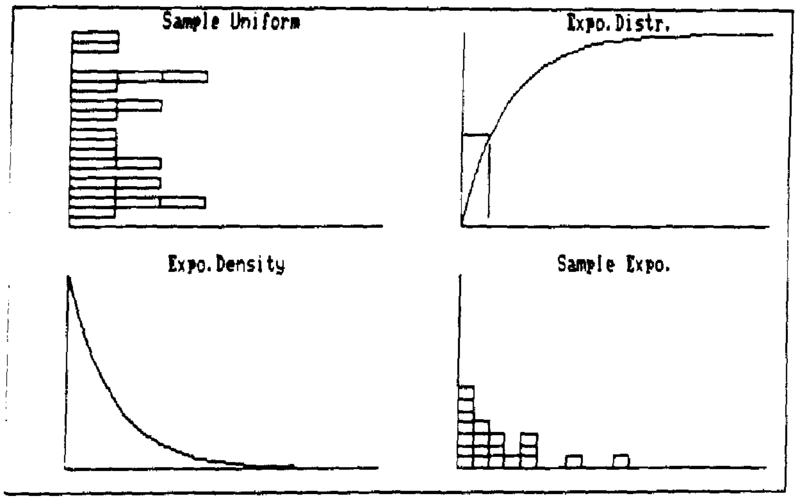


Figure 2: Display at the middle of the experiment.

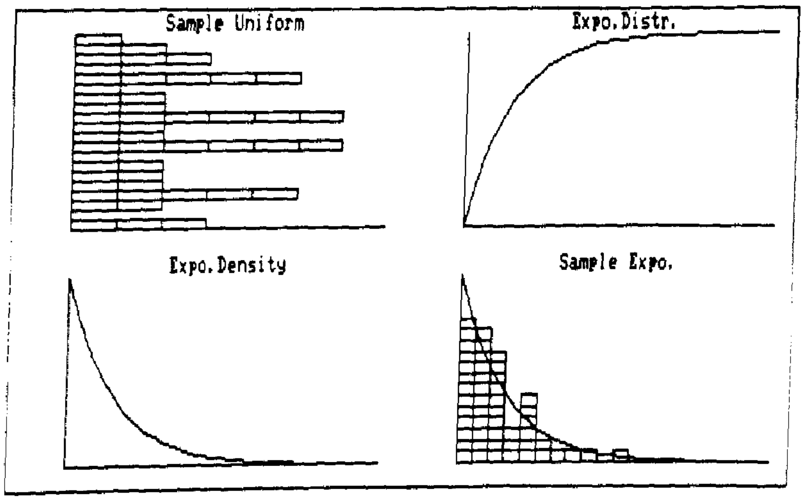


Figure 3: Display at the end of the experiment.

Naturally this is a poor substitute of the impression one gets from the procedure if you follow the dynamic movements on the screen. Such experiments not only just highlight the procedure but raise a lot of questions. For instance: What influence has the quality of the uniformly distributed random values to the exponential random values? How are uniformly distributed random numbers created? How are corresponding generators built? How to measure the quality of random number generators? What influence has the size of the random sample? etc... So, one experiment might initiate a whole bunch of experiments to follow.

Here is another experiment. The statistics $\bar{X}$ and $\widehat{\sigma^2} = \frac{1}{n-1}\sum(X_i - \bar{X})^2$ are both unbiased estimators of λ in the case of a Poisson distribution. Which is the one to be preferred? Fifty random samples of size $n = 5$ from a Poisson distribution with $\lambda=1$ were generated. For each random sample $\bar{x}$ and $\widehat{\sigma^2}$ were computed. A comparison is made by comparative stem and leaf displays.

<u>Stem amd leaf display of $\bar{x}$</u>

```
Factor of Stem: 1
1|2 represents 1.2

   3   0 | 024
  12   0 | 666666888
(27)   1 | 000000000000222222222224444
  11   1 | 66666688
   3   2 | 000
```

<u>Stem amd leaf display of $\widehat{\sigma^2}$</u>

```
Factor of Stem: 1
1|2 represents 1.2

  12   0 | 022333333333
  24   0 | 555555777778
( 9)   1 | 002223333
  17   1 | 55557777778
       2 |
   6   2 | 557
   3   3 | 3
   2   3 | 5
       4 |
       4 |
       5 |
       5 |
   1   6 | 3
```

Even though both estimators are unbiased their individual distributional properties have to be taken into account. Applying this experiment the student is forced to recapture concepts like distributional shape, central limit theorem, expectation, variability, mean square error and so on in the context of parameter estimation. Later on, he may be introduced to the idea of uniformly minimum variance unbiased estimation. In fact, it be shown that $\bar{X}$ is $UMVUE$ for λ in the case of a Poisson distribution and experiments like this can elucidate that it is so.

3 Proof Versus Experiment.

To prove or not to prove — that is the question teachers of statistics often have to face. Presumably there are many people who are reluctant to our line of approach. A proof is a proof and should not be replaced by such ambiguous things like a picture. Such a weak approach however offers some advantages since it may alleviate the understanding of theoretical facts and provoke the longing to prove them. And is a proof given by an author always a proof for the student? Let us consider an example.

When sampling from a normal distribution one can rely on the independence of the sample mean and variance. But how do textbooks cope with the problem of proving this fact?

One of our favourite books is that by Mood and Graybill [8]. To prove the proposition they use the joint moment generating function $m(t_1, t_2)$ of the random variables U and V which are given by

$$U = \frac{1}{n}\left(\sum_{i=1}^{n} Y_i\right)^2 \qquad V = \sum_{i=1}^{n}(Y_i - Y)^2$$

where n is the sample size and Y_i is the (theoretical) standardized random sample variable. They show after some manipulations of a quadratic form and exploitation of a special determinant that $m(t_1, t_2)$ can be factored as

$$m(t_1, t_2) = \left(\frac{1}{1-2t_1}\right)^{\frac{1}{n}} \left(\frac{1}{1-2t_2}\right)^{\frac{n-1}{2}} .$$

Now all (!) the reader has to do is to follow some arguments found in the theory of moment generating functions and he can state *quod erat demonstrandum.*

But if one carefully rereads the proof — it is spread over three pages of their book — one comes to the conclusion that for the beginner too many dependencies on other facts of statistical theory are included. It calls more for belief than being a proof. This is especially evident if one reads the following lines:

All the results of this section apply only to normal populations. It can be proved that for no other distributions are (1) the sample mean and sample variance independently distributed or (2) the sample mean exactly normally distributed.

Why not teach it the following way. State the propositions mentioned before and then show two scatter diagrams. One displays 50 points $(\bar{x}, s^2)$ based on samples from a normal disribution each having size 50 while the other shows the the same for an exponential distribution. Here they are:

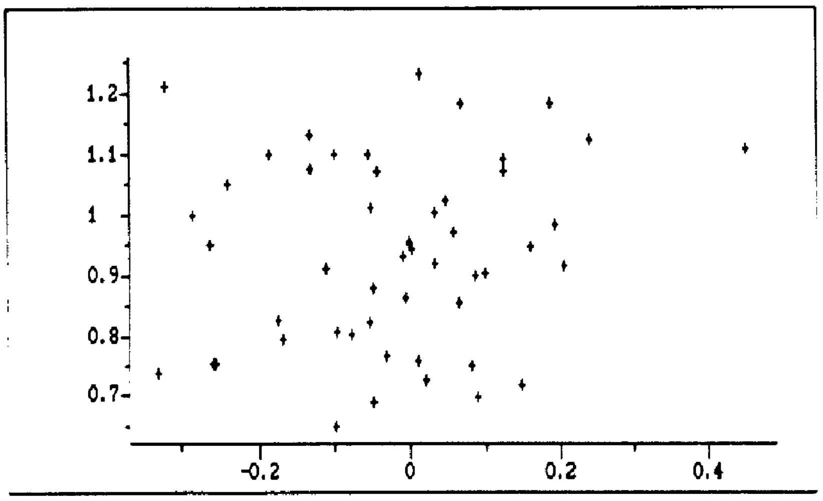


Figure 4: Scatter plot of $(\bar{x}, s^2)$ coming from a normal distribution.

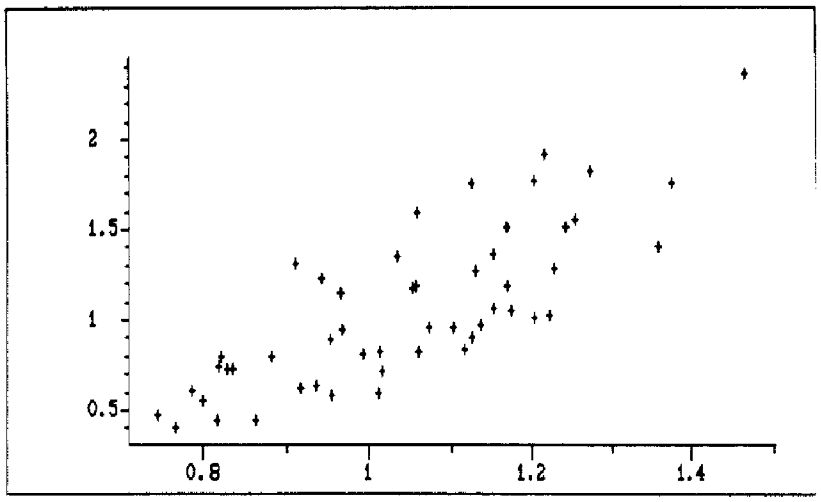


Figure 5: Scatter plot of $(\bar{x}, s^2)$ coming from an exponential distribution.

Clearly this is not a proof but we believe the beginner gets a better understanding what is said in the cited quotation from Mood and Graybill. To be fair let us look how the subject is dealt with after Boes joined the team [9]. They still use moment generating function arguments. But now they apply them to the special case $n = 2$. Here you succeed if you can prove that $Y_1 + Y_2$ and $Y_2 - Y_1$ are independent. The quotation given above is in the new edition too. And we still feel pity for the beginner.

We are not opposed to proving things. But to follow the lines of a proof usually calls for an amount of other theoretical facts the reader must be willing to rely on. We doubt if the beginner is always equipped in the appropriate way. So our line of approach certainly can help him see — to us this is better than just to believe. Even though we clearly do not give a proof students are likely to get a feeling of what this is all about. After all they have to accept certain theoretical facts later on which they will never be forced to *prove*. What they have to learn is that statistical methods are based on certain assumptions and that just conclusions may heavily rest upon their validity. Furthermore, examples like this enable to go one step further since now we can pose questions like "Do we have enough evidence to claim that $\bar{X}$ and S^2 are (not) independent?" provoking the need for tools like the χ^2 test of independence.

4 An Example of Type 2 Experiment

Statistical inference represents another area where computer simulation can excel. As Thompson [13] puts it

...the current generation of 32 bit chips can bring about the real computer revolution and change fundamentally the ways in which we approach the task of modeling and problem solving. ...the proliferation of fast computing to the desktop will encourage private developers to develop simulation-based procedures for a large and growing market of users who need to get from specific problems to useful solutions in the shortest time possible. We now have the ability to use the computer not as a fast calculator but as a device that changes the process of going from the microaxioms to the macrorealization.

As an example to conduct an experiment of type 2 let us take the following data set which are prices (DM) and ratings of quality for bicycles from 18 producers, [12]. The ratings range from 1 (good) to 5 (defective).

Price	Rating	Price	Rating	Price	Rating
270	5	509	3	660	5
300	5	510	3	700	3
400	3	550	3	720	1
450	3	610	2	750	3
499	3	615	3	790	1
500	5	630	2	800	2

Here is a scatter plot of this data set.

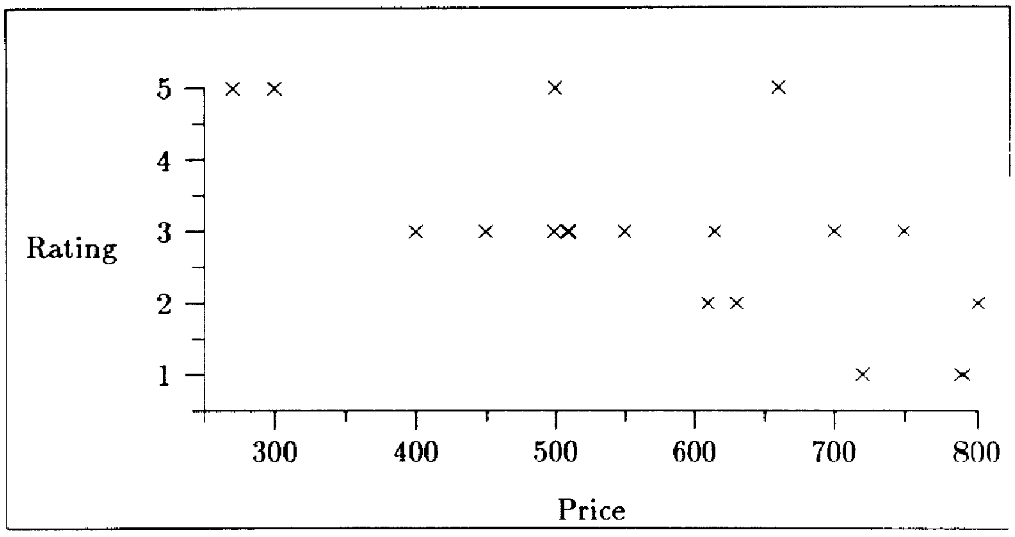


Figure 6: Prices and ratings of 18 bicycles.

The question arises whether there is a relationship between prices and ratings. To be precise it can be suspected that higher prices will cause the associated ratings to fall. But how can we get evidence?

The first thought coming into our mind is using Pearson's coefficient

$$r_P = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}.$$

For this data set we compute $r_P = -0.6511$. But what is the interpretation of getting this value? Significance tests based on r_P often make the assumption that the joint distribution of (X, Y) is a bivariate normal, [2]. This assumption is obviously too bold in this case since ratings are measured on an ordinal scale. As teachers of statistics we can recommend this approach only with mixed feelings since we *hope* that our conclusions are approximately valid.

A more appropriate approach would be to use Spearman's rank correlation coefficient

$$r_S = \frac{\sum(r_i - \bar{r})(r'_i - \bar{r}')}{\sqrt{\sum(r_i - \bar{r})^2 \sum(r'_i - \bar{r}')^2}}$$

where r_i=rank(x_i) and r'_i=rank(y_i). For this data set we compute $r_S = -0.5155$ using average ranks. Yet again, how can we interpret this value? Tables for the distribution of r_S as high as $n = 18$ are available assuming the existence of *no ties*. In fact, every tie constellation leads to an individual distribution of r_S and it is not clear to what extent the data situation at hand with that many ties can be treated as that without any. Furthermore, we must be aware that we may lose relevant information by using ranks r_i instead of the original x_i.

To circumvent this difficulty note that we need to know just how r_P is distributed if X and Y *are independent.* In other words a permutation test seems appropriate. This amounts to computing all values of

$$r_\pi = \frac{\sum(x_i - \bar{x})(\pi_i - \bar{\pi})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(\pi_i - \bar{\pi})^2}}$$

where $(\pi_1, \ldots, \pi_n)$ is one of the $18!$=6402373705728000 permutations of $(y_1, \ldots, y_n)$. This certainly is not a feasible task. On the other hand if we believe in the efficiency of statistical methods why not apply them to draw a conclusion. What we *can* do is to sample a certain amount of permutations and to judge on the basis of this *experiment* how likely it is to observe such an extreme data situation like the one we have at hand.

The following five steps can be viewed as a proposal to get an insight into the rationale of statistical tests.

(i) Set up an alternative hypothesis H_1 which we believe is true. Act as if it is *not* true, i.e. the null hypothesis H_0 is true. In this case we suppose that

H_1: High prices tend to be associated with low ratings.

Thus we work under

H_0: There is no relationship whatsoever between prices and ratings.

(ii) Which test statistic should be used to come to a decision? For our example r_π seems appropriate. A moment's thought shows that $T = \sum x_i \pi_i$ will lead to equivalent conclusions and this is the one we will use.

(iii) Let t_0 be the value of T computed from the data. Asking if our data situation is extreme under independence leads to asking if t_0 is an extreme value under H_0. This question can initiate the discussion of two approaches: the classical and the "nonparametric" one. The classical approach starts with a fixed level of significance α and t_0 will be defined as extreme if $P(T \le t_0) \le \alpha$. On the other hand the nonparametric approach amounts to assessing the critical level $\tilde{\alpha} = P(T \le t_0)$. While the classical approach assumes our knowledge of the H_0-distribution of T the latter offers more flexibility in that it is sufficient to give an estimate of $\hat{\alpha} = \hat{\alpha}(t_0)$ or even better a confidence bound $(0, \hat{\alpha}_u]$ of $\tilde{\alpha}$. Depending on t_0 we judge the data situation as extreme if we consider $\hat{\alpha}$ or $\hat{\alpha}_u$ as "too small".

(iv) Suppose there is a (18,2)-matrix ***bike*** whose first column contains the prices and the second the ratings. The expression

```
      +/×/bike
    29189
```

computes t_0.

To get a visual impression of how extreme t_0 is a sample of $M = 1000$ random permutations is generated. To this end suppose there is function ***RANDOMPERMUTATIONS_OF_SIZE***. Typing

```
      treal←bike[1000 RANDOMPERMUTATIONS_OF_SIZE 18;1]+.×bike[;2]
```

renders a vector ***treal*** of 1000 realisations t of T. A function ***STEM_AND_LEAF*** does the job of displaying their distribution in form of a stem-and-leaf display. Here we present a polished version of the output.

<u>Stem amd leaf display of 1000 realisations of $T = \sum x_i \pi_i$</u>

```
Factor of Stem: 1000

1|2 represents 1200

    1  28 | 9
    2  29 | 0
    9  29 | 2233333
   16  29 | 4555555
   30  29 | 66666677777777
   52  29 | 8888899999999999999999
   85  30 | 000000000000111111111111111111111
  131  30 | 2222222222222222222333333333333333333333333333
  195  30 | 4444444444444444444444444444444555555555555555555555555555555555
  257  30 | 66666666666666666666666666666677777777777777777777777777777777
  337  30 | 88888888888888888888888888888888888999999999999999999999999999999999999999999999
  425  31 | 0000000000000000000000000000000000000111111111111111111111111111111111111111111111111111
 (96)  31 | 222222222222222222222222222222222222222233333333333333333333333333333333333333333333333333333333
  479  31 | 444444444444444444444444444444444444444444444444455555555555555555555555555555555555555
  392  31 | 66666666666666666666666666666666666666666666666777777777777777777777777777777777777777
  306  31 | 888888888888888888888888888888888888888888888999999999999999999999999999999999999999999999
  216  32 | 0000000000000000000000000000000000011111111111111111111111111111111
  149  32 | 22222222222222222223333333333333333333333333333333
   99  32 | 4444444444444444444444445555555555555555555
   56  32 | 66666666666666677777777777
   30  32 | 888888888899
   18  33 | 0000111
   11  33 | 22223
    6  33 | 4445
    2  33 | 6
    1  33 | 8
```

(v) We arrive at a decision by the nonparametric approach as follows. There are

```
        +/treal≤29189
2
```

out of 1000 realisations of T being as small as t_0 or smaller leading to an estimate $\hat{\alpha} = 0.002$ of $\tilde{\alpha}$.

As an alternative we may use an upper confidence limit. The efficiency of several approaches to finding confidence regions can be discussed at this point. For instance, there are at least two $(1-\gamma)\times 100\%$ upper limits $UL(\tilde{\alpha})$ based on the normal approximation to the binomial distribution:

$$UL^{(1)}(\tilde{\alpha}) = \left(0, \alpha + z\sqrt{\frac{\hat{\alpha}(1-\alpha)}{M}}\right]$$

$$UL^{(2)}(\tilde{\alpha}) = \left(0, \frac{1}{M+z^2}\left\{s + \frac{1}{2} + \frac{z^2}{2} + z\sqrt{\frac{(s+\frac{1}{2})(M-s-\frac{1}{2})}{M} + \frac{z^2}{4}}\right\}\right],$$

where $s = M\hat{\alpha}$ and $z = z_{1-\gamma}$ is the $(1-\gamma)\times 100\%$-point of the standard normal distribution. For relevant literature covering this topic see [5] and the literature cited therein.

Interestingly, it turns out that the use of $UL^{(1)}(\tilde{\alpha})$ can be highly misleading since its nominal confidence level can be dramatically smaller than the actual one especially if $\hat{\alpha}$ is small, cf. [5]. It turns out $UL^{(2)}(\tilde{\alpha})$ does not share this shortcoming. Setting $1-\gamma=0.99$, $z=2.3263$ and $s=2$ we find the 99% upper bound of $\tilde{\alpha}$:

```
    UPPER_BOUND_OF_PROB 0.01 2 1000
0.0054
```

so that $UL^{(2)}(\tilde{\alpha})=(0, 0.0054]$.

Putting our findings together we arrive at the conclusion that H_0 is not tenable.

Even though this last example may seem strange we hold that it can serve as a vehicle to teach what statistical thinking is about. A whole bunch of old and new statistical ideas can be addressed to arrive at a reasonable solution. Nevertheless, it is not too streamlined and "normally distributed" as many synthetic examples in introductory texts. After all, it is this kind of data sets that the student has to face later on in the rough world outside.

5 A Theory of Experiment Construction

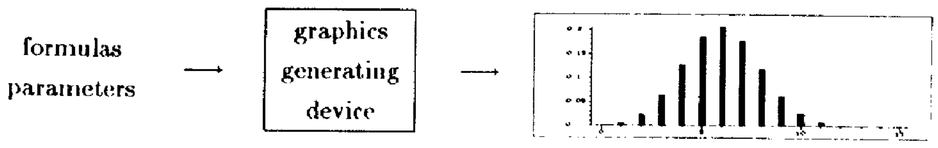
In this section we describe different levels of abstractions on the way to the entities we call "Statistical Experiments". The goal of the experiments is to demonstrate special ideas of the theory of statistics as well as statistical thinking. To justify the name "experiments" they must work under modified circumstances an experimenter is interested in. The experiments must not be black boxes. On the contrary one must be able to open the boxes to rearrange the ingredients.

5.1 A Problem

How can we get a feeling for the quality of the approximation of the binomial distribution by the Poisson distribution? An example will help to make our ideas clear. If a teacher is interested in clarifying the relationship between the binomial distribution and the Poisson distribution he usually proves the relationship by manipulating formulas:

$$P(X=x)=\binom{n}{x}p^x(1-p)^{n-x} \quad \xrightarrow[n\to\infty]{np=\lambda} \quad \frac{\lambda^x}{x!}e^{-\lambda} .$$

But presumably all this is not as good as an experiment where one can *see* the approximation work. To achieve this there must be a device to generate adequate graphical outputs.



This means that input and output of the device have to be well suited for the way a statistician thinks.

5.2 First Step: Translating Formulas Directly

Let us start with the formulas. Surely you must be able to handle such expressions like

$$P(X = x) = \binom{n}{x} p^x (1-p)^{n-x} \quad \text{and} \quad \frac{\lambda^x}{x!}e^{-\lambda} \quad .$$

The translation into a form understandable to a computer often is the first and the last argument to switch off the high-tech-instrument at hand. But there is *A Programming Language* that allows to type operations and operands of formulas very similarly to the mathematical notation.[2]

APL allows to start the designing process on a very high abstraction level compared to other computer languages. Pure *APL* enables to transform many mathematical notations into a computer readable form in a direct and easy way.[3] Here is a comparison of the probability functions above and their *APL*-counterparts.

Mathematics	*APL*[4]
$\binom{n}{x}p^x(1-p)^{n-x}$	`(x!n)×(p⋆x)×((1-p)⋆(n-x))`
$\lambda^x e^{-\lambda}/x!$	`(lambda⋆x)×(⋆-lambda)÷(!x)`

5.3 Second Step: Building Idioms for Frequently Used Terms

On the second level of abstraction one notices that certain patterns in formulas appear again and again. There are terms consisting of these patterns and one does not realize the atomic elements of the patterns any longer. As in natural languages these patterns are called "idioms". Examples related to statistics are $\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ or $\sum(x_i - \bar{x})^2 p_i$.

Such idioms may be identified with *APL*-idioms used by the programming statistician, but there are also a great number of *APL*-specific idioms decreasing the amount of time to solve programming problems. In building idioms and thinking in idioms one reaches the second level of abstraction.

[2]Iverson, [6], developed *APL* as a notational tool.

[3]Nobody is perfect. It may happen that the implemented functions run into numerical difficulties. For example it can be necessary to implement the !-function for yourself:

$$\binom{n}{x}p^x(1-p)^{n-x} = \exp\Big(\sum_{i=n-x+1}^{n} \ln i - \sum_{i=2}^{x} \ln i + x\ln p + (n-x)\ln(1-p)\Big).$$

APL: `⋆(+/⍟(n-x+1)↓⍳n)+(-+/⍟1↓⍳x)+(x×⍟p)+(n-x)×⍟(1-p)`

It is very nice to compute sums by "+/" and to handle vectors, for example `⍳n` ←→1 2 3 … n. Therefore, it is straightforward to generate the probability density function in one step: `(x!n)×(p⋆x)×(1-p)⋆n-x←0,⍳n`

[4]We know that some brackets are redundant.

5.4 Third Step: Enhancing the Language by Well Defined Functions

When idioms get longer and longer it becomes desirable to enhance the *APL*-language by adding new language elements. This means that user-defined functions have to be written. However the main problem is to decide how to design the new language elements. Questions such as

- What are the main basic tasks to be done?
- What are input informations that change from day to day?
- What is the structure of the desired output?
- What is an appropriate name for the operations done by the function?

have to be faced. The starting point for extending the language always is a concrete problem. To solve it in a satisfying way the structure of the whole space you are working in (workspace) has to be taken into consideration. The same job should not be done twice. Furthermore, some structural decisions for adding functions in the future have to be made. It is not a simple task for the statistician to work as a software engineer.

In the light of the approximation problem and related ones basic routines have to be defined to evaluate the probability density functions. The input should be parameters and x-values. The the output will be the corresponding probabilities.

What about the names?

`t0i1n2a3n4w5s`[5]

is a bad one. We want to have the same pattern in names of functions doing jobs of the same kind. So it is a good idea to use small letters for the names of "low level functions". Another design decision is to let the names of the functions computing a probability density function (cumulative distribution function) start with the letter "*d*" ("*c*"): *d_binomial c_binomial d_poisson c_poisson*. Although not being optimal in every respect we take the x-(or the p-)-values as left and the parameters as right argument of the functions in order to be compatible with the *APL*-function syntax:

Result ←left_argument function right_argument .

In designing functions in the way described above the notational abilities of the given language are extended. Moreover, an individual language is created. In this way an abstraction level is reached that is closer to the structure of the thoughts of the working statistician than working with the original *APL*-operators.

[5]This-is-not-a-name-with-sense.

5.5 Fourth Step: Combining Low Level Functions to Generate High Level Functions

To proceed with the approximation problem we need some functions to visualize the shape of distribution functions:

```
∇SHOW_BINOMIAL
```

Such a function has to activate graphical capabilities to draw the graph of the probability density function, to draw axis, a title, and so on. So we look for an easy way to produce output showing the main structure of the binomial distribution. What we have in mind is something like this:

```
        ∇SHOW_B n_p
    ⍝**** computation of the probability density function
    dbi←d_binomial n_p
    ⍝**** plot of the probability density function
    G_BARPLOT dbi
    ⍝**** this function shows the probability density function of the
    ⍝**** binomial distribution with n=n_p[1] and p=n_p[2]
```

SHOW_B works![6] It works because of a set of well defined graphical functions. *G_BARPLOT* is one of the high level routines producing graphical output. Other examples often needed by a statistician are:

G_HISTOGRAM *G_DENSITYTRACE* *G_BOX_AND_WHISKER* *G_XY_LINES* *G_Q_Q_PLOT* ...

These high level functions allow to draw sophisticated plots for special jobs. The high level functions themselves call functions of a lower abstraction level. On this level there exist functions like

G_points *G_singlelines* *G_vlines* *G_hlines* *G_marks* ...

Combining these ingredients very complicated pictures can be put together. This is not the right place to go into further details but two remarks are in order. First, there are functions in the set of the graphical functions to define the beginning and the ending of a picture as well as some to handle the organisation of different viewports, colours, modes, etc. Second, the *G_*-functions are based on Δ-functions written by Dr. Jürgen Steinecker which themselves involve the graphical abilities of the *APL*-implementation.

With the vocabulary of these routines in mind *SHOW_B* could be improved by defining *SHOW_BINOMIAL* to get pictures being a little bit more attractive.

[6]You have seen an output of *SHOW_B* some pages before.

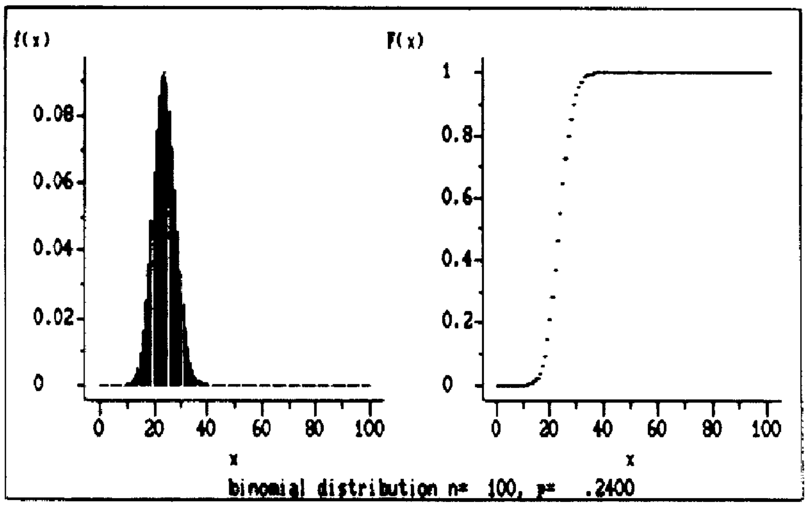


Figure 7: Output of ***SHOW_BINOMIAL***.

5.6 Fifth Step: Defining an Intermediate Level Function to the Posed Problem

To compare the shape of the binomial distribution with the shape of the the Poisson distribution we have to design a picture showing the essentials of both. Here are some ideas:

- Draw the probability density functions into two viewports:

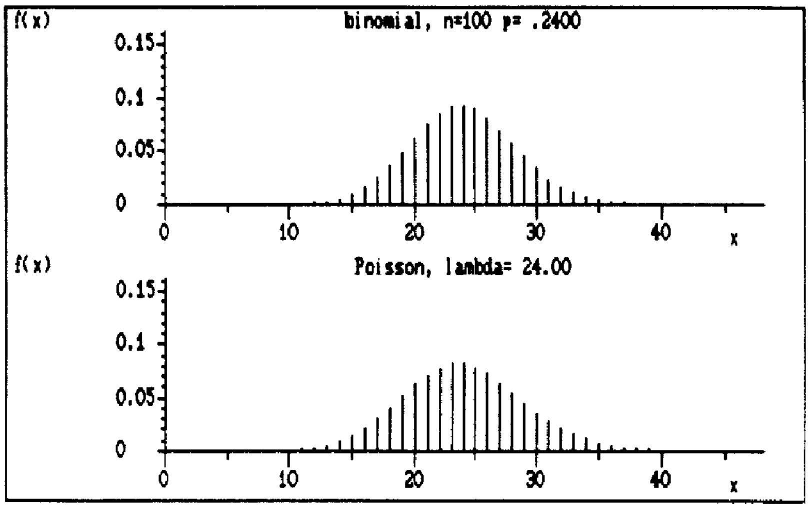


Figure 8: Comparing Poisson with binomial distribution (1).

- Draw the cumulative functions the same way.
- Draw the probability density function (or the cumulative distribution function) of the Poisson distribution into one viewport and the differences between the density functions (or the cumulative distribution functions) into the second one.
- The same as the one before but draw the differences not starting from the ***x_axis*** but from the points defined by the Poisson distribution.

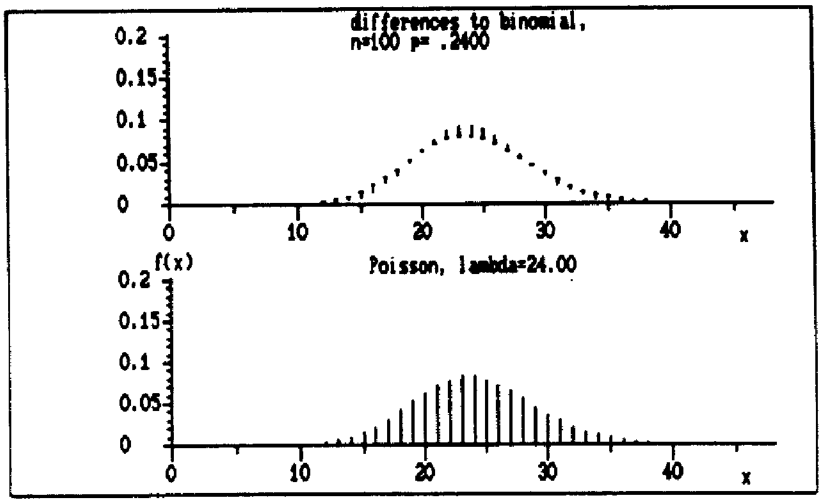


Figure 9: Comparing Poisson with binomial distribution (2).

- It is easy to create some more types by remembering that PP-plots and QQ-plots are efficient tools to compare distributions.

The internal structure of functions doing this job is straightforward:

```
∇COMPARE_BINOMIAL_AND_POISSON n_p
( Compute the values of binomial and Poisson distribution )
( Further computation like differences, etc. )
( Initialize graphic )
( Define viewport 1 and draw bars, lines, ... )
( Define viewport 2 and draw ... )
( Finish graphic )
```

This function can be viewed as a high level function but we classify it as an intermediate level function because of reserving the mark "high" for the remaining experiments. These are the ones that hopefully will make the teaching roar. Some remarks are in order.

- The user always has to remember the syntax of the function.
- The user has to make an explicit decision about the parameters.
- To visualize the process of approximation by increasing the parameter n the function has to be restarted again and again.
- The desire to make slight changes makes it necessary to edit the function.

With this in mind we are prepared to discuss the highest level described in this paper the level of statistical experiments.

5.7 Sixth Step: Reaching the Highest Level by Designing Experiments

An experiment should be a flexible instrument to demonstrate (statistical) phenomena in an open way. The experimenter should not be burdened with syntactical problems; nevertheless he should be able to make some essential modifications with very little effort. To achieve these aims we define the following structure of an experiment function:

```
        ∇EXP_<name_of_the_experiment>
    ⟨ Definiton part: here the environment of the experiment is defined ⟩
    ⟨ Working part: here the statements of the experiment's heart are gathered ⟩
```

The first part can be seen as defining the desk of the laboratory. A box (implemented as a function with the name *EXP_<name_of_the_experiment>_box*) is opened and all variables and other things that may be altered are declared. The working part of the experiment runs in this well defined environment. To create an experiment as open as possible the user must be allowed to have a look into the associated boxes and to change the ingredients he finds therein. The changes can either be accomplished using a programming editor or if you prefer an interactive mode by calling the function *DEF_BOX*.
Its syntax is: *DEF_BOX 'EXP_<name_of_the_experiment>'*

It is clear that the designer of experiments has to define the boxes too and therefore the whole range of changes. In this way much programming efforts can be avoided.

Now an experiment for the approximation problem will be constructed: We take the idea described above to define two viewports in which the probability density function of the Poisson distribution and the differences can be seen. The general structure of an experiment just mentioned is the beginning of a top down designing process:

```
        ∇EXP_APPROXIMATION_BINOMIAL_BY_POISSON
    ⟨ Define the environment ⟩
    ⟨ Do the experiment ⟩
```

First refinement of the definition part:

```
    ⟨ Define the environment ⟩ ≡
    ⟨ Some initializations ⟩
    start:
    sink←EXP_APPROXIMATION_BINOMIAL_BY_POISSON_box
```

... and on the other hand the first step for the working part:

```
    ⟨ Do the experiment ⟩ ≡
    ⟨ Compute some values ⟩
    ⟨ Initialize graphical facilities ⟩
    loop:  ⍝*** here starts the part that will be repeated some times ***
    ⟨ Compute new values for the next picture ⟩
    ⟨ Modify the first viewport ⟩
```

⟨ Modify the second viewport ⟩
⟨ Goto loop or stop ⟩
⟨ Finish graphical output ⟩
⟨ Clean up a little bit ⟩

We think a small impression is given by the short "top level descriptions". More interesting than discussing further details of the top down process is to have a look at the input values that an experimenter may wish to change.

1. First of all the location of the distributions has to be fixed. This is equivalent to choosing the parameter λ of the Poisson distribution.

   ```
   lambda ←24
   ```

2. It is important to define the region that will be displayed in the viewports.

   ```
   x_min ←0
   x_max ←2×lambda
   ```

   It follows that the mean of the Poisson distribution is located in the middle of the plot. Another idea would be to consider the variance in defining *x_max*, too.

3. To demonstrate the process of approximation it is necessary to define the values of the "running" parameter *n* of the binomial distribution. The bounds may be set by:

   ```
   n_min ←⌈lambda
   n_max ←100
   ```

   The amount of the increase of *n* can be defined by an increment factor, for example:

   ```
   n_incr_factor ←1.5
   ```

The resulting box function looks like this:

```
      ∇EXP_APPROXIMATION_BINOMIAL_BY_POISSON_box
   ⍝**** the mean of the Poisson distribution ****
   lambda ←24
   ⍝**** the minimal value of n of the binomial distribution ****
   n_min ←⌈lambda
   ⍝**** the maximal value of n of the binomial distribution ****
   n_max ←100
   ⍝**** the incrementation factor for the parameter n ****
   n_incr_factor ←1.5
   ⍝**** the minimum x-value for the window ****
   x_min ←0
   ⍝**** the maximum x-value for the window ****
   x_max ←2×lambda
```

A little demonstration: To start the experiment we just type

EXP_APPROXIMATION_BINOMIAL_BY_POISSON

and a sequence of displays will be generated. Some of these are copied here:

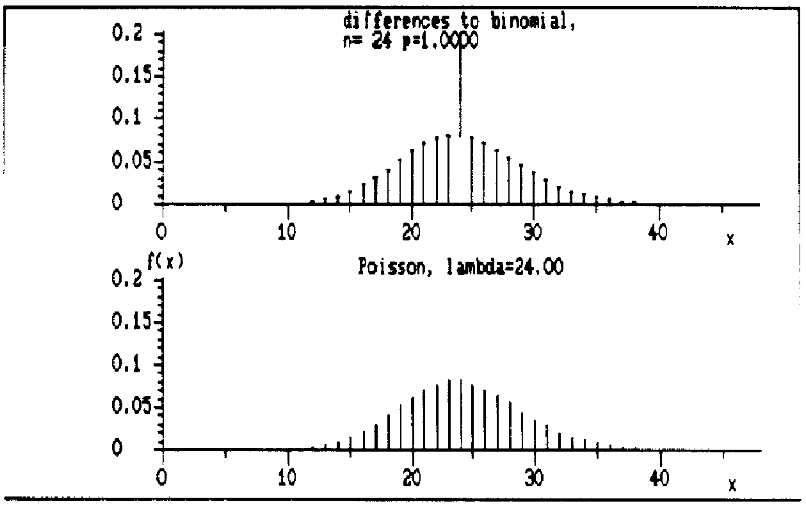


Figure 10: First display of the experiment.

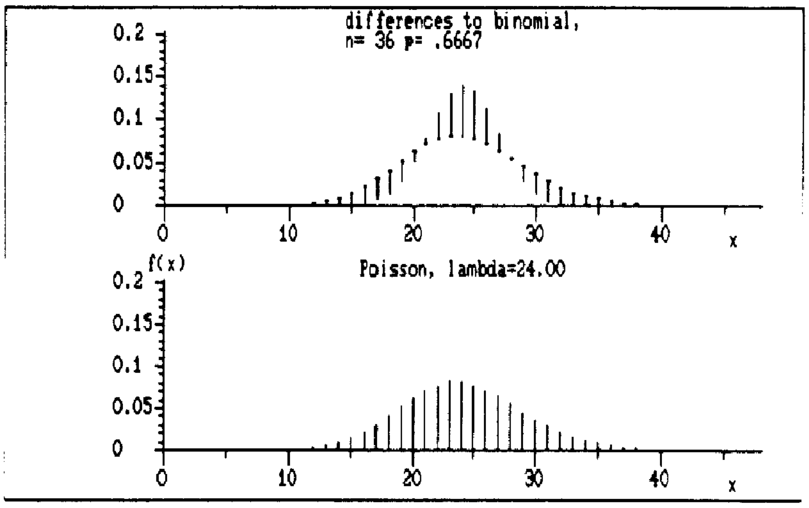


Figure 11: Second display of the experiment.

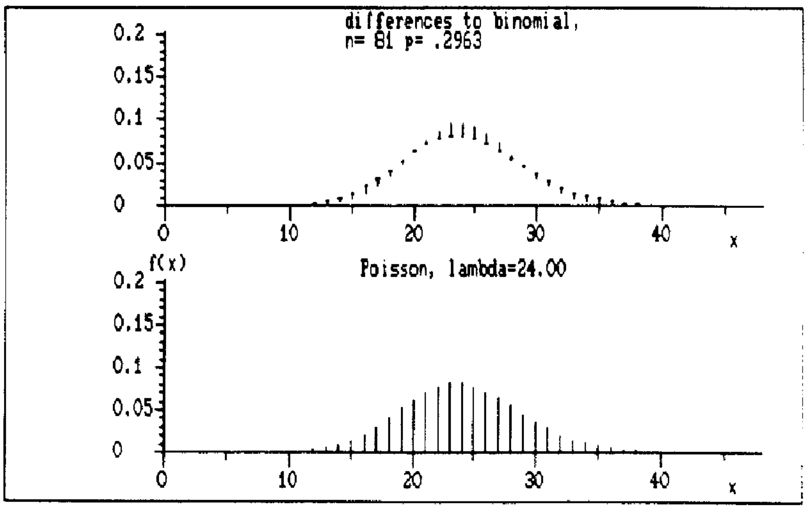


Figure 12: The last display of the experiment.

To observe the approximation process for small values of λ the laboratory desk can be modified by defining:

```
'x_max←100⋄lambda←20⋄n_min←5' DEF_BOX 'EXP_APPROXIMATION_BINOMIAL_BY_POISSON'
```

Using the abstraction level of the experiments properly enables to design attractive instruments for the classroom as well as for the teacher. Attractive means the ease of changing a given experiment by simple modification rules. The main keywords "run and look" expand to "define, run and look".

5.8 Comment: User Defined Experiments by Combining Modules of Other Experiments

The real power of the experiments has not been described completely as yet. Since the experiment functions are open to the user he can modify them in a deeper sense: By extracting modules of existing experiments to combine them to get new ones.

To explain this idea remember the relationship between the exponential distribution and the Poisson distribution. Is there an easy way to illustrate this relationship without much further ado? The answer is yes. We can exploit our former experiment illustrating how to generate exponentially distributed random numbers. All we have to do is to copy those parts of the experiment concerning the upper two viewports. Then we define a third viewport to cumulate realizations of the exponential distributions and count the phenomena in a fixed interval.

In the remaining fourth viewport the counts can be saved and compared to the theoretic results. Here is the last output of the new experiment:

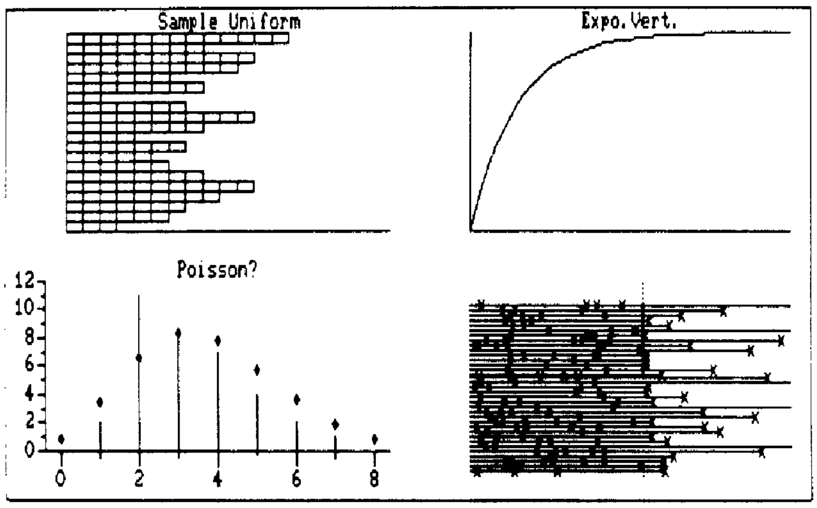


Figure 13: A mutation of the first experiment.

6 How it was done.

Here we will show how the type 1 experiment concerning the random number generation was constructed. The function ***DRAW_EXPO*** realizes the described experiment. The right argument ***N*** stands for the size of the random sample.

```
∇ DRAW_EXPO N
⟨initialise global ⟩
⟨initialise N-W-corner ⟩
⟨initialise S-W-corner ⟩
⟨initialise N-E-corner ⟩
⟨initialise S-E-corner ⟩
⟨loop over n ⟩
⟨end global ⟩
```

The global initialisation sets the frame for this special experiment. The value of the parameter λ is unimportant because the graph is spread over the whole width of the window. Therefore λ is set to 1. Futhermore suitable graphical parameters for the parameters $L = \lambda = 1.0$ and ***N*** will be chosen. As for the interpretation of the graphical parameters one should refer to our paper [11].

```
⟨initialise global ⟩ ≡
⍝ Parameters for experiment
L←1 ⋄ eps←0.05 ⋄ I←1
⍝ Parameter for graphics
XOh←eps×XO←0.999 i_exponential L ⋄ Nph←(⌈N +3)+XOh×N ⋄ Nuh←⌈N÷8
NXOh←+N×XOh
uh←20⍴0 ⋄ uk←20 eqcl 0,1
ph←20⍴0 ⋄ pk←20 eqcl 0,XO
⍝ Start graphic
G_begin
G_setcolor'white'
```

The function `i_exponential` computes values of the inverse distribution function F^{-1} of the exponential distribution while `k eqcl xu,xo` splits the interval `(xu,xo)` into `k` equidistant classes. We are not going into details with respect to the graphical routines and refer to the paper mentioned before.

Some local variables have occured and must be declared.

```
⟨Local Variables of DRAW_EXPO ⟩ ≡
sink,eps,I,XO,Nph,XOh,Nuh,NXOh,uh,uh,ph,pk
```

The initialisation of the parts of the graphic always is done in the same way:

- Selection of the viewport
- Decription of the window

- Creation of an unmarked coordinate system
- Drawing of the permanent graphs
- Labeling

Let's start in the upper left quarter: the N-W-corner.

```
⟨initialise N-W-corner    ⟩ ≡
2 2 G_setviewport 1 1
'nxfnyfnxsnys'G_setwindow 0,Nuh,0,1
G_hlines 0 ◊ G_vlines 0
G_label'Sample Uniform'
```

Now we are turning to the bottom left quarter: the S-W-corner. We will have to compute a couple of pairs $(x, \lambda e^{-\lambda x})$ in order to draw the density of the exponential function. The x-values lying in the interval (0,*X0*) will be used to compute the distribution function $1 - e^{-\lambda x}$. That is why they are stored in *X*.

```
⟨initialise S-W-corner    ⟩ ≡
2 2 G_setviewport 2 1
'nxfnyfnxsnys'G_setwindow 0,X0,0,Nph
G_hlines 0 ◊ G_vlines 0
G_lines X,[1.5](X←50 eqcl 0,X0)d_exponential L
G_label'Expo.Density'
```

Le‘’s continue with the upper right quarter: the N-E-corner. ***c_exponential*** computes the distribution function F of an exponential, ***d_exponential*** which was called in the last section computes the density f. We will need the distribution function from time to time to refresh the display. The computed pairs of points are stored into *XY*.

```
⟨initialise N-E-corner    ⟩ ≡
2 2 G_setviewport 1 2
'nxfnyfnxsnys'G_setwindow 0,X0,0,1
G_hlines 0 ◊ G_vlines 0
G_lines XY←X,[1.5]X c_exponential L
G_label'Expo.Distr.'
```

We should add some variables to the list of local variables.

```
⟨Local Variables of DRAW_EXPO    ⟩ + ≡
X,XY
```

There is only one part left the — you got it right — S-E-corner.

```
⟨initialise S-E-corner    ⟩ ≡
2 2 G_setviewport 2 2
'nyfnxfnxsnys'G_setwindow 0,X0,0,Nph
```

```
G_hlines 0 ◊ G_vlines 0
G_label'Sample Expo.'
```

Before we are turning to the stepwise execution of the experiment, let's talk about the end of the demonstration. As announced, the graph of the density is to be drawn in the S-E-corner. Furthermore the whole picture must be closed.

```
⟨end global   ⟩ ≡
2 2 G_setviewport 2 2
'nxfnyfnxsnys'G_setwindow 0,X0,0,Nph
G_lines X,[1.5]X d_exponential L
⍝ Stop graphic
G_end
```

The core of the experiment is the run through the following loop:

N-W draw a random value *P* from a (0,1)-uniform distribution. Raise the corresponding pillar.

N-E compute *PP* as the inverted distribution function at *P*. Draw the line from (0,*P*) to (*PP*,*P*) and from (*PP*,*P*) to (*PP*,0).

S-E raise the pillar of the histogram class containing *PP*.

refresh check, if refreshing is necessary. There is no way to avoid the destruction of the graph by the drawing and deleting in the second step. The solution to the problem is the refreshing after every fifth step.

Each window is processed in the same way.

1. Select the viewport.
2. Fix the window.
3. Compute and draw the new values — the graphic is built step by step.

The control of the loop is selfexplaining. (At least we hope so.)

```
 ⟨loop over n   ⟩ ≡
LZZ:2 2 G_setviewport 1 1
 'nxfnyfnxsnys'G_setwindow 0,Nuh,0,1
 G_setcolor'white'
 uh[i]←uh[i←+/uk<P←UNIFORM 1]+1
 0 0.05 1 1 G_bars 1 2ρ⌽(0.5×+/uk[i,i+1]),uh[i]
 2 2 G_setviewport 1 2
 'nxfnyfnxsnys'G_setwindow 0,X0,0,1
 XX←2 4ρ(eps+0),P,(PP-eps),P,PP,(P-eps),(PP←P i_exponential L),0+eps
```

```
  G_singlelines XX
  G_setcolor'black'
  G_singlelines XX
  G_setcolor'white'
  2 2 G_setviewport 2 2
  'nxfnyfnxsnys'G_setwindow 0,XO,0,Nph
  ph[i]←ph[i←20⌊+/pk<PP]+NXOh
  (0,XOh,1)G_bars 1 2ρ(0.5×+/pk[i,i+1]),ph[i]
  →LO×⍳0≠5|I
  2 2 G_setviewport 1 2
  'nxfnyfnxsnys'G_setwindow 0,XO,0,1
  G_hlines 0 ◊ G_vlines 0
  G_lines XY
LO:→LZZ×⍳N≥I←I+1
```

The Function *DRAW_EXPO* is almost complete. We only have to declare some more variables to be local.

```
⟨Local Variables of DRAW_EXPO ⟩ + ≡
P,PP,XX
```

7 Writing a Paper Not a Program

This section is meant as a first proof of the before mentioned statement: you do not have to be a computer scientist to use your computer effectively when teaching statistics. What we will demonstrate here might be put under the heading *APL* and literate programming. For those who are not familiar with *D.E. Knuth's* idea of literate programming we would mention his paper [7]: *Literate programming* in *The Computer Journal 1984*. But we believe the main ideas and the flavour of this approach will become clear in the sequel. Instead of lecturing about literate programming let us practice literate programming. To this end let us discuss a small statistical problem. Talking about the problem and its solution will be done in a mixture of natural language, mathematical language and *APL*.

1. α-trimmed mean. As we all know the mean $\bar{x}$ is quite sensitive regarding "outliers". Observations x_ν whose absolute value differ widely from all other observations tend to disturb the interpretation of $\bar{x}$ as a measure of location for the bulk of observations.

If one wants to stick to this — in one way reasonable — interpretation one has to get rid of these outliers before $\bar{x}$ is calculated.

The so called ***α-trimmed mean*** is one possibility to do this. The idea is quite simple. Remove the α percent smallest and the α percent largest observations from the data set. Then the mean is calculated as usual.

An *APL*-function which achieves this shall be developed. The value of α is the left argument, right argument is the vector of observations. As an explicit result the calculated ***α-trimmed mean*** is returned.

```
∇ r←alpha trimmean x  1
```

See also section 2.

2. The body of the function can be outlined quite easily.

1. Sort x in ascending or descending order.
2. Drop α percent of the observations on either end.
3. Calculate the mean of the remaining part.

This boils done to the following program steps.

```
∇+ r←alpha trimmean x   1
⟨sort x   3⟩
⟨trim x   4⟩
⟨mean x   5⟩
```

3. Thanks to the powerful *APL*-language (e.g. ⍋) the coding is almost no problem. At first let us do the sorting

```
⟨sort x   3⟩ ≡
x←x[⍋x]
```

This code is used in section 2.

4. now the trimming step

```
⟨trim x   4⟩ ≡
x←n↓(-n←⌊alpha×⍴x)↓x
```

This code is used in section 2.

5. and last but not least the calculation of the mean.

```
⟨mean x   5⟩ ≡
r←(+/x)÷⍴x
```

This code is used in section 2.

6. We did not pay any attention to the special structure of x, we just assumed that x is a data vector. But one thing we ought to do is to clean up our environment. So let us make as many variables local as possible.

```
⟨Local Variables of trimmean(1)   6⟩ ≡
n
```

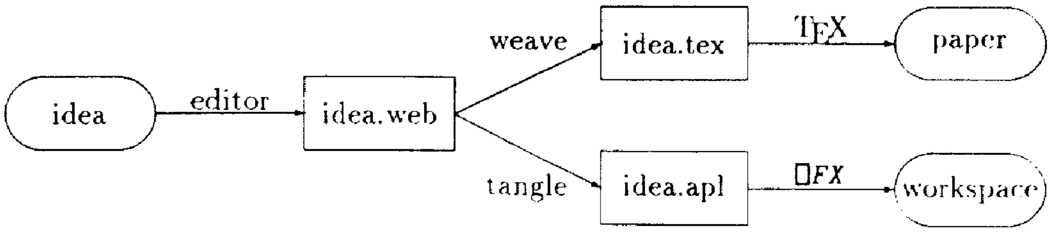
We hope everybody will agree that this small paper inside our paper is a clear discussion of a statistical method and its implementation. (We would confess that there still is room for criticism based on personal taste etc.) So it should be no problem to implement the outlined function. But if it is such an easy task — everything what can be said and should be said has been said — why not let a friendly ghost do the work.

Wouldn't it be nice to find somewhere a file containing the described function. To transfer this code into an *APL*-workspace then should be no problem — just think of ⎕*FX*. Here is the magic file.

```
r←alpha trimmean x;n
⍝ Initially declared in WEB-file <trimmean.web>, line 22
⍝ Main WEB-file <work.web>
⍝∇ trimmean: 1
⍝ 6: n
⍝ 2:, 3:
x←x[⍋x]
⍝ :3, 4:
x←n↓(-n←alpha×ρx)↓x
⍝ :4, 5:
r←(+/x)+ρx
⍝ :5, :2
TO_expand
⍝ Main WEB-file: work.web
```

How was this achieved? The structure of the system we use — let us call it *APLWEB*-system — is made explicit in the following picture.

The *APLWEB*-system



The starting point is the WEB-file incorporating our ideas. This is the file we create — using an editor which can handle more than one code table. The ideas are expressed in natural language, mathematical language, graphical language, etc. whatever is appropriate. We may include as many commands to a formatter — our choice is TEX — as we wish. The *APLWEB*-system can be controlled by just a few commands. Those used in the small example are:

@* starts a main section

@␣ starts a section

@< starts a top level description, must be closed by @>.

∇ starts function definition mode.

There are some more. But as it is not our main theme to introduce the *APLWEB*-system for *APL* we will not go further into the details. Hopefully it was sufficient to give an impression of this terrific tool. If the reader cannot understand how he managed to live without WEB up to now, we were successful. We would like to praise our colleague Christoph v. Basum [1] for this byproduct of his Ph.D.-thesis.

References

[1] Basum v., Ch. (1990): Making *APL* readable, Technical Report, University of Bielefeld.

[2] Bickel, P.J. / Doksum, K.A. (1977): *Mathematical Statistics: Basic Ideas and Selected Topics*, Holden Day, San Francisco.

[3] Courant, R. / Robbins, H. (1967): *What is Mathematics*, Oxford Univ. Press, Oxford.

[4] Daniels, H.E. (1975): *Statistics in Universities — a Personal View*, JRSS, Ser. A, Vol. 138.

[5] Hanfeng Chen (1990): *The Accuracy of Approximate Intervals for a Binomial Parameter*, JASA 85, pp. 514–518.

[6] Iverson, K. E. (1980): *Notation as a Tool of Thought*, Comm. of the ACM, Vol. 23, pp. 444–465.

[7] Knuth, D. E. (1984): *Literate Programming*, The Computer Journal, Vol. 27, pp. 97–111.

[8] Mood, A.M. / Graybill, A.F. (1963): *Introduction to the Theory of Statistics*, 2nd ed., McGraw-Hill, Tokyo.

[9] Mood, A.M. / Graybill, A.F./Boes, D.C. (1974): *Introduction to the Theory of Statistics*, 3rd ed., McGraw-Hill, Tokyo.

[10] Naeve, P. (1978): *CAI and Computational Statistics*, in: COMPSTAT LECTURES 1, Physica, Würzburg.

[11] Naeve, P. / Trenkler, D. / Wolf, P. (1989): *Bemerkungen zum Konzept der Graphikroutinen*, Technical Report, University of Bielefeld.

[12] test — Zeitschrift der Stiftung Warentest, (1989): *Billigräder fielen durch*, Vol. 24, No. 4, pp. 72–79.

[13] Thompson, J.R. (1989) *Empirical Model Building*, John Wiley, New York.