

Quantitative Methods for Political Economy

Seminar 1: R fundamentals and workflow

Friday 9 October 2026

Wouter Leenders

wouter.leenders@kcl.ac.uk

Before we begin...

Does everyone have R and RStudio installed?

Installation instructions: [Installing R and RStudio](#)

Open RStudio now and let me know if you need help.

Start with a fresh session

Work in pairs. Begin with a fresh R session.

Your turn

- Use Session > Restart R.
- Find the Console (results) and Environment (objects).
- Check that the Environment is empty.

Scripts

Create a new R script using File > New File > R Script. Save it as 01-seminar.R. Throughout the seminar, write code in the script editor and run it from there.

Your turn

- Type these calculations on three separate lines:

```
2 + 2  
5 * 3  
2^4
```

- Place your cursor on a line and press Cmd+Enter (Mac) or Ctrl+Enter (Windows). Read the result in the Console.
- Save your changes with Cmd+S (Mac) or Ctrl+S (Windows).
- As you continue, add each new working example below the previous one.

Recap: objects, vectors and missing values

```
study_hours <- c(6, 8, NA, 10)
study_hours * 2
length(study_hours)
mean(study_hours)
```

Your turn

- Predict the results before running.
- Why does the missing value not become zero?
- Which command counts all four entries?

Recap: observations, variables and columns

```
students <- data.frame(  
  student_id = c("A", "B", "C"),  
  study_hours = c(6, 8, 10)  
)  
students  
students$study_hours
```

Your turn

- In your data frame, what is an observation? And what is a variable?
- Explain what \$ does. What does `students$study_hours` return?

Recap: packages

What is the difference between `install.packages("ggplot2")` and `library(ggplot2)`?

Which one is needed again after restarting R?

- Installation adds a package to your computer
- Loading a package makes it available in the current session

Functions and arguments

```
seq(from = 2, to = 10, by = 2)
mean(c(6, 8, NA))
mean(c(6, 8, NA), na.rm = TRUE)
```

A function has a name (`seq`) and arguments (`from`, `to`, `by`).

If you omit an argument with a default value, R uses that default. What is the default for `mean()`'s `na.rm`?

NA records a missing value. This is not the same as a zero!

Your turn

- Use Tab to complete `seq` as you type the first line in your script.
- Type `?seq` in your script and run it to open the help page for `seq()`.
- Explain what `na.rm` does.

Incomplete expressions

```
label <- "seminar
```

An unmatched quote or bracket leaves an expression incomplete.

If you run an incomplete expression from your script, the Console shows “+”. Press Escape to cancel, then complete the command in your script.

Your turn

- Identify the missing quote. Type the completed expression in your script and run it.
- Leave only complete commands in the saved script.

Exercise: repair and explain

Code to check

```
weekly_hours <- c(4, 6, 8)
mean(weekly_hours
Weekly_hours * 10
mean(c(4, 6, NA))
```

Your turn

- Work in pairs: check each line and repair it if needed before adding it to your script and running it.
- Decide whether to keep or omit missing values in the fourth line.

Exercise: repair and explain

Code to check

```
weekly_hours <- c(4, 6, 8)
mean(weekly_hours
Weekly_hours * 10
mean(c(4, 6, NA))
```

Corrected code

```
weekly_hours <- c(4, 6, 8)
mean(weekly_hours)
weekly_hours * 10
mean(c(4, 6, NA)) # Returns NA
# Or: average observed values (5)
mean(c(4, 6, NA), na.rm = TRUE)
```

The fourth line is valid as written.

Your turn

- Work in pairs: check each line and repair it if needed before adding it to your script and running it.
- Decide whether to keep or omit missing values in the fourth line.

Scripts make your analysis replicable

A saved script records commands in order, so you and others can repeat the analysis.

Running the script recreates the objects it needs. External data files must be saved separately.

Add this mean below the vector exercise in 01-seminar.R:

```
mean(study_hours, na.rm = TRUE)
```

Your turn

- Save the script and run the new line.
- Read from top to bottom: is `study_hours` created before it is used?

One analysis, one project

Script: a .R file containing the commands for your analysis.

Project: a folder organising your scripts, input data and outputs.

Create seminar-1: File > New Project > New Directory > New Project.

Your turn

- Reopen 01-seminar.R. Use File > Save As to save it in this folder.
- Create data and output here.
- Copy both supplied CSV files into data.
- Run `getwd()` from your script.

Project folder

```
seminar-1/  
  seminar-1.Rproj  
  01-seminar.R  
  data/  
    survey.csv  
    turnout.csv  
  output/
```

Opening `seminar-1.Rproj` opens the project and sets its folder as the working directory.

Project setup checklist

Work through these checks with your partner before continuing.

Project and session

- **Did I create the project?** The seminar-1 folder should contain seminar-1.Rproj.
- **Where is it?** Find the project folder in Finder/File Explorer. Make sure the Files pane shows that folder too.
- **Am I working in the project?** The project menu at the top right should show seminar-1. Open its .Rproj file if needed.
- **Is R using the right folder?** Run `getwd()` from your script: it should give the path to your project folder.

Files and folders

- **Where is my script saved?** Save 01-seminar.R in the project folder, beside seminar-1.Rproj. Use File > Save As to choose that location.
- **Where are my input data?** The project's data folder should contain survey.csv and turnout.csv.
- **Where will my results go?** The output folder should also be inside the project. It can be empty now; exported results will go there later.

Paths within the project

Add these checks to your script and run them:

```
getwd()  
file.exists("data/survey.csv")  
file.exists("data/turnout.csv")
```

`data/survey.csv` is relative to the project folder.

Keep personal absolute paths out of shared scripts. Open the project to establish the working directory.

Your turn

- Expect TRUE for both files.
- If you see FALSE, check the project folder, filename and location of the supplied CSV.

Restart and rerun

In Tools > Global Options > General:

- Turn off restoring .RData at startup.
- Set saving the workspace on exit to Never.

Only objects created by commands in your script will return after restarting.

Your turn

- Save your script, then use Session > Restart R.
- Select all code in your script: Cmd+A (Mac) or Ctrl+A (Windows).
- Run it with Cmd+Enter (Mac) or Ctrl+Enter (Windows).
- Check that `study_hours` reappears in Environment.

Checkpoint: what is saved?

1. What is the difference between a script and an object?
2. Where will `output/results.csv` be written?
3. Will another student need your username in their file path?

Your turn

- Compare answers and help a neighbour restart successfully.

Packages: install once, load each session

Put this at the top of your script. A line starting with # is a comment; R skips it.

```
# install.packages("tidyverse")  
library(tidyverse)
```

Your turn

- If tidyverse is not installed, remove # from the first line, run it once, then put # back.
- Run `library(tidyverse)` to load the packages.
- Installation may take longer; pair up while it finishes.

CSV: a small survey

Synthetic teaching data: these are invented responses.

```
id,country,study_hours  
001,UK,8  
002,France,.  
003,Germany,12  
004,UK,10
```

The first line supplies names; commas separate fields.

Your turn

- Count observations and variables.
- What might the dot represent?

Import and inspect

Add this code below the existing code in 01-seminar.R, then run it:

```
survey <- read_csv("data/survey.csv")  
survey  
glimpse(survey)
```

Your turn

- Read the import message.
- Is study_hours numeric?
- What type is id? Were its leading zeros retained?

Tell R how to read the columns

Replace the survey import in your script with this version:

```
survey <- read_csv(  
  "data/survey.csv",  
  na = c("", "NA", "."),  
  col_types = cols(  
    id = col_character(),  
    country = col_character(),  
    study_hours = col_double()  
  )  
)
```

Your turn

- Run the revised import, then run `glimpse(survey)` again.
- Why specify `id` as character even if R already guessed correctly?

A warning needs investigation

Try this example in your script; the data are embedded in the code.

```
bad_csv <- "id,hours\n001,eight\n002,10"
bad <- read_csv(I(bad_csv),
  col_types = cols(
    id = col_character(), hours = col_double()
  ))
problems(bad)
```

Your turn

- Identify the failed value and check it in the source text `bad_csv`.
- Afterwards, select this example and comment it out with `Cmd+Shift+C` (Mac) or `Ctrl+Shift+C` (Windows).

This warning is intentional. `I()` marks literal CSV text; `\n` starts a new line.

Exercise: missing is not zero

Add these checks below the corrected survey import in your script:

```
survey$study_hours  
mean(survey$study_hours)  
mean(survey$study_hours, na.rm = TRUE)  
problems(survey)
```

Your turn

- Explain each result.
- How many responses enter the final mean?

Exercise: missing is not zero

Add these checks below the corrected survey import in your script:

```
survey$study_hours  
mean(survey$study_hours)  
mean(survey$study_hours, na.rm = TRUE)  
problems(survey)
```

Your turn

- Explain each result.
- How many responses enter the final mean?

Expected mean: 10 hours across three observed responses.

The first mean is missing; the corrected import should have no parsing problems.

Headers, metadata and delimiters

Add these examples to your script; each small dataset is embedded in the code.

```
read_csv(I("Survey export\nx,y\n1,2"), skip = 1)
read_csv(I("1,2\n3,4"),
         col_names = c("x", "y"))
read_delim(I("x|y\n1|2"), delim = "|")
```

Your turn

- Match each argument to a feature of the input.

For tabs use `read_tsv()`. `read_csv2()` expects semicolon separators and comma decimals.

Column names with spaces

Add this example to your script and inspect the names after each step.

```
example <- read_csv(I("Study hours\n8\n10"))
names(example)
example$`Study hours`
example <- rename(example, study_hours = `Study hours`)
names(example)
```

Backticks identify an unusual column name. Renaming once makes later code easier to read.

Your turn

- Compare the two outputs of `names(example)`.

Write data to disk

Add this code to your script after the survey checks:

```
write_csv(survey, "output/survey-clean.csv")
check <- read_csv("output/survey-clean.csv",
  col_types = cols(id = col_character()))
check
nrow(check)
check$id
```

Your turn

- Find `survey-clean.csv` in your project's output folder.
- Confirm four rows and IDs 001, 002, 003, 004.

CSV stores values, not R's full type information.

Independent import challenge

Open the supplied `data/turnout.csv` in a text editor to inspect it.

It contains invented turnout percentages. The word `missing` denotes an unobserved value.

Your turn

- At the end of your script, import the file with numeric turnout and a recognised missing value.
- Inspect it with `glimpse()`, `check_problems()`, and export it to `output/turnout-clean.csv`.
- Use your survey import as a guide. Save the script and explain each argument you change.

Import challenge: one solution

Compare this with the turnout code at the end of your script:

```
turnout <- read_csv("data/turnout.csv",  
  na = c("", "NA", "missing"),  
  col_types = cols(  
    area = col_character(), turnout = col_double()  
  ))  
glimpse(turnout)  
problems(turnout)  
write_csv(turnout, "output/turnout-clean.csv")
```

Three rows; turnout is 72.5, NA and 68.0. No parsing problems are expected.

Import checklist

- Is the path correct?
- Are headings and separators correct?
- Do column types and missing values make sense?
- Are there parsing problems?
- Does the script preserve each decision?

Your turn

- Use this checklist on your survey import.

Final test

Your turn

- Save your script and restart R using Session > Restart R.
- Select all code in your script: Cmd+A (Mac) or Ctrl+A (Windows).
- Run it with Cmd+Enter (Mac) or Ctrl+Enter (Windows).

Check that both clean CSV files are recreated in output.

Explain to your neighbour:

- how the script finds the input file;
- why the ID is character and the dot becomes missing;
- how you know the analysis can be repeated.