# R Training

## Introduction to R, Rstudio, and coding

---

Yebelay Berehan

February 12, 2022

Email: yebelay.ma@gmail.com

## 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
4. Data Reading and Writing

# 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)

# Helpful Links

- General R information: https://www.r-project.org/about.html
- R source code contributors: https://www.r-project.org/contributors.html
- About RStudio: https://www.rstudio.com/about/

## 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

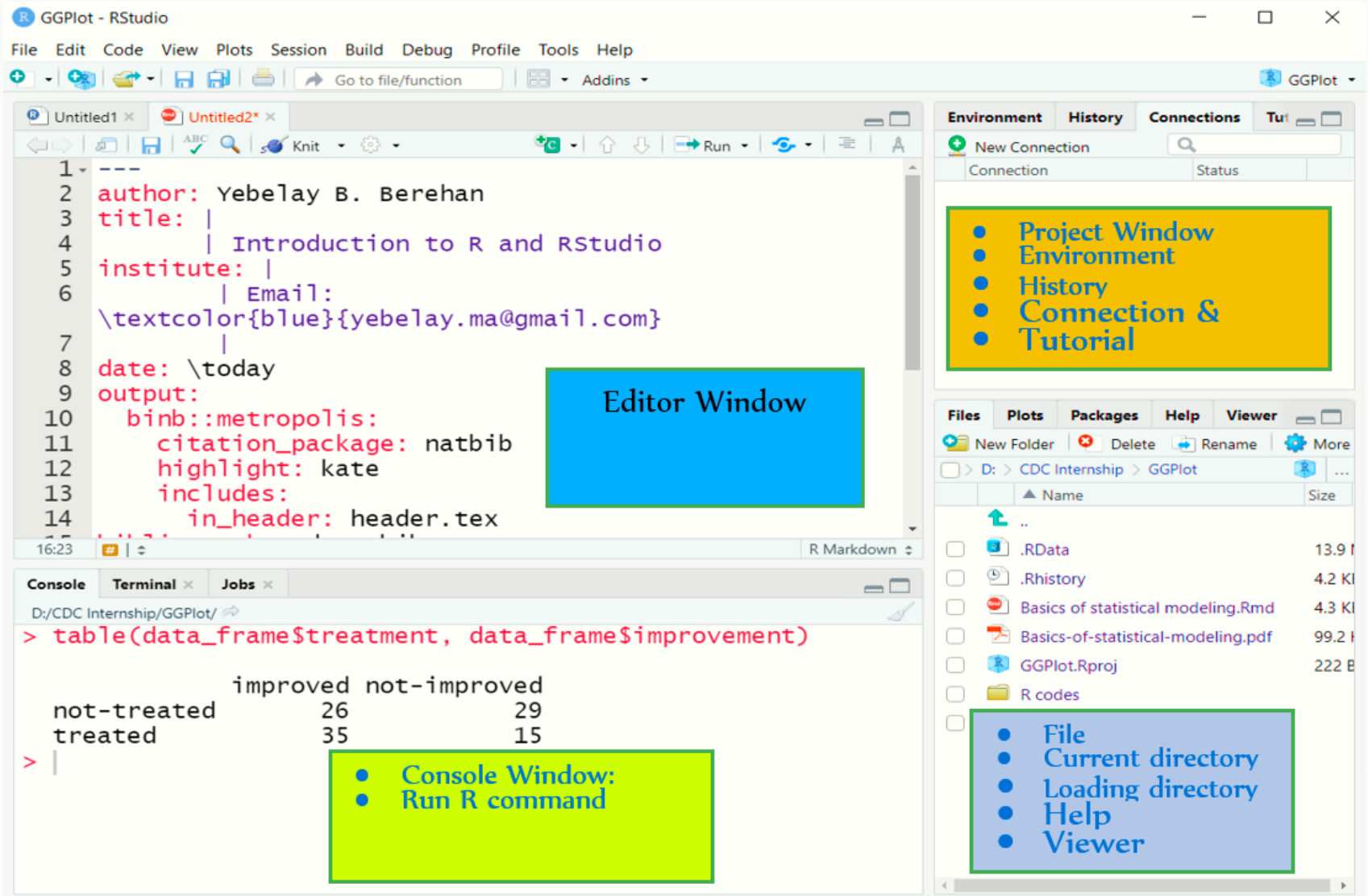
- Visit https://cran.r-project.org/
- Or simply google "download R" to find the link to download page.
- Also, check out "Install R" tutorial video by RStudio, Inc.

## 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



# Getting Started

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

**Source editor pane**

- Write and edit R scripts

**Console pane**

- Interactively run R commands

**Environment/history pane**

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

**Files/Plots/Packages/Help pane**

- Files: navigate directories
- Plots: view generated plots
- Packages: manage installed packages in the library

# 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.

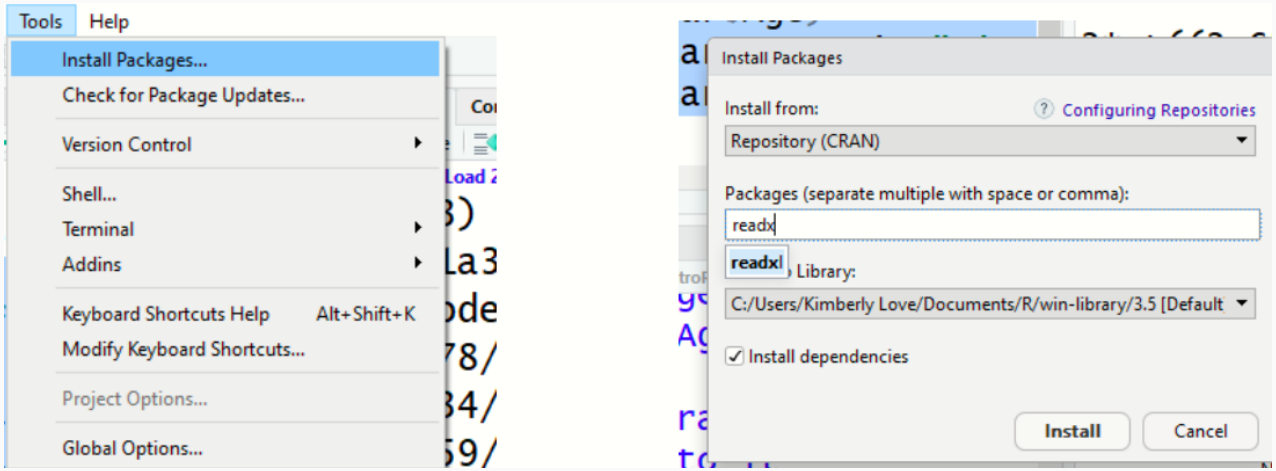
# What is R package

- Packages are collections of R functions, data, and compiled code in a well-defined format.
- The directory where packages are stored is called the library.
- R comes with a standard set of packages.
- Others are available for download and installation.
- Once installed, they have to be loaded into the session to be used.

# Installing and Loading Packages

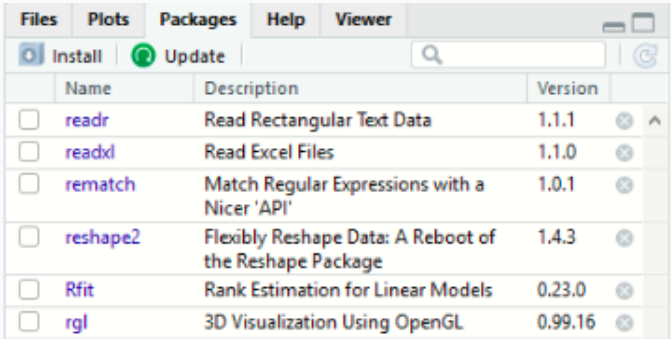
## Installing Packages

- Option 1: Menu



# Installing and Loading Packages

- Option 2: Packages Window




- Option 3: Code

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

## Loading Packages

```
.libPaths() # get library location
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

**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.t
```

- 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 and Workspace

- The workspace is a working environment where R will store and remember user-defined objects: vectors, matrices, data frames, lists, functions, variables etc.
- At the end of an R session, the user can save an image of the current workspace that is automatically reloaded the next time R is started.
- Except for functions most of the user-defined objects are usually referred to as R objects as a way to designate them.
- An R object makes it easy for humans to designate its content which could be a single numerical value or a large table of data.

# 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.

Create an R project

- 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 <- that could be read as "assign.." or "place into" or "read in" etc.

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

- 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():‘

```
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

# Matrix

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))
str(df) # `str()` displays internal structure of object
```

- We can enter data directly in to a data frame by using the built-in data editor.
- We can access the editor by using either the edit() or fix()

# Data frames

**commands:**

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

# Data frames

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

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

```
colnames(mydata)
head(mydata) # to see 6 lines of data
tail(mydata, 4) # last four rows
colSums(is.na(mydata)) # to missing data
dim(mydata)
length(mydata)
str(mydata)
summary(mydata)
```

# Subsetting

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

# 4. Reading and Writing data

- Importing data is rather easy in R but that may also depend on the nature of the data to be imported and from what format.
- Mostly data are in tabular form such as a spreadsheet or a comma-separated file (.csv).
- Base R has a series of read functions to import tabular data from plain text files with columns delimited by: space, tab, comma, with or without a header containing the column names.
- With an added package it is also possible to import directly from a Microsoft Excel spreadsheet format or other foreign formats from various sources.

# Importing from local files

- In base R the standard commands to read text files are based on the read.table() function.
- The following table lists the collection of the base R **read** functions.
- For more details use the help command **help(read.table)** that will display help for all.

**Table 1:** Details of dataset readings

| Function name | Assumes header | Separator | Decimal | File type |
|---|---|---|---|---|
| read.table() | No | " " | . | .text |
| read.csv() | Yes | "," | . | .csv |
| read.csv2() | Yes | ";" | , | .csv |
| read.delim() | Yes | "tab" | . | .text |
| read.delim2() | Yes | "tab" | , | .text |

# Reading raw data from other sources

## Import data

- There are many ways to get data into R and out of R.
- Import text file data using read.table() and comma separated files using `read.csv()` functions.

```
-# syntax:
  read.table("file name with full path", arguments)
```

```
# Examples:# Creates a data frame named myData
  mydata<- read.table(file = "datafile.txt",sep=" ",
                      header=TRUE)
  mydata<- read.csv(file = "datafile.csv")
```

# Reading raw data from other sources

- File names are specified in the same way as `file.choose()` function can be used to select the file interactively. i.e.

```
mydata <-read.csv(file.choose(),sep=",",header=T)
```

## Useful arguments

- Check these arguments carefully when you load your data

```
myData<-read.csv(file="datafile.csv",header= TRUE,
sep=",", strip.white =TRUE, na.strings= " ")
```

# Reading raw data from other sources

- You can reduce possible errors when loading a data file
- The header = TRUE argument tells R that the first row of your file contains the variable names
- The sep = "," argument tells R that fields are separated by comma
- The strip.white = TRUE argument removes white space before or after factors that has been mistakenly inserted during data entry.
- The na.strings = " " argument replaces empty cells by NA (missing data in R)

# Read data

**Stata to R:** Different packages for stata version >=13 vs. <13

```
library(foreign) # Versions before stata 13
data <-  read.dta(file="XXX.dta") # Other options
```

```
library(readstata13) # Versions from stata 13
data <-  read.dta13(file="XXX.dta") # Other options
```

**Excel to R:** There are several packages

```
library(readxl)
data <-  read_xlsx(path="XXX.xlsx", sheet = 1,
col_names = TRUE)
```

```
library(readr)
data <- read_csv("D:/CDC Internship/R Training/Ethiopia_R_t
```

# Read data

**CSV to R:** There are several packages

```
library(readr)
data <- read_csv(file="XXX.csv")
```

**Text file to R:** Available in R base, used for text and csv files

```
data <- read.table(file="XXX.txt",header=TRUE,
                   sep="")
```

```
data <- read.table(file="XXX.csv",header=TRUE,
                   sep=",")
```

# Exporting Data

**R to Stata:** Use the libraries haven or readstata13

```
write.dta(data,file="XXX.dta")# Other options
save.dta13(data, file="XXX.dta") # Other options
```

**R to Excel:** Note the package readxl does not work here

```
library(xlsReadWrite)
write.xls(data,"data.xls")# Other options
```

**R to csv:** Use readr package

```
write_csv(data, "data.csv", na = "")
```

**R to a text file:**

```
write.table(data, "data.txt", sep="\t")
```