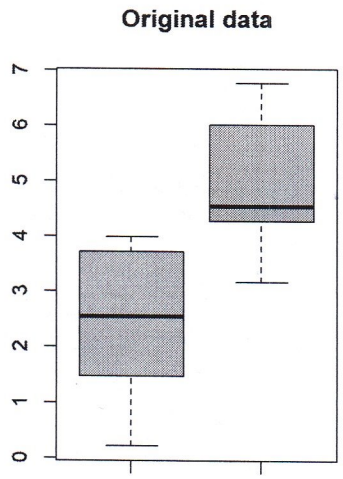
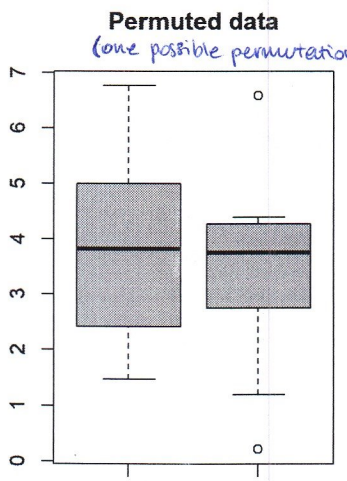
01 Permutation Tests - Two Independent Populations

```
# 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 <- 10
n2 <- 10
n  <- n1 + n2

# Simulation:
set.seed(240279)
x1 <- runif(n1,0,4)      ~ U([0,4])
x2 <- runif(n2,0,4)+3    ~ U([3,7])

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')
```





(one possible permutation)

We see that by permuting data the two distributions seems to change. If this change appears very often while permuting we end with the conclusion that the original data were kind of "extreme" ($\Rightarrow H_1$)

$\Rightarrow$ We want to compare the boxplot of the original data with (theoretically) all the boxplots of all the possible permutations. We want an indicator of how the boxplots are different, for instance we can use the absolute value of the difference of the means ($|\bar{x}_1 - \bar{x}_2|$)

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

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

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

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

(a lot smaller than the original one)

```
# 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
```

= # possible permutations

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

the test statistic is invariant w.r.t. the permutations within each sample ⟹ number of possible stats = $\frac{n!}{2(n_1!)(n_2!)}$

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

this 2 is because the test statistic has a absolute value and so also the swapping of the samples will generate the same value of the test statistic ($|\bar{x}_1 - \bar{x}_2| = |\bar{x}_2 - \bar{x}_1|$)

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

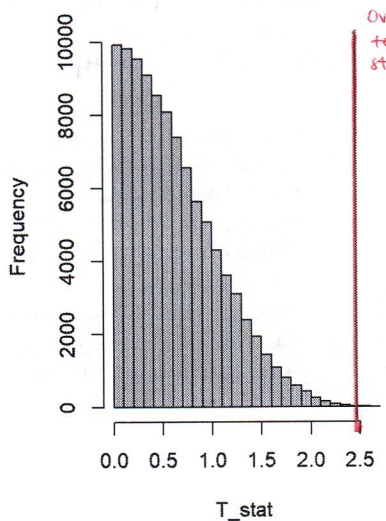
most extreme case, where the samples we have is the most extreme one. This is also the resolution of our p-value. The p-value can assume only: $\frac{1}{n!/2n_1!n_2!}, \frac{2}{n!/2n_1!n_2!}, \dots$

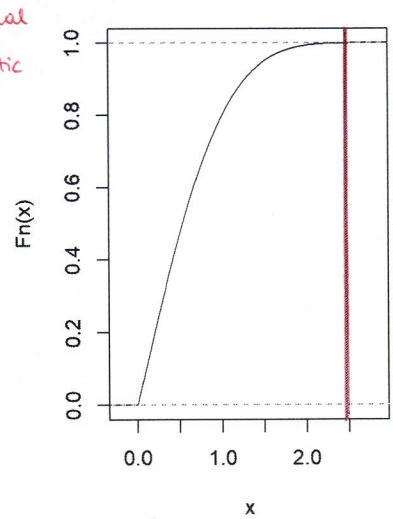
```
## [1] 1.082509e-05
```

The resolution of the p-value can be an issue: if n is too small (and consequently also n_1 and n_2) the p-value will be able to assume only few discrete values. We want to avoid the discretization of the p-value.
(Case of low resolution of p-value)

```
# 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)
```

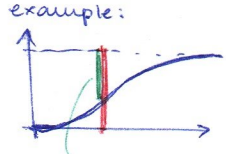
estimate of the distribution of the test statistic obtained by permuting data in all the possible permutations)





cumulative distribution function of the permutational distribution

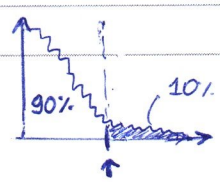
the p-value is the length of the segment (on the red line) going from the black line (ecdf) to the "---" line:
example:



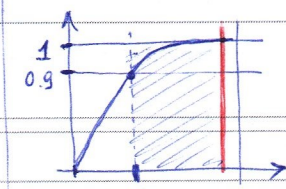
the p-value is the length of this segment

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

equivalent to go to the histogram of Tstat and look for the point that:



If we want a critical region of the level of 10% (test with prob. of type I error equal to 10%):



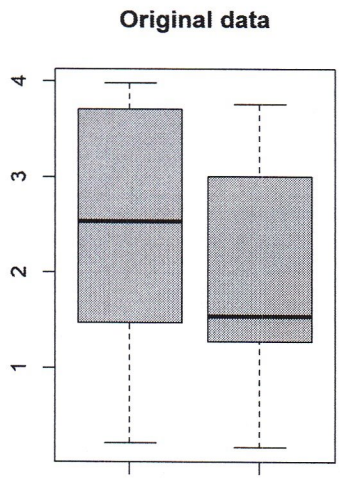
from this value it begins the critical region

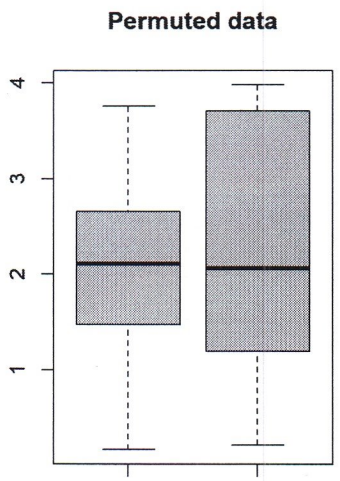
```
## [1] 3e-04
```

⟹ our original sample is very extreme

⟹ we reject H_0

```
### ------------------------------------------------------------------------
### 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')
```





this time the difference between the boxplots is not that big. if this is continued by a lot of permutations than our original sample is not that extreme.

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

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

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

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

→ the permuted sample is even more extreme that the original

```
# 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
```

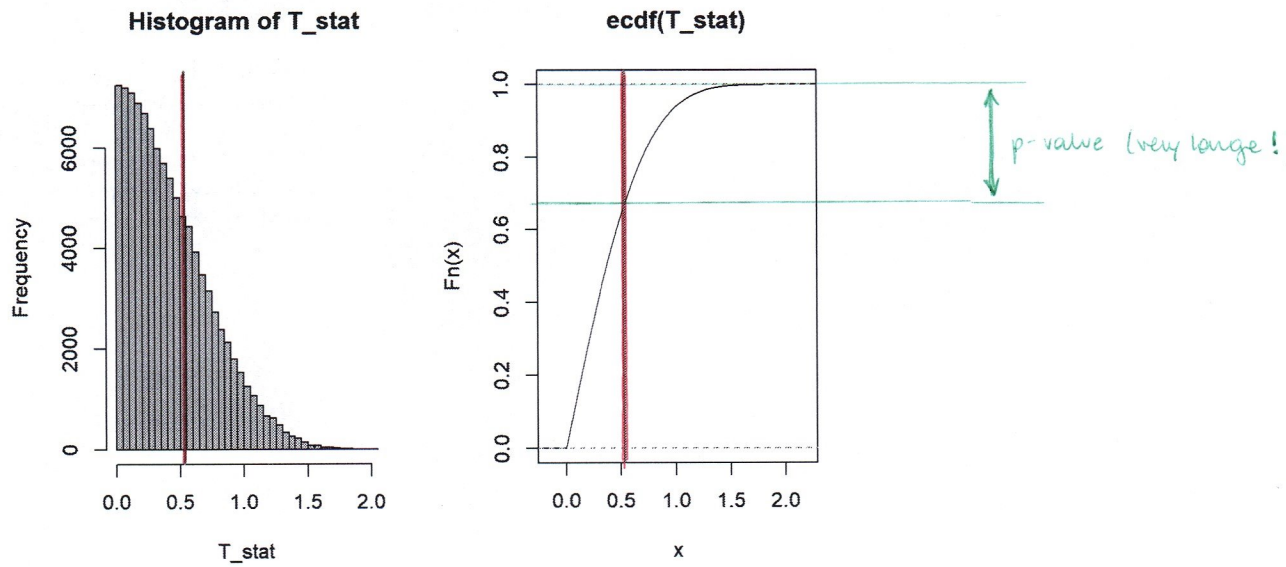
```
# Number of distinct values of T*:
factorial(n)/(2*factorial(n1)*factorial(n2))
```

```
## [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
```

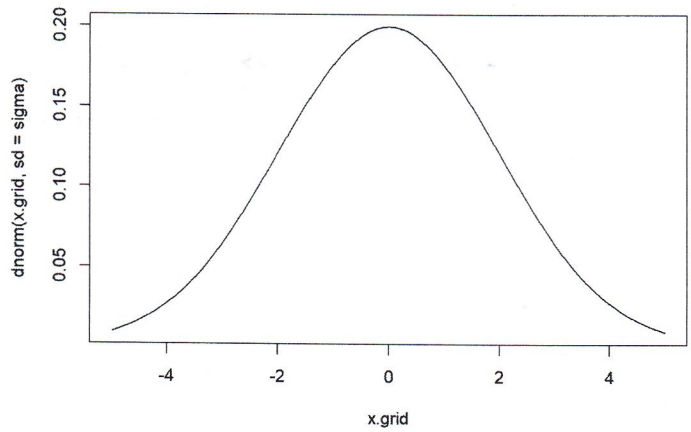
→ this means that 33% of the times that we permuted the data we got a value of the test statistic more extreme than the original one

Notice: as we already pointed out in the theory, the distribution of the T_stat depends on the data! In the two examples we have the same procedure but different data
⟹ the two histograms of T_stat are different!
(one goes to zero ~ 2.5, the other ~ 2.0 / 1.8)

20/10/2020

```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Nonparametric Inference
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
# Let's reprise again why should I use a nonparametric test, instead of a parametric one...
# the idea is that the real probability of committing a type I error (alpha...) should be
# less or equal of the nominal one...

### ------------------------------------------------------------------------
### Assess Empirical Level of Type-I error
### ------------------------------------------------------------------------
# Let's work with 2 sample t-tests
library(progress)  <- Useful for the progress bars (for computations)

rm(list=ls())
alpha = 0.05
n     = 10
B     = 1000
seed  = 2781991
sigma = 2

### Normal data
### ------------------------------------------------------------------------
x.grid <- seq(-5, 5, by=0.01)
plot(x.grid, dnorm(x.grid,sd=sigma), type='l')
```

Goal: we want to compare two samples from a given distribution (we want to compare their mean)

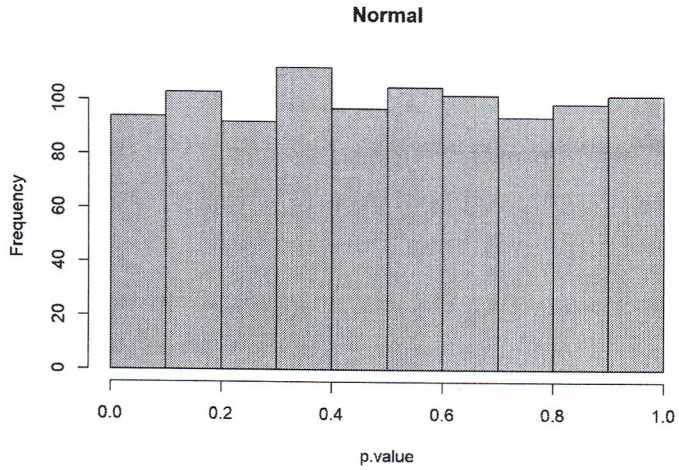


We estimate the distribution of the p-values in these settings:

```
p.value <- numeric(B)

pb = progress_bar$new(total=B)
pb$tick(0)
```

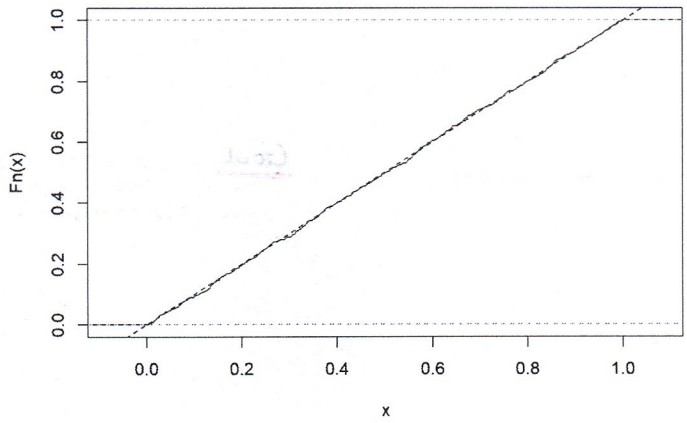
```
set.seed(seed)
for(j in 1:B){
  x1 = rnorm(n,sd=sigma)
  x2 = rnorm(n,sd=sigma)
  p.value[j] = t.test(x1,y=x2)$p.value
pb$tick()
  }
```

```
hist(p.value, main = 'Normal')
```



The p-values are distributed as a uniform (and so the test is calibrated)

```
plot(ecdf(p.value), main = 'Normal')
abline(0,1, lty=2, col='red')
```



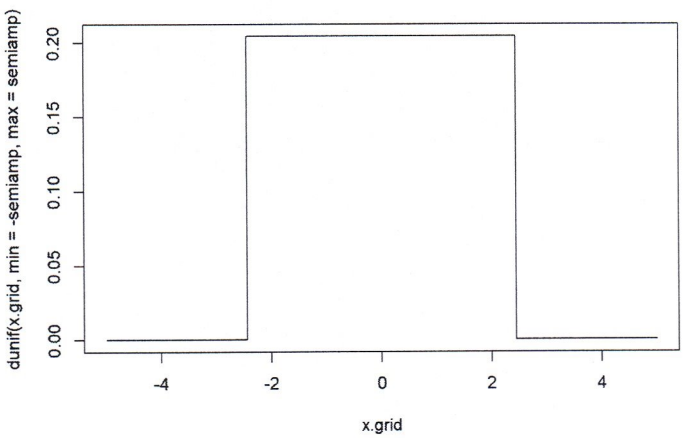
Empirical cumulative distribution function:

```
estimated.alpha <- sum(p.value < alpha)/B
c(estimated.alpha - sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975),
  estimated.alpha,
  estimated.alpha + sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975))
```

```
## [1] 0.03388273 0.04700000 0.06011727
```

Empirical type I error rate ⟹ We can't refuse the hypothesis that the empirical level is equal to the theoretical one

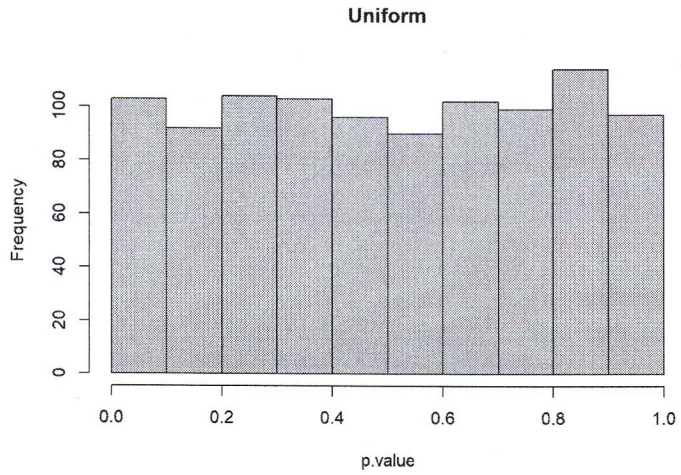
```
### Leptocurtic data (e.g. Uniform)
### ------------------------------------------------------------------------------
# Let's keep the same variance, for uniformity...
semiamp = sqrt(6)
plot(x.grid, dunif(x.grid,min=-semiamp,max=semiamp), type='l')
```




```
p.value <- numeric(B)
pb=progress_bar$new(total=B)
pb$tick(0)
```

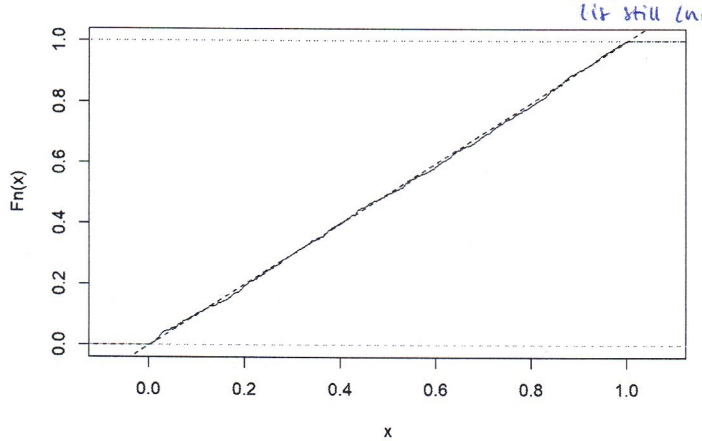
```
set.seed(seed)
for(j in 1:B){
  x1 = runif(n,min=-semiamp,max=semiamp)
  x2 =  runif(n,min=-semiamp,max=semiamp)
  p.value[j] = t.test(x1,y=x2)$p.value
  pb$tick()
}
```

```
hist(p.value, main = 'Uniform')
```



The uniformity of the p-values still stands (a bit)

```
plot(ecdf(p.value), main = 'Uniform')
abline(0,1, lty=2, col='red')
```



Empirical cumulative distribution function (it still (nearly) okay)

```
estimated.alpha <- sum(p.value < alpha)/B
c(estimated.alpha - sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975),
  estimated.alpha,
  estimated.alpha + sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975))
```

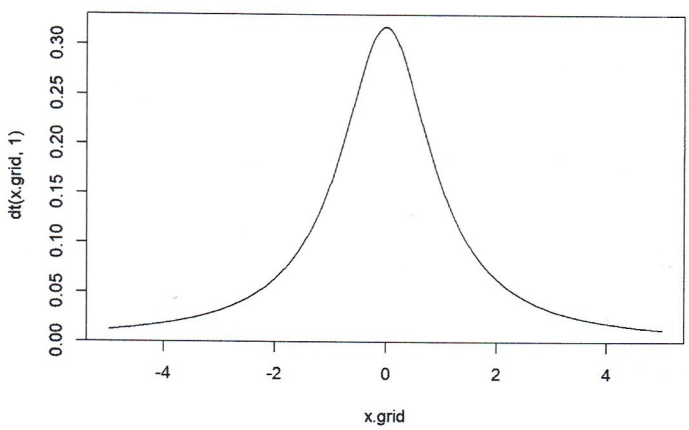
```
## [1] 0.0408699 0.0550000 0.0691301
```

```
# tends not to be a problem with "thin-tailed" data... but!
```

```
### Platicurtic data (e.g. Student's t)
### --------------------------------------------------------------------------
x.grid <- seq(-5, 5, by=0.01)
plot(x.grid, dt(x.grid, 1), type='l')
```

(distributions with fat tails)

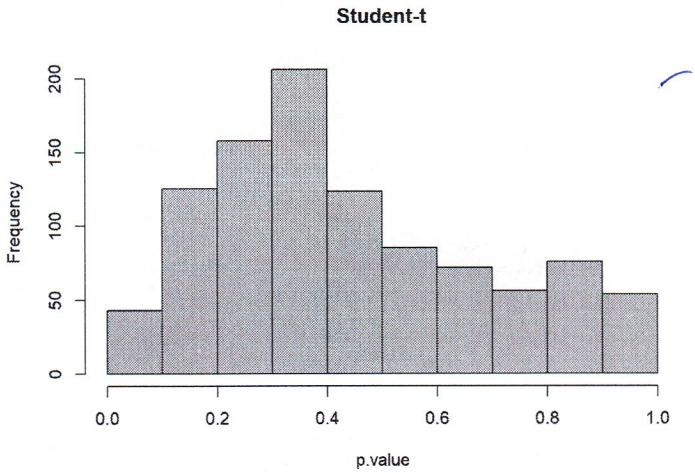
t-Student distribution:



```
p.value <- numeric(B)
pb=progress_bar$new(total=B)
pb$tick(0)
```

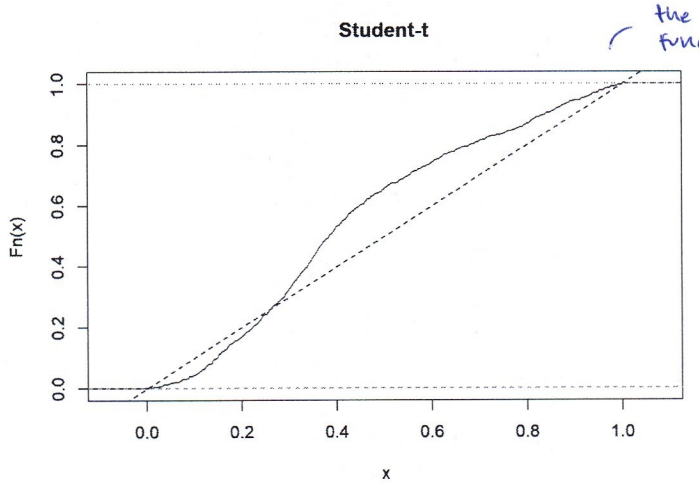
```
set.seed(seed)
for(j in 1:B){
  x.1 = rt(n, 1)
  x.2 = rt(n,1)
  p.value[j] = t.test(x.1,y=x.2)$p.value
  pb$tick()
}
```

```
hist(p.value, main = 'Student-t')
```



Here things go really bad.

```
plot(ecdf(p.value), main = 'Student-t')
abline(0,1, lty=2, col='red')
```



the empirical cumulative distribution function underlines that it's sooo bad

```
estimated.alpha <- sum(p.value < alpha)/B
c(estimated.alpha - sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975),
  estimated.alpha,
  estimated.alpha + sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975))
```

```
## [1] 0.009759756 0.018000000 0.026240244
```

very very low

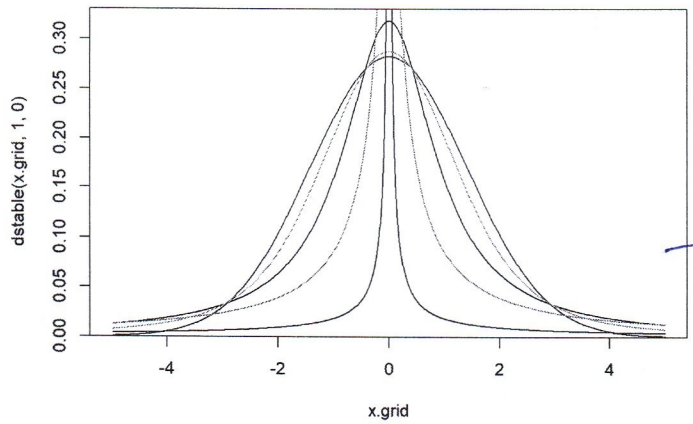
```
# I can actually replicate this with several cases...
# install.packages('stabledist')
library(stabledist)
```

← package used for simulate from "stable" distributions

```
# stabledist is a quite useful package used to simulate from "stable" distributions, where a
# stable distribution intended is in the Levy sense: i.e. if a linear combination of two
# variables generated from a stable distr  P is still distributed as P.
# Parametrisation is done with:
#   alpha = stability, from 0 to 2
#   beta  = skewness parameter
#   mu    = "mean", mean is actually defined defined only for alpha>1
#   variance is defined only for alpha=2

# very useful to simulate heavy tailed data
# special cases are...
```

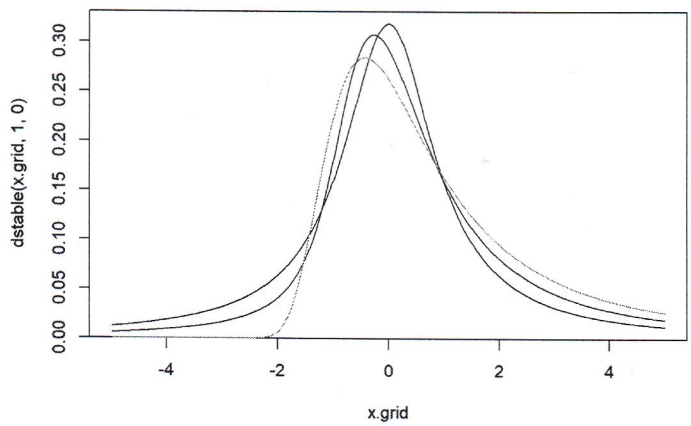
(1 → Cauchy, 2 → normal)

```
x.grid <- seq(-5, 5, by=0.01)
plot(x.grid, dstable(x.grid,1,0), type='l') #cauchy
lines(x.grid, dstable(x.grid,2,0), type='l',col="red") #normal
lines(x.grid, dstable(x.grid,1.5,0), type='l',col="orange")
lines(x.grid, dstable(x.grid,0.5,0), type='l',col="green")
lines(x.grid, dstable(x.grid,0.1,0), type='l',col="blue")
```



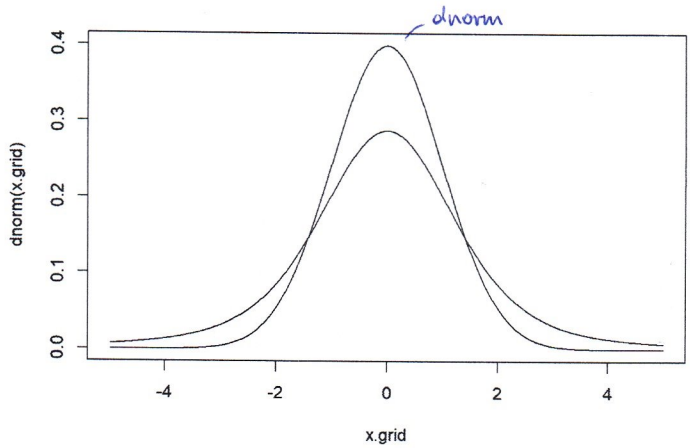
we have some heavy tails distributions that we can use to check some results (to check the validity of some results under "extreme" distributions)

```
plot(x.grid, dstable(x.grid,1,0), type='l') #cauchy
lines(x.grid, dstable(x.grid,1,0.5), type='l',col="red") #normal
lines(x.grid, dstable(x.grid,1,1), type='l',col="orange")
```



1.

```
# Let's try to run a test on a stable with alpha=1.5 (mean, no variance)
# and on a alpha=0.5 (no mean, no variance)

x.grid <- seq(-5, 5, by=0.01)
plot(x.grid, dnorm(x.grid), type='l')
lines(x.grid, dstable(x.grid,1.5,0), type='l',col='red')
```



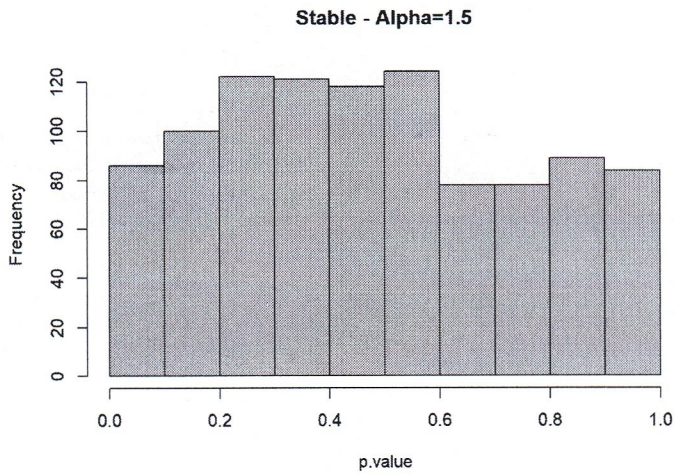
```
p.value <- numeric(B)
pb=progress_bar$new(total=B)
pb$tick(0)
```

} this is for the progress bar

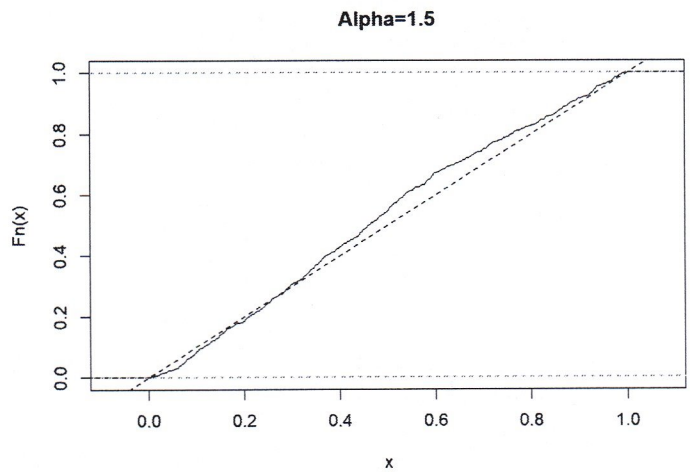
```
set.seed(seed)
for(j in 1:B){
  x.1 = rstable(n,1.5,0)
  x.2 = rstable(n,1.5,0)
  p.value[j] = t.test(x.1,y=x.2)$p.value
}
```

if we add: pb$tick() ← it prints the progress bar

```
hist(p.value, main = 'Stable - Alpha=1.5')
```




```
plot(ecdf(p.value), main = 'Alpha=1.5')
abline(0,1, lty=2, col='red')
```



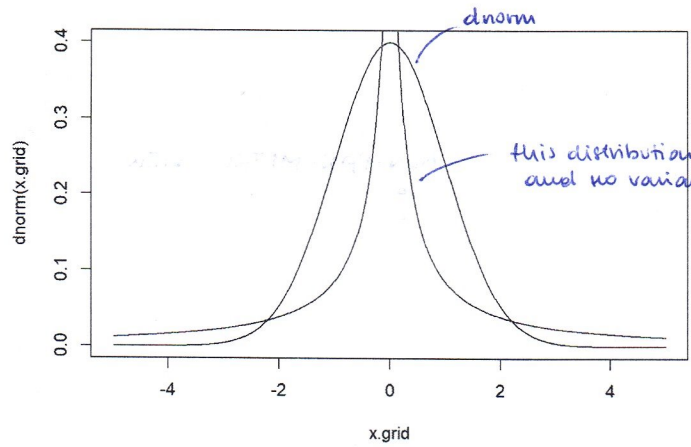

```
estimated.alpha <- sum(p.value < alpha)/B
c(estimated.alpha - sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975),
  estimated.alpha,
  estimated.alpha + sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975))
```

```
## [1] 0.01532345 0.02500000 0.03467655
```

2.

```
plot(x.grid, dnorm(x.grid), type='l')
lines(x.grid, dstable(x.grid,.5,0), type='l',col='red')
```

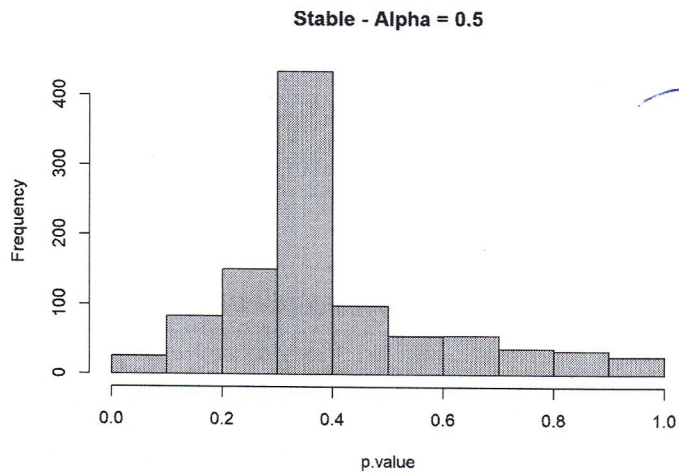
Example for which we'll try nonparametric settings := **Ex.**



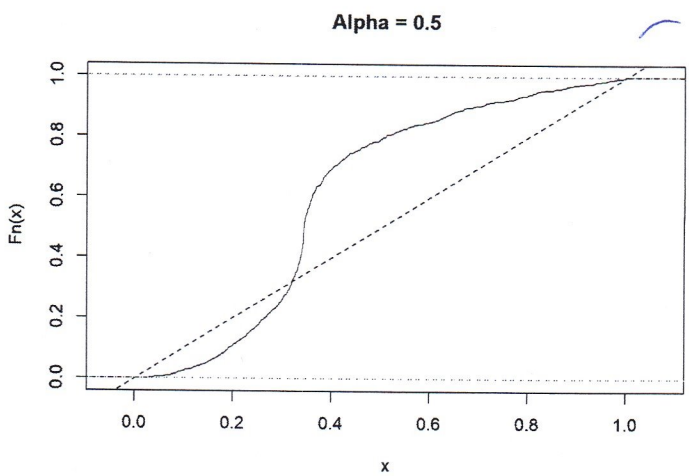

```
p.value <- numeric(B)
pb=progress_bar$new(total=B)
pb$tick(0)
```

```
set.seed(seed)
for(j in 1:B){
  x.1 = rstable(n,.5,0)
  x.2 = rstable(n,.5,0)
  p.value[j] = t.test(x.1,y=x.2)$p.value
  pb$tick()
}
```

```
hist(p.value, main = 'Stable - Alpha = 0.5')
```




```
plot(ecdf(p.value), main = 'Alpha = 0.5')
abline(0,1, lty=2, col='red')
```



```
estimated.alpha <- sum(p.value < alpha)/B
c(estimated.alpha - sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975),
  estimated.alpha,
  estimated.alpha + sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975))
```

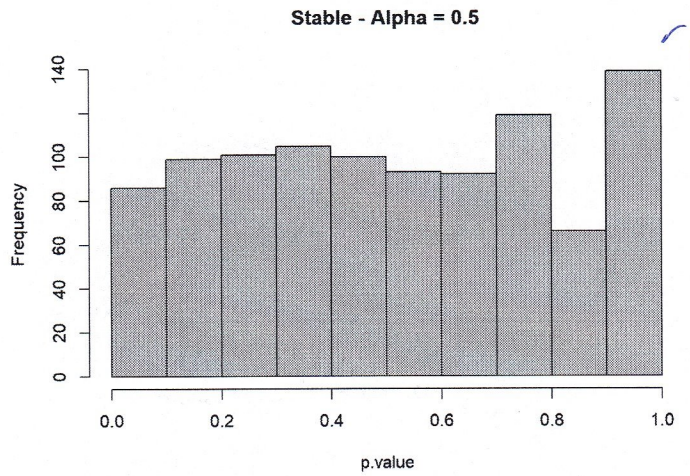
```
## [1] 0.001213513 0.006000000 0.010786487
```

```
# let's keep this last example as a testbed...
# t-test goes bananas, does not cover at all.
# let's try to compare the situation with nonparametric tests...
# let's start with Wilcoxon...

p.value <- numeric(B)
pb=progress_bar$new(total=B)
pb$tick(0)
```

NON-PARAMETRIC VIEW
- Wilkoxon test
- Permutational test

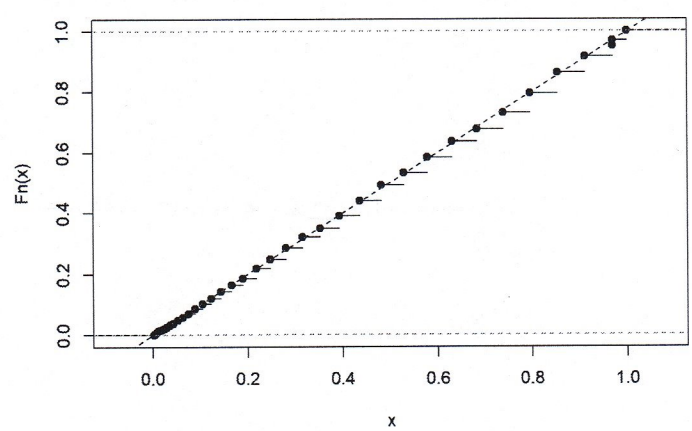
```
set.seed(seed)
for(j in 1:B){
  x.1 = rstable(n,.5,0)
  x.2 = rstable(n,.5,0)
  p.value[j] = wilcox.test(x.1,y=x.2,correct = T)$p.value
  pb$tick()
}
```

```
hist(p.value, main = 'Stable - Alpha = 0.5')
```



still some issues in the calibration (but it's just something related to the quantization of the p-value)

```
plot(ecdf(p.value), main = 'Alpha = 0.5')
abline(0,1, lty=2, col='red')
```



the p-value is quantized, so it's not a continuous function like before (≠ t-test) (the issues with the histogram are related to that)

```
estimated.alpha <- sum(p.value < alpha)/B
c(estimated.alpha - sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975),
  estimated.alpha,
  estimated.alpha + sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975))
```

```
## [1] 0.02530064 0.03700000 0.04869936
```

← way better than the t-test

Permutation test (MEAN)

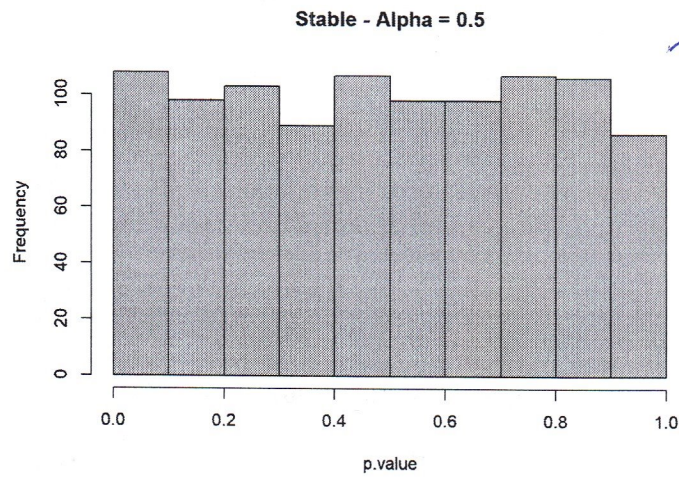
```r
# quite unfortunately for you, there are no good implementations in R for permutation tests...
# so we need to "weaponise" prof.
# Vantinis code
perm_t_test=function(x,y,iter=1e4){
  T0       = abs(mean(x)-mean(y))
  T_stat   = numeric(iter)
  x_pooled = c(x,y)
  n        = length(x_pooled)
  n1       = length(x)
  for(perm in 1:iter){
    # 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))
    }
  # p-value
  p_val <- sum(T_stat>=T0)/iter
  return(p_val)
}
```

test based on the absolute difference of the two means

```r
p.value <- numeric(B)
pb=progress_bar$new(total=B)
pb$tick(0)
```

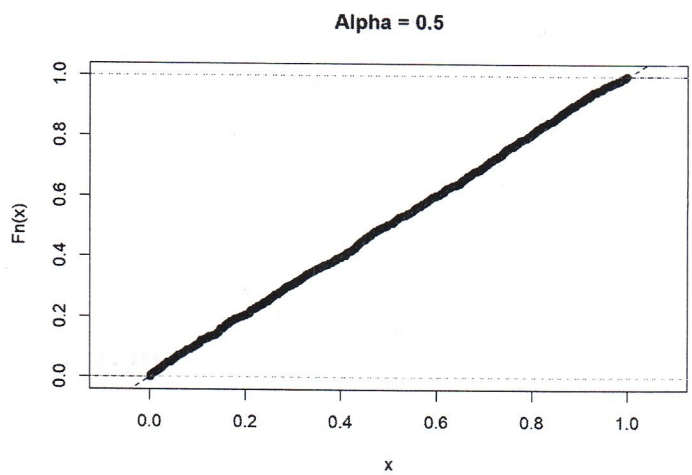
```r
set.seed(seed)
for(j in 1:B){
  x.1 = rstable(n,.5,0)
  x.2 = rstable(n,.5,0)
  p.value[j] = perm_t_test(x.1,x.2, iter=1000)
  pb$tick()
}
```

```r
hist(p.value, main = 'Stable - Alpha = 0.5')
```



the test is actually well calibrated

```r
plot(ecdf(p.value), main = 'Alpha = 0.5')
abline(0,1, lty=2, col='red')
```



```
estimated.alpha <- sum(p.value < alpha)/B
c(estimated.alpha - sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975),
  estimated.alpha,
  estimated.alpha + sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975))
```

```
## [1] 0.04174957 0.05600000 0.07025043
```

→ we cannot refute the hypothesis of the empirical α being equal to the theoretical one

Conclusion: moving under nonparametric settings we improved and we're now able to test something that we were not able to test before with a control of the type I error

We saw that the t-test for EX. went crazy, the (nonparametric) Wilcoxon was already better, while the perm. was good.

```
### ------------------------------------------------------------------------
### Assess Empirical Level of Type-II error
### ------------------------------------------------------------------------
#How to choose between the two (at Least in this specific setting?) Power calculations!
#i.e. we fix the data and the significance level, cycle over a grid of effect sizes, and see what is the most powerful in id
entifying things...

delta_grid=c(1,2,3,4,5)
set.seed(seed)
power_wilcox=numeric(length(delta_grid))
for(ii in 1:length(delta_grid)){
      p.value <- numeric(B)
      pb=progress_bar$new(total=B)
      pb$tick(0)
      delta=delta_grid[ii]
      for(j in 1:B){
        x.1 = rstable(n,.5,0)
        x.2 = rstable(n,.5,0,delta=delta)
        p.value[j] <- wilcox.test(x.1,y=x.2,correct = T)$p.value
        pb$tick()
      }
      estimated.power <- sum(p.value < alpha)/B
      power_wilcox[ii]=estimated.power
}
```

we want to be able to control the α level but also the power of a test!

Same settings as before, but this time one distribution is shifted w.r.t. the other.
(we shift by 1, then by 2, then by 3, 4 and 5)

Power of a test := $1-\beta$

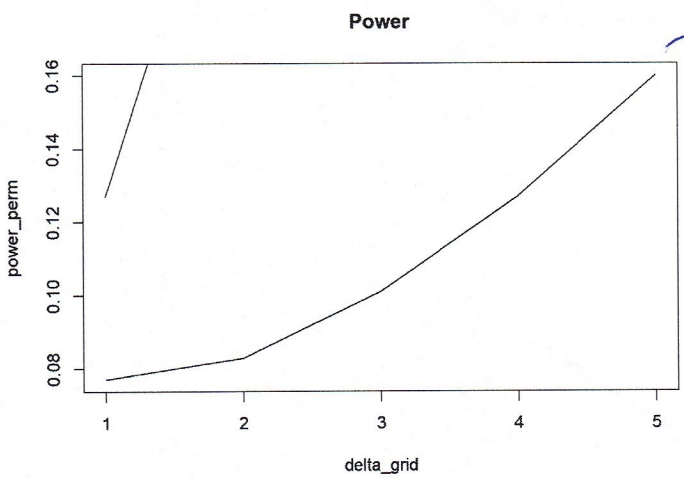
where β = IP(type II error rate)
$\Updownarrow$ IP(false negative)
$\Updownarrow$ IP(accept H_0 when it's false)

even if there is a difference we're not capturing it with our test. We want power ~ 1 (we want to be able to capture differences if there are)

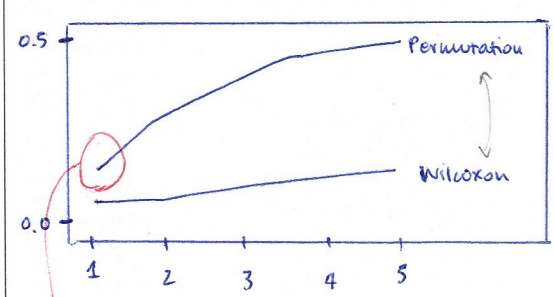
→ if we add here: "power_wilcox" we get [0.127, 0.243, 0.324, 0.408, 0.425] and the values are very low! This means that the probability of type II error is actually very high, not good!

```
power_perm=numeric(length(delta_grid))
set.seed(seed)
for(ii in 1:length(delta_grid)){
  p.value <- numeric(B)
  pb=progress_bar$new(total=B)
  pb$tick(0)
  delta=delta_grid[ii]
  for(j in 1:B){
    x.1 = rstable(n,.5,0)
    x.2 = rstable(n,.5,0,delta=delta)
    p.value[j] = perm_t_test(x.1,x.2,iter=1000)
    pb$tick()
    }
  estimated.power <- sum(p.value < alpha)/B
  power_perm[ii]=estimated.power
}
```

```
plot(delta_grid,power_perm,type='l', main='Power')
lines(delta_grid,power_wilcox,col='red')
```



Setting up the right limits:



However here it still too small. That's probably because we're using as a test statistic something that consider the mean, when our distribution DOES NOT ALLOW A MEAN!

→ We change the test statistic into something that consider the MEDIAN instead of the MEAN.

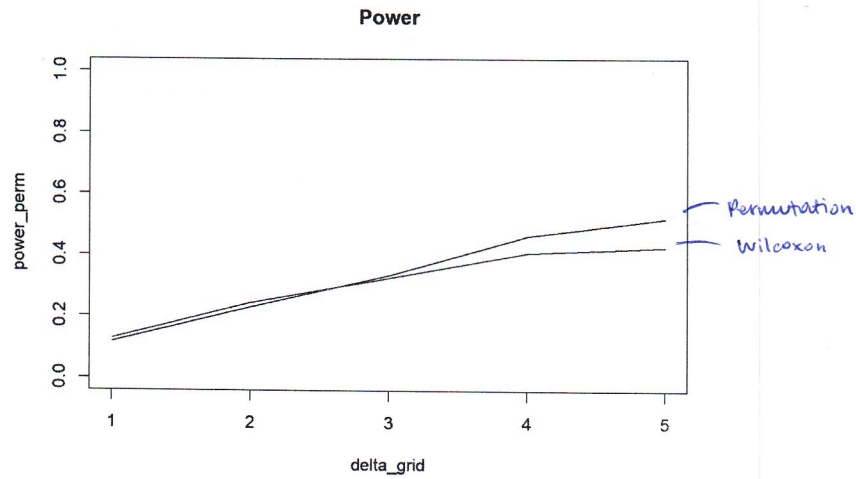
Nota: qua mi sa che quello in alto in realtà è Wilcoxon, comunque sia il problema che si ha tra media e mediana rimane (si mi sa che è Wilcoxon perché quando andiamo di mediana (e non media) il permutation migliore, il wilcoxon rimane)

Perm. test (MEDIAN)

```r
# power with a t-test structured like this is abysmal...
# Let's work on a median test...

perm_median_test=function(x,y,iter=1e4){
  T0       = abs(median(x)-median(y))
  T_stat   = numeric(iter)
  x_pooled = c(x,y)
  n        = length(x_pooled)
  n1       = length(x)
  for(perm in 1:iter){
    # 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(median(x1_perm) - median(x2_perm))
  }
  # p-value
  p_val <- sum(T_stat>=T0)/iter
  return(p_val)
}
```

```r
power_perm=numeric(length(delta_grid))
set.seed(seed)
for(ii in 1:length(delta_grid)){
  p.value <- numeric(B)
  pb=progress_bar$new(total=B)
  pb$tick(0)
  delta=delta_grid[ii]
  for(j in 1:B){
    x.1 = rstable(n,.5,0)
    x.2 = rstable(n,.5,0,delta=delta)
    p.value[j] <- perm_median_test(x.1,x.2,iter=1000)
    pb$tick()
    }
  estimated.power <- sum(p.value < alpha)/B
  power_perm[ii]=estimated.power
}
```

```r
plot(delta_grid,power_perm,type='l', main='Power',ylim=c(0,1))
lines(delta_grid,power_wilcox,col='red')
```




```r
# what happens instead, if I run a nonparametric test when i could've used a parametric one?

# T-test vs nonparametric alternatives...
delta_grid=seq(.1,1,by=.2)
power_perm=numeric(length(delta_grid))
set.seed(seed)
for(ii in 1:length(delta_grid)){
  p.value <- numeric(B)
  pb=progress_bar$new(total=B)
  pb$tick(0)
  delta=delta_grid[ii]
  for(j in 1:B){
    x.1 = rnorm(100)
    x.2 = rnorm(100,mean=delta)
    p.value[j] = perm_t_test(x.1,x.2,iter=1000)
    pb$tick()
    }
  estimated.power <- sum(p.value < alpha)/B
  power_perm[ii]=estimated.power
}
```

Settings: normal distributions shifted by:
$\delta\_grid := [0.1, 0.3, 0.5, 0.7, 0.9]$

Approaches:
- permutation test (mean)
- Wilcoxon test
- (standard) t-test

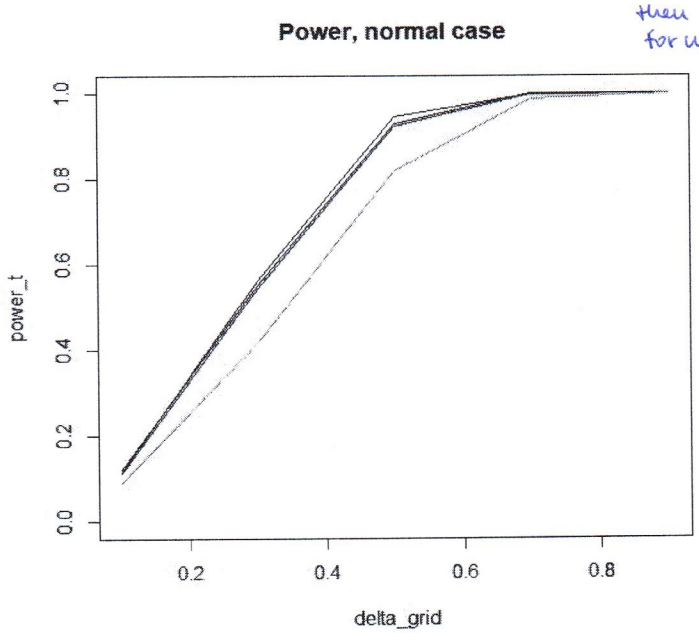
```
set.seed(seed)
power_wilcox=numeric(length(delta_grid))
for(ii in 1:length(delta_grid)){
  p.value <- numeric(B)
  pb=progress_bar$new(total=B)
  pb$tick(0)
  delta=delta_grid[ii]
  for(j in 1:B){
    x.1 = rnorm(100)
    x.2 = rnorm(100,mean=delta)
    p.value[j] <- wilcox.test(x.1,y=x.2,correct = T)$p.value
    pb$tick()
    }
  estimated.power <- sum(p.value < alpha)/B
  power_wilcox[ii]=estimated.power
}
```

```
set.seed(seed)
power_t=numeric(length(delta_grid))
# I can actually compute in an analytical fashion, but let's go with sim also here.
for(ii in 1:length(delta_grid)){
  p.value <- numeric(B)
  pb=progress_bar$new(total=B)
  pb$tick(0)
  delta=delta_grid[ii]
  for(j in 1:B){
    x.1 = rstable(n,.5,0)
    x.2 = rstable(n,.5,0,delta=delta)
    p.value[j] <- t.test(x.1,y=x.2)$p.value
    pb$tick()
    }
  estimated.power <- sum(p.value < alpha)/B
  power_t[ii]=estimated.power
}
```

```
set.seed(seed)
power_median=numeric(length(delta_grid))
for(ii in 1:length(delta_grid)){
  p.value <- numeric(B)
  pb=progress_bar$new(total=B)
  pb$tick(0)
  delta=delta_grid[ii]
  for(j in 1:B){
    x.1 <- rnorm(100)
    x.2 = rnorm(100,mean=delta)
    p.value[j] <- perm_median_test(x.1,x.2,iter=1000)
    pb$tick()
    }
  estimated.power <- sum(p.value < alpha)/B
  power_median[ii]=estimated.power
}
```

```
plot(delta_grid,power_t,ylim=c(0,1), main='Power, normal case',type='l')
lines(delta_grid,power_wilcox,col='red')
lines(delta_grid,power_perm,  col='blue')
lines(delta_grid,power_median,col='green')
```

Remember: if assumptions are met we want to use parametric settings (for the sake of computation).
If the assumptions are not met then we should always go for non-parametric settings.



```r
library(progress)
library(stabledist)
```

```r
### ------------------------------------------------------------------------------
### HOMEWORK
### Permutation test for equality of variances
### ------------------------------------------------------------------------------
### Show that it's valid.


### ------------------------------------------------------------------------------
### H0 : var(group_1) = var(group_2)
### H1 : var(group_1) > var(group_2)
### Test stat = (s^2)_1 / (s^2)_2
### ------------------------------------------------------------------------------
```

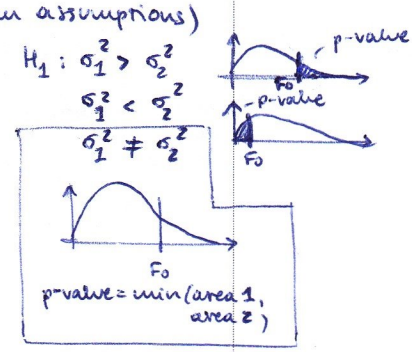
```r
### 1. Function
### ------------------------------------------------------------------------------
perm_var_test = function(x,y,iter=1e3){
  T0       = var(x)/var(y)
  T_stat   = numeric(iter)
  x_pooled = c(x,y)
  n        = length(x_pooled)
  n1       = length(x)
  for(perm in 1:iter){
    # permutation:
    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] <- var(x1_perm)/var(x2_perm)
  }
  # p-value
  p_val <- sum(T_stat>=T0)/iter
  return(p_val)
}
```

PARAMETRIC SETTINGS:
VARIANCE OF TWO POPULATIONS
(F-test, gaussian assumptions)

$H_0: \sigma_1^2 = \sigma_2^2$ $\quad H_1: \sigma_1^2 > \sigma_2^2$

$\sigma_1^2 < \sigma_2^2$

$\sigma_1^2 \neq \sigma_2^2$



$$F := \frac{S_1^2}{S_2^2}$$

#pop 1 = n_1
#pop 2 = n_2
under H_0:

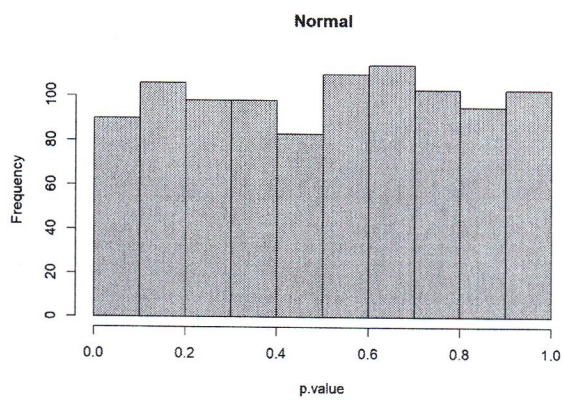
$$F \sim F(n_1 - 1, n_2 - 1)$$

Hint: always call group 1 the group that we suppose to have LARGEST variance

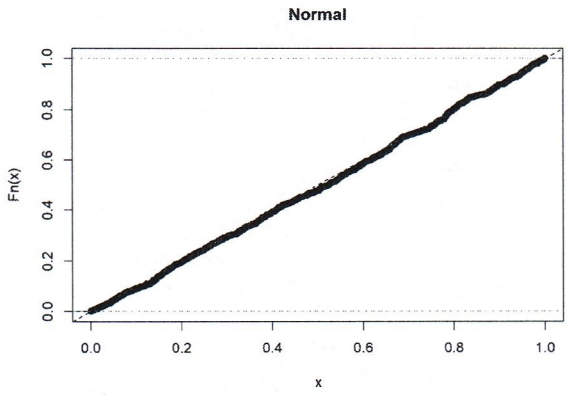
Note: this test is very sensible to gaussianity hypothesis violation. If Hp. are slightly violated this test goes crazy
→ non-param settings

```r
### 2. Type I error (validity)
### ------------------------------------------------------------------------------
B       = 1000
n       = 10
alpha   = 0.05
p.value = numeric(B)
set.seed(123)
```

```r
### 2.1. Gaussian
### ------------------------------------------------------------------------------
for(j in 1:B){
  x.1 = rnorm(n)
  x.2 = rnorm(n)
  p.value[j] = perm_var_test(x.1, x.2, iter=1000)
}

hist(p.value, main='Normal')
```



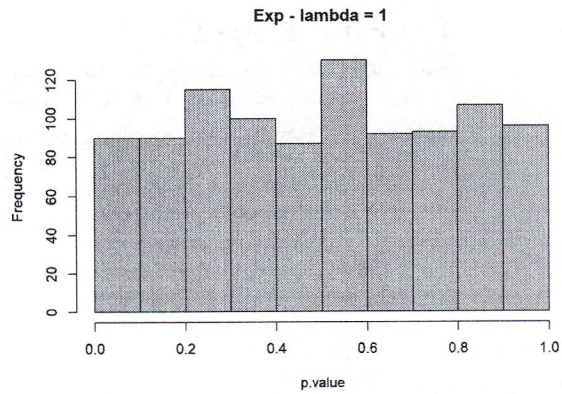

```r
plot(ecdf(p.value), main='Normal', xlim=c(0,1))
abline(0,1, lty=2, col='red')
```



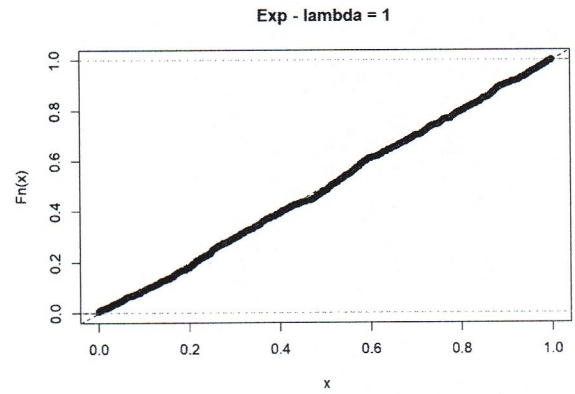

```
estimated.alpha = sum(p.value <alpha)/B
c(estimated.alpha - sqrt(estimated.alpha * (1-estimated.alpha)/B)*qnorm(0.975),
  estimated.alpha,
  estimated.alpha + sqrt(estimated.alpha * (1-estimated.alpha)/B)*qnorm(0.975))
```

```
## [1] 0.02956759 0.04200000 0.05443241
```

```
### 2.2. Exponential
### ------------------------------------------------------------------------------
for(j in 1:B){
  x.1 = rexp(n)
  x.2 = rexp(n)
  p.value[j] = perm_var_test(x.1, x.2, iter=1000)
}

hist(p.value, main='Exp - lambda = 1')
```



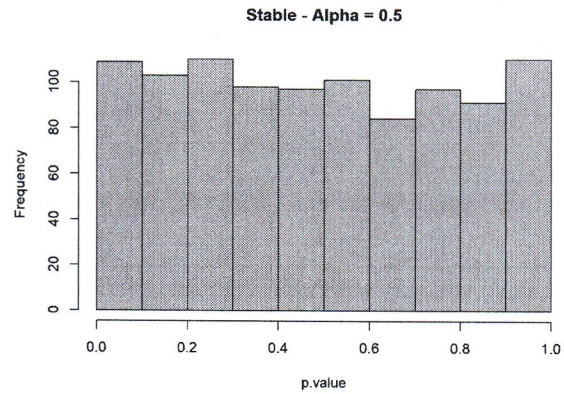

```
plot(ecdf(p.value), main='Exp - lambda = 1', xlim=c(0,1))
abline(0,1, lty=2, col='red')
```



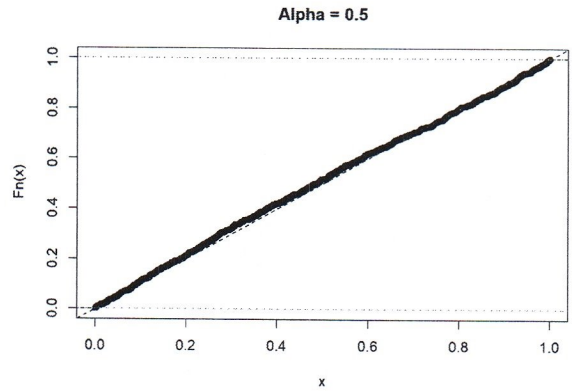

```
estimated.alpha = sum(p.value <alpha)/B
c(estimated.alpha - sqrt(estimated.alpha * (1-estimated.alpha)/B)*qnorm(0.975),
  estimated.alpha,
  estimated.alpha + sqrt(estimated.alpha * (1-estimated.alpha)/B)*qnorm(0.975))
```

```
## [1] 0.03301622 0.04600000 0.05898378
```

```
### 2.3. Stable with alpha = 0.5 (no mean, no variance)
###      (we hope to test somehow the variability of data)
### ---------------------------------------------------------------------------
for(j in 1:B){
  x.1 = rstable(n, .5, 0)
  x.2 = rstable(n, .5, 0)
  p.value[j] = perm_var_test(x.1, x.2, iter=1000)
}

hist(p.value, main='Stable - Alpha = 0.5')
```




```
plot(ecdf(p.value), main='Alpha = 0.5', xlim=c(0,1))
abline(0,1, lty=2, col='red')
```




```
estimated.alpha = sum(p.value <alpha)/B
c(estimated.alpha - sqrt(estimated.alpha * (1-estimated.alpha)/B)*qnorm(0.975),
  estimated.alpha,
  estimated.alpha + sqrt(estimated.alpha * (1-estimated.alpha)/B)*qnorm(0.975))
```

```
## [1] 0.03649188 0.05000000 0.06350812
```

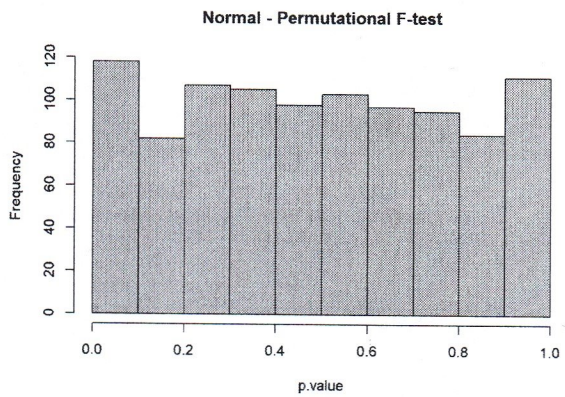
27/10/2020

```
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
### Correction of the assignment
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
# Assignment: come up with a permutation test to evaluate equality in dispersion parameters
# in a two  population, independent setting.
# Let's put ourselves in a relatively simple case:
# H0: same dispersion vs. H1: dispersion(y)>dispersion(x)

library(progress)
rm(list=ls())


perm_f_test = function(x,y,iter=1e3){
  T_stat   = numeric(iter)
  x_pooled = c(x,y)
  n        = length(x_pooled)
  n1       = length(x)
  T0       = var(y)/var(x)
  for(perm in 1:iter){
    # 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] <- var(x2_perm)/var(x1_perm)
  }
  # p-value
  p_val <- sum(T_stat>=T0)/iter
  return(p_val)
}

perm_f_test(rnorm(10),rnorm(10,sd=10))
```

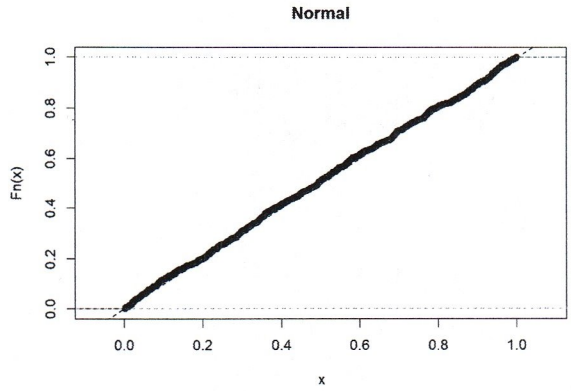
```
## [1] 0
```

```
B     = 1000
seed  = 1991
n     = 100
alpha = 0.05
#calibration

p.value = numeric(B)
pb      = progress_bar$new(total=B, format = "  computing [:bar] :percent eta: :eta")
pb$tick(0)
```

```
set.seed(seed)
for(j in 1:B){
  x1         = rnorm(n)
  x2         = rnorm(n)
  p.value[j] = perm_f_test(x1,x2)
  pb$tick()
}

hist(p.value, main = 'Normal - Permutational F-test')
```




```
plot(ecdf(p.value), main = 'Normal')
abline(0,1, lty=2, col='red')
```

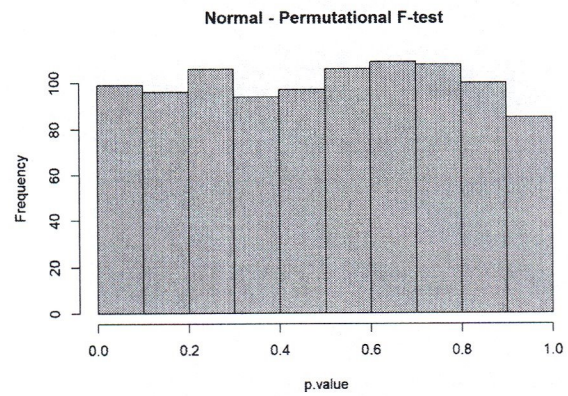



```
estimated.alpha = sum(p.value < alpha)/B
c(estimated.alpha - sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975),
  estimated.alpha,
  estimated.alpha + sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975))
```

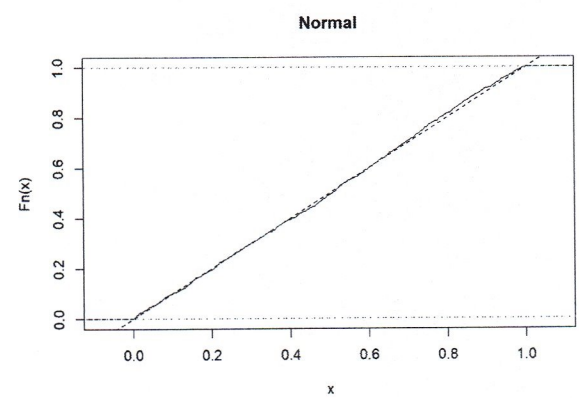
```
## [1] 0.0443961 0.0590000 0.0736039
```

```
p.value = numeric(B)
pb      = progress_bar$new(total=B)
pb$tick(0)
```

```
set.seed(seed)
for(j in 1:B){
  x1         = rnorm(n)
  x2         = rnorm(n)
  p.value[j] = var.test(x1,x2,alternative = 'greater')$p.value
  pb$tick()
}

hist(p.value, main = 'Normal - Permutational F-test')
```




```
plot(ecdf(p.value), main = 'Normal')
abline(0,1, lty=2, col='red')
```

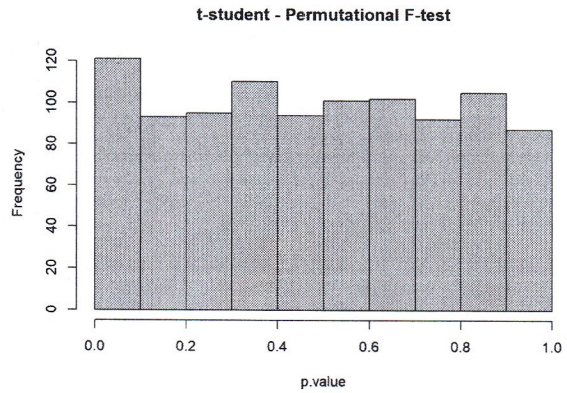


```
estimated.alpha = sum(p.value < alpha)/B
c(estimated.alpha - sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975),
  estimated.alpha,
  estimated.alpha + sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975))
```

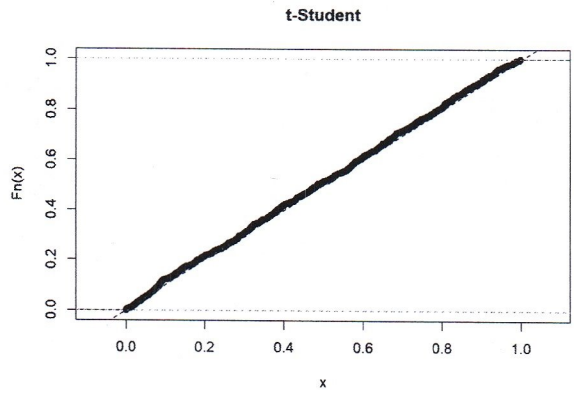
```
## [1] 0.03823888 0.05200000 0.06576112
```

```
p.value = numeric(B)
pb      = progress_bar$new(total=B, format = "  computing [:bar] :percent eta: :eta")
pb$tick(0)
```

```
set.seed(seed)
for(j in 1:B){
  x1         = rt(n,4)
  x2         = rt(n,4)
  p.value[j] = perm_f_test(x1,x2)
  pb$tick()
}

hist(p.value, main = 't-student - Permutational F-test')
```




```
plot(ecdf(p.value), main = 't-Student')
abline(0,1, lty=2, col='red')
```

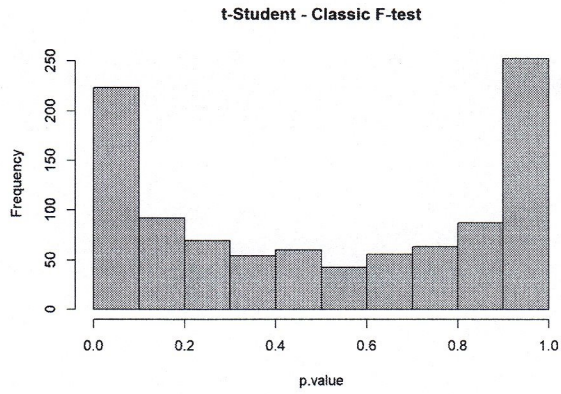



```
estimated.alpha = sum(p.value < alpha)/B
c(estimated.alpha - sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975),
  estimated.alpha,
  estimated.alpha + sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975))
```

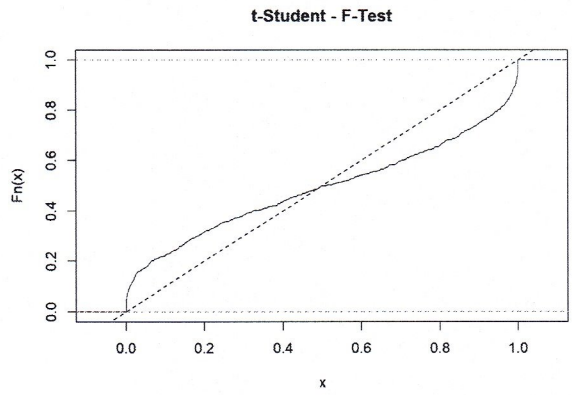
```
## [1] 0.03823888 0.05200000 0.06576112
```

```
p.value = numeric(B)
pb      = progress_bar$new(total=B)
pb$tick(0)
```

```
set.seed(seed)
for(j in 1:B){
  x1         = rt(n,4)
  x2         = rt(n,4)
  p.value[j] = var.test(x1,x2,alternative = 'greater')$p.value
  pb$tick()
}

hist(p.value, main = 't-Student - Classic F-test')
```




```
plot(ecdf(p.value), main = 't-Student - F-Test')
abline(0,1, lty=2, col='red')
```




```
estimated.alpha = sum(p.value < alpha)/B
c(estimated.alpha - sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975),
  estimated.alpha,
  estimated.alpha + sqrt(estimated.alpha*(1-estimated.alpha)/B)*qnorm(0.975))
```

```
## [1] 0.1514498 0.1750000 0.1985502
```

(too high!)

(a test that we would like to perform at level 5%. is actually performed at level ~20%.)

⇒ non-parametric settings!

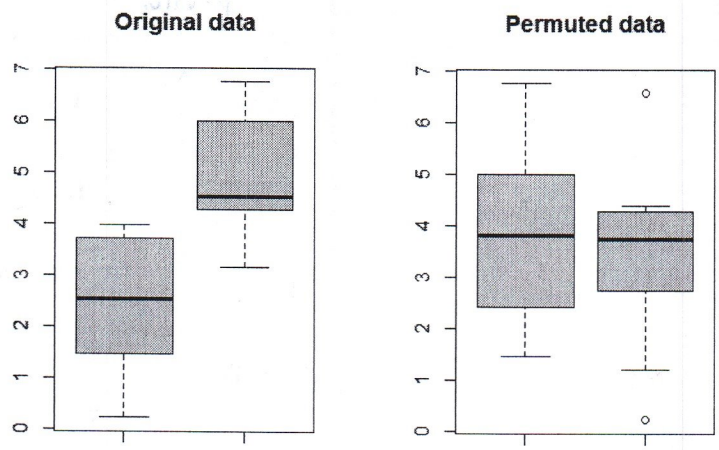
```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### PERMUTATION TESTS: TWO INDEPENDENT POPULATIONS
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------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

Simulated data:

```
# 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)

(we generate 2 samples from 2 different distributions $\Rightarrow$ H_0 false)



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

As 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))
```

(permuted)

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

```
# 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 possible values of the test statistic

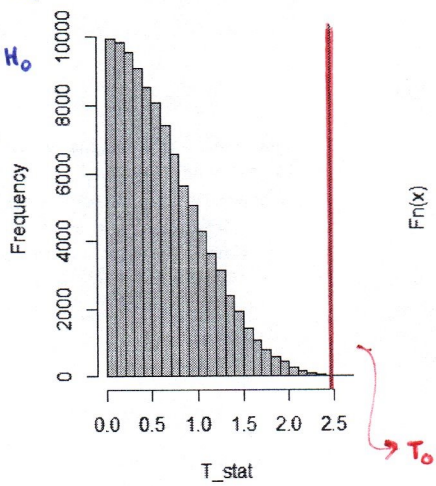
```
# Minimun achieveable p-value:
1/(factorial(n)/(2*factorial(n1)*factorial(n2)))
```

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

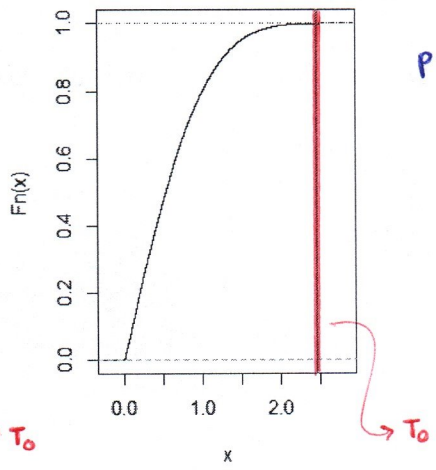
Conditional Monte Carlo estimate (which randomly picks permutations)

```
# 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)
```

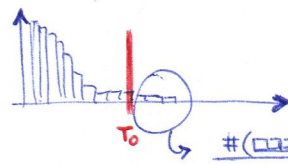
empirical cumulative distribution function (seems continuous but it's not)



permutational distribution under H_0 of $T = |\bar{x}_1 - \bar{x}_2|$ (absolute value of the difference of the samples means)

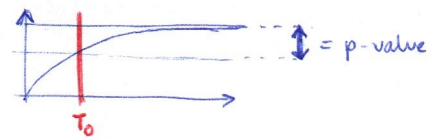


p-value = proportion of permutations that had a **T**-statistic value $\geq T_0$



$\frac{\#(\square\square\square)}{\#\text{ all permutations}}$

p-value with ecdf(T_stat):

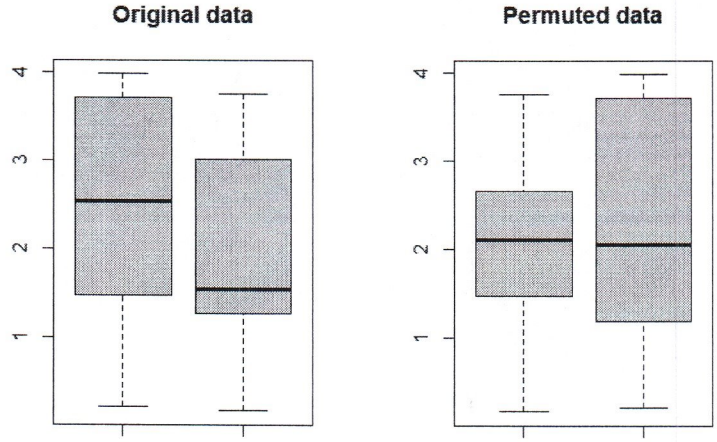



```
# p-value
p_val <- sum(T_stat>=T0)/B
p_val
```

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

⟹ we reject H_0

```
# ------------------------------------------------------------------------------
# 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 a lot. Let's check on other permutations!

```
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
```

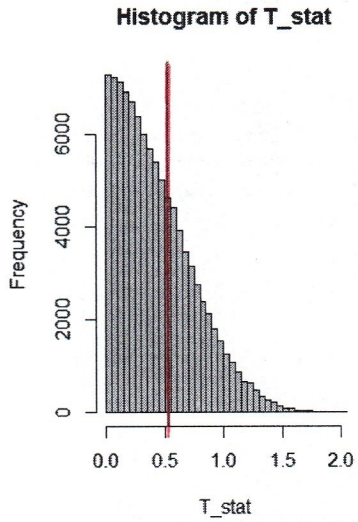
```
# Number of distinct values of T*:
factorial(n)/(2*factorial(n1)*factorial(n2))
```

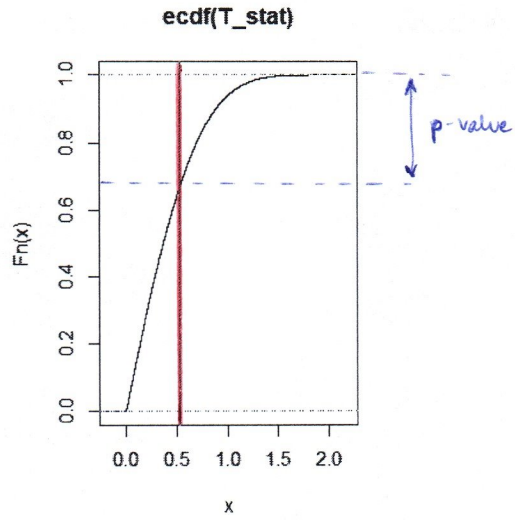
```
## [1] 92378
```

```
# 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
```

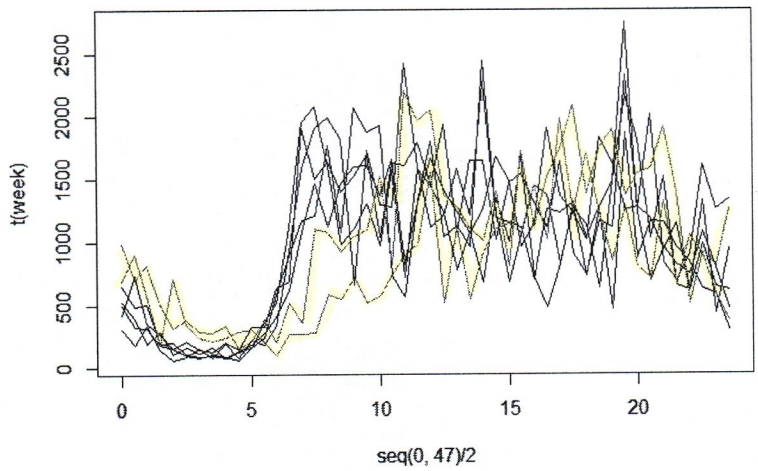
⟹ we do not reject H_0

```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### 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)
```

Note: permutations and reflections are made on the entire unit. We permute two rows, we reflect one row, we don't work on singular column.

Two datasets: 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. Each day is made by 48 half hours. Each row corresponds to one day.



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
```

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

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

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

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

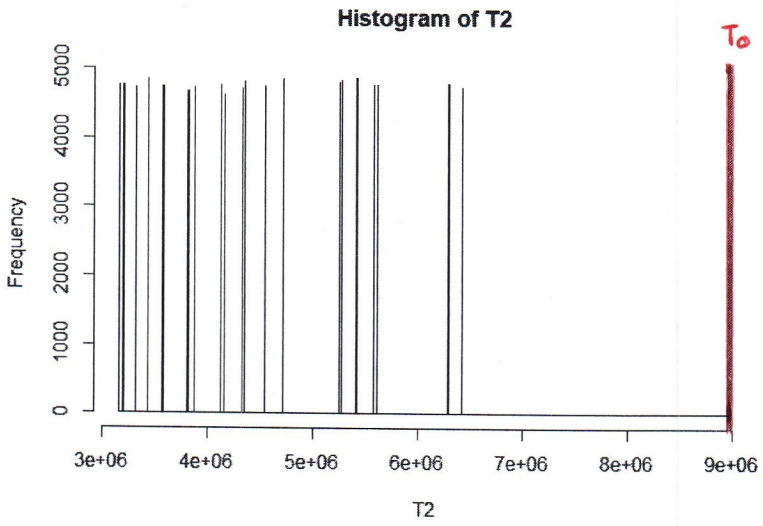
```
# number of different values of the test statistic
choose(7,5)
```

: How many groups of 5 and 2 can we do? We can align 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)
```

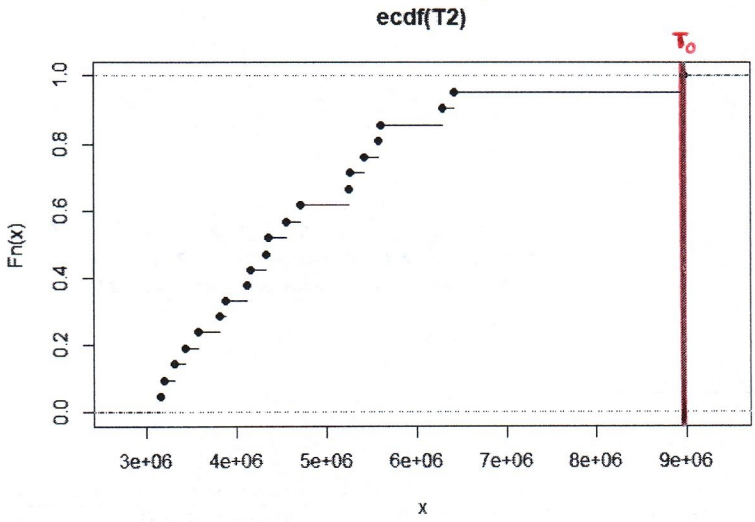
Remember that we're permuting the days (the vectors), we're not destroying vectors! (permutations over the rows, not the columns)



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

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

empirical cumulative distribution function

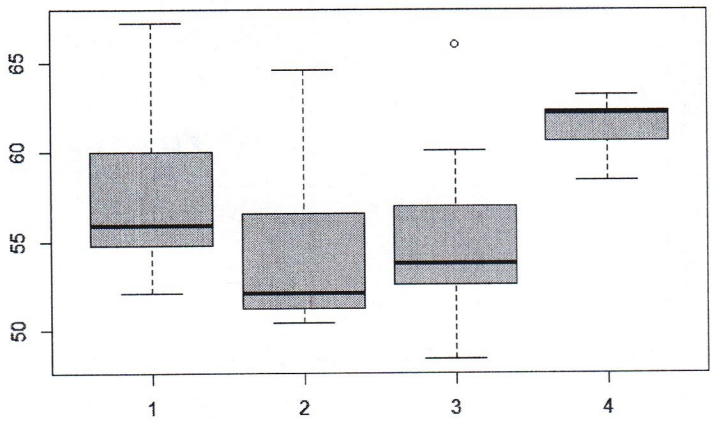



```
# p-value
p_val <- sum(T2>=T20)/B
p_val
```

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

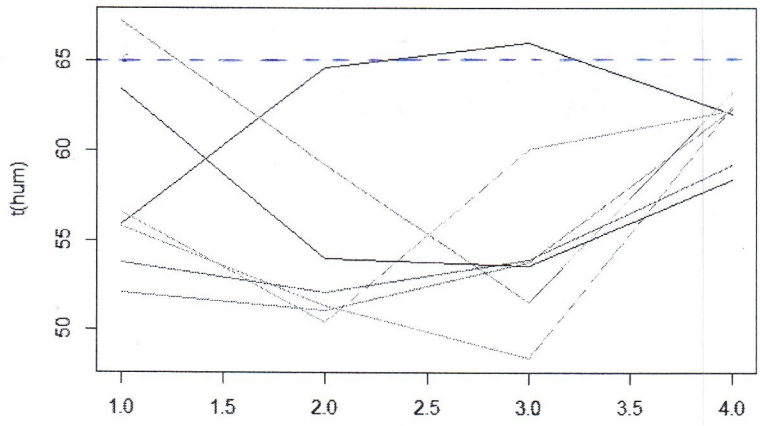
← smallest p-value achievable (we reject H_0)

```
# ------------------------------------------------------------------------------
# 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 (center $= [65,65,65,65]^T$), will 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

$\Longrightarrow 2^N = 2^7 = 128$ possible transform. (notice that this time are not permutations)

```
# 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
```

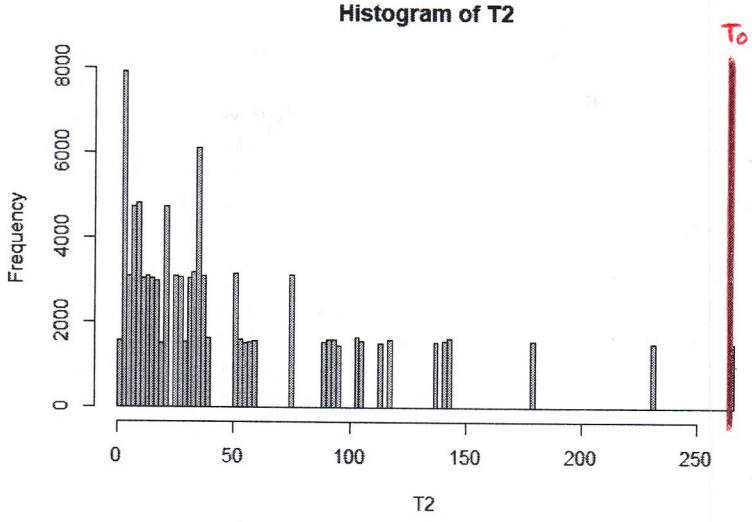
```
## [1] 128
```

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

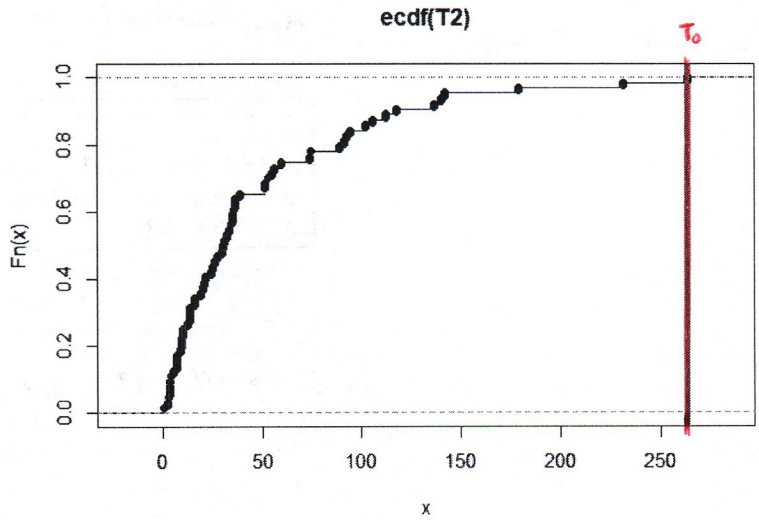
because if we reflect them all we would obtain a value test statistic equal to the original one

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

```
# 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)
```



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



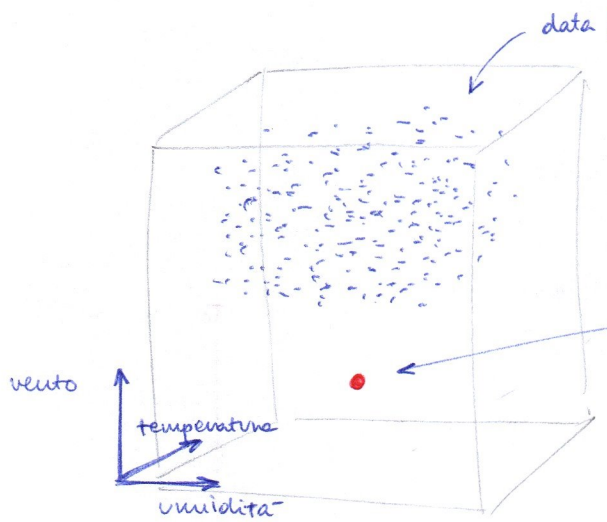

```
# p-value
p_val <- sum(T2>=T20)/B
p_val
```

```
## [1] 0.01535
```
⟹ we reject H_0

```
#-----------------------------------------------------------------------
# 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()
```

kind of same as before. when we have to work with paired populations, we actually work with one population: the population of differences

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



we want to check if this is the symmetry OF the distribution

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

```
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
```

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

Other possibilities

squared euclidean distance

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

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

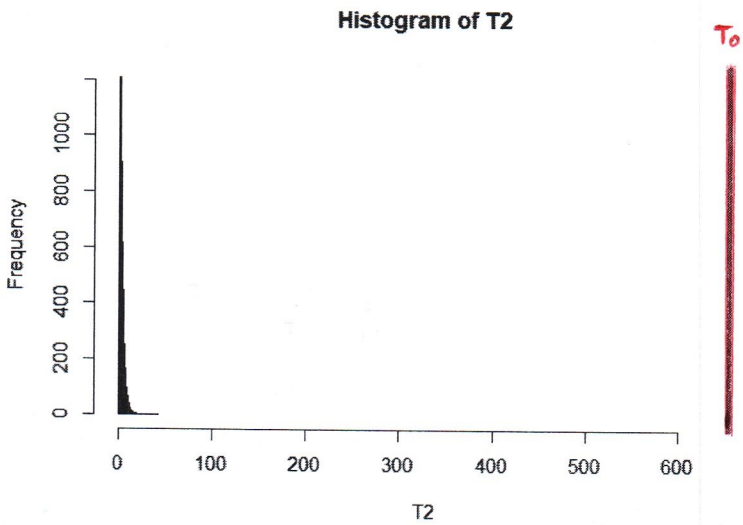
```
## [1] 1.1259e+15
```

→ need of Monte Carlo

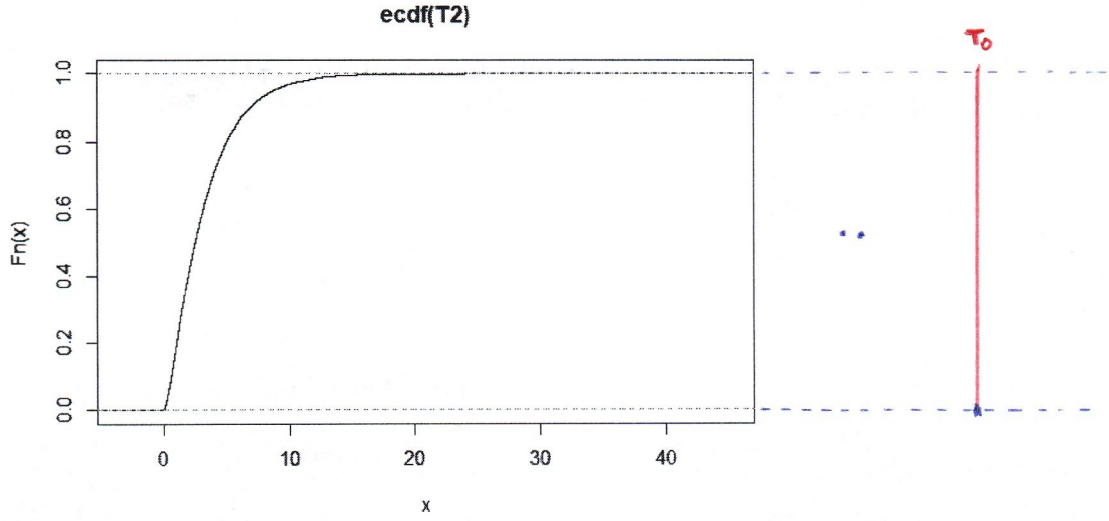
```
# Estimating the permutational distribution under H0
B <- 10000
T2 <- numeric(B)

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've randomly picking up $-1/+1$

we just multiply by $-1/+1$




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




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

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

```
2^50
```

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

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

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

head(chickwts)
```

The ANOVA is based on the F-test. F-test tends to be less robust w.r.t. the violation of normality assumptions (if normality is violated then we have a big problem ⟹ non-parametric settings)

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)
```

Is there effect based on the feed?

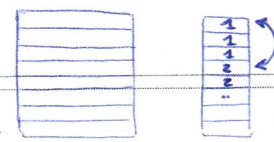
```
##      weight             feed
##  Min.   :108.0   casein   :12
##  1st Qu.:204.5   horsebean:10
##  Median :258.0   linseed  :12
##  Mean   :261.3   meatmeal :11
##  3rd Qu.:323.5   soybean  :14
##  Max.   :423.0   sunflower:12
```

```
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

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

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



we randomly change some labels

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

```
##             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
```

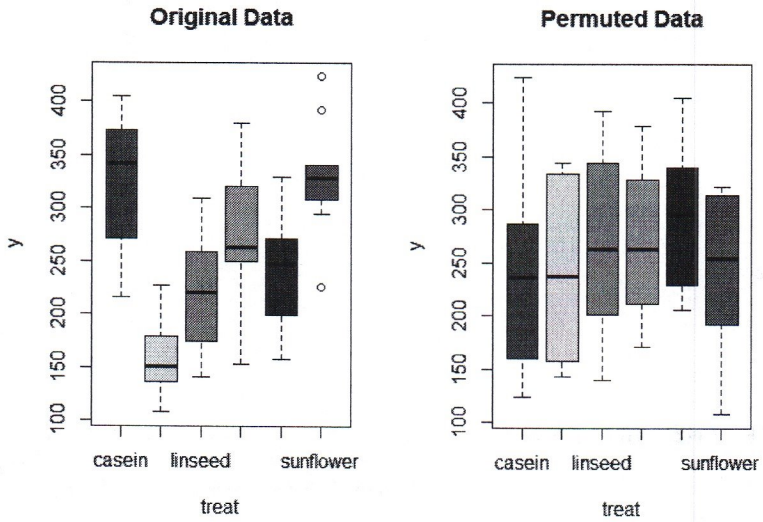
```
# Permutation test:
# Test statistic: F stat
T0 <- summary(fit)[[1]][1,4]
T0
```

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

```
# 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
```

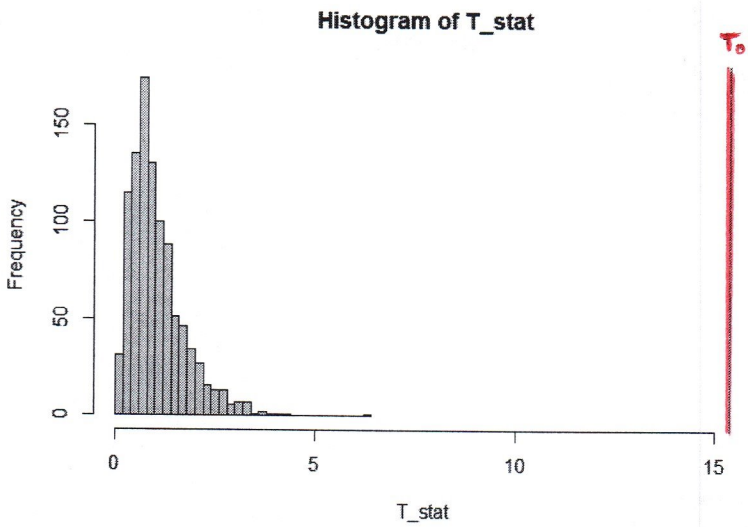
```
plot(feed, weight_perm, xlab='treat',col=rainbow(g),main='Permuted Data')
```



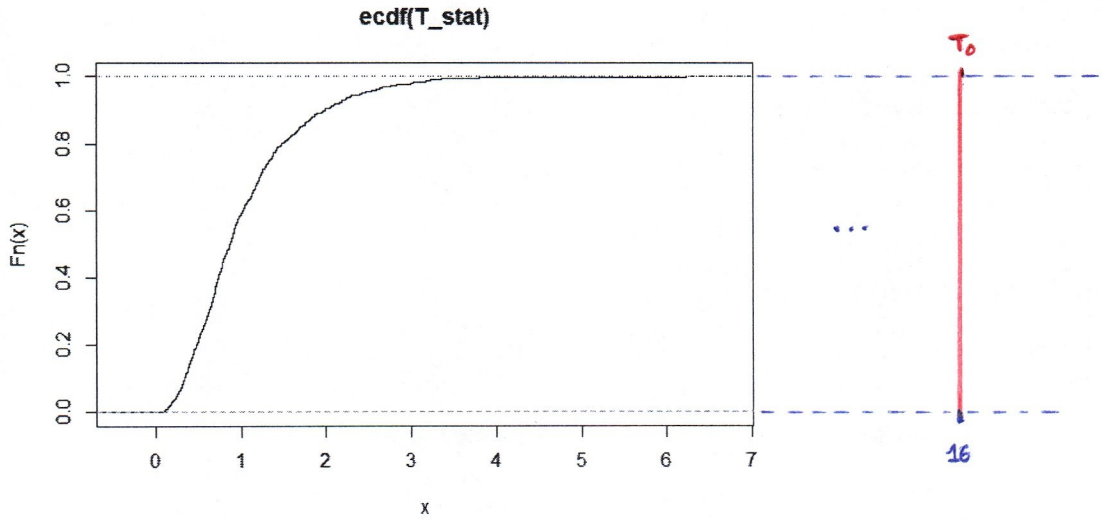
Visually the two are very different, let's make more permutations

```
# 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)
```

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



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



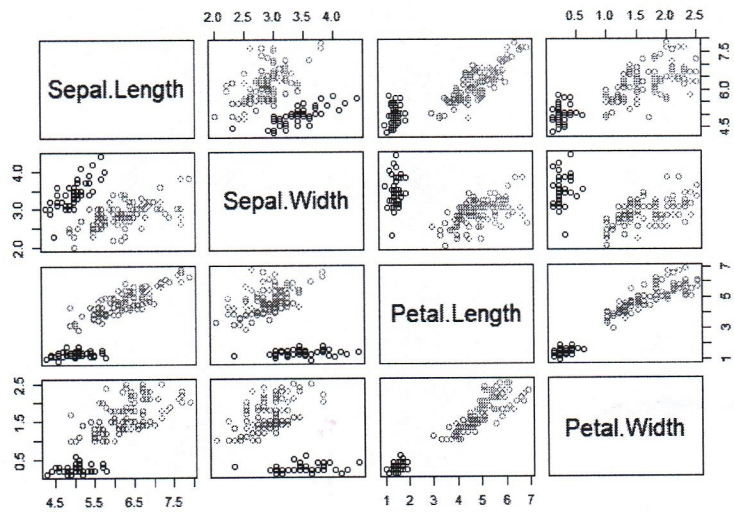

```
# p-value
p_val <- sum(T_stat>=T0)/B
p_val
```

```
## [1] 0
```
⟹ we 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)

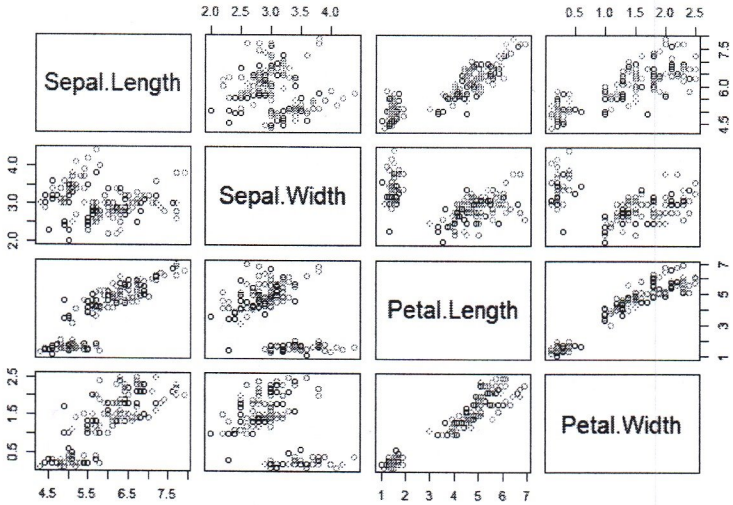



```
i1 <- which(species.name=='setosa')
i2 <- which(species.name=='versicolor')
i3 <- which(species.name=='virginica')
n1 <- length(i1)
n2 <- length(i2)
n3 <- length(i3)
n  <- n1+n2+n3

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

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

We use this as test statistic

```
##               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
```

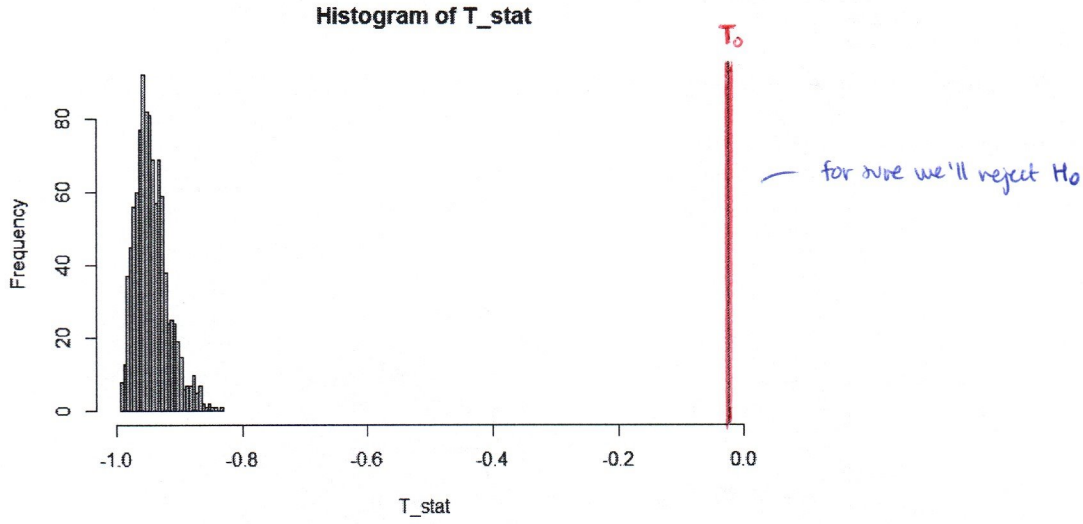
```
# TEST
# Test statistics: Wilks Lambda
T0 <- -summary.manova(fit,test="Wilks")$stats[1,2]
T0
```

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

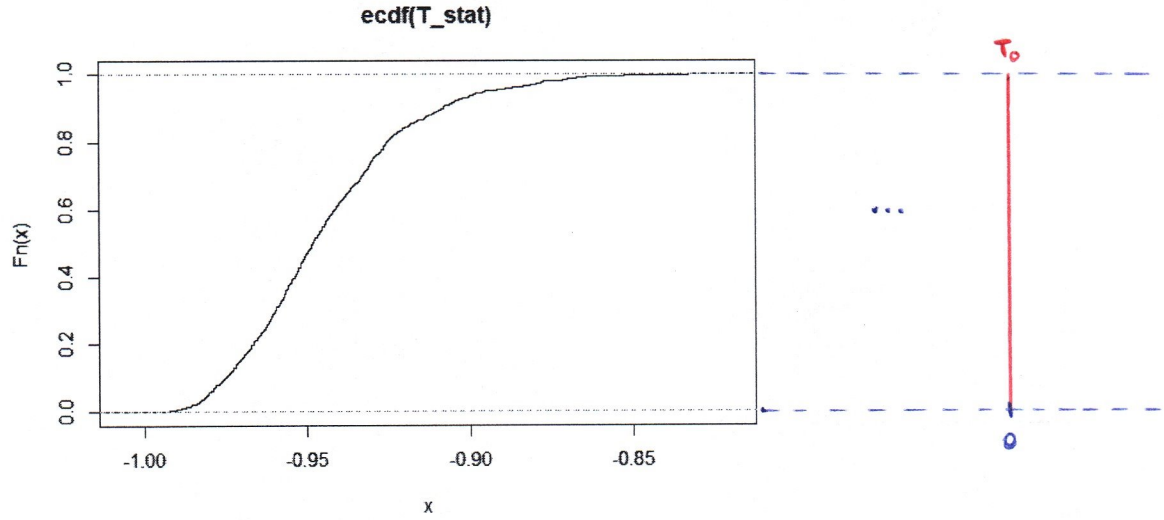
high Wilks → acceptance of H_0

```
# 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)
```

we put a "-", thanks to that we reject for large Wilks values (as in standard procedures (with other test statistics))




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



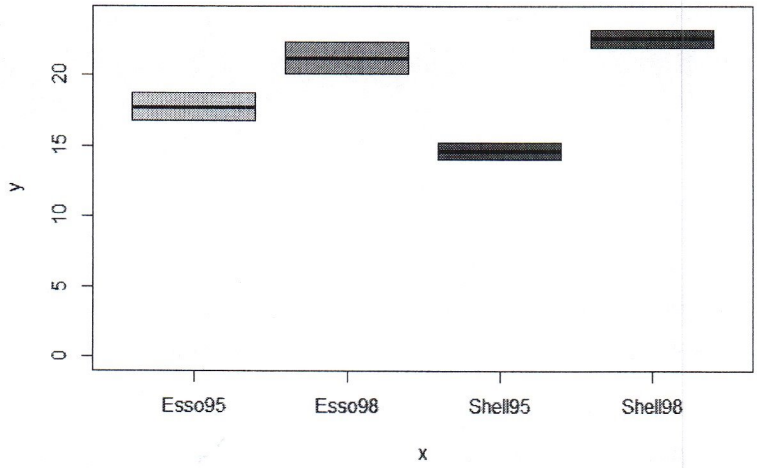

```
# p-value
p_val <- sum(T_stat>=T0)/B
p_val
```

```
## [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))
```

univariate response variable: fuel efficiency (km)
(based/influenced by 2 factor)

← 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 ANOVA-equivalent of Friedman-Levin method (explained later with linear models)

- fit reduced model
- compute the residuals
- permute the residuals
- 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
```

the residuals are permuted under H_0 (under H_0 we can permute them and have no effect (they're likelihood invariant under H_0))

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

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

```r
# 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
```

```r
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))
```

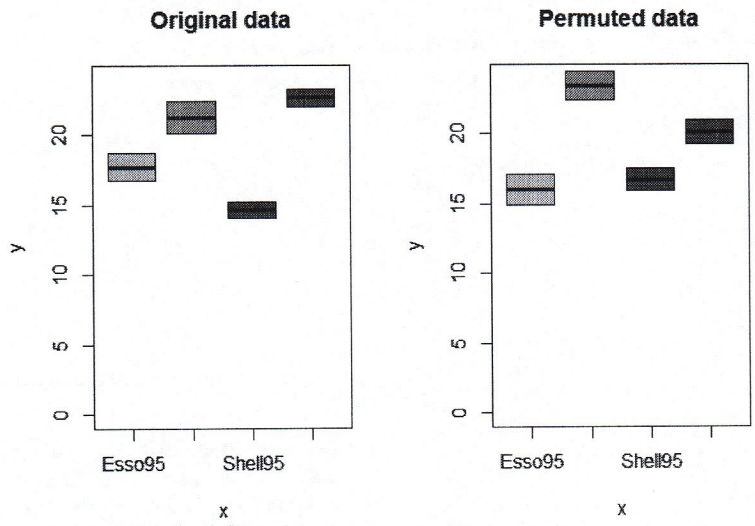
residuals

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
```

```r
# 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')
```




```r
# 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
```

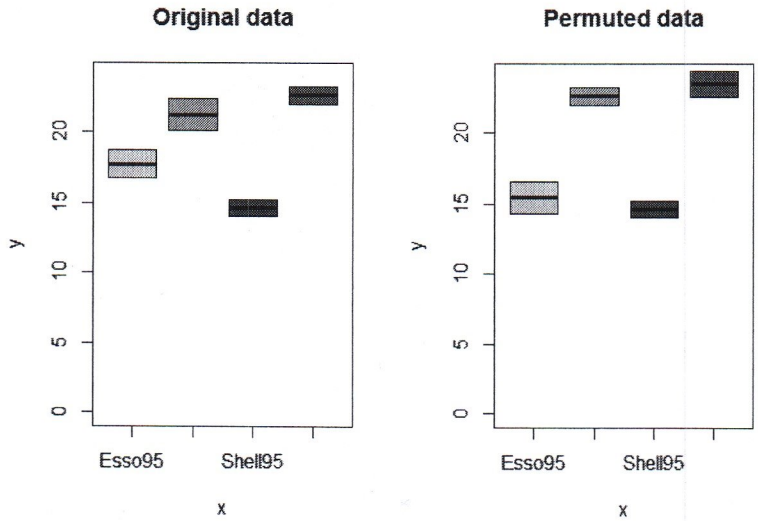
$\Longrightarrow$ we accept H_0: we get rid of the interaction

(2)

```r
# 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')
```



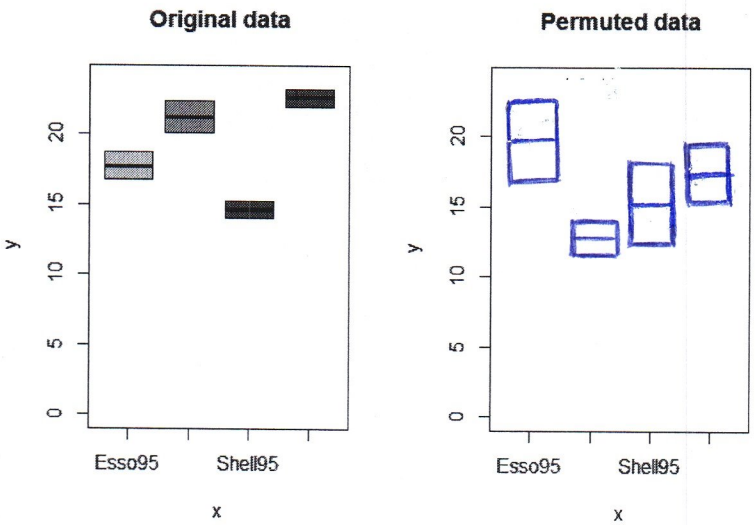
they seem very similar. If this happens for a lot of permutations ⇒ the factor station is not significant

(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
```

```
## [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
```

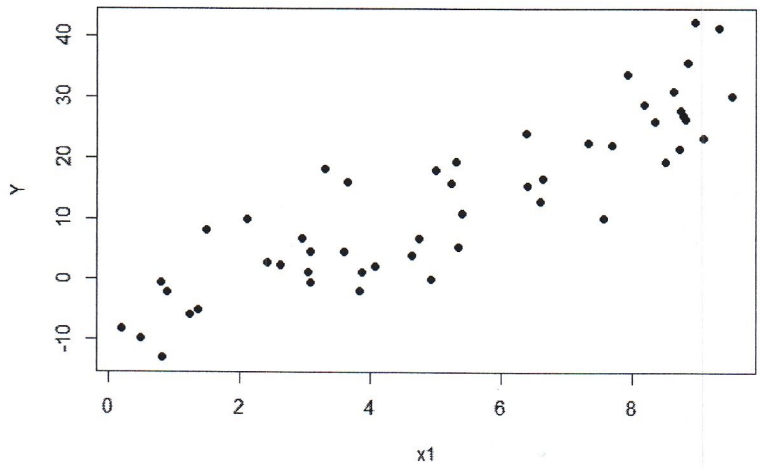
```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### PERMUTATION TEST: Linear models
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------

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

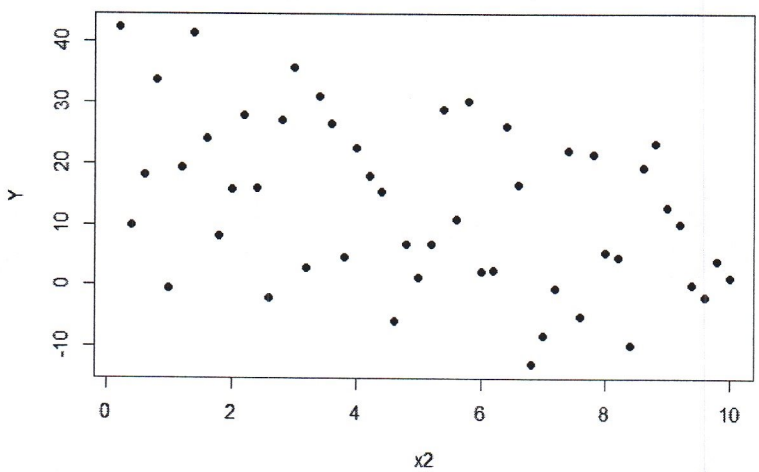
$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \varepsilon$$

we're trying to get rid of the distribution of the error

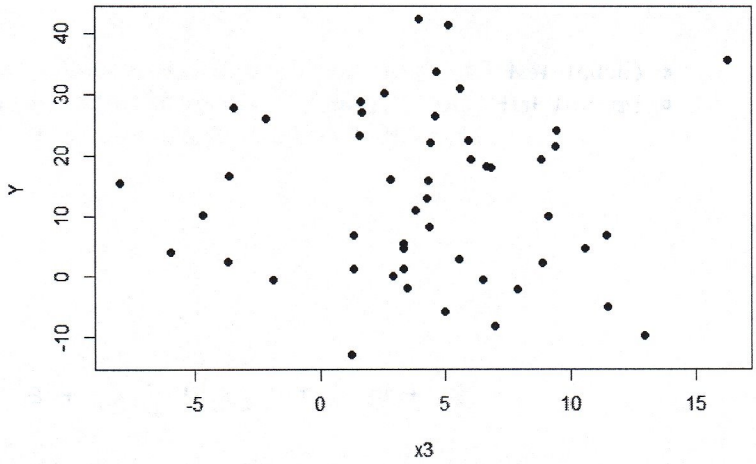
we're assuming all the errors to have the same distribution, just not gaussian !




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




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



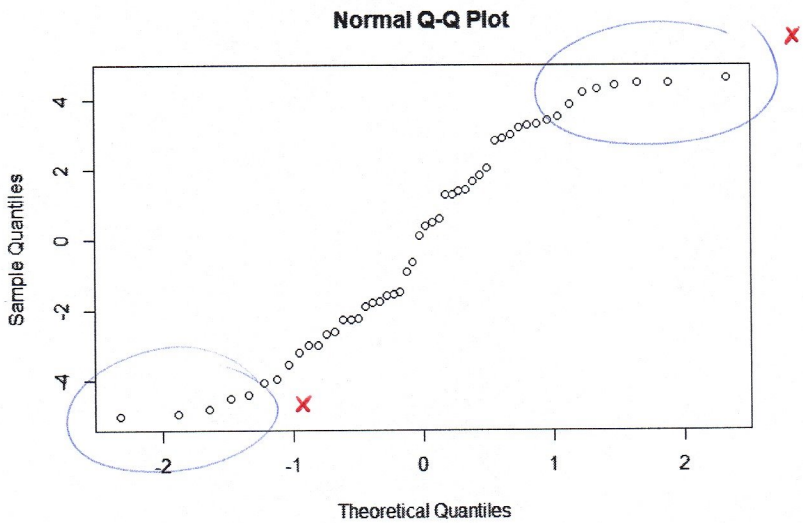

```
# parametric inference
result <- lm(Y ~ x1 + x2 + x3)
summary(result)
```

(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
```

✓ ✓

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



✗ ✗

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

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

✗

(Global)

```
# 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
```

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

← **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 inferentially identical to the previous one

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

```
# 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
```

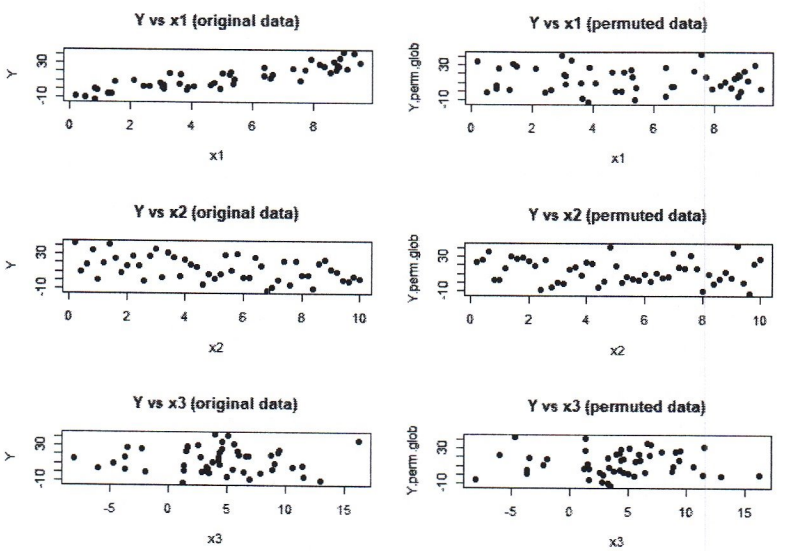
```
mean(Y) + res.H0glob[permutazione]
```

```
##  [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
```

```
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)
```

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

(and we see that there is a difference with the perms.)



here we see 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 would like to permute the errors, but we don't have them)

```
##               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: (Friedman-Levin idea)

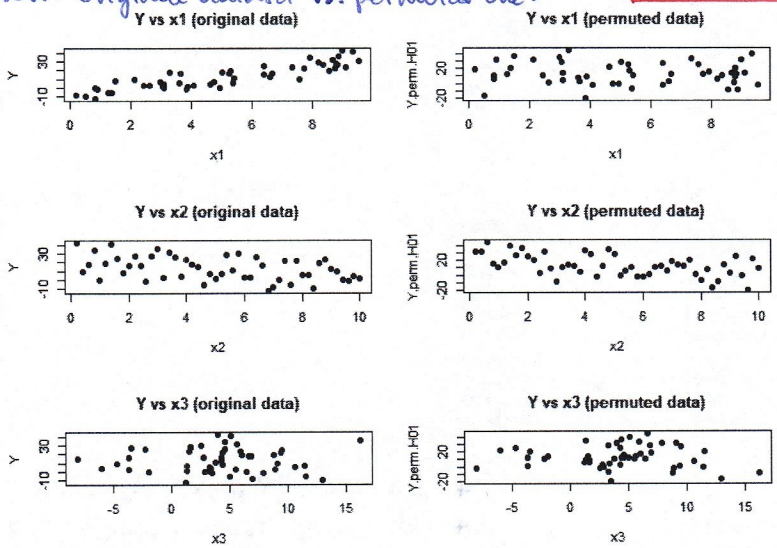
- fit the reduced model (without X_1)
- compute the fitted values of the model *
- compute the residuals
- permute the residuals
- sum the permuted residuals to the fitted values (*) and obtain another dataset
- introduce X_1: fit the whole model (complete model)

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

Idea 2: permute the residuals of the complete model

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

Comparison: original dataset vs. permuted one:



completely changed

UNCHANGED!

(Partial)

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

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

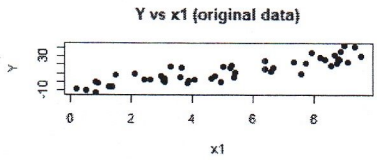
$H_0: \beta_2 = 0$

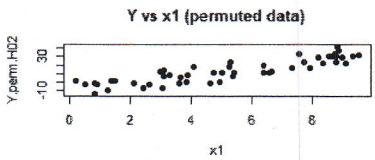
```
##               Estimate Std. Error     t value     Pr(>|t|)
## (Intercept)  0.1820590 1.44085541   0.1263548 9.000018e-01
## x1           4.1759699 0.16073115  25.9810872 3.913695e-29
## x2          -1.8607723 0.15955127 -11.6625349 2.451497e-15
## x3           0.1320238 0.09577616   1.3784623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

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

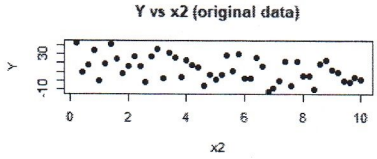
```
## [1] 11.66253
```

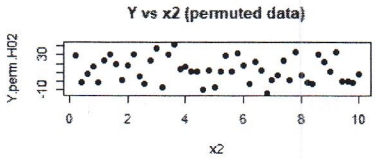
```
# permutations
# residuals of the reduced model

# 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)
```



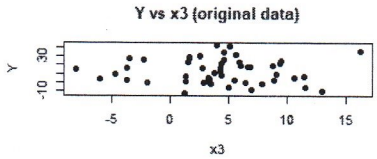


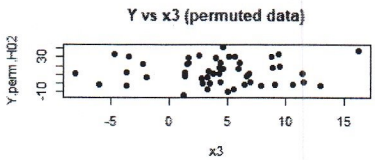
UNCHANGED





changed!





UNCHANGED

(Partial)

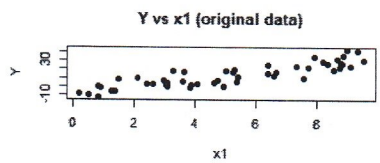
```
# Test on variable x3
# H0: beta3 = 0

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

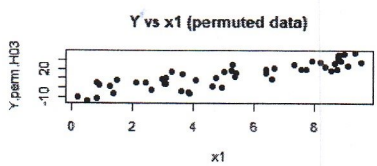
```
##               Estimate Std. Error     t value     Pr(>|t|)
## (Intercept)  0.1820590 1.44085541   0.1263548 9.000018e-01
## x1           4.1759699 0.16073115  25.9810872 3.913695e-29
## x2          -1.8607723 0.15955127 -11.6625349 2.451497e-15
## x3           0.1320238 0.09577616   1.3784623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

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

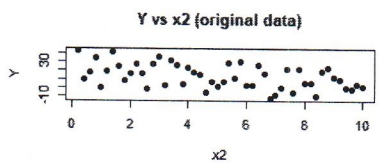
```
## [1] 1.378462
```

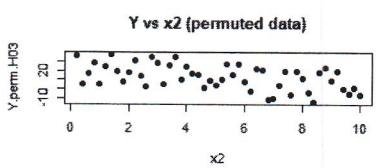
```
# permutations
# residuals of the reduced model

# 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)
```

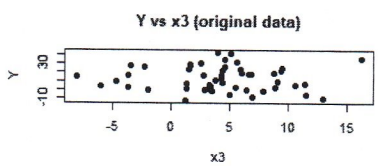


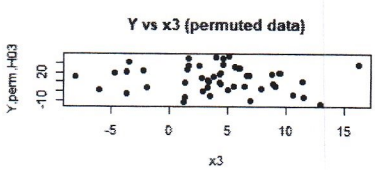


UNCHANGED









changed a little

```r
# 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])

}

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

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

✓

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

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

✓

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

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

✓

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

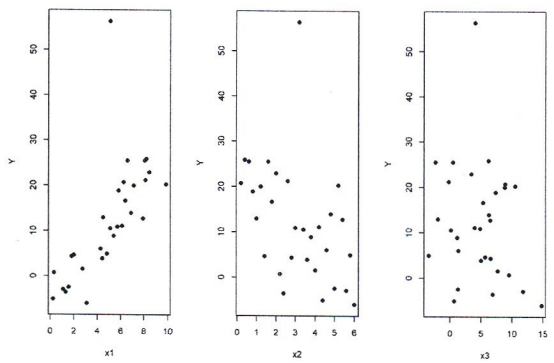
```
## [1] 0.168
```

✗

```r
# 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      ✓
```

```
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
### PERMUTATION TEST: Linear models
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
# Linear models
# Example on simulated data

set.seed(1992)
n  <- 30
# covariate values
x1 <- runif(n,0,10)
x2 <- (1:n)/5
x3 <- rnorm(n,5,5)

# generating model
b0 <- 2
b1 <- 3
b2 <- -2
b3 <- 0
Y  <- b0 + b1*x1 + b2*x2 + b3*x3 + stabledist::rstable(n,1.2,0)

par(mfrow=c(1,3))
plot(x1,Y,pch=16)
plot(x2,Y,pch=16)
plot(x3,Y,pch=16)
```



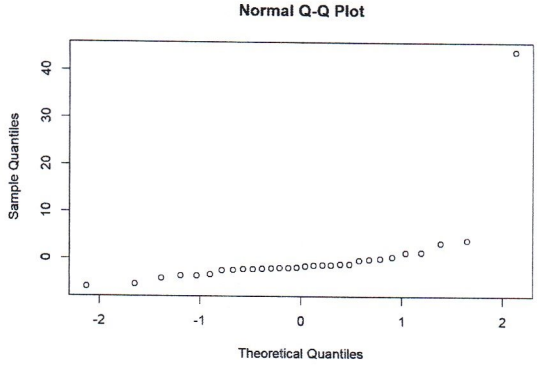

```
# parametric inference
result <- lm(Y ~ x1 + x2 + x3)
summary(result)
```

```
##
## Call:
## lm(formula = Y ~ x1 + x2 + x3)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.979 -2.335 -1.737 -0.260 44.004
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.2154     5.5706   0.936 0.357767
## x1            2.9268     0.6586   4.444 0.000146 ***
## x2           -2.1821     0.9874  -2.210 0.036115 *
## x3           -0.2114     0.3895  -0.543 0.591947
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.07 on 26 degrees of freedom
## Multiple R-squared:  0.5553, Adjusted R-squared:  0.504
## F-statistic: 10.82 on 3 and 26 DF,  p-value: 8.522e-05
```

it seems good. In the sense that we're rejecting where we should reject (β_3)

let's now check if the assumpt. of the model are met: normality of residuals?

```
par(mfrow=c(1,1))
qqnorm(result$residuals)
```



← This is telling us that we're doing something crazy

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

```
##
##  Shapiro-Wilk normality test
##
## data:  result$residuals
## W = 0.39707, p-value = 5.107e-10
```

the normality of the residuals is absolutely not respected

⇒ the p-values of the model do not make sense (even if they lead to right conclusions) in the sense that the statistical hp. on which the inference procedure is based are not satisfied

```
# 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
```

```
##    value
## 10.82281
```

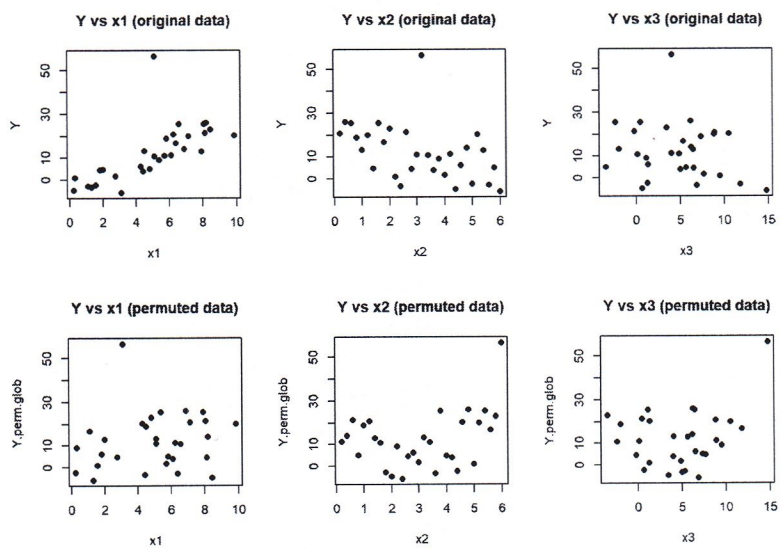
```
# permutations
permutazione <- sample(n)
Y.perm.glob <- Y[permutazione]

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

```
##  [1] 11.154834 13.947026 21.203688  4.953470 18.917391 20.797951 12.824425
##  [8] 10.608779 -2.970116 -5.042114  8.864195 -6.038753  4.403893  6.054960
## [15]  1.580460 12.942796 10.966838 -3.562276 25.451919  4.598725  3.861026
## [22] -2.506088 20.276649 25.873357  0.717380 20.002413 25.430729 16.703224
## [29] 22.925617 56.280482
```

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

```
##  [1] 11.154834 13.947026 21.203688  4.953470 18.917391 20.797951 12.824425
##  [8] 10.608779 -2.970116 -5.042114  8.864195 -6.038753  4.403893  6.054960
## [15]  1.580460 12.942796 10.966838 -3.562276 25.451919  4.598725  3.861026
## [22] -2.506088 20.276649 25.873357  0.717380 20.002413 25.430729 16.703224
## [29] 22.925617 56.280482
```

```
par(mfrow=c(2,3))
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)
```



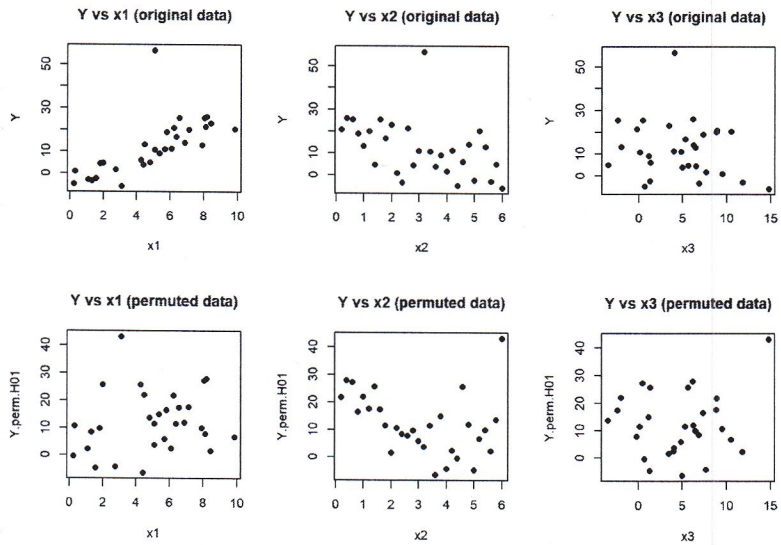

```
### ------------------------------------------------------------------------------
### Test on variable x1
### ------------------------------------------------------------------------------
# H0: beta1 = 0
# test statistic
summary(result)$coefficients
```

```
##               Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)  5.2153780  5.5705734  0.9362372 0.3577665015
## x1           2.9267668  0.6585953  4.4439532 0.0001458356
## x2          -2.1821340  0.9874066 -2.2099650 0.0361154584
## x3          -0.2113903  0.3895040 -0.5427167 0.5919472104
```

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

```
## [1] 4.443953
```

```
# 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

par(mfrow=c(2,3))
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)
```



from the first row to the second data changes a lot, this probably is because β_1 is important ($\neq 0$)

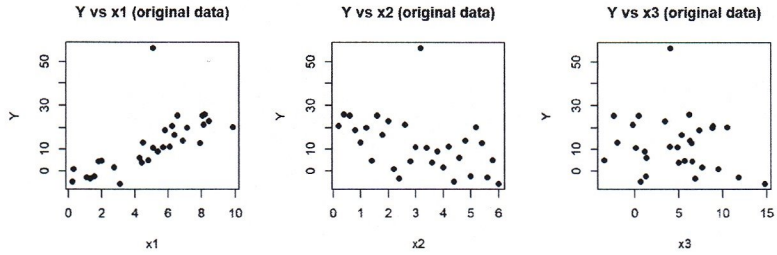
```
### ------------------------------------------------------------------------------
### Test on variable x2
### ------------------------------------------------------------------------------
# H0: beta2 = 0
# test statistic
summary(result)$coefficients
```

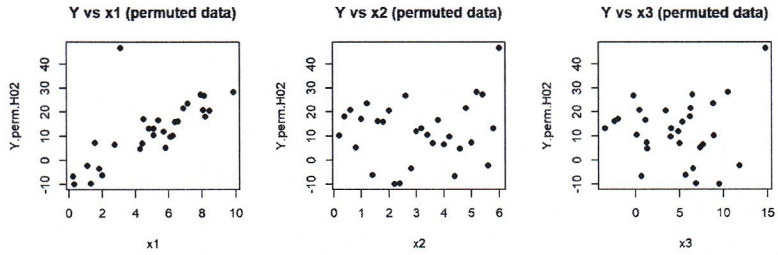
```
##               Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)  5.2153780  5.5705734  0.9362372 0.3577665015
## x1           2.9267668  0.6585953  4.4439532 0.0001458356
## x2          -2.1821340  0.9874066 -2.2099650 0.0361154584
## x3          -0.2113903  0.3895040 -0.5427167 0.5919472104
```

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

```
## [1] 2.209965
```

```
# 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

par(mfrow=c(2,3))
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)
```





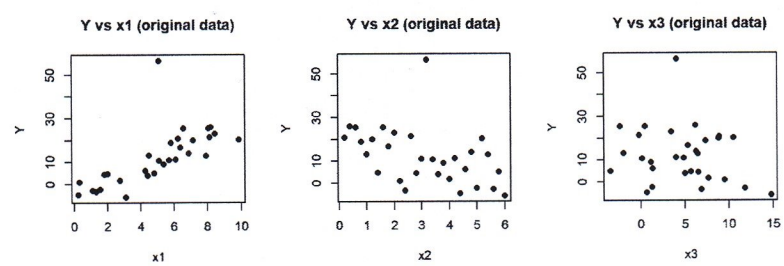

```
### ---------------------------------------------------------------------------
### Test on variable x3
### ---------------------------------------------------------------------------
# H0: beta3 = 0
# test statistic
summary(result)$coefficients
```

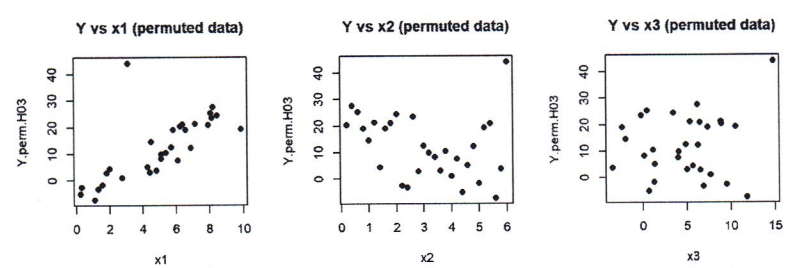
```
##               Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)  5.2153780  5.5705734  0.9362372 0.3577665015
## x1           2.9267668  0.6585953  4.4439532 0.0001458356
## x2          -2.1821340  0.9874066 -2.2099650 0.0361154584
## x3          -0.2113903  0.3895040 -0.5427167 0.5919472104
```

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

```
## [1] 0.5427167
```

```
# 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

par(mfrow=c(2,3))
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)
```





from the first row to the second data does not really change, this probably will lead us to say $\beta_3 = 0$

```
### ------------------------------------------------------------------------------
### 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])

}
```

p-values of the tests:

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

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

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

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

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

```
## [1] 0.025
```

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

```
## [1] 0.61
```

$\Longrightarrow$ we accept $\beta_3 = 0$

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

```
##
## Call:
## lm(formula = Y ~ x1 + x2 + x3)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.979 -2.335 -1.737 -0.260 44.004
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.2154     5.5706   0.936 0.357767
## x1            2.9268     0.6586   4.444 0.000146 ***
## x2           -2.1821     0.9874  -2.210 0.036115 *
## x3           -0.2114     0.3895  -0.543 0.591947
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.07 on 26 degrees of freedom
## Multiple R-squared:  0.5553, Adjusted R-squared:  0.504
## F-statistic: 10.82 on 3 and 26 DF,  p-value: 8.522e-05
```

General approach:

- calculate the statistic on the complete model
- calculate the residuals of the reduced model
- permute the residuals
- $\hat{y}_{\text{reduced model}}$ + permuted residuals = new data
- test statistic on the complete model with new data

on loop

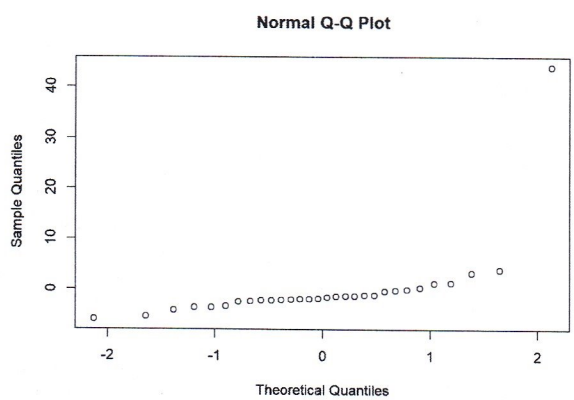
- p-value:

$$\frac{\text{sum}(\text{stat\_perm} >= \text{stat\_original})}{\#\ \text{iterations (loop)}}$$

```
### ---------------------------------------------------------------------------
### ---------------------------------------------------------------------------
### PERMUTATION TEST: Linear models - Homework
### ---------------------------------------------------------------------------
### ---------------------------------------------------------------------------
# Homework: find a test for H0: beta = mu (where mu is generic and not = 0)

# Example on simulated data
set.seed(1992)
n  <- 30
x1 <- runif(n,0,10)
x2 <- (1:n)/5
x3 <- rnorm(n,5,5)

# Generating model
b0 <- 2
b1 <- 3
b2 <- -2
b3 <- 0
Y  <- b0 + b1*x1 + b2*x2 + b3*x3 + stabledist::rstable(n,1.2,0)

### Test on variable x1
### H0: beta1 = a
### H0: Y        = b0 + a*x1 + beta2*x2 + beta3*x3
### H0: Y - a*x1 = b0        + beta2*x2 + beta3*x3

### ---------------------------------------------------------------------------
a = 2.8   # the true value is a = 3
### ---------------------------------------------------------------------------
### Parametric inference
result <- lm(Y - a*x1 ~ x1 + x2 + x3)
summary(result)
```

```
##
## Call:
## lm(formula = Y - a * x1 ~ x1 + x2 + x3)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.979 -2.335 -1.737 -0.260 44.004
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.2154     5.5706   0.936   0.3578
## x1            0.1268     0.6586   0.192   0.8489
## x2           -2.1821     0.9874  -2.210   0.0361 *
## x3           -0.2114     0.3895  -0.543   0.5919
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.07 on 26 degrees of freedom
## Multiple R-squared:  0.1892, Adjusted R-squared:  0.09559
## F-statistic: 2.022 on 3 and 26 DF,  p-value: 0.1355
```

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




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

```
##
##  Shapiro-Wilk normality test
##
## data:  result$residuals
## W = 0.39707, p-value = 5.107e-10
```

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

```
##               Estimate Std. Error    t value   Pr(>|t|)
## (Intercept)  5.2153780  5.5705734  0.9362372 0.35776650
## x1           0.1267668  0.6585953  0.1924805 0.84886136
## x2          -2.1821340  0.9874066 -2.2099650 0.03611546
## x3          -0.2113903  0.3895040 -0.5427167 0.59194721
```

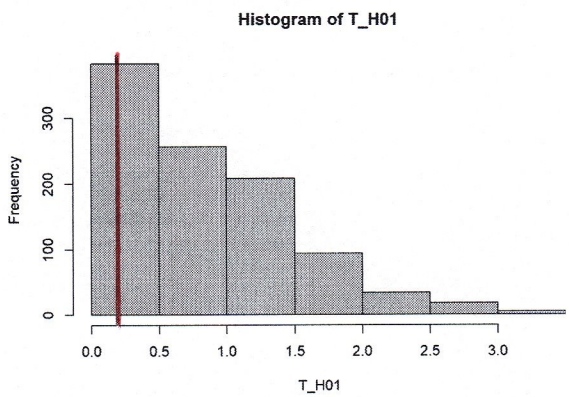
```
T0_x1 <- abs(summary(result)$coefficients[2,3])
T0_x1
```

```
## [1] 0.1924805
```

```
# permutations
# residuals of the reduced model

# reduced (H0) model:
# Y        = beta0 + 3*x1 + beta2*x2 + beta3*x3
# Y - 3*x1 = beta0        + beta2*x2 + beta3*x3
regr.H01         <- lm(Y-a*x1 ~ x2 + x3)
residui.H01      <- regr.H01$residuals
permutazione     <- sample(n)
residui.H01.perm <- residui.H01[permutazione]

# permuted Y:
Y.perm.H01 <- regr.H01$fitted + residui.H01.perm + a*x1

# p-values of the tests
B     <- 1000
T_H01 <- numeric(B)

for(perm in 1:B){
  permutazione     <- sample(1:n)
  residui.H01.perm <- residui.H01[permutazione]
  Y.perm.H01       <- regr.H01$fitted + residui.H01.perm + a*x1
  T_H01[perm]      <- abs(summary(lm(Y.perm.H01 - a*x1 ~ x1 + x2 + x3))$coefficients[2,3])
}

hist(T_H01, xlim=range(c(T_H01, T0_x1)))
sum(T_H01>=T0_x1)/B
```

```
## [1] 0.819
```

```
abline(v=T0_x1, col=3, lwd=4)
```



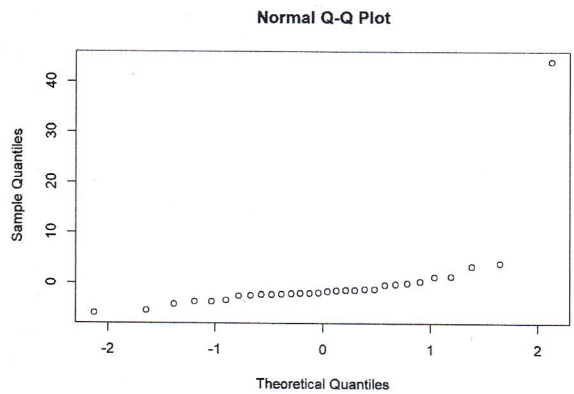

```
# comparison with the parametric test
summary(result)
```

```
##
## Call:
## lm(formula = Y - a * x1 ~ x1 + x2 + x3)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.979 -2.335 -1.737 -0.260 44.004
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.2154     5.5706   0.936   0.3578
## x1            0.1268     0.6586   0.192   0.8489
## x2           -2.1821     0.9874  -2.210   0.0361 *
## x3           -0.2114     0.3895  -0.543   0.5919
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.07 on 26 degrees of freedom
## Multiple R-squared:  0.1892, Adjusted R-squared:  0.09559
## F-statistic: 2.022 on 3 and 26 DF,  p-value: 0.1355
```

```
### ------------------------------------------------------------------------------
a = 1.8   # the true value is a = 3
### ------------------------------------------------------------------------------
### Parametric inference
result <- lm(Y - a*x1 ~ x1 + x2 + x3)
summary(result)
```

```
##
## Call:
## lm(formula = Y - a * x1 ~ x1 + x2 + x3)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.979 -2.335 -1.737 -0.260 44.004
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.2154     5.5706   0.936   0.3578
## x1            1.1268     0.6586   1.711   0.0990 .
## x2           -2.1821     0.9874  -2.210   0.0361 *
## x3           -0.2114     0.3895  -0.543   0.5919
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.07 on 26 degrees of freedom
## Multiple R-squared:  0.2983, Adjusted R-squared:  0.2174
## F-statistic: 3.685 on 3 and 26 DF,  p-value: 0.02463
```

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




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

```
##
##  Shapiro-Wilk normality test
##
## data:  result$residuals
## W = 0.39707, p-value = 5.107e-10
```

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

```
##               Estimate Std. Error    t value   Pr(>|t|)
## (Intercept)  5.2153780  5.5705734  0.9362372 0.35776650
## x1           1.1267668  0.6585953  1.7108636 0.09901390
## x2          -2.1821340  0.9874066 -2.2099650 0.03611546
## x3          -0.2113903  0.3895040 -0.5427167 0.59194721
```

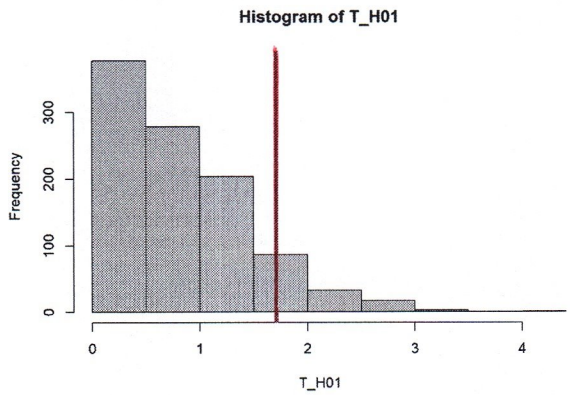
```
T0_x1 <- abs(summary(result)$coefficients[2,3])
T0_x1
```

```
## [1] 1.710864
```

```
# permutations
# residuals of the reduced model

# reduced (H0) model:
# Y        = beta0 + 3*x1 + beta2*x2 + beta3*x3
# Y - 3*x1 = beta0        + beta2*x2 + beta3*x3
regr.H01         <- lm(Y-a*x1 ~ x2 + x3)
residui.H01      <- regr.H01$residuals
permutazione     <- sample(n)
residui.H01.perm <- residui.H01[permutazione]

# permuted Y:
Y.perm.H01 <- regr.H01$fitted + residui.H01.perm + a*x1

# p-values of the tests
B     <- 1000
T_H01 <- numeric(B)

for(perm in 1:B){
  permutazione     <- sample(1:n)
  residui.H01.perm <- residui.H01[permutazione]
  Y.perm.H01       <- regr.H01$fitted + residui.H01.perm + a*x1
  T_H01[perm]      <- abs(summary(lm(Y.perm.H01 - a*x1 ~ x1 + x2 + x3))$coefficients[2,3])
}

hist(T_H01, xlim=range(c(T_H01, T0_x1)))
sum(T_H01>=T0_x1)/B
```

```
## [1] 0.09
```

```
abline(v=T0_x1, col=3, lwd=4)
```




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

```
##
## Call:
## lm(formula = Y - a * x1 ~ x1 + x2 + x3)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.979 -2.335 -1.737 -0.260 44.004
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.2154     5.5706   0.936   0.3578
## x1            1.1268     0.6586   1.711   0.0990 .
## x2           -2.1821     0.9874  -2.210   0.0361 *
## x3           -0.2114     0.3895  -0.543   0.5919
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.07 on 26 degrees of freedom
## Multiple R-squared:  0.2983, Adjusted R-squared:  0.2174
## F-statistic: 3.685 on 3 and 26 DF,  p-value: 0.02463
```

this method is preferred

```
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
### Correction of the assignment
### ------------------------------------------------------------------------------
### ------------------------------------------------------------------------------
set.seed(1992)
n  <- 30
# Covariate values
x1 <- runif(n,0,10)
x2 <- (1:n)/5
x3 <- rnorm(n,5,5)

# Generating model
b0 <- 2
b1 <- 3
b2 <- -2
b3 <- 0
Y  <- b0 + b1*x1 + b2*x2 + b3*x3 + stabledist::rstable(n,1.2,0)
result <- lm(Y ~ x1 + x2 + x3)

### ------------------------------------------------------------------------------
### Test on variable x2
### H0: beta2 = -2
### ------------------------------------------------------------------------------
# Test statistic
mu0   <- -2
summary(result)$coefficients
```

(alternative)

```
##               Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)  5.2153780  5.5705734  0.9362372 0.3577665015
## x1           2.9267668  0.6585953  4.4439532 0.0001458356
## x2          -2.1821340  0.9874066 -2.2099650 0.0361154584
## x3          -0.2113903  0.3895040 -0.5427167 0.5919472104
```

```
T0_x2 <- abs(summary(result)$coefficients[3,3]-mu0)
T0_x2
```

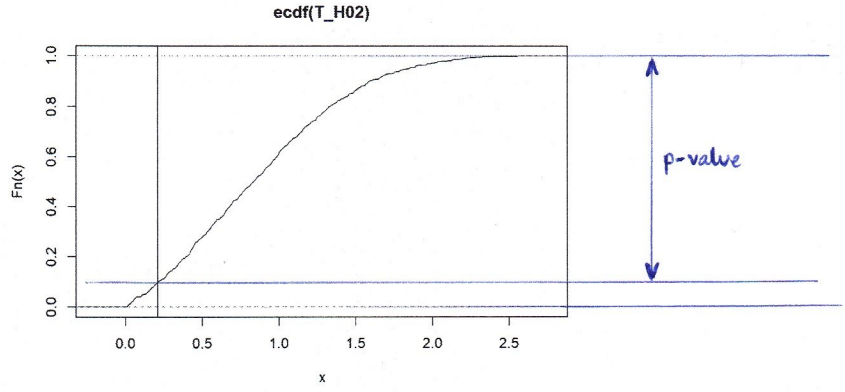
← test statistic

```
## [1] 0.209965
```

```
# Permutations
# Residuals of the reduced model

# Reduced model:
# Y = beta0 + (mu0*x2) + beta1*x1 + beta3*x3
regr.H02 <- lm(Y ~ I(1+x2*mu0) + x1 + x3 - 1) # we replace the intercept with a fixed value
summary(regr.H02)
```

```
##
## Call:
## lm(formula = Y ~ I(1 + x2 * mu0) + x1 + x3 - 1)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.484 -2.395 -1.209  0.311 44.394
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## I(1 + x2 * mu0)  0.85980    0.39302   2.188   0.0375 *
## x1               3.32389    0.41780   7.956  1.5e-08 ***
## x3              -0.09765    0.35888  -0.272   0.7876
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.005 on 27 degrees of freedom
## Multiple R-squared:  0.761,  Adjusted R-squared:  0.7344
## F-statistic: 28.65 on 3 and 27 DF,  p-value: 1.527e-08
```

```
residui.H02 <- regr.H02$residuals

B     <- 1000
T_H02 <- numeric(B)

for(perm in 1:B){
  permutazione     <- sample(n)
  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]-mu0)
}

plot(ecdf(T_H02))
abline(v=T0_x2)
```




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

```
## [1] 0.901
```

```
### -----------------------------------------------------------------------------
### -----------------------------------------------------------------------------
### PERMUTATIONAL (UNIVARIATE) CONFIDECE INTERVAL
### -----------------------------------------------------------------------------
### -----------------------------------------------------------------------------
# Let's compute the confidence interval for the mean of a univariate population.
# The idea is to assign to all points in R a p value, and then threshold over that.

uni_t_perm = function(data,mu0,B=10000){
  data_trans = data-mu0
  T0         = abs(mean(data_trans))
  T_perm     = numeric(B)
  n          = length(data)
  for(perm in 1:B){
    refl         = rbinom(n, 1, 0.5)*2 - 1
    T_perm[perm] = abs(mean(data_trans*refl))
  }
  return(sum(T_perm>=T0)/B)
}

library(pbapply)
library(parallel)

grid = seq(-3,3,by=0.001)

cl = makeCluster(2)
clusterExport(cl,varlist=list("data","uni_t_perm"))

data = stabledist::rstable(30,1.5,0)

perm_wrapper  = function(grid_point){uni_t_perm(data,grid_point,B=2000)}
pval_function = pbsapply(grid,perm_wrapper,cl=cl)
pval_function
plot(grid,pval_function,type='l')
```

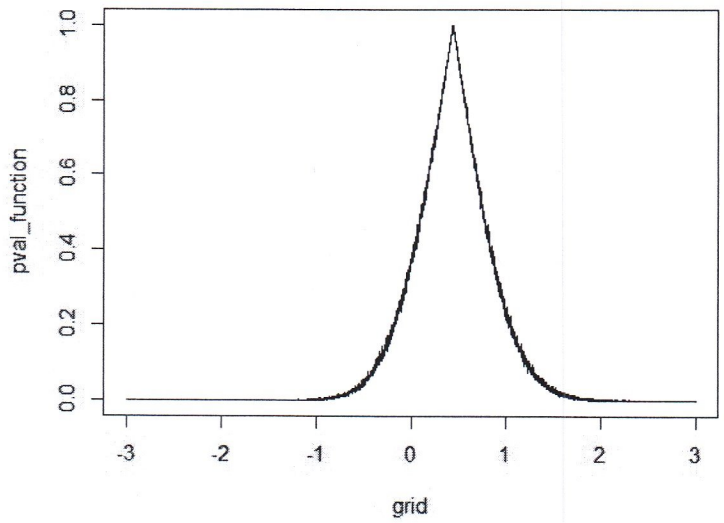
confidence region $= \{x : \text{p-value}(x) \geq \text{trashold}\}$

$= \{$ where we're not rejecting $H_0 \}$

$\Rightarrow$ we look for a p-value function:

$x \mapsto \text{p-value}(x)$

$\Rightarrow$ we test a grid of points and we see which tests are significant and which are not




```
range(grid[pval_function>0.05])
[1]  -0.526   1.425
```