# Lecture 9: Regression Diagnostics and Plotting with `ggplot2`

ECON 480 - Econometrics - Fall 2018

Ryan Safner

September 26, 2018

Solvable Regression Problem #1: Heteroskedasticity

Solvable Regression Problem #2: Outliers

Advanced Plotting in `R` with `ggplot2`

# Solvable Regression Problem #1: Heteroskedasticity

# Homoskedasticity

- Recall assumption #2 about the regression residuals ($\epsilon$) are that they are homoskedastic:

$$var(\epsilon) = \sigma_{\epsilon}^{2}$$

# Homoskedasticity

- Recall assumption #2 about the regression residuals ($\epsilon$) are that they are homoskedastic:

$$var(\epsilon) = \sigma_\epsilon^2$$

- A fancy way of saying that the **variance of the residuals is constant**, i.e. does not change over values of $X$

# Homoskedasticity

- Recall assumption #2 about the regression residuals ($\epsilon$) are that they are homoskedastic:

$$var(\epsilon) = \sigma^2_\epsilon$$

- A fancy way of saying that the **variance of the residuals is constant**, i.e. does not change over values of $X$
- Combined with assumption #1 (the mean of the residuals $E[\epsilon] = 0$) $\implies$ residuals are i.i.d. and come from the same distribution $\sim (0, \sigma^2_\epsilon)$

# Homoskedasticity

- If residuals are heteroskedastic, they do *not* have the same variance over values of $X$

$$var(\epsilon) \neq \sigma_\epsilon^2$$

# Homoskedasticity

- If residuals are heteroskedastic, they do *not* have the same variance over values of $X$

$$var(\epsilon) \neq \sigma^2_\epsilon$$

- **This does not cause $\hat{\beta}_1$ to be biased**, but it does cause the standard error of $\hat{\beta}_1$ to be incorrect

# Homoskedasticity

- If residuals are heteroskedastic, they do *not* have the same variance over values of $X$

$$var(\epsilon) \neq \sigma_{\epsilon}^{2}$$

- **This does not cause $\hat{\beta}_1$ to be biased**, but it does cause the standard error of $\hat{\beta}_1$ to be incorrect
    - Recall, $se(\hat{\beta}_1)$ is used to calculate our test statistic for hypothesis testing

# Homoskedasticity

- If residuals are heteroskedastic, they do *not* have the same variance over values of $X$

$$var(\epsilon) \neq \sigma_{\epsilon}^{2}$$

- **This does not cause $\hat{\beta}_1$ to be biased**, but it does cause the standard error of $\hat{\beta}_1$ to be incorrect
  - Recall, $se(\hat{\beta}_1)$ is used to calculate our test statistic for hypothesis testing
  - **May overstate the statistical significance of a finding!**

# Homoskedasticity

- The formula for $se(\hat{\beta}_1)$ assumes homoskedasticity, recall (from Lecture 8) it was:

$$se(\hat{\beta}_1) = \sqrt{\frac{(SER)^2}{n \times var(X)}}$$

# Homoskedasticity

- The formula for $se(\hat{\beta}_1)$ assumes homoskedasticity, recall (from Lecture 8) it was:

$$se(\hat{\beta}_1) = \sqrt{\frac{(SER)^2}{n \times var(X)}}$$

- When errors are heteroskedastic, the formula mutates to

$$se(\hat{\beta}_1) = \sqrt{\frac{\displaystyle\sum_{i=1}^{n}(X_i - \bar{X})^2 \hat{\epsilon}^2}{\left[\displaystyle\sum_{i=1}^{n}(X_i - \bar{X})^2\right]^2}}$$

# Homoskedasticity

- The formula for $se(\hat{\beta}_1)$ assumes homoskedasticity, recall (from Lecture 8) it was:

$$se(\hat{\beta}_1) = \sqrt{\frac{(SER)^2}{n \times var(X)}}$$

- When errors are heteroskedastic, the formula mutates to

$$se(\hat{\beta}_1) = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2 \hat{\epsilon}^2}{\left[\sum_{i=1}^{n}(X_i - \bar{X})^2\right]^2}}$$

- This is heteroskedasticity-robust ("robust") method of calculating $se(\hat{\beta}_1)$

# Homoskedasticity

- The formula for $se(\hat{\beta}_1)$ assumes homoskedasticity, recall (from Lecture 8) it was:

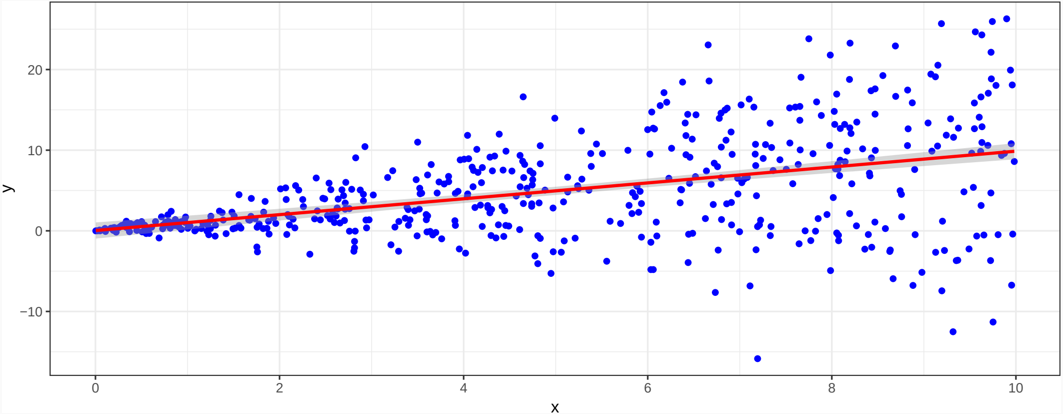
$$se(\hat{\beta}_1) = \sqrt{\frac{(SER)^2}{n \times var(X)}}$$

- When errors are heteroskedastic, the formula mutates to

$$se(\hat{\beta}_1) = \sqrt{\frac{\displaystyle\sum_{i=1}^{n}(X_i - \bar{X})^2\hat{\epsilon}^2}{\left[\displaystyle\sum_{i=1}^{n}(X_i - \bar{X})^2\right]^2}}$$

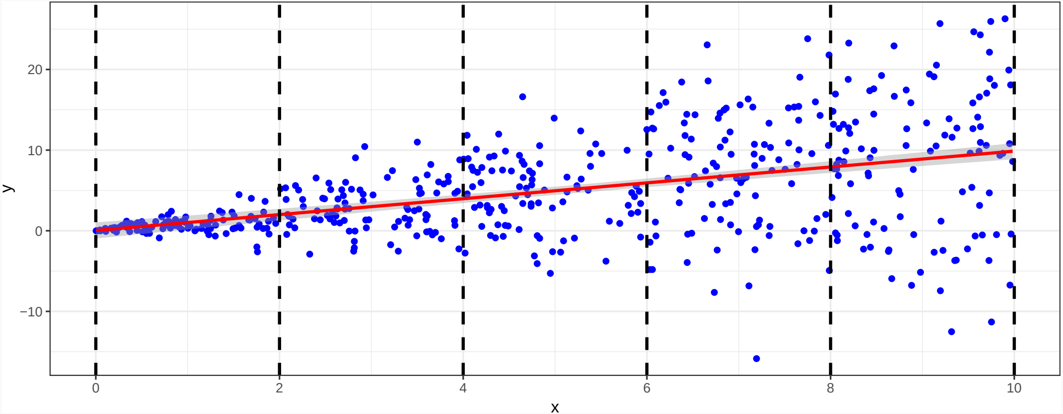
- This is heteroskedasticity-robust ("robust") method of calculating $se(\hat{\beta}_1)$
- Don't learn formula, **do learn what heteroskedasticity is and how it affects our model!**

# Visualizing Heteroskedasticity



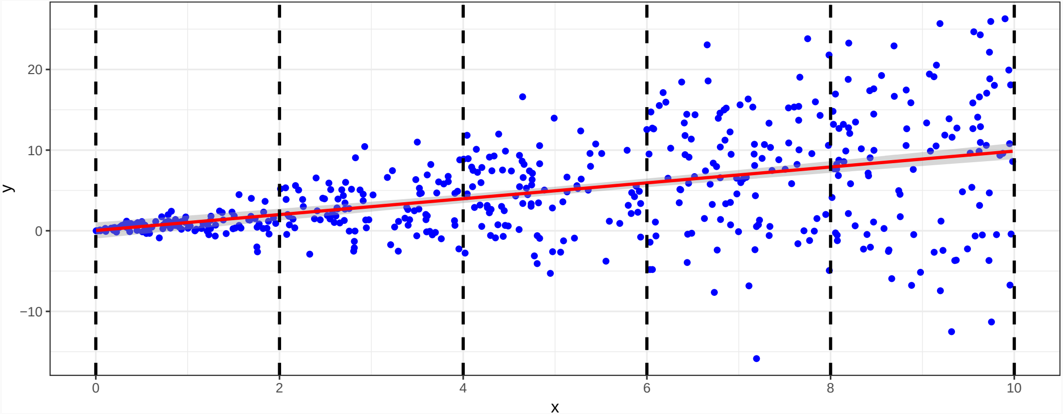
- The average residual (distance from point to OLS line) changes as *X* changes

# Visualizing Heteroskedasticity II



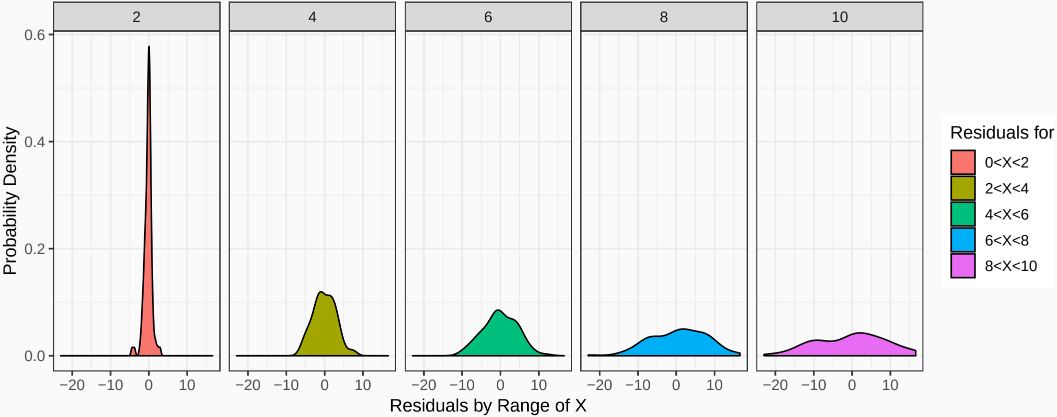
- We would expect the distribution of the residuals ($\hat{\epsilon}$) to be the same at every value of $X$

# Visualizing Heteroskedasticity II



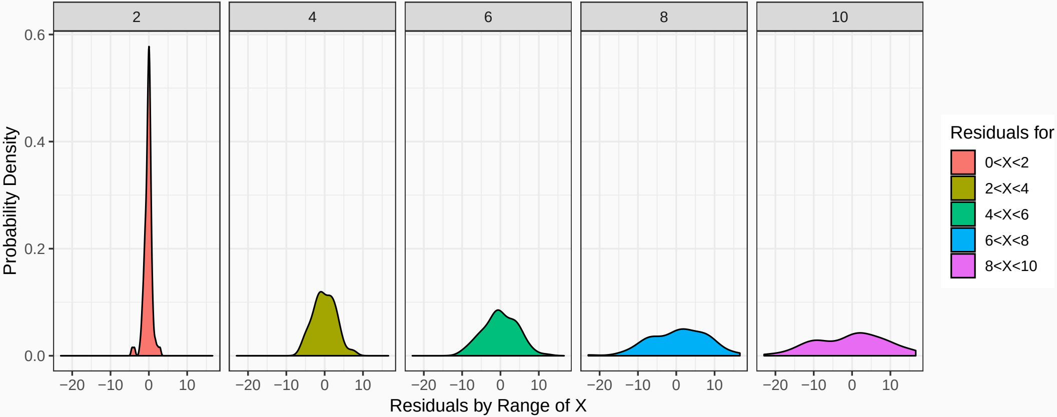
- We would expect the distribution of the residuals ($\hat{\epsilon}$) to be the same at every value of $X$
- Clearly there are very different distributions of residuals across $X$

## Visualizing Heteroskedasticity III



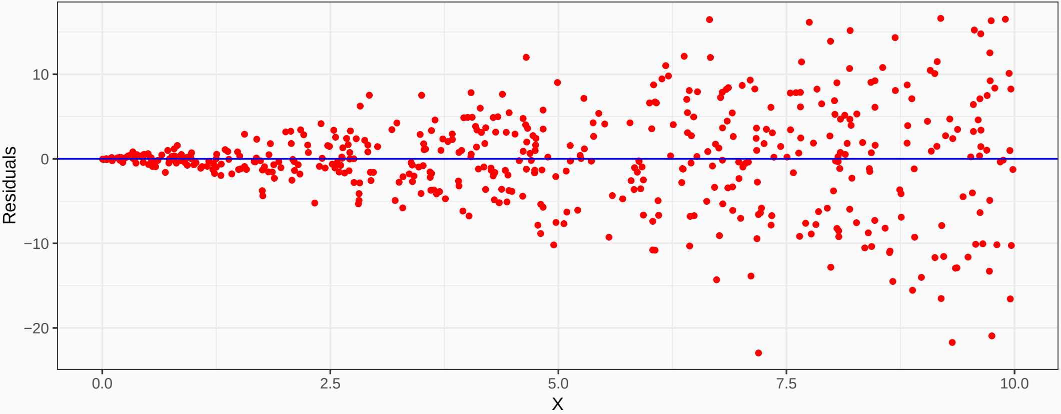
- We would expect the distribution of the residuals ($\hat{\epsilon}$) to be the same at every value of $X$

# Visualizing Heteroskedasticity III



- We would expect the distribution of the residuals ($\hat{\epsilon}$) to be the same at every value of $X$
- Clearly there are very different distributions of residuals across $X$

# Visualizing Heteroskedasticity IV



- We can also see in the residual plot that the size of residuals increases as *X* increases

# Detecting Heteroskedasticity

- Several tests to check if data is heteroskedastic

# Detecting Heteroskedasticity

- Several tests to check if data is heteroskedastic
- One common test is **Breusch-Pagan test**

# Detecting Heteroskedasticity

- Several tests to check if data is heteroskedastic
- One common test is **Breusch-Pagan test**
- Can use `bptest()` with `lmtest` package in `R`

# Detecting Heteroskedasticity

- Several tests to check if data is heteroskedastic
- One common test is **Breusch-Pagan test**
- Can use `bptest()` with `lmtest` package in R
    - $H_0$: homoskedastic

# Detecting Heteroskedasticity

- Several tests to check if data is heteroskedastic
- One common test is **Breusch-Pagan test**
- Can use `bptest()` with `lmtest` package in R
  - $H_0$: homoskedastic
  - If $p$-value < 0.05, reject $H_0 \implies$ heteroskedastic

# Detecting Heteroskedasticity

- Several tests to check if data is heteroskedastic
- One common test is **Breusch-Pagan test**
- Can use `bptest()` with `lmtest` package in R
  - $H_0$: homoskedastic
  - If $p$-value < 0.05, reject $H_0 \implies$ heteroskedastic

```
library(”lmtest”) #load lmtest package, install if first time
bptest(het.reg)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  het.reg
## BP = 112.99, df = 1, p-value < 2.2e-16
```

# Standard Errors: Technical Digression

- Before we generate robust SE’s, let’s look at plain SEs

# Standard Errors: Technical Digression

- Before we generate robust SE's, let's look at plain SEs
- Regression creates a **variance-covariance** matrix (`vcov`) of OLS estimators (betas), e.g.

$$\begin{bmatrix} cov(\hat{\beta}_0, \hat{\beta}_0) & cov(\hat{\beta}_0, \hat{\beta}_1) \\ cov(\hat{\beta}_0, \hat{\beta}_1) & cov(\hat{\beta}_1, \hat{\beta}_1) \end{bmatrix}$$

# Standard Errors: Technical Digression

- Before we generate robust SE's, let's look at plain SEs
- Regression creates a **variance-covariance** matrix ('`vcov`') of OLS estimators (betas), e.g.

$$\begin{bmatrix} cov(\hat{\beta}_0, \hat{\beta}_0) & cov(\hat{\beta}_0, \hat{\beta}_1) \\ cov(\hat{\beta}_0, \hat{\beta}_1) & cov(\hat{\beta}_1, \hat{\beta}_1) \end{bmatrix}$$

- Why a matrix?

# Standard Errors: Technical Digression

- Before we generate robust SE's, let's look at plain SEs
- Regression creates a **variance-covariance** matrix ('`vcov`') of OLS estimators (betas), e.g.

$$\begin{bmatrix} cov(\hat{\beta}_0, \hat{\beta}_0) & cov(\hat{\beta}_0, \hat{\beta}_1) \\ cov(\hat{\beta}_0, \hat{\beta}_1) & cov(\hat{\beta}_1, \hat{\beta}_1) \end{bmatrix}$$

- Why a matrix?
  - We currently have 2 OLS estimators $(\hat{\beta}_0, \hat{\beta}_1)$ for 1 $X_i$ variable

# Standard Errors: Technical Digression

- Before we generate robust SE’s, let’s look at plain SEs
- Regression creates a **variance-covariance** matrix (‘`vcov`’) of OLS estimators (betas), e.g.

$$\begin{bmatrix} cov(\hat{\beta}_0, \hat{\beta}_0) & cov(\hat{\beta}_0, \hat{\beta}_1) \\ cov(\hat{\beta}_0, \hat{\beta}_1) & cov(\hat{\beta}_1, \hat{\beta}_1) \end{bmatrix}$$

- Why a matrix?
  - We currently have 2 OLS estimators $(\hat{\beta}_0, \hat{\beta}_1)$ for 1 $X_i$ variable
  - We eventually will have $k$ $X_i$ variables, so a $\big(k + 1 \times k + 1\big)$ matrix

# Standard Errors: Technical Digression

- Before we generate robust SE's, let's look at plain SEs
- Regression creates a **variance-covariance** matrix (`vcov`) of OLS estimators (betas), e.g.

$$\begin{bmatrix} cov(\hat{\beta}_0, \hat{\beta}_0) & cov(\hat{\beta}_0, \hat{\beta}_1) \\ cov(\hat{\beta}_0, \hat{\beta}_1) & cov(\hat{\beta}_1, \hat{\beta}_1) \end{bmatrix}$$

- Why a matrix?
  - We currently have 2 OLS estimators $(\hat{\beta}_0, \hat{\beta}_1)$ for 1 $X_i$ variable
  - We eventually will have $k$ $X_i$ variables, so a $\big(k+1 \times k+1\big)$ matrix
- The 'diagonal' of the matrix contains the variance of each OLS estimator, since $cov(X, X) = var(X)$

# Standard Errors: Technical Digression

- Before we generate robust SE's, let's look at plain SEs
- Regression creates a **variance-covariance** matrix (`vcov`) of OLS estimators (betas), e.g.

$$\begin{bmatrix} cov(\hat{\beta}_0, \hat{\beta}_0) & cov(\hat{\beta}_0, \hat{\beta}_1) \\ cov(\hat{\beta}_0, \hat{\beta}_1) & cov(\hat{\beta}_1, \hat{\beta}_1) \end{bmatrix}$$

- Why a matrix?
  - We currently have 2 OLS estimators $(\hat{\beta}_0, \hat{\beta}_1)$ for 1 $X_i$ variable
  - We eventually will have $k$ $X_i$ variables, so a $\big(k+1 \times k+1\big)$ matrix
- The 'diagonal' of the matrix contains the variance of each OLS estimator, since $cov(X, X) = var(X)$
- Taking the diagonal and square rooting the terms gives us the SE of each estimator

# Standard Errors: In R

```r
# the variance-covariance matrix
vcov(het.reg)
```

```
##             (Intercept)           x
## (Intercept)  0.26024710 -0.038597979
## x           -0.03859798  0.007717254
```

```r
#the var(beta)'s are the diagonal of the matrix

diag(vcov(het.reg)) # look at just the diagonal values
```

```
## (Intercept)           x
## 0.260247096 0.007717254
```

```r
#convert into standard errors by square rooting
sqrt(diag(vcov(het.reg)))
```

```
##   (Intercept)           x
##   0.5101442   0.0878479
```

# Standard Errors: In R II

```r
# confirming the SE's match what R finds automatically with lm()
summary(het.reg)
```

```
## 
## Call:
## lm(formula = y ~ x, data = het.data)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -22.9678  -2.2192   0.0148   2.9127  16.6128 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  0.03632    0.51014   0.071    0.943    
## x            0.98398    0.08785  11.201   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 5.796 on 498 degrees of freedom
## Multiple R-squared:  0.2012, Adjusted R-squared:  0.1996
```

# Using Robust Standard Errors in R

```r
library(”sandwich”) # package that allows for robust SE estimation, install if first use
library(”lmtest”) # package that allows for coefficient tests, install if first use

# take original regression and change standard errors to robust SEs #

# create Robust Standard Errors for regression as 'het.reg$rse'
het.reg$rse <-sqrt(diag(vcovHC(het.reg, type=”HC1”)))
# same procedure as above but now we generate vcov with ”HC1” method (technical)
```

# Using Robust Standard Errors in R II

- Using `coeftest()` function in the `lmtest` package, hypothesis tests with robust SEs

```
coeftest(het.reg) # test with normal SEs
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.036318   0.510144  0.0712   0.9433
## x           0.983984   0.087848 11.2010   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(het.reg,vcov=vcovHC(het.reg,”HC1”)) # tests with robust SEs
```

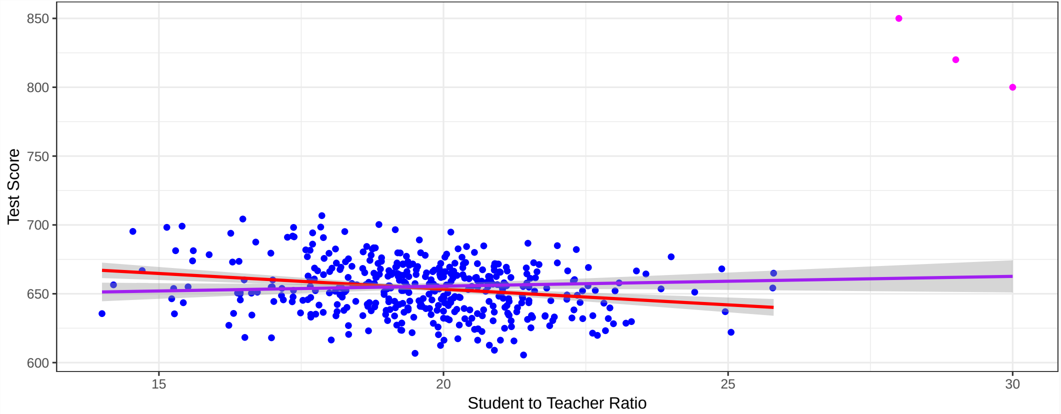
```
##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.036318   0.315781   0.115   0.9085
## x           0.983984   0.097101  10.134   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

# Using Robust Standard Errors in R: `stargazer` output

| | y | |
|---|---|---|
| | Normal SEs | Robust SEs |
| | (1) | (2) |
| x | $0.984^{***}$ | $0.984^{***}$ |
| | (0.088) | (0.097) |
| Constant | 0.036 | 0.036 |
| | (0.510) | (0.316) |
| *N* | 500 | 500 |
| $R^2$ | 0.201 | 0.201 |
| Residual Std. Error (df = 498) | 5.796 | 5.796 |
| *Notes:* | $^{***}$Significant at the 1 percent level. | |
| | $^{**}$Significant at the 5 percent level. | |
| | $^{*}$Significant at the 10 percent level. | |

# Solvable Regression Problem #2: Outliers

# Outliers Can Bias OLS



- Outliers can affect the slope (and intercept) of the line

# Outliers Can Bias OLS

| | testscr | |
|---|---|---|
| | With Outliers | Without Outliers |
| | (1) | (2) |
| str | 0.708 | −2.280*** |
| | (0.566) | (0.480) |
| Constant | 641.404*** | 698.933*** |
| | (11.215) | (9.467) |
| *N* | 423 | 420 |
| $R^2$ | 0.004 | 0.051 |
| Residual Std. Error | 23.764 (df = 421) | 18.581 (df = 418) |
| *Notes:* | | ***Significant at the 1 percent level. |
| | | **Significant at the 5 percent level. |
| | | *Significant at the 10 percent level. |

# Detecting Outliers

- Plot the data and look!

# Detecting Outliers

- Plot the data and look!
- A few methods to detect influence: ability of individual observations to affect OLS estimates

# Detecting Outliers with car Package

```r
library("car")

# Use Bonferonni test
outlierTest(school.outlier.reg) # will point out which obs #s seem outliers
```

```
##     rstudent unadjusted p-value Bonferonni p
## 422 8.822768         3.0261e-17   1.2800e-14
## 423 7.233470         2.2493e-12   9.5147e-10
## 421 6.232045         1.1209e-09   4.7414e-07
```

# DFBETAS

- **dfbetas** measure how different each OLS coefficient will be without each observation

# DFBETAS

- **dfbetas** measure how different each OLS coefficient will be without each observation
    - Scales the measure by standard error of OLS coefficient with observation deleted

# DFBETAS

- **dfbetas** measure how different each OLS coefficient will be without each observation
    - Scales the measure by standard error of OLS coefficient with observation deleted
    - Name means “difference in beta” from deleted observation

# DFBETAS

- **dfbetas** measure how different each OLS coefficient will be without each observation
    - Scales the measure by standard error of OLS coefficient with observation deleted
    - Name means “difference in beta” from deleted observation
        - e.g. $dfbeta_i = -3$: observation $i$ decreases coefficient by 3 standard errors

# DFBETAS

- **dfbetas** measure how different each OLS coefficient will be without each observation
    - Scales the measure by standard error of OLS coefficient with observation deleted
    - Name means “difference in beta” from deleted observation
        - e.g. $dfbeta_i = -3$: observation $i$ decreases coefficient by 3 standard errors
    - Downside: calculates this measure for *each* observation for *each* beta ($n \times k$ dfbetas)!

# DFBETAS IN R

```
dfbetas(school.outlier.reg)
```

```
##       (Intercept)           str
## 1     7.471830e-02 -6.728767e-02
## 2    -7.320670e-03  8.328531e-03
## 3    -1.346882e-02  1.119610e-02
## 4    -1.536989e-02  1.417645e-02
## 5    -1.716026e-02  1.432668e-02
## 6     7.630992e-02 -8.761930e-02
## 7    -2.033255e-02  1.010752e-02
## 8     4.622949e-02 -5.654660e-02
## 9     1.255955e-03 -1.042826e-02
## 10    3.869530e-02 -4.817225e-02
## 11    5.386822e-02 -6.283208e-02
## 12    4.334903e-02 -5.209908e-02
## 13    2.711156e-02 -3.570977e-02
## 14    3.538636e-03 -1.191752e-02
## 15   -7.150381e-02  6.384636e-02
## 16    1.283115e-02 -2.107274e-02
## 17   -1.045438e-01  9.754538e-02
## 18   -1.193732e-01  1.125672e-01
## 19    1.056295e-01 -1.142546e-01
## 20   -2.511522e-04 -7.236259e-03
## 21   -5.379359e-02  4.686663e-02
## 22    9.784139e-02 -1.060655e-01
## 23   -1.571693e-02  8.692185e-03
## 24    1.938071e-01 -2.028677e-01
```

# Dealing with Outliers

- Often, outliers may be the result of human error (measurement, transcribing, etc)

## Dealing with Outliers

- Often, outliers may be the result of human error (measurement, transcribing, etc)
- Outliers may be meaningful and accurate

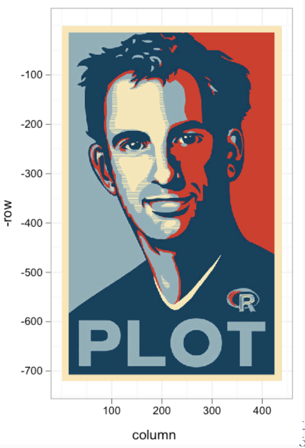
# Dealing with Outliers

- Often, outliers may be the result of human error (measurement, transcribing, etc)
- Outliers may be meaningful and accurate
- In any case, compare how including/dropping outliers affects regression and always discuss outliers!

# Advanced Plotting in R with `ggplot2`

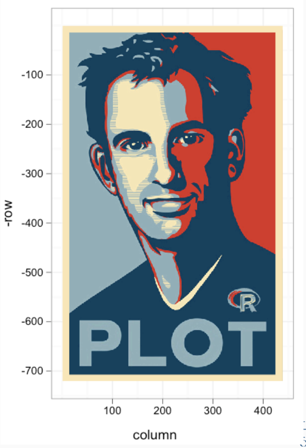
## ggplot2

- `ggplot2` is the most popular package



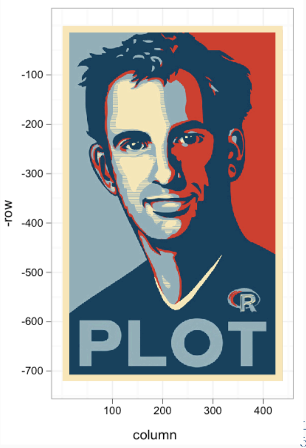
# ggplot2

- `ggplot2` is the most popular package
- Created by Hadley Wickham, part of `tidyverse` (really the ”Hadleyverse”)



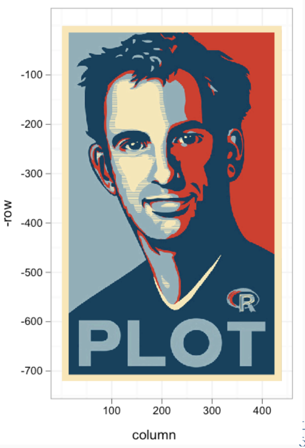
# ggplot2

- ggplot2 is the most popular package
- Created by Hadley Wickham, part of tidyverse (really the ”Hadleyverse”)
- Very powerful and beautiful, but requires a bit more of a learning curve



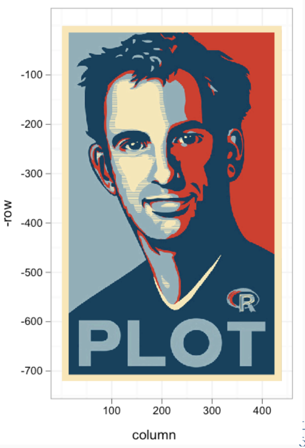
# ggplot2

- `ggplot2` is the most popular package
- Created by Hadley Wickham, part of `tidyverse` (really the ”Hadleyverse”)
- Very powerful and beautiful, but requires a bit more of a learning curve
- All those ”cool graphics” you’ve seen in the New York Times, fivethirtyeight, the Economist, Vox, etc. used ggplot2

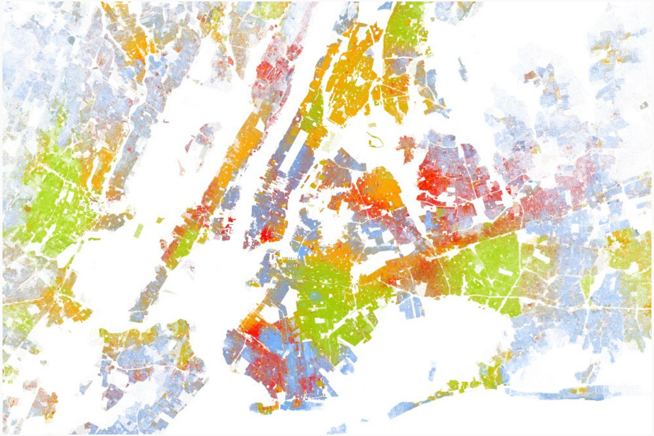


# ggplot2

- `ggplot2` is the most popular package
- Created by Hadley Wickham, part of **`tidyverse`** (really the ”Hadleyverse”)
- Very powerful and beautiful, but requires a bit more of a learning curve
- All those ”cool graphics” you’ve seen in the New York Times, fivethirtyeight, the Economist, Vox, etc. used ggplot2
- gg stands for a **grammar of graphics**



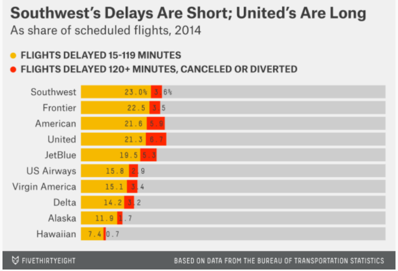
# A Picture is Worth a Thousand Words



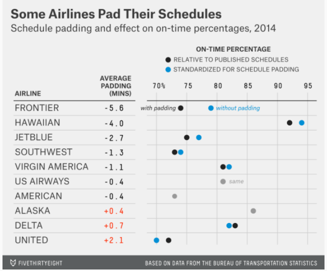
Dustin Cable's Racial Dot Map of NYC[1], The Best Map Ever Made of America's Racial Segretation; his (Python) code is open-source and available on Github

[1] 1 dot = 1 person, colors: White, African-American, Asian, Latino, All Other

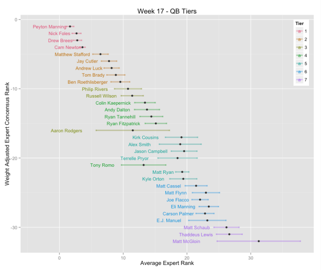
# ggplot2: All Your Figure Are Belong to Us



From fivethirtyyeight



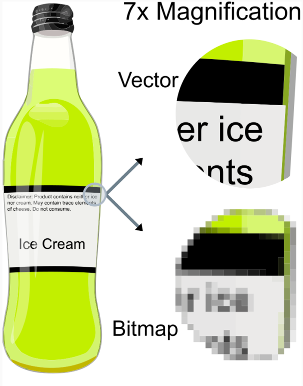
From fivethirtyyeight



From New York Times

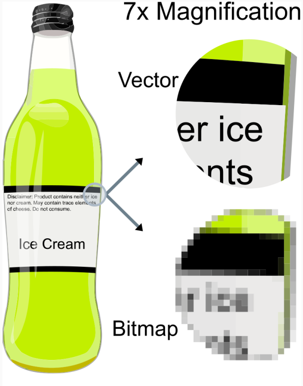
# ggplot2 FIGURES

- Figures produced are **vector-based** graphics (.pdf, .svg)
    - Can rescale to any size and not look ”pixely”
    - But big file size


7x Magnification
Vector
er ice
nts
Bitmap
Disclaimer: Product contains neither ice nor cream. May contain trace elements of cheese. Do not consume.
Ice Cream

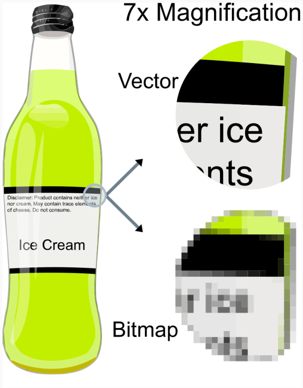
# ggplot2 FIGURES

- Figures produced are **vector-based** graphics (.pdf, .svg)
  - Can rescale to any size and not look ”pixely”
  - But big file size
- Can save figures as **raster-based** graphics (.png, .jpeg, .bmp, .gif)
  - Look *awful* when blown up
  - But small file size, great for MS Office and Web


7x Magnification
Vector
er ice
nts
Bitmap
Disclaimer: Product contains neither ice nor cream. May contain trace elements of cheese. Do not consume.
Ice Cream

# `ggplot2` FIGURES

- Figures produced are **vector-based** graphics (.pdf, .svg)
    - Can rescale to any size and not look ”pixely”
    - But big file size
- Can save figures as **raster-based** graphics (.png, .jpeg, .bmp, .gif)
    - Look *awful* when blown up
    - But small file size, great for MS Office and Web
- Suggestions:
    - If printing on paper, save graphics as .pdf
    - If posting to the web, save as .png and specify size


7x Magnification
Vector
er ice
nts
Bitmap
Disclaimer: Product contains neither ice nor cream. May contain trace elements of cheese. Do not consume.
Ice Cream

# Two Ways to Plot

1. Just the single `ggplot` command

```
ggplot(...) # make and view plot
ggplot(some.options) # remake plot with new options and view plot
```

# Two Ways to Plot

1. Just the single `ggplot` command
    - Will view plot right after producing it

```
ggplot(...) # make and view plot
ggplot(some.options) # remake plot with new options and view plot
```

# Two Ways to Plot

1. Just the single `ggplot` command
    - Will view plot right after producing it
    - Does not save as an object

```
ggplot(...) # make and view plot
ggplot(some.options) # remake plot with new options and view plot
```

# Two Ways to Plot

1. Just the single `ggplot` command
    - Will view plot right after producing it
    - Does not save as an object
    - Need to rerun or copy/paste full command producing plot in order to modify or view it again

```
ggplot(...) # make and view plot
ggplot(some.options) # remake plot with new options and view plot
```

# Two Ways to Plot

1. Just the single **ggplot** command

   - Will view plot right after producing it
   - Does not save as an object
   - Need to rerun or copy/paste full command producing plot in order to modify or view it again
   - Can still put it in a document

```
ggplot(...) # make and view plot
ggplot(some.options) # remake plot with new options and view plot
```

# Two Ways to Plot II

2. Create an object (as usual in R)

```r
plot.name<-ggplot(...) # make plot
plot.name<-plot.name+some.options # add new options to existing plot
plot.name # view plot
```

# Two Ways to Plot II

2. Create an object (as usual in R)
    - This allows you to save the plot for later (re)use

```r
plot.name<-ggplot(...) # make plot
plot.name<-plot.name+some.options # add new options to existing plot
plot.name # view plot
```

# Two Ways to Plot II

2. Create an object (as usual in R)
    - This allows you to save the plot for later (re)use
    - Also allows you to modify it

```
plot.name<-ggplot(...) # make plot
plot.name<-plot.name+some.options # add new options to existing plot
plot.name # view plot
```

# Two Ways to Plot II

2. Create an object (as usual in R)
    - This allows you to save the plot for later (re)use
    - Also allows you to modify it
    - Any time you want to view display it (i.e. for putting it in a document), just call up the plot by name

```
plot.name<-ggplot(...) # make plot
plot.name<-plot.name+some.options # add new options to existing plot
plot.name # view plot
```

# Grammar of Graphics

```
plot.name<-ggplot(data=mydf, mapping=aes(x=xvar,y=yvar))+
  geom_something(options)+
  moreoptions...
```

- `gg` “grammar of graphics” implies any graphic can be built from the same components/layers:

# Grammar of Graphics

```
plot.name<-ggplot(data=mydf, mapping=aes(x=xvar,y=yvar))+
  geom_something(options)+
  moreoptions...
```

- gg "grammar of graphics" implies any graphic can be built from the same components/layers:
  1. **Data**: base-layer describes the data used

# Grammar of Graphics

```
plot.name<-ggplot(data=mydf, mapping=aes(x=xvar,y=yvar))+
  geom_something(options)+
  moreoptions...
```

- gg "grammar of graphics" implies any graphic can be built from the same components/layers:
  1. **Data**: base-layer describes the data used
     - mydf is the dataframe containing xvar and yvar

# Grammar of Graphics

```
plot.name<-ggplot(data=mydf, mapping=aes(x=xvar,y=yvar))+
  geom_something(options)+
  moreoptions...
```

- `gg` “grammar of graphics” implies any graphic can be built from the same components/layers:
    1. **Data**: base-layer describes the data used
        - `mydf` is the dataframe containing `xvar` and `yvar`
        - `aes()` “aesthetics” identifies `xvar` (and if applicable `yvar`) from data to be “mapped” to a visual mark

# Grammar of Graphics

```
plot.name<-ggplot(data=mydf, mapping=aes(x=xvar,y=yvar))+
  geom_something(options)+
  moreoptions...
```

- `gg` “grammar of graphics” implies any graphic can be built from the same components/layers:
  1. **Data**: base-layer describes the data used
     - `mydf` is the dataframe containing `xvar` and `yvar`
     - `aes()` “aesthetics” identifies `xvar` (and if applicable `yvar`) from data to be “mapped” to a visual mark
  2. **Geoms**: visual marks that represent data observations or models, common examples:

# Grammar of Graphics

```
plot.name<-ggplot(data=mydf, mapping=aes(x=xvar,y=yvar))+
  geom_something(options)+
  moreoptions...
```

- gg "grammar of graphics" implies any graphic can be built from the same components/layers:
  1. **Data**: base-layer describes the data used
     - `mydf` is the dataframe containing `xvar` and `yvar`
     - `aes()` "aesthetics" identifies `xvar` (and if applicable `yvar`) from data to be "mapped" to a visual mark
  2. **Geoms**: visual marks that represent data observations or models, common examples:
     - e.g. `geom_point`, `geom_line`, `geom_bar`, `geom_histogram`, `geom_density`, `geom_boxplot`

# Grammar of Graphics

```
plot.name<-ggplot(data=mydf, mapping=aes(x=xvar,y=yvar))+
  geom_something(options)+
  moreoptions...
```

- gg "grammar of graphics" implies any graphic can be built from the same components/layers:
  1. **Data**: base-layer describes the data used
     - `mydf` is the dataframe containing `xvar` and `yvar`
     - `aes()` "aesthetics" identifies `xvar` (and if applicable `yvar`) from data to be "mapped" to a visual mark
  2. **Geoms**: visual marks that represent data observations or models, common examples:
     - e.g. `geom_point`, `geom_line`, `geom_bar`, `geom_histogram`, `geom_density`, `geom_boxplot`
  3. **Coordinates**: Cartesian coordinates are default

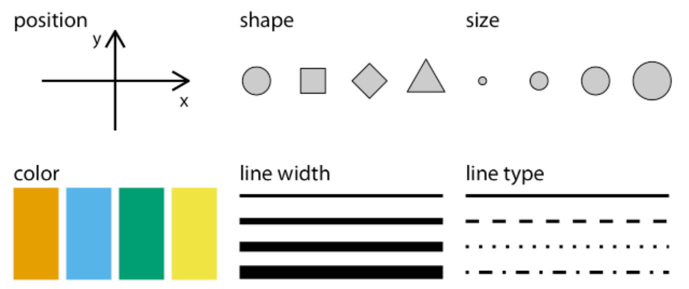
# Grammar of Graphics

```
plot.name<-ggplot(data=mydf, mapping=aes(x=xvar,y=yvar))+
  geom_something(options)+
  moreoptions...
```

- gg “grammar of graphics” implies any graphic can be built from the same components/layers:
  1. **Data**: base-layer describes the data used
     - `mydf` is the dataframe containing `xvar` and `yvar`
     - `aes()` “aesthetics” identifies `xvar` (and if applicable `yvar`) from data to be “mapped” to a visual mark
  2. **Geoms**: visual marks that represent data observations or models, common examples:
     - e.g. `geom_point`, `geom_line`, `geom_bar`, `geom_histogram`, `geom_density`, `geom_boxplot`
  3. **Coordinates**: Cartesian coordinates are default
     - change scales, axes, labels, etc; advanced options like maps

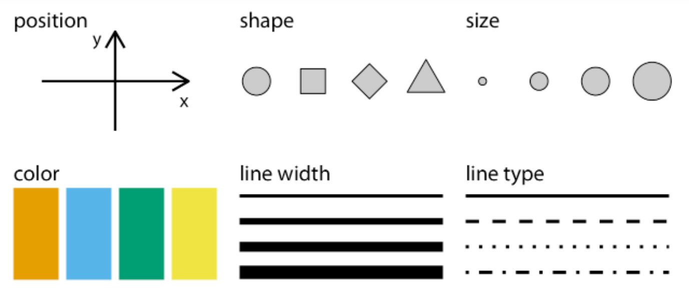
# Grammar of Graphics II

- Most important idea to master is `aes()` **aesthetics** that map data to visual markings



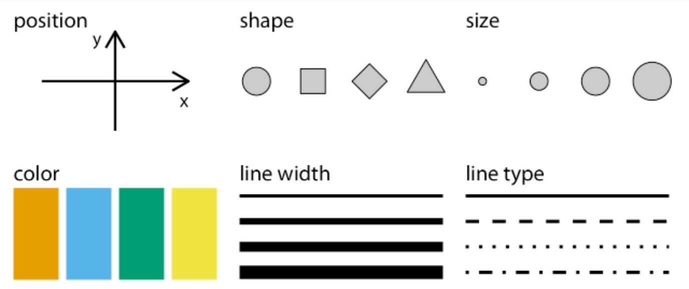
# Grammar of Graphics II

- Most important idea to master is **`aes()` aesthetics** that map data to visual markings
- Aesthetics come in many forms and many options, depending on the context of the data



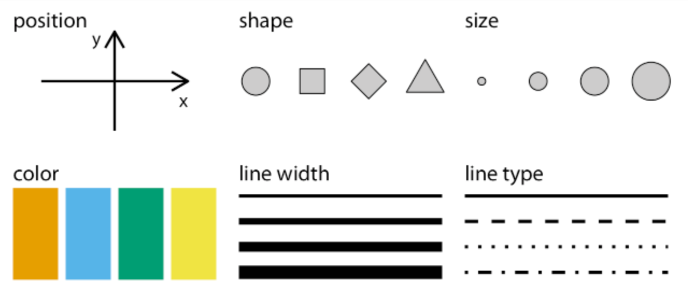
# Grammar of Graphics II

- Most important idea to master is `aes()` **aesthetics** that map data to visual markings
- Aesthetics come in many forms and many options, depending on the context of the data
  - Must identify position (e.g. what is x and y)



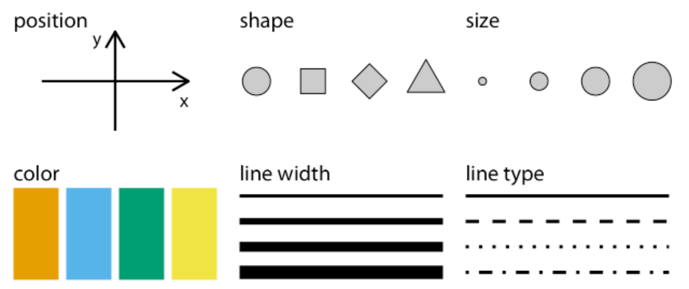
# Grammar of Graphics II

- Most important idea to master is **`aes()` aesthetics** that map data to visual markings
- Aesthetics come in many forms and many options, depending on the context of the data
  - Must identify position (e.g. what is `x` and `y`)
  - Determine the marking with various **`geoms`** (points, bars, lines, boxes, etc)



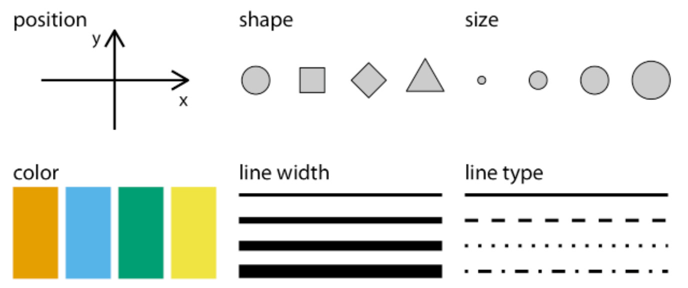
# Grammar of Graphics II

- Most important idea to master is `aes()` **aesthetics** that map data to visual markings
- Aesthetics come in many forms and many options, depending on the context of the data
  - Must identify position (e.g. what is `x` and `y`)
  - Determine the marking with various `geoms` (points, bars, lines, boxes, etc)
  - Can pass additional options into `geom` (color, size, shape, etc)



# Grammar of Graphics II

- Most important idea to master is **`aes()` aesthetics** that map data to visual markings
- Aesthetics come in many forms and many options, depending on the context of the data
    - Must identify position (e.g. what is `x` and `y`)
    - Determine the marking with various **`geoms`** (points, bars, lines, boxes, etc)
    - Can pass additional options into **`geom`** (color, size, shape, etc)
        - Particularly important if we want color, size, or shape to depend on a particular variable in dataset



# Example

For our example, we'll use the mpg dataset loaded with the `ggplot2` package

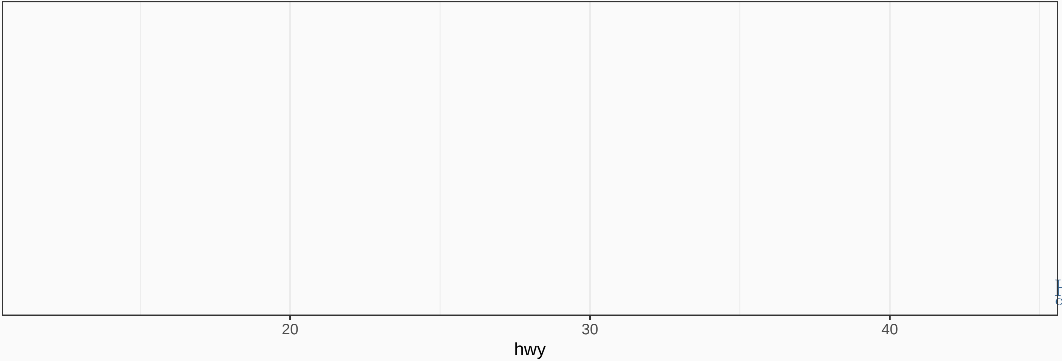
```
library("ggplot2") #load ggplot2
mpg #look at dataset
```

```
## # A tibble: 234 x 11
##    manufacturer model    displ  year   cyl trans   drv     cty   hwy fl
##    <chr>        <chr>    <dbl> <int> <int> <chr>   <chr> <int> <int> <chr>
##  1 audi         a4         1.8  1999     4 auto(l~ f        18    29 p
##  2 audi         a4         1.8  1999     4 manual~ f        21    29 p
##  3 audi         a4         2    2008     4 manual~ f        20    31 p
##  4 audi         a4         2    2008     4 auto(a~ f        21    30 p
##  5 audi         a4         2.8  1999     6 auto(l~ f        16    26 p
##  6 audi         a4         2.8  1999     6 manual~ f        18    26 p
##  7 audi         a4         3.1  2008     6 auto(a~ f        18    27 p
```

# gg Histogram: Base Layer

- Start with the base layer: establish the data source, define *x* variable

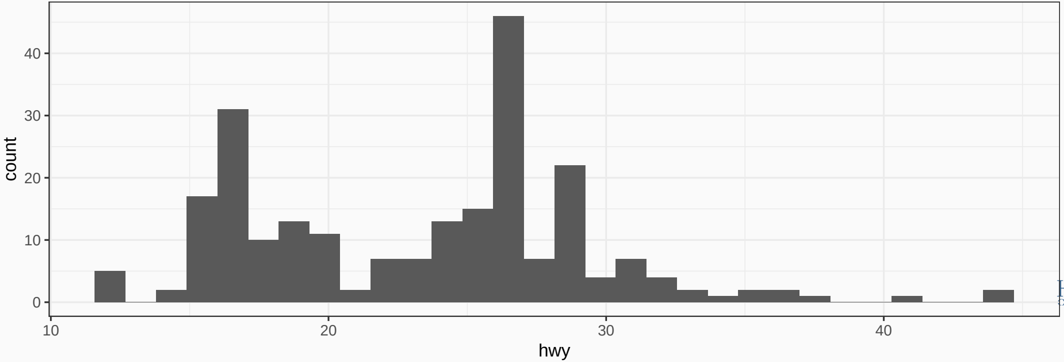
```
mpg.h<-ggplot(data=mpg,mapping=aes(x=hwy))
mpg.h
```



# gg Histogram: Adding Geoms

- Add a histogram layer of **hwy**

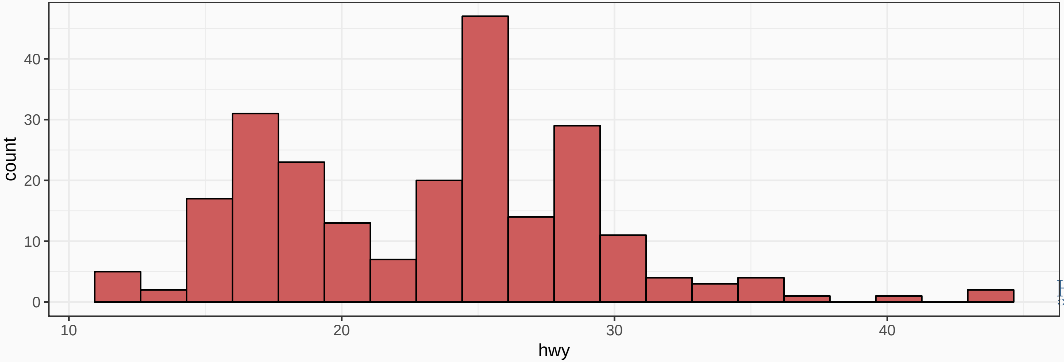
```
mpg.h1<-mpg.h+geom_histogram()
mpg.h1
```



# gg Histogram: Customizing Geoms

- Edit the histogram (# of bins, color, etc)

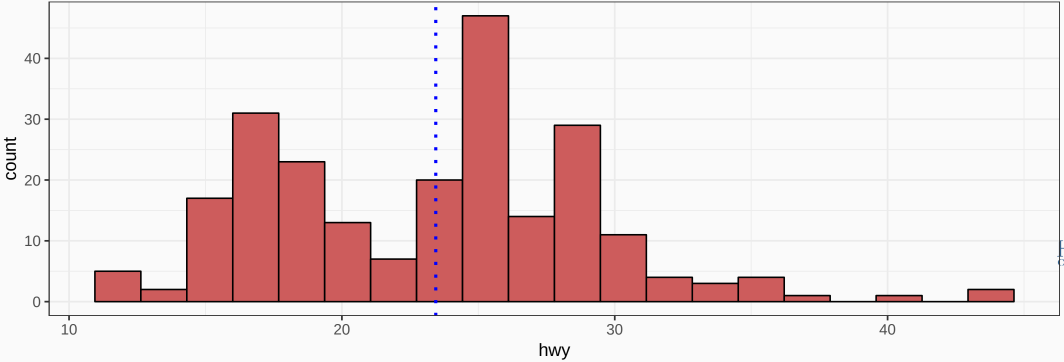
```
mpg.h2<-mpg.h+geom_histogram(bins=20, color=”black”,fill=”indianred”)
mpg.h2
```



# gg Histogram: Adding Other Layers

- Add a vertical line for the mean with another **geom** called **vline**

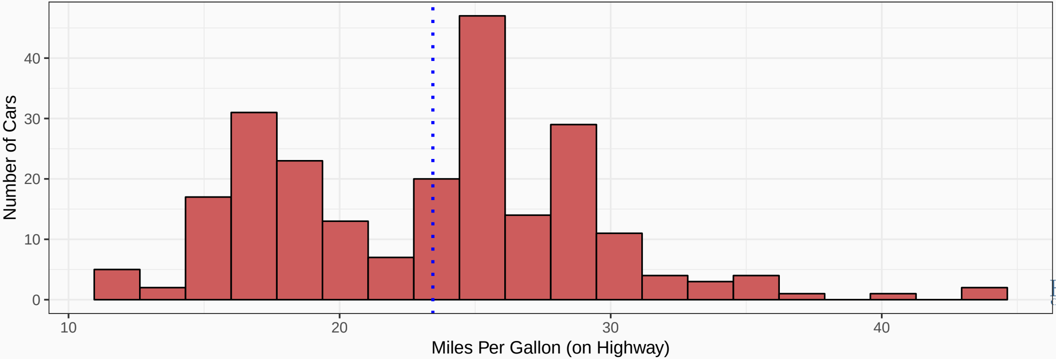
```
mpg.h2<-mpg.h2+
  geom_vline(xintercept=mean(mpg$hwy),linetype="dotted",color="blue",size=1)
mpg.h2
```



# gg Histogram: Editing Coordinates (Axes)

- Change the labels on the axes with `xlab()` and `ylab()`

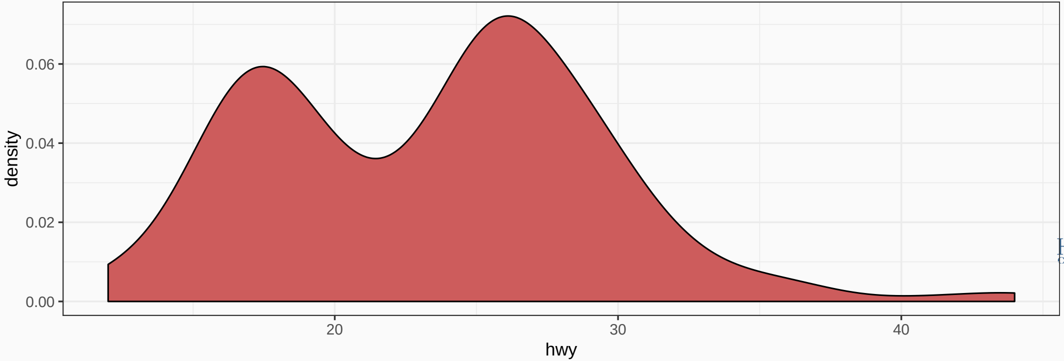
```
mpg.h2<-mpg.h2+xlab("Miles Per Gallon (on Highway)")+ylab("Number of Cars")
mpg.h2
```



# gg Other Geoms

- How about a **density plot**: use `geom_density()` instead of `geom_histogram()`

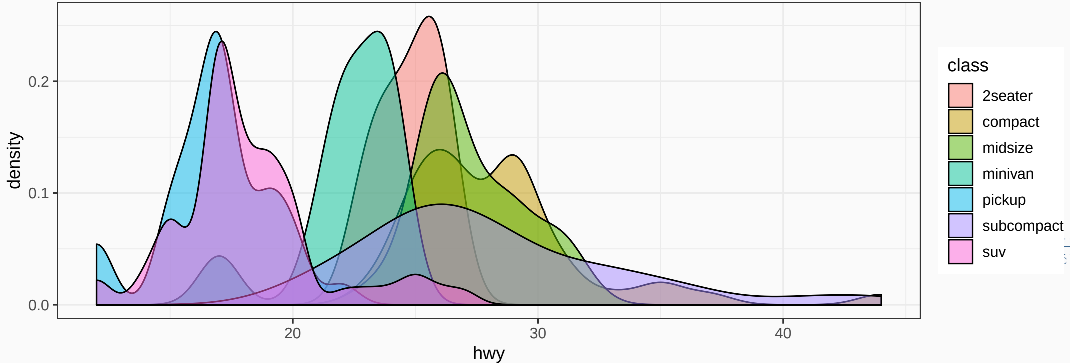
```
mpg.d<-ggplot(data=mpg,aes(x=hwy))+
  geom_density(fill="indianred")
mpg.d
```



# gg Other Geoms

- Let's make a separate density plot for each `class`, set `aes` to `fill` by `class`

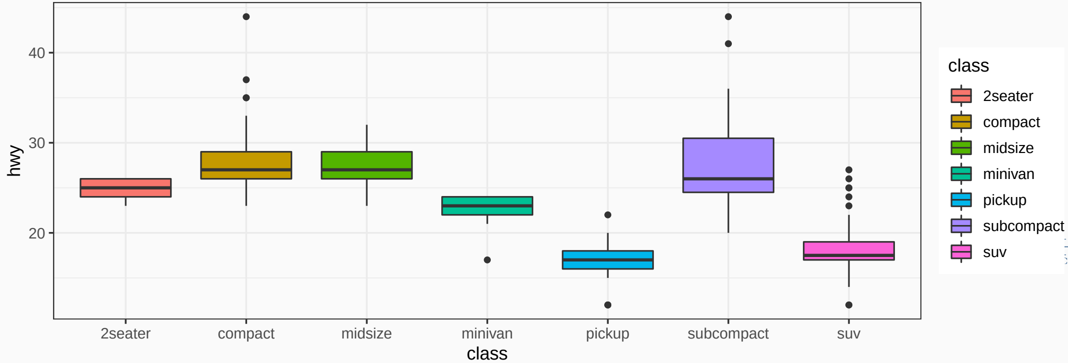
```
mpg.d<-ggplot(data=mpg,aes(x=hwy,fill=class))+
  geom_density(alpha=0.5) # alpha adds transparency
mpg.d
```



# gg Boxplot

- Instead of a density plot, a **`boxplot`** by **`class`** (note now **`x`** is **`class`** and **`y`** is **`hwy`**):

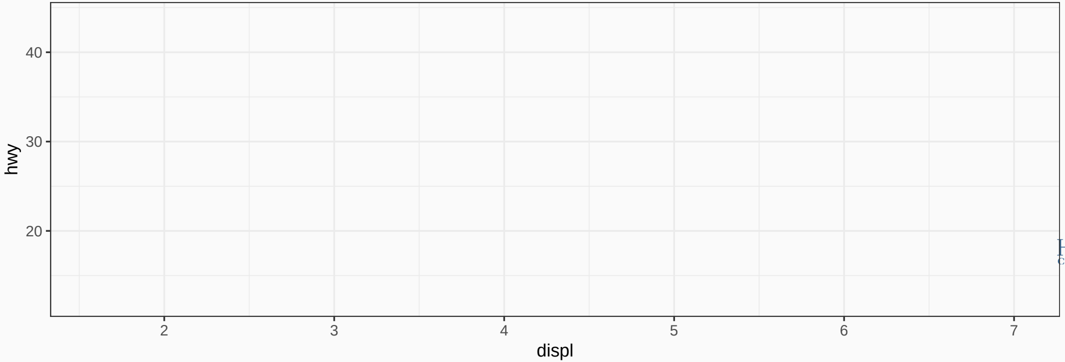
```r
mpg.b<-ggplot(data=mpg,aes(x=class,y=hwy,fill=class))+
  geom_boxplot()
mpg.b
```



# Scatterplot

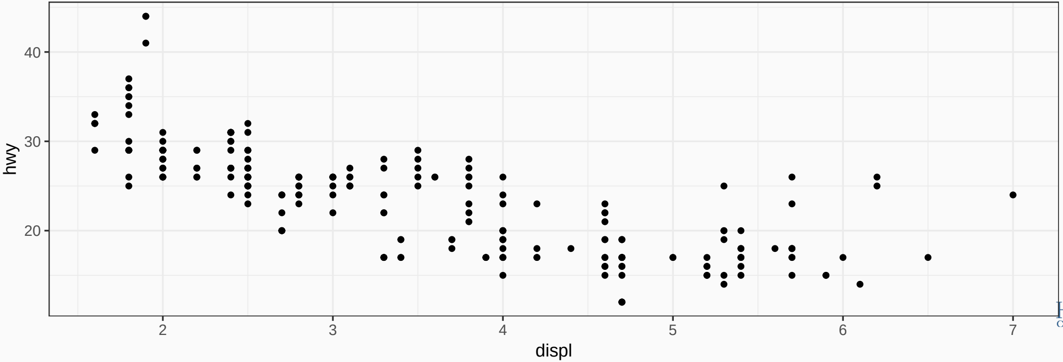
- Start with the base layer: establish data source, define *x* and *y* variables

```
mpg.p<-ggplot(data=mpg,aes(x=displ, y=hwy)) #use mtcars df, let x=displ, y=hwy

mpg.p
```



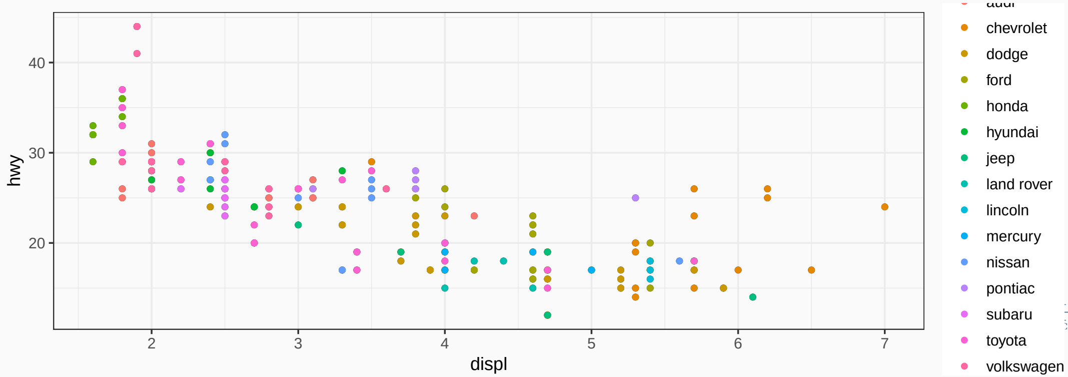
# Scatterplot: Geom Layer

```
mpg.p<-mpg.p+geom_point() # specify observations as points on graph

mpg.p
```



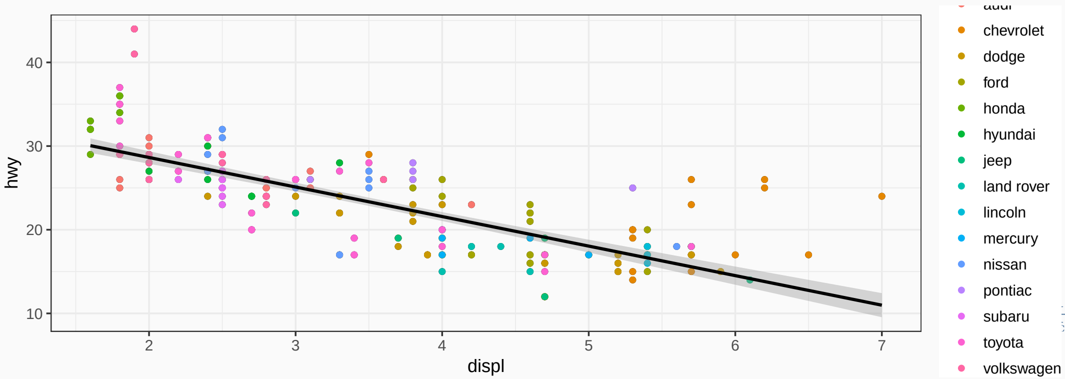
# Scatterplot: Geom Layer Options

```
mpg.p<-mpg.p+geom_point(aes(color=manufacturer)) # color data points by manuf.

mpg.p
```



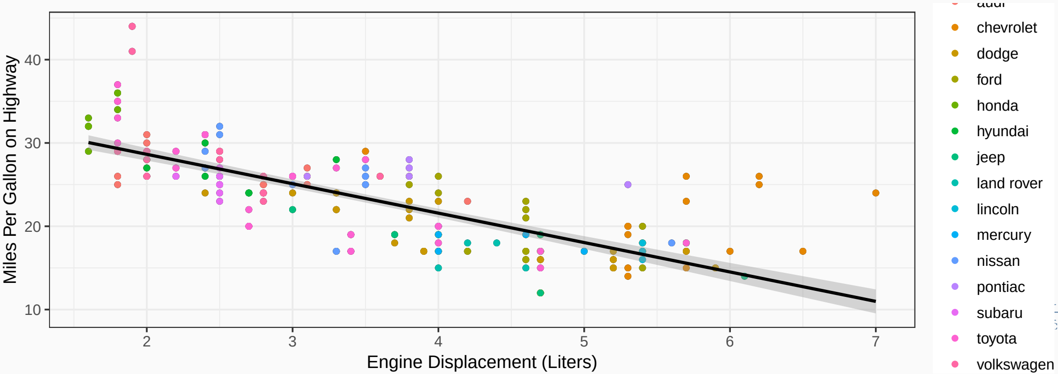
# Scatterplot: Geom Layer II

```
mpg.p<-mpg.p+geom_smooth(method=”lm”, color=”black”) # add a black OLS line

mpg.p
```



# Scatterplot: Coordinate Layer

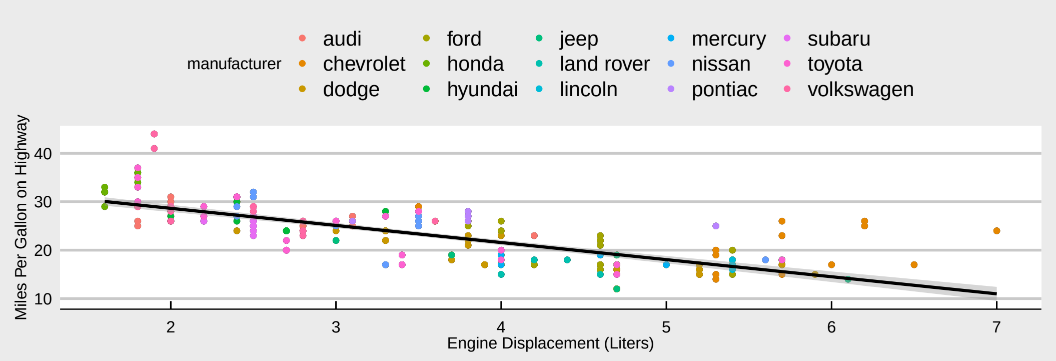
```
mpg.p<-mpg.p+xlab(”Engine Displacement (Liters)”)+
  ylab(”Miles Per Gallon on Highway”)
mpg.p
```



# Scatterplot: Coordinate Options

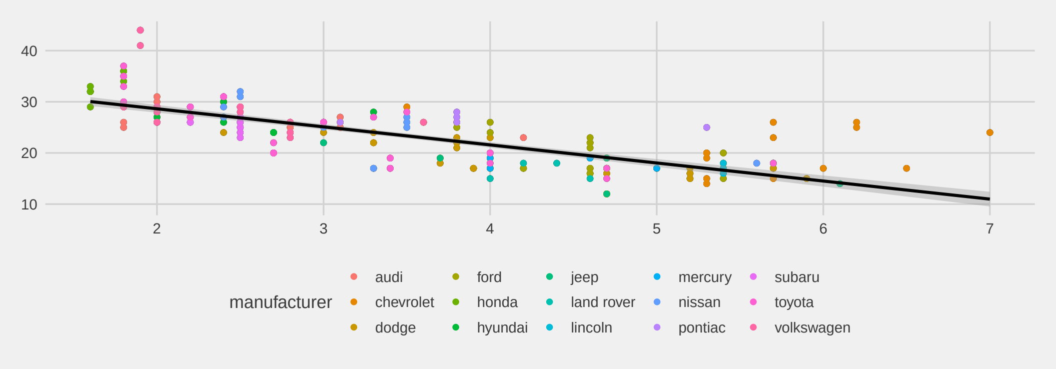
- Let's have some fun changing the theme

```
library("ggthemes") # need ggthemes package (install if first use)
mpg.p<-mpg.p+theme_economist_white() #make it look like The Economist magazine
mpg.p
```



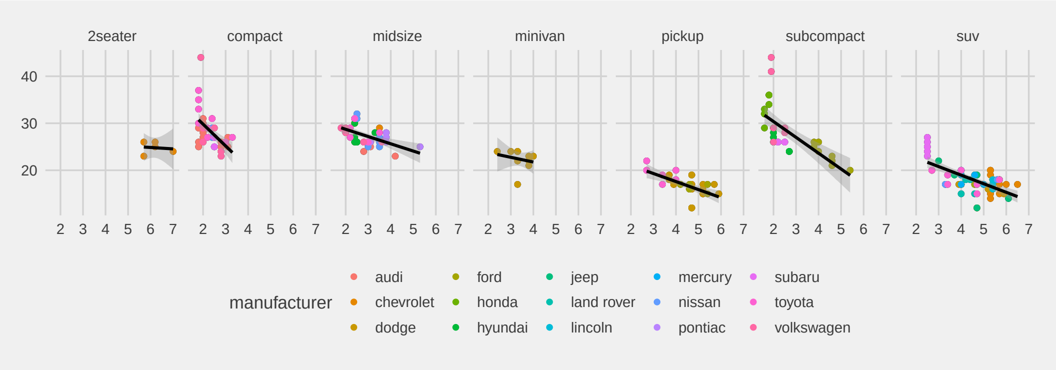
# Scatterplot: Coordinate Options II

```r
mpg.p<-mpg.p+theme_fivethirtyeight() #make it look like fivethirtyeight
mpg.p
```



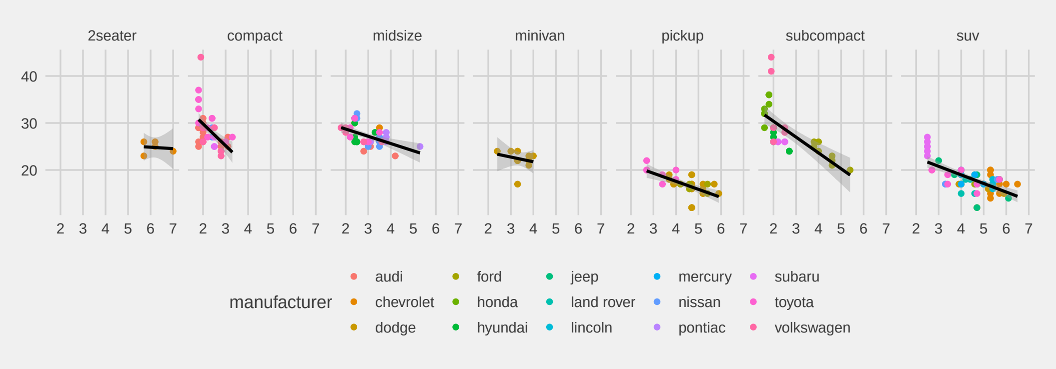
# Scatterplot: Coordinate Options: Facetting

```
# make columns of separate 'facet' figures for each class of car
mpg.p<-mpg.p+facet_grid(cols = vars(class)) # make 'columns' by variable 'class'
mpg.p
```

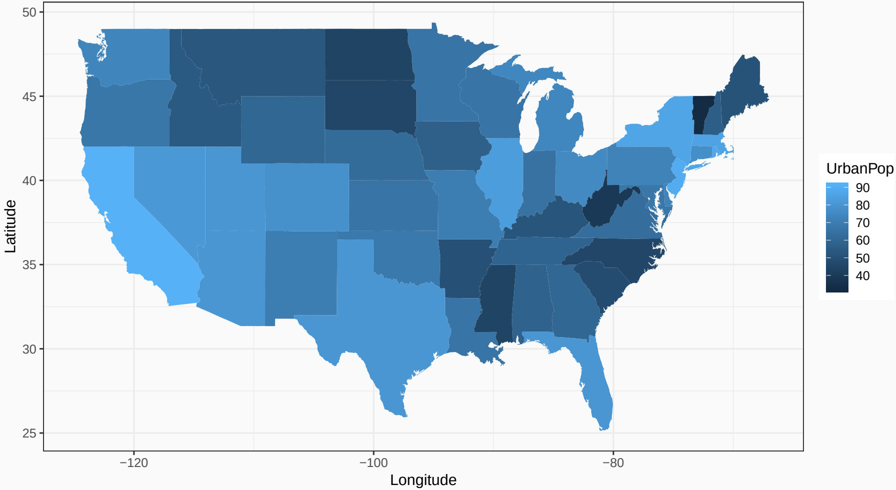


# All Together Now

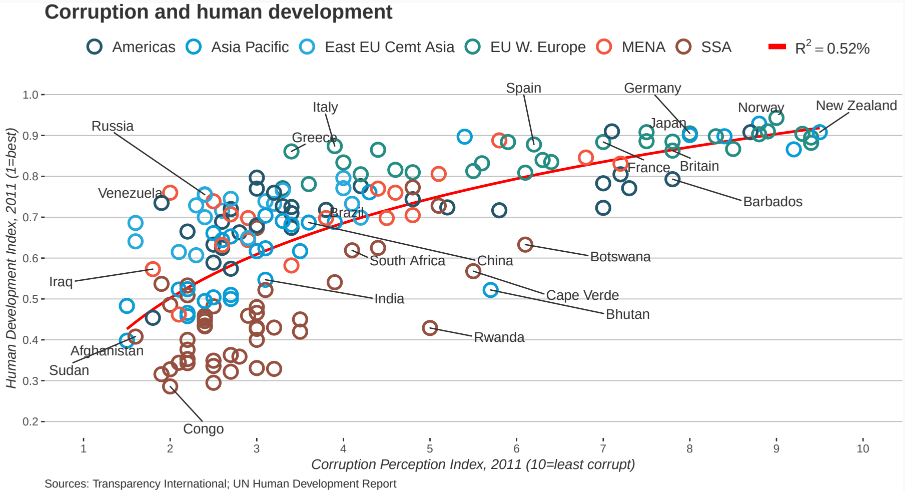
```
ggplot(data=mpg,aes(x=displ, y=hwy))+geom_point(aes(color=manufacturer))+
  geom_smooth(color=”black”,method=”lm”)+
  xlab(”Engine Displacement (Liters)”)+ylab(”Miles Per Gallon on Highway”)+
  theme_fivethirtyeight()+facet_grid(cols = vars(class))
```



# Advanced Uses of `ggplot2`: Maps (See Rmd for Code)



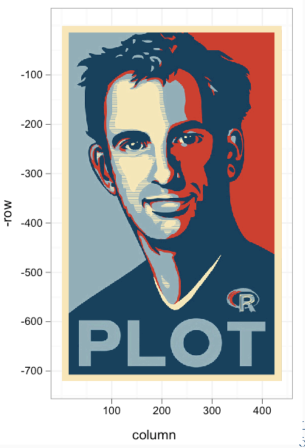
# ADVANCED USES OF ggplot2: MAPS (SEE RMD FOR CODE) II



Sources: Transparency International; UN Human Development Report

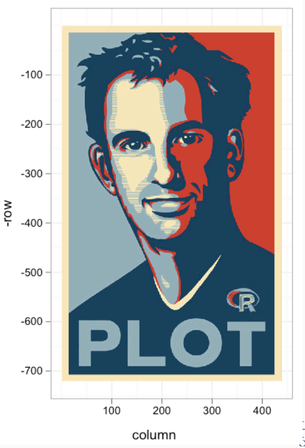
# Getting Help and Customizing to Your Heart's Content

- Every single command and option has extensive documentation with examples



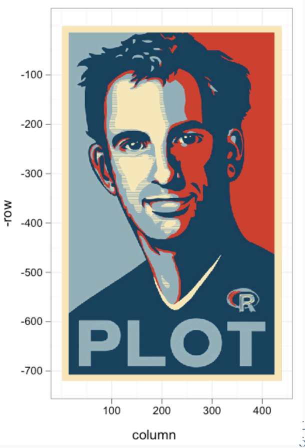
# Getting Help and Customizing to Your Heart's Content

- Every single command and option has extensive documentation with examples
    - Can type `?` in front of any function to get help and examples (e.g. `?aes()`, `?geom_smooth()`)



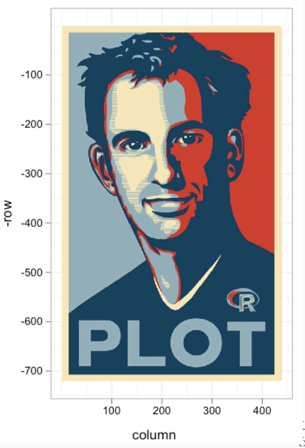
# Getting Help and Customizing to Your Heart's Content

- Every single command and option has extensive documentation with examples
  - Can type `?` in front of any function to get help and examples (e.g. `?aes()`, `?geom_smooth()`)
  - More help and examples from Wickham's R for Data Science book chapter on ggplot2



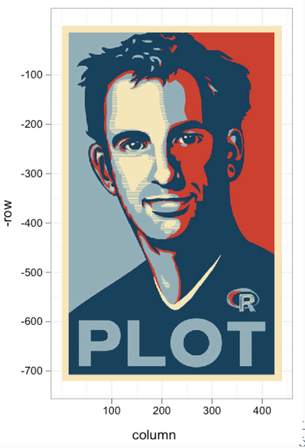
# Getting Help and Customizing to Your Heart's Content

- Every single command and option has extensive documentation with examples
    - Can type `?` in front of any function to get help and examples (e.g. `?aes()`, `?geom_smooth()`)
    - More help and examples from Wickham's R for Data Science book chapter on ggplot2
- Data work is a science, but it should also be **art**!



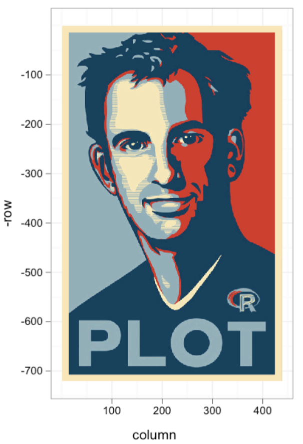
# Getting Help and Customizing to Your Heart's Content

- Every single command and option has extensive documentation with examples
  - Can type `?` in front of any function to get help and examples (e.g. `?aes()`, `?geom_smooth()`)
  - More help and examples from Wickham's R for Data Science book chapter on ggplot2
- Data work is a science, but it should also be **art**!
- I will *not* grade you on the beauty of your graphics (but we're all secretly judging!)

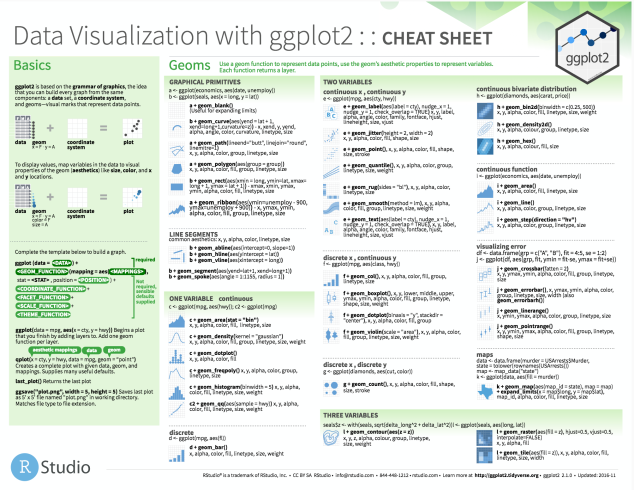


# Getting Help and Customizing to Your Heart's Content

- Every single command and option has extensive documentation with examples
  - Can type `?` in front of any function to get help and examples (e.g. `?aes()`, `?geom_smooth()`)
  - More help and examples from Wickham's R for Data Science book chapter on ggplot2
- Data work is a science, but it should also be **art**!
- I will *not* grade you on the beauty of your graphics (but we're all secretly judging!)
  - `hist()` and `plot()` are fine, you are not required to use `ggplot2` (but you really should! )



# More Help



# Data Visualization with ggplot2 : : CHEAT SHEET

ggplot2

## Basics

**ggplot2** is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.

**data** **geom** x = F · y = A + **coordinate system** = **plot**

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

**data** **geom** x = F · y = A color = F size = A + **coordinate system** = **plot**

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                        required
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
 stat = <STAT> , position = <POSITION>) +                       Not required,
<COORDINATE_FUNCTION> +                                         sensible
<FACET_FUNCTION> +                                              defaults
<SCALE_FUNCTION> +                                              supplied
<THEME_FUNCTION>
```

**ggplot(**data = mpg, **aes(**x = cty, y = hwy**))** Begins a plot that you finish by adding layers to. Add one geom function per layer.

aesthetic mappings · data · geom

**qplot(**x = cty, y = hwy, data = mpg, geom = “point")
Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

**last_plot()** Returns the last plot

**ggsave("plot.png", width = 5, height = 5)** Saves last plot as 5’ x 5' file named "plot.png" in working directory. Matches file type to file extension.

## Geoms

Use a geom function to represent data points, use the geom’s aesthetic properties to represent variables. Each function returns a layer.

### GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

**a + geom_blank()**
(Useful for expanding limits)

**b + geom_curve(**aes(yend = lat + 1, xend=long+1,curvature=z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

**a + geom_path(**lineend="butt", linejoin="round", linemitre=1)
x, y, alpha, color, group, linetype, size

**a + geom_polygon(**aes(group = group))
x, y, alpha, color, fill, group, linetype, size

**b + geom_rect(**aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

**a + geom_ribbon(**aes(ymin=unemploy - 900, ymax=unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

### LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

**b + geom_abline(**aes(intercept=0, slope=1))
**b + geom_hline(**aes(yintercept = lat))
**b + geom_vline(**aes(xintercept = long))

**b + geom_segment(**aes(yend=lat+1, xend=long+1))
**b + geom_spoke(**aes(angle = 1:1155, radius = 1))

### ONE VARIABLE continuous

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

**c + geom_area(stat = "bin")**
x, y, alpha, color, fill, linetype, size

**c + geom_density(**kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

**c + geom_dotplot()**
x, y, alpha, color, fill

**c + geom_freqpoly()** x, y, alpha, color, group, linetype, size

**c + geom_histogram(**binwidth = 5) x, y, alpha, color, fill, linetype, size, weight

**c2 + geom_qq(**aes(sample = hwy)) x, y, alpha, color, fill, linetype, size, weight

### discrete

d <- ggplot(mpg, aes(fl))

**d + geom_bar()**
x, alpha, color, fill, linetype, size, weight

### TWO VARIABLES

### continuous x , continuous y

e <- ggplot(mpg, aes(cty, hwy))

**e + geom_label(**aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE) x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

**e + geom_jitter(**height = 2, width = 2)
x, y, alpha, color, fill, shape, size

**e + geom_point()**, x, y, alpha, color, fill, shape, size, stroke

**e + geom_quantile()**, x, y, alpha, color, group, linetype, size, weight

**e + geom_rug(**sides = "bl"), x, y, alpha, color, linetype, size

**e + geom_smooth(**method = lm), x, y, alpha, color, fill, group, linetype, size, weight

**e + geom_text(**aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE), x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

### discrete x , continuous y

f <- ggplot(mpg, aes(class, hwy))

**f + geom_col()**, x, y, alpha, color, fill, group, linetype, size

**f + geom_boxplot()**, x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

**f + geom_dotplot(**binaxis = "y", stackdir = "center"), x, y, alpha, color, fill, group

**f + geom_violin(**scale = "area"), x, y, alpha, color, fill, group, linetype, size, weight

### discrete x , discrete y

g <- ggplot(diamonds, aes(cut, color))

**g + geom_count()**, x, y, alpha, color, fill, shape, size, stroke

### continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

**h + geom_bin2d(**binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

**h + geom_density2d()**
x, y, alpha, colour, group, linetype, size

**h + geom_hex()**
x, y, alpha, colour, fill, size

### continuous function

i <- ggplot(economics, aes(date, unemploy))

**i + geom_area()**
x, y, alpha, color, fill, linetype, size

**i + geom_line()**
x, y, alpha, color, group, linetype, size

**i + geom_step(direction = "hv")**
x, y, alpha, color, group, linetype, size

### visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

**j + geom_crossbar(**fatten = 2)
x, y, ymax, ymin, alpha, color, fill, group, linetype, size

**j + geom_errorbar()**, x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)

**j + geom_linerange()**
x, ymin, ymax, alpha, color, group, linetype, size

**j + geom_pointrange()**
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

### maps

data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))

**k + geom_map(**aes(map_id = state), map = map)
**+ expand_limits(**x = map$long, y = map$lat),
map_id, alpha, color, fill, linetype, size

### THREE VARIABLES

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))l <- ggplot(seals, aes(long, lat))

**l + geom_contour(aes(z = z))**
x, y, z, alpha, colour, group, linetype, size, weight

**l + geom_raster(**aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE)
x, y, alpha, fill

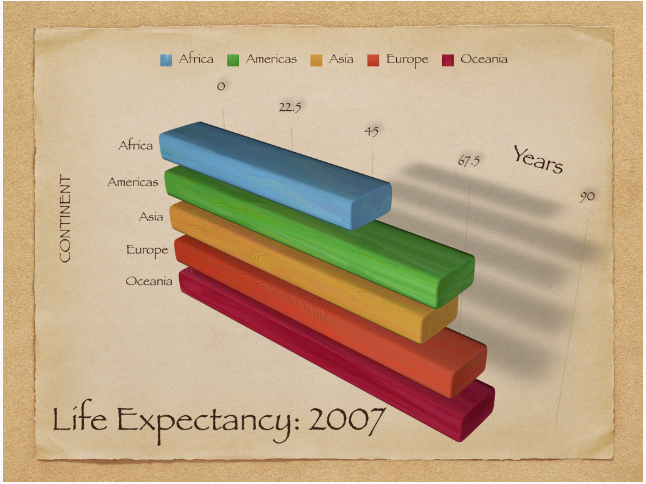
**l + geom_tile(**aes(fill = z)), x, y, alpha, color, fill, linetype, size, width

R Studio

RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at **http://ggplot2.tidyverse.org** • ggplot2 2.1.0 • Updated: 2016-11

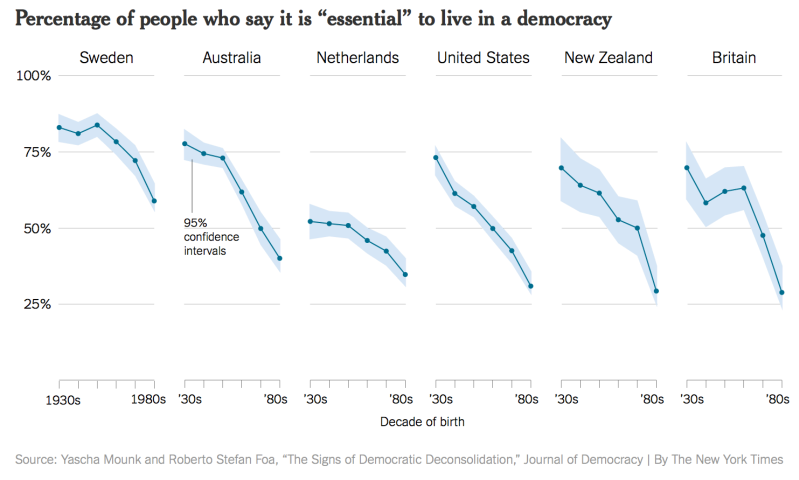
Rstudio’s Cheat Sheet

# Beautiful Graphics: Less is More



"Kill me now"

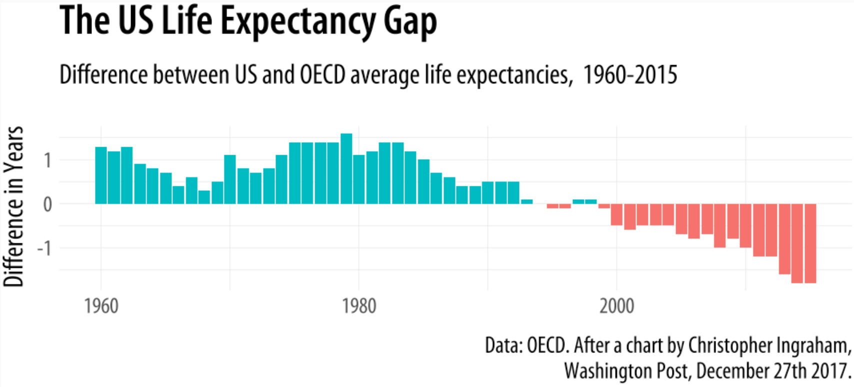
# Beautiful Graphics: Less is More II



Percentage of people who say it is “essential” to live in a democracy

Source: Yascha Mounk and Roberto Stefan Foa, “The Signs of Democratic Deconsolidation,” Journal of Democracy | By The New York Times

# Beautiful Graphics: Less is More III



Data: OECD. After a chart by Christopher Ingraham, Washington Post, December 27th 2017.

# Beautiful Graphics: Less is More IV

Less is More: Example Gif

# Resources (Best are Bolded)

On `ggplot2`

- R Studio's ggplot2 Cheat Sheet

On data visualization

# Resources (Best are Bolded)

On `ggplot2`

- R Studio's ggplot2 Cheat Sheet
- **`ggplot2`'s website reference section**

On data visualization

# Resources (Best are Bolded)

On ggplot2

- **R Studio's ggplot2 Cheat Sheet**
- **ggplot2's website reference section**
- STHDA's be awesome in ggplot2

On data visualization

# Resources (Best are Bolded)

On `ggplot2`

- **R Studio's ggplot2 Cheat Sheet**
- **`ggplot2`'s website reference section**
- STHDA's be awesome in ggplot2
- r-statistic's top 50 ggplot2 visualizations

On data visualization

# Resources (Best are Bolded)

On `ggplot2`

- **R Studio's ggplot2 Cheat Sheet**
- **`ggplot2`'s website reference section**
- STHDA's be awesome in ggplot2
- r-statistic's top 50 ggplot2 visualizations
- Hadley Wickham's R for Data Science book chapter on ggplot2

On data visualization

# Resources (Best are Bolded)

On `ggplot2`

- **R Studio's ggplot2 Cheat Sheet**
- **`ggplot2`'s website reference section**
- STHDA's be awesome in ggplot2
- r-statistic's top 50 ggplot2 visualizations
- Hadley Wickham's R for Data Science book chapter on ggplot2

On data visualization

- **Kieran Healy's Data Visualization: A Practical Guide**

# Resources (Best are Bolded)

On `ggplot2`

- **R Studio's ggplot2 Cheat Sheet**
- **`ggplot2`'s website reference section**
- STHDA's be awesome in ggplot2
- r-statistic's top 50 ggplot2 visualizations
- Hadley Wickham's R for Data Science book chapter on ggplot2

On data visualization

- **Kieran Healy's Data Visualization: A Practical Guide**
- **Claus Wilke's Fundamentals of Data Visualization**

# Resources (Best are Bolded)

On `ggplot2`

- **R Studio's ggplot2 Cheat Sheet**
- **`ggplot2`'s website reference section**
- STHDA's be awesome in ggplot2
- r-statistic's top 50 ggplot2 visualizations
- Hadley Wickham's R for Data Science book chapter on ggplot2

On data visualization

- **Kieran Healy's Data Visualization: A Practical Guide**
- **Claus Wilke's Fundamentals of Data Visualization**
- PolicyViz Better Presentations

# Resources (Best are Bolded)

On `ggplot2`

- **R Studio's ggplot2 Cheat Sheet**
- **`ggplot2`'s website reference section**
- STHDA's be awesome in ggplot2
- r-statistic's top 50 ggplot2 visualizations
- Hadley Wickham's R for Data Science book chapter on ggplot2

On data visualization

- **Kieran Healy's Data Visualization: A Practical Guide**
- **Claus Wilke's Fundamentals of Data Visualization**
- PolicyViz Better Presentations
- Karl Broman's How to Display Data Badly

# Resources (Best are Bolded)

On `ggplot2`

- **R Studio's ggplot2 Cheat Sheet**
- **`ggplot2`'s website reference section**
- STHDA's be awesome in ggplot2
- r-statistic's top 50 ggplot2 visualizations
- Hadley Wickham's R for Data Science book chapter on ggplot2

On data visualization

- **Kieran Healy's Data Visualization: A Practical Guide**
- **Claus Wilke's Fundamentals of Data Visualization**
- PolicyViz Better Presentations
- Karl Broman's How to Display Data Badly
- I Want Hue