

Data import

From flat files and excel files



Roland Krause | rworkshop | 2021-09-08

Learning objectives

You will learn to:

- Create and use *Rstudio projects*
- Download a data file to a dedicated subfolder
- Learn about **tibbles**
- Use **readr** to import flat-files data into *R*
- Use the interactive RStudio interface to visualise your data import
- Appreciate tibbles
- Use **readxl** to import excel files into *R*



Importing data

Getting started

- Represents probably the first step of your work
- R can handle multiple data types
 - Flat files (`.csv`, `.tsv`, ...)
 - Excel files (`.xls`, `.xlsx`)
 - Foreign statistical formats
 - `.sas` from SAS
 - `.sav` from SPSS
 - `.dta` from Stata
 - databases (SQL, SQLite ...)

Tidyverse implementation

- **R base** already provides functions for text files
 - `read.csv()`
 - `read.delim()`
 - ...
- **tidyverse** redefines these functions:
 - `speed`
 - **characters are not coerced to factors by default** (R base neither since `>=4.0`)
 - generates tibbles



Tibbles

Tibbles

- Have a refined print method that shows only the first 10 rows.
- Show all the columns that fit on screen and list the name of remaining ones.
- Each column reports its type (`double`, `int`, `lgl`, `chr`)
- Makes it much easier to work with large data.



Hint

Use `as_tibble()` to convert a `data.frame` to a tibble.

data.frame vs tibble

Data frame

```
swiss
```

	Fertility	Agriculture	Examination	Education	Cathol
Courtelary	80.2	17.0	15	12	9.
Delemont	83.1	45.1	6	9	84.
Franches-Mnt	92.5	39.7	5	5	93.
Moutier	85.8	36.5	12	7	33.
Neuveville	76.9	43.5	17	15	5.
Porrentruy	76.1	35.3	9	7	90.
Broye	83.8	70.2	16	7	92.
Glane	92.4	67.8	14	8	97.
Gruyere	82.4	53.3	12	7	97.
Sarine	82.9	45.2	16	13	91.
Veveyse	87.1	64.5	14	6	98.
Aigle	64.1	62.0	21	12	8.
Aubonne	66.9	67.5	14	7	2.
Avenches	68.9	60.7	19	12	4.
Cossonay	61.7	69.3	22	5	2.
Echallens	68.3	72.6	18	2	24.
Grandson	71.7	34.0	17	8	3.
Lausanne	55.7	19.4	26	28	12.
La Vallee	54.3	15.2	31	20	2.
Lavaux	65.1	73.0	19	9	2.
Morges	65.5	59.8	22	10	5.
Moudon	65.0	55.1	14	3	4.
Nyone	56.6	50.9	22	12	15.
Orbe	57.4	54.1	20	6	4.
Oron	72.5	71.2	12	1	2.
Payerne	74.2	58.1	14	8	5.
Paysd'enhaut	72.0	63.5	6	3	2.

Tibble

```
as_tibble(swiss)
```

```
# A tibble: 47 × 6
  Fertility Agriculture Examination Education Catholic Infant.
    <dbl>      <dbl>         <int>      <int>      <dbl>
1    80.2         17             15         12      9.96
2    83.1        45.1             6          9     84.8
3    92.5        39.7             5          5     93.4
4    85.8        36.5            12          7     33.8
5    76.9        43.5            17         15      5.16
6    76.1        35.3             9          7     90.6
7    83.8        70.2            16          7     92.8
8    92.4        67.8            14          8     97.2
9    82.4        53.3            12          7     97.7
10   82.9        45.2            16         13     91.4
# ... with 37 more rows
```

Tibbles

`tibbles` never use `rownames`. They are lost unless you assign them to a dedicated column

```
# library(tibble)
as_tibble(swiss, rownames = "Province")
```

```
# A tibble: 47 × 7
  Province      Fertility Agriculture Examination Education Catholic
  <chr>         <dbl>         <dbl>         <int>         <int>         <dbl>
1 Courtelary    80.2           17            15            12           9.96
2 Delemont      83.1           45.1           6             9           84.8
3 Franches-Mnt  92.5           39.7           5             5           93.4
4 Moutier       85.8           36.5           12            7           33.8
5 Neuveville    76.9           43.5           17            15           5.16
6 Porrentruy    76.1           35.3           9             7           90.6
7 Broye         83.8           70.2           16            7           92.8
8 Glane         92.4           67.8           14            8           97.2
9 Gruyere       82.4           53.3           12            7           97.7
10 Sarine       82.9           45.2           16            13          91.4
# ... with 37 more rows, and 1 more variable: Infant.Mortality <dbl>
```

Other enhancements

- no characters to factors (R base neither since `>=4.0`)
- colnames can be repaired, "`minimal`", "`unique`", "`universal`", "`check_unique`", see [details](#)

The tidyverse packages to import your data

readr

- **readr** package:
 - **read_csv()**: comma separated (,)
 - **read_csv2()**: separated (;)
 - **read_tsv()**: tab separated
 - **read_delim()**: general delimited files, auto-guesses delimiter
 - **read_fwf()**: fixed width files
 - **read_table()**: columns separated by white-space(s)

readxl

To import excel files (.xls and .xlsx):

- **read_excel()**
 - **read_xls()**
 - **read_xlsx()**



haven

- **read_sas()** for SAS
- **read_sav()** for SPSS
- **read_dta()** for Stata



Working directory set up with RStudio projects

Avoid using `setwd()` + absolute file paths

What's wrong with `setwd()`? [Jenny Bryan's great blog post on project based workflows](#)

- `setwd()` set the working directory to a specific folder through an absolute file path

```
setwd("C:/Users/veronica.codoni/Projects/Survival_Analysis")
```

- What if this folder moved to /Downloads, or onto another machine?

```
setwd("C:/Users/veronica.codoni/Projects/Survival_Analysis")
```

```
Error in setwd("C:/Users/veronica.codoni/Projects/Survival_Analysis") :
```

This approach is **not self-contained** and **portable**!!

RStudio projects as alternative ☐

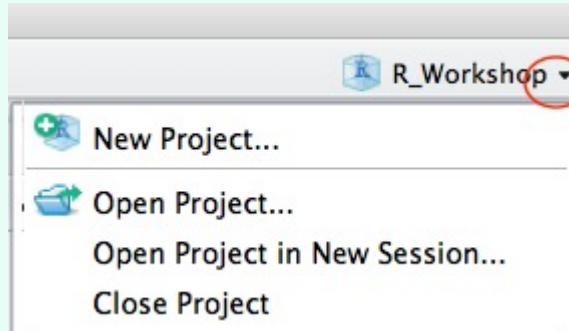
Start a new research project/data analysis by creating a new *Rstudio Project*

- it helps to keep all the files together
- it sets **automatically** the working directory to the project directory
- it **helps resolve file path issues** by using **relative** paths

Create a Project in Rstudio assures the project directory to be **stand-alone** and **portable**.

How to create RStudio projects

Use the **Project** button in the top right corner



- Name the project properly (keeping it short, no spaces)
- Once the project has been created, a **.Rproj** extension file is generated. This allows for automatic working directory set-up.

Exercise

- Create new project called **Rworkshop**
 - in a new folder
 - no **git** (even if you *should*)
- Create sub-folders **data** (**analysis**, **plots** ...)

Importing flat files



Reading flat files

Flat file example: *swiss.csv*

- Use the new RStudio project (finding your files will be easier)
- Download the [swiss.csv](#) file to your project folder, in the sub-folder `data`
- Open the file with a text viewer and have a look at its content
- Does the delimiter fit the file extension?

```
"Fertility","Agriculture","Examination","Education","Catholic","Infant.Mortality"  
80.2,17,15,12,9.96,22.2  
83.1,45.1,6,9,84.84,22.2  
92.5,39.7,5,5,93.4,20.2  
85.8,36.5,12,7,33.77,20.3  
76.9,43.5,17,15,5.16,20.6  
76.1,35.3,9,7,90.57,26.6  
83.8,70.2,16,7,92.85,23.6  
92.4,67.8,14,8,97.16,24.9  
82.4,53.3,12,7,97.67,21  
...
```

Exercise, download a csv file in a project sub-folder

1. Download swiss.csv in data

In your browser:

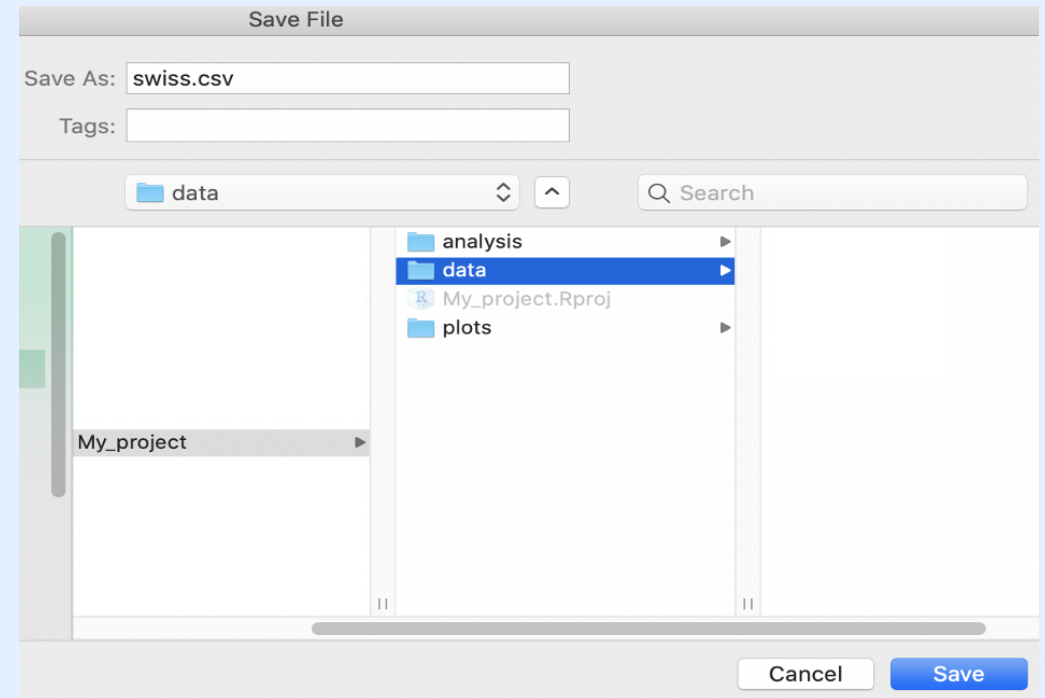
- Right-click on the **link**
- Choose **Save Link As**

- You **MUST** have created a RStudio project
- Use Finder/Explorer to select this project folder
- If not present **New Folder** called **data**
- Go inside this sub-folder **data**
- Hit **Save**

Flat file example: swiss.csv

- Create a new RStudio project (finding your files will be easier)
- Download the **swiss.csv** file to your project folder, in the sub-folder **data**
- Open the file with a text v

"Fertility", "Agriculture"
80.2, 17, 15, 12, 9.96, 22.2
83.1, 45.1, 6, 9, 84.84, 22.2
92.5, 39.7, 5, 5, 93.4, 20.2
85.8, 36.5, 12, 7, 33.77, 20.3
76.9, 43.5, 17, 15, 5.16, 20.6
76.1, 35.3, 9, 7, 90.57, 26.6
83.8, 70.2, 16, 7, 92.85, 23.6
92.4, 67.8, 14, 8, 97.16, 24.9
82.4, 53.3, 12, 7, 97.67, 21

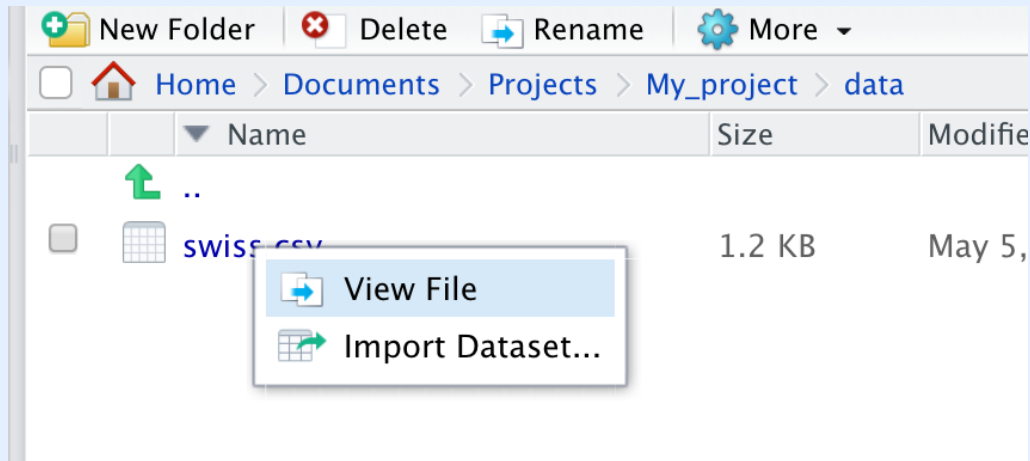


Exercise, download a csv file in a project sub-folder

2. Check if file is present and preview it

In RStudio:

- Use file panel
- Go in `data` and check if `swiss.csv` is present
- Right-click, menu should be:



- If `swiss.csv` is present in the `data` folder
- You can **View File** to see its content
- Cheat and **Import Dataset** it
- However, we want to import it programmatically
 - Ensures reproducibility
 - Enables sharing with others

Exercise, download a csv file in a project sub-folder



3. Import the file with `readr::read_delim()`

The file says ".csv" but could be wrong

```
# library(readr) or
# library(tidyverse)
read_delim("data/swiss.csv")
```

```
Rows: 47 Columns: 6
— Column specification —————
Delimiter: ","
dbl (6): Fertility, Agriculture, Examination, Education, Cathol
```

```
□ Use `spec()` to retrieve the full column specification for th
□ Specify the column types or set `show_col_types = FALSE` to q
```

```
# A tibble: 47 x 6
  Fertility Agriculture Examination
    <dbl>         <dbl>         <dbl>
1      80.2          17          15
2      83.1         45.1           6
3      92.5         39.7           5
4      85.8         36.5          12
5      76.9         43.5          17
6      76.1         35.3           9
7      83.8         70.2          16
8      92.4         67.8          14
9      82.4         53.3          12
10     82.9         45.2          16
# ... with 37 more rows, and 3 more
```

What do we observe with `readr::read_delim()`?

- Guesses automatically the field separator: ";"
- Reports the dimensions of the table read
- Also guessed the column types (all `double` here)
- Returns a `tibble` by default

Watch out! we didn't assign the tibble to a name!

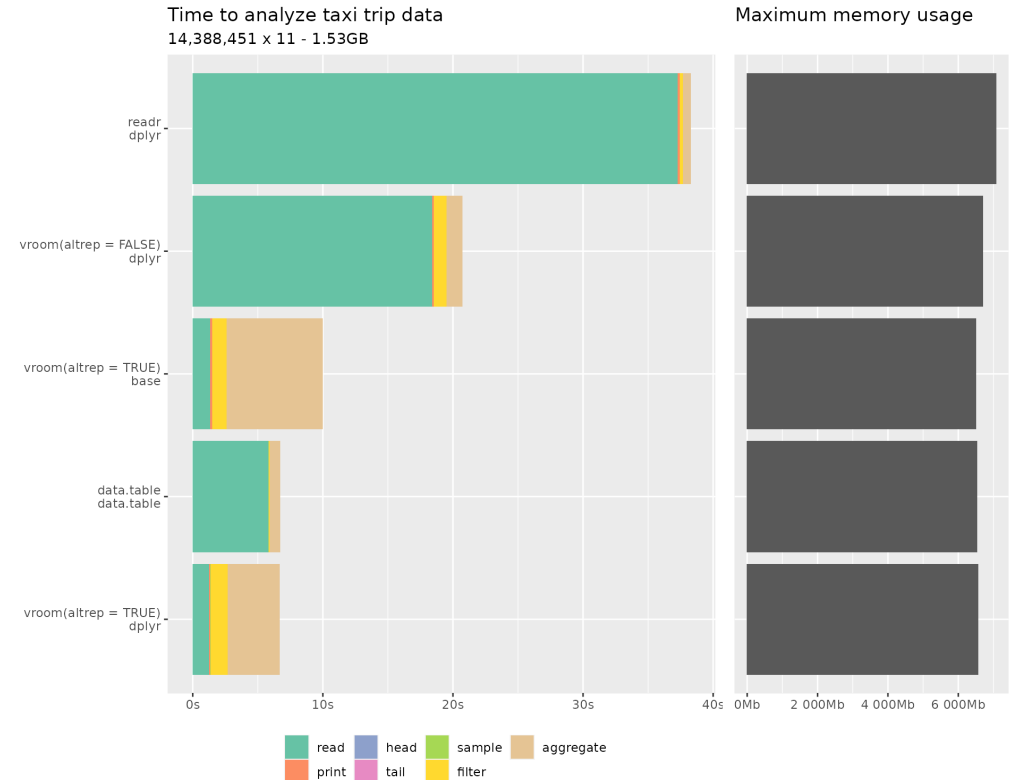
To do so:

```
swiss_db <- read_delim("data/swiss.csv")
```

readr 2.0 engine is now vroom

readr is the official import package of the tidyverse

- [Jim Hester](#) also developed **vroom**. All of the functionality are included in **readr** version 2.0.
- Great features
 - Column type guessing
 - Reporting parsing problems
 - Progress bar
 - Reading from urls
 - Reading compressed files
- Amazing features in **readr** with version 2.0 (released July 2021):
 - Delimiter guessing
 - **FAST!**
 - multi-threaded
 - use [ALTREP](#)
 - Column selection
 - Merge multiple files into one **tibble**



source: [vroom benchmarks](#)

Catch and report reading issues

Toy example with 2 created issues

```
read_delim("data/swiss2.csv")
```

Rows: 5 Columns: 5

— Column specification —
Delimiter: ","
chr (1): fer
dbl (4): agri, exa, edu, cath

- Use ``spec()`` to retrieve the full column specification for the data
- Specify the column types or set ``show_col_types = FALSE`` to suppress column types

```
# A tibble: 5 × 5
  fer    agri    exa    edu  cath
<chr> <dbl> <dbl> <dbl> <dbl>
1 80.2    17     15    12  NA
2 a      45.1     6     9 84.8
3 92.5    39.7     5     5  NA
4 85.8    36.5    12     7 33.8
5 76.9    43.5    17    15  5.16
```

- two missing data
- **fer** is no longer of type **double**

Specifying column types

readr record issues, **problems()** to see them

```
sw2 <- readr::read_delim("data/swiss2.csv",
  col_types = list(fer = col_double(),
                   agri = col_double(),
                   exa = col_integer(),
                   edu = col_integer(),
                   cath = col_double()))
```

sw2

```
# A tibble: 5 × 5
  fer    agri    exa    edu  cath
<dbl> <dbl> <int> <int> <dbl>
1 80.2    17     15    12  NA
2 NA      45.1     6     9 84.8
3 92.5    39.7     5     5  NA
4 85.8    36.5    12     7 33.8
5 76.9    43.5    17    15  5.16
```

```
problems(sw2)
```

```
# A tibble: 2 × 5
  row    col expected actual file
<int> <int> <chr>    <chr> <chr>
1     2     1 a double a /builds/r-training/rworkshop/site
2     3     1 a double a /builds/r-training/rworkshop/site
```


Choosing which columns to read

Lighter column type definitions

here, we want to skip the wrong columns: `fer` and `cath`.

We let `read_delim` guessing the `agri` type (?), see all correspondences [here](#).

```
readr::read_delim("data/swiss2.csv",  
  col_types = list(fer = "_",  
                   agri = "?",  
                   exa = "i",  
                   edu = "i",  
                   cath = "_"))
```

```
# A tibble: 5 x 3  
  agri    exa    edu  
  <dbl> <int> <int>  
1  17      15     12  
2  45.1      6      9  
3  39.7      5      5  
4  36.5     12      7  
5  43.5     17     15
```

Columns selection (support `tidyselect` syntax)

```
readr::read_delim("data/swiss2.csv",  
  col_select = c(agri, edu))
```

```
# A tibble: 5 x 2  
  agri    edu  
  <dbl> <dbl>  
1  17      12  
2  45.1      9  
3  39.7      5  
4  36.5      7  
5  43.5     15
```

Column selection with tidyselect

This package is the backend for programatic column selection in the [tidyverse](#). Features are described in [this article](#).

Operators

- `[c()]` for **combining** bare names
- `[:]` for selecting a **range**
- `[-]` for **negating** a selection

Helpers

- `everything()` **all** columns
- `last_col()` **last** column

Using patterns

- `starts_with()` with quoted prefix
- `ends_with()` with quoted suffix
- `contains()` with quoted string

```
readr::read_delim("data/swiss2.csv", show_col_types = FALSE,  
  col_select = starts_with("e"), n_max = 1)
```

```
# A tibble: 1 x 2  
  exa     edu  
  <dbl> <dbl>  
1    15     12
```

```
readr::read_delim("data/swiss2.csv", show_col_types = FALSE,  
  col_select = contains("e"), n_max = 1)
```

```
# A tibble: 1 x 3  
  fer     exa     edu  
  <dbl> <dbl> <dbl>  
1  80.2    15     12
```

```
readr::read_delim("data/swiss2.csv", show_col_types = FALSE,  
  col_select = fer:edu, n_max = 1)
```

```
# A tibble: 1 x 4  
  fer agri     exa     edu  
  <dbl> <dbl> <dbl> <dbl>  
1  80.2    17    15     12
```

Header, what to do when absent

Example

- Using the URL <https://basv53.uni.lu/lectures/data/example.csv>
- This toy data contains 3 columns

Content is:

```
dog,red,1
cat,blue,2
chicken,green,6
```

Naive approach

```
exa <- "https://basv53.uni.lu/lectures/data/example.csv"
readr::read_delim(exa)
```

Rows: 2 Columns: 3

— Column specification —

```
Delimiter: ","
chr (2): dog, red
dbl (1): 1
```

- Use ``spec()`` to retrieve the full column specification for the file
- Specify the column types or set ``show_col_types = FALSE`` to suppress column types

```
# A tibble: 2 × 3
  dog      red    `1`
  <chr>   <chr> <dbl>
1 cat     blue     2
2 chicken green     6
```

Satisfying?

Provide the header

Read all lines

- Colnames are self-created
- `show_col_types = FALSE` suppresses the verbose mode

```
read_delim(exa, col_names = FALSE, show_col_types = FALSE)
```

```
# A tibble: 3 × 3  
  X1      X2      X3  
  <chr> <chr> <dbl>  
1 dog    red      1  
2 cat    blue     2  
3 chicken green    6
```

Satisfying?

Supply the colnames

- Remember we are still reading directly from the url in the `exa` variable

```
read_delim(exa, col_names = c("animal", "color", "value"),  
            show_col_types = FALSE)
```

```
# A tibble: 3 × 3  
  animal color value  
  <chr>   <chr> <dbl>  
1 dog    red      1  
2 cat    blue     2  
3 chicken green    6
```

Better

Exercise

Override the detected column types

Import `example.csv` using the URL but

- **Skip** the `colour` column
- Read in the `value` column as **integer**

Hints

Column types are specified using `col_types = list()`

Function	Short	Description
<code>col_logical()</code>	l	TRUE/FALSE, 1/0
<code>col_integer()</code>	i	integers
<code>col_double()</code>	d	floating point values.
<code>col_number()</code>	n	numbers containing the grouping_mark
<code>col_date(format = "")</code>	D	with the locale's date_format
<code>col_time(format = "")</code>	t	with the locale's time_format
<code>col_datetime(format = "")</code>	T	ISO8601 date times
<code>col_factor(levels, ordered)</code>	f	a fixed set of values
<code>col_character()</code>	c	everything else
<code>col_skip()</code>	-	-, don't import this column
<code>col_guess()</code>	?	parse using the "best" type based on the input

Solution

Using one-letter shortcuts

```
readr::read_delim(exa,  
  col_names = c("animal", "color", "value"),  
  col_types = list(  
    animal = "c",  
    color = " ",  
    value = "i"  
  ))
```

```
# A tibble: 3 × 2  
  animal value  
  <chr>   <int>  
1 dog         1  
2 cat         2  
3 chicken     6
```

Even more compact

```
readr::read_delim(exa,  
  col_names = c("animal", "color", "value"),  
  col_types = "c_i")
```

```
# A tibble: 3 × 2  
  animal value  
  <chr>   <int>  
1 dog         1  
2 cat         2  
3 chicken     6
```

- Use the single character code in the column order

Skipping lines

Our example

```
# A tibble: 3 × 2
  animal value
  <chr>   <int>
1 dog         1
2 cat         2
3 chicken     6
```

Comment

Do not read lines beginning with a character.

```
readr::read_delim(
  exa,
  comment = "d",
  col_names = FALSE,
  show_col_types = FALSE)
```

```
# A tibble: 2 × 3
  X1      X2      X3
<chr> <chr> <dbl>
1 cat   blue    2
2 chicken green   6
```

Skip lines

```
readr::read_delim(
  exa,
  skip = 1,
  col_names = FALSE,
  show_col_types = FALSE)
```

```
# A tibble: 2 × 3
  X1      X2      X3
<chr> <chr> <dbl>
1 cat   blue    2
2 chicken green   6
```

Limited number of lines

```
readr::read_delim(
  exa,
  skip = 1,
  n_max = 1,
  col_names = FALSE,
  show_col_types = FALSE
)
```

```
# A tibble: 1 × 3
  X1      X2      X3
<chr> <chr> <dbl>
1 cat   blue    2
```

Alternatives

Need for (more) speed

Check `fread()` from [data.table](#)

- Stable
- No dependencies
- Overhauled
- Could feed from `bash` command
- Could be faster than `readr` functions



Importing Excel files



Reading Excel files



readxl package is part of tidyverse and makes it easy to import tabular data from Excel spreadsheets with several options.

More details are described in this [pkgdown website](#)

- **readxl** author: [Hadley Wickham](#) and [Jenny Bryan](#)
- **read_excel()** reads both **xls** and **xlsx** files and detects the format from the extension. Otherwise:
 - **read_xls()**
 - **read_xlsx()**
- Return tibbles
- Column type guessing
- Discovers the minimal data rectangle and returns that, by default
- Exert more control with **range**, **skip**, and **n_max**
- **excel_sheets()** returns the sheet names
- Combine multiple sheets, jointly with **purrr::map()**
- No external library dependencies, e.g., Java or Perl

Before we stop

You learned to:

- Appreciate the tibble features
- Learn set-up working directory with RStudio [projects](#)
- Learn the [tidyselect](#) syntax
- Use [readr::read_delim](#) to import your flat files
- Adjust the imported data types
- Use [readxl](#) to import excel files

Further reading

- [R for Data Science](#)

Acknowledgments ☐ ☐

Development

- Eric Koncina (initial installment)
- Veronica Codoni (swiss data set)
- Aurelién Ginolhac ([vroom](#) development)
- Roland Krause ([readr](#) 2.0 update)

Input and inspiration

- Jim Hester ([vroom](#), [readr](#) development)
- Jenny Bryan (advice with project organization)
- Hadley Wickham
- Nicholas Tierney

Thank you for your attention!