Mathematical Statistics

SF1901 Probability Theory and Statistics: Autumn 2016
Lab 0 for
TCOMK

1 Preparation

This computer exercise is a bit different from the other two, and has some overlap with computer exercise 1 (Lab 1). Unlike the other two computer exercises it does not render you bonus points. Its purpose is to help you get acquainted with MATLAB. Try to focus on learning how to use MATLAB, and make sure that you understand the commands you use.
Preparation: Read this instruction before you go to the computer exercise and review the exponential distribution.

The theme for this computer exercise is *simulation*. In the part of the course which treats probability theory you will learn how to compute different quantities, such as probabilities, expected values etc., given a certain random distribution. For more complicated random models it can be very time-consuming, or even impossible to do exact computations.

In such circumstances simulation can be an alternative. You then use a computer to simulate outcomes from a certain distribution, and then you can use for instance the mean of the outcomes to estimate the expected value, or some empirical quantile to estimate a probability. In this computer exercise we will do this for some rather simple problems (where explicit computations are possible), but the same basic principles can be applied to more difficult problems where explicit computations are not possible.

Let us say that we want to estimate the expected value of a certain distribution. Suppose that the distribution function is F, and let X be random variable with this distribution function. Also let us denote the expected value we want to estimate by μ.

If we generate observations $x_1, x_2, \ldots, x_n$ from F, we can estimate μ using

$$\hat{\mu} = \frac{1}{n} \sum_{k=1}^{n} x_k. \tag{1}$$

That this is a reasonable estimate of μ follows from the law of large numbers.

If we instead want to estimate $F(a) = \mathbb{P}(X \leq a)$ for some number a, then this can be done by calculating the proportion of the simulated observations that satisfies $\leq a$. We can write this as

$$\hat{F}(a) = \frac{\text{number of } x_k \text{ which are} \leq a}{n} = \frac{1}{n} \sum \mathbb{I}(x_k \leq a); \tag{2}$$

Here $\mathbb{I}$ denotes a so called *indicator function*, which takes the value 1 if the condition within parentheses is fulfilled, and 0 otherwise. This means that $\mathbb{I}(x_k \leq a)$ is equal to 1 exactly for those k such that $x_k \leq a$, which in turn means that the sum above counts the number of indices k which fulfill the condition.

2 Introduction to MATLAB

In all the computer exercises in this course we will use *MATLAB*, which is an interactive program for numerical computations. It is also a programming language. You will find MATLAB installed on most of the computers at KTH, and one of the advantages with MATLAB is that it will look basically the same regardless of the type of computer, on which you run it. If you would like to read more about MATLAB there are several guides to download or buy.

Start by logging on to your usual account. Then start MATLAB by clicking on the icon. The program is terminated by typing the command `exit`. To begin with you can think of MATLAB as an advanced calculator which evaluates expressions. You type in what you want to be done and MATLAB answers.

```
>> 3*11.5 + 2.3^2/4
ans =
      35.8225
```

Variables are assigned values using the sign `=`, and these values will be stored in the memory.

Try to assign values to some variables.

```
>> a = 1;
>> b = sqrt(36);
>> width = 3.89;
>> who

Your variables are:

a          b          width
```

As you can see the command `who` shows a list of variables that have been stored in the memory. A variable, for instance `b`, can be erased from memory using `clear b`. If you put a semi-colon , `;`, at the end of a row, the result of the commands on that row will not be written on the screen. MATLAB can also handle vectors and matrices, just as easily as scalars.

```
>> x = [1 3 7]

x =

     1 3 7

>> y = [2 1 8]'

y =

      2
      1
      8

>> z = [1 2 ; 3 4]'

z =

      1   3
      2   4

>> w = rand(1, 4)

w =

     [0.2190    0.0470    0.678    0.6793]
```

The sign ' translates to transpose, as you can see from above, and the semi-colon is used to separate rows in matrices. The function `rand(m, n)` will generate an $m \times n$-matrix of random numbers uniformly distributed between 0 and 1. The notation in algebraic expressions is the usual one, but you should keep in mind that the sign for multiplication `*` is interpreted as matrix multiplication. Element-wise multiplication of two matrices `A` and `B` of the same dimensions is written `A.*B`. MATLAB has most common functions built such as

```
exp log sin asin cos acos tan atan
```

Please observe that `log` denotes the natural logarithm. Try to plot a function, for instance by executing the following commands.

```
>> x = 0.5:0.1:2
>> help log
>> y = log(x)
>> plot(x,y)
```

In the first row `x` is assigned a vector running from 0.5 to 2 in steps of 0.1. The help function `help` will give you information about a function if you type `help` and the name of the function. If you only type `help` a list of all available functions will be shown sorted by *toolbox*).

MATLAB contains a great number of functions related to probability theory and statistics. See `help stats`. Look in particular at what you find under the heading *Random Number Generators*, which you will find at the beginning of the long list of functions.

3 The expectation of the exponential distribution

The function in MATLAB used to simulate exponentially distributed random variables is called `exprnd`. Please use the help command `help` to find out exactly how the function `exprnd` works, i.e. type `help exprnd`! The function can also be used to simulate vectors (or matrices) of independent exponentially distributed random variables. For example the code

```
>> n = 1000;
>> x = exprnd(2.5, n, 1);
```

will generate an $n \times 1$-vector of values coming form an exponential distribution with expectation 2.5. Suppose now that we do not know that the expectation of this distribution is exactly 2.5, and that we would like to estimate it using the simulated data. This can be done by computing the mean.

```
>> mean(x)
```

How good is your estimate? Try to redo the simulation and the computation of the mean a few times. Also try using different values for n!

4 Tail probabilities for the exponential distribution

We will now use simulation to estimate the tail probability

$$\mathbb{P}(X > x) = 1 - F_X(x) \tag{3}$$

for an exponential distribution. The syntax `(x>5)` in MATLAB will return a vector of the same size as `x`, but for which the elements are either 1 or 0 depending on whether the corresponding element in `x` satisfies the condition `> 5` or not.

One way to estimate $1-\hat{F}(5)$ is therefore to simulate a vector `x` containing data as above, and then to type

```
>> mean(x>5)
```

Which value do you get? What is the true tail probability? Try different `n`, and different tail probabilities (i.e. use some other value than 5)!