432 Class 14 Slides

github.com/THOMASELOVE/432-2018

2018-03-01

Setup

```
library(skimr); library(MASS)
library(robustbase); library(quantreg)
library(lmtest); library(sandwich)
library(boot); library(broom)
library(rms)
library(tidyverse)

decim <- function(x, k) format(round(x, k), nsmall=k)
```

Today's Materials

- Crime in the United States
- Sandwich Estimation of Standard Errors
- Bootstrapping Regression Coefficients

Next Time

1. Robust Linear Regression Methods with Huber weights
2. Robust Linear Regression with bisquare weights (biweights)
3. Bounded Influence Regression & Least Trimmed Squares
4. Penalized Least Squares using `ols` in the `rms` package
5. Quantile Regression on the Median

Some Motivating Graphs

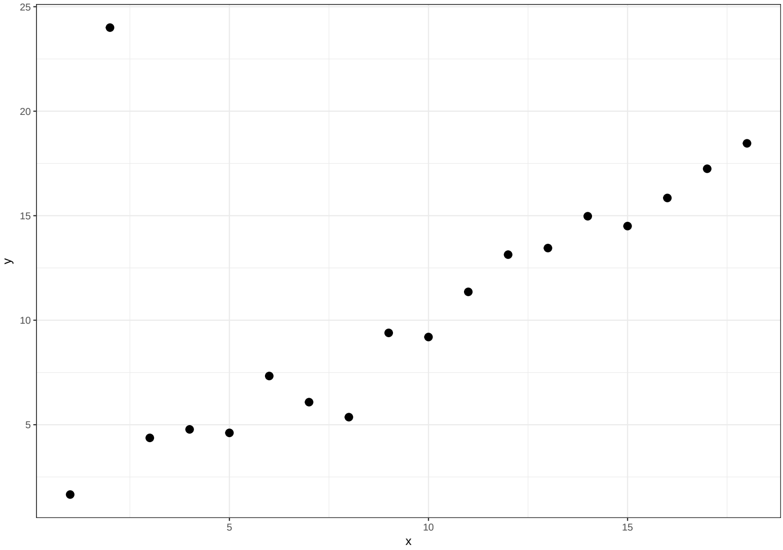
A Simple Regression Model

Suppose we were looking at a simple regression on a new batch of data.

```
set.seed(20170421)
newd <- data_frame(x = 1:18, y = rnorm(x, mean = x))
newd$y[2] <- 24

head(newd)
```

```
# A tibble: 6 x 2
      x     y
  <int> <dbl>
1     1  1.66
2     2 24.0
3     3  4.37
4     4  4.78
5     5  4.61
6     6  7.33
```

Scatterplot of `newd`



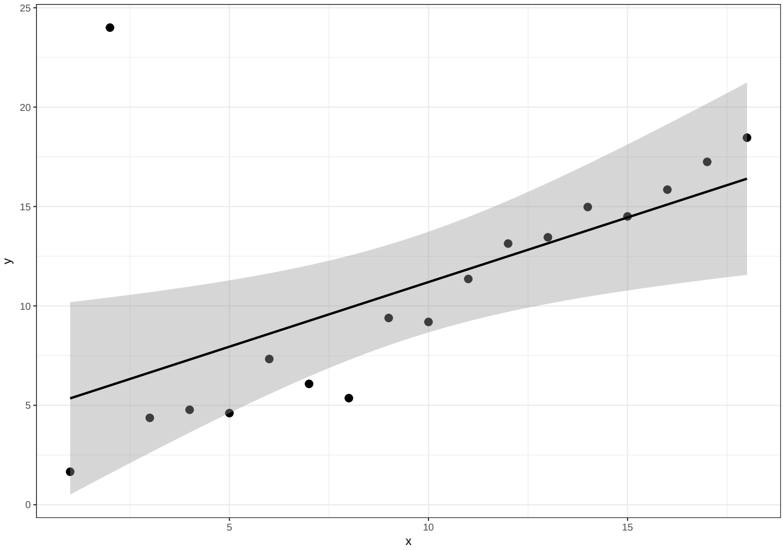
Code for Last Slide and Next Slide

```
(p <- ggplot(newd, aes(x = x, y = y)) +
    geom_point(size = 3) +
    theme_bw()
)
```

Add OLS line

```
p + geom_smooth(method = "lm", col = "black")
```

OLS regression line



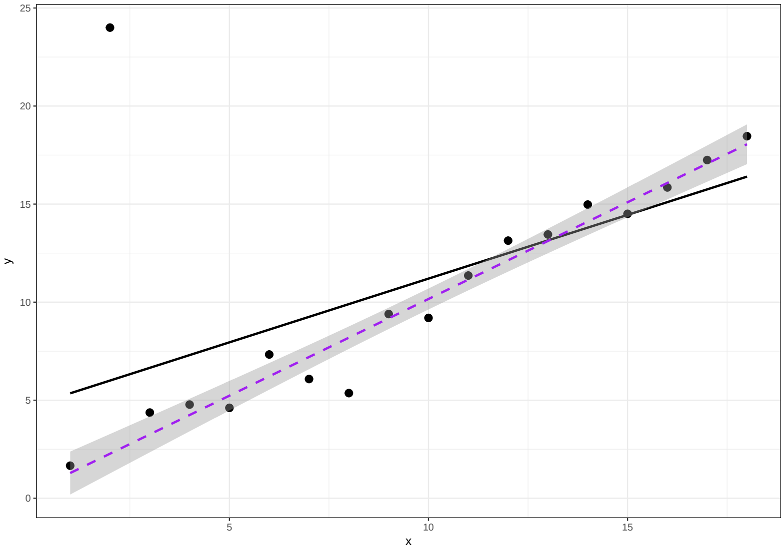
That outlier seems like a problem.

Suppose we compare the ordinary least squares regression line we saw above to a new line, fit without including the outlier at the top left of the plot.

Code for next plot

```
p + geom_smooth(method = "lm", se = FALSE, col = "black") +
    geom_smooth(method = "lm", data = filter(newd, y < 20),
                col = "purple", linetype = "dashed")
```

New Plot showing the outlier effect

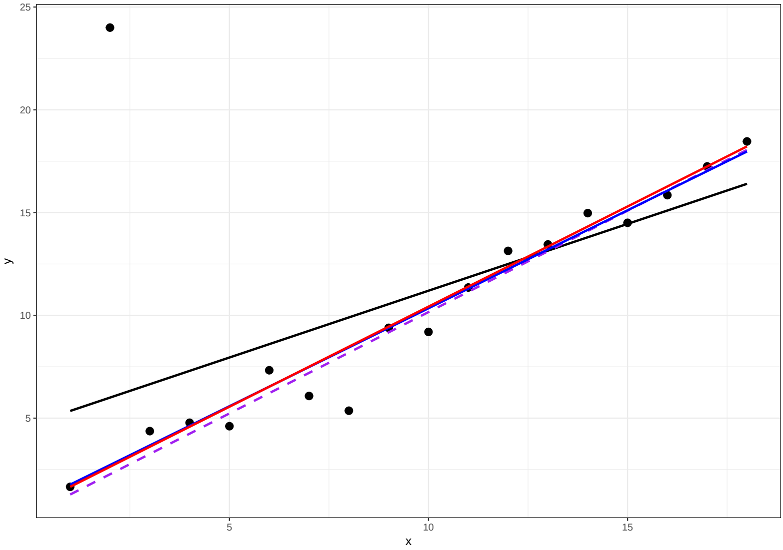


Add robust regression lines

Now, let's add a line from a robust regression [the robust (Huber weights) line] with the "rlm" method, and a quantile regression line using the "rq" method.

- Linear model (OLS) will be in black
- Linear model (OLS) without the outlier in dashed purple
- Robust Linear Model (via `rlm`) in blue
- Quantile Regression Model (via `rq`) in red

Comparison of models



Comparison of models (code)

```r
p + geom_smooth(method = "lm", col = "black", se = FALSE) +
    geom_smooth(method = "lm", data = filter(newd, y < 20),
                col = "purple", se = FALSE,
                linetype = "dashed") +
    geom_smooth(method = "rlm", col = "blue", se = FALSE) +
    geom_smooth(method = "rq", col = "red", se = FALSE)
```

Sources and Resources

Key sources for this document include:

- http://stats.idre.ucla.edu/r/dae/robust-regression/
- http://www.alastairsanderson.com/R/tutorials/robust-regression-in-R/
- John Fox's appendix on Applied Robust Regression
- https://cran.r-project.org/web/packages/robust/robust.pdf
- https://cran.r-project.org/web/packages/rms/rms.pdf
- http://www.statmethods.net/advstats/bootstrapping.html

The `crimestat` Data

Data Source

The `crimestat` data gathered here refer to 2016, mainly, and were obtained from:

- http://www.worldatlas.com/articles/the-most-dangerous-states-in-the-u-s.html
- https://www.statista.com/statistics/242302/percentage-of-single-mother-households-in-the-us-by-state/
- and a few different Wikipedia sites,

but the use of these data in this context is due to an older data set that appears in *Statistical Methods for Social Sciences*, Third Edition by Alan Agresti and Barbara Finlay (Prentice Hall, 1997), and which is the primary example at http://stats.idre.ucla.edu/r/dae/robust-regression/

The `crimestat` data set

For each of 51 states (including the District of Columbia), we have the state's ID number, postal abbreviation and full name, as well as:

- **crime** - the violent crime rate per 100,000 people
- **poverty** - the official poverty rate (% of people living in poverty in the state/district) in 2014
- **single** - the percentage of households in the state/district led by a female householder with no spouse present and with her own children under 18 years living in the household in 2016
- **trump** - whether Donald Trump won the popular vote in the 2016 presidential election in that state/district (which we'll ignore for today)

The crimestat data set

```
crimestat <- read.csv("crimestat.csv") %>% tbl_df
crimestat
```

```
# A tibble: 51 x 7
     sid state crime poverty single trump state.full
   <int> <fct> <dbl>   <dbl>  <dbl> <int> <fct>
 1     1 AL      427    19.2   9.02     1 Alabama
 2     2 AK      636    11.4   7.63     1 Alaska
 3     3 AZ      400    18.2   8.31     1 Arizona
 4     4 AR      480    18.7   9.41     1 Arkansas
 5     5 CA      396    16.4   7.25     0 California
 6     6 CO      309    12.1   6.75     0 Colorado
 7     7 CT      237    10.8   8.04     0 Connecticut
 8     8 DE      489    13.0   6.52     0 Delaware
 9     9 DC     1244    18.4   8.41     0 District of Colu~
10    10 FL      540    16.6   8.29     1 Florida
# ... with 41 more rows
```

Numerical Summaries

```
crimestat %>% select(poverty, single, crime) %>% skim
```

```
Skim summary statistics
 n obs: 51
 n variables: 3

Variable type: numeric
 variable missing complete  n   mean      sd     p0     p25 median    p75    p100     hist
    crime       0        51 51 364.41 179.05 99.3   260.2  326.5  427.35 1244.4
  poverty       0        51 51  14.87   3.08  9.2   12.15  14.8   17.2    21.9
   single       0        51 51   7.69   1.61  4.48   6.75   7.63   8.5    11.59
```

Modeling `crime` with `poverty` and `single`

Our main goal will be to build a linear regression model to predict **crime** using **poverty** and **single**.

We'll start by building an ordinary least squares model on the two predictors (after centering them, so that the intercept is meaningful) and looking at some diagnostics.

```
crimestat <- crimestat %>%
    mutate(pov_c = poverty - mean(poverty),
           single_c = single - mean(single))
```

Fitting an OLS model

Our first model `mod1` using OLS

```
(mod1 <- lm(crime ~ pov_c + single_c, data = crimestat))
```

```
Call:
lm(formula = crime ~ pov_c + single_c, data = crimestat)

Coefficients:
(Intercept)        pov_c     single_c
     364.41        16.11        23.84
```

```
confint(mod1)
```

```
                 2.5 %    97.5 %
(Intercept) 318.296950 410.51481
pov_c        -3.218922  35.44816
single_c    -13.121152  60.80677
```

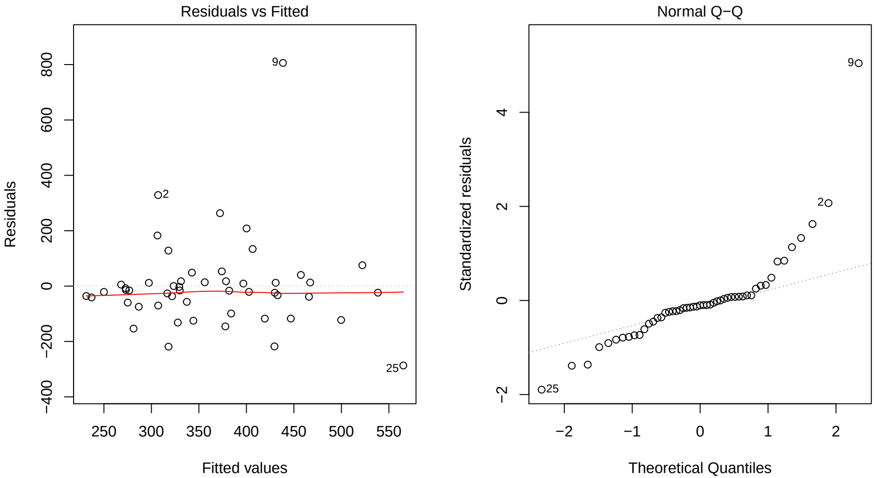
glance(mod1) and tidy(mod1)

```
  r.squared adj.r.squared    sigma statistic     p.value df
1  0.196879     0.1634156 163.771  5.883417 0.005184941  3
     logLik      AIC     BIC deviance df.residual
1 -330.8419 669.6837 677.411  1287405          48
```

```
         term  estimate std.error statistic      p.value
1 (Intercept) 364.40588 22.932525 15.890351 9.475916e-21
2       pov_c  16.11462  9.615642  1.675876 1.002655e-01
3    single_c  23.84281 18.384226  1.296917 2.008596e-01
```

Neither predictor meets our usual standard of having an estimate which is at least twice as large as the standard error.

Residual Plots for our model?



Potential Problems: States 9, 25 and maybe 2

Who are the outlier states?

Points 9, 25 and maybe 2 look like they could be problematic. Which states are those?

```
filter(crimestat, row_number() %in% c(9, 25, 2))
```

```
# A tibble: 3 x 9
    sid state crime poverty single trump state.full   pov_c
  <int> <fct> <dbl>   <dbl>  <dbl> <int> <fct>        <dbl>
1     2 AK      636    11.4   7.63     1 Alaska       -3.47
2     9 DC     1244    18.4   8.41     0 District of~  3.53
3    25 MS      278    21.9  11.4      1 Mississippi   7.03
# ... with 1 more variable: single_c <dbl>
```

Augmented Data Set with OLS results

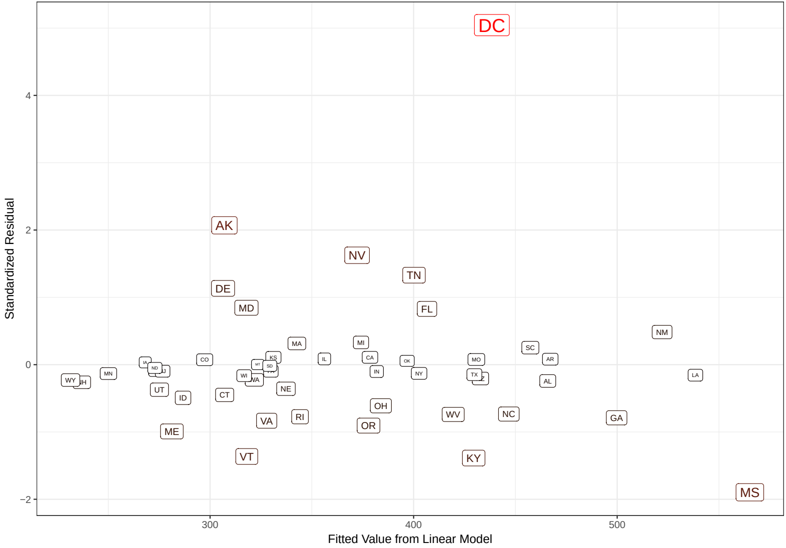
```
crime_with_mod1 <- augment(mod1, crimestat)
head(crime_with_mod1, 3)
```

```
  sid state crime poverty single trump state.full     pov_c
1   1    AL 427.4    19.2   9.02     1    Alabama  4.327451
2   2    AK 635.8    11.4   7.63     1     Alaska -3.472549
3   3    AZ 399.9    18.2   8.31     1    Arizona  3.327451
     single_c  .fitted  .se.fit    .resid       .hat
1  1.33117647 465.8801 39.84145 -38.48010 0.05918290
2 -0.05882353 307.0446 39.96271 328.75545 0.05954371
3  0.62117647 432.8371 34.99599 -32.93709 0.04566282
    .sigma      .cooksd .std.resid
1 165.4029 0.0012304477 -0.2422405
2 157.9444 0.0904293530  2.0699826
3 165.4310 0.0006759806 -0.2058720
```

Standardized Residuals vs. Fitted Values (code)

```
ggplot(crime_with_mod1, aes(x = .fitted, y = .std.resid,
                            size = abs(.std.resid),
                            col = -abs(.std.resid))) +
    geom_label(aes(label = state)) +
    guides(size = FALSE, col = FALSE) +
    scale_color_continuous(low = "red", high = "black") +
    theme_bw() +
    labs(x = "Fitted Value from Linear Model",
         y = "Standardized Residual")
```

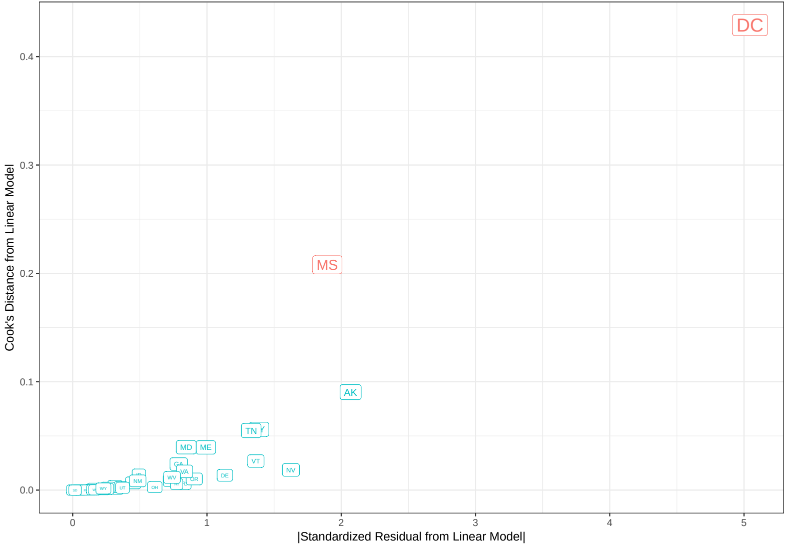
Standardized Residuals vs. Fitted Values



Cook’s Distance vs. |Standardized Residuals| (code)

```
ggplot(crime_with_mod1, aes(x = abs(.std.resid), y = .cooksd,
                            size = .cooksd,
                            col = .cooksd < 0.2)) +
    geom_label(aes(label = state)) +
    guides(size = FALSE, col = FALSE) +
    scale_color_discrete() +
    theme_bw() +
    labs(x = "|Standardized Residual from Linear Model|",
         y = "Cook's Distance from Linear Model")
```

Cook's Distance vs. |Standardized Residuals|



What about just "robustifying" the standard errors of the coefficients?

Would Sandwich Estimation help for our original model?

from David Freedman:

> *The "Huber Sandwich Estimator" (for which Peter Huber is not to be blamed) can be used to estimate the variance of the MLE (maximum likelihood estimate) when the underlying model is incorrect. If the model is nearly correct, so are the usual standard errors, and robustification is unlikely to help much. On the other hand, if the model is seriously in error, the sandwich may help on the variance side, but the parameters being estimated by the MLE are likely to be meaningless.*

Sandwich estimation is mainly used to help address heteroscedasticity in linear regression, not so much with outliers. So, I doubt it will get us all the way to significance, but let's see...

Using `lmtest::coeftest` to get standard errors

```r
mod1 <- lm(crime ~ pov_c + single_c, data = crimestat)

# requires lmtest package
coeftest(mod1)
```

```

t test of coefficients:

            Estimate Std. Error t value Pr(>|t|)
(Intercept) 364.4059    22.9325 15.8904   <2e-16 ***
pov_c        16.1146     9.6156  1.6759   0.1003
single_c     23.8428    18.3842  1.2969   0.2009
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Using `coeftest` for Robust (Huber) Standard Errors

```r
# requires lmtest and sandwich packages
coeftest(mod1, vcov = sandwich)
```

```

t test of coefficients:

            Estimate Std. Error t value Pr(>|t|)
(Intercept)  364.406     22.248 16.3794  < 2e-16 ***
pov_c         16.115     10.898  1.4787  0.14574
single_c      23.843     13.948  1.7095  0.09382 .
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Using coefci for Robust (Huber) Standard Errors

```r
# requires lmtest and sandwich packages
coefci(mod1, vcov = sandwich, level = 0.95)
```

```
                2.5 %    97.5 %
(Intercept) 319.673648 409.13812
pov_c        -5.796371  38.02561
single_c     -4.200619  51.88624
```

Bootstrapping Regression Coefficients

Bootstrapped Regression Coefficient Estimates

I'd be happier using bootstrapped estimates in this setting.

Source: statmethods.net link

```r
# requires boot package
# build function to obtain regression weights
bs <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- lm(formula, data=d)
  return(coef(fit))
}

# now do R = 1000 replications
set.seed(432222)
results <- boot(data=crimestat, statistic=bs,
    R=1000, formula = crime ~ pov_c + single_c)
```

Bootstrapping Estimates with 1,000 replications

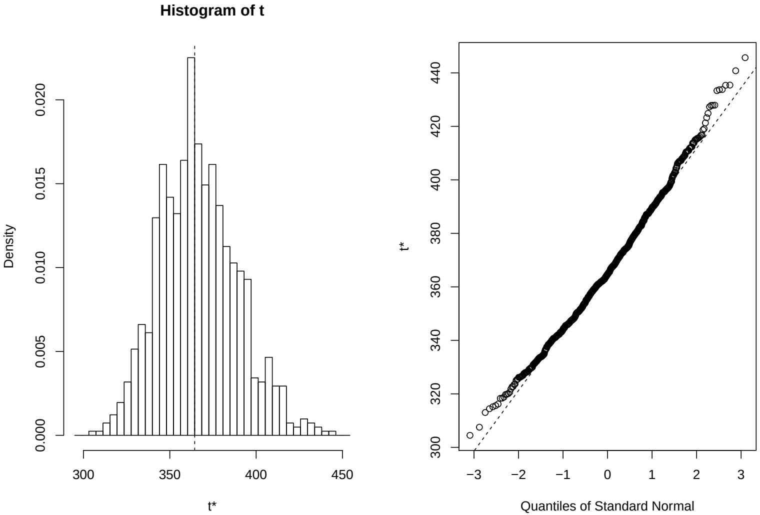
```
results
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = crimestat, statistic = bs, R = 1000, formula = cri
    pov_c + single_c)


Bootstrap Statistics :
     original    bias    std. error
t1* 364.40588  2.107998    22.63315
t2*  16.11462  1.235138    11.58621
t3*  23.84281 -1.224433    15.51142
```

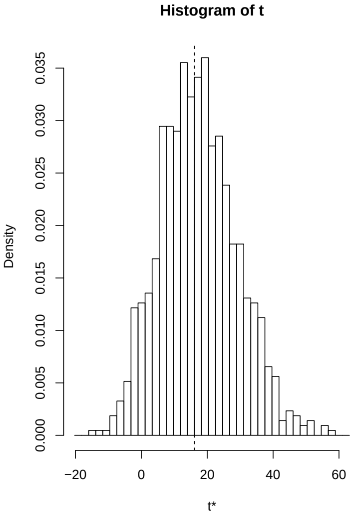
Plots of Bootstrapped Estimates (next 3 slides)

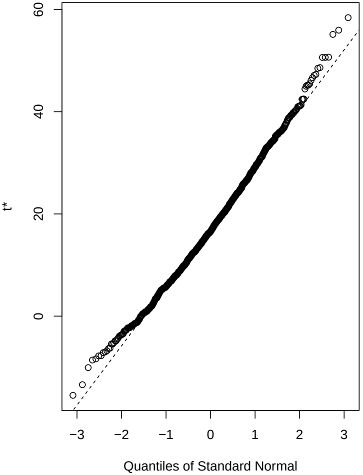
```r
plot(results, index = 1) # intercept
plot(results, index = 2) # pov_c slope
plot(results, index = 3) # single_c slope
```

Intercept Estimates (bootstrap)

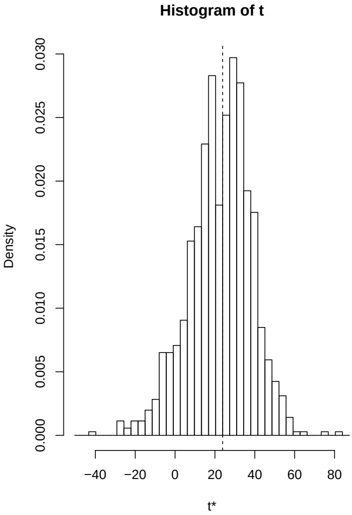


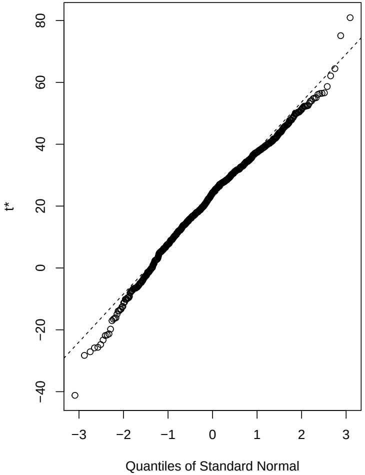
pov_c Slope Estimates (bootstrap)





`single_c` Slope Estimates (bootstrap)





Obtain 95% Confidence Intervals

```
boot.ci(results, type="bca", index=1) # intercept
boot.ci(results, type="bca", index=2) # pov_c slope
boot.ci(results, type="bca", index=3) # single_c slope
```

95% Bootstrap CI for Intercept

```r
boot.ci(results, type="bca", index=1) # intercept
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "bca", index = 1)

Intervals :
Level       BCa
95%   (328.9, 421.8 )
Calculations and Intervals on Original Scale
```

95% Bootstrap CI for Slope of pov_c

```
boot.ci(results, type="bca", index=2) # pov_c slope
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "bca", index = 2)

Intervals :
Level       BCa
95%   (-2.90, 41.12 )
Calculations and Intervals on Original Scale
```

95% Bootstrap CI for Slope of `single_c`

```r
boot.ci(results, type="bca", index=3) # single_c slope
```

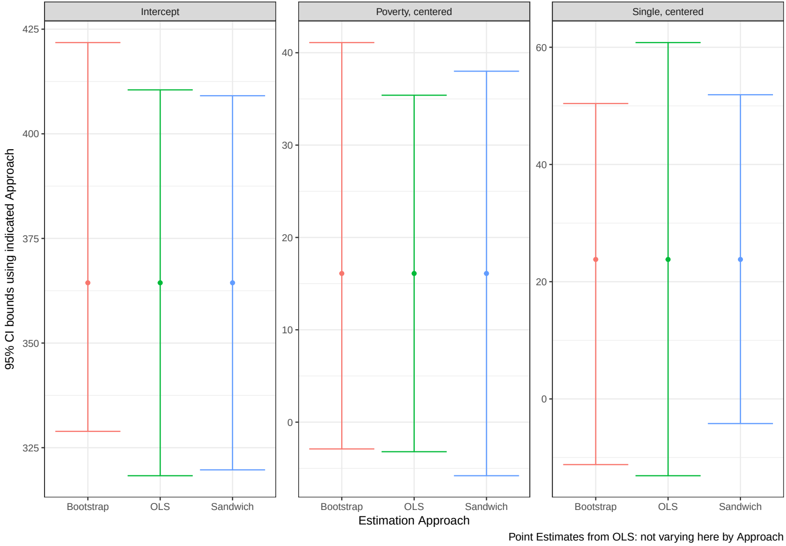
```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "bca", index = 3)

Intervals :
Level       BCa
95%   (-11.19,  50.37 )
Calculations and Intervals on Original Scale
```

Standard, Sandwich and Bootstrapped 95% CIs for the Coefficients of our OLS model

Fitted OLS Model: `crime` = 364.4 + 16.1 x `pov_c` + 23.8 x `single_c`

Fit	Intercept CI	`pov_c` CI	`single_c` CI
OLS	(318.3, 410.5)	(-3.2, 35.4)	(-13.1, 60.8)
OLS with sandwich	(319.7, 409.1)	(-5.8, 38.0)	(-4.2, 51.9)
OLS, bootstrapped	(328.9, 421.8)	(-2.9, 41.1)	(-11.2, 50.4)

Comparison Plot (code, next two slides)



Code for plot on prior slide (part 1)

```
res_class14 <- data_frame(
    approach = c(rep("OLS",3), rep("Sandwich",3),
                 rep("Bootstrap",3)),
    parameter = c(rep(c("Intercept", "Poverty, centered",
                        "Single, centered"),3)),
    estimate = c(rep(c(364.4, 16.1, 23.8),3)),
    conf.low = c(318.3, -3.2, -13.1, 319.7, -5.8, -4.2,
                 328.9, -2.9, -11.2),
    conf.high = c(410.5, 35.4, 60.8, 409.1, 38.0, 51.9,
                  421.8, 41.1, 50.4)
)
```

Code for plot on prior slide (part 2)

```
ggplot(res_class14, aes(x = approach, y = estimate,
                        col = approach)) +
    geom_point() +
    geom_errorbar(aes(ymin = conf.low, ymax = conf.high)) +
    labs(x = "Estimation Approach",
         y = "95% CI bounds using indicated Approach",
         caption = "Point Estimates from OLS:
         not varying here by Approach") +
    guides(col = FALSE) +
    theme_bw() +
    facet_wrap(~ parameter, scales = "free_y")
```

Good luck on the Quiz!

Due Monday at Noon.