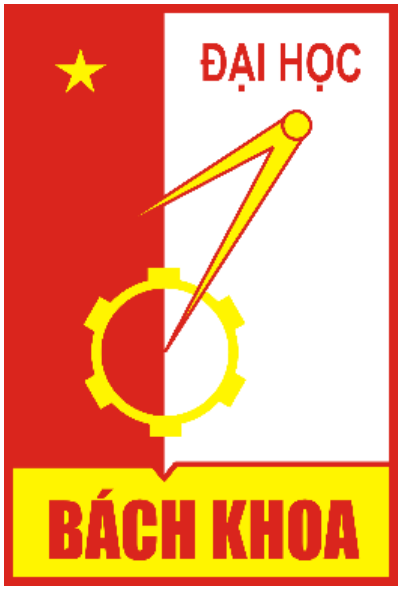
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

----- *** -----



BÁO CÁO CUỐI KÌ TIN SINH HỌC

Đề tài: Exploratory Analysis of Biological Data using R 2017

Giảng viên hướng dẫn: TS. Nguyễn Hồng Quang

Sinh viên thực hiện	MSSV
Vũ Văn Trung	20183846

Hà Nội, ngày 04 tháng 08 năm 2022

Lời nói đầu

Trong thời gian thực hiện và hoàn thành bài tập lớn môn học "Tin sinh học" với đề tài "Exploratory Analysis of Biological Data using R 2017", chúng em xin được gửi lời cảm ơn tới thầy Nguyễn Hồng Quang - Bộ môn Truyền thông và Mạng máy tính – Trường Công nghệ Thông tin và Truyền thông – Trường Đại học Bách khoa Hà Nội đã hướng dẫn và giúp đỡ chúng em hoàn thành môn học này. Thông qua môn học, chúng em đã được tiếp cận những kiến thức mới cũng như vận dụng những kiến đã học vào việc hoàn thiện bài tập lớn. Từ đó, giúp chúng em có cái nhìn bao quát và thực tế hơn về những kiến thức từ sách vở mà chúng em đã được học. Mặc dù bản thân chúng em đã cố gắng nhưng do thời gian, kiến thức và kinh nghiệm có hạn nên sản phẩm và báo cáo còn có thể có nhiều thiếu xót, vì vậy chúng em rất mong nhận được sự thông cảm và nhận xét từ thầy để bài tập lớn của chúng em được hoàn thiện hơn.

Project gồm 5 module: - Module 1: Exploratory Data Analysis - Module 2: Regression - Module 3: Demension Reduction - Module 4: Clustering - Module 5: Hypothesis Testing - Kết luận

Một lần nữa, chúng em xin chân thành cảm ơn thầy!

Mục lục

Mã nguồn

Chương 1: R_EDA-Introduction
Chương 2: R_EDA-Regression
Chương 3: R_EDA-DimensionReduction
Chương 4: R_EDA-Clustering
Chương 5: R_EDA-HypothesisTesting

Module 1: Exploratory Data Analysis

I.Lý thuyết

EDA - Exploratory Data Analysis(Phân tích khám phá dữ liệu) là một phương pháp phân tích dữ liệu chủ yếu sử dụng các kỹ thuật về biểu đồ, hình vẽ nhằm:

- Khám phá cấu trúc cơ bản của tập dữ liệu
- Xác định các biến quan trọng
- Phát hiện xu hướng
- Phát hiện những điểm dị thường và ngoại lệ
- Phát hiện lỗi và thiếu dữ liệu
- Phát triển các mô hình thống kê

Cùng với đó, mục đích của EDA là tạo ra giả thuyết chứ không phải kiểm tra giả thuyết của chúng ta dựa trên tập dữ liệu nên EDA thường là bước đầu tiên của quá trình phân tích dữ liệu.
EDA nhấn mạnh việc nhìn nhận dữ liệu theo các cách khác nhau thông qua:

- Tính toán và lập các bảng mô tả cơ bản của các thuộc tính của dữ liệu như phạm vi, phương tiện, phương sai,…
- Tạo các hình mô tả như các hộp, biểu đồ, biểu đồ phân tán,…
- Áp dụng các phép biến đổi cần thiết như xếp hạng,…
- So sánh các quan sát với các mô hình thống kê như biểu đồ hồi quy tuyến tính và phi tuyến tính
- Đơn giản hóa dữ liệu.
- Xác định cấu trúc cơ bản thông qua phân cụm
- ⇨ Mục tiêu cuối cùng là xác định mô hình thống kê nào phù hợp sử dụng để kiểm tra giả thuyết và dự đoán.

Bài Workshop này sử dụng ngôn ngữ lập trình R cho EDA bởi:

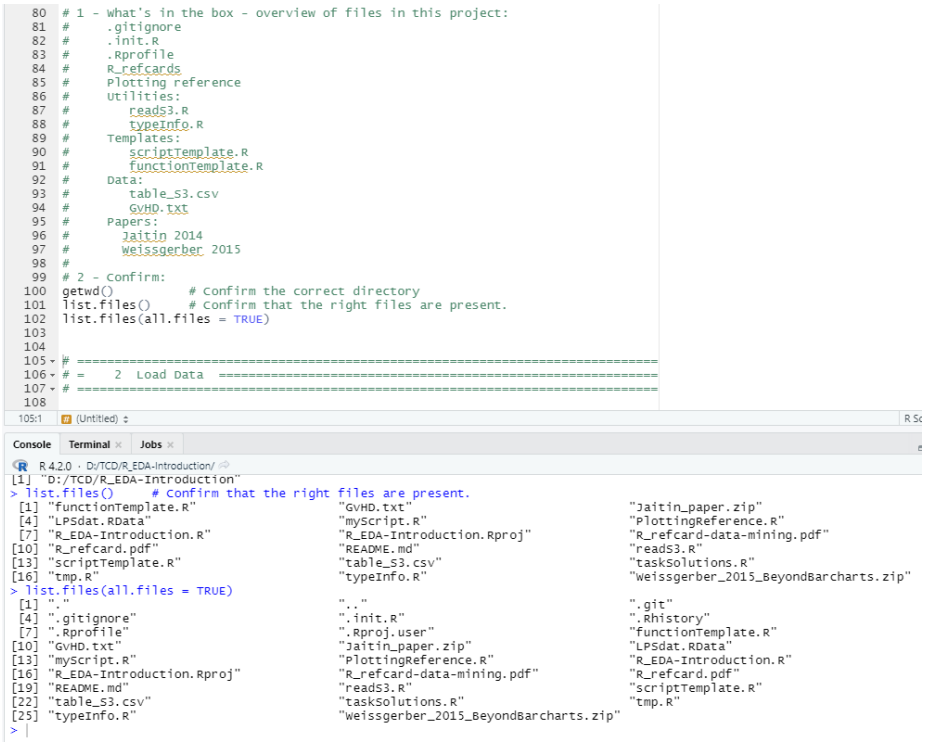
- R là ngôn ngữ lập trình đầy đủ tính năng
- Có thể coi R là “bàn làm việc thống kê”
- Thao tác dữ liệu dễ dàng
- Dễ dàng truy cập vào các hình mô tả
- Cộng đồng lớn

Đối với EDA thì sử dụng các hình mô tả là điều cần thiết và tuân theo quy tắc: sử dụng ít các từ ngữ. Có 1 lưu ý khá quan trọng khi sử dụng các hình mô tả là:

- Đảm bảo tất cả các yếu tố trên hình mô tả là cần thiết
- Đảm bảo tất cả các yếu tố trên hình mô tả là thông tin
- Đảm bảo tất cả các thông tin trong tệp dữ liệu đều đã được hiển thị.

II.Thực hành

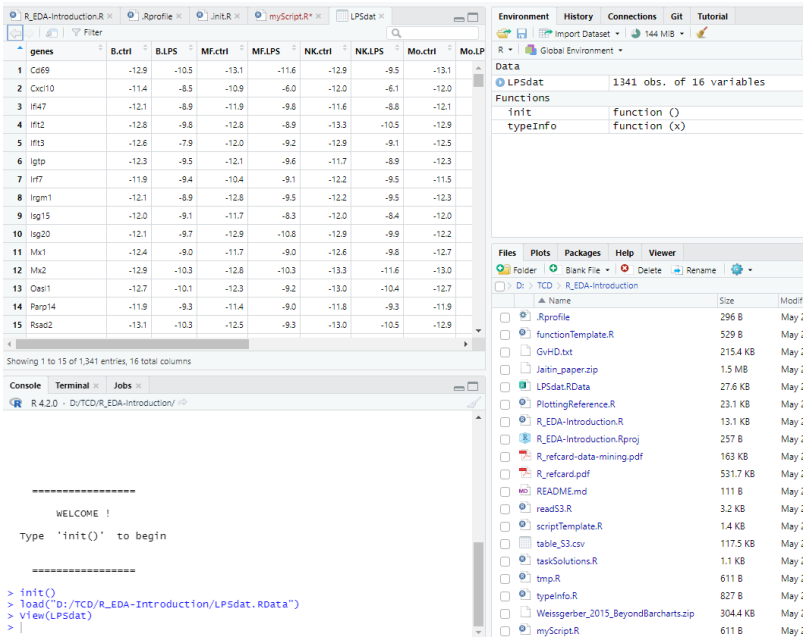
1.Project files and Setup:



```
# 1 - What's in the box - overview of files in this project:
#     .gitignore
#     .init.R
#     .Rprofile
#     R_refcards
#     Plotting reference
#     Utilities:
#        readS3.R
#        typeInfo.R
#     Templates:
#        scriptTemplate.R
#        functionTemplate.R
#     Data:
#        table_S3.csv
#        GvHD.txt
#     Papers:
#       Jaitin 2014
#       Weissgerber 2015
#
# 2 - Confirm:
getwd()          # Confirm the correct directory
list.files()     # Confirm that the right files are present.
list.files(all.files = TRUE)


# ============================================================================
# =    2  Load Data  =========================================================
# ============================================================================
```

```
Console  Terminal  Jobs
R 4.2.0 · D:/TCD/R_EDA-Introduction/
[1] "D:/TCD/R_EDA-Introduction"
> list.files()      # Confirm that the right files are present.
 [1] "functionTemplate.R"                "GvHD.txt"                          "Jaitin_paper.zip"
 [4] "LPSdat.RData"                      "myScript.R"                        "PlottingReference.R"
 [7] "R_EDA-Introduction.R"              "R_EDA-Introduction.Rproj"          "R_refcard-data-mining.pdf"
[10] "R_refcard.pdf"                     "README.md"                         "readS3.R"
[13] "scriptTemplate.R"                  "table_S3.csv"                      "taskSolutions.R"
[16] "tmp.R"                             "typeInfo.R"                        "weissgerber_2015_BeyondBarcharts.zip"
> list.files(all.files = TRUE)
 [1] "."                                 ".."                                ".git"
 [4] ".gitignore"                        ".init.R"                           ".Rhistory"
 [7] ".Rprofile"                         ".Rproj.user"                       "functionTemplate.R"
[10] "GvHD.txt"                          "Jaitin_paper.zip"                  "LPSdat.RData"
[13] "myScript.R"                        "PlottingReference.R"               "R_EDA-Introduction.R"
[16] "R_EDA-Introduction.Rproj"          "R_refcard-data-mining.pdf"         "R_refcard.pdf"
[19] "README.md"                         "readS3.R"                          "scriptTemplate.R"
[22] "table_S3.csv"                      "taskSolutions.R"                   "tmp.R"
[25] "typeInfo.R"                        "weissgerber_2015_BeyondBarcharts.zip"
>
```

2.Load data:



	genes	B.ctrl	B.LPS	MF.ctrl	MF.LPS	NK.ctrl	NK.LPS	Mo.ctrl	Mo.LP
1	Cd69	-12.9	-10.5	-13.1	-11.6	-12.9	-9.5	-13.1	
2	Cxcl10	-11.4	-8.5	-10.9	-6.0	-12.0	-6.1	-12.0	
3	Ifi47	-12.1	-8.9	-11.9	-9.8	-11.6	-8.8	-12.1	
4	Ifit2	-12.8	-9.8	-12.8	-8.9	-13.3	-10.5	-12.9	
5	Ifit3	-12.6	-7.9	-12.0	-9.2	-12.9	-9.1	-12.5	
6	Igtp	-12.3	-9.5	-12.1	-9.6	-11.7	-8.9	-12.3	
7	Irf7	-11.9	-9.4	-10.4	-9.1	-12.2	-9.5	-11.5	
8	Irgm1	-12.1	-8.9	-12.8	-9.5	-12.2	-9.5	-12.3	
9	Isg15	-12.0	-9.1	-11.7	-8.3	-12.0	-8.4	-12.0	
10	Isg20	-12.1	-9.7	-12.9	-10.8	-12.9	-9.9	-12.2	
11	Mx1	-12.4	-9.0	-11.7	-9.0	-12.6	-9.8	-12.7	
12	Mx2	-12.9	-10.3	-12.8	-10.3	-13.3	-11.6	-13.0	
13	Oasl1	-12.7	-10.1	-12.3	-9.2	-13.0	-10.4	-12.7	
14	Parp14	-11.9	-9.3	-11.4	-9.0	-11.8	-9.3	-11.9	
15	Rsad2	-13.1	-10.3	-12.5	-9.3	-13.0	-10.5	-12.9	

Showing 1 to 15 of 1,341 entries, 16 total columns

```
Console  Terminal  Jobs
R 4.2.0 · D:/TCD/R_EDA-Introduction/

    ================
        WELCOME !
  Type  'init()'  to begin
    ================

> init()
> load("D:/TCD/R_EDA-Introduction/LPSdat.RData")
> View(LPSdat)
>
```

Environment | History | Connections | Git | Tutorial

Import Dataset · 144 MiB

R · Global Environment

Data
- LPSdat — 1341 obs. of 16 variables

Functions
- init — function ()
- typeInfo — function (x)

Files | Plots | Packages | Help | Viewer

Folder · Blank File · Delete · Rename

D: › TCD › R_EDA-Introduction

Name	Size	Modif
.Rprofile	296 B	May
functionTemplate.R	529 B	May
GvHD.txt	215.4 KB	May
Jaitin_paper.zip	1.5 MB	May
LPSdat.RData	27.6 KB	May
PlottingReference.R	23.1 KB	May
R_EDA-Introduction.R	13.1 KB	May
R_EDA-Introduction.Rproj	257 B	May
R_refcard-data-mining.pdf	163 KB	May
R_refcard.pdf	531.7 KB	May
README.md	111 B	May
readS3.R	3.2 KB	May
scriptTemplate.R	1.4 KB	May
table_S3.csv	117.5 KB	May
taskSolutions.R	1.1 KB	May
tmp.R	611 B	May
typeInfo.R	827 B	May
Weissgerber_2015_BeyondBarcharts.zip	304.4 KB	May
myScript.R	611 B	May

3. Subsetting

```
18  # gene names and the expression values for Mo.LPS
19  # for the top ten expression values.
20  LPSdat[order(LPSdat$Mo.LPS, decreasing = TRUE)[1:10], c("genes", "Mo.LPS")]
21
22  # [END]
23
24
```

24:1 (Untitled) R Script

Console Terminal Jobs

R 4.2.0 · D:/TCD/R_EDA-Introduction/

```
> LPSdat[order(LPSdat$Mo.LPS, decreasing = TRUE)[1:10], c("genes", "Mo.LPS")]
      genes Mo.LPS
205  H2-Eb1   -6.1
2    Cxcl10   -6.4
367  H2-Ab1   -6.5
708    Fth1   -6.6
129    Srgn   -6.8
720    Lyz2   -7.0
1213   Ftl1   -7.4
357    Cst3   -7.5
714  Ifitm3   -7.5
704  Fcer1g   -7.6
>
```

```
21
22  #which of the data columns has the highest standard deviation?
23  head(LPSdat)
24  sd(LPSdat[ , 2])
25  for (i in 2:15) {
26      cat(sprintf("%s %f\n",colnames(LPSdat), sd(LPSdat[ , i])))
27  }
28
29  # [END]
30
31
```

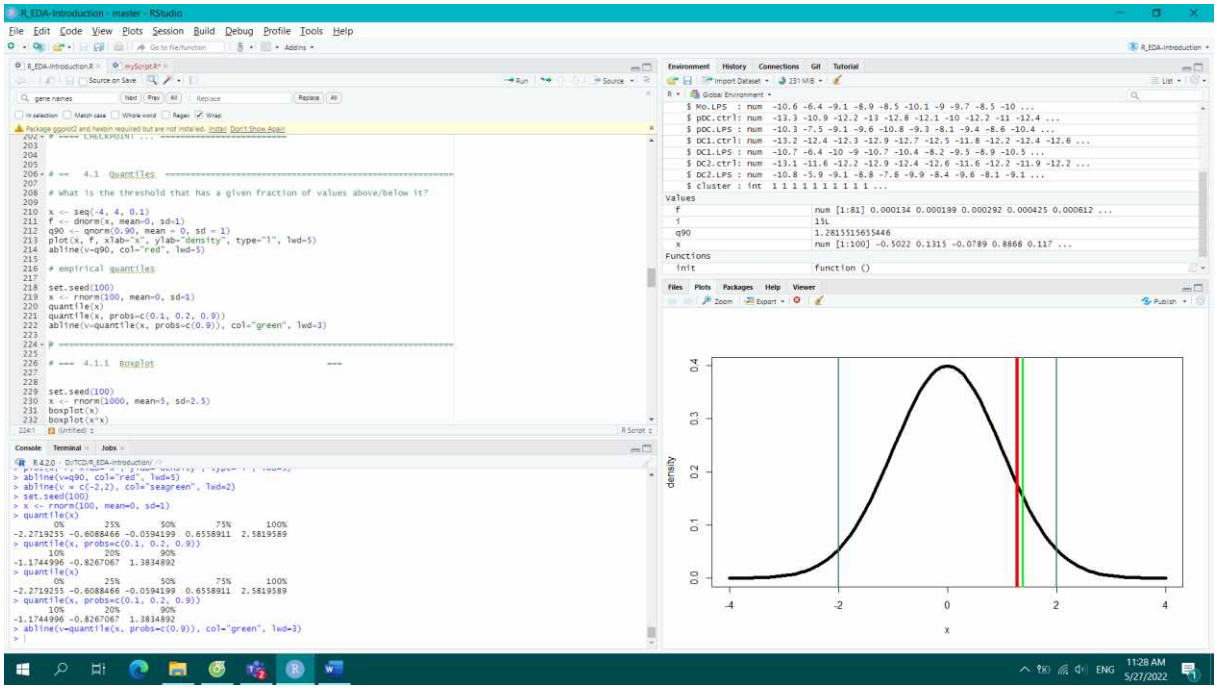
28:1 (Untitled) R Script

Console Terminal Jobs

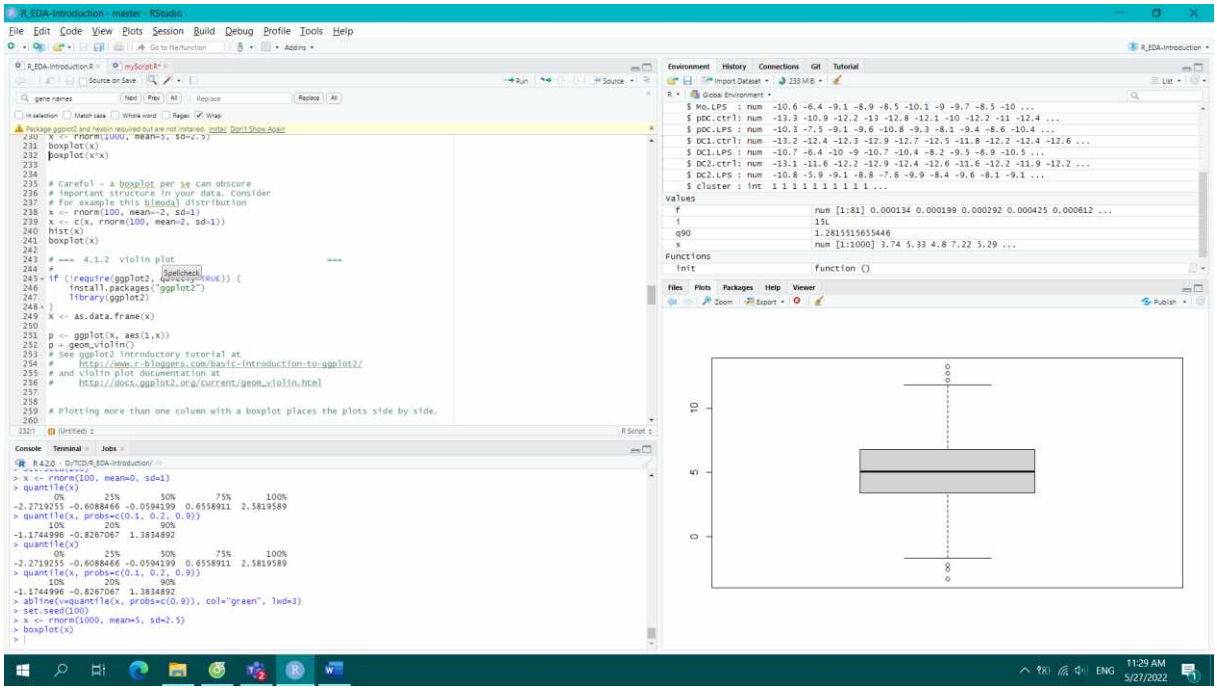
R 4.2.0 · D:/TCD/R_EDA-Introduction/

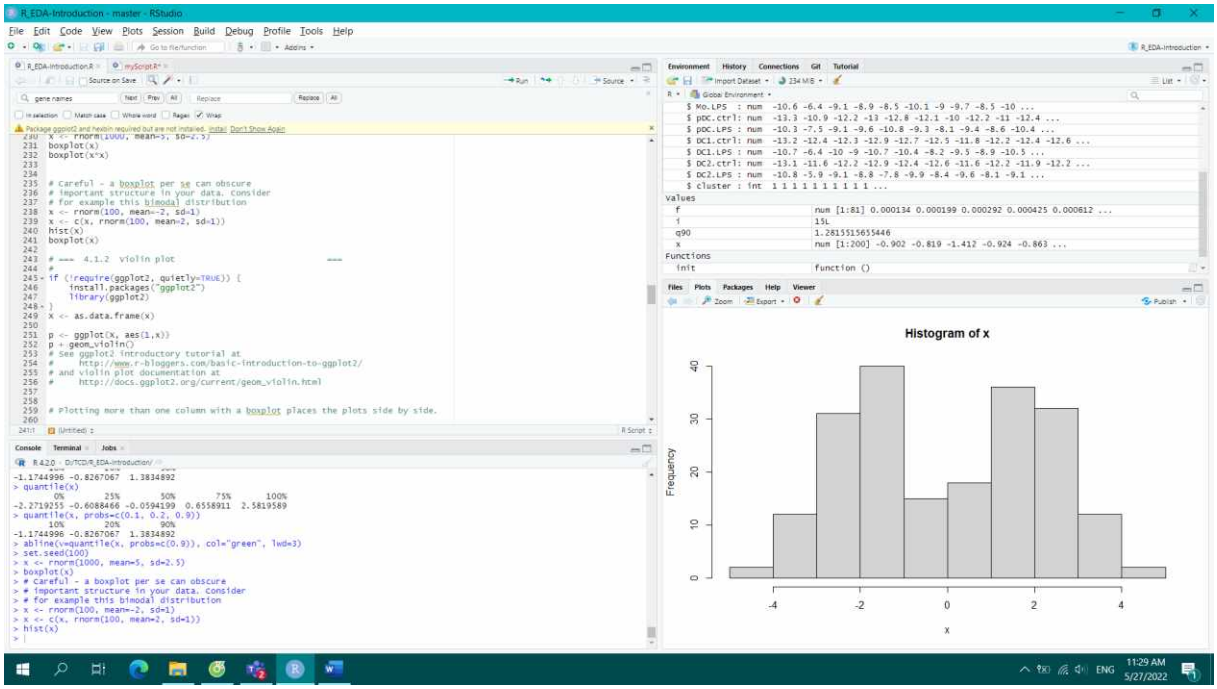
```
+ }
genes 0.946103
 B.ctrl 0.946103
 B.LPS 0.946103
 MF.ctrl 0.946103
 MF.LPS 0.946103
 NK.ctrl 0.946103
 NK.LPS 0.946103
 Mo.ctrl 0.946103
 Mo.LPS 0.946103
 pDC.ctrl 0.946103
 pDC.LPS 0.946103
 DC1.ctrl 0.946103
 DC1.LPS 0.946103
 DC2.ctrl 0.946103
 DC2.LPS 0.946103
 cluster 0.946103
```

4.1 Quantiles

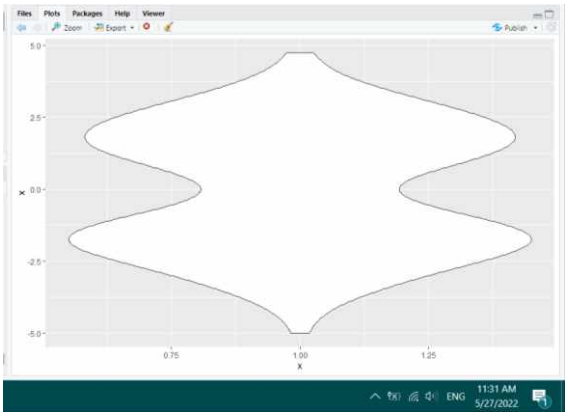


4.2 Box Plot

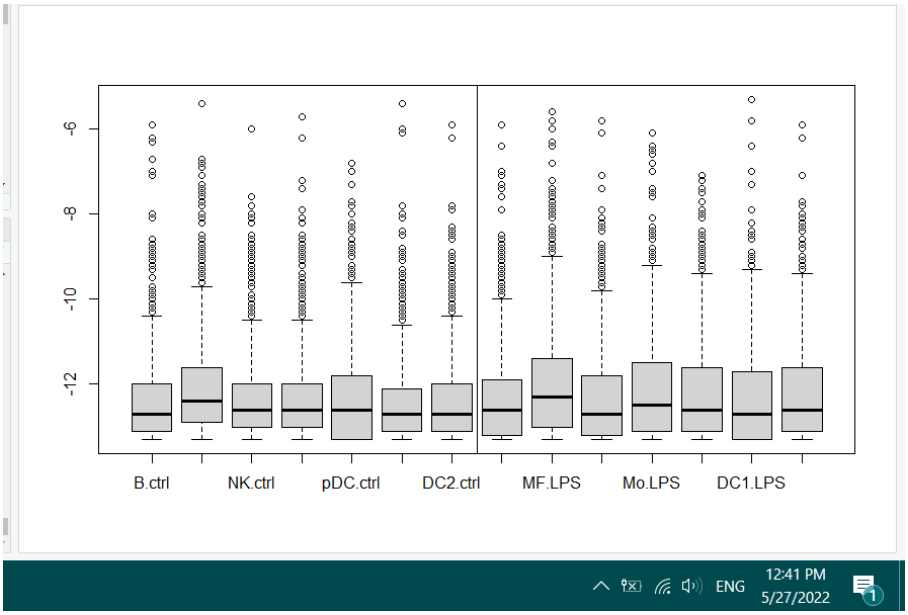




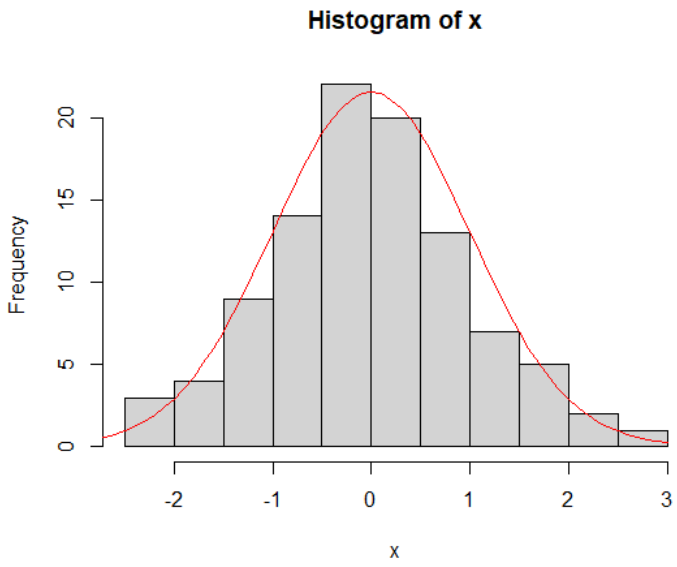
4.3 Violin plot



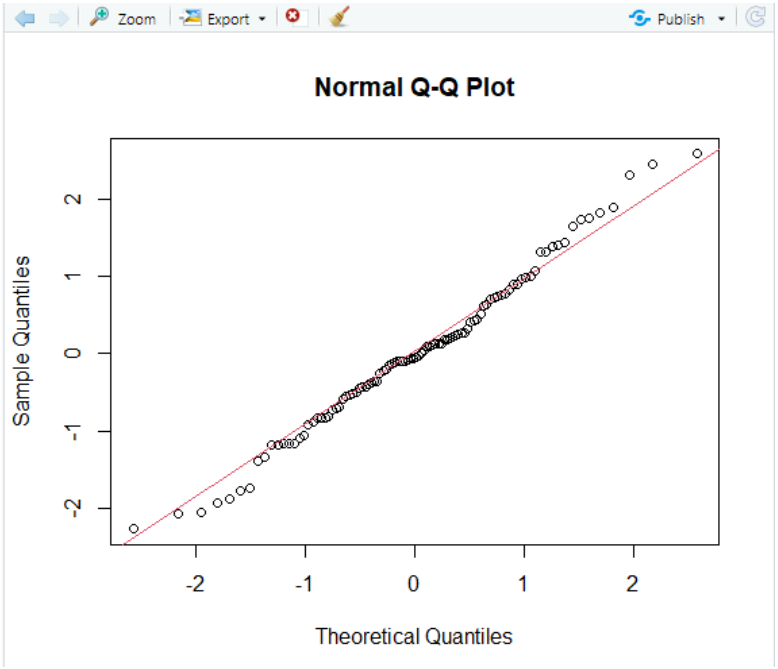
4.4 Plotting more than one column with a boxplot places the plots side by side.



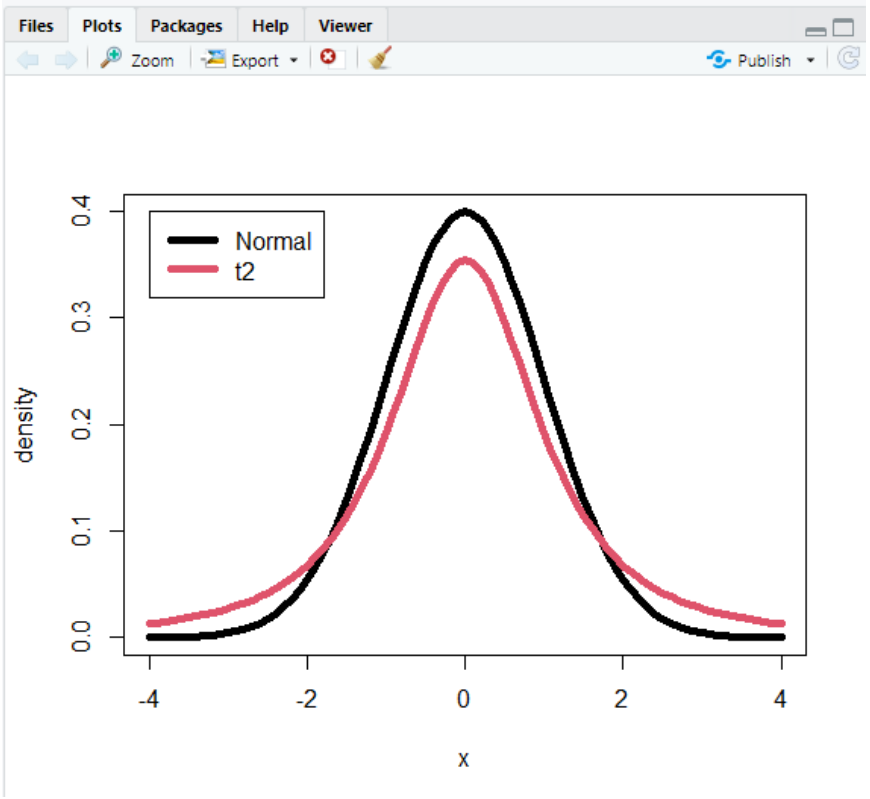
4.5 Overlay a histogram with a line plot



4.6 QQ plots

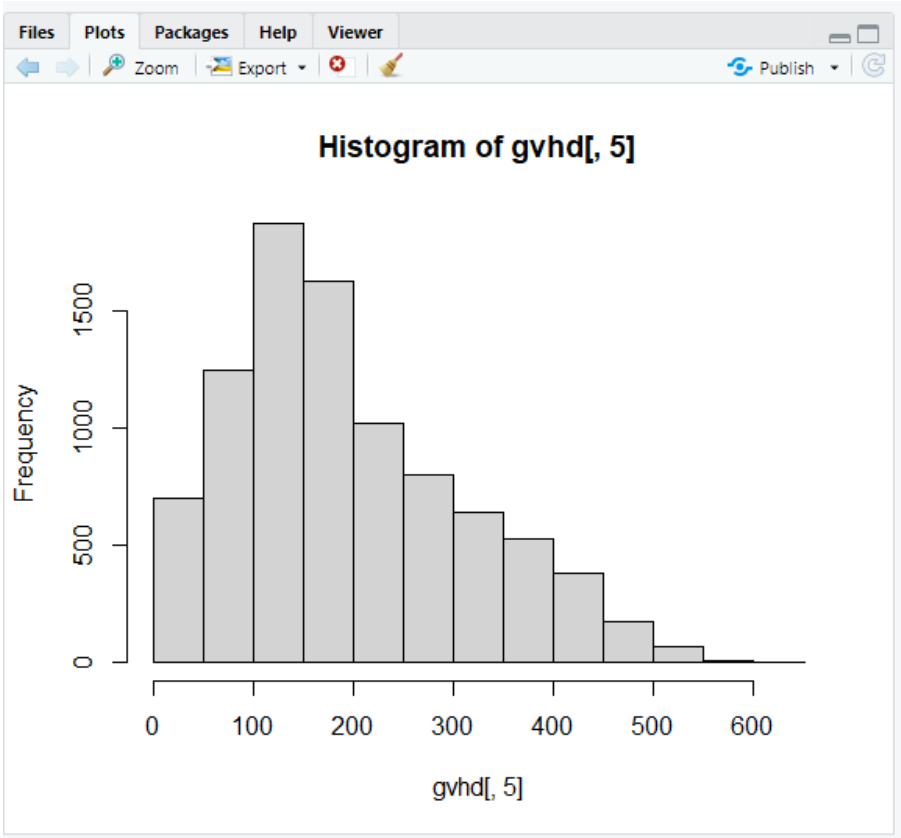


4.7 Lines and legends

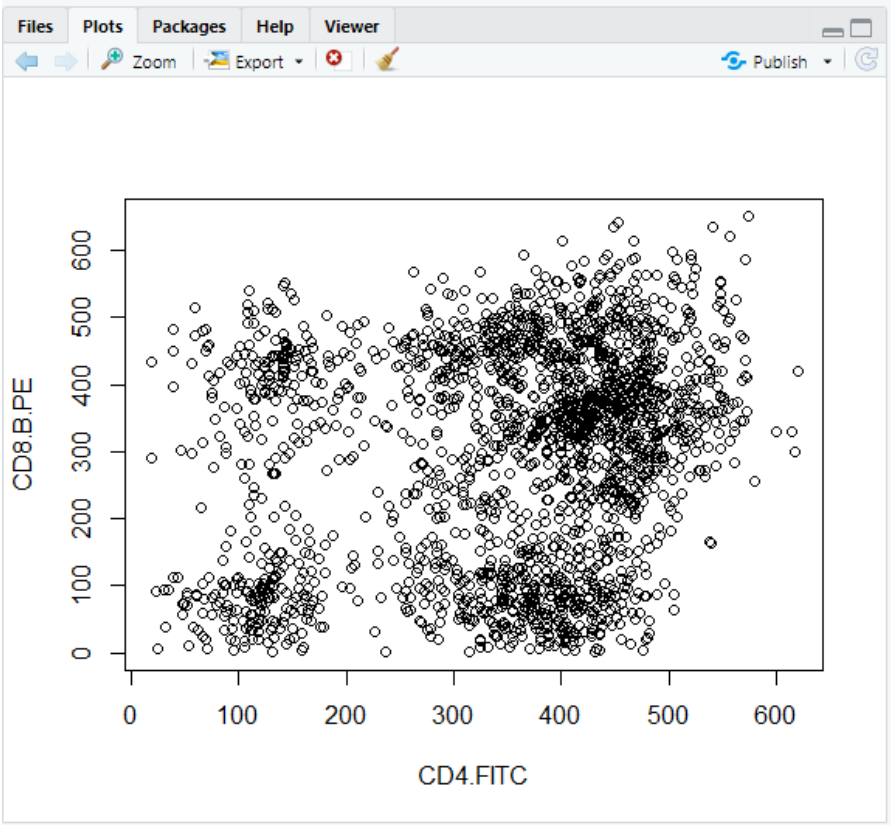


5. Exploring flow cytometry data

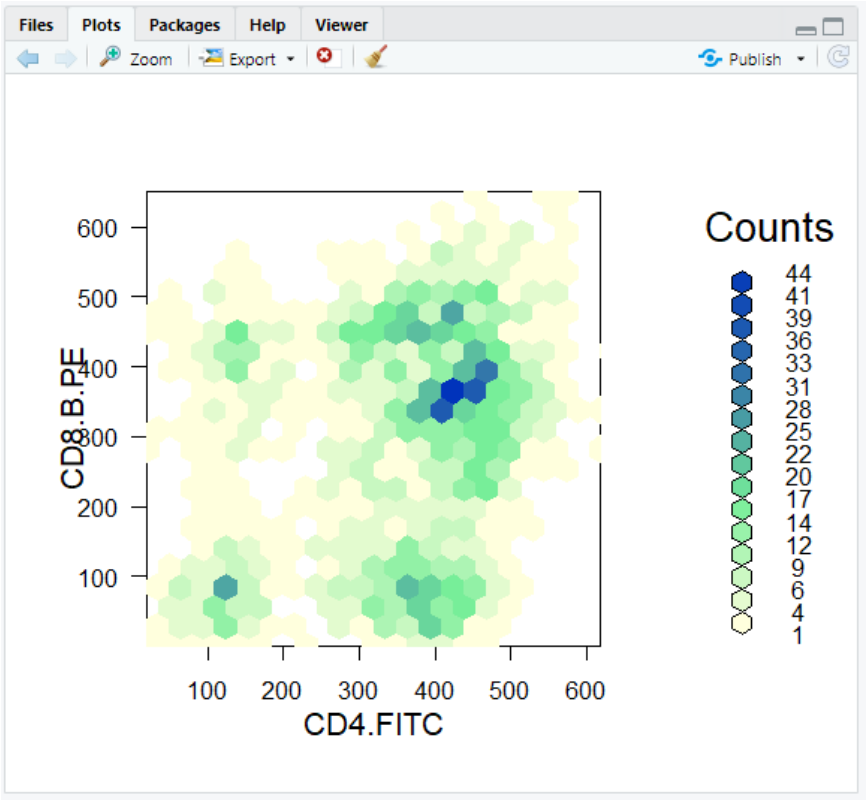
File data: GvHD.txt



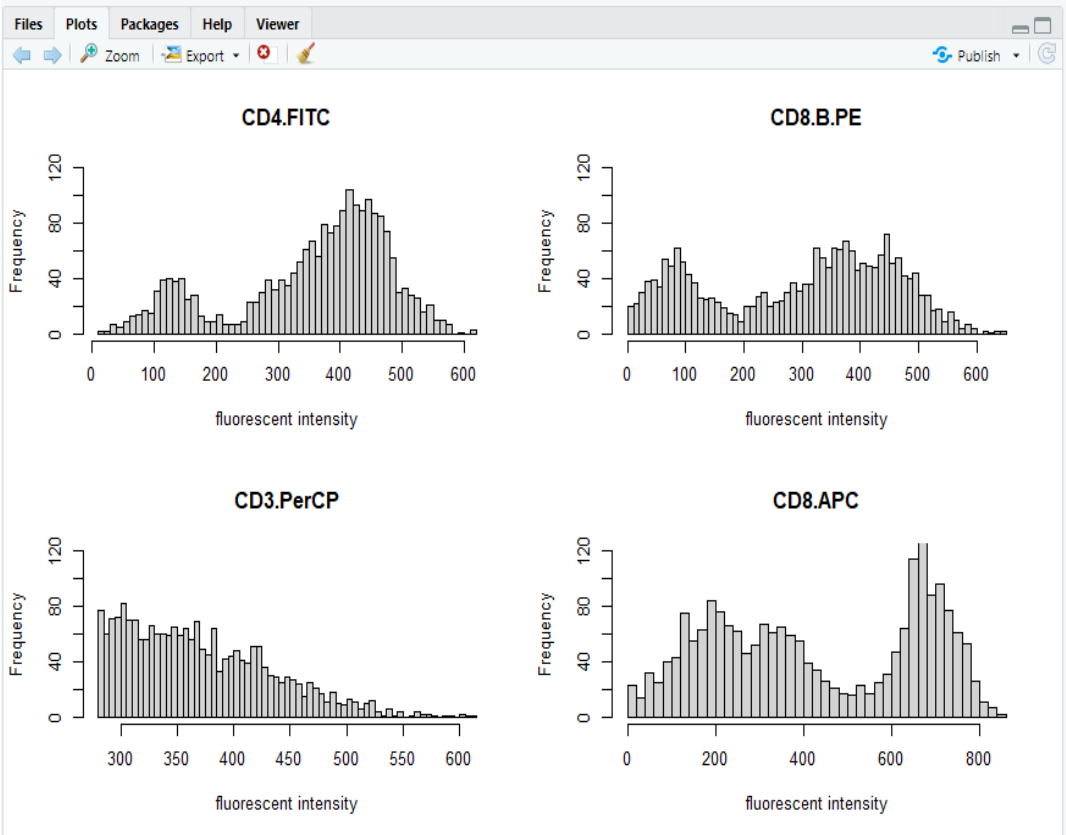
5.1 Biểu diễn các điểm



5.2 Sử dụng hình lục giác và màu sắc



5.3 Trellis plots: all against all

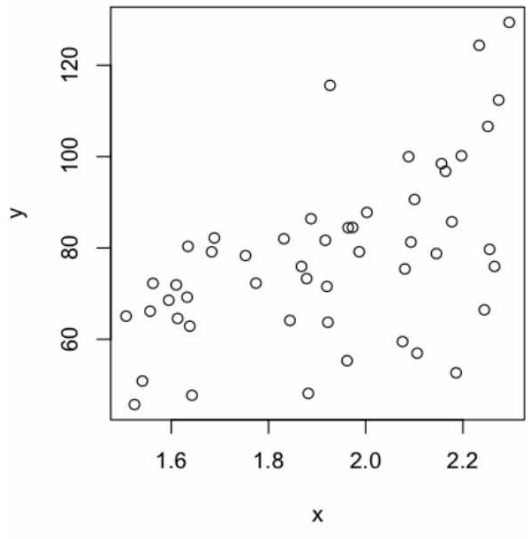


Module 2: Regression

I. Lý thuyết

1. Đặt vấn đề

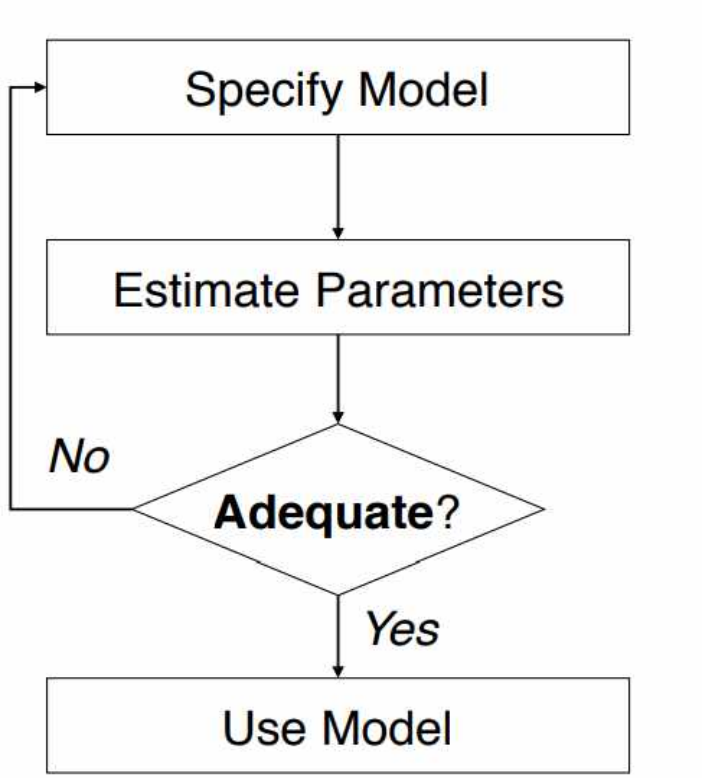
Khi thực hiện đo lường nhiều biết cho mỗi thành viên trong một quần thể, biểu đồ phân tán có thể cho chúng ta thấy rằng giá trị không hoàn toàn độc lập, chẳng hạn có xu hướng biến này phụ thuộc vào biến kia



Ví dụ sự phụ thuộc:

- Chiều cao so với cân nặng
- Xác suất tử vong so với tuổi

2. Mô hình hóa



Hồi quy tuyến tính là một trong những mô hình khả thi mà chúng ta có thể áp dụng để phân tích dữ liệu

3.Tác động của các mối tương quan đối với dữ liệu;

- Khi một biến này phụ thuộc vào biến kia, các biến ở một mức độ nào đó có mối tương quan với nhau.

Hệ số giá trị tương quan của Pearson nằm trong khoảng từ -1 đến 1, với 0 cho thấy không có tương quan.

4. Hồi quy tuyến tính

Hồi quy tuyến tính là một trong những mô hình khả thi mà chúng ta có thể áp dụng để phân tích dữ liệu.

Hồi quy tuyến tính giả định một mô hình cụ thể:

$$y_i = \alpha + \beta x_i + \varepsilon_i$$

xi là biến độc lập. Tùy thuộc vào ngữ cảnh, còn được gọi là "biến dự báo", "biến hồi quy", "biến được kiểm soát", "biến thao tác", "biến giải thích", "biến hiển thị" và / hoặc "biến đầu vào"

yi là biến phụ thuộc, còn được gọi là "biến phản hồi", "hồi quy và", "biến đo lường", "biến quan sát", "biến phản hồi", "biến giải thích", "biến kết quả", "biến thử nghiệm" và / hoặc "biến đầu ra".

εi là "lỗi" - không theo nghĩa là "sai", mà theo nghĩa là tạo ra những sai lệch so với mô hình lý tưởng hóa. Các εi được giả định là độc lập và N (0,) (phân phối chuẩn), chúng cũng có thể được gọi là phần dư

Mô hình này có hai tham số: hệ số hồi quy α và hệ số chặn β

Phân tích hồi quy tuyến tính bao gồm:

- Ước lượng các tham số;

- Đặc điểm mô hình tốt như thế nào.

* Hồi quy tuyến tính: estimation

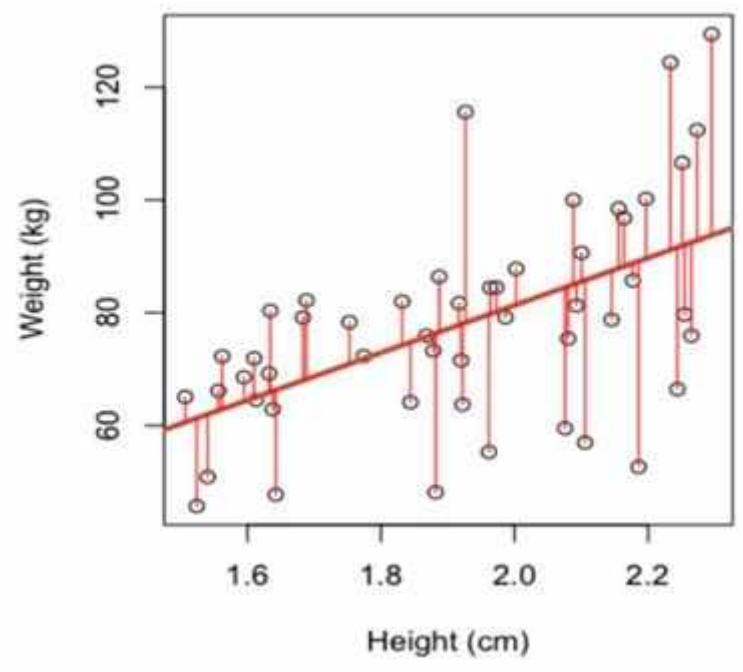
Đối với mô hình tuyến tính, với các tham số ước lượng a, b

$$SSE = \sum_{i=1}^{n} \left(y_i - a - b(x_i)\right)^2$$

Ước lượng: chọn các tham số a, b sao cho SSE càng nhỏ càng tốt. Chúng ta gọi đây là: ước lượng bình phương nhỏ nhất, Phương pháp bình phương nhỏ nhất này có một giải tích cho trường hợp tuyến tính

* Hồi quy tuyến tính: residuals

Dư lượng : Đường liền nét màu đỏ là đường vừa với hình vuông nhỏ nhất (đường hồi quy), được xác định bởi độ dốc và điểm giao cắt cụ thể. Các đường màu đỏ giữa đường hồi quy và các điểm dữ liệu thực tế là phần dư.



* Hồi quy tuyến tính: quality control

- Mối quan hệ giữa phản hồi và bộ hồi quy phải là tuyến tính (ít nhất là gần đúng).

- Thuật ngữ lỗi, ε phải có giá trị trung bình bằng 0

- ε nên có phương sai không đổi

- Các lỗi phải được phân phối bình thường (bắt buộc đối với các bài kiểm tra và khoảng thời gian)

5. Hồi quy phi tuyến tính

nls() có nhiều chức năng giải nén giống như lm()

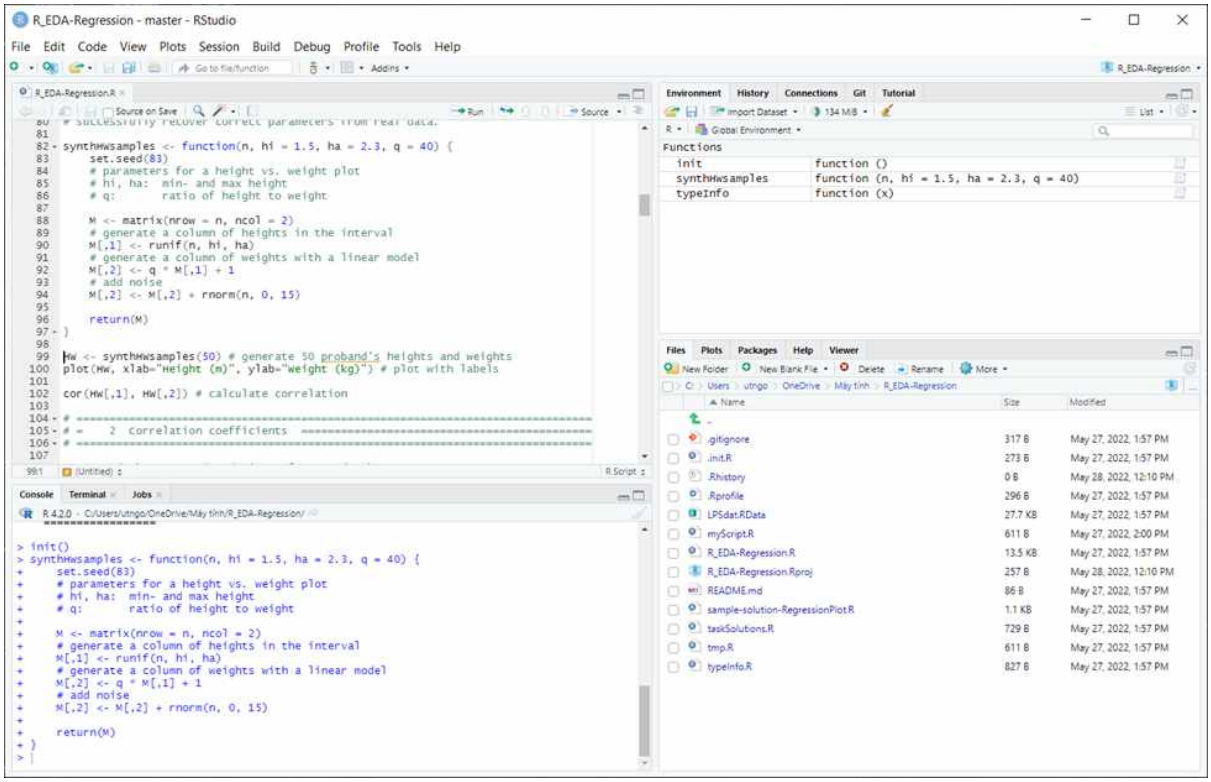
res = nls(formula, data=data, start=c(parameters))

Các tùy chọn khác nhau tồn tại liên quan đến thuật toán ước tính các tham số. Ước lượng mạnh mẽ với nls() đặc biệt hữu ích

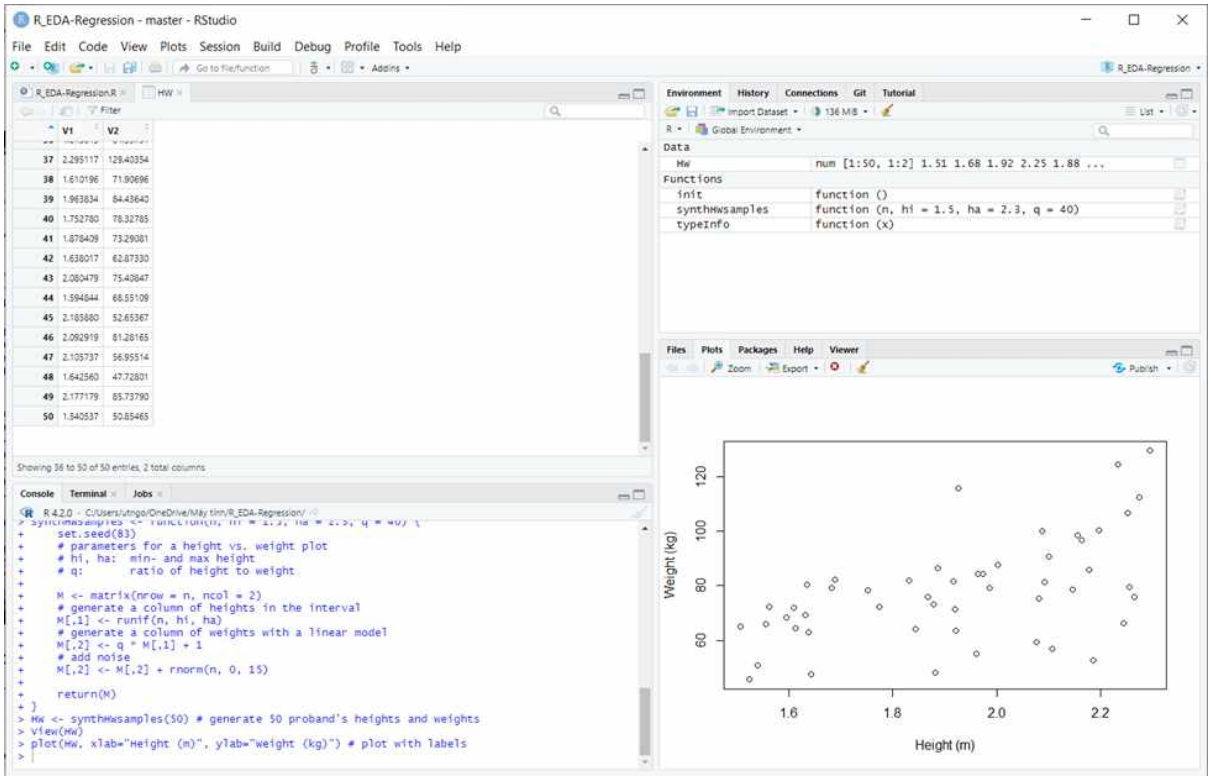
II.Thực hành

1. Synthetic data: a linear model

Tạo độ cao ngẫu nhiên trong 1 khoảng thời gian, sau đó tính trọng lượng giả định theo phương trình tuyến tính đơn giản



Tạo ra 50 giá trị chiều cao cân nặng và vẽ biểu đồ:



Tính giá trị tương quan giữa chiều cao và cân nặng

```
+     # parameters for a height vs. weight plot
+     # hi, ha:  min- and max height
+     # q:       ratio of height to weight
+
+     M <- matrix(nrow = n, ncol = 2)
+     # generate a column of heights in the interval
+     M[,1] <- runif(n, hi, ha)
+     # generate a column of weights with a linear model
+     M[,2] <- q * M[,1] + 1
+     # add noise
+     M[,2] <- M[,2] + rnorm(n, 0, 15)
+
+     return(M)
+ }
> HW <- synthHWsamples(50) # generate 50 proband's heights and weights
> View(HW)
> plot(HW, xlab="Height (m)", ylab="Weight (kg)") # plot with labels
> cor(HW[,1], HW[,2]) # calculate correlation
[1] 0.5408063
>
```

Cần xác định hệ quy chiếu để biết giá trị tương quan này là thấp hay cao,…

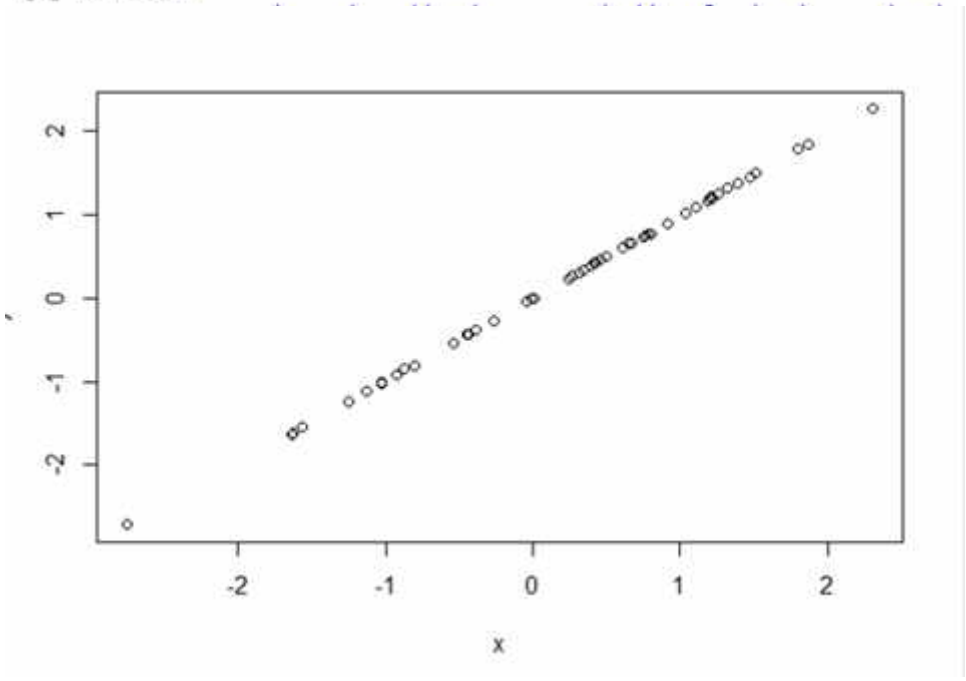
2. Correlation coefficients

Tạo 50 giá trị ngẫu nhiên theo phân phối chuẩn

Xác định các giá trị của Y tương ứng với các giá trị của X,với r là phần tram dữ liệu tương quan hoàn hảo với nhau:

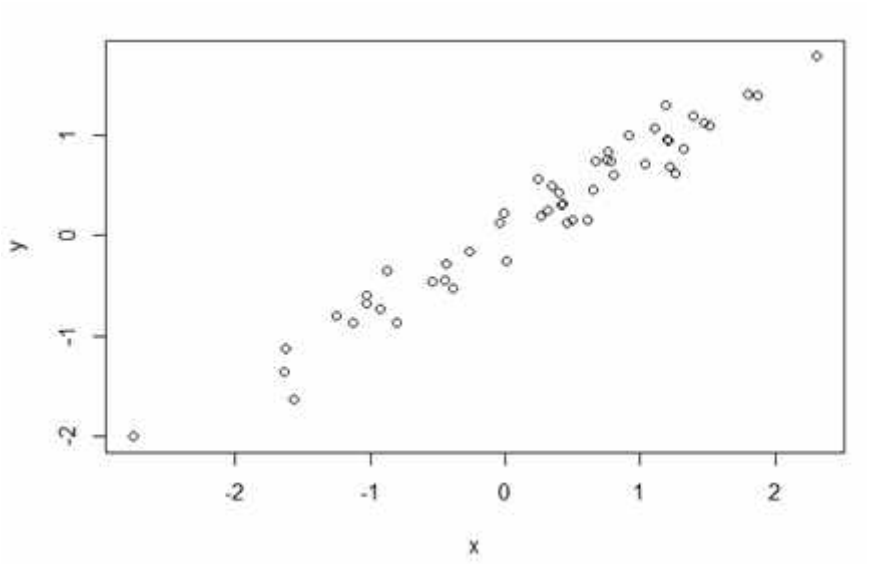
-với r = 0.99

```
> r <- 0.99; y <- (r * x) + ((1-r) * rnorm(50)); plot(x,y); cor(x,y)
[1] 0.9999577
```



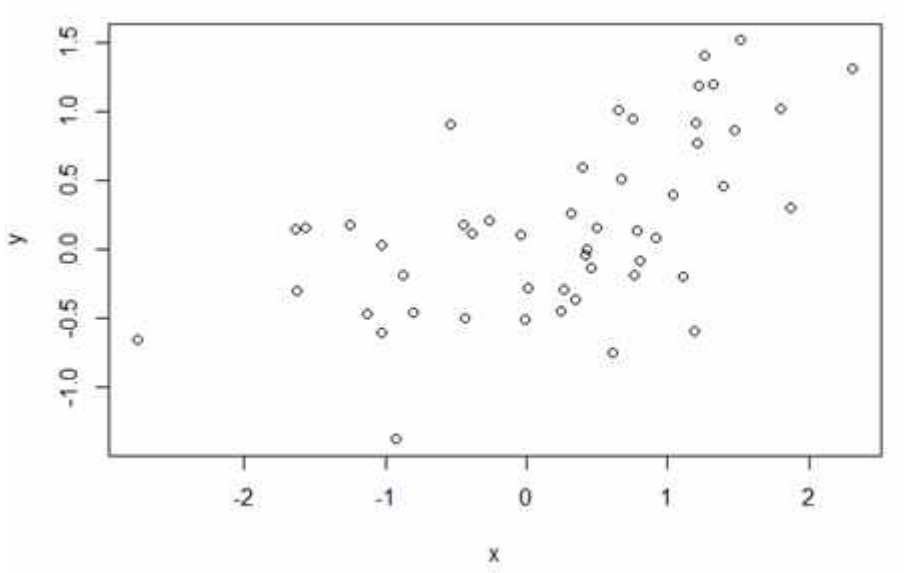
- với r = 0.8

```
> r <- 0.8;  y <- (r * x) + ((1-r) * rnorm(50)); plot(x,y); cor(x,y)
[1] 0.9741214
```



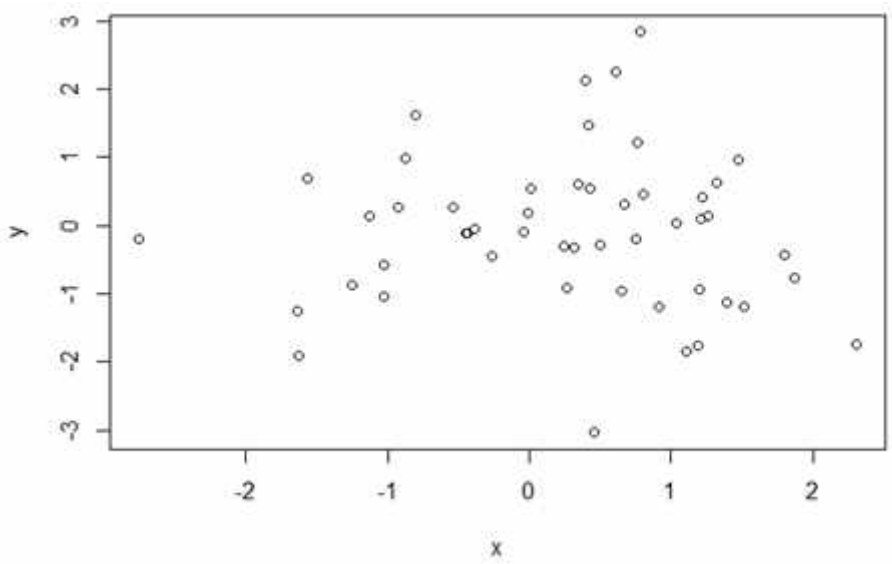
-với r = 0.4

```
> r <- 0.4;  y <- (r * x) + ((1-r) * rnorm(50)); plot(x,y); cor(x,y)
[1] 0.5711197
```

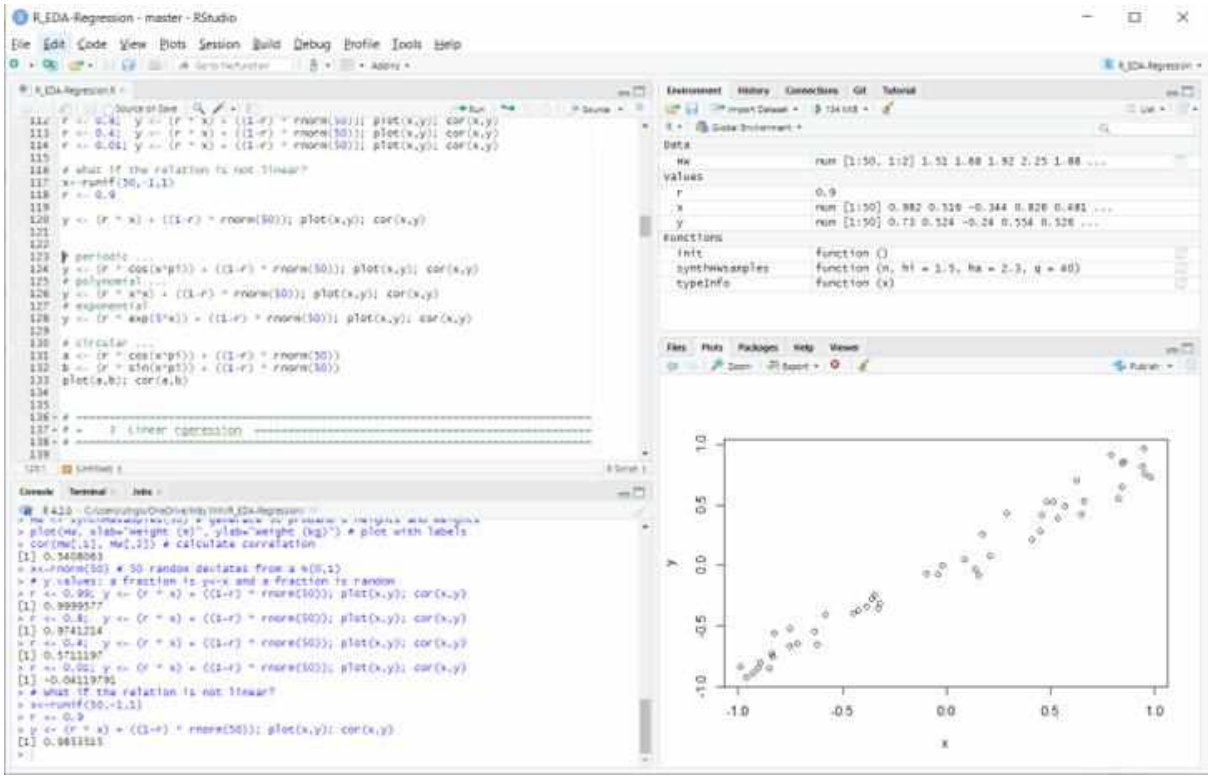


-với r = 0.01

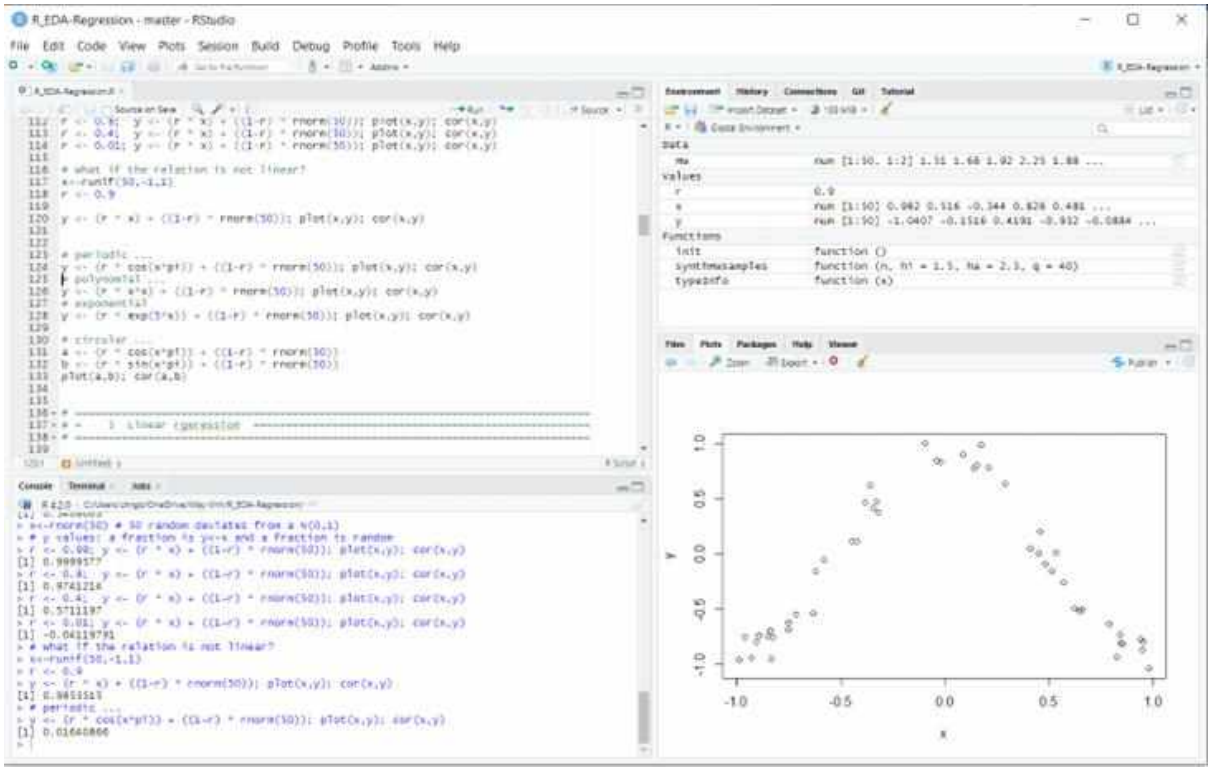
```
> r <- 0.01; y <- (r * x) + ((1-r) * rnorm(50)); plot(x,y); cor(x,y)
[1] -0.04119791
```



Nếu là mối quan hệ phi tuyến tính:tạo 50 giá trị phân phối đồng nhất trong (-1,1)

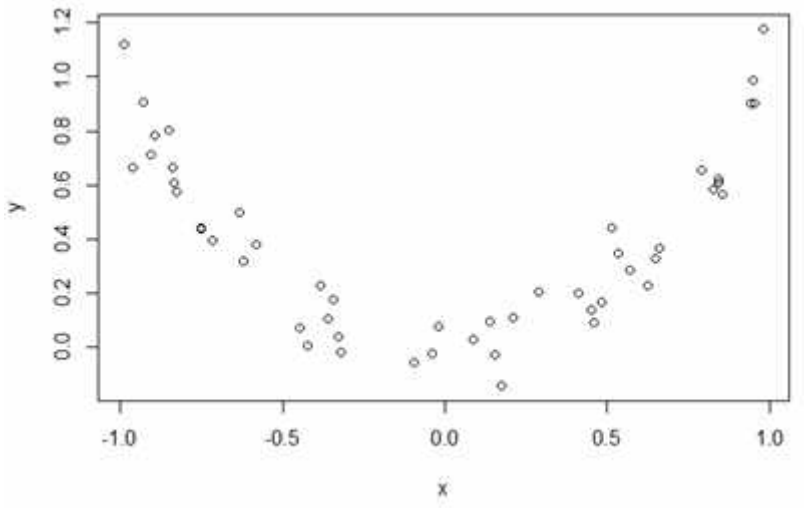


-Xem xét mối tương quan của chuỗi thời gian trong chu kì tế bào:



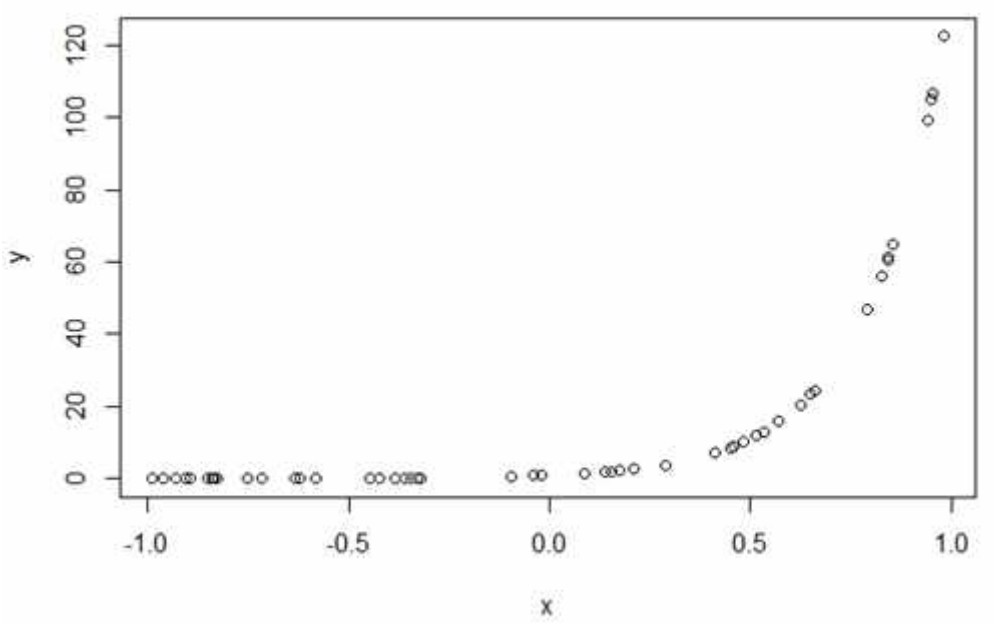
- Polynomial

```
> # polynomial ...
> y <- (r * x*x) + ((1-r) * rnorm(50)); plot(x,y); cor(x,y)
[1] -0.005038197
```



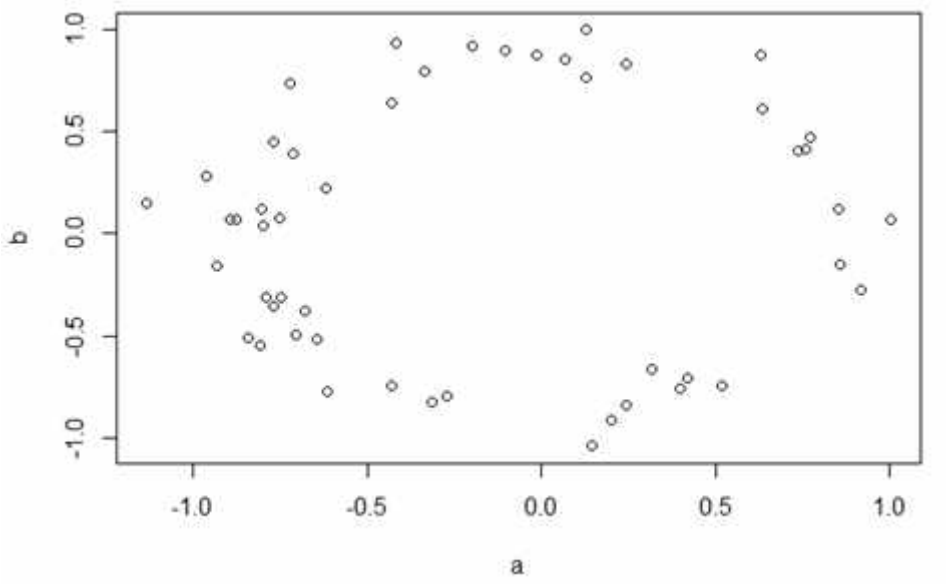
- Exponential

```
> # exponential
> y <- (r * exp(5*x)) + ((1-r) * rnorm(50)); plot(x,y); cor(x,y)
[1] 0.7193067
```



- Circular

```
> # circular ...
> a <- (r * cos(x*pi)) + ((1-r) * rnorm(50))
> b <- (r * sin(x*pi)) + ((1-r) * rnorm(50))
> plot(a,b); cor(a,b)
[1] 0.07419048
```



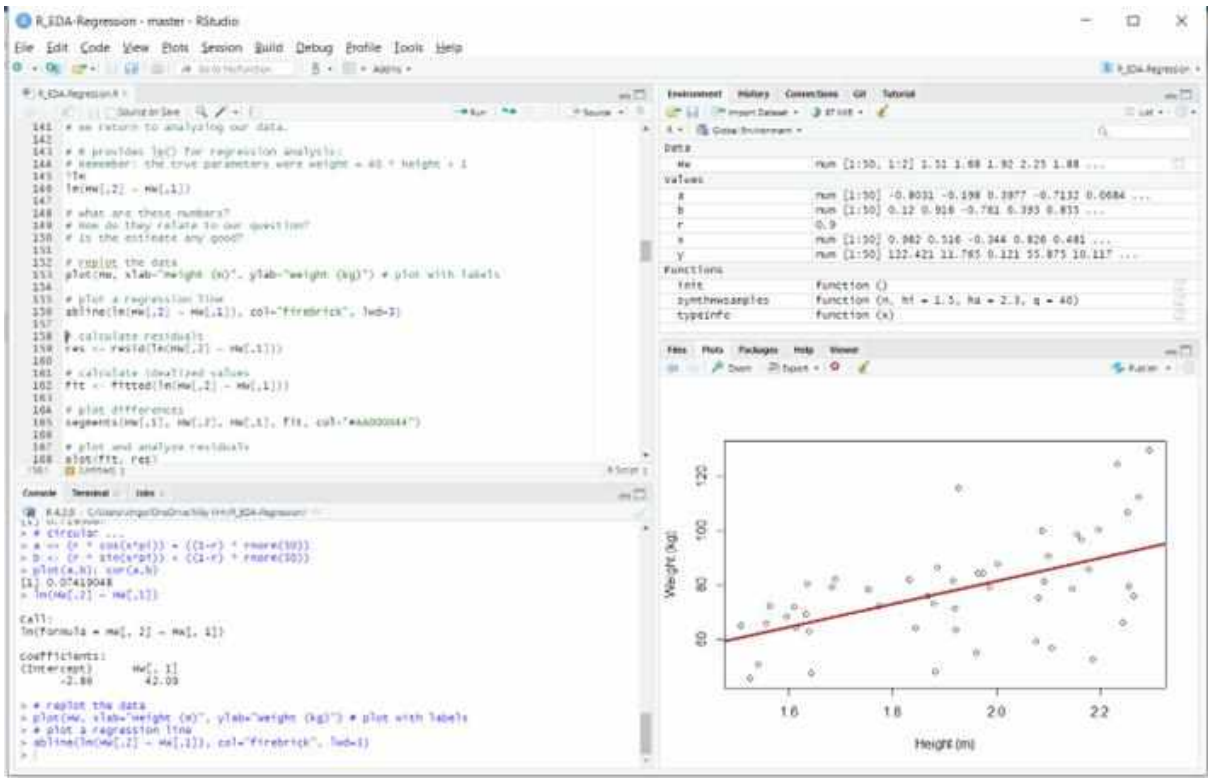
Nhận xét: hệ số tương quan chọn ra những thứ tương quan tuyến tính khá tốt, nhưng đối với các mô hình phi tuyến tính thì hệ số tương quan trở nên vô nghĩa.

3. Linear regression

- hàm lm() để phân tích hồi quy tuyến tính của cột chiều cao cân nặng 2 so với cột chiều cao cân nặng 1

```
> lm(HW[,2] ~ HW[,1])

Call:
lm(formula = HW[, 2] ~ HW[, 1])

Coefficients:
(Intercept)       HW[, 1]
      -2.86         42.09
```

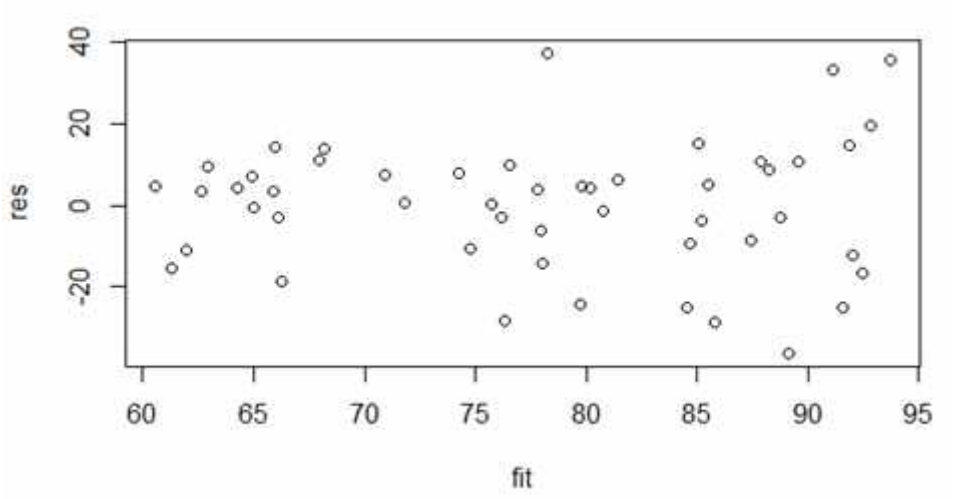
Vẽ đường hồi quy lấy các tham số từ mô hình hồi quy:



tính toán phần dư và giá trị lý tưởng

```
> res <- resid(lm(HW[,2] ~ HW[,1]))
> # calculate idealized values
> fit <- fitted(lm(HW[,2] ~ HW[,1]))
>
```

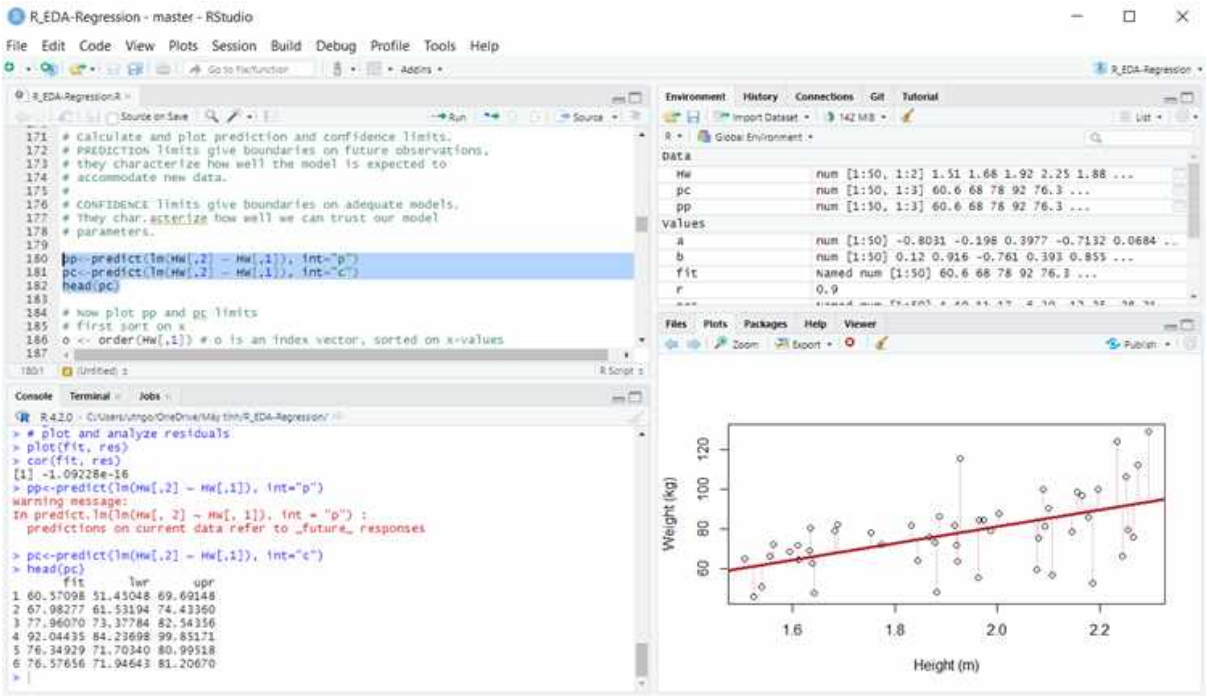
Đồ thị phần dư:



Độ tương quan giữa phần dư và giá trị lý tưởng gần như bằng không.

```
> cor(fit, res)
[1] -1.09228e-16
```

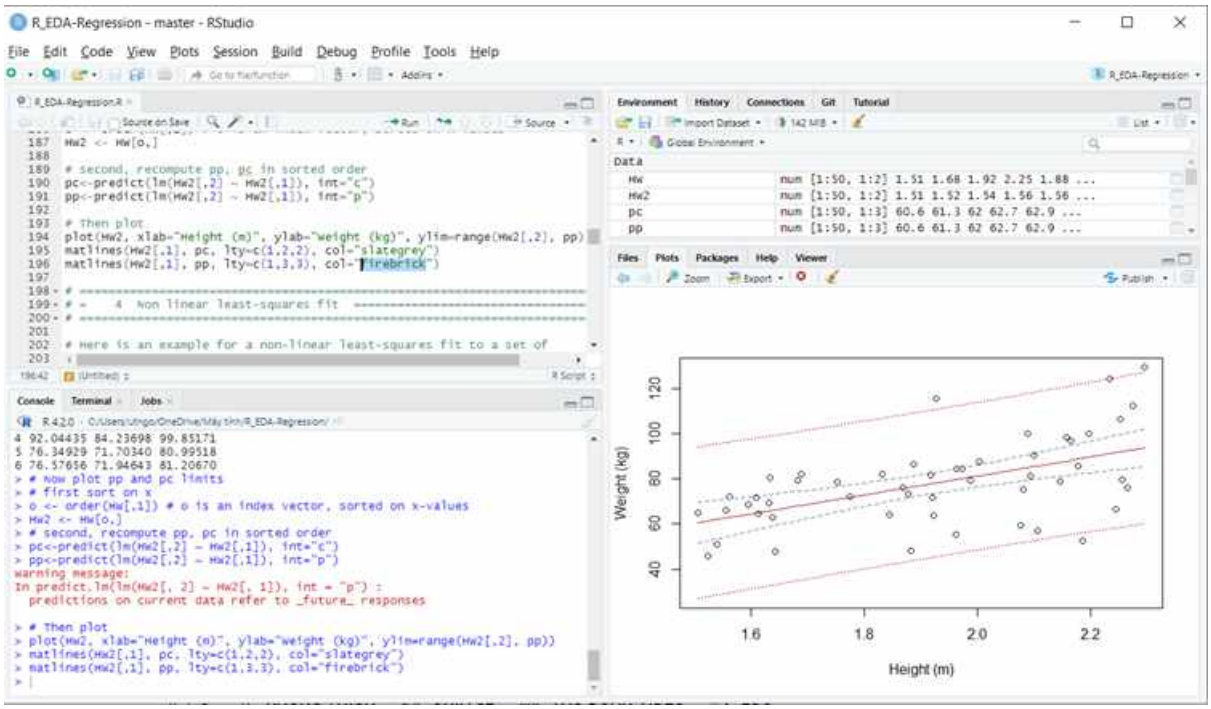
-Tính toán và lập biểu đồ giới hạn dự đoán và giới hạn tin cậy.



Các giá trị đang không được sắp xếp theo thứ tự nhất định nên cần sắp xếp lại theo giá trị tăng dần hoặc giảm dần theo trục x

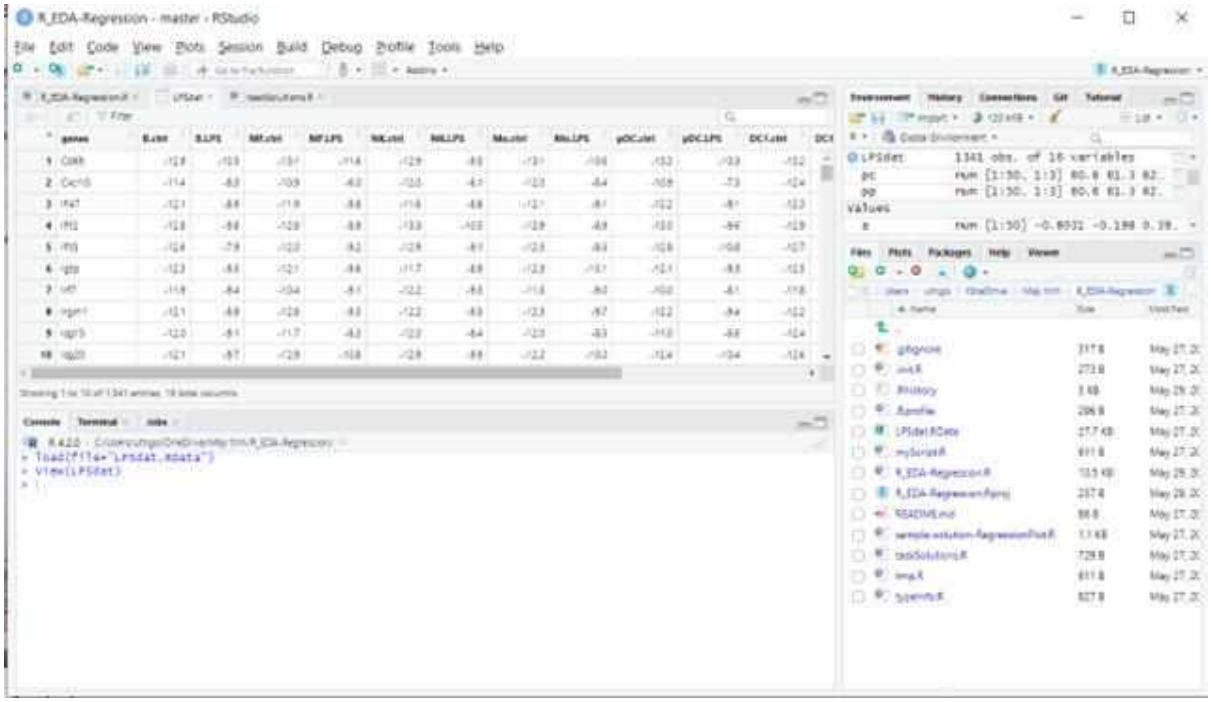
Sau đó tính toán lại các giá trị pp và pc ở trên sau khi đã sắp xếp

```
Console  Terminal  Jobs
R 4.2.0 · C:/Users/utngo/OneDrive/Máy tính/R_EDA-Regression/
       fit      lwr      upr
1 60.57098 51.45048 69.69148
2 67.98277 61.53194 74.43360
3 77.96070 73.37784 82.54356
4 92.04435 84.23698 99.85171
5 76.34929 71.70340 80.99518
6 76.57656 71.94643 81.20670
> # Now plot pp and pc limits
> # first sort on x
> o <- order(HW[,1]) # o is an index vector, sorted on x-values
> HW2 <- HW[o,]
> # second, recompute pp, pc in sorted order
> pc<-predict(lm(HW2[,2] ~ HW2[,1]), int="c")
> pp<-predict(lm(HW2[,2] ~ HW2[,1]), int="p")
Warning message:
In predict.lm(lm(HW2[, 2] ~ HW2[, 1]), int = "p") :
  predictions on current data refer to _future_ responses

>
```



4. Non linear least-squares fit

Tải dữ liệu

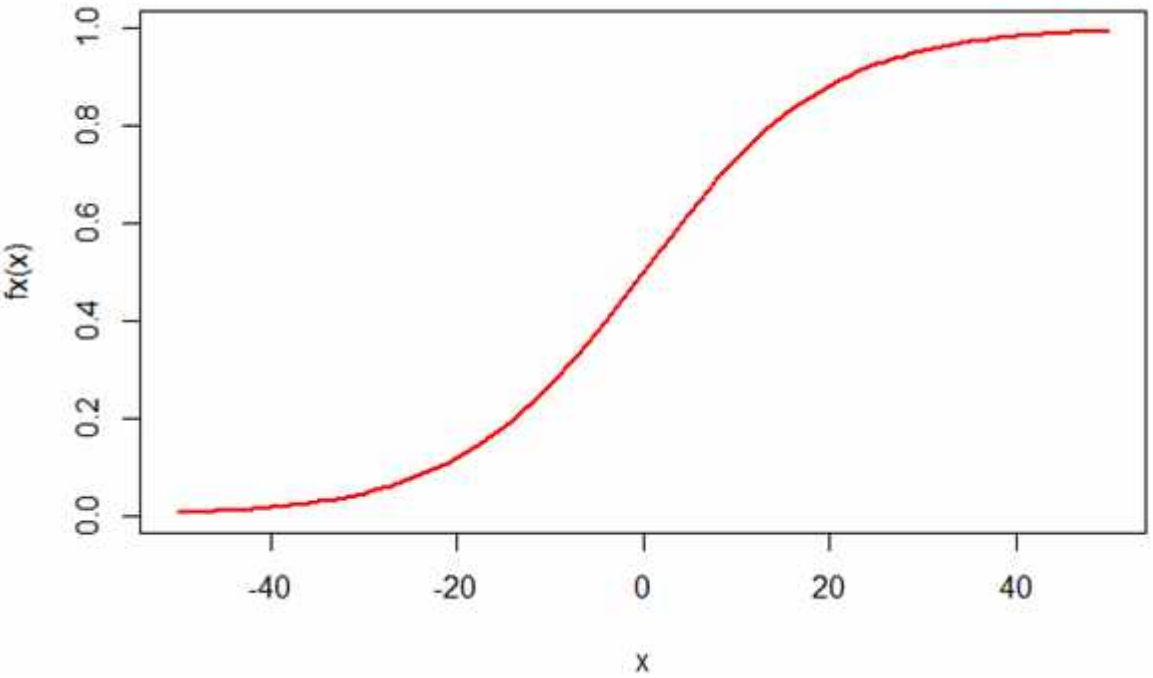


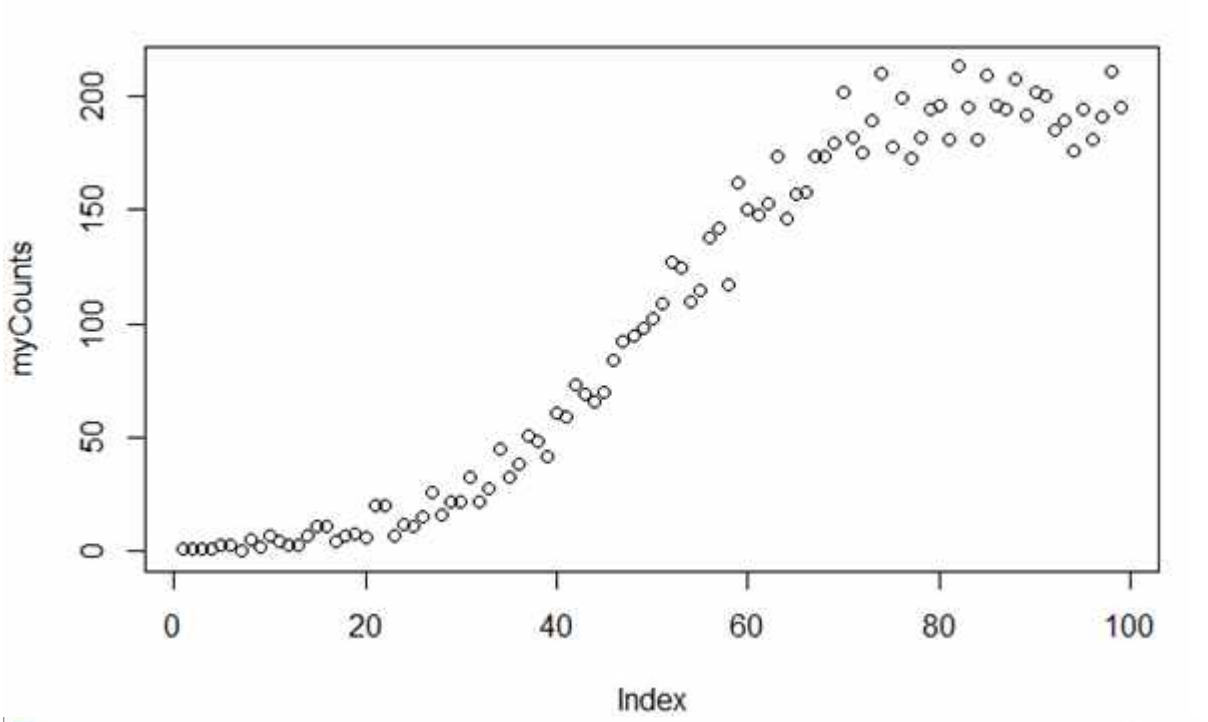
Example: mô phỏng nguy cơ mắc bệnh ở một độ tuổi nhất định theo a logistic function centred 50 năm và lây lan khi khởi phát khoảng từ 0 đến 100 năm.

4.1 Synthetic data: a logistic function

Sử dụng một chiến lược có thể được sử dụng để tạo mẫu cho bất kỳ phân phối mục tiêu tùy ý nào. Để đạt được điều này, ta tạo ra một phương sai x ngẫu nhiên, phân bố đồng đều trong một khoảng thời gian mà ta quan tâm. Với mỗi phương án, ta tính giá trị hàm f (x) tương ứng. Sau đó, chúng ta "tung xúc xắc" chấp nhận hay từ chối phương án đầu tiên: Chúng ta tạo ra phương án z thống nhất thứ hai trong phạm vi của hàm mục tiêu trong khoảng thời gian của chúng ta. Nếu f (x) nhỏ hơn z, chúng ta chấp nhận x là một mẫu trong phân phối của chúng ta

```
ageOfOnset <- function (n) {
    X <- c()
    i <- 1
    while (i < n) {
        age<-floor(runif(1,1,100))
        s<-runif(1)   # sample uniform between 0 and 1
        # i.e. in the range of the function
        if (s < 1-(1/(1+exp(0.1*(age-50))))) {
            # if s is smaller than the function value ...
            X <- append(X, age) # ... add this age to the vector
            i<-i+1 # ... and increment i.
        }
        # Else, try again until n successful attempts.
    }
    return(X)
}
> fx <- function(x) {1-(1/(1+exp(0.1*(x))))}
> curve(fx, xlim=c(-50,50), col="red", lwd="2")
> ages <- ageOfOnset(10000)
> head(ages, 20)
 [1] 31 90 56 83 47 69 47 99 95 56 64 70 37 76 88 76 59 78 56 55
> myCounts <- tabulate(ages)
> head(myCounts)
[1] 1 1 1 1 3 3
> plot(myCounts)
```





R_EDA-Regression - master - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

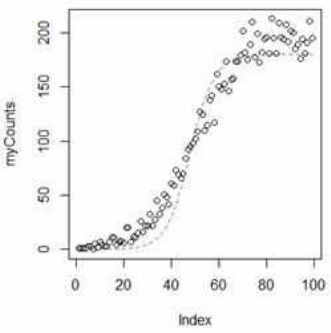
```
#   tm is the mean age of onset
#   B is the risk

# Try some "reasonable" starting parameters, otherwise the
# curve-fit might not converge.

curve(fz(x, S=180, tm=48, B=0.2), add=TRUE, col="slategrey", lwd=1, lty=2)

# now use nls() to fit the formula against our data

age <- c(1:99)
nlsFz <- nls(myCounts ~ fz(age, S, med, B), start=c(S=120, med=30, B=0.9))

nlsFz

summary(nlsFz)
coef(nlsFz)

p <- coef(nlsFz)
```

```
> # Non-linear least-squares fit of a self-defined function!
> fz <- function(t, S, tm, B) { S*(1-(1/(1+exp(B*(t-tm))))) }
> curve(fz(x, S=180, tm=48, B=0.2), add=TRUE, col="slategrey", lwd=1, lty=2)
> age <- c(1:99)
> nlsFz <- nls(myCounts ~ fz(age, S, med, B), start=c(S=120, med=30, B=0.9))
>
> nlsFz
Nonlinear regression model
  model: myCounts ~ fz(age, S, med, B)
   data: parent.frame()
      S     med       B
198.654  49.220   0.103
 residual sum-of-squares: 7413

Number of iterations to convergence: 9
Achieved convergence tolerance: 4.066e-06
>
```

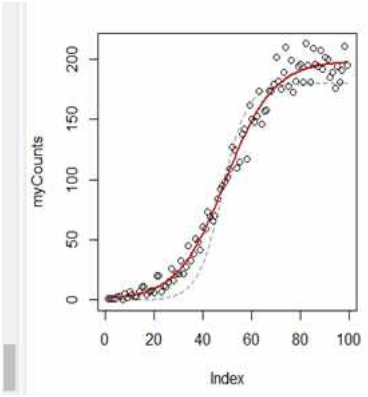
Environment | History | Connections | Git | Tutorial

nlsFz	List of 6
pc	num [1:50, 1:3] 60.6 61.3 62..
pp	num [1:50, 1:3] 60.6 61.3 62..
values	
a	num [1:50] -0.8031 -0.198 0.39..

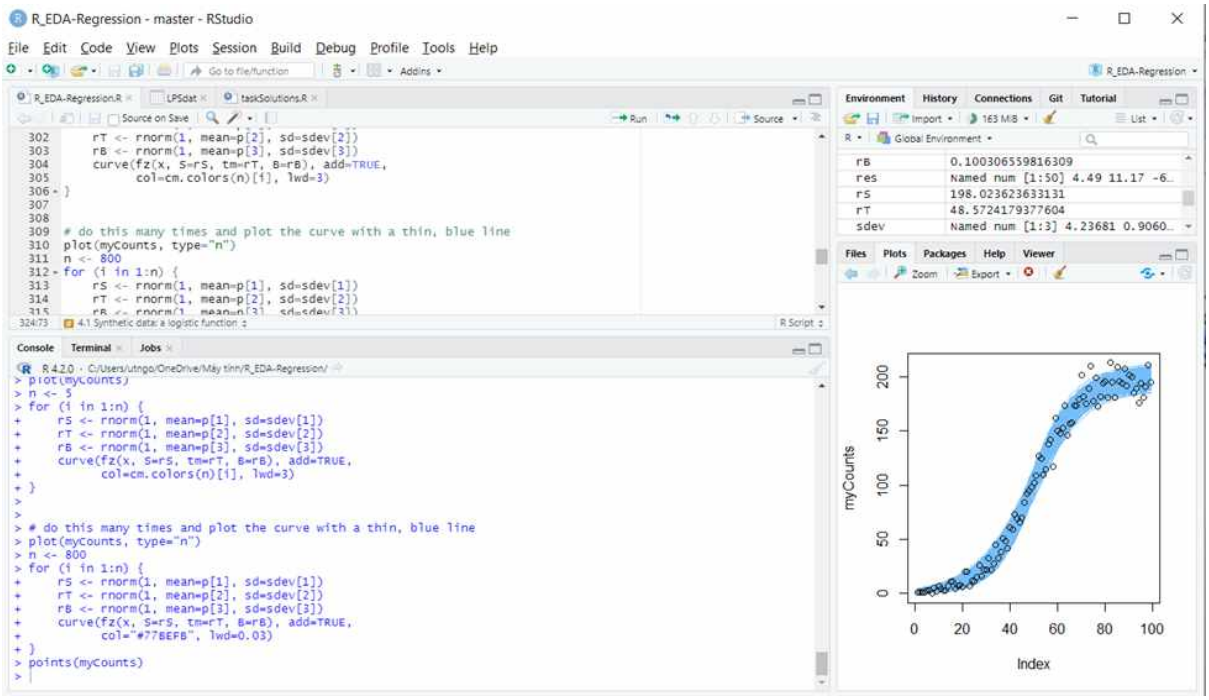
Files | Plots | Packages | Help | Viewer



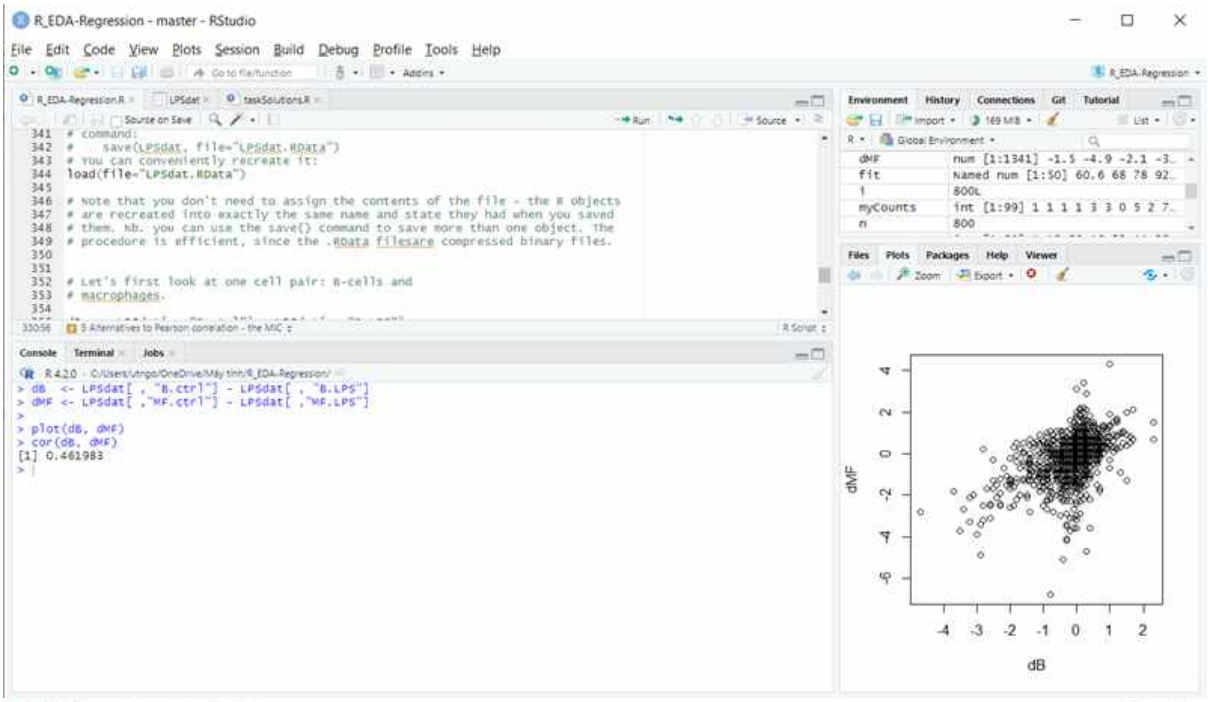
```
> summary(nlsFz)

Formula: myCounts ~ fz(age, S, med, B)

Parameters:
     Estimate Std. Error t value Pr(>|t|)
S   1.987e+02  2.156e+00   92.13   <2e-16 ***
med 4.922e+01  4.600e-01  107.00   <2e-16 ***
B   1.030e-01  3.993e-03   25.81   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.787 on 96 degrees of freedom

Number of iterations to convergence: 9
Achieved convergence tolerance: 4.066e-06

> coef(nlsFz)
        S       med         B
198.654484  49.220216   0.103039
>
> p <- coef(nlsFz)
```

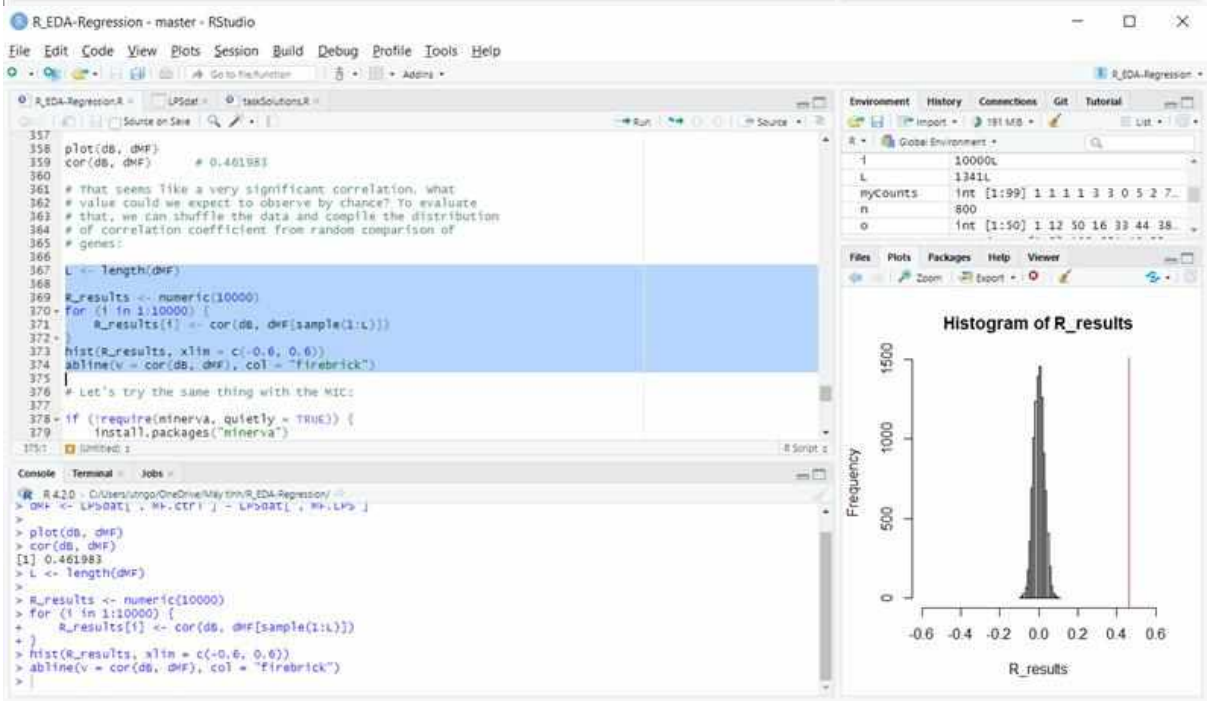


4.2 Evaluating the fit



5. Alternatives to Pearson correlation - the MIC





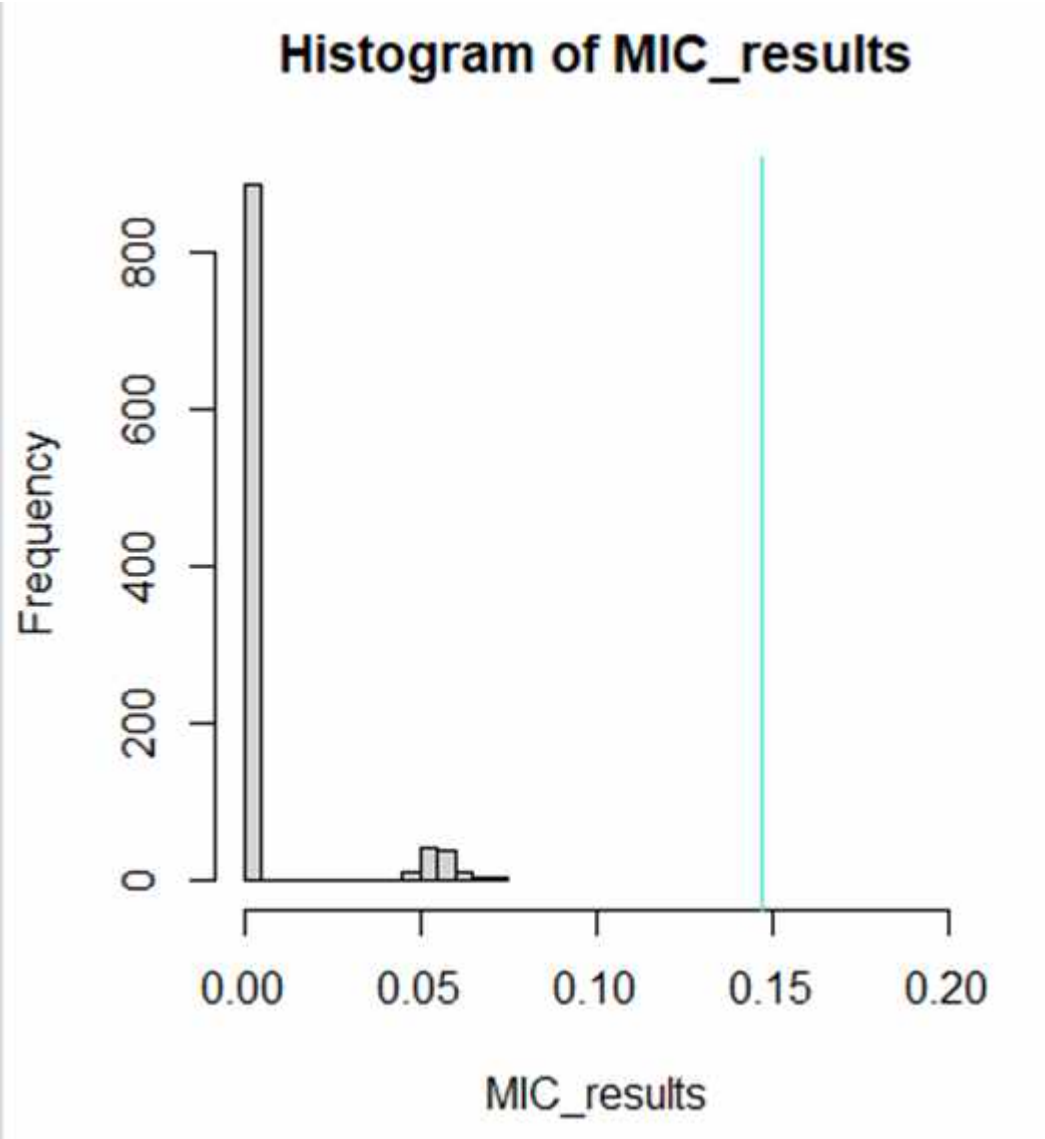
```r
hist(R_results, xlim = c(-0.6, 0.6))
abline(v = cor(dB, dMF), col = "firebrick")

# Let's try the same thing with the MIC:

if (!require(minerva, quietly = TRUE)) {
    install.packages("minerva")
    library(minerva)
}
```

387:1 (Untitled)

Console Terminal Jobs

R 4.2.0 · C:/Users/utngo/OneDrive/Máy tính/R_EDA-Regression/

```
+     library(minerva)
+ }
> ?mine
> mine(dB, dMF)  # 0.1468241
$MIC
[1] 0.1468241

$MAS
[1] 0.0190443

$MEV
[1] 0.1468241

$MCN
[1] 2

$`MIC-R2`
[1] -0.06660419

$GMIC
[1] 0.1321531

$TIC
[1] 20.24044

>
```


Histogram of MIC_results
Frequency
0
200
400
600
800
0.00
0.05
0.10
0.15
0.20
MIC_results

Module 3: Dimension Reduction

I.Lý thuyết

Mục đích của bài học

- Nắm được phương pháp PCA – Principal Component Analysis hay Phân tích thành phần chính nhằm giảm kích thước dữ liệu
- Có thể thực hiện PCA trên dữ liệu và diễn giải kết quả
- Có thể sử dụng dữ liệu để xác định các dữ liệu đang được quan tâm
- Tìm hiểu về các lựa chọn thay thế như Phép chiếu và các phương pháp nhúng

1. Giới thiệu về PCA – Principal Component Analysis: Phân tích thành phần chính

Mục tiêu của PCA là chuyển đổi một số biến có thể tương quan thành một số nhỏ hơn của các biến không tương quan được gọi là các thành phần chính. Số lượng biến nhỏ hơn có thể được sử dụng để giảm chiều dữ liệu và trực quan hóa.
Hệ số tương quan: Mối tương quan giữa dữ liệu trong thế giới thực là phổ biến, nguyên nhân là do các mối quan hệ hoặc các yếu tố gây nhiễu.
Vậy tương quan ở đây thực sự có nghĩa là gì?

2. PCA và hệ số tương quan

PCA chuyển đổi một tập hợp các quan sát về các biến có thể tương quan thành một tập hợp các giá trị của các biến không tương quan được gọi là các thành phần chính. Thành phần chính đầu tiên là phép chiếu dữ liệu thành một thứ nguyên có phương sai càng cao càng tốt (càng nhiều dự thay đổi trong dữ liệu càng tốt); mỗi thành phần thành công đến lượt nó có phương sai cao nhất có thể với điều kiện là nó trực giao với các thành phần trước. Dó đó, PCs cung cấp một cái nhìn tốt nhất vè cấu trúc của dữ liệu và giải thích phương sai của nó. Điều này rất hữu ích cho EDA khi không thể trực quan hóa các dữ liệu lớn.

3. R và PCA

Mục đích:

- Giảm thứ nguyên
- Phân tích tầm quan trọng tương đối của các thứ nguyên

R có hai phương thức khác nhau cho PCA: prcomp() và princomp(). 2 phương thức sử dụng các cách tiếp cận toán học khác nhau nhưng kết quả hầu như giống hệt nhau. Prcomp() ổn định hơn về mặt số học. Tuy nhiên, chúng cũng sử dụng

các tên khác nhau có các phần tử trong danh sách kết quả của chúng. Sau đây là 1 vài ví dụ về sự khác nhau của 2 phương thức:

prcomp()	princomp()	
center	center	The vector that was subtracted to center the data
sdev	sdev	Standard deviations for each dimension of the rotated data
rotation	loadings	The actual principal components
x	scores	The rotated data, i.e. after projection along each PC

e.g. use data$x for the rotated results of a prcomp() call, but use data$scores if the result came from princomp()

4. Tổng kết

- PCA là một công cụ khá hữu ích cho EDA
- Rất tốt để phát hiện các biến gây nhiễu
- Ứng dụng vào dữ liệu biểu hiện gen để có thể xác định các mẫu đặc trưng:
    - PCA của gen
    - PCA của các mẫu
- Nhược điểm: Mất thời gian cho việc giải thích các biến
- Các phương pháp dựa trên mô hình có thể được áp dụng nhanh chóng

II. Thực hành

1. PCA Introduction

Ví dụ về dữ liệu tổng hợp

- 500 mẫu được phân phối như sau:

```
set.seed(2707)
x1 <- rnorm(500,0,1)
y1 <- rnorm(500,0,1)
```

- Tạo ra y2 phụ thuộc vào x1

```
80  y2 <- 2*x1 + y1
81  mean(y2)
82  y2 <- y2-mean(y2)
83  mean(y2)
84  sd(y2)
85  y2 <- y2 / sd(y2)
86  sd(y2)
87  print(sd(y2), digits=22)
88
89
90  # Create a lattice plot with two rows and two columns
91
```

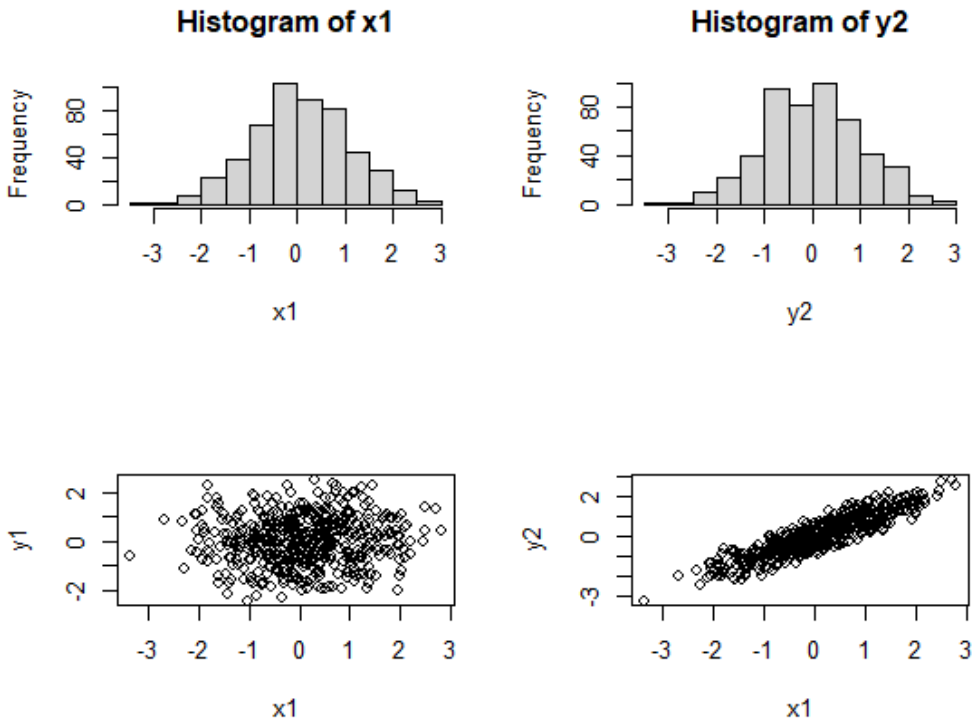
77:21 (Untitled)

Console | Terminal | Jobs

R 4.2.0 · D:/TCD/R_EDA-DimensionReduction/

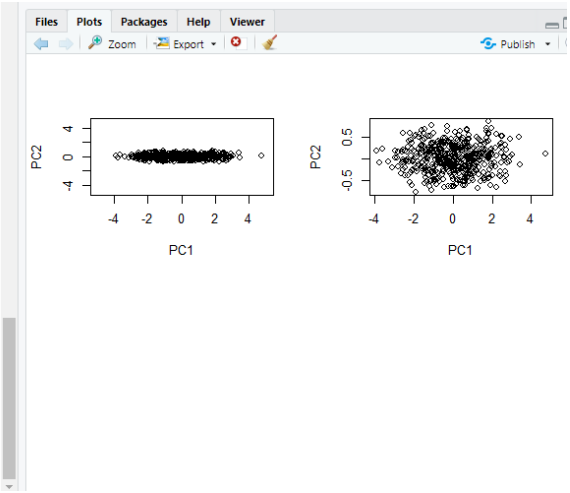
```
> y2 <- y2-mean(y2)
> mean(y2)
[1] 9.265592e-18
> sd(y2)
[1] 2.286244
> y2 <- y2 / sd(y2)
> sd(y2)
[1] 1
> print(sd(y2), digits=22)
[1] 0.9999999999999998889777
```

- Tạo 4 biểu đồ minh họa



- Tính PCA của x1 với y2 và hiển thị các kết quả

```
> pcaSample <- prcomp(cbind(x1,y2))
> # here are the information items from the returned list of results
> pcaSample
Standard deviations (1, .., p=2):
[1] 1.3841321 0.3117919

Rotation (n x k) = (2 x 2):
          PC1        PC2
x1 -0.7096365 -0.7045679
y2 -0.7045679  0.7096365
> pcaSample$sdev
[1] 1.3841321 0.3117919
> pcaSample$rotation
          PC1        PC2
x1 -0.7096365 -0.7045679
y2 -0.7045679  0.7096365
> summary(pcaSample)
Importance of components:
                          PC1     PC2
Standard deviation     1.3841 0.31179
Proportion of Variance 0.9517 0.04829
Cumulative Proportion  0.9517 1.00000
> head(pcaSample$x)
           PC1          PC2
[1,] -2.3671309 -0.022112496
[2,]  0.6864310 -0.403159564
[3,]  1.1365843 -0.143493494
[4,] -0.3451804 -0.004860618
[5,]  0.6207848 -0.472388914
[6,] -0.1397211 -0.105823361
> plot(pcaSample$x, xlim=c(-5,5), ylim=c(-5,5))
> plot(pcaSample$x)
>
```



2. EDA với phương pháp PCA

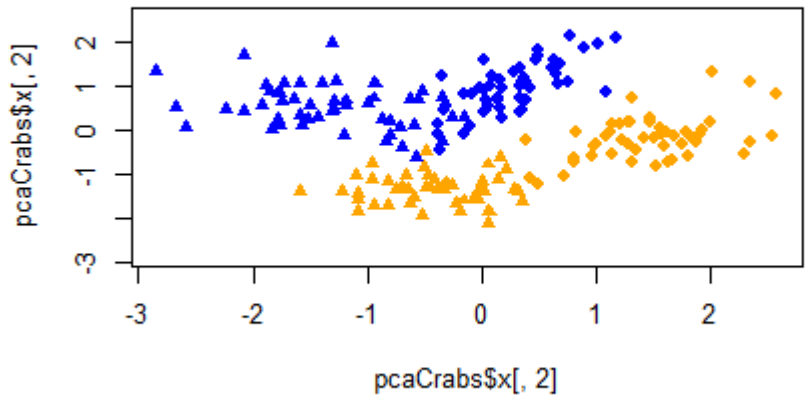
- Load dữ liệu

```
> library(MASS)
> data(crabs)
> head(crabs)
  sp sex index   FL  RW   CL   CW  BD
1  B   M     1  8.1 6.7 16.1 19.0 7.0
2  B   M     2  8.8 7.7 18.1 20.8 7.4
3  B   M     3  9.2 7.8 19.0 22.4 7.7
4  B   M     4  9.6 7.9 20.1 23.1 8.2
5  B   M     5  9.8 8.0 20.3 23.0 8.2
6  B   M     6 10.8 9.0 23.0 26.5 9.8
```

Các thông tin gồm có: màu sắc (xanh và cam – B & O), giới tính (đực và cái – M & F) và các chỉ số cơ thể.

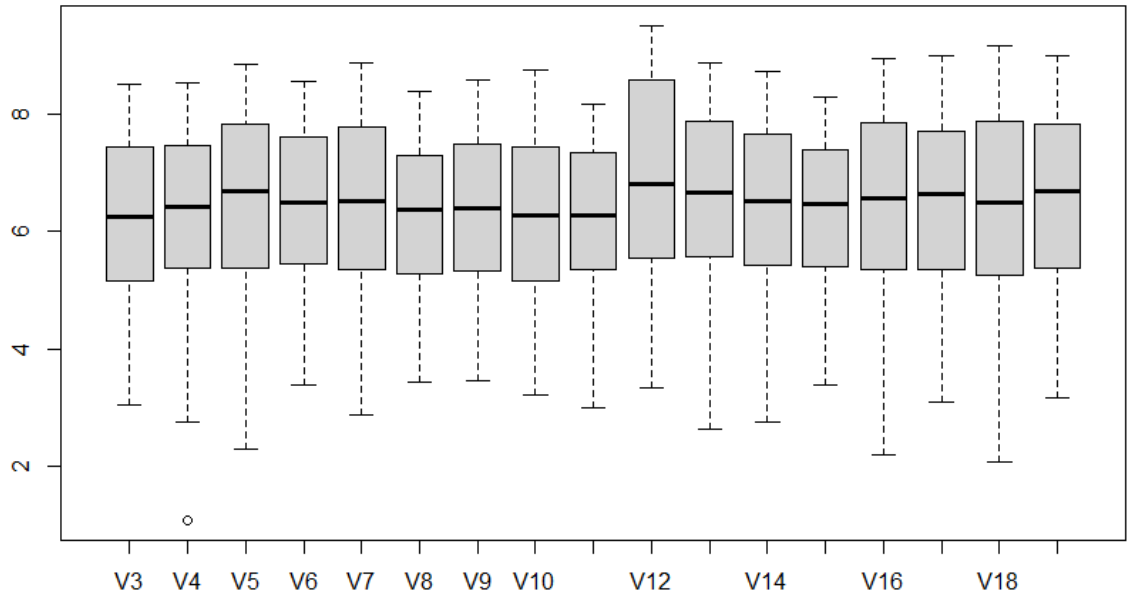
2.1 Sự quan trọng tương đối của PCs

Task: Vẽ biểu đồ gồm giới tính và màu sắc của cua: màu cam và xanh hình tròn cho cua cái còn hình tam giác cho cua đực



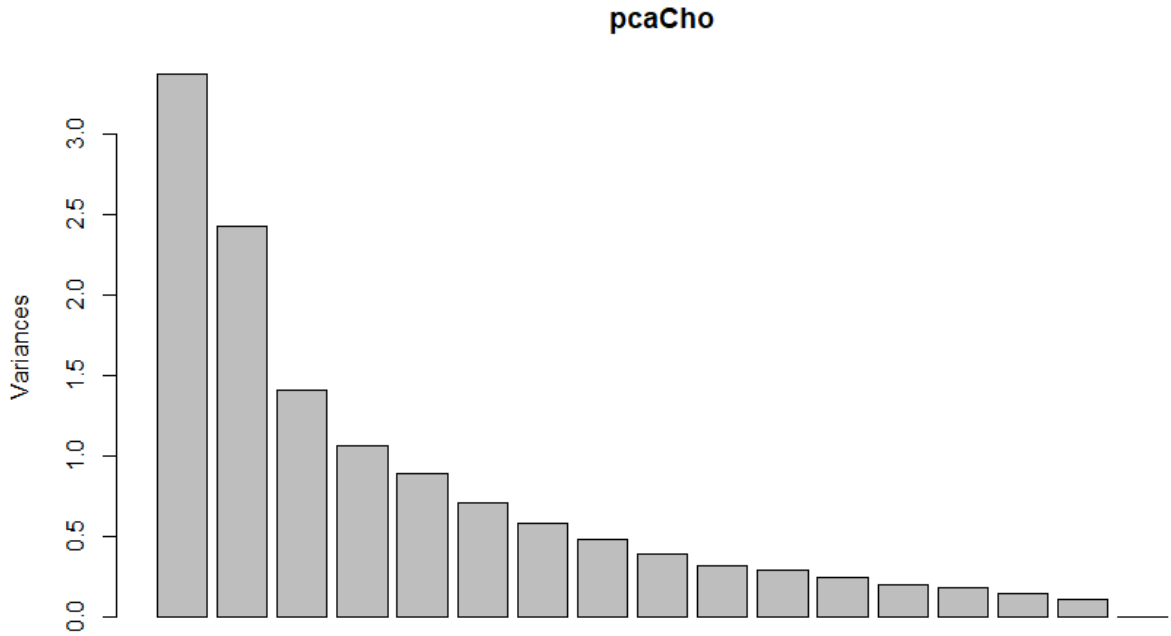
2.2 Dữ liệu chu trình tế bào

Tải tập dữ liệu Chodata gồm 237 genes được biết hoặc nghi vấn có liên quan trong chu kỳ tế bào. Tạo boxplot so sánh xu hướng chung

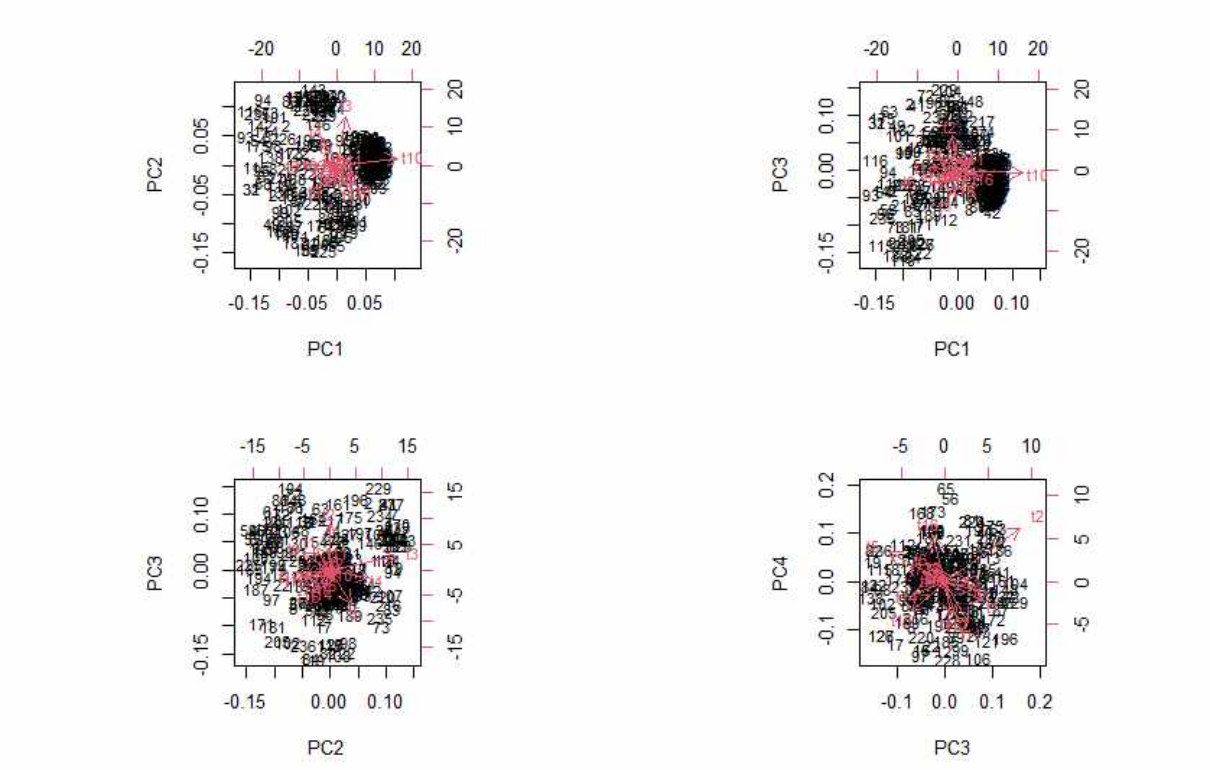


2.3 Khám phá thành phần chính – PCs

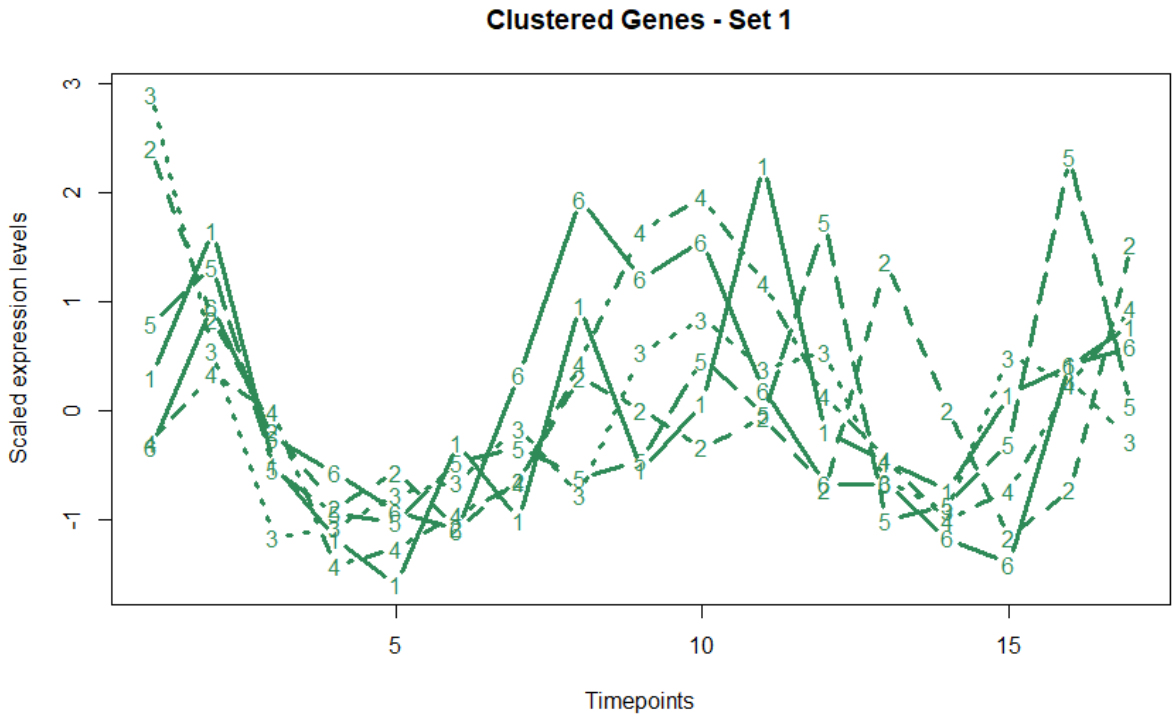
- Tính PCS



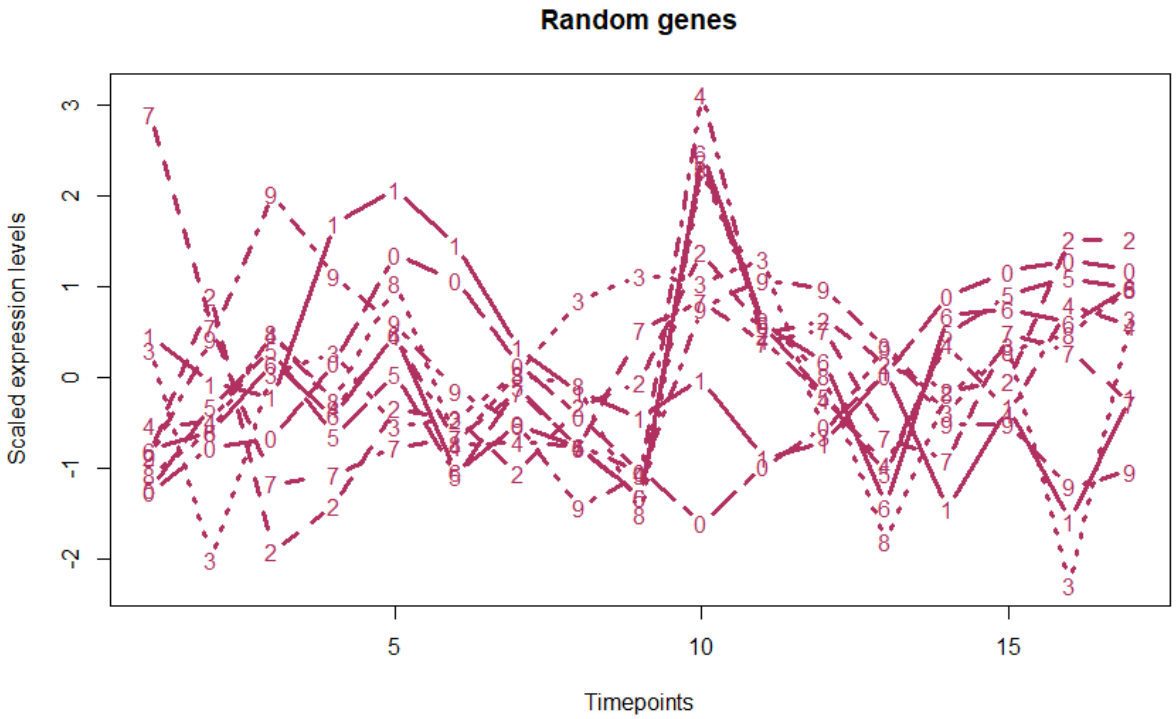
- Khám phá các mối tương quan dựa theo vài thành phần chính đầu tiên



2.4 Khám phá một số genes tương tự

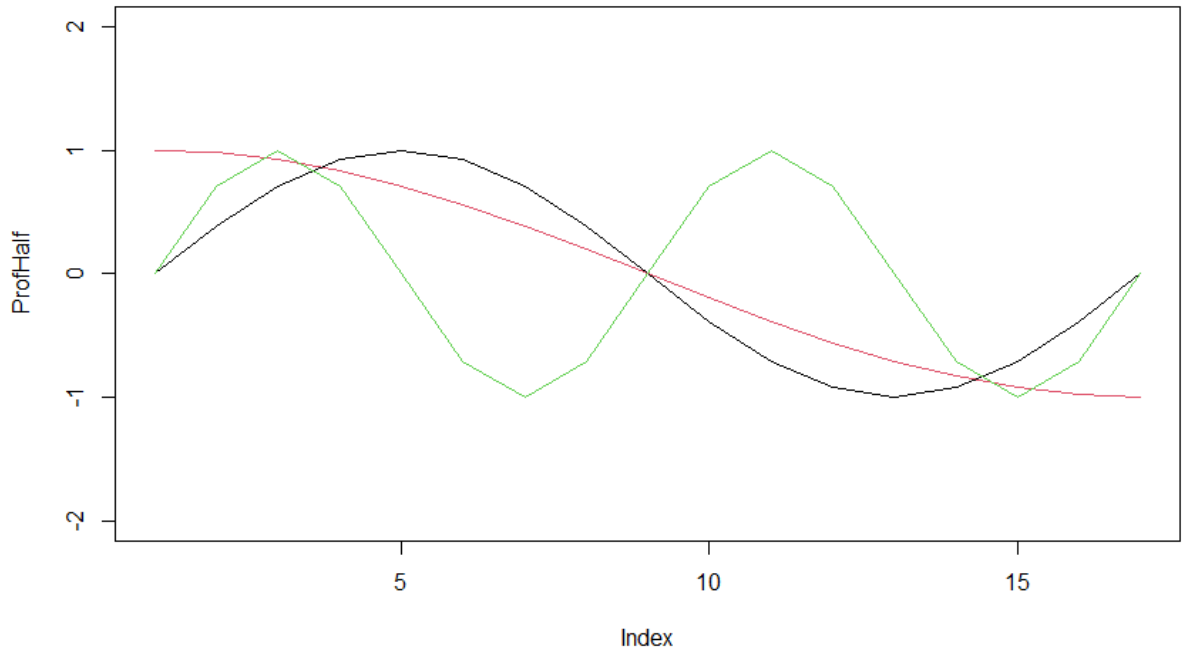


Task: Chọn và vẽ biểu đồ của 10 genes ngẫu nhiên để so sánh



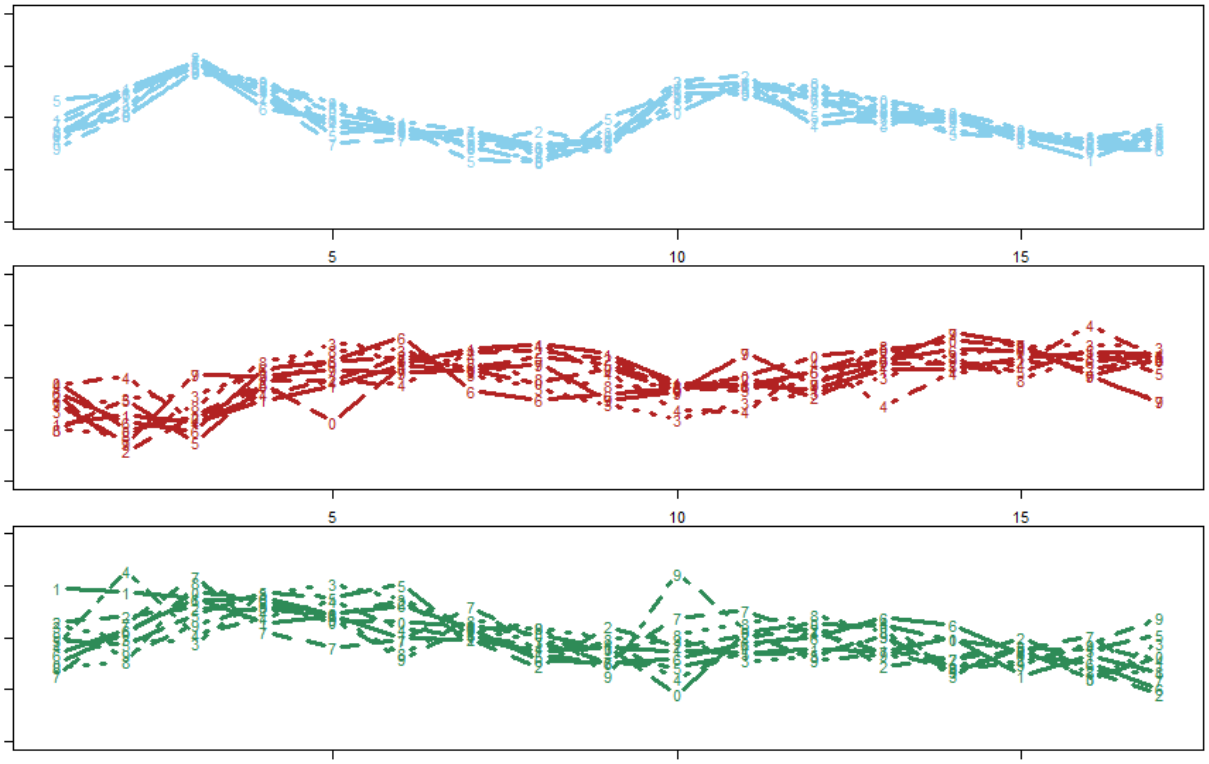
3. Khám phá dữ liệu liên quan đến mô hình

- Sử dụng 3 mô hình đơn giản: 1 nửa dóng hình sin, 1 sóng hình sin và 2 sóng hình sin. Sau đây là 1 ví dụ:



- Chọn các chỉ số từ biểu đồ và vẽ các tọa độ song song của các giá trị thực tế

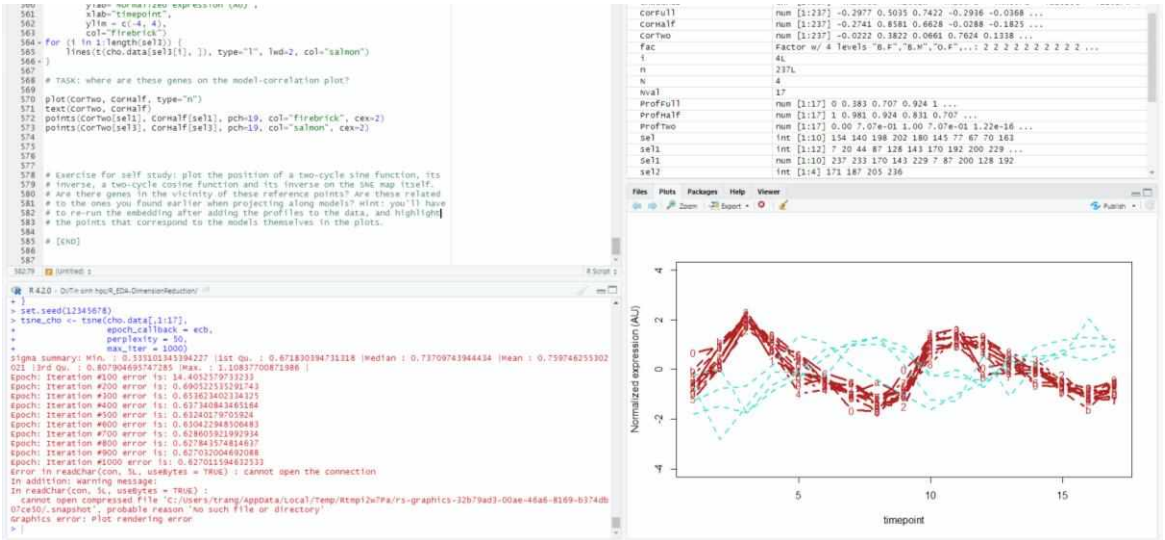
Khởi tạo sel1, sel2, sel3 gồm 10 genes bất kỳ. Vẽ trên cùng 1 biểu đồ với trục X là Timepoints, trục Y là Genes tương tự (giá trị giới hạn trục từ -4 đến 4)



4. t-SNE

t-SNE là một thuật toán nhằm giảm chiều dữ liệu một các hiệu quả. Tiếp theo chúng ta sẽ sử dung thuật toán t-SNE để khám phá một số dữ liệu trước đây.

- Chọn một tập hợp con như ở trên và xem xét nó trong quá trình lặp với 3 biểu đồ sin lúc trước

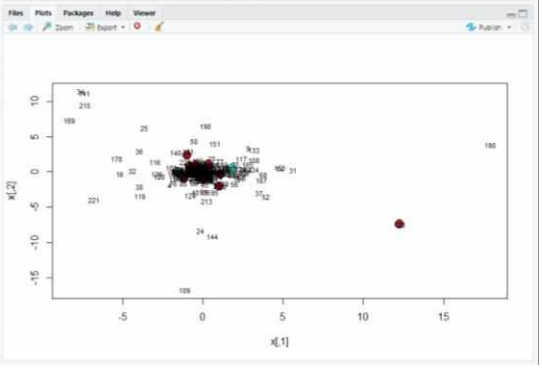


- Chọn một nhóm genes có giá trị gần với nhóm genes đã chọn ban đầu

```
        ylim = c(-4, 4),
        col="firebrick")
for (i in 1:length(sel1)) {
    lines(t(cho.data[sel1[i], ]), type="l", lwd=2, col="salmon")
}

# TASK: Where are these genes on the model-correlation plot?

plot(CorTwo, CorHalf, type="n")
text(CorTwo, CorHalf)
points(CorTwo[sel1], CorHalf[sel1], pch=19, col="firebrick", cex=2)
points(CorTwo[sel3], CorHalf[sel3], pch=19, col="salmon", cex=2)



# Exercise for self study: plot the position of a two-cycle sine function, its
# inverse, a two-cycle cosine function and its inverse on the SNE map itself.
# Are there genes in the vicinity of these reference points? Are these related
# to the ones you found earlier when projecting along models? Hint: you'll have
# to re-run the embedding after adding the profiles to the data, and highlight
# the points that correspond to the models themselves in the plots.

# (END)
```

CorHalf	num [1:237] -0.2741 0.8581 0.6628 -0.0288 -0.1825 ...
CorTwo	num [1:237] -0.0222 0.3822 0.0661 0.7624 0.1338 ...
fac	Factor w/ 4 levels "B.F","B.M","O.F",..: 2 2 2 2 2 2 2 2 2 2 ...
f	4L
n	237L
N	4
Nval	17
ProfFull	num [1:17] 0 0.383 0.707 0.924 1 ...
ProfHalf	num [1:17] 1 0.981 0.924 0.831 0.707 ...
ProfTwo	num [1:17] 0.00 7.07e-01 1.00 7.07e-01 1.22e-16 ...
Sel	int [1:10] 154 140 198 202 180 145 77 67 70 163
sel1	int [1:12] 7 20 44 87 128 143 170 192 200 229 ...
sel2	int [1:10] 217 233 170 143 229 7 87 200 128 192
sel3	int [1:4] 171 187 205 236

```
> }
> set.seed(12345678)
> tsne_cho <- tsne(cho.data[,1:17],
+                  epoch_callback = ecb,
+                  perplexity = 50,
+                  max_iter = 1000)
sigma summary: Min. : 0.533101345394227 |1st Qu. : 0.671830394731318 |Median : 0.737097439444434 |Mean : 0.7397462155022
021 |3rd Qu. : 0.807904695747285 |Max. : 1.10837700871986 |
Epoch: Iteration #100 error is: 14.4052579733233
Epoch: Iteration #200 error is: 0.690522535291743
Epoch: Iteration #300 error is: 0.653823402334325
Epoch: Iteration #400 error is: 0.637340843465164
Epoch: Iteration #500 error is: 0.632401797059924
Epoch: Iteration #600 error is: 0.630422948506483
Epoch: Iteration #700 error is: 0.628605921992934
Epoch: Iteration #800 error is: 0.627843574814637
Epoch: Iteration #900 error is: 0.627032004692088
Epoch: Iteration #1000 error is: 0.627011594632533
Error in readChar(con, 5L, useBytes = TRUE) : cannot open the connection
In addition: Warning message:
In readChar(con, 5L, useBytes = TRUE) :
  cannot open compressed file 'C:/Users/trang/AppData/Local/Temp/RtmpI2w7Pa/rs-graphics-32b79ad1-00ae-46a6-8169-b174db
07ce50/.snapshot', probable reason 'No such file or directory'
Graphics error: Plot rendering error
>
```



Module 4: Clustering

I.Lý thuyết

1. Mục tiêu

• Hiểu các nguyên tắc phân cụm;

• Có thể áp dụng các phương pháp phân cấp - và phân vùng cơ bản cho dữ liệu;

• Biết cách diễn giải kết quả;

• Hiểu kiểm soát chất lượng cụm;

• Biết về các lựa chọn thay thế.

2. Sự phức tạp trong dữ liệu tương tác

• Các miền trong cấu trúc protein

• Các gen cốt lõi

• Protein có chức năng tương tự

(dựa trên các thuộc tính tương tự được đo lường, ví dụ: cốt lõi)

3. Giới thiệu về phân cụm

Phân cụm là một ví dụ về học tập không giám sát, hữu ích cho việc phân tích các mẫu trong dữ liệu có thể dẫn đến khám phá lớp học.

Phân cụm là sự phân vùng của một tập dữ liệu thành các nhóm phần tử tương tự với nhau hơn là các phần tử trong các nhóm khác.

Phân cụm là một phương pháp hoàn toàn chung chung có thể được áp dụng cho gen, mẫu hoặc cả hai.

4. Phân cụm theo thứ bậc

- Cho N mục và chỉ số khoảng cách:

1. Gán mỗi mục vào "cụm" của riêng nó. Khởi tạo ma trận khoảng cách giữa các cụm là khoảng cách giữa các mục.

2. Tìm cặp cụm gần nhất và hợp nhất

chúng thành một cụm duy nhất.

3. Tính toán khoảng cách mới giữa các cụm.

4. Lặp lại 2-3 cho đến khi tất cả các cụm đã được hợp nhất thành một cụm duy nhất.

- Giải phẫu của phân cụm theo thứ bậc

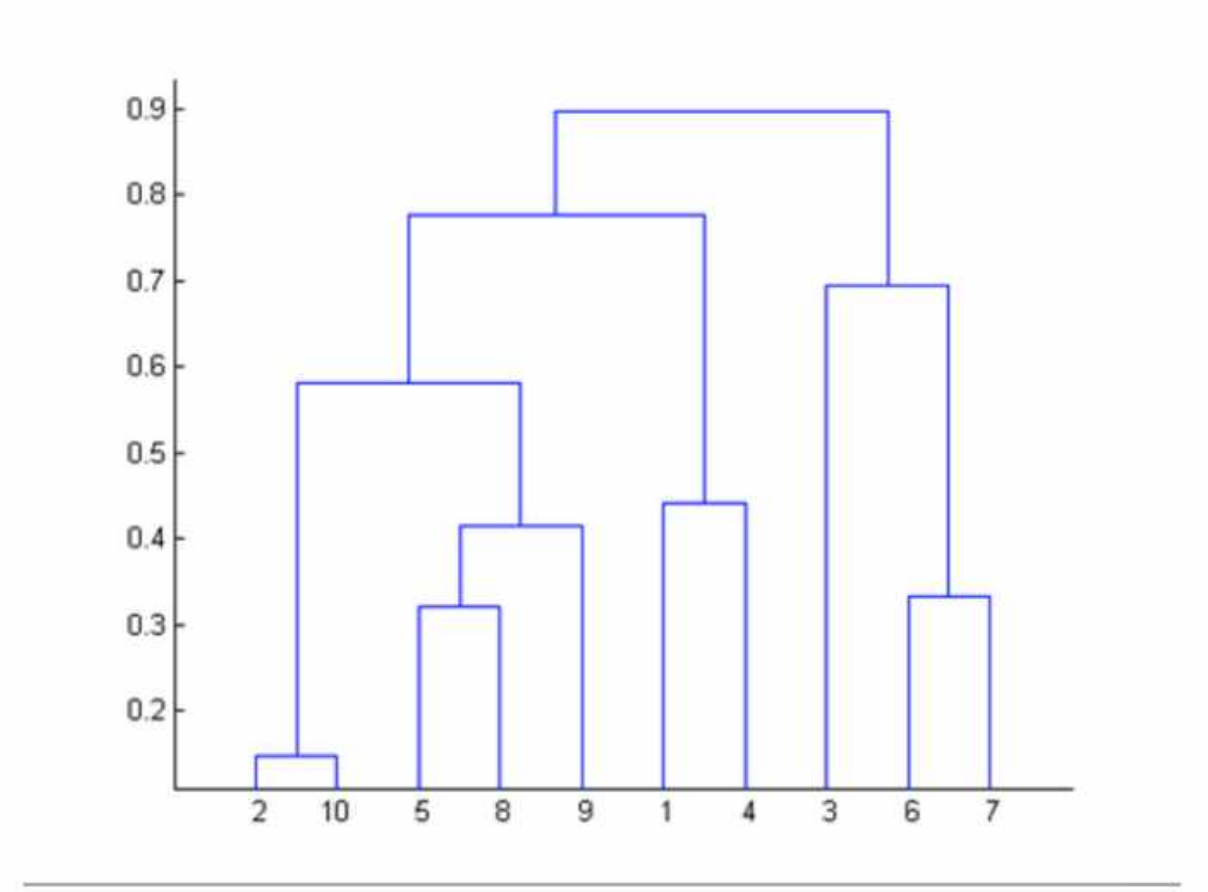
  • Ma trận khoảng cách

  • Phương pháp liên kết

  - Đầu ra: dendrogram

  • một cây xác định mối quan hệ giữa các đối tượng và khoảng cách giữa các cụm

  • một chuỗi các cụm lồng nhau



Ưu nhược điểm của phân cụm phân cấp

Ưu điểm	**Nhược điểm**

Có thể có cụm nhỏ lồng bên trong những cụm lớn	Các cụm có thể không được đại diện một cách tự nhiên bằng cấu trúc phân cấp
Không cần xác định trước số lượng cụm	Cần thiết phải "cắt" hình ảnh dendrogram để tạo ra các cụm
Phương thức liên kết linh hoạt	Việc phân cụm từ dưới lên có thể dẫn đến cấu trúc kém ở phần ngọn của cây. Các lần tham gia sớm không thể "hoàn tác"

5. Các phương pháp phân chia (phân vùng)

- Giải phẫu của một phương pháp dựa trên phân vùng

  • ma trận dữ liệu

  • chức năng khoảng cách

  • số lượng nhóm

- Đầu ra

  • phân công nhóm của mọi đối tượng

- Phương pháp:

+ Chọn K nhóm

  • khởi tạo các trung tâm nhóm

  • hay còn gọi là centroid, medoid

  • gán từng đối tượng cho tâm gần nhất theo số liệu khoảng cách

  • chỉ định lại (hoặc tính toán lại) các trung tâm

  • lặp lại 2 bước cuối cùng cho đến khi nhiệm vụ ổn định

6. So sánh K-means với K-medoids

K-means	K-medoids
Centroid là trung bình các cụm	Centroid là 1 đối tượng thực tế, giảm thiểu tổng số trong khoảng cách cụm
Centroid cần được tính toán lại mỗi lần lặp lại	Centroid có thể được xác định bằng cách tra cứu nhanh vào ma trận khoảng cách
Khởi tạo khó khăn vì khái niệm về centroid có thể không rõ ràng trước khi bắt đầu	Khởi tạo chỉ đơn giản là K đối tượng được chọn ngẫu nhiên
Thuật toán: kmeans	Thuật toán: PAM

Ưu nhược điểm của phân chia (phân vùng)

Ưu điểm	Nhược điểm
Số lượng nhóm được xác định rõ ràng	Phải chọn số lượng nhóm
Sự phân công rõ ràng, xác định của một đối tượng cho một nhóm	Đôi khi các đối tượng không vừa khít với bất kỳ cụm nào
Các thuật toán đơn giản để suy luận	Có thể tối ưu cục bộ. Thường yêu cầu khởi động lại nhiều lần với các lần khởi tạo ngẫu nhiên.

K-means

1. Chia dữ liệu thành K cụm Khởi tạo các trọng tâm với giá trị trung bình của các cụm

2. Gán từng mục cho cụm có tâm gần nhất

3. Khi tất cả các đối tượng đã được chỉ định, hãy tính toán lại trọng tâm (giá trị trung bình)

4. Lặp lại 2-3 cho đến khi các trung tâm không còn di chuyển

=> K-means và các phương pháp phân cụm phân cấp là các kỹ thuật đơn giản, nhanh chóng và hữu ích

II. Thực hành

1. Load dữ liệu

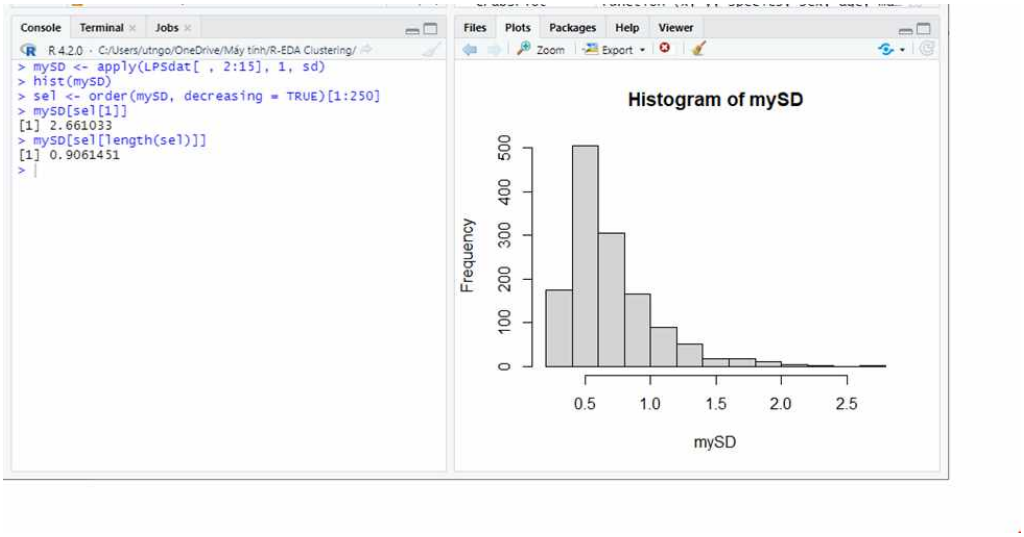
myClustering.R | LPSdat

Filter

	genes	B.ctrl	B.LPS	MF.ctrl	MF.LPS	NK.ctrl	NK.LPS	Mo.ctrl	Mo.LPS	pDC.ctrl	pDC.LPS	DC1.ctrl	DC1.LPS	DC2.ctrl	DC2.LPS	cluster
1	Cd69	-12.9	-10.5	-13.1	-11.6	-12.9	-9.5	-13.1	-10.6	-13.3	-10.3	-13.2	-10.7	-13.1	-10.8	1
2	Cxcl10	-11.4	-8.5	-10.9	-6.0	-12.0	-6.1	-12.0	-6.4	-10.9	-7.5	-12.4	-6.4	-11.6	-5.9	1
3	Ifi47	-12.1	-8.9	-11.9	-9.8	-11.6	-8.8	-12.1	-9.1	-12.2	-9.1	-12.3	-10.0	-12.2	-9.1	1
4	Ifit2	-12.8	-9.8	-12.8	-8.9	-13.3	-10.5	-12.9	-8.9	-13.0	-9.6	-12.9	-9.0	-12.9	-8.8	1
5	Ifit3	-12.6	-7.9	-12.0	-9.2	-12.9	-9.1	-12.5	-8.5	-12.8	-10.8	-12.7	-10.7	-12.4	-7.8	1
6	Igtp	-12.3	-9.5	-12.1	-9.6	-11.7	-8.9	-12.3	-10.1	-12.1	-9.3	-12.5	-10.4	-12.6	-9.9	1
7	Irf7	-11.9	-9.4	-10.4	-9.1	-12.2	-9.5	-11.5	-9.0	-10.0	-8.1	-11.8	-8.2	-11.6	-8.4	1
8	Irgm1	-12.1	-8.9	-12.8	-9.5	-12.2	-9.5	-12.3	-9.7	-12.2	-9.4	-12.2	-9.5	-12.2	-9.6	1
9	Isg15	-12.0	-9.1	-11.7	-8.3	-12.0	-8.4	-12.0	-8.5	-11.0	-8.6	-12.4	-8.9	-11.9	-8.1	1
10	Isg20	-12.1	-9.7	-12.9	-10.6	-12.9	-9.9	-12.2	-10.0	-12.4	-10.4	-12.6	-10.5	-12.2	-9.1	1
11	Mx1	-12.4	-9.0	-11.7	-9.0	-12.6	-9.8	-12.7	-8.9	-11.5	-8.9	-12.5	-8.6	-12.3	-8.7	1
12	Mx2	-12.9	-10.3	-12.8	-10.3	-13.3	-11.6	-13.0	-11.2	-13.3	-10.7	-13.0	-11.0	-13.2	-10.3	1
13	Oasl1	-12.7	-10.1	-12.3	-9.2	-13.0	-10.4	-12.7	-9.3	-12.1	-9.8	-13.1	-9.0	-12.9	-9.7	1
14	Parp14	-11.9	-9.3	-11.4	-9.0	-11.8	-9.3	-11.9	-9.2	-11.2	-8.7	-12.4	-9.3	-11.9	-9.0	1
15	Rsad2	-13.1	-10.3	-12.5	-9.3	-13.0	-10.5	-12.9	-9.5	-12.6	-10.6	-13.0	-10.2	-13.0	-8.8	1
16	Usp18	-12.9	-9.8	-12.3	-10.3	-12.6	-9.5	-12.7	-9.5	-12.6	-8.7	-12.6	-10.1	-12.2	-9.1	1
17	I830012O16Rik	-13.1	-9.4	-12.9	-11.1	-13.3	-11.1	-13.0	-10.3	-13.0	-12.0	-13.2	-12.3	-13.1	-9.2	1
18	Ifit1	-12.8	-9.3	-12.8	-9.1	-12.9	-9.0	-12.8	-8.8	-12.6	-10.5	-13.2	-10.5	-12.6	-8.4	1
19	Amica1	-12.7	-11.0	-12.8	-13.0	-12.9	-11.1	-12.3	-11.1	-10.9	-9.4	-12.0	-10.6	-12.2	-10.7	2
20	Arf4	-11.4	-10.9	-11.2	-10.6	-11.5	-10.5	-11.8	-10.2	-11.4	-9.8	-11.5	-10.6	-11.5	-10.5	2
21	Batf	-12.8	-11.5	-12.4	-11.3	-12.6	-10.2	-12.2	-10.6	-12.2	-11.9	-12.3	-11.7	-12.7	-11.3	2
22	BC006779	-12.0	-10.1	-12.9	-10.4	-12.7	-10.4	-12.4	-10.6	-12.1	-10.1	-12.7	-10.7	-12.3	-10.3	2
23	Bcl3	-12.1	-11.2	-12.8	-11.3	-11.9	-10.8	-11.8	-10.6	-12.0	-11.4	-12.4	-11.1	-11.5	-10.4	2

- Select top 250 standard deviations of expression values:

Biểu đồ dữ liệu của mySD:

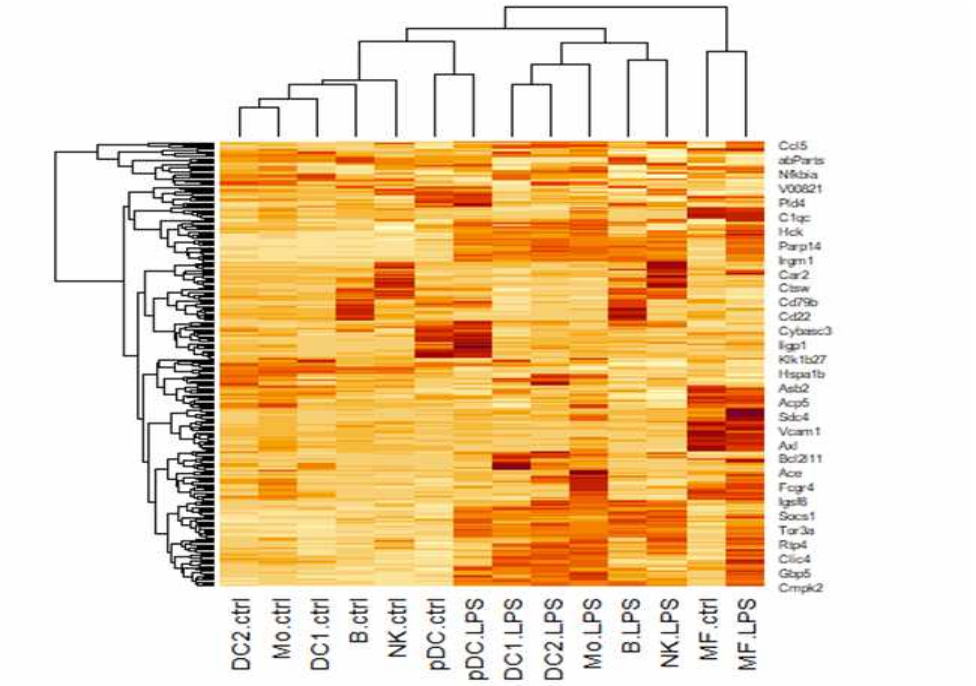


-Để thuận tiện cho việc phân cụm sử dụng các đối tượng dữ liệu là ma trận số đơn giản

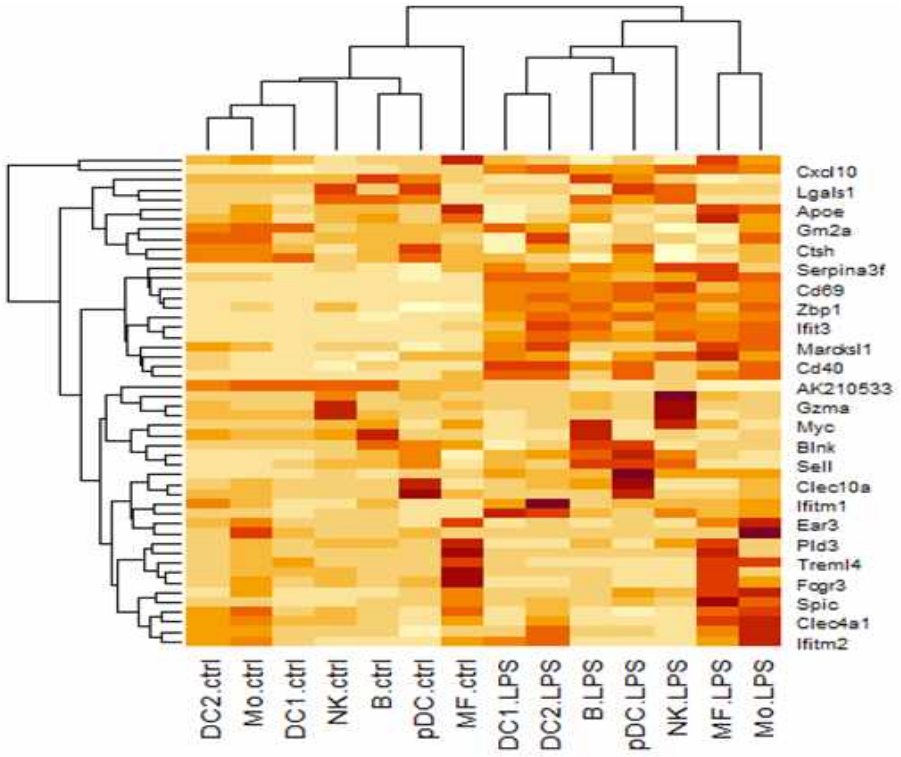
```
> dat <- matrix(numeric(250 * 14), nrow = 250)
> for (i in 1:length(sel)) {
+   dat[i, ] <- as.numeric(LPSdat[sel[i], 2:15])
+ }
> rownames(dat) <- LPSdat$genes[sel]
> colnames(dat) <- colnames(LPSdat[2:15])
> str(dat)
 num [1:250, 1:14] -11.4 -6.2 -5.9 -12.6 -8.6 -12.6 -12.8
 -6.3 -12.4 -12.2 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:250] "Cxcl10" "abParts" "M34473" "Ccl4_Scy
A4" ...
  ..$ : chr [1:14] "B.ctrl" "B.LPS" "MF.ctrl" "MF.LPS"
 ...
```

2. HEATMAPS

Heatmaps là một phần cơ bản của phân tích biểu hiện gen.

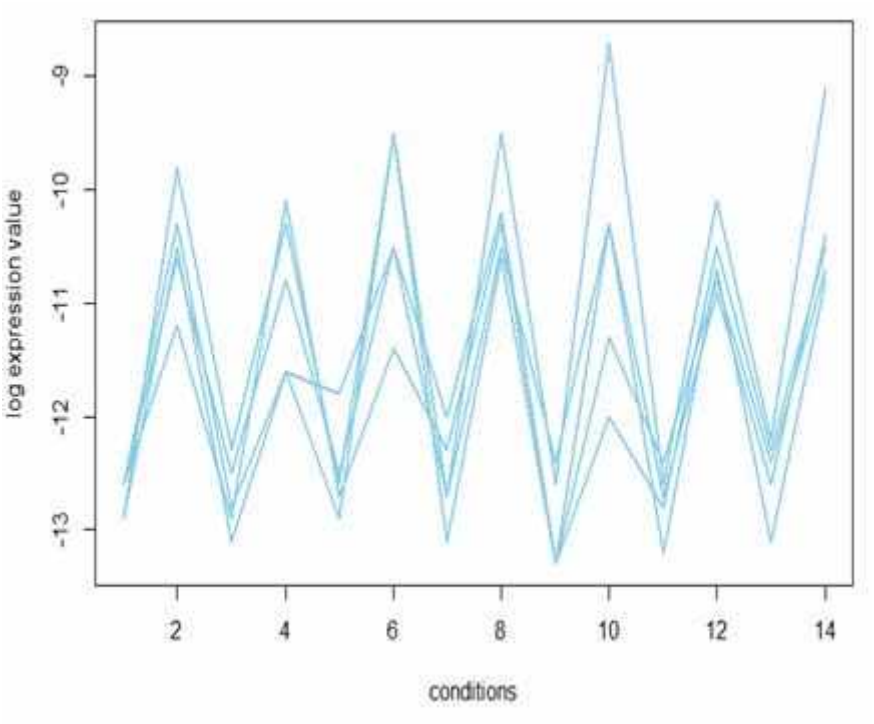


Để dễ quan sát vẽ bản đồ nhiệt gen thứ 5:

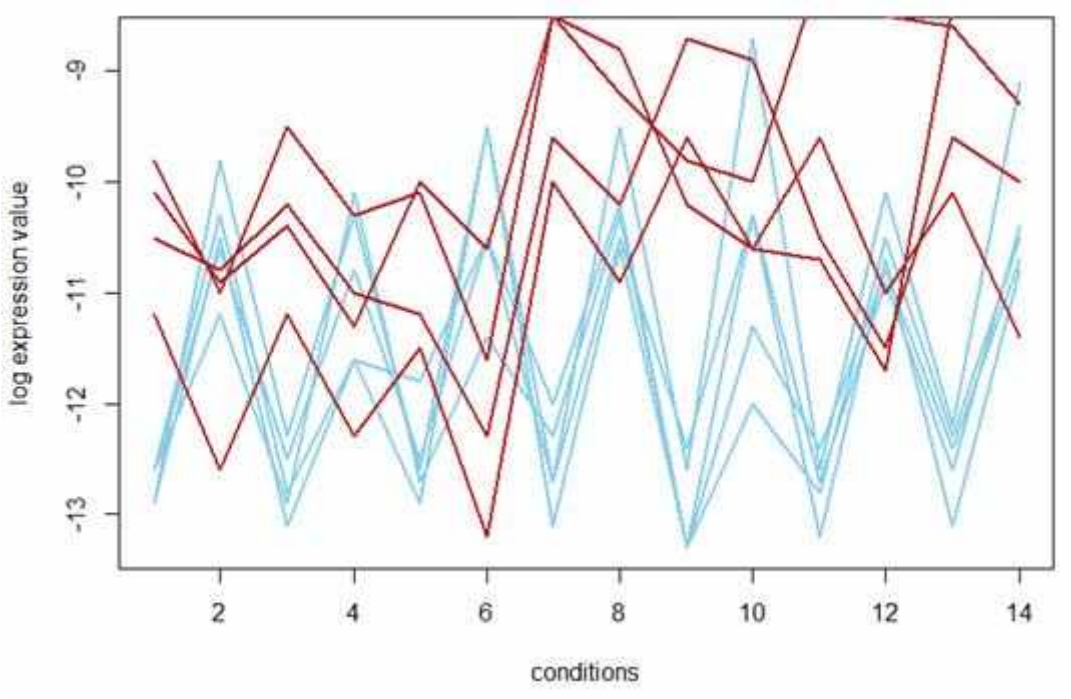



```
> # What's the actual range of values?
> range(dat[,1])
[1] -13.3  -5.9
```

Sử dụng biểu đồ "tọa độ song song" - matplot () để xem các cấp độ biểu thức thực tế

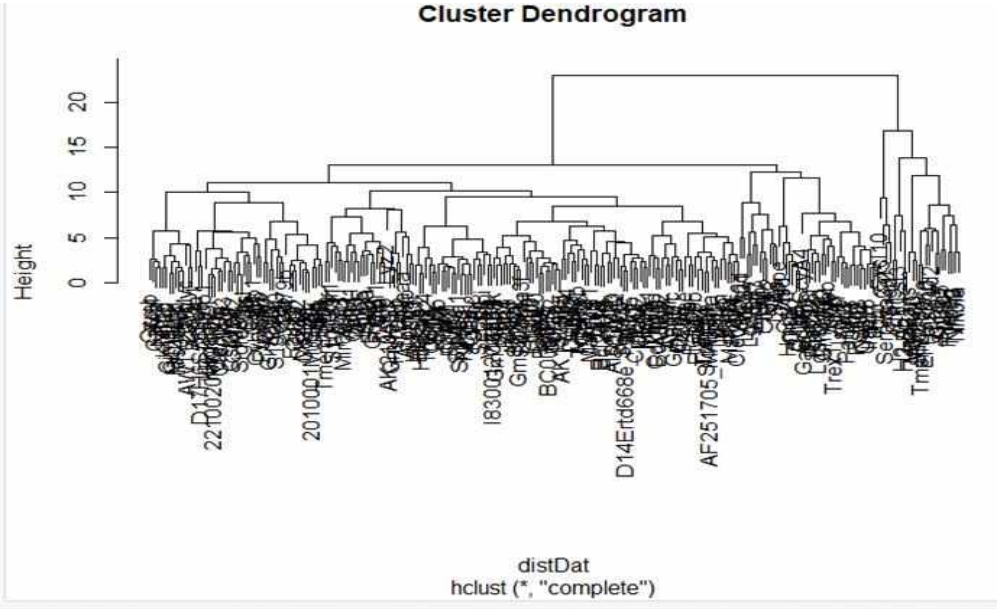


Chồng gen ở set 2 vào đồ thi bằng hàm lines()



3. Hierarchical clustering(Phân cụm phân cấp)

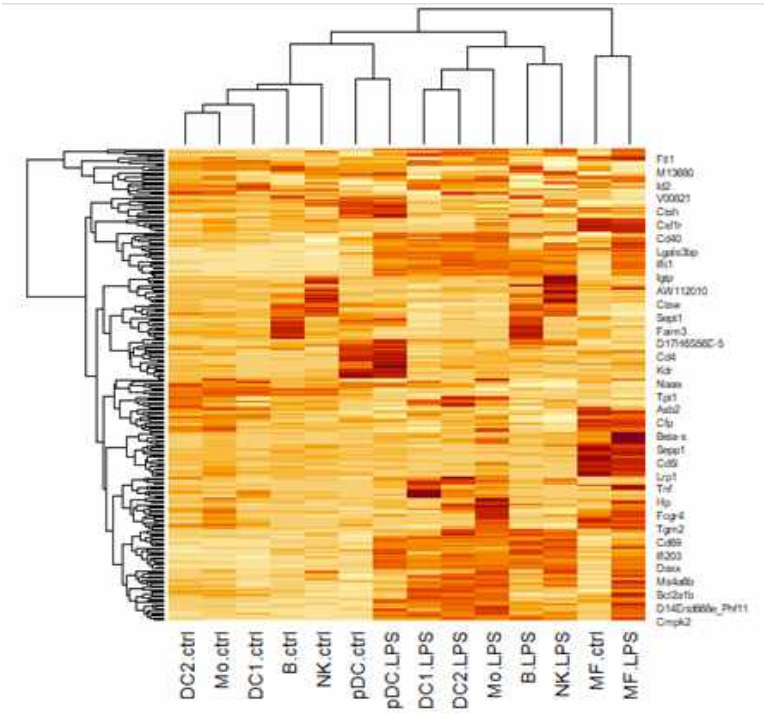
Phân cụm phân cấp có lẽ là kỹ thuật cơ bản nhất. Các dendrograms trên các hàng và cột. Để phân cụm phân cấp, trước tiên chúng ta cần tạo một bảng khoảng cách. Có nhiều cách để xác định khoảng cách hãy chọn mặc định: "Khoảng cách Euclid".



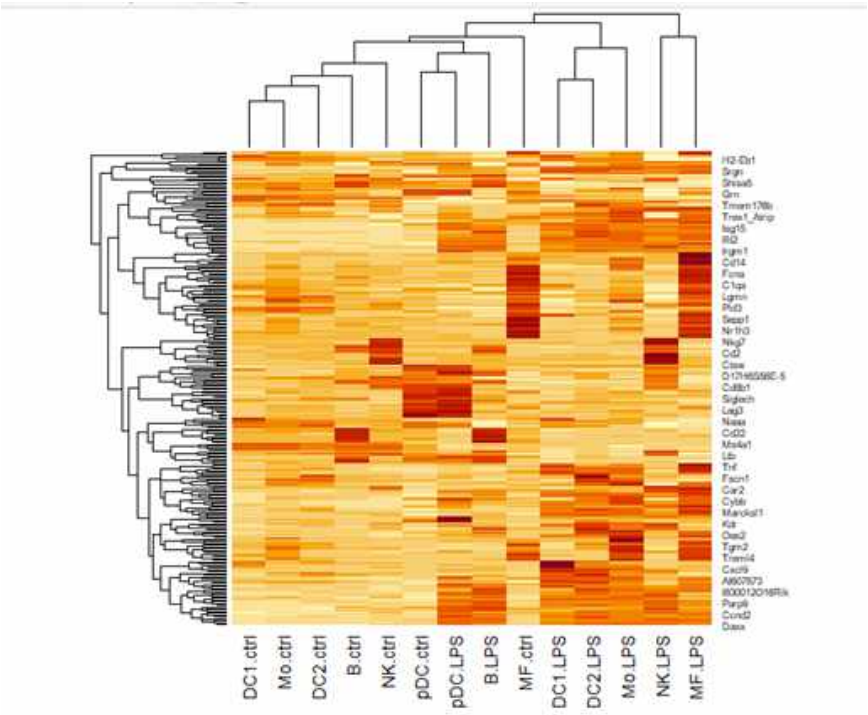
3.1 Exploring distance metrics:

Vẽ biểu đồ heatmap:

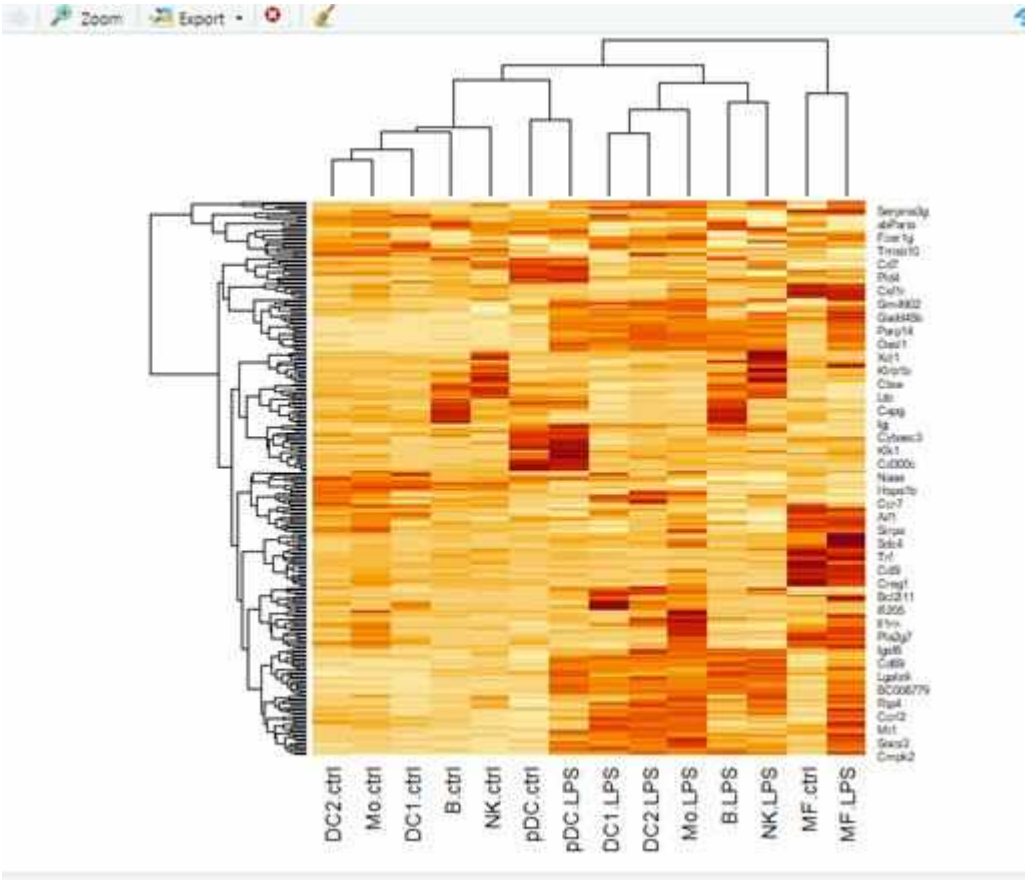
Sử dụng method=euclidian (khoảng cách Euclidian)



Sử dụng method=maximum (khoảng cách maximum)



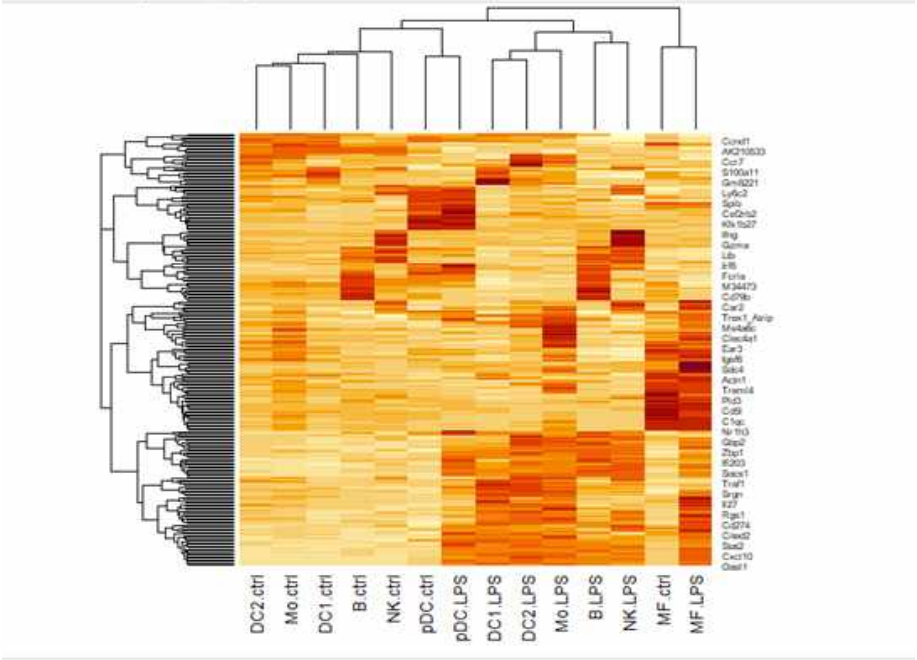
Sử dụng method=minkowski (khoảng cách minkowski)



Bạn cũng có thể xây dựng riêng hàm khoảng cách

Ví dụ

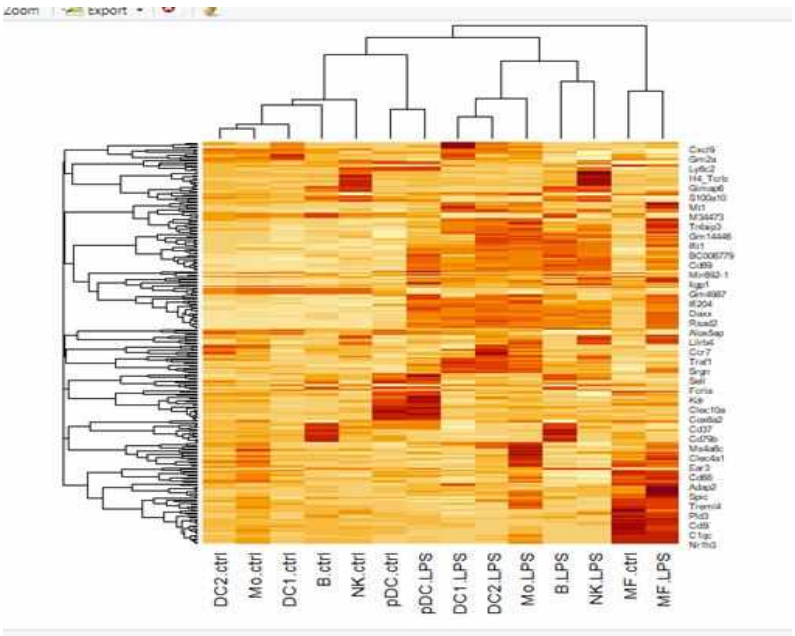
```
dCor <- function(x) as.dist(2 - cor(t(x)))
heatmap(dat, distfun = dCor)
```



Hay sử dụng **thuộc tính tương quan** làm khoảng cách

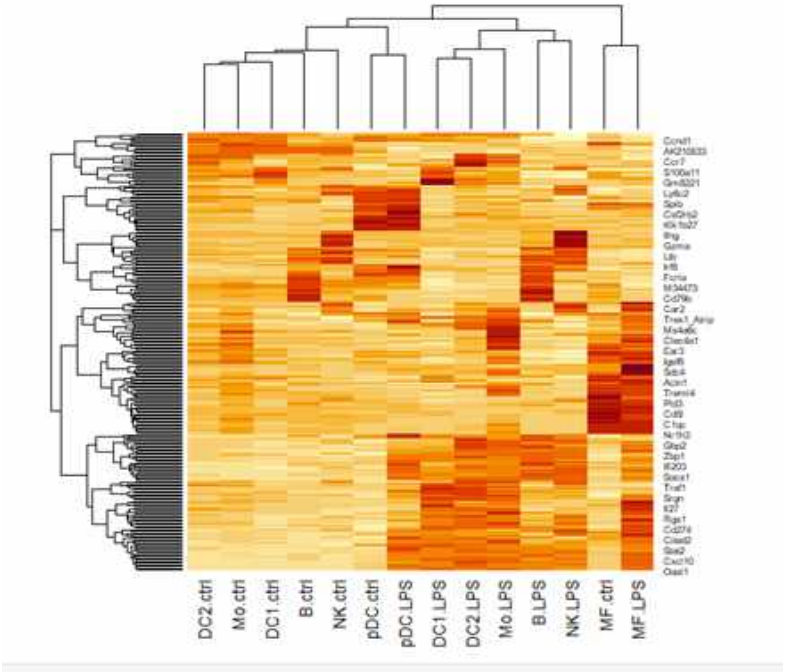
(1 - abs(pearson correlation))

```
dCorAbs <- function(x) as.dist(1 - abs(cor(t(x))))
heatmap(dat, distfun = dCorAbs)
```



(2 – pearson correlation)

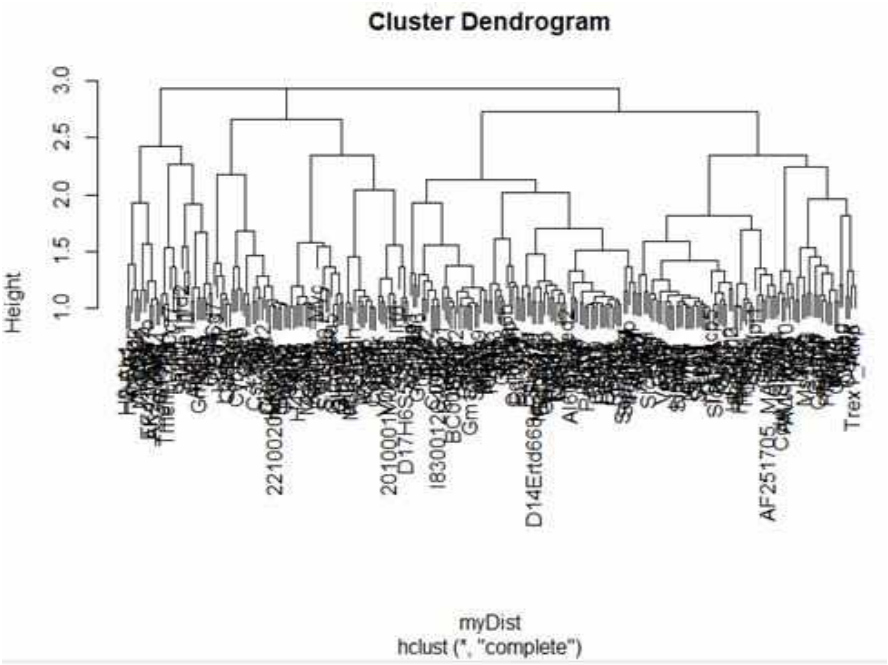
```
dCor <- function(x) as.dist(2 - cor(t(x)))
heatmap(dat, distfun = dCor)
```



Tính toán **tương quan** của dữ liệu dat sử dụng hàm dCor(2 minus correlation)

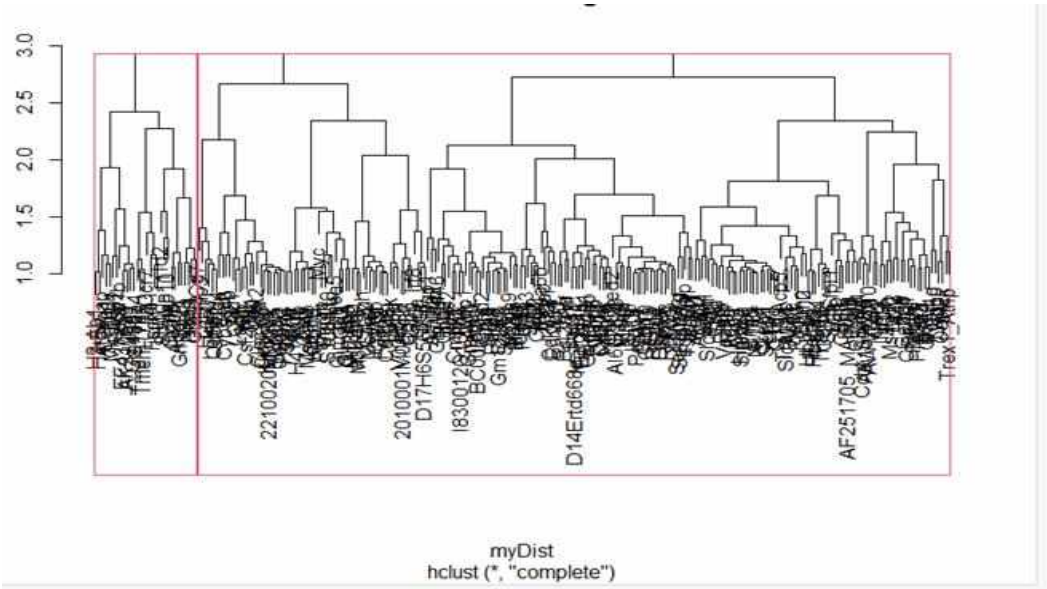
```
> N <- nrow(dat)
> myDist <- matrix(numeric(N*N), nrow = N)
> rownames(myDist) <- rownames(dat)
> colnames(myDist) <- rownames(dat)
> for (i in 1:N) {
+   for (j in i:N){
+     d <- 2 - cor(dat[i, ], dat[j, ])
+     myDist[i, j] <- d
+     myDist[j, i] <- d
+   }
+ }
> myDist <- as.dist(myDist)
> hc <- hclust(myDist)
> plot(hc)
>
```

Tạo biểu đồ dendrogram (myDist) từ ma trận khoảng cách (tương quan dữ liệu sử dụng hàm dCor đã tính toán)

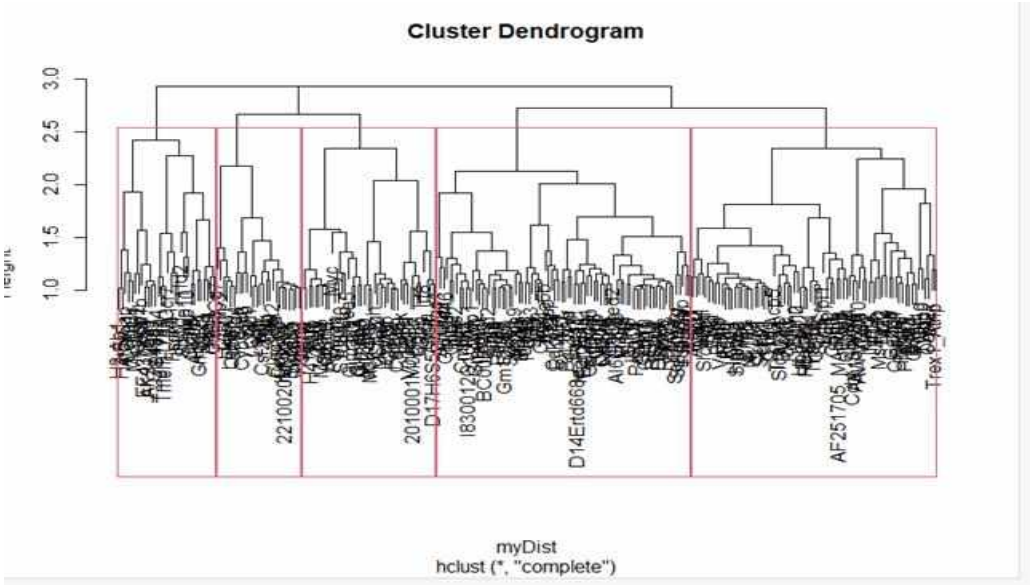


3.2.Phân cụm Dendrograms:

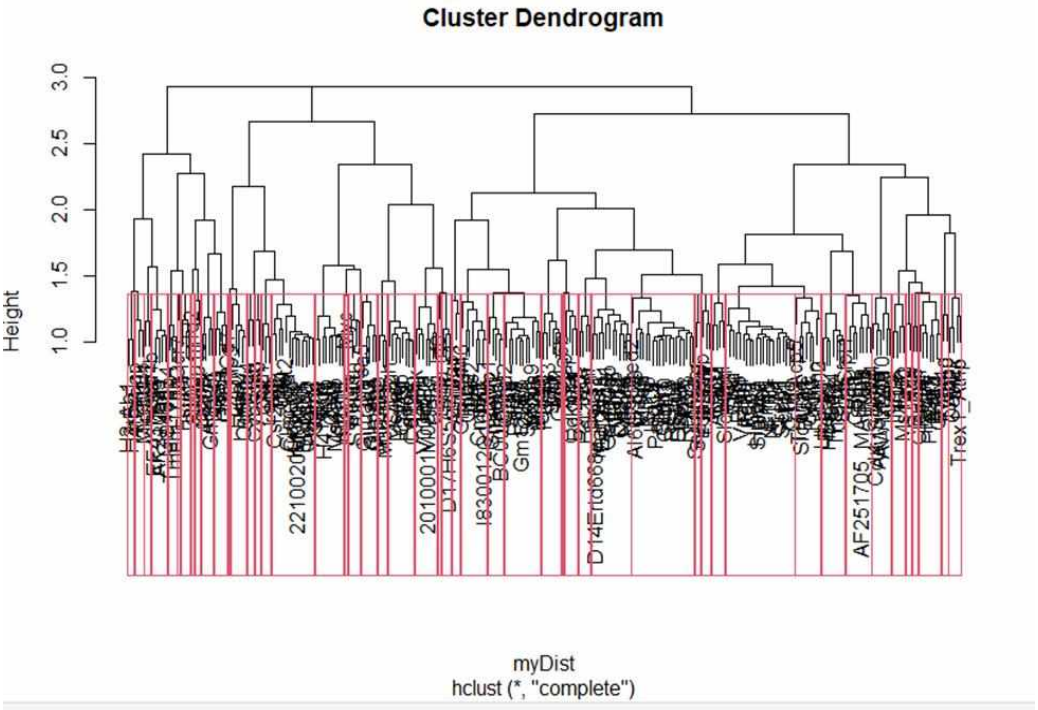
K=2



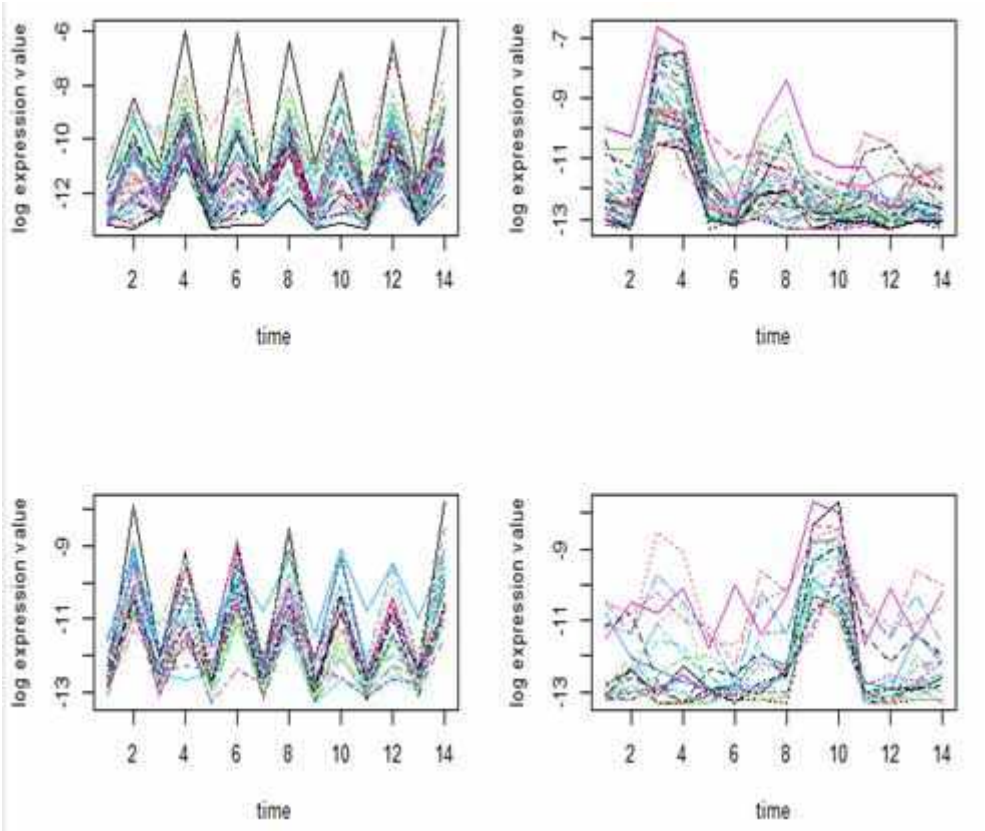
K=5



K=50

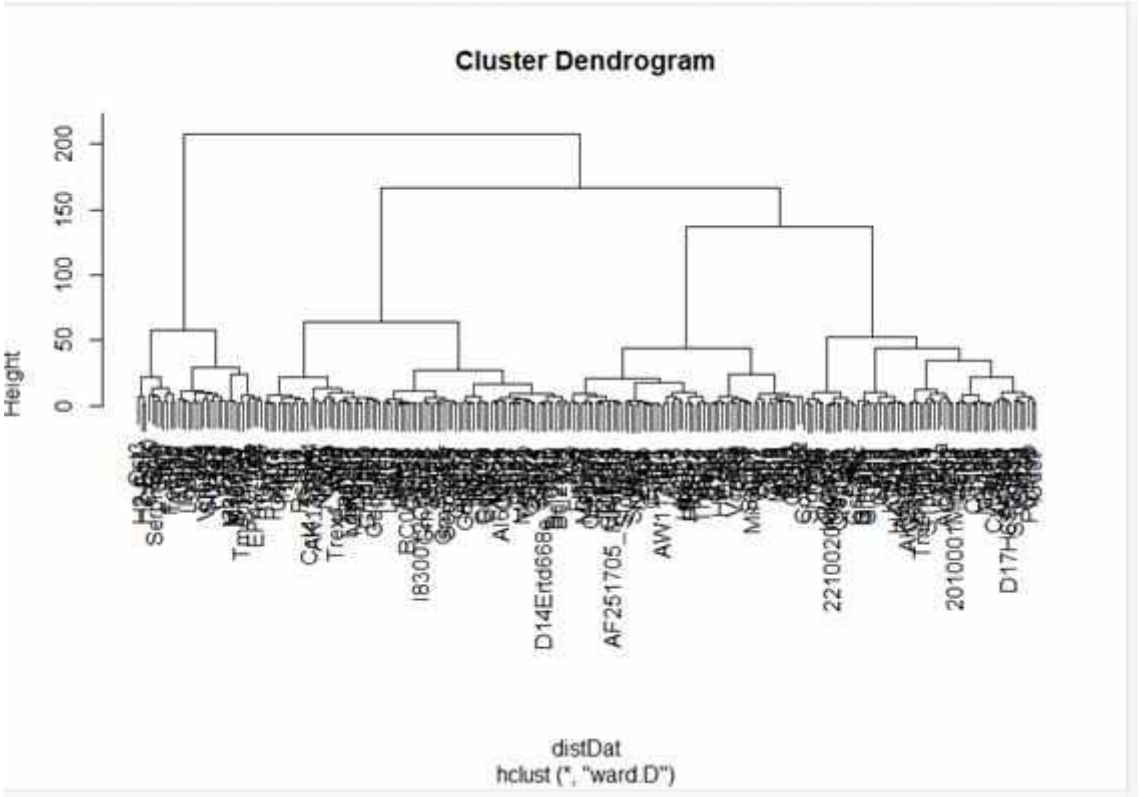


Vẽ 4 lớp lớn nhất (lớp 1,6,5,8)



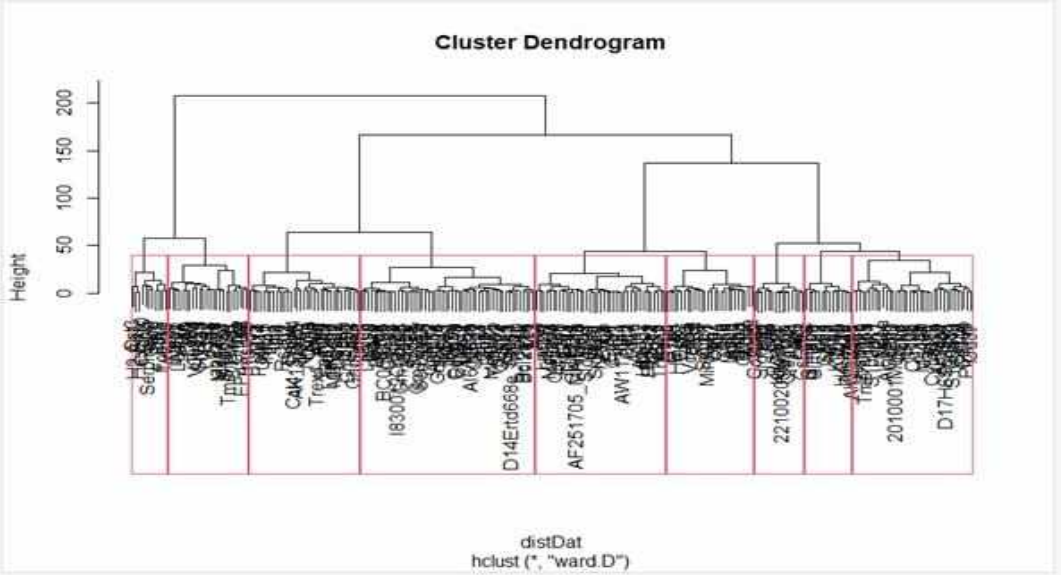
Với ma trận khoảng cách là khoảng cách Euclidian sử dụng phân cụm liên kết

```
hc.ward <-hclust(distDat, method = "ward.D", members=NULL)
```

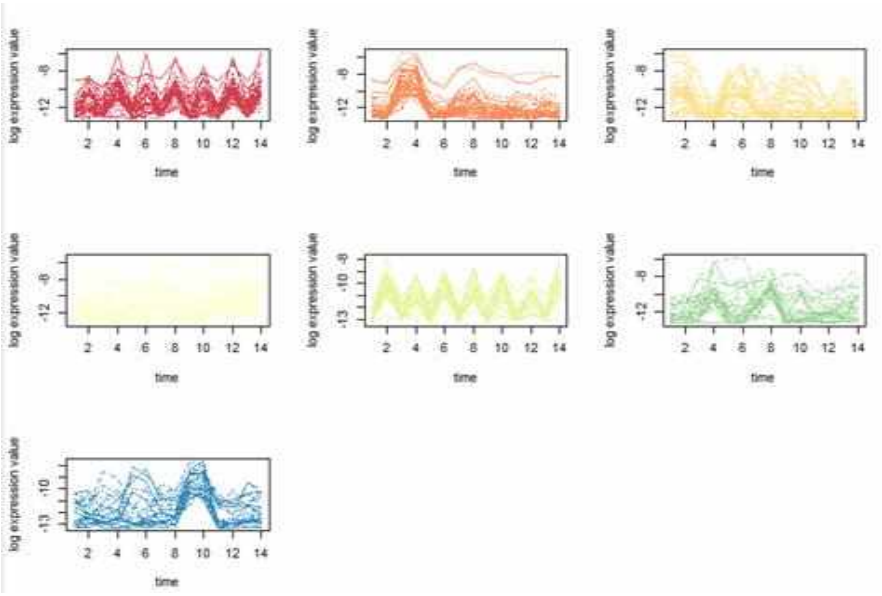


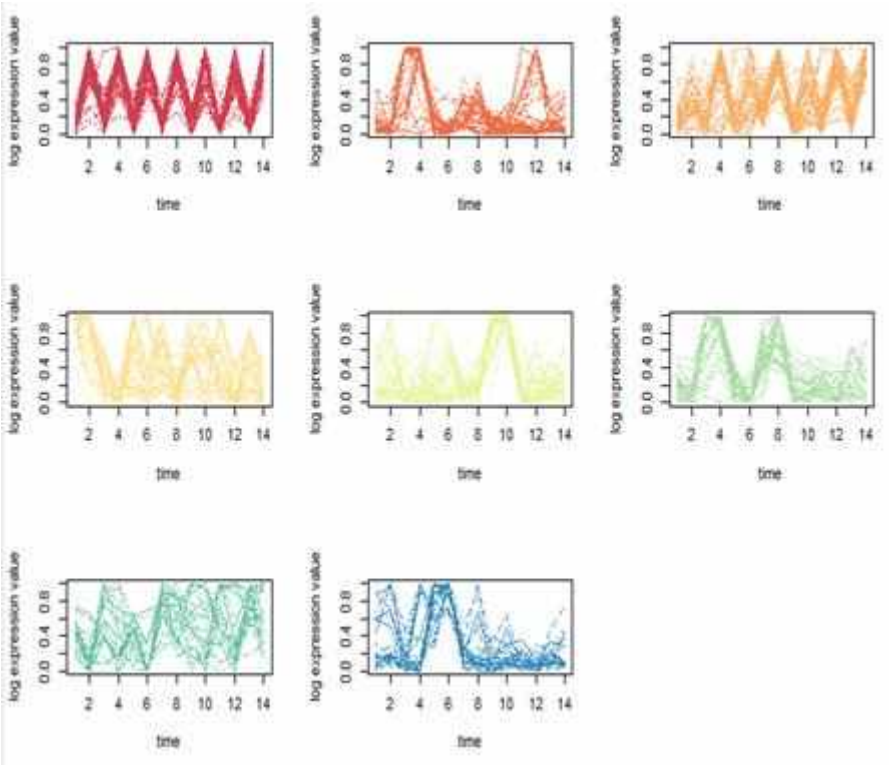
Chia dendrogram distDat thành các nhóm với k=9 và sắp xếp

```
# draw rectangles
rect.hclust(hc.ward,k=9)

# This looks reasonable ...
# Now retrieve the actual indices and use them to generate
# paralell coordinate plots.

class.ward<-cutree(hc.ward, k = 9)
sort(table(class.ward))
```



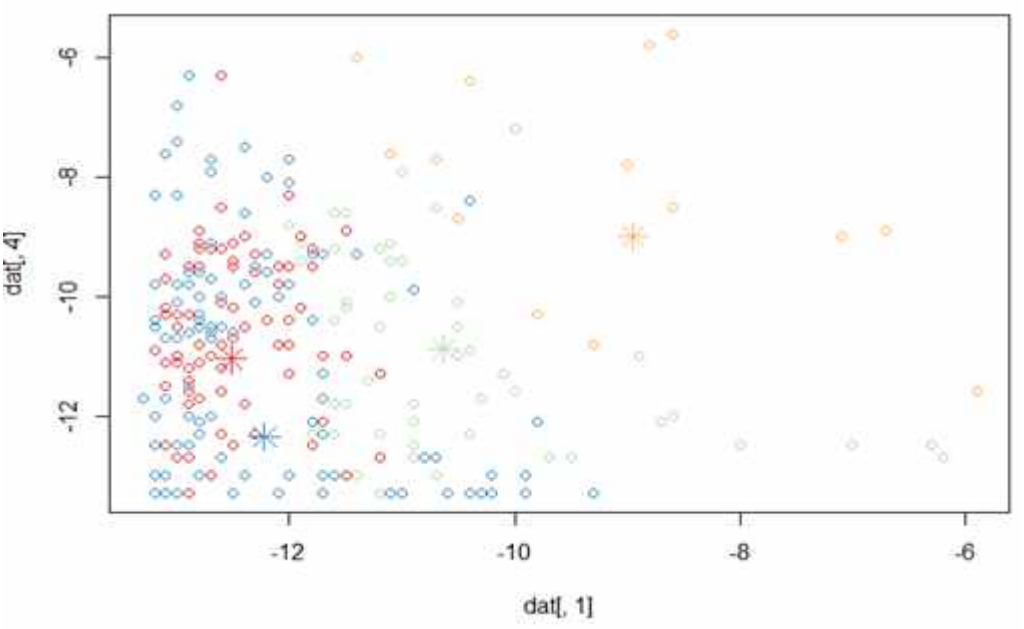
Vẽ 9 cụm có sử dụng thư viện màu



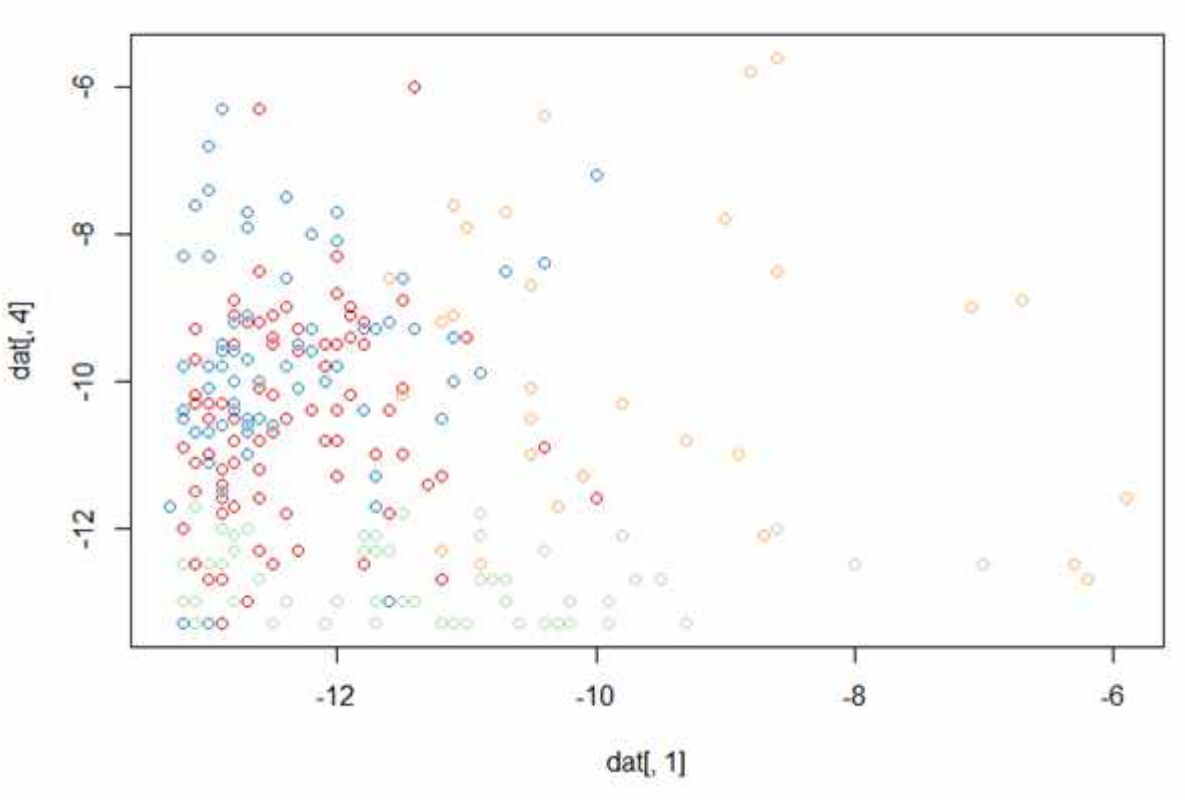


4. PARTITIONING CLUSTERING:

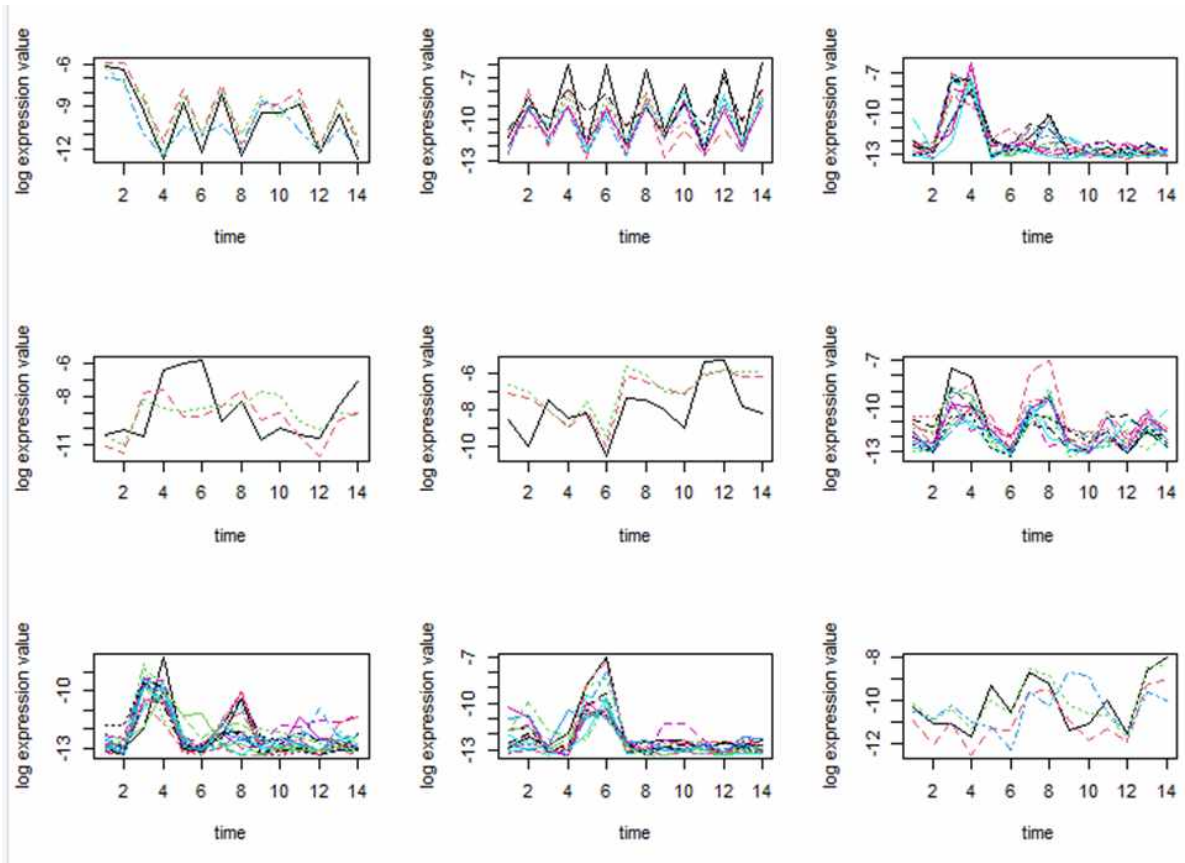
4.1. K-means(k=4)



4.2. K-medoids(k=4)

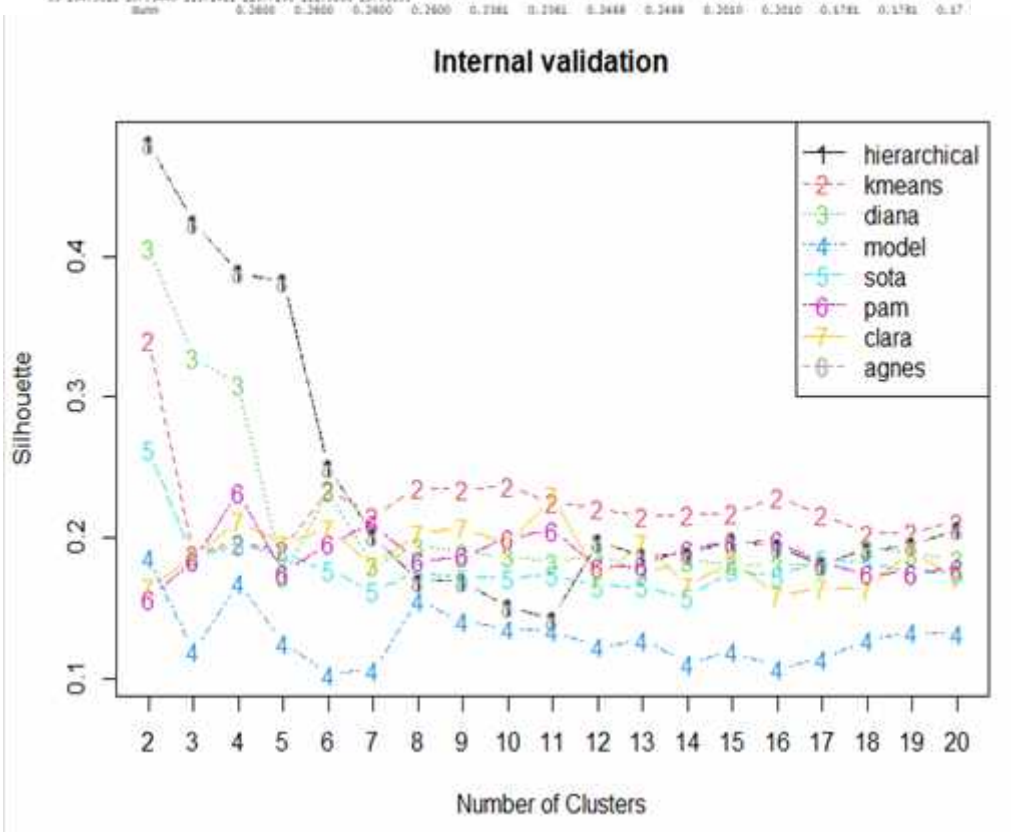


5. AFFINITY PROPAGATION CLUSTERING:



6. CLUSTER QUALITY METRICS

```
Cluster sizes:
 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

validation measures:
                                 2      3      4      5      6      7      8      9     10     11     12     13
 14     15     16     17     18     19     20

Hierarchical connectivity  18.7607 18.7607 18.8607 22.7540 36.3115 56.6782 62.2567 63.0012 82.2123 84.5661 90.5234 90.5234 101.60
99 104.9012 107.9440 113.1421 119.7190 121.3286 137.6893
           dunn             0.2600  0.2600  0.2600  0.2600  0.2381  0.2381  0.2468  0.2468  0.2010  0.2010  0.1781  0.1781  0.17
81  0.1781  0.1781  0.1781  0.1781  0.1781  0.2243
           silhouette       0.4830  0.4230  0.3883  0.3821  0.2504  0.2016  0.1689  0.1690  0.1512  0.1418  0.1976  0.1871  0.18
78  0.1983  0.1907  0.1869  0.1914  0.1944  0.2057
kmeans     connectivity  41.8988  69.7377  97.0238 106.5583 106.8869 111.9937 117.7167 122.1167 131.2036 144.1111 143.4167 134.1119 160.92
98 161.4484 163.2181 166.2969 178.7833 181.2795 177.4900
           dunn             0.0953  0.1098  0.1190  0.1197  0.1248  0.0941  0.0941  0.1013  0.0959  0.1501  0.1113  0.1113  0.11
13  0.1173  0.1158  0.1198  0.1221  0.1221  0.1221
           silhouette       0.3400  0.1886  0.1959  0.1862  0.2348  0.2150  0.2353  0.2342  0.2367  0.2232  0.2210  0.2148  0.21
67  0.2170  0.2297  0.2167  0.2052  0.2040  0.2110
diana      connectivity  27.9651  35.5948  48.8706  70.2981  82.2349 106.7179 134.3171 151.1583 154.2683 158.1766 118.7174 160.4183 167.43
85 172.1947 173.8592 177.9143 178.7127 188.6883 188.6883
           dunn             0.1256  0.1401  0.1826  0.1435  0.1574  0.0986  0.1014  0.1038  0.1102  0.1102  0.1172  0.1213  0.12
37  0.1280  0.1313  0.1322  0.1348  0.1359  0.1390
           silhouette       0.4051  0.3282  0.3086  0.1726  0.2131  0.1798  0.1942  0.1890  0.1865  0.1824  0.1879  0.1750  0.18
31  0.1815  0.1804  0.1802  0.1829  0.1884  0.1841
model      connectivity  58.0858 124.2242 111.5156 142.2754 173.1175 169.6796 169.8432 188.8925 180.9849 196.3071 227.4067 228.1687 234.40
52 245.5992 261.6099 264.9643 264.5840 267.4437 262.1655
           dunn             0.0048  0.0742  0.0777  0.1031  0.0504  0.0427  0.0888  0.0981  0.1022  0.0988  0.0894  0.0874  0.06
74  0.0874  0.0781  0.0781  0.0781  0.0889  0.0889
           silhouette       0.1835  0.1195  0.1673  0.1251  0.1027  0.1058  0.1552  0.1452  0.1331  0.1158  0.1222  0.1274  0.11
04  0.1192  0.1079  0.1138  0.1273  0.1324  0.1315
sota       connectivity  15.1067  79.4187  81.2456  98.6099 103.5673 106.2679 148.1464 148.1325 188.5798 172.6111 174.7766 178.4790 178.47
90 185.4390 183.4390 204.1337 204.8421 224.9103 226.0827
           dunn             0.0634  0.0867  0.0894  0.0894  0.0894  0.0894  0.0894  0.0894  0.0894  0.0904  0.0904  0.0894  0.08
94  0.0894  0.0894  0.0894  0.0894  0.1131  0.1131
           silhouette       0.2621  0.1855  0.1941  0.1885  0.1768  0.1619  0.1755  0.1715  0.1716  0.1731  0.1680  0.1652  0.15
73  0.1764  0.1728  0.1848  0.1819  0.1749  0.1748
pam        connectivity  79.7988  89.0621  81.6448 107.5328  99.3294 127.1754 119.7889 167.7095 176.9437 174.0810 190.7340 195.1234 194.46
55 194.0484 196.5601 212.4198 214.8429 223.1796 226.5690
           dunn             0.0581  0.0995  0.1119  0.0867  0.0867  0.0867  0.1126  0.1218  0.0888  0.0888  0.0868  0.0888  0.11
04  0.1304  0.1804  0.1062  0.0987  0.1102  0.1102
           silhouette       0.1570  0.1838  0.2116  0.1749  0.1954  0.2090  0.1834  0.1861  0.1998  0.2052  0.1791  0.1804  0.18
19  0.1865  0.1982  0.1817  0.1743  0.1750  0.1780
clara      connectivity  82.8197  83.6790  90.4311  93.9684 113.2790 146.1270 141.1754 126.0813 171.1056 149.1710 186.3084 181.7976 218.81
44 204.4833 216.5230 229.9087 228.8710 181.1139 206.9849
           dunn             0.0581  0.0621  0.0910  0.0944  0.0838  0.0968  0.0878  0.1139  0.1069  0.1011  0.1316  0.1195  0.11
54  0.1200  0.0827  0.1012  0.1031  0.1453  0.1070
           silhouette       0.1840  0.1881  0.2115  0.1912  0.2059  0.1805  0.2029  0.2070  0.1913  0.2291  0.1710  0.1858  0.16
80  0.1835  0.1584  0.1839  0.1691  0.1904  0.1712
agnes      connectivity  18.7607  18.7607  19.0607  22.7540  36.3115  56.6782  62.2567  63.0012  82.2123  84.5661  90.5234  90.5234 101.60
99 104.9012 107.9440 113.1421 119.7190 121.3286 137.6893
           dunn             0.2600  0.2600  0.2600  0.2600  0.2381  0.2381  0.2468  0.2468  0.2010  0.2010  0.1781  0.1781  0.17
```



Module 5: Hypothesis Testing

I. Lý thuyết

1. Mục tiêu

- Hiểu được các ý tưởng chính đằng sau 1 bài kiểm tra thống kê.
- Biết về các khái niệm true/false – positives/negatives, pvalue cùng tầm quan trọng.
- Có thể áp dụng kiểm tra tham số và phi tham số đơn giản với bộ dữ liệu.
- Biết các diễn giải kết quả.
- Hiểu các vấn đề đằng sau các thử nghiệm.
- Biết phải làm gì với các thử nghiệm trong bối cảnh phân tích dữ liệu biểu thức.
- Có thể thực hiện các thí nghiệm mô phỏng và hoán vị như những lựa chọn thay thế cho các bài kiểm tra.

2. Hypothesis testing là gì

Hypothesis testing hay còn gọi là Kiểm định giả thiết thống kê là một phương pháp giúp chúng ta đưa ra kết luận chung dựa trên số liệu thu thập được.
Trong kiểm định giả thiết thống kê, chúng ta thường áp dụng các câu hỏi như:

- Một mẫu cụ thể có phải là một phần của phân phối hay nó có phải là một ngoại lệ không?
- Có thể lấy hai bộ mẫu từ cùng một phân phối hay chúng đến từ phân phối khác nhau?

1. Một số khái niệm trong Hypothesis testing

Giả thiết Ho- Null Hypothesis
Giả thiết thay thể Ha- Alternative Hypothesis
Trong đó:

- Ho là mệnh đề giả định rằng các giá trị đều bằng nhau hay không có sự khác biệt đáng kể giữa các mẫu.
- Ha là mệnh đề thay thế cho Ho – phủ định Ho(reject Ho), nghĩa là các giá trị không bằng nhau

2. Lỗi trong Hypothesis testing

Truth / Decision	H_0	H_1
Accept H_0	$1 - \alpha$	β "False negative" "Type II error"
Reject H_0	α "False positive" "Type I error"	$1 - \beta$

Trong đó Alpha là một hệ số đặt ra bởi người dùng. Giá trị Alpha = 0.05 nghĩa là chúng ta chấp nhận 5% rủi ro rằng chúng ta đã sai trong việc bác bỏ Ho. Trong vài trường hợp Alpha có thể lớn hơn 0.1

3. P-value là gì

P-value là giá trị mà chúng ta sẽ nhận được sau khi chạy t-test và dùng nó để kết luận. Có thể tính theo công thức sau:

P-value < α	Chúng ta sẽ bác bỏ Ho	Cho nên thừa nhận Ha
P-value > α	Chúng ta đã sai khi bác bỏ Ho	Cho nên phải thừa nhận Ho

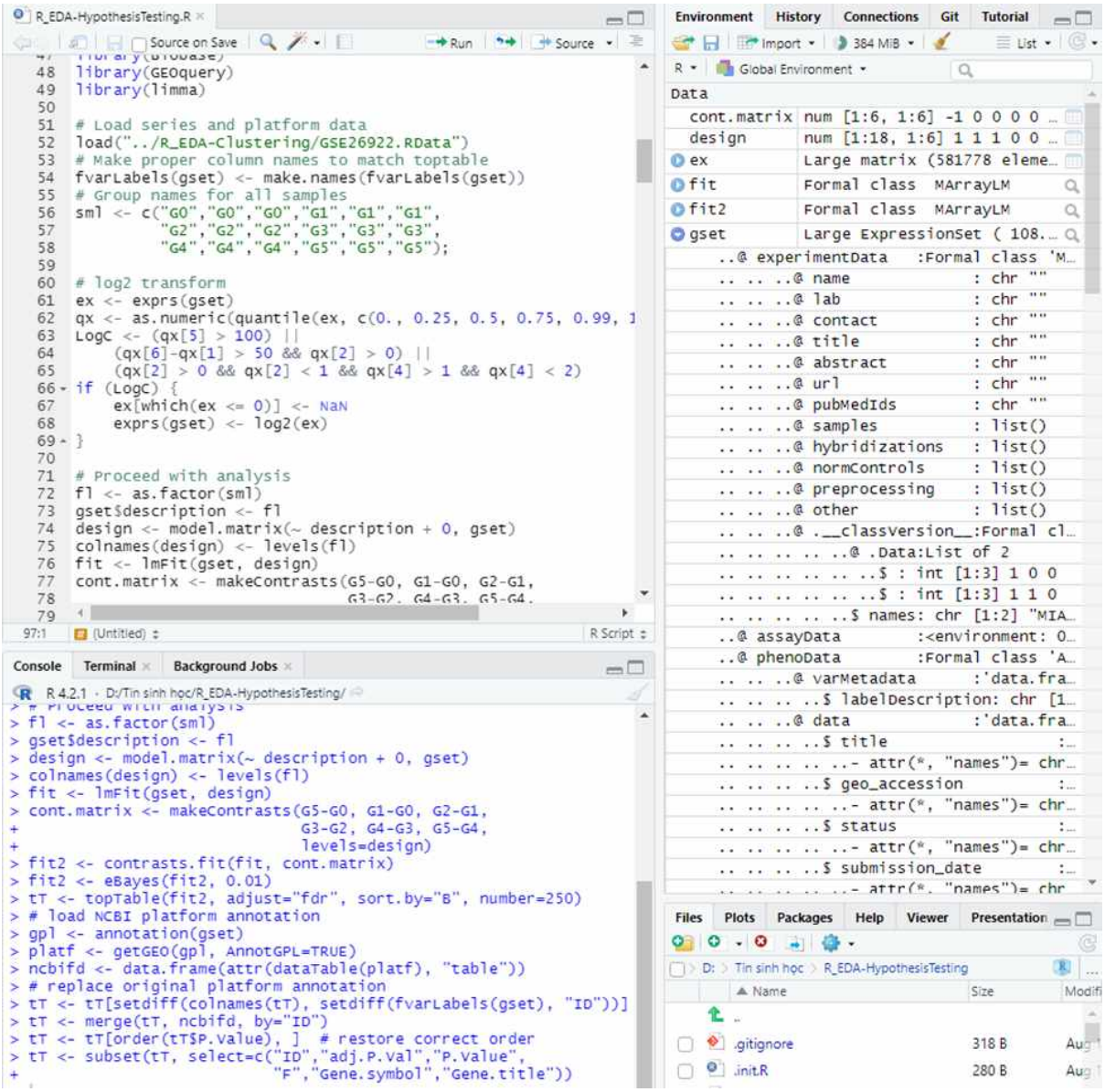
Ví dụ khi tính ra p-value =0.1 trong khi Alpha = 0.05, tức là chúng ta cho phép có 5% khả năng chúng ta sai khi bác bỏ Ho trong khi thực tế chỉ tính ra 1%. Chính vì vậy chúng ta không sai khi bác bỏ Ho hay nói cách khác là phải bác bỏ Ho

II. Thực hành

1. Load tập dữ liệu GSE26922

```
> load("../R_EDA-Clustering/GSE26922.RData")
```

Thực hiện tạo lại bảng và cập nhật các chú thích của NCBI platform



- gene có P.value lớn nhất trong tT

```
> # No. 1 in tT
> tT$ID[1]
[1] "8117594"
```

=>gene có id 8117594

	ID	adj.P.Val	P.Value	F	Gene.symbol	Gene.title
195	8117594	8.715553e-12	4.109444e-16	544.52465	HIST1H2BM	histone cluster 1, H2bm

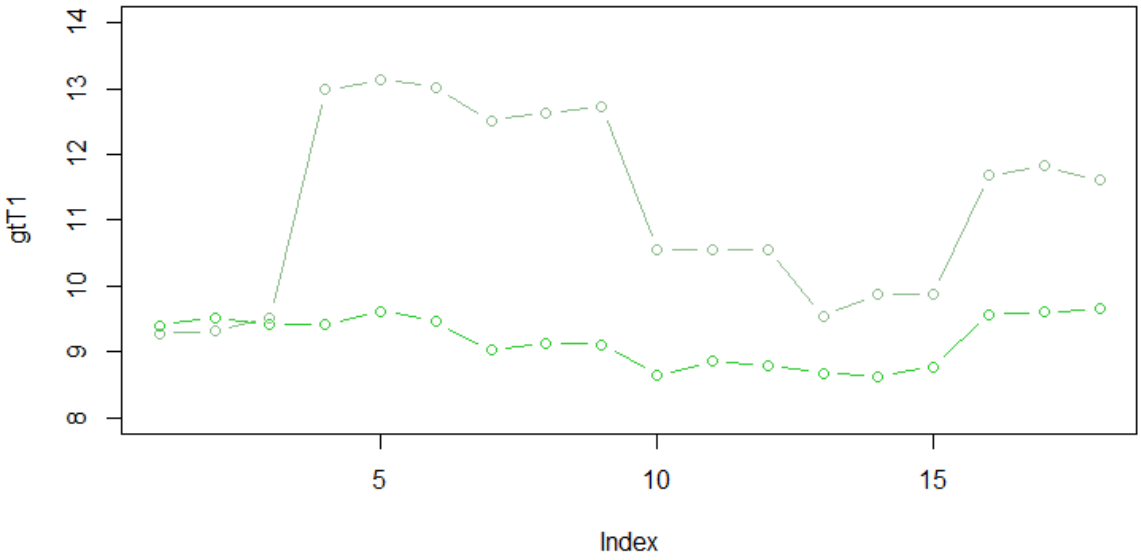
- gene có P.value nhỏ nhất trong tT

```
> # No. 250 in tT
> tT$ID[nrow(tT)]
[1] "8009476"
```

=> gene có id 8009476

vẽ biểu đồ

```
> gtT1 <- ex[ntT1, ]
> gtT250 <- ex[ntT250, ]
> plot(gtT1,    ylim=c(8,14), type="b", col="darkseagreen")
> lines(gtT250,               type="b", col="limegreen")
```



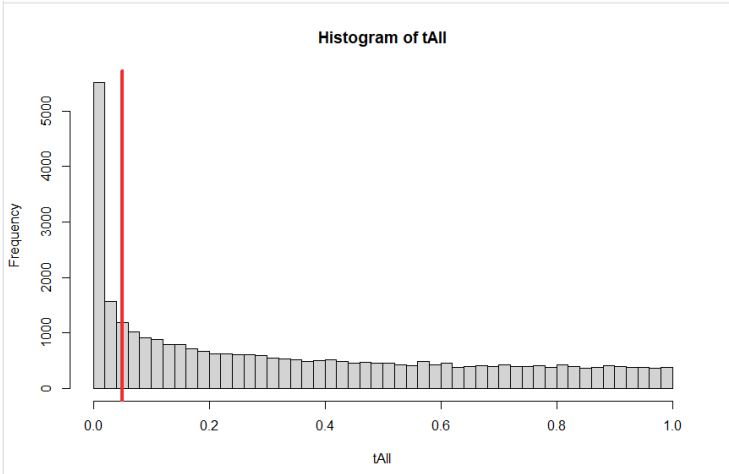
Có thể thấy gen có p.value lớn nhất có sự chênh lệch lớn nhất giữa control value và actual measured value còn gene có p.value nhỏ nhất có sự chênh lệch nhỏ

```
> # values against the T0 values
> g1 <- t.test(ex[ntT1, 1:3], ex[ntT1, 4:18])
> g1

        Welch Two Sample t-test

data:  ex[ntT1, 1:3] and ex[ntT1, 4:18]
t = -6.3838, df = 15.232, p-value = 1.143e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.888416 -1.443850
sample estimates:
mean of x mean of y
 9.370971 11.537104
```

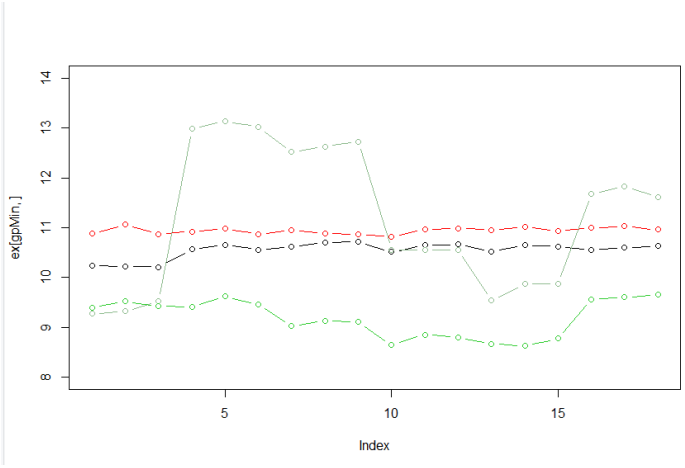
```
> g250 <- t.test(gtT250[1:3], gtT250[4:18])
> g250

        Welch Two Sample t-test

data:  gtT250[1:3] and gtT250[4:18]
t = 2.9441, df = 15.999, p-value = 0.009528
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.08873719 0.54524681
sample estimates:
mean of x mean of y
 9.445678  9.128686
```

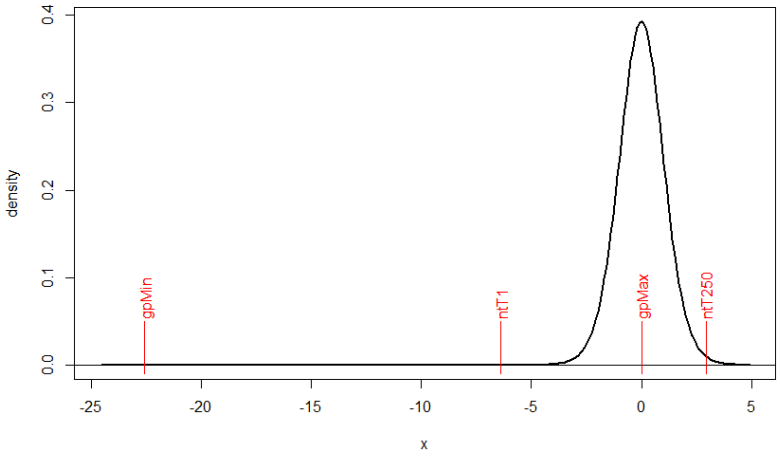
- giá trị của p-value của H0

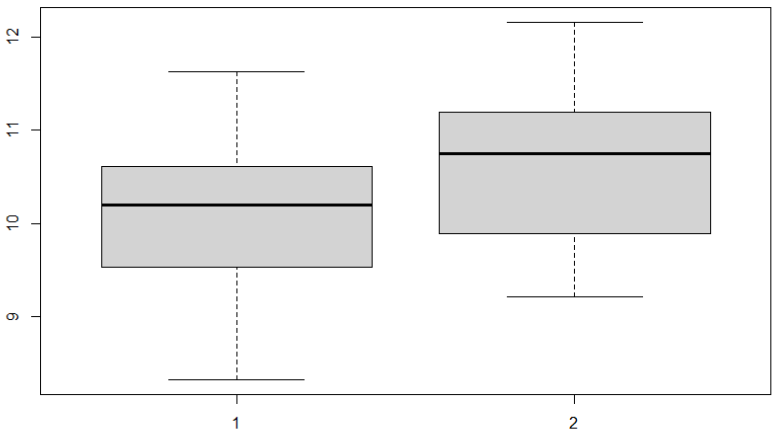
```
> g1$p.value       # this is the p-value of H0
[1] 1.142902e-05
>
```

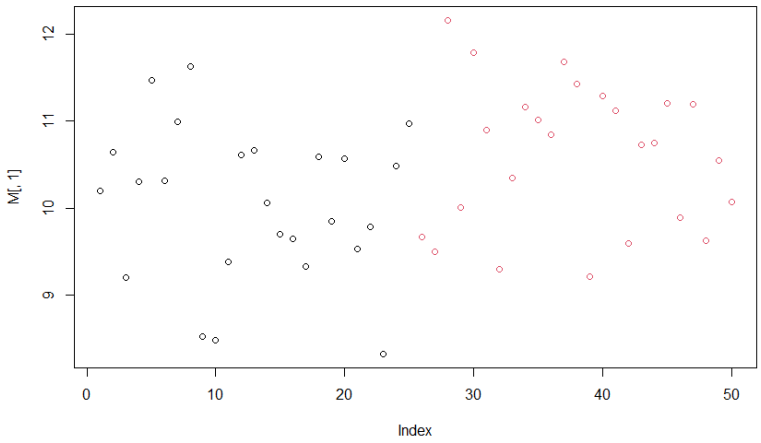


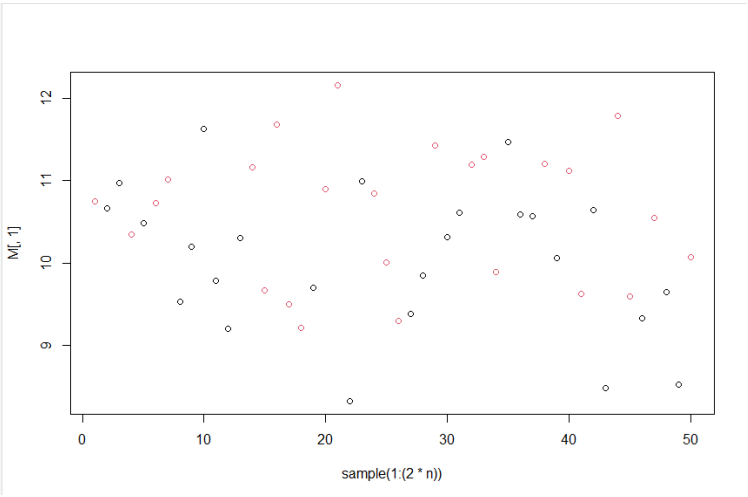

```
> range(tAll)
[1] 2.553457e-13 9.999931e-01
> # How many rows have a p less than 0.05?
> sum(tAll < 0.05)   # Crafty code! Why does this work?
[1] 7716
```

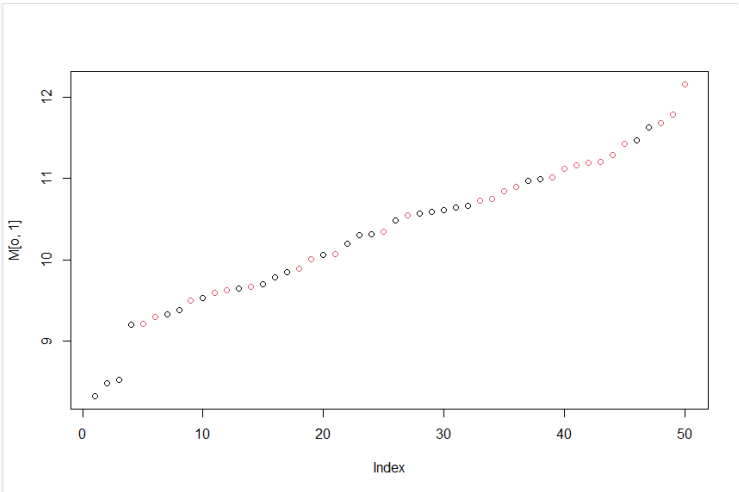

ex[gpMin,]
Index
8
9
10
11
12
13
14
5
10
15

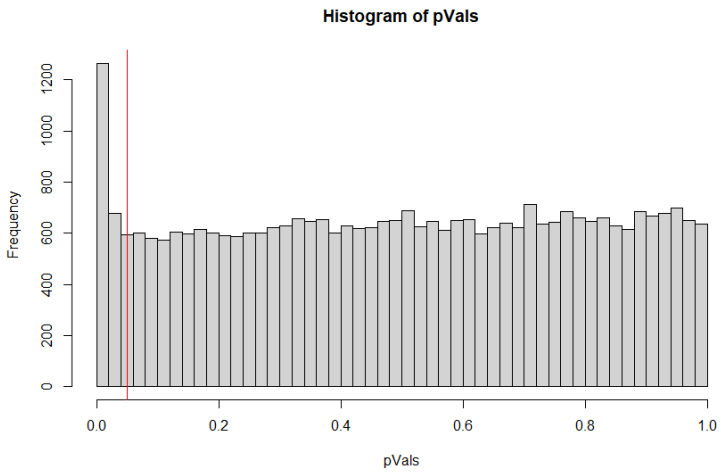

density
x
gpMin
ntT1
gpMax
ntT250
0.0
0.1
0.2
0.3
0.4
-25
-20
-15
-10
-5
0
5


9
10
11
12
1
2

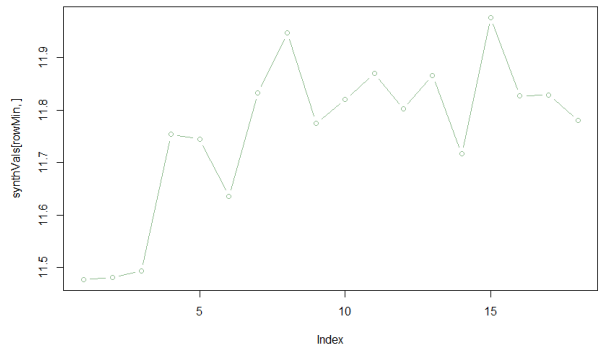

M[, 1]
9
10
11
12
0
10
20
30
40
50
Index


M[, 1]
9
10
11
12
0
10
20
30
40
50
sample(1:(2 * n))


M[o, 1]
9
10
11
12
0
10
20
30
40
50
Index

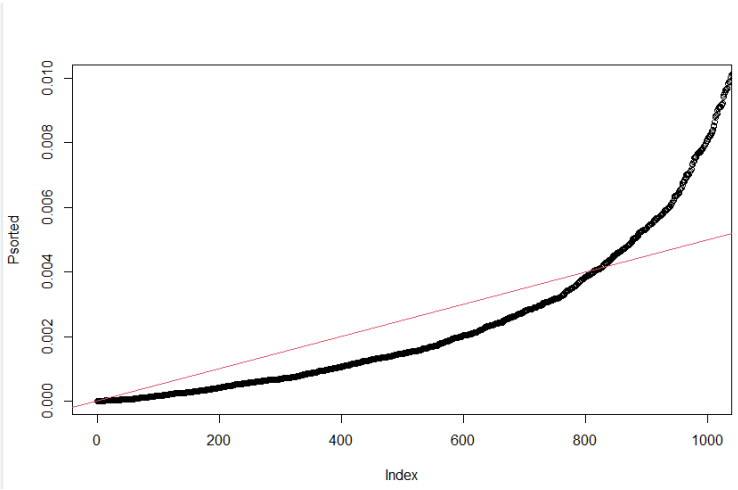



```
> as.numeric(TRUE)  # 1 ... TRUE is coerced into 1
[1] 1
> sum(as.numeric(pVals < 0.05)) # ... so summing
[1] 2242
> # over all TRUE
> # values counts
> # them.
> min(pVals)
[1] 2.846206e-10
```




```
> 0.05 / n  # 1.546982e-06
[1] 1.546982e-06
> # our real data ...
> sum(as.numeric(tAll < 0.05))       # as is
[1] 7716
> sum(as.numeric(tAll < 0.05 / n))   # Bonferroni corrected
[1] 352
> # the synthetic data ...
> sum(as.numeric(pVals < 0.05))      # as is
[1] 2242
> sum(as.numeric(pVals < 0.05 / n))  # Bonferroni corrected
[1] 7
```

```
> # Let's illustrate this with sample data
> set.seed(100)
> N <- 10000
> alpha <- 0.05
> y1 <- matrix(rnorm(9000*4, 0, 1), 9000, 4)
> y2 <- matrix(rnorm(1000*4, 5, 1), 1000, 4)
> y <- rbind(y1, y2)
> myt.test <- function(y){
+     t.test(y, alternative="two.sided")$p.value
+ }
> P <- apply(y, 1, myt.test)
> sum(P<alpha)
[1] 1456
> Psorted <- sort(P)
```




```
> sum(p<0.05)
[1] 3
> p <- p.adjust(P, method="fdr")
> sum(p<0.05)
[1] 825
> # Calculate the true FDR
> sum(p[1:9000]<0.05)/sum(p<0.05)
[1] 0.03757576
```

Tài liệu tham khảo

[1] ***https://bioinformaticsdotca.github.io/EDA_2017***

[2] Silde bài giảng môn tin sinh học- TS.Nguyễn Hồng Quang

[3] Youtube: Bioinformatics DotCa