Lab 2

Steven Boyd

10/7/2021

Getting started

Today, we are going to work on loading and tidying data. Step 1 is to download the following zip file from the course git repository "gdp__per__cap__1995__2015.zip". Save it in a place that you can locate it with a relative file path. Use "getwd()" if you are unsure what your working directory is.

First, we will make sure everyone can extract data from a zip file. One option is to open the zipped file and use your OS' built in extraction capabilities (assuming it has those). However, it is also possible to access the data within R Studio. In fact, readr can grab data inside a zip without extracting at all.

Note that your file path may be different. Adjust accordingly!

```
gdp_per_cap <- read_csv('data/gdp_per_cap_1995_2015.zip')
```

```
## Multiple files in zip: reading 'b9988620-72da-4463-b88c-93fa96b7075f_Data.csv'
```

```
## Rows: 269 Columns: 25
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (25): Series Name, Series Code, Country Name, Country Code, 1995 [YR1995...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Notice the warning text. It is telling us that there are multiple csv files within the zip file. This data was downloaded from the World Bank earlier this year in this format. Read the name of the file carefully. What does it say at the end of the file name?

If you look inside the zip file, you'll notice another csv with *almost* the same name. The only difference is in the suffix. The one marked "metadata" contains information *about* the data, but no data we can actually analyze. It is common for csv files like this to be generated automatically when you download data from a huge database like the World Bank's DataBank.

We don't really need to worry that R didn't read the metadata, because it is only useful as documentation. If you do have multiple csv files in a zip file, you can sepcify which one you want to read, but you need to unzip first.

```
gdp_per_cap_2 <- read_csv(unz(description =  'data/gdp_per_cap_1995_2015.zip',
                  filename =   'b9988620-72da-4463-b88c-93fa96b7075f_Data.csv'))
```

```
## Rows: 269 Columns: 25

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (25): Series Name, Series Code, Country Name, Country Code, 1995 [YR1995...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

We can confirm that these contain the same data by examining each:

```r
str(gdp_per_cap)
```

```
## spec_tbl_df [269 x 25] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Series Name  : chr [1:269] "GDP per capita, PPP (current international $)" "GDP per capita, PPP (
##  $ Series Code  : chr [1:269] "NY.GDP.PCAP.PP.CD" "NY.GDP.PCAP.PP.CD" "NY.GDP.PCAP.PP.CD" "NY.GDP.PC
##  $ Country Name : chr [1:269] "Afghanistan" "Albania" "Algeria" "American Samoa" ...
##  $ Country Code : chr [1:269] "AFG" "ALB" "DZA" "ASM" ...
##  $ 1995 [YR1995]: chr [1:269] ".." "2665.10903649849" "7082.78475736955" ".." ...
##  $ 1996 [YR1996]: chr [1:269] ".." "2979.3330899284" "7377.69949168115" ".." ...
##  $ 1997 [YR1997]: chr [1:269] ".." "2716.69355834805" "7465.89932260811" ".." ...
##  $ 1998 [YR1998]: chr [1:269] ".." "3020.93057263444" "7816.77057806543" ".." ...
##  $ 1999 [YR1999]: chr [1:269] ".." "3471.65635908955" "8068.2961352061" ".." ...
##  $ 2000 [YR2000]: chr [1:269] ".." "3862.48437763265" "8446.58797859344" ".." ...
##  $ 2001 [YR2001]: chr [1:269] ".." "4301.38282973192" "8775.11702646461" ".." ...
##  $ 2002 [YR2002]: chr [1:269] "877.014423503263" "4661.37914319822" "9293.83679843142" ".." ...
##  $ 2003 [YR2003]: chr [1:269] "927.857547917432" "4994.92469274671" "10019.3578261386" ".." ...
##  $ 2004 [YR2004]: chr [1:269] "925.441616196776" "5423.20168382235" "10591.034224285" ".." ...
##  $ 2005 [YR2005]: chr [1:269] "1023.05162974192" "5865.31759085604" "11405.6403648217" ".." ...
##  $ 2006 [YR2006]: chr [1:269] "1077.76190658441" "6557.80133674236" "11776.0413453018" ".." ...
##  $ 2007 [YR2007]: chr [1:269] "1228.70413531195" "7274.52468523841" "12311.0390253609" ".." ...
##  $ 2008 [YR2008]: chr [1:269] "1272.57320417194" "8228.32755984622" "12643.1491551178" ".." ...
##  $ 2009 [YR2009]: chr [1:269] "1519.69254818311" "8819.51009173416" "12722.3781829231" ".." ...
##  $ 2010 [YR2010]: chr [1:269] "1710.57564538432" "9636.10870469977" "13095.4468249877" ".." ...
##  $ 2011 [YR2011]: chr [1:269] "1699.48799733991" "10207.7335023376" "13500.0416554875" ".." ...
##  $ 2012 [YR2012]: chr [1:269] "1914.77435127371" "10526.3189737173" "13303.3317888752" ".." ...
##  $ 2013 [YR2013]: chr [1:269] "2015.51496204541" "10570.9681040326" "13056.8036185828" ".." ...
##  $ 2014 [YR2014]: chr [1:269] "2069.42464180385" "11259.2967004521" "13003.267118909" ".." ...
##  $ 2015 [YR2015]: chr [1:269] "2087.30532306683" "11658.8660596051" "12015.6313951015" ".." ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   `Series Name` = col_character(),
##   ..   `Series Code` = col_character(),
##   ..   `Country Name` = col_character(),
##   ..   `Country Code` = col_character(),
##   ..   `1995 [YR1995]` = col_character(),
##   ..   `1996 [YR1996]` = col_character(),
##   ..   `1997 [YR1997]` = col_character(),
##   ..   `1998 [YR1998]` = col_character(),
##   ..   `1999 [YR1999]` = col_character(),
##   ..   `2000 [YR2000]` = col_character(),
##   ..   `2001 [YR2001]` = col_character(),
##   ..   `2002 [YR2002]` = col_character(),
##   ..   `2003 [YR2003]` = col_character(),
##   ..   `2004 [YR2004]` = col_character(),
##   ..   `2005 [YR2005]` = col_character(),
##   ..   `2006 [YR2006]` = col_character(),
##   ..   `2007 [YR2007]` = col_character(),
##   ..   `2008 [YR2008]` = col_character(),
##   ..   `2009 [YR2009]` = col_character(),
##   ..   `2010 [YR2010]` = col_character(),
##   ..   `2011 [YR2011]` = col_character(),
##   ..   `2012 [YR2012]` = col_character(),
##   ..   `2013 [YR2013]` = col_character(),
##   ..   `2014 [YR2014]` = col_character(),
##   ..   `2015 [YR2015]` = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```r
str(gdp_per_cap_2)
```

```
## spec_tbl_df [269 x 25] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Series Name  : chr [1:269] "GDP per capita, PPP (current international $)" "GDP per capita, PPP (
##  $ Series Code  : chr [1:269] "NY.GDP.PCAP.PP.CD" "NY.GDP.PCAP.PP.CD" "NY.GDP.PCAP.PP.CD" "NY.GDP.PCA
##  $ Country Name : chr [1:269] "Afghanistan" "Albania" "Algeria" "American Samoa" ...
##  $ Country Code : chr [1:269] "AFG" "ALB" "DZA" "ASM" ...
##  $ 1995 [YR1995]: chr [1:269] ".." "2665.10903649849" "7082.78475736955" ".." ...
##  $ 1996 [YR1996]: chr [1:269] ".." "2979.3330899284" "7377.69949168115" ".." ...
##  $ 1997 [YR1997]: chr [1:269] ".." "2716.69355834805" "7465.89932260811" ".." ...
##  $ 1998 [YR1998]: chr [1:269] ".." "3020.93057263444" "7816.77057806543" ".." ...
##  $ 1999 [YR1999]: chr [1:269] ".." "3471.65635908955" "8068.2961352061" ".." ...
##  $ 2000 [YR2000]: chr [1:269] ".." "3862.48437763265" "8446.58797859344" ".." ...
##  $ 2001 [YR2001]: chr [1:269] ".." "4301.38282973192" "8775.11702646461" ".." ...
##  $ 2002 [YR2002]: chr [1:269] "877.014423503263" "4661.37914319822" "9293.83679843142" ".." ...
##  $ 2003 [YR2003]: chr [1:269] "927.857547917432" "4994.92469274671" "10019.3578261386" ".." ...
##  $ 2004 [YR2004]: chr [1:269] "925.441616196776" "5423.20168382235" "10591.034224285" ".." ...
##  $ 2005 [YR2005]: chr [1:269] "1023.05162974192" "5865.31759085604" "11405.6403648217" ".." ...
##  $ 2006 [YR2006]: chr [1:269] "1077.76190658441" "6557.80133674236" "11776.0413453018" ".." ...
##  $ 2007 [YR2007]: chr [1:269] "1228.70413531195" "7274.52468523841" "12311.0390253609" ".." ...
##  $ 2008 [YR2008]: chr [1:269] "1272.57320417194" "8228.32755984622" "12643.1491551178" ".." ...
##  $ 2009 [YR2009]: chr [1:269] "1519.69254818311" "8819.51009173416" "12722.3781829231" ".." ...
##  $ 2010 [YR2010]: chr [1:269] "1710.57564538432" "9636.10870469977" "13095.4468249877" ".." ...
##  $ 2011 [YR2011]: chr [1:269] "1699.48799733991" "10207.7335023376" "13500.0416554875" ".." ...
##  $ 2012 [YR2012]: chr [1:269] "1914.77435127371" "10526.3189737173" "13303.3317888752" ".." ...
##  $ 2013 [YR2013]: chr [1:269] "2015.51496204541" "10570.9681040326" "13056.8036185828" ".." ...
##  $ 2014 [YR2014]: chr [1:269] "2069.42464180385" "11259.2967004521" "13003.267118909" ".." ...
##  $ 2015 [YR2015]: chr [1:269] "2087.30532306683" "11658.8660596051" "12015.6313951015" ".." ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   `Series Name` = col_character(),
##   ..   `Series Code` = col_character(),
##   ..   `Country Name` = col_character(),
##   ..   `Country Code` = col_character(),
##   ..   `1995 [YR1995]` = col_character(),
##   ..   `1996 [YR1996]` = col_character(),
##   ..   `1997 [YR1997]` = col_character(),
##   ..   `1998 [YR1998]` = col_character(),
##   ..   `1999 [YR1999]` = col_character(),
##   ..   `2000 [YR2000]` = col_character(),
##   ..   `2001 [YR2001]` = col_character(),
##   ..   `2002 [YR2002]` = col_character(),
##   ..   `2003 [YR2003]` = col_character(),
##   ..   `2004 [YR2004]` = col_character(),
##   ..   `2005 [YR2005]` = col_character(),
##   ..   `2006 [YR2006]` = col_character(),
##   ..   `2007 [YR2007]` = col_character(),
##   ..   `2008 [YR2008]` = col_character(),
##   ..   `2009 [YR2009]` = col_character(),
##   ..   `2010 [YR2010]` = col_character(),
##   ..   `2011 [YR2011]` = col_character(),
##   ..   `2012 [YR2012]` = col_character(),
##   ..   `2013 [YR2013]` = col_character(),
##   ..   `2014 [YR2014]` = col_character(),
##   ..   `2015 [YR2015]` = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

"Tidy"ing the data

Is this data in "tidy" format? Why or why not?

Your Answer

What can we do to `gdp_per_cap` to make it tidy?

Your Answer

Finish this code to make the data tidy:

gdp_per_cap_tidy <- gdp_per_cap %>% pivot__________(c(`1995 [YR1995]`, `_______`, `1997 [YR1997]`, `1998 [YR1998]`, `1999 [YR1999]`, `2000 [YR2000]`, `_______`, `2002 [YR2002]`, `2003 [YR2003]`, `2004 [YR2004]`, `2005 [YR2005]`, `2006 [YR2006]`, `2007 [YR2007]`, `2008 [YR2008]`, `_______`, `2010 [YR2010]`, `2011 [YR2011]`, `2012 [YR2012]`, `2013 [YR2013]`, `_______`, `2015 [YR2015]`), names_to = "________", _________ = "gdppc")

There are more efficent ways to specify your columns. Try to come up with at least one.

Cleaning up the data

Let's look at our new tidy data. Now we are getting closer to something we can work with! Is there anything you would want to change about it?

```
#head(gdp_per_cap_tidy)
```

Your Answer

\textit{There may be easier methods that make use of regular expressions. If you are familiar with regex and want to try, please feel free. If you don't know regex, you should be able manage this problem with dplyr functions like recode() or case_when():}

Fill in this code to make the "year" variable more useable:

gdp_per_cap_tidy <- ___________ %>% ________(year = ________(year, '1995 [YR1995]' = 1995, '1996 [YR1996]' = 1996, '_________________' = 1997, '1998 [YR1998]' = 1998, '1999 [YR1999]' = _____, '2000 [YR2000]' = 2000, '2001 [YR2001]' = 2001, '2002 [YR2002]' = 2002, '2003 [YR2003]' = 2003, '2004 [YR2004]' = 2004, '2005 [YR2005]' = 2005, '_________________' = 2006, '2007 [YR2007]' = 2007, '2008 [YR2008]' = 2008, '2009 [YR2009]' = _____, '2010 [YR2010]' = 2010, '2011 [YR2011]' = 2011, '2012 [YR2012]' = 2012, '2013 [YR2013]' = _____, '2014 [YR2014]' = 2014, '2015 [YR2015]' = 2015))

Do we need all of the variables in the dataset? Which would you remove and why? Are there any you would rename? Fill in the code to rename and remove variables you want to change:

gdp_per_cap_tidy <- gdp_per_cap_tidy %>% rename(_________) %>% select(_________)

Subsetting the Data

This dataset now looks much cleaner, is easier to work with, and does not contain as much unnecessary information. Now suppose that we only want to keep countries for which we have at least some data (i.e. remove countries which are missing "gdppc" for all years in the data). How would you approach this problem? Fill in the code to do it for you.

gdp_per_cap_tidy_subset <- _____________ %>% group_by(_________) %>% _________(!all(________(gdppc)))

This doesn't seem to work. Any guesses as to why? We can check if values we expect to be "NA" actually are by running this code:

It appears that a lot of of our missing data is not being recognized as NA. What values appear where you expect to see NA? One thing we can check when our code isn't running as expected is the class of your values. Run the following code to check what class "gdppc" is:

```
#class(gdp_per_cap_tidy$gdppc)
```

It is not numeric, but we want it to be. How can we coerce these values to be numeric? Does anyone know why this might solve the code issue we just experienced?

Your Answer

As you can see, when we told R that the column should contain numbers, there were some values that it didn't know how to handle (because they weren't numbers). By default, it sets these values to "NA" which actually makes them easier for us to filter out.

Now, let's try removing countries with no data again (try running that code that didn't work):

We've managed to remove all the countries (the dataset actually includes regions and other aggregate categories of states as well) that had no data. For our final task, let's compute the year-over-year change in GDP per capita for each observation as a percentage. After adding it to the dataset, create a histogram so you can see the distribution of annual growth or decline in GDP per capita. What other visualization would you use to understand something else about this data?

Complete this code:

gdp_per_cap_change <- gdp_per_cap_tidy_subset %>% _______(_______) %>% _______(country, year, by_group = TRUE) %>% _______(rate = 100 * (gdppc - ____ (gdppc)) /_____(gdppc)) %>% ungroup()

Code for your plot goes here:

Merging datasets

Located in the same folder as the GDP data is zipped GNI data. It was downloaded from the same source and has a similar structure. See if you can follow the steps above to convert it to a tidy dataset and then merge the two using a join function. Would you be able to join them if you were working with the raw data?

I will post my code on Friday morning.