Series 2

1. The dataset `prostcancer` contains medical data measured on 97 men who were about to receive a radical prostatectomy. The aim is to find out how the predictors influence the response **psa**, which is an important indicator for further development of the cancer. The variables are

psa	level of prostate specific antigen (response)
cavol	cancer volume
weight	prostate weight
age	age of patient
bph	benign prostatic hyperplasia amount
svi	seminal vesicle invasion
cp	capsular penetration
gleason	gleason score
pgg45	percent of Gleason score 4 or 5

Try to find a good linear model that represents the data. Therefore, take the following steps:

1. Produce diagnostic plots such as scatterplots of the response **psa** versus the predictors (hint: use `pairs(prostate, pch = ".")` and residual plots. Are the model assumptions valid? Which variables need to be log-transformed?
2. Perform a stepwise variable selection. Do forward selection and backward elimination lead to the same model? Which variables have a significant influence on the **psa**-level and in which direction? Compare these results to an all-subsets regression using Mallows-C_p.

R-hints:

```
> ## Reading the dataset
> prostate <- read.table("http://stat.ethz.ch/Teaching/Datasets/prostcancer.dat",
                         header = TRUE)
> pairs(prostate, pch = ".")

> ## Fit the full model
> prost.full  <- lm(psa ~ cavol+weight+age+bph+svi+cp+gleason+pgg45,
                    data = prostate)

> ## Fit the empty model. This is not very useful in itself, but is required
> ## as a starting model for stepwise forward variable selection
> prost.empty <- lm(psa ~ 1, data = prostate)

> ## Backward elimination, starting from the full model
> prost.bw    <- step(prost.full, direction = "backward")
> ## Forward selection, starting from the empty model
> prost.fw    <- step(prost.empty, direction = "forward",
                      scope = psa ~ cavol+weight+age+bph+svi+cp+gleason+pgg45)

> ## Loading the package for all-subsets regression
> library(leaps)

## All subsets model choice, compare to the stepwise methods
prost.alls  <- regsubsets(...)

## Load function to produce a nice figure of C_p versus p
source("ftp://stat.ethz.ch/Teaching/maechler/CompStat/cp-plot.R")
p.regsubsets(prost.alls)
```

2. **a)** The artificial dataset shown in Figure 1 is generated from a mixture of two normal distributions (see part b)). Imagine you wouldn't know anything about the distribution and you'd want to estimate the density by use of a kernel density estimator with Gaussian kernel. Guess a good bandwidth.

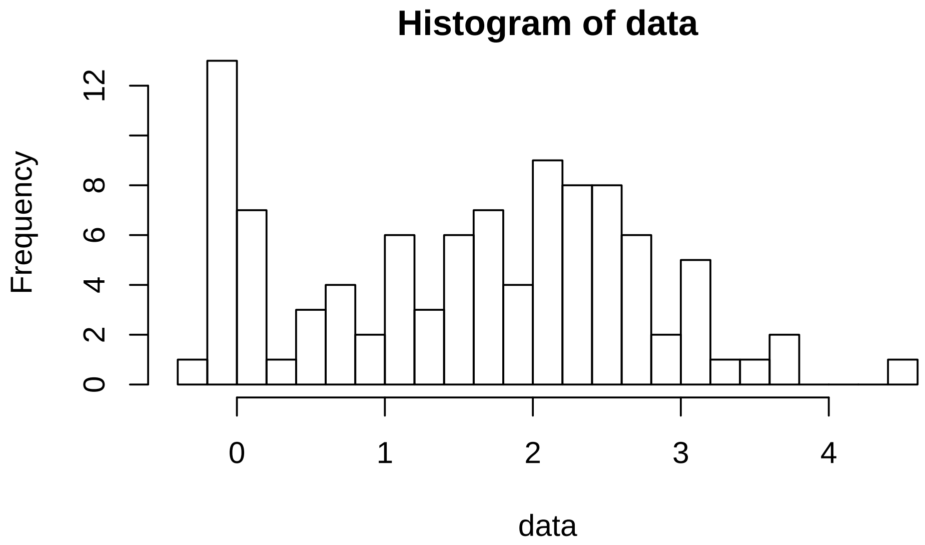

Figure 1: Artificial data from mixture of normal distributions

b) Carry out a simulation with `R` to evaluate the quality of bandwidths for kernel density estimation for a mixture of normal distributions where a point is generated by a $\mathcal{N}(0, 0.01)$-distribution with probability 0.2 and by a $\mathcal{N}(2, 1)$-distribution with probability 0.8. Repeat the following code 200 times for multiple bandwidth values: $h = 0.02,\ 0.1,\ 0.3,\ 0.6,\ 1,\ 1.5$, as well as h equal to your own guess from part a) and to a plug-in estimate of the optimal bandwidth (see below):

1. Generate a dataset of 100 points from the mixture distribution from above:

```
> set.seed(1)
> data <- numeric(100)
> for(i in 1:100){
   p <- runif(1, min = 0, max = 1)
   if (p < 0.2)
     data[i] <- rnorm(1, mean = 0, sd = sqrt(0.01))
   else
     data[i] <- rnorm(1, mean = 2, sd = 1)
 }
```

2. Compute the kernel density estimator with the function `density`. Consult `help(density)` to understand the syntax.

```
> ke <- density(data, bw = 0.1, n = 61, from = -1, to = 5)
```

An estimated plug-in bandwidth is obtained by using `bw ="sj"`.

3. In the lecture, we saw that the goodness of a kernel estimate can be measured using the integrated mean squared error (IMSE). In practice, the IMSE can be approximated by averaging the squared difference between the kernel estimator and the true density values at the multiple datapoints $-0.2,\ 0,\ 0.2,\ 0.5,\ 1,\ 1.5,\ 2,\ 2.5,\ 3,\ 4$:

```
> index <- c(9, 11, 13, 16, 21, 26, 31, 36, 41, 51)
> ke$x[index]  ## -0.2  0.0  0.2  0.5  1.0  1.5  2.0  2.5  3.0  4.0
```

```
> ## Compute the true density at the given datapoints
> dmix <- 0.2 * dnorm(ke$x[index], mean = 0, sd = sqrt(0.01)) +
     0.8 * dnorm(ke$x[index], mean = 2, sd = 1)
```

```
> ## Take the mean of the squared differences
> quality <- mean((ke$y[index] - dmix)^2)
```

Note that the commands given above are suitable for a single execution, but not necessarily for the full simulation. For example, you will need a vector of the qualities of all simulation runs. You may define a matrix for the qualities to store the results for all different bandwidths. Consider `help(matrix), help(apply)`.

Compute and compare the averaged quality of the kernel estimators for the different bandwidths.

c) (Optional) Repeat the whole project with the Epanechnikov kernel, which can be obtained by specifying `kernel="epanechnikov"` in `density`.

Preliminary discussion: Friday, March 09.

Deadline: Friday, March 16.