Introduction to R and RStudio

Jenny Wu
Director of Bioinformatics
Genomics High Throughput Facility
Chao Family Comprehensive Cancer Center
UC Irvine
Outline

• Overview of R and RStudio
• How to navigate RStudio
• R basic data types and data structures
• Data frames and Data I/O
• Sample Dataset hands on embedded

https://www.r-project.org/
https://www.rstudio.com/
What’s R

• R is a programing language and software environment for statistical computing and graphics developed in 1993.

• The R language has its roots in the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues.

• R is developed by the R Development Core Team and can be considered as a different implementation of S.

• R is the language of Data Science with strengths in data management, data visualization, and statistics.
Why R?

• Freely available under GNU general public license
• Open source (non-proprietary)
• Can run on various operating systems like Windows, Linux and Mac
• Active community development to solve problems (2000+ libraries)
• If you are trying to solve a problem, someone else has probably written a R solution that you could use
• Connects with other languages such as SQL, python etc.
• Bioconductor
What’s R Studio

• Full featured integrated development environment (IDE) and modeled after Matlab’s IDE.

• RStudio has many features: it includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management syntax highlighting code completion smart indentation

• Useful cheat sheets: https://www.rstudio.com/resources/cheatsheets/
Why RStudio?

• Available in open source and commercial editions and runs on the desktop (Windows, Mac, and Linux) or in a browser connected to RStudio Server
• Auto Code completion
• Syntax highlighting
• Integrated help and documentation
• Develop and test interactive graphics
• Author reports, books, slides
Navigating RStudio

**On HPC3 start a new session of Rstudio**

- Rstudio can be divided into different panes
- You can customize the configuration under “View” -> “Panes”

**Source**

- Area to write R code that you want to save for later, including functions, commands, objects etc
- Similar to a Word Document but for code

**Console**

- Area to type R commands to execute
- Meant for interactive use and quick checks
Navigating RStudio

Environment/History

The Environment tab will provide a detailed list of every function or symbol that is defined in the Console.

Miscellaneous

Files, Plots

Packages

• Lists all downloaded packages
• Able to download and update packages
• Packages are user-written R code to solve a specific problem

Help

• Contains documentation for anything R-related
• Get help by typing in script or console:
  • help("topic") or ?topic
Navigating RStudio: Source and Console

- R comments are prefixed by #
- Comments are useful notes about the R code but are not R code
- R code is not prefixed by anything in source pane but in console with “>”
- R output is prefixed by > and is displayed in the console pane
- It is good practice to load libraries (library()) and set path first (setwd())
Useful R shortcuts

• Ctrl+Enter: send code to R console
• Ctrl+Shift+C: comment/uncomment
• Ctrl+1/2/3/4: switch window focus
• Ctrl+Alt+Enter: send code to terminal
• Ctrl+L: clear console
Packages

- Packages: bundles of code that add new functions to R to solve a specific problem.
- Basic and contributed packages
- Packages are submitted to Comprehensive R Archive Network (CRAN, official repository for R packages)
- R has a huge number of external packages on CRAN and github
- Browse packages at r-pkg.org
R Variables

Let’s start typing in HPC3 RStudio console

• All variables (scalars, vectors, matrices, etc) created by R are called objects
• Use arrow to assign value to a variable
• To view the contents of any R object, just type its name
• To view all the objects in the current session, use ls()
Basic Data Types in R

There are 5 atomic (basic) data types: numeric, integer, complex, character, logical

```r
> chrNum <- 10L
> coord_start <- 2419
> coord_end <- coord_start + 10000
> gene_name <- "ISLET1"
> isCTRL <- TRUE
> typeof(chrNum)
> typeof(coord_start)
```
Basic Data Structures in R

• Vector: set of obj of the same type in an 1-d array
  > gender <- c("Male", "Male", "Female", "Female")
  > data <- 1:6
  > mat <- matrix(data, nrow=2)

• Matrix: set of obj arranged in higher dimension
  > genotype <- factor(c("WT", "WT", "Mut", "Mut"))
  > summary(genotype)
  > typeof(genotype)

• Factors: used to represent categorical data
  > gender_cat <- c(gender, gender)#concatenate

• List: set of obj of different types

• Data frame: a 2-dimensional rectangular table of values where every value in a column must be the same type.
  > data2 <- data+data
  > mat2 <- mat*mat
  > mat_dot <- mat %*% mat
  > cat("\014") #clear console, same as ctrl+L
The Iris dataset from UCI ML Repository

Iris Data Set
Download: Data Folder  Data Set Description

Abstract: Famous database; from Fisher, 1936

<table>
<thead>
<tr>
<th>Data Set Characteristics:</th>
<th>Multivariate</th>
<th>Number of Instances:</th>
<th>150</th>
<th>Area:</th>
<th>Life</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute Characteristics:</td>
<td>Real</td>
<td>Number of Attributes:</td>
<td>4</td>
<td>Date Donated</td>
<td>1988-07-01</td>
</tr>
<tr>
<td>Associated Tasks:</td>
<td>Classification</td>
<td>Missing Values?</td>
<td>No</td>
<td>Number of Web Hits:</td>
<td>4737160</td>
</tr>
</tbody>
</table>
Load the iris dataset

```r
library(datasets)# load a library
data("iris") # load an existing dataset
class(iris) # check the class
## [1] "data.frame"
dim(iris) # check the dimension of the data frame
## [1] 150  5
head(iris)# check the first few lines. You can use tail() too
## Sepal.Length  Sepal.Width  Petal.Length  Petal.Width Species
## 1       5.1         3.5          1.4         0.2   setosa
## 2       4.9         3.0          1.4         0.2   setosa
## 3       4.7         3.2          1.3         0.2   setosa
## 4       4.6         3.1          1.5         0.2   setosa
## 5       5.0         3.6          1.4         0.2   setosa
## 6       5.4         3.9          1.7         0.4   setosa
summary(iris$Species)
View(iris) # Visualize in source pane
```
Explore the iris dataset

```r
names(iris)

summary(iris)
##     Sepal.Length   Sepal.Width   Petal.Length   Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##        Species
##     setosa    :50
## versicolor  :50
##   virginica  :50

str(iris)  # structure of iris object
plot(iris)# basic plot

t.test(iris$Sepal.Length, iris$Petal.Length)# basic t test
```
Data Frames

• Data frame format is familiar - a typical csv/Excel file in the wild
• Data frames are objects

• Data frame columns hold homogeneous data
• The entire data frame holds heterogeneous data
Manipulating Data Frames

• Create a data frame with the `data.frame()` function

• Subsetting is the action of selecting specific pieces of our data

• R provides subsetting operators that allow us to select data in complex and useful ways

• Subsetting Operators: `[]`, `[[ ]]`, `$`
  • Syntax to select all rows, `m[ , ]`
  • Syntax to select all columns, `m[ , ]`

```r
> iris[2:4,]
> iris[1:3, "Species"]
> iris$Petal.Length
> iris[iris$Species == "virginica", 1:3]
> iris[iris$Petal.Length < 2, 1:5]
> iris[iris$Petal.Length > 2 & iris$Species == "versicolor", ]
> x <- iris[2,]
> typeof(x)
> class(x)
> y <- iris[[2]]
> typeof(y)
> class(y)
```
Subsetting Data Frames

• R has a built-in function `subset()` that allows you to select and filter data all in the same function

```r
> ?subset
> subset(iris, Petal.Length < 2) [, "Species"]
> subset(iris, Species == "virginica") [1:5,]
> x <- subset(iris, Species == "virginica") [1:5,]
> typeof(x)
> class(x)
```
Data I/O

• Pay attention to your working dir and the location of I/O files
• Base R provides basic functions for csv/txt file I/O
• More fancy functions with *readr, readxl* packages

```r
> getwd()
> ?read.csv
> countsTable <- read.csv("/dfs6/pub/ucightf/workshop/counts.csv", header=TRUE, stringsAsFactors=FALSE)
> head(countsTable)
> dim(countsTable)
> x <- countsTable[rowSums(countsTable>0)>6,]  # remove the rows that have 0s in all samples
> ?write.csv
> write.csv(x,"counts1.csv",quote=F)
> saveRDS(x,"counts1.rds")
> x <- readRDS("counts1.rds")
```
Executing R scripts from command line

- Automate your R scripts
- Integrate R into production
- Call R through other tools or systems

```plaintext
> module load R
> Rscript yoursript.R

# Here is Linux shell prompt not R console
```
Instruction to install R and Rstudio

• Find out what is the latest version of R, you can look at the CRAN (Comprehensive R Network) website, http://cran.r-project.org/.

• https://docs.google.com/document/d/1-IMGaaFzdiFKGqCc_l98uyvImqlgUiPPC1mysPEot-0/
Useful Links

- https://cran.r-project.org/
- https://cran.r-project.org/doc/manuals/R-intro.html
- https://cran.r-project.org/other-docs.html
- https://www.stat.berkeley.edu/~spector/R.pdf
- https://rseek.org/
- https://www.datacamp.com/