Histograms

Histograms are a crucial tool for understanding the distribution of a single variable. In order for a histogram to work well, the variable of interest should take on enough values to require "binning" or lumping several values into the same column of the figure.

DW-NOMINATE Data

To see how histograms work, let's work with estimates of the ideology of the U.S. House of Representatives. These data are called DW-NOMINATE Scores and are available at voteview.com. The cleaned data `nominate.csv` includes the 100th through the 113th Congress and contains a variable called `ideology_score` that indicates a representatives ideology for a particular Congress. Note that these data are at the legislator-Congress level, so that each row in the data set is an individual legislator in a particular Congress. Smaller values (i.e., more negative) indicate that a representative is more liberal and larger values (i.e., more positive) indicate that a representative is more conservative. For example, the Ron Paul is one of the most conservative representatives in the data set and has a score of about 1.3 (it varies across Congresses). Dennis Kucinich is among the most liberal and has a score of about -0.7.

To get started, let's load the cleaned data.

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

We can use the `glimpse()` function in the `tibble` package to check that the data loaded properly and get a quick overview of the data. It shows use the variable names, let's us know what type of variable we're working with (e.g., character, integer, double), and shows us the first few values.

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

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

I've given the variables very descriptive names, but if you need more information, you can find the codebook at DW-NOMINATE's data page.

Histograms

Now let's try a histogram. To do this, we'll need to load the R package `ggplot2`. Though `ggplot2` is a complex package, we'll get some sense of how it works this semester. It is well-documented online, so feel free to read about it as much as you want or need on your own.

There are three critical components to each plot created using `ggplot2`, which we'll refer to as a 'ggplot.'

1. the data: a formal data frame in R.
2. the aesthetic: the relationship between the variables, specified in the call to `ggplot`. You can specify how the data points vary in space, color, size, shape, etc.
3. the geometry: the type of plot.

There are other components, such as scales, facets, and coordinate systems, but these the three critical components usually provide what we need.

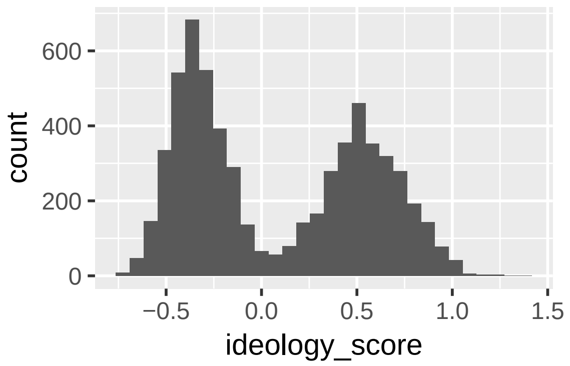
Let's go ahead and load `ggplot2` because we'll be using it a lot and create a histogram of the variable `ideology_score`.

Remember that for each ggplot, we have (at least) three elements.

1. data: the first argument to `ggplot()`. Because the variable we want to plot, `ideology_score` is contained in the data frame `nominate`, we use `nominate` as the first argument.
2. aesthetic: the second argument to `ggplot()`. Because we want to create a histogram, we want value of `ideology_score` to correspond to the location along the x-axis. To create this correspondence, we use `aes(x = ideology_score)` as the second argument.
3. geometry: added to `ggplot()`. Because we want to create a histogram, we want to use `geom_histogram()`. We simply add, using `+`, the desired geometries to the plot.

```r
# load package
library(ggplot2)

# specify data and aesthetic, then add the geometry
ggplot(nominate, aes(x = ideology_score)) +  geom_histogram()
```



This histogram makes sense. We have a grouping of Democrats (presumably) on the left and a grouping of Republicans (again, presumably) on the right. If you wanted, you could probably come up with a model that explains why we expect few politicians in the center.

There are a few subtle points that I should emphasize about the line `ggplot(nominate, aes(x = ideology_score)) +  geom_histogram()`.

1. The first argument supplied to `ggplot()` is the data. The second argument is the aesthetics.
2. The second argument, `aes(...)` is itself a function. This function just creates the "aesthetic mapping," a link between the variables in the data set and space, color, size, shape, etc, in the chart.
3. `geom_histogram()` is also a function, but it is added to the plot using `+`. We'll see several other functions that can be added to the plot.

Review Exercises

1. List and describe the three critical components of a ggplot.

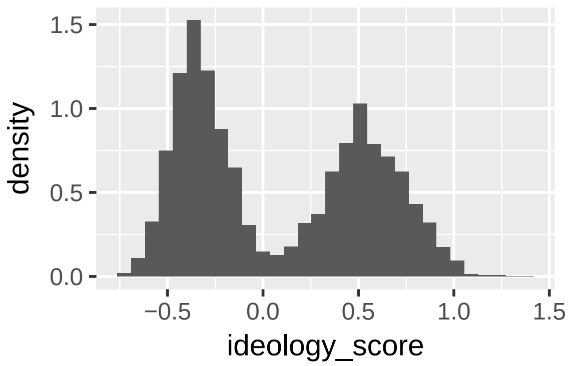
2. In the following snippet of code, label the data, the aesthetics, and the geometry: `ggplot(nominate, aes(x = ideology_score)) +  geom_histogram()`.
3. In this section, we've seen three new functions: `ggplot()`, `aes()`, and `geom_histogram()`. Describe what each one does and how to use it.

But it uses counts! What about density?

Our textbook uses density on the y-axis, not counts. This is important, because if the bin widths of the histograms are not equal, then counts would create a highly misleading histogram. However, if the bin widths are equal, then counts and densities produce identical histograms. Also, we usually use equal bin widths in practice. Therefore, as long as bin widths are equal, it doesn't matter whether we use counts or densities.

You may have noticed that `geom_hist()` puts counts on the y-axis by default. If we want to change that default behavior, we can add `y = ..density..` to our aesthetics. `geom_hist()` assumes you want `y = ..count..` by default. In practice (i.e., with equal bin widths), it doesn't matter. Let's just go with counts when we're making histograms with ggplot2.

```
# specify data and aesthetic, then add the geometry
ggplot(nominate, aes(x = ideology_score, y = ..density..)) +  geom_histogram()
```



Subsets of Data Frames

But these ideology scores that we're plotting go across 14 different Congresses (the 100th through the 113th). Ideally, we'd plot just one at a time. To do that, rather than give the whole data set to the `ggplot()`, we'll give a subset.

To create a subset of the original data set that only contains the 100th Congress, we'll use the `subset()` function. For this function, the first argument is the data frame to be subsetted and the second arguments is a logical statement that identifies the cases to be kept. If the logical statement is `TRUE`, then the case is kept. If the logical statement is `FALSE`, then the case is dropped.

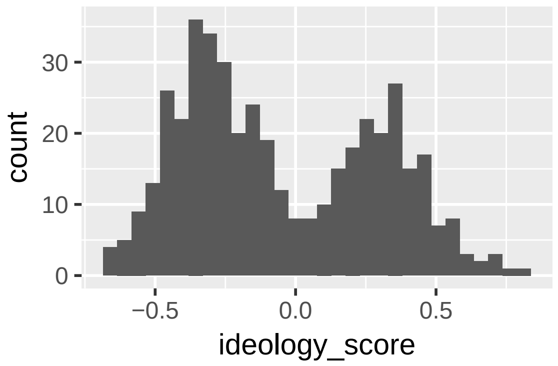
```
# subset data to only 100th congress
nominate100 <- subset(nominate, congress == 100)
```

Literally, this means…

1. find a subset of the data frame `nominate` where the variable `congress` equals 100 (remember that we need double equals signs to test for equality), and
2. assign that new (smaller) data frame to `nominate100`.

Now use just use the same code as before, but change the data frame. Rather than supplying `nominate` as the first argument, we'll supply `nominate100`.

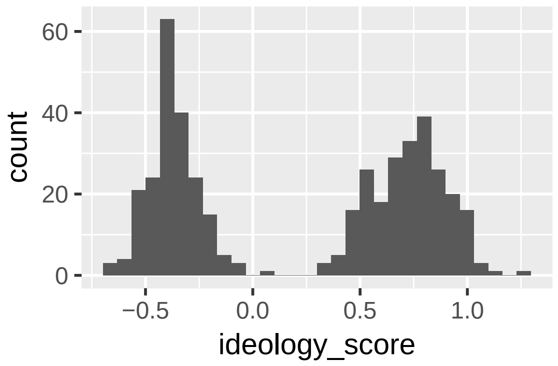
```r
# create histogram for 100th congress
ggplot(nominate100, aes(x = ideology_score)) +  geom_histogram()
```



Now let's repeat that process, but for the 113th Congress.

```r
# subset data to only 113th congress
nominate113 <- subset(nominate, congress == 113)

# create histogram
ggplot(nominate113, aes(x = ideology_score)) +  geom_histogram()
```



Review Exercises

1. Explain what the `subset()` function does and how to use it. What are the first and second arguments?
2. Study the histograms of the ideology scores in the 100th and 113th Congresses. In what ways are they similar? In what ways are they different? Why might that be?

Faceting

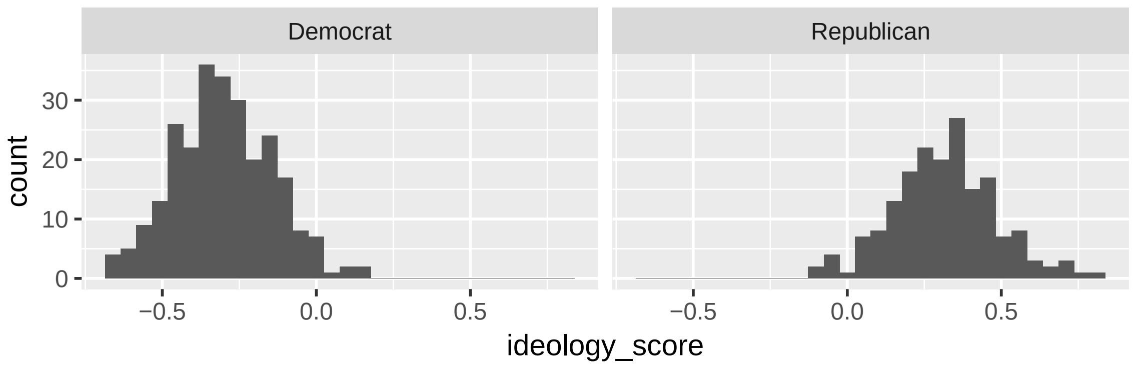
Let's investigate this divergence in the parties more carefully.

Is the left hump actually Democrats? Is the right hump actually Republicans? To to that, we might apply a facet to the histogram. A facet simply breaks the data frame into subsets and draws one histogram per subset.

Create a facet by adding the function `facet_wrap()` to the plot. You'll have to specify the faceting variable as an argument to the `facet_wrap()` function. In this case, we'll do it by `party`, adding `+ facet_wrap(~`

`party)` (i.e., "create a facet by party") to the ggplot we've been using. Notice the tilde (`~`) in front of the faceting variable.

```
# build histogram
ggplot(nominate100, aes(x = ideology_score)) +
  geom_histogram() +
  facet_wrap(~ party)
```



This is quite helpful. We see that our intuition was correct. The two humps in the previous histograms were due to homogeneity within parties and heterogeneity between parties.

Review Exercises

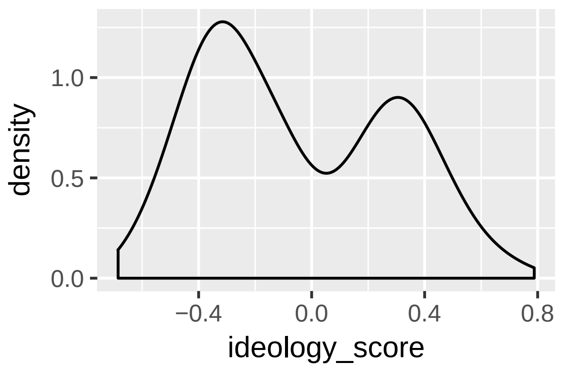
1. What does a facet do to a plot?
2. Explain what the `facet_wrap()` function does and how to use it.
3. Suppose I added `+ facet_wrap(party)`. Would that work? What is missing?
4. Suppose you `+ facet_wrap( ~ state)` to `ggplot(nominate100, aes(x = ideology_score)) + geom_histogram()`. What would happen?

Density Plots

In the book, the authors sometimes use a rough sketch of a histogram using a smooth curve rather than a complete histograms with vertical bars. For out purposes, a density plot is a smooth curve that approximates a histogram.

We can easily create a density plot rather than a histogram by using `geom_density()` as the geometry rather than `geom_histogram()`.

```
# create density plot
ggplot(nominate100, aes(x = ideology_score)) +
  geom_density()
```



Review Exercises

1. How is a density plot similar to and different from a histogram?
2. What does the function `geom_density()` do and how do you use it?

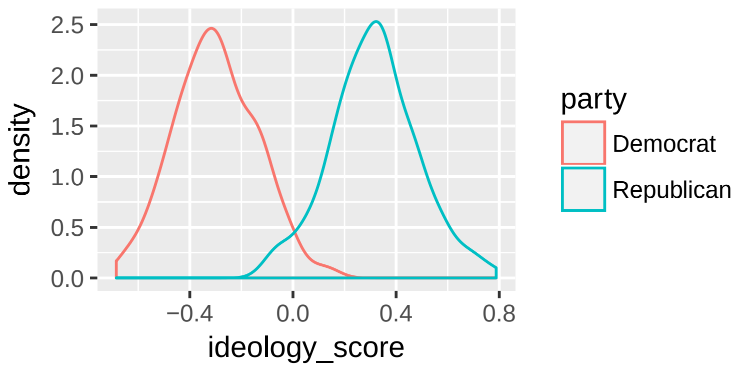
Color and Fill

Color

A density plot has the advantage it uses less ink. Because of this, we could use color rather than faceting to distinguish Republicans and Democrats.

To use color to distinguish Republicans and Democrats, we simply add `color = party` to the aesthetics, this will draw two different colored lines for the density for Republicans and the density for Democrats.

```
# build density plot
ggplot(nominate100,
       aes(x = ideology_score, color = party)) +
  geom_density()
```

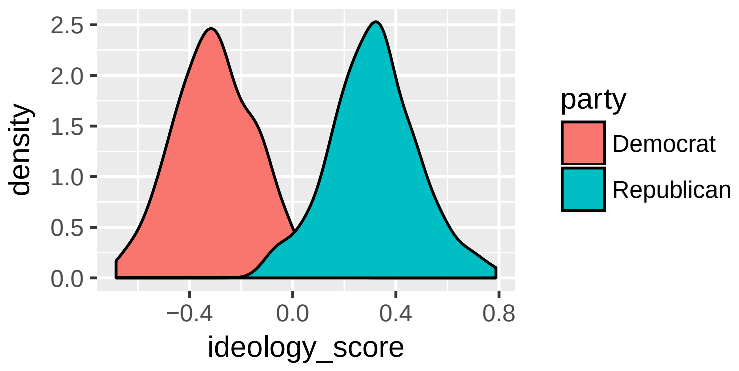


This look really good! Since we are most interested in the overlap between the parties, I think this density plot makes more sense for us, so let's stick with it.

Fill

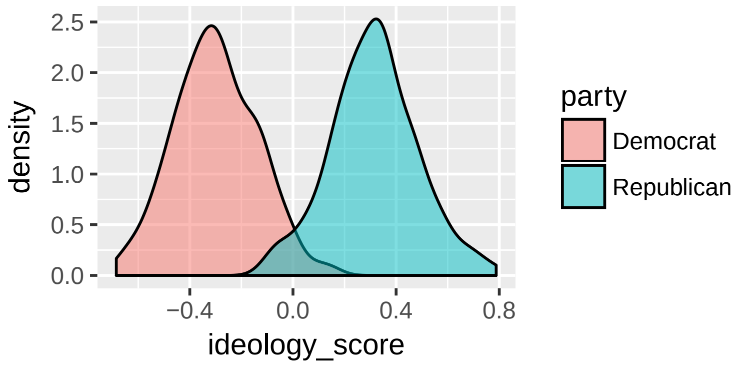
Instead of coloring the lines differently, we could "fill"" the density with different colors. While the color aesthetic represents the color of the line itself, the fill aesthetic represents the color inside the line. We use fill distinguish the parties by adding `fill = party` to out aesthetics.

```
# build density plot
ggplot(nominate100,
       aes(x = ideology_score, fill = party)) +
  geom_density()
```



Notice that the Republican fill (blue-green) completely covers the Democrat fill (orange-red). But here's a hint. If we supply and `alpha` argument to `geom_density()` it will may the colors transparent. `alpha = 0` would be completely transparent. `alpha = 1` would be not transparent at all.

```
# build density plot
ggplot(nominate100,
       aes(x = ideology_score, fill = party)) +
  geom_density(alpha = 0.5)
```



This slight transparency allows us to see the distribution for Democrats and the distribution for Republicans, even when the two overlap.

Review Questions

1. Explain the difference between the color and fill aesthetics.
2. Explain how you could use the color aesthetic to fit, on a single plot, the separate densities of the ideology scores for representative all 50 states.
3. Explain how alpha transparency works and why you might use it. Suppose I used `alpha = 0.1` instead of `alpha = 0.5` in the plot above. How would the plot change?

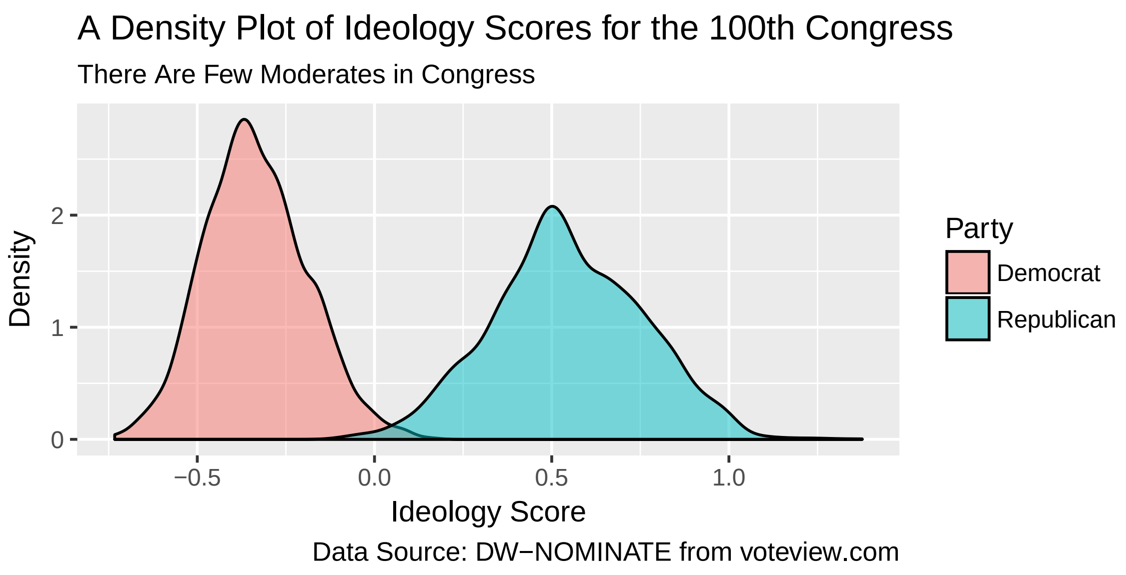
Labels

By default, `ggplot()` uses the variable names to label the axes and legend. By default, there is no title, subtitle, or caption.

You will usually want to improve the axis labels and legends. You will sometimes want to add a title, subtitle, and/or caption. We make this changes by adding the `labs()` function to the plot.

To the `labs()` function, you simply supply one argument per aesthetic, such as `x`, `y`, `color`, or `fill`. You can also supply arguments for `title`, `subtitle`, or `caption` if you wish. These argument are character strings (surrounded by quotes), such as `"Ideology Score"` or `"A Density Plot of Ideology Scores for the 100th Congress"`.

```r
# build density plot
ggplot(nominate, aes(x = ideology_score, fill = party)) +
  geom_density(alpha = 0.5) +
  labs(x = "Ideology Score",
       y = "Density",
       fill = "Party",
       title = "A Density Plot of Ideology Scores for the 100th Congress",
       subtitle = "There Are Few Moderates in Congress",
       caption = "Data Source: DW-NOMINATE from voteview.com")
```



This plot looks pretty nice.

Review Exercises

1. By default, how does `ggplot()` label the axes and legends?
2. How can you change the default labels? How can you add a title, subtitle, or caption?
3. Explain what the `labs()` function does and how to use it.
4. Explain, in as much detail as you can, what each part the last block of code above does.

Themes

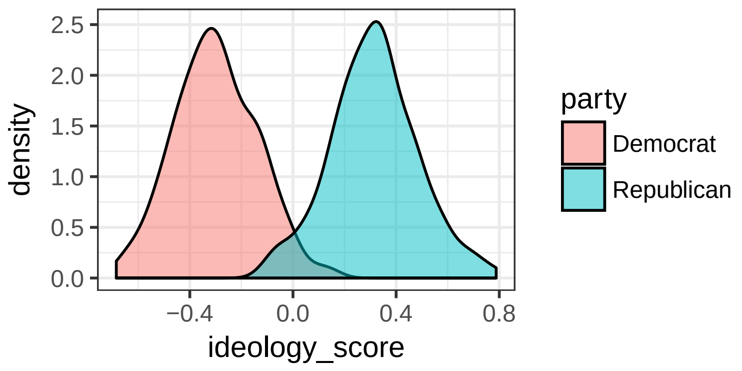
We can use themes to control the overall look of our ggplots. The theme controls elements such as the color of the background, the font family and size, the color and size of the grid, etc.

There are the six themes provided with `ggplot2`.

Theme	Description
`theme_grey()`	The signature `ggplot2` theme with a grey background and white gridlines, designed to put the data forward yet make comparisons easy.
`theme_bw()`	The classic dark-on-light `ggplot2` theme. May work better for presentations displayed with a projector.
`theme_linedraw()`	A theme with only black lines of various widths on white backgrounds, reminiscent of a line drawings. `Serves a purpose similar to theme_bw`.
`theme_light()`	A theme similar to `theme_linedraw` but with light grey lines and axes, to direct more attention towards the data.
`theme_minimal()`	A minimalist theme with no background annotations.
`theme_classic()`	A classic-looking theme, with x and y axis lines and no gridlines.

Let's see what this looks like by adding the theme `theme_bw()` (my favorite!) to our plot.

```r
# build density plot
ggplot(nominate100, aes(x = ideology_score, fill = party)) +
  geom_density(alpha = 0.5) +
  theme_bw()
```



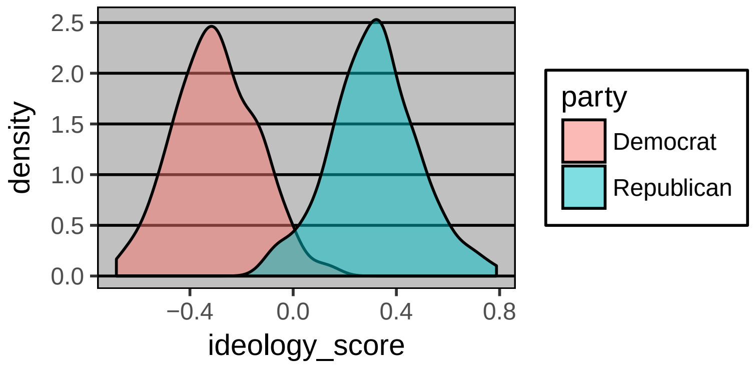
If you are interested, the `ggthemes` package includes several other themes.

Theme	Description
`theme_economist()`	based on the Economist
`theme_economist_white()`	based on the Economist
`theme_excel()`	based on Excel
`theme_few()`	based on Few's "Practical Rules for Using Color in Charts"
`theme_fivethirtyeight()`	based on fivethirtyeight.com plots
`theme_gdocs()`	based on Google docs
`theme_igray()`	inverse grey
`theme_pander()`	based on the pander package
`theme_solarized()`	based on the Solarized palette
`theme_solarized_2()`	based on the Solarized palette
`theme_stata()`	based on default Stata graphics
`theme_tufte()`	based on Tufte–minimum ink, maximum data
`theme_wsj()`	based on Wall Street Journal

Let's try the infamous (famously ugly) Excel theme.

```r
# load packages
library(ggthemes)  # for additional themes and even more fun!

# build density plot
ggplot(nominate100, aes(x = ideology_score, fill = party)) +
  geom_density(alpha = 0.5) +
  theme_excel()
```



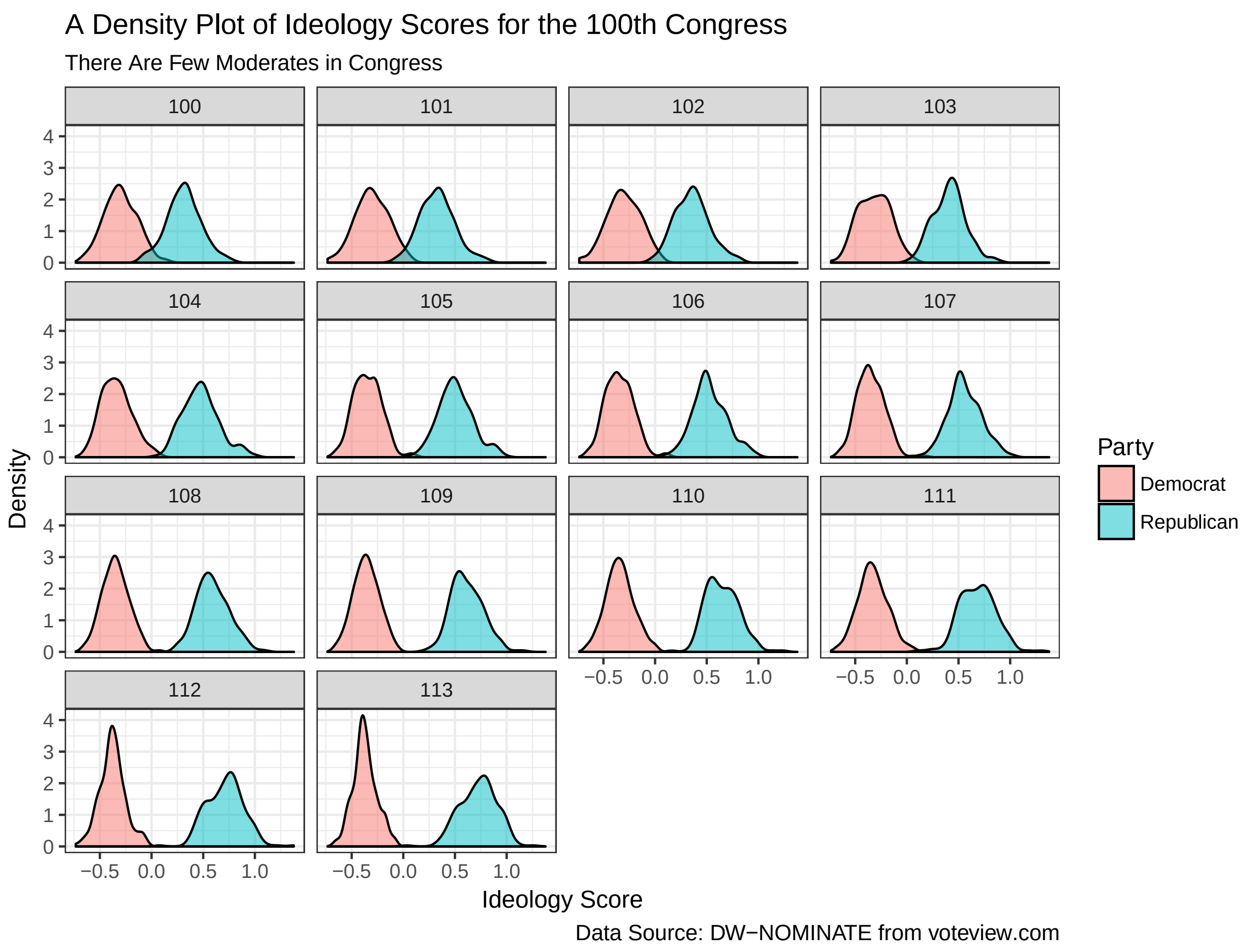
Review Exercises

1. What are themes in ggplot? What do they control/change?
2. How do you change the theme of a ggplot?
3. What are a few theme options from ggplot? From ggthemes?
4. Try a few different themes from both ggplot2 and ggthemes. What is your favorite and why?

Putting It All Together

We'll add two things back in. First, we'll put out nice labels back in. Second, let's go back to the `nominate` data frame (we've been using `nominate100`) and facet by Congress.

```r
# build density plot
ggplot(nominate, aes(x = ideology_score, fill = party)) +
  geom_density(alpha = 0.5) +
  facet_wrap(~congress) +
  labs(x = "Ideology Score",
       y = "Density",
       fill = "Party",
       title = "A Density Plot of Ideology Scores for the 100th Congress",
       subtitle = "There Are Few Moderates in Congress",
       caption = "Data Source: DW-NOMINATE from voteview.com") +
  theme_bw()
```



Saving ggplots

It is quite easy to save a ggplot as a `.png` or some other file. You just use the function `ggsave()`, which, by default, saves the last plot you created.

1. Create the plot you want.
2. Decide the file type you want to save your figure as. `.png` is a fine choice for our purposes.[1]
3. Choose where you want to save your figure. This should be somewhere in your working directory. Suppose you are making the figure for a paper. You might have a project folder called `cool-paper`–you've already set `cool-paper` as your working directory. If you're following my advice, then you'll have a subdirectory `doc` in `cool-paper`. That's where you'll keep your paper, maybe as a `.docx` file. I recommend having a subdirectory `figs` in `doc` where you save your figures. That way your figures don't clutter the directory where you keep your paper. If you follow that advice, then you'll almost always want to save your figures to `doc/figs`.
4. Choose a name for your figure. I recommend something compact and descriptive, which is easier said than done. For our final density plot, I might use `ideology-density-by-congress.png`. The `.png` extension indicates the file type. `ggsave()` use the extension to guess the file type you want to use.
5. Use `ggsave()`.

There are three important arguments to `ggsave()`.

[1]For high-quality figures, you might want to consider '.pdf', but these are not as easy to work with in Word '.docx' files.

1. `filename`: This is the first argument to `ggsave()`. It is just the file path–the place you want to save the file and the name of the file. If you are following my suggestions, it would be `"doc/figs/ideology-density-by-congress.png"`.
2. `height`: You should explicitly name this argument. It controls the height of the figure (in inches, by default). `height = 4` is a good starting point, but you'll want to experiment.
3. `width`: You should explicitly name this argument. It controls the width of the figure (in inches, by default). `width = 5` is a good starting point, but you'll want to experiment.

For the last figure we created, I experimented a little and discovered that 5 inches tall and 8 inches wide works well.

```r
# save last plot as png
ggsave("doc/figs/ideology-density-by-congress.pdf", height = 5, width = 8)
```

Just to review and wrap up, here is the fill R script that I might use to create and save this figure.

```r
# set working directory
setwd("~/Dropbox/classes/pols-209")

# load packages
library(ggplot2)  # for creating figures
library(readr)  # for read_csv()
library(tibble)  # for glimpse()

# load data
nominate <- read_csv("data/nominate.csv")  # data are in the data subfolder

# quick look at the data
glimpse(nominate)

# build plot
ggplot(nominate, aes(x = ideology_score, fill = party)) +
  geom_density(alpha = 0.5) +
  facet_wrap(~ congress) +
  labs(x = "Ideology Score",
       y = "Density",
       fill = "Party",
       title = "A Density Plot of Ideology Scores for the 100th Congress",
       subtitle = "There Are Few Moderates in Congress",
       caption = "Data Source: DW-NOMINATE from voteview.com") +
  theme_bw()

# save last plot as png (to the figs subfolder of the doc subfolder)
ggsave("doc/figs/ideology-density-by-congress.pdf", height = 5, width = 8)
```

Review Exercises

1. Explain how to save a ggplot. Be sure to discuss each of the five steps.
2. Identify and describe each of the three arguments we usually supply to `ggsave()`.
3. Explain, in as much detail as you can, what each part the last block of code above does.