R Coding Session: A Journey into Data Manipulation and Visualisation

Day 1: Quick Intro to R & Tidyverse Environment

Indira Puteri Kinasih

June 5, 2023

Universiti Brunei Darussalam

Sessions

Sessions

What is R?

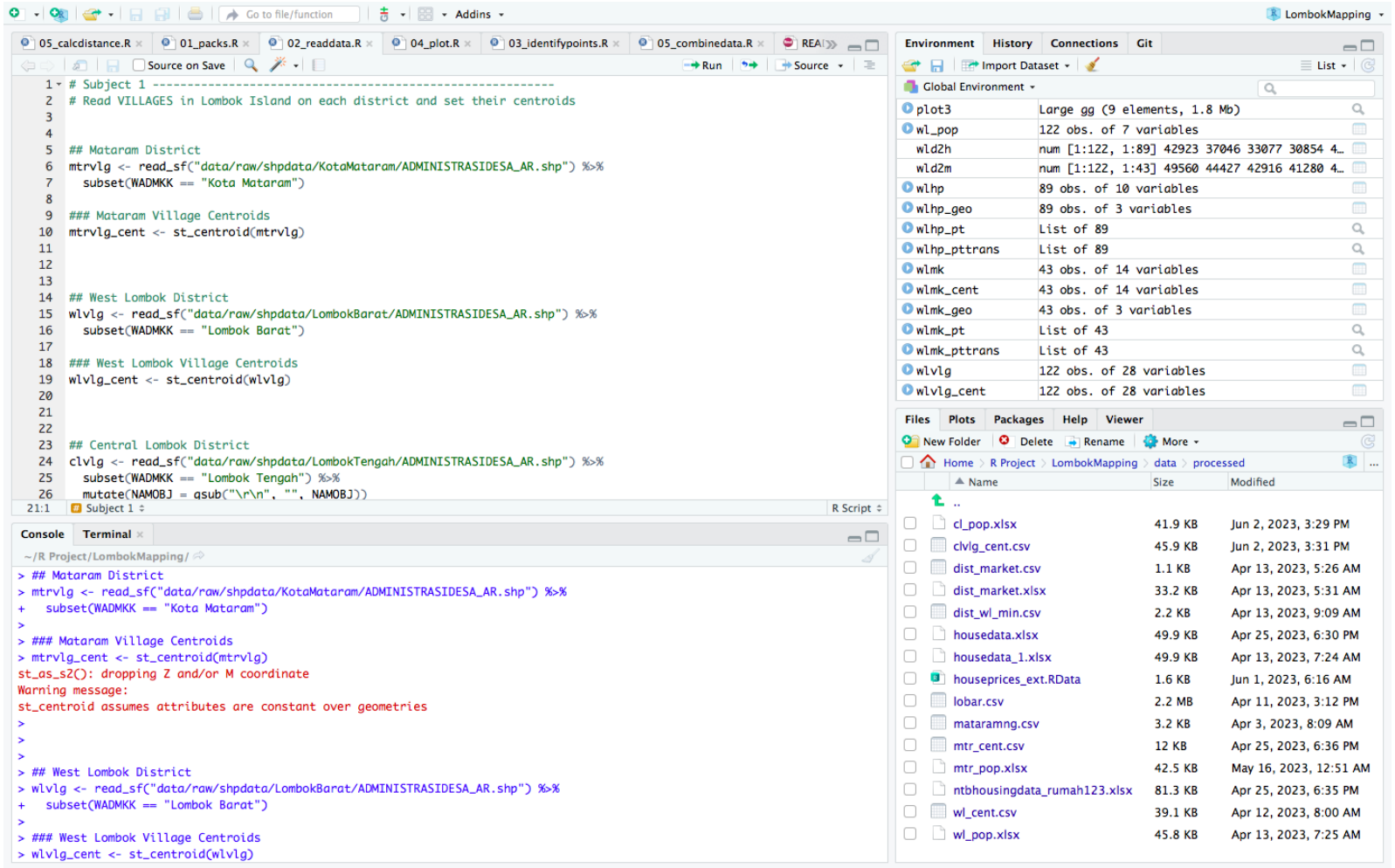
- a programming language and software environment for statistical computing and graphics;
- provides a wide variety of statistical and graphical techniques;
- widely used in academia, industry, and data analysis fields

R and R Studio

- R Studio can be considered as an extension for the programming language R, offering a more intuitive interface that includes the **R Console**, a **script editor**, and additional valuable features such as **R markdown** and integration with GitHub;

Source: Douglas et al., 2023

R Studio Orientation



R Studio Orientation

- **Script**: In the top-left quadrant, there is a simple text editor for writing your R code. It highlights the code, and allows you to easily run a section of your code (highlight a section and hit `command` and `enter`);
- **Console**: In the bottom-left quadrant, there is a console for entering R commands. This closely resembles a command-line interface where you can execute individual lines, and the console will display the corresponding results. Note, you can use the up arrow ↑ to easily access previously executed lines of code

R Studio Orientation

- **Environment**: In the top-right quadrant, the environment displays information that you have stored inside of variables. When working with scripts, it is common to create numerous variables, and the environment area assists in maintaining a record of the stored values and their respective variables;
- **Plots**, **Packages**, etc.: Within the lower-right section, there are several tabs available to access diverse information. This area serves as the rendering space for visualizations and allows you to access documentation and view the packages you have loaded

Getting Started

we can try some simple basic arithmetic expressions, using script or directly in R console, ex:

```
# Arithmetic/Number Operation
2 + 2
[1] 4
```

The other operators are $-$, $*$, / for subtraction, multiplication and division respectively;

Math functions include: $log()$, $log10()$, $exp()$, $sqrt()$. ***Please give it a try :)***

Getting Started

besides number and arithmetic expression, we also can give R a **character** input ex:

```
# Character Input
``Hi, how are you doing?''
[1] ``Hi, how are you doing?''
```

make sure to put “ ” between the characters you want to input

Objects in R

- "everything in R is an **object**";
- it can be single number, character string, plot output, a summary of your statistical analysis, or a set of R commands that perform a specific task
- to view the object's value, type the name of the object

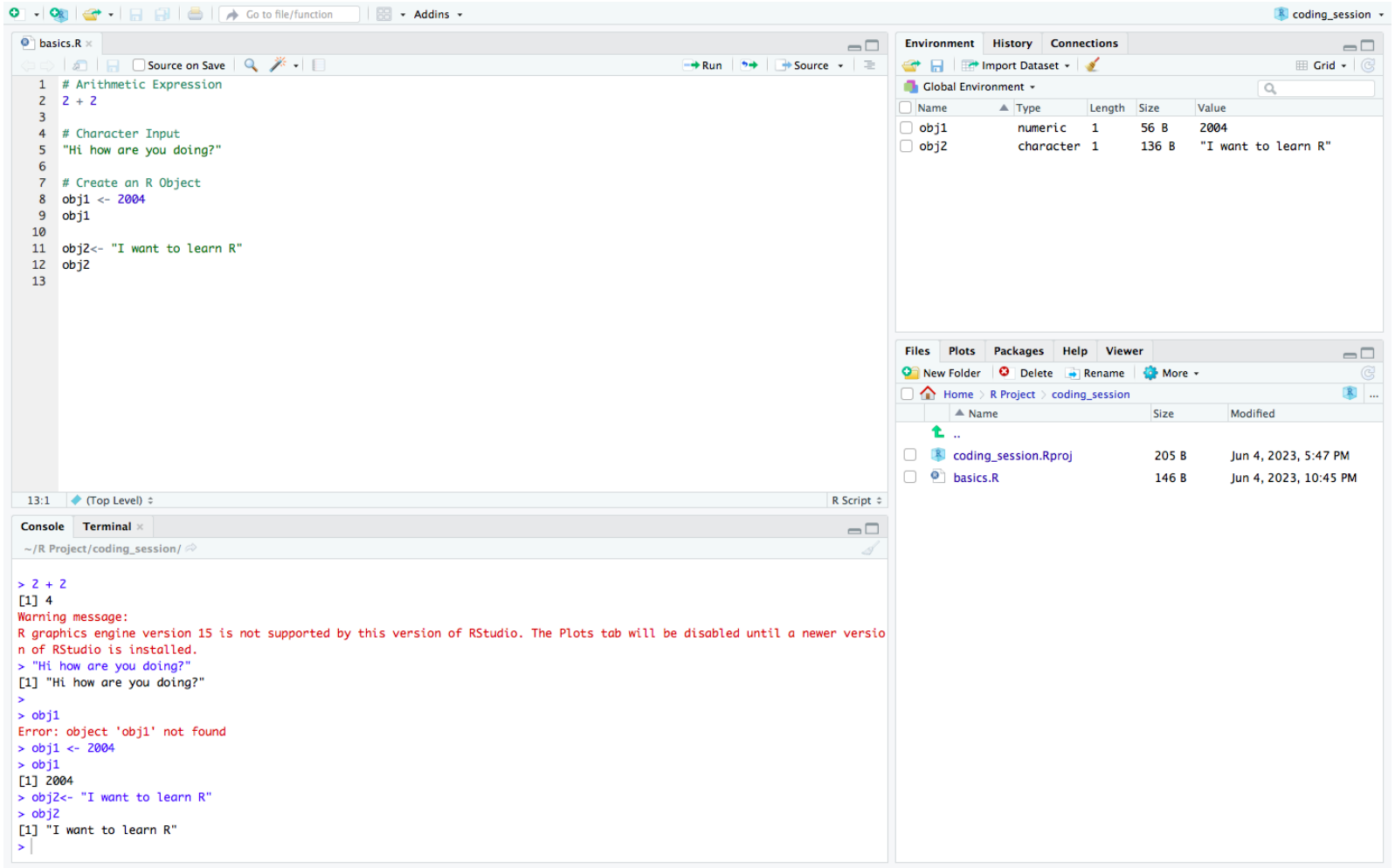
Creating Objects

- create an object = give a name to that object;
- assign a value to this object using the assignment operator <−

```
# Create an Object
obj1 <- 2003
obj1
[1] 2003
```

```
obj2 <- ''I want to learn R''
obj2
[1] ''I want to learn R''
```

Creating Objects



Creating Objects

we can do several stuff with our objects;

```
# Summing two objects
obj3 <- 1982 obj4 <- obj1 + obj3
obj4
[1] 3986
```

Sometimes, we will find an `Error` message

```
obj5 <- obj4 - obj0
obj5
Error object 'obj0' not found
```

why? because we have not created/defined the object `obj0` yet

Functions in R

- A function is a construct/object that holds a set of **instructions** to carry out a particular action or task;
- The base installation of R comes with many functions already defined;
- It is also possible for us to create our own functions to accomplish tasks that are tailored to our objectives.

Functions in R

Example of a function in R is `c()`. It is a function, which stands for *concatenate*, is utilized to **combine multiple values and store them in a vector**, which is a data structure for holding sequential elements.

```
shoes_sz <- c(41, 39, 36, 40, 38, 42, 39)
shoes_sz
[1] 41 39 36 40 38 42 39
```

Functions in R

Now, can you try to calculate the descriptive statistics of `shoes_sz` using functions `mean()`, `var()`, `sd()`, `length()`, and also `summary()` ?

Sessions

Introduction to Tidyverse

- A compilation of packages that specifically target data science
- has significantly advanced the field of R programming.

Source: Roye, 2020

Introduction to Tidyverse

Package	Description
`ggplot2`	Grammar for creating graphics
`purrr`	R functional programming
`tibble`	Modern and effective table system
`dplyr`	Grammar for data manipulation
`tidyr`	Set of functions to create tidy data
`stringr`	Function set to work with characters
`readr`	An easy and fast way to import data
`forcats`	Tools to easily work with factors

Table 1: Important Packages in Tidyverse

Style Guide

- **Avoid using more than 80 characters** per line to allow reading the complete code.
- Always use a space after a comma, never before.
- The operators ($==$, $+$, $-$, $<-$, %>%, etc.) must have a space before and after.
- **No space** between the name of a function and the first parenthesis, nor between the last argument and the final parenthesis of a function.

Style Guide

- **Avoid** reusing names of functions and common variables. Ex: `c <- 5` vs. `c()`;
- Sort the script separating the parts with the comment form `#` and `---`
- **Avoid** accent marks or special symbols in names, files, routes, etc.
- Object names must follow a constant structure: `day_one`, `day_1`.

Pipe Operator

- Using proper indentation is recommended when working with multiple arguments of a function or when chaining functions together using the `pipe` operator (%>%);
- It allows the output of a function applied to the first argument to be passed as the input to the next function.

Pipe Operator

```
# Example of a simple pipe application
c(41, 39, 36, 40, 38, 42, 39) %>% mean()
[1] 39.28571
```

Read & Write Data

Function	Description
`read_csv()` or `read_csv2()`	coma or semicolon (CSV)
`read_delim()`	general separator
`read_table()`	whitespace-separated

Table 2: Functions to Read Data

Table & Vector Manipulation

The `dplyr` and `tidyr` packages provide us with a data manipulation grammar, a set of useful verbs to solve common problems

Function	Description
`mutate()` or `read_csv2()`	add new variables or modify existing ones
`select()`	select variables
`filter()`	filter
`summarise()` or `read_csv2()`	summarize/reduce
`arrange()`	sort
`group_by()`	grouped
`rename()`	rename column

Table 3: Functions to Read Data

Filter & Sort

```
ny_data <-
read_csv("data/bym_nyc_study.csv")%>%
filter(med_hhincome > 50000)%>%
arrange(-med_hhincome)
ny_data
```

Sessions

References I

Douglas, A., Roos, D., Mancini, F., Couto, A., & Lusseau, D. (2023). *An introduction to r*. https://intro2r.com/chap1.html

Roye, D. (2020). *A very short introduction to tidyverse*. https://dominicroye.github.io/en/2020/a-very-short-introduction-to-tidyverse/#pipe