Lecture 6: Data Transformation

More Data Transformation with dplyr

Yujin Jeong

STATS 195

Drought Data

Drought data

In this section, we will use California Drought data which you can download from Canvas.

```
library(readr)
drought <- read_csv("CADrought.csv",
               col_types = cols(ReleaseDate = col_date(format = "%Y%m%d"),
                                ValidEnd = col_date(format = "%Y-%m-%d"),
                                ValidStart = col_date(format = "%Y-%m-%d")))
```

Exploring the data

Let’s view the structure of the data set:

```
str(drought)
```

```
spc_tbl_ [9,106 × 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ ReleaseDate      : Date[1:9106], format: "2018-09-18" "2018-09-11" "2018-09-04" "2018-08-28" ...
 $ FIPS             : chr [1:9106] "06001" "06001" "06001" "06001" ...
 $ County           : chr [1:9106] "Alameda County" "Alameda County" "Alameda County" "Alameda County" ...
 $ State            : chr [1:9106] "CA" "CA" "CA" "CA" ...
 $ None             : num [1:9106] 0 0 0 0 0 0 0 0 0 0 ...
 $ D0               : num [1:9106] 100 100 100 100 100 100 100 100 100 100 ...
 $ D1               : num [1:9106] 0 0 0 0 0 0 0 0 0 0 ...
 $ D2               : num [1:9106] 0 0 0 0 0 0 0 0 0 0 ...
 $ D3               : num [1:9106] 0 0 0 0 0 0 0 0 0 0 ...
 $ D4               : num [1:9106] 0 0 0 0 0 0 0 0 0 0 ...
 $ ValidStart       : Date[1:9106], format: "2018-09-18" "2018-09-11" "2018-09-04" "2018-08-28" ...
 $ ValidEnd         : Date[1:9106], format: "2018-09-24" "2018-09-17" "2018-09-10" "2018-09-03" ...
 $ StatisticFormatID: num [1:9106] 2 2 2 2 2 2 2 2 2 2 ...
```

Note that D0 is the least intense level of droughts and D4 the most intense. The columns from `None` to `D4` always add up to 100%.

Exploring the data

Before we start using the dataset for our analysis, we should do some sanity checks to make sure that the data we have is indeed the data we expect. For example: California has 58 counties, so we should have all 58 counties represented. We check this using `summarize` and the function `n_distinct`.

```
# Sanity check: should have all 58 counties
drought %>% summarize(distinct = n_distinct(County))
```

```
## # A tibble: 1 × 1
##   distinct
##      <int>
## 1       58
```

Exploring the data

Let’s select and keep important columns.

- `` `ReleaseDate` `` seems to be the same as `` `ValidStart` ``, and `` `ValidEnd` `` is always 6 days later than `` `ValidStart` ``. Therefore, we may keep only the `` `ValidStart` `` column.
- `` `State` `` is the same for all values in our dataset, so we can safely drop it.
- We are not going to use `` `FIPS` ``, `` `StatisticFormatID` ``, and `` `None` ``.

```
# select important columns
drought_small <- drought %>% select(County, Date = ValidStart, D0:D4)
```

Some dplyr practice

Select the rows with D4 being 100.

Some dplyr practice

Select the rows with D4 being 100.

```
drought_small %>% filter(D4 == 100)
```

Some dplyr practice

Which county experienced the most number of weeks with 100% land area in D4?

Some dplyr practice

Which county experienced the most number of weeks with 100% land area in D4?

```
drought_small %>%
    filter(D4 == 100) %>%
    # FILL IN!
```

Some dplyr practice

Which county experienced the most number of weeks with 100% land area in D4?

Hint:

```
drought_small %>%
    filter(D4 == 100) %>%
    group_by(County) %>%
    # FILL IN!
```

Some dplyr practice

Which county experienced the most number of weeks with 100% land area in D4?

```
drought_small %>%
    filter(D4 == 100) %>%
    group_by(County) %>%
    summarize(count = n()) %>%
    arrange(desc(count))
```

```
## # A tibble: 22 × 2
##    County                 count
##    <chr>                  <int>
##  1 Santa Barbara County      68
##  2 Ventura County            68
##  3 Fresno County             30
##  4 Kings County              30
##  5 Madera County             30
##  6 Mariposa County           30
##  7 Merced County             30
##  8 San Joaquin County        30
##  9 Stanislaus County         30
## 10 Tulare County             30
## # … with 12 more rows
## # ℹ Use `print(n = ...)` to see more rows
```

Some dplyr practice

Which counties experienced 100% land area in D4 at any time? (Hint: try distinct.)

Some dplyr practice

Which counties experienced 100% land area in D4 at any time? (Hint: try distinct.)

```
drought_small %>% filter(D4 == 100) %>%
    distinct(County)
```

```
## # A tibble: 22 × 1
##    County
##    <chr>
##  1 Alpine County
##  2 Amador County
##  3 Calaveras County
##  4 El Dorado County
##  5 Fresno County
##  6 Kings County
##  7 Madera County
##  8 Mariposa County
##  9 Merced County
## 10 Mono County
## # … with 12 more rows
## # ℹ Use `print(n = ...)` to see more rows
```

Some dplyr practice

Select rows with date "2018-09-18" and return the entries in descending order of D0.

Some dplyr practice

Select rows with date "2018-09-18" and return the entries in descending order of D0.

```
drought_small %>% filter(Date == "2018-09-18") %>% arrange(desc(D0))
```

Tidying and Plotting

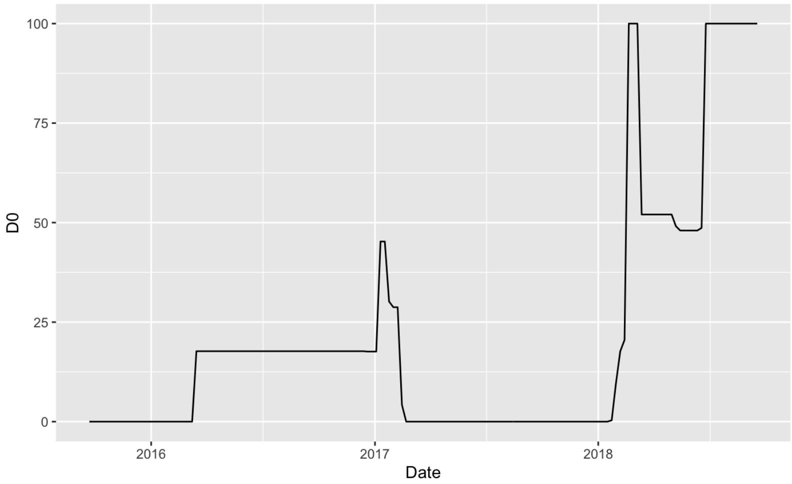
For now, let's focus on Santa Clara county. Threfore, first we should use `dplyr`'s `filter` to get a new dataset with just the observations from Santa Clara:

```
# filter for just Santa Clara County
drought_sc <- drought_small %>% filter(County == "Santa Clara County")
```

Tidying and Plotting

Next, let’s make a line plot of `D0` against `Date`:

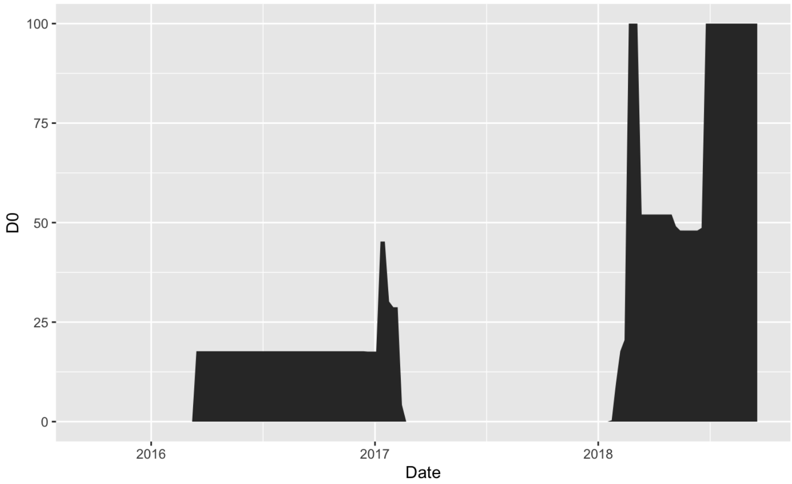
```
ggplot(data = drought_sc) +
    geom_line(mapping = aes(x = Date, y = D0))
```



Tidying and Plotting

To make an area plot instead, replace `geom_line` with `geom_area`:

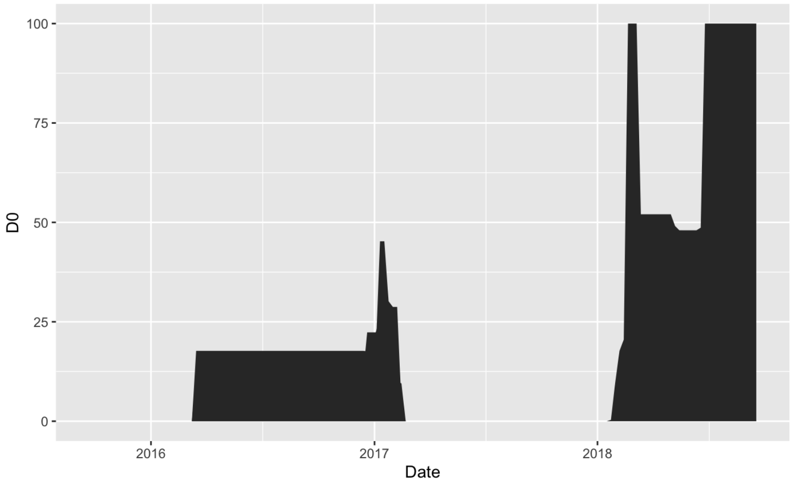
```
ggplot(data = drought_sc) +
    geom_area(mapping = aes(x = Date, y = D0))
```



Tidying and Plotting

How can we plot `D0` and `D1` against `Date`? We could try this:

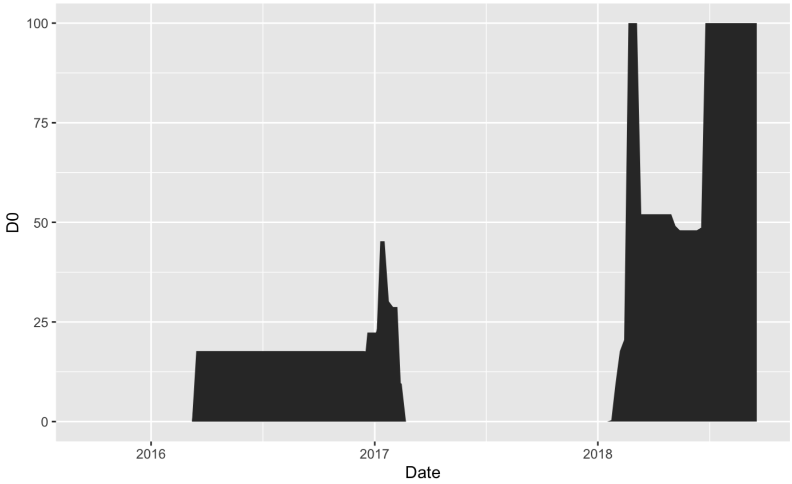
```
ggplot(data = drought_sc) +
    geom_area(mapping = aes(x = Date, y = D0)) +
    geom_area(mapping = aes(x = Date, y = D1))
```



There are a number of issues with this approach! What are they?

Tidying and Plotting

There are a number of issues with this approach:



- It doesn't scale. To display `D0` to `D4`, we need 3 more lines of code. What if there were 10 drought levels instead?
- Both areas are filled in with black and are stacked on top of each other, so we can't tell the difference between the two.
- The y-axis reads `D0` even though the values plotted are for `D0` and `D1`.

How should we reorganize the dataset to fix these issues?

Tidying and Plotting

How should we reorganize the dataset to fix these issues?

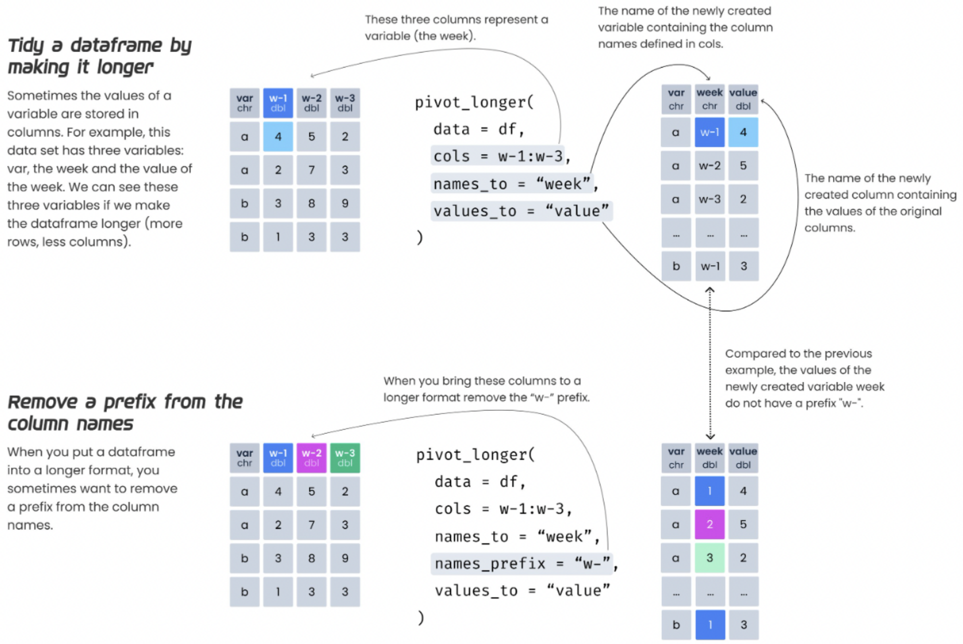
We need to reorganize the dataset such that there is

- a `` `Drought level` `` column with factors `` `D0` `` to `` `D4` ``, and
- a corresponding `` `Percent of land area` `` which gives the percent of land area in that level of drought.

In Hadley Wickham’s terminology, the original dataset is not "tidy" and we have to make it so.

Tidying and Plotting

We can use `tidyr`'s `pivot_longer` function to accomplish this.



Source: Christian Burkhart Twitter.

Tidying and Plotting

```
library(tidyr)
drought_sc_long <- drought_sc %>% pivot_longer(D0:D4,
                                 names_to = "Drought level",
                                 values_to = "Percent of land area")
```

```
drought_sc
```

```
# A tibble: 157 × 7
   County             Date          D0    D1    D2    D3 <d
   <chr>              <date>     <dbl> <dbl> <dbl> <dbl> <d
 1 Santa Clara County 2018-09-18   100     0     0     0
 2 Santa Clara County 2018-09-11   100     0     0     0
 3 Santa Clara County 2018-09-04   100     0     0     0
 4 Santa Clara County 2018-08-28   100     0     0     0
 5 Santa Clara County 2018-08-21   100     0     0     0
 6 Santa Clara County 2018-08-14   100     0     0     0
 7 Santa Clara County 2018-08-07   100     0     0     0
 8 Santa Clara County 2018-07-31   100     0     0     0
 9 Santa Clara County 2018-07-24   100     0     0     0
10 Santa Clara County 2018-07-17   100     0     0     0
# … with 147 more rows
```

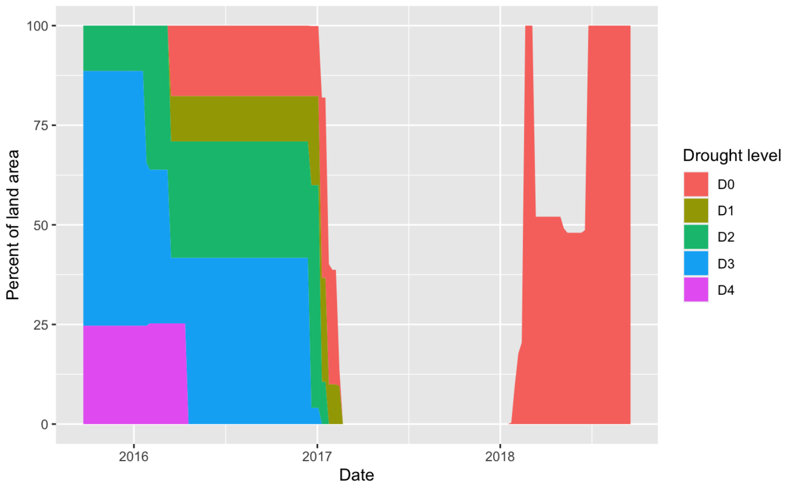
```
drought_sc_long
```

```
# A tibble: 785 × 4
   County             Date       `Drought level` `Percent o
   <chr>              <date>     <chr>                <dbl>
 1 Santa Clara County 2018-09-18 D0                   100
 2 Santa Clara County 2018-09-18 D1                   0
 3 Santa Clara County 2018-09-18 D2                   0
 4 Santa Clara County 2018-09-18 D3                   0
 5 Santa Clara County 2018-09-18 D4                   0
 6 Santa Clara County 2018-09-11 D0                   100
 7 Santa Clara County 2018-09-11 D1                   0
 8 Santa Clara County 2018-09-11 D2                   0
 9 Santa Clara County 2018-09-11 D3                   0
10 Santa Clara County 2018-09-11 D4                   0
# … with 775 more rows
```

Tidying and Plotting

Now we can easily plot `D0` to `D4` on the same chart (because our column names have spaces in them, we need to surround them with backticks (`) so that R interprets the code properly):

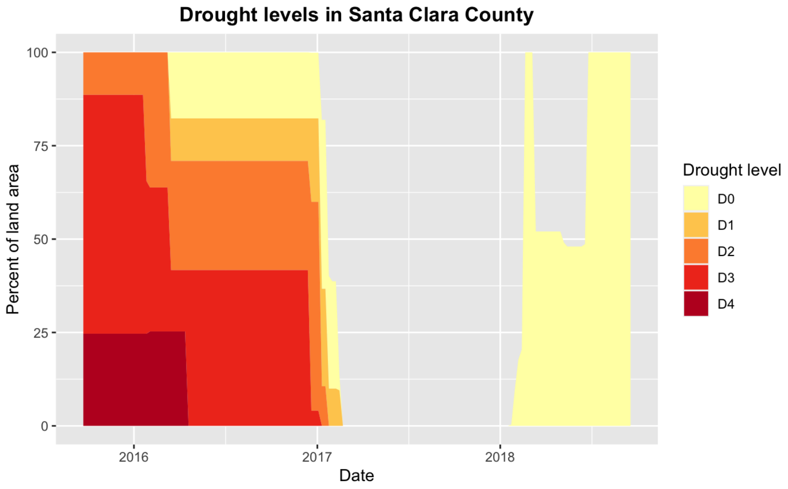
```
# make area plot
ggplot(data = drought_sc_long) +
    geom_area(mapping = aes(x = Date, y = `Percent of land area`,
                            fill = `Drought level`))
```



Tidying and Plotting

Let's use `scale_fill_brewer` to make the area colors more theme-appropriate, and add a title:

```
# make area plot
ggplot(data = drought_sc_long) +
    geom_area(mapping = aes(x = Date, y = `Percent of land area`,
                            fill = `Drought level`)) +
    labs(title = "Drought levels in Santa Clara County") +
    scale_fill_brewer(palette = "YlOrRd") +
    theme(plot.title = element_text(face = "bold", hjust = 0.5))
```



Scaling our code

The plot we just made was for Santa Clara County. What if we were interested in another county, say Monterey County, instead? Do we have to type up all the code again?

Scaling our code

The plot we just made was for Santa Clara County. What if we were interested in another county, say Monterey County, instead? Do we have to type up all the code again?

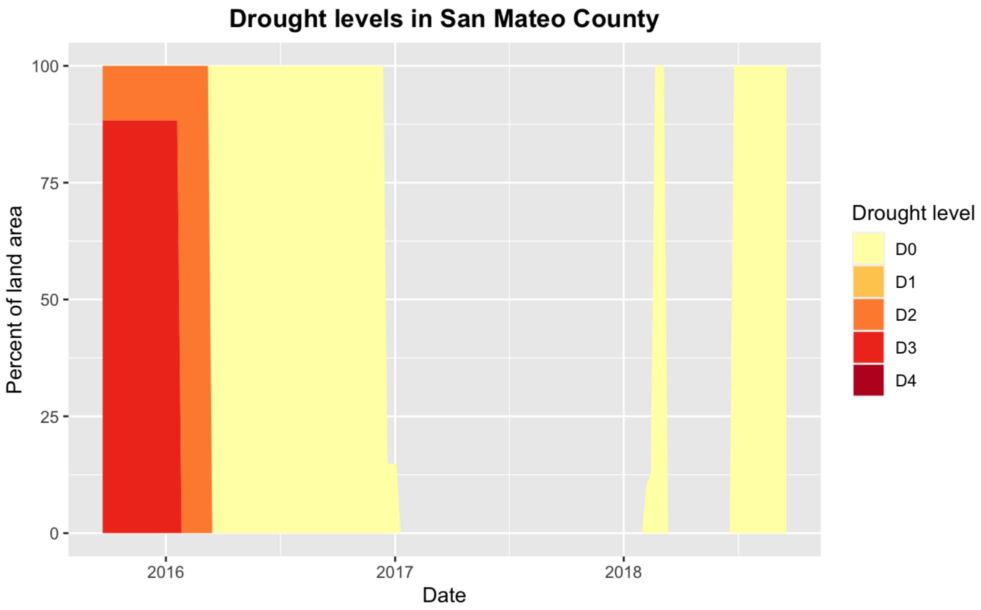
NO! Let's make it as a function!

```
droughtPlot<-function(county="Santa Clara County",df=drought){
  df %>% select(County, Date = ValidStart, D0:D4) %>%
    filter(County == county) %>%
    pivot_longer(D0:D4, names_to = "Drought level",
                                  values_to = "Percent of land area") %>%
    ggplot() +
    geom_area(mapping = aes(x = Date, y = `Percent of land area`,
                            fill = `Drought level`)) +
    labs(title = paste("Drought levels in", county)) + #Note this change
    scale_fill_brewer(palette = "YlOrRd") +
    theme(plot.title = element_text(face = "bold", hjust = 0.5))
}
```

Scaling our code

Let’s plot drought levels of San Mateo Country.

```
droughtPlot("San Mateo County")
```



2016 Voting Data

2016 Voting Data

In this section, we will use 2016 US presidential election county data which you can download from Canvas.

```
election <- read_csv("2016_US_County_Level_Presidential_Results.csv")
str(election)
```

```
spc_tbl_ [3,141 × 11] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ ...1          : num [1:3141] 0 1 2 3 4 5 6 7 8 9 ...
 $ votes_dem     : num [1:3141] 93003 93003 93003 93003 93003 ...
 $ votes_gop     : num [1:3141] 130413 130413 130413 130413 130413 ...
 $ total_votes   : num [1:3141] 246588 246588 246588 246588 246588 ...
 $ per_dem       : num [1:3141] 0.377 0.377 0.377 0.377 0.377 ...
 $ per_gop       : num [1:3141] 0.529 0.529 0.529 0.529 0.529 ...
 $ diff          : num [1:3141] 37410 37410 37410 37410 37410 ...
 $ per_point_diff: chr [1:3141] "15.17%" "15.17%" "15.17%" "15.17%" ...
 $ state_abbr    : chr [1:3141] "AK" "AK" "AK" "AK" ...
 $ county_name   : chr [1:3141] "Alaska" "Alaska" "Alaska" "Alaska" ...
 $ combined_fips : num [1:3141] 2013 2016 2020 2050 2060 ...
```

There are 3,141 rows in total, matching the number of counties in the US.

2016 Voting Data

The dataset contains the following columns:

```
names(election)
```

```
## [1] "...1"           "votes_dem"      "votes_gop"      "total_votes"    "per_dem"        "per_gop"
## [7] "diff"           "per_point_diff" "state_abbr"     "county_name"    "combined_fips"
```

- `per_dem` and `per_gop` refer to the percentage of votes to Democrats and Republicans respectively.
- `diff` represents the absolute difference between Republican votes - Democrat votes.
- `per_point_diff` represents this difference as a percentage of total votes.
- `combined_fips` is a 5-digit code identifying the county.

We are interested in whether a given county had more Republican or Democrat votes. We recompute the `diff` and `per_point_diff` columns as:

```
election <- election %>% mutate(diff = votes_gop - votes_dem,
                                per_point_diff = diff / total_votes * 100)
```

Exploring the data

Compute percentage of popular vote won by each party:

Exploring the data

Compute percentage of popular vote won by each party:

```
paste0("Republican % of popular vote: ",
       round(sum(election$votes_gop) / sum(election$total_votes) * 100, digits = 1),
       "%")
paste0("Democrat % of popular vote: ",
       round(sum(election$votes_dem) / sum(election$total_votes) * 100, digits = 1),
       "%")
```

```
## [1] "Republican % of popular vote: 47.3%"
## [1] "Democrat % of popular vote: 47.5%"
```

Although Clinton lost the presidential election, she won the popular vote.

Exploring the data

Compute number of counties won by each party:

Exploring the data

Compute number of counties won by each party:

```
paste0("# of counties won by Republican: ",
       sum(election$votes_gop > election$votes_dem))
paste0("# of counties won by Democrat: ",
       sum(election$votes_gop < election$votes_dem))
```

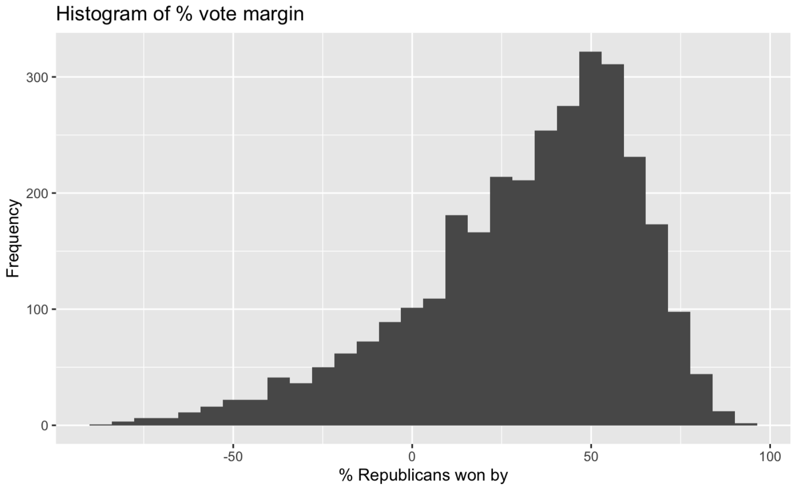
```
## [1] "# of counties won by Republican: 2654"
## [1] "# of counties won by Democrat: 487"
```

Painting a completely different picture, Trump won 2654 out of 3141 counties (or 84.5% of all counties). Clinton only won 487 counties. This suggests that Clinton won in counties with large populations, or that the margin of victory was slimmer in the counties that Trump won compared with the counties that Clinton won.

Exploring the data

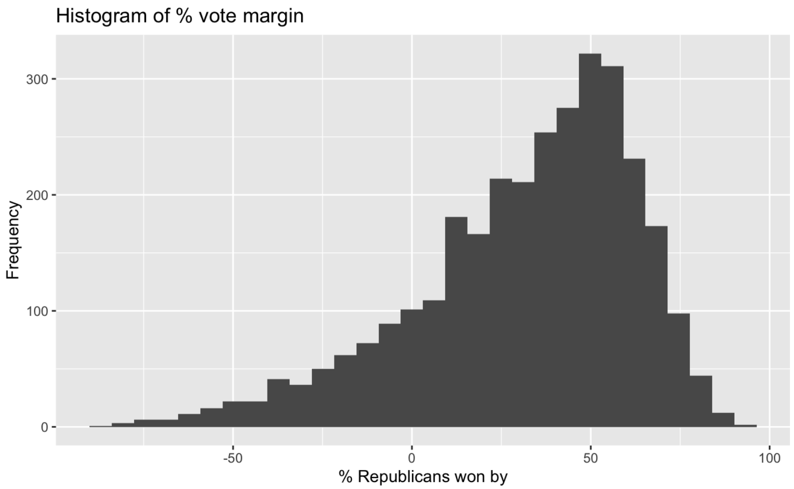
One theory is that Clinton won her counties by a huge margin percentage-wise, while Trump won his counties by a slim margin percentage-wise. To test this theory, we could plot a histogram of the `per_point_diff`:

```
ggplot() +
    geom_histogram(data = election, mapping = aes(x = per_point_diff)) +
    labs(title = "Histogram of % vote margin",
         x = "% Republicans won by", y = "Frequency")
```



Exploring the data

One theory is that Clinton won her counties by a huge margin percentage-wise, while Trump won his counties by a slim margin percentage-wise. To test this theory, we could plot a histogram of the `per_point_diff`:

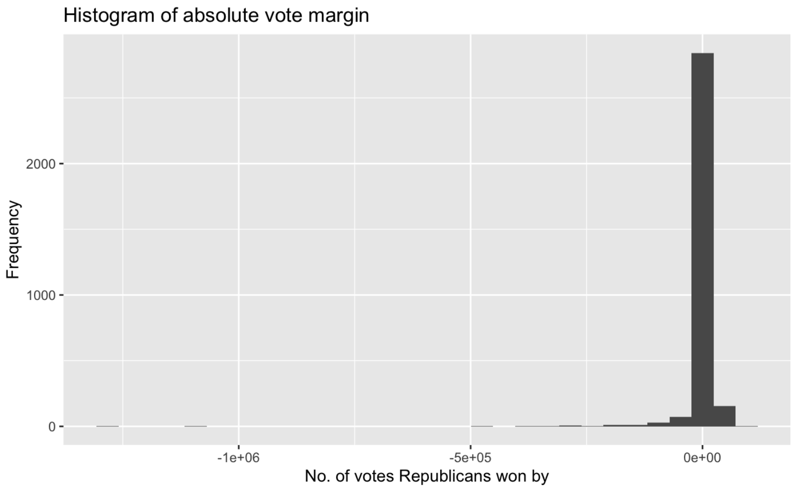


The chart does not support the theory that Trump had narrower margins of victory in the counties that he won: he won a sizable number of counties with > 50% vote difference.

Exploring the data

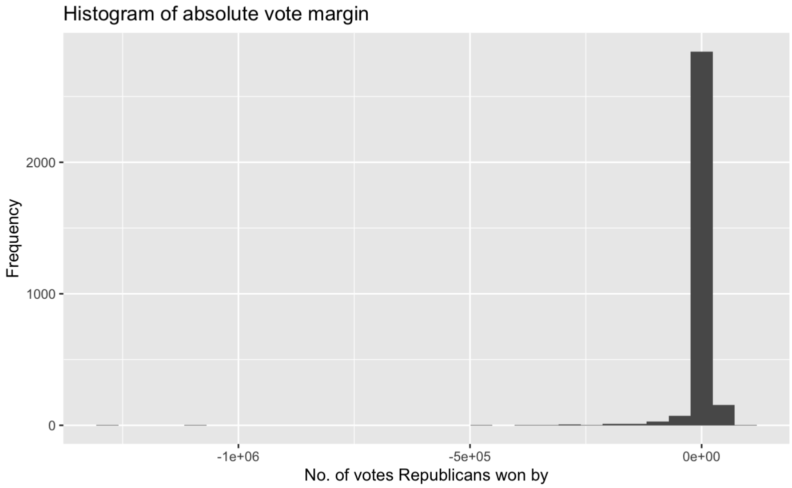
Let’s try plotting a histogram of `diff` to look at absolute differences instead:

```
ggplot() +
    geom_histogram(data = election, mapping = aes(x = diff)) +
    labs(title = "Histogram of absolute vote margin",
        x = "No. of votes Republicans won by", y = "Frequency")
```



Exploring the data

Let’s try plotting a histogram of `` `diff` `` to look at absolute differences instead:



This chart is very different! In the counties that Clinton won, she won it by extremely large margins in terms of absolute votes. Thus, even though she won very few counties compared to Trump, these large margins meant that she could actually win the popular vote.

Exploring the data

Show the top 50 counties with largest absolute vote difference:

```
election %>% select(State = state_abbr, County = county_name, diff) %>%
    # FILL IN!
```

Exploring the data

Show the top 50 counties with largest absolute vote difference:

```
election %>% select(State = state_abbr, County = county_name, diff) %>%
    mutate(abs_diff = abs(diff)) %>%
    arrange(desc(abs_diff)) %>%
    select(State, County, `Vote difference` = diff) %>%
    head(n = 50)
```

```
# A tibble: 50 × 3
   State County                `Vote difference`
   <chr> <chr>                             <dbl>
 1 CA    Los Angeles County             -1273485
 2 IL    Cook County                    -1088369
 3 NY    Kings County                    -461433
 4 WA    King County                     -459368
 5 NY    New York County                 -456546
 6 PA    Philadelphia County             -455124
 7 CA    Alameda County                  -395162
 8 CA    Santa Clara County              -346020
 9 NY    Queens County                   -334839
10 MA    Middlesex County                -292756
# … with 40 more rows
```

Drawing Maps

Let's start by drawing a map of the USA. The `maps` package contains a lot of outlines of continents, countries, states, and counties. `ggplot2`'s `map_data()` function puts these outlines in data frame format, which then allows us to plot them with `ggplot()`.

```
# install.packages("maps")
library(maps)
map_USA <- map_data("usa")
head(map_USA)
```

```
##        long      lat group order region subregion
## 1 -101.4078 29.74224     1     1   main      <NA>
## 2 -101.3906 29.74224     1     2   main      <NA>
## 3 -101.3620 29.65056     1     3   main      <NA>
## 4 -101.3505 29.63911     1     4   main      <NA>
## 5 -101.3219 29.63338     1     5   main      <NA>
## 6 -101.3047 29.64484     1     6   main      <NA>
```

Each row in the `map_USA` dataset is one point on the outline of the USA. We are going to use `ggplot2`'s `geom_polygon` to connect these points.

Drawing Maps

It turns out that you can’t draw a map of the US with just one polygon: there are islands (e.g. Long Island) which form separate polygons. To draw just the "main" mainland, filter as follows:

```
map_USA_main <- map_USA %>% filter(region == "main")
```

```
ggplot() +
    geom_polygon(data = map_USA_main, mapping = aes(x = long, y = lat)) +
    coord_quickmap() # this just keeps the correct aspect ratio
```

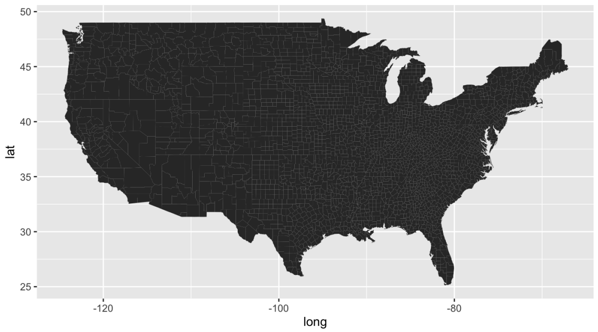
Drawing Maps

It turns out that it's not so easy to get mapping data at the county level with FIPS codes. I've provided a dataset on canvas to save you this trouble:

```
map_county_fips <- readRDS("county_map_fips.rds")
```

Now let's draw a county map using code that's very similar to what we had for drawing the map of the USA:

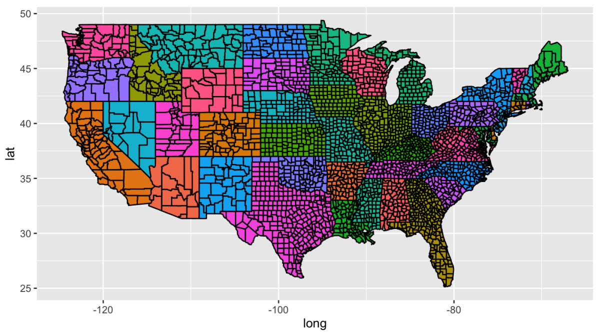
```
ggplot() +
    geom_polygon(data = map_county_fips, mapping = aes(x = long, y = lat, group = group)) +
    coord_quickmap()
```



Drawing Maps

Now, let's draw a map with black outlines for the counties, and different colors for each state:

```
ggplot() +
    geom_polygon(data = map_county_fips,
                 mapping = aes(x = long, y = lat, group = group, fill = region),
                 col = "black") +
    coord_quickmap() +
    theme(legend.position="none")
```



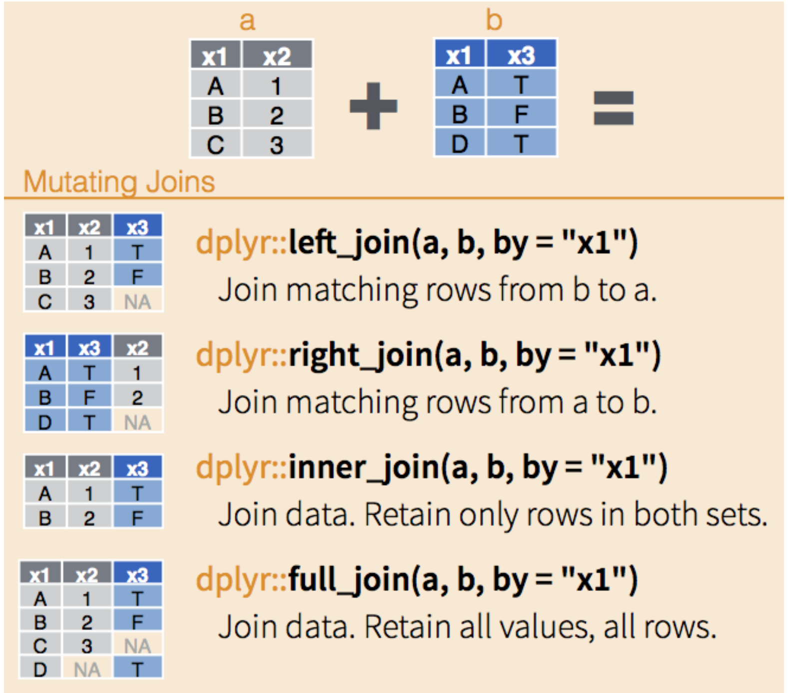
Now we want the fills of the counties to depend on our elections data.

Combining Data with dplyr

In order to have the fills of the counties depend on our elections data, we need to get information from our `election` tibble to the `map_county_fips` tibble.

We can achieve this by using `dplyr`'s **left-join**.

Combining Data with dplyr



From: https://mikoontz.github.io/data-carpentry-week/lesson_joins.html

Combining Data with dplyr

First, extract the columns we need from `election` into a new, smaller data frame:

```
county_per_diff <- election %>%
    select(fips = combined_fips, percent_diff = per_point_diff)
```

Combining Data with dplyr

Next, we join this smaller data frame to the mapping data using `left_join`:

```
# join elections data to mapping data
map_county_per_diff <- map_county_fips %>%
    left_join(county_per_diff, by = "fips")
```

```
head(map_county_fips)
```

```
##        long      lat group order  region subregion fips
## 1 -86.50517 32.34920     1     1 alabama   autauga 1001
## 2 -86.53382 32.35493     1     2 alabama   autauga 1001
## 3 -86.54527 32.36639     1     3 alabama   autauga 1001
## 4 -86.55673 32.37785     1     4 alabama   autauga 1001
## 5 -86.57966 32.38357     1     5 alabama   autauga 1001
## 6 -86.59111 32.37785     1     6 alabama   autauga 1001
```

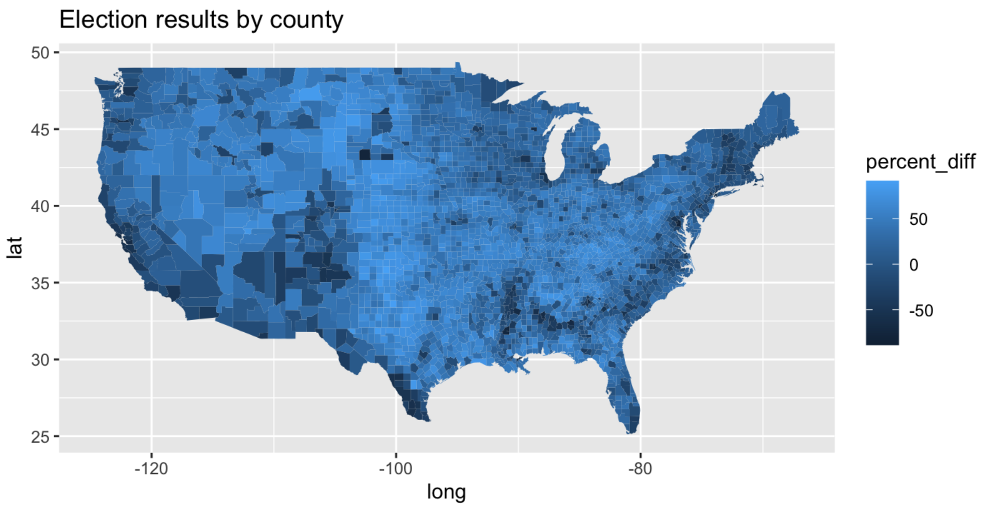
```
head(map_county_per_diff)
```

```
##        long      lat group order  region subregion fips
## 1 -86.50517 32.34920     1     1 alabama   autauga 1001
## 2 -86.53382 32.35493     1     2 alabama   autauga 1001
## 3 -86.54527 32.36639     1     3 alabama   autauga 1001
## 4 -86.55673 32.37785     1     4 alabama   autauga 1001
## 5 -86.57966 32.38357     1     5 alabama   autauga 1001
## 6 -86.59111 32.37785     1     6 alabama   autauga 1001
```

Combining Data with dplyr

Finally, we use `ggplot()` to plot the data:

```
ggplot(data = map_county_per_diff, mapping = aes(x = long, y = lat, group = group)) +
    geom_polygon(aes(fill = percent_diff)) +
    coord_quickmap() +
    labs(title = "Election results by county")
```



Final Touch

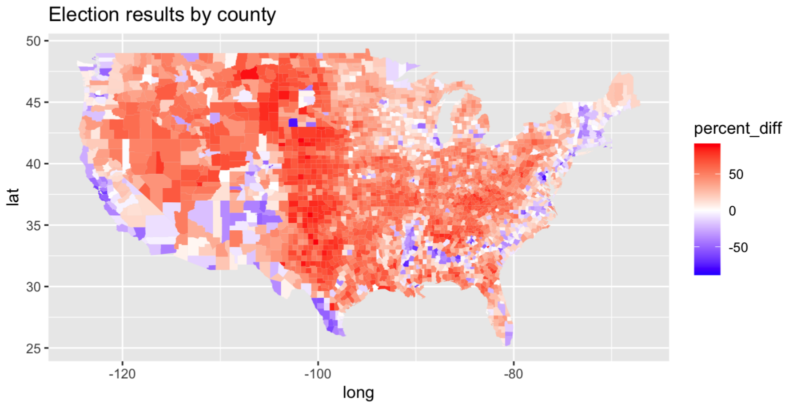
There are two things we can do to improve on it:

- The color scale is currently different shades of blue. Something more informative would be counties that voted very Republican being red, those that voted very Democrat being blue, and those that voted evenly being white.
- The "lat" and "long" axes, as well as the grey background with grids, are not helpful for interpreting maps.

Final Touch

Let’s change the color scale first.

```
g1 <- ggplot(data = map_county_per_diff, mapping = aes(x = long, y = lat, group = group)) +
    geom_polygon(aes(fill = percent_diff)) +
    scale_fill_gradient2(low = "blue", high = "red") +
    coord_quickmap() +
    labs(title = "Election results by county")
g1
```



Final Touch

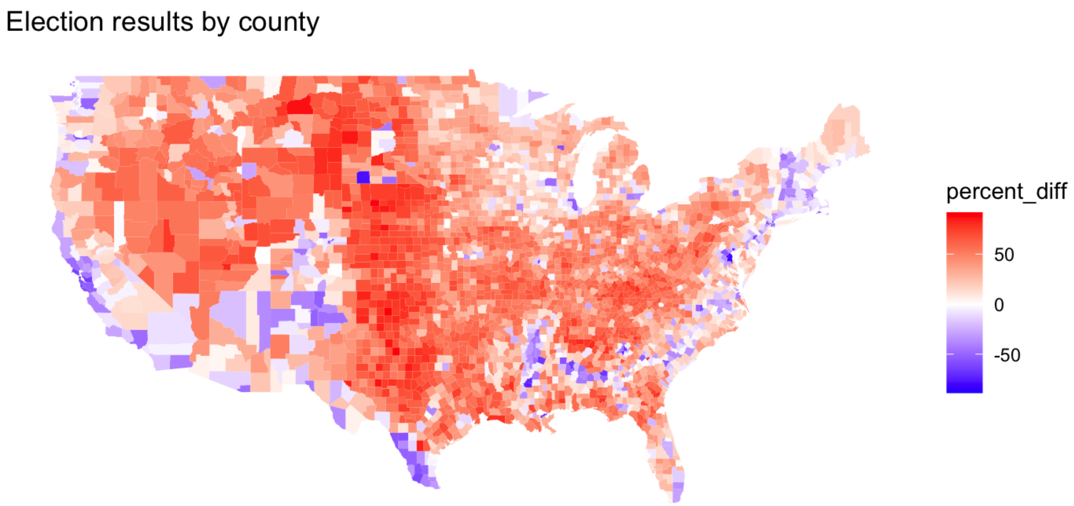
To remove parts of the plot which are not helpful, let's add this theme

```
map_theme <- theme(
    axis.title.x = element_blank(),
    axis.text.x  = element_blank(),
    axis.ticks.x = element_blank(),
    axis.title.y = element_blank(),
    axis.text.y  = element_blank(),
    axis.ticks.y = element_blank(),
    panel.background = element_rect(fill = "white")
)
```

Final Touch

Next, add `map_theme` to the `g1` function call:

```
g1 + map_theme
```



Final Touch

If we want to draw state boundaries as well:

```
map_state <- map_data("state")
ggplot(data = map_county_per_diff, mapping = aes(x = long, y = lat, group = group)) +
    geom_polygon(aes(fill = percent_diff)) +
    geom_polygon(data = map_state, fill = NA, color = "black") +
    scale_fill_gradient2(low = "blue", high = "red") +
    coord_quickmap() +
    labs(title = "Election results by county") +
    map_theme
```