Statistical analysis 12.04.17

```r
# Load datasets from R
library(datasets)

# Load the library we are going to use
library(data.table)
library(psych)
library(MASS)
```

Little reminder on comparisons

Element wise comparison, compares each element of the vector.

```r
a = c(1,2,3,4)

# OR is joining - use ONE |
b = a[((a>3) | (a<2))]
print(b)
```

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

```r
# AND is the intersection - use ONE &
b = a[((a>3) & (a<2))]
print(b)
```

```
## numeric(0)
```

```r
######## Operators ########
# ==  equals to
# <=  smaller than
# =>  greater than
# !=  different than
# &&  Conditional and
# ||  conditional or
# |   elementwise or
# &   elementwise and
```

Check what your datasets

Go to : http://127.0.0.1:8617/help/library/datasets/html/00Index.html Or write : ?datasets then index bottom of the page

Categorical datasets

```r
print(HairEyeColor)
```

```
## , , Sex = Male
##
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    32   11    10     3
##   Brown    53   50    25    15
##   Red      10   10     7     7
##   Blond     3   30     5     8
##
## , , Sex = Female
##
##        Eye
## Hair    Brown Blue Hazel Green
##   Black    36    9     5     2
##   Brown    66   34    29    14
##   Red      16    7     7     7
##   Blond     4   64     5     8
```

Explore the structure of the data. Here it is a table NOT a data.table

```
print(str(HairEyeColor))
```

```
##  table [1:4, 1:4, 1:2] 32 53 10 3 11 50 10 30 10 25 ...
##  - attr(*, "dimnames")=List of 3
##   ..$ Hair: chr [1:4] "Black" "Brown" "Red" "Blond"
##   ..$ Eye : chr [1:4] "Brown" "Blue" "Hazel" "Green"
##   ..$ Sex : chr [1:2] "Male" "Female"
## NULL
```

Get the marginal count for one variable: the sum of all the counts by this variable

```
eyes = margin.table(HairEyeColor, 2)
print(eyes)
```

```
## Eye
## Brown  Blue Hazel Green
##   220   215    93    64
```

```
# convert to proportions
prop.table(eyes)
```

```
## Eye
##     Brown      Blue     Hazel     Green
## 0.3716216 0.3631757 0.1570946 0.1081081
```

```
# test significantly differ from H0 equal proportions
chisq.test(eyes)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  eyes
## X-squared = 133.47, df = 3, p-value < 2.2e-16
```

```
# test with an H0 with custom population proportions (p = c(...,...))
chisq.test(eyes, p=c(.41,.32,.15,.12))
```

```
##
##  Chi-squared test for given probabilities
```

```
##
## data:  eyes
## X-squared = 6.4717, df = 3, p-value = 0.09079
```

```
# Different structure for a data.table
dt = data.table(a = rep(23,4))
str(dt)
```

```
## Classes 'data.table' and 'data.frame':   4 obs. of  1 variable:
##  $ a: num  23 23 23 23
##  - attr(*, ".internal.selfref")=<externalptr>
```

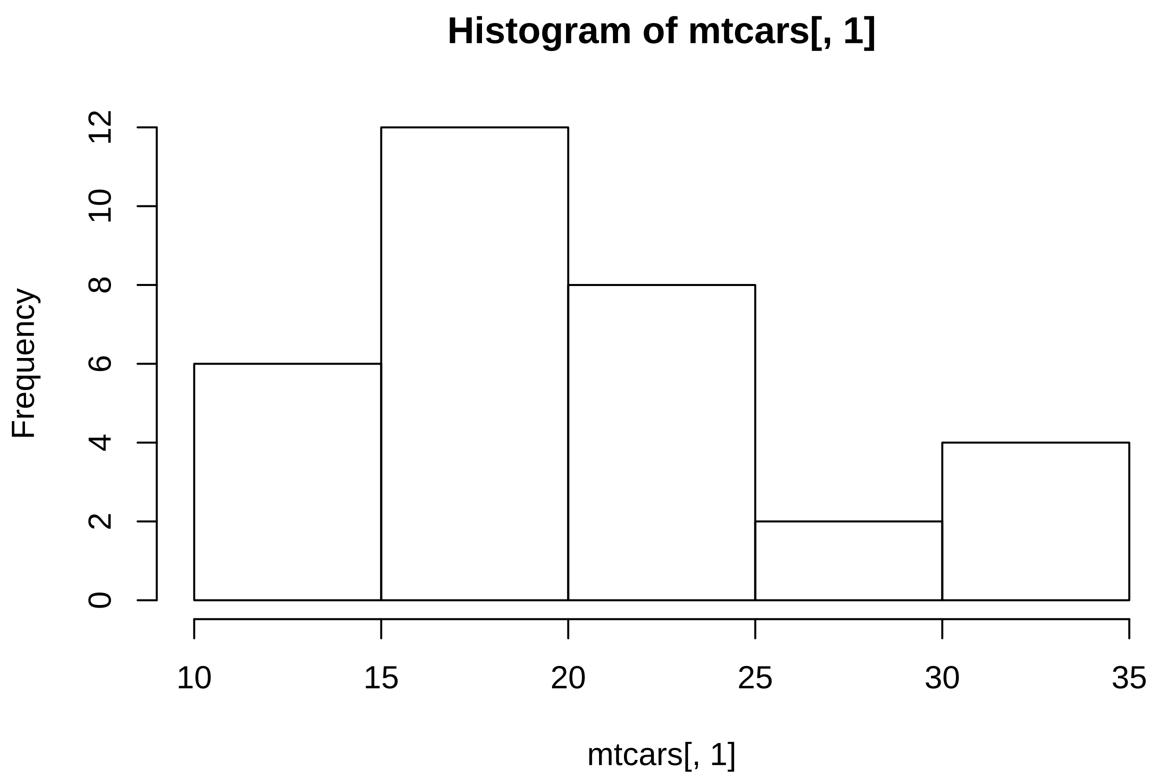
T-Test

SAMPLE MEAN - 2 SAMPLE MEAN - 2 tail - 1 tail

```
# Using dataset mtcars

summary(mtcars)
```

```
##       mpg             cyl             disp             hp       
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0  
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
##       drat             wt             qsec             vs        
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
##        am              gear            carb      
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
##  Median :0.0000   Median :4.000   Median :2.000  
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812  
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

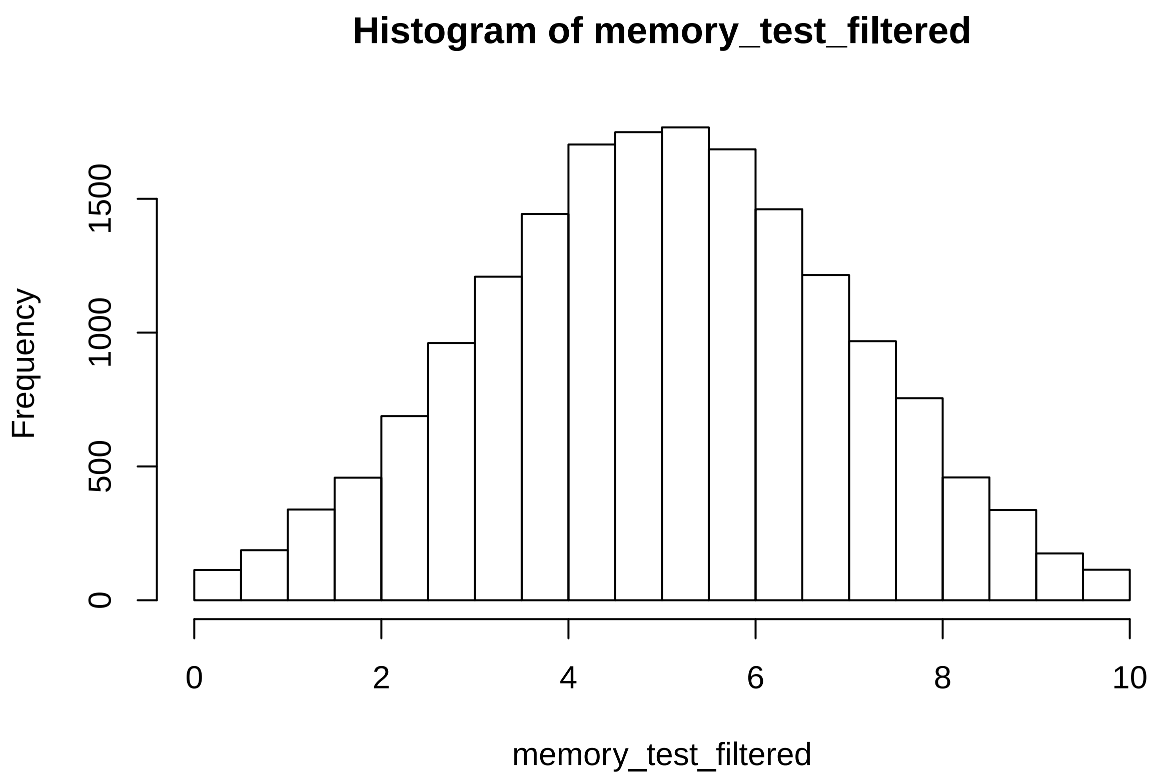
```
# extract mpg
hist(mtcars[,1])
```




```r
# test significantly different than 20 -- get confidence interval of 0.8
t.test(mtcars[,1], mu = 20, conf.level = 0.8)
```

```
##
##  One Sample t-test
##
## data:  mtcars[, 1]
## t = 0.08506, df = 31, p-value = 0.9328
## alternative hypothesis: true mean is not equal to 20
## 80 percent confidence interval:
##  18.69549 21.48576
## sample estimates:
## mean of x
##  20.09062
```

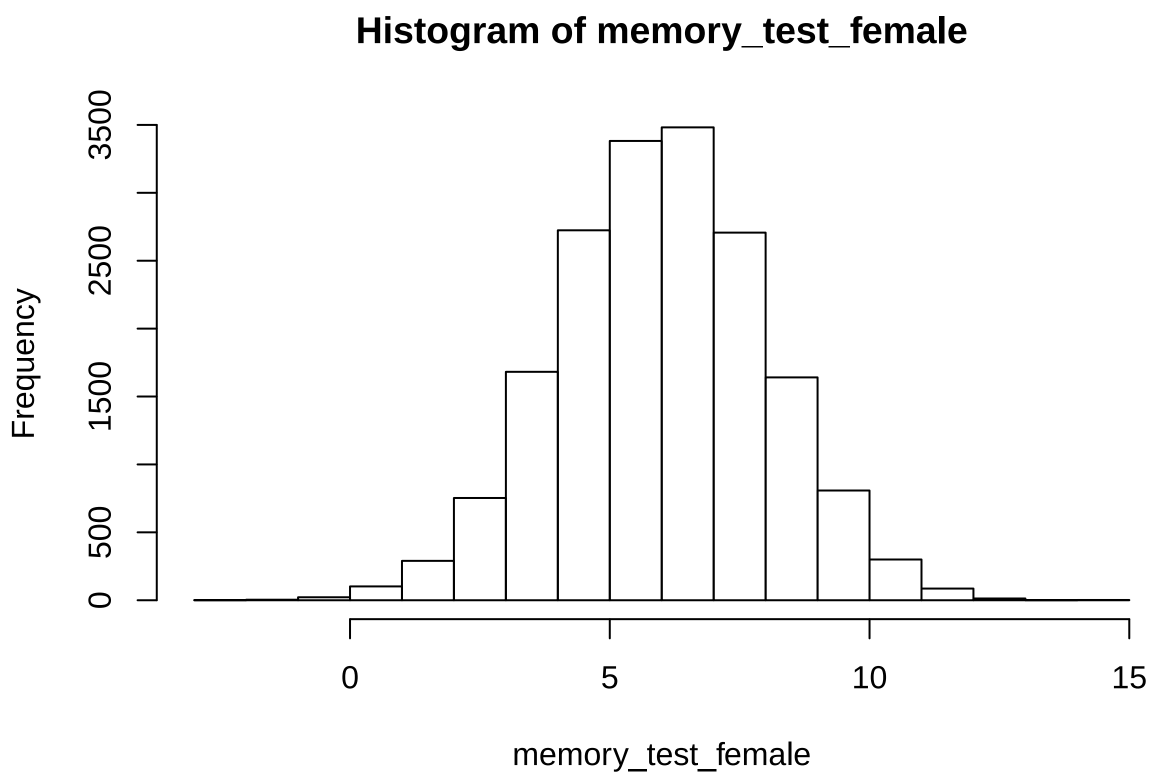
```r
# create fake data
# Sample 18000 scores from a normal distribution of mean 5 and sd 2
memory_test = rnorm(18000, mean = 5, sd = 2)

# take only the values between 0 and 10 (you could use also the function ifelse())
memory_test_filtered = memory_test[memory_test>0 & memory_test<10]
hist(memory_test_filtered)
```




```r
# test significantly different than 5
t.test(memory_test_filtered, mu=5)
```

```
##
##  One Sample t-test
##
## data:  memory_test_filtered
## t = 0.62811, df = 17785, p-value = 0.5299
## alternative hypothesis: true mean is not equal to 5
## 95 percent confidence interval:
##  4.980870 5.037172
## sample estimates:
## mean of x
##  5.009021
```

```r
# sample the same amount of subjects this time centered on 6
memory_test_female = rnorm(18000, mean = 6, sd = 2)
hist(memory_test_female)
```




```r
describe(memory_test_female)
```

```
##    vars     n mean sd median trimmed  mad   min   max range  skew kurtosis
## X1    1 18000    6  2   6.01       6 1.99 -2.09 14.77 16.86 -0.01        0
##      se
## X1 0.01
```

```r
# Filter it in the same way take only values between 0 and 10
memory_test_female_filtered = memory_test_female[memory_test_female>0 & memory_test_female<10]
hist(memory_test_female_filtered)
```


```r
# One tailed (alternative / alt = "greater")
t.test(memory_test_female_filtered, memory_test_filtered, alt = "greater")
```

```
##
##  Welch Two Sample t-test
##
## data:  memory_test_female_filtered and memory_test_filtered
## t = 44.282, df = 35350, p-value < 2.2e-16
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  0.8581758       Inf
## sample estimates:
## mean of x mean of y
##  5.900304  5.009021
```

PAIRED T-TEST

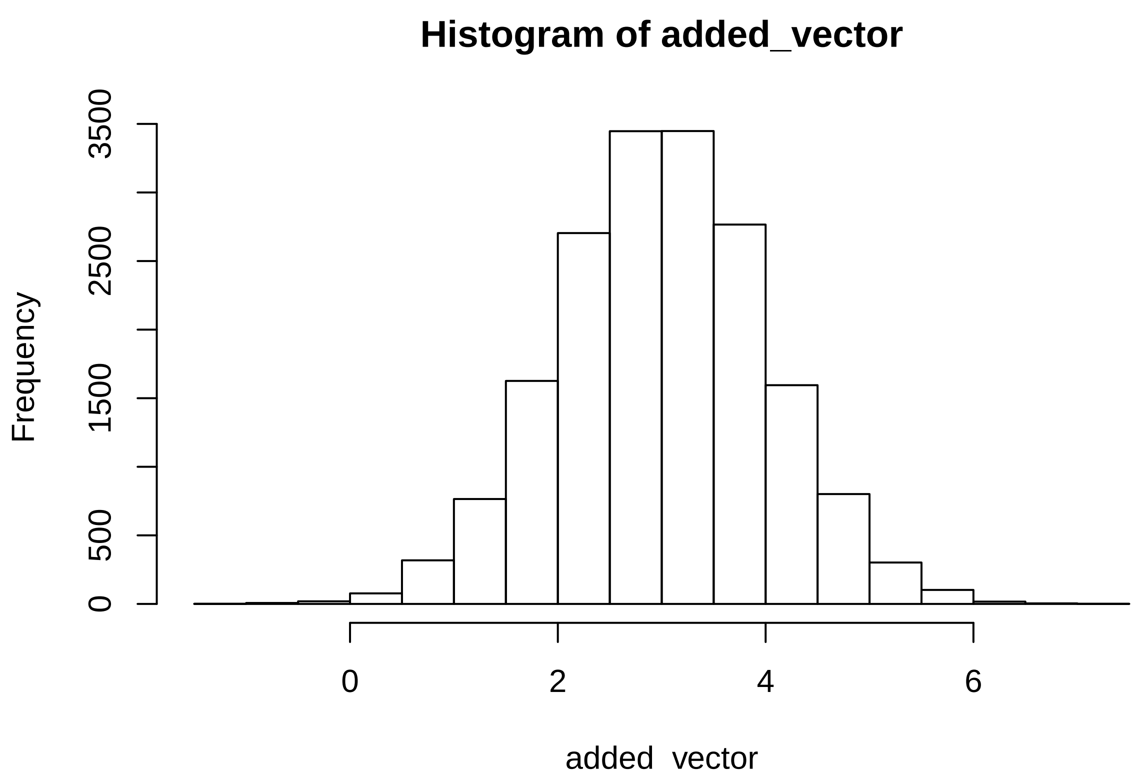
T1 -> Intervention -> T2 (repeated measure)

```r
## GROUP T1 - first test before the intervention
# sample fake data centered on 2
memory_test = rnorm(18000, mean = 2, sd = 1)

# a lot of the data is below 0
# but since we are going to use a paired t-test we want to keep the same amount of
# subjects and not delete them

# We need to REPLACE the values for subjects below 0
# If we replace all the values below 0 by 0 it will strongly alter the shape of the distribution
```

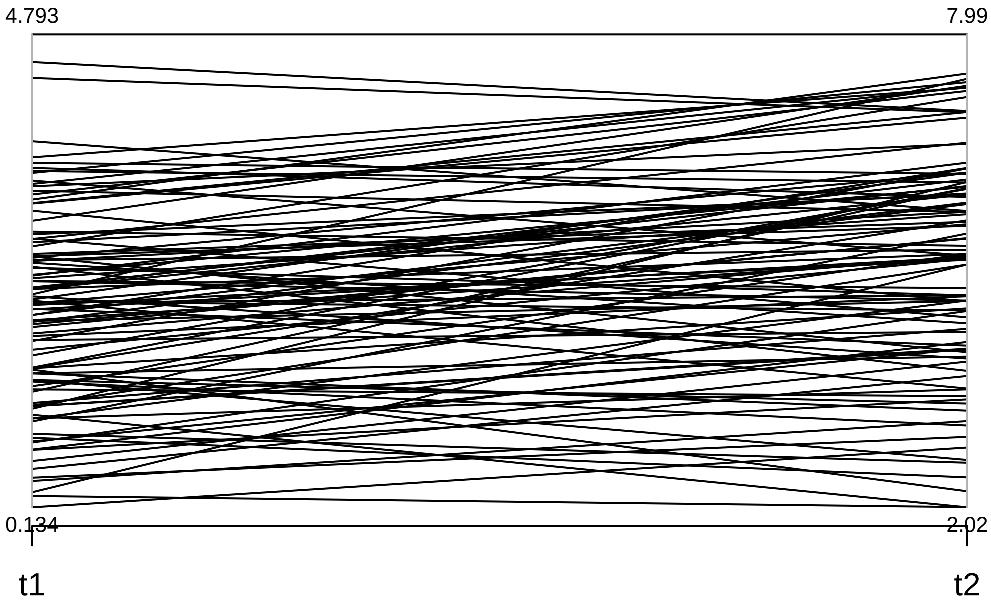
```r
memory_test[memory_test<0] = 0

# look at the first bin on the left is a lot higher than its symetric bin on the other side
# The distribution is skewed, which is not bad in itself, you just have to account for it by using
# robust statistics if the skeweness is too important
hist(memory_test)
```


```r
# Another way, to smooth your distribution a bit could be to replace by 0 + a random value between 0 an
# (this is fake data -- not something you will want to do with real data)
memory_test = rnorm(18000, mean = 2, sd = 1)

# get the number of subjects below 0
size_below_zero = length(memory_test[memory_test<0])

# get a vector of values between 0 and 3
sampled_vector = seq(from = 0, to = 3, by = 0.01)

# sample from that vector the same amount of time that there are subject below 0 in the memory test
sample_for_replacement = sample(sampled_vector, size = size_below_zero,replace = T)

# replace the values
memory_test[memory_test<0] = sample_for_replacement
hist(memory_test)
```


```
## GROUP T2 cw in between !
# We create the second vector by adding a random number sampled from a normal distribution
# with mean of 3 and sd of 1
added_vector = rnorm(length(memory_test), mean = 3, sd = 1)
hist(added_vector)
```



```r
# our intervention has a positive effect so we add this positive vector
memory_test_t2 = copy(memory_test) + added_vector

# we build our data.table with the two samples
pairs = data.table(t1 = memory_test, t2 = memory_test_t2)

# Check visually evolution of the 100 first subjects (parcoord is part of the MASS package)
parcoord(pairs[0:100,], var.label = T)
```




```r
# Paired T test with H0 difference is not significantly different than 0
t.test(memory_test_t2, memory_test , paired = T)
```

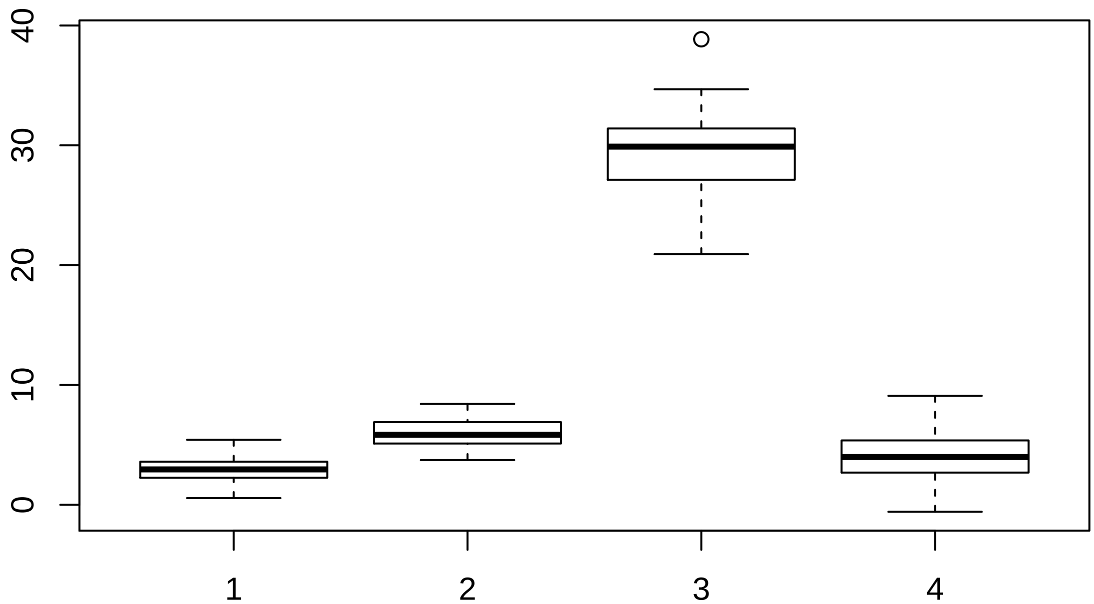
```
##
##  Paired t-test
##
## data:  memory_test_t2 and memory_test
## t = 402.29, df = 17999, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  2.99013 3.01941
## sample estimates:
## mean of the differences
##                3.00477
```

ANOVA

One variable

```r
# Check that the number of days of holidays between countries is different
chinese = rnorm(n = 150, mean = 3, sd = 1)
japan = rnorm(n = 30, mean = 6, sd = 1)
danemark = rnorm(n = 20, mean = 30, sd = 4)
usa = rnorm(n = 400, mean = 4, sd = 2)
```

```r
boxplot(chinese, japan, danemark, usa)
```




```r
# We need an indicator variable, we build it by hand
group = c(rep('chinese', 150),rep('japanese', 30), rep('danish', 20), rep('american', 400))

# This is our value variable
values = c(chinese, japan, danemark, usa)

# We create a data.table with those two corresponding vector
dt = data.table(g = group, v = values)
print(dt)
```

```
##             g        v
##   1:  chinese 4.522336
##   2:  chinese 3.245479
##   3:  chinese 5.111065
##   4:  chinese 2.120024
##   5:  chinese 1.944806
##  ---
## 596: american 5.418048
## 597: american 4.548744
## 598: american 6.585812
## 599: american 0.472787
## 600: american 1.810241
```

```r
# Now we can perform our anova test
aov_results = aov(dt, formula = v ~ g)
summary(aov_results)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## g             3  12747    4249    1241 <2e-16 ***
## Residuals   596   2040       3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# We can also test group difference one to one using Tukey
# Here they are all significant
tukey_results = TukeyHSD(aov_results)
print(tukey_results)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = v ~ g, data = dt)
##
## $g
##                        diff        lwr         upr p adj
## chinese-american   -1.06648  -1.522852  -0.6101093 0e+00
## danish-american    25.23529  24.143116  26.3274654 0e+00
## japanese-american   1.90587   1.003559   2.8081801 5e-07
## danish-chinese     26.30177  25.167084  27.4364587 0e+00
## japanese-chinese    2.97235   2.019022   3.9256782 0e+00
## japanese-danish   -23.32942 -24.705432 -21.9534105 0e+00
```

```r
# To use the function stack() the number of observation need to be similar
chinese = rnorm(n = 50, mean = 3, sd = 1)
japan = rnorm(n = 50, mean = 6, sd = 1)
danemark = rnorm(n = 50, mean = 30, sd = 4)
usa = rnorm(n = 50, mean = 4, sd = 2)
stack_variable = stack(data.frame(cbind(chinese, japan, danemark, usa)))
result = aov(stack_variable, formula = values ~ ind)
summary(result)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## ind           3  26189    8730    1473 <2e-16 ***
## Residuals   196   1162       6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```