Introduction to R and the Tidyverse

ISAIR 2025

Ahyoung Lim

LONDON SCHOOL *of* HYGIENE & TROPICAL MEDICINE

Learning objectives

By the end of Day 1, participants will be able to:

- Write, save, and execute **R scripts** to perform basic R operations
- Install and load **R packages**
- Create an **R project** in R studio and understand the concept of working directories
- **Load data** from a package or using `readr` package
- Understand the basics of **`ggplot2`** and create basic visualisations

What is R?

- **R** is a free and open-source programming language for statistical computing and graphics.
- **RStudio** is an integrated development environment or IDE which can be used for R programming.

Running commands in R

R requires typed instructions from its user:

- Interactively - Type an R command into the console and press enter
- Using a script - You can type the command into a script and either click Run or press ctrl+Enter
    - You can't save what you write in the console panel.
    - A script is a file where you can write a sequence of commands.
    - Scripts can be saved so you don't have to re-write code every time you re-open Rstudio.

Assignment operator (<–) assigns whatever is on the right of the arrow to the name of the object on the left of the arrow.

Good coding practice

Naming objects:

- R is case sensitive (e.g., `T` and `t` are different objects)
- Must not start with numbers or contain spaces
- Use meaningful object names and stick to using all lower case; if you need to use spaces, then you can use an underscore (_)

Formatting R scripts:

- Start your R script with a header (e.g., title, author, and date)
- Keep your code well-structured and group related commands (e.g., use section headers ending with four or more -, =, or # characters)
- Try to comment to help others or your future self to understand what your R code is trying to do

Exercise 1

1. Open RStudio
2. Type `4+5` into the console and press enter.
3. Type `a <- 5` and press enter.
4. Type `a` and press enter.
5. Type `a+6` and press enter.

Exercise 2

1. Open an R script
2. Type `55/9` in the script and click the run button
3. Put the cursor back on the line where you typed `55/9` and press Ctrl+Enter
4. Type `a <- 156` and run the line of code.
5. On a new line type `b` and then press Alt+- (Alt and the minus key) and see what happens in the script.
6. Assign `b` to a value of value of your choosing and run the line of code.
7. Type `# This is a comment` in the script. R will ignore anything after a number/hash key.
8. Press and hold Alt on the line with the comment and press the up or down arrow keys.

Getting help

- Use `help('command')` or `?command` to open the documentation of a function or dataset (e.g. ‘`?mean`’).
- The documentation usually provides information on:
    - How the function is meant to be used.
    - The arguments it can accept.
    - The value it returns.

Trouble shooting

- R will use errors, warnings, and messages to signal unintended behaviour and potential errors.

- Debugging your R code:
    - Check every “, (, { or [is closed, and that the letter case is correct.
    - Check the examples associated with the packages you are using.
    - Search online (e.g., stack overflow)

Using AI to learn R

1. Debug your R code:
    - The purpose of your code ("*I'm trying to do xyz*")
    - The code you used ("*This is the code I used…*")
    - The error you received ("*This is the error I received…*")
2. Ask what a function does/figure out what your R code does
3. Specify packages in your prompts ("*I'm trying to do xyz using dplyr…*")

💡 Don't rely on AI as your first step when your code doesn't work. Try debugging on your own first- it's an essential part of learning. ChatGPT is just one of many tools you can use to enhance your understanding.

⚠ ChatGPT might confidently provide incorrect answers, so always double-check the code independently and test it thoroughly.

Working directories

- `setwd(),getwd()`
- Rstudio Projects
- Why R project?

Installing and loading packages

```
install.packages("NameofPackage")  #  Installs a package for use later
library(NameofPackage)      # Loads the package to be used in this session
```

Data frames

Table 1: Example data frame

country	continent	year	lifeExp	pop	gdpPercap
Afghanistan	Asia	1952	28.801	8425333	779.4453
Afghanistan	Asia	1957	30.332	9240934	820.8530
Afghanistan	Asia	1962	31.997	10267083	853.1007

- Character: Text such as “Hello”
- Numeric: A number such as 5 or 1.34
- Logical: TRUE or FALSE
- Factor: Numbers that have text labels. Such as 0 = “Male”, 1 = “Female”
- Vectors: A one dimensional collection of objects all of the same type such as (TRUE, FALSE, TRUE, TRUE)

Loading data frames from a package

```r
library(gapminder)
data("gapminder")
```

Working with data frame

```
head(gapminder) # See the top 6 rows
tail(gapminder) # See the bottom 6 rows
names(gapminder) # See the variables names
summary(gapminder) # Summarise the data
str(gapminder) # See the structure of the object
class(gapminder) # See the class (In this case a data.frame)
```

Reading data into r

```
library(readr)


# Read in a csv file
dataname <- read_csv('PathtoData/data.csv’)

# Write a csv file
write_csv(name_of_object, path = 'PathtoData/data.csv')
```

Tidyverse and tidy data

- Tidyverse is a collection of R packages designed for data science

- Tidyverse relies on Tidy data:
  - Each variable forms a column.
  - Each observation forms a row.
  - Each value has it's own cell.

- Two main packages in the tidyverse for working with tidy data:
  - ggplot2 (grammar of graphics)
  - dplyr (grammar of data manipulation)

ggplot

- Grammar of graphics
  - geoms: describe the shapes that make up a plot
    - Scatter plot: `geom_point()`
    - Time series graph: `geom_line()`
    - Histogram: `geom_bar()`
  - `aes()` stand for aesthetics

- Useful resources:
  - R Graph Gallery https://www.r-graph-gallery.com/
  - R for Data Science https://r4ds.hadley.nz/
  - R Graphics Cookbook https://r-graphics

Making a graph

```
gm2007 <- filter(gapminder, year == 2007)

# Scatter plot
ggplot(gm2007) + # Define an empty plot with gm2007
    geom_point(mapping = aes(x = lifeExp, y = gdpPercap, col = continent))

# Bar chart
ggplot(gm2007) +
    geom_bar(mapping = aes(x = continent))
```

Saving graphs and themes

```
ggsave("outputs/scatterplot.png", plot = g1)
```

```
g1 + theme_cleveland()
g1 + theme_bw()
g1 + theme_dark()
```

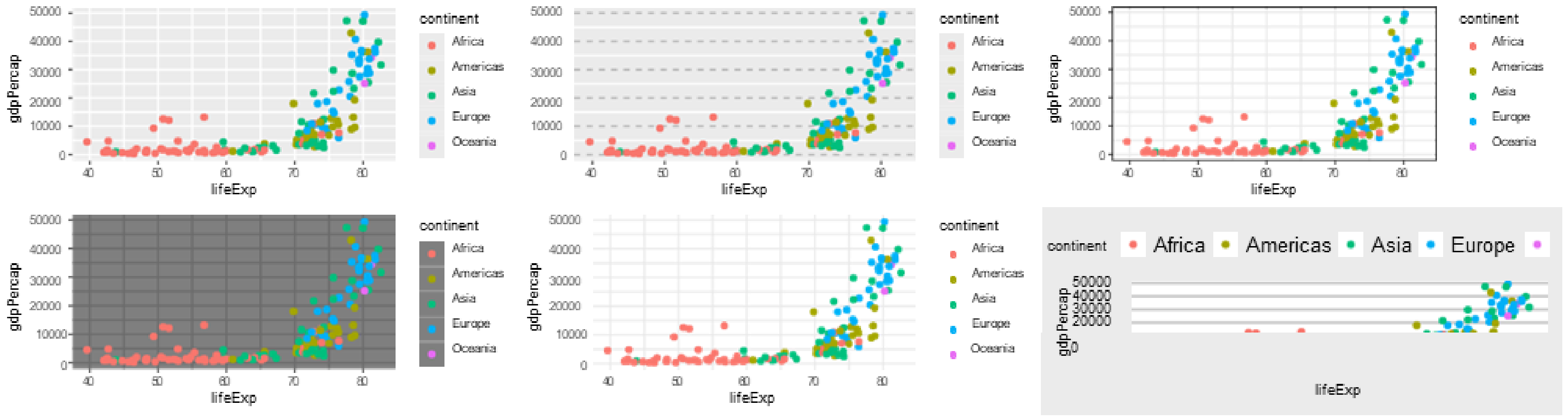


Figure 3: Themes in ggplot2

Practical for Day 1

💻 Exercises 3 to 5

📊 Create your own plot!

- Explore different ggplot features for customisation in the lecture notes and online resources.
- Use the gapminder data to create and save a new plot (ideally something different from the example plots in the lecture notes).
- In the breakout rooms (Day 2, 13:00-14:00), you will discuss:
    - The `ggplot2` features you used.
    - Why you chose particular customisations
    - Any challenges you faced

Exercise 3

1. Download the course data from the website if you haven't already and extract the folder onto your computer.
2. Click the drop down menu in the top right hand corner
3. Select new project
4. Select existing directory
5. Navigate to the course folder
6. Click create project
7. Close and Reopen Rstudio
8. Reload the project from the menu in the top right hand corner.

***Optional: Set up an R project in your existing directory and load one of your own datasets**

Exercise 4

1. Load the gapminder package
2. Load the gapminder data frame using `data("gapminder")`
3. Look at the top 6 and bottom 6 rows
4. Look at the structure of the data frame
5. Look at the names of the columns (variables) of the gapminder data

Exercise 5

1. Load the gapminder data.
2. Create an empty plot with the `ggplot()` function.
3. Add a line plot using `geom_line()`.
4. Inside `geom_line()` put `aes(x = year, y = lifeExp, group = country)`.
5. Run the command to produce a line graph of Life expectancy for the different countries.
6. After the `aes()` type `alpha = 0.2` this reduces the transparency of the colour.
7. Add `col = "white"` to the `geom_line` (Inside `aes()` or outside?).
8. Add `theme_dark()` to the end of the plot to change the overall style.

LONDON
SCHOOL of
HYGIENE
&TROPICAL
MEDICINE

Learning objectives

By the end of Day 2, participants will be able to:

- Apply the main verbs of `dplyr` to manipulate data efficiently.
- Use the `group_by()` and `summarise()` to group data and calculate summary statistics for each group.
- Understand and use the pipe operator (`%>%`) to chain multiple operations together
- Understand the basics of R Markdown

dplyr

- Grammar of data manipulation
- Key verbs/functions:
    - `select()` picks variables based on their names.
    - `filter()` picks rows based on their values. filter to get certain rows
    - `arrange()` changes the ordering of the rows.
    - `mutate()` adds new variables that are functions of existing variables.

Logical operators in R

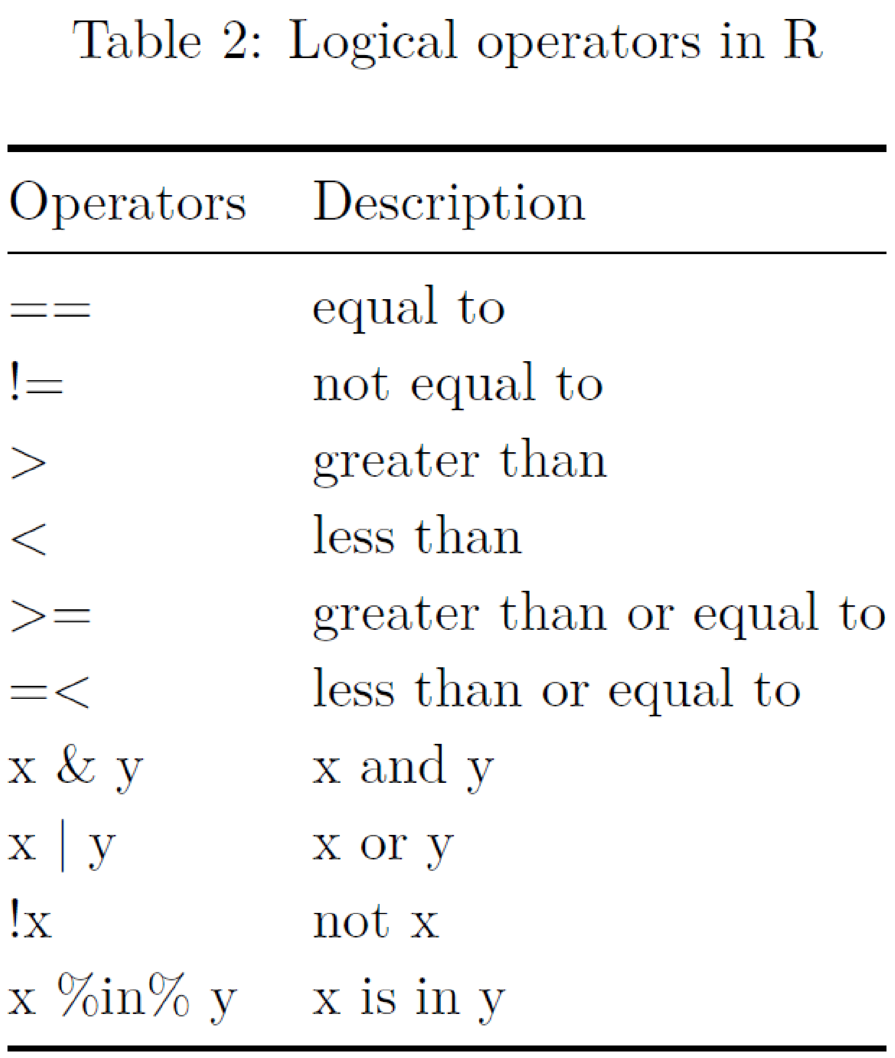


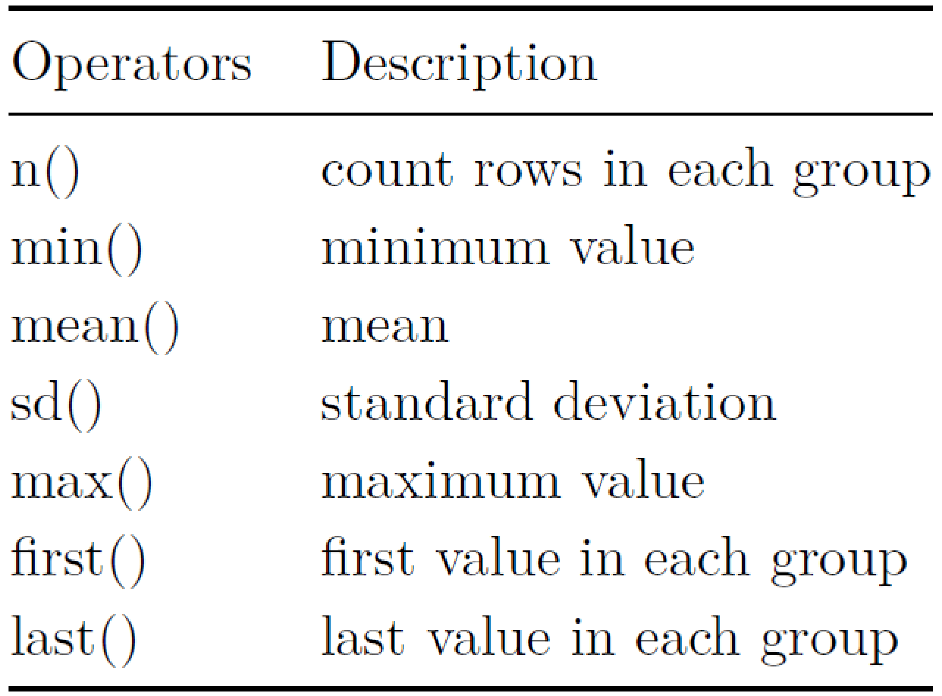
Table 2: Logical operators in R

Operators	Description
==	equal to
!=	not equal to
>	greater than
<	less than
>=	greater than or equal to
=<	less than or equal to
x & y	x and y
x | y	x or y
!x	not x
x %in% y	x is in y

Grouping data and calculating summaries of the groups

- `group_by()` groups the data by one or more variables
- `summarise()` summarises the data based on groupings

Table 3: Useful functions for summarise



Operators	Description
n()	count rows in each group
min()	minimum value
mean()	mean
sd()	standard deviation
max()	maximum value
first()	first value in each group
last()	last value in each group

Pipe operator

- In R a pipe is represented by the symbol `%>%`
- Shortcut to create `%>%` : ctrl+shift+m
- With the vocabulary of dplyr and more generally the `%>%` can be read as saying "*and then*."
- The pipe operator allows even more complex data manipulations to be performed in one function call.

```
# Piping
gapminder %>%
  group_by(year)  %>%
  summarise(meanLifeExp = mean(lifeExp), countries = n())

# Without piping
summarise(group_by(gapminder, year), meanLifeExp = mean(lifeExp), countries = n())
```

R Markdown

- Markdown is a document format that allows code and text to be combined together.
- The R Markdown document can be used to create reports and notebooks without having to copy and paste graphs or results to other documents.

Exercise 6

1. Select the country and continent column from the gapminder data frame
2. Filter the gapminder data to get years after 1980
3. Assign the data from step 2 to a new object called `gapminder_after_1980`
4. Sort `gapminder_after_1980` by `lifeExp`

Exercise 7

1. Type gapminder then create a pipe by pressing ctrl+shift+m
2. Press enter to go onto a new line
3. Group the data by year and continent then add a pipe and go to new line
4. Summarise by calculating the mean population
5. Run the entire command by highlighting it all and pressing run or ctrl+enter
6. Create your own piped command using dplyr verbs.

Practical for Day 2

- 💻 Exercises 6 and 7
- 🧠 Health coverage in Haiti