LAB Permutation Tests

```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### PERMUTATION TESTS: TWO INDEPENDENT POPULATIONS
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
```

Simulated example:

```
# We sample n1 data from a first population X1 and n2 data froma second population X2
# Test:
# H0:  X1 =^d  X2
# H1:  X1 !=^d X2

# ------------------------------------------------------------------
# Case 1. H0 FALSE
# To understand the behaviour of the test, we sample from two populations
# with different means

# Parameters:
n1 <- n2 <- 10
n <- n1 + n2

# Simulation:
set.seed(240279)
x1 <- runif(n1,0,4)
x2 <- runif(n2,0,4)+3

x_pooled <- c(x1,x2)

par(mfrow=c(1,2))
boxplot(x1,x2,main='Original data')

# How data change if we apply one random permutation?
permutation <- sample(1:n)

x_perm  <- x_pooled[permutation]
x1_perm <- x_perm[1:n1]
x2_perm <- x_perm[(n1+1):n]

boxplot(x1_perm,x2_perm,main='Permuted data')
```

(n_1 and n_2 small)

Case 1. H_0 FALSE. We generate 2 samples from 2 different distributions → H_0 : false

To understand the behaviour of the test, we sample from two populations with different means

Parameters: $n_1 = n_2 = 10$, $n = n_1 + n_2$

Simulation: set.seed(240279), x_1 ~ runif(n1,0,4), x_2 ~ runif(n2,0,4)+3

x_pooled <- c(x1,x2) : we put together all the values

boxplot(x1,x2,main='Original data')

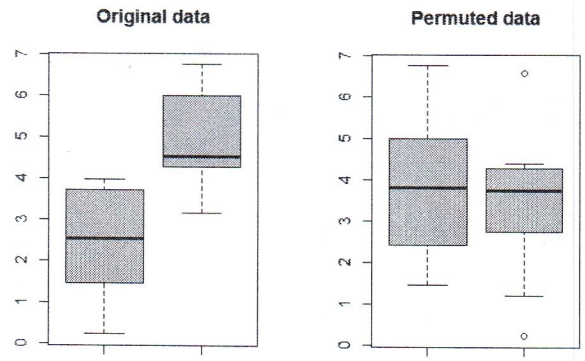
How data change if we apply one random permutation?

permutation <- sample(1:n)

x_perm <- x_pooled[permutation]
x1_perm <- x_perm[1:n1]
x2_perm <- x_perm[(n1+1):n]

} we create the new permutated samples

boxplot(x1_perm,x2_perm,main='Permuted data')



We have to understand if this difference between original and permutated happened by chance or it's usually like that :
if this change is remarkable (like in this case) for many permuted samples → we'll have a small p-value

As a test statistic we use: $T = |\bar{x}_1 - \bar{x}_2|$

```
abs(mean(x1)-mean(x2))
```

```
## [1] 2.465564
```

(original)

```
abs(mean(x1_perm)-mean(x2_perm))
```

```
## [1] 0.4176959
```

(permutated)

```
# The mean difference is lower.
# Was that a case?

# TEST
# Test statistic: absolute difference between the two means
T0 <- abs(mean(x1) - mean(x2))
T0
```

```
## [1] 2.465564
```

```
# Cardinality of the permutational space:
factorial(n)
```

```
## [1] 2.432902e+18
```

```
# Number of distinct values of T*:
factorial(n)/(2*factorial(n1)*factorial(n2))
```

```
## [1] 92378
```

← we can have 92k of possible values of the test statistic

```
# Minimun achieveable p-value:
1/(factorial(n)/(2*factorial(n1)*factorial(n2)))
```

```
## [1] 1.082509e-05
```

Conditional Monte Carlo estimate (which randomly peaks permutations)

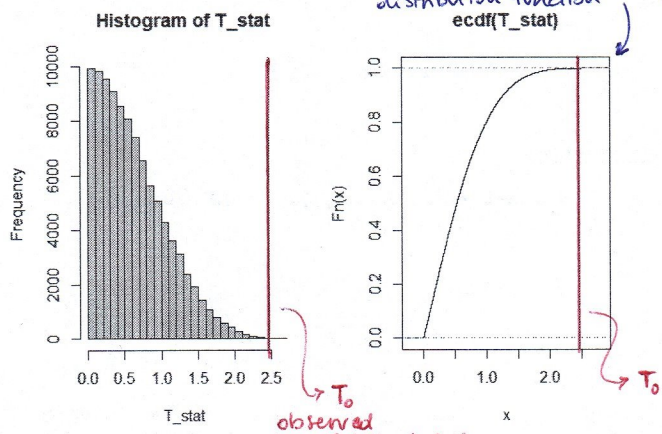
```r
# CMC to estimate the p-value
B <- 100000 # Number of permutations
T_stat <- numeric(B) # Vector where we will store the values of T*

# To estimate the p-value we use a loop
# Inside the loop, we do the following:
# 1. choose a random permutation of data (with the command sample)
# 2. calculate and save the test statistic obtained with the permuted data
for(perm in 1:B){
  # permutation:
  permutation <- sample(1:n)
  x_perm <- x_pooled[permutation]
  x1_perm <- x_perm[1:n1]
  x2_perm <- x_perm[(n1+1):n]
  # test statistic:
  T_stat[perm] <- abs(mean(x1_perm) - mean(x2_perm))
}

# Permutational distribution of T
hist(T_stat,xlim=range(c(T_stat,T0)),breaks=30)
abline(v=T0,col=3,lwd=2)

plot(ecdf(T_stat))
abline(v=T0,col=3,lwd=2)
```

Note: if the number of permutations is low we can explorate all of them

Here we're doing something useless from the computational point of view: we're randomly picking 100k permutations among 92k (it would be better (in terms of costs) to explore all the 92k permutations)

T statistic for the original data

Empirical cumulative distribution function (seems continuous but it's not!)

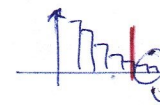


Permutational distribution under H_0 of $T = |\bar{X}_1 - \bar{X}_2|$ (absolute value of the difference of the sample means)

T_0 observed value of test statistic

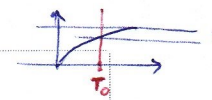
T_0

p-value = proportion of permutations that had a T-statistic value $\geq T_0$:



$\frac{\#(\geq T_0)}{\#\text{ all permutations}}$

p-value (with ecdf(T_stat)):

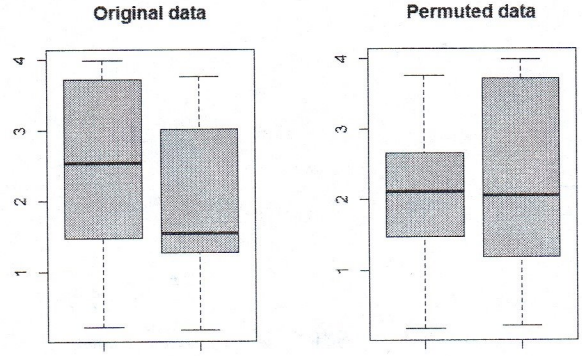



```r
# p-value
p_val <- sum(T_stat>=T0)/B
p_val
```

```
## [1] 3e-04
```

⟹ we reject H_0

```r
# ----------------------------------------------------------------------
# Case 2. H0 TRUE
# now we simulate data from two populations with the same distribution

# Simulation:
set.seed(240279)
x1 <- runif(n1,0,4)
x2 <- runif(n2,0,4)+0
x_pooled <- c(x1,x2)

par(mfrow=c(1,2))
boxplot(x1,x2,main='Original data')

# How data change if we apply one random permutation?
permutation <- sample(1:n)

x_perm <- x_pooled[permutation]
x1_perm <- x_perm[1:n1]
x2_perm <- x_perm[(n1+1):n]

boxplot(x1_perm,x2_perm,main='Permuted data')
```



Here it doesn't seem like the permutation influenced so much

```r
abs(mean(x1)-mean(x2))
```

```
## [1] 0.5344363
```

(original)

```
abs(mean(x1_perm)-mean(x2_perm))
```

```
## [1] 0.1823041
```

(permuted)

```
# TEST
# Test statistic: absolute difference between the two means
T0 <- abs(mean(x1) - mean(x2))
T0
```

```
## [1] 0.5344363
```

```
# Cardinality of the permutational space:
factorial(n)
```

```
## [1] 2.432902e+18
```

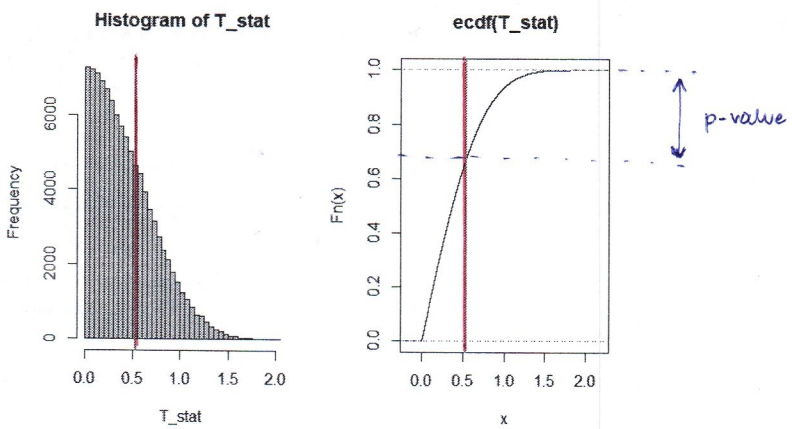
```
# Number of distinct values of T*:
factorial(n)/(2*factorial(n1)*factorial(n2))
```

```
## [1] 92378
```

```
# Minimun achieveable p-value:
1/(factorial(n)/(2*factorial(n1)*factorial(n2)))
```

```
## [1] 1.082509e-05
```

```
# CMC to estimate the p-value
B <- 100000 # Number of permutations
T_stat <- numeric(B) # Vector where we will store the values of T*

for(perm in 1:B){
  # permutation:
  permutation <- sample(1:n)
  x_perm <- x_pooled[permutation]
  x1_perm <- x_perm[1:n1]
  x2_perm <- x_perm[(n1+1):n]
  # test statistic:
  T_stat[perm] <- abs(mean(x1_perm) - mean(x2_perm))
}

# Permutational distribution of T
hist(T_stat,xlim=range(c(T_stat,T0)),breaks=30)
abline(v=T0,col=3,lwd=2)

plot(ecdf(T_stat))
abline(v=T0,col=3,lwd=2)
```




```
# p-value
p_val <- sum(T_stat>=T0)/B
p_val
```

```
## [1] 0.33116
```

$\Longrightarrow$ we don't reject H_0

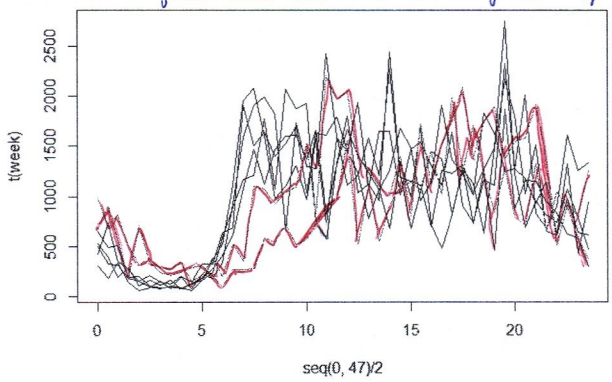
```
dev.off()
```

```
### ----------------------------------------------------------------------
### ----------------------------------------------------------------------
### PERMUTATION TESTS (MULTIVARIATE)
### ----------------------------------------------------------------------
### ----------------------------------------------------------------------

# Example 1: Two (independent) multivariate population test
# Hourly accesses to AreaC: working days vs week-end days

d1 <- read.csv('accessi-orari-areac-2016-09-12-00_00_00.csv', header=T)
d2 <- read.csv('accessi-orari-areac-2016-09-13-00_00_00.csv', header=T)
d3 <- read.csv('accessi-orari-areac-2016-09-14-00_00_00.csv', header=T)
d4 <- read.csv('accessi-orari-areac-2016-09-15-00_00_00.csv', header=T)
d5 <- read.csv('accessi-orari-areac-2016-09-16-00_00_00.csv', header=T)
d6 <- read.csv('accessi-orari-areac-2016-09-17-00_00_00.csv', header=T)
d7 <- read.csv('accessi-orari-areac-2016-09-18-00_00_00.csv', header=T)

week <- rbind(d1[,2], d2[,2], d3[,2], d4[,2], d5[,2], d6[,2], d7[,2])
matplot(seq(0,47)/2,t(week), type='l', col=c(1,1,1,1,1,2,2), lty=1)
```

Permutations and reflections are made on the entire unit. We permute two rows, we reflect one row, we don't work on singular columns!

Two dataset: t_1: 5 × 48 (working days)
t_2: 2 × 48 (weekends)

Each column contains the number of cars entering the area C of Milan in a given half an hour in a given day. Every day is made by 48 half an hours.
Each row correspond to one day.



We cannot apply Hotelling T^2 because we don't have a sufficient sample size to have a full rank estimate of the sample variance/covariance matrix

We define the two groups:

```
t1 <- week[1:5,]
t2 <- week[6:7,]

# Computing a proper test statistic
# (i.e., squared distance between the two sample mean vectors)
t1.mean <- colMeans(t1)
t2.mean <- colMeans(t2)

n1 <- dim(t1)[1]
n2 <- dim(t2)[1]
n  <- n1 + n2

T20 <- as.numeric((t1.mean-t2.mean) %*% (t1.mean-t2.mean))
T20
```

← squared euclidean distance between the two samples' mean (like Mahalanobis's distance in which the sample of variance/covariance matrix is the identity). We would use it in a parametric framework if we had known that the cov/var matrix is I.

```
## [1] 8976210
```

```
# Selecting a proper permutation strategy (i.e., data point permutations)

# number of possible data point permutations
factorial(7)
```

```
## [1] 5040
```

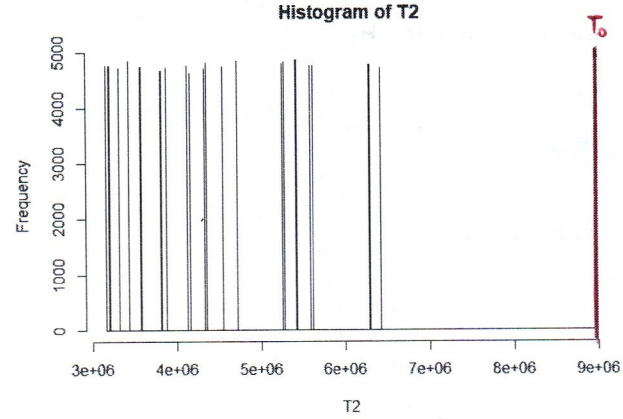
```
# number of different values of the test statistic
choose(7,5)
```

How many groups of 5 and 2 can we do?
we can allign the 7 days in 21 possible ways
→ the permutational distribution will be a discrete uniform with just 21 values

```
## [1] 21
```

```
# Estimating the permutational distribution under H0
B <- 100000
T2 <- numeric(B)

for(perm in 1:B){
  # Random permutation of indexes
  # When we apply permutations in a multivariate case, we keep the units together
  # i.e., we only permute the rows of the data matrix
  t_pooled <- rbind(t1,t2)
  permutation <- sample(n)
  t_perm <- t_pooled[permutation,]
  t1_perm <- t_perm[1:n1,]
  t2_perm <- t_perm[(n1+1):n,]

  # Evaluation of the test statistic on permuted data
  t1.mean_perm <- colMeans(t1_perm)
  t2.mean_perm <- colMeans(t2_perm)
  T2[perm]  <- (t1.mean_perm-t2.mean_perm) %*% (t1.mean_perm-t2.mean_perm)
}

# plotting the permutational distribution under H0
hist(T2,xlim=range(c(T2,T20)),breaks=1000)
abline(v=T20,col=3,lwd=4)
```

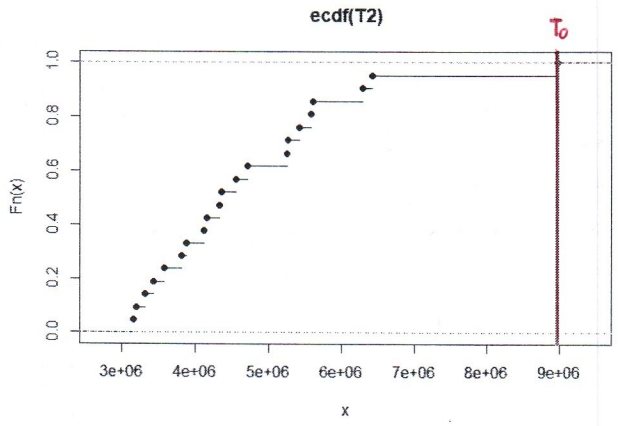
Remember that we're permutating the days (the vectors) we're not destroying the vectors!
(permutation of the rows, not the columns)



← 21 test statistics (21 possible test statistics, our one is pretty extreme) (is the most extreme)

```
plot(ecdf(T2))
abline(v=T20,col=3,lwd=4)
```

empirical cumulative distribution function

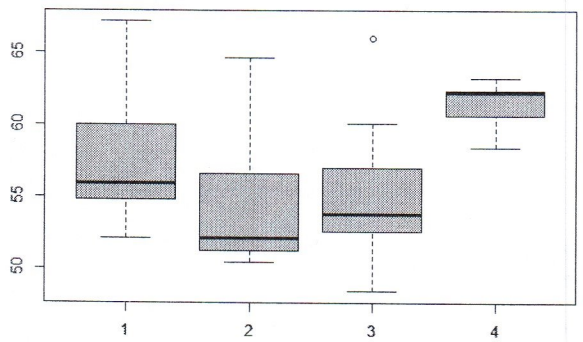



```
# p-value
p_val <- sum(T2>=T20)/B
p_val
```

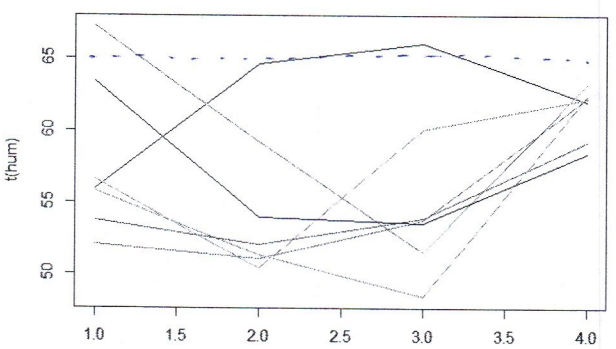
```
## [1] 0.04868
```

← smallest p-value achievable (we reject H_0)

```
#
# Example 2: Center of simmetry of one multivariate population
# Relative humidity in Milan during the summer months

hum <- read.csv2('307_Umidita_relativa_2008_2014.csv', header=T)
hum <- hum[,3]
hum <- matrix(hum, ncol=12, byrow=T)[,6:9]

boxplot(hum)
```




```
matplot(t(hum), type='l', lty=1)
```



7 years (each line is a year) and 4 features (average relative humidity in the 1st, 2nd, 3rd and 4th trimester of the year)

H_0 : center = $[65, 65, 65, 65]^T$

Basically we want to check:

if we make a reflection of the lines w.r.t. the horizontal line (given by $[65,65,65,65]^T$), would we obtain a dataset with exactly the same likelihood

looking at the plot we're expecting strong evidence to reject

This time, for each curve we can decide to reflect it or not

$\Rightarrow 2^N = 2^7 = 128$ possible transformations (this time are not permutations)

```r
# center of simmetry under H0
mu0      <- c(65, 65, 65, 65)

# Computing a proper test statistic
# (i.e., squared distance between the sample mean vector and the hypothesized center of simmetry)
x.mean   <- colMeans(hum)
n <- dim(hum)[1]
p <- dim(hum)[2]

T20 <- as.numeric((x.mean-mu0) %*% (x.mean-mu0) )

# Selecting a proper likelihood-invariant strategy (i.e., data point reflections)
# We are assuming that under H0 the data distribution is simmetric

# number of possible data point reflections
2^7
```

: we can decide wether to reflect or not a curve : 2^N

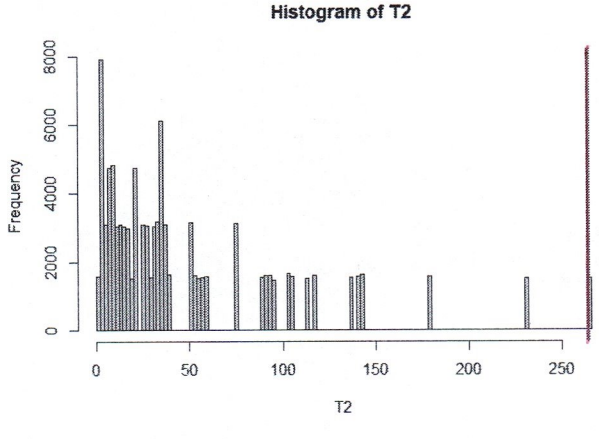
```
## [1] 128
```

```r
# number of different values of the test statistic
2^7/2
```

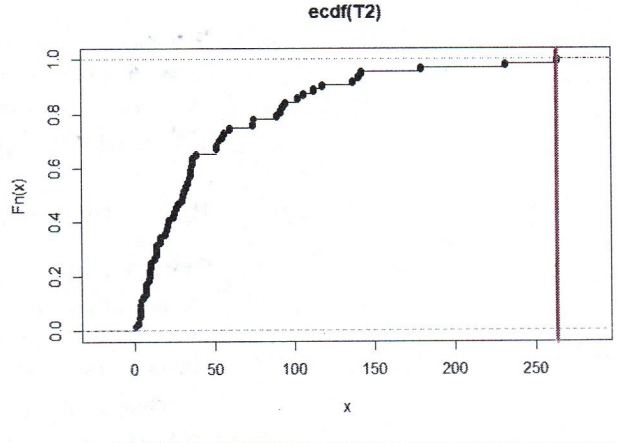
because if we reflect them all we would obtain a value test statistic equal to the original one

```
## [1] 64
```

```r
# Estimating the permutational distribution under H0
B <- 100000
T2 <- numeric(B)

for(perm in 1:B){
  # In this case we use changes of signs in place of permutations

  # Permuted dataset
  signs.perm <- rbinom(n, 1, 0.5)*2 - 1
  hum_perm <- mu0 + (hum - mu0) * matrix(signs.perm,nrow=n,ncol=p,byrow=FALSE)
  x.mean_perm <- colMeans(hum_perm)
  T2[perm]  <- (x.mean_perm-mu0)  %*% (x.mean_perm-mu0)
}

# plotting the permutational distribution under H0
hist(T2,xlim=range(c(T2,T20)),breaks=100)
abline(v=T20,col=3,lwd=4)
```




```r
plot(ecdf(T2))
abline(v=T20,col=3,lwd=4)
```




```r
# p-value
p_val <- sum(T2>=T20)/B
p_val
```

```
## [1] 0.01535
```

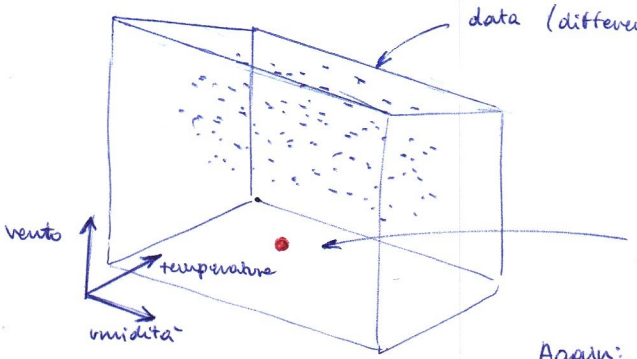
$\Longrightarrow$ we reject H_0

kind of same as before

when we have to work with paired populations we work with one population: the population of differences

```
#-------------------------------------------------------------------------
# Example 3: two paired multivariate population test
# (i.e., same days for Milan and Barcelona)
# The data set contains observations of temperature, humidity and wind in Milan and Barcelona
# on 50 different days

t1 <- read.table('barcellona.txt', header=T)
t2 <- read.table('milano.txt', header=T)

library(rgl)
open3d()
```

```
plot3d(t1-t2, size=3, col='orange', aspect = F)
points3d(0,0,0, size=6)
a = scene3d()
rgl.close()
x11()
rglwidget(a)
```



Again: if it's the center of symmetry we can reflect the data and we shouldn't notice the difference

In this case we can use the Hotelling T^2 statistic (the sample size allows us) ($p = 3$, $n = 50$)

```
p  <- dim(t1)[2]
n1 <- dim(t1)[1]
n2 <- dim(t2)[1]
n <- n1+n2

# Evaluate the test statistic
t1.mean <- colMeans(t1)
t2.mean <- colMeans(t2)
t1.cov  <-  cov(t1)
t2.cov  <-  cov(t2)
Sp      <- ((n1-1)*t1.cov + (n2-1)*t2.cov)/(n1+n2-2)
Spinv   <- solve(Sp)

delta.0 <- c(0,0,0)

diff <- t1-t2
diff.mean <- colMeans(diff)
diff.cov <- cov(diff)
diff.invcov <- solve(diff.cov)

#T20 <- as.numeric(n1 * (diff.mean-delta.0)  %*% (diff.mean-delta.0))
#T20 <- as.numeric(n1 * (diff.mean-delta.0) %*% solve(diag(diag(diff.cov))) %*% (diff.mean-delta.0))
T20 <- as.numeric(n1 * (diff.mean-delta.0) %*% diff.invcov %*% (diff.mean-delta.0))

# With coupled data, we have to reduce the number of permutations
# the only permutations that preserve the likelyhood under H0 are
# permutations within each couple (i.e., exchange Milan and Barcelona within the same day)
# We are assuming the the differences are simmetrically distributed under H0

# number of possible data point reflections
2^50
```

other possibilities

squared euclidean distance

we're neglecting the covariance structure, we're rescaling each component by its standard deviation

we're rescaling also using the variance/covariance structure (Mahalanobis's distance^2 of the sample mean to the null point)

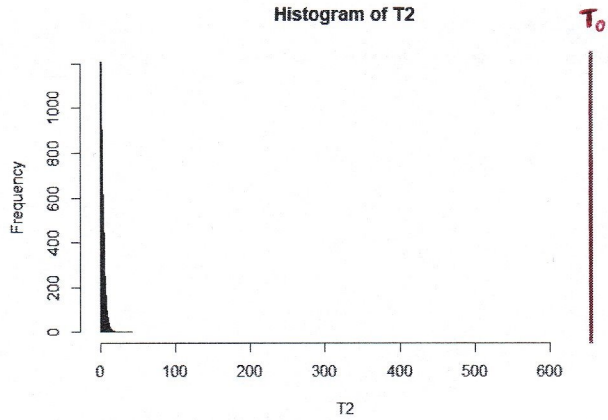
```
## [1] 1.1259e+15
```

→ need of MonteCarlo

```
# Estimating the permutational distribution under H0
B <- 10000
T2 <- numeric(B)

for(perm in 1:B)
  {
  # Random permutation
  # obs: exchanging data within couples means changing the sign of the difference
  signs.perm <- rbinom(n1, 1, 0.5)*2 - 1

  diff_perm <- diff * matrix(signs.perm,nrow=n1,ncol=p,byrow=FALSE)
  diff.mean_perm <- colMeans(diff_perm)
  diff.cov_perm <- cov(diff_perm)
  diff.invcov_perm <- solve(diff.cov_perm)

  #T2[perm] <- as.numeric(n1 * (diff.mean_perm-delta.0) %*% (diff.mean_perm-delta.0))
  #T2[perm] <- as.numeric(n1 * (diff.mean_perm-delta.0) %*% solve(diag(diag(diff.cov_perm))) %*% (diff.mean_perm-delta.0))
  T2[perm] <- as.numeric(n1 * (diff.mean_perm-delta.0) %*% diff.invcov_perm %*% (diff.mean_perm-delta.0))
  }

# plotting the permutational distribution under H0
hist(T2,xlim=range(c(T2,T20)),breaks=100)
abline(v=T20,col=3,lwd=4)
```

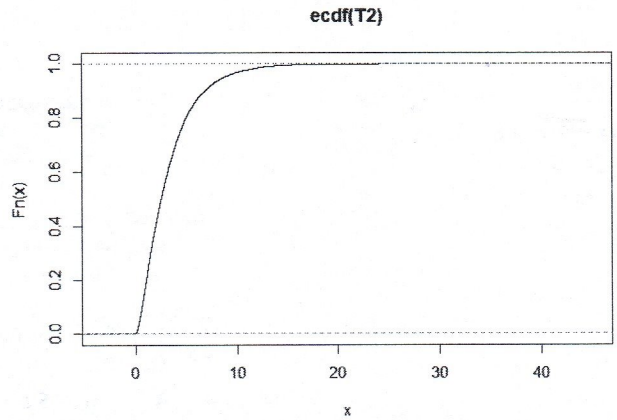
instead of randomly mixing data we're randomly picking up 1/-1

here we just multiply the 1/-1 to the



T_0

```
plot(ecdf(T2))
abline(v=T20,col=3,lwd=4)
```



(not even visible)

```
# p-value
p_val <- sum(T2>=T20)/B
p_val
```

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

```
2^50
```

```
## [1] 1.1259e+15
```

↗ test ANOVA has F-test statistic

F tests (ANOVA too) tends to be less robust w.r.t. the violation of normality assumptions (⇒ if normality is violated then it's a big problem ⇒ non-parametric testings)

```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### PERMUTAZION TESTS: ANOVA
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------

# One-way ANOVA
# (p=1, g=6)

head(chickwts)
```

(generalization of the two population test)

```
##   weight      feed
## 1    179 horsebean
## 2    160 horsebean
## 3    136 horsebean
## 4    227 horsebean
## 5    217 horsebean
## 6    168 horsebean
```

```
attach(chickwts)
summary(chickwts)
```

```
##      weight             feed
##  Min.   :108.0   casein   :12
##  1st Qu.:204.5   horsebean:10
##  Median :258.0   linseed  :12
##  Mean   :261.3   meatmeal :11
##  3rd Qu.:323.5   soybean  :14
##  Max.   :423.0   sunflower:12
```

is there effect based on the feed?

```r
g <- nlevels(feed)
n <- dim(chickwts)[1]

layout(cbind(1,2))
plot(feed, weight, xlab='treat',col=rainbow(g),main='Original Data')

# H0: tau1 = tau2 = tau3 = tau4 = tau5 = tau6 = 0
# the chickens belong to the same population

# H1: (H0)^c
# the chickens belong to several different population
# Parametric test:
fit <- aov(weight ~ feed)
summary(fit)
```

$H_0: \tau_1 = \dots = \tau_6 = 0$: the 6 groups share the same distribution

$H_1: \exists \tau_j \neq 0 \quad j = 1, \dots, 6$

← just because we want to use the F statistic of the parametric test

Under H_0 they come from the same distribution, so if we shuffle the observations and permute the elements it shouldn't be noticed:

(we change randomly some labels)

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## feed         5 231129   46226   15.37 5.94e-10 ***
## Residuals   65 195556    3009
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

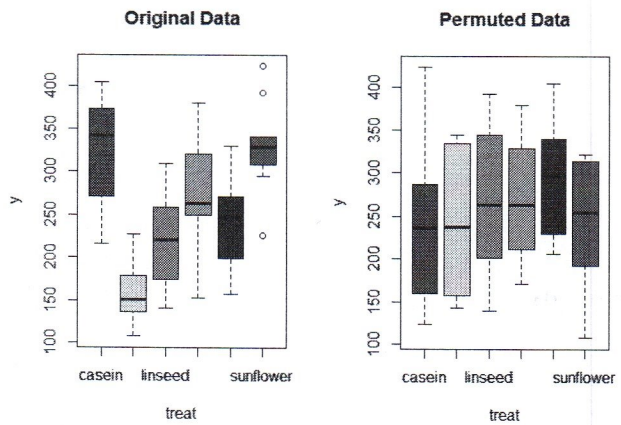
```r
# Permutation test:
# Test statistic: F stat
T0 <- summary(fit)[[1]][1,4]
T0
```

```
## [1] 15.3648
```

```r
# what happens if we permute the data?
permutazione <- sample(1:n)
weight_perm <- weight[permutazione]
fit_perm <- aov(weight_perm ~ feed)
summary(fit_perm)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## feed         5  34466    6893   1.142  0.347
## Residuals   65 392219    6034
```

```r
plot(feed, weight_perm, xlab='treat',col=rainbow(g),main='Permuted Data')
```

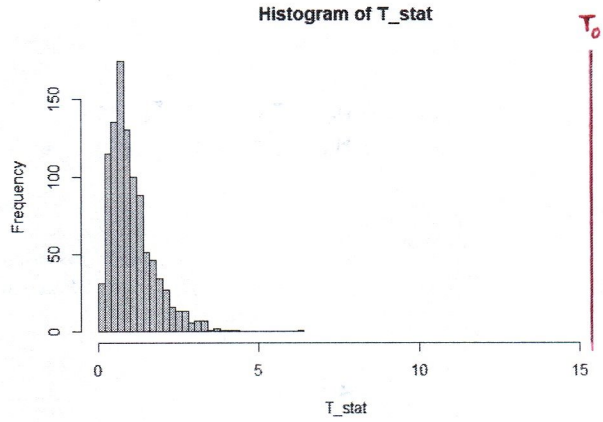


Visually the two are very different, let's make more permutations

```r
# CMC to estimate the p-value
B <- 1000 # Number of permutations
T_stat <- numeric(B)
n <- dim(chickwts)[1]

for(perm in 1:B){
  # Permutation:
  permutation <- sample(1:n)
  weight_perm <- weight[permutation]
  fit_perm <- aov(weight_perm ~ feed)

  # Test statistic:
  T_stat[perm] <- summary(fit_perm)[[1]][1,4]
}

layout(1)
hist(T_stat,xlim=range(c(T_stat,T0)),breaks=30)
abline(v=T0,col=3,lwd=2)
```

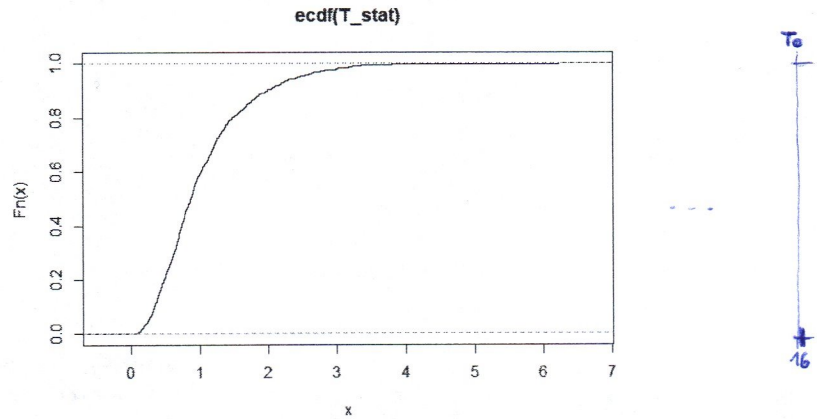
(permutational)

CMC to get the distribution of the test statistic (from which we'll get the p-value)



⇒ For sure we'll reject H_0

```
plot(ecdf(T_stat))
abline(v=T20,col=3,lwd=4)
```




```
# p-value
p_val <- sum(T_stat>=T0)/B
p_val
```

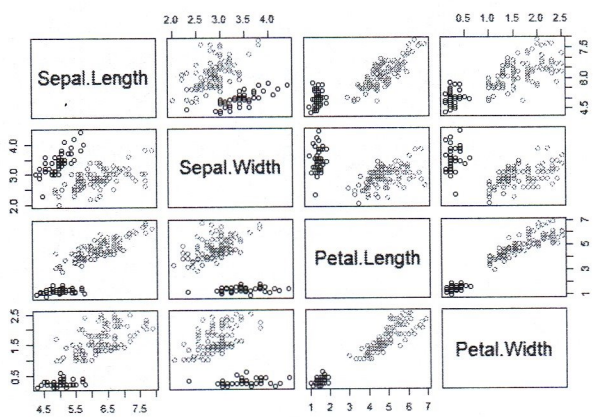
```
## [1] 0
```

⟹ reject H_0

```
# we reject the null hypothesis

detach(chickwts)

# ------------------------------------------------------------
# MANOVA
data(iris)
attach(iris)
species.name <- factor(Species, labels=c('setosa','versicolor','virginica'))
iris4        <- iris[,1:4]
plot(iris4,col=species.name)
```

Multivariate one-way ANOVA

(Same code as one-way ANOVA but with a different statistic)
(Moreover, same way of reasoning)

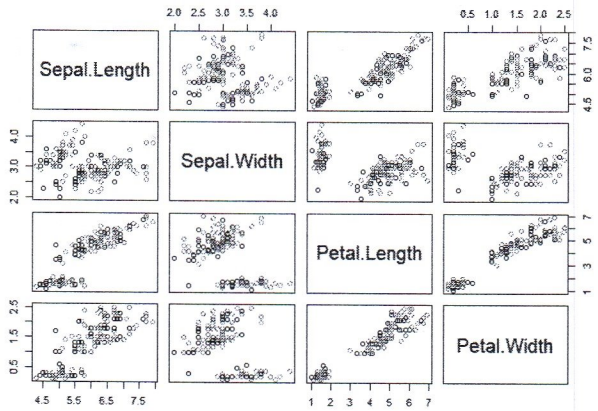


```
i1 <- which(species.name=='setosa')
i2 <- which(species.name=='versicolor')
i3 <- which(species.name=='virginica')
n1 <- length(i1)
n2 <- length(i2)
n3 <- length(i3)
n  <- n1+n2+n3

g  <- length(levels(species.name))
p  <- 4
detach(iris)

# MANOVA
# parametric test:
fit <- manova(as.matrix(iris4) ~ species.name)
summary.manova(fit,test="Wilks")
```

test statistic used

```
##               Df    Wilks approx F num Df den Df    Pr(>F)
## species.name   2 0.023439   199.15      8    288 < 2.2e-16 ***
## Residuals    147
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# How to perform a permutation test in this case?
# Multivariate framework -> We permute the labels associated to each unit
permutation <- sample(1:n)
species.name.perm <- species.name[permutation]
plot(iris4,col=species.name.perm)
```




```
fit.perm <- manova(as.matrix(iris4) ~ species.name.perm)
summary.manova(fit.perm,test="Wilks")
```

```
##                   Df   Wilks approx F num Df den Df Pr(>F)
## species.name.perm   2 0.94786  0.97683      8    288 0.4542
## Residuals         147
```

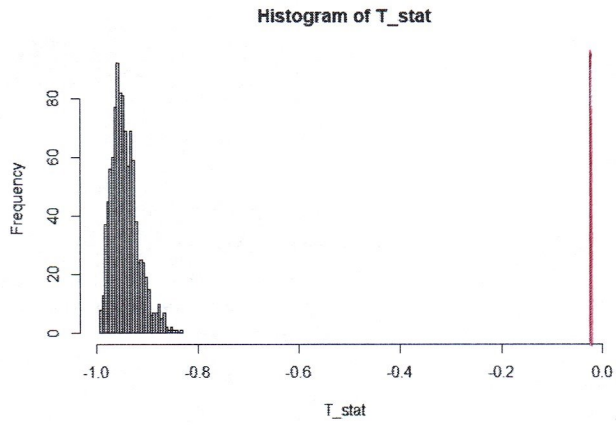
```
# TEST
# Test statistics: Wilks Lambda
T0 <- -summary.manova(fit,test="Wilks")$stats[1,2]
T0
```

```
## [1] -0.02343863
```

high Wilks → we accept H_0

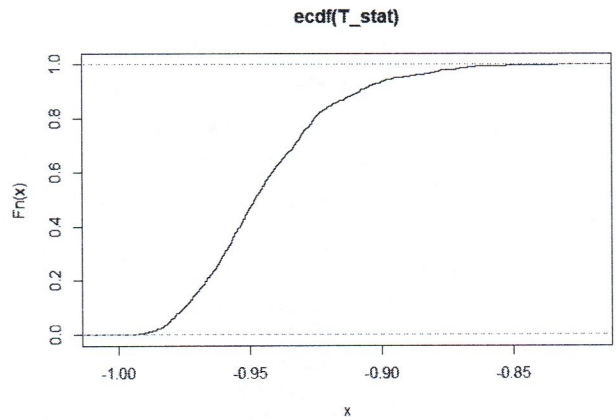
```
# Note that wilk's lambda is significant for small values!
# It is sufficient to change its sign to use it in a permutation test

# Permutations
B <- 1000
T_stat <- numeric(B)

for(perm in 1:B){
  # choose random permutation
  permutation <- sample(1:n)
  species.name.perm <- species.name[permutation]
  fit.perm <- manova(as.matrix(iris4) ~ species.name.perm)
  T_stat[perm] <- -summary.manova(fit.perm,test="Wilks")$stats[1,2]
}

layout(1)
hist(T_stat,xlim=range(c(T_stat,T0)),breaks=30)
abline(v=T0,col=3,lwd=2)
```

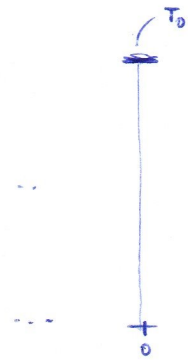
with this → we reject for large Wilks values



⇒ for sure we'll reject H_0

```
plot(ecdf(T_stat))
abline(v=T20,col=3,lwd=4)
```





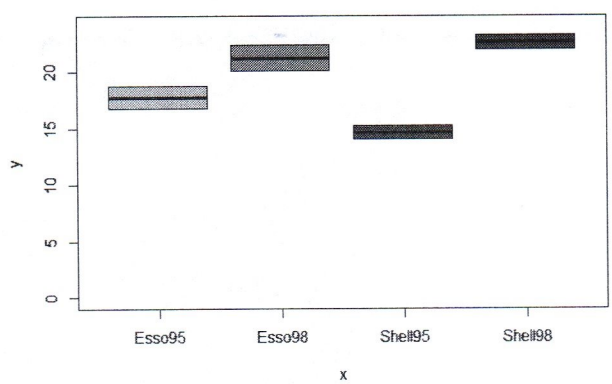

```
# p-value
p_val <- sum(T_stat>=T0)/B
p_val
```

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

```
# ------------------------------------------------------------
# Two-way ANOVA (Pb. 4, 14/09/06)
# Variable:  Distance     [km/l]
# factor1:   Fuel station (0=Esso, 1=ShelL)
# factor2:   Type of fuel (0=95,   1=98)

km           <- c(18.7, 16.8, 20.1, 22.4, 14.0, 15.2, 22.0, 23.3)
station      <- factor(c('Esso','Esso','Esso','Esso','Shell','Shell','Shell','Shell'))
fuel         <- factor(c('95','95','98','98','95','95','98','98'))
station_fuel<- factor(c('Esso95','Esso95','Esso98','Esso98','Shell95','Shell95','Shell98','Shell98'))

M            <- mean(km)
Mstation     <- tapply(km,      station, mean)
Mfuel        <- tapply(km,       fuel, mean)
Mstation_fuel <- tapply(km, station_fuel, mean)


plot(station_fuel, km, col=rainbow(5)[2:5], ylim=c(0,24))
```

unidimensional response variable: fuel efficiency (in km) and two factors (that potentially can influence)

← interaction between the two factors




```
# Parametric test:
summary.aov(aov(km ~ station + fuel + station:fuel))
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## station      1   1.53    1.53   1.018 0.37001
## fuel         1  66.70   66.70  44.357 0.00264 **
## station:fuel 1  10.35   10.35   6.884 0.05857 .
## Residuals    4   6.01    1.50
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

we remove the interaction term

```
# Without interaction
summary.aov(aov(km ~ station + fuel))
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## station      1   1.53    1.53   0.468 0.52440
## fuel         1  66.70   66.70  20.378 0.00632 **
## Residuals    5  16.37    3.27
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

we remove the station

```
# Without station
summary.aov(aov(km ~ fuel))
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## fuel         1   66.7   66.70   22.36 0.00323 **
## Residuals    6   17.9    2.98
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

try to obtain the same with a non parametric approach

(1)

```
# Permutation test
# The model we have to test is the following:
# km = mu + alpha*station + beta*fuel + gamma*station*fuel
# We have 3 different tests:
# 1. factor station   (H0: alpha=0)
# 2. factor fuel      (H0: beta=0)
# 3. interaction      (H0: gamma=0)

# We apply different permutations for developing the different tests!
# We start by testing the interaction:
# H0: gamma=0 against H_1: gamma!=0

# test statistic:
summary.aov(aov(km ~ station + fuel + station:fuel))
```

We use the ANOVA equivalent of Friedman-Lenin method (explained later with linear models)

- fit reduced model
- compute the residual
- permute the residuals → the residuals are permuted under **H0** (under H0 we can permute them and have no effect (they're likelihood invariant under H0))
- summing permuted res with the fitted values
- fit the full model

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## station      1   1.53    1.53   1.018 0.37001
## fuel         1  66.70   66.70  44.357 0.00264 **
## station:fuel 1  10.35   10.35   6.884 0.05857 .
## Residuals    4   6.01    1.50
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
T0_station_fuel <- summary.aov(aov(km ~ station + fuel + station:fuel))[[1]][3,4]
T0_station_fuel
```

```
## [1] 6.883624
```

```
# permutation
# the idea is to permute the residuals under H0:
# km = mu + alpha*station + beta*fuel
# additive model
aov.H0station_fuel <- aov(km ~ station + fuel)
aov.H0station_fuel
```

reduced model

```
## Call:
##    aov(formula = km ~ station + fuel)
##
## Terms:
##                   station     fuel Residuals
## Sum of Squares   1.53125 66.70125  16.36625
## Deg. of Freedom        1        1         5
##
## Residual standard error: 1.809213
## Estimated effects may be unbalanced
```

```
residuals.H0station_fuel <- aov.H0station_fuel$residuals
n <- 8
permutation <- sample(1:n)
residuals.H0station_fuel <- residuals.H0station_fuel[permutation]
# permuted y values:
km.perm.H0station_fuel <- aov.H0station_fuel$fitted + residuals.H0station_fuel
summary.aov(aov(km.perm.H0station_fuel ~ station + fuel + station:fuel))
```

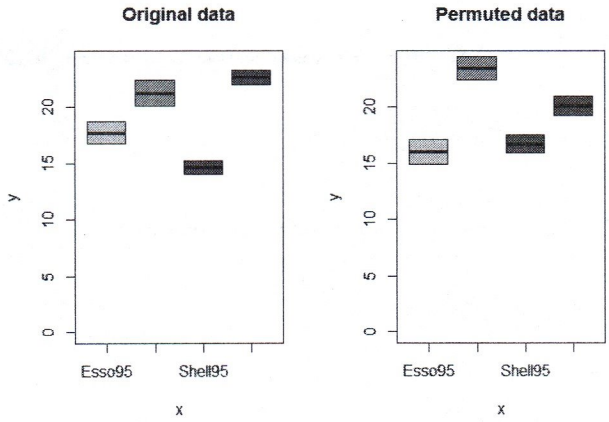
residuals

permutation

$\hat{y}$ + permuted residuals
new model (complete)

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## station      1   3.38    3.38   1.817 0.24891
## fuel         1  58.59   58.59  31.503 0.00495 **
## station:fuel 1   8.30    8.30   4.464 0.10216
## Residuals    4   7.44    1.86
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# How data has changed?
layout(rbind(1:2))
plot(station_fuel, km, col=rainbow(5)[2:5], ylim=c(0,24),main='Original data')
plot(station_fuel, km.perm.H0station_fuel, col=rainbow(5)[2:5], ylim=c(0,24),main='Permuted data')
```




```
# TEST of interaction
B <- 1000
T_station_fuel <- numeric(B)
for(perm in 1:B){
  permutation <- sample(n)
  residuals.H0station_fuel <- residuals.H0station_fuel[permutation]
  km.perm.H0station_fuel <- aov.H0station_fuel$fitted + residuals.H0station_fuel
  T_station_fuel[perm] <- summary.aov(aov(km.perm.H0station_fuel ~ station + fuel + station:fuel))[[1]][3,4]
}

# p-value
sum(T_station_fuel >= T0_station_fuel)/B
```

```
## [1] 0.08
```

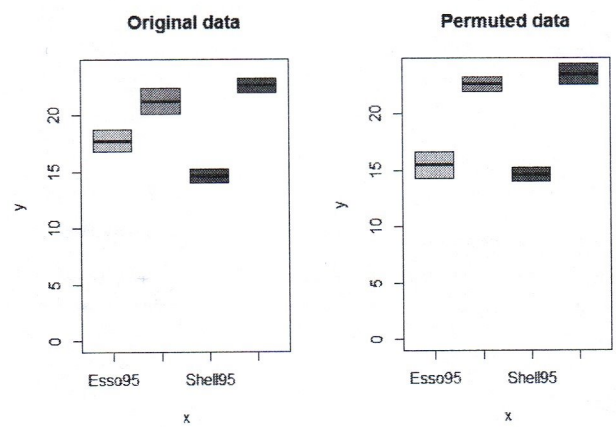
⟹ get rid of the interaction (accept H_0)

(2)

```
# The interaction is not significant.
# We can remove it and perform a test for the two main effects

# TEST OF FACTOR STATION   (H0: alpha=0)
T0_station <- summary.aov(aov(km ~ station + fuel))[[1]][1,4]
# residuals under H0:
# km = mu + beta*fuel
aov.H0station <- aov(km ~ fuel)
residuals.H0station <- aov.H0station$residuals
# permuted y values:
km.perm.H0station <- aov.H0station$fitted + residuals.H0station[permutation]
summary.aov(aov(km.perm.H0station ~ station + fuel))
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## station      1   0.00    0.00    0.00 1.000000
## fuel         1 129.61  129.61   85.87 0.000246 ***
## Residuals    5   7.55    1.51
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# How data has changed?
layout(rbind(1:2))
plot(station_fuel, km, col=rainbow(5)[2:5], ylim=c(0,24),main='Original data')
plot(station_fuel, km.perm.H0station, col=rainbow(5)[2:5], ylim=c(0,24),main='Permuted data')
```



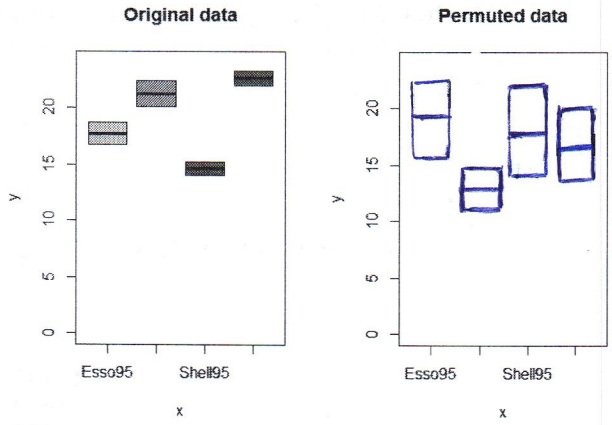
seems very similar.
If this happens for a lot of permutations ⟹ the factor station is not significant

(3)

```
# TEST OF FACTOR FUEL   (H0: beta=0)
T0_fuel <- summary.aov(aov(km ~ station + fuel))[[1]][2,4]
# residuals under H0:
# km = mu + alpha*station
aov.H0fuel <- aov(km ~ station)
residuals.H0fuel <- aov.H0fuel$residuals
# permuted y values:
km.perm.H0fuel <- aov.H0fuel$fitted + residuals.H0fuel[permutation]
summary.aov(aov(km.perm.H0fuel ~ station + fuel))
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## station      1  88.44   88.44  73.520 0.000356 ***
## fuel         1  10.35   10.35   8.605 0.032506 *
## Residuals    5   6.01    1.20
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# How data has changed?
layout(rbind(1:2))
plot(station_fuel, km, col=rainbow(5)[2:5], ylim=c(0,24),main='Original data')
plot(station_fuel, km.perm.H0fuel, col=rainbow(5)[2:5], ylim=c(0,24),main='Permuted data')
```



there are a lot of differences: if this is the case for many permutations ⇒ the factor fuel is significant

(2) + (3)

```
# TEST OF FACTOR STATION ANF TEST OF FACTOR FUEL
# p-values
B <- 1000
T_fuel <- T_station <- numeric(B)
for(perm in 1:B){
  permutation <- sample(n)

  km.perm.H0station <- aov.H0station$fitted + residuals.H0station[permutation]
  T_station[perm] <- summary.aov(aov(km.perm.H0station ~ station + fuel))[[1]][1,4]

  km.perm.H0fuel <- aov.H0fuel$fitted + residuals.H0fuel[permutation]
  T_fuel[perm] <- summary.aov(aov(km.perm.H0fuel ~ station + fuel))[[1]][2,4]
}

sum(T_station >= T0_station)/B
```

(both but separately)

```
## [1] 0.565
```

```
sum(T_fuel >= T0_fuel)/B
```

```
## [1] 0.028
```

```
# Comparison with the parametric test
summary.aov(aov(km ~ station + fuel))
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## station      1   1.53    1.53   0.468 0.52440
## fuel         1  66.70   66.70  20.378 0.00632 **
## Residuals    5  16.37    3.27
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

(4)

```
# We can remove also the factor station
# TEST ON THE FACTOR FUEL
T0_fuel <- summary.aov(aov(km ~ fuel))[[1]][1,4]
# residuals under H0
# km = mu
residuals.H0fuel <- km - M

# Note that in this case, permuting the residuals under H0
# and permuting the data is exactly the same:
permutation <- sample(n)
km.perm.H0fuel <- M + residuals.H0fuel[permutation]
km.perm <- km[permutation]

km.perm.H0fuel
```

→ this time the model (reduced) doesn't have station

```
## [1] 22.0 16.8 14.0 20.1 18.7 15.2 23.3 22.4
```

```
km.perm
```

```
## [1] 22.0 16.8 14.0 20.1 18.7 15.2 23.3 22.4
```

```
# CMC to estimate the p-value
B <- 1000
T_fuel <- numeric(B)
for(perm in 1:B){
  permutation <- sample(n)
  km.perm <- km[permutation]
  T_fuel[perm] <- summary.lm(aov(km.perm ~ fuel ))$f[1]

}
sum(T_fuel >= T0_fuel)/B
```

```
## [1] 0.008
```

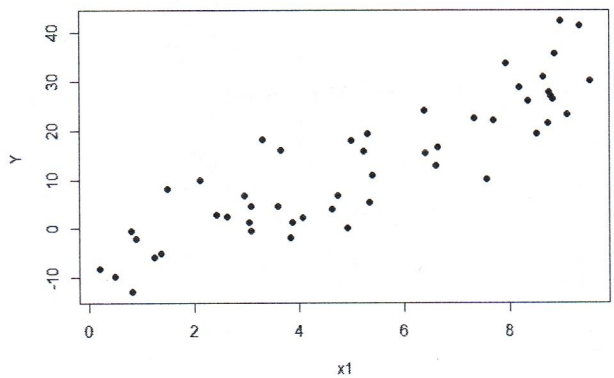
```
### ---------------------------------------------------------------------
### ---------------------------------------------------------------------
### PERMUTATION TEST: Linear models
### ---------------------------------------------------------------------
### ---------------------------------------------------------------------

# Linear models
# Example on simulated data

set.seed(24021979)
n <- 50
# covariate values
x1 <- runif(n,0,10)
x2 <- (1:n)/5
x3 <- rnorm(n,5,5)

# generating model
b0 <- 2
b1 <- 4
b2 <- -2
b3 <- 0
Y <- b0 + b1*x1 + b2*x2 + b3*x3 + runif(n,-5,5)

plot(x1,Y,pch=16)
```

- **Global** test (all regressor): permutation of the response
- **Partial** test (one regressor): ASYMPTOTICALLY EXACT
  we work on the residuals (permutations)

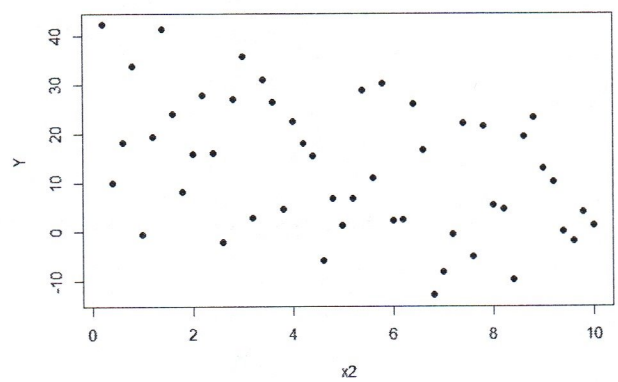
$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \varepsilon$$

we're trying to get rid of the distr. of the error

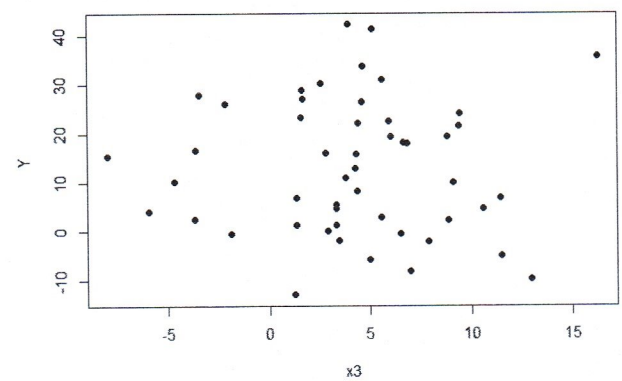
! we're assuming all the errors to have the same distribution, just not gaussian




```
plot(x2,Y,pch=16)
```




```
plot(x3,Y,pch=16)
```



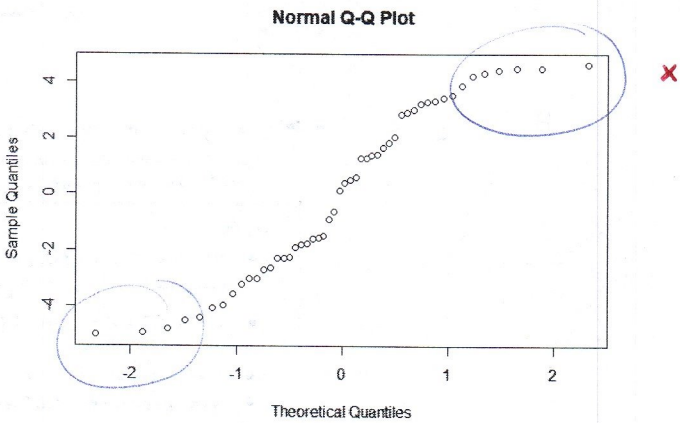
```r
# parametric inference
result <- lm(Y ~ x1 + x2 + x3)
summary(result)
```

(as comparison)

```
##
## Call:
## lm(formula = Y ~ x1 + x2 + x3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0412 -2.5301  0.2557  2.9685  4.6008
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.18206    1.44086   0.126    0.900
## x1           4.17597    0.16073  25.981  < 2e-16 ***
## x2          -1.86077    0.15955 -11.663 2.45e-15 ***
## x3           0.13202    0.09578   1.378    0.175
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.185 on 46 degrees of freedom
## Multiple R-squared:  0.9493, Adjusted R-squared:  0.946
## F-statistic: 287.1 on 3 and 46 DF,  p-value: < 2.2e-16
```

✓ ✓

```r
qqnorm(result$residuals)
```



✗

```r
shapiro.test(result$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  result$residuals
## W = 0.93082, p-value = 0.005935
```

✗

```r
# permutation inference
# we want to perform different tests

# Overall model
# H0: beta1 = beta2 = beta3 = 0
# test statistic
T0_glob <- summary(result)$f[1]
T0_glob
```

(Global)

$H_0: \beta_1 = \beta_2 = \beta_3 = 0$

Note: we can use different test statistics. Moreover, remember that: given a test statistic we can also apply monotonic transformations to the test statistics obtaining tests which are inferencial identical to the previous ones

```
##    value
## 287.1341
```

```r
# permutations
permutazione <- sample(n)
Y.perm.glob <- Y[permutazione]

# in this case permuting the responses or the residuals is the same
res.H0glob <- Y - mean(Y)
Y.perm.glob
```

```
##  [1]  22.48241717  26.38239058  35.74159214   2.21250032   2.40708113
##  [6]  15.92132568  30.09713093  27.05470139  28.79799953  24.14881023
## [11]  19.37519779  -8.25398893  26.04981205  -5.07195264  -0.43453339
## [16]  -1.87633222  15.37941576  18.22361867   8.08065421  23.37314319
## [21]  22.10003708  -5.75465816   1.33931967  41.39818009  19.34967537
## [26]  -0.44439887   6.67252665   3.99175228   2.76803385   9.95027224
## [31]   1.32310819  10.94871942   4.58105052   6.74628679  33.66110255
## [36]  18.01712082  16.52693700  31.04898964  15.87302185  -9.81517813
## [41]  10.04966981  -2.17551423   4.53115582  12.79666281   5.43513788
## [46]  42.28576482   0.04698356 -12.97294939  21.56536855  27.77744131
```

```r
mean(Y) + res.H0glob[permutazione]
```

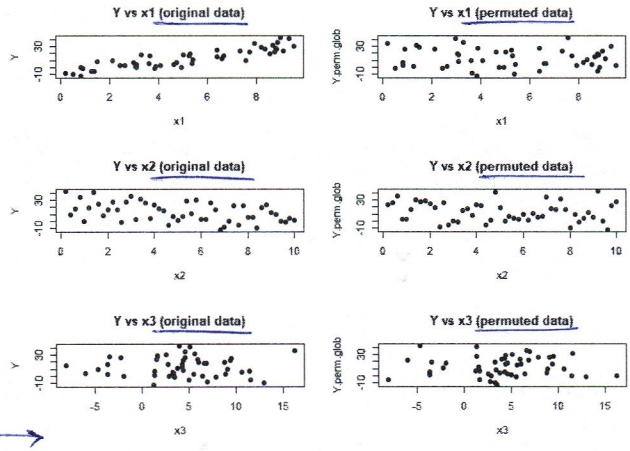
```
##  [1]  22.48241717  26.38239058  35.74159214   2.21250032   2.40708113
##  [6]  15.92132568  30.09713093  27.05470139  28.79799953  24.14881023
## [11]  19.37519779  -8.25398893  26.04981205  -5.07195264  -0.43453339
## [16]  -1.87633222  15.37941576  18.22361867   8.08065421  23.37314319
## [21]  22.10003708  -5.75465816   1.33931967  41.39818009  19.34967537
## [26]  -0.44439887   6.67252665   3.99175228   2.76803385   9.95027224
## [31]   1.32310819  10.94871942   4.58105052   6.74628679  33.66110255
## [36]  18.01712082  16.52693700  31.04898964  15.87302185  -9.81517813
## [41]  10.04966981  -2.17551423   4.53115582  12.79666281   5.43513788
## [46]  42.28576482   0.04698356 -12.97294939  21.56536855  27.77744131
```

(*) we add:

```
pairs(cbind(Y, x1, x2, x3), main = "Original Data")
pairs(cbind(Y.perm.glob, x1, x2, x3), main = "Permuted Data")
```

```
layout(matrix(1:6,nrow=3,byrow=FALSE))
plot(x1,Y,main='Y vs x1 (original data)',pch=16)
plot(x2,Y,main='Y vs x2 (original data)',pch=16)
plot(x3,Y,main='Y vs x3 (original data)',pch=16)
plot(x1,Y.perm.glob,main='Y vs x1 (permuted data)',pch=16)
plot(x2,Y.perm.glob,main='Y vs x2 (permuted data)',pch=16)
plot(x3,Y.perm.glob,main='Y vs x3 (permuted data)',pch=16)
```

here we see that there is a difference with the permutation



here we see the changes

here there are no big changes

(*) →

(Partial)

```
# Test on variable x1
# H0: beta1 = 0

# test statistic
summary(result)$coefficients
```

$H_0: \beta_1 = 0$ → we permute the residuals (we wanted to permute the errors but all we have are the residuals)

```
##               Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)  0.1820590 1.44085541   0.1263548 9.000018e-01
## x1           4.1759699 0.16073115  25.9810872 3.913695e-29
## x2          -1.8607723 0.15955127 -11.6625349 2.451497e-15
## x3           0.1320238 0.09577616   1.3784623 1.747282e-01
```

```
T0_x1 <- abs(summary(result)$coefficients[2,3])
T0_x1
```

```
## [1] 25.98109
```

```
# permutations
# residuals of the reduced model

# reduced model:
# Y = beta0 + beta2*x2 + beta3*x3
regr.H01 <- lm(Y ~ x2 + x3)
residui.H01 <- regr.H01$residuals
residui.H01.perm <- residui.H01[permutazione]

# permuted y:
Y.perm.H01 <- regr.H01$fitted + residui.H01.perm

layout(matrix(1:6,nrow=3,byrow=FALSE))
plot(x1,Y,main='Y vs x1 (original data)',pch=16)
plot(x2,Y,main='Y vs x2 (original data)',pch=16)
plot(x3,Y,main='Y vs x3 (original data)',pch=16)
plot(x1,Y.perm.H01,main='Y vs x1 (permuted data)',pch=16)
plot(x2,Y.perm.H01,main='Y vs x2 (permuted data)',pch=16)
plot(x3,Y.perm.H01,main='Y vs x3 (permuted data)',pch=16)
```

Idea:

- we fit the reduced model (model without X_1)
- we compute the fitted values of the model
- we compute the residuals
- we permute the residuals
- we sum the permuted residuals to the original fitted values and generate another dataset
- we introduce now X_1 and we fit the complete model

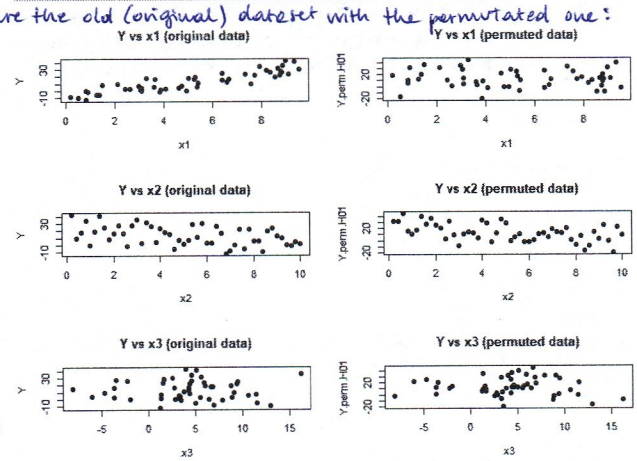
Test statistic: $\hat{\beta}_1$ / $\hat{\beta}_1^2$ / $std(\hat{\beta}_1)$ ...

Friedman-Levin idea

let's compare the old (original) dataset with the permutated one:

completely changed →

UNCHANGED! →



Idea 2:
permute the residuals of the complete model, so:

- fit the model with X_1
- fitted values
- residuals
- permutation of the residuals
- fitted values + permuted residual

(Partial)

```
# Test on variable x2
# H0: beta2 = 0

# test statistic
summary(result)$coefficients
```

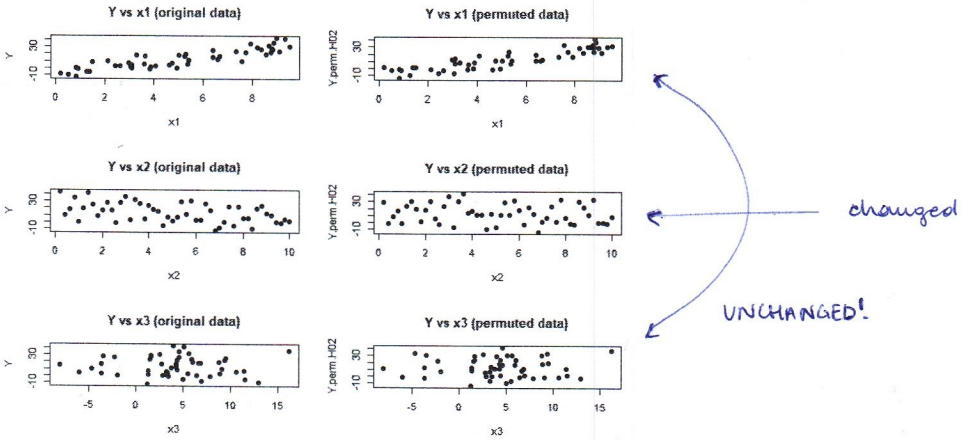
$H_0: \beta_2 = 0$

```
##               Estimate Std. Error     t value     Pr(>|t|)
## (Intercept)  0.1820590 1.44085541   0.1263548 9.000018e-01
## x1           4.1759699 0.16073115  25.9810872 3.913695e-29
## x2          -1.8607723 0.15955127 -11.6625349 2.451497e-15
## x3           0.1320238 0.09577616   1.3784623 1.747282e-01
```

```
T0_x2 <- abs(summary(result)$coefficients[3,3])
T0_x2
```

```
## [1] 11.66253
```

```
# permutations
# residuals of the reduced model

# reduced model:
# Y = beta0 + beta1*x1 + beta3*x3
regr.H02 <- lm(Y ~ x1 + x3)
residui.H02 <- regr.H02$residuals
residui.H02.perm <- residui.H02[permutazione]

# permuted y:
Y.perm.H02 <- regr.H02$fitted + residui.H02.perm

layout(matrix(1:6,nrow=3,byrow=FALSE))
plot(x1,Y,main='Y vs x1 (original data)',pch=16)
plot(x2,Y,main='Y vs x2 (original data)',pch=16)
plot(x3,Y,main='Y vs x3 (original data)',pch=16)
plot(x1,Y.perm.H02,main='Y vs x1 (permuted data)',pch=16)
plot(x2,Y.perm.H02,main='Y vs x2 (permuted data)',pch=16)
plot(x3,Y.perm.H02,main='Y vs x3 (permuted data)',pch=16)
```



(Partial)

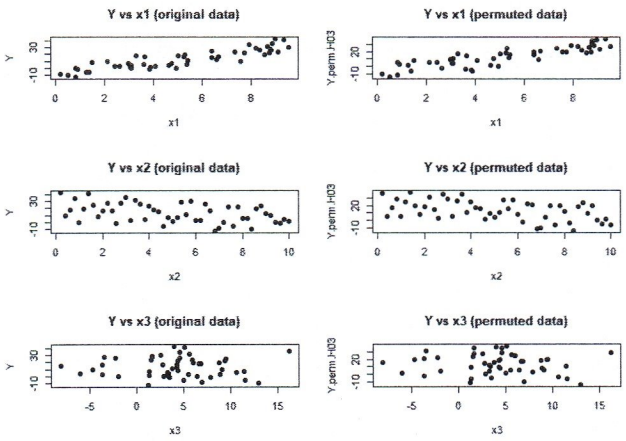
```
# Test on variable x3
# H0: beta3 = 0

# test statistic
summary(result)$coefficients
```

```
##               Estimate Std. Error     t value     Pr(>|t|)
## (Intercept)  0.1820590 1.44085541   0.1263548 9.000018e-01
## x1           4.1759699 0.16073115  25.9810872 3.913695e-29
## x2          -1.8607723 0.15955127 -11.6625349 2.451497e-15
## x3           0.1320238 0.09577616   1.3784623 1.747282e-01
```

```
T0_x3 <- abs(summary(result)$coefficients[4,3])
T0_x3
```

```
## [1] 1.378462
```

```
# permutations
# residuals of the reduced model

# reduced model:
# Y = beta0 + beta1*x1 + beta2*x2
regr.H03 <- lm(Y ~ x1 + x2)
residui.H03 <- regr.H03$residuals
residui.H03.perm <- residui.H03[permutazione]

# permuted y:
Y.perm.H03 <- regr.H03$fitted + residui.H03.perm

layout(matrix(1:6,nrow=3,byrow=FALSE))
plot(x1,Y,main='Y vs x1 (original data)',pch=16)
plot(x2,Y,main='Y vs x2 (original data)',pch=16)
plot(x3,Y,main='Y vs x3 (original data)',pch=16)
plot(x1,Y.perm.H03,main='Y vs x1 (permuted data)',pch=16)
plot(x2,Y.perm.H03,main='Y vs x2 (permuted data)',pch=16)
plot(x3,Y.perm.H03,main='Y vs x3 (permuted data)',pch=16)
```



UNCHANGED!

→ changed a little

```
# p-values of the tests
B <- 1000
T_H0glob <- T_H01 <- T_H02 <- T_H03 <- numeric(B)

for(perm in 1:B){
  permutazione <- sample(n)

  Y.perm.glob <- Y[permutazione]
  T_H0glob[perm] <- summary(lm(Y.perm.glob ~ x1 + x2 + x3))$f[1]

  residui.H01.perm <- residui.H01[permutazione]
  Y.perm.H01 <- regr.H01$fitted + residui.H01.perm
  T_H01[perm] <- abs(summary(lm(Y.perm.H01 ~ x1 + x2 + x3))$coefficients[2,3])

  residui.H02.perm <- residui.H02[permutazione]
  Y.perm.H02 <- regr.H02$fitted + residui.H02.perm
  T_H02[perm] <- abs(summary(lm(Y.perm.H02 ~ x1 + x2 + x3))$coefficients[3,3])

  residui.H03.perm <- residui.H03[permutazione]
  Y.perm.H03 <- regr.H03$fitted + residui.H03.perm
  T_H03[perm] <- abs(summary(lm(Y.perm.H03 ~ x1 + x2 + x3))$coefficients[4,3])

}
```

(we permute all the singular cases here, it's still the same as doing it separately)

we fit 4 different models every time

```
sum(T_H0glob>=T0_glob)/B
```

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

✓

```
sum(T_H01>=T0_x1)/B
```

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

✓

```
sum(T_H02>=T0_x2)/B
```

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

✓

```
sum(T_H03>=T0_x3)/B
```

```
## [1] 0.168
```

✗

```
# comparison with the parametric test
summary(result)
```

```
##
## Call:
## lm(formula = Y ~ x1 + x2 + x3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0412 -2.5301  0.2557  2.9685  4.6008
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.18206    1.44086   0.126    0.900
## x1           4.17597    0.16073  25.981  < 2e-16 ***
## x2          -1.86077    0.15955 -11.663 2.45e-15 ***
## x3           0.13202    0.09578   1.378    0.175
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.185 on 46 degrees of freedom
## Multiple R-squared:  0.9493, Adjusted R-squared:  0.946
## F-statistic: 287.1 on 3 and 46 DF,  p-value: < 2.2e-16
```

✓ ✓ ✗ ✓