Data Import

PSY 410: Data Science for Psychology

Dr. Sara Weston

2026-04-15

Getting real data into R

Your data isn't inside a package

So far we've used datasets that were already loaded in R — `mpg`, `flights`, `bfi`.

. . .

But your thesis data isn't sitting inside a package. It's a CSV on your desktop, an Excel file from your advisor, or a Qualtrics export with 200 columns named `Q1`, `Q2`, `Q3`…

. . .

Today we learn how to get it into R — and what to check when things don't import cleanly.

Reading CSV files

read_csv() — your workhorse

```
# The basic pattern
my_data <- read_csv("path/to/file.csv")
```

- Part of the `readr` package (loaded with tidyverse)
- Returns a **tibble**, not a base R data frame
- Smarter about guessing column types than `read.csv()`

Why not read.csv()?

	`read_csv()`	`read.csv()`
Package	readr (tidyverse)	base R
Returns	tibble	data.frame
Speed	Much faster	Slower
Type guessing	Better	Converts strings to factors
Consistent	Yes	Depends on options

Always use `read_csv()` in this class.

What read_csv() does automatically

```
# It figures out column types from the first 1000 rows
my_data <- read_csv("survey.csv")

# It tells you what it decided:
glimpse(my_data)
```

Types it recognizes: `dbl` (number), `chr` (text), `lgl` (TRUE/FALSE), `date`, `dttm` (date-time)

When the guess is wrong

Sometimes R guesses wrong — especially with:

- ID numbers (should be character, not numeric)
- Zip codes, phone numbers
- Dates in unusual formats
- Columns that are mostly empty

```
# Check for problems after reading
my_data <- read_csv("survey.csv")
problems(my_data)  # Shows any parsing issues
```

Specifying column types

```r
# Tell read_csv() what types to use
my_data <- read_csv("survey.csv",
  col_types = cols(
    participant_id = col_character(),  # Don't treat as number
    age = col_integer(),
    gender = col_character(),
    score = col_double()
  )
)
```

col_types shortcuts

Shortcut	Type
`"c"`	character
`"d"`	double
`"i"`	integer
`"l"`	logical
`"D"`	date
`"?"`	let R guess

```r
# Compact version
read_csv("survey.csv", col_types = "cdiccc?")
```

Handling messy headers

Real data often has ugly column names:

```r
# Skip rows at the top (metadata, notes)
read_csv("messy.csv", skip = 2)

# Only read specific rows
read_csv("messy.csv", n_max = 100)

# Rename columns after import
my_data <- read_csv("messy.csv") |>
  rename(age = `Age (years)`, score = `Total Score`)
```

Missing values

By default, `read_csv()` recognizes these as NA:

`NA`, `N/A`, `NaN`, `""` (empty), `"."`, `"NULL"`

```
# Add your own NA strings
read_csv("survey.csv", na = c("NA", "N/A", "-999", ""))
```

Psychology datasets often use `-999` or `99` as missing value codes!

Writing CSV files

```
# Save your cleaned data
write_csv(my_cleaned_data, "data/cleaned_survey.csv")
```

Save intermediate cleaned files — don't re-clean every time you start R.

Pair coding break

Your turn: 10 minutes

I've created a messy CSV file for you. Download it here. With a partner:

1. Read it back in with `read_csv()` — what types does R guess?
2. Fix the missing values: `-999` and `"N/A"` should be `NA`
3. Rename the columns to clean snake_case names
4. What does `problems()` show you?

Before we move on

Upload your code to Canvas for participation credit. Paste what you have into today's in-class submission — it doesn't need to work perfectly.

Excel files

readxl::read_excel()

Excel is everywhere in psychology. The `readxl` package handles it:

```
library(readxl)  # Or just use readxl:: directly

# Read the first sheet
my_data <- read_excel("survey_results.xlsx")

# Read a specific sheet
my_data <- read_excel("survey_results.xlsx", sheet = "Demographics")

# Read a specific range
my_data <- read_excel("survey_results.xlsx",
  sheet = "Scores",
  range = "A1:F50"
)
```

Excel pitfalls

Things that look fine in Excel but break in R:

- **Merged cells** — R sees them as one value + empty cells
- **Colors as data** — R can't read cell colors
- **Multiple tables on one sheet** — use `range` to isolate them
- **Formatted numbers** — "1,234" might read as text

> Tip
>
> The best fix is usually to clean the Excel file first, or use `range` to read just the clean part.

Listing sheet names

```
# See what sheets are in a file
excel_sheet_names("survey_results.xlsx")
```

Useful when you don't know the file structure yet.

Other file formats

SPSS and SAS files

Psychology labs often have data in SPSS (`.sav`) or SAS formats:

```
library(haven)

# Read SPSS
spss_data <- read_sav("study.sav")

# Read SAS
sas_data <- read_sas("study.sas")
```

These return tibbles, just like `read_csv()`.

A note about SPSS labels

SPSS files often have **variable labels** and **value labels**:

```
# Labels are stored as attributes
attr(spss_data$gender, "label")   # "Participant Gender"
attr(spss_data$gender, "levels")  # 1="Male", 2="Female"

# To drop labels and work with plain values:
library(haven)
spss_data <- zap_labels(spss_data)
```

Getting data out of Qualtrics

The most common workflow in psychology:

1. In Qualtrics: **Download** → **CSV** (or use the API — advanced)
2. The export includes metadata rows at the top — skip them:

```
# Qualtrics CSVs typically have 3 header rows
qualtrics_data <- read_csv("my_survey_export.csv", skip = 2)
```

3. Clean up the automatic column names (they're ugly)
4. Handle the timestamp columns if you need them

> **Note**
>
> The first two rows after the header contain Qualtrics' own descriptions. `skip = 2` gets past them. Always check with `glimpse()` after importing.

Putting it together

A realistic import workflow

```
# 1. Read the file
raw_data <- read_csv("qualtrics_export.csv",
  skip = 2,
  na = c("", "N/A", "-999")
)

# 2. Check what you got
glimpse(raw_data)
problems(raw_data)

# 3. Rename columns
clean_data <- raw_data |>
  rename(
    id = ResponseId,
    age = Q1,
    condition = Q2
  )

# 4. Fix types if needed
clean_data <- clean_data |>
  mutate(
    id = as.character(id),
    age = as.numeric(age)
  )

# 5. Save the cleaned version
write_csv(clean_data, "data/clean_survey.csv")
```

Get a head start: Assignment 3

Try this mini-challenge before next class:

```
# 1. Import the messy CSV from the pair exercise
messy <- read_csv("data/messy_survey.csv", na = c("", "NA", "-999", "N/A"))

# 2. Check for problems
problems(messy)

# 3. Fix at least one column type issue with col_types or
#    readr::parse_number()

# 4. Save your cleaned version
write_csv(clean, "data/clean_survey.csv")
```

Assignment 3 will have you do this end to end with real messy data.

Solution

Wrapping up

The import toolkit

Function	What it does
`read_csv()`	Read CSV files
`write_csv()`	Save CSV files
`read_excel()`	Read Excel files
`read_sav()`	Read SPSS files
`problems()`	Check for import issues
`col_types`	Control column types

Before next class

Read:

- R4DS Ch 9: Layers

Practice:

- Try importing a CSV or Excel file you have lying around
- Check `problems()` — does it flag anything?
- Practice renaming messy column names

Key takeaways

1. **`read_csv()` is your default** — it's faster and smarter than `read.csv()`
2. **Check your imports** — `glimpse()` and `problems()` are your friends
3. **Name your missing values** — psychology data loves `-999`
4. **Excel is tricky** — use `readxl` and be suspicious of formatting
5. **Save cleaned files** — don't re-clean every time

The one thing to remember

Your analysis is only as good as your import. Check it before you trust it.

Next time: Layers & Aesthetics