

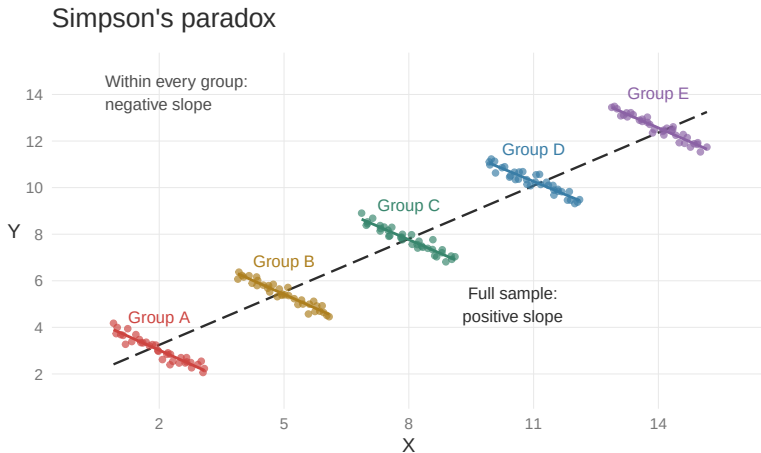
Introduction to R

Quantitative Methods in Political Economy
Pre-sessional Lecture 2

Wouter Leenders

wouter.leenders@kcl.ac.uk

A question from lecture 1



What is R?

- A language and software environment for statistical computing and graphics
- Free to download and use
- Open source: its code is available to inspect and extend

What is RStudio?

- An **integrated development environment** for working with R
- Brings together your code, results, data objects, and plots
- Available in free version

Resources for learning R

- Wickham, Çetinkaya-Rundel & Grolemund, *R for Data Science* (2nd ed.)
<https://r4ds.hadley.nz/>
- Stack Overflow
- YouTube
- AI: usual caveats apply...

Installing R and RStudio

Follow the step-by-step instructions on KEATS:

[Installing R and RStudio \(PDF\)](#)

Once both are installed, open RStudio to work with R.

The RStudio interface

The usual four-pane layout:

- **Top left: Source**
Write and edit scripts containing your commands
- **Bottom left: Console**
Run commands and read output or error messages
- **Top right: Environment**
See the data and objects you have created
- **Bottom right: Files, Plots, Packages, Help**
Browse files, inspect plots, and read documentation

Running your first command

Type this into the Console and press Enter:

```
1 + 1
```

Now open a script using **File > New File > R Script** and type the same command.

- Place the cursor on the line and click **Run**
- Windows/Linux: Ctrl + Enter. Mac: Cmd + Return

Save your script as an `.R` file so you can **reproduce your work**.

Creating objects in R

R stores values in **objects**. Use `<-` to assign a value to a name:

```
x <- 2
```

This creates an object called `x` with the value 2.

To display its value, run either command:

```
x  
print(x)
```

You can also use `x = 2` for assignment. We will use `<-`.

Using objects in calculations

Once an object exists, use its name in a calculation:

```
x + 3
```

R uses the value currently stored in x. Here, the result is 5.

Create another object:

```
new_object <- -10
```

Object names are **case-sensitive**: x and X are different names.

Arithmetic with objects

Try these calculations:

```
x / new_object      # Divide
x - new_object      # Subtract
x * new_object      # Multiply
x ^ 3                # Raise to a power
```

Use parentheses to control the order of operations:

```
(3 - x) * 4
3 - x * 4
```

Text, copied values, and missing values

An object can hold text. Put text inside quotation marks:

```
z <- "this is the letter Z"
```

Assign the current value of one object to another:

```
object_2 <- x
```

R represents a **missing value** using NA:

```
p <- NA
```

Exercise: objects and calculations

1. Run `x <- 10`, then `x / new_object`.
Why has the result changed?
2. Create `long_number` with the value `3245 / 899`.
3. Create `long_number_100` with the value `long_number * 100`.
Why might saving this result be useful?

Exercise: parentheses and object names

1. Create `y` with the value `(3 - x) * long_number`.
What changes if you remove the parentheses?
2. Try running `new_object <- 2`.
Why does this command fail, while `new_object <- 2` works?

Compare your results with the person next to you.

Vectors

A **vector** stores several values in a single object.

For example, Singapore's latitude and longitude are approximately:

```
location <- c(1.290270, 103.851959)
```

Use `c()` to combine values into a vector:

```
a <- c(0, 1, NA, 10, -1.2)
```

a

The vector `a` contains five values, including one missing value.

Sequences and vector calculations

Use `:` to create a sequence of consecutive integers:

```
seq_values <- 8:91
```

Apply a calculation to every value in a vector:

```
a * 3  
# [1]  0.0  3.0  NA 30.0 -3.6
```

The missing value stays missing.

This is useful when performing the same calculation on many observations.

Character vectors and vector length

Vectors can also contain text:

```
word_vector <- c("red", "red", "white", "purple")  
word_vector
```

Use `length()` to count the number of elements:

```
length(word_vector)  
# [1] 4
```

Repeated values count as separate elements.

Exercise: vectors

1. Create `combo <- x + a`.
Are the resulting values what you expected?
2. Try `word_vector * a`.
Why does this fail?
3. Try `seq_values / a`.
What do you notice about the results and the warning?
4. Create `mix <- c(2, x, "help")`.
What happens to the numbers, and why?

Combining vectors

Use `c()` to join the elements of existing vectors:

```
a_seq_combined <- c(a, seq_values)
```

The new vector contains all the values of `a`, followed by all the values of `seq_values`.

Check its length:

```
length(a_seq_combined)
```

Creating data frames

We often import an existing data set, but we can also create one in R.

Start by creating a vector for each column:

```
ID <- c(1, 2, 3, 4)
```

```
Name <- c("Martha", "Tim", "Rob", "Georgia")
```

```
Maths <- c(87, 91, 97, 95)
```

```
Science <- c(83, 99, 84, 76)
```

Each vector contains information about the same four students, in the same order.

Combining columns into a data frame

Use `data.frame()` to combine the columns:

```
students_df <- data.frame(ID, Name, Maths, Science)
students_df
```

This keeps the marks numeric and the names as text.

Click `students_df` in the Environment pane, or run:

```
View(students_df)
```

A data frame in R

ID	Name	Maths	Science
1	Martha	87	83
2	Tim	91	99
3	Rob	97	84
4	Georgia	95	76

- Each **row** is an observation: one student
- Each **column** is a variable: a characteristic of that student

Exercise: creating a data set

Create a data frame from **John Snow (1855), Table IX**, covering the first seven weeks of the 1854 cholera epidemic in London.

Water-supply group	Houses	Cholera deaths
Southwark & Vauxhall	40,046	1,263
Lambeth	26,107	98
Rest of London	256,423	1,422

- Create three vectors named `supply_area`, `houses`, and `deaths`
- Combine them into a data frame named `cholera_data`
- Inspect the result in RStudio

Packages in R

Packages add functions to R. We will use `ggplot2` to create graphs.

Install the package once:

```
install.packages("ggplot2")
```

Load it in each new R session when you want to use it:

```
library(ggplot2)
```

Checking a variable's type

Before plotting, check that `deaths` contains numbers:

```
str(cholera_data)
is.numeric(cholera_data$deaths)
```

The `$` symbol selects a column from a data frame.

If the column contains numbers stored as text, convert it:

```
cholera_data$deaths <- as.numeric(cholera_data$deaths)
```

Numeric columns stay numeric when created directly with `data.frame()`.

Calculating deaths per 10,000 houses

The groups contain different numbers of houses. Compare deaths relative to the number of houses:

```
cholera_data$deaths_per_10000 <-  
  cholera_data$deaths / cholera_data$houses * 10000
```

This creates a new column in `cholera_data`.

- Southwark & Vauxhall: approximately 315.4
- Lambeth: approximately 37.5
- Rest of London: approximately 55.5

Creating a bar chart with ggplot2

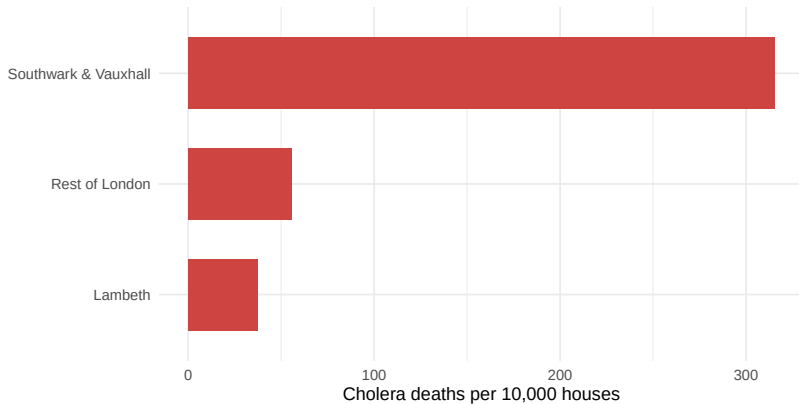
Compare deaths per 10,000 houses across water-supply groups:

```
cholera_plot <- ggplot(cholera_data,  
  aes(x = deaths_per_10000, y = supply_area)) +  
  geom_col(fill = "#CD4440", width = 0.65) +  
  labs(x = "Cholera deaths per 10,000 houses",  
       y = NULL) +  
  theme_minimal(base_size = 16)
```

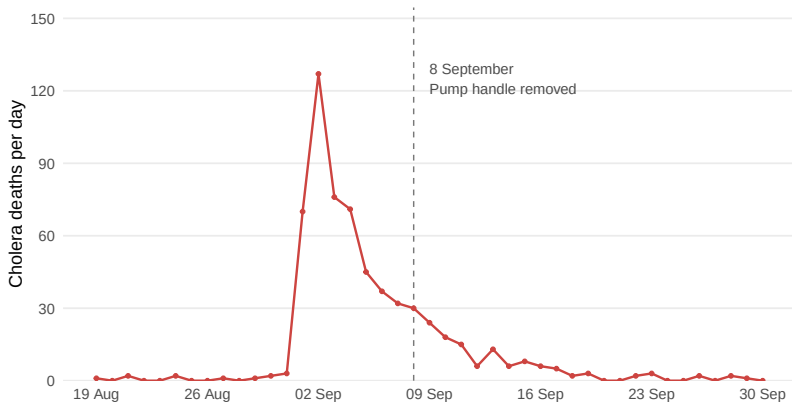
cholera_plot

geom_col() draws bars using the values already in the data.

Cholera mortality and water supply



Cholera deaths over time: Soho, 1854



Data: Snow (1855), Table I

John Snow's map: Soho, 1854



- Dots: mapped deaths
- Triangles: water pumps
- Largest: Broad Street

Source: Snow (1855), Map I