Vignette of ggplot2

Create: Alex Khaykin | Extend: Ross Boehme

Create: 2023-03-30 | Extend: 2023-04-26

INTRODUCTION

ggplot2 was the first of Hadley Wickham's tidy packages and was intended to simplify and streamline the appearance of R graphics. In this vignette, we will walk through key plots in ggplot2 using the 'congress_age' dataset from fivethirtyeight and best tidy practices.

First, load the fivethirtyeight package and the congress_age dataset:

```
install.packages("fivethirtyeight",repos = "http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/rossboehme/AppData/Local/R/win-library/4.2'
## (as 'lib' is unspecified)
```

```
## package 'fivethirtyeight' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\rossboehme\AppData\Local\Temp\RtmpG0rQcu\downloaded_packages
```

```
library(fivethirtyeight)
```

```
## Warning: package 'fivethirtyeight' was built under R version 4.2.3
```

```
## Some larger datasets need to be installed separately, like senators and
## house_district_forecast. To install these, we recommend you install the
## fivethirtyeightdata package by running:
## install.packages('fivethirtyeightdata', repos =
## 'https://fivethirtyeightdata.github.io/drat/', type = 'source')
```

```
data("congress_age")
str(congress_age)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    18635 obs. of  13 variables:
##  $ congress  : int  80 80 80 80 80 80 80 80 80 80 ...
##  $ chamber   : chr  "house" "house" "house" "house" ...
##  $ bioguide  : chr  "M000112" "D000448" "S000001" "E000023" ...
##  $ firstname : chr  "Joseph" "Robert" "Adolph" "Charles" ...
```

```
##  $ middlename: chr  "Jefferson" "Lee" "Joachim" "Aubrey" ...
##  $ lastname  : chr  "Mansfield" "Doughton" "Sabath" "Eaton" ...
##  $ suffix    : chr  NA NA NA NA ...
##  $ birthday  : Date, format: "1861-02-09" "1863-11-07" ...
##  $ state     : chr  "TX" "NC" "IL" "NJ" ...
##  $ party     : chr  "D" "D" "D" "R" ...
##  $ incumbent : logi  TRUE TRUE TRUE TRUE FALSE FALSE ...
##  $ termstart : Date, format: "1947-01-03" "1947-01-03" ...
##  $ age       : num  85.9 83.2 80.7 78.8 78.3 78 77.9 76.8 76 75.8 ...
```

Load the tidyverse and ggplot2 packages:

```
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.8     v dplyr   1.1.0
## v tidyr   1.3.0     v stringr 1.5.0
## v readr   2.1.4     v forcats 0.5.2
## v purrr   1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(dplyr)
```

What are the top 10 most common first names of congresspeople?

First, we need to use the dplyr package to count and then sort the number of first names in the dataset.

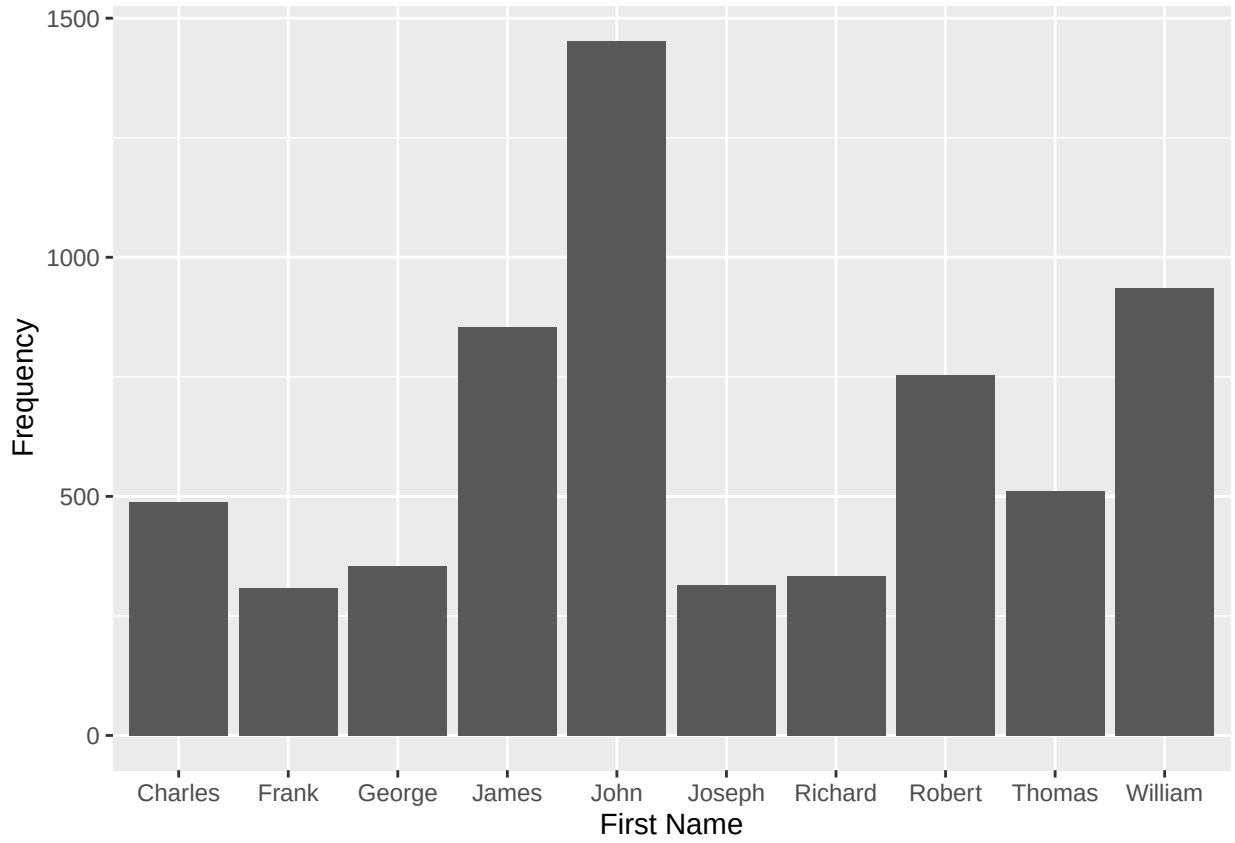
```
first_names <- congress_age %>%
  group_by(firstname) %>%
  count(firstname) %>%
  arrange(desc(n))
head(first_names)
```

```
## # A tibble: 6 x 2
## # Groups:   firstname [6]
##   firstname     n
##   <chr>     <int>
## 1 John       1453
## 2 William     935
## 3 James       855
## 4 Robert      753
## 5 Thomas      512
## 6 Charles     488
```

This uses the **group_by** function to group the congresspeople by their first names so they we can **count** them, and then we **arrange** them in descending (**desc**) order by the count (**n**) we generated.

Barplot with geom_bar Barplots with geom_bar are a very quick way to look at summary data like counts. Although geom_bar will do the counting for you, here I are passing a dataframe that has already been summarized in counts so I will use the **stat="identity"** parameter inside of geom_bar.

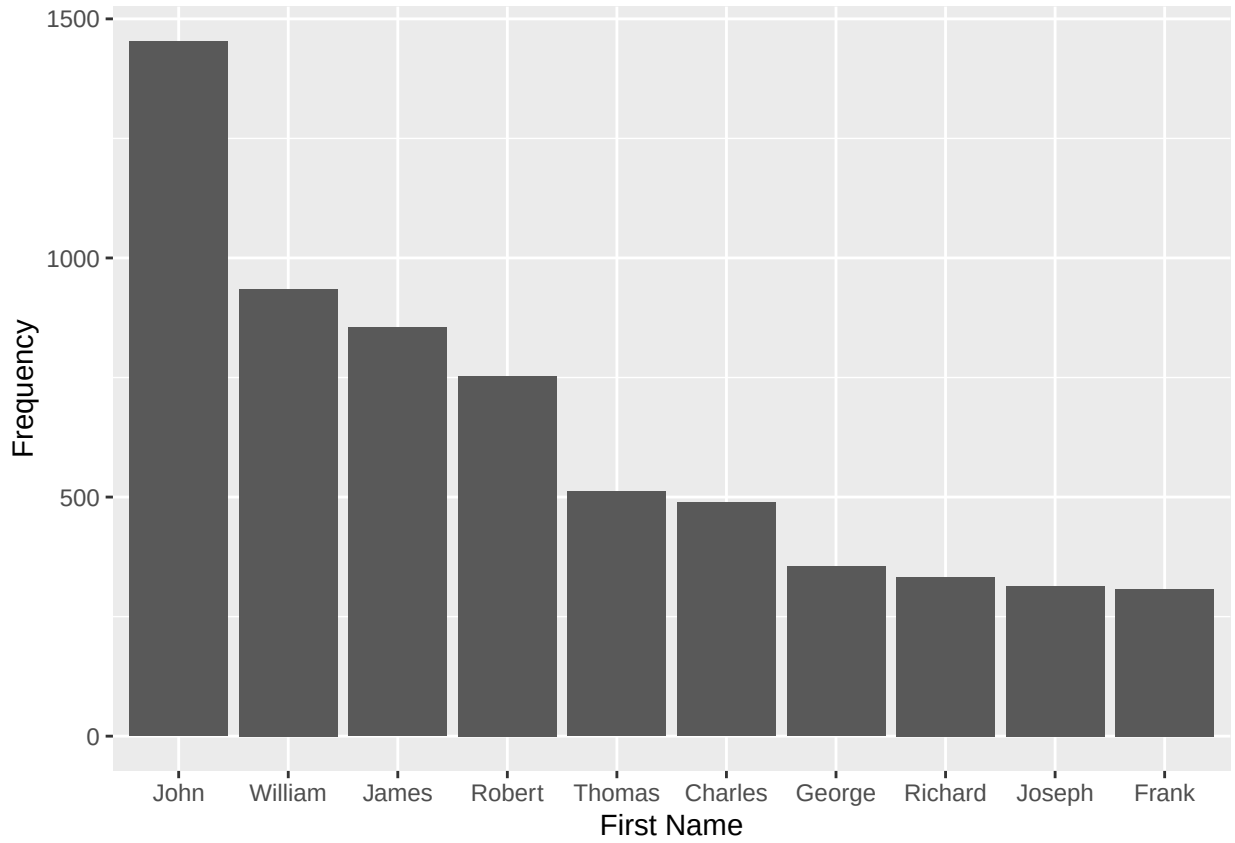
```
first_names[1:10,] %>%
  ggplot(aes(y = n, x = firstname)) +
  geom_bar(stat = "identity") +
  labs(x = "First Name", y = "Frequency")
```



Please note that x and y labels are added by using the **labs()** function. Unlike with the dplyr or tidyverse, ggplot requires + signs rather than a %>% to separate the statements. For all ggplots the aesthetic mapping **aes()** is vital as well as some form of geom statement. What is passed through the aesthetic determines what is on the x and y axis.

For example, by default ggplot will place the x axis into alphabetical order rather than take the order provided by the table. To fix this I can pass an additional parameter **scale_x_discrete()**:

```
level_order <- first_names[1:10, "firstname"]
first_names[1:10,] %>%
  ggplot(aes(y = n, x = firstname)) +
  geom_bar(stat = "identity") +
  labs(x = "First Name", y = "Frequency") +
  scale_x_discrete(limits = level_order$firstname)
```



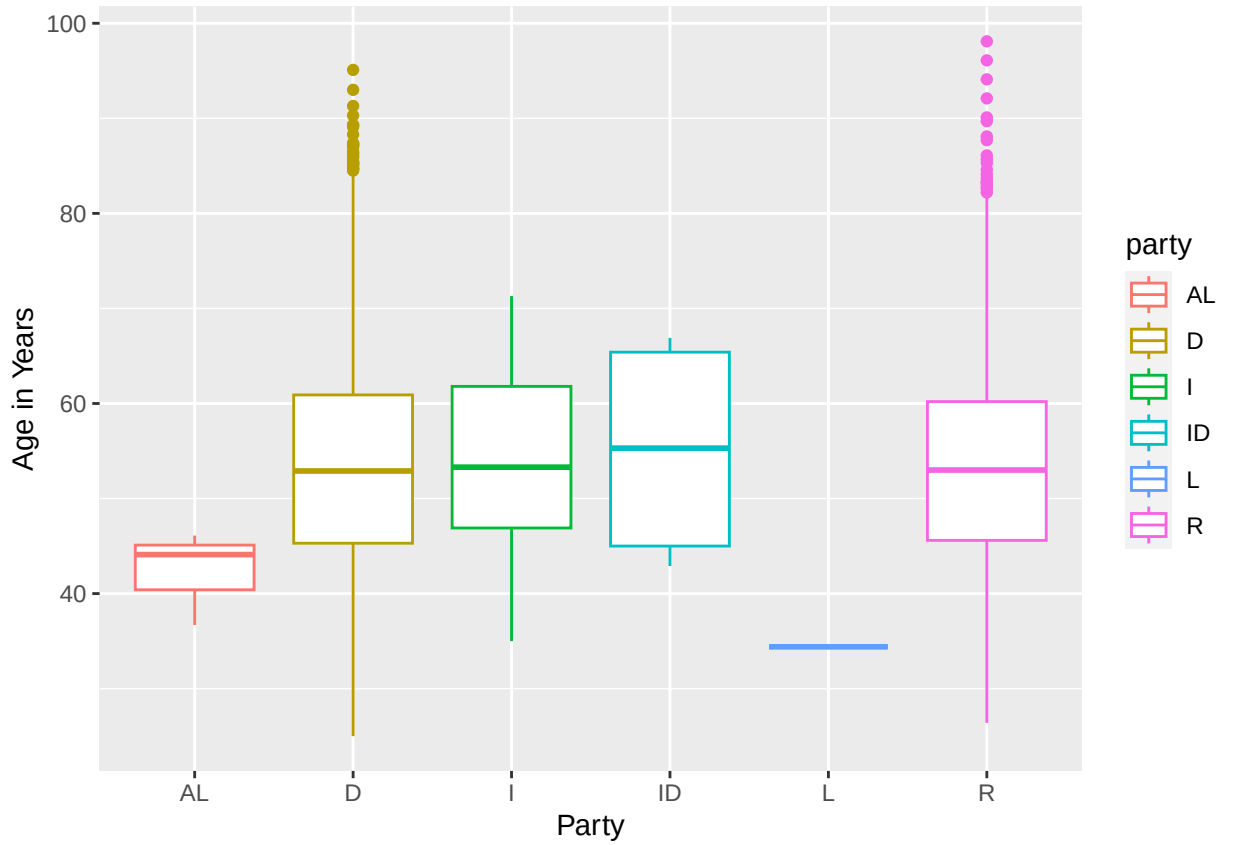
In the history of the US congress the frequency of the name John has outstripped other first name with William, James, and Robert not far behind.

Does the median age of a congress persons differ by political party?

In order to answer this question I will use a box plot on the raw dataset without any tidy manipulation.

Box plot with geom_boxplot. This will create a conventional box and whisker plot.

```
congress_age %>%
  ggplot(aes(x = party, y = age, colour = party)) +
  geom_boxplot() +
  labs(x = "Party", y = "Age in Years")
```

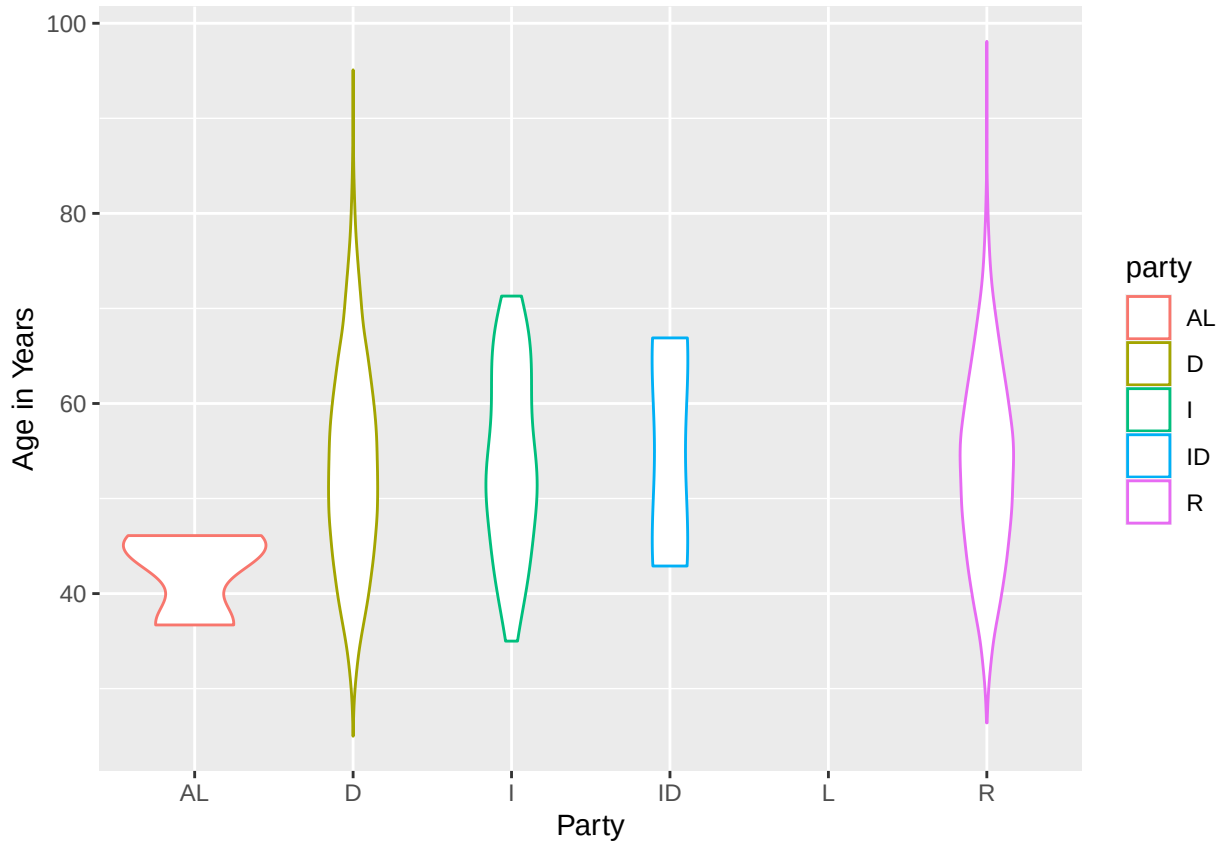


The median age is similar across all political parties, except the Libertarian(L) and the American Independent party(AL).

Violin plot with geom_violin. Violin plots are an alternative to box plots that add more information than a box plot in terms of the underlying distribution of the data. I will create a violin plot with the same data as above to demonstrate the additional information that can be obtained.

```
congress_age %>%
  ggplot(aes(x = party, y = age, colour = party)) +
  geom_violin() +
  labs(x = "Party", y = "Age in Years")
```

```
## Warning: Groups with fewer than two data points have been dropped.
```

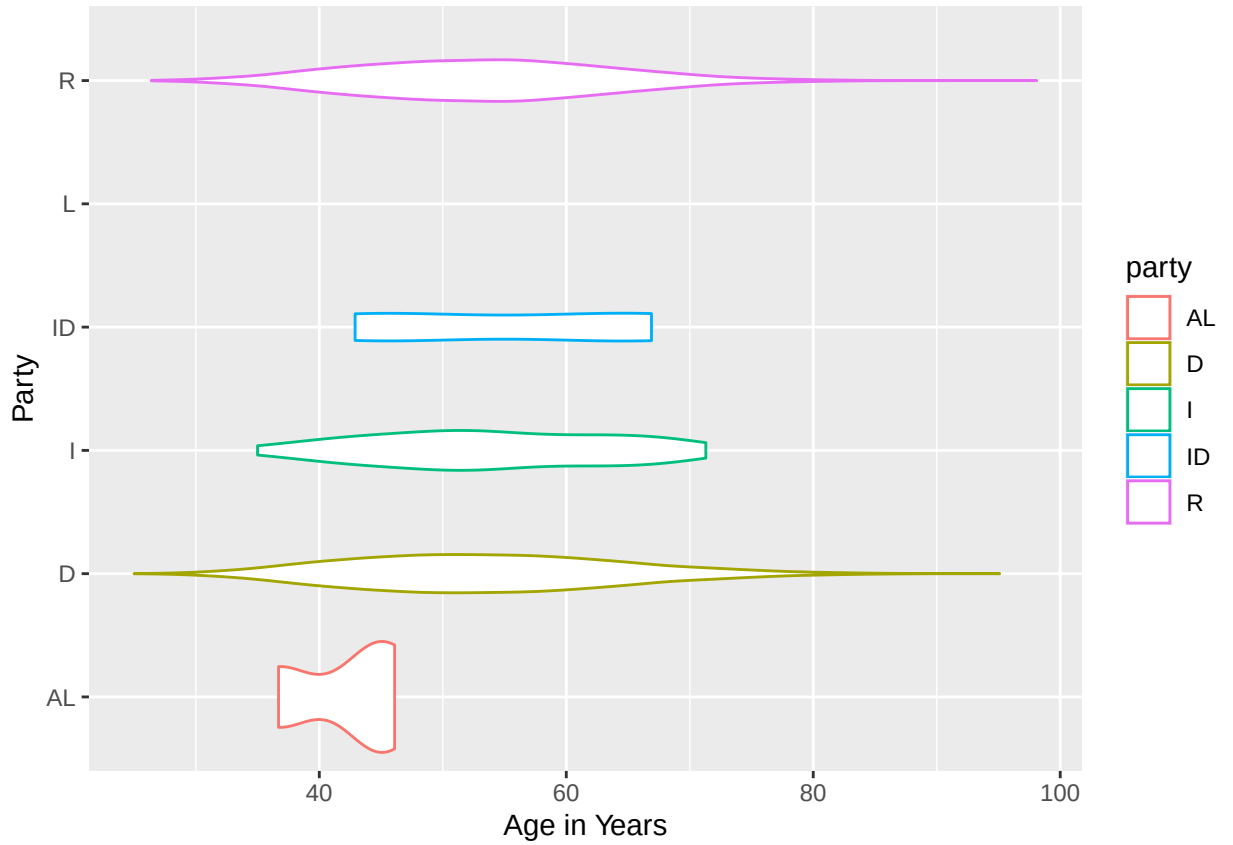


From this plot I can that the distribution of ages is most similar between Democrats(D) and Republicans(R). The Libertarian group is not shown because there was only one in the dataset.

What if we would like to visualize this plot horizontally instead? I can employ **coord_flip()** to flip the coordinates of the plot:

```
congress_age %>%
  ggplot(aes(x = party, y = age, colour = party)) +
  geom_violin() +
  labs(x = "Party", y = "Age in Years") +
  coord_flip()
```

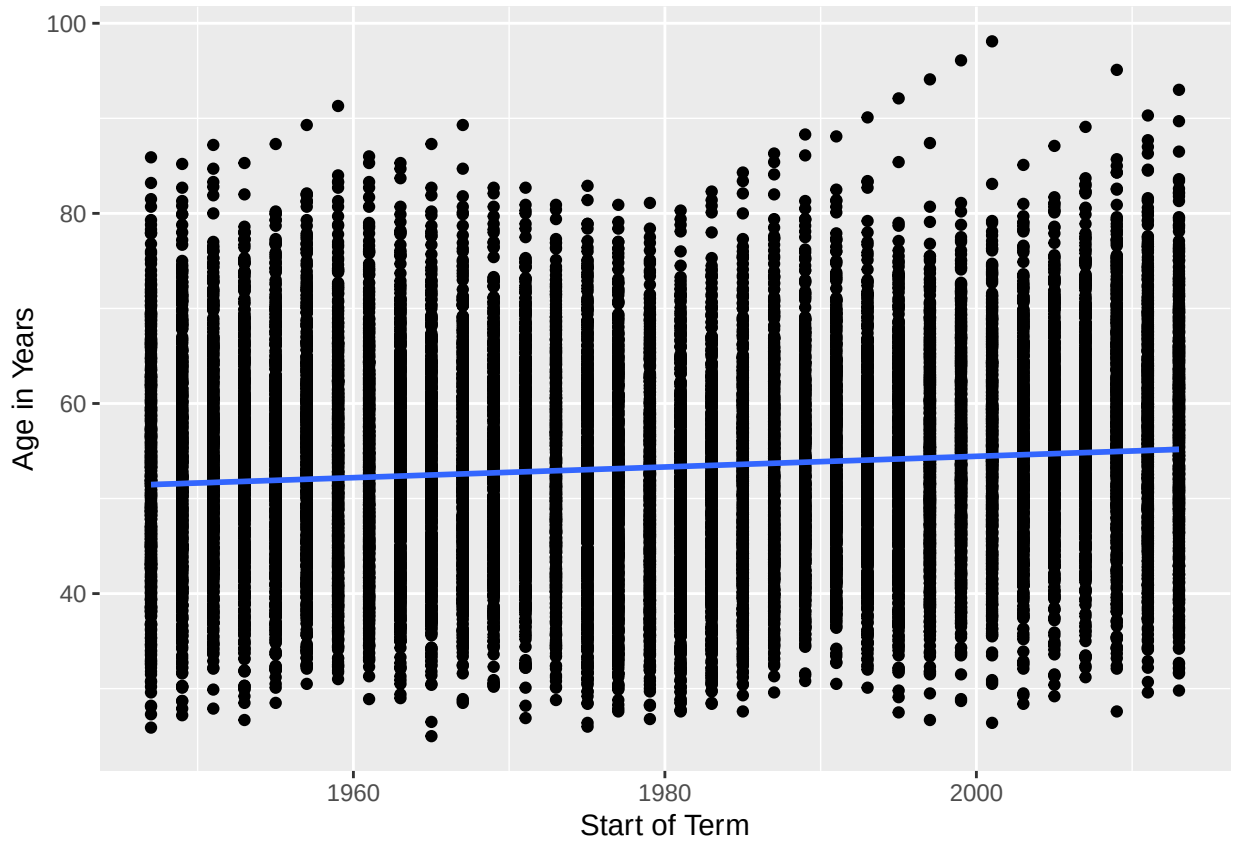
```
## Warning: Groups with fewer than two data points have been dropped.
```



Scaterplot with geom_point and geom_smooth. Scaterplots in ggplot are accomplished with **geom_point()** function and one can choose to add an an optional regression line to the data using either **geom_smooth()** or **geom_abline()**. However geom_abline requires that you have already calculated the line of best fit or another line before ploting. Geom_smooth is the ggplot replacement for baseR abline().

```
congress_age %>%
  ggplot(aes(x = termstart, y = age)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(x = "Start of Term", y = "Age in Years")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



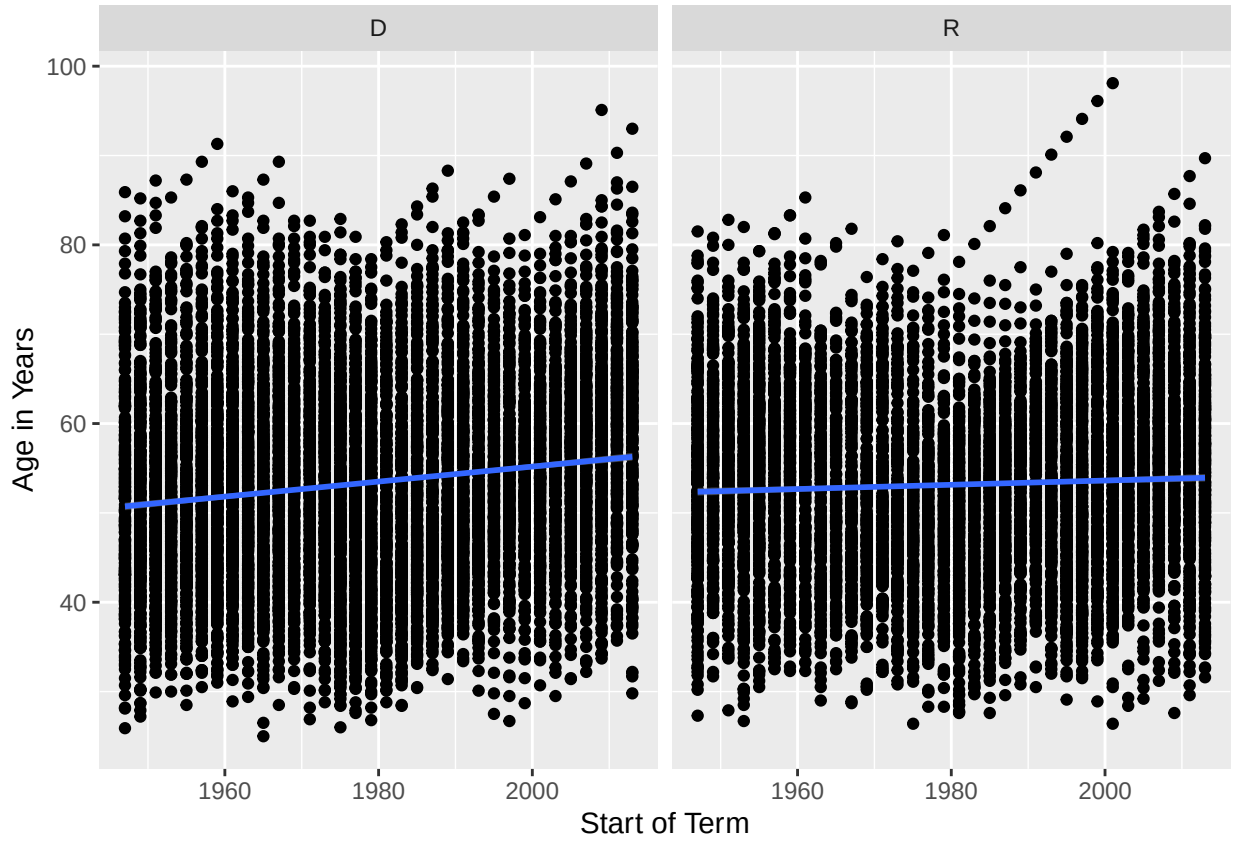
The scatterplot and regression line demonstrate that over time we are electing older people to congress.

Facet_wrap is one way to create multiple plot pannel within the same plot.

Lets use the above regression plot to test whether there is a difference between democrats and republicans at age at start of term. To create panels in a ggplot one can use either **facet_wrap()** or **facet_grid()**. Both functions perform similarly although facet_grid will create plots even for missing data where as facet_wrap will not. Here I used facet_wrap to demonstrate how the wraping works by adding "~z" where z is grouping variable.

```
congress_age %>%
  filter(party == "D" | party == "R") %>%
  ggplot(aes(x = termstart, y = age)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(x = "Start of Term", y = "Age in Years") +
  facet_wrap(~party)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Extend - Ross Boehme

While Alex did an excellent job of demonstrating the various graphs ggplot2 could create, he largely kept the original data intact. Therefore to extend his analysis, I'll demonstrate how the lubridate and dplyr packages from the tidyverse can be used to create new fields and edit existing ones. In addition, I'll graph those created fields using ggplot2 to expand on Alex's demonstrations.

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(dplyr)
library(ggplot2)
```

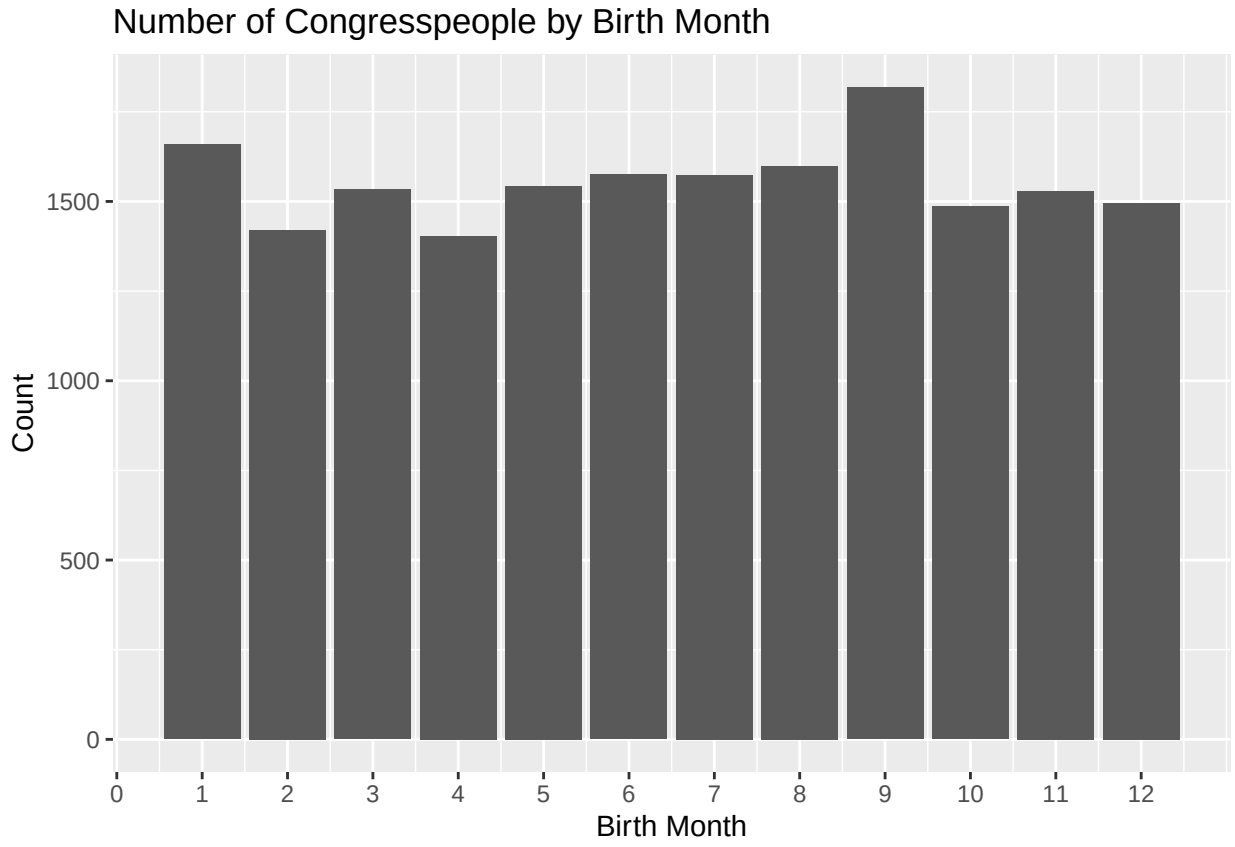
Pulling Birth Month from Birthday Field and Re-arranging Birthday Format With *Lubridate*

Lubridate contains many date-related functions. For example, lubridate::month can extract the month number from a YMD field like the below. I then graph with ggplot, showing that the most common birth month is the 9th (September).

In addition, I'll use the lubridate::format function to convert the birthday field from YMD to the more conventional DMY.

```
congress_age$birth_month <- lubridate::month(as.POSIXlt(congress_age$birthday, format="%Y/%m/%d"))

ggplot(data = congress_age, aes(x = birth_month)) +
  geom_bar() +
  labs(x = "Birth Month", y = "Count") +
  scale_x_continuous(breaks = seq(0, 12, by = 1)) +
  labs(title = "Number of Congresspeople by Birth Month")
```




```
congress_age <- congress_age %>%
  dplyr::mutate(birthday = format(birthday, "%d/%m/%Y"))
```

Generating Full State Name from State Abbreviation

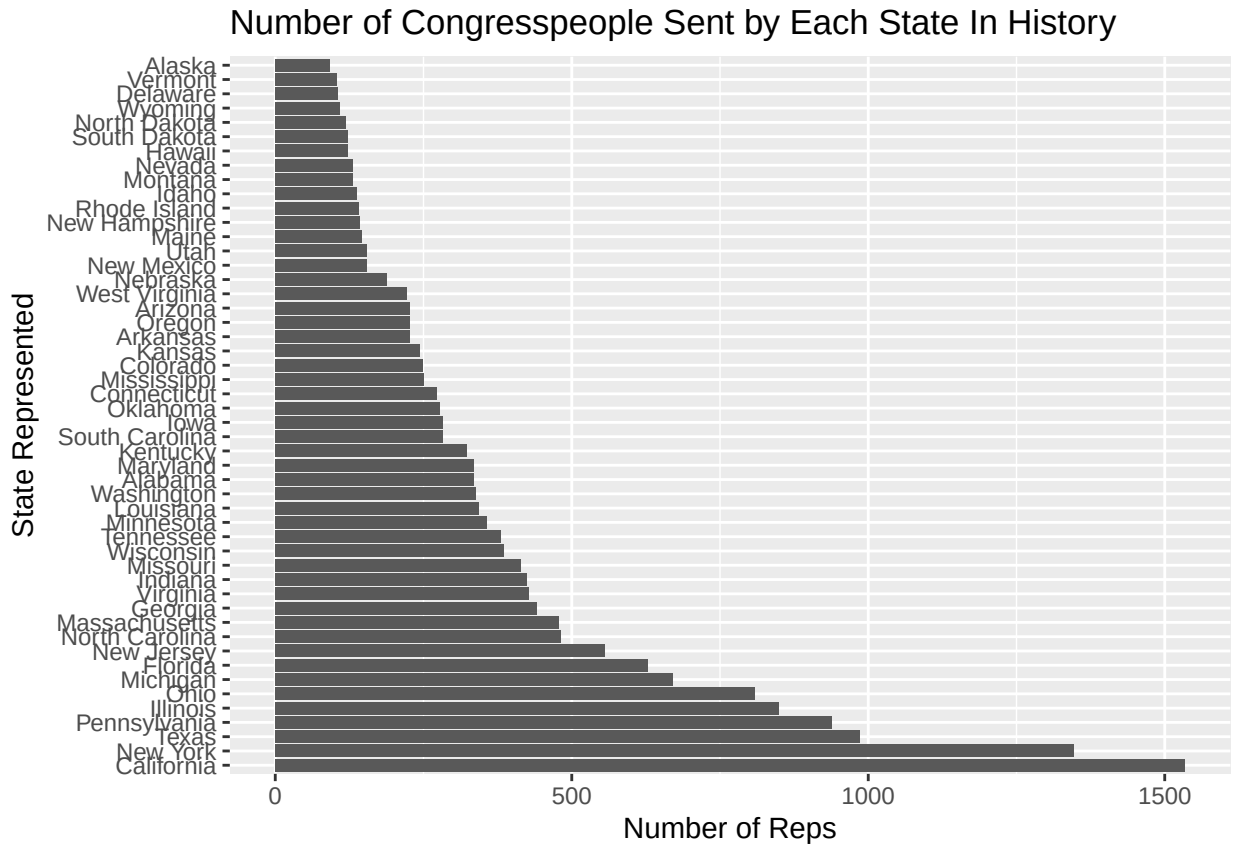
Second, I will generate the full state name from the state abbreviation. This leverages a built in R library ("state") which contains a dictionary, matching state abbreviations to full names.

I use the state.name function from state, combining it with dplyr's mutate function to create a new column, "state_name".

Alaska, Vermont, Delaware, Wyoming, and North Dakota are the five states who have sent the fewest congresspeople to D.C. This aligns with my domain knowledge: They are all less populous states, thus they would have fewer house of representatives members.

```
congress_age <- congress_age %>%
  dplyr::mutate(state_name = state.name[match(state, state.abb)])

ggplot(data = congress_age, aes(x = fct_infreq(state_name))) +
  geom_bar() +
  labs(x = "State Represented", y = "Number of Reps") +
  labs(title = "Number of Congresspeople Sent by Each State In History") +
  coord_flip()
```

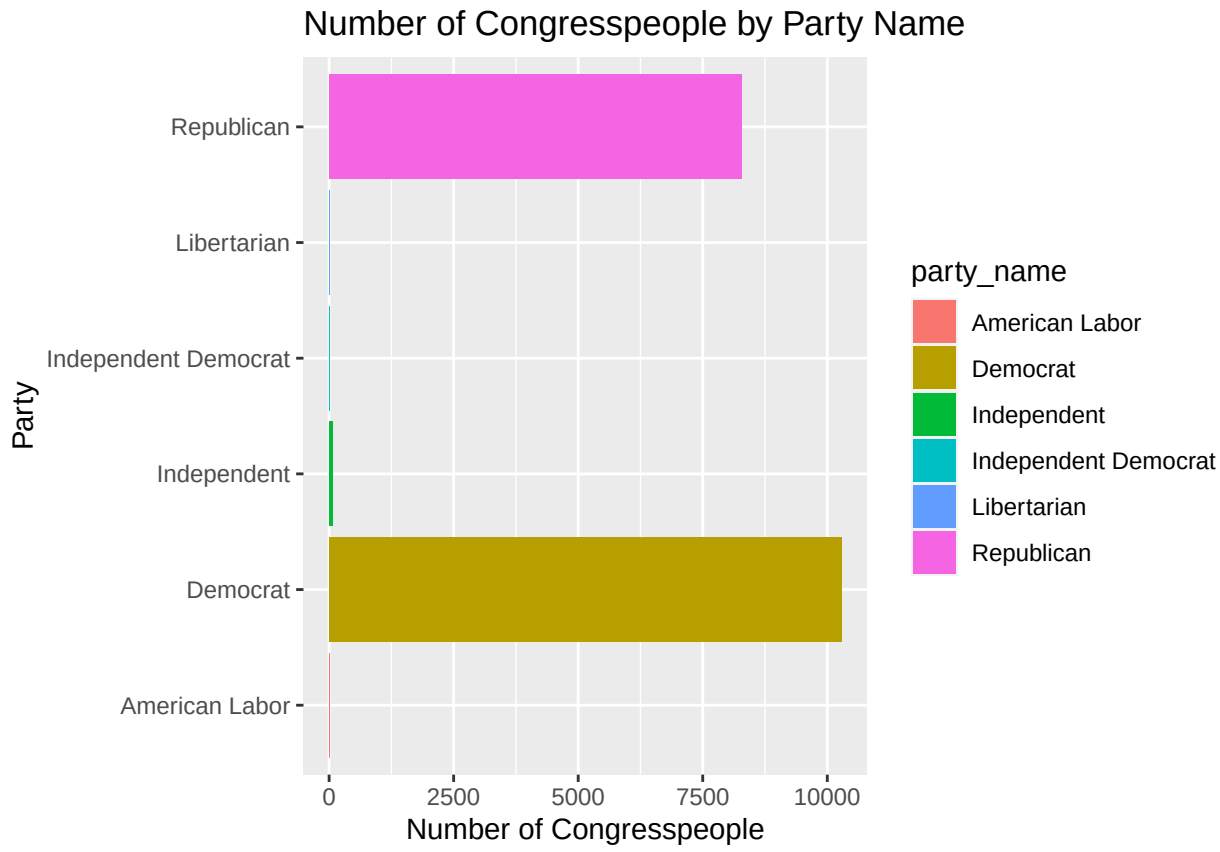


Mapping Full Party Name to Party Abbreviation

The current party names are abbreviated. For example: "R", "D", and "I". I'll use mutate and a small dictionary inside CASE WHEN to add the full party names (e.g."Republican", "Democrat", and "Independent"). Abbreviations' definitions come from the official Senate website.

```
congress_age <- congress_age %>%
  dplyr::mutate(party_name = case_when(
                  party == "D" ~ "Democrat",
                  party == "R" ~ "Republican",
                  party == "I" ~ "Independent",
                  party == "ID" ~ "Independent Democrat",
                  party == "AL" ~ "American Labor",
                  party == "L" ~ "Libertarian"
    ))
```

```
ggplot(data = congress_age, aes(x = party_name,fill=party_name)) +
  geom_bar() +
  labs(x = "Party", y = "Number of Congresspeople") +
  labs(title = "Number of Congresspeople by Party Name") +
  coord_flip()
```



Calculated Field Comparing Age to Average Age of Congressperson

For my last example using dplyr, I'll create a calculated field: Comparing the age of each congressperson (observation) to the the average age of all congresspeople of their chamber (House, Senate). To do so I'll use mutate once again.

Since this calculation is made on an observation level, it's difficult to graph. Therefore, I'll create a simple snippet showing the result, where Joseph Mansfield had the largest difference in age compared to his chamber (33.5 years), followed by Robert Doughton (30.8 years), and Adolph Sabath (28.3 years).

```
congress_age <- congress_age %>%
  group_by(chamber) %>%
  dplyr::mutate(age_vs_chamber_mean = abs(age - mean(age))) %>%
  ungroup()

snippet <- congress_age[c('firstname','lastname','birthday','chamber','age','age_vs_chamber_mean')]
head(snippet)
```

```
## # A tibble: 6 x 6
##   firstname lastname  birthday   chamber   age age_vs_chamber_mean
##   <chr>     <chr>     <chr>      <chr>   <dbl>                <dbl>
## 1 Joseph    Mansfield 09/02/1861 house    85.9                 33.5
## 2 Robert    Doughton  07/11/1863 house    83.2                 30.8
## 3 Adolph    Sabath    04/04/1866 house    80.7                 28.3
## 4 Charles   Eaton     29/03/1868 house    78.8                 26.4
## 5 William   Lewis     22/09/1868 house    78.3                 25.9
## 6 James     Gallagher 16/01/1869 house    78                   25.6
```