International Journal of Psychosocial Rehabilitation, Vol. 24, Issue 10, 2020
ISSN: 1475-7192

Using R language to analyze and programming vital data by applying it to a human diseases

[1]MSc Qasim Mahdi Haref, [2]MSc Rafat Talib Hashim, [3]MSc Sara Ali Abdulkareem

Abstract

R is an open-source software platform for statistical data analysis. The R project started in 1993 as a project launched by two New Zealand statisticians, Ross Ilhaka and Robert Gentleman, and their goal was to create a new research platform in statistical computing. Since then, this pilot project has grown to include more than twenty statisticians and computer scientists from all over the world. Because it is an open-source platform, R has been rapidly accredited by statistical departments from universities around the world, and its expansion nature has attracted them as a platform for academic research, and the free platform has also played an important role. And not long ago, statisticians, data scientists, and machine learning began publishing research papers containing R code to implement new work assignments, among most academic journals. Platform R made this process very easy: anyone can post a working package within the platform in the "R Archive Full Network" "CRAN", and it is available to everyone. As of this writing, thousands of R platform users have contributed over 6,100 work packages, extending the capabilities of the platform to fields as diverse as economics, clinical trial analysis, social science, and web data. Anyone can search for applications in MRAN for the topic they want. Many companies and other organizations are working on expanding the R project while maintaining the original essence through the non-profit R Foundation (based in Vienna, Austria). The Bio Conductor has created more than 900 additional work packages, making this project a pioneering programmatic in genetic and genetic data analysis. R Studio has created a great interactive development environment in the R language, boosting user productivity all over the world. Revolution Analytics supported the R project with an open revolution that made it easy to embed it in any other application. In this article, we will try to take a practical example through which we give an overview of data analysis using R.

Keywords: R language, human diseases, programming

I. Introduction

R is a Programming application combined with the semantics of inspiration inspired by another programming language called Scheme [11]. MSI was created by John Chambers at Bell Labs. There are some differences from the C language, but most of the code has not changed [12]. The R language was created by Ross Ihaka and Robert

[1] Department of computer engineering technologies, Imam Khadum College (IKC), Iraq.
[2] Department of computer engineering technologies, Imam Khadum College (IKC), Iraq.
[3] Computer department –presidency of Diyala University- Iraq.

Received: 15 May 2020 | Revised: 5 Jun 2020| Accepted: 20 July 2020

Gentleman at the University of Auckland in New Zealand [13], and the language is currently being developed by the central R development team which includes John Gamblers among its members. The R language was partially named from the names of two of its creators and as a kind of approach with the designation of S [14]. The initial visualization of the project appeared in 1992 with a preliminary version released in 1995 and the first stable beta version in 2000. [15] [16] [17] The R language includes a number of data types that can be distinguished by the class command that gives the data type. Logical types include the true and false variables that are common in other programming languages. Numeric can hold any number whatever its capacity and the presence of any number of decimal places; Integers include any integer; Complex, including real numbers; Symbols (symbols) include all symbols and symbols are stored in their form without regard to the fact of their content such as numbers and signs of addition and subtraction as in other programming languages; Row, and stores symbols in numbers from the hexadecimal count system. The example below shows the use of the CLASS function to get the variable type [18]: R is a programming language and analysis tool developed in 1993 by Robert Gentleman and Ross Ihaka at the University of Auckland, New Zealand. It is widely used by programmers, statisticians, data scientists, and data mining. It is one of the most popular analytics tools used in data analytics and business analytics. It has many applications in areas such as healthcare, academics, consulting, finance, media, and many more. Its wide application in statistics, data visualization, and machine learning have increased the demand for certified professionals in the R language. In this article, we will take as an example data used by researcher Sabina Chiaretti and colleagues to study the properties of gene expression in acute lymphocytic disease (Acute Lymphocytic). Leukemia, or ALL, is a type of leukemia characterized by abnormal proliferation of lymphoblasts in the bone marrow, which are primitive, immature, and undifferentiated blood cells. This disease affects both B-Cells and B-Cells. T-Cells. In this study, the researchers measured the gene expression of 12625 genes using DNA chips (Microarray) for 128 patients newly diagnosed with ALL disease, as a group of them were infected with B cells and a group in T cells.

Upload data and prepare the software environment:

The reason we choose this example is that data is directly available on the Bioconductor site in the ALL package. In addition to this data, we also need to install the following affy and RColorBrewer packages. We can install these packages using the following commands:

```
source("http://bioconductor.org/biocLite.R")
biocLite("affy")
biocLite("ALL")
## RColorBrewer is located in the CRAN
install.packages("RColorBrewer")
library(affy)
library(ALL)

## load the data
data(ALL)
```

We note that the data is stored in an object of the ExpressionSet type specifically designed to store the gene expression data for microchip experiments as it contains the results of each sample and general information about the experiment such as the name of the technology used, the number of genes, information about patients and the published research paper ... etc. We can see some of this information as follows:

```
                                                    ALL <
                    ExpressionSet (storageMode: lockedEnvironment)
                            assayData: 12625 features, 128 samples
                                         element names: exprs
                                             protocolData: none
                                                           phenoData
                        sampleNames: 01005 01010 ... LAL4 (128 total)
                varLabels: cod diagnosis ... date last seen (21 total)
                                        varMetadata: labelDescription
                                                     featureData: none
                         'experimentData: use 'experimentData(object)
                                                   pubMedIds: 1468442
```

For example, if we wanted to know the type of data available on patients, we can use the phenoData and varMetadata functions as follows:

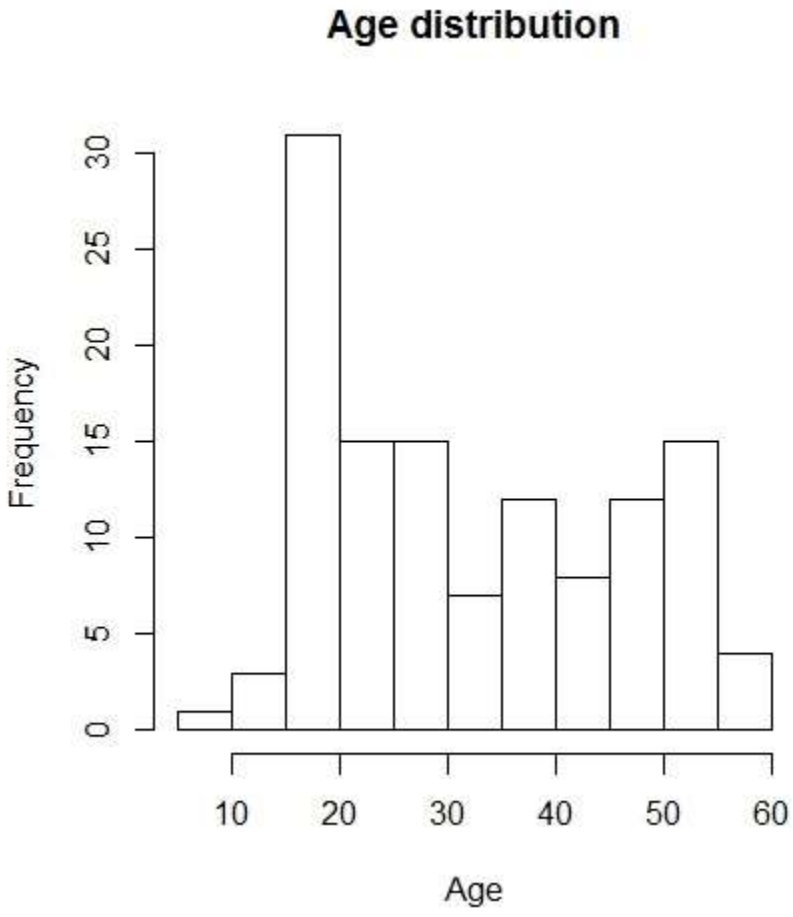
Received: 15 May 2020 | Revised: 5 Jun 2020| Accepted: 20 July 2020

```
                                                          pd <- phenoData(ALL) <
                                                                  varMetadata(pd)
labelDescription
 cod                                                                      Patient ID
 diagnosis                                                           Date of diagnosis
 sex                                                              Gender of the patient
 age                                                         Age of the patient at entry
 BT                                                    does the patient have B-cell or T-cell ALL
 remission      Complete remission(CR), refractory(REF) or NA. Derived from CR
 CR                                                                Original remisson data
 date.cr                                                   Date complete remission if achieved
 t(4;11)        did the patient have t(4;11) translocation. Derived from citog
 t(9;22)        did the patient have t(9;22) translocation. Derived from citog
 cyto.normal                    Was cytogenetic test normal? Derived from citog
 citog         original citogenetics data, deletions or t(4;11), t(9;22) status
 mol.biol                                                           molecular biology
 fusion protein                        which of p190, p210 or p190/210 for bcr/able
 mdr                                                                multi-drug resistant
 .kinet                                                  ploidy: either diploid or hyperd
 ccr                                  Continuous complete remission? Derived from f.u
 relapse                                                        Relapse? Derived from f.u
 transplant     did the patient receive a bone marrow transplant? Derived from f.u
 f.u                                                               follow up data available
 date last seen                                                    date patient was last seen
```

There are two things that are widely used to know how to distribute data, the first is a table and it is used in the case of dealing with factional data (for example: male, female or 1, 2, 3) and it allows to count the number of data from each category. The second is hist. for plotting the repeating table of data and used in the case of continuous or intermittent data. Usually repeating diagrams with a probability density function using the density command. We can use these commands to identify the distribution of the data we have, for example, how many patients are infected in T cells and B cells, the age distribution of patients, etc.

Received: 15 May 2020 | Revised: 5 Jun 2020| Accepted: 20 July 2020

```
BT <- table(ALL$BT) <
BT <
B B1 B2 B3 B4  T T1 T2 T3 T4
2  10 15 1  5  12 23 36 19 5

Total number of B patients ##
sum(BT[1:5]) <
95 [1]
Total number of T patients ##
sum(BT[6:10]) <
33 [1]

number of male and female patients
table(ALL$sex) <
F  M
83 42

hist(ALL$age,xlab="Age",main="Age distribution") <
```



A frequency chart for the distribution of age groups

We notice from the iterative chart that there are two large peaks centered on the age of 18 years and the second smaller near 52 years, which gives us a preliminary idea that there are samples that contain a large proportion of young people and another proportion of the elderly.

Study the expression of some genes:

To simplify the example, suppose we have the following genes known to contribute to apoptosis and we want to see the difference between its activity in patients with T cells and patients with B cells. Where we know that cancer cells are cells that grow in an unnatural way and do not respond to the signals coming from the cellular environment that invites them to stop reproduction or death. Nor does it respond to chemicals that induce her to die. Suppose if we study the following genes:

```
                                                                genes <
   "ABL1"      "Tial1"        "SIVA1"     "foxo3b"     "FOXO3" [1]
  "CDKN1A"    "LOC100271831" "MAPK3"     "ABL1"       "BCLAF1" [6]
   "LOC731605" "Tle1"         "DAP"       "ABL1"       "CUL1" [11]
```

Since we are using microchips, we must first convert the gene names to the corresponding ID in the microchips, we can use R or directly the DAVID site and get the following ID:

```
,"gn.list <- c("36199_at","39020_at","2031_s_at", "39723_at<
                 ,"at","1636_g_at", "39730_at","34740_at_1635" +
                  ("g_at","38050_at","1000_at","1001_at_41763" +
```

For example, we take the genetic expression of these genes in 15 patients with B cells and 15 patients with T cells, as follows:

```
                    gn <- featureNames(ALL) <
                t <- is.element(gn,gn.list) <
  (small.eset <- exprs(ALL)[t,c(81:110)] <
```

Among the diagrams that we can use to take an initial look at the activity of these genes in this group of patients are thermal maps, where we can draw a matrix in which each line represents a gene and each column represents a patient from a patient. And color each cell according to the percentage of activity of the gene. For more information, patients are ranked in the matrix by gene expression. We can do this automatically by using a clustering algorithm. For example, here we use the hierarchical clustering. Before drawing the thermal map, we choose a group of meaningful colors so that we can see clearly as follows:

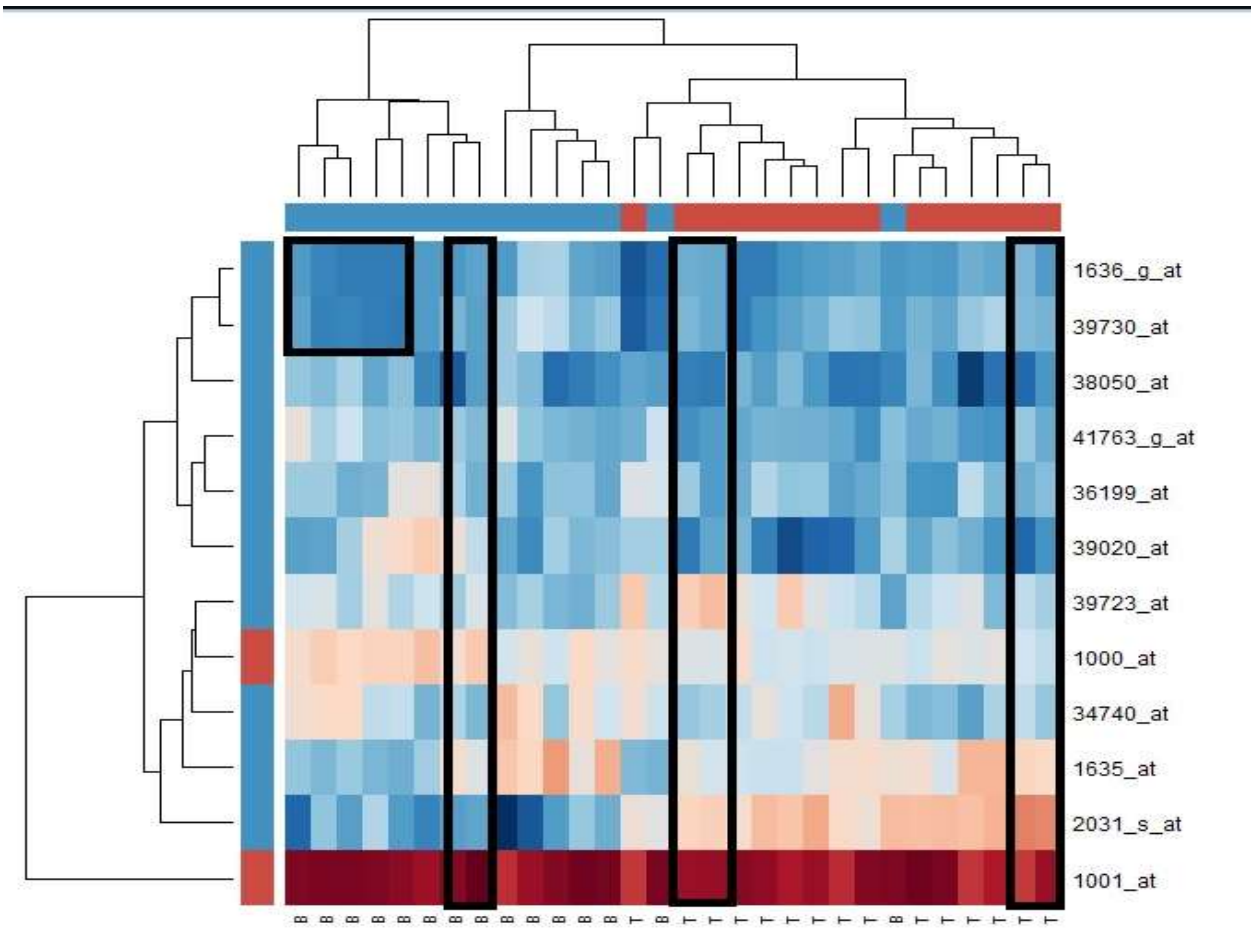
Received: 15 May 2020 | Revised: 5 Jun 2020| Accepted: 20 July 2020

```
                                        library(RColorBrewer) <
hmcol <- colorRampPalette(brewer.pal(10,"RdBu"))(256) <
```

We then rename the patients, denoted with T for patients with T cells and B for patients with B cells.

```
cell <- c(rep('B',15),rep('T',15)) <
      colnames(small.eset) <- cell <
```

Since the arrangement of patients and genes will change in the array after the assembly is performed, we can add two colored columns to the matrix edge. In the column for patients, we color in red, patients in T cells, and blue in patients in B cells. As for genes, we know that all genes have a role in the death of the cell except the first and second genes, so we will color them in red and the rest in blue.



Thermal map showing the expression of the selected genes in the 30 patients

We note that some patients (columns) have a similar genetic expression in the majority of genes, and some are similar in some genes (shown in the black square), but this scheme in this way does not give us enough information, or it may not be easy to read at first sight. We can make an effort and rely on the results of the assembly algorithm shown on the footnotes of the statement where the patients were grouped in the form of a hierarchical scheme, but we can represent in a better way. It would be better, then, if we could compute the extent of similarity between gene expression among patients by using a mathematical method that gives us a value that represents this similarity. If we consider each column as a ray, we can calculate the distance between these rays using classical methods, for example, by calculating the Euclidian distance, which calculates the distance between the two rays $ X $ and $ Y $ as follows:

$$ d = \sqrt {\sum_ {i = 1} ^ {n} (x_i-y_i) ^ 2} $$

Correlation coefficient can also be calculated (usually results are better). Among the frequently used correlation coefficients is the Spearman correlation coefficient as follows:

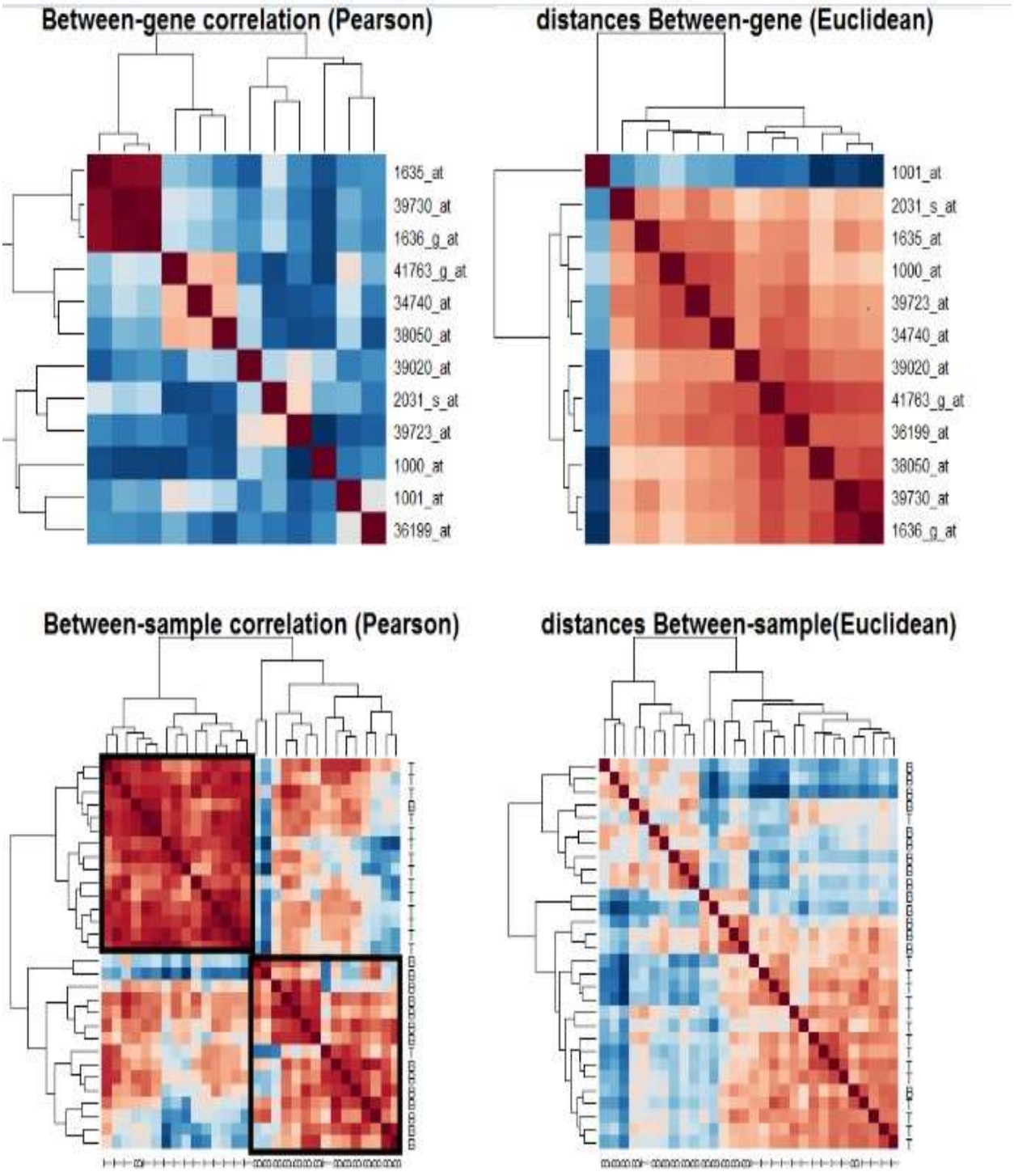
$$ \rho = \frac {\sum_i (x_i- \bar {x}) (y_i- \bar {y})} {\sqrt {\sum_i (x_i- \bar {x}) ^ 2 \sum_i (y_i- \bar {y}) ^ 2}} $$

We can write a function to calculate these parameters, or directly use the cor.dist function to calculate the Spearman correlation coefficient and the euc function to calculate the Euclidean distance. These functions are available in the bioDist package on the Bioconducor website. In the following example, we will calculate the values of gene expression similarities in all genes in all patients (or in other words, the similarity between the lines of the thermal map in the first example) using both methods (correlation coefficient and distance) and calculate the similarity of gene expression between patients in all genes (columns). This is as follows:

```
                                    biocLite("bioDist") <
                          d.gene.cor <- cor.dist(small.eset) <
,heatmap(as.matrix(d.gene.cor),sym=TRUE,col=hmcol <
                ,'main='Between-gene correlation (Pearson) +
                          (xlab='probe set id',labCol=NA +

                          d.gene.euc <- euc(small.eset) <
,heatmap(as.matrix(d.gene.euc),sym=TRUE,col=hmcol <
                ,'main='distances Between-gene (Euclidean) +
                          (xlab='probe set id',labCol=NA +

                       d.sample.cor <- cor.dist(t(small.eset)) <
,heatmap(as.matrix(d.sample.cor),sym=TRUE,col=hmcol <
              ,'main='Between-sample correlation (Pearson) +
                                    ('xlab='cell type +

                       d.sample.euc <- euc(t(small.eset)) <
,heatmap(as.matrix(d.sample.euc),sym=TRUE,col=hmcol <
                ,'main='distances Between-sample(Euclidean)+
                                    ('xlab='cell type +
```

Received: 15 May 2020 | Revised: 5 Jun 2020| Accepted: 20 July 2020

We can compile these heat maps in an image for a more complete look



Thermo graph represents the ratio of similarity of gene expression in patients or between genes using different methods

II. Conclusion

We note that using the correlation coefficient to calculate the similarity between genes and between patients gives us a better look. For example, we notice that we can divide patients into two groups (represented by the black squares) on the right, which contain patients in B cells, but a patient in T cells permeates them. In the box on the left, we see a grouping of patients in T cells interspersed with an infected patient in B cells. We can conclude that the gene expression changes according to the diseased cells. As for the two patients who were not classified by their peers, there are two possibilities. Either there were some errors in the experiment or perhaps the result is really correct. For example, the researcher can take these two samples and study more closely to understand the reason for their similarity with the other class, or do the experiment again for these two patients and make sure. We hope that this amount of examples will help the reader to get some sort of idea on how to use the R language to analyze data even if the example is not comprehensive (for example, how do you choose genes, what statistical tools can be used, etc.).

References

1. Karl Rexer, Heather Allen, & Paul Gearan (2011); 2011 Data Miner Survey Summary, presented at Predictive Analytics World, Oct. 2011.
2. Wrathematics (27 August 2011). "How Much of R Is Written in R". librestats. Archived from the original on 12 June 2018. Retrieved 7 August 2018.
3. Eddelbuettel, Dirk; Francois, Romain (2011). "Rcpp: Seamless R and C++ Integration". Journal of Statistical Software. 40 (8). doi:10.18637/jss.v040.i08.
4. Luo J, Wu M, Gopukumar D, Zhao Y. Big data application in biomedical research and health care: a literature review. Biomed InformInsights 2016;8:1.
5. Gaitanou P, Garoufallou E, Balatsoukas P. The effectiveness of big data in health care: a systematic review. In: Metadata and semanticsresearch. 2014:141–53.
6. Lillo-Castellano JM, Mora-Jimenez I, Santiago-Mozos R, Chavarria-Asso F, Cano-González A, García-Alberola A, et al. Symmetrical com-pression distance for arrhythmia discrimination in cloud-based big-data services. IEEE J Biomed Health Inform 2015;19:1253–63.
7. Andreu-Perez J, Poon CC, Merrifield RD, Wong ST, Yang GZ. Big data for health. IEEE J Biomed Health Inform 2015;19:1193–1208.
8. Archenaa J, Anita EM. A survey of big data analytics in healthcare and government. Procedia Comput Sci 2015;50:408–13.
9. Borne K. Top 10 big data challenges–a serious look at 10 big data V's. MAPR, 2014:NO4, 80. (11) (PDF) Big Data Analytics in Medicine and Healthcare. Available from: https://www.researchgate.net/publication/325076139_Big_Data_Analytics_in_Medicine_and_Healthcare [accessed May 20 2020].
10. Muenchen, Robert (19 June 2017). "The Popularity of Data Science Software". Retrieved 21 November 2018.

Received: 15 May 2020 | Revised: 5 Jun 2020| Accepted: 20 July 2020