Statistical Computing with R
Masters in Data Science 503 (S4)
Fourth Batch, SMS, TU, 2025

Shital Bhandary
Associate Professor
Statistics/Bio-statistics, Demography and Medical Education
Patan Academy of Health Sciences, Lalitpur, Nepal
Faculty, Masters in Medical Research, NHRC/Kathmandu University
Faculty, FAIMER Fellowship in Health Professions Education, India/USA

Review Preview

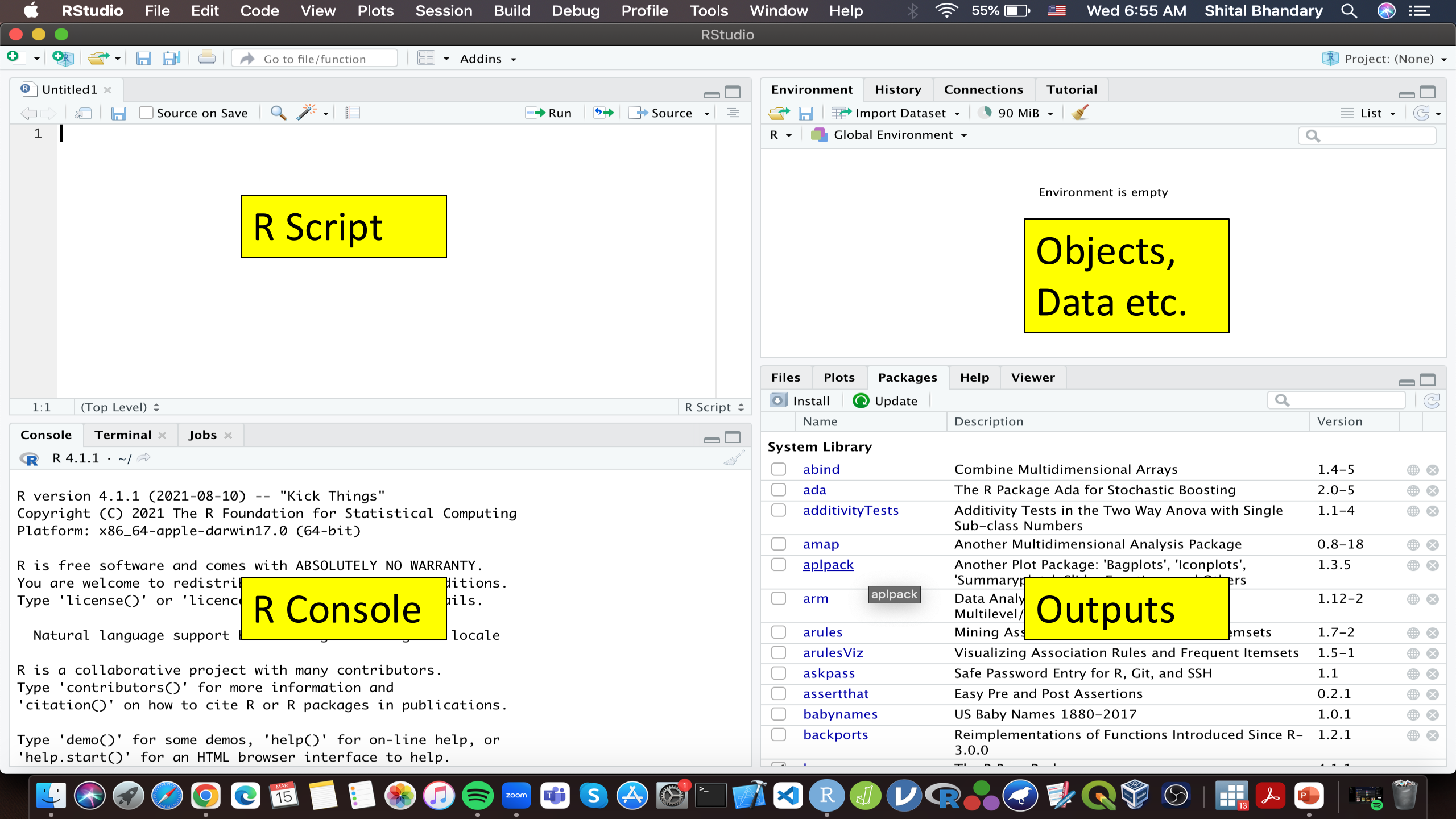
- Basics of R
- Chapter from “R for Everyone” book
- **We discussed it in the last class**

- Basics of coding in R
- Chapter from “Hands-on Programming with R” book
- **We will discuss this in today’s class**

R script: Programming in R

- We must use R script to store the codes and reproduce the results
- We can start a new script in R Studio using “+” sign then R script
- OR
- We can use File → New File → R script

Note: R script is a text file with .R extension


RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
55% Wed 6:55 AM Shital Bhandary
RStudio
Go to file/function
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1
R Script
1:1
(Top Level)
R Script
Environment
History
Connections
Tutorial
Import Dataset
90 MiB
List
R
Global Environment
Environment is empty
Objects,
Data etc.
Files
Plots
Packages
Help
Viewer
Install
Update
Name
Description
Version
System Library
abind
Combine Multidimensional Arrays
1.4-5
ada
The R Package Ada for Stochastic Boosting
2.0-5
additivityTests
Additivity Tests in the Two Way Anova with Single Sub-class Numbers
1.1-4
amap
Another Multidimensional Analysis Package
0.8-18
aplpack
Another Plot Package: 'Bagplots', 'Iconplots',
1.3.5
arm
1.12-2
arules
1.7-2
arulesViz
Visualizing Association Rules and Frequent Itemsets
1.5-1
askpass
Safe Password Entry for R, Git, and SSH
1.1
assertthat
Easy Pre and Post Assertions
0.2.1
babynames
US Baby Names 1880-2017
1.0.1
backports
Reimplementations of Functions Introduced Since R-3.0.0
1.2.1
Console
Terminal
Jobs
R 4.1.1 · ~/
R version 4.1.1 (2021-08-10) -- "Kick Things"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
R Console
Natural language support
locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Outputs
aplpack

RStudio

Go to file/function | Addins | Project: (None)

Untitled1*

Source on Save | Run | Source

```
1  #Column vector
2  x <- c(1:30)
3  y <- x^3
4  plot(x, y)
```

#Column vector
x <- c(1:30)
y <- x^3
plot(x, y)

1:1 (Top Level) R Script

Console | Terminal | Background Jobs

R 4.3.2 · ~/

```
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> #Column vector
> x <- c(1:30)
> y <- x^3
> plot(x, y)
>
```

This appears here after the codes are selected and “Run”!

Environment | History | Connections | Tutorial

Import Dataset | 136 MiB | List

R | Global Environment

Values

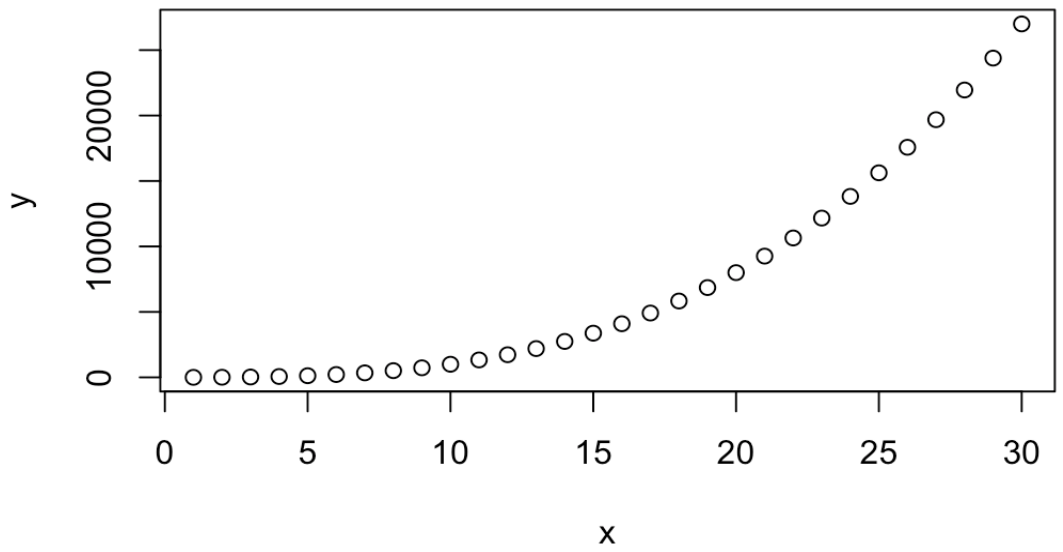
x	int [1:30] 1 2 3 4 5 6 7 8 9 10 ...
y	num [1:30] 1 8 27 64 125 216 343 512 729 1000 ...

This shows how ”x” and “y” objects are stored after the script was “Run” and compiled with success.

Files | Plots | Packages | Help | Viewer | Presentation

Zoom | Export | Publish

This appeared as plot(x,y) was ”Run” in script!



Lets get a data file from internet to use pipes: Must use R script in R Studio!

 # Load in the Iris data from internet:

```
iris <- read.csv(url("http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"), header = FALSE)

head(iris)
```

 # Add column names for V1, V2, V3, V4 and V5 columns to the Iris data

```
names(iris) <- c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width", "Species")
```

Saving the data frame as “csv” file in laptop: Use R script in R Studio!

- You can save the data from internet as CSV file in local computer
- write.csv(DataFrame Name, "Path to export the DataFrame\\File Name.csv", row.names=FALSE)
- **write.csv(iris, “iris.csv”)** #Will save CSV file in working directory
- To know your working directory: getwd()

```
> getwd()
[1] "/Users/shitalbhandary"
```

Class/Home work!

- Covert iris data to long format using reshape2 package (install it if not done so first) and save it as iris_long object in R
- Hint: Use melt function of the reshape2 package (search/find/use)
- Save this as csv file in your working directory
- When do we used long data (instead of wide data)?
  - https://www.thedataschool.com.au/mipadmin/the-shape-of-data-long-vs-wide/

R script example for discussion/practice: Test.R

store the current directory
- initial.dir <- getwd()

change to the new Test directory
- setwd("~/mac/Test")

load the necessary libraries
- library(magrittr) #for pipes

set the output file (it will bypass R and R Studio)
- sink("session3.out")

load the dataset from Test folder
- iris <- read.csv("iris.csv")

Do the analysis
- plot(iris)

This will not appear in R studio!
- summary(iris)
- iris %>% cor(Sepal.Length, Sepal.Width)

close the output file
- sink()

unload the libraries
- detach("package:magrittr")

change back to the original directory
- setwd(initial.dir)

Using forward pipe operator/s in R: **library(magrittr)** required!

Compute the square root of `iris$Sepal.Length` and assign it to the new variable (**Hotkey for pipe i.e. "%>%" is: Ctrl+Shit+M in PC or CMD + Shift + M in MacOS)**

```
iris$Sepal.Length.SQRT <- iris$Sepal.Length %>% sqrt()
```

Compute the square root of `iris$Sepal.Length` and assign it to the **same** variable

```
iris$Sepal.Length %<>% sqrt
```

Return `Sepal.Length` iris$Sepal.Length

 # Be careful while using %<>% as the original data will be lost!

The “tee” pipe operator “%T%”: library(magrittr) required!

```
set.seed(123) # Why?
rnorm(200) %>%
matrix(ncol = 2) %>%
plot %>%
colSums
```

```
set.seed(123)
rnorm(200) %>%
matrix(ncol = 2) %T>%
plot %>%
colSums
```

Normally, code ends after plot command but the “tee” pipe operator allows it to continue for the next argument (without the plot being plotted!)

The exposing pipe operator “%$%”: **library(magrittr)** required!

```
iris %>%
        subset(Sepal.Length > mean(Sepal.Length)) %$%
        cor(Sepal.Length, Sepal.Width)
```

The %$% operation comes handy for functions where “data” argument is not required/used like built-in “cor” function of R!

What will you get with this code:

```
        cor(iris$Sepal.Length, iris$Sepal.Width)
```

When NOT to use pipes?

- In chapter 18 of the web version of the text book “R for Data Science”, the authors have given four suggestions:

  - Your pipes are longer than (say) ten steps
  - You have multiple inputs or outputs
  - You are starting to think about a directed graph with a complex dependency structure
  - You're doing internal package development

More here: https://stackoverflow.com/questions/38880352/should-i-avoid-programming-packages-with-pipe-operators

More references and notes:

- References on “magrittr” package is available here: https://cran.r-project.org/web/packages/magrittr/vignettes/magrittr.html

- tidiverse package and its associated packages i.e. dplyr, ggplot etc. **automatically** loads the “magrittr” so that we can use the pipe operators easily

- R Studio, on the other hand, does not load the magrittr packages “automatically”, we need to load it if tidiverse packages are not used

Questions/queries?

- Final notes:
  - Use %>% frequently
  - Use %<>% if and only if it is required
  - Use %T% and %$% only if you think it is required

Thank you!

@shitalbhandary