COMPSCIX 415.2 Homework 1

Robert Clements

DUE DATE: Feb 6, 2018 @ 6:15PM

In this assignment…

…we will work on getting your work environment all set up correctly with R, RStudio, git, Github, and a comprehensive set of R packages that we will use for the remainder of this course. From here we will work on getting comfortable with these tools by creating your first R Markdown document.

Feel free to play around with R Markdown settings and themes. There is no *right* answer on this homework, this is purely meant for practice. However, your final document should include all of the components that I specifically ask for.

What to Turn In

You will turn in a compiled (or knitted) R Markdown document, in html or pdf format (not Word).

Prerequisite

Let's make sure everything is working first.

Install R

If you haven't already, install R from here: https://cran.rstudio.com/.

Install RStudio

Now install RStudio from here: https://www.rstudio.com/products/rstudio/download/#download.

Install git

If you don't already have it, install git from this link here: https://git-scm.com/downloads. You can check if you already have it by opening a terminal and typing `which git`.

If you like GUI's, the default download will have some available. I've never used these and can't vouch for them. There are other choices here: https://git-scm.com/downloads/guis. These are not required for this course, you will be able to use git from RStudio, but if you're adventurous or well-versed in git, feel free to use what you want.

Now that you have git installed, let's do some configuration. First we'll set up a git username **(note that this is not the same as your Github username)**. The git username is a way to associate you with your commits.

Open a terminal or shell of some kind. git Bash would be fine for this. You can also use the Terminal pane (right next to the Console pane) in RStudio, and type the following. Be sure to replace "Your Name" with your name.

```
git config --global user.name "Your Name"
```

Confirm that you have set the Git username correctly:

```
$ git config --global user.name
```

Now we will do the same thing with your email address. If you do not wish for your email address to be used here, follow the instructions at https://help.github.com/articles/about-commit-email-addresses/ for using a no-reply email address.

```
git config --global user.email "email@example.com"
```

Confirm that you have set the email address correctly in Git:

```
$ git config --global user.email
```

Get on Github

If you don't have an account, please create one on Github by going here: https://github.com/. This is required, and I'll ask for a URL later in this assignment. Remember to read this for advice on choosing a username: http://happygitwithr.com/github-acct.html.

Make sure RStudio works with git

Follow these steps:

- Open RStudio
- Click Tools -> Global Options -> Git/SVN
- If Git executable shows "(none)":
- click Browse and select the git executable installed on your system
- On a Mac, this will likely be one of
- /usr/bin/git
- /usr/local/bin/git
- /usr/local/git/bin/git
- On Windows, git.exe will likely be somewhere in Program Files
- Click OK

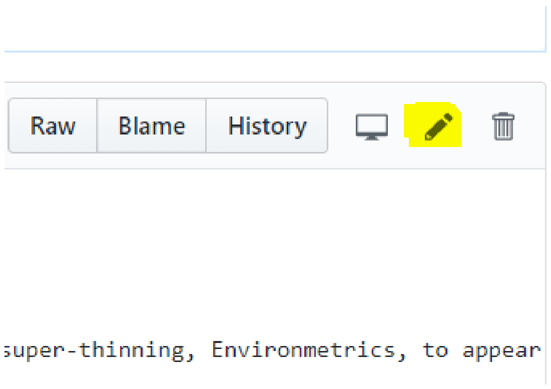
Install the Tidyverse and mdsr

We will be using many functions from the tidyverse packages. Install it like this:

```r
install.packages("tidyverse")
install.packages("mdsr")
```

Now, to load the core tidyverse packages and mdsr into your R session, type:

```r
library(tidyverse)
library(mdsr)
```



super-thinning, Environmetrics, to appear

Figure 1:

If you really want to compile your docs into pdf's...

...you will need to install a *full* version of TeX from here: https://www.latex-project.org/get/#tex-distributions. This is not a fast install, so don't do this if you're in a hurry. Note that html will be fine, and better if you're using git (like you should be). PDF files can not be version controlled in the same way html and other text-based files can. Also, RStudio can sometimes be slow at compiling into a pdf which can slow down your progress. And lastly, html can handle *interactive* documents, which are really useful.

If you absolutely insist on turning in pdf's, compile your docs into html *while you are still working on them*, and then once you are all done, compile the final document as a pdf to turn in.

Ok, now we should be all set to go!

Exercises

Here are the exercises for you to complete.

Create your first git repository

Go to your Github account and create a new repository called compscix-415-2-assignments. This will be the repository for all of your homework assignments for the next 10 weeks along with your midterm and final. Initialize a readme file.

Go to the readme.md file, click on the pencil icon (Figure 1) in the upper right corner of the box so you can edit the readme. Write a description for this repository and then commit your changes.

Now, we want to clone this repository onto your laptop. Click on the green Clone or Download button (Figure 2) and copy the URL.

There are many ways to clone the repo. We'll use RStudio.

Open RStudio, click on File -> New Project -> Version Control -> Git. You can figure out the rest from here.

Start a new R Markdown document

Go to File -> New File -> R Markdown. Give it the title: "COMPSCIX 415.2 Homework 1" and click OK.

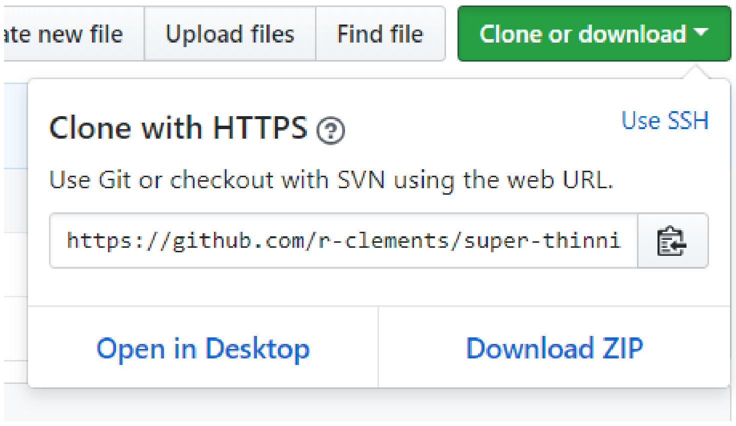


Figure 2:

A new R Markdown file will open with a default template that you can read. Save this file with this name: homework_1_lastname_firstname. Replace lastname and firstname with your last name and first name.

Change the author name and date at the top of the document in the YAML.

Commit and push

Let's commit this file and push it to Github. Use the Git pane in RStudio to do both.

Tell me where your repository is

Delete everything in your R Markdown file after the first code chunk that says

```
knitr::opts_chunk$set(echo = TRUE)
```

Now type the following line:

```
My Github repository for my assignments can be found at this URL: [your URL](your URL)
```

and replace "your URL" with your github repository URL which you can copy from your web browser.

Let's look at some data

Load the mdsr and tidyverse packages by including the following line in an R chunk and then running the chunk by clicking the green arrow in the upper right corner of the code chunk, or you can put your cursor in the chunk and click the dropdown arrow next to the Run button in the upper right corner of the source pane, or you can use the keyboard shortcut (find it):

```
library(mdsr)
library(tidyverse)
```

At this point you may want to knit your document to see what it looks like. Click the knit button at the top of the source pane. Your doc should knit and open in a new window.

You may notice that some extraneous messages from R will show up on the document. This happens because when you knit, R is running the code and displaying the output from that code on your doc. Often, when you load packages, there will be some output to the console. This isn't really useful for your doc, so let's suppress these types of messages and warnings.

Add the following options to your R chunk above to suppress extraneous R messages from your final compiled document:

“‘`{r load_packages, warning=FALSE, message=FALSE}`

The mdsr package is the accompanying package to the text book Modern Data Science with R and has a lot of data sets that we may use throughout the course. Yes, R packages sometimes contain data sets. There are some R packages that contain nothing **but** data sets - no functions at all.

Ok, we'll load the WorldCities data by including the following in an R chunk and running it (same chunk as above, or a new one, it's up to you how you want to organize your R Markdown file):

```r
data("WorldCities")
```

Let's look at the help page for this data set. Yes, even data sets have help pages! No need to include this part in your document, just run the code in the console for now.

```r
?WorldCities
```

QUESTION 1: How many observations and variables are there in this data set? What are some of the variable names? Type up your answer in your document using complete sentences.

Let's take a glimpse of the data by including the following in another R chunk and running it:

```r
glimpse(WorldCities)
```

We can also view the data in RStudio by either clicking on the data set name in the Environment pane, or by typing `View(WorldCities)` in the console. Try it both ways.

WorldCities is a data.frame, it has rows and columns, and the columns can be of different types (numeric, character, factor, logical, integer, double, etc.). Each row is an observation of all of the columns, or variables. Extracting columns or rows is one of the most common tasks we have to do, so let's do this next.

Extract parts of the data

First let's filter down the dataset so that we aren't outputting pages of data to our document. Here's the easiest way to grab the top 200 rows of the data. Include this and run it.

```r
WorldCities <- head(WorldCities, 200) # 200 rows
```

The quickest way to pull a column from a data.frame is to use $. Include this in your document and run it.

```r
country_col <- WorldCities$country
```

The above will assign the values of the country column of WorldCities to the variable name country_col. `<-` is the assignment operator. Now let's take a look at the unique countries. Include this in your document and run it.

```r
unique(country_col)
```

```
## [1] "AD" "AE" "AF" "AG" "AI" "AL" "AM" "AO" "AR"
```

QUESTION 2: There is a column called region in the data set. Can you extract this and show only the unique values?

Compile and commit

At this point you may want to compile your document to see what it looks like. Click the knit button at the top of the source pane. Your doc should knit and open in a new window.

From here, change the style, text, or fix any errors in your code that show up. Use the R Markdown cheat sheet to help with this. You can find it by clicking Help -> Cheatsheets -> R Markdown Cheat Sheet.

Commit your changes with a useful comment such as "finished questions 1 and 2".

Extract data the tidy way

We can also extract columns in these ways:

```
country_col <- select(WorldCities, country)
country_col <- WorldCities %>% select(country)
```

Here, `select` is a function (we'll call it a verb in the future), which selects the column "country" from WorldCities. Remember that you can look at the help file:

`?select`

If there are multiple packages that have a `select` function, all of these will be listed and you should pick the one from package `dplyr`.

The `%>%` is a pipe operator. It works by plugging in WorldCities as the first argument in the `select` function - it is equivalent to `select(WorldCities, country)`. For example, `country_col %>% unique()` will output the unique values in `country_col` because it is equivalent to running `unique(country_col)`. These pipes will be used very often, and make your code much more readable. You can also string together multiple pipes, you just have to remember to read your code from left to right, and that is the order that your code will be run.

`first_this %>% then_this %>% then_this`

and your results will sequentially be plugged into the next function as the first argument. Example:

```
WorldCities %>% select(region) %>% head(5)
```

```
##           region
## 1 Europe/Andorra
## 2 Europe/Andorra
## 3     Asia/Dubai
## 4     Asia/Dubai
## 5     Asia/Dubai
```

This works by first running `select(WorldCities, region)` and then taking the result of that and plugging it into `head` as the first argument.

QUESTION 3: Can you extract and show the unique entries from the country column in WorldCities using one line of code and two %>% operators? The output will look like this:

```
##     country
## 1        AD
## 3        AE
## 15       AF
## 65       AG
## 66       AI
## 67       AL
## 87       AM
## 104      AO
## 131      AR
```

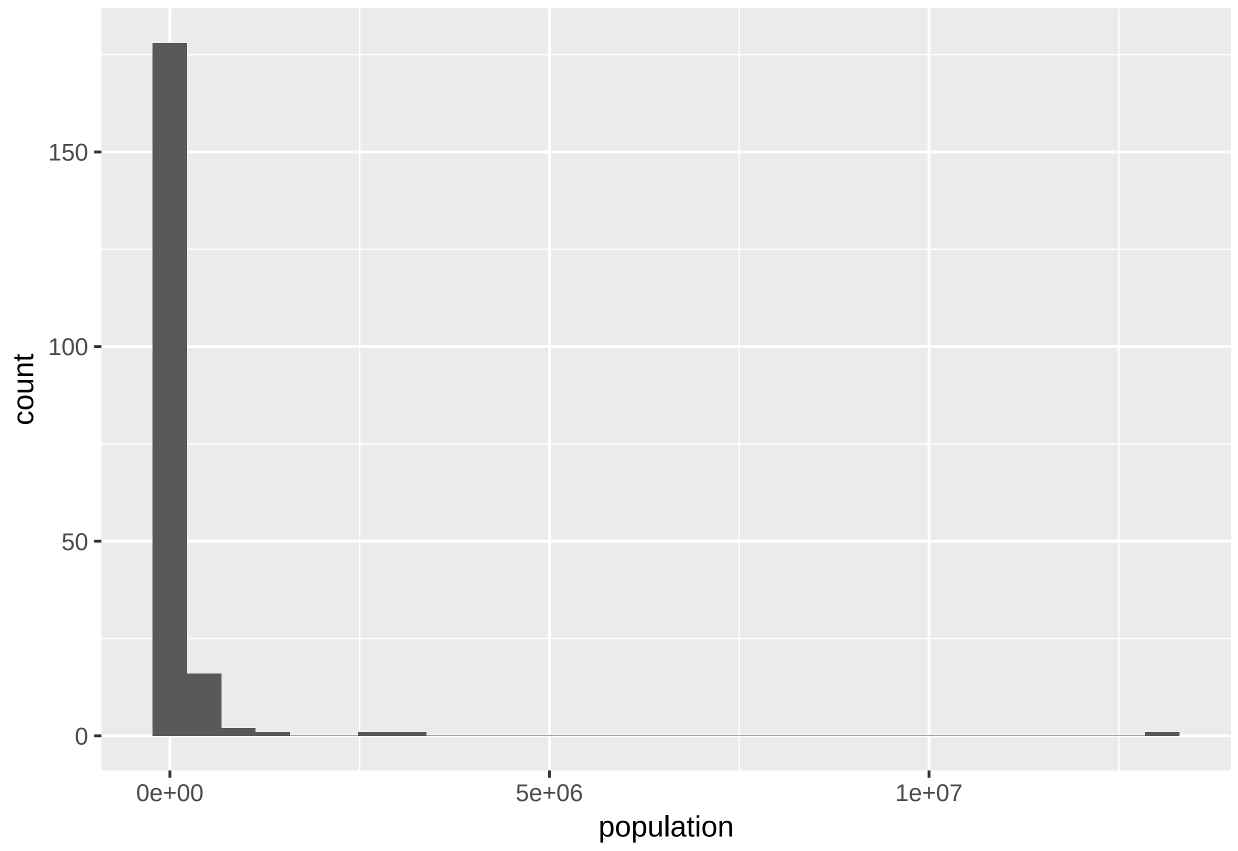
Visualize it

Let's get an idea of the distribution of city populations. We'll talk about the best types of plots for distributions next week. For now, we'll just make a basic histogram using the `ggplot` function.

Include this in your markdown and run it.

```
WorldCities %>% ggplot(aes(x = population)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We build plots using layers (the + sign) with `ggplot`. For instance, if we want to change the x-axis label from "population" to "City Population", we just add another layer like this:

```
WorldCities %>% ggplot(aes(x = population)) +
  geom_histogram() +
  xlab('City Population')
```

We will get into details about this later.

QUESTION 4: Make one more histogram of the population and add a new x-label, a new y-label, a new title (use `ggtitle('my title')`), and change the theme of the plot using `theme_bw()`.

Turn in your completed assignment

Save your document and knit it. If you're happy with it, commit your changes with the comment "finished assignment" and push your R Markdown file and your html file to Github.

You can turn in your assignment to me in one of two ways in Canvas: (1) upload your html document; or (2) type in the URL to your html file in Github.