

# Quant 2, Lab 2

## Bootstrapping & Cluster Analysis

Sylvan Zheng

# Outline

- ▶ Tip of the week: Pacman

# Outline

- ▶ Tip of the week: Pacman
- ▶ Bootstrapping

# Outline

- ▶ Tip of the week: Pacman
- ▶ Bootstrapping
- ▶ Clustered analysis

# Outline

- ▶ Tip of the week: Pacman
- ▶ Bootstrapping
- ▶ Clustered analysis
- ▶ Bootstrapping x Clustered analysis

# Outline

- ▶ Tip of the week: Pacman
- ▶ Bootstrapping
- ▶ Clustered analysis
- ▶ Bootstrapping x Clustered analysis
- ▶ Intro to git (if we have time)

# Pacman

- ▶ R Package for package management

# Pacman

- ▶ R Package for package management
- ▶ `pacman::p_load`: loads or installs automatically



# Pacman

- ▶ R Package for package management
- ▶ `pacman::p_load`: loads or installs automatically
- ▶ `pacman::p_install_version`: installs a specific version

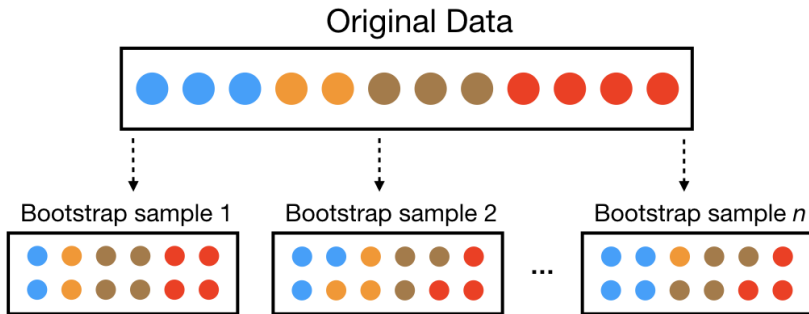
# Bootstrapping

- ▶ I heard you like samples

# Bootstrapping

- ▶ I heard you like samples
- ▶ So I made a sample of samples from a sample

# Bootstrapping



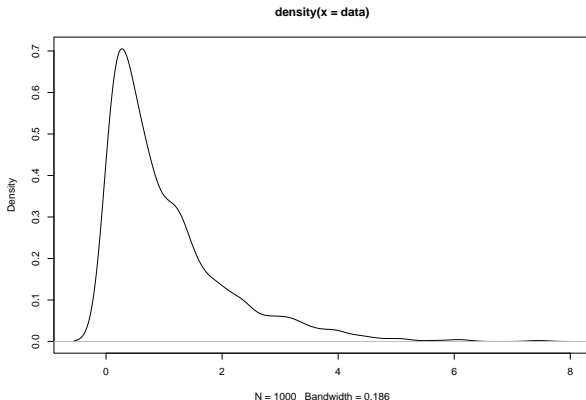
# Bootstrapping

- ▶ Get confidence intervals for estimators we can't (or don't want to) derive variances for

# Bootstrap Example: Quantile CIs

- ▶ Let's say we are interested in estimating the 90th percentile value

```
N <- 1000  
data <- mysterious_dgp(N)  
density(data) %>% plot()
```



# Bootstrap Example: Quantile CIs

- ▶ We can estimate the sample 90th quantile...

```
quantile(data, 0.9)
```

```
##          90%
```

```
## 2.402436
```

# Bootstrap Example: Quantile CIs

- ▶ We can estimate the sample 90th quantile...
- ▶ But how would we calculate the confidence interval on this estimate?

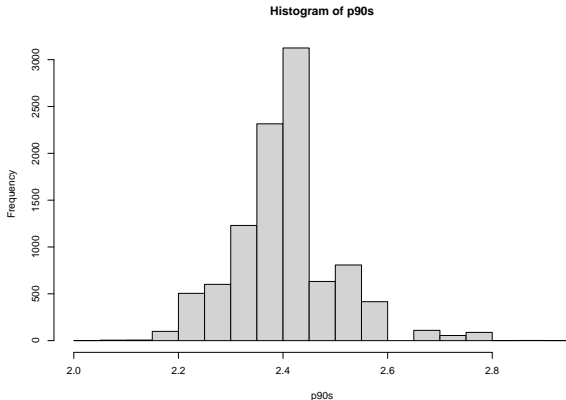
```
quantile(data, 0.9)
```

```
##          90%  
## 2.402436
```



# Bootstrap Example: Quantile CIs

```
N.bs <- 10000  
# Draw 10K samples from our data with replacement  
sample.of.samples <- map(1:N.bs, \(x) sample(data, N, replace = T))  
  
# Calculate the p90 for each sample  
p90s <- map(sample.of.samples, \(x) quantile(x, 0.9)) %>% unlist()  
hist(p90s)
```



# Bootstrap Example: Quantile CIs

- Estimate the CI with the bootstrap sample statistics

```
quantile(p90s, c(0.025, 0.975))
```

```
##      2.5%      97.5%
```

```
## 2.222548 2.677523
```

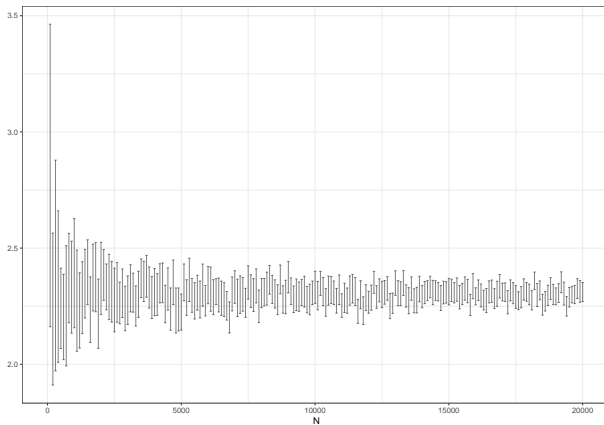
## Just for comparison...

```
samples <- map(1:N.bs, \(x) mysterious_dgp(N))  
  
true.p90s <- map(samples, \(x) quantile(x, 0.9)) %>% unlist()  
  
quantile(true.p90s, c(0.025, 0.975))
```

```
##      2.5%      97.5%  
## 2.119951 2.482374
```

- Bias corrected and/or studentized bootstrapping are also options

# Bootstrap validity is also asymptotic



# Cluster Analysis

- ▶ Data and treatments often **clustered**

# Cluster Analysis

- ▶ Data and treatments often **clustered**
- ▶ State wide minimum wage policy change

# Cluster Analysis

- ▶ Data and treatments often **clustered**
- ▶ State wide minimum wage policy change
- ▶ Classroom wide distribution of technology

# Cluster Analysis

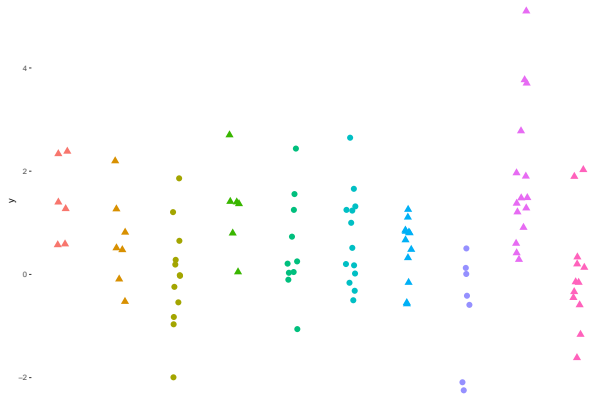
- ▶ Data and treatments often **clustered**
- ▶ State wide minimum wage policy change
- ▶ Classroom wide distribution of technology
- ▶ Etc.



# Cluster simulation

```
make_clustered_data <- function(G, N) {  
  g <- sample(1:G, size = N, replace = T)  
  treat.g <- sample(c(0, 1), size = G, replace = T)  
  while (is.na(sd(treat.g))) {  
    treat.g <- sample(c(0, 1), size = G, replace = T)  
  }  
  treat.i <- sapply(g, \(x) treat.g[x])  
  effect.g <- rnorm(1:G)  
  g.i <- sapply(g, \(x) effect.g[x])  
  y <- rnorm(N) + g.i + 0.5 * treat.i  
  data.frame(  
    g = g,  
    treat = treat.i,  
    y = y  
  )  
}
```

# Cluster simulation



# Cluster simulation

```
model <- lm(y ~ treat, data = df)
summary(model) %>% tidy()
```

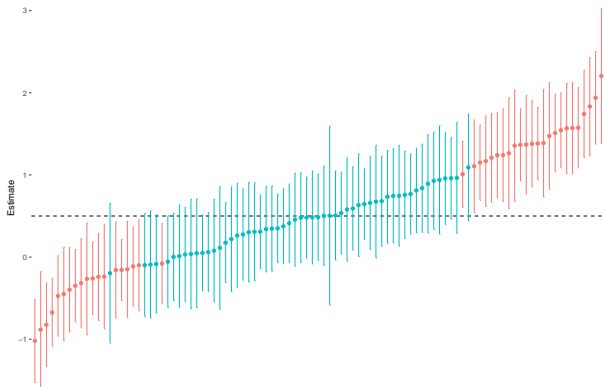
```
## # A tibble: 2 x 5
##   term          estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)    0.221     0.179      1.23 0.220
## 2 treat          0.731     0.235      3.12 0.00241
```

# Cluster simulation

```
make_sim <- function(x) {  
  df <- make_clustered_data(G = 10, N = 100)  
  model <- summary(lm(y ~ treat, data = df))$coefficients["treat", ]  
  model  
}  
res <- map(1:100, make_sim) %>%  
  bind_rows() %>%  
  mutate(  
    lo = Estimate - 1.96 * `Std. Error`,  
    hi = Estimate + 1.96 * `Std. Error`,  
    cover = lo < 0.5 & hi > 0.5,  
    i = 1:100  
  )
```

# Cluster simulation

- Only 56% cover the true value



# Cluster Bootstrapping: Vanilla

- ▶ Let's try to bootstrap SEs for  $\beta_{treat}$

```
vanilla_bootstrap <- function(a) {  
  resampled.data <- df %>% sample_frac(1, replace = T)  
  tidy(lm(y ~ treat, data = resampled.data)) %>% filter(term == "trea  
}  
vb.results <- map(1:1000, vanilla_bootstrap) %>%  
  bind_rows()  
sd(vb.results$estimate)
```

```
## [1] 0.2306454
```

# Cluster Bootstrapping: Block / Pair

- ▶ Bootstrap procedure with clustered data needs to reflect the clustered structure

```
get_block_bs_sample <- function(df) {  
  sample(size = n_clusters, df$g, replace = T) %>%  
    map(\(x) filter(df, g == x)) %>%  
    bind_rows()  
}
```

# Cluster Bootstrapping: Block / Pair

- ▶ Bootstrap procedure with clustered data needs to reflect the clustered structure
- ▶ Block/Pair bootstrapping resamples at the **cluster** level

```
get_block_bs_sample <- function(df) {  
  sample(size = n_clusters, df$g, replace = T) %>%  
    map(\(x) filter(df, g == x)) %>%  
    bind_rows()  
}
```



## Cluster Bootstrapping: Block / Pair

```
bb.estimates <- map(1:1001, \(a) {  
  resampled.data <- get_block_bs_sample(df)  
  tidy(lm(y ~ treat, data = resampled.data)) %>% filter(term == "trea  
}) %>%  
  bind_rows() %>%  
  .$estimate
```

# Cluster Bootstrapping: Block / Pair

```
sd(bb.estimates)
```

```
## [1] NA
```

# Cluster Bootstrapping: Block / Pair

```
table(get_block_bs_sample(df)$treat)
```

```
##  
##  0  1  
## 74 37
```

```
table(get_block_bs_sample(df)$treat)
```

```
##  
##  0  1  
## 14 94
```

## Cluster Bootstrapping: Wild

- ▶ Let's not draw conventional bootstrap samples

## Cluster Bootstrapping: Wild

- ▶ Let's not draw conventional bootstrap samples
- ▶ Instead, **randomize the sign of the residual** at the **cluster level**

# Cluster Bootstrapping: Wild

- ▶ Let's not draw conventional bootstrap samples
- ▶ Instead, **randomize the sign of the residual** at the **cluster level**
- ▶ Then, construct  $y_{\text{star}} = y_{\text{pred}} + \text{randomized\_residual}$  and estimate  $y_{\text{star}} \sim \text{treat}$

# Cluster Bootstrapping: Wild

- ▶ Let's not draw conventional bootstrap samples
- ▶ Instead, **randomize the sign of the residual** at the **cluster level**
- ▶ Then, construct  $y_{\text{star}} = y_{\text{pred}} + \text{randomized\_residual}$  and estimate  $y_{\text{star}} \sim \text{treat}$
- ▶ Do this  $N_{\text{boot}}$  times

# Cluster Bootstrapping: Wild

- ▶ Let's not draw conventional bootstrap samples
- ▶ Instead, **randomize the sign of the residual** at the **cluster level**
- ▶ Then, construct  $y_{\text{star}} = y_{\text{pred}} + \text{randomized\_residual}$  and estimate  $y_{\text{star}} \sim \text{treat}$
- ▶ Do this  $N_{\text{boot}}$  times
- ▶ Lab activity: implement in pairs using `wild_bs.R` as a starting template



# Cluster Bootstrapping: Wild

```
model <- lm(y ~ treat, df)
df$y_pred <- predict(model, df)
df$residuals <- df$y - df$y_pred
wild_bootstrap_3 <- function(a) {
  flip <- sample(c(-1, 1), n_clusters, replace = T)
  bs.resid.sign <- sapply(df$g, \(x) flip[x])
  df$y_star <- df$y_pred + df$residuals * bs.resid.sign

  tidy(lm(y_star ~ treat, data = df)) %>% filter(term == "treat")
}
wb3.estimates <- lapply(1:2000, wild_bootstrap_3) %>% bind_rows()
```

# Cluster Bootstrapping: Wild

```
sd(wb3.estimate$estimate)
```

```
## [1] 0.4036346
```

# Wild Cluster via Sandwich

```
library(sandwich)
vcv.wild <- vcovBS(
  model,
  type = "wild-rademacher", cluster = ~g, R = 1000
)
coeftest(model, vcv.wild)
```

```
##
## t test of coefficients:
##
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.22076    0.23435   0.9420  0.34851
## treat        0.73145    0.40175   1.8207  0.07171 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

## Cluster Robust SEs with fixest, estimatr

```
fixest::feols(y ~ treat, data = df, cluster = ~g)
```

```
## OLS estimation, Dep. Var.: y
## Observations: 100
## Standard-errors: Clustered (g)
##           Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 0.220756   0.247647  0.891414  0.39591
## treat       0.731449   0.429948  1.701252  0.12311
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 1.14716   Adj. R2: 0.080828
```

```
estimatr::lm_robust(y ~ treat, data = df, cluster = g)
```

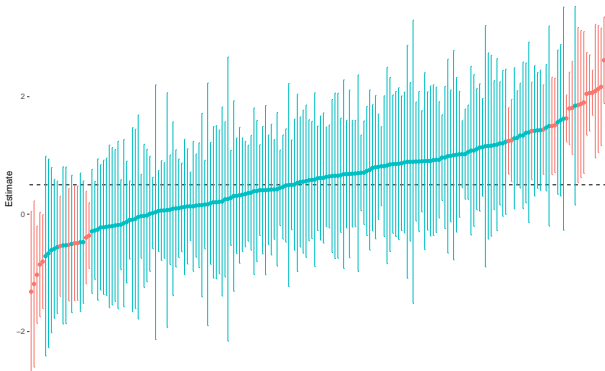
```
##           Estimate Std. Error  t value  Pr(>|t|)    CI Lower CI Up
## (Intercept) 0.2207557  0.2694841  0.819179 0.4759311 -0.6663693 1.107
## treat       0.7314494  0.4645722  1.574458 0.1643331 -0.3935894 1.856
##           DF
## (Intercept) 2.832031
## treat       6.269208
```

# Coverage test: wild

```
conduct_sims(wild_sim, "Wild Bootstrap", 200)
```

```
## Warning: Removed 1 row containing missing values or values outside t.  
## ('geom_point()').
```

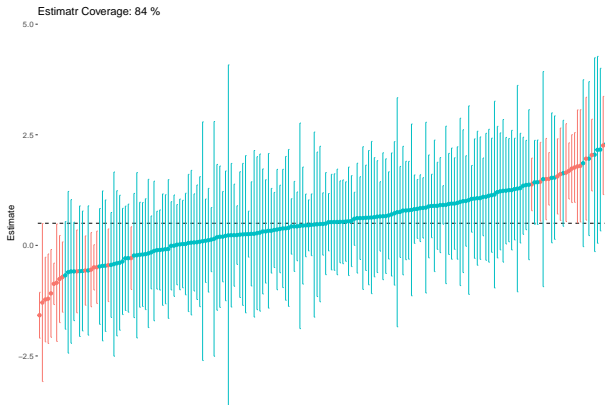
Wild Bootstrap Coverage: 84.9246231155779 %



# Coverage test: estimatr

```
conduct_sims(estimatr_sim, "Estimatr", 200)
```

```
## Warning in sqrt(diag(vcov_fit$Vcov_hat)): NaNs produced
```



# Git intro

- ▶ Source (ie, code and source) management tool

# Git intro

- ▶ Source (ie, code and source) management tool
- ▶ Mac/Linux: Check if installed with `git --version`



# Git intro

- ▶ Source (ie, code and source) management tool
- ▶ Mac/Linux: Check if installed with `git --version`
- ▶ `git clone` and `git pull` easily sync eg, lab materials to your local computer