Importing and cleaning data with R

Saul Diaz Infante Velasco

2/23/23

Table of contents

Preface **1**

Introduction **3**

I. Introduction to Importing Data in R **5**

1. Importing data from flat files with utils **7**
1.1. read.csv 7
1.1.1. Instructions 100 XP 7
References 8
1.2. stringsAsFactors 8
Instructions 100 XP 8
1.3. Any changes? 9
1.4. read.delim 10
Instructions 100 XP 10

2. Summary **13**

References **15**

Preface

Understanding how to prep your data is an essential skill when working in R. It's what you have to do before you can reveal the insights that matter. In this part, you'll learn how to import your data from a variety of sources, among others .csv, .xls, text files, and more. You'll then gain the skills you'll need to prepare your data for analysis, including converting data types, filling in missing values, and using fuzzy string matching. Throughout the track, you'll have the chance to apply your skills to real-world data such as customer asset portfolios and restaurant reviews. Start this track and gain the data prepping skills you need to clean your dirty data.

This part encloses four chapters. The first and second considers the general functions to load common formats. The rhird chapter treat some techinques to manage partial information. The fourth chapter is abour reshaping and we may close with a project in order to apply correspongin content of this course.

1. Introduction to Importing Data in R
2. Intermediate Importing Data in R
3. Cleaning Data in R
4. Reshaping Data with tidyr

Introduction

Importing data into R should be the easiest step in your analysis. Unfortunately, that is almost never the case. Data can come in many formats, ranging from .csv and text files, to statistical software files, to databases and HTML data. Knowing which approach to use is key to getting started with the actual analysis. In this course, you'll start by learning how to read .csv and text files in R. You will then cover the readr and data.table packages to easily and efficiently import flat file data. After that, you will learn how to read .xls files in R using readxl and gdata.

This content has been taken from the dataCamp course with the same title. See

https://app.datacamp.com/learn/courses/introduction-to-importing-data-in-r

for more details.

Part I.

Introduction to Importing Data in R

1. Importing data from flat files with utils

A lot of data comes in the form of flat files: simple tabular text files. Learn how to import the common formats of flat file data with base R functions.

1.1. read.csv

The `utils` package, which is automatically loaded in your R session on startup, can import CSV files with the `read.csv()` function.

In this exercise, you'll be working with `swimming_pools.csv` (view); it contains data on swimming pools in Brisbane, Australia (Source: data.gov.au). The file contains the column names in the first row. It uses a comma to separate values within rows.

Type `dir()` in the console to list the files in your working directory. You'll see that it contains swimming_pools.csv, so you can start straight away.

1.1.1. Instructions `100 XP`

- Use read.csv() to import "swimming_pools.csv" as a data frame with the name pools.
- Print the structure of pools using str().

ex_001.R

1. *Importing data from flat files with utils*

```
# Import swimming_pools.csv: pools
pools <- read.csv("swimming_pools.csv")
# Print the structure of pools
str(pools)
```

References

- http://s3.amazonaws.com/assets.datacamp.com/production/course_1477/datasets/swimming_pools.csv
- https://data.gov.au/dataset/swimming-pools-brisbane-city-council

1.2. stringsAsFactors

With `stringsAsFactors`, you can tell R whether it should convert strings in the flat file to factors.

For all importing functions in the utils package, this argument is TRUE, which means that you import strings as factors. This only makes sense if the strings you import represent categorical variables in R. If you set `stringsAsFactors` to `FALSE`, the data frame columns corresponding to strings in your text file will be character.

You'll again be working with the `swimming_pools.csv` (view in data folder) file. It contains two columns (`Name` and `Address`), which shouldn't be factors.

Instructions `100 XP`

- Use `read.csv()` to import the data in `"swimming_pools.csv"` as a data frame called pools; make sure that strings are imported as characters, not as factors.

- Using `str()`, display the structure of the dataset and check that you indeed get character vectors instead of factors.

ex_002.R

```
# Import swimming_pools.csv correctly: pools
pools <- read.csv("swimming_pools.csv", stringsAsFactors = FALSE)

# Check the structure of pools
str(pools)
```

1.3. Any changes?

Consider the code below that loads data from `swimming_pools.csv` in two distinct ways:

```
# Option A
pools <- read.csv("swimming_pools.csv", stringsAsFactors = TRUE)

# Option B
pools <- read.csv("swimming_pools.csv", stringsAsFactors = FALSE)
```

ex_003.R

```
library(projmgr)

# the following could be run in RMarkdown
todo_path <- system.file(
    "extdata",
    "todo-ex.yml",
     package = "projmgr",
     mustWork = TRUE
```

```
)

my_todo <- read_todo(todo_path)
report_todo(my_todo)
```

1.4. read.delim

Aside from `.csv` files, there are also the `.txt` files which are basically text files. You can import these functions with `read.delim()`. By default, it sets the sep argument to `"\t"` (fields in a record are delimited by tabs) and the header argument to `TRUE` (the first row contains the field names).

In this exercise, you will import hotdogs.txt (view), containing information on sodium and calorie levels in different hotdogs (Source: UCLA). The dataset has 3 variables, but the variable names are not available in the first line of the file. The file uses tabs as field separators.

Instructions `100 XP`

- Import the data in `"hotdogs.txt"` with `read.delim()`. Call the resulting data frame `hotdogs`. The variable names are not on the first line, so make sure to set the header argument appropriately.
- Call `summary()` on `hotdogs`. This will print out some summary statistics about all variables in the data frame.

ex_005

```
# Import hotdogs.txt: hotdogs
hotdogs <- read.delim(
    "hotdogs.txt",
    sep = '\t',
```

```
    header = FALSE
)

# Summarize hotdogs
summary(hotdogs)
```

Instructions 50 XP How many variables in the resulting pools data frame have different types if you specify the stringsAsFactors argument differently?

The swimming_pools.csv (view) file is available in your current working directory so you can experiment in the console.

2. Summary

In summary, this book has no content whatsoever.

References