The Average and SD in R

The Basics: `mean()` and `sd()`

Calculating an average and standard deviation in R is straightforward. The `mean()` function calculates the average and the `sd()` function calculates the standard deviation. However, both of these functions are designed to work with vectors, not data frames, and so we must remember to use the `data$variable` syntax.

To see how this works, let's load the `nominate.rds` and remind ourselves what variables we're working with.

```
# load data
nominate <- readRDS("data/nominate.rds")
# note: make sure the file 'nominate.rds' is in the 'data' subdirectory
#   and your working directory is set appropriately.

# quick look at data
tibble::glimpse(nominate)
```

```
## Observations: 6,159
## Variables: 6
## $ congress               <int> 100, 100, 100, 100, 100, 100, 100, 100,...
## $ state                  <fctr> ALABAMA, ALABAMA, ALABAMA, ALABAMA, AL...
## $ congressional_district <int> 1, 2, 3, 4, 5, 6, 7, 1, 1, 2, 3, 4, 5, ...
## $ party                  <fctr> Republican, Republican, Democrat, Demo...
## $ name                   <fctr> CALLAHAN, DICKINSON, NICHOLS  B, BEVIL...
## $ ideology_score         <dbl> 0.358, 0.349, -0.039, -0.203, -0.152, -...
```

Then let's calculate the average and standard deviation of the `ideology_score` variable.

```
# calculate average
mean(nominate$ideology_score)
```

```
## [1] 0.08695941
```

```
# calculate stardard deviation
sd(nominate$ideology_score)
```

```
## [1] 0.4749944
```

Importantly, the SD that R calculates with `sd()` is not quite the same as the SD in the textbook. See Section 7 on p. 74 for more details. In short, the textbook uses the formula $\sqrt{\frac{\text{sum of squared deviations from mean}}{\text{number of observations}}}$. R uses the forumula $\sqrt{\frac{\text{sum of squared deviations from mean}}{\text{number of observations - 1}}}$.

We can see this below:

```
# create example numeric vector
x <- c(6, 3, 2, 7)

# calculate number of observations
N <- length(x)  # counts the number of elements in x

# using FPP formula
deviations <- x - mean(x)
s <- deviations^2  # the "s" in r.m.s.
m <- sum(s)/N  # the mean; the "m" in r.m.s.
```

```r
sd <- sqrt(m)  # the sqare root, the "r" in r.m.s.
print(sd)  # this is the SD
```

```
## [1] 2.061553
```

```r
# using R's formula
deviations <- x - mean(x)  # same as above
s <- deviations^2  # same as above
m_plus <- sum(s)/(N - 1)  # divide by N - 1 rather than N
sd_plus <- sqrt(m_plus)  # same as above
print(sd_plus)  # this is the SD+
```

```
## [1] 2.380476
```

```r
# compute using sd()
sd(x)  # same as R's formula above
```

```
## [1] 2.380476
```

```r
# correct using forula on p. 75

cf <- sqrt((N - 1)/N)  # conversion factor
cf*sd(x)  # same as using FFP formula above
```

```
## [1] 2.061553
```

An Aside on Data Frames

Remember that `mean(ideology_score)` does not work. It does not work because `ideology_score` is stored inside a data frame. Therefore, as with all varables in data sets, we need to use the `data$variable` syntax. If you find this confusing, you should think of a data frame as a box of vectors. The vectors in the box can be different types (e.g., numeric, character), but they all have the same length (i.e., number of elements). When you ask R to calculate `mean(ideology_score)`, it looks around (in the "environment") for an object called `ideology_score`. However, it will not find `ideology_score`, because `ideology_score` is hidden in the box (i.e., the data frame). In order to force R to look inside the box, you have to tell it what box to look inside. `nominate$ideology_score`, then, means "the `ideology_score` vector in the `nominate` data set."

It's probably more interesting to look at an individual Congress, though, so let's subset the data and take the average and standard deviation of only the 100th Congress.

```r
# average and sd for the 100th congress
nominate100 <- subset(nominate, congress == 100)  # create a new data frame w/ only the 100th congress
mean(nominate100$ideology_score)  # calculate the average
```

```
## [1] -0.04942369
```

```r
sd(nominate100$ideology_score)  # calculate the standard deviation
```

```
## [1] 0.343292
```

Within an individual Congress, we can calculate the average and standard deviation for Republicans and Democrats separately.

```r
# calculate average and sd for republicans in 100th congress
nominate100rep <- subset(nominate, (congress == 100) & (party == "Republican")) # subset to 100th congr
mean(nominate100rep$ideology_score)  # calculate average
```

```
## [1] 0.3141508
```

```r
sd(nominate100rep$ideology_score)  # calculate sd
```

```
## [1] 0.1686674
```

```r
# calculate average and sd for democrats in 100th congress
nominate100dem <- subset(nominate, (congress == 100) & (party == "Democrat")) # subset to 100th congres
mean(nominate100dem$ideology_score)  # calculate average
```

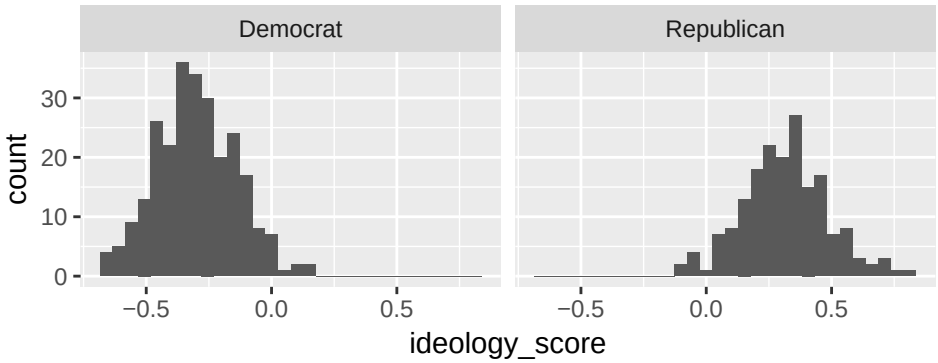
```
## [1] -0.2997308
```

```r
sd(nominate100dem$ideology_score)  # calculate sd
```

```
## [1] 0.1596674
```

For the 100th Congress, you can see that, on average, Democrats are more liberal than Republicans. Within the parties, you can see a similar spread in the distributions. The histogram below shows the same pattern.

```r
# plot separate histograms for democrats and republicans for the 100th congress
library(ggplot2)  #
ggplot(nominate100, aes(x = ideology_score)) +  # data set: nominate100, aesthetics: x = ideology_score
    geom_histogram() +  # geometry: histogram
    facet_wrap(~ party)  # facets: create plots by party
```



In general, this is how we'll want to use averages and standard deviations–calculating for distinct subgroups (e.g., Republicans and Democrats) and comparing. Remember that the last step of the scientific method I discribed is "comparisons."

Moving Further: `summarize()` and `group_by()`

The `dplyr` package offers a two functions that, when combined, allow us to do this quite effectively.

`group_by()`

First, the `group_by()` function allows us to point out the interesting groups in our data frame. These are the groups that *you*, the researcher, find interesting. In our case, the variable `party` defines the groups we are interested in. The first argument to `group_by()` is the data set we would like to group. The second argument to `group_by()` is the variable that we would like to group by. So `group_by(nominate, party)`

means "group the `nominate` data set by `party`." One group will be Republicans, another group will be Independents, and the remaining group will be Democrats.

```
# load packages
library(dplyr)  # for summarize() and group_by()

# group the data set
nominate100_grouped <- group_by(nominate100, party)
```

`summarize()`

Grouped data frames are especially useful when combined with the `summarize()`. The `summarize()` function calculates statistics *by group*.

`summarize()` creates a new data frame with several variables. There will be one variable for each grouping variable. In the current example, we just have one: `party`. It will also create a variable for each statistic.

The first argument to the `summarize()` function is the grouped data set. The remaining arguments are explicly named. The name of the remaining arguments will be the variable names in the new data set. The arguments are the calculations that you'd like to apply to each group.

The following example should make the logic clear.

```
# calculate the mean and standard deviation for each party in nominate100
summarize(nominate100_grouped,
          mean_ideology = mean(ideology_score),
          sd_ideology = sd(ideology_score))
```

```
## # A tibble: 2 × 3
##        party mean_ideology sd_ideology
##       <fctr>         <dbl>       <dbl>
## 1   Democrat    -0.2997308   0.1596674
## 2 Republican     0.3141508   0.1686674
```

Multiple Grouping Factors

In the previous example, we grouped only by `party` and used data from only the 100th Congress. But with `group_by()` and `summarize()` we can quickly do this for each party *and* each Congress.

```
# calculate the mean and standard deviation for each party and congress in nominate
nominate_grouped <- group_by(nominate, party, congress)  # group data by party *and* congress
summarize(nominate_grouped,
          mean_ideology = mean(ideology_score), # calculate the mean for each group
          sd_ideology = sd(ideology_score))  # calculate the standard deviation for each group
```

```
## Source: local data frame [28 x 4]
## Groups: party [?]
##
##       party congress mean_ideology sd_ideology
##      <fctr>    <int>         <dbl>       <dbl>
## 1  Democrat      100    -0.2997308   0.1596674
## 2  Democrat      101    -0.3024198   0.1619839
## 3  Democrat      102    -0.3018587   0.1630104
## 4  Democrat      103    -0.3138217   0.1566859
## 5  Democrat      104    -0.3383846   0.1479384
## 6  Democrat      105    -0.3455714   0.1364599
```

```
## 7  Democrat      106    -0.3444836   0.1372905
## 8  Democrat      107    -0.3495425   0.1331503
## 9  Democrat      108    -0.3489375   0.1294971
## 10 Democrat      109    -0.3591337   0.1238202
## # ... with 18 more rows
```

This new data frame is much bigger, so there is a lot to look at. Here are some comparisons you might make:

1. The average ideology for Republicans across Congresses. Are Republicans getting more conservative, more moderate, or staying about the same?
2. The average ideology for Democrats across Congresses. Are Democrats getting more liberal, more moderate, or staying about the same?
3. The standard deviation of ideology for Republicans across Congresses. Are Republicans becoming more ideological similar, dissimilar, or staying about the same?
4. The standard deviation of ideology for Democrats across Congresses. Are Democrats becoming more ideological similar, dissimilar, or staying about the same?

Based on the output above, what would you say?

Plotting the Average and SD

It turns out that the table created by `summarize()` is a data frame. If we store this data frame as an object (say, `smry`), then it's easy to plot these data using `ggplot2`.

```
# store summarize() data frame as an object
smry <- summarize(nominate_grouped,
                  mean_ideology = mean(ideology_score),
                  sd_ideology = sd(ideology_score))
```

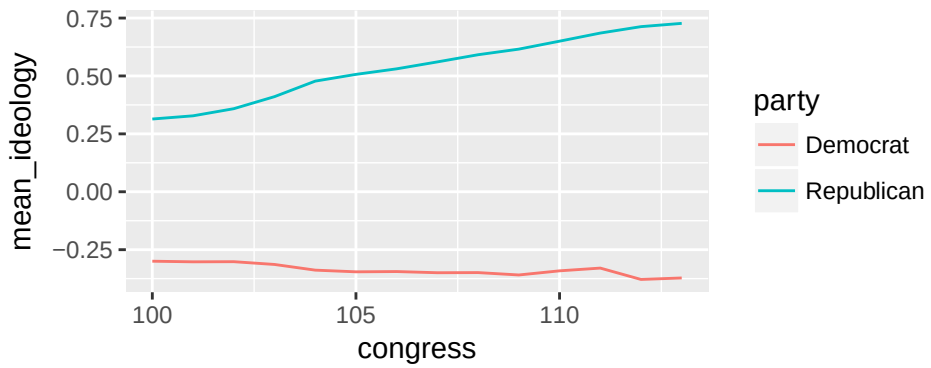
I have in mind a plot where the x-axis represents the Congress (i.e., 100, 101, etc.) and the y-axis represents the average (or standard deviation). I want one line for each party that are different colors.

To create this plot, we'll use ggplot2. This means that we'll take the same approach of supplying the data and aesthetics and then adding the geometry.

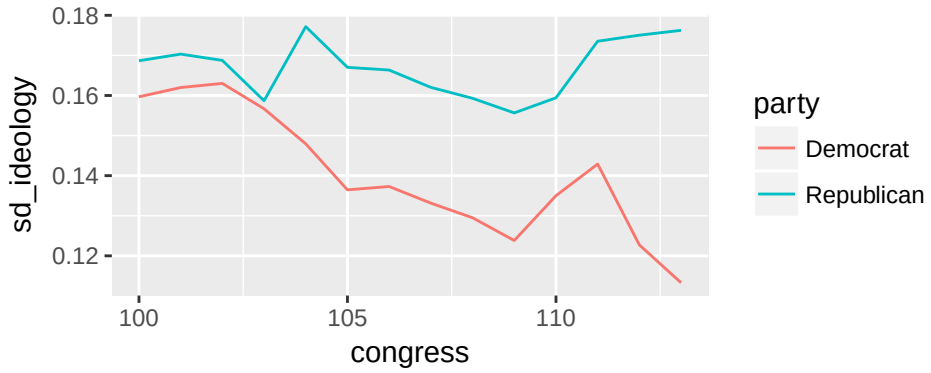
In this case, we want to use the data frame `smry`. The aesthetics we want are `x = congress`, `y = mean_ideology` (or `y = sd_ideology`), and `color = party`. The plot I have in mind is a line plot, so we'll add `geom_line()`, which simply connects the x-y values with a line.

```
# load packages
library(ggplot2)

# line plot of average
ggplot(smry, aes(x = congress, y = mean_ideology, color = party)) +
  geom_line()
```



```r
# line plot of standard deviation
ggplot(smry, aes(x = congress, y = sd_ideology, color = party)) +
  geom_line()
```



This plot of averages and standard deviation makes it clear that polarization is happening for two reasons.

1. Republics are moving futher to the right, on average.
2. Democrats are becoming more ideologically similar.

Both of these tendencies lead to less overlap in the distributions.

If we wanted, we could refine these ggplots in the usual ways. For completeness, I've include the entire script necessary to re-create the figure.

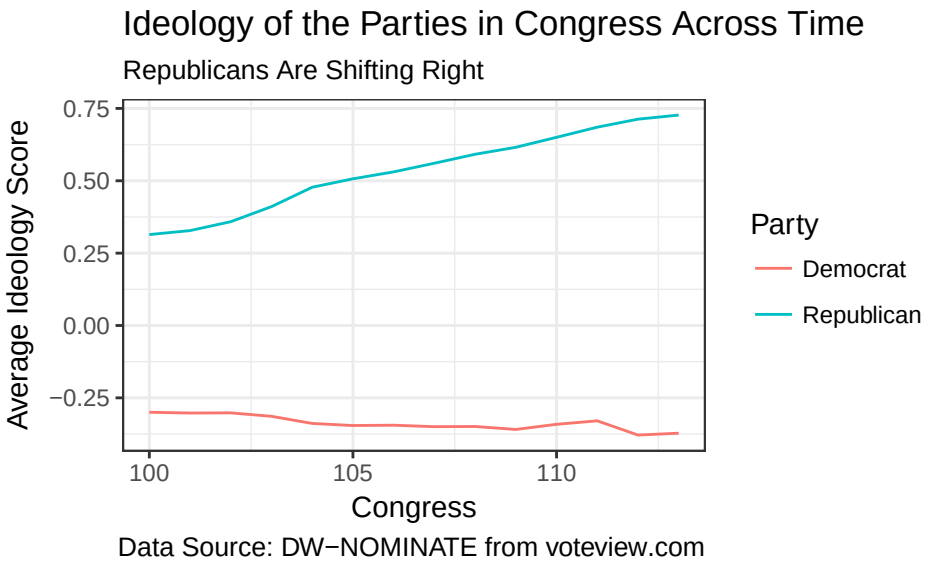
```r
# load packages
library(dplyr)
library(ggplot2)

# load data
nominate <- readRDS("data/nominate.rds")
# note: make sure the file 'nominate.rds' is in the 'data' subdirectory
#   and your working directory is set appropriately.

# group the data frame
nominate_grouped <- group_by(nominate, party, congress)  # group data by party *and* congress

# calculate average and sd for each group
smry <- summarize(nominate_grouped,
                  mean_ideology = mean(ideology_score),
                  sd_ideology = sd(ideology_score))

# line plot of average
ggplot(smry, aes(x = congress, y = mean_ideology, color = party)) +
  geom_line() +
  labs(x = "Congress",
       y = "Average Ideology Score",
       color = "Party",
       title = "Ideology of the Parties in Congress Across Time",
       subtitle = "Republicans Are Shifting Right",
       caption = "Data Source: DW-NOMINATE from voteview.com") +
  theme_bw()
```



Review Exercises

1. In these notes, we've introduced serveral new functions: `mean()`, `sd()`, `group_by()`, `summarize()`, and `geom_line()`. For each of these functions, answer the following questions:
   a. What does the function do?
   b. What arguments does the function usually take? Note that functions take many arguments, but there are a few that we ususually use.
   c. When would you use the function?
   d. What package is the function in?
2. Suppose I create the object `x` using the code `x <- c(1, 2, NA, 4)`. If I run `mean(x)` what will I get? What about `sd(x)`? What is one way to solve the problem? Is it a good solution?
3. Sometimes we prefer to use the median as the measure of location of a distribution rather than the average. We can easily calculate the median in R using `median()`, which works similarly to `mean()`. Re-create the final figure above using the median rather than the mean.
4. Similarly, we sometimes use the inter-quartile range (IQR, the dfference between the 75th and 25th percentile) as a measure of dispersion rather than the SD. Re-create the final figure above using the IQR rather than the mean.
5. Load the `state-legislators` data. You choose the filetype. Group the data set by year, state, and party. Using that grouped data, create a line plot, much like the final figure above, but with faceting by state. (Hint: This will create one big figure with 50 line plots–one for each state.) What interesting patterns do you see?