You’ve heard me say it a thousand times: to master data science, you need to practice.

You need to “practice small” by [practicing individual techniques and functions](http://www.sharpsightlabs.com/blog/simple-practice-maps-tidyverse/).

To get some practice, my recommendation is to find reasonably sized datasets online and plot them.

Wikipedia is a nearly-endless source of good datasets. The great thing about Wikipedia is that many of the datasets are small and well contained. They are also fairly clean, with just enough messiness to make them a bit of a challenge.

As a quick example, this week, we’ll plot some economic data.

**Plotting public debt using ggplot2**

Here, we’re going to scrape some data from Wikipedia and plot it using ggplot2.

It sounds easy enough, but make note of the complexity here. We’re actually using several dozen techniques in mildly complex ways to get this done.

If you’re a beginner, take a careful look at this. You need to break this down, identify the tools that are being used, and make sure that you know them. (Hint: 80% of this is ggplot2 and dplyr … make sure you master them.)

If you’re at an intermediate level and you now most of the R data science toolkit, take a look at how everything is being put together.

Don’t just run the code. Study it. You’ll become a better data scientist by studying other people’s code.

#=============

# LOAD PACKGES

#=============

library(tidyverse)

library(rvest)

library(stringr)

#=============

# READ WEBPAGE

#=============

html.public\_debt\_by\_country <- read\_html('https://en.wikipedia.org/wiki/List\_of\_countries\_by\_public\_debt')

#==================

# SCRAPE TABLE DATA

#==================

df.public\_debt\_by\_country <- html.public\_debt\_by\_country %>%

html\_nodes('table') %>%

.[[1]] %>%

html\_table()

#===============

# RENAME COLUMNS

#===============

names(df.public\_debt\_by\_country)

colnames(df.public\_debt\_by\_country) <- c('country'

,'debt\_as\_pct\_of\_gdp'

, 'measure\_year'

, 'gross\_debt\_as\_pct\_of\_gdp\_IMF'

, 'net\_debt\_as\_pct\_of\_gdp\_IMF'

, 'measure\_year\_IMF'

, 'region'

)

#========================================================

# DROP EXTRA COLUMNS

# - The Wikipedia table has a couple of different sources

# for very similar data.

# - We will drop the IMF data

#========================================================

df.public\_debt\_by\_country %>%

select(-gross\_debt\_as\_pct\_of\_gdp\_IMF

, -net\_debt\_as\_pct\_of\_gdp\_IMF

, -measure\_year\_IMF

) ->

df.public\_debt\_by\_country

#=====================================================

# REMOVE RECORD FOR 'World'

# - The original data has a summary record for 'World'

# - we will remove this

#=====================================================

df.public\_debt\_by\_country <- df.public\_debt\_by\_country %>% filter(country != 'World')

#============================

# COERCE data.frame TO tibble

#============================

df.public\_debt\_by\_country <- df.public\_debt\_by\_country %>% as\_tibble()

#=============

# GET MAP DATA

#=============

df.map <- map\_data('world')

# INSPECT map data

df.map %>% glimpse()

df.map %>% names()

# INSPECT debt data

df.public\_debt\_by\_country %>% glimpse()

#=============================

# RENAME 'region' to 'country'

#=============================

df.map %>%

rename(country = region) ->

df.map

df.map %>% glimpse()

#===============================

# IDENTIFY JOIN MISMATCHES

# - we will use an anti-join to

# identify mis-matches between

# the country variable on our

# two different datasets

#===============================

anti\_join(df.public\_debt\_by\_country

,df.map

,by = 'country'

)

# country debt\_as\_pct\_of\_gdp measure\_year

# 1 Antigua and Barbuda 89.0 2012

# 2 Burma NA NA

# 3 People's Republic of China 20.1 2016

# 4 Congo, Democratic Republic of the 18.2 2016

# 5 Congo, Republic of the 49.3 2016

# 6 Cote d'Ivoire 50.9 2016

# 7 Gambia, The NA NA

# 8 Gibraltar 7.5 2008

# 9 Hong Kong 38.4 2016

# 10 Korea, North NA NA

# 11 Korea, South 46.1 2016

# 12 Saint Kitts and Nevis 83.0 2013

# 13 Saint Vincent and the Grenadines 67.0 2013

# 14 Trinidad and Tobago 61.0 2016

# 15 Tuvalu 41.1 2013

# 16 United Kingdom 92.2 2016

# 17 United States 73.8 2016

# 18 World 64.0 2012

#============================================

# GET COUNTRY NAMES FROM df.map

# - these will be the new names that we will

# use when we re-code the names in

# df.public\_debt\_by\_country

#============================================

df.map %>%

group\_by(country) %>%

summarise() %>%

print(n = Inf)

# RECODE

df.public\_debt\_by\_country %>%

mutate(country = recode(country

,`Antigua and Barbuda` = 'Antigua'

,`Burma` = 'Myanmar'

,`People's Republic of China` = 'China'

,`Congo, Democratic Republic of the` = 'Democratic Republic of the Congo'

,`Congo, Republic of the` = 'Republic of Congo'

,`Cote d'Ivoire` = 'Ivory Coast'

,`Gambia, The` = 'Gambia'

#,`Gibraltar` = ''

#,`Hong Kong` = ''

,`Korea, North` = 'North Korea'

,`Korea, South` = 'South Korea'

,`Saint Kitts and Nevis` = 'Saint Kitts'

,`Saint Vincent and the Grenadines` = 'Saint Vincent'

,`Trinidad and Tobago` = 'Trinidad'

#,`Tuvalu` = ''

,`United Kingdom` = 'UK'

,`United States` = 'USA'

)

) ->

df.public\_debt\_by\_country

#===========================

# RE-INSPECT JOIN MISMATCHES

# note: these last 3 are OK

#===========================

anti\_join(df.public\_debt\_by\_country

,df.map

,by = 'country'

)

#=====

# JOIN

#=====

df.map\_public\_debt <- left\_join(df.map

,df.public\_debt\_by\_country

,by = 'country'

)

# INSPECT

df.map\_public\_debt %>% glimpse()

#============================

# PLOT

# - this is just a basic plot

#============================

df.map\_public\_debt %>%

ggplot(aes(x = long, y = lat, group = group)) +

geom\_polygon(aes(fill = debt\_as\_pct\_of\_gdp))

#=============

# CREATE THEME

#=============

theme.map <- theme(

text = element\_text(family = 'Helvetica Neue', color = '#444444')

,panel.background = element\_rect(fill = '#CCCCCC')

,plot.background = element\_rect(fill = '#CCCCCC')

,legend.background = element\_rect(fill = '#CCCCCC')

,panel.grid = element\_blank()

,plot.title = element\_text(size = 18, face = 'bold')

,plot.subtitle = element\_text(size = 12)

,legend.key = element\_blank()

,axis.text = element\_blank()

,axis.ticks = element\_blank()

,axis.title = element\_blank()

)

#========================

# CREATE WORLD PLOT / MAP

#========================

plot.debt\_to\_gdp\_map <- df.map\_public\_debt %>%

ggplot(aes(x = long, y = lat, group = group)) +

geom\_polygon(aes(fill = debt\_as\_pct\_of\_gdp)) +

theme.map +

labs(title = str\_c('Countries in the developed world are'

,'\ncarrying high levels of debt compared to GDP'

)

,fill = str\_c('Net public debt','\nas a % of GDP')

) +

scale\_fill\_gradientn(colors = c('#009933', '#ffff00', 'orange', '#e60000')

,values = scales::rescale(c(30, 50, 70, 100, 200))

)

Here, we’ll actually plot this two different times.

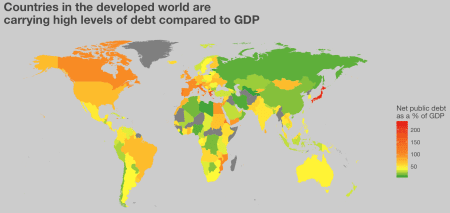
First, we’ll plot the entire map of the world.

#------

# WORLD

#------

plot.debt\_to\_gdp\_map

[[](http://www.sharpsightlabs.com/join-newsletter-get-crash-course/)](http://www.sharpsightlabs.com/join-newsletter-get-crash-course/)

Next, we’ll sub-set this down and look only at a specific region: Europe.

To do this, we can use coord\_cartesian() to specify a specific range for lat/long. This essentially “zooms in” on a specific part of the map.

#-------

# EUROPE

#-------

plot.debt\_to\_gdp\_map +

coord\_cartesian(xlim = c(-15, 50), ylim = c(30, 75)) +

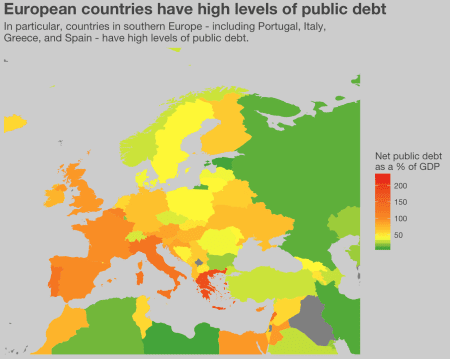
labs(title = "European countries have high levels of public debt"

,subtitle = str\_c('In particular, countries in southern Europe - including Portugal, Italy,'

,'\nGreece, and Spain - have high levels of public debt.'

)

)

[[](http://www.sharpsightlabs.com/join-newsletter-get-crash-course/)](http://www.sharpsightlabs.com/join-newsletter-get-crash-course/)