Lab 1 Human Genome Analysis 2020 : Introduction to R

Learning objectives

- Become aware of the field of computational advanced genetics
- Install and run R packages on your computer
- Use KnitR to generate a html file of your lab report
- Basic computation in R
- How to create and compute with vectors
- Simple Graphs

Overview

In recent years, the field of genomic analysis has sifted towards requiring some knowledge of R, Python/Perl/C and the use of high performance computers (often requiring some fundamental Unix skills) available at national computing centers for working with large data sets. While there are many great software packages available for particular computational problems in evolutionary biology, many software programs do not have a user interface (e.g. drop down menus and such) and are run in command line mode. The lab sessions in this course have been designed to give students an introduction to working with R code in addition to learning some standalone software packages.

Computer Operating Systems

Everything we do in this course can be done in Windows, Apple's OS X, Linux and Unix and on most laptop computers sold in the past few years. One of the goals of this course is for you to be able to set up an environment to program and run R on your own computer or the computers in your research laboratory.

Open Source Software

The Open Source movement treats program source code in a similar manner to the way scientists publish their results: publicly and open to unfettered examination and discussion. Examples include:

- Linux operating system
- Firefox web browser
- Python
- R
- BioPerl, BioJava, BioPython
- EMBOSS, Bioconductor, Cytoscape, and many of the programs we will use in bioinformatics.

We can also look at the code to see how they solved they problem, what algorithms they used and even use the code in our programs as long as we properly acknowledge the source.

And because… * Open Science is Kinder Science (https://www.nceas.ucsb.edu/news/open-science-kinder-science)

R

R is the largest and most comprehensive public domain statistical computing environment. The core R package is enhanced by several hundred user-supplied add-on packages, including many for gene expression analysis, in the Comprehensive R Archive Network (CRAN) (http://cran.r-project.org/). Omegahat Project for Statistical Computing. BioConductor (http://www.bioconductor.org/) is an open source and open development software project for the analysis and comprehension of genomic data and is based primarily on the R programming language. R and Bioconductor are free, Open Source and available for Windows, MacOS and a wide variety of UNIX platforms.

R manuals, help and tutorials

Many introductory and advance tutorials have been developed for R. Here are a few

- The offical R manuals (http://cran.r-project.org/manuals.html#R-admin)
- CRAN's Introduction to R (http://cran.r-project.org/doc/manuals/R-intro.html)
- R for Data Science (https://r4ds.had.co.nz/) by Garrett Grolemund and Hadley Wickham
- R Graphics Cookbook (http://www.cookbook-r.com/Graphs/) by Winston Chang
- Data Carpentries Genomic Workshop Sessions (https://github.com/datacarpentry/genomics-workshop/)
- Data Analysis and Visualization in R for Ecologists (https://datacarpentry.org/R-ecology-lesson

/index.html)

There are also many workshops and online R courses that you could take to follow up what you learn in this class.

On the Computer

Installing R

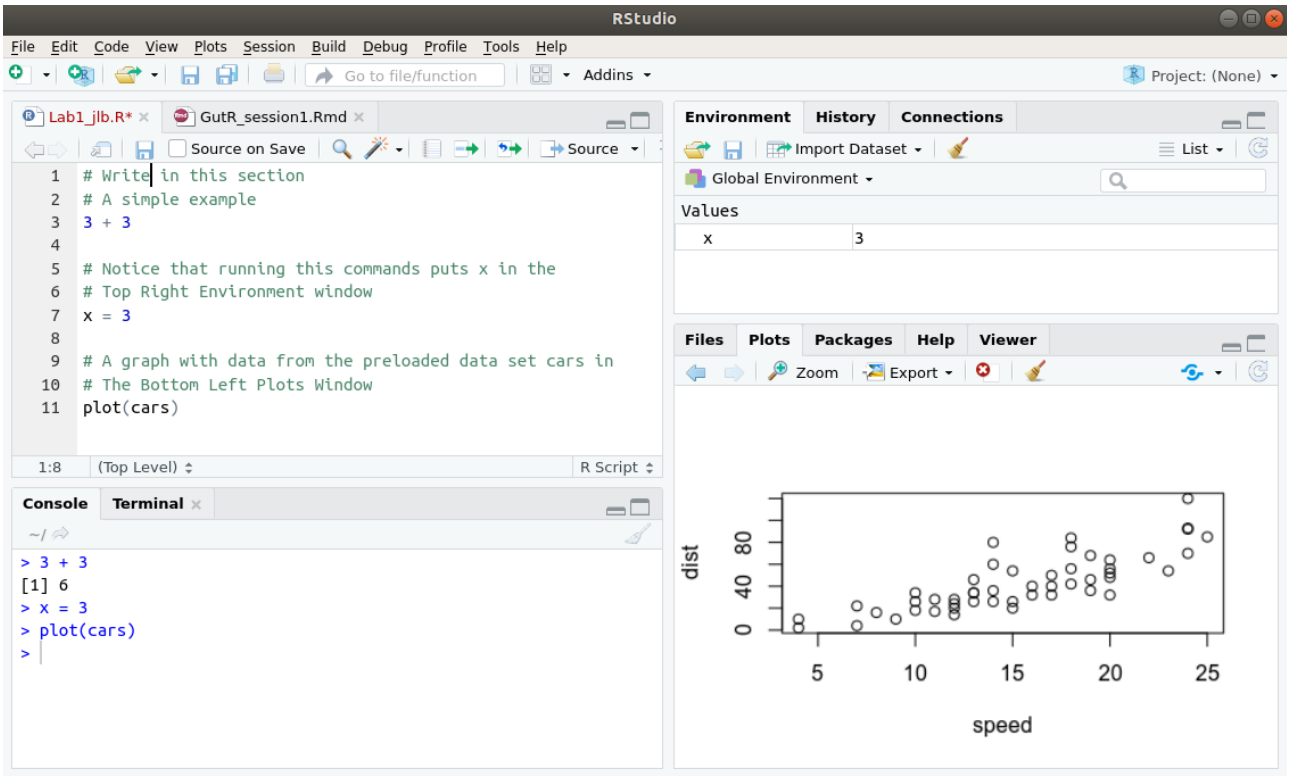
1. Download the proper distribution of R from CRAN (http://cran.r-project.org/) and follow the installation instructions.
2. Install R Studio (http://www.rstudio.org/), a nice graphical interface for working with R.

Working in RStudio

1. Open RStudio
2. Click on File > New File > R Script
3. Save this file in a new folder named HumanGenomics as lab1_yourname.R

The default R studio appearance includes 4 windows.

1. The R script(s) and data view.
2. Console.
3. Workspace and history.
4. Files, plots, packages and help.



Basic Layout in RStudio

The R script(s) and data view window (upper left window)

You should keep good notes as in a lab notebook and keep track of your R commands that you run so that you can or others can repeat the analyses. This is done in a file that is usually save with the .R extension (e.g. mylab1_notes.R). The hash tag # signifies notes and is not run as R commands. To create a new R script you can either go to File -> New -> R Script. You will also create a R markdown file that will generate a lab report for you to load into Moodle, but more on that below.

In this window you can type R commands into this file and run them. Just leave the cursor anywhere on the line where the command is and press Ctrl-R or click on the 'Run' icon above. Output will appear in the Console below.

Type

3 + 3

into the file and click run. You should see the result in the Console Window.

Console Window (bottom left window)

The console is where you can type R commands and see output, however many people prefer to type R commands into the R script file so that they are save and can easily be modified and rerun. If you do run commands in the Script window, the output appears in the Console Window

Workspace and History tabs (upper right window)

The workspace tab shows all the active objects. The history tab shows a list of commands used so far.

Files, Plots, Packages and Help (bottom right window)

There are data sets that come with the R package and used in tutorials. If you run the following command you will see a graph of related to the cars data set in the Plots window

Basic computation

R can be used as an ordinary calculator. Here are a few examples:

```
3 * 3
```

```
## [1] 9
```

Note order of operations

```
3 + 3 / 3
```

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

Where

```
(3 + 3) / 3
```

```
## [1] 2
```

Here are a few other simple computations

```
    log (10)       # Natural logarithm with base e=2.718282
```

```
## [1] 2.302585
```

```
    exp(2)         # exponential function
```

```
## [1] 7.389056
```

```
    3^3            # 3 raised to the third power
```

```
## [1] 27
```

```
    sqrt (9)       # Square root
```

```
## [1] 3
```

```
    abs (1-7)      # Absolute value of 1-7
```

```
## [1] 6
```

R data types

There are 5 basic data types in R

- Numeric
- Integer
- Logical
- Character
- Complex

Numerics

Decimal values are called numerics in R. It is the default computational data type. Note that R is case sensitive, e.g., object names gene, GENE, Gene are all different. To find out what valule an object holds, simply type its name.

```
x = 3.5
```

Simply typing x will give the value of x

```
x
```

```
## [1] 3.5
```

or you can compute with x

```
sqrt(x)
```

```
## [1] 1.870829
```

Integers

In order to create an integer variable in R, invoke the as.integer function

```
x = 3.33
y = as.integer(x)
y
```

```
## [1] 3
```

Logical

A logical value is created by comparison between variables.

```
x = 1; y = 2
z = x > y
z
```

```
## [1] FALSE
```

Standard logical operations are “&” (and), “|” (or), and “!” (negation).

```
x = TRUE; y = FALSE
x & y
```

```
## [1] FALSE
```

```
x | y
```

```
## [1] TRUE
```

```
!x
```

```
## [1] FALSE
```

Character

A character object is used to represent string values in R. It is defined by double quotes "".

```
x = "ATGAAA"
y = "TTTTGA"
x
```

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

but you can't do standard math on strings

```
# x + y
# Error in x + y : non-numeric argument to binary operator
```

How you can use special commands for working with strings

```
DNA = paste(x,y)
DNA
```

```
## [1] "ATGAAA TTTTGA"
```

We will learn more aobut working with strings in subsequent labs.

Complex

We will not work with complex numbers in our labs, but for completeness. A complex value in R is defined by the imaginary value i.

```
x = 1 + 2i
x
```

```
## [1] 1+2i
```

Important note: since there are many built-in functions in R, make sure that the new object names you assign are not already used by the system. A simple way of checking this is to type in the name you want to use. If the system returns an error message telling you that such object is not found, it is safe to use the name. For example, c (for concatenate) is a built-in function used to combine elements so NEVER assign an object to c!

Vectors

A vector is a sequence of data elements of the same basic type. data elements in a vector are officially called components. Assignment operator (<-) stores the value (object) on the right side of (<-) expression in the left side. Once assigned, the object can be used just as an ordinary component of the computation. The c function concanenates the components into a vector.

```
x<- c(1,10,100)
x
```

```
## [1]   1  10 100
```

Now you can do scalar computations on a vector

```
x * 2
```

```
## [1]   2  20 200
```

or use sum, sort, min, max, length and many other operations. For example

```
sum (x)
```

```
## [1] 111
```

You can also do vector arithmatic

```
x<- c(1,10,100)
y<- c(1,2,3)
x * y
```

```
## [1]   1  20 300
```

Vectors can also be made of characters

```
codons<- c("AUG", "UAU", "UGA")
codons
```

```
## [1] "AUG" "UAU" "UGA"
```

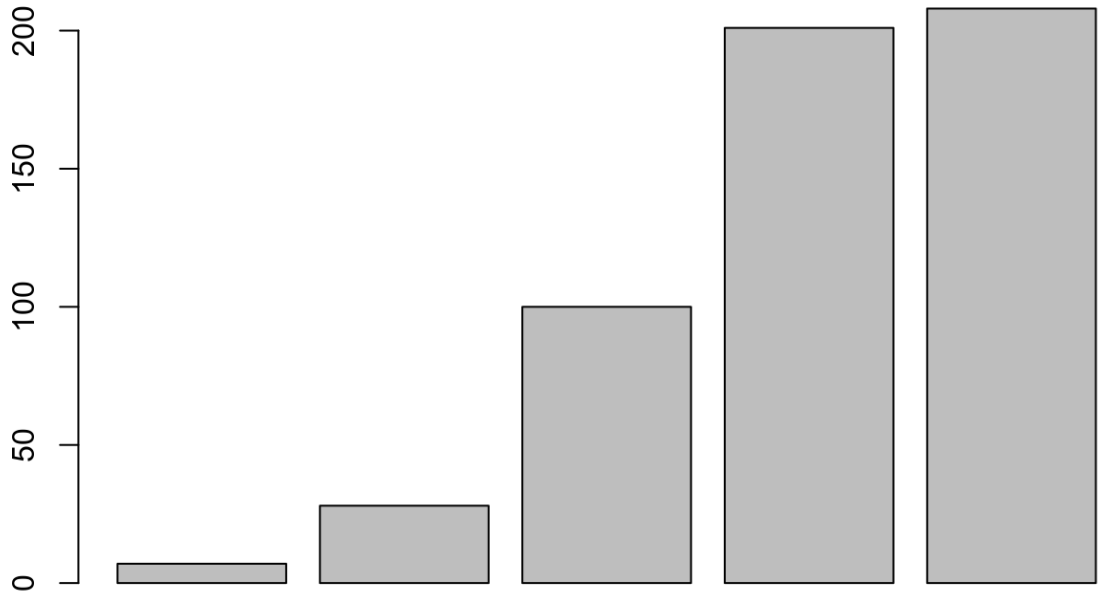
Simple Graphs

Just for fun and we will end this lab with a simple graph

```
RNA_levels<- c(7, 28, 100, 201, 208)
```

A simple bar plot

```
barplot(RNA_levels)
```



Much more on graphing in R later in the course. Check out the R graph gallery (http://rgraphgallery.blogspot.com/) to see some beautiful graphs.

Excercises

Document and save your R script file (e.g. lab1_jlb.R)

Exercise 1

For x = 2 and y = 8, compute the sum, difference, product and quotient of x and y

Exercise 2

For x = 3.5 and y = 5, Test x^5 is greater than y^4 and return a logical value

Exercise 3

Create a vector of the values 211, 62, 108, 43 and 129. Determine the sum of the vector. Divide each value in the vector by the sum to determine relate frequency.

Exercise 4

Create a vector of the nucleotides A, T, C and G. Sort the vector.

What to turn in?

Your mylab1_notes.R file that includes the examples abd exerices. Be sure to include comments to mark each section.