USC Department Of Translational Genomics
Keck Medicine of USC

TRGN 527: Applied Data Science and Bioinformatics

UNIT I. Introduction and Basic Data Science

Week 1 – Lecture 3

Enrique I. Velazquez Villarreal, M.D., Ph.D., M.P.H., M.S. | **Assistant Professor**

Dept. of Translational Genomics

USC | Keck School of Medicine | Norris Comprehensive Cancer Center

Leader of the USC Bioinformatics Core – *USC CaRE2 Health Equity Center*

David W. Craig, Ph.D. | **Professor and Vice Chair**

Dept. of Translational Genomics

USC | Keck School of Medicine | Norris Comprehensive Cancer Center

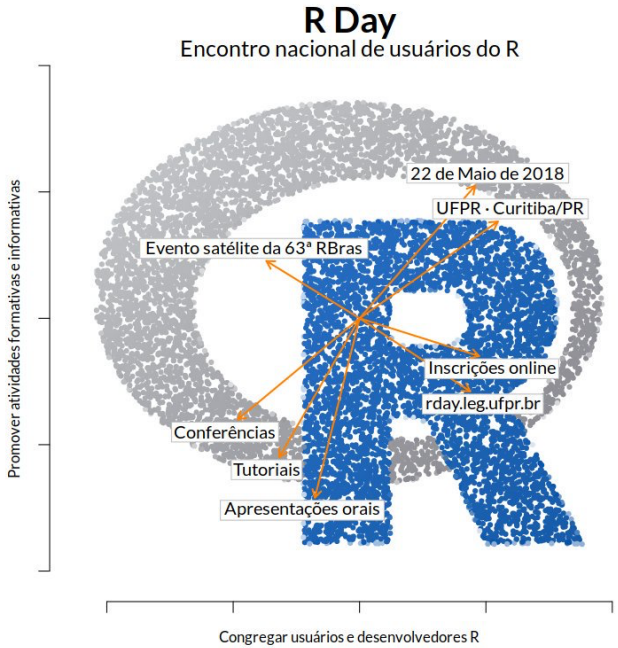
Co-Director, Institute of Translational Genomics

Topics

- Data aggregation and pivoting examples. Joining of tables. Loading in data, data frames, Data types (numerical, categorical, ordinal).

Input and Output

- Statistical analyses begin with data
- Most data is stuck inside files and databases
- Statistical analyses ends with reports.
- Basic functions in R for input and output are:
  - read.csv
  - read.table
  - print
  - cat
  - format



Basic Syntax in R

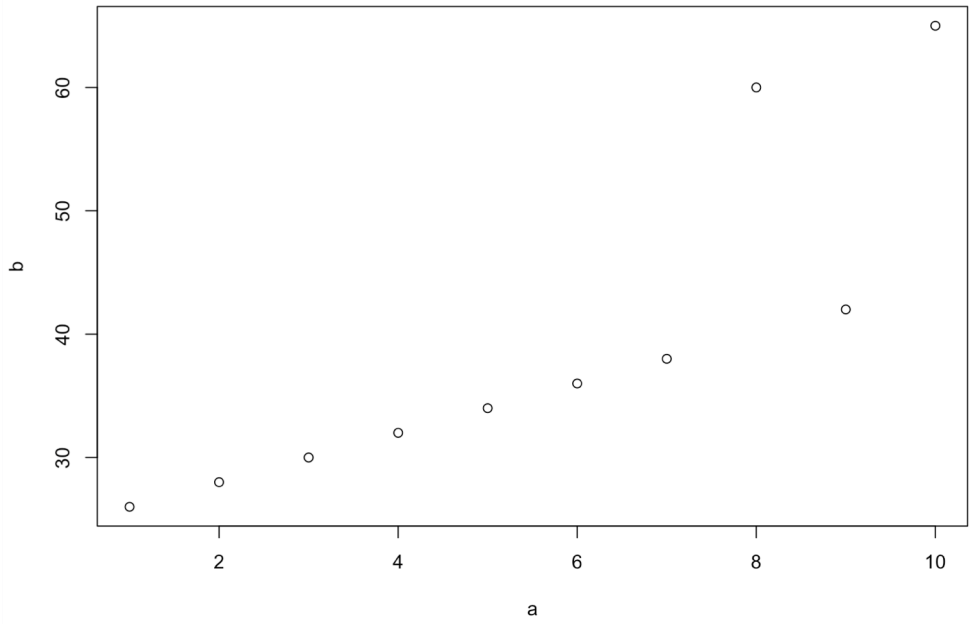
- First you should check that R and RStudio are installed on your system.
- Open Rstudio.

```
63  # Basic operations in R:
64
65  x <- 20
66  y <- 26
67
68  x+y
69
70  z <- 5*x
71  z
72
73  a <- 365
74  a
75
76  ls()
77
78  rm(a,z)
```

Basic Plot

- Assign the numbers 1-10 to variable “a” and 26,28,30,32,34,36,38,60,42,65 to variable “b”.
- Print the number of values assigned to a variable.
- Create a plot.

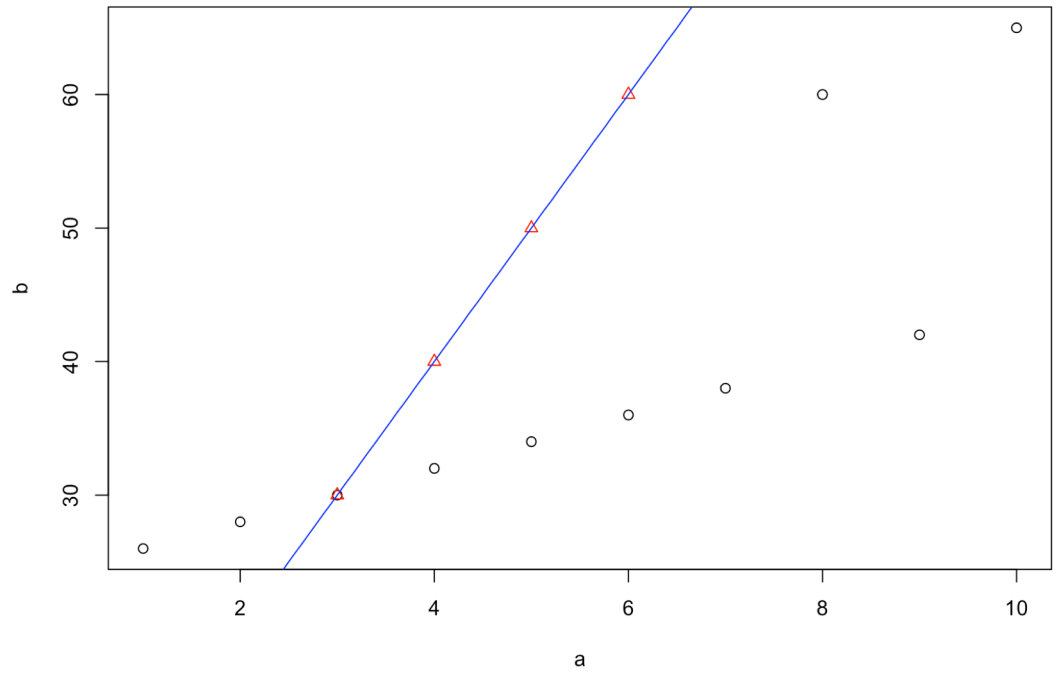
```
101  # Basic Plot
102
103  a <- 1:10
104  a
105  length(a)
106
107  b <- c(26,28,30,32,34,36,38,60,42,65)
108  length(b)
109
110  plot(a,b)
111  c <- seq(from=10, to=100, by=10)
112  c
```



Basic Plot

- Assign the numbers 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100 to vector c.
- Use function seq() to define series of points.
- Add the points to the plot
- Specify color and plotting character

```
101  # Basic Plot
102
103  a <- 1:10
104  a
105  length(a)
106
107  b <- c(26,28,30,32,34,36,38,60,42,65)
108  length(b)
109
110  plot(a,b)
111  c <- seq(from=10, to=100, by=10)
112  c
113
114  points(a, c, col="red", pch=2)
115  lines(a, c, col="blue")
```



Vector Elements

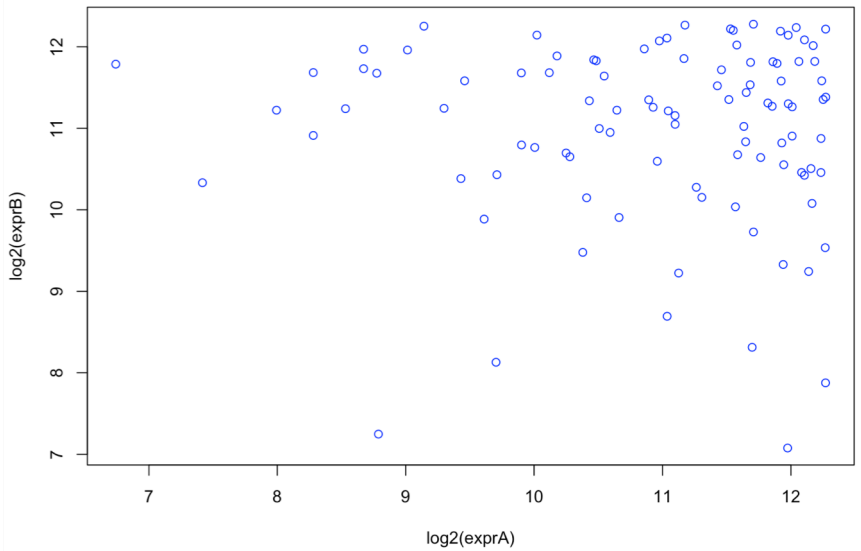
- Retrieve individual vector elements
  - Refer to vector element 5
  - Ask which vector element has a value smaller or equal to 50
  - Check for either the value or index of vector of vector elements that match the condition.

```
119  # Retrieve Individual Vector Elements
120
121  b
122
123  b[5]
124
125  b<=50
126
127  b[b<=50]
128
129  which(b<=50)
```

Reading Data Files

- Loading data file into R.
  - First, look at the content of the data “gene_expression_table.csv” (download the file from BB)
    - The file contains gene expression data.
  - The file consist of three columns and one header row.
  - Load the content of file “gene_expression_table.csv” into the variable data.
    - The path is skipped because the file sits in the same folder from where R is called.
    - The argument header is set to TRUE, because the first line in the data file contains the column names.
    - As file separator, the tabulator is set with the argument sep.
  - Make these names available to the system with attach() command.
  - Create a plot of the data.

```
136  # Importing data
137
138  getwd()
139
140  setwd("/Users/enriquevelazquez/Documents/R_working_directory")
141
142  data <- read.csv("gene_expression_example.csv", header = TRUE, sep=",")
143
144  names(data)
145
146  attach(data)
147
148  plot(log2(exprA), log2(exprB), col="blue")
```



Writing Data Files

- Write data into a file.
  - Use arguments for write.table() function.

```
152 # Writing into files
153
154 write.table(data, "output-data.csv", append=FALSE, col.names=TRUE,
155           row.names = FALSE, quote = FALSE, sep = ",")
156
157 system("head -3 output-data.csv")
158
159 write.table(data, "output-data.csv", append=FALSE, col.names=TRUE,
160             row.names = FALSE, quote = TRUE, sep = ",")
161
162 system("head -3 output-data.csv")
163
164 write.table(data, "output-data.csv", append=FALSE, col.names=TRUE,
165             row.names = TRUE, quote = TRUE, sep = ",")
166
167 system("head -3 output-data.csv")
168
169 write.table(data, "output-data.csv", append=FALSE, col.names=FALSE,
170             row.names = FALSE, quote = FALSE, sep = ",")
171
172 system("head -3 output-data.csv")
173
174 write.table(exprA, "output-data.csv", append=FALSE, col.names=FALSE,
175             row.names = FALSE, quote = FALSE, sep = ",")
176
177 system("head -3 output-data.csv")
178
179 write.table(list(exprA,exprB), "output-data.csv", append=FALSE, col.names=FALSE,
180             row.names = FALSE, quote = FALSE, sep = ",")
181
182 system("head -3 output-data.csv")
```

Data Types

- A variety of data types are available in R:
  - Vectors (numerical, character, logical)
  - Matrices
  - Lists
  - Data frames

Vectors

- Everything in R is an object

```
265  # Vectors
266
267  t <- c(2,5,7.4,9,-1,3) # numeric vector
268  r <- c("t","c","g", "a") # character vector
269  g <- c(TRUE,FALSE,TRUE,TRUE,TRUE,FALSE) #logical vector
```

Matrices

- Matrices are an extension of numeric/character vectors.

```
272
273 #Matrices
274
275 TRGNData <- c(2.1, 2.2, 2.3, 2.4, 2.5, 2.6) # numeric vector
276 mat <- matrix(TRGNData, 2, 3) # creating a matrix
277 mat
278
279 mat2 <- matrix(nrow = 2, ncol = 2) # creating another matrix
280 mat2
281
282 dim(mat2)  # checking the dimensions of the matrix
283
284 mat2 <- matrix(c(1:3))
285 class(mat2)       # checking that the matrix has vectors with a class attribute
286
287 mat2 <- matrix(c(1:3))
288 class(mat2)
289 typeof(mat2)    # shows that the matrix is an integer vector
290
291 # Other ways to construct a matrix
292
293 mat3 <- matrix(1:6, nrow = 2, ncol = 3)
294
295 mat4 <- 1:10
296 dim(mat4) <- c(2, 5)
297
298 # Another way is to bind columns or rows using cbind() and rbind().
299
300 x <- 1:3
301 y <- 10:12
302 cbind(x, y)
303
304 rbind(x, y)
305
306 # Using by-row argument to specify how the matrix is filled
307
308 mat5 <- matrix(c(1, 2, 3, 11, 12, 13),
309                nrow = 2,
310                ncol = 3,
311                byrow = TRUE)
312 mat5
```

Lists

- DEFINITION: Lists are arrays with mixed data types.
- Use the function read.csv() to import data
  - This means that although the first column contains text type data, it is allowable to perform calculation with the data in column two.

```
186
187  # Lists
188
189  data <- read.csv("gene_expression_example.csv", header = TRUE, sep=",")
190
191  data[1,]
192
193  data[1,2]+1
194
```

Lists

- Manually create a list of elements that can be retrieved.
  - xray[[1]] identifies the first vector of the list.
  - xray[[1]][2] identifies the second element of the first vector.
  - As a note, xray[1,2] does not work for list, but only for arrays.

```
195 ##########
196
197 # Creating lists
198
199 xray <- list(ID=c("PTEN", "TERT", "CDK2NA"), Score=c(0.26, 0.20, 0.30))
200
201 xray
202
203 xray[1]
204
205 xray[[1]]
206
207 xray$ID[2]
208
209 xray[[1]][2]
210
211 names(xray)
212
213 attach(xray)
214
215 ID
216
217 ID[1]
218
```

Data Frames

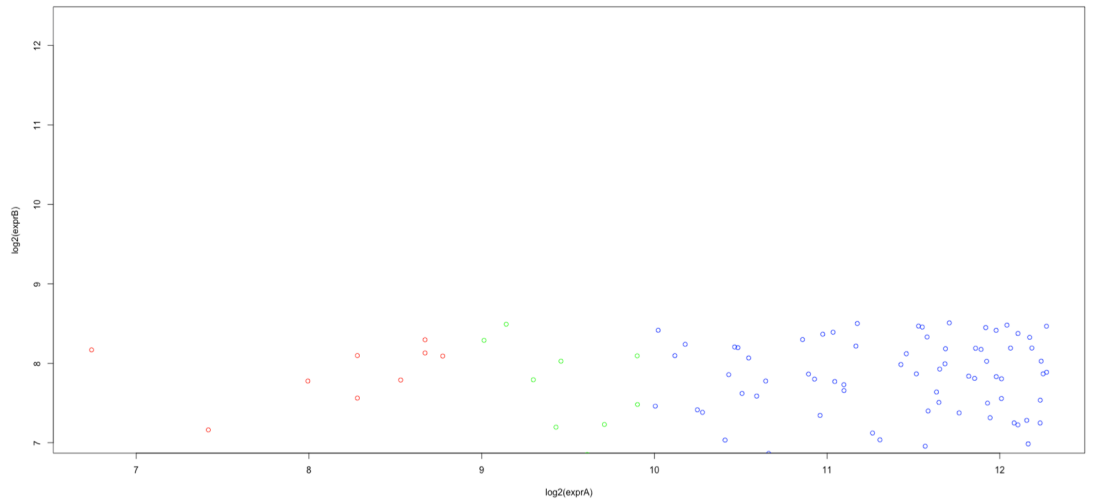
- DEFINITION: Data frame is a list with named entries.
- When we transfer a list to a data frame, the display of the data looks nicer.
- A data frame resembles a table, whereas a list consists of a collection of vectors.

```
221
222  # Data Frames
223
224  xray
225
226  data.frame(xray)
227
```

Programming Structures

- R is a full-fledged programming language.
- It was designed to write complex programs.
- Here is a demonstration of using two fundamental functions: loops and conditionals
- The basic structure of a loop is: *for(index in vector) {…}*
- Below is an example where a *for* construct loops through all element indices of vector exprA, where the expression value fulfills a cerntain condition.

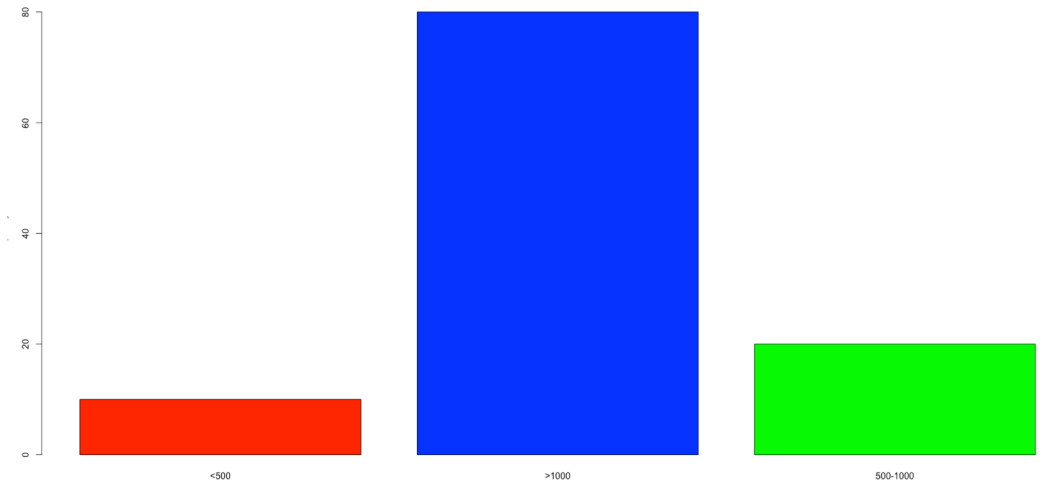
```
240
241  # Composite plots
242
243  plot(log2(exprA), log2(exprB), col= "blue", type = "n")
244  for(i in which(exprA<500)) {points(log2(exprA[i]),log(exprB[i]), col="red")}
245  for(i in which(exprA>1000)) {points(log2(exprA[i]),log(exprB[i]), col="blue")}
246  for(i in which(exprA>=500 & exprA <=1000)) {points(log2(exprA[i]),log(exprB[i]), col="green")}
247
```



Programming Structures

- Write a complex one liner in order to count the number of genes with expression values of a particular range.
- Use the *if* function.

```
250
251  # Composite plots 2
252
253  a = 0; b = 0; c = 0;
254  for(i in exprA) {
255    if(i<500) {a=a+1};
256    if(i>1000){b=b+1}else{c=c+1}
257  };
258  barplot(c(a,b,c),
259          names=c("<500",">1000","500-1000"),
260          col=c("red", "blue", "green"),
261          yla="Frequency")
262
```



Entering Data from Keyboard

- Exercise
  - If you have a small amount of data, too small to justify the overhead of creating an input file, how can you enter data directly into your workspace?
  - Build three vectors with the names of the: 1) Essential Amino acids, 2) Conditionally essential amino acids and 3) Non-essential amino acids.

Entering Data from Keyboard

- Answer:

```
316
317 # Exercise 1
318
319 essential_aa <- c("Histidine", "Isoleucine", "Leucine", "Lysine", "Methionine", "Phenylalanine", "Threonine", "Tryptophan", "Valine")
320 cond_essential_aa <- c("Arginine", "Cysteine", "Glutamine", "Glycine", "Proline", "Tyrosine")
321 non_essential_aa <- c("Alanine", "Aspartic_acid", "Asparagine", "Glutamic_acid", "Serine", "Selenocysteine", "Pyrrolysine")
322
```

R Markdown

file:///Users/enriquevelazquez/Documents/Enrique/TRGN_599_App_DataScience_Bioinf/TRGN_599_Rmarkdow

TRGN599_Week_1_Lecture_3

EIVV

Getting Started

Basic operations in R:

```
x <- 20
y <- 26

x+y
```

```
## [1] 46
```

```
z <- 5*x
z
```

```
## [1] 100
```

```
a <- 365
a
```

```
## [1] 365
```

```
ls()
```

```
## [1] "a" "x" "y" "z"
```

```
rm(a,z)
```

Basic Plot

```
a <- 1:10
a
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
length(a)
```

```
## [1] 10
```

```
b <- c(26,28,30,32,34,36,38,60,42,65)
length(b)
```