Statistical Tests

Antonio Osamu Katagiri Tanaka - A01212611@itesm.mx

March 27, 2020

```
# Clear all objects (from the workspace)
rm(list = ls())

# Suppress Warning messages
options(warn = -1)

# Turn off scientific notation like 1e+06
# options(scipen=999)

# Load Libs
library(GEOquery)
```

```
## Loading required package: Biobase

## Loading required package: BiocGenerics

## Loading required package: parallel

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colMeans,
##     colnames, colSums, dirname, do.call, duplicated, eval, evalq,
##     Filter, Find, get, grep, grepl, intersect, is.unsorted, lapply,
##     lengths, Map, mapply, match, mget, order, paste, pmax, pmax.int,
##     pmin, pmin.int, Position, rank, rbind, Reduce, rowMeans, rownames,
##     rowSums, sapply, setdiff, sort, table, tapply, union, unique,
##     unsplit, which, which.max, which.min

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

## Setting options('download.file.method.GEOquery'='auto')

## Setting options('GEOquery.inmemory.gpl'=FALSE)
```

```
library(Biobase)
library(limma)
```

```
##
## Attaching package: 'limma'
## The following object is masked from 'package:BiocGenerics':
##
##     plotMA
```

```r
library(affy)
library(siggenes)
```

```
## Loading required package: multtest
```

```
## Loading required package: splines
```

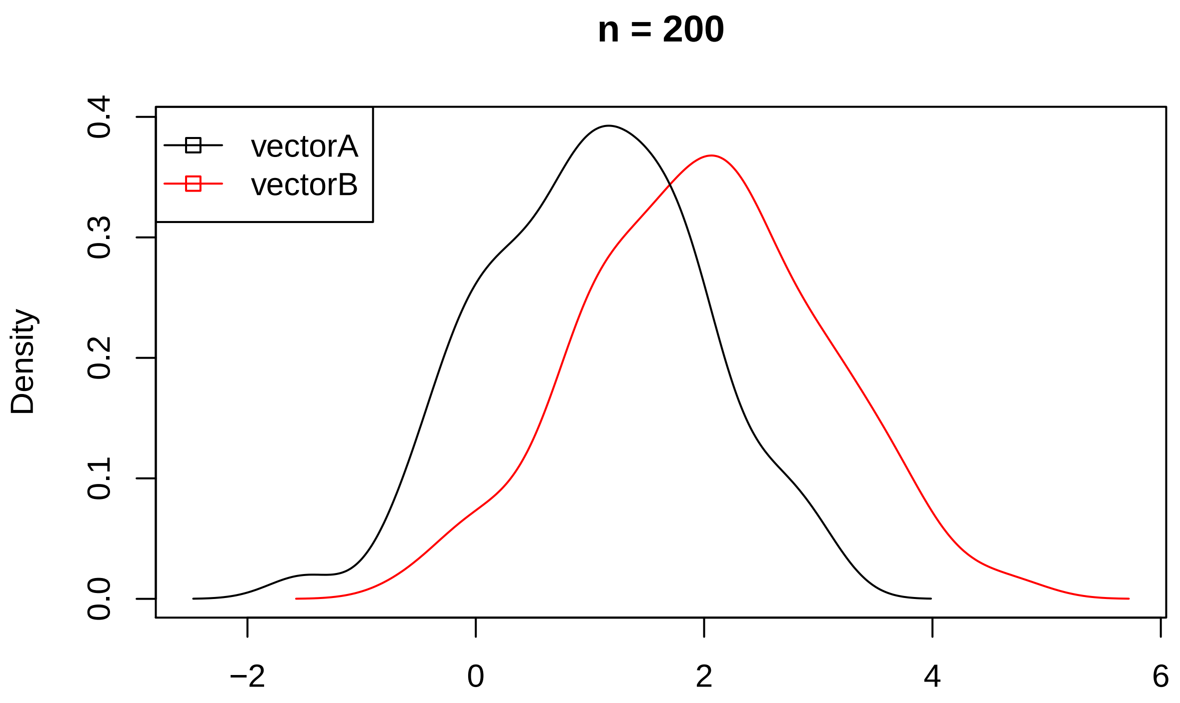
```r
source("./statistical_tests_lib.R")
```

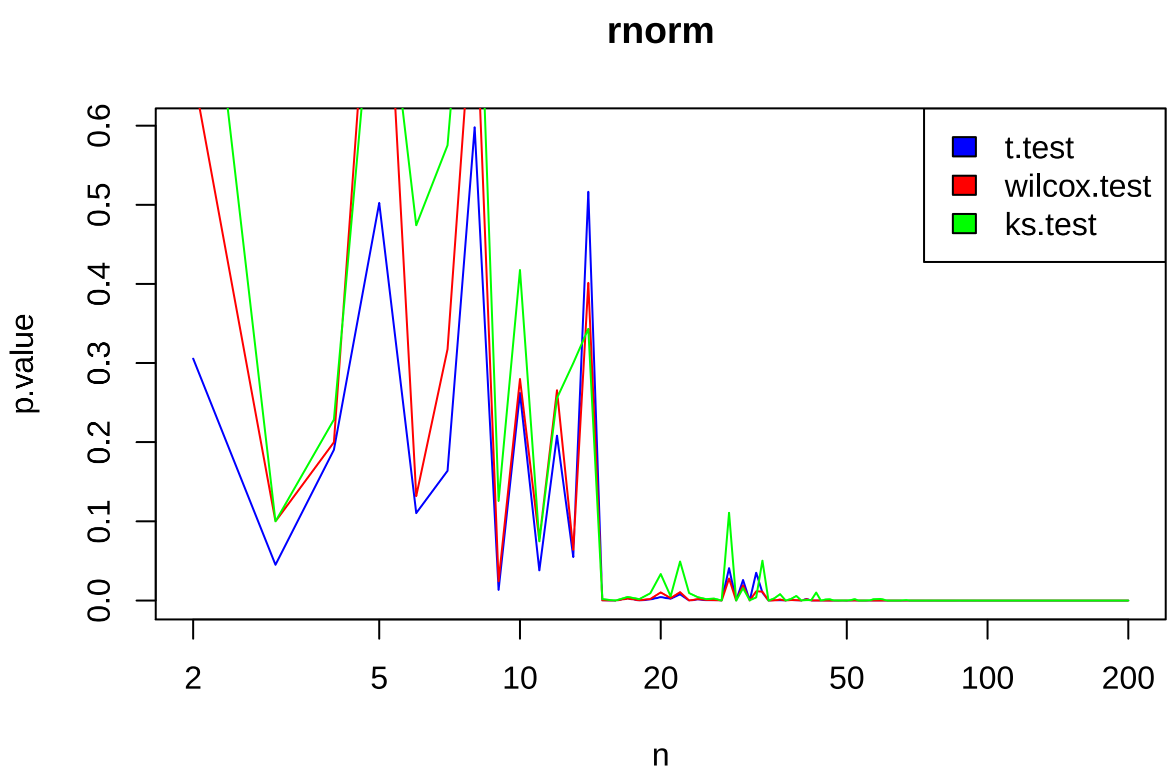
---

1) Obtener p-values con un t-test, Wilcoxon y Kolmogorov para:

A) 2 vectores de datos con distribución normal (rnorm) y diferente media, para número de muestras n= 2. . . . . . 20. Graficar los resultados y comparar.

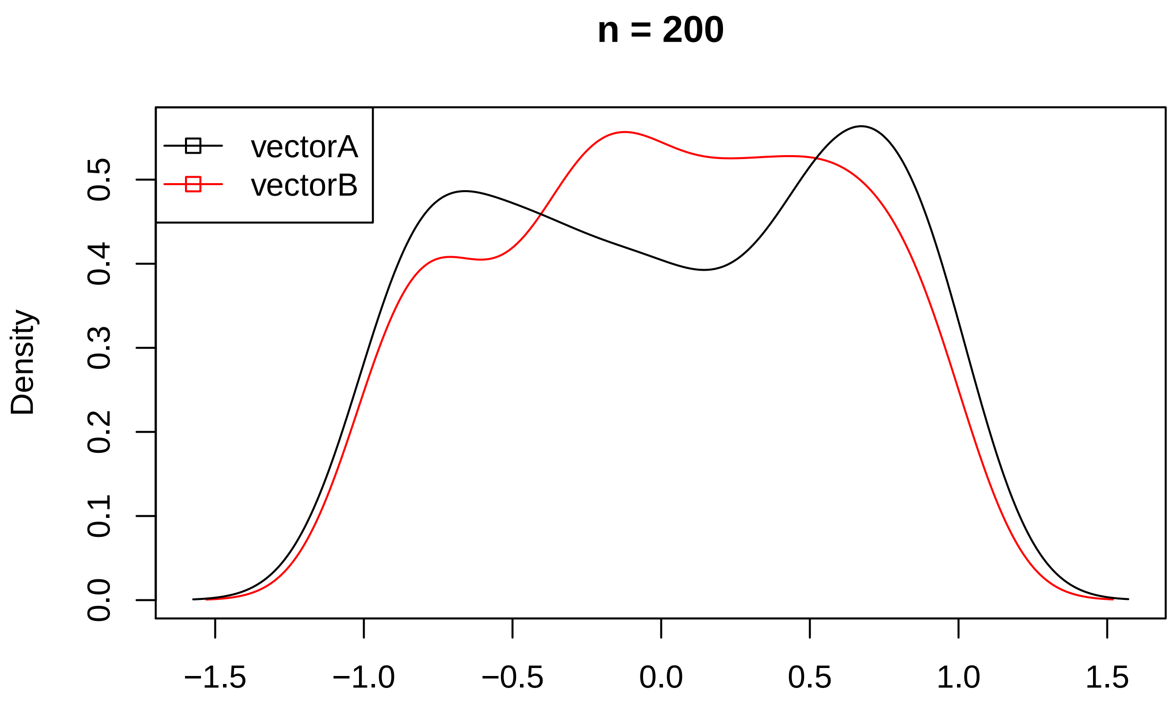
```r
getPvalues <- function(dist) {
  ns = 2:200
  ttest_pval = c()
  wtest_pval = c()
  ktest_pval = c()

  for (n in ns) {
    if (dist == "rnorm") {
      vectorA = rnorm(n = n, mean = 1, sd = 1)
      vectorB = rnorm(n = n, mean = 2, sd = 1)
    } else {
      vectorA = runif(n, -1, 1)
      vectorB = runif(n, -1, 1)
    }

    if (n == max(ns)) {
      plot.densities(vectorA, vectorB, main = paste0("n = ", n))
    }

    ttest = t.test(vectorA, vectorB)
    ttest_pval = c(ttest_pval, ttest$p.value)
    #print(ttest$p.value)

    wtest = wilcox.test(vectorA, vectorB)
    wtest_pval = c(wtest_pval, wtest$p.value)
    #print(wtest$p.value)

    ktest = ks.test(vectorA, vectorB)
    ktest_pval = c(ktest_pval, ktest$p.value)
    #print(ktest$p.value)
  }

  if (dist == "rnorm") {
    plot(
      ns,
      ttest_pval,
      main = "rnorm",
      xlab = "n",
      ylab = "p.value",
      type = "l",
      col = "blue",
      log = "x"
    )
  } else {
    plot(
      ns,
      ttest_pval,
      main = "runif",
      xlab = "n",
      ylab = "p.value",
      type = "l",
      col = "blue"
    )
  }

  lines(ns, wtest_pval,
        col = "red")

  lines(ns, ktest_pval,
        col = "green")

  legend("topright",
         c("t.test", "wilcox.test", "ks.test"),
         fill = c("blue", "red", "green"))
}

getPvalues("rnorm")
```

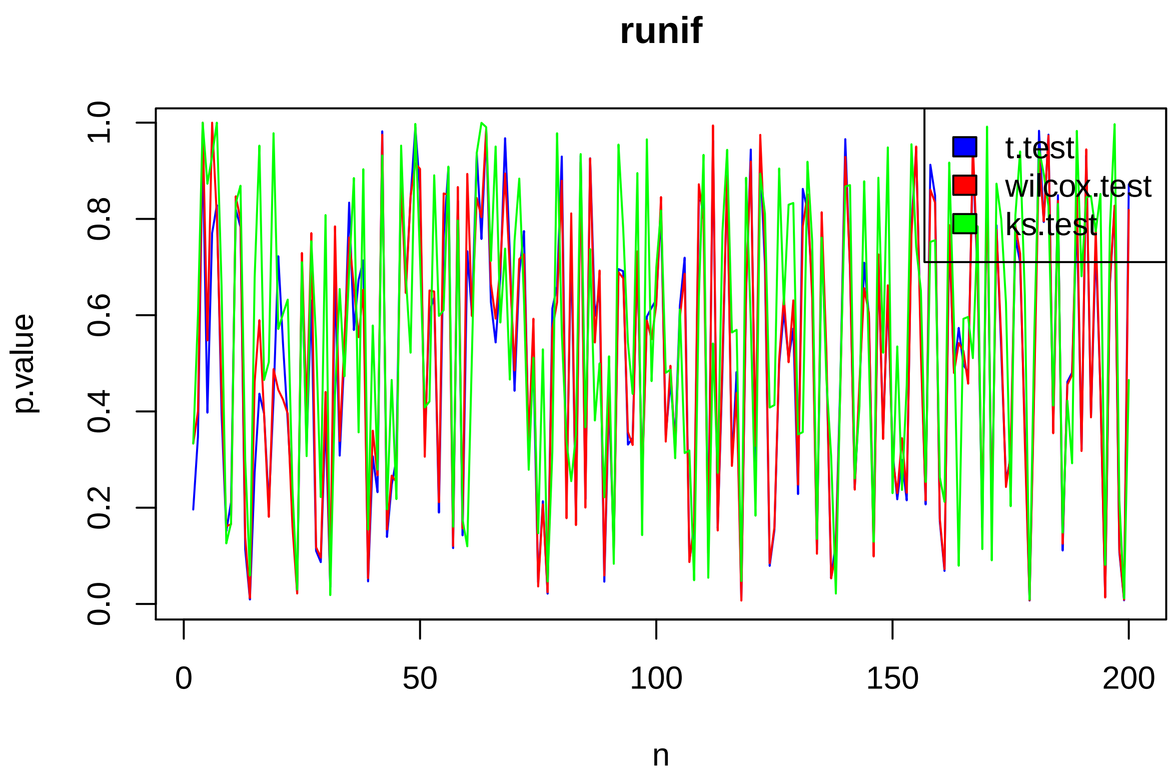




B) Repetir (A) con 2 vectores de datos generados con la función runif.

```r
getPvalues("runif")
```





2) Obtener p-values con un t-test, Wilcoxon, Kolmogorov y SAM para una base de datos GEO de su elección.

Dataset details

Title: Severe acute respiratory syndrome expression profile

Summary: Expression profiling of peripheral blood mononuclear cells (PBMC) from 10 adult patients with severe acute respiratory syndrome (SARS). Results provide insight into the host immune response to the SARS coronavirus.

Organism: Homo sapiens

Platform: GPL201: [HG-Focus] Affymetrix Human HG-Focus Target Array

Citation:
Reghunathan R, Jayapal M, Hsu LY, Chng HH et al. *Expression profile of immune response genes in patients with Severe Acute Respiratory Syndrome.* BMC Immunol 2005 Jan 18;6:2. PMID: 15655079

Reference Series: GSE1739

Sample count: 14

Value type: count

Series published: 2005/01/18

Dataset taken from: GDSbrowser, GSE1739 query, and GSE1739 geo2r

The GSE1739 was selected for the Bioinformatics class assignments and activities. The reason behind this decision is the similarities that SARS shares with the novel coronavirus COVID-19. The purpose id to work with up-to-date data that is relecant to the current crisis.

Let's download .soft.gz from GEO and take a look to the data

```
databaseID = "GSE1739"
gse <- getGEO(databaseID, GSEMatrix = FALSE, destdir = "./NCBI_GEO")
```

```
## Using locally cached version of GSE1739 found here:
## ./NCBI_GEO/GSE1739.soft.gz

## Reading file....

## Parsing....

## Found 15 entities...

## GPL201 (1 of 16 entities)

## GSM30361 (2 of 16 entities)

## GSM30362 (3 of 16 entities)

## GSM30363 (4 of 16 entities)

## GSM30364 (5 of 16 entities)

## GSM30365 (6 of 16 entities)

## GSM30366 (7 of 16 entities)

## GSM30367 (8 of 16 entities)

## GSM30368 (9 of 16 entities)

## GSM30369 (10 of 16 entities)

## GSM30370 (11 of 16 entities)

## GSM30371 (12 of 16 entities)

## GSM30372 (13 of 16 entities)

## GSM30373 (14 of 16 entities)

## GSM30374 (15 of 16 entities)
```

```
# names of all the GSM objects contained in the GSE
lst_gsm = GSMList(gse)
names(lst_gsm)
```

```
##  [1] "GSM30361" "GSM30362" "GSM30363" "GSM30364" "GSM30365" "GSM30366"
##  [7] "GSM30367" "GSM30368" "GSM30369" "GSM30370" "GSM30371" "GSM30372"
## [13] "GSM30373" "GSM30374"
```

```
# and get the 1st GSM object on the list
lst_gsm[[1]]
```

```
## An object of class "GSM"
## channel_count
## [1] "1"
## contact_address
## [1] " "
## contact_city
## [1] "Singapore"
## contact_country
## [1] "Singapore"
## contact_institute
## [1] "NUS"
## contact_name
## [1] "Jayapal,,Manikandan"
## contact_zip/postal_code
## [1] "117597"
## data_row_count
## [1] "8793"
## description
## [1] "PBMC normal sample RNA for Control"
## geo_accession
## [1] "GSM30361"
## last_update_date
## [1] "May 27 2005"
## molecule_ch1
## [1] "total RNA"
## organism_ch1
## [1] "Homo sapiens"
## platform_id
## [1] "GPL201"
## series_id
## [1] "GSE1739"
## source_name_ch1
## [1] "PBMC normal sample RNA"
## status
## [1] "Public on Jan 18 2005"
## submission_date
## [1] "Sep 08 2004"
## supplementary_file
## [1] "NONE"
## taxid_ch1
## [1] "9606"
## title
## [1] "N1"
## type
## [1] "RNA"
## An object of class "GEODataTable"
## ****** Column Descriptions ******
##               Column               Description
## 1             ID_REF
## 2              VALUE      raw signal intensity
## 3           ABS_CALL present, absent, marginal
## 4 DETECTION P-VALUE                    p-value
## ****** Data Table ******
##       ID_REF  VALUE ABS_CALL DETECTION P-VALUE
## 1 1007_s_at  321.8        P          0.035163
## 2   1053_at  204.8        P          0.006532
## 3    117_at  538.6        P          0.001141
## 4    121_at 1277.8        P          0.011447
## 5 1255_g_at   51.3        A          0.418069
## 8788 more rows ...
```

```r
# and the names of the GPLs represented
lst_gpl = GPLList(gse)
names(lst_gpl)
```

```
## [1] "GPL201"
```

Let's prepare the data for further analysis

First, we need to make sure that all of the GSMs are from the same platform:

```r
gsmplatforms <- lapply(lst_gsm, function(x) {
  Meta(x)$platform_id
})
length(gsmplatforms)
```

```
## [1] 14
```

```r
# If there are more GPLs, we can filter the original GSMList to include only those GSMs within
# a specific platform and use this list for further processing
```

```
gsmlist = Filter(function(gsm) {
  Meta(gsm)$platform_id == 'GPL201'
},
GSMList(gse))
length(gsmlist)
```

```
## [1] 14
```

So, now we would like to know what column represents the data that we would like to extract. Looking at the first few rows of the Table of a single GSM will likely give us an idea (and by the way, GEO uses a convention that the column that contains the single measurement for each array is called the VALUE column, which we could use if we don't know what other column is most relevant).

```
Table(gsmlist[[1]])[1:5, ]
```

```
##      ID_REF  VALUE ABS_CALL DETECTION P-VALUE
## 1 1007_s_at  321.8        P          0.035163
## 2   1053_at  204.8        P          0.006532
## 3    117_at  538.6        P          0.001141
## 4    121_at 1277.8        P          0.011447
## 5 1255_g_at   51.3        A          0.418069
```

go through the necessary steps to make a compliant ExpressionSet

```
gset = getGEO(databaseID, GSEMatrix = TRUE, destdir = "./NCBI_GEO")[[1]][1]
```

```
## Found 1 file(s)
```

```
## GSE1739_series_matrix.txt.gz
```

```
## Using locally cached version: ./NCBI_GEO/GSE1739_series_matrix.txt.gz
```

```
## Parsed with column specification:
## cols(
##   ID_REF = col_character(),
##   GSM30361 = col_double(),
##   GSM30362 = col_double(),
##   GSM30363 = col_double(),
##   GSM30364 = col_double(),
##   GSM30365 = col_double(),
##   GSM30366 = col_double(),
##   GSM30367 = col_double(),
##   GSM30368 = col_double(),
##   GSM30369 = col_double(),
##   GSM30370 = col_double(),
##   GSM30371 = col_double(),
##   GSM30372 = col_double(),
##   GSM30373 = col_double(),
##   GSM30374 = col_double()
## )
```

```
## Using locally cached version of GPL201 found here:
## ./NCBI_GEO/GPL201.soft
```

```
pD_gset = pData(phenoData(gset))
pD_gset[,c(1,12)]
```

```
##          title                       description
## GSM30361    N1 PBMC normal sample RNA for Control
## GSM30362    N2 PBMC normal sample RNA for Control
## GSM30363    N3 PBMC normal sample RNA for Control
## GSM30364    N4 PBMC normal sample RNA for Control
## GSM30365    S1        SARS patient blood sample 1
## GSM30366    S2        SARS patient blood sample 1
## GSM30367    S3        SARS patient blood sample 1
## GSM30368    S4        SARS patient blood sample 4
## GSM30369    S5        SARS patient blood sample 5
## GSM30370    S6        SARS patient blood sample 6
## GSM30371    S7        SARS patient blood sample 7
## GSM30372    S8        SARS patient blood sample 8
## GSM30373    S9        SARS patient blood sample 9
## GSM30374   S10       SARS patient blood sample 10
```

```
#We will indeed use the VALUE column. We then want to make a matrix of these values like so:

# get the probeset ordering
probesets <- Table(GPLList(gse)[[1]])$ID
# make the data matrix from the VALUE columns from each GSM
# being careful to match the order of the probesets in the platform
# with those in the GSMs
data.matrix <- do.call('cbind', lapply(gsmlist, function(x)
{
  tab <- Table(x)
  mymatch <-
    match(probesets, tab$ID_REF)
  return(tab$VALUE[mymatch])
}))
data.matrix <-
  apply(data.matrix, 2, function(x) {
    as.numeric(as.character(x))
  })
```

A) Verifique si necesita normalizar los datos con quantile normalization. Obtenga p-values antes y después de normalizar.

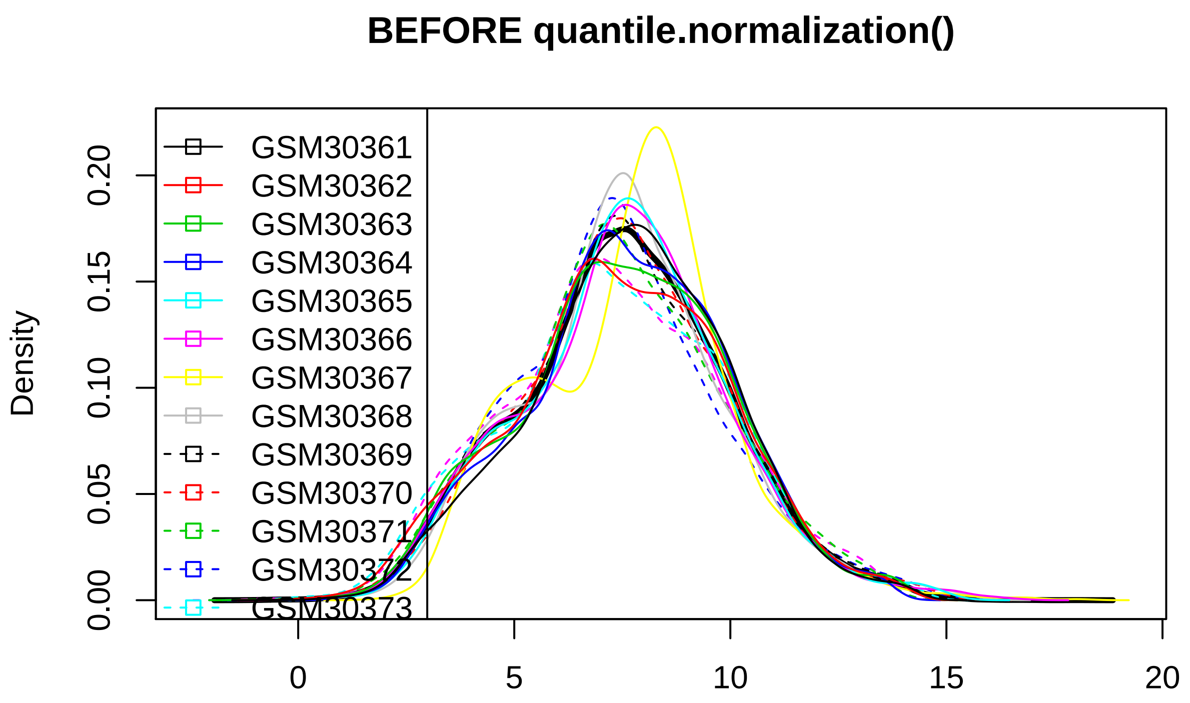
```
dm_all <- log2(data.matrix)

# Normal (control) samples
dm_norm = dm_all[1:5,1:4]
dm_norm
```

```
##       GSM30361 GSM30362  GSM30363  GSM30364
## [1,]  8.330021 8.006186  8.370687  8.518850
## [2,]  7.678072 8.197217  7.764208  8.032321
## [3,]  9.073070 8.519636  9.047124  8.500244
## [4,] 10.319446 9.781688 10.010948 10.016948
## [5,]  5.680887 5.459432  5.375039  4.722466
```

```
# SARS samples
dm_sars = dm_all[1:5,5:14]
dm_sars
```

```
##      GSM30365 GSM30366  GSM30367  GSM30368  GSM30369 GSM30370  GSM30371
## [1,] 7.680887 7.806711  8.033974  8.292322  7.997179 7.879583  7.622052
## [2,] 7.753551 8.053111  5.809929  7.668885  7.488644 7.089583  7.566054
## [3,] 9.132114 8.513333  8.106432 10.018617  9.144403 9.201389 10.172177
## [4,] 9.627351 9.988827 10.186981 10.215654 10.021813 9.954778  9.737585
## [5,] 5.741467 5.189825  4.478972  6.203593  5.672425 5.409391  6.002252
##       GSM30372 GSM30373 GSM30374
## [1,]  7.624978 8.186857 7.050937
## [2,]  7.309249 6.072535 7.260214
## [3,] 10.452653 8.000000 8.724855
## [4,] 10.087728 9.471472 9.545737
## [5,]  5.057450 5.133399 6.213347
```

```
plot.densities(dm_all,  main = "BEFORE quantile.normalization()")
```



There is small-to-large variability within groups and small variability across groups, so, let's use quantile normalization

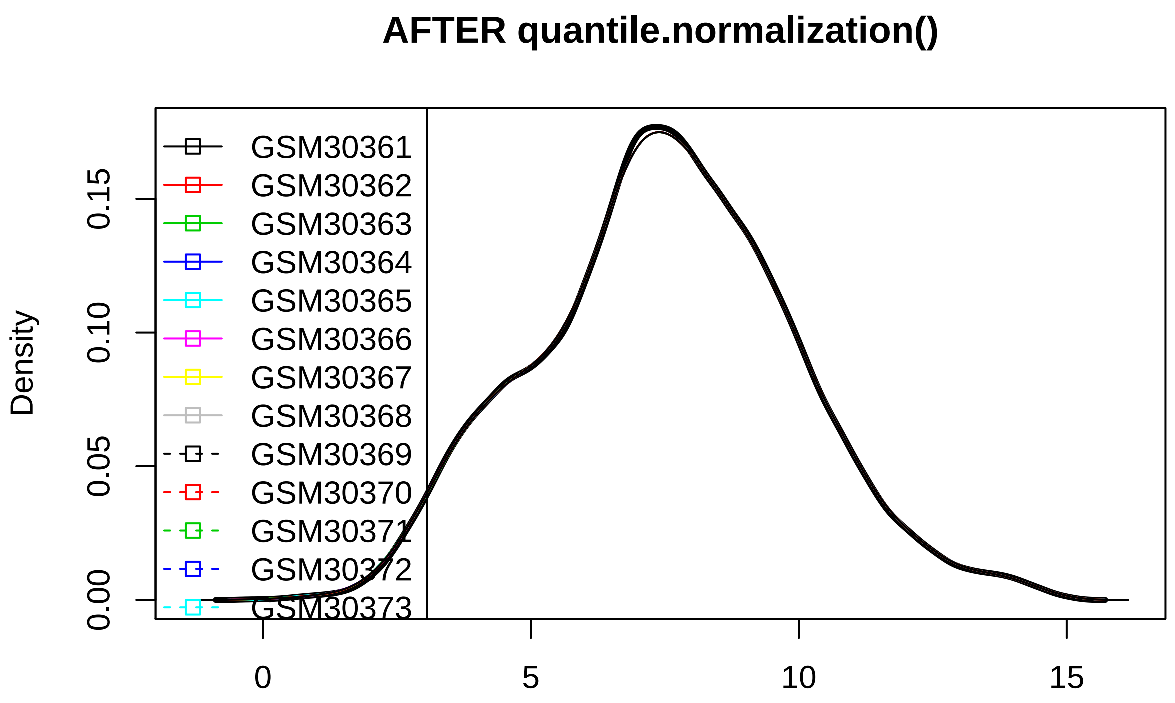
```
dm_all_qn <- quantile.normalization(dm_all)

# Normal (control) samples
dm_norm_qn = dm_all_qn[1:5,1:4]
dm_norm_qn
```

```
##       GSM30361 GSM30362 GSM30363 GSM30364
## [1,]  8.161136 7.974546 8.248358 8.339551
## [2,]  7.484716 8.128818 7.679727 7.878717
## [3,]  8.921154 8.436491 8.919724 8.325097
## [4,] 10.295475 9.696545 9.947566 9.923834
## [5,]  5.363346 5.529863 5.348546 4.545038
```

```
# SARS samples
dm_sars_qn = dm_all_qn[1:5,5:14]
dm_sars_qn
```

```
##      GSM30365  GSM30366  GSM30367  GSM30368  GSM30369 GSM30370  GSM30371
## [1,] 7.600519  7.757225  7.655003  8.379657  8.062848 7.919740  7.774365
## [2,] 7.691611  8.013826  5.676884  7.691611  7.530020 7.100157  7.720626
## [3,] 9.154704  8.527373  7.759264 10.053997  9.154704 9.220894 10.115654
## [4,] 9.645563 10.051500 10.366689 10.270498 10.033226 9.959321  9.731034
## [5,] 5.638145  5.224321  4.086702  6.078995  5.683874 5.350561  6.114970
##       GSM30372 GSM30373 GSM30374
## [1,]  7.859851 8.239157 7.328423
## [2,]  7.522946 6.324003 7.528983
## [3,] 10.489992 8.078678 8.777586
## [4,] 10.138742 9.368622 9.554931
## [5,]  5.124808 5.425214 6.534890
```

```
plot.densities(dm_all_qn, main = "AFTER quantile.normalization()")
```



Compute the p-values for t-test, Wilcoxon, Kolmogorov & SAM

```
compare_groups = c(rep("control", 4), rep("sars", 10))
ex = exprs(gset)
ex_log2 = log2(ex + 1)
cbind(compare_groups, ex_log2[,])
```

```
##          compare_groups
## GSM30361 "control"      "8.33449676839042"
## GSM30362 "control"      "8.01178633120661"
## GSM30363 "control"      "8.37503943134693"
## GSM30364 "control"      "8.52277766952458"
## GSM30365 "sars"         "7.68790052248075"
## GSM30366 "sars"         "7.81313985089045"
## GSM30367 "sars"         "8.03946743827853"
## GSM30368 "sars"         "8.29691620687929"
## GSM30369 "sars"         "8.00281501560705"
## GSM30370 "sars"         "7.8856963733394"
## GSM30371 "sars"         "7.62935662007961"
## GSM30372 "sars"         "7.63226821549951"
## GSM30373 "sars"         "8.19179950106508"
## GSM30374 "sars"         "7.06177619758669"
```

```
print("")
```

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

```
# # Perform the stat tests
# vd_ctrl_ttest = apply(ex_log2,1,function(x){
#   aux = t.test(x[which(compare_groups == "control")],x[which(compare_groups == "sars")])
#   aux$p.value
# })
# vd_ctrl_ttest
# vd_ctrl_wtest = apply(ex_log2,1,function(x){
#   aux = wilcox.test(x[which(compare_groups == "control")],x[which(compare_groups == "sars")])
```

```r
#   aux$p.value
# })
# vd_ctrl_wtest
# vd_ctrl_ktest = apply(ex_log2,1,function(x){
#   aux = ks.test(x[which(compare_groups == "control")],x[which(compare_groups == "sars")])
#   aux$p.value
# })
# vd_ctrl_ktest

# # Perform the stat tests
# t.test(ex_log2[,which(compare_groups == "control")],ex_log2[,which(compare_groups == "sars")])
# wilcox.test(ex_log2[,which(compare_groups == "control")],ex_log2[,which(compare_groups == "sars")])
# ks.test(ex_log2[,which(compare_groups == "control")],ex_log2[,which(compare_groups == "sars")])
# dm_all_sam = sam(dm_all, c(rep(1, 4), rep(0, 10)), method = "d.stat")
```

```r
calPvalues <- function(data.matrix) {
  pval = de.test(
    x = data.matrix,
    classes = c(rep(1, 4), rep(0, 10)),
    test = c("ttest", "kolmogorov", "wilcoxon")
  )
  data.matrix_sam <- sam(data.matrix,
                         c(rep(1, dim(data.matrix)[2])),
                         method = "d.stat",
                         gene.names = colnames(data.matrix))

  return(
    list(
      "ttest_pval" = pval$ttest,
      "wtest_pval" = pval$kolmogorov,
      "ktest_pval" = pval$wilcoxon,
      "sam_pval"   = unname(data.matrix_sam@p.value)
    )
  )

  # #Mostrar resultados de SAM con cierto valor de delta
  # data.matrix_sam_sum <- summary(data.matrix_sam, 1.9)
  #
  # #Acceder a matrix de Genes significativos
  # dim(data.matrix_sam_sum@mat.sig)
}

# [TODO: make correlation matrix plot w/individual plots for each]
plt_pval <- function(pval) {
  for (i in 1:length(pval)) {
    ns = 1:length(pval[[i]])
    plot(
      ns,
      pval[[i]],
      main = "runif",
      xlab = "",
      ylab = "p.value",
      #log = "y",
      col = "blue",
      type = "p"
    )
  }
}

# lines(ns, ttest_pval,
#       col = "red",
#         type = "p")
#
# lines(ns, wtest_pval,
#         col = "green",
#         type = "p")
#
# lines(ns, ktest_pval,
#         col = "magenta",
#         type = "p")
#
# legend(
#    "topright",
#    c("sam", "t.test", "wilcox.test", "ks.test"),
#    fill = c("blue", "red", "green", "magenta")
# )
```

p-values BEFORE quantile.normalization()

```
pval = calPvalues(dm_all)
```

```
## We're doing 16384 complete permutations
## and randomly select 100 of them.
```

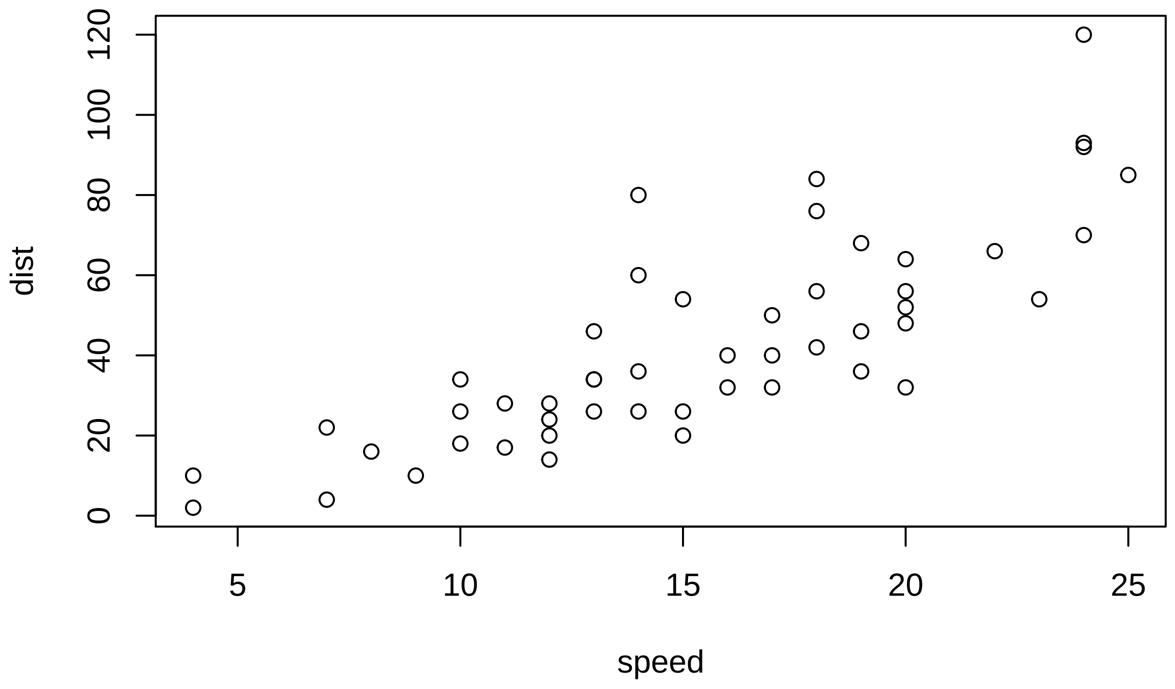
p-values AFTER quantile.normalization()

```
pval_qn = calPvalues(dm_all_qn)
```

```
## We're doing 16384 complete permutations
## and randomly select 100 of them.
```

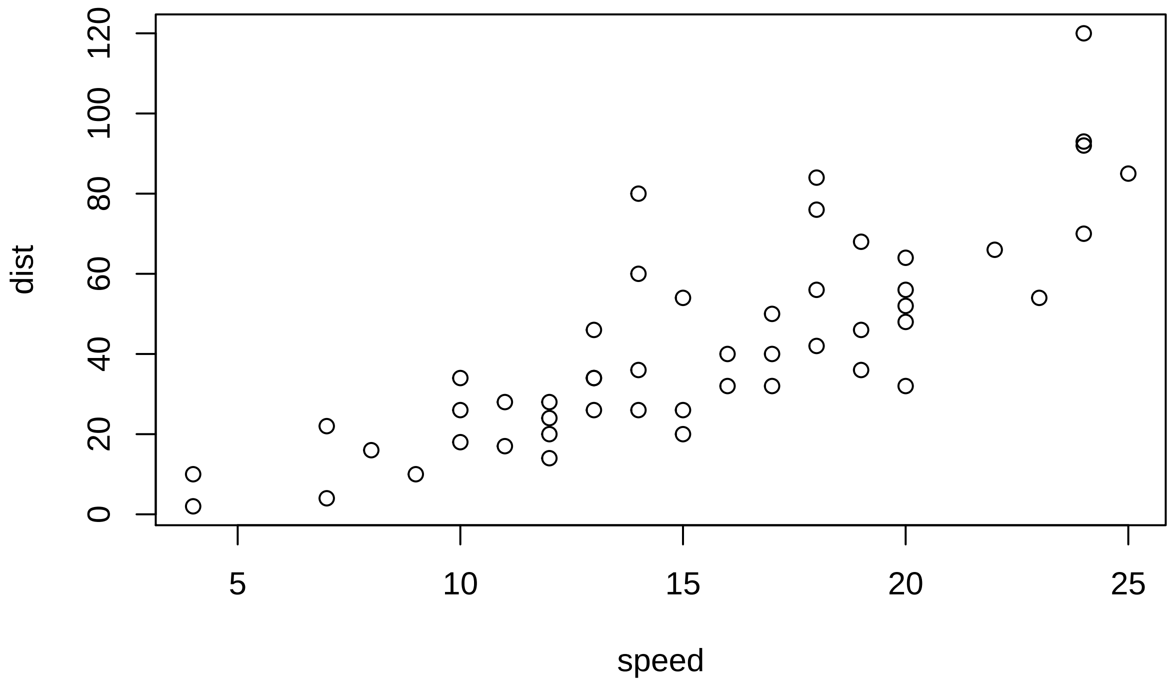
B) Compare los p-values y grafique. Comente sobre la correlación entre las distintas pruebas estadísticas y la cantidad de genes significativos.

```
plot(cars)
```



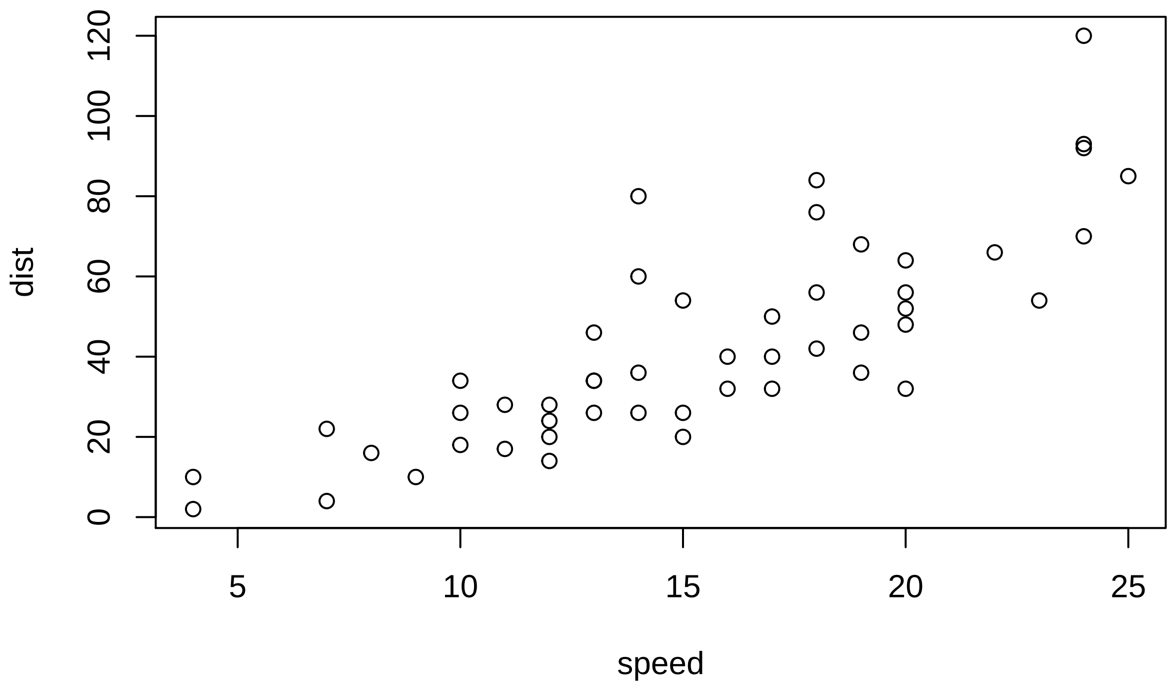
C) Usando el GPL de los datos de GEO, dar una significancia biológica de los genes significativos.

```r
plot(cars)
```



3) Repetir el ejercicio (2) pero con una base de datos de GEO con menos o más muestras, según sea el caso.

```
plot(cars)
```



```
# Enable Warning messages
options(warn = 0)
```