LAB 01

TOPICS:

- Basic commands (scalars, vectors, matrices, and operations)
- Import/Export of dataframes
- Examples of univariate statistical analyses with plots
- Visualization of Multivariate Data
- Visualization of Categorical Data
- 3d plots, functions, "for" cycles

```
### ---------------------------------------------------------------------
### ---------------------------------------------------------------------
### Basic commands
### ---------------------------------------------------------------------
### ---------------------------------------------------------------------

## Vector/Matrices & Linear Algebra in R
# scalars
a <- 1
b <- 3
4 -> c
a <- b

# vectors
v <- c(2,3,5,4)
v
```

```
## [1] 2 3 5 4
```

```
u <- seq(2,5,len=4)
u
```

```
## [1] 2 3 4 5
```

```
u <- seq(2,5,by=1)
u
```

```
## [1] 2 3 4 5
```

```
z <- 2:5
z
```

```
## [1] 2 3 4 5
```

```
# matrices
W <- rbind(c(11,13,15),c(12,14,16))
W
```

```
##      [,1] [,2] [,3]
## [1,]   11   13   15
## [2,]   12   14   16
```

```
W <- cbind(c(11,12),c(13,14),c(15,16))
W
```

```
##      [,1] [,2] [,3]
## [1,]   11   13   15
## [2,]   12   14   16
```

```
W <- matrix(data = c(11,12,13,14,15,16), nrow = 2, ncol = 3, byrow = F)
W
```

```
##      [,1] [,2] [,3]
## [1,]   11   13   15
## [2,]   12   14   16
```

```
## Extraction of elements from a vector or a matrix
v
```

```
## [1] 2 3 5 4
```

```
v[2]
```

```
## [1] 3
```

```
v[c(2,3)]
```

```
## [1] 3 5
```

```
v[-3]
```

```
## [1] 2 3 4
```

```
v[-c(1,4)]
```

```
## [1] 3 5
```

```
W
```

```
##      [,1] [,2] [,3]
## [1,]   11   13   15
## [2,]   12   14   16
```

```
W[2,3]
```

```
## [1] 16
```

```
W[2,c(2,3)]
```

```
## [1] 14 16
```

```
W[2,]
```

```
## [1] 12 14 16
```

```
W[,c(2,3)]
```

```
##      [,1] [,2]
## [1,]   13   15
## [2,]   14   16
```

```
# Remark: in R vectors "are not" matrices n*1 o 1*n:
# vectors have only one index, whereas matrices have
# two indices (for rows and columns)

v
```

```
## [1] 2 3 5 4
```

```
rbind(v)
```

```
##   [,1] [,2] [,3] [,4]
## v    2    3    5    4
```

```
cbind(v)
```

```
##      v
## [1,] 2
## [2,] 3
## [3,] 5
## [4,] 4
```

```
# Algebraic operations in R
# Remark: by default, operations are done component-wise

a <- 1
b <- 2
c <- c(2,3,4)
d <- c(10,10,10)
Z <- matrix(c(1,10,3,10,5,10), nrow = 2, ncol = 3, byrow = F)

# Sum and multiplication (component-wise).
# (this default is different from that of matlab!)
a+b # scalar + scalar
```

```
## [1] 3
```

```
c+d # vector + vector
```

```
## [1] 12 13 14
```

```
a*b # scalar * scalar
```

```
## [1] 2
```

```
c*d # vector * vector (component-wise)
```

```
## [1] 20 30 40
```

```
c+a # vettor + scalar
```

```
## [1] 3 4 5
```

```
c^2 # attention: operations are always component-wise!
```

```
## [1]  4  9 16
```

```
exp(c)
```

```
## [1]  7.389056 20.085537 54.598150
```

```
sum(c) # sums the components of c
```

```
## [1] 9
```

```
prod(c) # returns the product of the components of c
```

```
## [1] 24
```

```
# Operations on matrices
V <- t(W) # transpose of a matrix
Z+W # matrix + matrix (component-wise)
```

```
##      [,1] [,2] [,3]
## [1,]   12   16   20
## [2,]   22   24   26
```

```
Z*W # matrix * matrix (component-wise)
```

```
##      [,1] [,2] [,3]
## [1,]   11   39   75
## [2,]  120  140  160
```

```
#(!) V*W # matrix * matrix (component-wise) (error!)
V %*% W # Matrix multiplication
```

```
##      [,1] [,2] [,3]
## [1,]  265  311  357
## [2,]  311  365  419
## [3,]  357  419  481
```

```
W %*% V
```

```
##      [,1] [,2]
## [1,]  515  554
## [2,]  554  596
```

```
W+a # matrix + scalar
```

```
##      [,1] [,2] [,3]
## [1,]   12   14   16
## [2,]   13   15   17
```

```
W+c # matrix + vector
```

```
##      [,1] [,2] [,3]
## [1,]   13   17   18
## [2,]   15   16   20
```

```
W+2:5
```

```
## Warning in W + 2:5: longer object length is not a multiple of shorter object
## length
```

```
##      [,1] [,2] [,3]
## [1,]   13   17   17
## [2,]   15   19   19
```

```
# Remark: R uses the "recycling", i.e., it tries to make the
# terms dimensions compatible by recicling data if missing

# Inverse of a matrix (square and invertible)
A <- matrix(c(11,13,12,14), ncol=2, nrow=2, byrow=TRUE)
det(A)
```

```
## [1] -2
```

```
solve(A)
```

```
##      [,1] [,2]
## [1,]   -7  6.5
## [2,]    6 -5.5
```

```
# Solution of a linear system Ax=b
b <- c(1,1)
solve(A,b)
```

```
## [1] -0.5  0.5
```

```
## Categorical data
# The command 'factor' converts the argument (vector of numbers or strings)
# in a vector of realizations of a categorical random variable, whose possible
# values are collected in 'Levels'

district <- c('MI',  'MI',  'VA',  'BG',  'LO', 'LO',  'CR',  'Alt', 'CR',  'MI',
              'Alt', 'CR',  'LO',  'VA',  'MI',  'Alt', 'LO',  'MI')
district <- factor(district,levels=c('MI','LO','BG','CR','VA','Alt'))
district
```

```
##  [1] MI  MI  VA  BG  LO  LO  CR  Alt CR  MI  Alt CR  LO  VA  MI  Alt LO  MI
## Levels: MI LO BG CR VA Alt
```

```
resass <- table(district) # table of absolute frequences
resass
```

```
## district
##  MI  LO  BG  CR  VA Alt
##   5   4   1   3   2   3
```

```
resrel <- table(district)/length(district) # table of relative frequences
resrel
```

```
## district
##         MI         LO         BG         CR         VA        Alt
## 0.27777778 0.22222222 0.05555556 0.16666667 0.11111111 0.16666667
```

```
## Lists
## Objects made of objects (objects can be of different type).
exam <- list (course = 'Applied Statistics',
              date = '27/09/2014',
              enrolled = 7,
              corrected = 6,
              student_id = as.character(c(45020,45679,46789,43126,42345,47568,45674)),
              evaluation = c(30,29,30,NA,25,26,27))
exam
```

```
## $course
## [1] "Applied Statistics"
##
## $date
## [1] "27/09/2014"
##
## $enrolled
## [1] 7
##
## $corrected
## [1] 6
##
## $student_id
## [1] "45020" "45679" "46789" "43126" "42345" "47568" "45674"
##
## $evaluation
## [1] 30 29 30 NA 25 26 27
```

```
exam$evaluation
```

```
## [1] 30 29 30 NA 25 26 27
```

```
exam[[6]]
```

```
## [1] 30 29 30 NA 25 26 27
```

```
## data.frame
## Objects made of vectors of the same lengths, possibly of different types.
## (Remark: they look like matrices by they aren't!)
exam <- data.frame(
  student_id = factor(as.character(c(45020,45679,46789,43126,42345,47568,45674))),
  evaluation_W = c(30,29,30,NA,25,26,17),
  evaluation_O = c(28,30,30,NA,28,27,NA),
  evaluation_P = c(30,30,30,30,28,28,28),
  outcome  = factor(c('Passed','Passed','Passed','To be repeated','Passed','Passed','To be repeated')))
exam
```

```
##   student_id evaluation_W evaluation_O evaluation_P        outcome
## 1      45020           30           28           30         Passed
## 2      45679           29           30           30         Passed
## 3      46789           30           30           30         Passed
## 4      43126           NA           NA           30 To be repeated
## 5      42345           25           28           28         Passed
## 6      47568           26           27           28         Passed
## 7      45674           17           NA           28 To be repeated
```

```
exam$evaluation_W    # a data.frame is a particular kind of list!
```

```
## [1] 30 29 30 NA 25 26 17
```

```
exam[[2]]
```

```
## [1] 30 29 30 NA 25 26 17
```

```
exam[2,]
```

```
##   student_id evaluation_W evaluation_O evaluation_P outcome
## 2      45679           29           30           30  Passed
```

```
attach(exam)
evaluation_W
```

```
## [1] 30 29 30 NA 25 26 17
```

```
detach(exam)
```

```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Data Import/Export
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
record <- read.table('record.txt', header=T)

head(record)
```

```
##           m100  m200  m400 m800 m1500 m3000 Marathon
## argentin 11.61 22.94 54.50 2.15  4.43  9.79   178.52
## australi 11.20 22.35 51.08 1.98  4.13  9.08   152.37
## austria  11.43 23.09 50.62 1.99  4.22  9.34   159.37
## belgium  11.41 23.04 52.00 2.00  4.14  8.88   157.85
## bermuda  11.46 23.05 53.30 2.16  4.58  9.81   169.98
## brazil   11.31 23.17 52.80 2.10  4.49  9.77   168.75
```

```
dim(record)
```

```
## [1] 55  7
```

```
dimnames(record)
```

```
## [[1]]
##  [1] "argentin" "australi" "austria"  "belgium"  "bermuda"  "brazil"
##  [7] "burma"    "canada"   "chile"    "china"    "columbia" "cookis"
## [13] "costa"    "czech"    "denmark"  "domrep"   "finland"  "france"
## [19] "gdr"      "frg"      "gbni"     "greece"   "guatemal" "hungary"
## [25] "india"    "indonesi" "ireland"  "israel"   "italy"    "japan"
## [31] "kenya"    "korea"    "dprkorea" "luxembou" "malaysia" "mauritiu"
## [37] "mexico"   "netherla" "nz"       "norway"   "png"      "philippi"
## [43] "poland"   "portugal" "rumania"  "singapor" "spain"    "sweden"
## [49] "switzerl" "taipei"   "thailand" "turkey"   "usa"      "ussr"
## [55] "wsamoa"
##
## [[2]]
## [1] "m100"     "m200"     "m400"     "m800"     "m1500"    "m3000"    "Marathon"
```

```
# Transform times in seconds
record[,4:7] <- record[,4:7]*60
head(record)
```

```
##           m100  m200  m400  m800 m1500 m3000 Marathon
## argentin 11.61 22.94 54.50 129.0 265.8 587.4  10711.2
## australi 11.20 22.35 51.08 118.8 247.8 544.8   9142.2
## austria  11.43 23.09 50.62 119.4 253.2 560.4   9562.2
## belgium  11.41 23.04 52.00 120.0 248.4 532.8   9471.0
## bermuda  11.46 23.05 53.30 129.6 274.8 588.6  10198.8
## brazil   11.31 23.17 52.80 126.0 269.4 586.2  10125.0
```

```
# to save a data frame (or a matrix)
write.table(record, file = 'record_mod.txt')

# Remark. The file containing 'record_mod.txt' will be saved in the working directory
# to save several objects in the workspace
W <- matrix(data = c(11,12,13,14,15,16), nrow = 2, ncol = 3, byrow = F)
V <- t(W)
a <- 1

save(W,V,a, file = 'variousobjects.RData')

# to save the entire workspace: save.image('FILENAME.RData')
save.image("myworkspace.RData")

# this command remove all the variable of the workspace
rm(list=ls())
ls()
```

```
## character(0)
```

```
# to Load a workspace (i.e., .RData)
load("variousobjects.RData")

ls()
```

```
## [1] "a" "V" "W"
```

```
### ---------------------------------------------------------------------------
### ---------------------------------------------------------------------------
### Example: analysis of quantitative data (with plots)
### ---------------------------------------------------------------------------
### ---------------------------------------------------------------------------
record <- read.table('record_mod.txt', header=T)
head(record)
```

```
##           m100  m200  m400  m800 m1500 m3000 Marathon
## argentin 11.61 22.94 54.50 129.0 265.8 587.4  10711.2
## australi 11.20 22.35 51.08 118.8 247.8 544.8   9142.2
## austria  11.43 23.09 50.62 119.4 253.2 560.4   9562.2
## belgium  11.41 23.04 52.00 120.0 248.4 532.8   9471.0
## bermuda  11.46 23.05 53.30 129.6 274.8 588.6  10198.8
## brazil   11.31 23.17 52.80 126.0 269.4 586.2  10125.0
```

```
# some synthetic indices
colMeans(record)
```

```
##        m100        m200        m400        m800       m1500       m3000
##    11.61855    23.64164    53.40582   124.58182   259.52727   566.85818
##    Marathon
## 10395.19636
```

```
sapply(record, mean)
```

```
##        m100        m200        m400        m800       m1500       m3000
##    11.61855    23.64164    53.40582   124.58182   259.52727   566.85818
##    Marathon
## 10395.19636
```

```
sapply(record, sd)
```

```
##         m100         m200         m400         m800        m1500        m3000
##    0.4522103    1.1110602    2.6783367    6.4934466   19.9455319   49.4601474
##     Marathon
## 1825.7726951
```

```
sapply(record, var)
```

```
##         m100         m200         m400         m800        m1500        m3000
## 2.044941e-01 1.234455e+00 7.173488e+00 4.216485e+01 3.978242e+02 2.446306e+03
##     Marathon
## 3.333446e+06
```

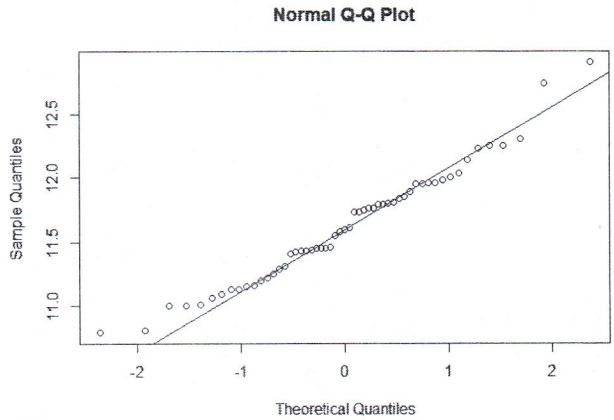
```
cov(record)
```

```
##                m100         m200        m400        m800        m1500
## m100       0.2044941    0.4787135   1.010955    2.136788    6.569596
## m200       0.4787135    1.2344547   2.550142    5.223808   15.476232
## m400       1.0109549    2.5501422   7.173488   15.624737   42.087172
## m800       2.1367879    5.2238081  15.624737   42.164848  116.772727
## m1500      6.5695960   15.4762323  42.087172  116.772727  397.824242
## m3000     16.5891232   39.0096808 103.014285  277.348485  956.093939
## Marathon 566.6616242 1390.7175616 3449.547725 9238.943636 31970.827879
##                m3000     Marathon
## m100        16.58912     566.6616
## m200        39.00968    1390.7176
## m400       103.01428    3449.5477
## m800       277.34848    9238.9436
## m1500      956.09394   31970.8279
## m3000     2446.30618   81258.0017
## Marathon 81258.00170 3333445.9341
```

```
cor(record)
```

```
##               m100      m200      m400      m800      m1500     m3000  Marathon
## m100     1.0000000 0.9527911 0.8346918 0.7276888 0.7283709 0.7416988 0.6863358
## m200     0.9527911 1.0000000 0.8569621 0.7240597 0.6983643 0.7098710 0.6855745
## m400     0.8346918 0.8569621 1.0000000 0.8984052 0.7878417 0.7776369 0.7054241
## m800     0.7276888 0.7240597 0.8984052 1.0000000 0.9016138 0.8635652 0.7792922
## m1500    0.7283709 0.6983643 0.7878417 0.9016138 1.0000000 0.9691690 0.8779334
## m3000    0.7416988 0.7098710 0.7776369 0.8635652 0.9691690 1.0000000 0.8998374
## Marathon 0.6863358 0.6855745 0.7054241 0.7792922 0.8779334 0.8998374 1.0000000
```

```
### ------------------------------------------------------------------------
### Descriptive/inferential analysis on the variable m100
attach(record)

### t-test for the mean value of the quantity
### H0: mean==11.5 vs H1: mean!=11.5

# Recall: qqplot to verify (qualitatively) the Gaussian assumption on the
# distribution generating sample
qqnorm(m100) # quantile-quantile plot
qqline(m100, col='red') # theoretical line
```




```
# Recall: Shapiro-Wilk test to verify (quantitatively) the Gaussian assumption on the
# distribution generating sample
shapiro.test(m100)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  m100
## W = 0.97326, p-value = 0.2569
```

```
alpha       <- .05
sample.mean <- mean(m100)
mean.H0     <- 11.5
sample.sd   <- sd(m100)
n           <- length(m100)
tstat       <- (sample.mean - mean.H0)/(sample.sd/sqrt(n))
cfr.t       <- qt( 1 - alpha/2, n-1 )
abs(tstat) < cfr.t  # cannot reject H0 (accept H0)
```

```
## [1] TRUE
```

```
pval  <- ifelse(tstat >= 0, (1 - pt(tstat, n-1))*2, pt(tstat, n-1)*2)
pval
```

p-value

```
## [1] 0.05709702
```

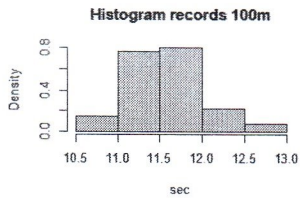
```
IC <- c(inf    = sample.mean - sample.sd/sqrt(n) * qt( 1 - alpha/2, n-1 ),
        center = sample.mean,
        sup    = sample.mean + sample.sd/sqrt(n) * qt( 1 - alpha/2, n-1 ))
IC
```

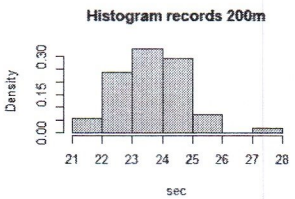
```
##      inf   center      sup
## 11.49630 11.61855 11.74080
```

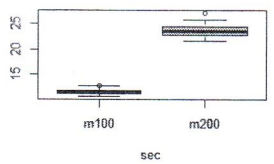
```
# automatically
t.test(m100, mu = mean.H0, alternative = 'two.sided', conf.level = 1-alpha)
```

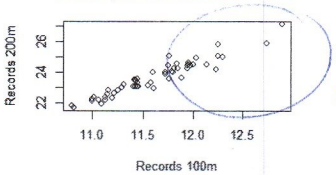
```
##
##  One Sample t-test
##
## data:  m100
## t = 1.9441, df = 54, p-value = 0.0571
## alternative hypothesis: true mean is not equal to 11.5
## 95 percent confidence interval:
##  11.4963 11.7408
## sample estimates:
## mean of x
##  11.61855
```

```
### ------------------------------------------------------------------------
### Simple linear regression (variable m200 vs m100)

# More than one plot in a unique device (commands par or layout)
# (command par)
x11()
par(mfrow=c(2,2))
hist(m100,prob=T,main="Histogram records 100m",xlab="sec")
hist(m200,prob=T,main="Histogram records 200m",xlab="sec")
boxplot(record[,1:2],main="Boxplot records 100m e 200m",xlab="sec")
plot(m100,m200, main='Scatter plot records 100m e 200m',xlab="Records 100m",ylab="Records 200m")
```







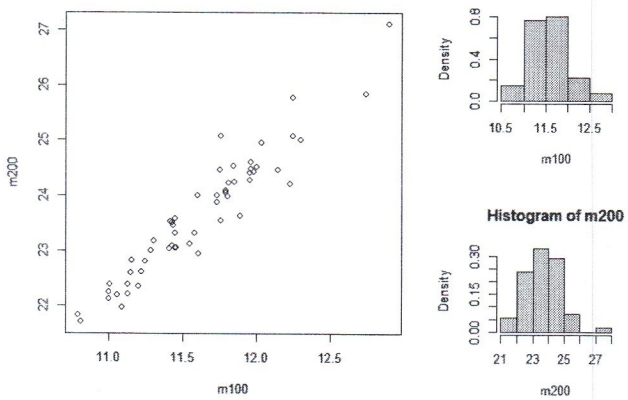


Here we can notice that σ^2 seems to change when we go up with the values of (x,y)
→ ETEROSCHEDASTICITY??
We want σ^2 to be constant!

!!

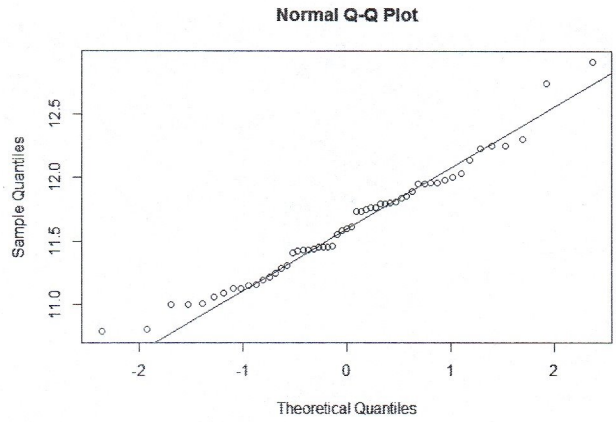
```
dev.off()
```

```
# command layout
x11()
layout(cbind(c(1,1), c(2,3)), widths=c(2,1), heights=c(1,1))
plot(m100,m200)
hist(m100, prob=T)
hist(m200, prob=T)
```




```
dev.off()
```

```
# Fit of the linear model (command lm)
# Model: m200=beta0*m100+eps, eps~N(0,sigma^2)
regression <- lm(m200 ~ m100)
```




```
regression
```

```
##
## Call:
## lm(formula = m200 ~ m100)
##
## Coefficients:
## (Intercept)         m100
##      -3.557        2.341
```

```
summary(regression)
```

```
##
## Call:
## lm(formula = m200 ~ m100)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.86303 -0.16559 -0.00756  0.16599  1.10722
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -3.5570     1.1914  -2.985  0.00428 **
## m100          2.3410     0.1025  22.845  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3405 on 53 degrees of freedom
## Multiple R-squared:  0.9078, Adjusted R-squared:  0.9061
## F-statistic: 521.9 on 1 and 53 DF,  p-value: < 2.2e-16
```

IP of observing $\beta_1 = 2.3410$ if $H_0: \beta_1 = 0$ is true
⟹ since it's so low either we're seeing a miracle or H_0 is false
⟹ $H_0: \beta_1 = 0$ REJECTED

```
coef(regression)
```

```
## (Intercept)        m100
##   -3.556967    2.340965
```

```
vcov(regression)
```

```
##             (Intercept)        m100
## (Intercept)   1.4195378 -0.12199717
## m100         -0.1219972  0.01050021
```

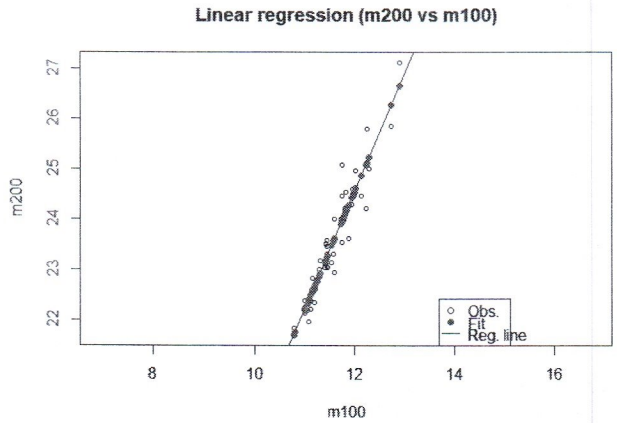
```
residuals(regression)
```

```
##           1           2           3           4           5           6
## -0.681631757 -0.311836293 -0.110258139 -0.113438848 -0.220487075  0.250657607
##           7           8           9          10          11          12
## -0.392342968  0.056356617 -0.014607931 -0.007559704  0.401777888  0.458523975
##          13          14          15          16          17          18
##  0.159030651 -0.434330192  0.343151507  0.006994624 -0.107968774  0.045211935
##          19          20          21          22          23          24
## -0.038860119  0.172946971 -0.063643383  0.036994624  0.379946396 -0.187077430
##          25          26          27          28          29          30
## -0.137559704  0.056536751  0.309741861  0.322922570  0.127476898  0.097452497
##          31          32          33          34          35          36
## -0.022547503  0.049030651  0.660150932  0.355163132 -0.863029777  1.107223560
##          37          38          39          40          41          42
## -0.657101831  0.031115480 -0.351173885 -0.241402821 -0.049849068 -0.432776440
##          43          44          45          46          47          48
## -0.287968774  0.130175333  0.236332216 -0.236897296 -0.086415022  0.251802289
##          49          50          51          52          53          54
##  0.062922570 -0.088655584  0.510633206 -0.047788640  0.127959172 -0.144101256
##          55
## -0.416921697
```

```
fitted(regression)
```

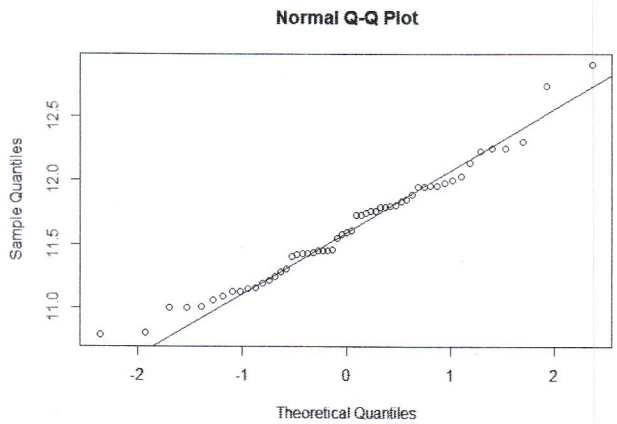
```
##        1        2        3        4        5        6        7        8
## 23.62163 22.66184 23.20026 23.15344 23.27049 22.91934 24.86234 22.19364
##        9       10       11       12       13       14       15       16
## 24.53461 24.41756 23.59822 26.64148 24.44097 22.40433 23.17685 24.04301
##       17       18       19       20       21       22       23       24
## 22.49797 22.54479 21.74886 22.21705 22.19364 24.04301 24.16005 23.24708
##       25       26       27       28       29       30       31       32
## 24.41756 24.18346 23.20026 23.24708 22.87252 23.90255 23.90255 24.44097
##       33       34       35       36       37       38       39       40
## 25.11985 24.60484 25.07303 23.97278 24.27710 22.77888 23.48117 23.55140
##       41       42       43       44       45       46       47       48
## 25.11985 23.97278 22.49797 24.08982 23.22367 25.23690 24.06642 22.56820
##       49       50       51       52       53       54       55
## 23.24708 22.70866 23.94937 24.48779 21.70204 22.33410 26.26692
```

```
x11()
plot(m100, m200, asp=1,cex=0.75)
abline(coef(regression))
points(m100, fitted(regression), col='red', pch=19)
legend('bottomright',c('Obs.','Fit','Reg. line'),col=c('black','red','black'),lwd=c(1,1,1),lty=c(-1,-1,1),pch=c(c(1,19,-1)))
title(main='Linear regression (m200 vs m100)')
```




```
dev.off()
```

```
# Test F "by hand" (H0: beta0=0 vs H1: beta0!=0)
SSreg <- sum((fitted(regression) - mean(m200))^2)
```




```
SSres <- sum(residuals(regression)^2)
SStot <- sum((m200 - mean(m200))^2)

n     <- length(m200)
Fstat <- (SSreg/1) / (SSres/(n-2))
P     <- 1 - pf(Fstat, 1, n-2)
P # reject H0
```

```
## [1] 0
```

```
# Confidence and prediction intervals (command predict)
newdata <- data.frame(m100=c(10,11,12))
pred_nd <- predict(regression, newdata)
pred_nd
```

pointwise

```
##        1        2        3
## 19.85268 22.19364 24.53461
```

```
IC_nd <- predict(regression, newdata, interval = 'confidence', level = .99)
IC_nd
```

confidence interval

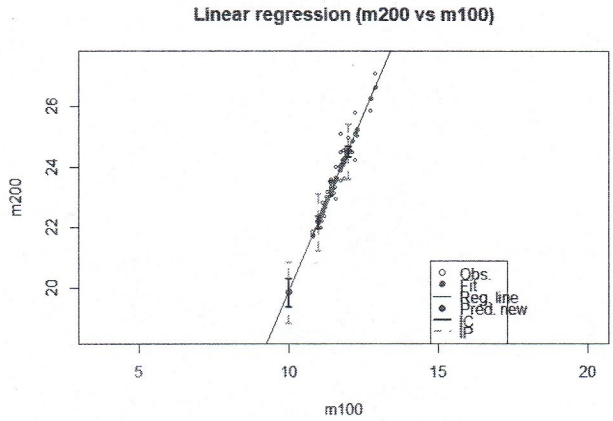
```
##        fit      lwr      upr
## 1 19.85268 19.39288 20.31248
## 2 22.19364 21.98453 22.40276
## 3 24.53461 24.37350 24.69572
```

```
IP_nd <- predict(regression, newdata, interval = 'prediction', level = .99)
IP_nd
```

prediction interval (which takes into account the variance around the mean) !

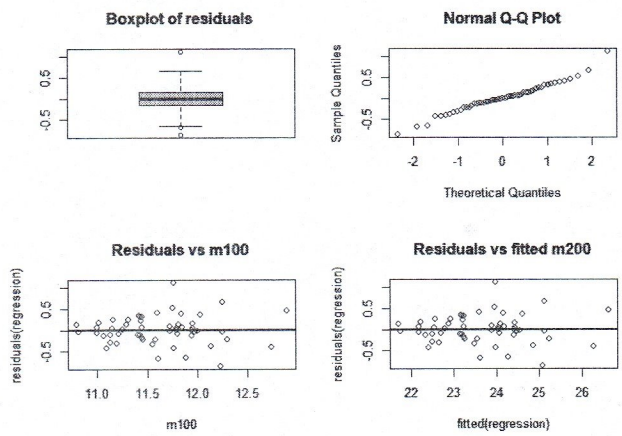
```
##        fit      lwr      upr
## 1 19.85268 18.83330 20.87206
## 2 22.19364 21.26013 23.12716
## 3 24.53461 23.61066 25.45856
```

```
x11()
plot(m100, m200, asp=1,ylim=c(18.5,27.5),cex=0.5)
abline(coef(regression))
points(m100, fitted(regression), col='red', pch=20)
points(c(10,11,12),pred_nd,col='blue',pch=16)
matlines(rbind(c(10,11,12),c(10,11,12)),t(IP_nd[,-1]),type="l",lty=2,col='dark grey',lwd=2)
matpoints(rbind(c(10,11,12),c(10,11,12)),t(IP_nd[,-1]),pch="-",lty=2,col='dark grey',lwd=2,cex=1.5)
matlines(rbind(c(10,11,12),c(10,11,12)),t(IC_nd[,-1]),type="l",lty=1,col='black',lwd=2)
matpoints(rbind(c(10,11,12),c(10,11,12)),t(IC_nd[,-1]),pch="-",lty=1,col='black',lwd=2,cex=1.5)
legend('bottomright',c('Obs.','Fit','Reg. line','Pred. new','IC','IP'),col=c('black','red','black','blue','black','dark gre
y'),lwd=c(1,1,1,1,2,2),lty=c(-1,-1,1,-1,1,2),pch=c(c(1,19,-1,19,-1,-1)))
title(main='Linear regression (m200 vs m100)')
```



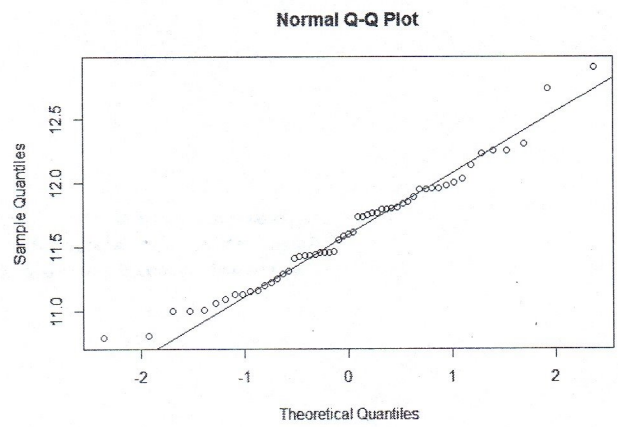

```
dev.off()
```

```
# diagnostic of residuals
x11()
par (mfrow=c(2,2))
boxplot(residuals(regression), main='Boxplot of residuals')
qqnorm(residuals(regression))
plot(m100, residuals(regression), main='Residuals vs m100')
abline(h=0, lwd=2)
plot(fitted(regression), residuals(regression), main='Residuals vs fitted m200')
abline(h=0, lwd=2)
```




```
dev.off()
```

```
detach(record)
```

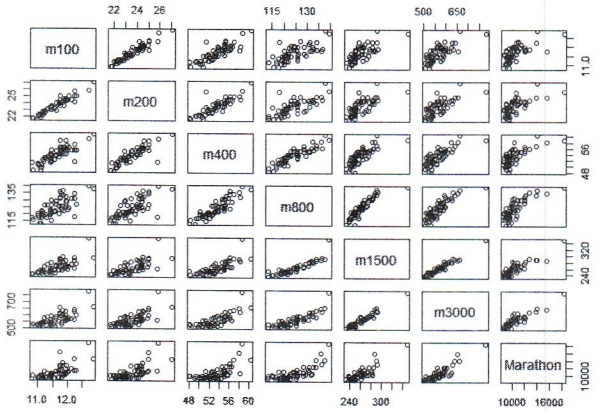



```
graphics.off()
```

```
### ---------------------------------------------------------------------------
### ---------------------------------------------------------------------------
### Visualization of Multivariate Data
### ---------------------------------------------------------------------------
### ---------------------------------------------------------------------------
### Example 1: dataset record (all the variables)
record <- read.table('record_mod.txt', header=T)
head(record)
```

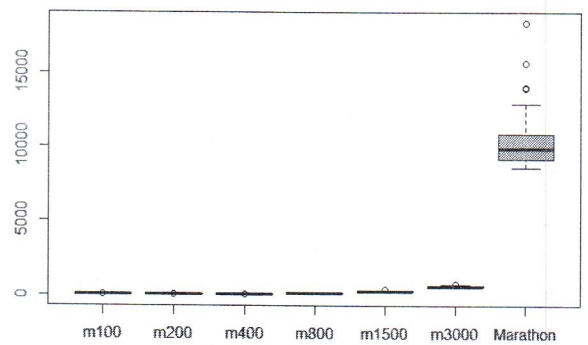
```
##           m100  m200  m400  m800 m1500 m3000 Marathon
## argentin 11.61 22.94 54.50 129.0 265.8 587.4  10711.2
## australi 11.20 22.35 51.08 118.8 247.8 544.8   9142.2
## austria  11.43 23.09 50.62 119.4 253.2 560.4   9562.2
## belgium  11.41 23.04 52.00 120.0 248.4 532.8   9471.0
## bermuda  11.46 23.05 53.30 129.6 274.8 588.6  10198.8
## brazil   11.31 23.17 52.80 126.0 269.4 586.2  10125.0
```

```
# Scatterplot
x11()
pairs(record)  # or plot(record)
```

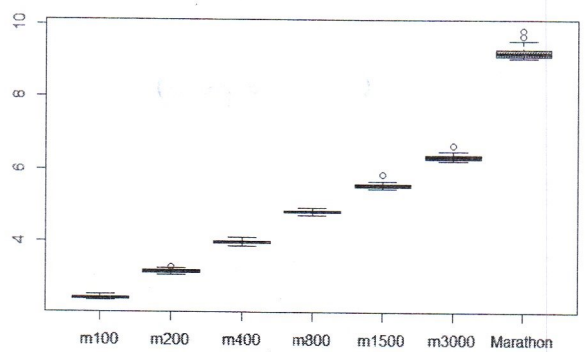


← this is so related to the covariance matrix (the covariance matrix gives, as in here, the informations about the correlations between variables)

```
# Boxplot
boxplot(record, col='gold')
```




```
boxplot(log(record), col='gold')
```




```
# Starplot
x11()
stars(record, col.stars=rep('gold',55))
```

each statistical unit is a star and each "ray" (raggio) represents the length of a different variable

→ detect outliers / check patterns (useful with a small number of units)

```
# Radarplot
stars(record, draw.segments=T)
```

(same ↑)

```
# Chernoff faces
x11()
source('faces.R')
faces(record)
```

(same ↑)

similar faces have similar values of the features

```
### ------------------------------------------------------------------------
### Example 2: cerebral aneurysm
aneurysm <- read.table('aneurysm.txt', header=T, sep=',')
head(aneurysm)
```

(Politecnico project)

```
##           R1         R2         C1         C2 POSLH ROT
## 1  4.6123692  1.8189348 -0.7113495 -0.8126038     1   1
## 2  3.0635290  0.5358821 -0.2321406  0.2195964     1   2
## 3 -2.5073081 -1.8296361 -0.1477957  1.0275683     2   1
## 4 -0.4831741 -1.3670522 -0.1344654  0.7809211     2   2
## 5  8.6775054 -0.7214820 -0.8779464 -0.8643967     1   1
## 6 -1.4005311  0.3016918  0.3694719  0.8115340     2   1
```

```
dim(aneurysm)
```

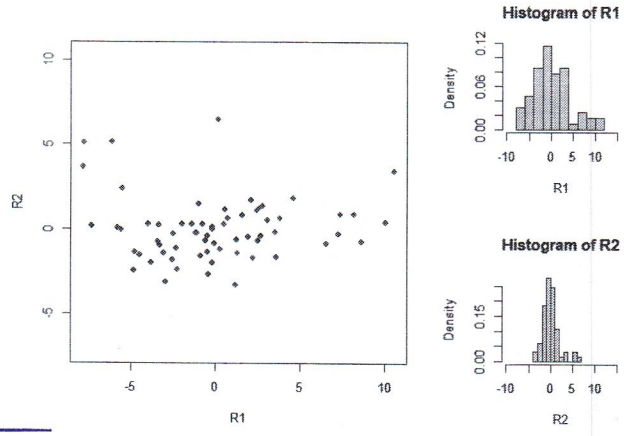
```
## [1] 65  6
```

```
aneurysm.geometry <- aneurysm[,1:4]
aneurysm.position <- factor(aneurysm[,5])

head(aneurysm.geometry)
```

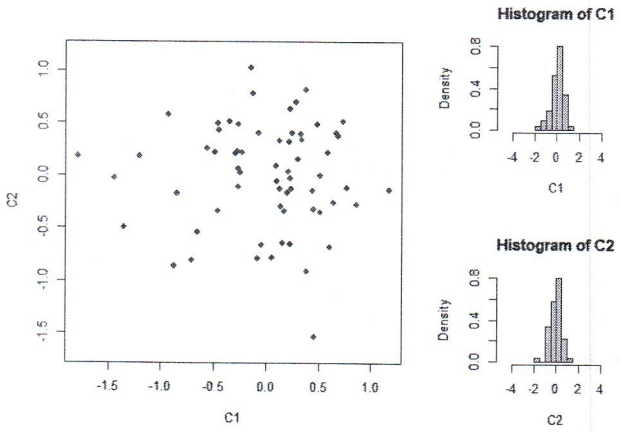
```
##           R1         R2         C1         C2
## 1  4.6123692  1.8189348 -0.7113495 -0.8126038
## 2  3.0635290  0.5358821 -0.2321406  0.2195964
## 3 -2.5073081 -1.8296361 -0.1477957  1.0275683
## 4 -0.4831741 -1.3670522 -0.1344654  0.7809211
## 5  8.6775054 -0.7214820 -0.8779464 -0.8643967
## 6 -1.4005311  0.3016918  0.3694719  0.8115340
```

useful plot (colored)

```
color.position <- ifelse(aneurysm.position == '1', 'red', 'blue')

attach(aneurysm.geometry)

x11()
layout(cbind(c(1,1), c(2,3)), widths=c(2,1), heights=c(1,1))
plot(R1,R2, asp=1, col=color.position,pch=16)
hist(R1, prob=T, xlim=c(-10,15))
hist(R2, prob=T, xlim=c(-10,15))
```




```
x11()
layout(cbind(c(1,1), c(2,3)), widths=c(2,1), heights=c(1,1))
plot(C1,C2, asp=1, col=color.position,pch=16)
hist(C1, prob=T, xlim=c(-5,5))
hist(C2, prob=T, xlim=c(-5,5))
```




```
graphics.off()

detach(aneurysm.geometry)

# some statistical indices
sapply(aneurysm.geometry,mean)
```

```
##           R1           R2           C1           C2
## 1.181730e-16 1.798174e-16 1.490403e-16 6.452879e-17
```

```
sapply(aneurysm.geometry,sd)
```

```
##        R1        R2        C1        C2
## 4.1859403 1.8563342 0.5871577 0.4940132
```

```
cov(aneurysm.geometry)
```

```
##              R1           R2           C1           C2
## R1  1.752210e+01 -9.654685e-16 -1.305303e+00 -4.165673e-01
## R2 -9.654685e-16  3.445977e+00 -5.419870e-02 -1.502647e-01
## C1 -1.305303e+00 -5.419870e-02  3.447541e-01  2.946215e-16
## C2 -4.165673e-01 -1.502647e-01  2.946215e-16  2.440490e-01
```

we can try to make 2 for each function: one for the blue ones and one for the red ones (from the plot) and see their difference

```
cor(aneurysm.geometry)
```

```
##              R1           R2           C1           C2
## R1  1.000000e+00 -1.242479e-16 -5.310843e-01 -2.014436e-01
## R2 -1.242479e-16  1.000000e+00 -4.972537e-02 -1.638560e-01
## C1 -5.310843e-01 -4.972537e-02  1.000000e+00  1.015713e-15
## C2 -2.014436e-01 -1.638560e-01  1.015713e-15  1.000000e+00
```

```
# Attention: rounded zeros!
round(sapply(aneurysm.geometry,mean),1)
```

```
## R1 R2 C1 C2
##  0  0  0  0
```

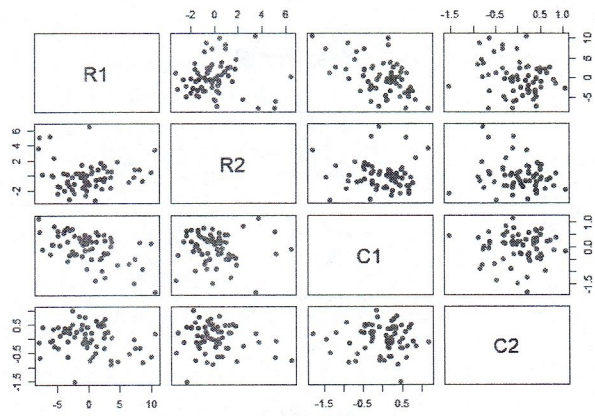
```
round(cov(aneurysm.geometry),1)
```

```
##      R1   R2   C1   C2
## R1 17.5  0.0 -1.3 -0.4
## R2  0.0  3.4 -0.1 -0.2
## C1 -1.3 -0.1  0.3  0.0
## C2 -0.4 -0.2  0.0  0.2
```

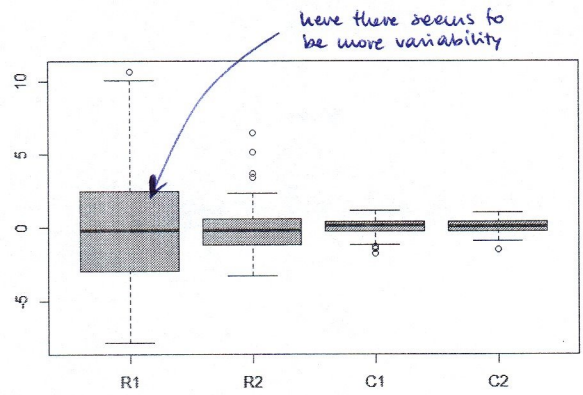
```
round(cor(aneurysm.geometry),1)
```

```
##      R1   R2   C1   C2
## R1  1.0  0.0 -0.5 -0.2
## R2  0.0  1.0  0.0 -0.2
## C1 -0.5  0.0  1.0  0.0
## C2 -0.2 -0.2  0.0  1.0
```

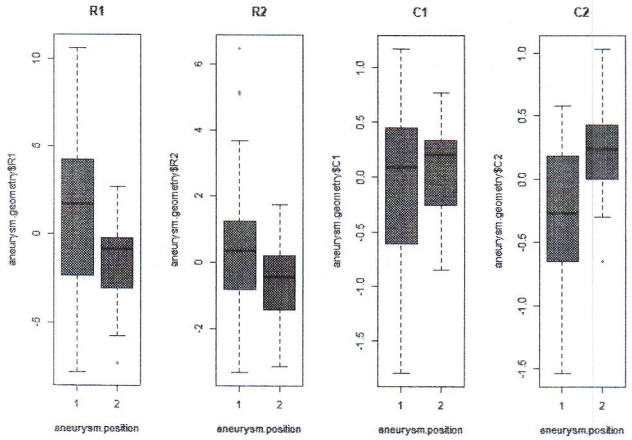
```
# Scatterplot
x11()
pairs(aneurysm.geometry, col=color.position, pch=16)
```



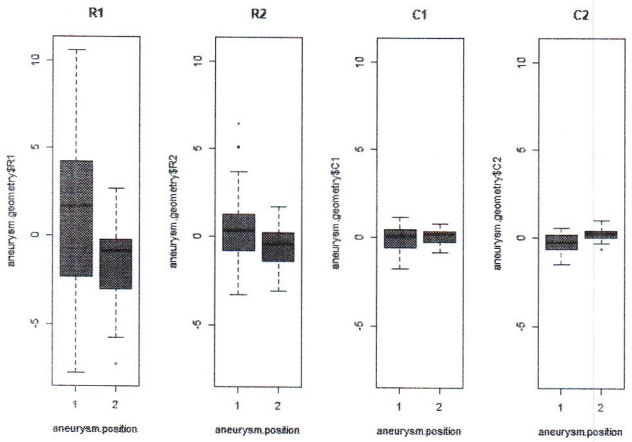

```
# Boxplot
boxplot(aneurysm.geometry, col='gold')
```




```
# Stratified boxplots
x11()
par(mfrow = c(1,4))
boxplot(aneurysm.geometry$R1 ~ aneurysm.position, col=c('red','blue'), main='R1')
boxplot(aneurysm.geometry$R2 ~ aneurysm.position, col=c('red','blue'), main='R2')
boxplot(aneurysm.geometry$C1 ~ aneurysm.position, col=c('red','blue'), main='C1')
boxplot(aneurysm.geometry$C2 ~ aneurysm.position, col=c('red','blue'), main='C2')
```



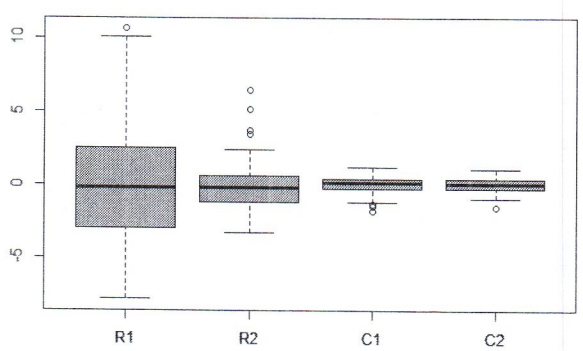

```
# Stratified boxplots (same scale)
par(mfrow = c(1,4))
boxplot(aneurysm.geometry$R1 ~ aneurysm.position, col=c('red','blue'), main='R1', ylim=range(aneurysm.geometry))
boxplot(aneurysm.geometry$R2 ~ aneurysm.position, col=c('red','blue'), main='R2', ylim=range(aneurysm.geometry))
boxplot(aneurysm.geometry$C1 ~ aneurysm.position, col=c('red','blue'), main='C1', ylim=range(aneurysm.geometry))
boxplot(aneurysm.geometry$C2 ~ aneurysm.position, col=c('red','blue'), main='C2', ylim=range(aneurysm.geometry))
```




```
dev.off()
```

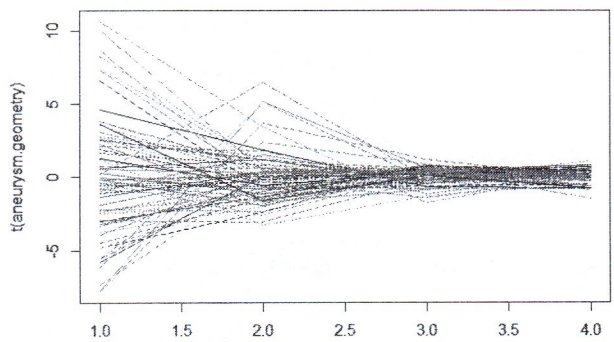
```
## windows
##       2
```

```
# Chernoff faces
source('faces.R')
```

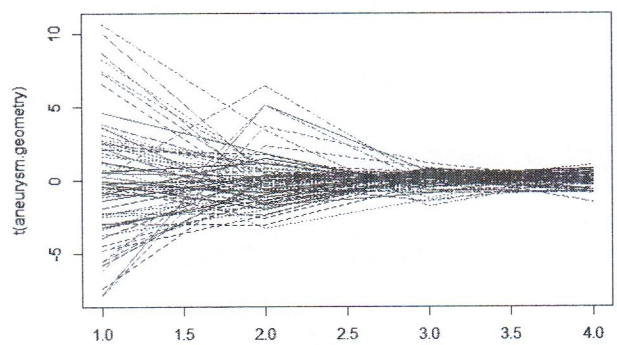



```
x11()
faces(aneurysm.geometry)
```

```
# matplot
matplot(t(aneurysm.geometry),type='l')
```



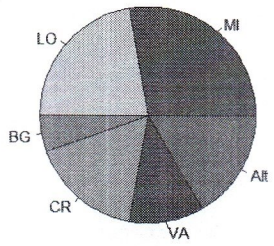

```
matplot(t(aneurysm.geometry),type='l',col=color.position)
```



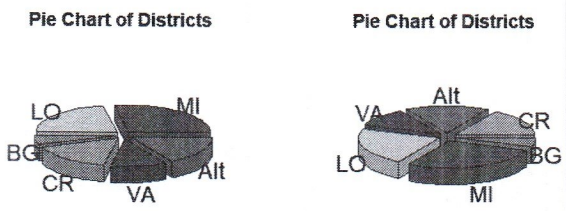

```
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
### Visualization of Categorical Data
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
district <- c('MI',  'MI',  'VA',  'BG',  'LO', 'LO',  'CR',  'Alt', 'CR',  'MI',
              'Alt', 'CR',  'LO',  'VA',  'MI',  'Alt', 'LO',  'MI')
district <- factor(district,levels=c('MI','LO','BG','CR','VA','Alt'))
district
```

```
##  [1] MI  MI  VA  BG  LO  LO  CR  Alt CR  MI  Alt CR  LO  VA  MI  Alt LO  MI
## Levels: MI LO BG CR VA Alt
```

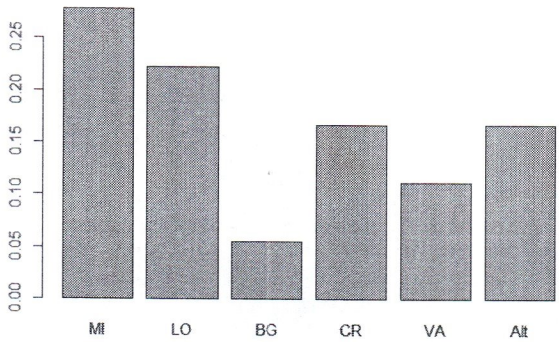
```
# Pie chart (no ordering of levels)
x11()
pie(table(district),col=rainbow(length(levels(district))))
```




```
# 3D Pie chart (not recommended!!)
library(plotrix)
x11(width = 14)
par(mfrow=c(1,2))
pie3D(table(district)[1:length(levels(district))],labels=levels(district),explode=0.1,
      main="Pie Chart of Districts ",col=rainbow(length(levels(district))))
set.seed(180317)
shuffle = sample(1:length(levels(district)), size=length(levels(district)), replace = F)
pie3D(table(district)[shuffle],labels=levels(district)[shuffle],explode=0.1,
      main="Pie Chart of Districts ",col=rainbow(length(levels(district)))[shuffle])
```



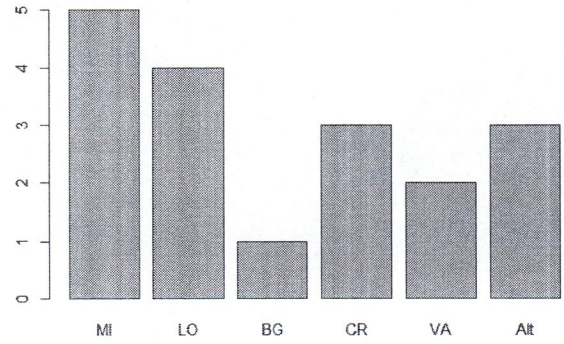

```
# Barplot (levels are ordered)
x11()
barplot(table(district)/length(district))
```




```
# or
plot(district)   # barplot of absolute frequences

# Remark: R is an object-oriented language; a function (e.g.,
#       the function plot()) may behave differently depending on the object
#       it takes as input
is(district)[1]
```

```
## [1] "factor"
```

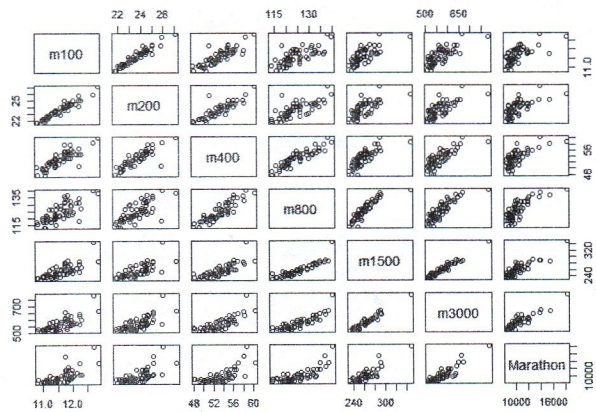
```
plot(district)
```



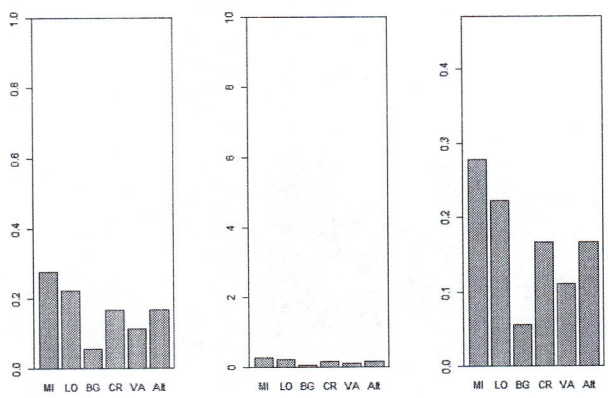

```
# record is a data frame
is(record)[1]
```

```
## [1] "data.frame"
```

```
plot(record) # scatterplot
```

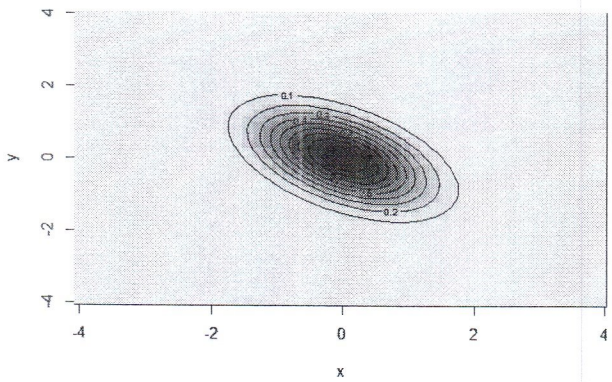



```
# Remark 2: be careful to the scale of representation
x11(width=14, height=5)
par(mfrow=c(1,3))
barplot(table(district)/length(district),ylim=c(0,1)); box()
barplot(table(district)/length(district),ylim=c(0,10)); box()
barplot(table(district)/length(district),ylim=c(0,0.47)); box()
```

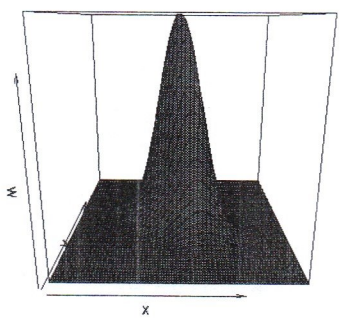



```
graphics.off()
```

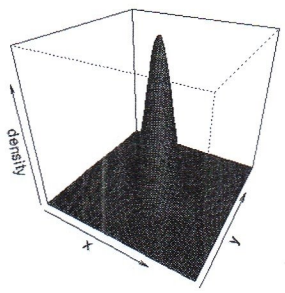
```r
### ---------------------------------------------------------------------
### ---------------------------------------------------------------------
### 3d plots
### ---------------------------------------------------------------------
### ---------------------------------------------------------------------

# For instance, let's plot a bivariate Gaussian density
x <- seq(-4,4,0.15)
y <- seq(-4,4,0.15)

# To build a function in R
gaussian <- function(x, y)
{exp(-(x^2+y^2+x*y))}

w <- matrix(NA, nrow = length(x), ncol=length(y))

# for
for(i in 1:length(x)){
  for(j in 1:length(y))
  {w[i,j] <- gaussian(x[i], y[j])}
}

# or
w <- outer(x, y, gaussian)
# help(outer)

x11()
image(x, y, w)
contour(x, y, w, add=T)
```




```r
persp(x, y, w, col='red')
```




```r
persp(x, y, w, col='red', theta=30, phi=30, shade=.05, zlab='density')
```



```r
# To dowload a package:
# from RStudio: Tools -> Instal Packages -> type PACKAGENAME and click install
# from R: Packages -> Instal Packages -> Choose a CRAN mirror
#         (e.g., Italy (Milano)) -> Choose the package and click OK

library(rgl)
persp3d(x, y, w, col='red', alpha=1)
lines3d(x,x, gaussian(x,x), col='blue', lty=1)
lines3d(x,x, 0, col='blue', lty=2)

dev.off()
```

3D plot s.t. we can interact

LAB 02

TOPICS:

- Probability density functions. Cumulative distribution functions. Quantiles
- Random number generation
- QQplots

```
### ------------------------------------------------------------------------
### Probability density functions. Cumulative distribution functions. Quantiles
### ------------------------------------------------------------------------
### Example: Gaussian distribution
# Probability density function (paf)
dnorm(0)                  # density function at 0 for a distribution N(0,1)
```

```
## [1] 0.3989423
```

```
dnorm(0,mean = 1, sd = 2) # density function at 0 for a distribution N(1,4)
```

```
## [1] 0.1760327
```

```
# Cumulative distribution function (cdf)
pnorm(0)      # P(Z<0), with Z ~ N(0,1)
```

```
## [1] 0.5
```

```
pnorm(0,1,2) # P(X<0), with X ~ N(1,4)
```

```
## [1] 0.3085375
```

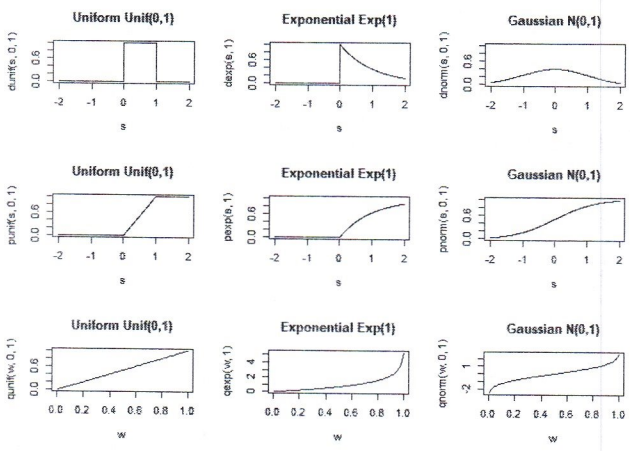
```
# Quantiles (inverse of cdf)
qnorm(0.95)       #  =z s.t.  P(Z<z)=0.95, with Z ~ N(0,1)
```

```
## [1] 1.644854
```

```
qnorm(0.95,1,2)  #  =z s.t.  P(Z<z)=0.95, with Z ~ N(1,4)
```

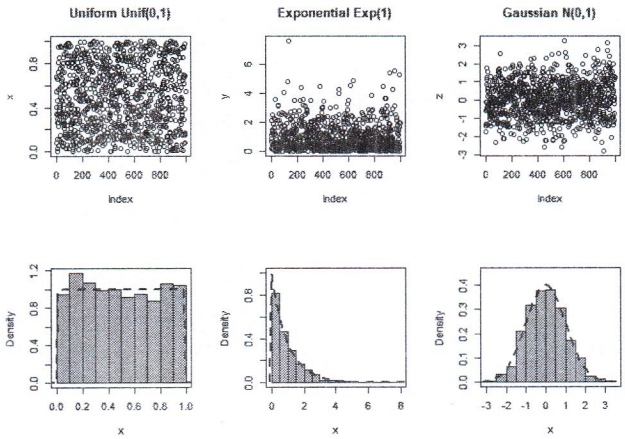
```
## [1] 4.289707
```

```
### ------------------------------------------------------------------------
### Commands to generate random numbers, obtain pdfs
### cdf and its inverse for the most popular models:
#   Commands rnorm(),  dnorm(),  pnorm(),  qnorm(),
#            rexp(),   dexp(),   pexp(),   qexp(),
#            runif(),  dunif(),  punif(),  qunif(),
#            rbinom(), dbinom(), pbinom(), qbinom(),
#            rpois(),  dpois(),  ppois(),  qpois(),
#            rgamma(), dgamma(), pgamma(), qgamma(),
### ------------------------------------------------------------------------

### Other examples of distributions
x11(width=21, height=21)
layout(matrix(c(1,2,3,4,5,6,7,8,9), 3, byrow=T))

s <- seq(-2, 2, by = 0.01)

plot(s, dunif(s, 0, 1), main='Uniform Unif(0,1)', type = 'l', ylim = c(0,1))
plot(s, dexp(s, 1),     main='Exponential Exp(1)', type = 'l', ylim = c(0,1))
plot(s, dnorm(s, 0, 1), main='Gaussian N(0,1)', type = 'l', ylim = c(0,1))

plot(s, punif(s, 0, 1), main='Uniform Unif(0,1)', type = 'l', ylim = c(0,1))
plot(s, pexp(s, 1),     main='Exponential Exp(1)', type = 'l', ylim = c(0,1))
plot(s, pnorm(s, 0, 1), main='Gaussian N(0,1)', type = 'l', ylim = c(0,1))

w <- seq(0.01/2, 1 - 0.01/2, by = 0.01)

plot(w, qunif(w, 0, 1), main='Uniform Unif(0,1)', type = 'l')
plot(w, qexp(w, 1),     main='Exponential Exp(1)', type = 'l')
plot(w, qnorm(w, 0, 1), main='Gaussian N(0,1)', type = 'l')
```



```
###
### Random number generation
###
set.seed(20032020)
x <- runif( n=1000, min=0, max=1)
y <- rexp( n=1000, rate=1)
z <- rnorm( n=1000, mean=0, sd=1)

x11(width=21, height=14)
par(mfrow=c(2,3))
plot(x, main='Uniform Unif(0,1)')
plot(y, main='Exponential Exp(1)')
plot(z, main='Gaussian N(0,1)')

hist(x, main='', col='grey', xlab='x', prob=T)
lines(seq(-0.2, 1.2, length=100), dunif(seq(-0.2, 1.2, length=100)), col='blue', lty=2, lwd=2)
box()
hist(y, main='', col='grey', xlab='x', prob=T, ylim=c(0,1))
lines(seq(-1, 9, length=100), dexp(seq(-1, 9, length=100)), col='blue', lty=2, lwd=2)
box()
hist(z, main='', col='grey', xlab='x', prob=T, ylim=c(0,.45))
lines(seq(-4, 4, length=100), dnorm(seq(-4, 4, length=100)), col='blue', lty=2, lwd=2)
box()
```

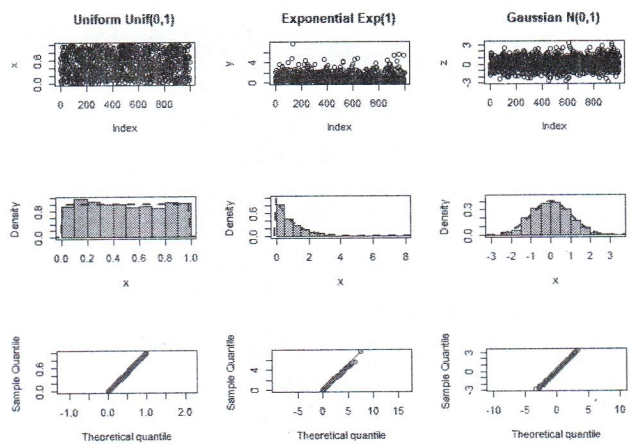
it allows to get the same generation of random numbers every time we run the code




```
graphics.off()
```

```
###
### QQplot
###
### The QQplot `qqplot()` can be used to verify if a sample comes from a given distribution.
#   Remark: the QQplot can be plotted for any distribution - not necessarily Gaussian.

x11(width=21, height=21)
layout(matrix(c(1,2,3,4,5,6,7,8,9), 3, byrow=T))

plot(x, main='Uniform Unif(0,1)')
plot(y, main='Exponential Exp(1)')
plot(z, main='Gaussian N(0,1)')

hist(x, main='', col='grey', xlab='x', prob=T)
lines(seq(-0.2, 1.2, length=100), dunif(seq(-0.2, 1.2, length=100)), col='blue', lty=2, lwd=2)
box()
hist(y, main='', col='grey', xlab='x', prob=T, ylim=c(0,1))
lines(seq(-1, 9, length=100), dexp(seq(-1, 9, length=100)), col='blue', lty=2, lwd=2)
box()
hist(z, main='', col='grey', xlab='x', prob=T, ylim=c(0,.45))
lines(seq(-4, 4, length=100), dnorm(seq(-4, 4, length=100)), col='blue', lty=2, lwd=2)
box()

qqplot(qunif((1:1000/1000-0.5/1000)), x, col='red', xlab='Theoretical quantile', ylab='Sample Quantile', asp=1)
abline(0, 1, col='blue')
qqplot(qexp((1:1000/1000-0.5/1000)), y, col='red', xlab='Theoretical quantile', ylab='Sample Quantile', asp=1)
abline(0, 1, col='blue')
qqplot(qnorm((1:1000/1000-0.5/1000)), z, col='red', xlab='Theoretical quantile', ylab='Sample Quantile', asp=1)
abline(0, 1, col='blue')
```

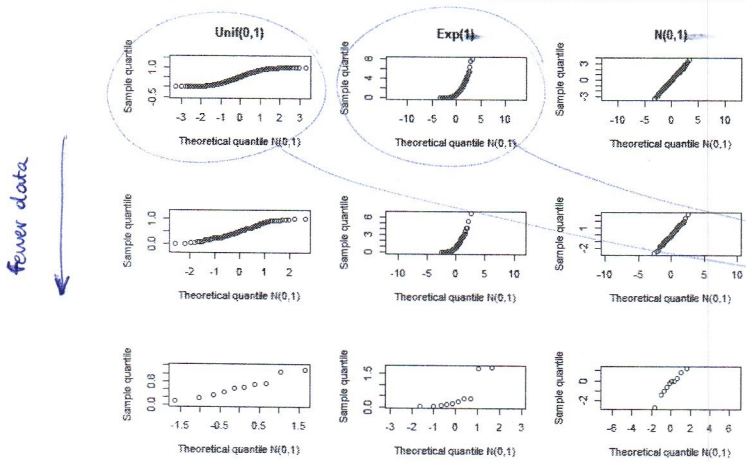
compare the theoretical quantiles of the distribution and the sample quantiles



these QQplots say that the assumptions we made are satisfied (the assumptions on the distributions)

```
# Nevertheless, the QQplot is mostly used to qualitatively verify if a sample
# comes from a Gaussian distribution
```

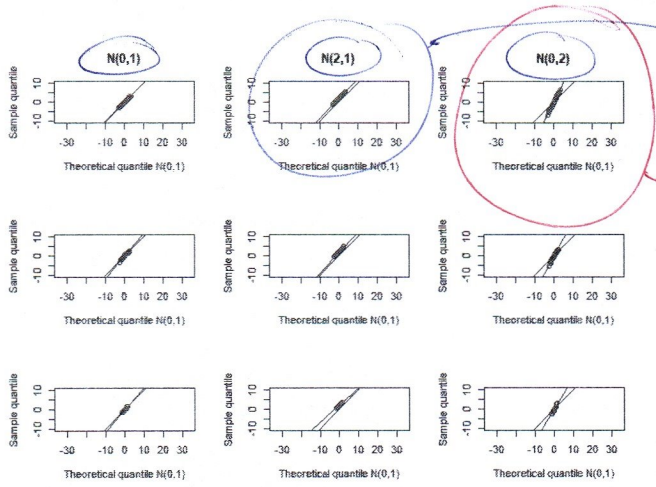
```
x11()
layout(matrix(c(1,2,3,4,5,6,7,8,9), 3, byrow=T))

x <- runif( n=1000, min=0, max=1)
y <- rexp( n=1000, rate=1)
z <- rnorm( n=1000, mean=0, sd=1)

qqplot(qnorm((1:1000/1000-1/2000)), x, col='red', xlab='Theoretical quantile N(0,1)', ylab='Sample quantile', asp=1, main='U
nif(0,1)')
qqplot(qnorm((1:1000/1000-1/2000)), y, col='red', xlab='Theoretical quantile N(0,1)', ylab='Sample quantile', asp=1, main='E
xp(1)')
qqplot(qnorm((1:1000/1000-1/2000)), z, col='red', xlab='Theoretical quantile N(0,1)', ylab='Sample quantile', asp=1, main='N
(0,1)')

x <- runif( n=100, min=0, max=1)
y <- rexp( n=100, rate=1)
z <- rnorm( n=100, mean=0, sd=1)

qqplot(qnorm((1:100/100-1/200)), x, col='red', xlab='Theoretical quantile N(0,1)', ylab='Sample quantile', asp=1)
qqplot(qnorm((1:100/100-1/200)), y, col='red', xlab='Theoretical quantile N(0,1)', ylab='Sample quantile', asp=1)
qqplot(qnorm((1:100/100-1/200)), z, col='red', xlab='Theoretical quantile N(0,1)', ylab='Sample quantile', asp=1)

x <- runif( n=10, min=0, max=1)
y <- rexp( n=10, rate=1)
z <- rnorm( n=10, mean=0, sd=1)

qqplot(qnorm((1:10/10-1/20)), x, col='red', xlab='Theoretical quantile N(0,1)', ylab='Sample quantile', asp=1)
qqplot(qnorm((1:10/10-1/20)), y, col='red', xlab='Theoretical quantile N(0,1)', ylab='Sample quantile', asp=1)
qqplot(qnorm((1:10/10-1/20)), z, col='red', xlab='Theoretical quantile N(0,1)', ylab='Sample quantile', asp=1)
```



QQ Plots with the aim of comparing the data generated by uniform, exponential and gaussian with gaussian distribution

These plots can be obtained by using the command `qqnorm()` that automatically computes the theoretical quantiles of the Gaussian distribution. Further, R creates automatically the line with the command `qqline()`, that plots the straight line through the first and third quartile.

```
## ------------------------------------------------------------------------
## What happens when the sample comes from a non-standard Gaussian?

x11(width=21, height=21)
layout(matrix(c(1,2,3,4,5,6,7,8,9), 3, byrow=T))

x <- rnorm( n=1000, mean=0, sd=1)
y <- rnorm( n=1000, mean=2, sd=1)
z <- rnorm( n=1000, mean=0, sd=2)

qqplot(qnorm((1:1000/1000-1/2000)), x, col='red', xlab='Theoretical quantile N(0,1)', ylab='Sample quantile', asp=1, ylim=c
(-5,5)*2, main='N(0,1)')
abline(0,1)
qqline(x, col='red')
qqplot(qnorm((1:1000/1000-1/2000)), y, col='red', xlab='Theoretical quantile N(0,1)', ylab='Sample quantile', asp=1, ylim=c
(-5,5)*2, main='N(2,1)')
abline(0,1)
qqline(y, col='red')
qqplot(qnorm((1:1000/1000-1/2000)), z, col='red', xlab='Theoretical quantile N(0,1)', ylab='Sample quantile', asp=1, ylim=c
(-5,5)*2, main='N(0,2)')
abline(0,1)
qqline(z, col='red')

x <- rnorm( n=100, mean=0, sd=1)
y <- rnorm( n=100, mean=2, sd=1)
z <- rnorm( n=100, mean=0, sd=2)

qqplot(qnorm((1:100/100-1/200)), x, col='red', xlab='Theoretical quantile N(0,1)', ylab='Sample quantile', asp=1, ylim=c(-5,
5)*2)
abline(0,1)
qqline(x, col='red')
qqplot(qnorm((1:100/100-1/200)), y, col='red', xlab='Theoretical quantile N(0,1)', ylab='Sample quantile', asp=1, ylim=c(-5,
5)*2)
abline(0,1)
qqline(y, col='red')
qqplot(qnorm((1:100/100-1/200)), z, col='red', xlab='Theoretical quantile N(0,1)', ylab='Sample quantile', asp=1, ylim=c(-5,
5)*2)
abline(0,1)
qqline(z, col='red')

x <- rnorm( n=10, mean=0, sd=1)
y <- rnorm( n=10, mean=2, sd=1)
z <- rnorm( n=10, mean=0, sd=2)

qqplot(qnorm((1:10/10-1/20)), x, col='red', xlab='Theoretical quantile N(0,1)', ylab='Sample quantile', asp=1, ylim=c(-5,5)*
2)
abline(0,1)
qqline(x, col='red')
qqplot(qnorm((1:10/10-1/20)), y, col='red', xlab='Theoretical quantile N(0,1)', ylab='Sample quantile', asp=1, ylim=c(-5,5)*
2)
abline(0,1)
qqline(y, col='red')
qqplot(qnorm((1:10/10-1/20)), z, col='red', xlab='Theoretical quantile N(0,1)', ylab='Sample quantile', asp=1, ylim=c(-5,5)*
2)
abline(0,1)
qqline(z, col='red')
```



fewer data

mean ≠ 0 means that we have a vertical translation

variance ≠ 1 means that we have a different slope

If the data are Gaussian, the slope of the `qqline` is an estimate of the standard deviation, the intercept is an estimate of the mean. This comes from the observation that: if x is the quantile of order α of $N(\mu, \sigma^2)$, than $z = \frac{(x-\mu)}{\sigma}$ is the quantile of order α of $Z \sim N(0,1)$, i.e., $x = \mu + \sigma * z$.

```
### Test of Gaussianity: Shapiro-Wilks test:    H0: X ~ N     vs     H1=H0^C
shapiro.test(x)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  x
## W = 0.92465, p-value = 0.3974
```

```
shapiro.test(y)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  y
## W = 0.98624, p-value = 0.9898
```

```
shapiro.test(z)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  z
## W = 0.92378, p-value = 0.3895
```