Module 3: R

Visualization

Instructor: Anjali Silva, PhD

TA: Tia Harrison, MSc

Data Sciences Institute, University of Toronto

9 July 2022

Course Documents

- Visit: https://github.com/anjalisilva/IntroductionToR
- All course material will be available via IntroductionToR GitHub repository (https://github.com/anjalisilva/IntroductionToR). Folder structure is as follows:
    - Lessons - All files*: These folders contain all files.
    - Lessons - Data only: This folder contains data only.
    - Lessons - Lesson Plans only: This folder contains lesson plans only.
    - Lessons - PDF only: This folder contains slide PDFs only.
    - README - README file
    - .gitignore - Files to ignore specified by instructor

Course Contacts

- Instructor: Anjali Silva Email: a.silva@utoronto.ca (Must use the subject line DSI-IntroR. E.g., DSI-IntroR: Inquiry about Lecture I.)
- TA: Tia Harrison Email: tia.harrison@mail.utoronto.ca

Overview

- ggplot
- Initializing plots, specifying variables, and choosing chart types
- Customizing plots with labels, axes, color, size, multiple graph types, multiple plots, and overall look

(Wickham and Grolemund, 2017, Chapter 3; Healy, 2018, Chapter 3; Alexander, 2022, Chapter 6)

ggplot

ggplot

ggplot2 is a package that allows us to make graphics in R. It's loaded with the tidyverse.

Initializing a plot

the `ggplot()` function initializes the plot. In the arguments, you will identify the base of your plot. This includes:

- the data we want to graph from
- the aesthetics we will use

What doesn't go in the `ggplot()` function:

- the type of graph we want
- the way we want the axes to look
- the labels we want

```
ggplot(data = my_data, mapping = aes())
```

Aesthetic

In the `mapping` argument, we specific our aesthetic using `aes()`

This is where we indicate what variable we want for the x axis, the y axis, the color, or any other feature that the plot in question might have.

```
aes(x = variable1,
    y = variable2,
    shape = variable3,
    color = variable4,
    size = variable5,
    fill = variable6)
```

In combination

This takes `data` and passes it to ggplot, which initializes a plot using that data and specifies that variable1 will be represented on the x axis and variable2 on the y axis.

```
data %>%
  ggplot(aes(x = variable1, y = variable2))
```

Adding layers

After initializing, you still won't have a plot. You have to add layers -- which includes the type of plot you want, as well as tweaks and formatting specifics.

When we add layers to a ggplot object, we use the `+` between lines:

```
data %>%
  ggplot(aes(x = variable1, y = variable2)) +
  # a layer +
  # another layer
```

Geoms

Geom layers add types of plot. There are more than 40 geoms! Some common ones:

Bar plots

```
geom_bar()
```

Histograms

```
geom_histogram()
```

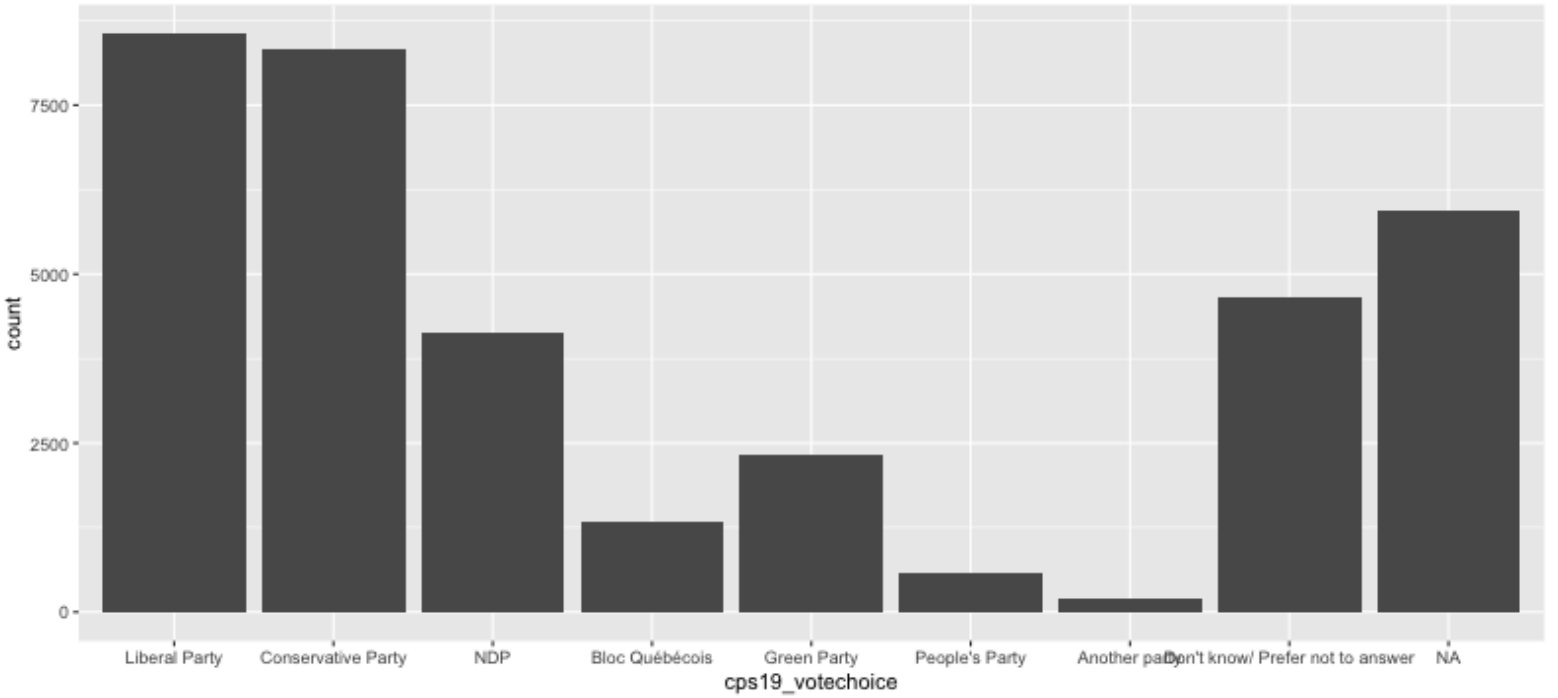
Scatterplots

```
geom_point()
```

Examples

Bar plot

```
CES_data %>%
  ggplot(aes(x = cps19_votechoice, )) +
  geom_bar()
```



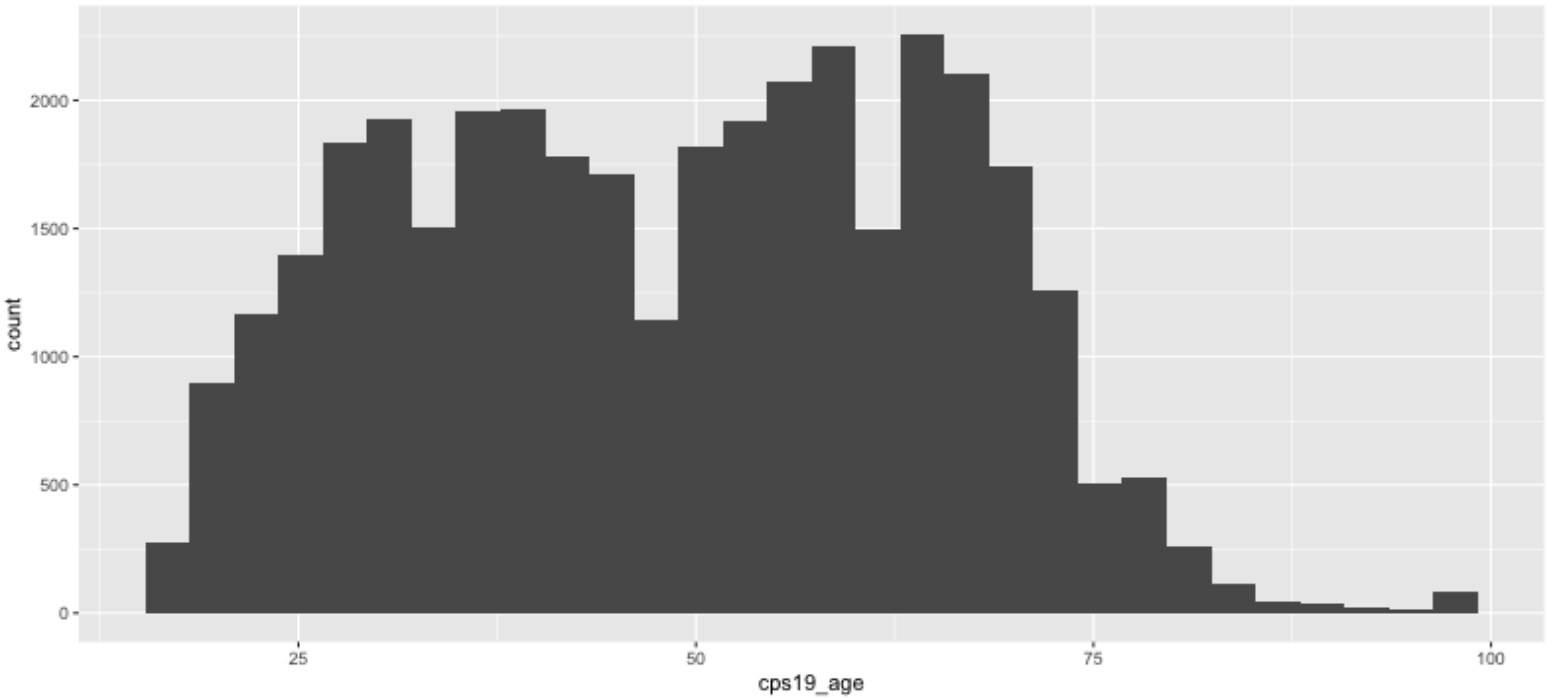
Options for barplots

These are the defaults in a geom_bar:

```
geom_bar(
  stat = "count", # can change to "prop" for proportion
  position = "stack", # can change to "dodge" or "fill"
  width = NULL, # can put a bar width here
  na.rm = FALSE, # can remove NAs
  orientation = NA,  # can specify "x" or "y"
  show.legend = NA # can add or remove a legend
)
```

Histogram

```
CES_data %>%
  mutate(age = 90 - cps19_yob) %>%
  ggplot(aes(x = cps19_age)) +
  geom_histogram()
```

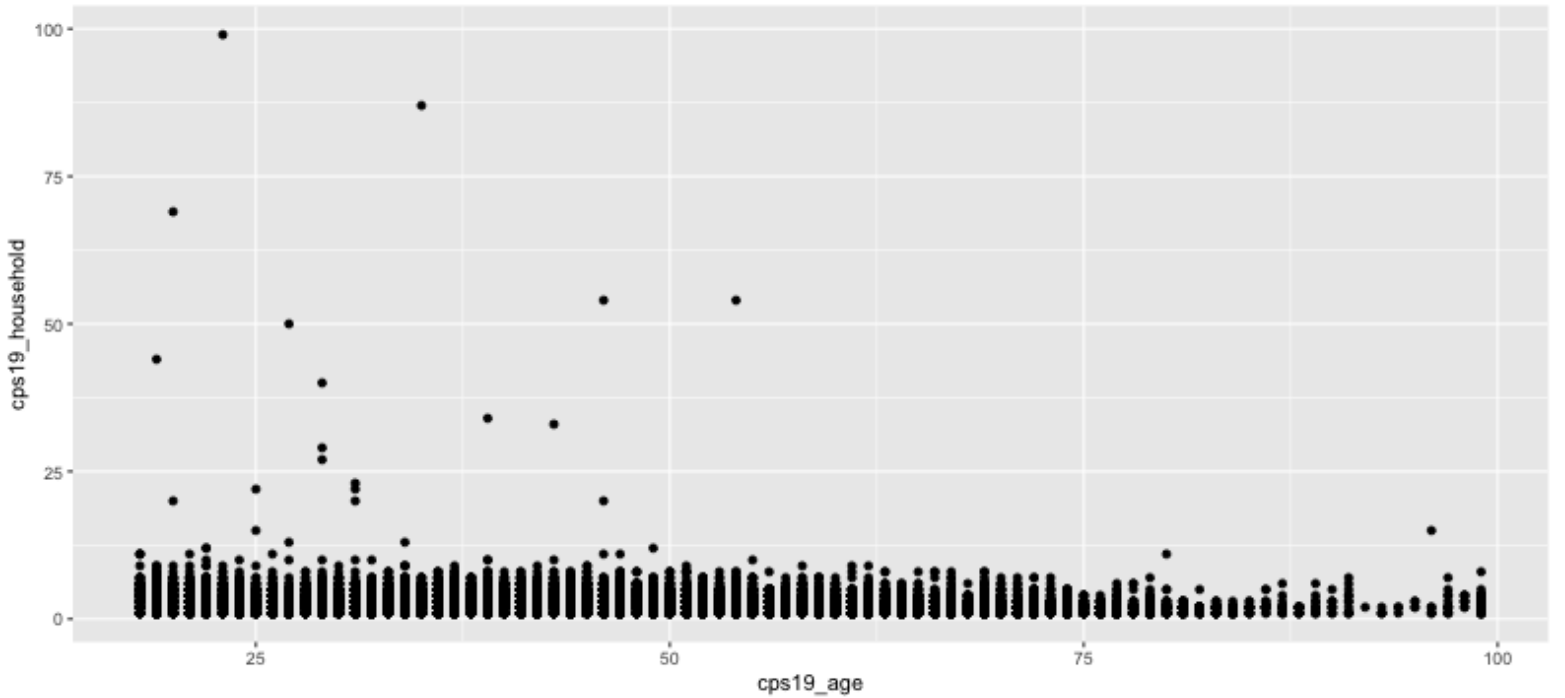


Options for histograms

```
geom_histogram(
  stat = "bin", # can change to "count"
  position = "stack", # can change to "identity", "dodge", or "fill"
  binwidth = NULL, # can specify the range of each bin
  bins = NULL, # can specify the number of bins
  na.rm = FALSE, # can tell it to ignore NAs
  orientation = NA, # can specify "x" or "y"
  show.legend = NA # can add or remove a legend
)
```

Scatter plot

```
CES_data %>%
  mutate(age = 90 - cps19_yob) %>%
  ggplot(aes(x = cps19_age, y = cps19_household)) +
  geom_point()
```



Options for scatterplots

```
geom_point(
  position = "identity", # can change to "jitter"
  na.rm = FALSE, # can ignore NAs
  show.legend = NA # can add or remove a legend
)
```

Exercises

Using different variables:

1. Make a barplot
2. Make a histogram
3. Make a scatterplot

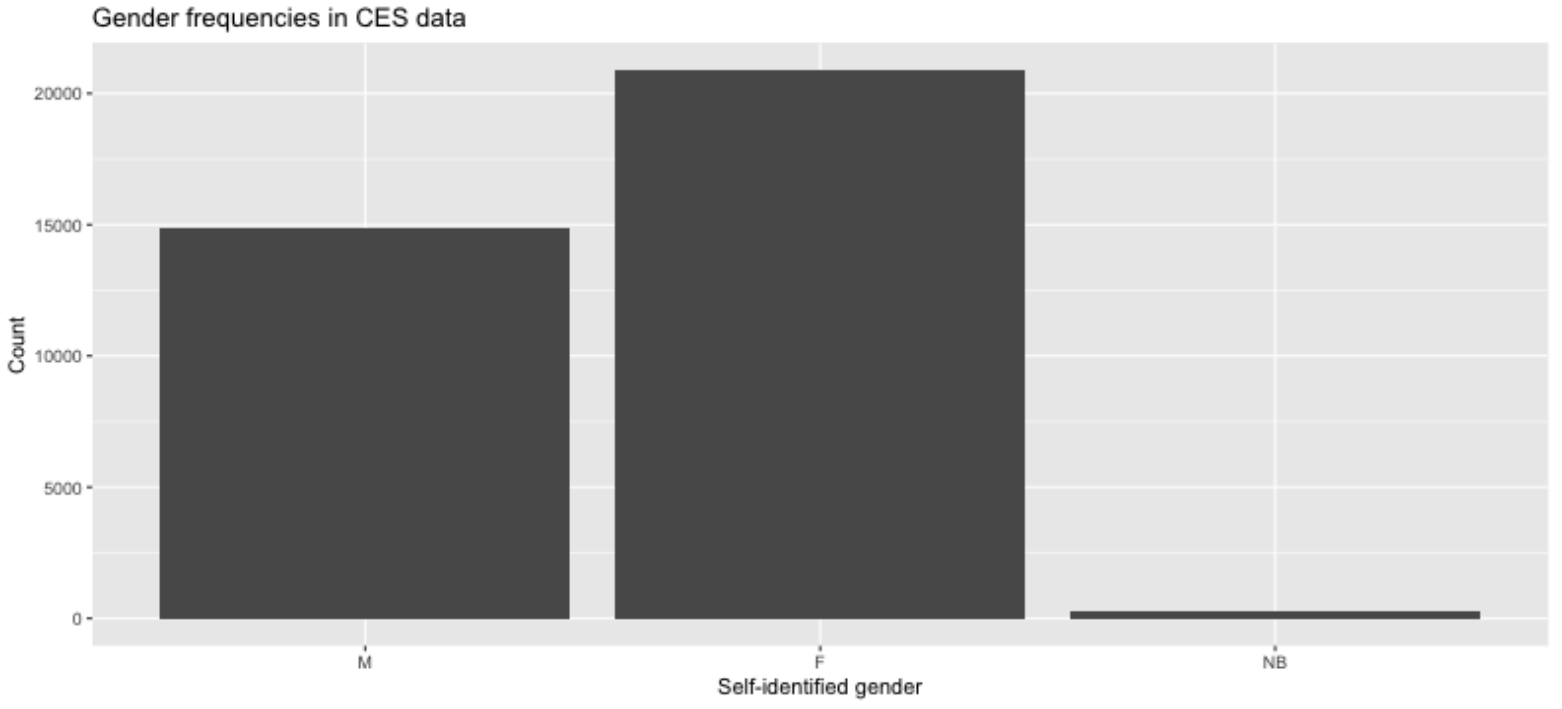
Customizing plots

Labels

We can change the way that labels appear to improve the look and readability.

```
labs(x = "X name",
     y = NULL,
     title = "Title")
```

```
CES_data %>%
  ggplot(aes(x = cps19_gender)) +
  geom_bar() +
  labs(x = "Self-identified gender",
       y = "Count",
       title = "Gender frequencies in CES data")
```



Axes

How we change the axes depends on what types of variables we have.

The layers take the form: `scale_<which axis>_<what type of axis>()`.
There are VERY many `scale_` options.

Similar layers can be added for x and y axes, as well as other graph features like color and size.

Manipulate a continuous x axis

```
scale_x_continuous(breaks = , # use a vector to specify locations
                   minor_breaks = , # also can be a vector
                   n.breaks = , # can specify the number of breaks
                   labels = , # change the labels on the breaks
                   limits = , # set the upper and lower limits
                   position = # "left", "right", "top", "bottom")
```

Manipulate a discrete x axis

```
scale_x_discrete(breaks = , # character vector of breaks
                 limits = , # set possible values for scale
                 drop = , # TRUE or FALSE to drop unused factor levels
                 labels = , # change the labels on the breaks
                 psiition = # "left", "right", "top", "bottom")
```

Fill

We can change the fill to represent a variable:

```
data %>%
  ggplot(aes(x = variable1,
             fill = variable2)) +
  geom_()
```

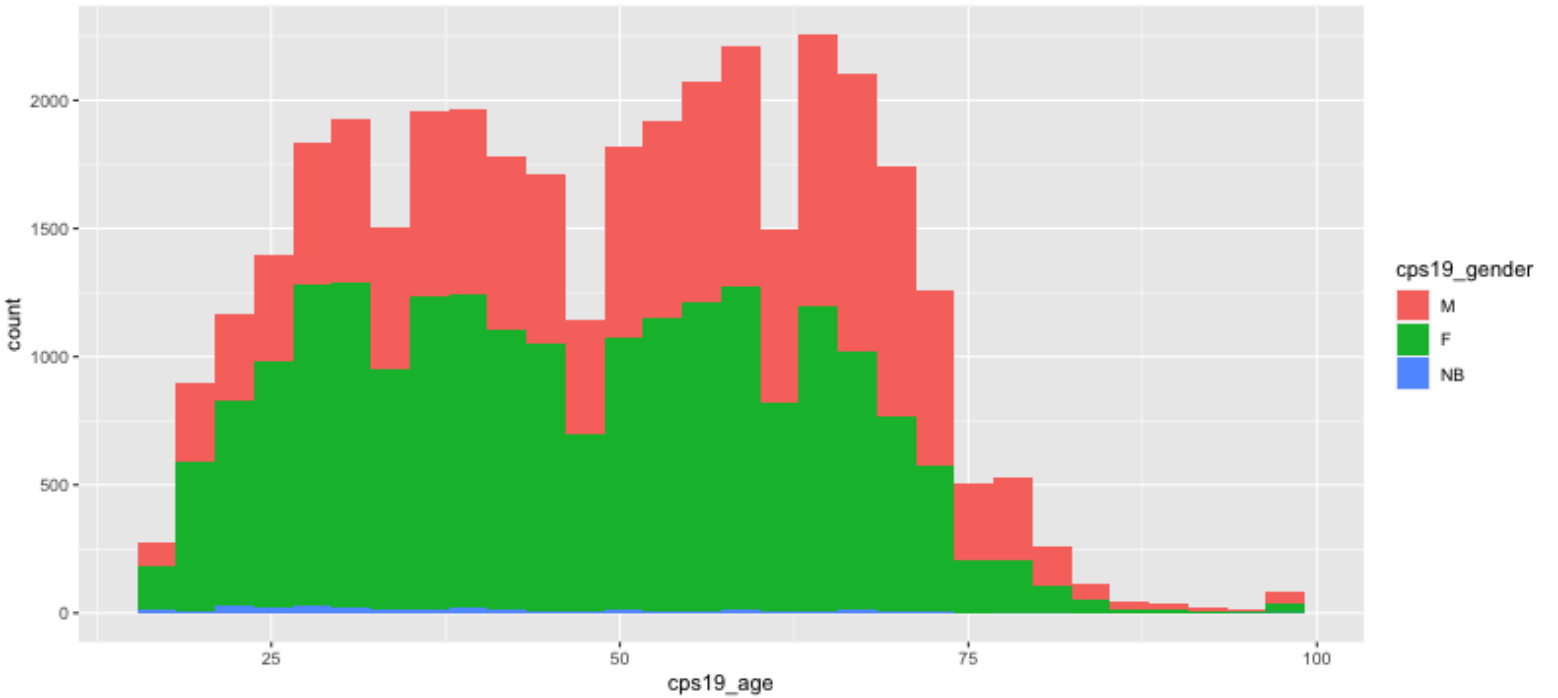
or to be a color of choice:

```
data %>%
  ggplot(aes(x = variable1)) +
  geom_(fill = "my_color")
```

The difference is where the `fill =` is located. If it is in the aes(), then it will represent a variable. If it is not in the aes(), it will just change the look of the graph.

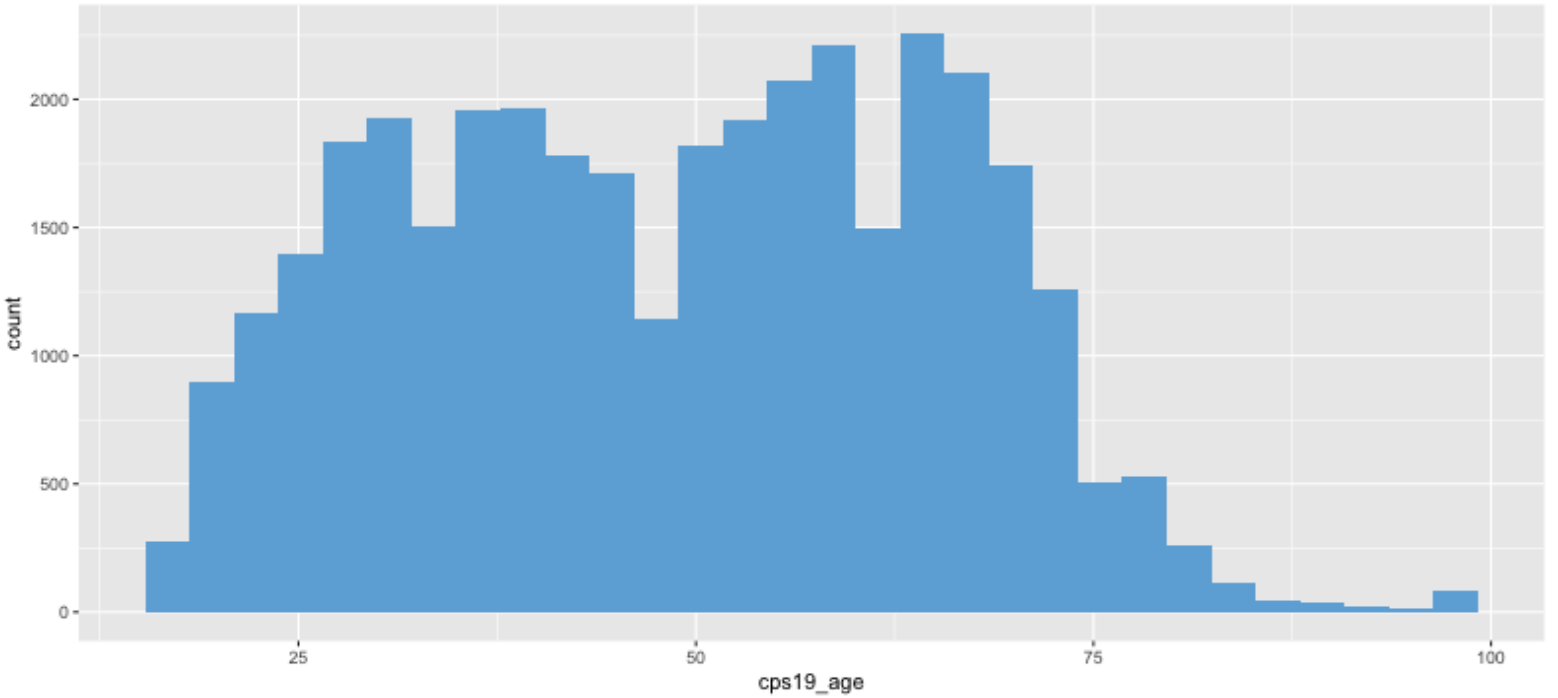
Fill to represent a variable

```
CES_data %>%
  ggplot(aes(x = cps19_age,
             fill = cps19_gender)) +
  geom_histogram()
```



Changing fill to a specific color

```
CES_data %>%
  ggplot(aes(x = cps19_age)) +
  geom_histogram(fill = "#6DAEDB")
```



Color

For some geoms, you will need to change color rather than fill.

We can change the color to represent a variable:

```
data %>%
  ggplot(aes(x = variable1,
             color = variable2)) +
  geom_()
```

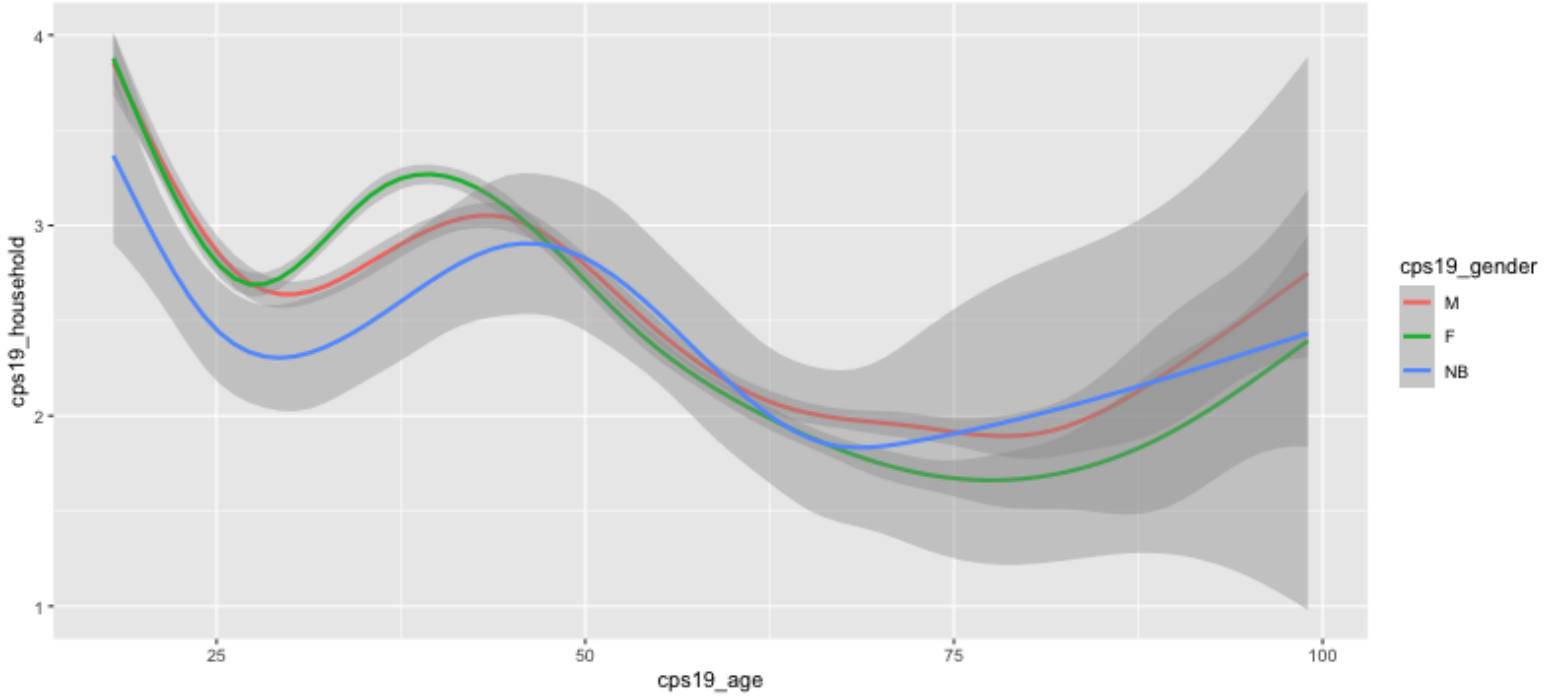
or to be a color of choice:

```
data %>%
  ggplot(aes(x = variable1)) +
  geom_(color = "my_color")
```

The difference is where the `color =` is located. If it is in the aes(), then it will represent a variable. If it is not in the aes(), it will just change the look of the graph.

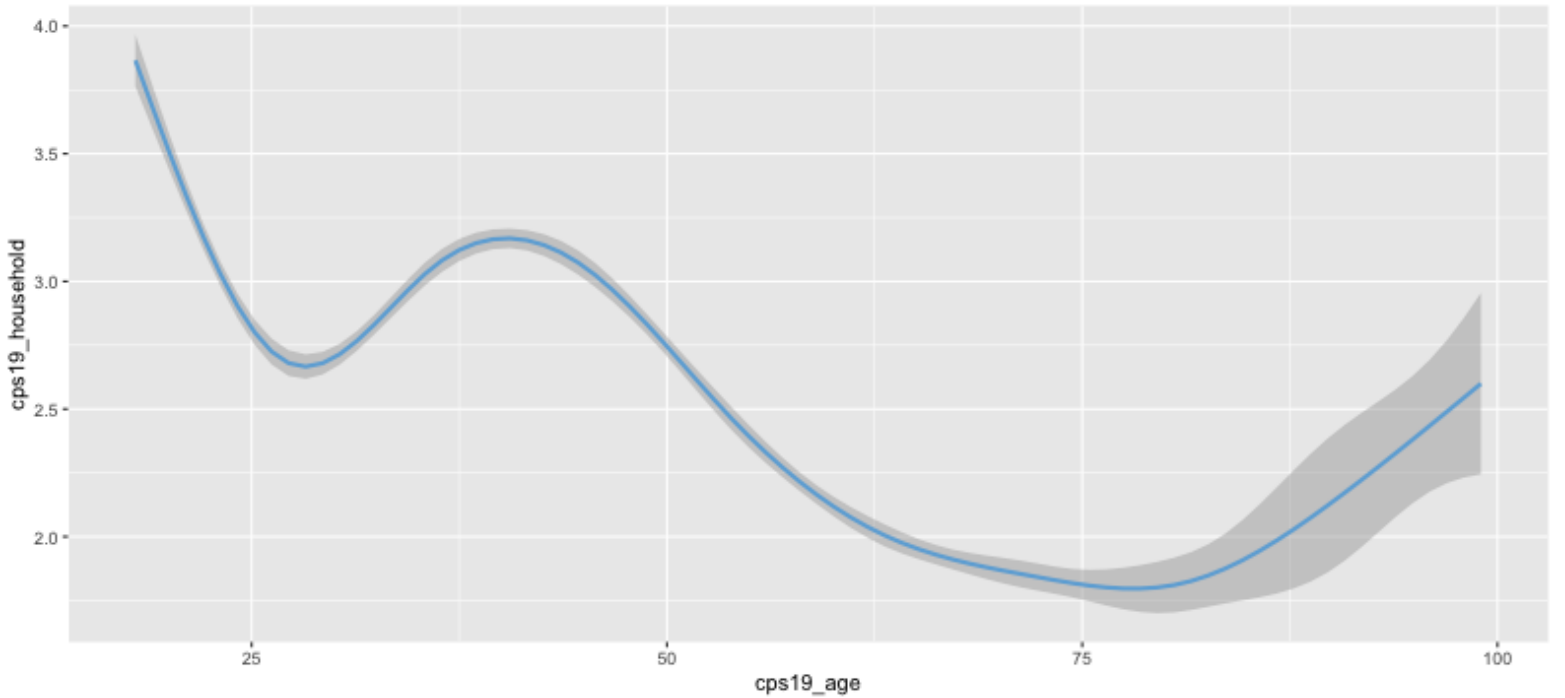
Color to represent a variable

```
CES_data %>%
  ggplot(aes(x = cps19_age,
             y = cps19_household,
             color = cps19_gender)) +
  geom_smooth()
```



Color for visual effect

```
CES_data %>%
  ggplot(aes(x = cps19_age,
             y = cps19_household)) +
  geom_smooth(color = "#6DAEDB")
```



Size

We can change the size to represent a variable:

```
data %>%
  ggplot(aes(x = variable1,
             y = variable2,
             size = variable3)) +
  geom_()
```

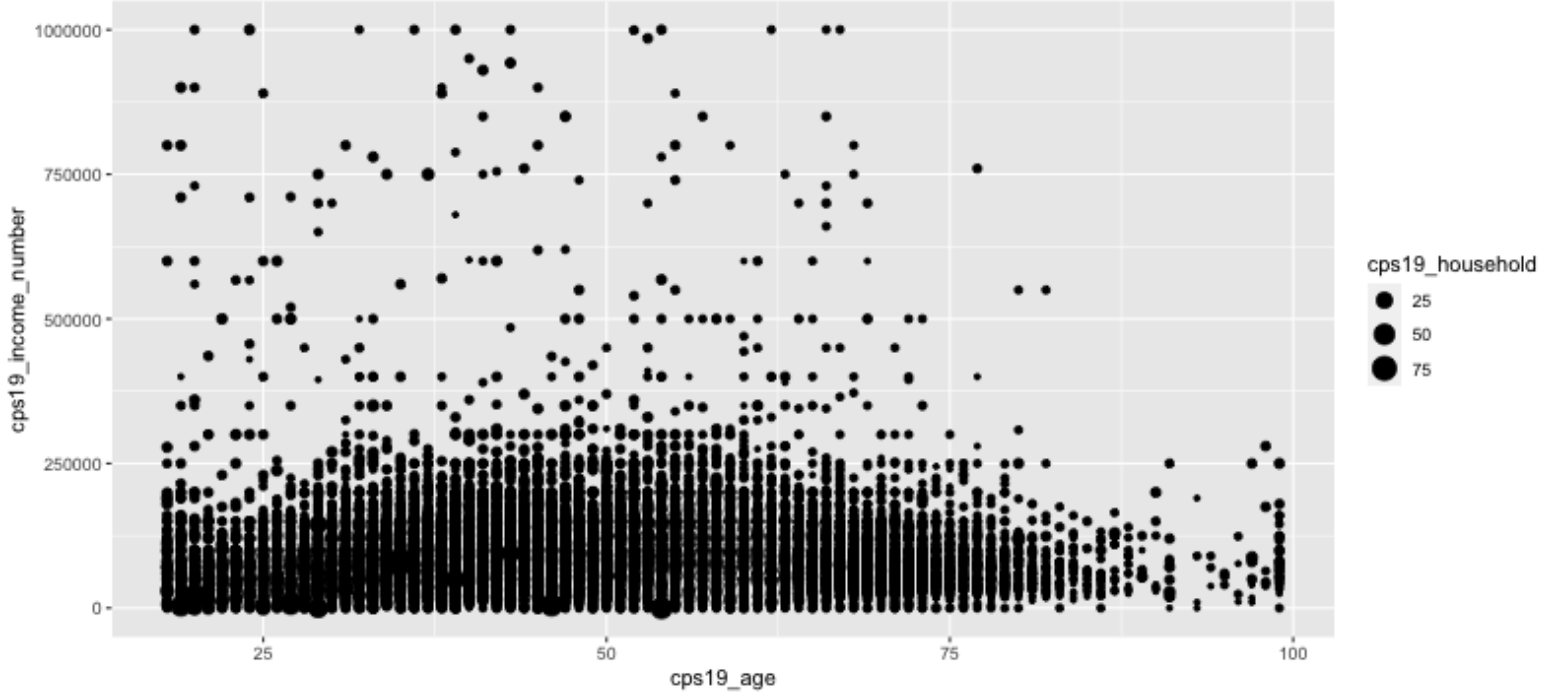
or to be a color of choice:

```
data %>%
  ggplot(aes(x = variable1,
             y = variable2)) +
  geom_(size = "my_size")
```

The difference is where the `fill =` is located. If it is in the aes(), then it will represent a variable. If it is not in the aes(), it will just change the look of the graph.

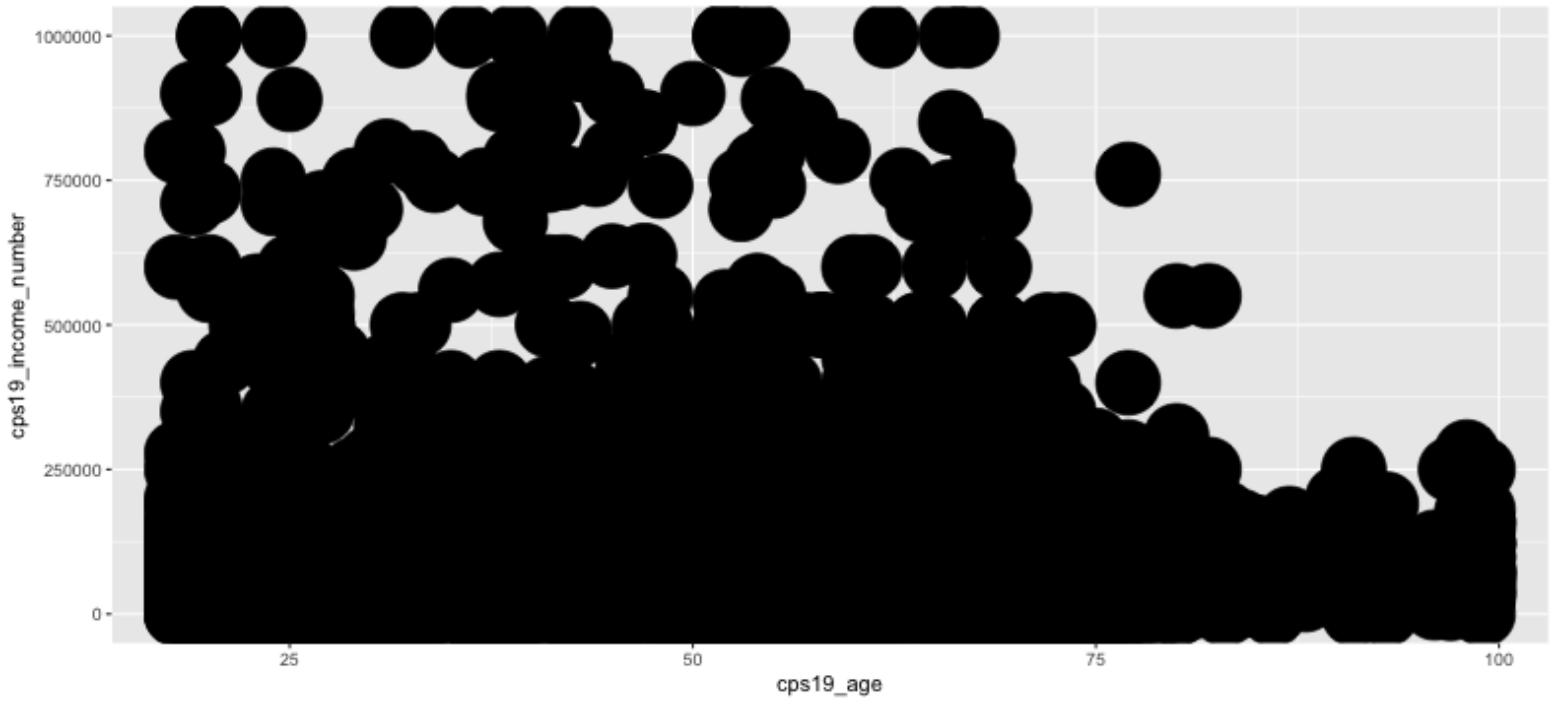
Size to represent a variable

```
CES_data %>%
  ggplot(aes(x = cps19_age,
             y = cps19_income_number,
             size = cps19_household)) +
  geom_point()
```



Size for visual effect

```
CES_data %>%
  ggplot(aes(x = cps19_age,
             y = cps19_income_number)) +
  geom_point(size = 15)
```



Using multiple geoms

We can layer geoms on top of one another.

Geoms can either share their aes():

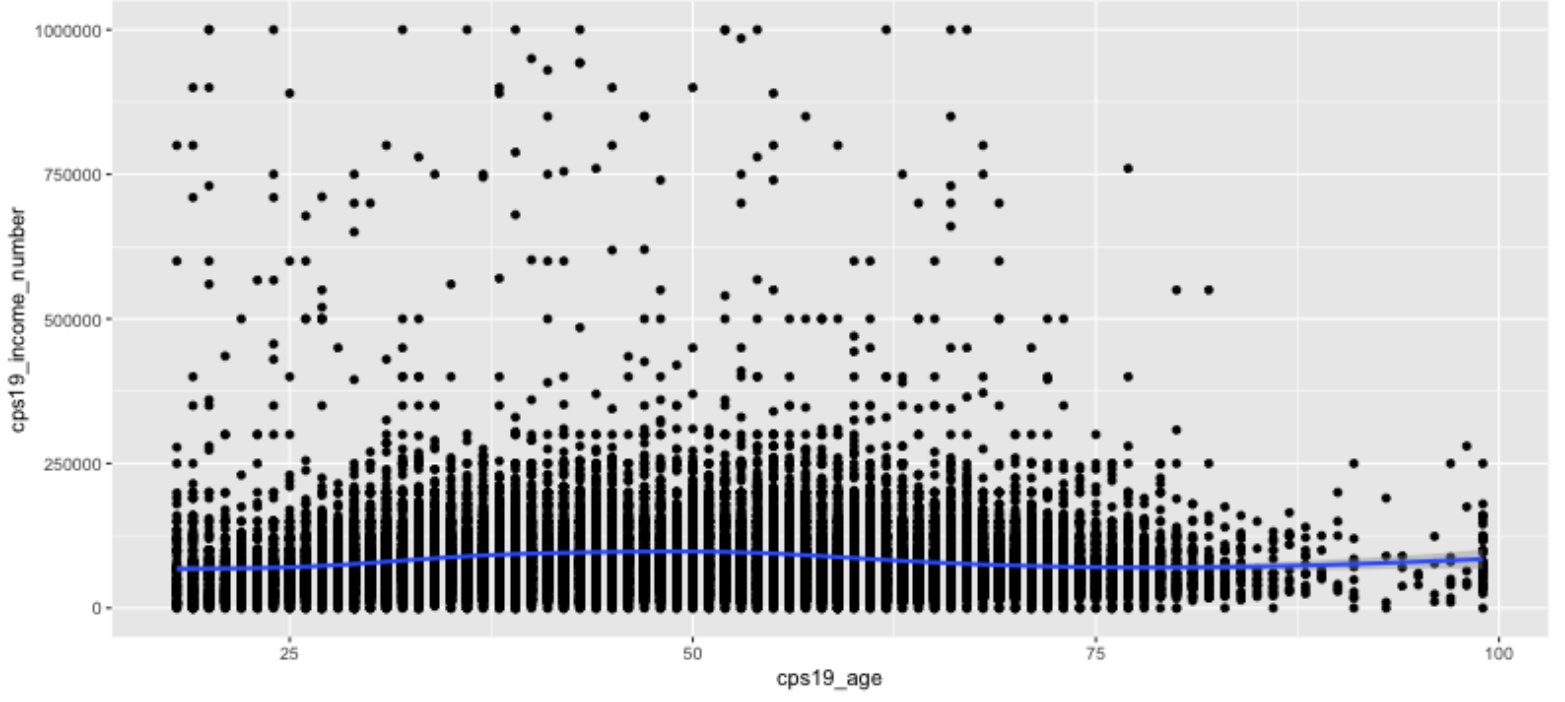
```
data %>%
  ggplot(aes(x = variable1,
             y = variable2)) +
  geom_() +
  geom_()
```

or they can have their own:

```
data %>%
  ggplot() +
  geom_(aes(x = variable1,
            y = variable2)) +
  geom_(aes(x = variable1,
            y = variable3))
```

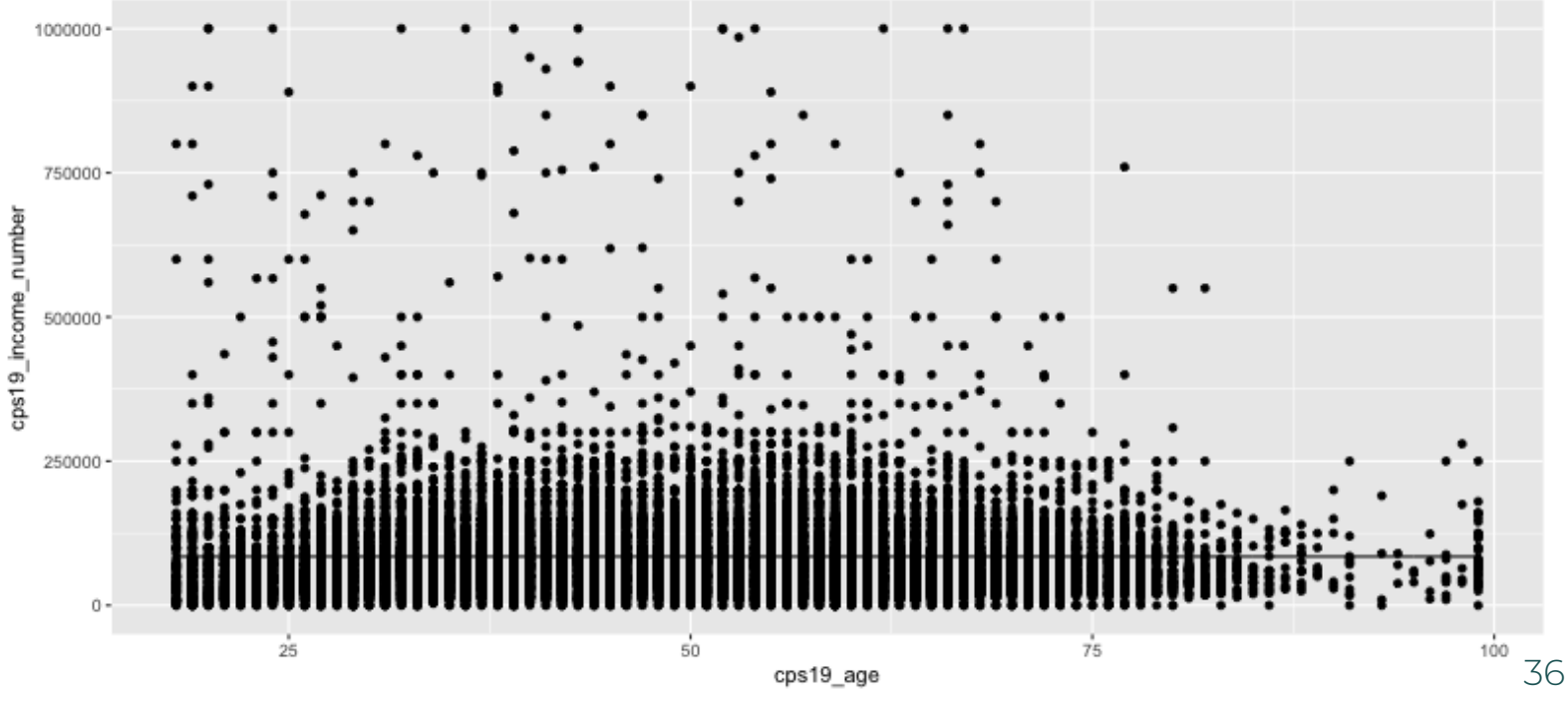
Geoms that share aesthetics

```
CES_data %>%
  ggplot(aes(x = cps19_age,
             y = cps19_income_number)) +
  geom_point() +
  geom_smooth()
```



Separate geom aesthetics

```
CES_data %>%
  ggplot() +
  geom_point(aes(x = cps19_age,
                 y = cps19_income_number)) +
  geom_line(aes(x = cps19_age,
                y = mean(cps19_income_number, na.rm = TRUE)))
```



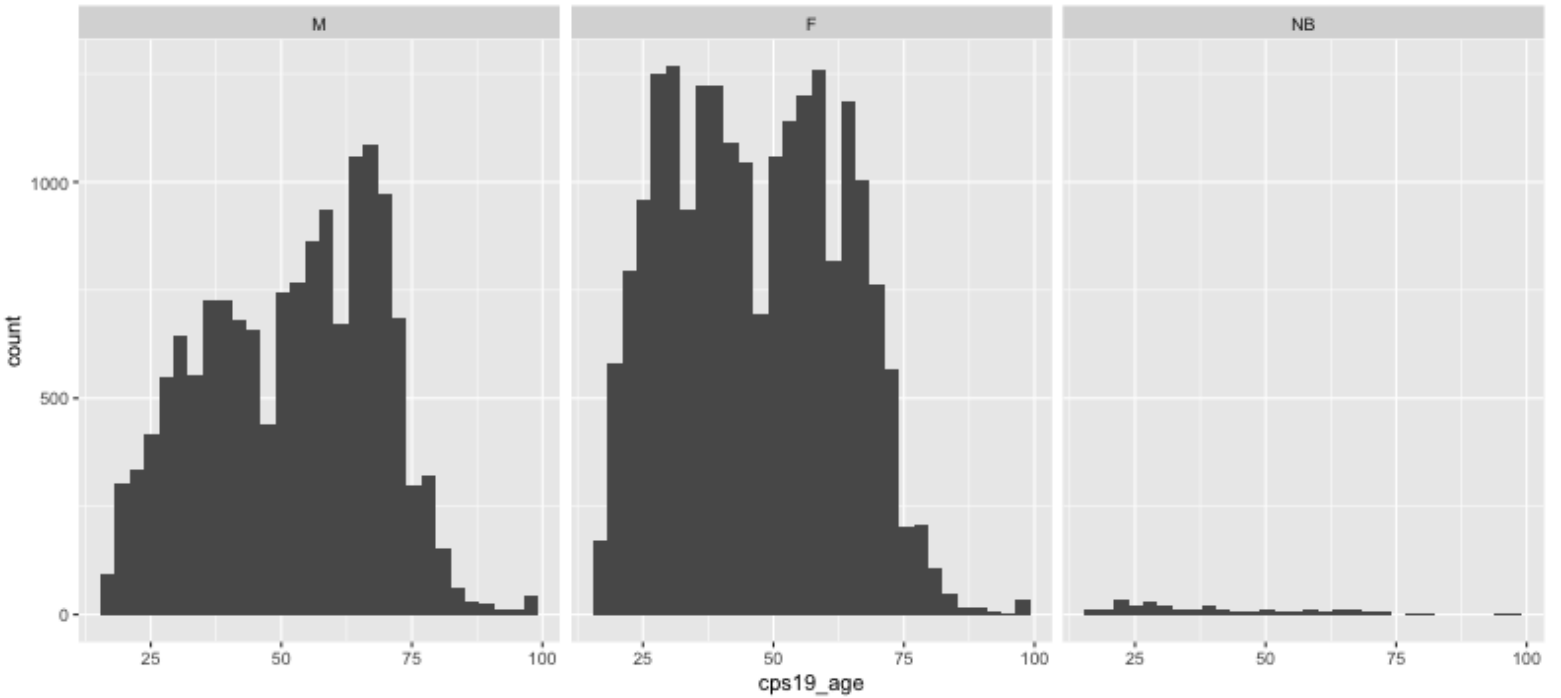
Facets

Facets give you side-by-side graphs for different categories.

```
facet_wrap(facets = "variables you want to facet by")
facet_grid(facets = "variables that you want to facet by")
```

Facets

```
CES_data %>%
  ggplot(aes(x = cps19_age)) +
  geom_histogram() +
  facet_wrap(facets = "cps19_gender")
```



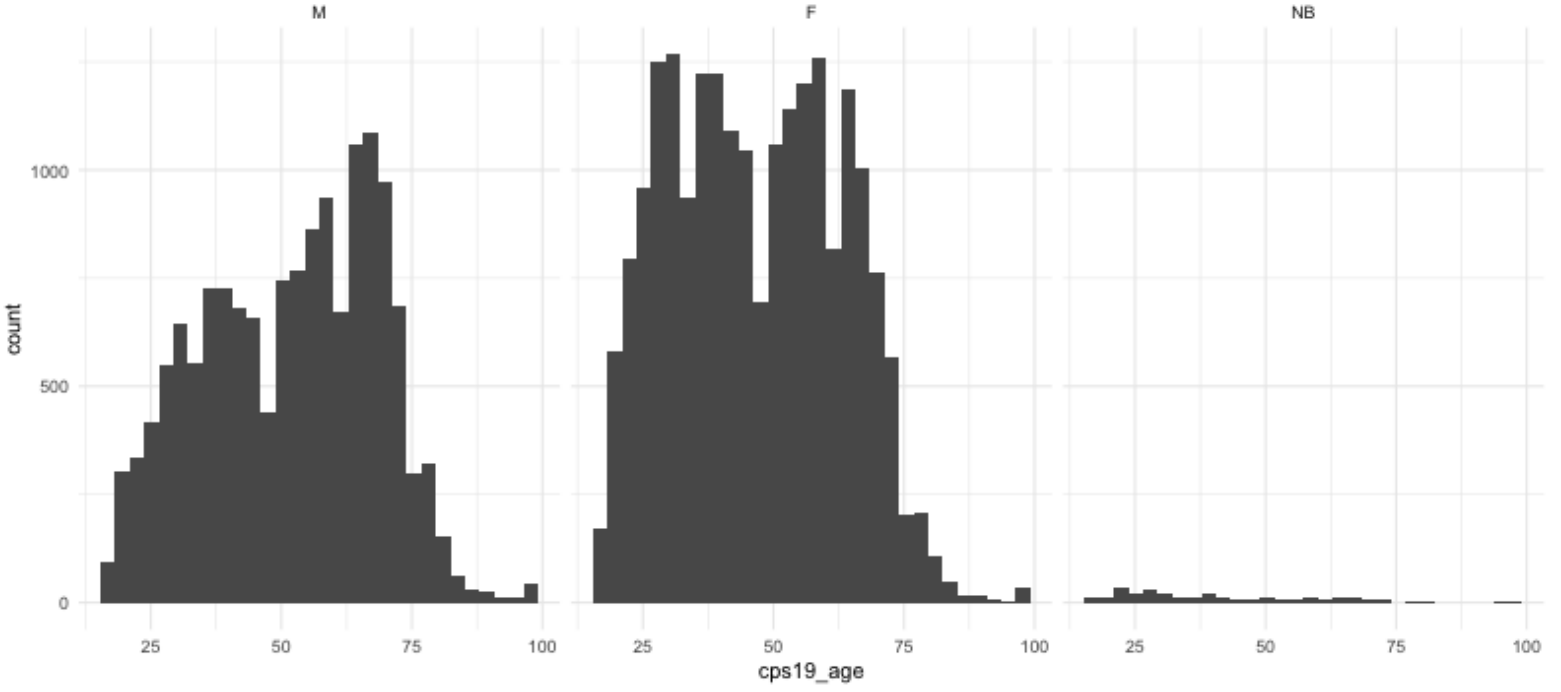
Themes

Themes are added at the end. They control the overall look.

```
theme_bw()
theme_classic()
theme_light()
theme_minimal()
```

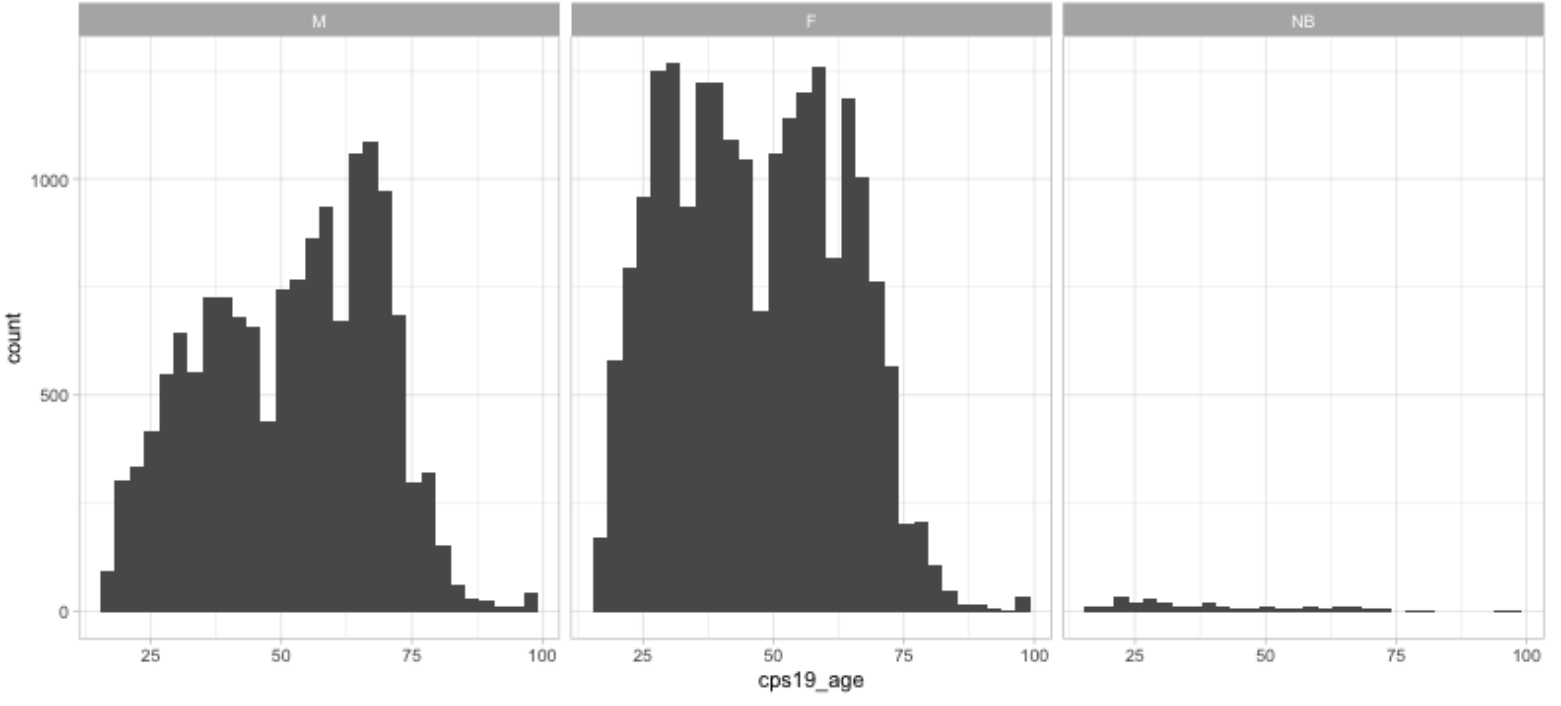
Themes

```
CES_data %>%
  ggplot(aes(x = cps19_age)) +
  geom_histogram() +
  facet_wrap(facets = "cps19_gender") +
  theme_minimal()
```



Themes

```
CES_data %>%
  ggplot(aes(x = cps19_age)) +
  geom_histogram() +
  facet_wrap(facets = "cps19_gender") +
  theme_light()
```

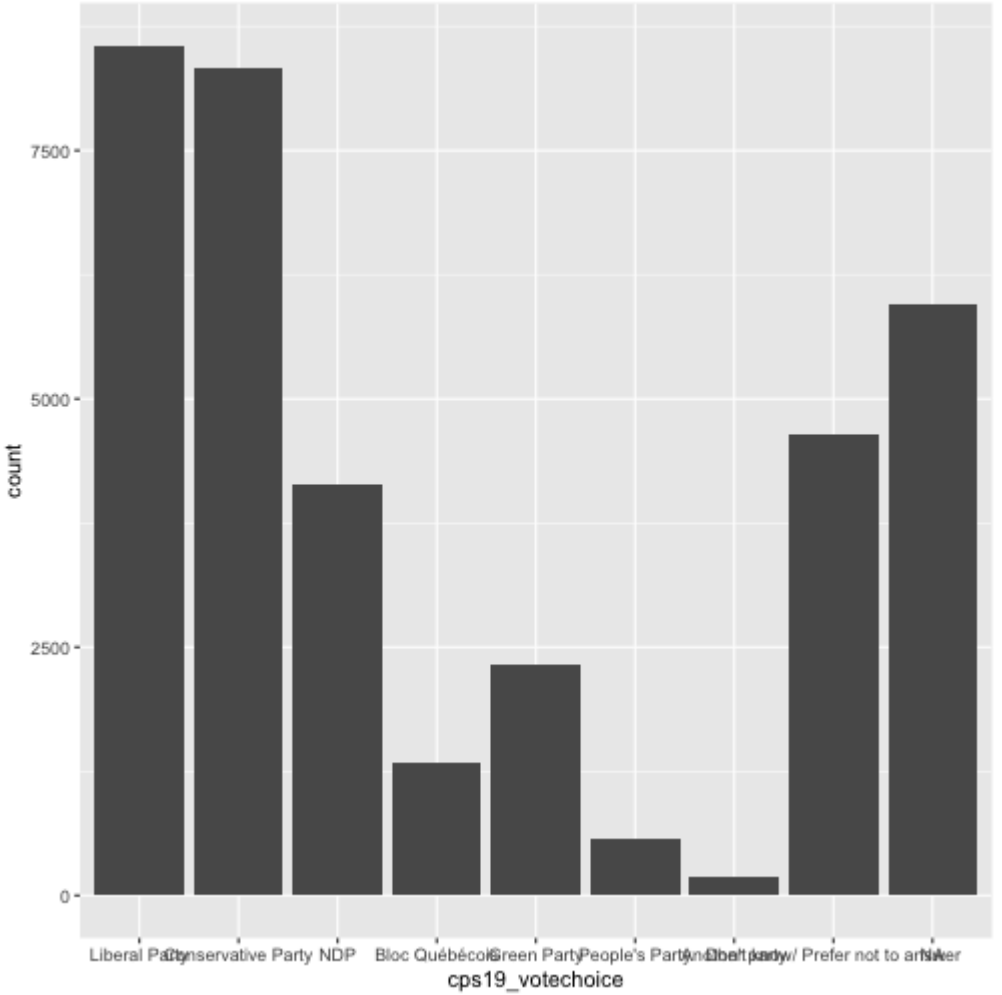


Example: Graphing CES data

We can make a bar graph representing the response to the question: "Which party do you think you will vote for?", named `cps19_votechoice`.

```
CES_data %>%
  ggplot() +
  geom_bar(aes(x = cps19_votechoice))
```

Bar graphs

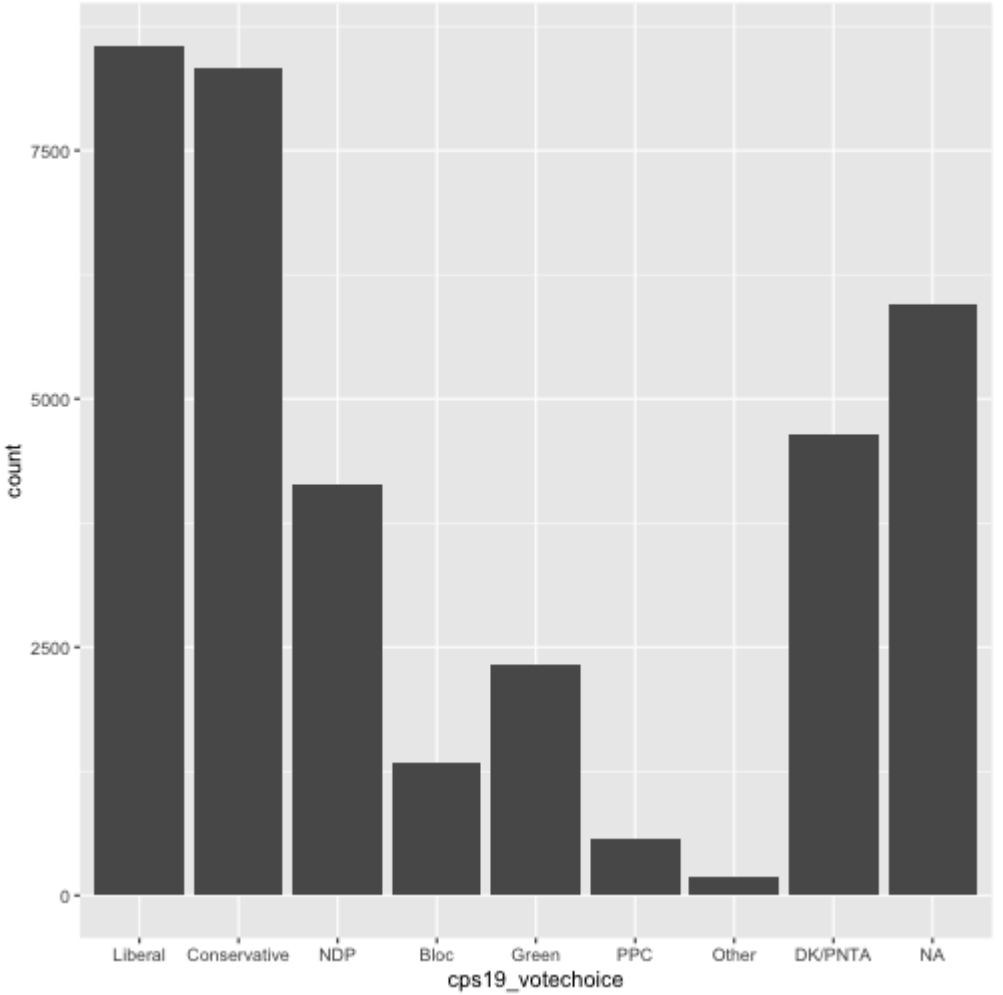

7500
5000
2500
0
count
Liberal Party
Conservative Party
NDP
Bloc Québécois
Green Party
People's Party
Don't know/ Prefer not to answer
cps19_votechoice

Bar graphs

You may want to use more short forms in the responses to make the graph more readable. To manipulate the axis labels, we use the `scale_x_discrete` function and specify what labels we want:

```
CES_data %>%
  ggplot() +
  geom_bar(aes(x = cps19_votechoice)) +
  scale_x_discrete(labels = c(
    "Liberal Party" = "Liberal",
    "Conservative Party" = "Conservative",
    "Bloc Québécois" = "Bloc",
    "Green Party" = "Green",
    "People's Party" = "PPC",
    "Another party" = "Other",
    "Don't know/ Prefer not to answer" = "DK/PNTA")
  )
```

Bar graphs

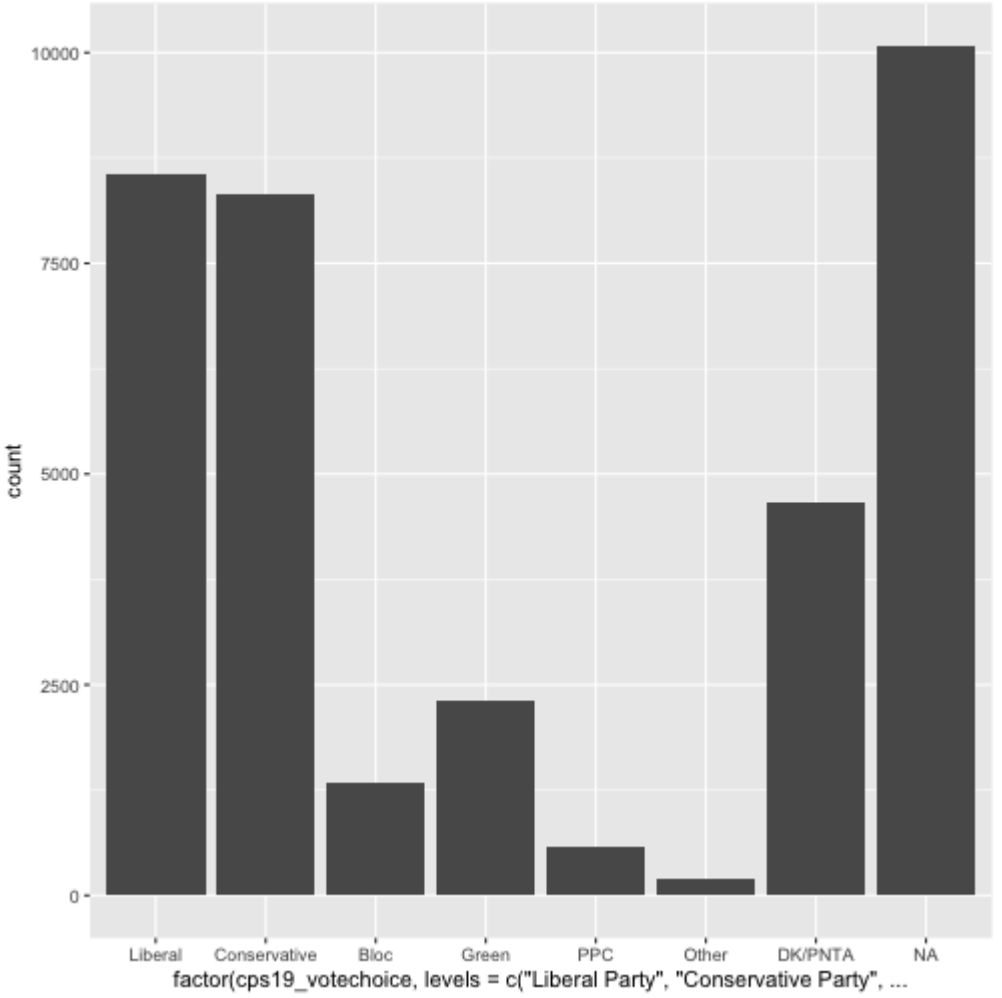

7500
5000
2500
0
count
Liberal
Conservative
NDP
Bloc
Green
PPC
Other
DK/PNTA
NA
cps19_votechoice

Bar graphs

We can also reorder bars from in a way that makes more sense. To do this, we take the variable and make it into a factor. Factors have a specific order, given in the `levels` argument:

```
CES_data %>%
  ggplot() +
  geom_bar(aes(x = factor(cps19_votechoice,
                          levels = c("Liberal Party",
                                     "Conservative Party",
                                     "Bloc Québécois",
                                     "Green Party",
                                     "People's Party",
                                     "Another party",
                                     "Don't know/ Prefer not to answer")
  scale_x_discrete(labels = c(
    "Liberal Party" = "Liberal",
    "Conservative Party" = "Conservative",
    "Bloc Québécois" = "Bloc",
    "Green Party" = "Green",
    "People's Party" = "PPC",
    "Another party" = "Other",
    "Don't know/ Prefer not to answer" = "DK/PNTA")
  )
```

Bar graphs

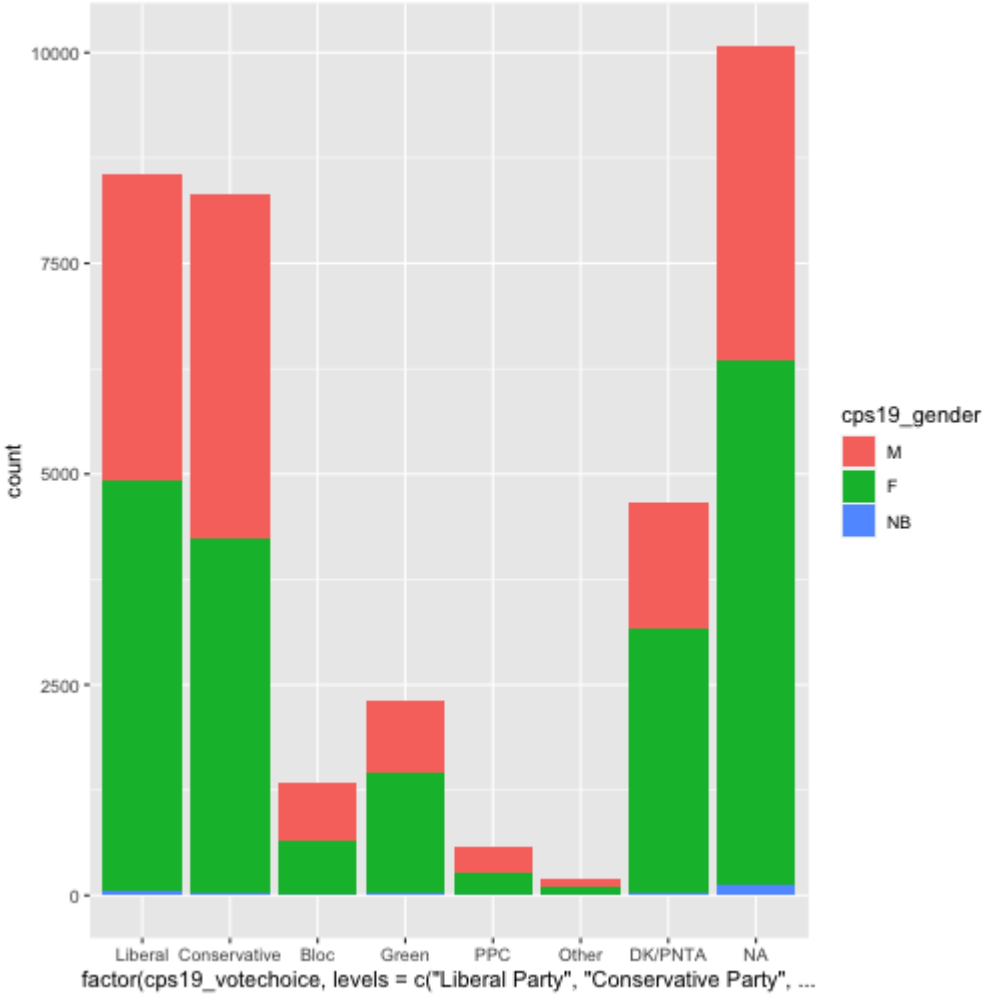

10000
7500
5000
2500
0
count
Liberal
Conservative
Bloc
Green
PPC
Other
DK/PNTA
NA
factor(cps19_votechoice, levels = c("Liberal Party", "Conservative Party", ...

Bar graphs

What if we want to compare the voting intentions between genders? We can use the `fill` argument in the `aes()` function to do that:

```
CES_data %>%
  ggplot() +
  geom_bar(aes(x = factor(cps19_votechoice,
                          levels = c("Liberal Party",
                                     "Conservative Party",
                                     "Bloc Québécois",
                                     "Green Party",
                                     "People's Party",
                                     "Another party",
                                     "Don't know/ Prefer not to answer")),
               fill = cps19_gender)) +
  scale_x_discrete(labels = c(
    "Liberal Party" = "Liberal",
    "Conservative Party" = "Conservative",
    "Bloc Québécois" = "Bloc",
    "Green Party" = "Green",
    "People's Party" = "PPC",
    "Another party" = "Other",
    "Don't know/ Prefer not to answer" = "DK/PNTA"
  ))
```

Bar graphs

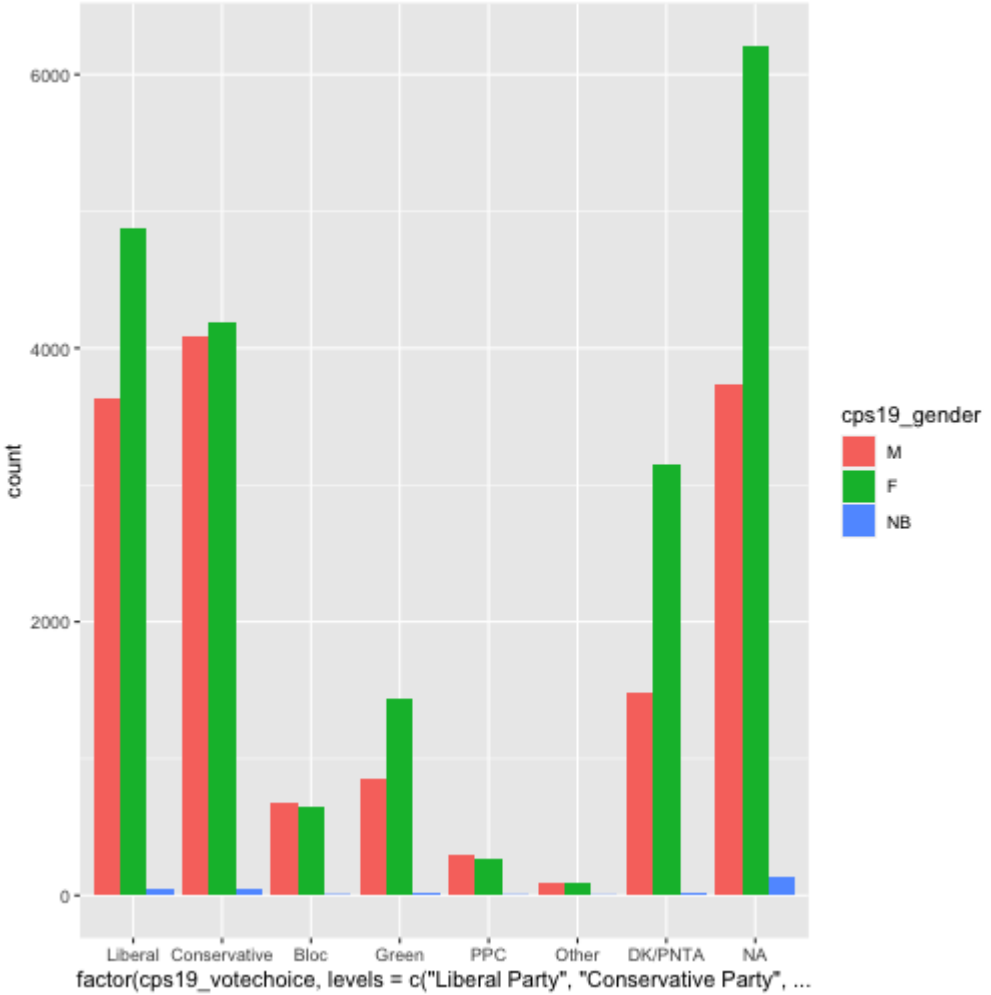

10000
7500
5000
2500
0
count
Liberal
Conservative
Bloc
Green
PPC
Other
DK/PNTA
NA
factor(cps19_votechoice, levels = c("Liberal Party", "Conservative Party", ...
cps19_gender
M
F
NB

Bar graphs

If we don't want the bars to be stacked, we need to change the `position` argument in the `geom_bar()` function:

```
CES_data %>%
  ggplot() +
  geom_bar(aes(x = factor(cps19_votechoice,
                          levels = c("Liberal Party",
                                     "Conservative Party",
                                     "Bloc Québécois",
                                     "Green Party",
                                     "People's Party",
                                     "Another party",
                                     "Don't know/ Prefer not to answer")
               fill = cps19_gender),
           position = "dodge") +
  scale_x_discrete(labels = c(
    "Liberal Party" = "Liberal",
    "Conservative Party" = "Conservative",
    "Bloc Québécois" = "Bloc",
    "Green Party" = "Green",
    "People's Party" = "PPC",
    "Another party" = "Other",
    "Don't know/ Prefer not to answer" = "DK/PNTA"
```

Bar graphs

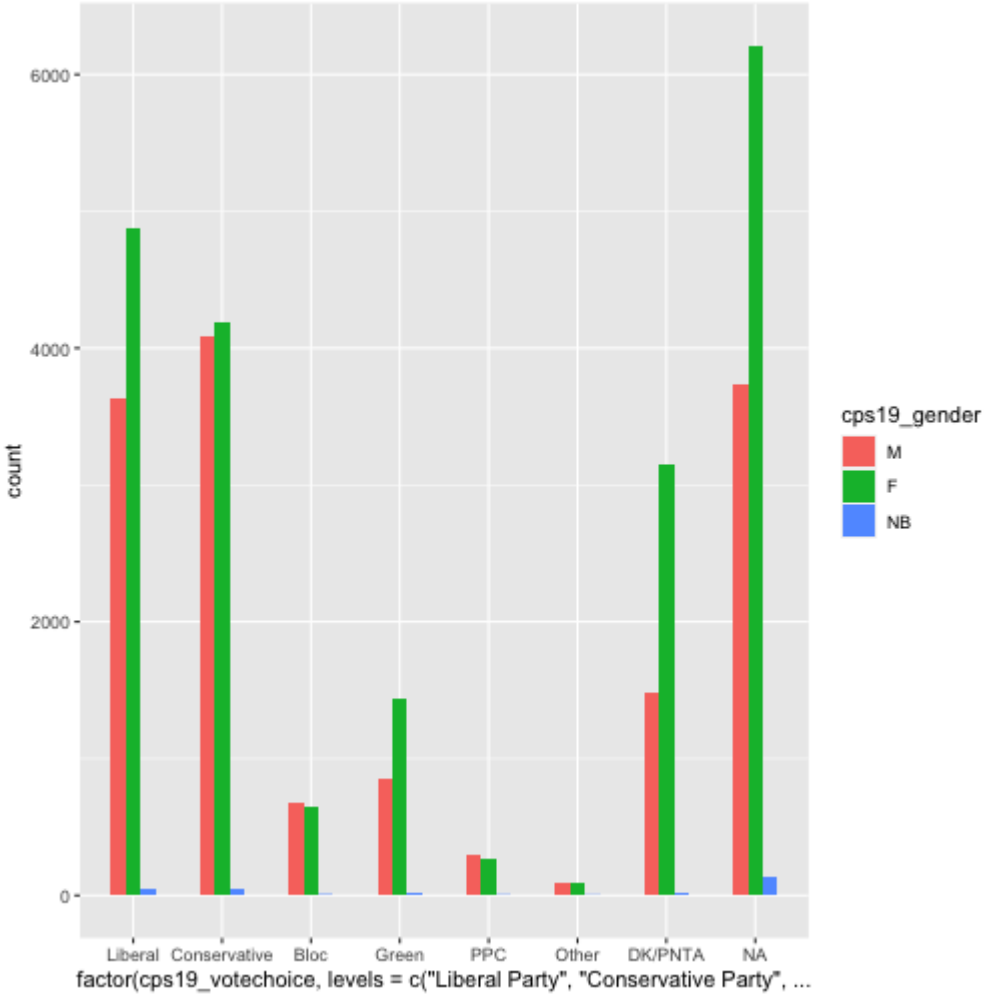

6000
4000
2000
0
count
Liberal
Conservative
Bloc
Green
PPC
Other
DK/PNTA
NA
factor(cps19_votechoice, levels = c("Liberal Party", "Conservative Party", ...
cps19_gender
M
F
NB

Bar graphs

We can change the widths of the bars as well:

```
CES_data %>%
  ggplot() +
  geom_bar(aes(x = factor(cps19_votechoice,
                          levels = c("Liberal Party",
                                     "Conservative Party",
                                     "Bloc Québécois",
                                     "Green Party",
                                     "People's Party",
                                     "Another party",
                                     "Don't know/ Prefer not to answer")
               fill = cps19_gender),
           position = "dodge",
           width = 0.5) +
  scale_x_discrete(labels = c(
    "Liberal Party" = "Liberal",
    "Conservative Party" = "Conservative",
    "Bloc Québécois" = "Bloc",
    "Green Party" = "Green",
    "People's Party" = "PPC",
    "Another party" = "Other",
    "Don't know/ Prefer not to answer" = "DK/PNTA"
```

Bar graphs


6000
4000
2000
0
count
Liberal
Conservative
Bloc
Green
PPC
Other
DK/PNTA
NA
factor(cps19_votechoice, levels = c("Liberal Party", "Conservative Party", ...
cps19_gender
M
F
NB

Histograms

We can make a histogram representing the response to the question: "How do you feel about the federal political parties below? Set the slider to a number from 0 to 100, where 0 means you really dislike the party and 100 means you really like the party." and the Conservative Party, named `cps19_party_rating_24`.

```
CES_data %>%
  ggplot() +
  geom_histogram(aes(x = cps19_party_rating_24))
```

Histograms

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 2659 rows containing non-finite values (stat_bin).
```

Histograms

The histogram splits the range of values for Conservative Party rating into 30 bins automatically, but what if we want a different number of bins? We can change the `bins` argument in the `geom_histogram()` function:

```
CES_data %>%
  ggplot() +
  geom_histogram(aes(x = cps19_party_rating_24),
                 bins = 10)
```

Histograms

```
## Warning: Removed 2659 rows containing non-finite values (stat_bin).
```

Histograms

If we want to look at distributions of a variable in different groups, we can use something called faceting. To show what the Conservative Party ratings look like across different views on education spending, we can add the `facet_wrap` function:

```
CES_data %>%
  ggplot() +
  geom_histogram(aes(x = cps19_party_rating_24),
                 bins = 10) +
  facet_wrap(facets = "cps19_spend_educ")
```

Histograms

```
## Warning: Removed 2659 rows containing non-finite values (stat_bin).
```

Histograms

We can use the `nrow` arguments to say how many rows we want the facets to form:

```
CES_data %>%
  ggplot() +
  geom_histogram(aes(x = cps19_party_rating_24),
                 bins = 10) +
  facet_wrap(facets = "cps19_spend_educ",
             nrow = 1)
```

Histograms

```
## Warning: Removed 2659 rows containing non-finite values (stat_bin).
```

Histograms

The order of the facets could be better. We use the same `factor` and `levels` method as before:

```
CES_data %>%
  ggplot() +
  geom_histogram(aes(x = cps19_party_rating_24),
                 bins = 10) +
  facet_wrap(~factor(cps19_spend_educ,
                     levels = c("Spend less",
                                "Spend about the same as now",
                                "Spend more",
                                "Don't know/ Prefer not to answer")),
             nrow = 1)
```

Histograms

```
## Warning: Removed 2659 rows containing non-finite values (stat_bin).
```

Histograms

For any graph, we probably want better labels than the variable names. We add the `labels()` function to specific labels for the x-axis, the y-axis, and the title:

```
CES_data %>%
  ggplot() +
  geom_histogram(aes(x = cps19_party_rating_24),
                 bins = 10) +
  facet_wrap(~factor(cps19_spend_educ,
                     levels = c("Spend less",
                                "Spend about the same as now",
                                "Spend more",
                                "Don't know/ Prefer not to answer")),
             nrow = 1) +
  labs(x = "Conservative Party rating",
       y = "Count",
       title = "Views on education spending and the Conservative Party")
```

Histograms

```
## Warning: Removed 2659 rows containing non-finite values (stat_bin).
```

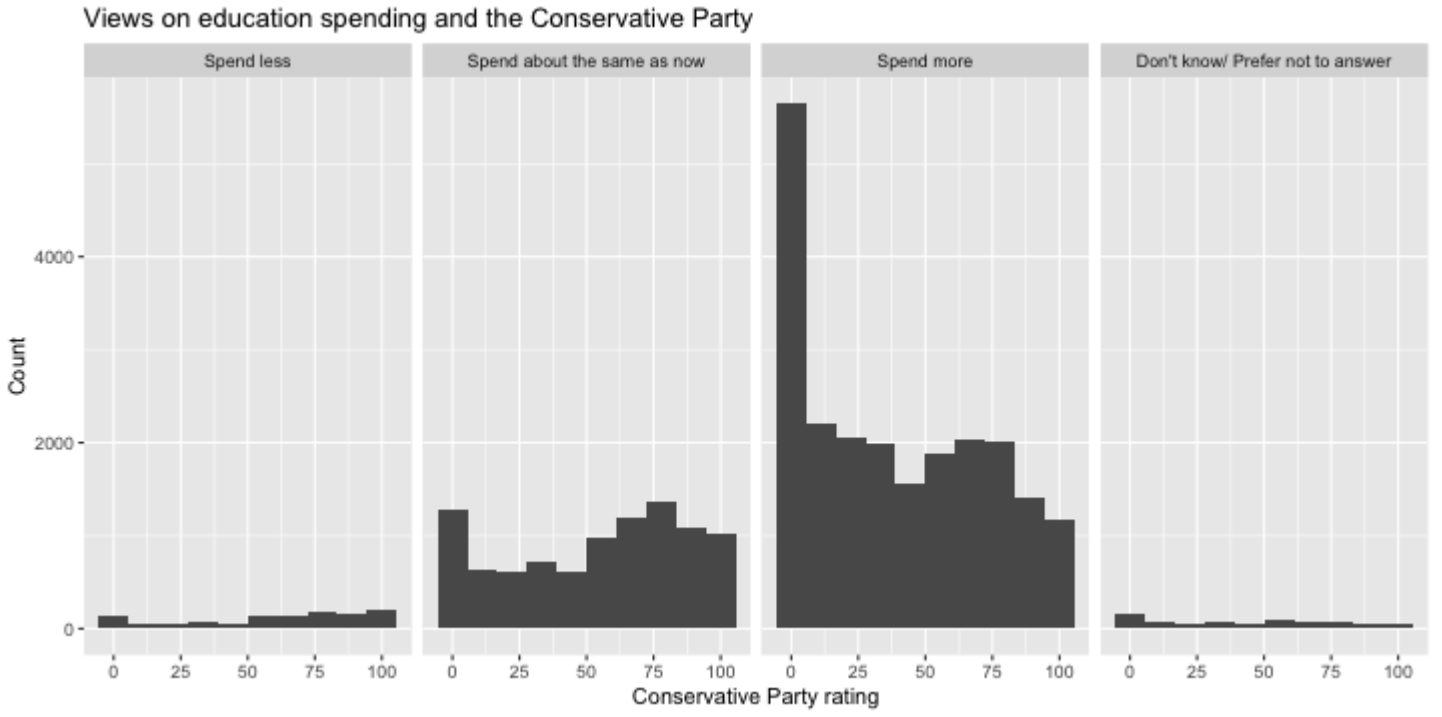
Histograms

We can widen the graph by editing the code chunk, specifying `fig.height` and `fig.width`:

```
CES_data %>%
  ggplot() +
  geom_histogram(aes(x = cps19_party_rating_24),
                 bins = 10) +
  facet_wrap(~factor(cps19_spend_educ,
                     levels = c("Spend less",
                                "Spend about the same as now",
                                "Spend more",
                                "Don't know/ Prefer not to answer")),
             nrow = 1) +
  labs(x = "Conservative Party rating",
       y = "Count",
       title = "Views on education spending and the Conservative Party")
```

Histograms

```
## Warning: Removed 2659 rows containing non-finite values (stat_bin).
```

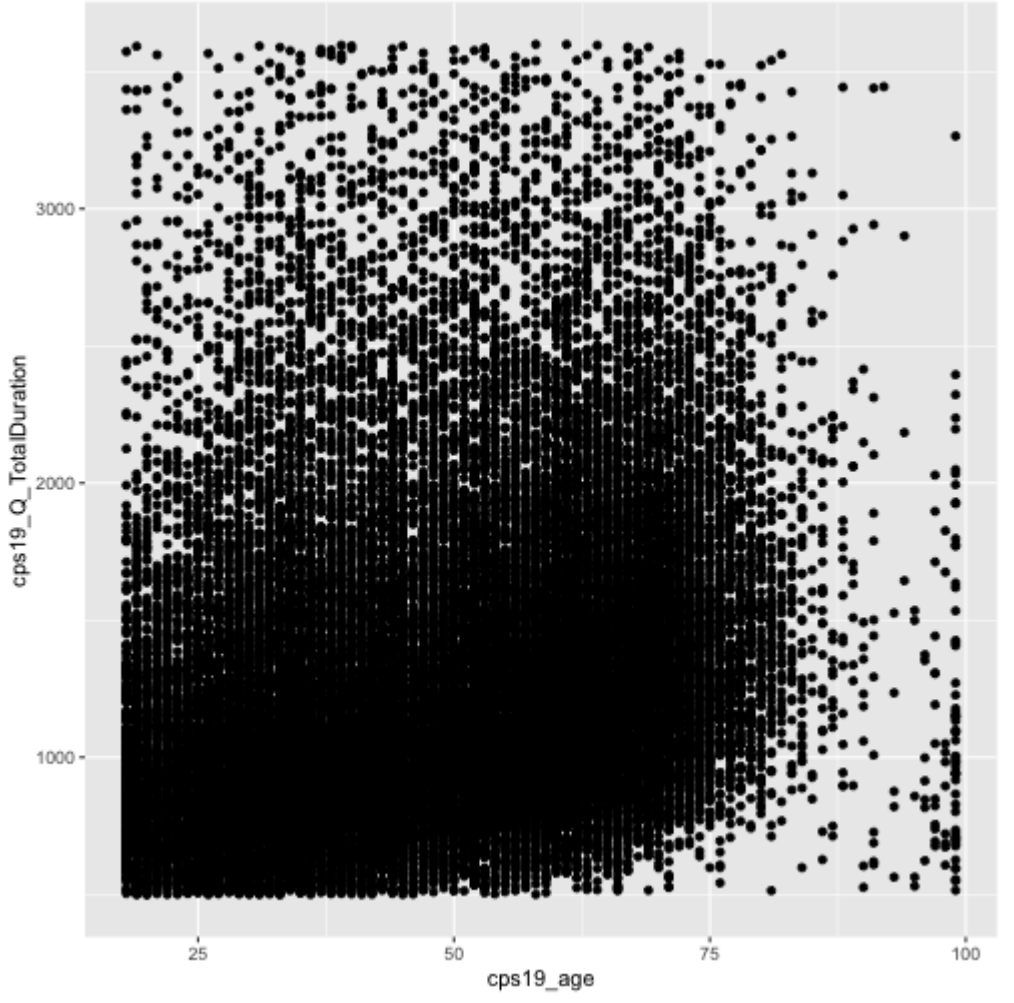


Scatterplots

We can make a scatterplot representing the relationship between the ages of the survey-takers and the time they spent on the survey, named `cps19_age` and `cps19_Q_TotalDuration`, using `geom_point()`:

```
CES_data %>%
  ggplot() +
  geom_point(aes(x = cps19_age,
                 y = cps19_Q_TotalDuration))
```

Scatterplots



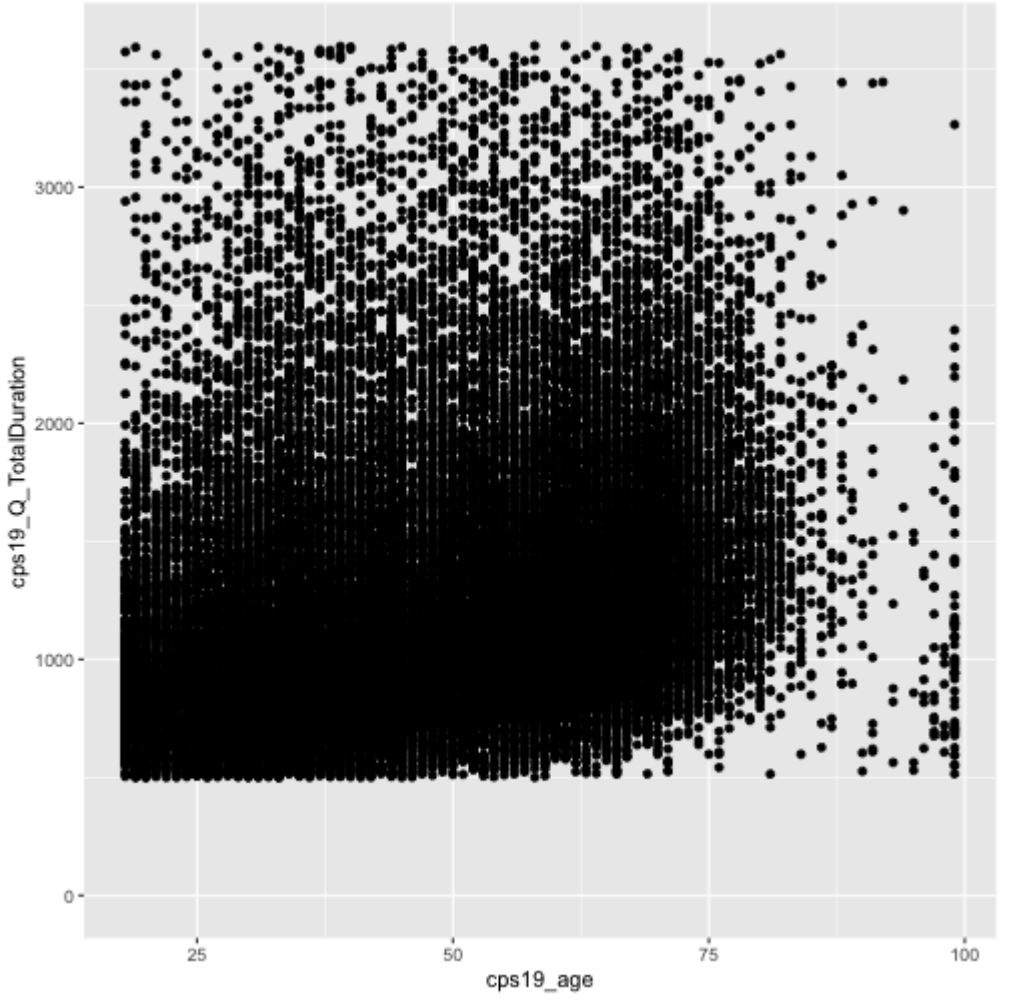
Scatterplots

There are some very large values for time spent on survey that makes it hard to see the rest. We can look at only the values in between 0 seconds and 3600 seconds, or 1 hour. Those that took longer are considered to be 'inattentive'.

One way to do this is to set limits on the axis using `scale_y_continuous()`:

```
CES_data %>%
  ggplot() +
  geom_point(aes(x = cps19_age,
                 y = cps19_Q_TotalDuration)) +
  scale_y_continuous(limits = c(0,3600))
```

Scatterplots



Scatterplots

Many points sitting exactly on top of each other, like what's happening with age, makes it hard to read the graph. If we add the arguement `position = "jitter"` to the `geom_point()` function, ggplot will slightly separate points that are in exactly the same spot:

```
CES_data %>%
  ggplot() +
  geom_point(aes(x = cps19_age,
                 y = cps19_Q_TotalDuration),
             position = "jitter") +
  scale_y_continuous(limits = c(0,3600))
```

Scatterplots

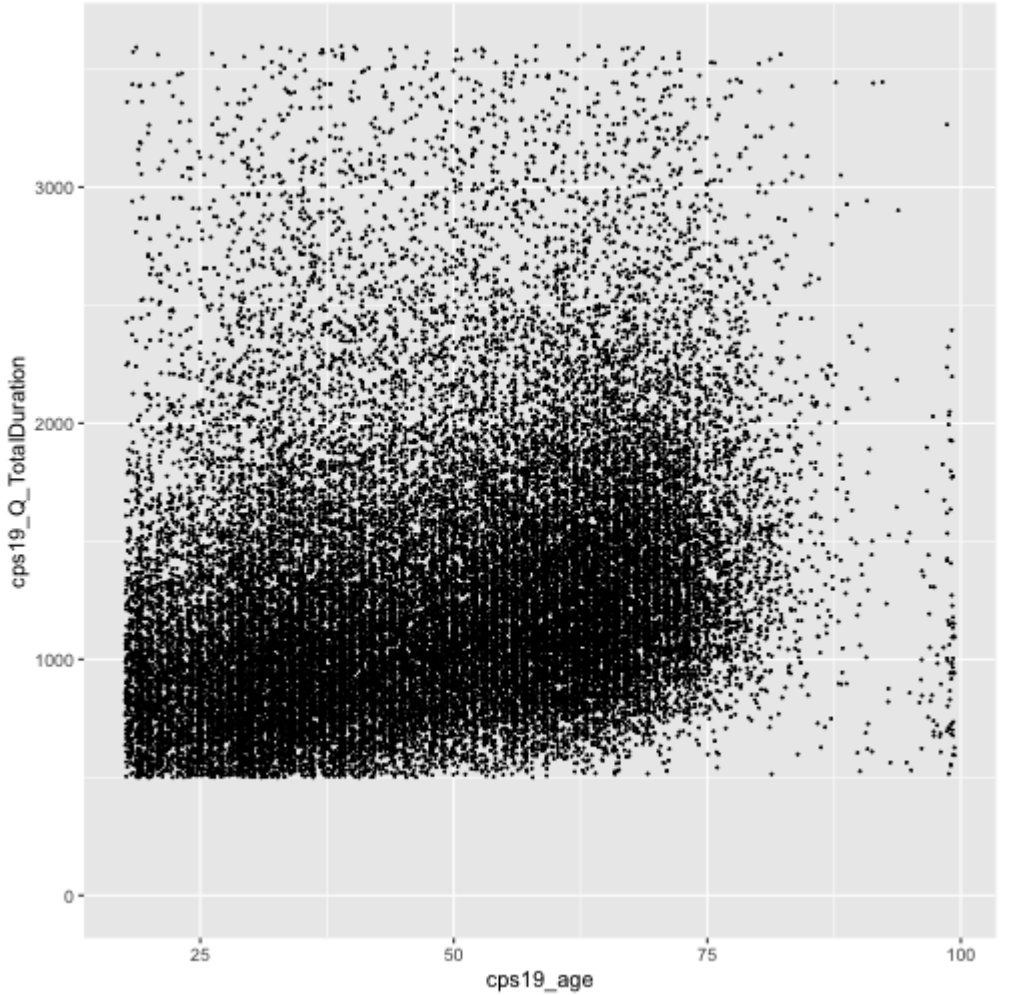
Many points sitting exactly on top of each other, like what's happening with age, makes it hard to read the graph. If we add the arguement `position = "jitter"` to the `geom_point()` function, ggplot will slightly separate points that are in exactly the same spot:

Scatterplots

Now it's just formed a cloud. To get separation, we can decrease the size of the individual points with the `size` argument in `geom_point()`:

```
CES_data %>%
  ggplot() +
  geom_point(aes(x = cps19_age,
                 y = cps19_Q_TotalDuration),
             position = "jitter",
             size = 0.1) +
  scale_y_continuous(limits = c(0,3600))
```

Scatterplots

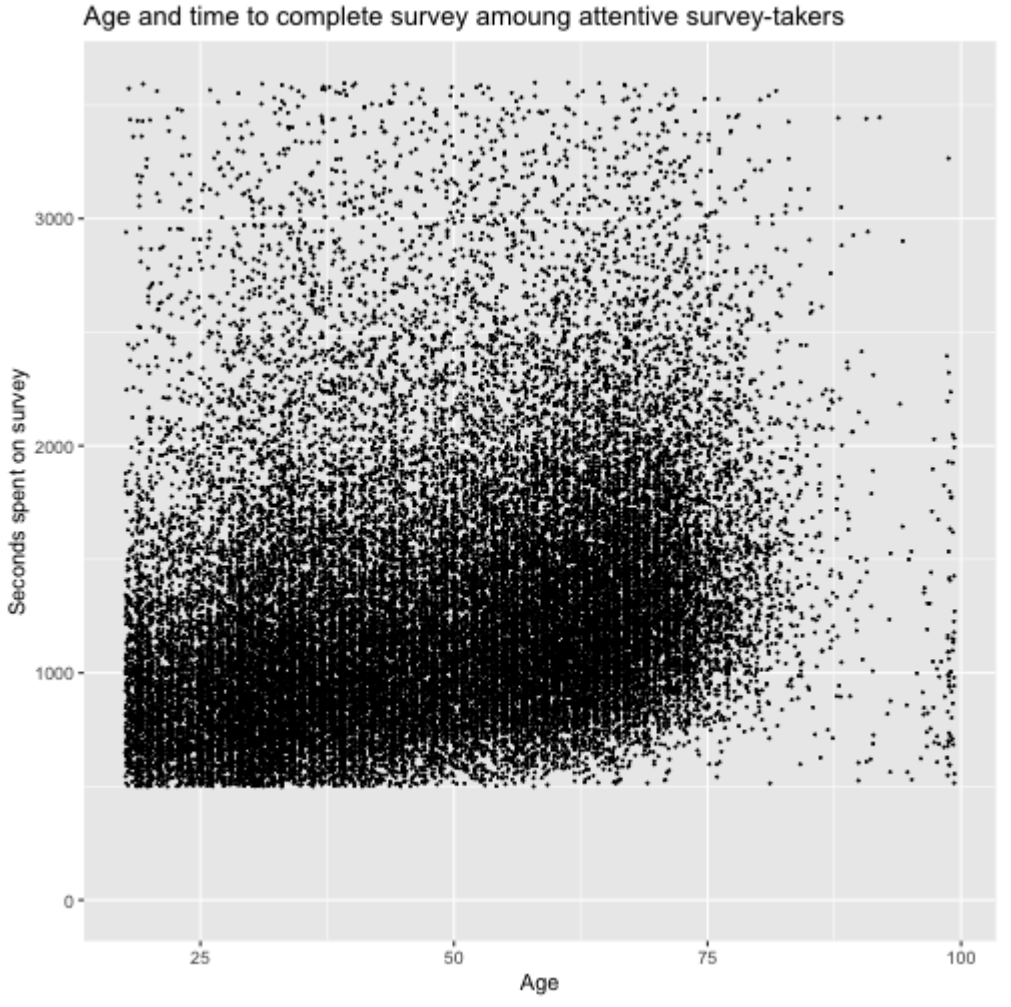


Scatterplots

Again, we can add labels:

```
CES_data %>%
  ggplot() +
  geom_point(aes(x = cps19_age,
                 y = cps19_Q_TotalDuration),
             position = "jitter",
             size = 0.1) +
  scale_y_continuous(limits = c(0,3600)) +
  labs(x = "Age",
       y = "Seconds spent on survey",
       title = "Age and time to complete survey amoung attentive survey-
```

Scatterplots



Exercises

Exercises

Take your:

1. barplots,
2. histograms, and
3. scatterplot from before

and customize them. Try to add each different customization to at least one plot.

Any questions?