LAB 04

TOPIC:

- Testing for multivariate normality
- Experiment

```
library(mvtnorm)
library(mvnormtest)
```

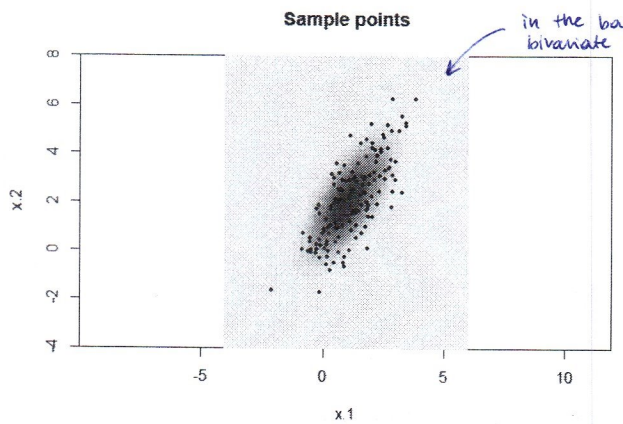
```
### ------------------------------------------------------------------------
###
### Testing the hypothesis of multivariate normality
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------

# We generate a sample of n=150 observation from bivariate Gaussian (as in LAB_3.R)
mu <- c(1,2)
mu
```

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

```
sig <- matrix(c(1,1,1,2), 2)
sig
```

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

```
n   <-  150

set.seed(20200403)
X <- rmvnorm(n, mu, sig)

x.1 <- seq(-4,6,0.15); x.2 <- seq(-4,8,0.15)
w <- matrix(NA, length(x.1), length(x.2))
for(i in 1:length(x.1)){
  for(j in 1:length(x.2)){
    w[i,j] <- dmvnorm(c(x.1[i],x.2[j]),mu,sig)
  }
}

x11()
image(x.1, x.2, w,asp=1, ylim=c(-4,8), main="Sample points")
points(X[,1],X[,2],pch=20,cex=.75)
```



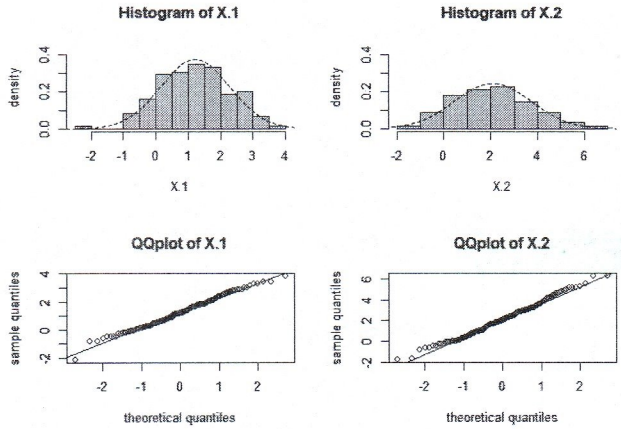
in the background there is the bivariate distribution that we have used to generate the data

```
dev.off()
```

```
### ------------------------------------------------------------------------
### Test of normality
### ------------------------------------------------------------------------
### Approach 1: look at some linear combinations of the original variables
### Example: original variables, principal components

x11()
par(mfrow=c(2,2))

hist(X[,1], prob=T, ylab='density', xlab='X.1', main='Histogram of X.1',ylim=c(0,0.45))
lines((-1000):1000 /100, dnorm((-1000):1000 /100,mean(X[,1]),sd(X[,1])), col='blue', lty=2)

hist(X[,2], prob=T, ylab='density', xlab='X.2', main='Histogram of X.2',ylim=c(0,0.45))
lines((-1000):1000 /100, dnorm((-1000):1000 /100,mean(X[,2]),sd(X[,2])), col='blue', lty=2)

qqnorm(X[,1], main='QQplot of X.1',xlab='theoretical quantiles', ylab='sample quantiles')
qqline(X[,1])

qqnorm(X[,2], main='QQplot of X.2',xlab='theoretical quantiles', ylab='sample quantiles')
qqline(X[,2])
```

1. (simpler but weaker) ← it's easier to use it to find a direction s.t. the projections are not gaussian (in that case the data is not gaussian) rather than to prove gaussianity

We check the projections on the axes:




```
# we perform univariate tests of normality on the two components
shapiro.test(X[,1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  X[, 1]
## W = 0.99154, p-value = 0.5138
```

H_0: gaussianity
H_1: not gaussianity

✓

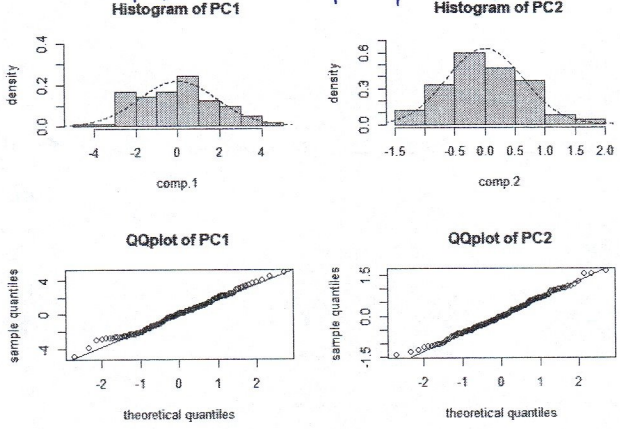
```
shapiro.test(X[,2])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  X[, 2]
## W = 0.99073, p-value = 0.4324
```

✓

```
# Recall: Shapiro-Wilk test
# H0: X~N vs H1: H0^c
# test statistics: W=(angular coeff. of the qqline)^2/sample variance
# One can prove that:
# - w<1
# - the empirical distribution under H0 is concentrated on values near 1
#   (if n=50 more than 90% of the obs. is between 0.95 and 1)
#   (if n=5 more than 90% of the obs. is between 0.81 and 1).
#
# If the data do NOT come from a Gaussian distribution, the distribution of
# the test statistics moves toward smaller values:
# Small values of the statistics W give evidence against H0
# => reject H0 for small values of W

# we look at the directions of the PCs
pc.X <- princomp(X,scores=T)
#pc.X$scores

x11()
par(mfrow=c(2,2))
hist(pc.X$scores[,1], prob=T, ylab='density', xlab='comp.1', main='Histogram of PC1',ylim=c(0,0.41))
lines((-1000):1000 /100, dnorm((-1000):1000 /100,mean(pc.X$scores[,1]),sd(pc.X$scores[,1])), col='blue', lty=2)
hist(pc.X$scores[,2], prob=T, ylab='density', xlab='comp.2', main='Histogram of PC2',ylim=c(0,0.7))
lines((-1000):1000 /100, dnorm((-1000):1000 /100,mean(pc.X$scores[,2]),sd(pc.X$scores[,2])), col='blue', lty=2)
qqnorm(pc.X$scores[,1], main='QQplot of PC1',xlab='theoretical quantiles', ylab='sample quantiles')
qqline(pc.X$scores[,1])
qqnorm(pc.X$scores[,2], main='QQplot of PC2',xlab='theoretical quantiles', ylab='sample quantiles')
qqline(pc.X$scores[,2])
```

We check the projections on the principal components:




```
shapiro.test(pc.X$scores[,1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  pc.X$scores[, 1]
## W = 0.98993, p-value = 0.36
```

✓

```
shapiro.test(pc.X$scores[,2])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  pc.X$scores[, 2]
## W = 0.99253, p-value = 0.6247
```

✓

```
# Problem:
# Which level should I use in each test, to get a global level of alpha?
```

2.

```
### ------------------------------------------------------------------------
### Approach 2
### Consider the squared Mahalanobis distances of the data from the (sample) mean
### and test if they are a sample from a chi-square distribution

# Recall:
# Theorem: if X~N(mu,Sigma) r.v. in R^p, det(Sigma)>0
#          then d2(X,mu)=(X-mu)'Sigma^-1(X-mu) ~ Chi-sq(p)

x11()
plot(X, asp=1,xlab='X.1',ylab='X.2')

library(car)
```

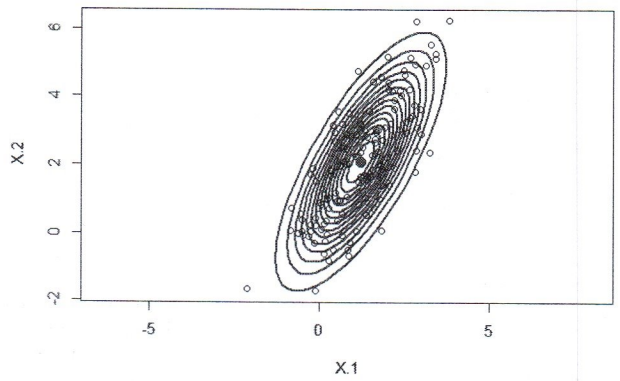
Theorem:

$$\left.\begin{array}{l} X \sim N(\mu, \Sigma) \\ \det(\Sigma) \neq 0 \\ \dim(X) = p \end{array}\right\} \Rightarrow (\underline{X}-\underline{\mu})^T \Sigma^{-1} (\underline{X}-\underline{\mu}) \sim \chi^2(p)$$

```
for(prob in (1:14)/15)
  dataEllipse(X, levels = prob , add=T)
```

here we have to use the sample variance/covariance matrix instead of Σ

NOTE: the results are stated with the true μ and Σ so we'll have some errors from the beginning

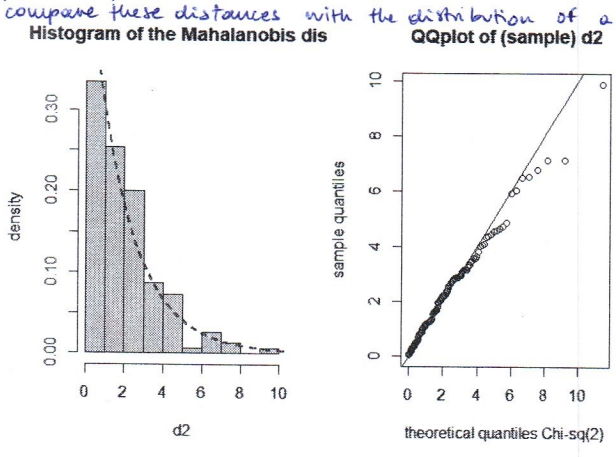



```
d2 <- mahalanobis(X, colMeans(X), cov(X))

x11(width=13)
par(mfrow=c(1,2))
hist(d2, prob=T, main='Histogram of the Mahalanobis dist.',xlab='d2',ylab='density', col='grey84')
lines(0:2000/100, dchisq(0:2000/100,2), col='blue', lty=2, lwd=2)
qqplot(qchisq((1:n - 0.5)/n, df = 2), d2, main='QQplot of (sample) d2',xlab='theoretical quantiles Chi-sq(2)',
       ylab='sample quantiles')
abline(0, 1)
```

MAHALANOBIS' DISTANCE

Graphical exploration:
We compare these distances with the distribution of a χ^2:

- histogram
- QQplot




```
### we can perform a chi.sq goodness-of-fit test

d2.class <- cut(d2, qchisq((0:10)/10, df = 2))
d2.freq  <- table(d2.class)

chisq.test(x = d2.freq, p = rep(1/10, 10), simulate.p.value = T)
```

```
##
##  Chi-squared test for given probabilities with simulated p-value (based
##  on 2000 replicates)
##
## data:  d2.freq
## X-squared = 11.733, df = NA, p-value = 0.2514
```

✓

H_0: data $\sim \chi^2$

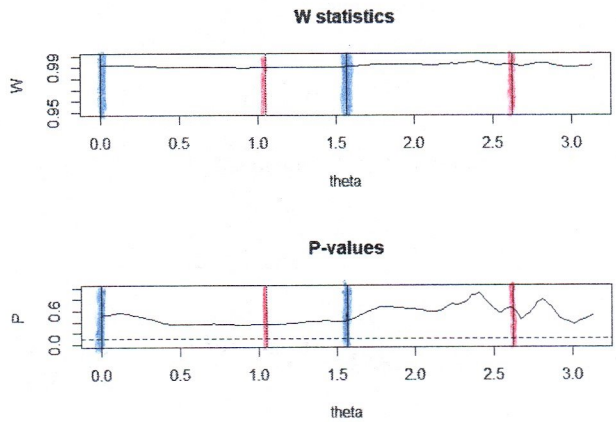
```
# Test: does the population probabilities (given in x) equal those in p?

# Remark: since the mean and covariance matrix are unknown, we can only
#     have approximate solutions:
#     the Mahalanobis distance is computed with estimates of the mean vector
#     and of the covariance matrix; the sample of distances is not iid

graphics.off()
```

3.

(?)

```
### ----------------------------------------------------------------------------
### Approach 3: test of all the directions simultaneously, by looking at the min
### of the Shapiro-Wilk statistics

# We reject H0: X ~ N if we observe a "low" value of W along at least
# one direction, i.e., if the minimum of W along the direction is "low"

# Looking at all directions, is equivalent to looking at min(W) along
# the directions (test statistic)

# Example with our simulated data: we compute the W statistics for all
# the directions.
theta   <- seq(0, pi - pi/180, by = pi/180)
W       <- NULL
P       <- NULL
for(i in 1:length(theta))
{
  a   <- c(cos(theta[i]), sin(theta[i]))
  w   <- shapiro.test(X %*% a)$statistic
  p   <- shapiro.test(X %*% a)$p.value
  W   <- c(W, w)
  P   <- c(P, p)
}

x11()
par(mfrow = c(2,1))
plot(theta, W, main = 'W statistics', ylim = c(0.95,1), type='l')
abline(v=c(0, pi/2), col = 'blue')
abline(v= atan(princomp(X)$loadings[2,]/princomp(X)$loadings[1,]), col='red')
abline(v= atan(princomp(X)$loadings[2,]/princomp(X)$loadings[1,]) + pi, col='red')

plot(theta, P, main = 'P-values', ylim = c(0,1), type='l')
abline(v=c(0, pi/2), col = 'blue')
abline(h=0.10, col = 'blue', lty = 2) # set alpha=10%
abline(v= atan(princomp(X)$loadings[2,]/princomp(X)$loadings[1,]), col='red')
abline(v= atan(princomp(X)$loadings[2,]/princomp(X)$loadings[1,]) + pi, col='red')
```

← we define a vector θ which spans all the possible angles, we project data on the direction defined by θ and we calculate the p-value of the shapiro test



the vertical lines are those we were looking at with the first approach (axes (x,y) PC1, PC2)

(?)

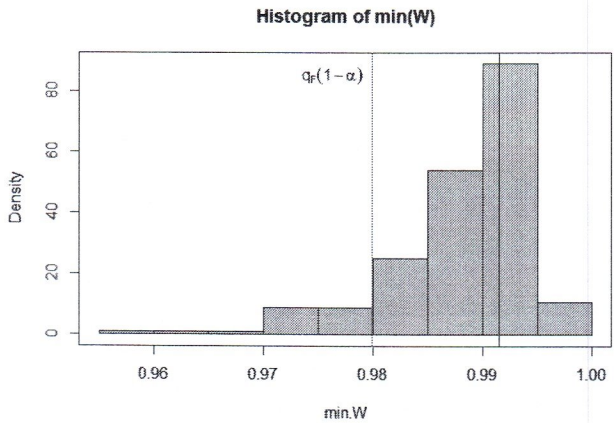
```
# Note: to set the rejection region, we should look at the distribution of
# of min(W) under H0 [not the distribution of W.a for all the directions a]
# --> To see how much we reject globally if we set a threshold
#     alpha=10% at the univariate tests based on W.a, see Experiment.R

# Hence, to build the rejection region for the test,
# we just need to set the threshold as the quantile of order alpha
# of the distribution of min(W) under H0

# Formally:
# H0: X ~ N vs H1: H0^c
# test statistics: min(W) ~ F under H0
# Reject H0 if min(W)<qF(alpha),
# with qF(alpha) s.t. P(min(W) < qF(alpha)|H0) = alpha

# The distribution F of min(W) is not known, but it can be approximated with
# a Monte Carlo method. That is, we approximate the distribution of min(W) with a
# histogram generated by simulating Gaussian samples. The quantile qF(alpha) is
# estimated with the sample quantile of order alpha from the samples.

# Note: an explicit expression is available for min(W). It is computed
# by the function mshapiro.test

# Example: for 200 sample we can compute min(W) and look at its distribution
min.W=NULL
for(i in 1:200){
  Y <- rmvnorm(n, mu, sig)
  min.W <- c(min.W, mshapiro.test(t(Y))$stat)
}
x11()
hist(min.W, prob=T, col='grey81', main='Histogram of min(W)'); box()
abline(v=quantile(min.W, probs = .1), col=2)
text(quantile(min.W, probs = .1), 85, labels = expression(q[F](1-alpha)),
     pos=2)

# => I'll reject H0 if min(W)<qF(alpha), with qF(alpha)~.98
quantile(min.W, probs = .1)
```

```
##       10%
## 0.9799275
```

```
# Actual observation of min(W)
min.W0 = mshapiro.test(t(X))$stat
min.W0 # accept H0
```

```
##         W
## 0.9915438
```

```
abline(v=min.W0, col='blue')
```




```
# Approximate the p-value with Monte Carlo: count how many realizations under H0
# are associated with a min(W) lower than the actual observation

# Proportion of the realization that has min(W) lower than min.W0
sum(min.W < min.W0)/200
```

```
## [1] 0.625
```

```
# Very high p-value => accept H0

### The function mcshapiro.test implements a procedure that:
### 1- approximates the distribution of the statistics min(W) via MC
### 2- performs a test of normality based on the (approximate) distribution of min(W)
### 3- returns an approximate p-value of the test at point 2-
mcshapiro.test <- function(X, devstmax = 0.01, sim = ceiling(1/(4*devstmax^2)))
{
  library(mvnormtest)
  n   <- dim(X)[1]
  p   <- dim(X)[2]
  mu  <- rep(0,p)
  sig <- diag(p)
  W   <- NULL
  for(i in 1:sim)
  {
    Xsim <- rmvnorm(n, mu, sig)
    W   <- c(W, mshapiro.test(t(Xsim))$stat)
    # mshapiro.test(X): compute the statistics min(W) for the sample X
  }
  Wmin   <- mshapiro.test(t(X))$stat   # min(W) for the given sample
  pvalue <- sum(W < Wmin)/sim          # proportion of min(W) more extreme than the observed Wmin
  devst  <- sqrt(pvalue*(1-pvalue)/sim)
  list(Wmin = as.vector(Wmin), pvalue = pvalue, devst = devst, sim = sim)
}

mcshapiro.test(X)
```

uncertainty acceptable

max number of montecarlo simulations

mcshapiro.test()

MONTECARLO MULTIVARIATE SHAPIRO TEST (automatically implements this)

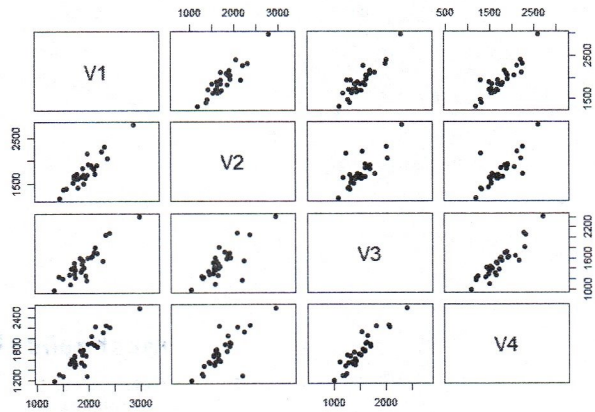
```
## $Wmin
## [1] 0.9915438
##
## $pvalue
## [1] 0.6188
##
## $devst
## [1] 0.009713631
##
## $sim
## [1] 2500
```

```
graphics.off()
```

```
### ----------------------------------------------------------------------------
# Example: test of normality for the dataset stiff
# Dataset: each sample unit is a board. For each board, four measures of
# stiffness are token (X1: sending a shock wave, X2: while vibrating the board,
# X3 and X4: static tests)

stiff <- read.table('stiff.dat')
stiff
```

```
##       V1   V2   V3   V4
## 1   1889 1651 1561 1778
## 2   2403 2048 2087 2197
## 3   2119 1700 1815 2222
## 4   1645 1627 1110 1533
## 5   1976 1916 1614 1883
## 6   1712 1712 1439 1546
## 7   1943 1685 1271 1671
## 8   2104 1820 1717 1874
## 9   2983 2794 2412 2581
## 10 1745 1600 1384 1508
## 11 1710 1591 1518 1667
## 12 2046 1907 1627 1898
## 13 1840 1841 1595 1741
## 14 1867 1685 1493 1678
## 15 1859 1649 1389 1714
## 16 1954 2149 1180 1281
## 17 1325 1170 1002 1176
## 18 1419 1371 1252 1308
## 19 1828 1634 1602 1755
## 20 1725 1594 1313 1646
## 21 2276 2189 1547 2111
## 22 1899 1614 1422 1477
## 23 1633 1513 1290 1516
## 24 2061 1867 1646 2037
## 25 1856 1493 1356 1533
## 26 1727 1412 1238 1469
## 27 2168 1896 1701 1834
## 28 1655 1675 1414 1597
## 29 2326 2301 2065 2234
## 30 1490 1382 1214 1284
```

```
x11()
plot(stiff, asp=1, pch=19)
```

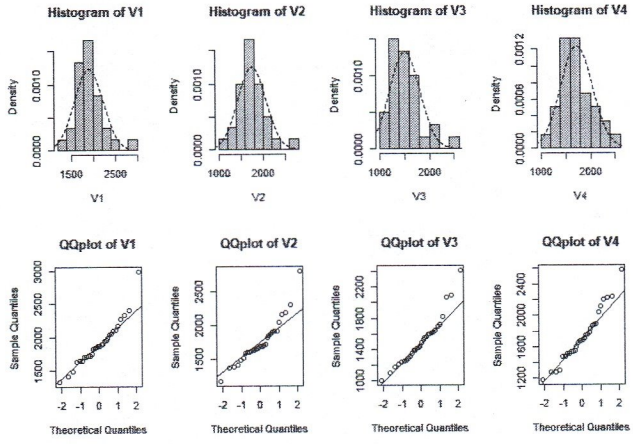



```
# Normality of the components
x11(width=12)
par(mfcol=c(2,4))

for(i in 1:4)
{
  hist(stiff[,i], prob=T, main=paste('Histogram of V', i, sep=''), xlab=paste('V', i, sep=''))
  lines(900:2800, dnorm(900:2800,mean(stiff[,i]),sd(stiff[,i])), col='blue', lty=2)
  qqnorm(stiff[,i], main=paste('QQplot of V', i, sep=''))
  qqline(stiff[,i])
  print(shapiro.test(stiff[,i])$p)
}
```

```
## [1] 0.05118027
```

```
## [1] 0.0174619
```

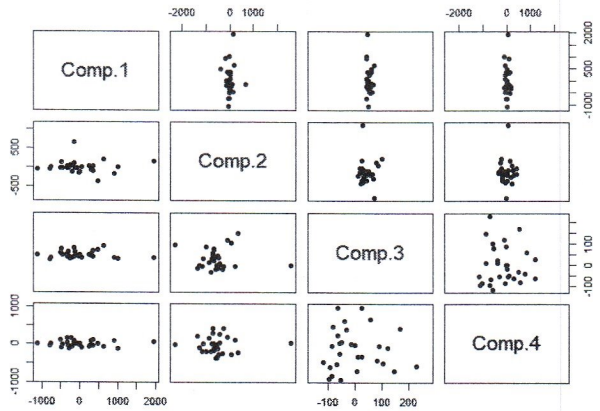
```
## [1] 0.05750619
```




```
## [1] 0.3337193
```

we have some doubts because of these 4 p-values

```
# Normality of the principal components
PCs <- data.frame(princomp(stiff)$scores)

x11()
plot(PCs, asp=1, pch=19)
```

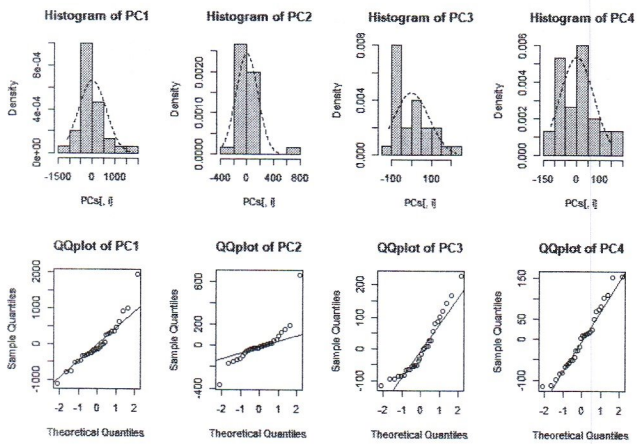



```
x11(width=13)
par(mfcol=c(2,4))
for(i in 1:4)
{
  hist(PCs[,i], prob=T, main=paste('Histogram of PC', i, sep=''))
  lines(seq(min(PCs[,i]), max(PCs[,i]), length=2000), dnorm(seq(min(PCs[,i]), max(PCs[,i]), length=2000),mean(PCs[,i]),sd(PC
s[,i])), col='blue', lty=2)
  qqnorm(PCs[,i], main=paste('QQplot of PC', i, sep=''))
  qqline(PCs[,i])
  print(shapiro.test(PCs[,i])$p)
}
```

```
## [1] 0.03982978
```

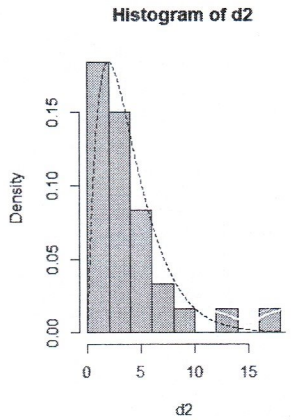
```
## [1] 4.088399e-05
```
← so bad! ⟹ the gaussianity is not satisfied

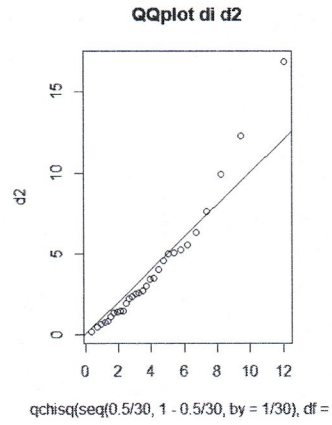
```
## [1] 0.02052519
```




```
## [1] 0.3345126
```
(Just because we want to try the methods (we already know the data are not gaussian) :)

```
# Mahalanobis distances of the data from the sample mean
M <- colMeans(stiff)
S <- cov(stiff)

d2 <- matrix(mahalanobis(stiff, M, S))

windows()
par(mfrow=c(1,2))

hist(d2, prob=T)
lines(0:2000/100, dchisq(0:2000/100,4), col='blue', lty=2)

qqplot(qchisq(seq(0.5/30, 1 - 0.5/30 ,by = 1/30), df = 4), d2,  main='QQplot di d2')
abline(0, 1)
```






```
d2.class <- cut(d2, qchisq((0:6)/6, df = 3))
d2.freq  <- table(d2.class)

chisq.test(x = d2.freq, p = rep(1/6, 6), simulate.p.value = T)
```

```
##
##  Chi-squared test for given probabilities with simulated p-value (based
##  on 2000 replicates)
##
## data:  d2.freq
## X-squared = 2.8, df = NA, p-value = 0.7601
```

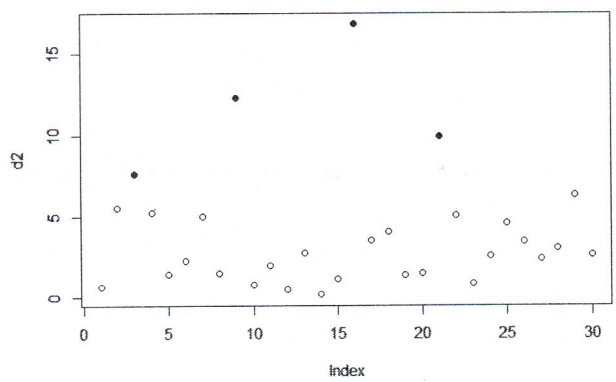
→ we don't have enough evidence to reject the null hypothesis (H_0: gaussianity)
(in fact H_0: χ^2 and this leads to gaussianity)

```
# test of all the directions simultaneously
mcshapiro.test(stiff)
```

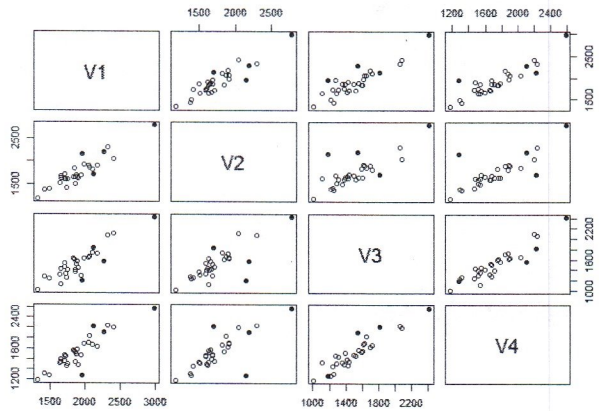
```
## $Wmin
## [1] 0.7534355
##
## $pvalue
## [1] 0.002
##
## $devst
## [1] 0.0008935323
##
## $sim
## [1] 2500
```

→ enough evidence to reject H_0 : gaussianity

```
### The data don't seem Gaussian. What can we do?
### Identify clusters
### Identify (and possibly remove) outliers
### Transform the data (e.g., Box-Cox transformations, see Johnson-Wichern Chap.4.8,
###                     R functions powerTransform(); bcPower())
### Work without the Gaussian assumption (e.g., permutation tests)

### Let's try to identify and remove outliers:
# We remove the data too far (in the sense of the Mahalanobis distance)
# from the center of the distribution

x11()
plot(d2, pch=ifelse(d2<7.5,1,19))
```




```
x11()
plot(stiff, pch=ifelse(d2<7.5,1,19))
```

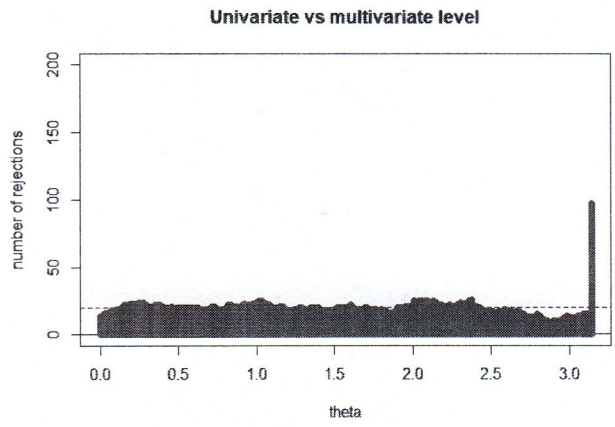



```
stiff.noout <- stiff[which(d2<7.5),]

mcshapiro.test(stiff.noout)
```

```
## $kWmin
## [1] 0.9505474
##
## $pvalue
## [1] 0.8664
##
## $devst
## [1] 0.006804441
##
## $sim
## [1] 2500
```

```
# In this case removing the outliers solves the problem of non-gaussianity
```

```
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
### Experiment
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
### Experiment: we repeat 200 times the sample generation (under H0) and the
### univariate tests along "all" (i.e., a fine grid of) the directions.
### We count how many univariate rejections and how many multivariate rejections we get.

library(mvtnorm)
mu <- c(1,2)
sig <- matrix(c(1,1,1,2), 2)
n   <-  150
theta   <- seq(0, pi - pi/180, by = pi/180)
Tot <- 0
Sin <- rep(0, 180)

x11()
plot(theta, rep(0, 180), pch = '', ylim = c(0,200), xlim = c(0, pi), ylab='number of rejections', main = 'Univariate vs mult
ivariate level')
abline(h = 20, lty = 2)
abline(h = 0, lty = 1)
points(theta, Sin, col = 'blue', pch = 16)
points(pi, Tot, col = 'red', pch = 16)

for(i in 1:200){
  Y <- rmvnorm(n, mu, sig)
  theta   <- seq(0, pi - pi/180, by = pi/180)
  W       <- NULL
  P       <- NULL
  for(i in 1:length(theta)){
    a   <- c(cos(theta[i]), sin(theta[i]))
    w   <- shapiro.test(Y %*% a)$statistic
    p   <- shapiro.test(Y %*% a)$p.value
    W   <- c(W, w)
    P   <- c(P, p)
  }
  Sin <- Sin + as.numeric(P < (0.10))
  Tot <- Tot + (sum(P < (0.10)) > 0)
  points(theta, Sin, col = 'blue', pch = 16)
  points(pi, Tot, col = 'red', pch = 16)
}
```




```
# To globally reject at 10% we should
# (a) decrease the level of the univariate tests (e.g., Bonferroni corrections)
# or
# (b) think multivariate
```