

STAC32: Applications of Statistical Methods

Lecture notes

Reading in data

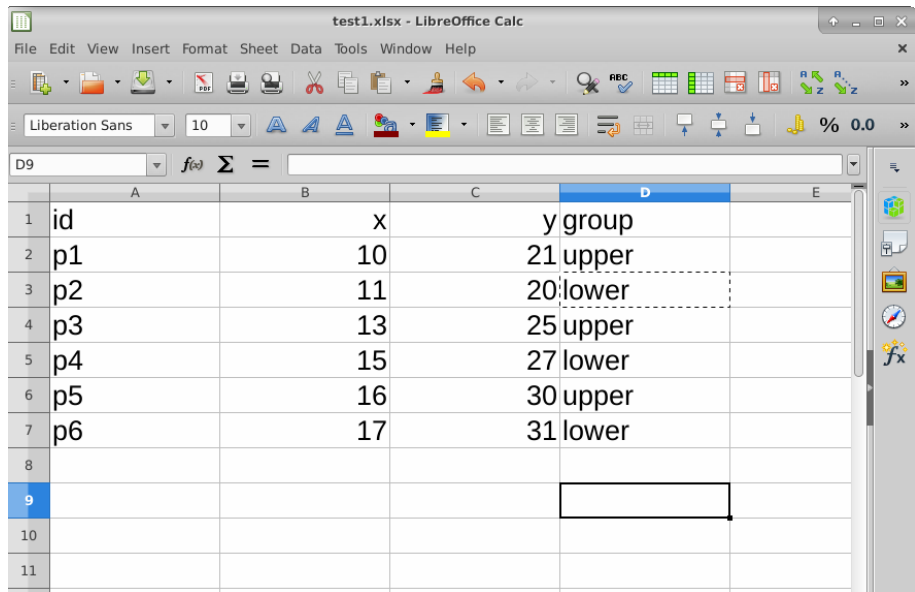
Introduction

- First thing we need to do is to read in data, so that we can use our software to analyze.
- Consider these:
 - Spreadsheet data saved as .csv file.
 - “Delimited” data such as values separated by spaces.
 - Actual Excel spreadsheets.

Packages for this section

```
library(tidyverse)
```

A spreadsheet



The screenshot shows a LibreOffice Calc spreadsheet titled "test1.xlsx". The interface includes a menu bar (File, Edit, View, Insert, Format, Sheet, Data, Tools, Window, Help), a toolbar with various icons, and a status bar at the bottom. The spreadsheet has five columns labeled A, B, C, D, and E. The data is organized into rows, with row 9 highlighted. The data in the spreadsheet is as follows:

| | A | B | C | D | E |
|----|----|----|----|-------|---|
| 1 | id | x | y | group | |
| 2 | p1 | 10 | 21 | upper | |
| 3 | p2 | 11 | 20 | lower | |
| 4 | p3 | 13 | 25 | upper | |
| 5 | p4 | 15 | 27 | lower | |
| 6 | p5 | 16 | 30 | upper | |
| 7 | p6 | 17 | 31 | lower | |
| 8 | | | | | |
| 9 | | | | | |
| 10 | | | | | |
| 11 | | | | | |

Save as .csv

- .csv or “comma-separated values” is a way of turning spreadsheet values into plain text.
- Easy to read into R (or SAS, later)
- but does not preserve formulas. (This is a reason for doing all your calculations in your statistical software, and only having data in your spreadsheet.)
- File, Save As Text CSV (or similar).
- used name test1.csv.

The .csv file

```
id,x,y,group  
p1,10,21,upper  
p2,11,20,lower  
p3,13,25,upper  
p4,15,27,lower  
p5,16,30,upper  
p6,17,31,lower
```

To read this in:

- Fire up rstudio.cloud.
- Upload this .csv file. (Bottom right, next to New Folder, Upload.) Click Choose File, find the file, click Open. Click OK. See the file appear bottom right.

Make a new notebook

- ...and get rid of the template document (leaving the first four lines).
- Make a code chunk and in it put this. Run it.

```
library(tidyverse)
```


Reading in the file

- Use `read_csv` with the name of the file, in quotes. Save the read-in file in something, here called `mydata`. Make a new code chunk for this:

```
mydata <- read_csv("test1.csv")
```

```
## Parsed with column specification:
## cols(
##   id = col_character(),
##   x = col_double(),
##   y = col_double(),
##   group = col_character()
## )
```

More on the above

- `read_csv` guesses what kind of thing is in each column. Here it correctly guesses that:
- `id` and `group` are text (categorical variables). `id` is actually “identifier variable”: identifies individuals.
- `x` and `y` are integers (quantitative variables that here have no decimal point). Decimal numbers would be labelled `num` or `double`.

R Studio on your own computer

- Put the .csv file in the same folder as your project. Then read it in as above like `read_csv("test1.csv")`.
- Or, use `f=file.choose()` which brings up a file selector (as if you were going to find a file to load or save it). Find your .csv file, the address of which will be saved in `f`, and then:

```
mydata=read_csv(f)
```

- When you have selected the file, comment out the `file.choose` line by putting a `#` on the front of it. That will save you having to find the file again by mistake. (Keyboard shortcut: go to the line, type control-shift-C or Mac equivalent with Cmd.)

Looking at what we read in

- Again, type the name of the thing to display it:

```
mydata
```

| id | x | y | group |
|----|----|----|-------|
| p1 | 10 | 21 | upper |
| p2 | 11 | 20 | lower |
| p3 | 13 | 25 | upper |
| p4 | 15 | 27 | lower |
| p5 | 16 | 30 | upper |
| p6 | 17 | 31 | lower |

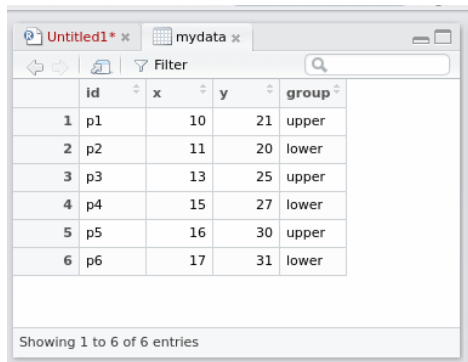
- This is a “tibble” or data frame, the standard way of storing a data set in R.
- Tibbles print as much as will display on the screen. If there are more rows or columns, it will say so.
- You will see navigation keys to display more rows or columns (if there are more).

View-ing your data frame

- Another way to examine your data frame is to View it, like this:

```
View(mydata)
```

...or find your data frame in the Global Environment top right and click it. - This pops up a “data frame viewer” top left:



| | id | x | y | group |
|---|----|----|----|-------|
| 1 | p1 | 10 | 21 | upper |
| 2 | p2 | 11 | 20 | lower |
| 3 | p3 | 13 | 25 | upper |
| 4 | p4 | 15 | 27 | lower |
| 5 | p5 | 16 | 30 | upper |
| 6 | p6 | 17 | 31 | lower |

Showing 1 to 6 of 6 entries

This View

- Read-only: cannot edit data
- Can display data satisfying conditions: click on Filter, then:
 - for a categorical variable, type name of category you want
 - for a quantitative variable, use slider to describe values you want.
- Can sort a column into ascending or descending order (click little arrows next to column name).
- Clicking the symbol with arrow on it left of Filter “pops out” View into separate (bigger) window.

Summarizing what we read in

- It is always a good idea to look at your data after you have read it in, to make sure you have believable numbers (and the right number of individuals and variables).
- Quick check for errors: these often show up as values too high or too low, so the min and/or max will be unreasonable.
- Five-number summary:

```
summary(mydata)
```

```
##           id                x                y
## Length:6           Min.      :10.00      Min.      :20.00
## Class :character    1st Qu.:11.50      1st Qu.:22.00
## Mode  :character    Median :14.00      Median :26.00
##                               Mean  :13.67      Mean  :25.67
##                               3rd Qu.:15.75      3rd Qu.:29.25
##                               Max.   :17.00      Max.   :31.00
##
##           group
```

Reading from a URL

- Any data file on the Web can be read directly.
- Example data:.
- Use URL instead of filename.
- I like to save the URL in a variable first (because URLs tend to be long), and then put that variable in the `read_` function:

```
my_url <- "http://www.utsc.utoronto.ca/~butler/c32/global.csv"  
global <- read_csv(my_url)
```

```
## Parsed with column specification:  
## cols(  
##   warehouse = col_character(),  
##   size = col_double(),  
##   cost = col_double()  
## )
```


The data

global

| warehouse | size | cost |
|-----------|------|-------|
| A | 225 | 11.95 |
| B | 350 | 14.13 |
| A | 150 | 8.93 |
| A | 200 | 10.98 |
| A | 175 | 10.03 |
| A | 180 | 10.13 |
| B | 325 | 13.75 |
| B | 290 | 13.30 |
| B | 400 | 15.00 |
| A | 125 | 7.97 |

Space-delimited files

- Another common format for data is a text file with the values separated by spaces. Top of some other data:

```
cup tempdiff
Starbucks 13
Starbucks 7
Starbucks 7
Starbucks 17.5
Starbucks 10
Starbucks 15.5
Starbucks 6
Starbucks 6
SIGG 12
SIGG 16
SIGG 9
SIGG 23
SIGG 11
```

Reading the coffee data

- This file was on my computer so I uploaded it to <http://rstudio.cloud> first.
- This time, `read_delim`, and we also have to say what the thing is separating the values:

```
coffee <- read_delim("coffee.txt", " ")
```

```
## Parsed with column specification:
## cols(
##   cup = col_character(),
##   tempdiff = col_double()
## )
```

- Name of the cup, text, and tempdiff, a decimal number.

Looking at the values (some)

coffee

| cup | tempdiff |
|-----------|----------|
| Starbucks | 13.0 |
| Starbucks | 7.0 |
| Starbucks | 7.0 |
| Starbucks | 17.5 |
| Starbucks | 10.0 |
| Starbucks | 15.5 |
| Starbucks | 6.0 |
| Starbucks | 6.0 |
| SIGG | 12.0 |
| SIGG | 16.0 |
| SIGG | 9.0 |
| SIGG | 23.0 |
| SIGG | 11.0 |

Reading from the Web; the soap data

- Use the URL in place of the filename.
- Save the URL in a variable first:

```
url <- "http://www.utsc.utoronto.ca/~butler/c32/soap.txt"
soap <- read_delim(url, " ")
```

```
## Parsed with column specification:
## cols(
##   case = col_double(),
##   scrap = col_double(),
##   speed = col_double(),
##   line = col_character()
## )
```

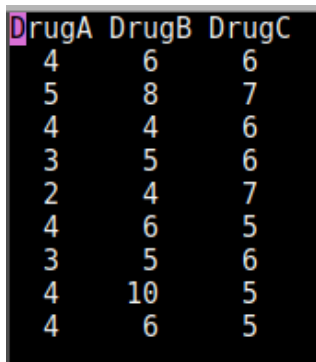
The soap data (some)

soap

| case | scrap | speed | line |
|------|-------|-------|------|
| 1 | 218 | 100 | a |
| 2 | 248 | 125 | a |
| 3 | 360 | 220 | a |
| 4 | 351 | 205 | a |
| 5 | 470 | 300 | a |
| 6 | 394 | 255 | a |
| 7 | 332 | 225 | a |
| 8 | 321 | 175 | a |
| 9 | 410 | 270 | a |
| 10 | 260 | 170 | a |
| 11 | 241 | 155 | a |
| 12 | 331 | 190 | a |
| 13 | 275 | 140 | a |

Data aligned in columns

- Sometimes you see data aligned in columns, thus:



| DrugA | DrugB | DrugC |
|-------|-------|-------|
| 4 | 6 | 6 |
| 5 | 8 | 7 |
| 4 | 4 | 6 |
| 3 | 5 | 6 |
| 2 | 4 | 7 |
| 4 | 6 | 5 |
| 3 | 5 | 6 |
| 4 | 10 | 5 |
| 4 | 6 | 5 |

- `read_delim` will not work: values separated by more than one space.
- The number of spaces between values is not constant, because there is one fewer space before the 10.
- `read_table` works for this.

Reading in column-aligned data

```
drugs <- read_table("migraine.txt")
```

```
## Parsed with column specification:
```

```
## cols(
```

```
##   DrugA = col_double(),
```

```
##   DrugB = col_double(),
```

```
##   DrugC = col_double()
```

```
## )
```

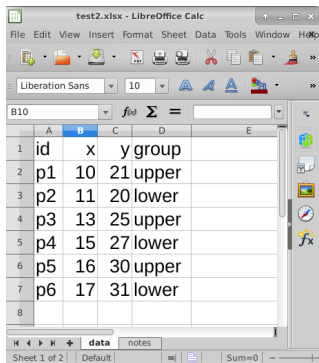

The data

drugs

| DrugA | DrugB | DrugC |
|-------|-------|-------|
| 4 | 6 | 6 |
| 5 | 8 | 7 |
| 4 | 4 | 6 |
| 3 | 5 | 6 |
| 2 | 4 | 7 |
| 4 | 6 | 5 |
| 3 | 5 | 6 |
| 4 | 10 | 5 |
| 4 | 6 | 5 |

Reading an Excel sheet directly

- Here is my spreadsheet from before, but tarted up a bit:



| | A | B | C | D | E |
|---|----|----|----------|---|---|
| 1 | id | x | y group | | |
| 2 | p1 | 10 | 21 upper | | |
| 3 | p2 | 11 | 20 lower | | |
| 4 | p3 | 13 | 25 upper | | |
| 5 | p4 | 15 | 27 lower | | |
| 6 | p5 | 16 | 30 upper | | |
| 7 | p6 | 17 | 31 lower | | |
| 8 | | | | | |

- It is now a workbook with a second sheet called “notes” (that we don't want).
- Install package `readxl` first.

Reading it in

- Read into R, saying that we only want the sheet “data”. Upload spreadsheet first.
- Excel spreadsheets must be “local”: cannot read one in from a URL.

```
library(readxl)
mydata2 <- read_excel("test2.xlsx", sheet = "data")
mydata2
```

| id | x | y | group |
|----|----|----|-------|
| p1 | 10 | 21 | upper |
| p2 | 11 | 20 | lower |
| p3 | 13 | 25 | upper |
| p4 | 15 | 27 | lower |
| p5 | 16 | 30 | upper |
| p6 | 17 | 31 | lower |