

Practical: Data I/O

BaseIRBootcamp 2017

Slides

Here a link to the lecture slides for this session: **LINK**

(https://therbootcamp.github.io/_sessions/D1S3_DataIO/Data_IO.html)

Overview

In this practical you'll learn how to read and save data. By the end of this practical you will know how to:

1. Create a project
2. Identify the location of a file
3. Read in data of various types
4. Use R's file connections
5. Scrape the internet (if you get there)

Functions

Here are the main read-in functions:

| Function | Description |
|---------------------------------------|---------------------------|
| <code>read_csv()</code> | Read flat csv file |
| <code>read_sas()</code> | Read SAS file |
| <code>read_sav()</code> | Read SPSS file |
| <code>readRDS()</code> | Read RDS file |
| <code>file(...,'r'), readLines</code> | Read from file connection |

Here are the main export functions:

| Function | Description |
|---------------------------------------|--------------------------|
| <code>write_csv()</code> | Write flat csv file |
| <code>write_sas()</code> | Write SAS file |
| <code>write_sav()</code> | Write SPSS file |
| <code>saveRDS()</code> | Save RDS file |
| <code>file(...,'w'), readLines</code> | Write to file connection |

Tasks

The tutorial begins with the titanic data set, which contains records of 1313 passengers on their name (`Name`), their passenger class (`PClass`), their age (`Age`), their sex (`Sex`), whether they survived (`Survived`), and finally a numeric coding of their sex (`SexCode`).

Later you will be working with a randomly generated artificial data set of mine, which will help demonstrate differences in speed and file size.

Please download the following data files and save them to an appropriate location on the disc:

titanic.csv

(https://raw.githubusercontent.com/therbootcamp/therbootcamp.github.io/master/_sessions/data/titanic.csv)

titanic.sav

(https://raw.githubusercontent.com/therbootcamp/therbootcamp.github.io/master/_sessions/data/titanic.sav)

titanic.sas7bdat

(https://raw.githubusercontent.com/therbootcamp/therbootcamp.github.io/master/_sessions/data/titanic.sas7bdat)

my_data.csv

(https://raw.githubusercontent.com/therbootcamp/therbootcamp.github.io/master/_sessions/data/my_data.csv)

If you can, please verify whether the `.sav` and `sas7bdat` are appropriate SPSS and SAS files, respectively.

Read to tibble

1. Create a new **project** either in the location of the downloaded data files or its parent folder (one folder higher). To do this go to *File/New Project...*, select existing directory, and enter the desired folder. This step will ease reading and saving data, as it sets the working directory to the proximity of the data files. Confirm this using `getwd()`.
2. Identify the filepath (incl. filename) for each of the three data files. RStudio makes it really easy. Simply write quotation marks in your script editor, e.g., `" "`, move the (text) cursor between the marks, and press the tab key. This opens a file menu that allows you to conveniently select the path to the file. The result is a character string that exactly defines the location of the file relative to the current working directory. Oftentimes, it is convenient to do this directly within the read function, e.g., `read_csv("")`.
3. Load the packages `readr` and `haven` using `library()`, and read in the three titanic data files using `read_csv()`, `read_sas()`, and `read_sav()`. Call them `df_csv`, `df_sas`, and `df_sav`, respectively. Inspect each of the imported objects using `print()` and `str()`. What kind of object are they?
4. Try to verify that the three data sets are identical. To do this use the *is-equal-to* operator `==` introduced in the last tutorial and the arithmetic mean function `mean()`. Remember, that logical values can be coerced to 0 and 1. Note, that the data sets contain NA's, that is missing values. One way to see this is via the summary function `summary()`. Specifically, the variable `Age` contains missing values. Use the help file `?mean` to find out how to compute the mean while ignoring missing values and then use `==` to verify the equality between the three data sets.

Write to disk

5. The first column in the titanic set contains row numbers. Older read and write functions in R would by default add a row number column to the data, when writing to disk. Newer functions, however, do not show this behavior. Eliminate the first column from `df_csv` using a negative index. Negative indices mean *omit* rather than *select*. E.g., `c(1, 2, 3)[-2]` returns the vector without the second value. Then write the reduced data frame to disk using `write_csv()`. Use the exact same file path. Observe whether R gives you a warning before (over-)writing the data?
6. Determine the file path to `my_data.csv` and read it in using `read_csv()`. Now write the data back to disk using `write_sas`, `write_sav`, and also `saveRDS`. Each of these functions take as the first argument the data frame and as the second argument the file path. To obtain the latter adapt the filepath of `my_data.csv` by hand to make sure that the file path has the correct file ending, i.e., `.sas7bdat`, `.sav`, and `.RDS`, respectively. While you write the individual data sets pay attention to the speed of execution.
7. You may have noticed that the `saveRDS` function was the slowest. This is because `.RDS` generally leads to the highest degree of compression, i.e., the smallest data files. To verify this use the `file.info()` on each of the four files, `my_data.csv`, `my_data.sas7bdat`, `my_data.sav`, and `my_data.RDS` (using the correct path). Look out for the `size` element, which gives the size in `bytes`.

File connections

8. The most basic way to handle files is via file connections. The first step of working with file connections is to establish a file connections using, e.g., `file()`. The main arguments to `file` are a location (a file path, url, etc.) and a mode indicator, e.g., `r` for Open for reading in text mode. Try opening a connection to the `titanic.csv` dataset. To do this, you must assign the output of `file()` to an object, which then is the connection, e.g., `con = file("my_path", "r")`.
9. Now that the connection is open, you can read the file using, e.g., `readLines`. `readLines` iterates through the file line by line and returns each line as a character string. Store the output in an object. Try reading the file. When done close the connection using `close(my_con)` and inspect the data.
10. As you can see the read in data looks much messier than with using `read_csv()`. If you're up for the challenge you can try to nonetheless parse this data. A good way is to split each line using the `stri_split_fixed` function using the comma `,` (from the `stringi` package). This will return a list of vectors, where each vector is one row of the data frame. Next step is to bind the rows to a matrix using `do.call(rbind, my_list)`. What's then left is transform the matrix to a data frame and to take care of column types and names.

Scrapping the internet

11. Finally on to something rather different. Connections can also be established to data located outside one's own disk, such as, for instance, the world wide web. Specifically, using `url()` we can establish a connection to a webpage, which we then can use to retrieve information. This is used, for instance, by the very convenient `rvest` package. To illustrate how easy it can be, consider the following code that downloads and parses Milestones table of R's Wikipedia page.

```
# load package
install.packages('rvest')
library(rvest)
library(magrittr)

# get html
url = 'https://en.wikipedia.org/wiki/R_(programming_language)'
page = read_html(url)

# get table
# use XPath from inspect page (e.g., in Chrome)
table = page %>% html_node(xpath = '//*[@id="mw-content-text"]/div/table[2]') %>% html_
table()

# create tibble
as.tibble(table)
```

Additional reading

- For more details on all steps of data analysis check out Hadley Wickham's R for Data Science (<http://r4ds.had.co.nz/>).
- For more advanced content on objects check out Hadley Wickham's Advanced R (<http://adv-r.had.co.nz/>).
- For more on pirates and data analysis check out the respective chapters in YaRrr! The Pirate's Guide to R YaRrr! Chapter Link (<https://bookdown.org/ndphillips/YaRrr/htests.html>)