Lecture 14: Data Import, part 2

STAT 450, Fall 2021

Today we continue our discussion on how to read data sets into R. We will also discuss how to write data sets from R to file.

Reference for `readr` package: https://readr.tidyverse.org/

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

`read_csv()`

Comma-separated values (CSV) files are one of the most common types of data files that you will encounter. Each entry on each line of a CSV file is separated by a comma.

Documentation for `read_csv()` is provided in the help menu:

```
help(read_csv)
```

Some useful arguments:

- `file`: name of the file path (should be in quotations)
- `col_names`: Either TRUE, FALSE or a character vector of column names. If TRUE, the first row of the input will be used as the column names (default). If FALSE, column names will be generated automatically.
- `skip`: Number of lines to skip before reading data

Let's look at some examples with inline CSV files to demonstrate how these arguments work. The following examples are taken directly from Section 11.2 of *R for Data Science*

By default, `read_csv()` uses the first line of the data for the column names, which is a common convention.

```
read_csv("a,b,c
1,2,3
4,5,6")
```

```
## # A tibble: 2 x 3
##       a     b     c
##   <dbl> <dbl> <dbl>
## 1     1     2     3
## 2     4     5     6
```

Sometimes there are a few lines of metadata at the top of the file. You can use `skip = n` to skip the first `n` lines.

```
read_csv("The first line of metadata
  The second line of metadata
  x,y,z
  1,2,3", skip = 2)
```

```
## # A tibble: 1 x 3
##       x     y     z
##   <dbl> <dbl> <dbl>
## 1     1     2     3
```

The data might not have column names. You can use `col_names = FALSE` to tell `read_csv()` not to treat the first row as headings, and instead label them sequentially from `X1` to `Xn`:

```
read_csv("1,2,3\n4,5,6", col_names = FALSE)
```

```
## # A tibble: 2 x 3
##      X1    X2    X3
##   <dbl> <dbl> <dbl>
## 1     1     2     3
## 2     4     5     6
```

`\n` is a shortcut for adding a new line

Alternatively, you can pass `col_names` a character vector which will be used as the column names:

```
read_csv("1,2,3\n4,5,6", col_names = c("x", "y", "z"))
```

```
## # A tibble: 2 x 3
##       x     y     z
##   <dbl> <dbl> <dbl>
## 1     1     2     3
## 2     4     5     6
```

Compared to base R

`read_csv()` is a replacement of the older, equivalent base R function `read.csv()`

`read_csv()` is typically much faster `read.csv()`

`read_csv()` returns a tibble, whereas `read.csv()` returns a traditional R `data.frame`

Exercise: Download the data set `test.csv` from Blackboard and place it in your working directory. Use `read_csv()` to read this data set into R, and name the columns "x", "y", and "d".

Exercise: Identify what is wrong with the following inline CSV file:

```
read_csv("a,b\n1,2,3\n4,5,6")
```

Other data formats

`read_csv2()` reads semicolon separated file (common in countries where `,` is used as the decimal place)

`read_tsv()` reads tab delimited files

`read_delim()` reads in files with any deliminator (specify with the `delim` argument)

Example: Inline example with a space as the deliminator:

```
read_delim("a b c
1 2 3
4 5 6", delim = " ")
```

```
## # A tibble: 2 x 3
##       a     b     c
##   <dbl> <dbl> <dbl>
## 1     1     2     3
## 2     4     5     6
```

readxl

The `readxl` package can be used to read an Excel spreadsheets into R.

Reference: https://readxl.tidyverse.org/index.html

Example: Read in the Excel spreadsheet `grades.xlsx`, which can be downloaded from Blackboard.

```
library(readxl)
grades1 <- read_excel("grades.xlsx", sheet = 1, skip = 1)
grades1
```

```
## # A tibble: 5 x 3
##   Student Score Grade
##     <dbl> <dbl> <chr>
## 1       1  0.83 B
## 2       2  0.89 B+
## 3       3  0.98 A
## 4       4  0.9  A-
## 5       5  0.83 B
```

```
grades2 <- read_excel("grades.xlsx", sheet = 2, skip = 1)
grades2
```

```
## # A tibble: 4 x 3
##   Student Score Grade
##     <dbl> <dbl> <chr>
## 1       1  0.82 B
## 2       2  0.8  B-
```

```
## 3       3  0.87 B+
## 4       4  0.99 A
```

Writing to a file

Use the function `write_csv()` to write a data set from R to a CSV file located in your working directory.

For example, the following code writes the `mpg` data frame from R to a CSV file.

```
write_csv(mpg, path = "mpg.csv")
```

Exercise: Use `filter()` to subset the rows of `mpg` that correspond to the car manufacturer `toyota`. Call the subsetted data frame `toyota_mpg`. Next, use `write_csv()` to write the data frame `toyota_mpg` to a CSV file located in your working directory.