DATA 607 - Project 2

Adam Douglas

Due 10/7/2018

Introduction

The concept of tidy data was popularized by Hadley Wickham in his paper titled "Tidy Data"[1]. By putting data into a tidy format, one could perform data analysis within R much quicker and easier, because the format works very well with R's vectorized functions.

This project will highlight these concepts by taking 3 examples of "untidy" data and, using tools that Wickham himself created, transform them into a tidy format. Then, we can see how much easier analysis can be done thanks to tidy data.

Set-Up

First, we will load the packages we require to do our tidying work. The two tools we will use most for this task are `tidyr` and `dplyr`. These are both included, along with some other useful tools for visualization and analysis, in a larger package known as `tidyverse`.

```
library(tidyverse)
```

Example 1

Our first example comes from the United Nations (UN) Department of Economic and Social Affairs[2]. The UN tracks, amongst many other things, migration of peoples from one area of the globe to another. The data, freely available on the UN website, was highlighted by my classmate Juanelle Marks.

We will be looking specifically at the data titled "By destination and origin" which shows migration to and from various countries across several years.

Raw Data

The raw data exists as an Excel file. The only transformation made to the original file was to remove extra tabs that we're not loading (for a smaller file size) and to filter out subtotal rows by color.

Since this exists as Excel, we will use the `readxl` package from the Tidyverse to assist in the loading of the data.

```
library(readxl)
```

Unfortunately, readxl does not yet work with URLs, so we load the data from a local file. Using the `read_excel` function, this is quite easy to do.

```
# Load the proper sheet and only the cell range we want
UN <- read_excel("UN_MigrantStockByOriginAndDestination_2017.xlsx",
          sheet="Table 1",
          range="A16:IG1906")
```

[1] http://vita.had.co.nz/papers/tidy-data.pdf
[2] http://www.un.org/en/development/desa/population/migration/data/estimates2/estimates17.shtml

```r
# Display the data
UN
```

```
## # A tibble: 1,890 x 241
##     X__1    X__2 X__3  X__4   X__5 X__6  Total `Other North` `Other South`
##    <dbl>   <dbl> <chr> <chr> <dbl> <chr> <chr> <chr>         <chr>
##  1  1990 1990014 Buru~ <NA>    108 B R   3331~ 8943          50676
##  2  1990 1990015 Como~ <NA>    174 B     14079 672           847
##  3  1990 1990016 Djib~ <NA>    262 B R   1222~ 1827          5484
##  4  1990 1990017 Erit~ <NA>    232 I     11848 345           737
##  5  1990 1990018 Ethi~ <NA>    231 B R   1155~ 7358          22075
##  6  1990 1990019 Kenya <NA>    404 B R   2972~ 35132         65430
##  7  1990 1990020 Mada~ <NA>    450 C     23917 3563          2851
##  8  1990 1990021 Mala~ <NA>    454 B R   1127~ 11744         19158
##  9  1990 1990022 Maur~ 1       480 C     3613  75            292
## 10  1990 1990023 Mayo~ <NA>    175 B     15229 1142          1354
## # ... with 1,880 more rows, and 232 more variables: Afghanistan <chr>,
## #   Albania <chr>, Algeria <chr>, `American Samoa` <chr>, Andorra <chr>,
## #   Angola <chr>, Anguilla <chr>, `Antigua and Barbuda` <chr>,
## #   Argentina <chr>, Armenia <chr>, Aruba <chr>, Australia <chr>,
## #   Austria <chr>, Azerbaijan <chr>, Bahamas <chr>, Bahrain <chr>,
## #   Bangladesh <chr>, Barbados <chr>, Belarus <chr>, Belgium <chr>,
## #   Belize <chr>, Benin <chr>, Bermuda <chr>, Bhutan <chr>, `Bolivia
## #   (Plurinational State of)` <chr>, `Bosnia and Herzegovina` <chr>,
## #   Botswana <chr>, Brazil <chr>, `British Virgin Islands` <chr>, `Brunei
## #   Darussalam` <chr>, Bulgaria <chr>, `Burkina Faso` <chr>,
## #   Burundi <chr>, `Cabo Verde` <chr>, Cambodia <chr>, Cameroon <chr>,
## #   Canada <chr>, `Caribbean Netherlands` <chr>, `Cayman Islands` <chr>,
## #   `Central African Republic` <chr>, Chad <chr>, `Channel Islands` <chr>,
## #   Chile <chr>, China <chr>, `China, Hong Kong SAR` <chr>, `China, Macao
## #   SAR` <chr>, Colombia <chr>, Comoros <chr>, Congo <chr>, `Cook
## #   Islands` <chr>, `Costa Rica` <chr>, `Côte d'Ivoire` <chr>,
## #   Croatia <chr>, Cuba <chr>, Curaçao <chr>, Cyprus <chr>, Czechia <chr>,
## #   `Dem. People's Republic of Korea` <chr>, `Democratic Republic of the
## #   Congo` <chr>, Denmark <chr>, Djibouti <chr>, Dominica <chr>,
## #   `Dominican Republic` <chr>, Ecuador <chr>, Egypt <chr>, `El
## #   Salvador` <chr>, `Equatorial Guinea` <chr>, Eritrea <chr>,
## #   Estonia <chr>, Ethiopia <chr>, `Faeroe Islands` <chr>, `Falkland
## #   Islands (Malvinas)` <chr>, Fiji <chr>, Finland <chr>, France <chr>,
## #   `French Guiana` <chr>, `French Polynesia` <chr>, Gabon <chr>,
## #   Gambia <chr>, Georgia <chr>, Germany <chr>, Ghana <chr>,
## #   Gibraltar <chr>, Greece <chr>, Greenland <chr>, Grenada <chr>,
## #   Guadeloupe <chr>, Guam <chr>, Guatemala <chr>, Guinea <chr>,
## #   `Guinea-Bissau` <chr>, Guyana <chr>, Haiti <chr>, `Holy See` <chr>,
## #   Honduras <chr>, Hungary <chr>, Iceland <chr>, India <chr>,
## #   Indonesia <chr>, `Iran (Islamic Republic of)` <chr>, ...
```

Looking at the raw data, we first see some missing column names, which we can fix easily:

```r
# Using the Excel document, fix the first 6 column names
UN <- UN %>% rename(year = X__1,
             sort = X__2,
             destination = X__3,
             notes = X__4,
             code = X__5,
```

```
              typeOfData = X__6)
```

Tidying

The next piece we tackle is the fact that we have observations (source countries) saved as individual variables. We can use the `gather` function from `tidyr` to fix this:

```r
# This creates a new column called "source" from the column headers
#   and puts the values into a "people" column, effectively gathering
#   the data into a longer (rather than wider) format

tidyUN <- UN %>% gather(key = "source", value="people", 7:241)

# Select a subset to see how the data looks now
tidyUN %>% filter(destination == "France", year == 2017)
```

```
## # A tibble: 235 x 8
##     year    sort destination notes  code typeOfData source         people
##    <dbl>   <dbl> <chr>       <chr> <dbl> <chr>      <chr>          <chr>
##  1  2017 2017178 France      <NA>    250 B          Total          7902783
##  2  2017 2017178 France      <NA>    250 B          Other North    ..
##  3  2017 2017178 France      <NA>    250 B          Other South    ..
##  4  2017 2017178 France      <NA>    250 B          Afghanistan    4832
##  5  2017 2017178 France      <NA>    250 B          Albania        6796
##  6  2017 2017178 France      <NA>    250 B          Algeria        1452409
##  7  2017 2017178 France      <NA>    250 B          American Samoa 2
##  8  2017 2017178 France      <NA>    250 B          Andorra        996
##  9  2017 2017178 France      <NA>    250 B          Angola         21610
## 10  2017 2017178 France      <NA>    250 B          Anguilla       9
## # ... with 225 more rows
```

Finally, we notice that the people column is stored as a character vector, not as numeric which would be more appropriate for this sort of variable. Before we can fix that, we have to get rid of the `..` used in the Excel sheet for missing data and replace it with an NA.

```r
# Replace the ..'s with NA
tidyUN$people[tidyUN$people == ".."] <- NA

# Change the column type
tidyUN$people <- as.integer(tidyUN$people)
```

Our data should now be in a tidy format and ready for analysis.

Analysis

There are many different types of analyses we could perform on this data, and that only increases if we were to include other data from the UN site. For this demonstration, we'll keep to some simple examples.

First, let's take a random country like France and see where the top 10 migrant populations come from:

```r
france <- tidyUN %>% filter(year == 2017, destination == "France",
                  source!= "Total") %>%
  group_by(as.character(year), source) %>%
  summarize(n = sum(people, na.rm=TRUE)) %>%
  top_n(10,n) %>%
```

Table 1: Migrant Population in France by Country of Origin (Top 10)

Year	Country	# People
2017	Algeria	1,452,409
2017	Morocco	940,552
2017	Portugal	724,000
2017	Tunisia	394,506
2017	Italy	373,182
2017	Spain	309,049
2017	Turkey	301,950
2017	Germany	237,178
2017	United Kingdom	188,161
2017	Belgium	155,548

Table 2: Migrant Population by Country (Top 10)

Year	Country	# People
2017	United States of America	49,776,970
2017	Saudi Arabia	12,185,284
2017	Germany	12,165,083
2017	Russian Federation	11,651,509
2017	United Kingdom	8,841,717
2017	United Arab Emirates	8,312,524
2017	France	7,902,783
2017	Canada	7,861,226
2017	Australia	7,035,560
2017	Spain	5,947,106

```
  arrange(desc(n))

kable(france, col.names = c("Year","Country","# People"),
      caption="Migrant Population in France by Country of Origin (Top 10)",
      format.args = list(big.mark = ",")) %>%
  kable_styling(bootstrap_options = "striped", full_width = FALSE, position = "left")
```

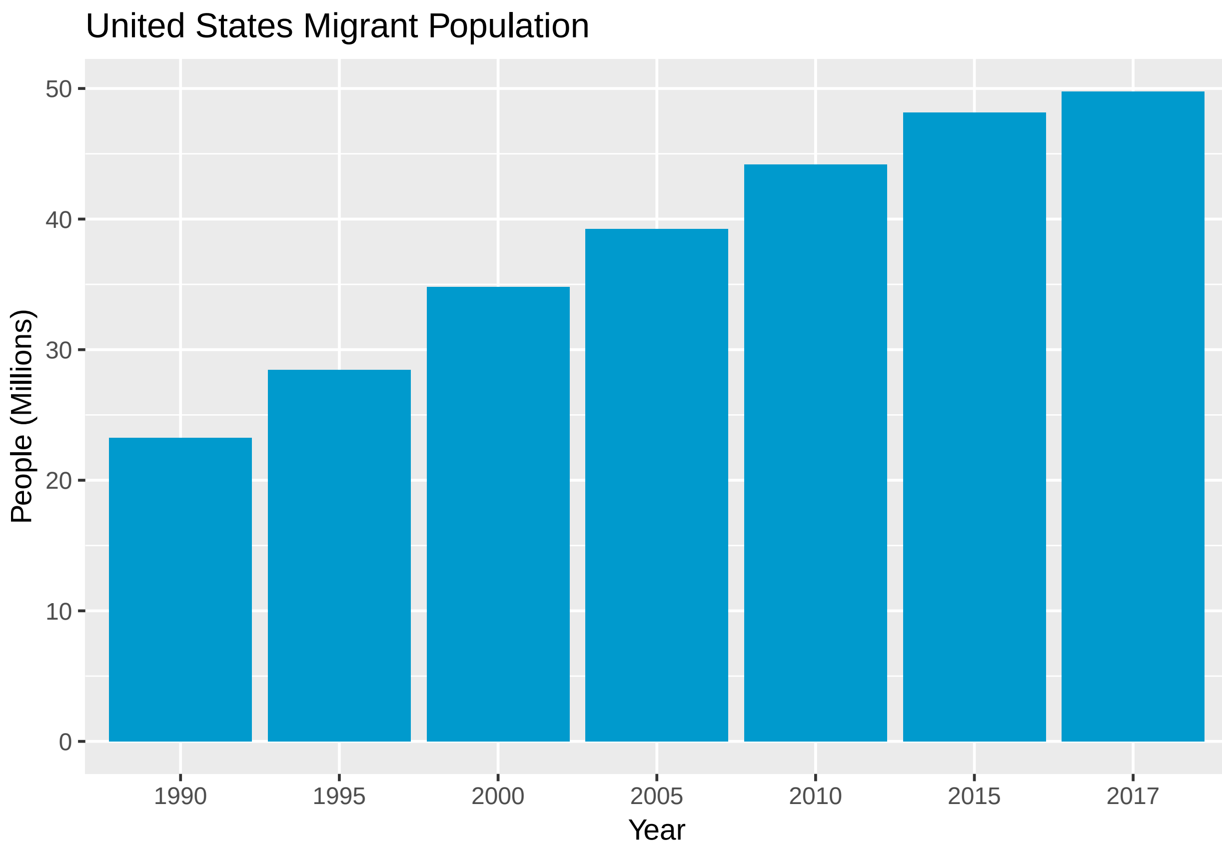
We can also look at which country holds the largest migrant population:

```
top5Dest <- tidyUN %>% filter(year == 2017, source == "Total") %>%
  group_by(as.character(year), destination) %>%
  summarize(n = sum(people, na.rm=TRUE)) %>%
  top_n(10,n) %>%
  arrange(desc(n))

kable(top5Dest, col.names = c("Year","Country","# People"),
      caption="Migrant Population by Country (Top 10)",
      format.args = list(big.mark = ",")) %>%
  kable_styling(bootstrap_options = "striped", full_width = FALSE, position = "left")
```

We can also track over time

```
US <- tidyUN %>% filter(destination == "United States of America",
                  source== "Total") %>%
```

```r
  group_by(as.character(year)) %>%
  rename(Year = `as.character(year)`) %>%
  summarize(millionPeople = sum(people, na.rm=TRUE)/1000000)

US %>% ggplot(aes(x=Year, y=millionPeople)) +
  geom_col(fill="deepskyblue3") + ylab("People (Millions)") +
  ggtitle("United States Migrant Population")
```



Example 2

The next example is taken from the Department of Education's National Student Loan Data System (NSLDS)[3]. The specific file is a portfolio summary showing outstanding interest and balances by acedemic year and loan type.

Raw Data

The raw data is in a table inside an Excel document. So, we will use the `readxl` package like before.

```r
loan <- read_excel("PortfolioSummary.xls")

loan
```

[3]https://catalog.data.gov/dataset/national-student-loan-data-system/resource/02a63933-37ef-4b14-a45a-90dd7b523b29

```
## # A tibble: 39 x 10
##    `Federal Student~ X__1  X__2  X__3  X__4  X__5  X__6  X__7  X__8  X__9 
##    <chr>             <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
##  1 Includes outstan~ <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA> 
##  2 Data Source: Nat~ <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA> 
##  3 <NA>              <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA> 
##  4 <NA>              <NA>  Dire~ <NA>  Fede~ <NA>  Perk~ <NA>  Tota~ <NA> 
##  5 Federal Fiscal Y~ <NA>  Doll~ Reci~ Doll~ Reci~ Doll~ Reci~ Doll~ Undu~
##  6 2007              <NA>  106.8 7     401.~ 22.6~ 8.19~ 2.79~ 516   28.3~
##  7 2008              <NA>  122.5 7.70~ 446.5 23.6~ 8.5   2.89~ 577   29.8~
##  8 2009              <NA>  154.~ 9.19~ 493.~ 25    8.69~ 3     657   32.1~
##  9 2010              <NA>  224.5 14.4  516.~ 25.1~ 8.40~ 2.89~ 749.~ 34.2~
## 10 2011              <NA>  350.~ 19.3~ 489.~ 23.8~ 8.30~ 2.89~ 848.~ 36.5 
## # ... with 29 more rows
```

Looking at the data, we have our work cut out for us. First we'll do some basic cleanup:

```r
# The first 3 lines are sheet headers and unnecessary
loan <- loan[-(1:3),]

# Fix our variable names
names(loan) <- c("year","period","Direct Loan Dollars Outstanding",
                 "Direct Loan Recipients","FFEL Dollars Outstanding",
                 "FFEL Loan Recipients","Perkins Loan Dollars Outstanding",
                 "Perkins Loan Recipients","Total Dollars Outstanding",
                 "Total Recipients")

# The next 2 lines are column headers and redundant
loan <- loan[-(1:2),]

# If we look at the end of the data, there are more unnecessary rows
#   left over from the Excel file.
loan <- head(loan,-5)

loan
```

```
## # A tibble: 29 x 10
##    year  period `Direct Loan Dollar~ `Direct Loan Reci~ `FFEL Dollars Out~
##    <chr> <chr>  <chr>                <chr>              <chr>             
##  1 2007  <NA>   106.8                7                  401.89999999999998
##  2 2008  <NA>   122.5                7.7000000000000002 446.5             
##  3 2009  <NA>   154.90000000000001   9.1999999999999993 493.30000000000001
##  4 2010  <NA>   224.5                14.4               516.70000000000005
##  5 2011  <NA>   350.10000000000002   19.399999999999999 489.80000000000001
##  6 2012  <NA>   488.30000000000001   22.800000000000001 451.69999999999999
##  7 2013  Q1     508.69999999999999   23.399999999999999 444.89999999999998
##  8 <NA>  Q2     553                  24.100000000000001 437               
##  9 <NA>  Q3     569.20000000000005   24.300000000000001 429.5             
## 10 <NA>  Q4     609.10000000000002   25.600000000000001 423               
## # ... with 19 more rows, and 5 more variables: `FFEL Loan
## #   Recipients` <chr>, `Perkins Loan Dollars Outstanding` <chr>, `Perkins
## #   Loan Recipients` <chr>, `Total Dollars Outstanding` <chr>, `Total
## #   Recipients` <chr>
```

Now our data frame looks more complete now.

Tidying

We have a few issues outstanding with the data before we can call it tidy. First, we are missing some data in the `year` and `period` columns.

```
# Fill in years
loan <- fill(loan,year)

# Missing periods are for the whole year
loan[(is.na(loan$period)),2] <- "YR"

loan
```

```
## # A tibble: 29 x 10
##    year  period `Direct Loan Dollar~ `Direct Loan Reci~ `FFEL Dollars Out~
##    <chr> <chr>  <chr>                <chr>              <chr>
##  1 2007  YR     106.8                7                  401.89999999999998
##  2 2008  YR     122.5                7.7000000000000002 446.5
##  3 2009  YR     154.90000000000001   9.1999999999999993 493.30000000000001
##  4 2010  YR     224.5                14.4               516.70000000000005
##  5 2011  YR     350.10000000000002   19.399999999999999 489.80000000000001
##  6 2012  YR     488.30000000000001   22.800000000000001 451.69999999999999
##  7 2013  Q1     508.69999999999999   23.399999999999999 444.89999999999998
##  8 2013  Q2     553                  24.100000000000001 437
##  9 2013  Q3     569.20000000000005   24.300000000000001 429.5
## 10 2013  Q4     609.10000000000002   25.600000000000001 423
## # ... with 19 more rows, and 5 more variables: `FFEL Loan
## #   Recipients` <chr>, `Perkins Loan Dollars Outstanding` <chr>, `Perkins
## #   Loan Recipients` <chr>, `Total Dollars Outstanding` <chr>, `Total
## #   Recipients` <chr>
```

Finally, we see that observations (type of loan) is put into columns as if they were variables. We can easily fix this with `gather` from the `dplyr` package.

```
# Because we have multiple variables in columns, we need to be
#   more careful in our use of "gather"

# Move all into a single column
tidyLoan <- gather(loan, key="type", value="amount", -year, -period)

# Get the word "loan" out so we normalize the names
tidyLoan$type <- str_replace(tidyLoan$type,"\\s{1}(Loan)","")

# Now split the "type" column, because it really contains two variable types
tidyLoan <-
  tidyLoan %>%
  extract(type, c("loanType", "measure"),"([[:alpha:]]+)\\s{1}(.+)")

# Then spread those into their respective columns
tidyLoan <- tidyLoan %>% spread(measure, amount)

# Finally, fix column names and column types
tidyLoan <- rename(tidyLoan, dollars = `Dollars Outstanding`, recipients = Recipients)

tidyLoan$dollars <- as.numeric(tidyLoan$dollars)
```

```
tidyLoan$recipients <- as.numeric(tidyLoan$recipients)

tidyLoan
```

```
## # A tibble: 116 x 5
##    year  period loanType dollars recipients
##    <chr> <chr>  <chr>      <dbl>      <dbl>
##  1 2007  YR     Direct     107.         7
##  2 2007  YR     FFEL       402.        22.6
##  3 2007  YR     Perkins      8.2        2.8
##  4 2007  YR     Total      516         28.3
##  5 2008  YR     Direct     122.         7.7
##  6 2008  YR     FFEL       446.        23.7
##  7 2008  YR     Perkins      8.5        2.9
##  8 2008  YR     Total      577         29.9
##  9 2009  YR     Direct     155.         9.2
## 10 2009  YR     FFEL       493.        25
## # ... with 106 more rows
```

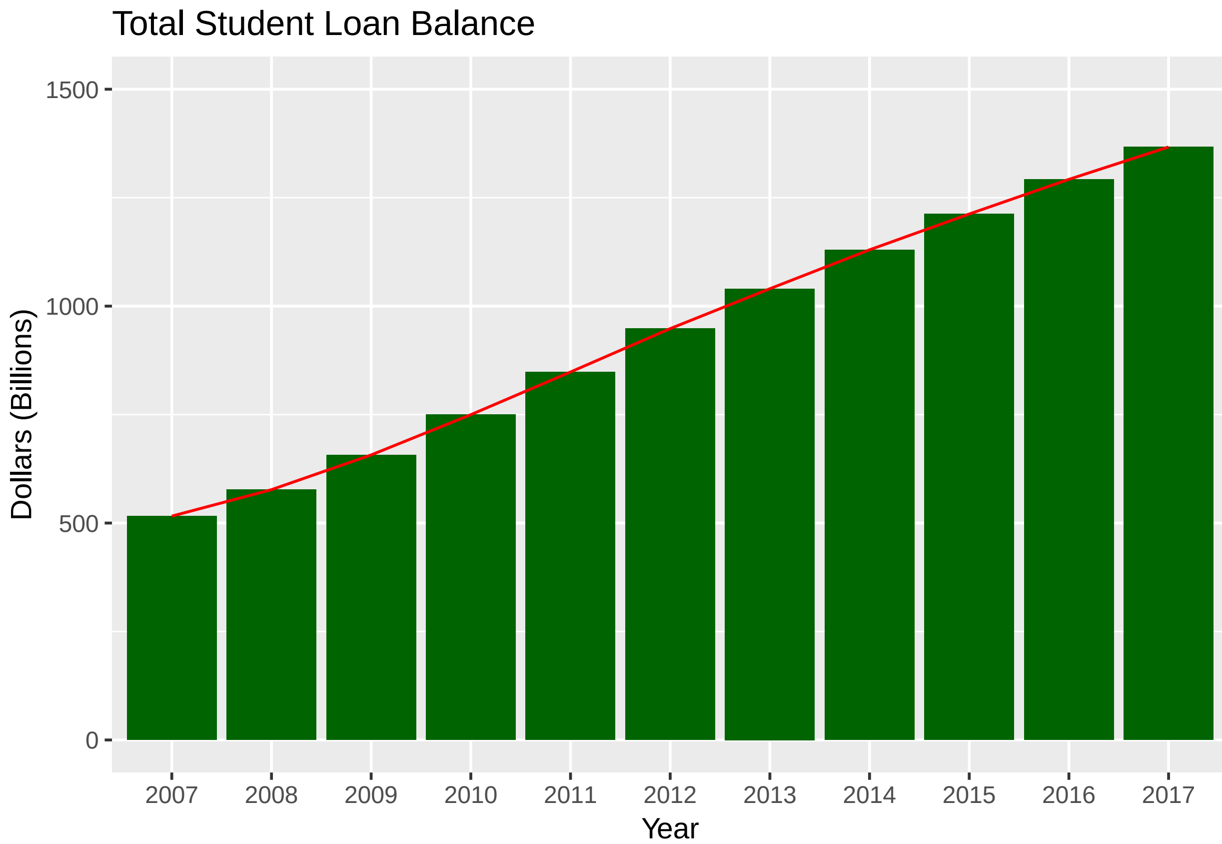
We now have our data in a tidy format with 116 observations (the original 29 times 4 - 3 loan types and 1 total) and can begin analyzing.

Analysis

Alone, this data does not afford too much in the way of analysis, however we can look at a few items of interest.

First, over time, the total loan amounts:

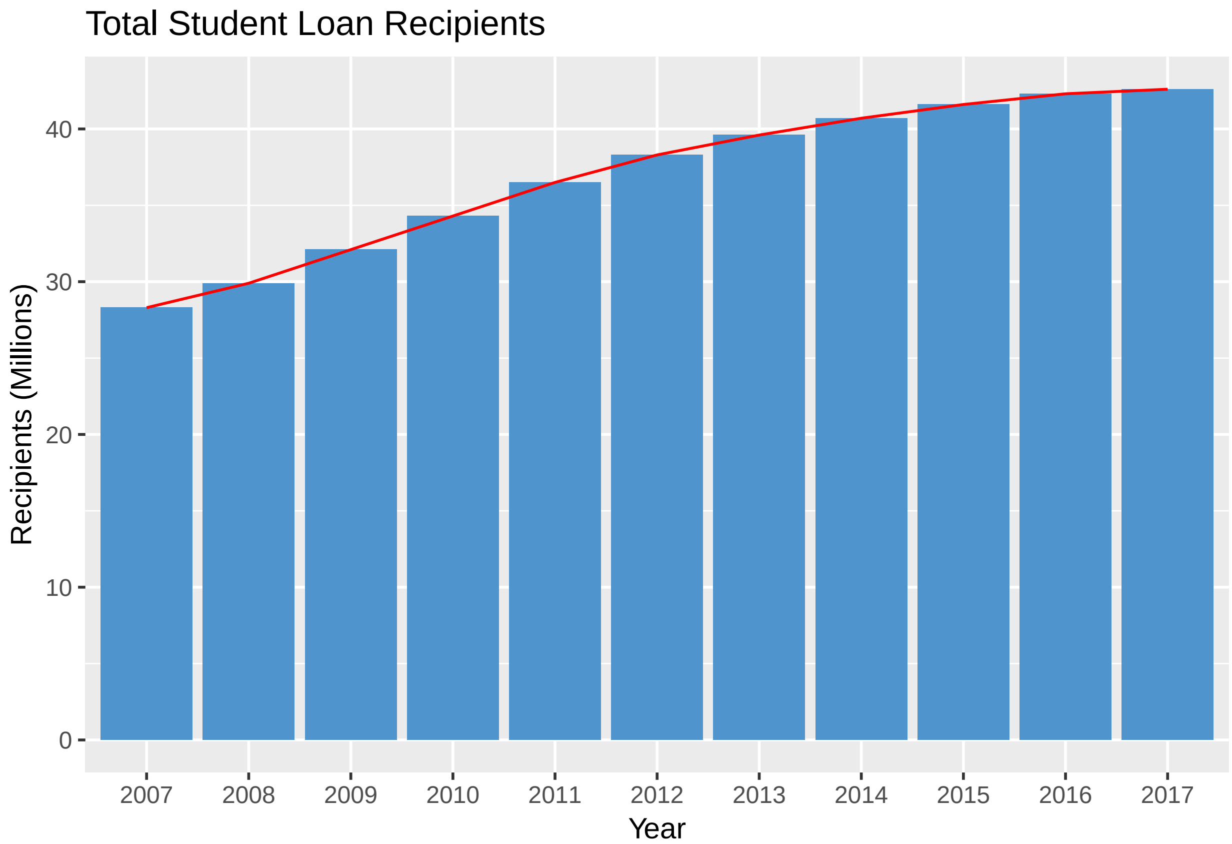
```
# Filter to a single value per year
tidyLoan %>%
  filter(loanType == "Total", period == "YR" | period == "Q4") %>%
  ggplot(aes(x=year, y=dollars)) +
    geom_col(fill= "darkgreen") +
    geom_line(aes(x=year, y=dollars, group=1), col="red") +
    ggtitle("Total Student Loan Balance") +
    xlab("Year") + ylab("Dollars (Billions)") +
    ylim(0,1500)
```



The graph shows that there has been a **significant** increase in student loan balances over the 10 years represented. In fact, within the first 6 years it doubled.

Is this because more students are going to college?

```
tidyLoan %>%
  filter(loanType == "Total", period == "YR" | period == "Q4") %>%
  ggplot() +
    geom_col(aes(x=year, y=recipients),fill= "steelblue3") +
    geom_line(aes(x=year, y=recipients, group=1), col="red") +
    ggtitle("Total Student Loan Recipients") +
    xlab("Year") + ylab("Recipients (Millions)")
```



Looking at these graphs, it appears that there is more outstanding balance over time than can be explained by simply having more students. One may conclude that there is another variable at work here, perhaps cost of education is rising? Further analysis would be required to confirm or deny that.

Example 3

This final example comes from the NYS Data website [4] and is a list of Supplemental Nutrition Assistance Program (SNAP) caseloads and expenditures. This program, previously referred to as "food stamps" is an important safety net low-income families and children.

Raw Data

Because we have the data in a CSV format, we can pull it directly from GitHub:

```
SNAP <- read_csv(url("https://raw.githubusercontent.com/lysanthus/Data607/master/Project2/SNAP.csv"),co

SNAP
```

```
## # A tibble: 11,542 x 14
##      Year Month  `Month Code` `District Code` District   `Total SNAP House~
##     <int> <chr>         <int>           <int> <chr>                   <int>
##  1   2002 Janua~            1               1 Albany                   8598
##  2   2002 Janua~            1               2 Allegany                 1812
```

[4] https://data.ny.gov/Human-Services/Supplemental-Nutrition-Assistance-Program-SNAP-Cas/dq6j-8u8z

```
## 3  2002 Janua~          1               3 Broome                6000
## 4  2002 Janua~          1               4 Cattaraug~            3041
## 5  2002 Janua~          1               5 Cayuga                2159
## 6  2002 Janua~          1               6 Chautauqua            5523
## 7  2002 Janua~          1               7 Chemung               2702
## 8  2002 Janua~          1               8 Chenango              1704
## 9  2002 Janua~          1               9 Clinton               2945
## 10 2002 Janua~          1              10 Columbia              1365
## # ... with 11,532 more rows, and 8 more variables: `Total SNAP
## #   Persons` <int>, `Total SNAP Benefits` <int>, `Temporary Assistance
## #   SNAP Households` <int>, `Temporary Assistance SNAP Persons` <int>,
## #   `Temporary Assistance SNAP Benefits` <int>, `Non-Temporary Assistance
## #   SNAP Households` <int>, `Non-Temporary Assistance SNAP Persons` <int>,
## #   `Non-Temporary Assistance SNAP Benefits` <int>
```

Looking at our raw CSV data, we can see that the data is broken out into years and months and divided by county. Furthermore, there are 3 types of observations stored as variable types: "Temporary-Assistance", "Non-Temporary Assistance" and "Total". We also have 3 actual variables: "Households", "Persons", "Benefits".

Tidying

Because we have a relatively "clean" dataset, we can proceed with reshaping it into a tidy format. We will use a similar methodology that we used in example 2 above.

```r
# Move all into a single column
tidySNAP <- gather(SNAP, key="type", value="amount", 6:14)

# Get the word "SNAP" out so we normalize the names
tidySNAP$type <- str_replace(tidySNAP$type,"\\s{1}(SNAP)","")

# Get the word "Assistance" out so we normalize the names
tidySNAP$type <- str_replace(tidySNAP$type,"\\s{1}(Assistance)","")

# Now split the "type" column, because it really contains two variable types
tidySNAP <-
  tidySNAP %>%
  extract(type, c("benefitType", "measure"),"([-[:alpha:]]+)\\s{1}(.+)")

# Then spread those into their respective columns
tidySNAP <- tidySNAP %>% spread(measure, amount)

tidySNAP
```

```
## # A tibble: 34,626 x 9
##     Year Month `Month Code` `District Code` District  benefitType Benefits
##    <int> <chr>        <int>           <int> <chr>     <chr>          <int>
##  1  2002 April            4               1 Albany    Non-Tempor~   671390
##  2  2002 April            4               1 Albany    Temporary     767340
##  3  2002 April            4               1 Albany    Total        1438730
##  4  2002 April            4               2 Allegany  Non-Tempor~   159474
##  5  2002 April            4               2 Allegany  Temporary     113401
##  6  2002 April            4               2 Allegany  Total         272875
##  7  2002 April            4               3 Broome    Non-Tempor~   529332
##  8  2002 April            4               3 Broome    Temporary     446036
##  9  2002 April            4               3 Broome    Total         975368
```

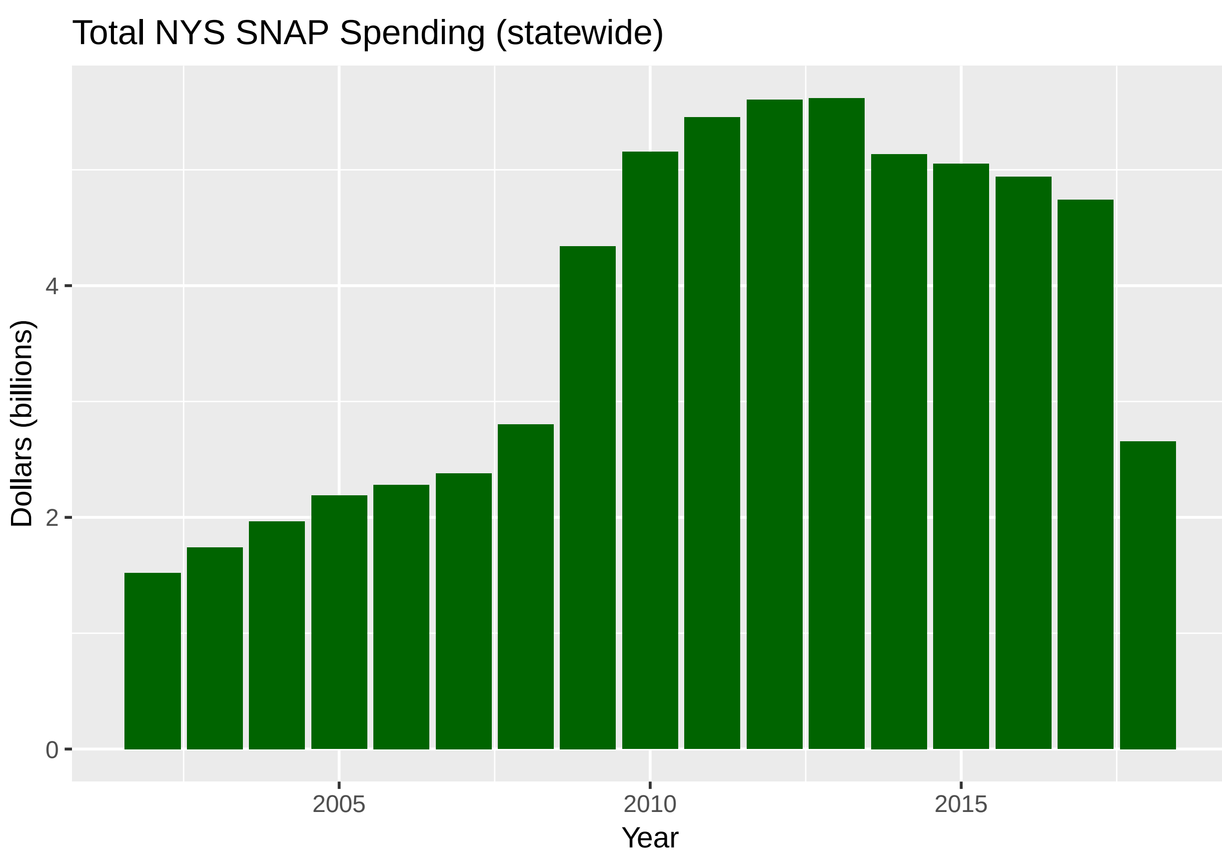
```
## 10  2002 April            4               4 Cattarau~ Non-Tempor~   297399
## # ... with 34,616 more rows, and 2 more variables: Households <int>,
## #   Persons <int>
```

Now we have a proper tidy data set to analyze.

Analysis

First, let's look at how many dollars are being spent per year:

```
tidySNAP %>%
    filter(benefitType == "Total") %>%
    group_by(Year) %>% summarize(spend = sum(Benefits, rm.na=TRUE)/1000000000,
                                 people = sum(Persons, rm.na=TRUE),
                                 perPerson = spend/people) %>%
    ggplot() +
    geom_col(aes(x=Year, y=spend),fill= "darkgreen") +
    ggtitle("Total NYS SNAP Spending (statewide)") +
    xlab("Year") + ylab("Dollars (billions)")
```



This gives us some idea of the total amounts being spent, though there is no evidence that these numbers are adjusted for inflation in any way.

We could also compare the dollars per resident of each county to get a per-capita spending value (with the assistance of a census dataset).

```
# Get the census data
census <- read_csv(url(
```

Table 3: Top 10 Total SNAP Spending by County (2017)

Year	County	Spend per Capita
2017	Chautauqua	287.51
2017	Montgomery	267.77
2017	Sullivan	260.97
2017	Chemung	258.54
2017	Oneida	253.71
2017	Erie	251.29
2017	Monroe	250.59
2017	Broome	238.14
2017	Oswego	230.11
2017	Onondaga	227.42

```
  "https://raw.githubusercontent.com/lysanthus/Data607/master/Project2/census.csv"),
  col_names = TRUE)

# We're going to limit to 2017 only
census <- census %>% filter(Year == 2017, Geography != "New York State")

# Remove the word "County" for easy joining
census$Geography <- str_replace(census$Geography,"\\sCounty","")

# Same thing with the SNAP data
SNAP2017 <- tidySNAP %>%
        filter(benefitType == "Total", Year == "2017") %>%
        group_by(Year, District) %>% summarize(spend = sum(Benefits, rm.na=TRUE),
                                 people = mean(Persons, rm.na=TRUE))

# Join the datasets and get some measures
SNAPbyCounty <-
  left_join(SNAP2017,census, by=c("District"="Geography")) %>%
  select(year = Year.x, county= District, spend, people, population = Population) %>%
  mutate(spendPC = round(spend / population,2), recipPC = (people / population))

top10SNAP <- SNAPbyCounty %>% top_n(10,spendPC) %>% arrange(desc(spendPC)) %>%
  select(year, county, spendPC)

kable(top10SNAP, col.names = c("Year","County","Spend per Capita"),
      caption="Top 10 Total SNAP Spending by County (2017)") %>%
  kable_styling(bootstrap_options = "striped", full_width = FALSE, position = "left")
```