LAB 05

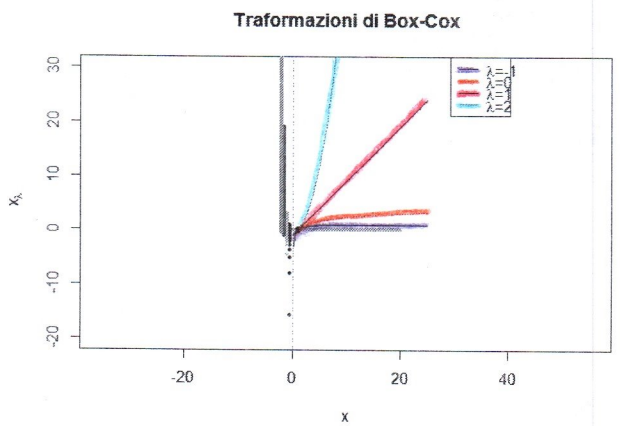
TOPIC:

- Box-Cox transformations
- Tests and confidence regions for the mean of a multivariate Gaussian
- Further material

```
library(car)
library(mvtnorm)
load("mcshapiro.test.RData")
```

```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Box-Cox transformation
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Univariate Box-Cox transformation (Johnson-Wichern Chap.4.8)
# The Box-Cox transformations are based on the family of power transformation
# (for x>0)
# x_lambda = (x^lambda-1)/lambda if lambda!=0
#            ln(x)              if lambda==0
# that is continuous in lambda for x>0.
# The parameter lambda is determined as the maximum likelihood solution
# under the Gaussian assumption (see J-W)
```

```
# Let's plot the transformations for some values of lambda
box_cox <- function(x,lambda){
  if(lambda!=0)
    return((x^lambda-1)/lambda)
  return(log(x))
}

x<-seq(0,25,by=0.01)

x11()
plot(x,box_cox(x,0),col='gold',type='l',xlab='x',ylab=expression(x[lambda]),ylim=c(-20,30),xlim=c(-5,25),asp=1)
title(main='Traformazioni di Box-Cox')
curve(box_cox(x,-1),from=0,to=25,col='blue',add=TRUE)
curve(box_cox(x,1),from=0,to=25,col='red',add=TRUE)
curve(box_cox(x,2),from=0,to=25,col='springgreen',add=TRUE)
points(1,0,pch=19,col='black')
abline(v=0,lty=2,col='grey')
legend('topright',c(expression(paste(lambda,'=-1')),expression(paste(lambda,'=0')),
                    expression(paste(lambda,'=1')),expression(paste(lambda,'=2'))),col=c('blue','gold','red','springgreen'),
lty=c(1,1,1,1,1))

xx=seq(0.01,20.01,.05)

par(cex=.5)
points(xx, rep(0,length(xx)), col='grey', pch=19)
points(-.5+rep(0,length(xx)),box_cox(xx,-1),  col='blue', pch=19)
points(-.5+rep(-.5,length(xx)),log(xx),  col='gold', pch=19)
points(-.5+rep(-1,length(xx)),box_cox(xx,1), col='red', pch=19)
points(-.5+rep(-1.5,length(xx)),box_cox(xx,2), col='springgreen', pch=19)
points(1,0,pch=19,col='black')
```

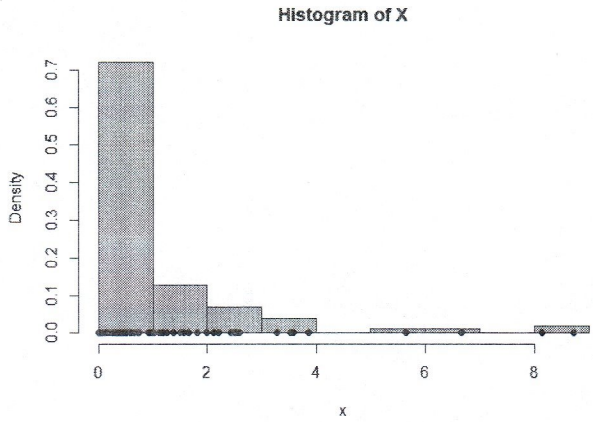


$\lambda = 0 \Rightarrow \log(x)$

Remember: these transformations can be applied only to positive data (we can change the sign in case)

```
# For lambda<1: observations <1 are "spread", observations >1 are "shrinked"
# For lambda>1: observations <1 are "shrinked", observations >1 are "spread"

graphics.off()
```

```
### ------------------------------------------------------------------------
### A trivial example
### ------------------------------------------------------------------------
n <- 100

set.seed(05042016)
x <- rnorm(n)^2

x11()
hist(x, col='grey', prob=T, xlab='x', main ='Histogram of X')
points(x,rep(0,n), pch=19)
```

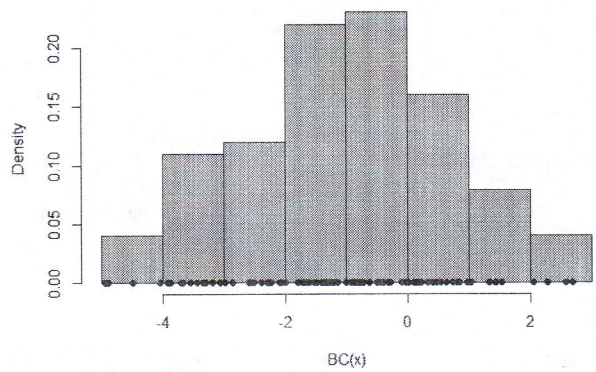



```
# we would need to spread the obs with small values and shrink the obs with large values
# => we expect a small lambda
```

```
# Univariate Box-Cox transformation
# We compute the optimal lambda of the univariate Box-Cox transformation (command powerTransform of library car)
lambda.x <- powerTransform(x)
lambda.x
```

```
## Estimated transformation parameter
##         x
## 0.1918285
```

```
# lambda<1: observations <1 are "spread", observations >1 are "shrinked"

# Transformed sample with the optimal lambda (command bcPower of library car)
bc.x <- bcPower(x, lambda.x$lambda)      # it transforms the data of the first argument through the Box-Cox transformation w
ith lambda given as second argument
hist(bc.x, col='grey', prob=T, main='Histogram of BC(X)', xlab='BC(x)')
points(bc.x,rep(0,n), pch=19)
```




```
shapiro.test(x)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  x
## W = 0.61593, p-value = 8.728e-15
```

✘

```
shapiro.test(bc.x)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  bc.x
## W = 0.99143, p-value = 0.7784
```

✓

```
# Multivariate Box-Cox transformation (Johnson-Wichern Cap.4.8)
# Similar to univariate transformation, but jointly on all the variables
rm(x)
```

```
### ------------------------------------------------------------------------
### Simulated Example
### ------------------------------------------------------------------------
b <- read.table('data_sim.txt')
head(b)
```

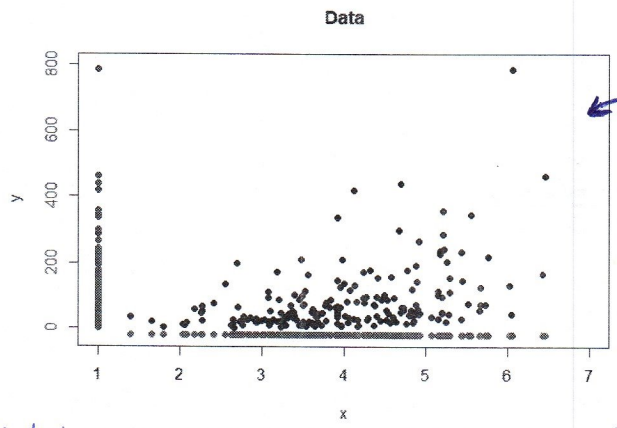
: bivariate distribution simulation

```
##          x         y
## 1 5.446189 146.53762
## 2 2.084034  18.09554
## 3 4.238034  95.42054
## 4 4.690763 439.80116
## 5 4.042480  82.17827
## 6 3.573651  22.85204
```

```
dim(b)
```

```
## [1] 200   2
```

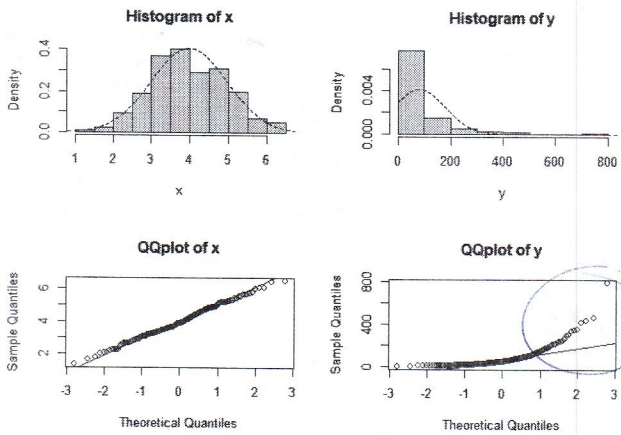
```
attach(b)

x11()
plot(b,pch=19,main='Data', xlim=c(1,7), ylim=c(-20,800))
points(x, rep(-20,dim(b)[1]), col='red', pch=19)
points(rep(1,dim(b)[1]), y, col='blue', pch=19)
```



the cloud of points doesn't look gaussian and looking at the projections we see that the problem may be on the y-axes.

We check our hypothesis on what could be the problem (y) we study the projections on the axes:

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

hist(x, prob=T,col='grey85')
lines(0:1000 / 100, dnorm(0:1000 / 100,mean(x),sd(x)), col='blue', lty=2)

hist(y, prob=T,col='grey85')
lines(0:1000, dnorm(0:1000,mean(y),sd(y)), col='blue', lty=2)

qqnorm(x, main='QQplot of x')
qqline(x)

qqnorm(y, main='QQplot of y')
qqline(y)
```



strong evidence that data are not gaussian

```
# For y: left tail too light, right tail too heavy

shapiro.test(x)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  x
## W = 0.99443, p-value = 0.6653
```

✓

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

```
##
##  Shapiro-Wilk normality test
##
## data:  y
## W = 0.67337, p-value < 2.2e-16
```

✗

```
graphics.off()

mcshapiro.test(b)
```

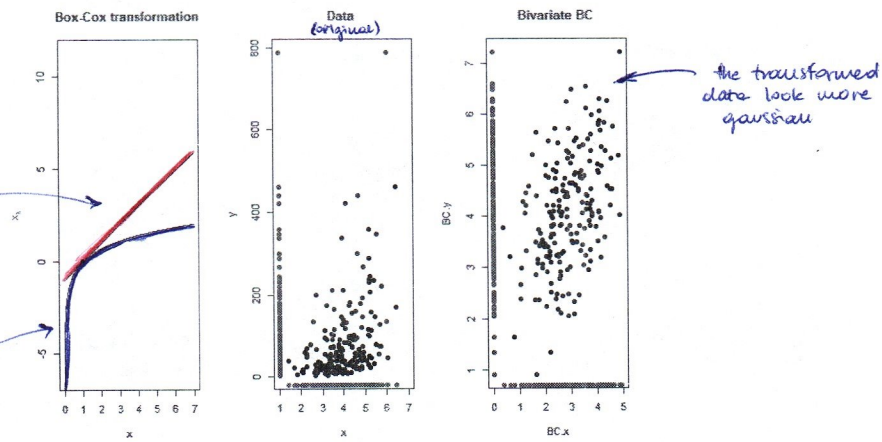
```
## $Wmin
## [1] 0.674787
##
## $pvalue
## [1] 0
##
## $devst
## [1] 0
##
## $sim
## [1] 2500
```

(on the same example)

```
###
### Bivariate Box-Cox transformation
###
# Compute the optimal Lambda of a bivariate Box-Cox transformation
# (command powerTransform, with a multivariate input)
lambda <- powerTransform(cbind(x,y))
lambda
```

```
## Estimated transformation parameters
##         x         y
## 0.97717477 0.02386159
```

```
# lambda[1] close to 1
# lambda[2] close to 0, it is almost a log transform;
# lambda[2]<1:  obs <1 "spred", obs >1 "shrinked"
# => it adds weight to the left tail, lightens the right tail

# Compute the transformed data with optimal Lambda (of the bivariate transf.)
# (command bcPower)
BC.x <- bcPower(x, lambda$lambda[1])
BC.y <- bcPower(y, lambda$lambda[2])

xx <- seq(0,7,by=0.01)

x11(width=21, height=7)
par(mfrow=c(1,3))

plot(xx,box_cox(x=xx,lambda=lambda$lambda[1]),col='red',lty=1,type='l',xlab=expression(x),ylab=expression(x[lambda]),ylim=c
(-5,10),asp=1)
title(main='Box-Cox transformation')
lines(xx,box_cox(x=xx,lambda=lambda$lambda[2]),col='blue')
points(1,0,pch=19,col='black')
abline(a=-1,b=1,lty=2,col='grey')
legend('bottomright',c(expression(lambda[x]),expression(lambda[y]),expression(paste(lambda,'=1'))),col=c('blue','red','grey'
),lty=c(1,1,1))

plot(b,pch=19,main='Data',xlim=c(1,7))
points(rep(1,200),y,pch=19,col='blue') # projection on y
points(x,rep(-20,200),pch=19,col='red')  # projection on x

plot(BC.x,BC.y,pch=19,main='Bivariate BC',xlim=c(0,5))
points(rep(0,200),BC.y,pch=19,col='blue') # projection on y
points(BC.x,rep(0.7,200),pch=19,col='red')  # projection on x
```



boxcox trans. applied at x (almost a straight line since $\lambda \simeq 1$)

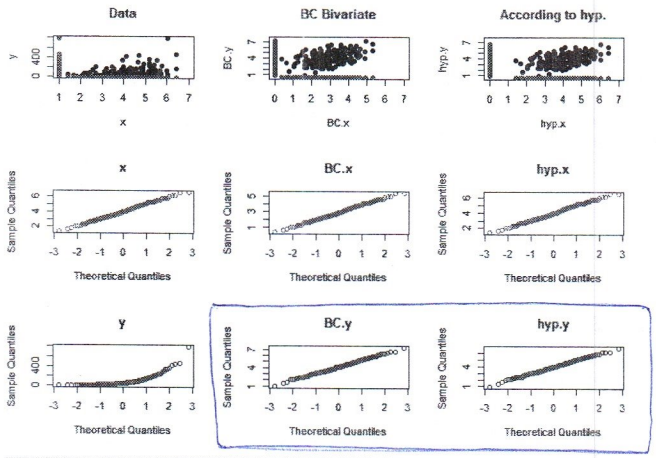
boxcox trans. applied at y ($\lambda \simeq 0.239$)

the transformed data look more gaussian

```
graphics.off()
```

Remember anyway that gaussianity must be guaranteed on any possible direction, not only on x and y, in order to have a 2-dimensional gaussian distribution

```
# Let's formulate an hypothesis of transformation:
# since we get lambda[1]~1 and lambda[2]~0, we could reasonably consider:
hyp.x <- x
hyp.y <- log(y)

# Remark: from the application-oriented viewpoint, explaining a log
# transform could be simpler than introducing Box-Cox transformations

x11(width=14, height=10)
par(mfrow=c(3,3))
plot(b,pch=19,main='Data',xlim=c(1,7))
points(rep(1,200),y,pch=19,col='blue')   # projection on y
points(x,rep(-20,200),pch=19,col='red')  # projection on x

plot(BC.x,BC.y,pch=19,main='BC Bivariate',xlim=c(0,7),ylim=c(0.5,7.5))
points(rep(0,200),BC.y,pch=19,col='blue')   # projection on y
points(BC.x,rep(0.5,200),pch=19,col='red')  # projection on x

plot(hyp.x,hyp.y,pch=19,main='According to hyp.',xlim=c(0,7),ylim=c(0.5,7.5))
points(rep(0,200),hyp.y,pch=19,col='blue')  # projection on y
points(hyp.x,rep(0.5,200),pch=19,col='red') # projection on x

qqnorm(x, main="x",col='red')
qqnorm(BC.x, main="BC.x",col='red')
qqnorm(hyp.x, main="hyp.x",col='red')

qqnorm(y, main="y",col='blue')
qqnorm(BC.y, main="BC.y",col='blue')
qqnorm(hyp.y, main="hyp.y",col='blue')
```



both $\lambda = 0.02$ and $\lambda = 0$ works well for y

```
dev.off()

# Univariate Shapiro-Wilk (H0: univariate gaussianity along a given direction)
shapiro.test(x)$p
```

```
## [1] 0.6652598
```

✓

```
shapiro.test(BC.x)$p
```

```
## [1] 0.6871995
```

✓

```
shapiro.test(hyp.x)$p
```

```
## [1] 0.6652598
```

✓

```
shapiro.test(y)$p
```

```
## [1] 1.916013e-19
```

✗

```
shapiro.test(BC.y)$p
```

```
## [1] 0.9912947
```

✓

```
shapiro.test(hyp.y)$p
```

```
## [1] 0.9939556
```

✓

```
# MC-Shapiro test (H0: multivariate gaussianity)
mcshapiro.test(cbind(x,y))$p
```

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

✗

```
mcshapiro.test(cbind(BC.x,BC.y))$p
```

```
## [1] 0.8972
```

✓

```
mcshapiro.test(cbind(hyp.x,hyp.y))$p
```

```
## [1] 0.9964
```

✓

mcshapiro.test is based on a Monte Carlo procedure so it's based on realizations (simulations), it may seem that the second is better than the first, but it's not 100% like that. Moreover, the goal is to obtain gaussianity, not the heigher p-value.

Remember: in this case we were lucky because we just had to fix one component (we worked on the marginal distributions) but having K gaussian variables doesn't guarantee to have a K-dimensional gaussian distribution!

```
# High p-value for the transformed data: with the obtained transformation we
# have no evidence of non-gaussianity, i.e., we can assume that transformed
# data are Gaussian

detach(b)

graphics.off()
```

```
### ------------------------------------------------------------------------
### Box-Cox transformation on the dataset stiff
### ------------------------------------------------------------------------
stiff <- read.table('stiff.dat')
head(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
```

→ these data are not gaussian (we already saw it)

```
dim(stiff)
```

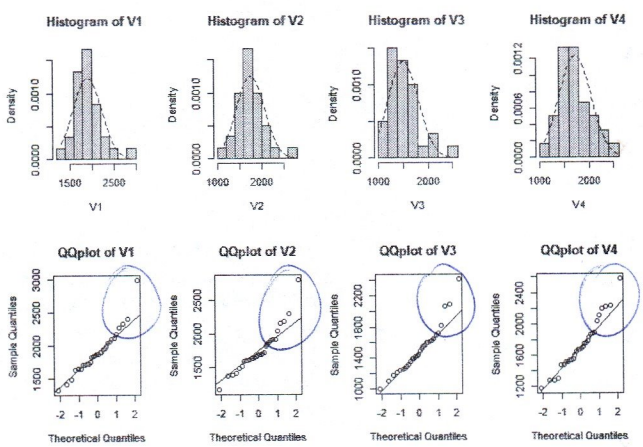
```
## [1] 30  4
```

```
lambda.mult <- powerTransform(stiff)
lambda.mult
```

```
## Estimated transformation parameters
##          V1          V2          V3          V4
##  0.09437843 -0.27947125  0.14835794  0.75472854
```

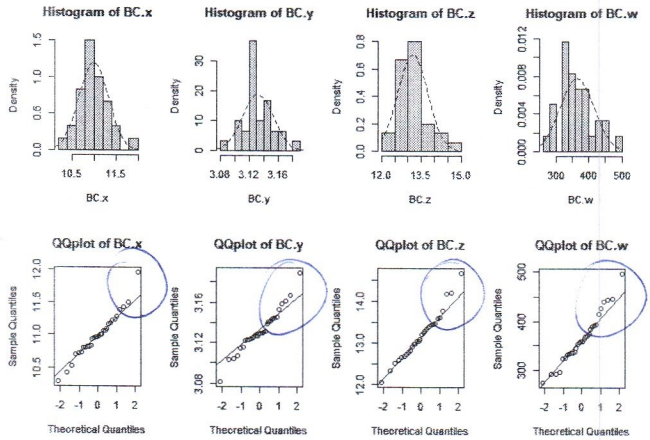
```
BC.x <- bcPower(stiff[,1],  lambda.mult$lambda[1])
BC.y <- bcPower(stiff[,2],  lambda.mult$lambda[2])
BC.z <- bcPower(stiff[,3],  lambda.mult$lambda[3])
BC.w <- bcPower(stiff[,4],  lambda.mult$lambda[4])
```

```
# Plot of the original variables
attach(stiff)
x11(width=14, height=7)
par(mfrow=c(2,4))
xx<-seq(1320,3000,length=100)
hist(V1, prob=T,breaks=8,col='grey85')
lines(xx, dnorm(xx,mean(V1),sd(V1)), col='blue', lty=2)
yy<-seq(1150,2800,length=100)
hist(V2, prob=T,breaks=8,col='grey85')
lines(yy, dnorm(yy,mean(V2),sd(V2)), col='blue', lty=2)
zz<-seq(1000,2420,length=100)
hist(V3, prob=T,breaks=8,col='grey85')
lines(zz, dnorm(zz,mean(V3),sd(V3)), col='blue', lty=2)
ww<-seq(1100,2600,length=100)
hist(V4, prob=T,breaks=8,col='grey85')
lines(ww, dnorm(ww,mean(V4),sd(V4)), col='blue', lty=2)

qqnorm(V1, main='QQplot of V1')
qqline(V1)

qqnorm(V2, main='QQplot of V2')
qqline(V2)

qqnorm(V3, main='QQplot of V3')
qqline(V3)

qqnorm(V4, main='QQplot of V4')
qqline(V4)
```




```
detach(stiff)
```

```
# Plot of the transformed variables
x11(width=14, height=7)
par(mfrow=c(2,4))
xx<-seq(10,12,length=100)
hist(BC.x, prob=T,breaks=8,col='grey85')
lines(xx, dnorm(xx,mean(BC.x),sd(BC.x)), col='blue', lty=2)
yy<-seq(3,3.2,length=100)
hist(BC.y, prob=T,breaks=8,col='grey85')
lines(yy, dnorm(yy,mean(BC.y),sd(BC.y)), col='blue', lty=2)
zz<-seq(12,15,length=100)
hist(BC.z, prob=T,breaks=8,col='grey85')
lines(zz, dnorm(zz,mean(BC.z),sd(BC.z)), col='blue', lty=2)
ww<-seq(250,500,length=100)
hist(BC.w, prob=T,breaks=8,col='grey85')
lines(ww, dnorm(ww,mean(BC.w),sd(BC.w)), col='blue', lty=2)

qqnorm(BC.x, main='QQplot of BC.x')
qqline(BC.x)

qqnorm(BC.y, main='QQplot of BC.y')
qqline(BC.y)

qqnorm(BC.z, main='QQplot of BC.z')
qqline(BC.z)

qqnorm(BC.w, main='QQplot of BC.w')
qqline(BC.w)
```



the result is still not completely satisfing (but already better)

```
# One transformed variable at a time
shapiro.test(BC.x)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  BC.x
## W = 0.97172, p-value = 0.5874
```

✓

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

```
##
##  Shapiro-Wilk normality test
##
## data:  BC.y
## W = 0.96952, p-value = 0.5263
```

✓

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

```
##
##  Shapiro-Wilk normality test
##
## data:  BC.z
## W = 0.97475, p-value = 0.6753
```

✓

```
shapiro.test(BC.w)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  BC.w
## W = 0.96898, p-value = 0.5117
```

✓

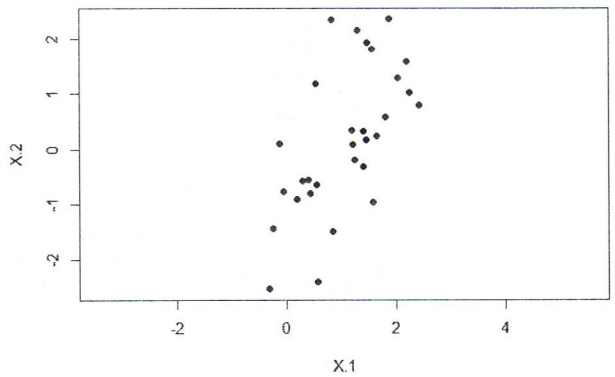
```
# All together
mcshapiro.test(cbind(BC.x,BC.y,BC.z,BC.w))
```

```
## $Wmin
## [1] 0.7306063
##
## $pvalue
## [1] 0
##
## $devst
## [1] 0
##
## $sim
## [1] 2500
```

✗ ← it's not always the case that we can just adjust the marginal distributions to get a gaussian distribution

```r
# Note: the higher the dimensionality, the more difficult to
# recover the gaussianity

# In this case, Box-Cox transformations are not enough. We saw that removing
# outliers indeed solve the problems of non-Gaussianity.
# Alternative approaches:
# - Asymptotics
# - Non-parametrics (e.g., permutation tests)

graphics.off()
```

```r
### ----------------------------------------------------------------------
### ----------------------------------------------------------------------
### Test and confidence regions for the mean of a multivariate Gaussian
### ----------------------------------------------------------------------
### ----------------------------------------------------------------------
### Example with simulated data from a bivariate Gaussian
### ----------------------------------------------------------------------
mu <- c(1,0)
sig <- matrix(c(1,1,1,2),nrow=2)

set.seed(123)
x <- rmvnorm(n=30, mean=mu, sigma=sig)
x <- data.frame(X.1=x[,1],X.2=x[,2])

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




```r
dev.off()
```

```r
n <- dim(x)[1]
p <- dim(x)[2]

x.mean   <- sapply(x,mean)
x.cov    <- cov(x)
x.invcov <- solve(x.cov)
```

```r
### ----------------------------------------------------------------------
### Test for the mean of a multivariate Gaussian
### ----------------------------------------------------------------------
# Premiss: general rule to perform a test
# 1)  Formulate the test (and test the Gaussian assumption, if needed)
# 2)  Compute the test statistics
# 3a) Having set the level of the test, verify whether the test statistics
#     belongs to the region of rejection (i.e., if there is statistical
#     evidence to reject H0)
# 3b) Compute the p-value of the test
### ----------------------------------------------------------------------
### Test on the mean of level alpha=1%
### H0: mu == mu0 vs H1: mu != mu0
### with mu0=c(1,0)
### ----------------------------------------------------------------------
mcshapiro.test(x)
```

$H_0: \mu = \mu_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ ← this is the mean we used to generate the data

```
## $Wmin
## [1] 0.963085
##
## $pvalue
## [1] 0.5272
##
## $devst
## [1] 0.009985192
##
## $sim
## [1] 2500
```

✓

```r
alpha <- 0.01
mu0 <- c(1,0)

# T2 Statistics
x.T2       <- n * (x.mean-mu0) %*% x.invcov %*% (x.mean-mu0)
# Radius of the ellipsoid
cfr.fisher <- ((n-1)*p/(n-p))*qf(1-alpha,p,n-p)
# Test:
x.T2 < cfr.fisher   # no statistical evidence to reject H0 at level alpha
```

Fisher quantile

we want to see if our test statistic lies inside or outside the rejection region

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

⟹ we do not reject H_0

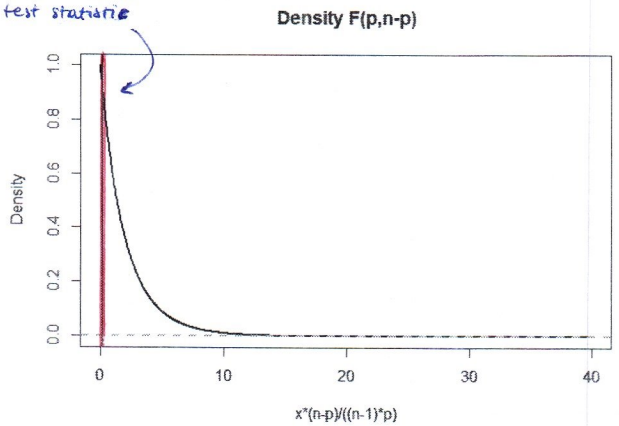
BE CAREFUL ON α AND $1-\alpha$

```
# Rejection region: {x.T2>cfr.fisher}
# (we reject for large values of the T2 statistics)

# Compute the p-value
P <- 1-pf(x.T2*(n-p)/((n-1)*p), p, n-p)
P
```

```
##           [,1]
## [1,] 0.8168201
```

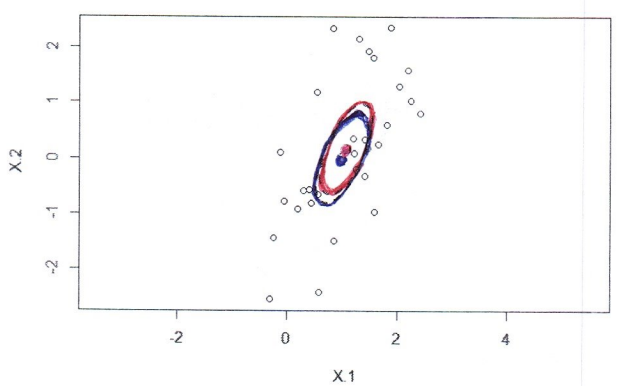
```
x11()
xx <- seq(0,40,by=0.05)
plot(xx,df(xx*(n-p)/((n-1)*p),p,n-p),type="l",lwd=2,main='Density F(p,n-p)',xlab='x*(n-p)/((n-1)*p)',ylab='Density')
abline(h=0,v=x.T2*(n-p)/((n-1)*p),col=c('grey','red'),lwd=2,lty=c(2,1))
```

x value of our test statistic




```
# The P-value is high because the test statistics is central with respect to
# its distribution under H0
# => we cannot reject for any reasonable level (we would reject for a level alpha>81.7%)

dev.off()
```

```
# Region of rejection (centred in mu0)
x11()
plot(x, asp = 1)
ellipse(mu0, shape=x.cov/n, sqrt(cfr.fisher), col = 'blue', lty = 2, center.pch = 16)

# We add a red point in correspondence of the sample mean
points(x.mean[1], x.mean[2], pch = 16, col ='red', cex = 1.5)

# Confidence region (centred in x.mean)
# { m \in R^2 s.t. n * (x.mean-m)' %*% (x.cov)^-1 %*% (x.mean-m) < cfr.fisher }

ellipse(x.mean, x.cov/n, sqrt(cfr.fisher), col = 'red', lty = 2, lwd=2, center.cex=1)
```



○ ⇒ defines the rejection region ($RR = O^c$)

○ ⇒ confidence region (same shape as the rejection region but centered in the sample mean)

• = μ_0

• = sample μ (sample mean)

everything that lays out the blue ellips is in the rejection region

⇒ if we have a test statistics that lays outside the blue ellipse we get to the conclusion that we reject H_0

```
# Remark: the radius and the shape of the ellipse are the same, but the centre changes:
# - Rejection region: the centre is the mean mu0 under H0 (blue ellipse)
# - Confidence region: the centre is the sample mean (red ellipse)

# Which relation between the two ellipses?
# - If the rejection region does NOT contain the sample mean (i.e., we
#   are in the acceptance region), then we cannot reject H0
#   (i.e., if the sample mean falls within the ellipse we accept H0)
# - If the mean under H0 (mu0) is contained in the confidence region
#   of level 1-alpha, then we do not reject H0 at level alpha
# => the confidence region of level 1-alpha contains all the mu0
#    that we would accept at level alpha

# Note: by definition, the confidence region of level 1-alpha
# produces ellipsoidal regions that contain the true mean
# 100(1-alpha)% of the times. If H0 is true (i.e., mu0 is
# the true mean), those ellipsoidal regions will contain mu0
# 100(1-alpha)% of the times
```

```
### ------------------------------------------------------------------------------
### Asymptotic test on the mean
### H0: mu == mu0 vs H1: mu != mu0
### with mu0=c(1,0)
### ------------------------------------------------------------------------------
# Note: we don't need to verify the Gaussian assumption!
# Warning: we are going to use an asymptotic test, but we only have n = 30 data!
```

(in practice we should use more data)

```
mu0   <- c(1,0)
x.T2A   <- n * (x.mean-mu0) %*%  x.invcov  %*% (x.mean-mu0)
cfr.chisq <- qchisq(1-alpha,p)
x.T2A < cfr.chisq # no statistical evidence to reject H0 at level alpha
```

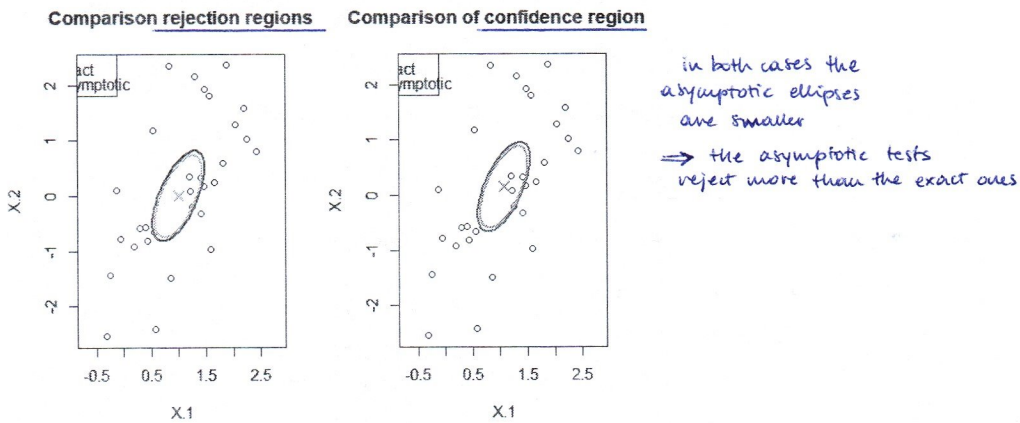
```
##      [,1]
## [1,] TRUE
```

→ again we do not reject H_0

```
# Compute the p-value
PA <- 1-pchisq(x.T2A, p)
PA
```

```
##           [,1]
## [1,] 0.8097058
```

```
x11(width=14, height=7)
par(mfrow=c(1,2))
plot(x, asp = 1,main='Comparison rejection regions')
ellipse(mu0, shape=x.cov/n, sqrt(cfr.fisher), col = 'blue', lty = 1, center.pch = 4, center.cex=1.5, lwd=2)
ellipse(mu0, x.cov/n, sqrt(cfr.chisq), col = 'lightblue', lty = 1, center.pch = 4, center.cex=1.5, lwd=2)
points(mu0[1], mu0[2], pch = 4, cex = 1.5, lwd = 2, col ='lightblue')
legend('topleft', c('Exact', 'Asymptotic'),col=c('blue','lightblue'),lty=c(1),lwd=2)

plot(x, asp = 1,main='Comparison of confidence regions')
ellipse(x.mean, x.cov/n, sqrt(cfr.fisher), col = 'red', lty = 1, center.pch = 4, center.cex=1.5, lwd=2)
ellipse(x.mean, x.cov/n, sqrt(cfr.chisq), col = 'orange', lty = 1, center.pch = 4, center.cex=1.5, lwd=2)
points(x.mean[1], x.mean[2], pch = 4, cex = 1.5, lwd = 2, col ='orange')
legend('topleft', c('Exact', 'Asymptotic'),col=c('red','orange'),lty=c(1),lwd=2)
```



in both cases the asymptotic ellipses are smaller

⇒ the asymptotic tests reject more than the exact ones

```
graphics.off()
```

```
### ------------------------------------------------------------------------------
###  We change the null Hypothesis
### ------------------------------------------------------------------------------
mu0 <- c(1.5,-0.5)
mu0
```

$H_0: \mu = \mu_0 = \begin{bmatrix} 1.5 \\ -0.5 \end{bmatrix}$

```
## [1]  1.5 -0.5
```

```
mu
```

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

```
# Remark: now H0 is false (we will want to reject H0)

### Test of level 1%
### H0: mu == mu0=c(1.5,-0.5) vs H1: mu != mu0=c(1.5,-0.5)
### ------------------------------------------------------------------------------
### Ellipsoidal region
x.T2 <- n * t(x.mean-mu0) %*% x.invcov %*% (x.mean-mu0)
x.T2 < cfr.fisher
```

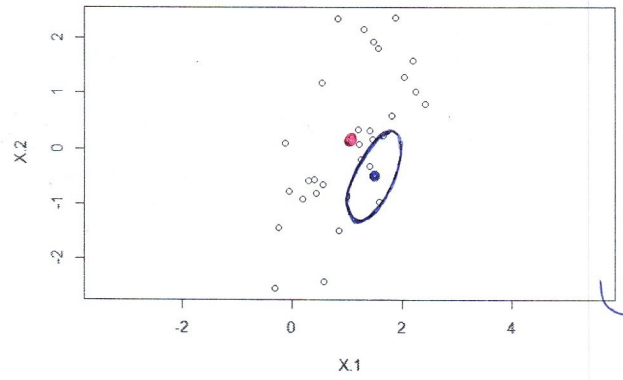
```
##      [,1]
## [1,] FALSE
```

→ we reject H_0

```
# Compute the p-value
P <- 1-pf(x.T2*(n-p)/((n-1)*p), p, n-p)
P
```

```
##              [,1]
## [1,] 2.678471e-06
```

```
x11()
plot(x, asp = 1)
ellipse(mu0, shape=x.cov/n, sqrt(cfr.fisher), col = 'blue', lty = 2, center.pch = 16)
points(x.mean[1], x.mean[2], pch = 16, col = 'red', cex=1.5)
```



○ defines the border of the rejection region (rejection region = $\bigcirc^c$)

● = μ_0

● = sample mean

→ the sample mean is inside the rejection region ⟹ we reject H_0

```
dev.off()
```

```
# How would we communicate the results?
# For instance, we could provide confidence intervals along interesting
# directions (if they don't contain the mean under H0, it means that
# we reject the associated univariate test along that direction)
```

```
# Let's try with simultaneous T2 confidence intervals on the coordinate directions:
# Recall: these are projections of the ellipsoidal confidence region

T2 <- cbind(inf = x.mean - sqrt(cfr.fisher*diag(x.cov)/n),
            center = x.mean,
            sup = x.mean + sqrt(cfr.fisher*diag(x.cov)/n))
T2
```

```
##            inf    center       sup
## X.1  0.5901592 1.0695188 1.5488785
## X.2 -0.6568668 0.1544118 0.9656904
```

→ contains 1.5
→ contains −0.5

{ each one contains the projection of the mean

```
# Both the intervals contain the mean under H0
# (i.e., mu0 is contained in the rectangular region determined by
# the projection of the ellipsoid along the coordinate directions)

# Remark: this is not in contrast with the previous findings
# Rejecting the global T2-test means that we reject H0 along at least one
# direction, not necessarily along the coordinate direction
```

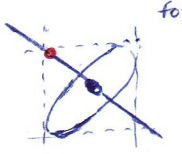
```
x11()
par(mfrow=c(1,1))
plot(x, asp = 1,main='Confidence and rejection regions')
ellipse(mu0, shape=x.cov/n, sqrt(cfr.fisher), col = 'blue', lty = 2, center.pch = 16)
points(x.mean[1], x.mean[2], pch = 16, col = 'red', cex=1.5)

ellipse(x.mean, shape=x.cov/n, sqrt(cfr.fisher), col = 'red', lty = 2, center.pch = 16)
rect(T2[1,1],T2[2,1],T2[1,3],T2[2,3], border='red', lwd=2)

# Let's try with Bonferroni intervals
k <- p # number of intervals I want to compute (set in advance)
cfr.t <- qt(1-alpha/(2*k),n-1)
Bf <- cbind(inf = x.mean - cfr.t*sqrt(diag(x.cov)/n),
            center = x.mean,
            sup = x.mean + cfr.t*sqrt(diag(x.cov)/n))
Bf
```

```
##            inf    center       sup
## X.1  0.6362019 1.0695188 1.5028358
## X.2 -0.5789432 0.1544118 0.8877668
```

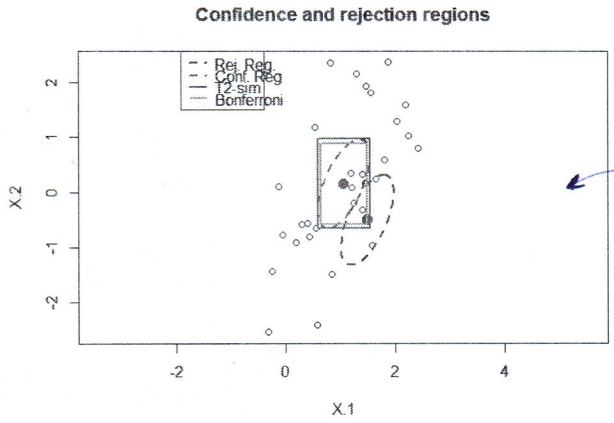
```
# Both the intervals contain the mean under H0
# (i.e., mu0 is contained in the rectangular region determined by
# the Bonferroni intervals along the coordinate directions)

# we add the Bonferroni intervals to the plot
rect(Bf[1,1],Bf[2,1],Bf[1,3],Bf[2,3], border='orange', lwd=2)
legend('topleft', c('Rej. Reg.', 'Conf. Reg','T2-sim', 'Bonferroni'),col=c('blue','red','red','orange'),lty=c(2,2,1,1),lwd=2
)
```

Our client is asking us to check if the mean of the population is $\mu_0 = \begin{bmatrix} 1.5 \\ -0.5 \end{bmatrix}$, we compute the rejection region (○) and we see that ● is in the rejection region ⟹ we conclude that the mean is not μ_0

Then the client asks us what is wrong: 1.5 or −0.5? We compute two separate confidence intervals and we see that both 1.5 and −0.5 are inside the CI's so we can't claim that the first component or second component is wrong, we can only say that there is a linear combination of the two that is wrong (because it doesn't fall in the ellipse)!



for example a projection on the —— direction is not containing the ●, even if a projection on x and a projection on y both contain ●



In this case Bonferroni's intervals lead to the same results than the singular (not corrected) intervals

```
# Remark: if we wanted to compute additional Bonferroni intervals
# along other directions, we would need to re-compute all the Bonferroni
# intervals with another correction k

graphics.off()
```

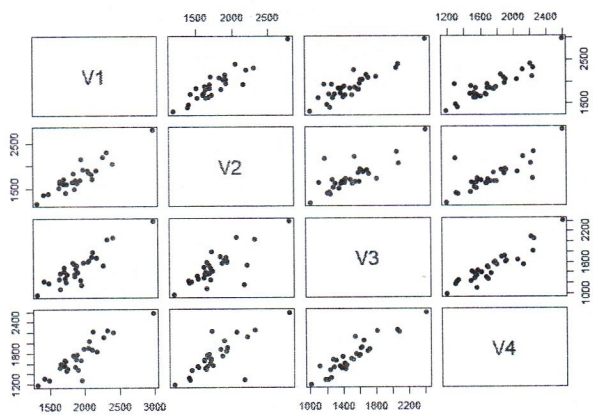
```
### ------------------------------------------------------------------------
### Example: analysis of a real dataset (dataset stiff)
### ------------------------------------------------------------------------
stiff <- read.table('stiff.dat')
head(stiff)
```

```
##      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
```

```
dim(stiff)
```

```
## [1] 30  4
```

```
n <- dim(stiff)[1]
p <- dim(stiff)[2]

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




```
dev.off()
```

```
### test of Gaussianity
mcshapiro.test(stiff)
```

```
## $Wmin
## [1] 0.7534355
##
## $pvalue
## [1] 0.0016
##
## $devst
## [1] 0.0007993597
##
## $sim
## [1] 2500
```

```r
# To recover Gaussianity, we remove 4 outliers (see LAB_4.R)
x.mean <- colMeans(stiff)
x.cov <- cov(stiff)
d2 <- matrix(mahalanobis(stiff, x.mean, x.cov))
stiff <- stiff[which(d2<7.5),]

mcshapiro.test(stiff)
```

```
## $Wmin
## [1] 0.9505474
##
## $pvalue
## [1] 0.8736
##
## $devst
## [1] 0.006645992
##
## $sim
## [1] 2500
```

✓

```r
n <- dim(stiff)[1]
p <- dim(stiff)[2]
```

```r
### Test for the mean of level 5%
### ------------------------------------------------------------------------
# 1) Formulate the test:
#    H0: mu == mu0 vs H1: mu != mu0
#    with mu0=c(1850, 1750, 1500, 1700)

mu0   <- c(1850, 1750, 1500, 1700)
alpha <- 0.05

# 2) Compute the test statistics
x.mean   <- colMeans(stiff)
x.cov    <- cov(stiff)
x.invcov <- solve(x.cov)

x.T2       <- n * (x.mean-mu0) %*% x.invcov %*% (x.mean-mu0)

# 3a) Verify if the test statistics belongs to the rejection region
cfr.fisher <- ((n-1)*p/(n-p))*qf(1-alpha,p,n-p)
x.T2 < cfr.fisher # we accept H0 at 5%
```

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

→ we accept H_0

```r
# 3b) Compute the p-value
P <- 1-pf(x.T2*(n-p)/((n-1)*p), p, n-p)
P
```

```
##           [,1]
## [1,] 0.0653894
```

```r
### ------------------------------------------------------------------------
### Confidence region for the mean of level 95%
### ------------------------------------------------------------------------
# 1)  Identify the type of region of interest: CR for the mean (ellipsoidal region)
#     { m \in R^4 t.c. n * (x.mean-m)' %*% x.invcov %*% (x.mean-m) < cfr.fisher }
# 2)  Characterize the region: compute the centre, direction of
#     the principal axes, length of the axes

# Centre:
x.mean
```

```
##       V1       V2       V3       V4
## 1840.423 1679.000 1473.846 1675.154
```

```r
# Directions of the principal axes:
eigen(x.cov/n)$vectors
```

```
##            [,1]       [,2]       [,3]       [,4]
## [1,] -0.5061946 -0.61649444  0.5544459 -0.2372579
## [2,] -0.4669544 -0.05759932 -0.7128900 -0.5200228
## [3,] -0.5062825  0.77807640  0.3533213 -0.1159277
## [4,] -0.5190362 -0.10589633 -0.2440115  0.8123089
```

```r
# Length of the semi-axes of the ellipse:
r <- sqrt(cfr.fisher)
r*sqrt(eigen(x.cov/n)$values)
```

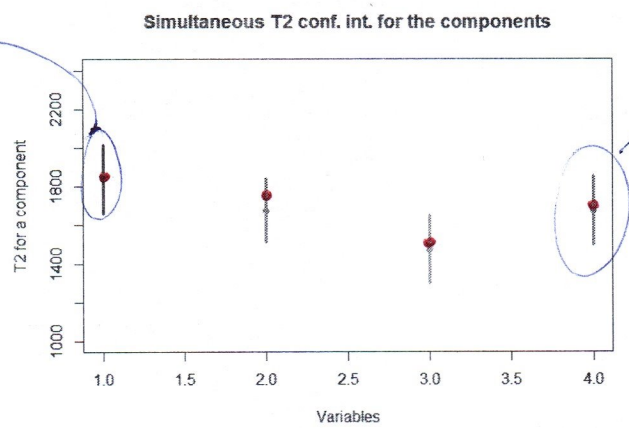
```
## [1] 338.09484  58.91961  55.27628  41.44214
```

```r
# Warning: Conf Reg => x.cov/n

# Question: how to plot the confidence region?
# (I don't know how to plot in R^4!)
# I can work with 'representations' of the confidence regions
# (e.g., projections along particular directions)

# We plot the projections of the ellipsoid in some directions
# of interest (e.g. the x and y coordinates)
# => We plot the simultaneous T2 confidence intervals in each
#    direction of interest (with global coverage alpha)

### Simultaneous T2 intervals on the components of the mean
### with global level 95%
### ------------------------------------------------------------------------
T2 <- cbind(inf = x.mean - sqrt(cfr.fisher*diag(x.cov)/n),
            center = x.mean,
            sup = x.mean + sqrt(cfr.fisher*diag(x.cov)/n))
T2
```

```
##         inf   center      sup
## V1 1662.533 1840.423 2018.313
## V2 1514.826 1679.000 1843.174
## V3 1295.504 1473.846 1652.188
## V4 1495.854 1675.154 1854.454
```

```
x11()
matplot(1:4,1:4,pch='',ylim=range(stiff),xlab='Variables',ylab='T2 for a component',
        main='Simultaneous T2 conf. int. for the components')
for(i in 1:4)segments(i,T2[i,1],i,T2[i,3],lwd=3,col=i)
points(1:4, T2[,2], pch=16, col=1:4)

# Is mu0 inside the rectangular region?
# We add it to the plot
points(1:4, mu0, lwd=3, col='orange')
```

projection of the 4-dimensional ellipse on the x_1 coordinate



projection of the 4-th dimensional ellipse on the x_4 coordinate

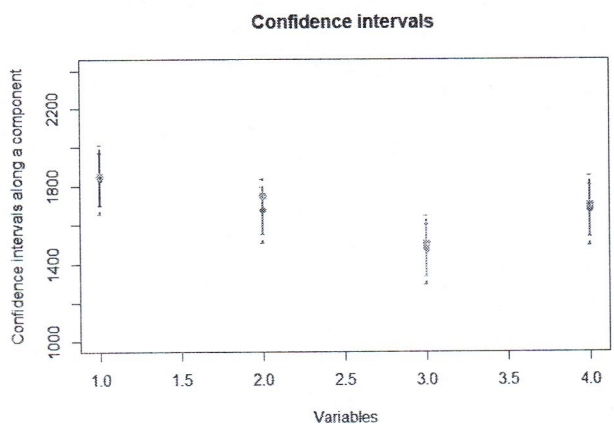
• = μ_0 projection on $x_1, \dots, x_4$

(again, this just give us a partial information)

```
# Yes, it is, because it is inside all the T2-intervals,

### Bonferroni intervals on the components of the mean
### with global Level 95%
### ---------------------------------------------------------------------------
k <- p
cfr.t <- qt(1 - alpha/(k*2), n-1)

Bf <- cbind(inf = x.mean - cfr.t*sqrt(diag(x.cov)/n),
            center = x.mean,
            sup = x.mean + cfr.t*sqrt(diag(x.cov)/n))
Bf
```

```
##         inf   center      sup
## V1 1706.611 1840.423 1974.235
## V2 1555.505 1679.000 1802.495
## V3 1339.694 1473.846 1607.998
## V4 1540.281 1675.154 1810.027
```

```
# Let's do a plot
x11()
matplot(1:4,1:4,pch='',ylim=range(stiff),xlab='Variables',ylab='Confidence intervals along a component',main='Confidence int
ervals')

for(i in 1:4) segments(i,T2[i,1],i,T2[i,3],lwd=2,col='grey35', lty=3)
points(1:4, T2[,1], pch='-', col='grey35')
points(1:4, T2[,3], pch='-', col='grey35')

for(i in 1:4) segments(i,Bf[i,1],i,Bf[i,3],lwd=2,col=i)
points(1:4, Bf[,2], pch=16, col=1:4)
points(1:4, Bf[,1], pch='-', col=1:4)
points(1:4, Bf[,3], pch='-', col=1:4)

# Is mu0 inside the Bonferroni confidence region?
# we add it to the plot
points(1:4, mu0, lwd=3, col='orange')
```




```
# Yes, it is, because it belongs to all the intervals along the components

graphics.off()
```

Further material

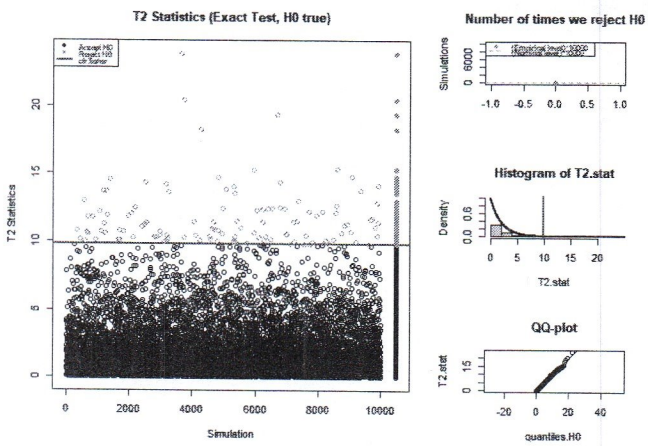
```
### Simulation to compute the actual level and the power of the T2 test
library(mvtnorm)
```

```
### ------------------------------------------------------------------------
### 1) Exact test (T2) & H0 TRUE
### ------------------------------------------------------------------------
mu0 <- c(1,0)
alpha <- 0.01

mu <- c(1,0)
sig <- matrix(c(1,1,1,2),nrow=2)
n=100
p=2

T2.stat <- NULL
# We generate 10000 samples under H0 and count how many times we reject H0 (true) [this is an estimate of the actual level]
for(i in 1:10000){
  x <- rmvnorm(n, mean=mu, sigma=sig)
  x.mean   <- colMeans(x)
  x.cov    <- cov(x)
  x.invcov <- solve(x.cov)
  x.T2       <- n * (x.mean-mu0) %*% x.invcov %*% (x.mean-mu0)
  T2.stat <- c(T2.stat, x.T2)
}
# The vector T2.stat collects 10000 independent realizations of the T2 statistics, computed under H0 => we know its distribu
tion!
```

```
x11()
layout(cbind(c(1,1,1),c(2,3,4)), widths=c(2,1), heights=c(1,1))
plot(T2.stat, xlab='Simulation', ylab='T2 Statistics',xlim=c(0,10500))
abline(h = ((n-1)*p/(n-p))*qf(1-alpha,p,n-p), col='red', lwd=2)  # criterion to rejecto H0
points(which(T2.stat>((n-1)*p/(n-p))*qf(1-alpha,p,n-p)),T2.stat[which(T2.stat>((n-1)*p/(n-p))*qf(1-alpha,p,n-p))], col='oran
ge')
points(rep(10500,10000),T2.stat, xlab='Simulations', ylab='T2 statistics',pch=19)
points(rep(10500, length(which(T2.stat>((n-1)*p/(n-p))*qf(1-alpha,p,n-p)))),T2.stat[which(T2.stat>((n-1)*p/(n-p))*qf(1-alph
a,p,n-p))], pch=19, col='orange')
title(main='T2 Statistics (Exact Test, H0 true)')
legend('topleft',c('Accept H0', 'Reject H0', 'cfr.fisher'), col=c('black','orange','red'),pch=c(19,19,-1),lty=c(-1,-1,1),lwd
=c(1,1,2),cex=.8)

plot(0,0, ylab='Simulations', xlab='',ylim=c(0,10000),main='Number of times we reject H0')
count=0
for(i in 1:10000)
{
  if(T2.stat[i]>((n-1)*p/(n-p))*qf(1-alpha,p,n-p))
  {
    count=count+1
    points(0,count,col='orange',pch=19)
  }
}
abline(h=10000*alpha,lty=2,col='grey')
legend('topleft', c('(Empirical level)*10000', '(Nominal level)*10000'), col=c('orange','grey'), lty=c(-1,2),pch=c(19,-1),ce
x=.8)

# we compute the proportion of times on the total number of simulations in which I reject H0: this is an empirical estimate
 of the Level of the test (recall: H0 is true!)
sum(T2.stat > ((n-1)*p/(n-p))*qf(1-alpha,p,n-p))/10000
```

```
## [1] 0.0117
```

```
# we test the distribution of T2 under H0: (it should be proportional to a Fisher)
hist(T2.stat, prob=TRUE,col='grey85',ylim=c(0,1),xlab='T2.stat',ylab='Density',main='Histogram of T2.stat')
xx <- seq(0,40,by=0.05)
lines(xx,df(xx*(n-p)/((n-1)*p),p,n-p),type="l",lwd=2)
abline(v = ((n-1)*p/(n-p))*qf(1-alpha,p,n-p), col='red', lwd=2)

quantiles.H0 <- ((n-1)*p/(n-p))*qf((1:10000 - 0.5)/10000,p,n-p)
qqplot(quantiles.H0, T2.stat, asp=1,main='QQ-plot')
abline(0,1)
```

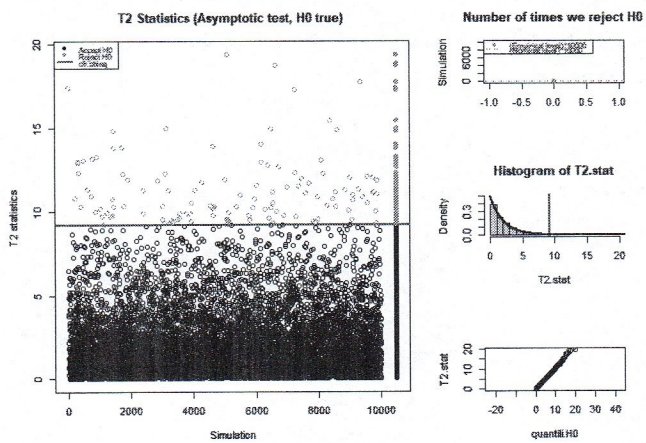


```
### ---------------------------------------------------------------------------
### 2) Asymptotic test (Chi-squared) & H0 TRUE
### ---------------------------------------------------------------------------
# I repeat the same experiment but with an
# asymptotic test
T2.stat <- NULL
for(i in 1:10000){
  x <- rmvnorm(n, mean=mu, sigma=sig)
  x.mean   <- colMeans(x)
  x.cov    <- cov(x)
  x.invcov <- solve(x.cov)
  x.T2       <- n * (x.mean-mu0) %*% x.invcov %*% (x.mean-mu0)
  T2.stat <- c(T2.stat, x.T2)
}
```

```
x11()
layout(cbind(c(1,1,1),c(2,3,4)), widths=c(2,1), heights=c(1,1))
plot(T2.stat, xlab='Simulation', ylab='T2 statistics',xlim=c(0,10500))
abline(h = qchisq(1-alpha,p), col='red', lwd=2)  # criterion to reject
points(which(T2.stat>qchisq(1-alpha,p)),T2.stat[which(T2.stat>qchisq(1-alpha,p))], col='orange')
points(rep(10500,10000),T2.stat, xlab='Simulations', ylab='T2 Statistics',pch=19)
points(rep(10500, length(which(T2.stat>qchisq(1-alpha,p)))),T2.stat[which(T2.stat>qchisq(1-alpha,p))], pch=19, col='orange')
title(main='T2 Statistics (Asymptotic test, H0 true)')
legend('topleft',c('Accept H0', 'Reject H0', 'cfr.chisq'), col=c('black','orange','red'),pch=c(19,19,-1),lty=c(-1,-1,1),lwd=
c(1,1,2),cex=.8)

plot(0,0, ylab='Simulation', xlab='',ylim=c(0,10000),main='Number of times we reject H0')
count=0
for(i in 1:10000){
  if(T2.stat[i]>qchisq(1-alpha,p)){
    count=count+1
    points(0,count,col='orange',pch=19)
  }
}
abline(h=10000*alpha,lty=2,col='grey')
legend('topleft', c('(Empirical level)*10000', '(Nominal level)*10000'), col=c('orange','grey'), lty=c(-1,2),pch=c(19,-1),ce
x=.8)

# we compute the proportion of times on the total number of simulations in which I reject H0 this is an empirical estimate o
f the level of the test (recall: H0 is true!)
sum(T2.stat > qchisq(1-alpha,p))/10000
```

```
## [1] 0.0119
```

```
# we test the distribution of T2 under H0: they should be (approximately) draws from a chi-squared
hist(T2.stat, prob=TRUE,col='grey85',ylim=c(0,.5),xlab='T2.stat',ylab='Density',main='Histogram of T2.stat')
xx <- seq(0,40,by=0.05)
lines(xx,dchisq(xx,p),type="l",lwd=2)
abline(v = qchisq(1-alpha,p), col='red', lwd=2)

quantili.H0 <- qchisq((1:10000 - 0.5)/10000,p)
qqplot(quantili.H0, T2.stat, asp=1)
abline(0,1)
```

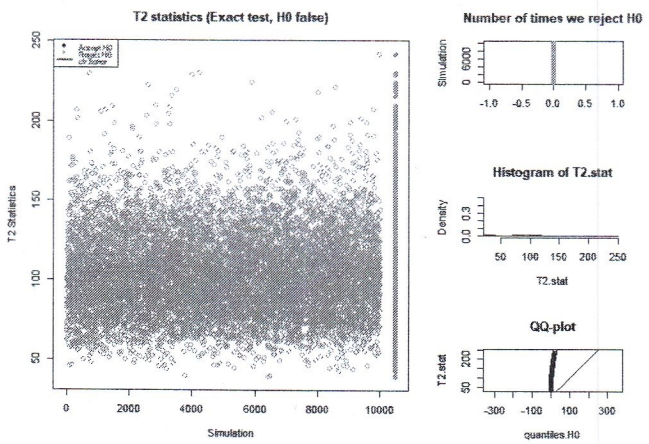



```
graphics.off()

### ---------------------------------------------------------------------------
### 3) H0 FALSE
### ---------------------------------------------------------------------------
# We keep the same test but change the distribution according to which we generate the data (=> H0 is false, we now aim to r
eject it)

mu <- c(2,1)

T2.stat <- NULL
for(i in 1:10000){
  x <- rmvnorm(n, mean=mu, sigma=sig)
  x.mean   <- colMeans(x)
  x.cov    <- cov(x)
  x.invcov <- solve(x.cov)
  x.T2       <- n * (x.mean-mu0) %*% x.invcov %*% (x.mean-mu0)
  T2.stat <- c(T2.stat, x.T2)
}
# The vector T2.stat collects 10000 independent realizations of the T2 statistics, computed under H0 (but now H0 is false!)
```

```
x11()
layout(cbind(c(1,1,1),c(2,3,4)), widths=c(2,1), heights=c(1,1))
plot(T2.stat, xlab='Simulation', ylab='T2 Statistics',xlim=c(0,10500))
abline(h = ((n-1)*p/(n-p))*qf(1-alpha,p,n-p), col='red', lwd=2)  # criterio di rifiuto
points(which(T2.stat>((n-1)*p/(n-p))*qf(1-alpha,p,n-p)),T2.stat[which(T2.stat>((n-1)*p/(n-p))*qf(1-alpha,p,n-p))], col='oran
ge')
points(rep(10500,10000),T2.stat, xlab='Simulazioni', ylab='Statistica T2',pch=19)
points(rep(10500, length(which(T2.stat>((n-1)*p/(n-p))*qf(1-alpha,p,n-p)))),T2.stat[which(T2.stat>((n-1)*p/(n-p))*qf(1-alph
a,p,n-p))], pch=19, col='orange')
title(main='T2 statistics (Exact test, H0 false)')
legend('topleft',c('Accept H0', 'Reject H0', 'cfr.fisher'), col=c('black','orange','red'),pch=c(19,19,-1),lty=c(-1,-1,1),lwd
=c(1,1,2),cex=.8)

plot(0,0, ylab='Simulation', xlab='',ylim=c(0,10000),main='Number of times we reject H0')
count=0
for(i in 1:10000){
  if(T2.stat[i]>((n-1)*p/(n-p))*qf(1-alpha,p,n-p)){
    count=count+1
    points(0,count,col='orange',pch=19)
  }
}

# we compute the proportion of times on the total number of simulations in which I reject H0 this is an empirical estimate o
f the POWER of the test for mu = c(2,1) (recall: H0 is false!)
sum(T2.stat > ((n-1)*p/(n-p))*qf(1-alpha,p,n-p))/10000
```

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

```
# we test the distribution of the T2 stat.: now H0 is false, we expect that they are NOT draws from a Fisher
hist(T2.stat, prob=TRUE,col='grey85',ylim=c(0,.5),xlab='T2.stat',ylab='Density',main='Histogram of T2.stat')
xx <- seq(0,40,by=0.05)
lines(xx,df(xx*(n-p)/((n-1)*p),p,n-p),type="l",lwd=2)
abline(v = ((n-1)*p/(n-p))*qf(1-alpha,p,n-p), col='red', lwd=2)

quantiles.H0 <- ((n-1)*p/(n-p))*qf((1:10000 - 0.5)/10000,p,n-p)
qqplot(quantiles.H0, T2.stat, asp=1,main='QQ-plot')
abline(0,1)
```




```
graphics.off()
# Remark: the power depends on the position of the theoretical mean with respect to the mean under H0
```