Post 2: R Integration with the Google Suite

Stat 133, Fall 2017

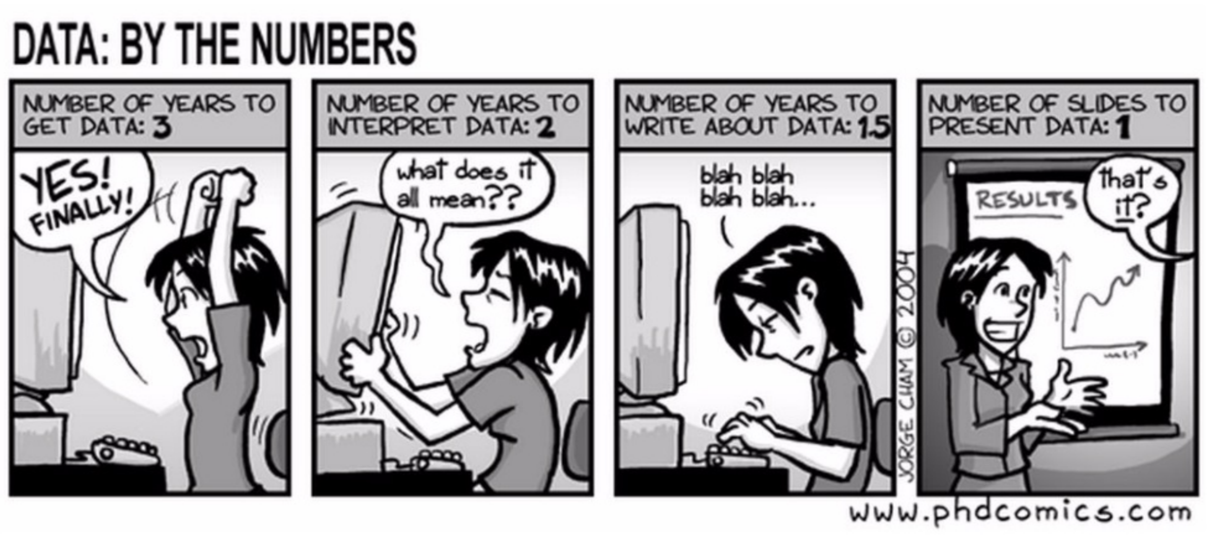
Tyler Larsen

November 30, 2017

Introduction

Up until this point, we have been given all the data we need, most often in the form of an already somewhat tidy .csv file. We've then been loading the data in one of two ways: either we have manually downloaded and then read in the file into an R script, or we have download files through links to Github.

But what happens if our data in neither hosted on Github, nor available in a readily available .csv file, and is instead hosted on Google Sheets? Luckily, there are packages to help us pull in that data directly to R, too. In particular, we will be looking at the `googlesheets` package, which is very helpful for the often time consuming "data gathering" phase of the data cycle.


DATA: BY THE NUMBERS
NUMBER OF YEARS TO GET DATA: 3
YES! FINALLY!
NUMBER OF YEARS TO INTERPRET DATA: 2
what does it all mean??
NUMBER OF YEARS TO WRITE ABOUT DATA: 1.5
blah blah blah blah...
NUMBER OF SLIDES TO PRESENT DATA: 1
RESULTS
that's it?
JORGE CHAM © 2004
www.phdcomics.com

http://www.phdcomics.com/comics/archive.php?comicid=462

Background

Installing googlesheets:

```r
#install.packages("googlesheets")
library("googlesheets")
```

`googlesheets` was built with `dplyr` in mind, so we must also install and load `dplyr`:

```r
#install.packages("dplyr")
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

A little bit about `googlesheets` function naming conventions:

- All functions that have anything to do with Google Sheets begin with `gs_`,
- All functions that operate on specific worksheets begin with `gs_ws_`,
- And all functions that communicate with Google Drive begin with `gd_`, but we won't be getting much into this.

And finally, two quick vocabulary terms:

- Sheet: refers to the entire file, including each tab (worksheet) within. In Excel, this would be the whole Excel file, while in Google it is the whole Google Sheet file.
- Worksheet: each page of the sheet, these can be navigated through the tabs found at the bottom of the sheet.

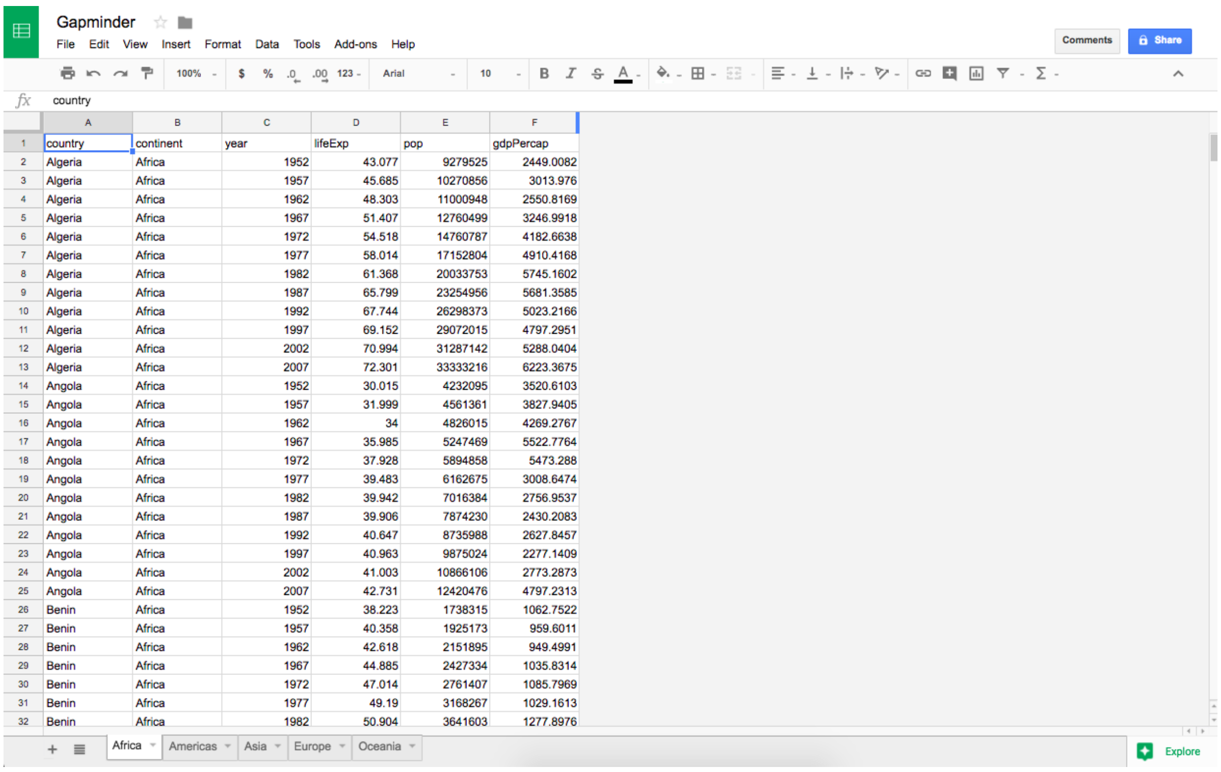
Example:

Now that we know a bit about the `googlesheets` package and what it can do, let's see for ourselves. The developers have made demonstrations easy, and within the package their is already a Google Sheet loaded with data from Gapminder. In order to access this sheet, we must first be authenticated. Running the code below will direct you to the Google Sheets sign in page, and then add the "Gapminder" sheet to your Google Sheets account.

```
gs_gap() %>%
  gs_copy(to = "Gapminder")
```

```
## Successful copy! New sheet is titled "Gapminder".
```

Now if you to to your Google Sheets account on your browswer, you should now see a new sheet titled "Gapminder" waiting for you. When you open it up,it should look like this:



fx country

	A	B	C	D	E	F
1	country	continent	year	lifeExp	pop	gdpPercap
2	Algeria	Africa	1952	43.077	9279525	2449.0082
3	Algeria	Africa	1957	45.685	10270856	3013.976
4	Algeria	Africa	1962	48.303	11000948	2550.8169
5	Algeria	Africa	1967	51.407	12760499	3246.9918
6	Algeria	Africa	1972	54.518	14760787	4182.6638
7	Algeria	Africa	1977	58.014	17152804	4910.4168
8	Algeria	Africa	1982	61.368	20033753	5745.1602
9	Algeria	Africa	1987	65.799	23254956	5681.3585
10	Algeria	Africa	1992	67.744	26298373	5023.2166
11	Algeria	Africa	1997	69.152	29072015	4797.2951
12	Algeria	Africa	2002	70.994	31287142	5288.0404
13	Algeria	Africa	2007	72.301	33333216	6223.3675
14	Angola	Africa	1952	30.015	4232095	3520.6103
15	Angola	Africa	1957	31.999	4561361	3827.9405
16	Angola	Africa	1962	34	4826015	4269.2767
17	Angola	Africa	1967	35.985	5247469	5522.7764
18	Angola	Africa	1972	37.928	5894858	5473.288
19	Angola	Africa	1977	39.483	6162675	3008.6474
20	Angola	Africa	1982	39.942	7016384	2756.9537
21	Angola	Africa	1987	39.906	7874230	2430.2083
22	Angola	Africa	1992	40.647	8735988	2627.8457
23	Angola	Africa	1997	40.963	9875024	2277.1409
24	Angola	Africa	2002	41.003	10866106	2773.2873
25	Angola	Africa	2007	42.731	12420476	4797.2313
26	Benin	Africa	1952	38.223	1738315	1062.7522
27	Benin	Africa	1957	40.358	1925173	959.6011
28	Benin	Africa	1962	42.618	2151895	949.4991
29	Benin	Africa	1967	44.885	2427334	1035.8314
30	Benin	Africa	1972	47.014	2761407	1085.7969
31	Benin	Africa	1977	49.19	3168267	1029.1613
32	Benin	Africa	1982	50.904	3641603	1277.8976

Next, we want to register the sheet in R as a `googlesheet` object. There are three functions we can use to do this: `gs_title`, `gs_key`, and `gs_url`. These three In this example, we'll use `gs_title`:

```
gap <- gs_title("Gapminder")
```

```
## Sheet successfully identified: "Gapminder"
```

Here's an overview of what the sheet looks like:

```
gap
```

```
##                   Spreadsheet title: Gapminder
##                  Spreadsheet author: tylerlarsen579
##   Date of googlesheets registration: 2017-11-28 09:21:58 GMT
##     Date of last spreadsheet update: 2017-11-28 09:21:55 GMT
##                          visibility: private
##                         permissions: rw
##                             version: new
##
## Contains 5 worksheets:
## (Title): (Nominal worksheet extent as rows x columns)
## Africa: 625 x 6
## Americas: 301 x 6
## Asia: 397 x 6
## Europe: 361 x 6
## Oceania: 25 x 6
##
## Key: 157DtAc_nXhlr2ePCTwzEDXDTJx0JU0vCmHeTvQurRog
## Browser URL: https://docs.google.com/spreadsheets/d/157DtAc_nXhlr2ePCTwzEDXDTJx0JU0vCmHeTvQurRog/
```

Next, we'll look at how we can create a Google Sheet of our own, starting from scratch. In order to do this, we will use the built in data frame `mtcars`, as well as the function `gs_new`. We can then view our new Google Sheet in the browser using the function `gs_browse()`.

```
# takes three arguments (only "input" is reqired)
# "cars" is the name of the new Google Sheet
# input is the dataframe we are sending to Google sheets
# trim eleimates the unnecessary rows and columns surrounding our data
cars_sheet <- gs_new("cars", input = mtcars, trim = TRUE)
```

```
## Sheet "cars" created in Google Drive.
```

```
## Range affected by the update: "R1C1:R33C11"
```

```
## Worksheet "Sheet1" successfully updated with 363 new value(s).
```

```
## Accessing worksheet titled 'Sheet1'.
```

```
## Sheet successfully identified: "cars"
```

```
## Accessing worksheet titled 'Sheet1'.
```

```
## Worksheet "Sheet1" dimensions changed to 33 x 11.
```

```
## Worksheet dimensions: 33 x 11.
```

```
cars_sheet %>% gs_browse()
```

Proof that this worked:

```
cars_sheet
```

```
##                   Spreadsheet title: cars
##                  Spreadsheet author: tylerlarsen579
##   Date of googlesheets registration: 2017-11-28 09:22:10 GMT
##     Date of last spreadsheet update: 2017-11-28 09:22:08 GMT
##                          visibility: private
##                         permissions: rw
##                             version: new
##
## Contains 1 worksheets:
## (Title): (Nominal worksheet extent as rows x columns)
## Sheet1: 33 x 11
##
## Key: 1GycCJX4V17vp4pJZkZlPkSV8OuheVPiSQq6iNIKN_Ao
## Browser URL: https://docs.google.com/spreadsheets/d/1GycCJX4V17vp4pJZkZlPkSV8OuheVPiSQq6iNIKN_Ao/
```

Discussion

This package also offers a host of other features and functions, more than I could possibly show in one blog post. Additional arguments to the `gs_new` function allow you to easily change the visability and the permissions on a Google Sheet. `gs_download()` lets the user download the Google Sheet as a csv, rather than editing the Google Sheet itself. `gs_edit_cells()` allows you to durectly edit the cells of a Google Sheet. `gs_delete()` lets you remotely delete files from your Google Sheets. The `gs_webapp_` function family allows for easy integration of Google Sheets into Shiny apps.

Take Home Message

Used in conjunction with other packages, `googlesheets` can be a very powerful tool. Packages such as `dplyr` are valuable when it comes to wrangling the collected data, while `ggplot` can be exremely useful when used in conjunction with `googlesheets` in order to visulualize the collected data.

One particularly interesting and useful application of this package is to Google surveys, which output their data to Google Sheets. This package is especially useful because it allows for live monitoring of survey results from the comfort of Rstudio, rather than having to use the limited analysis features provided by Google Sheets.

With `googlesheets`, the possibilities are endless.

Sources

https://www.linkedin.com/pulse/update-google-sheets-via-r-automatically-tanya-cashorali/ https://github.com/ucb-stat133/stat133-fall-2017/blob/master/slides/01-big-picture.pdf https://github.com/jennybc/googlesheets https://trinkerrstuff.wordpress.com/ https://rawgit.com/jennybc/googlesheets/master/vignettes/basic-usage.html#download-sheets-as-csv-pdf-or-xlsx-file https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf https://simplystatistics.org/2016/08/26/googlesheets/