Lab 1 Solutions

Oscar Cuadros

September 30, 2022

Goals for today

For the first lab, we will reverse engineer some plots using a relatively small, historical dataset. This should help you practice the "grammar of graphics'' as well as the process of downloading, completing, and knitting your assignments. This lab is based on the structure of the first assignment.

Load and look at the data

METHOD 1

- STEP 0: Installing packages in R (tidyverse, haven, readxl, rmarkdown)
- STEP 1: Call the libraries you need

```
library(tidyverse) # csv, tsv, fwf
library(haven) # sas(SAS), sav(SPSS), dta(Stata)
library(readxl) # xlsx, xls(Excel)
```

- STEP 2: Set your file path

```
setwd("C:/Users/cuadr/OneDrive/Escritorio/UCHICAGO/7. Intro QMSC 2022/Lab/2022/1")

# To reset your path:
# require("knitr")
# opts_knit$set(root.dir = "C:/Users/cuadr/OneDrive/Escritorio/UCHICAGO/7. Intro QMSC 2022/Lab/2022/1")
```

- STEP 3: Upload your data set in R

```
midwest <- read_dta("midwest.dta")

male_female <- read.delim("male_female_counts.txt")

college <- read.csv("College.csv")

world_wealth_inequality <- read_excel("world_wealth_inequality.xlsx")
```

- Now check your data!

```
View(midwest)
View(male_female)
View(college)
View(world_wealth_inequality)
```

METHOD 2

```
file <- "https://raw.githubusercontent.com/UChicago-pol-methods/IntroQSS-F22/main/data/carsonPDB.csv"
df_pdb <- read_csv(file)
```

```
## Rows: 4998 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): date
## dbl (5): PDBid, President, Total_Pgs, maps, Redaction_total
##
## 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.
```

```
df_pdb
```

```
## # A tibble: 4,998 x 6
##       PDBid date    President Total_Pgs  maps Redaction_total
##       <dbl> <chr>       <dbl>     <dbl> <dbl>           <dbl>
##  1 17061961 6/17/61         1         8     2              26
##  2 19061961 6/19/61         1         7     1              23
##  3 20061961 6/20/61         1         5     0               8
##  4 21061961 6/21/61         1         5     0              17
##  5 22061961 6/22/61         1         6     1              13
##  6 23061961 6/23/61         1         9     1              22
##  7 24061961 6/24/61         1         6     0               9
##  8 26061961 6/26/61         1         7     1              22
##  9 27061961 6/27/61         1         7     1              25
## 10 28061961 6/28/61         1         8     1              24
## # ... with 4,988 more rows
```

```
## # A tibble: 209 x 8
##    Country         Year Proportion_Consumptio~ Proportion_Cons~ Proportion_Impo~
##    <chr>          <dbl>                  <dbl>            <dbl>            <dbl>
##  1 United Kingdom  1893                   0.02             100.               NA
##  2 United Kingdom  1894                   0                100                NA
##  3 United Kingdom  1895                   0.01             100.               NA
##  4 United Kingdom  1896                   0.01             100.               NA
##  5 United Kingdom  1897                   0.01             100.               NA
##  6 United Kingdom  1898                   0.01             100.               NA
##  7 United Kingdom  1899                   0                100                NA
##  8 United Kingdom  1900                   0.01             100.               NA
##  9 United Kingdom  1901                   0                100                NA
## 10 United Kingdom  1902                   0                100                NA
## # ... with 199 more rows, and 3 more variables: Tons_Imported <dbl>,
## #   Tons_Produced <dbl>, Tons_Exported <dbl>
```

Don't worry if you cannot interpret the code above. If you can, excellent! All you need to know is that our data is now in the environment and saved as an object called `df` (for "dataframe'').

Before we worry about visualizing the data, let's look at what it contains. Run the chunk below to view the column headings and first several rows of data.

```
head(df)  #show col names and several rows of data
```

```
## # A tibble: 6 x 8
##   Country  Year Proportion_Cons~ Proportion_Cons~ Proportion_Impo~ Tons_Imported
##   <chr>   <dbl>            <dbl>            <dbl>            <dbl>         <dbl>
## 1 United~  1893             0.02             100.               NA         25000
## 2 United~  1894             0                100                NA          7000
## 3 United~  1895             0.01             100.               NA         14000
## 4 United~  1896             0.01             100.               NA         16000
## 5 United~  1897             0.01             100.               NA          8000
## 6 United~  1898             0.01             100.               NA         10000
## # ... with 2 more variables: Tons_Produced <dbl>, Tons_Exported <dbl>
```

What are the variables contained in the data set?

Country, year, proportion of consumed coal that was imported, proportion of the consumption that was imported from the UK, proportion of the imported that came from the UK, amount imported, amount produced, and amount exported.

What does each row represent? (i.e. what is the *unit of observation*?)

Each row is unique to a given country in a specific year. We would say the unit of observation is a "country-year.''

Notice that the first several rows of "Proportion_Imported_From_UK'' are empty. Why might this data be missing? (hint: look at column 1)

By definition, the UK cannot import coal from the UK. Therefore, it is nonsensical to think about this variable taking on any value for these observations.

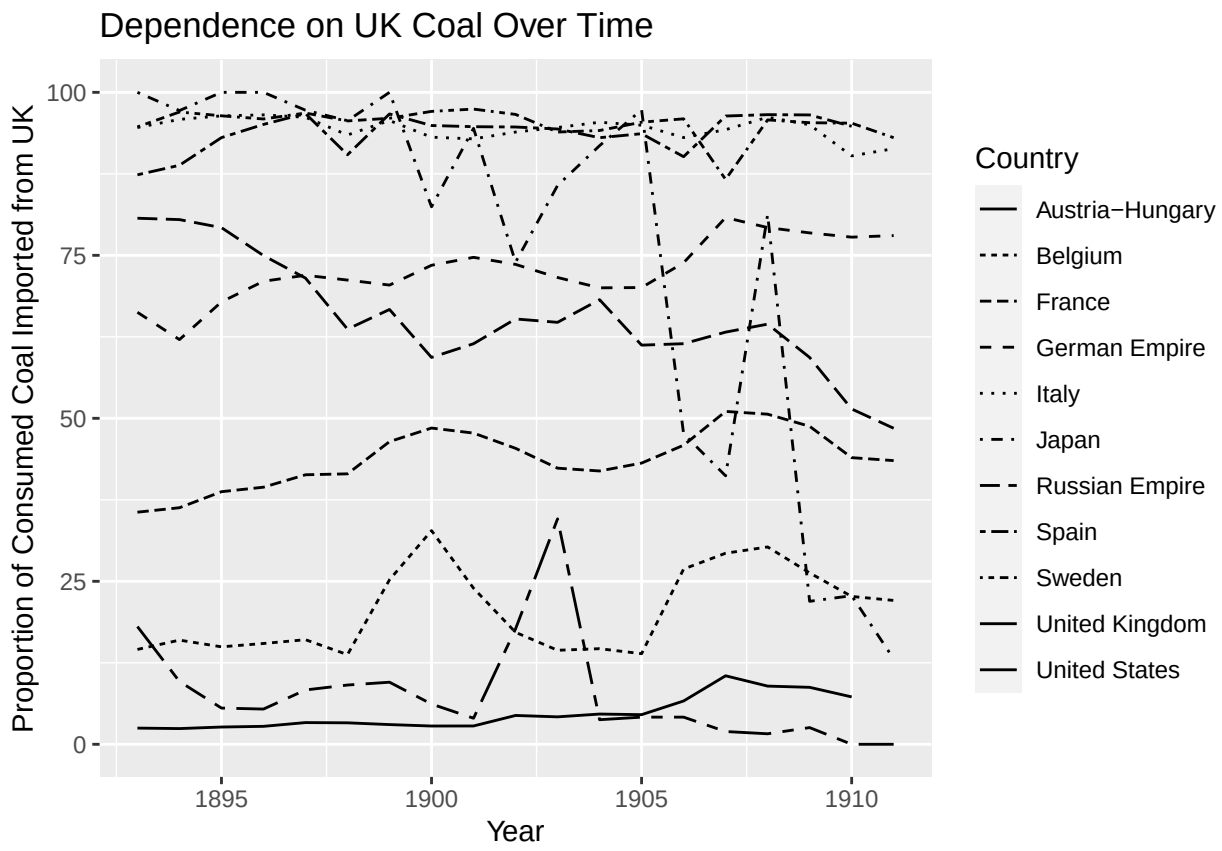
First plot

For the first plot, most of the code has been filled in for you. Alter the code to recreate this image.

```
plot_1 <- ggplot(data = df,
                 #create plot, specify data and global aes
                 mapping = aes(x = Year,
                               y = Proportion_Imported_From_UK,
                               linetype = Country)) +
  geom_line() + #generate line graph
  labs(x = "Year",
       y = "Proportion of Consumed Coal Imported from UK",
       title = "Dependence on UK Coal Over Time") #title and labels

plot_1
```

```
## Warning: Removed 21 row(s) containing missing values (geom_path).
```



Second plot

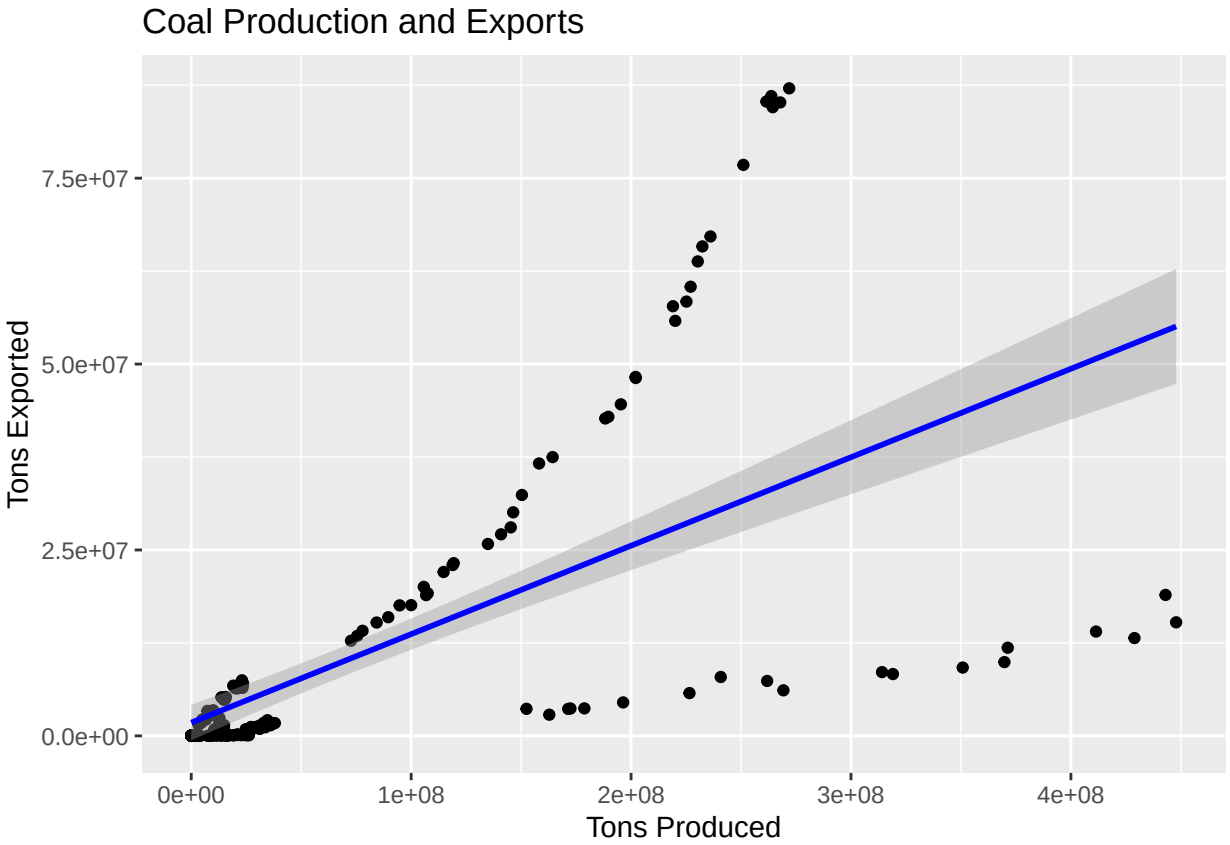
For the second plot, more of the code is missing. See if you can recreate the image. (hint, the line being fitted is a linear model)

```
plot_2 <- ggplot(data = df,
                 #create plot, specify data and global aes
                 mapping = aes(x = Tons_Produced,
                               y = Tons_Exported)) +
  geom_point() + #generate scatterplot
  geom_smooth(method = "lm",
              color = "blue") + #add lm
  labs(x = "Tons Produced",
       y = "Tons Exported",
       title = "Coal Production and Exports") #labels

plot_2
```

```
## Warning: Removed 2 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



What relationship does this plot represent?

It shows us what happens to coal exports as coal production increases.

Does the overall trend (as represented by the line) fit your expectations?

It should fit your expectations. In general, exports increase as production increases. However, the relationship is not 1:1, suggesting that countries don't export all of increased consumption.

Why do you think there is a cluster of points in the bottom right corner?

These observations probably come from a highly industrialized country which has large coal reserves: they produce a lot, but export less than expected, suggesting they consume much of it domestically.

What about the points in the lower left?

These are likely observations from countries who have few coal reserves. They are able to produce and export very little, relative to some other countries.

Third Plot

Since we suspect that the clustering is the result of different countries exporting different proportions of the coal they produce, we can alter the code above slightly to color the points by country and fit a separate line for each country. Complete the code to produce this plot.

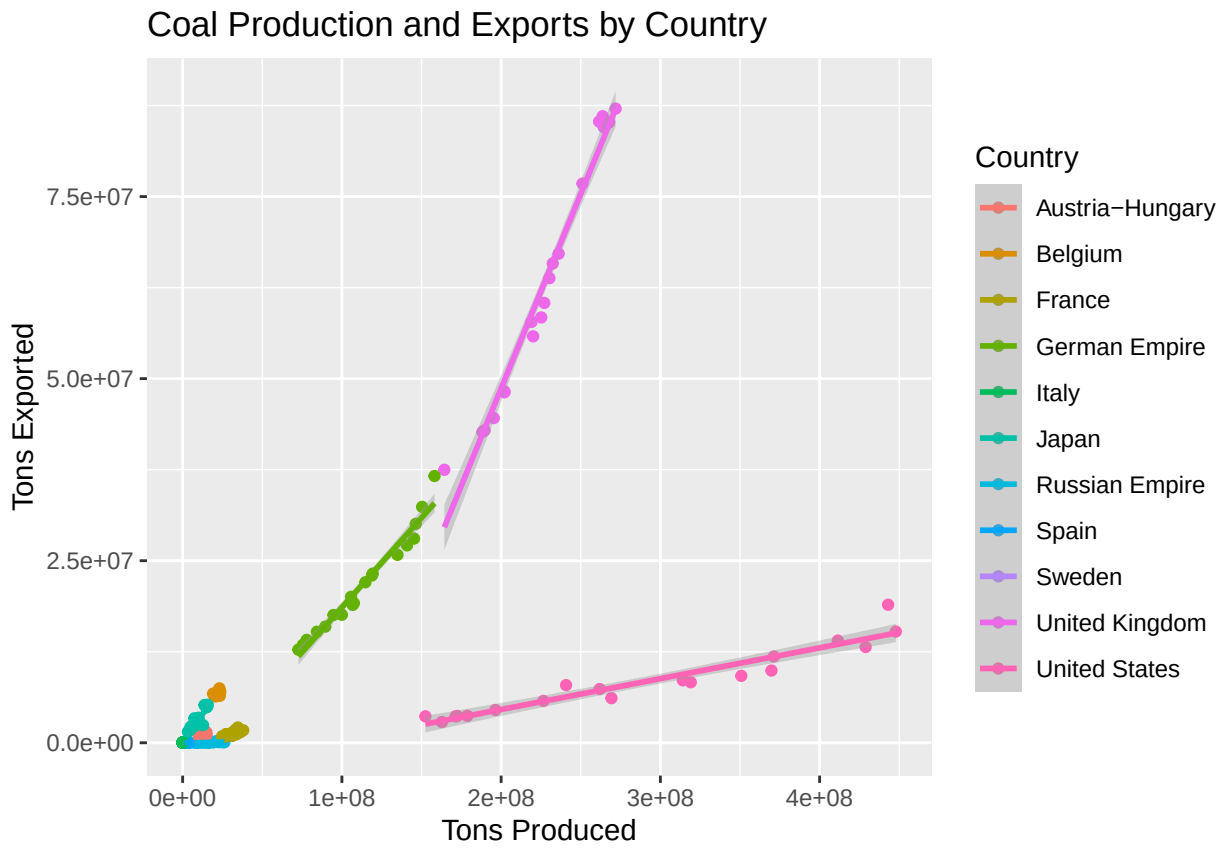
```
plot_3 <- ggplot(data = df,
                 #create plot, specify data and global aes
                 mapping = aes(x = Tons_Produced,
                               y = Tons_Exported,
```

```
                                color = Country)) +
  geom_point() + #scatterplot layer
  geom_smooth(method = "lm") + #lm layer
  labs(x = "Tons Produced",
       y = "Tons Exported",
       title = "Coal Production and Exports by Country") #labels

plot_3
```

```
## Warning: Removed 2 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



What issues do you see with this plot?

Some countries produce and export **A LOT** more than some of the others. This makes it difficult to really see what is happening with the countries who are clustered in the bottom left.

How might we address this concern with a different visualization?

One way we can solve this is by *faceting*. Faceting breaks up a single plot into a series of plots by the value of a categorical variable (in this case, "Country' ').

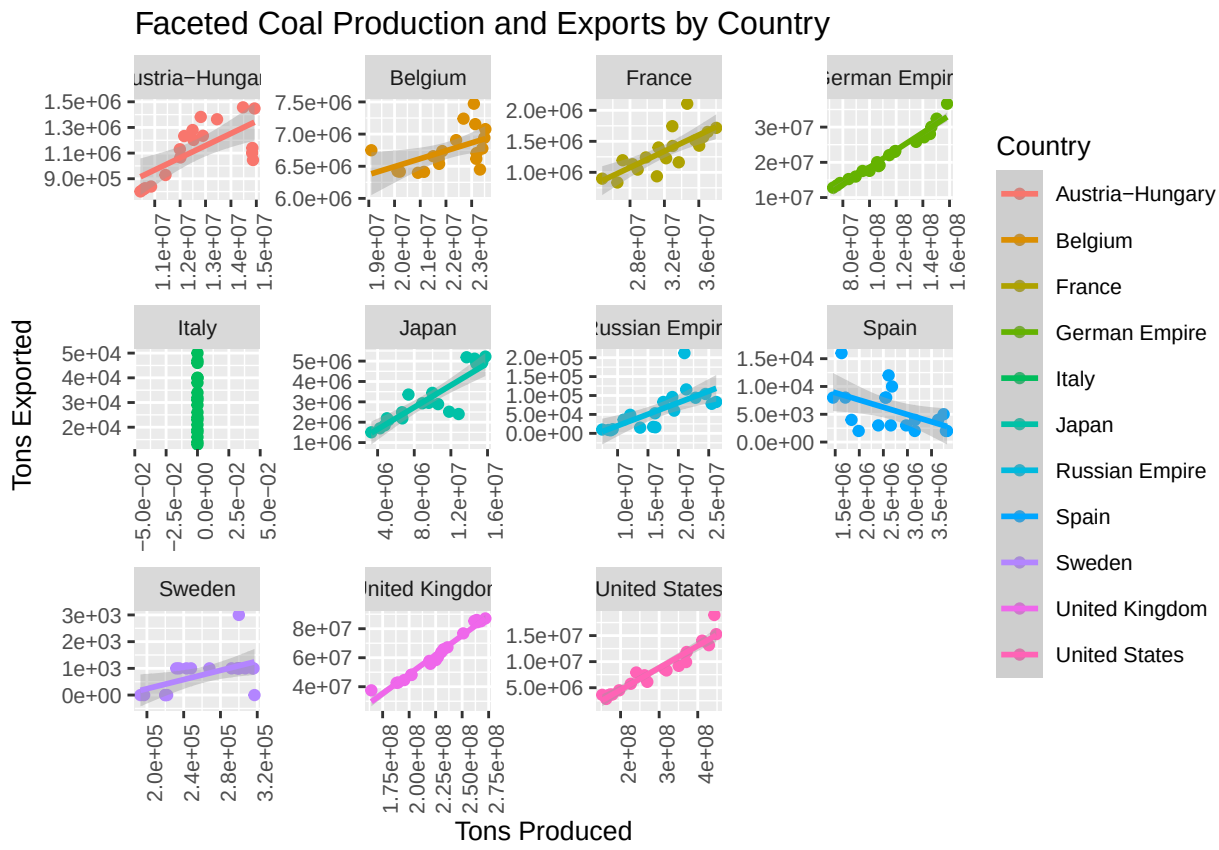
Fourth Plot

```
plot_4 <- ggplot(data = df, #create plot, specify data and global aes
                 mapping = aes(x = Tons_Produced,
                               y = Tons_Exported,
                               color = Country)) +
  geom_point() + #scatterplot layer
  geom_smooth(method = "lm") + #lm layer
  #facet by country, allow variable scales
  facet_wrap(vars(Country), scales = "free") +
  #resize and rotate x axis labels 45 degrees
  theme(text = element_text(size=10),
        axis.text.x = element_text(angle=90, hjust=1)) +
  #force scientific notation on axis labels
  scale_y_continuous(labels = scientific) +
  scale_x_continuous(labels = scientific) +
  labs(x = "Tons Produced",
       y = "Tons Exported",
       title = "Faceted Coal Production and Exports by Country") #labels

plot_4
```

```
## Warning: Removed 2 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



While colors can be a powerful visualization tool, line type, point shape, and other tools make your work more accessible to people who are colorblind. Alternatively, there are packages outside the tidyverse (e.g. `ggthemes`*) which contain colorblind friendly palettes or allow you to manually define a palette using several different color reference systems.*