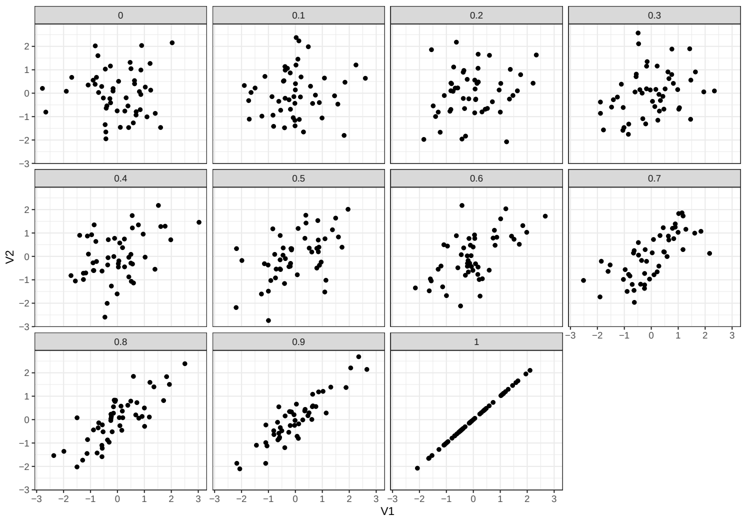
Measurement and visualization, 2

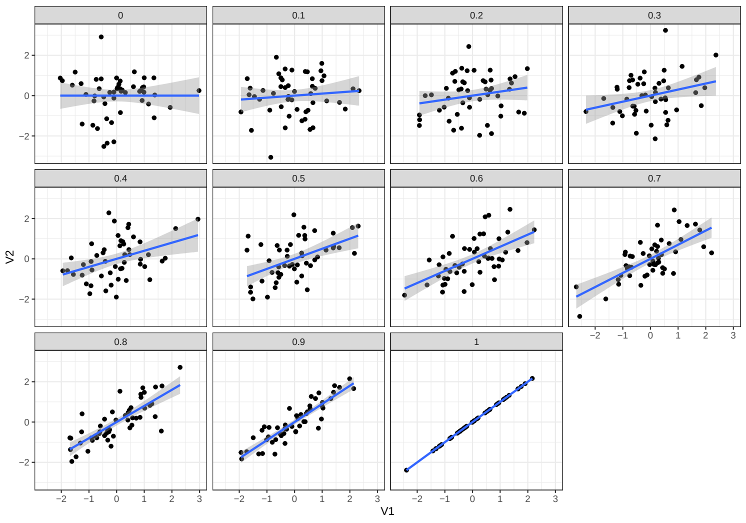
Frank Edwards

10/26/2021

Correlation



Correlation



Correlation (math time): Z-scores

First, we need the variables to be comparable, so we transform them to be on a standard deviation scale.

A *z*-score scales a variable measures the number of standard deviations an observation is away from it's mean.

$$z \text{ score of } x_i = \frac{x_i - \bar{x}}{S_x}$$

Where $\bar{x}$ is the mean, and S_x is the standard deviation of variable x. Z scores have a mean zero, and a range defined by the range of the data on a standard deviation scale.

For a normally (Gaussian) distributed variable, this will typically range between $[-3, 3]$

In R, we can transform a numeric into a *z*-score using `scale()`

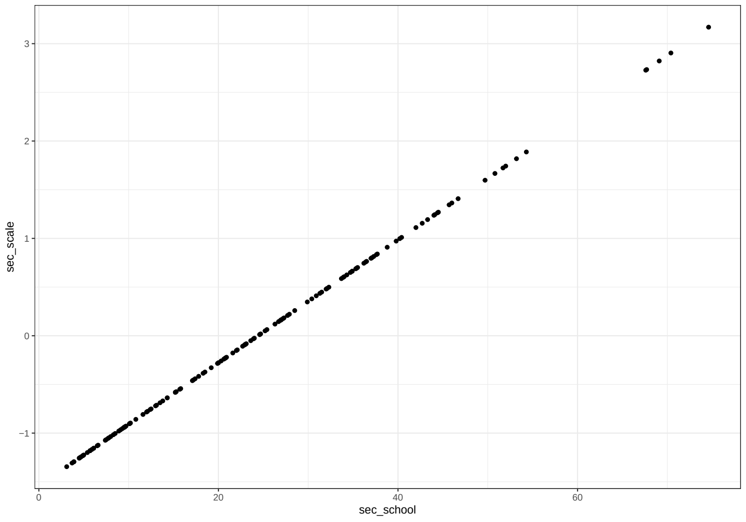
Z-scores in R

```
ipv<-read.csv("https://raw.githubusercontent.com/f-edwards/intro_st

ipv_scale<-ipv %>%
  mutate(sec_scale = scale(sec_school)) %>%
  select(sec_school, sec_scale)
summary(ipv_scale)
```

```
##    sec_school       sec_scale.V1
##  Min.   : 3.10   Min.   :-1.345006
##  1st Qu.:10.18   1st Qu.:-0.898292
##  Median :22.40   Median :-0.126408
##  Mean   :24.40   Mean   : 0.000000
##  3rd Qu.:34.90   3rd Qu.: 0.662840
##  Max.   :74.60   Max.   : 3.169492
##  NA's   :3       NA's   :3
```

Z-scores in R

```
ggplot(ipv_scale, aes(x=sec_school, y=sec_scale)) + geom_point()
```



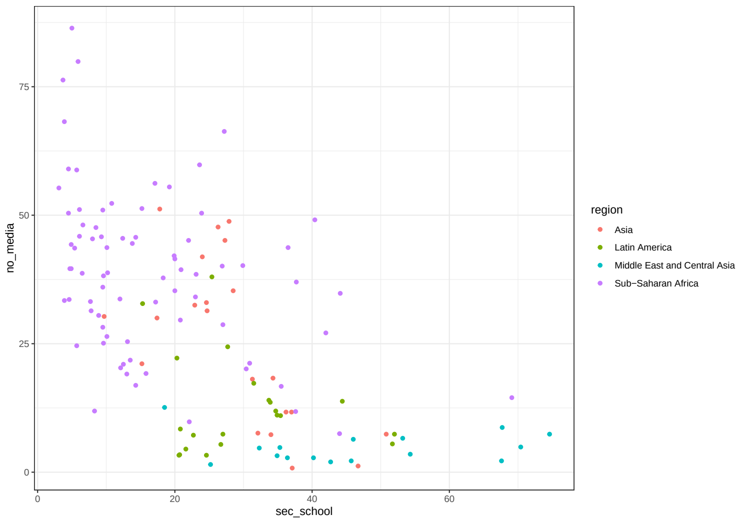
Correlation measures the degree to which two variables are associated with each other. We often use the letter r to denote a correlation.

$$r(x, y) = \frac{1}{n} \sum_{i=1}^{n} \frac{x_i - \bar{x}}{S_x} \times \frac{y_i - \bar{y}}{S_y}$$

Note that this is equal to the average of the product of the $z - scores$ of x and y

In R, you can use `cor()`

Returning to our example: Are sec_school and no_media correlated?



Obtaining the correlation coefficient

```
cor(ipv$sec_school, ipv$no_media, use="complete")
```

```
## [1] -0.6077951
```

```
## z score method
mean(scale(ipv$sec_school) * scale(ipv$no_media), na.rm=TRUE)
```

```
## [1] -0.6084724
```

Clustering

Latent structure

Data often *cluster* based on unobserved or unobservable characteristics. We can use *classification methods* to try to uncover these latent structures in data.

k-means is a straightforward method we can use to identify k latent groupings in our data, based on proximity of observations for specified variables.

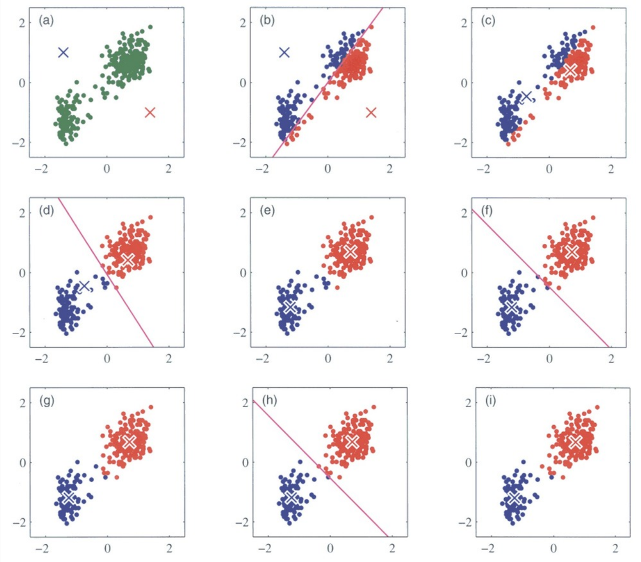
An algorithm is a sequential set of steps used to solve a problem.

A *centroid* is the mean value of a cluster within a group.

1. Choose the initial centroids for each of the k clusters
2. Assign each observation to the cluster with the nearest centroid
3. Assign a new centroid based on the within-cluster mean for assigned observations
4. Repeat steps 2 and 3 until the cluster assignments no longer change

We arbitrarily choose the number of clusters k, and R randomly selects starting centroid values for step 1.

The k-means algorithm



Implementing k-means for the IPV data

```r
ipv_scale<-ipv %>%
  select(sec_school, no_media) %>%
  mutate(sec_school = scale(sec_school),
         no_media = scale(no_media)) %>%
  filter(!(is.na(sec_school)), !(is.na(no_media)))

ipv_kmeans<-kmeans(ipv_scale,
                   centers = 3,
                   nstart=10)
```

Working with the k-means object

```
ipv_kmeans
```

```
## K-means clustering with 3 clusters of sizes 17, 46, 72
##
## Cluster means:
##   sec_school   no_media
## 1  1.8803248 -1.1669709
## 2  0.2071354 -0.7678187
## 3 -0.6135490  0.7910351
##
## Clustering vector:
##   [1] 2 1 1 1 1 3 3 3 3 3 3 2 2 3 3 3 3 2 3 3 3 3 3 3 2 2 3 3 3 3 2 2 2 2 2 2 2
##  [38] 1 3 3 3 3 1 3 2 2 3 3 1 1 3 2 2 2 2 3 2 2 2 1 1 2 2 1 3 2 3 3 3 3 3 3 3 2
##  [75] 3 3 3 1 2 3 3 2 2 2 3 3 2 3 3 3 3 3 3 2 2 2 1 2 2 2 2 2 2 3 3 3 2 2 2 2 2
## [112] 3 3 1 3 3 3 3 3 3 3 1 3 3 3 3 2 1 1 3 3 3 3 3 3
##
## Within cluster sum of squares by cluster:
## [1]  8.657364 24.241269 52.858371
##  (between_SS / total_SS =  68.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

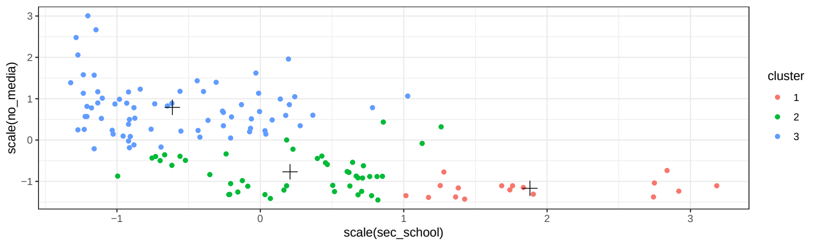
Pull out what we need from the list

```
ipv_clusters<-ipv %>%
  filter(!(is.na(sec_school)), !(is.na(no_media))) %>%
  mutate(cluster = factor(ipv_kmeans$cluster))

centers<-data.frame(ipv_kmeans$centers)

centers
```

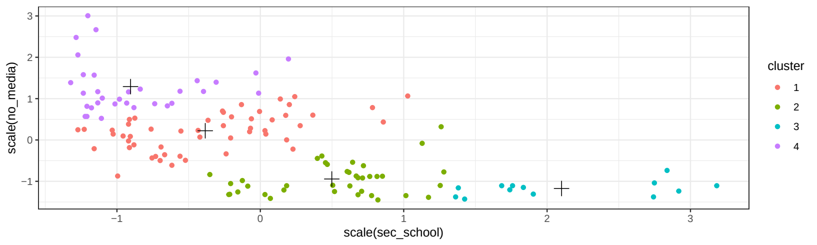
```
##    sec_school   no_media
## 1  1.8803248 -1.1669709
## 2  0.2071354 -0.7678187
## 3 -0.6135490  0.7910351
```

Plot it!

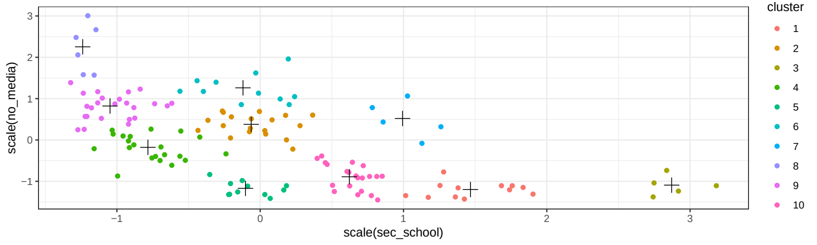
```r
ggplot(ipv_clusters, aes(x = scale(sec_school),
                         y =scale(no_media),
                         color = cluster)) +
  geom_point() +
  geom_point(data = centers,
             aes(x=sec_school,
                 y=no_media),
             color = "black", size = 4, shape = 3)
```



What if we thought there were 4 clusters?



What if we thought there were 10 clusters?



Summary

- Measurement and design matter!
- Always check your data, and think about how unit and item non-response may inform your conclusions
- Think about desirability and other forms of response bias as you interpret your results
- Design visuals and exploratory analyses to check hypotheses about what's going on in the data
- Think about the structure of your data, use descriptive statistics like correlations to describe relationships
- Think about latent structures in your data to capture clustering

Lab: more data visualization with ggplot

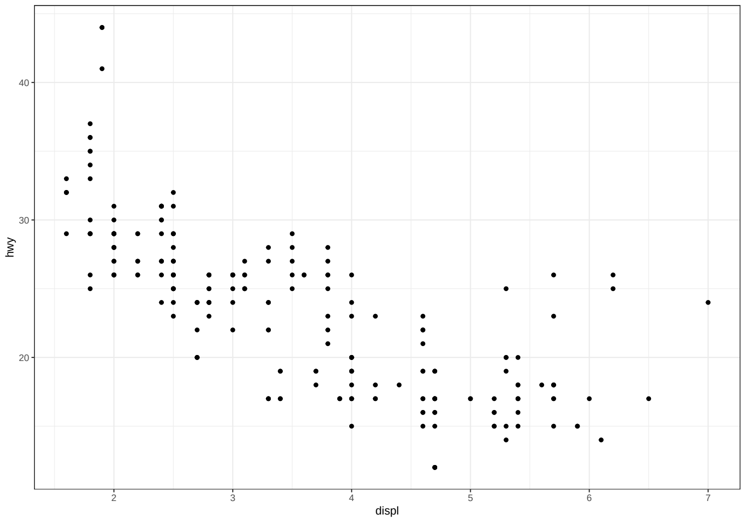
The mpg data

```
head(mpg)
```

```
## # A tibble: 6 x 11
##   manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
##   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
## 1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa~
## 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa~
## 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa~
## 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa~
## 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa~
## 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa~
```

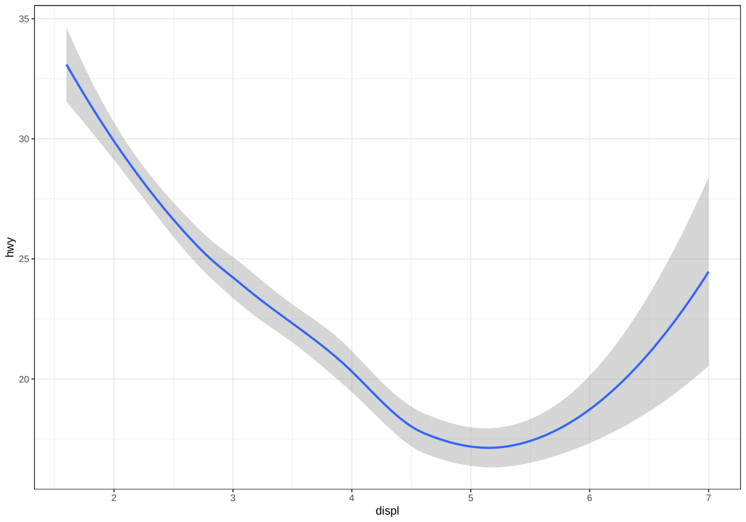
How are these plots similar?

```
ggplot(mpg,
       aes(x = displ, y = hwy)) +
  geom_point()
```



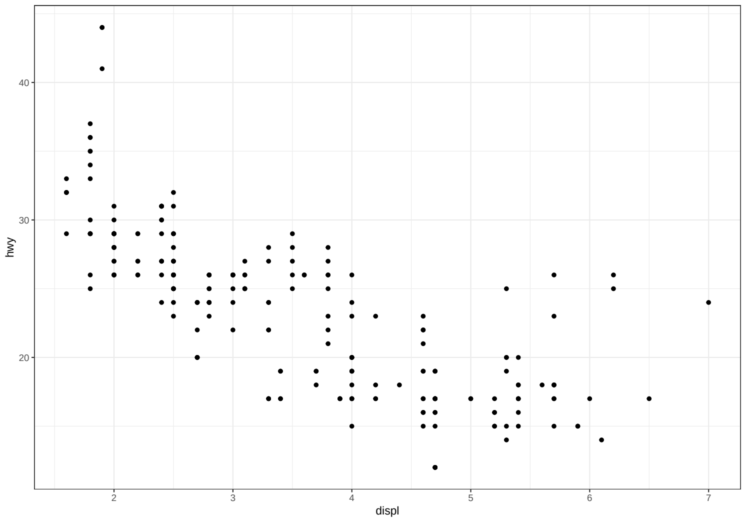
How are these plots similar?

```
ggplot(mpg,
       aes(x = displ, y = hwy)) +
  geom_smooth()
```



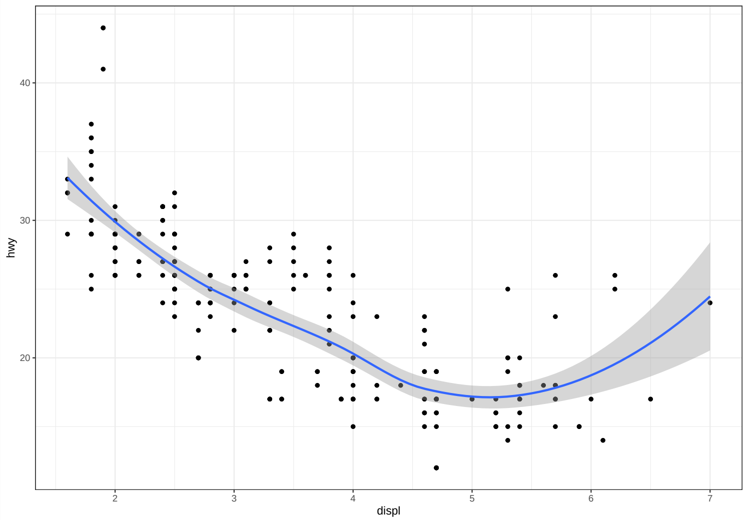
Geometric objects (geoms) map data to visual objects

```
ggplot(mpg,
       aes(x = displ, y = hwy)) +
  geom_point()
```



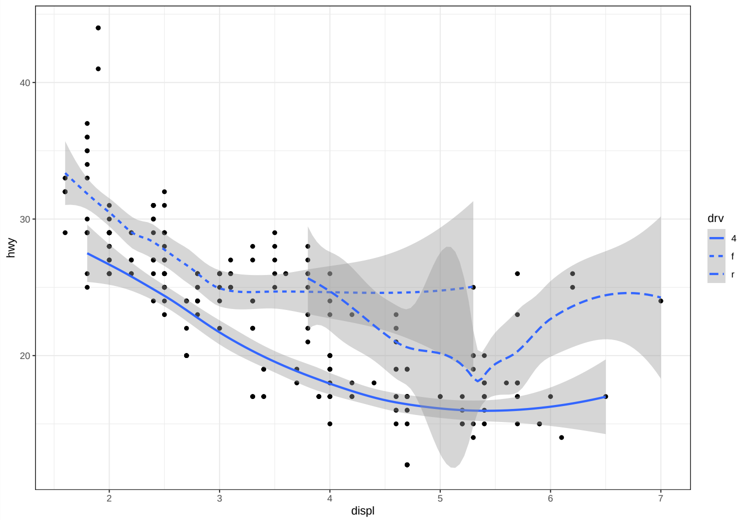
We can layer geoms

```
ggplot(mpg,
       aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```



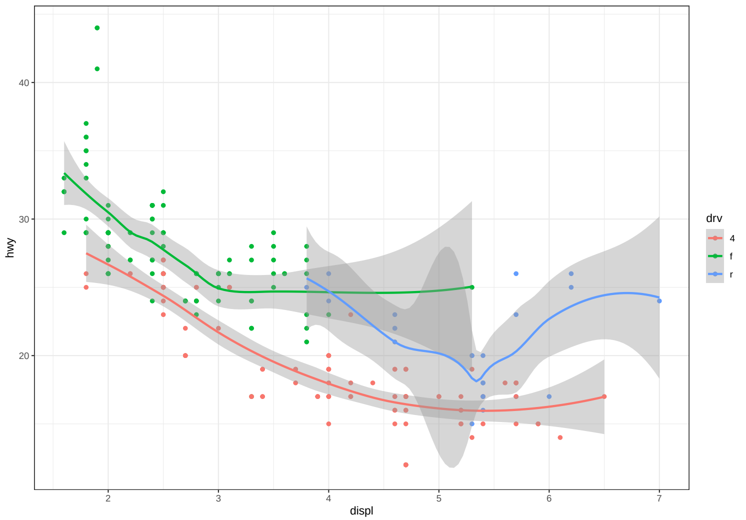
Add aesthetics to map variables to visual objects

```
ggplot(mpg,
       aes(x = displ, y = hwy, lty = drv)) +
  geom_point() +
  geom_smooth()
```



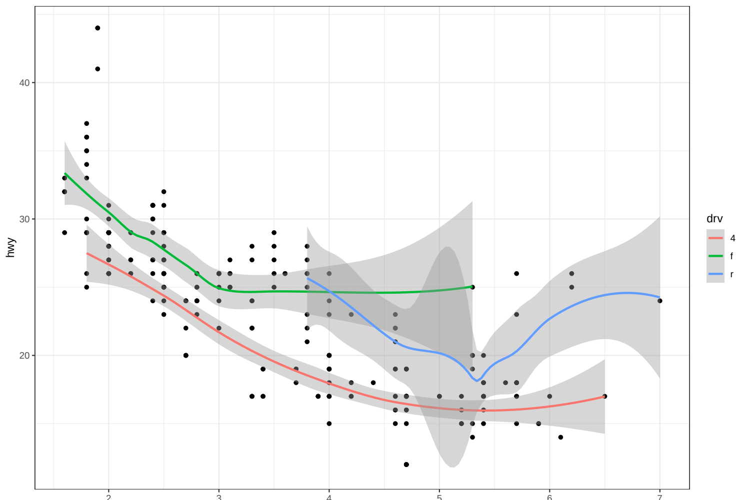
Add aesthetics to map variables to visual objects

```
ggplot(mpg,
       aes(x = displ, y = hwy, color = drv)) +
  geom_point() +
  geom_smooth()
```



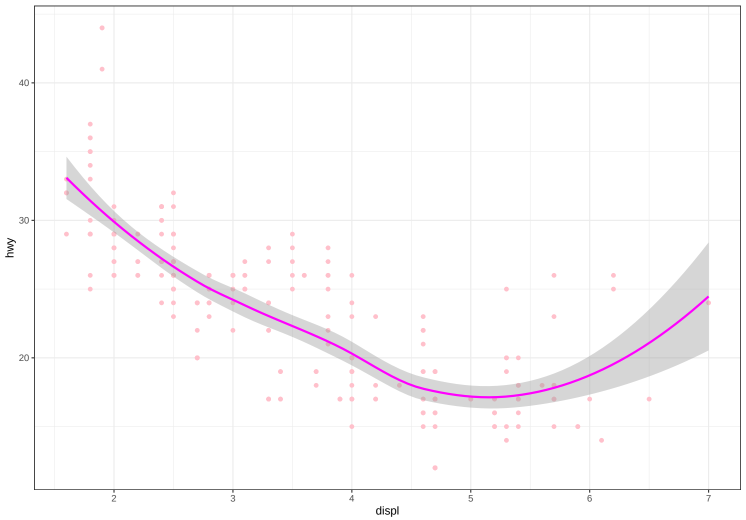
Global and local aesthetics

```
## What's the difference from the prior plot?
## could i make this more compact?
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy)) +
  geom_smooth(aes(x = displ, y = hwy, color = drv))
```



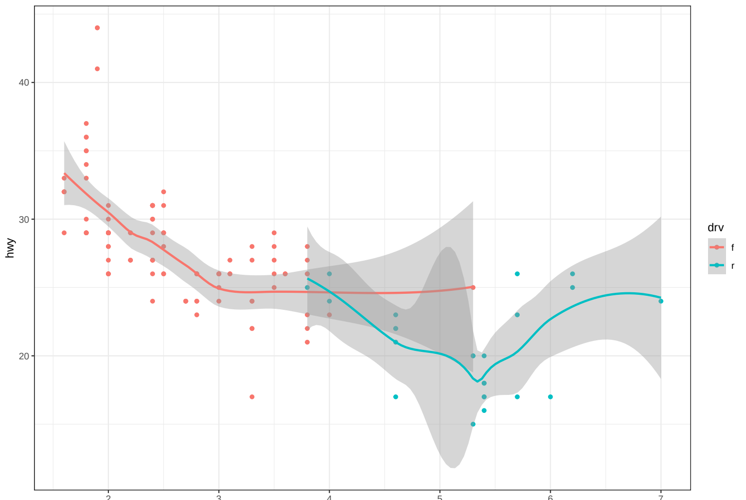
Modifying visual objects without variable mapping

```
ggplot(mpg,
       aes(x = displ, y = hwy)) +
  geom_point(color = "pink") +
  geom_smooth(color = "magenta")
```



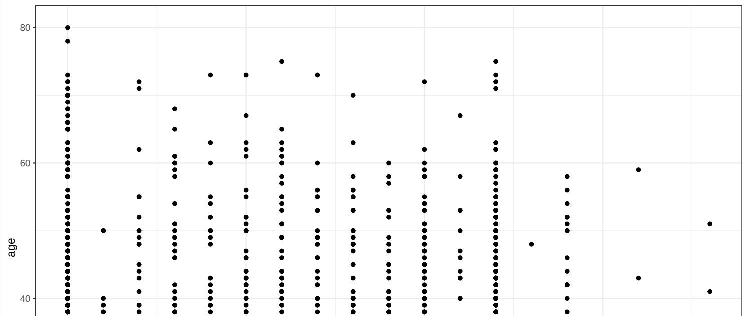
Modifying data prior to plotting

```
ggplot(mpg %>%
         filter(drv !="4"),
       aes(x = displ, y = hwy, color = drv)) +
  geom_point() +
  geom_smooth()
```



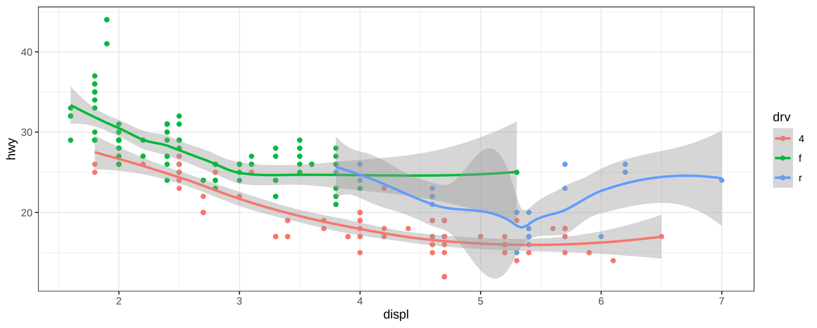
- Using the `afghan` data, visualize the relationship between age and educ.years. What is the best geom for examining this relationship?

```
afghan<-read_csv("https://raw.githubusercontent.com/kosukeimai/qss/

ggplot(afghan,
       aes(y = age, x = educ.years)) +
  geom_point()
```



This is not pretty

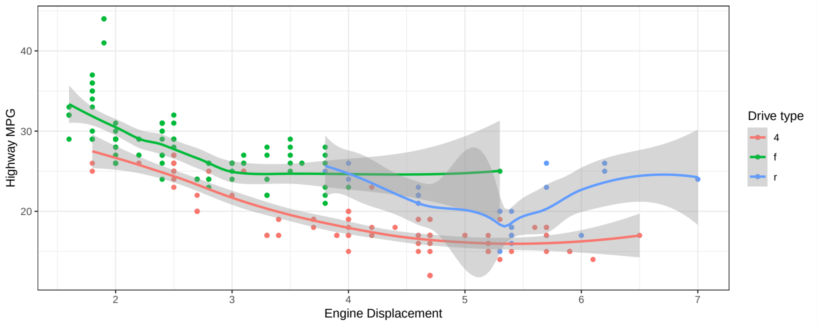
```
ggplot(mpg,
       aes(x = displ, y = hwy, color = drv)) +
  geom_point() +
  geom_smooth()
```



Axis labels

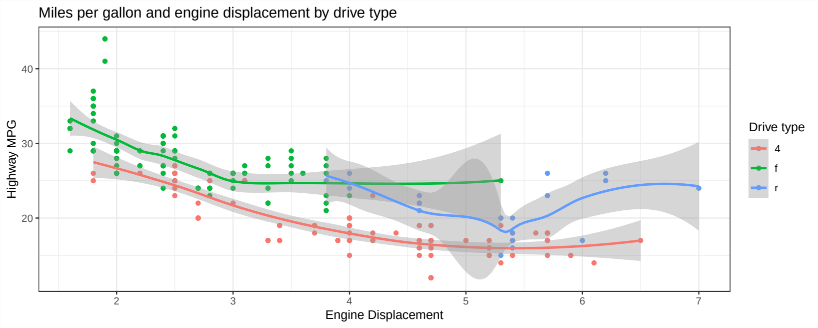
This is better!

```
ggplot(mpg,
       aes(x = displ, y = hwy, color = drv)) +
  geom_point() +
  geom_smooth() +
  labs(x = "Engine Displacement",
       y = "Highway MPG",
       color = "Drive type")
```



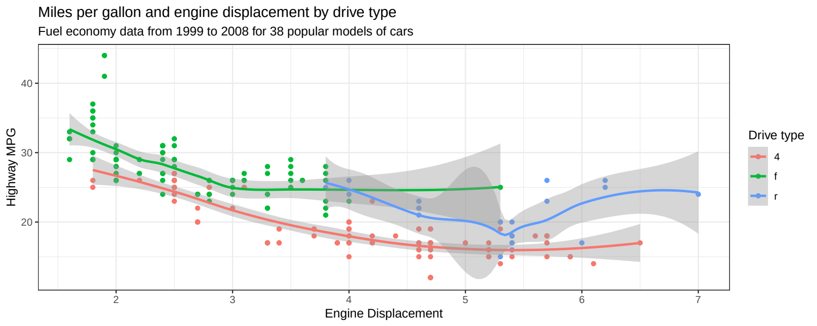
Titles

```
ggplot(mpg,
       aes(x = displ, y = hwy, color = drv)) +
  geom_point() +
  geom_smooth() +
  labs(x = "Engine Displacement",
       y = "Highway MPG",
       color = "Drive type",
       title = "Miles per gallon and engine displacement by drive type")
```



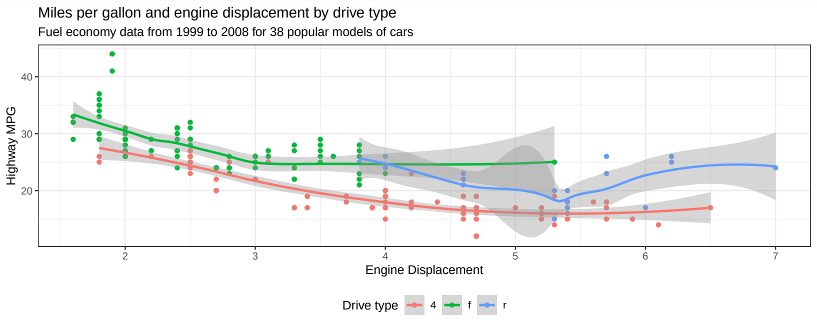
Subtitles

```
ggplot(mpg,
       aes(x = displ, y = hwy, color = drv)) +
  geom_point() +
  geom_smooth() +
  labs(x = "Engine Displacement",
       y = "Highway MPG",
       color = "Drive type",
       title = "Miles per gallon and engine displacement by drive type",
       subtitle = "Fuel economy data from 1999 to 2008 for 38 popular models of cars")
```



Moving things around with theme

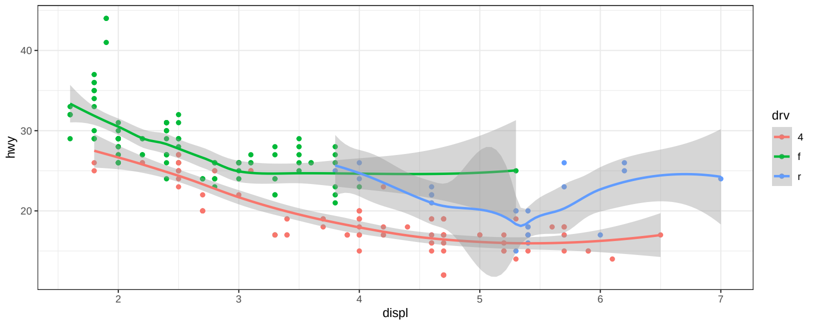
```
ggplot(mpg,
       aes(x = displ, y = hwy, color = drv)) +
  geom_point() +
  geom_smooth() +
  labs(x = "Engine Displacement",
       y = "Highway MPG",
       color = "Drive type",
       title = "Miles per gallon and engine displacement by drive type",
       subtitle = "Fuel economy data from 1999 to 2008 for 38 popular models of cars") +
  theme(legend.position = "bottom")
```



If two vehicles have identical MPG and displ, they will overlap, and we can't actually see them. A jitter adds a small amount of noise to help us see all the data.

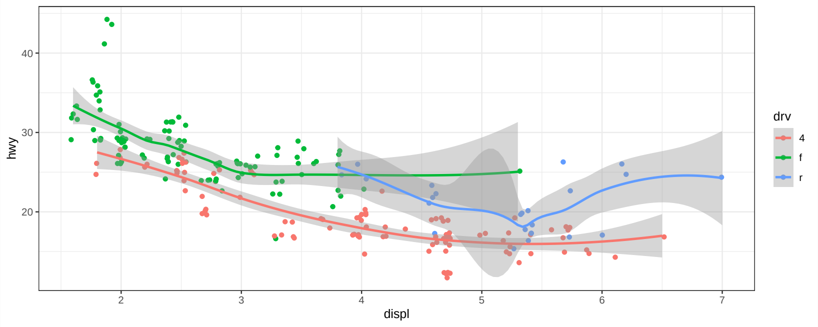
Improving scatterplots

```
ggplot(mpg,
       aes(x = displ, y = hwy, color = drv)) +
  geom_point() +
  geom_smooth()
```



Improving scatterplots

```
ggplot(mpg,
       aes(x = displ, y = hwy, color = drv)) +
  geom_jitter() +
  geom_smooth()
```



Joining data

Data for today: polling and the 2016 election

```r
polls<-read.csv("https://raw.githubusercontent.com/f-edwards/intro_stats/master/data/polls2016.csv")
## if not in the .RMD slide file
head(polls)
```

```
##      id state Clinton Trump days_to_election electoral_votes    population
## 1 26255    TX      38    41                24              38 Likely Voters
## 2 26253    WI      48    44                23              10 Likely Voters
## 3 26252    VA      54    41                23              13 Likely Voters
## 4 26251    NV      47    40                19               6 Likely Voters
## 5 26250    TX      46    48                23              38 Likely Voters
## 6 26249    NH      50    43                23               4 Likely Voters
```

```r
polls<-polls %>%
  filter(population=="Likely Voters") %>%
  select(state, Clinton, Trump, electoral_votes)
```

Data for today: election results

```
results<-read.csv("https://raw.githubusercontent.com/f-edwards/intro_stats/master/data/1976-2016-president.csv"
head(results)
```

```
##   year   state state_po state_fips state_cen state_ic       office
## 1 1976 Alabama       AL          1        63       41 US President
## 2 1976 Alabama       AL          1        63       41 US President
## 3 1976 Alabama       AL          1        63       41 US President
## 4 1976 Alabama       AL          1        63       41 US President
## 5 1976 Alabama       AL          1        63       41 US President
## 6 1976 Alabama       AL          1        63       41 US President
##                  candidate                      party writein candidatevotes
## 1           Carter, Jimmy                    democrat   FALSE         659170
## 2            Ford, Gerald                  republican   FALSE         504070
## 3          Maddox, Lester american independent party   FALSE           9198
## 4 Bubar, Benjamin ""Ben""                 prohibition   FALSE           6669
## 5                Hall, Gus        communist party use   FALSE           1954
## 6         Macbride, Roger                 libertarian   FALSE           1481
##   totalvotes  version notes
## 1    1182850 20171015    NA
## 2    1182850 20171015    NA
## 3    1182850 20171015    NA
## 4    1182850 20171015    NA
## 5    1182850 20171015    NA
## 6    1182850 20171015    NA
```

```
results<-results %>%
  filter(year==2016) %>%
  filter(candidate=="Clinton, Hillary" |
           candidate=="Trump, Donald J.") %>%
  group_by(state_po, candidate) %>%
```

Joining data frames

- We can join (or merge) two data frames together by common variables

Joining data frames

- We can join (or merge) two data frames together by common variables
- Joining variables must have identical column names, types, and values

Joining election results and election predictions

How are both datasets structured? What common variables could we join on?

```
head(polls)
```

```
##   state Clinton Trump electoral_votes
## 1    TX      38    41              38
## 2    WI      48    44              10
## 3    VA      54    41              13
## 4    NV      47    40               6
## 5    TX      46    48              38
## 6    NH      50    43               4
```

```
head(results)
```

```
## # A tibble: 6 x 3
## # Groups:   state_po [3]
##   state_po candidate        pct_vote
##   <chr>    <chr>               <dbl>
## 1 AK       Clinton, Hillary     36.6
## 2 AK       Trump, Donald J.     51.3
## 3 AL       Clinton, Hillary     34.4
## 4 AL       Trump, Donald J.     62.1
## 5 AR       Clinton, Hillary     33.7
## 6 AR       Trump, Donald J.     60.6
```

Restructuring data for join

- State abbreviation is a common column for both

Restructuring data for join

- State abbreviation is a common column for both
- Candidate is a column in results, and is spread over column names in polls

Restructuring data for join

- State abbreviation is a common column for both
- Candidate is a column in results, and is spread over column names in polls
- We want to join, such that the election results for each candidate are joined onto each poll for a state.

Restructuring data for join

- State abbreviation is a common column for both
- Candidate is a column in results, and is spread over column names in polls
- We want to join, such that the election results for each candidate are joined onto each poll for a state.
- For example, Nevada poll results for Clinton should match onto Nevada election results

Restructuring data for join

- State abbreviation is a common column for both
- Candidate is a column in results, and is spread over column names in polls
- We want to join, such that the election results for each candidate are joined onto each poll for a state.
- For example, Nevada poll results for Clinton should match onto Nevada election results
- Note that there is more than one poll available for most states, but only one election result

Rename columns to match

Rename state in polls to state_po to match across data.frames

```
polls<-polls %>%
  rename(state_po = state)
names(polls)
```

```
## [1] "state_po"        "Clinton"         "Trump"           "electoral_votes"
```

A note on tidy data

- Each column should be a variable

A note on tidy data

- Each column should be a variable
- Values should not be stored in columns

A note on tidy data

- Each column should be a variable
- Values should not be stored in columns

```
head(polls)
```

```
##   state_po Clinton Trump electoral_votes
## 1       TX      38    41              38
## 2       WI      48    44              10
## 3       VA      54    41              13
## 4       NV      47    40               6
## 5       TX      46    48              38
## 6       NH      50    43               4
```

Pivot polls from wide format to long format

- Take the candidate name from the column, add a new column for candidate

Pivot polls from wide format to long format

- Take the candidate name from the column, add a new column for candidate
- Note that this structure matches the structure of results

pivot_longer

`pivot_longer` reshapes wide data into long data. `pivot_wider` reshapes long data into wide data.

```
polls_long <- polls %>%
  pivot_longer(cols = Clinton:Trump,
               names_to = "candidate",
               values_to = "poll_result")

head(polls_long)
```

```
## # A tibble: 6 x 4
##   state_po electoral_votes candidate poll_result
##   <chr>              <int> <chr>           <int>
## 1 TX                    38 Clinton            38
## 2 TX                    38 Trump              41
## 3 WI                    10 Clinton            48
## 4 WI                    10 Trump              44
## 5 VA                    13 Clinton            54
## 6 VA                    13 Trump              41
```

Any other problems prior to joining?

```r
head(polls_long)
```

```
## # A tibble: 6 x 4
##   state_po electoral_votes candidate poll_result
##   <chr>              <int> <chr>           <int>
## 1 TX                    38 Clinton            38
## 2 TX                    38 Trump              41
## 3 WI                    10 Clinton            48
## 4 WI                    10 Trump              44
## 5 VA                    13 Clinton            54
## 6 VA                    13 Trump              41
```

```r
head(results)
```

```
## # A tibble: 6 x 3
## # Groups:   state_po [3]
##   state_po candidate        pct_vote
##   <chr>    <chr>               <dbl>
## 1 AK       Clinton, Hillary     36.6
## 2 AK       Trump, Donald J.     51.3
## 3 AL       Clinton, Hillary     34.4
## 4 AL       Trump, Donald J.     62.1
## 5 AR       Clinton, Hillary     33.7
## 6 AR       Trump, Donald J.     60.6
```

Renaming with a conditional

```
results_new <- results %>%
  mutate(candidate = case_when(
    candidate == "Clinton, Hillary" ~ "Clinton",
    candidate == "Trump, Donald J." ~ "Trump"
  ))
```

Renaming using string manipulation

```
results_new2 <- results %>%
  mutate(candidate =
           word(candidate, 1))

head(results_new2)
```

```
## # A tibble: 6 x 3
## # Groups:   state_po [3]
##   state_po candidate pct_vote
##   <chr>    <chr>        <dbl>
## 1 AK       Clinton,      36.6
## 2 AK       Trump,        51.3
## 3 AL       Clinton,      34.4
## 4 AL       Trump,        62.1
## 5 AR       Clinton,      33.7
## 6 AR       Trump,        60.6
```



```
## almost...
```

Renaming using string manipulation

```r
results_new2 <- results_new2 %>%
  mutate(candidate =
           str_replace(candidate, ",", ""))

head(results_new2)
```

```
## # A tibble: 6 x 3
## # Groups:   state_po [3]
##   state_po candidate pct_vote
##   <chr>    <chr>        <dbl>
## 1 AK       Clinton       36.6
## 2 AK       Trump         51.3
## 3 AL       Clinton       34.4
## 4 AL       Trump         62.1
## 5 AR       Clinton       33.7
## 6 AR       Trump         60.6
```

Putting it all together

```
results_new2 <- results %>%
  mutate(candidate =
           str_replace(
             word(candidate, 1),
             ",", ""))

head(results_new2)
```

```
## # A tibble: 6 x 3
## # Groups:   state_po [3]
##   state_po candidate pct_vote
##   <chr>    <chr>        <dbl>
## 1 AK       Clinton       36.6
## 2 AK       Trump         51.3
## 3 AL       Clinton       34.4
## 4 AL       Trump         62.1
## 5 AR       Clinton       33.7
```

Joins

Join them

- left_join() joins the object on the right to the object on the left, retaining all rows in the left hand object, but potentially removing rows in the right hand object.

Join them

- left_join() joins the object on the right to the object on the left, retaining all rows in the left hand object, but potentially removing rows in the right hand object.
- All columns are preserved.

```
polls_results<- polls_long %>%
  left_join(results_new2)

head(polls_results)
```

```
## # A tibble: 6 x 5
##   state_po electoral_votes candidate poll_result pct_vote
##   <chr>              <int> <chr>           <int>    <dbl>
## 1 TX                    38 Clinton            38     43.2
## 2 TX                    38 Trump              41     52.2
## 3 WI                    10 Clinton            48     46.5
## 4 WI                    10 Trump              44     47.2
## 5 VA                    13 Clinton            54     49.8
## 6 VA                    13 Trump              41     44.4
```

Check data structure to ensure we didn't create duplicates in the final object

We want to see the same number of rows in `polls_long` and `polls_results`

```
dim(polls_long)
```

```
## [1] 1454    4
```

```
dim(polls_results)
```

```
## [1] 1454    5
```