2. Downloading and loading data

A public clinical dataset, checked before analysis

Contents

Read the baseline CSV . . . 1
Where R looks for files . . . 2
Check the size and the unit of observation . . . 2
Inspect types, missing values, and ranges . . . 2
Set the category order . . . 3
A second file: one row per patient and visit . . . 3
Try it: count the vision groups . . . 3
Useful `read.csv()` arguments . . . 4
Common errors when loading data . . . 4
Optional: other file formats . . . 4
- Built-in practice datasets . . . 4
- Semicolon and tab-separated files . . . 5
- Try it: read the semicolon file . . . 5
- R's own format: `.rds` . . . 5
- Install the extra packages . . . 6
- A faster CSV reader: readr . . . 6
- Excel workbooks . . . 6
- SPSS, Stata, and SAS files . . . 6
- From a web address . . . 7
- Other sources . . . 7

A data frame stores variables in columns and observations in rows. We'll read the prepared CSVs from the Moorfields study by Fasler and colleagues (2019), then check what one row represents. The data notes document their preparation.

Read the baseline CSV

```
# Store the imported data frame as cohort
cohort <- read.csv("data/moorfields_baseline.csv", na.strings = c("", "NA"))

# Pick columns by name; c() combines the names
head(cohort[c("age", "sex", "baseline_va", "baseline_vision")], 3)
```

`read.csv()` imports the file; `<-` assigns the result to `cohort`. `na.strings` treats empty fields and `"NA"` as missing. `head(..., 3)` shows three rows of the selected columns. Change `3` to `10` and compare.

Each row is one patient with an exact day-zero visual acuity measurement. The browser already has this file, so no upload is needed.

Column	Meaning
`patient_id`	Published deidentified ID, used to check or join rows
`age`	Recorded baseline age in years, coarsened for privacy
`sex`	Recorded sex, Female or Male
`baseline_va`	ETDRS letter score; higher means better visual acuity
`baseline_vision`	Poor (35), Intermediate (>35 and <70), or Good (70 letters)

In the source, ages at or below 65 are recorded as 60, and ages over 90 as 99. These clusters reflect coarsening, not exact patient ages. We therefore label summaries **recorded baseline age**.

For local R, download moorfields_baseline.csv and moorfields_followup.csv into your Project's `data/` folder.

Where R looks for files

The **working directory** is the starting folder for a relative path such as `"data/moorfields_baseline.csv"`.

```
getwd() # Folder R uses for relative paths
list.files("data")
file.exists("data/moorfields_baseline.csv") # Check the path before importing
```

The last command should return `TRUE`. Change `.csv` to `.cvs`: what happens? In the browser, these files are in a temporary file system. Locally, open the Project so R uses its folder.

Check the size and the unit of observation

```
dim(cohort)
anyDuplicated(cohort$patient_id) # 0 means no repeated patient IDs
```

The prepared cohort has 3,355 rows and five columns, with no repeated IDs. The original data contain 3,357 patients; two lack an exact day-zero measurement. Repeated visit rows would give frequent attenders extra weight in a patient summary.

Inspect types, missing values, and ranges

```
str(cohort)
colSums(is.na(cohort)) # Missing values in each column
summary(cohort[c("age", "baseline_va")])
table(cohort$baseline_vision, useNA = "ifany")
```

`str()` shows each column's type. A column expected to be numeric but read as text needs investigation before analysis.

This prepared file has no missing values. That doesn't make the original records or later visits complete. Visual acuity is a letter-chart score, not a percentage; check its range against the data notes.

Question: which command above counts missing values, and which counts patients in each vision group? Try `summary(cohort$age)` as another check.

Set the category order

```
# Set the order used in tables and figures
cohort$sex <- factor(cohort$sex, levels = c("Female", "Male"))
cohort$baseline_vision <- factor(
  cohort$baseline_vision,
  levels = c("Poor", "Intermediate", "Good")
)

table(cohort$baseline_vision)
```

A factor stores categories in a chosen order. Without these levels, text would sort alphabetically, with Good before Poor. `$` selects one column.

A second file: one row per patient and visit

```
followup <- read.csv("data/moorfields_followup.csv", na.strings = c("", "NA"))
followup$timepoint <- factor(
  followup$timepoint,
  levels = c("Baseline", "Year 1", "Year 2")
)

table(followup$timepoint)
```

Here a patient can have a baseline row and selected year-one and year-two rows. Annual observations are the nearest valid visits to 52 and 104 weeks, within eight weeks on either side. A missing visit contributes no row; it isn't a zero acuity score.

This file illustrates repeated observations. Table 1 uses the one-row-per-patient baseline file; part 4 switches to the Ghana sepsis data.

Try it: count the vision groups

Fill the blank with the grouping column's name.

```
table(cohort$______, useNA = "ifany")
```

> **Hint**
>
> Use `baseline_vision`. `$` selects a column.

Solution.

> **Solution**
>
> ```
> table(cohort$baseline_vision, useNA = "ifany")
> ```
>
> There are 451 patients in Poor, 2,071 in Intermediate, and 833 in Good.

Useful `read.csv()` arguments

A small preview with text IDs

```
preview <- read.csv(
  "data/moorfields_baseline.csv",
  na.strings = c("", "NA"),
  nrows = 5, # Read a small preview
  colClasses = c(patient_id = "character") # Keep IDs as text
)

str(preview)
```

`nrows` limits the import to five rows; `colClasses` keeps the ID as text. This preserves leading zeros such as `"00123"`. Change `5` to `20`, then check `dim(preview)`.

Common errors when loading data

Troubleshooting an import

- File not found: use `getwd()` and `list.files("data")`; check spelling and hidden extensions such as `results.csv.csv`.
- `'\U' used without hex digits`: replace Windows backslashes (`"C:\Users\..."`) with forward slashes (`"C:/Users/..."`).
- Everything in one column: check the separator; try `read.csv2()` below.
- Numbers read as text: inspect entries with `unique()` or `table()`; values such as `"<5"`, `"n/a"`, or spaces may need cleaning or `na.strings`.
- Garbled accents: try `fileEncoding = "UTF-8"` or `"latin1"` for older exports.
- Excel dates read as numbers such as 45231: readxl usually converts them; otherwise use `as.Date(x, origin = "1899-12-30")`.

Optional: other file formats

The following examples write a copy of `cohort`, then read it back. Run **Set the category order** first. With your own files, start at the reading step. The CSVs already bundled with this page are sufficient for the seminar.

Built-in practice datasets

Example

R also includes `mtcars` and `iris` for practice.

```
head(mtcars, 3)
```

Semicolon and tab-separated files

Example and exercise

The unmodified Moorfields source uses semicolons. Our prepared CSVs use commas. `read.csv2()` expects semicolons with decimal commas; `read.delim()` reads tab-separated files. A different separator can be set with `read.csv(file, sep = "|")`.

```
# Make a semicolon-separated copy for this example
write.csv2(cohort, "cohort_semicolon.csv", row.names = FALSE)
readLines("cohort_semicolon.csv", n = 3)

semicolon <- read.csv2("cohort_semicolon.csv")
head(semicolon, 3)
```

`readLines()` shows the raw separators. Compare it with the imported rows.

Try it: read the semicolon file

The exercise setup writes `cohort_semicolon.csv` for you. Fill the blank with the reader for that separator.

```
semicolon <- ______("cohort_semicolon.csv")

dim(semicolon)
```

Hint

The semicolon reader's name has one extra character after `read.csv`.

Solution.

Solution

```
semicolon <- read.csv2("cohort_semicolon.csv")

dim(semicolon)
```

`read.csv2()` uses `sep = ";"` and `dec = ","` by default. Those arguments also work with `read.csv()`.

R's own format: `.rds`

Compare RDS with CSV

```
# RDS retains the object, including factor levels
saveRDS(cohort, "cohort.rds")
from_rds <- readRDS("cohort.rds")

write.csv(cohort, "cohort_copy.csv", row.names = FALSE)
from_csv <- read.csv("cohort_copy.csv")

levels(from_rds$baseline_vision)
class(from_csv$baseline_vision)
```

RDS retains the object and its factor levels. CSV returns the category as text, so its order must be set again. Use RDS between R scripts and CSV when sharing with other software.

Install the extra packages

Packages for the remaining examples

```
webr::install(c("readr", "readxl", "writexl", "haven"))

"File-reading packages are ready."
```

Locally, use `install.packages(c("readr", "readxl", "writexl", "haven"))` once. The cell above installs browser builds. If it fails here, continue with the CSV examples or use local R.

A faster CSV reader: readr

Example

```
from_readr <- readr::read_csv("data/moorfields_baseline.csv", na = c("", "NA"))

from_readr
```

`readr::read_csv()` returns a tibble, a compact data frame, and prints guessed types. `show_col_types = FALSE` hides that report. It uses `na` instead of `na.strings` and preserves names with spaces, which need backticks when used in code, such as `` `Visual acuity` ``.

Excel workbooks

Example

```
writexl::write_xlsx(cohort, "cohort.xlsx")

readxl::excel_sheets("cohort.xlsx") # Check the sheet names
from_excel <- readxl::read_excel("cohort.xlsx", sheet = 1)
head(from_excel, 3)
```

`excel_sheets()` lists available sheets; `sheet` selects one. Try its name instead of `1`, or add `range = "A1:C6"`. Merged headers, notes, and colours aren't a clean data structure; select a rectangular range or request a CSV export.

SPSS, Stata, and SAS files

Example

```
haven::write_sav(cohort, "cohort.sav")
from_spss <- haven::read_sav("cohort.sav")

head(from_spss$baseline_vision)
# Turn SPSS value labels into factor levels
from_spss$baseline_vision <- haven::as_factor(from_spss$baseline_vision)
levels(from_spss$baseline_vision)
```

SPSS stores numbers with value labels, such as 1 = Poor. haven preserves these labels; `as_factor()` turns them into R categories. SAS transport files (`.xpt`) are often used in clinical-trial submissions.

From a web address

Local R example

```r
url <- "https://example.org/study/data.csv"   # replace with the real address
remote <- read.csv(url)

# Keep a dated local copy, so the analysis doesn't change if the online file does
local_file <- paste0("data/download_", Sys.Date(), ".csv")
download.file(url, local_file, mode = "wb")
remote <- read.csv(local_file)
```

Replace the example URL with a real address. A dated local copy prevents later online changes from silently changing your analysis. `mode = "wb"` keeps downloaded bytes unchanged, including Excel files on Windows. Browser downloads also depend on the host allowing cross-origin requests.

Other sources

The data import cheatsheet summarizes readr and readxl.

← 1. Installing R

3. Building a Table 1 →