Introduction to R-Part 1

Overview of R and RStudio

Yebelay Berehan

yebelay.ma@gmail.com

2022-05-21

Outlines

1. What / Why R?
2. Rstudio & R
   a. The Source, Console, Help and Environment panes
   b. Functions and Data Objects
3. Working with R: Objects and Workspace
   a. R Objects & Project Management
   b. Good Coding practice

1. What is R?

- Computer language & environment for statistical computing & graphics.
- Script based (text computer code), not GUI based (menu / point & click).
- Tools for Data Handling and manipulation
- Large collection of statistical tools (packages) for Data Analysis;
    - contributed by many experts
- Graphical interface for Visualizing Data & results from statistical analyses
- Relatively simple and effective, widely used,
    - **free, open source**.

Why R?

- Open source (free!):
  - open for anyone to review and contribute.
- Maintained by top quality experts
- Built for statistical analysis
- Reproducible and transparent:
  - Saved R code can be used to easily reproduce any analysis and Collaborators can share their work in the R script format.
- Publication-ready data visualization
- Software compatibility
- Generating reports in various formats (MS word, PDF)

2. RStudio

What is RStudio? Why use it?

- Best Integrated Development Environment (IDE) for R.
- Powerful and makes using R easier
- RStudio can:
    - Organize your code, output, and plots.
    - Auto-complete code and highlight syntax.
    - Help view data and objects.
    - Enable easy integration of R code into documents.
- User-friendly interfaces.

Basic Setup

Installing R

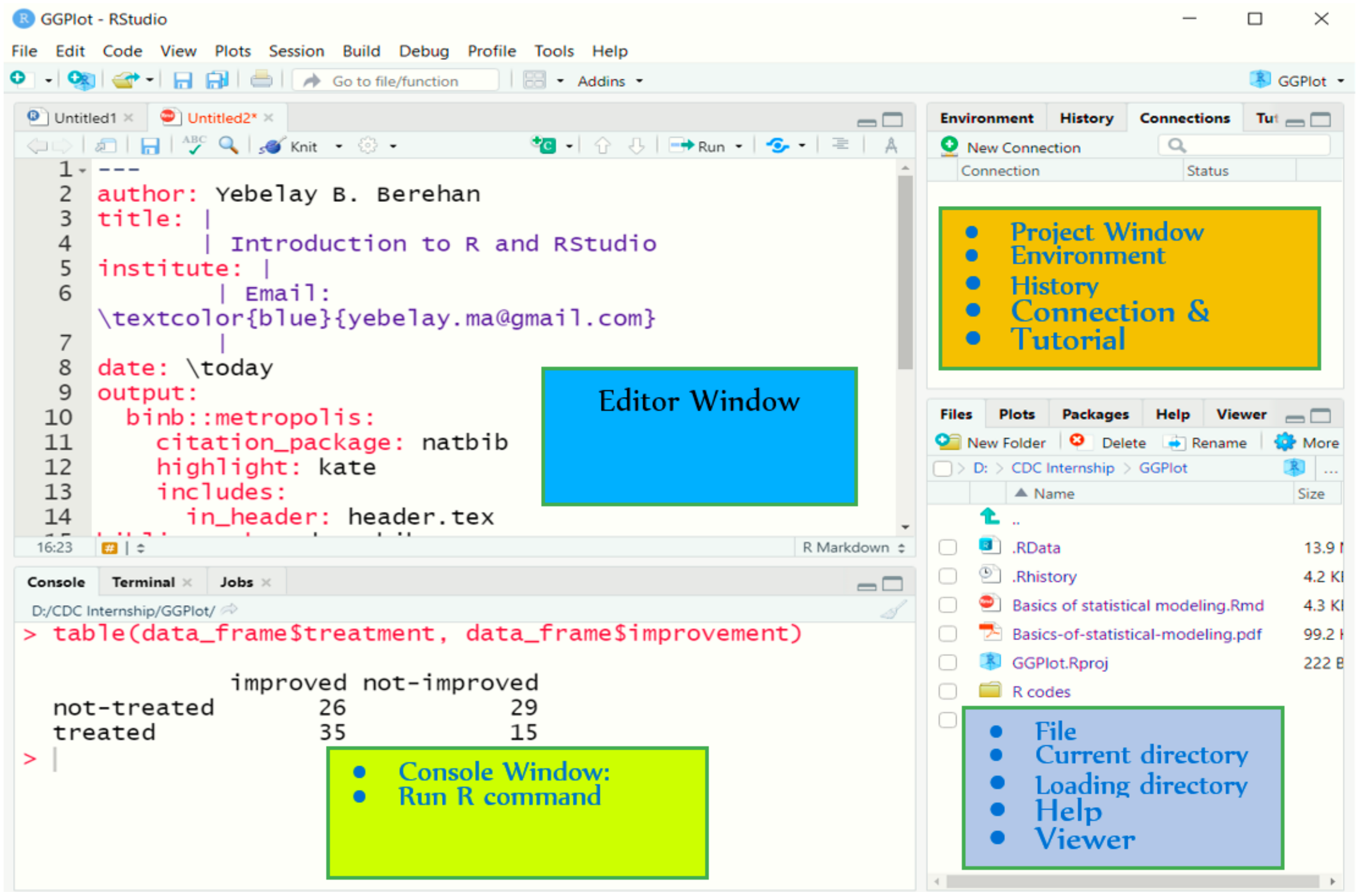
- Visit https://cran.r-project.org/
- Or simply google download R to find the link to download page.
- Also, check out **Install R** tutorial video by RStudio, Inc.

Installing RStudio

- Visit https://www.rstudio.com/products/rstudio/download/
- Or simply google download Rstudio to find the link to download page.
- Also, check out **Install RStudio** tutorial video by RStudio, Inc.

Choose the version for your computer and follow installation instructions.

RStudio Overview



```
---
author: Yebelay B. Berehan
title: |
       | Introduction to R and RStudio
institute: |
         | Email:
\textcolor{blue}{yebelay.ma@gmail.com}
         |
date: \today
output:
  binb::metropolis:
    citation_package: natbib
    highlight: kate
    includes:
      in_header: header.tex
```

```
> table(data_frame$treatment, data_frame$improvement)

             improved not-improved
  not-treated      26           29
  treated          35           15
>
```

Getting Started

- RStudio will open with 4 sections (called panes):

1. Source editor pane

- Write and edit R scripts

2. Console pane

- Interactively run R commands

3. Environment/history pane

- Environment: view objects in the global environment
- History: search and view command history

4. Files/Plots/Packages/Help pane

- Files: navigate directories
- Plots: view generated plots
- Packages: manage installed packages in the library
- Help: view help documentations for any package/function

Customization

Panes

- The size and position of panes can be customized.
- On the top right of each pane, there are buttons to adjust the pane size.
- Also, place your mouse pointer/cursor on the border line between panes and when the pointer changes its shape, click and drag to adjust the pane size.
- For more options, go to **View > Panes** on the menu bar.
- Alternatively, try **Tools > Global Options > Pane Layout**.

Appearances

- The overall appearance can be customized as well.
- Go to **Tools > Global options > Appearnce** on the menu bar to change themes, fonts, and more.

Installing and Loading Packages

- Packages are collections of R functions, data, and compiled code in a well-defined format.
- There are three categories of packages.

1. Base Packages: Providing the basic functionality, maintained by the R Core Development group. Currently, there are 14 packages, these are

```
rownames(installed.packages(priority="base"))
```

```
##  [1] "base"      "compiler"  "datasets"  "graphics"  "grDevices" "grid"
##  [7] "methods"   "parallel"  "splines"   "stats"     "stats4"    "tcltk"
## [13] "tools"     "utils"
```

2. Recommended Packages: also a default package, mainly include additional more complex statistical procedures. These are 15 packages

```
rownames(installed.packages(priority="recommended"))
```

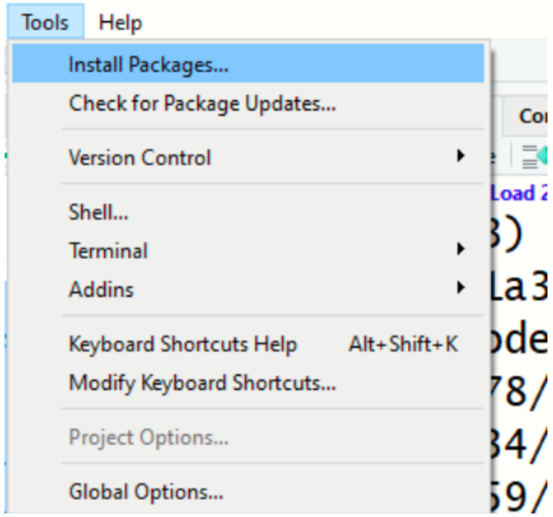
```
##  [1] "boot"       "class"      "cluster"    "codetools"  "foreign"
##  [6] "KernSmooth" "lattice"    "MASS"       "Matrix"     "mgcv"
```

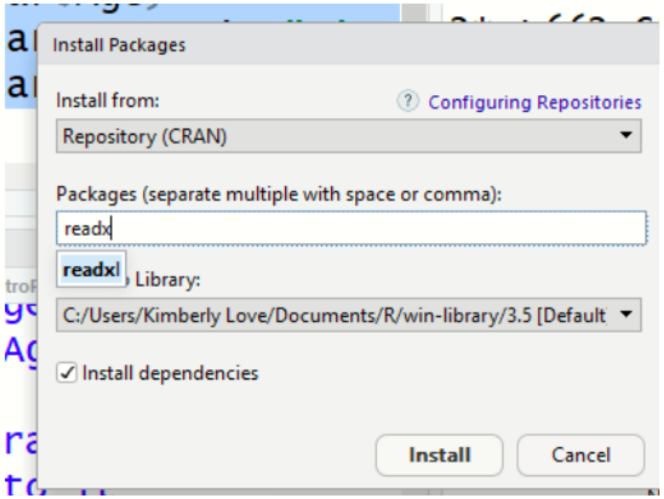
3. Contributed packages: Due to the open nature of R, anyone can contribute new packages at any time.

- Currently, the CRAN package repository features 19022 available packages.

Installing Packages

- **Option 1:** Menu





Install from: Repository (CRAN)

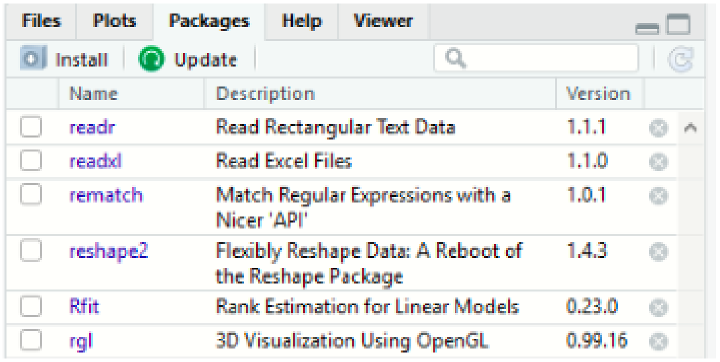
Packages (separate multiple with space or comma): readx

readxl

Library: C:/Users/Kimberly Love/Documents/R/win-library/3.5 [Default

✓ Install dependencies

- **Option 2:** Packages Window



	Name	Description	Version
☐	readr	Read Rectangular Text Data	1.1.1
☐	readxl	Read Excel Files	1.1.0
☐	rematch	Match Regular Expressions with a Nicer 'API'	1.0.1
☐	reshape2	Flexibly Reshape Data: A Reboot of the Reshape Package	1.4.3
☐	Rfit	Rank Estimation for Linear Models	0.23.0
☐	rgl	3D Visualization Using OpenGL	0.99.16

- **Option 3:** Code

```
install.packages("readxl")
```

Loading Packages

```
library()    # see all packages installed
search()     # see packages currently loaded
```

Updating R and RStudio

Updating R

- Go to CRAN and download new version
- **More efficient:** install `installr` package, load it, and run `updateR()`
    - Updates R and Optionally updates all packages
    - **May be better to do this in `basic Rgui`**
- Version should update automatically in RStudio
    - Check/change R version under **Tools>Global Options>R version**
- Then update the R packages with the code:

```
update.packages(ask = FALSE, checkBuilt = TRUE)
```

- To updating RStudio: Go to RStudio and download new version
- Click on Help>Check for Updates, follow menu prompts

Functions and Help

- Information about a function `read.table` can be accessed by typing the following into the console:

```
help(read.table) # help about function read.table
?read.table # same thing
help.start() # general help
example("read.table") # show an example of function read.table
Sys.Date()
```

- Arguments are the inputs to a function.
- In this case, the only argument to `help()` is `read.table`.
- Help files provide documentation on how to use functions and what functions produce.

3. Working with R Objects

Organize with an RStudio project

- It is a good habit to immediately create a project for handling the analysis of new data and keep everything together.
- The workspace is a working environment where R will store and remember user-defined objects: **vectors, matrices, data frames, lists, variables** etc.
- To Create an R project, go to
- **File > New Project and then choose: New Directory> Name for the directory > Click on Create Project**
- For more complex project it may be useful to create sub-directories to contain data, scripts and other documents separately.
- Can also type the below function into the Console, but we won't do that in this session.

```
prodigenr::setup_project("C:/Users/yebel/Desktop/LearningR")
```

Creating R objects

- Objects can be created in the form of
  - `variable <- value` or `variable = value` or `variable -> value`.
  - Variable names can be letters, numbers, and the dot or underline characters but not dot followed by numbers `.4you` is illegal).
- the symbol `<-` (`Alt + -`) that could be read as `assign` or `place into` or `read in` etc.

```
# need to placed in quotes as diabetic is string.
A <-"Diabetic"
```

- The standard data objects in R are: **scalars, vectors, factors, matrices and arrays, lists, and data frames**.
- Data types assigned to each objects are: **logical, numeric, integer, character, complex.**

Vector

- A set of scalars arranged in a one-dimensional array.
- Data values are all the same mode(data type), but can hold any mode.
  - e.g:(-2, 3.4, 3), (TRUE, FALSE, TRUE), ("blue", "gray", "red")
- Vectors can be created using the following functions:
- `c()` function to combine individual values
  - `x <- c(10.4, 5.6, 3.1, 6.4, 21.7)`
- `seq()` to create more complex sequences
  - `seq(from=1, to=10, by=2) or seq(1,10 )`
- `rep()` to create replicates of values
  - `rep(1:4, times=2, each=2)`

Some useful functions in vector

- `class(x):` returns class/type of vector x
- `length(x):` returns the total number of elements
- `x[length(x)]:` returns last value of vector x
- `rev(x):` returns reversed vector
- `sort(x):` returns sorted vector
- `unique(x):` returns vector without multiple elements
- `range(x):` Range of x
- `quantile(x):` Quantiles of x for the given probabilities
- `which.max(x):` index of maximum
- `which.min(x):` index of minimum

Factors

- A factor is used to store predefined categorical data
- Can be ordered and unordered
    - e.g. :("yes", "no", "no", "yes", "yes"), ("male", "female", "female", "male")
- Factors can be created using `factor()`

```r
size <- factor(c("small", "large", "small", "medium"))
```

- The levels of a factor can be displayed using `levels()`.

Matrix

- Matrix is a rectangular array arranged in rows and columns.
- The individual items in a matrix are called its elements or entries.
- Matrices can be created by:

1. `matrix()`
2. converting a vector into a matrix
3. binding together vectors

- Matrices can be created using the functions:
  - `matrix()` creates a matrix by specifying rows and columns
  - `dim()` sets dimensions to a vector
  - `cbind` combines columns
  - `rbind` combines rows

e.g.

```
m1<-matrix(data = 1:6, nrow = 3, ncol = 2)
m2<-cbind(1:3,5:7,10:12)
x=1:6
dim(x) <- c(2, 3)
```

- Note: `dim()` can also be used to retrieve dimensions of an object!

Assign names to rows and columns of a matrix

```
rownames(m1) <- c("A", "B", "C")
colnames(m1)<- c("a","b")
```

Data frames

- a data set in R is stored a data frame.
- Two-dimensional, arranged in rows and columns created using the function: `data.frame()`
- e.g.

```
df <- data.frame(ID = 1:3, Sex = c("F", "F", "M"), Age = c(17, 18,18))
```

- We can enter data directly by access the editor using either the `edit()` or `fix()`

```
 new.data<-data.frame()  # creates an "empty" data frame
 new.data<-edit(new.data) # request the changes or  fix(new.data)
```

- We'll use the data set called **diabetes data** to do this exploration.

```
library(readr)
diabetes <- read_csv("diabetes.csv")
#View(diabetes)
```

Quick intro to functions

- Use `head()` and `tail()` to view the **first (and last) five rows**
- Use `View()` to **view an entire** `data.table` object
- Use `str()` to view the **structure** of `data.table` object
- Use `tables()` to **show all loaded `data.table` objects**
- **Sorting** and **ordering** rows using `setorder()` and `order()`
- Arguments are always enclosed in parentheses
- `colnames()` to look variable names
- `colSums(is.na())` to sum missing data
- Use `subset()` to subset data.
- Functions in R take named arguments.

Subsetting

```
diabetes[] # the whole data frame
diabetes[1, 1] # 1st element in 1st column
diabetes[1, 6] # 1st element in the 6th column
diabetes[, 1] # first column in the data frame
diabetes[1] # first column in the data frame
diabetes[1:3, 3]
diabetes[3, ] # the 3rd row
diabetes[1:6, ] # the 1st to 6th rows
diabetes[c(1,4), ] # rows 1 and 4 only
diabetes[c(1,4), c(1,3) ]
diabetes[, -1] # the whole except first column
```