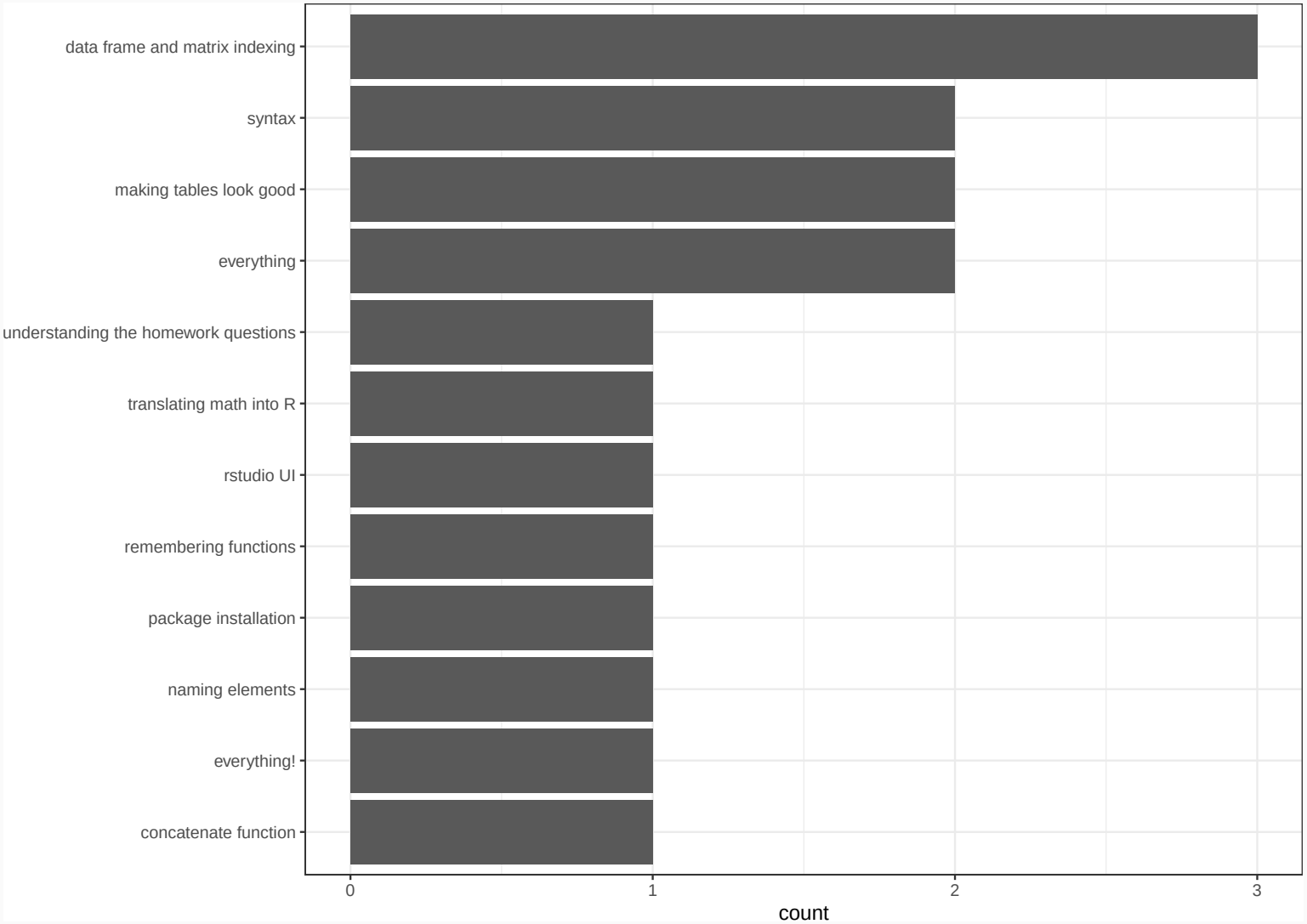
Causality, 2

Frank Edwards

9/16/2020

Challenges in R so far



Returning to Pager’s experiment

The causal effect

For observation *i* is equal to callback_crimTRUE_i - callback_crimFALSE_i

The fundamental problem of causal inference is that we only observe one of these outcomes

The counterfactual and potential outcomes

```
##   callback crimrec callback_crimT callback_crimF
## 1        1       1              1             NA
## 2        0       0             NA              0
## 3        1       0             NA              1
## 4        1       0             NA              1
## 5        0       1              0             NA
## 6        0       1              0             NA
```

Randomized experiments (or RCTs)

- By randomizing assignment to treatment, we can treat units as equivalent

Randomized experiments (or RCTs)

- By randomizing assignment to treatment, we can treat units as equivalent
- If units are equivalent, we can estimate the average treatment effect as a difference in means on the outcome between the treatment and control group

Randomized experiments (or RCTs)

- By randomizing assignment to treatment, we can treat units as equivalent
- If units are equivalent, we can estimate the average treatment effect as a difference in means on the outcome between the treatment and control group
- If we don't randomize, we have no assurance that the treated and control groups are equivalent, meaning we don't have a strong case that we've observed the counterfactual

Obtaining a sample average treatment effect

The sample average treatment effect is defined as:

$$\text{SATE} = \frac{1}{n} \sum_{i=1}^{n} Y_i(1) - Y_i(0)$$

Obtaining a sample average treatment effect

The sample average treatment effect is defined as:

$$\text{SATE} = \frac{1}{n} \sum_{i=1}^{n} Y_i(1) - Y_i(0)$$

In practice, since we only observe $Y_i(1)$ OR $Y_i(0)$, we instead estimate a *difference-in-means* of the outcome between the treatment and control: $mean(Y(1)) - mean(Y(0))$. If assignment has been randomized, these values are identical.

Why we randomize

An experiment on voting and a social pressure

Civic duty: The whole point of democracy is that citizens are active participants in government; that we have a voice in government. Your voice starts with your vote. On August 8, remember your rights and responsibilities as a citizen. Remember to vote. DO YOUR CIVIC DUTY – VOTE

An experiment on voting and a social pressure

Civic duty: The whole point of democracy is that citizens are active participants in government; that we have a voice in government. Your voice starts with your vote. On August 8, remember your rights and responsibilities as a citizen. Remember to vote. DO YOUR CIVIC DUTY – VOTE

Hawthorne effect (surveillance): This year, we're trying to figure out why people do or do not vote. We'll be studying voter turnout in the August 8 primary election. Our analysis will be based on public records, so you will not be contacted again or disturbed in any way. Anything we learn about your voting or not voting will remain confidential and will not be disclosed to anyone else. DO YOUR CIVIC DUTY – VOTE

An experiment on voting and a social pressure

```
data(social)
head(social)
```

```
##      sex yearofbirth primary2004   messages primary2006 hhsize
## 1   male        1941           0 Civic Duty           0      2
## 2 female        1947           0 Civic Duty           0      2
## 3   male        1951           0  Hawthorne           1      3
## 4 female        1950           0  Hawthorne           1      3
## 5 female        1982           0  Hawthorne           1      3
## 6   male        1981           0    Control           0      3
```

Obtaining mean voting by treatment/control

```
control<-mean(
  social[social$messages == "Control", "primary2006"]
  )

treatment<-mean(
  social[social$messages != "Control", "primary2006"]
  )
control
```

```
## [1] 0.2966383
```

```
treatment
```

```
## [1] 0.3382829
```

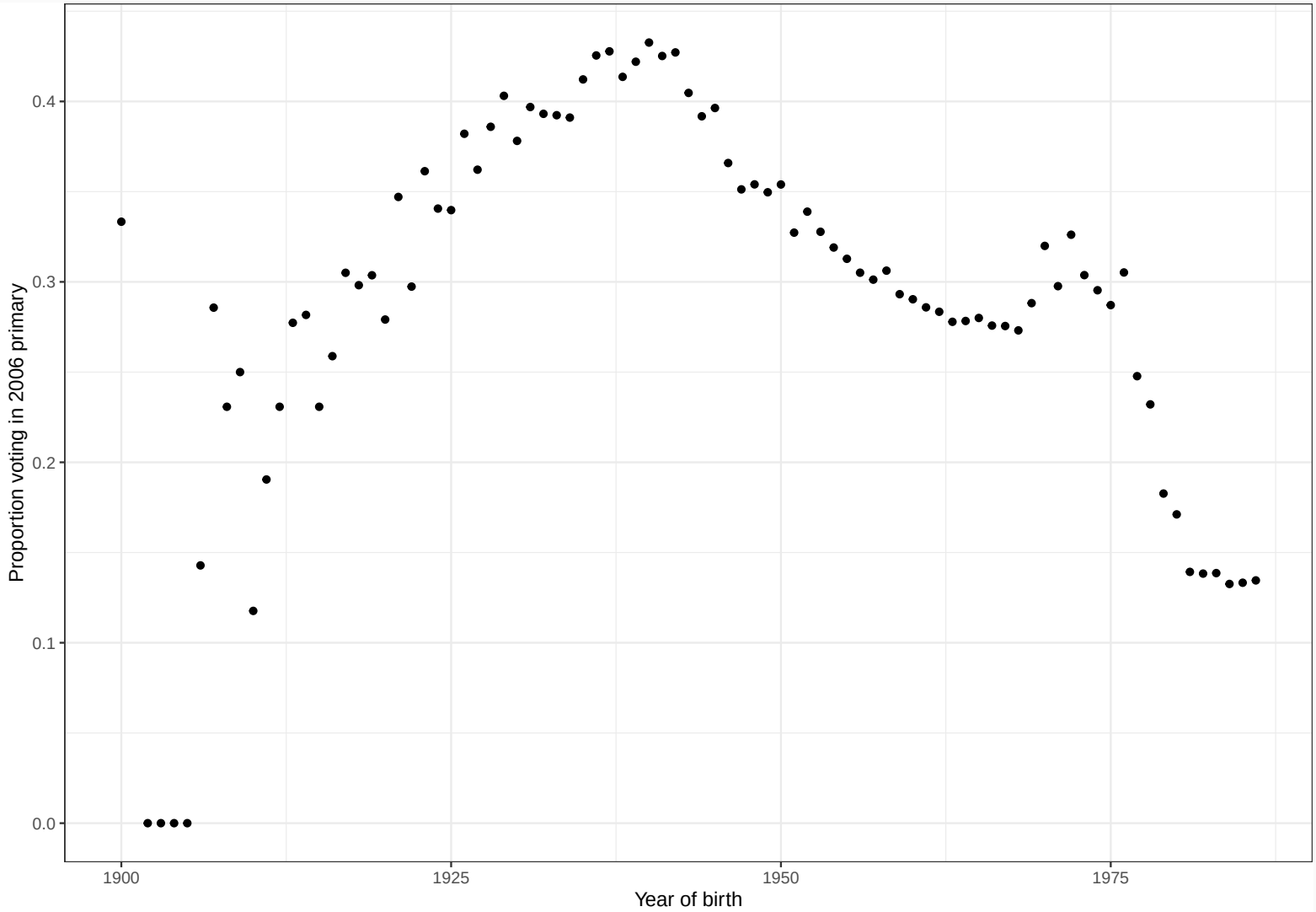
The difference in means (causal effect)

```
effect <- treatment - control
```

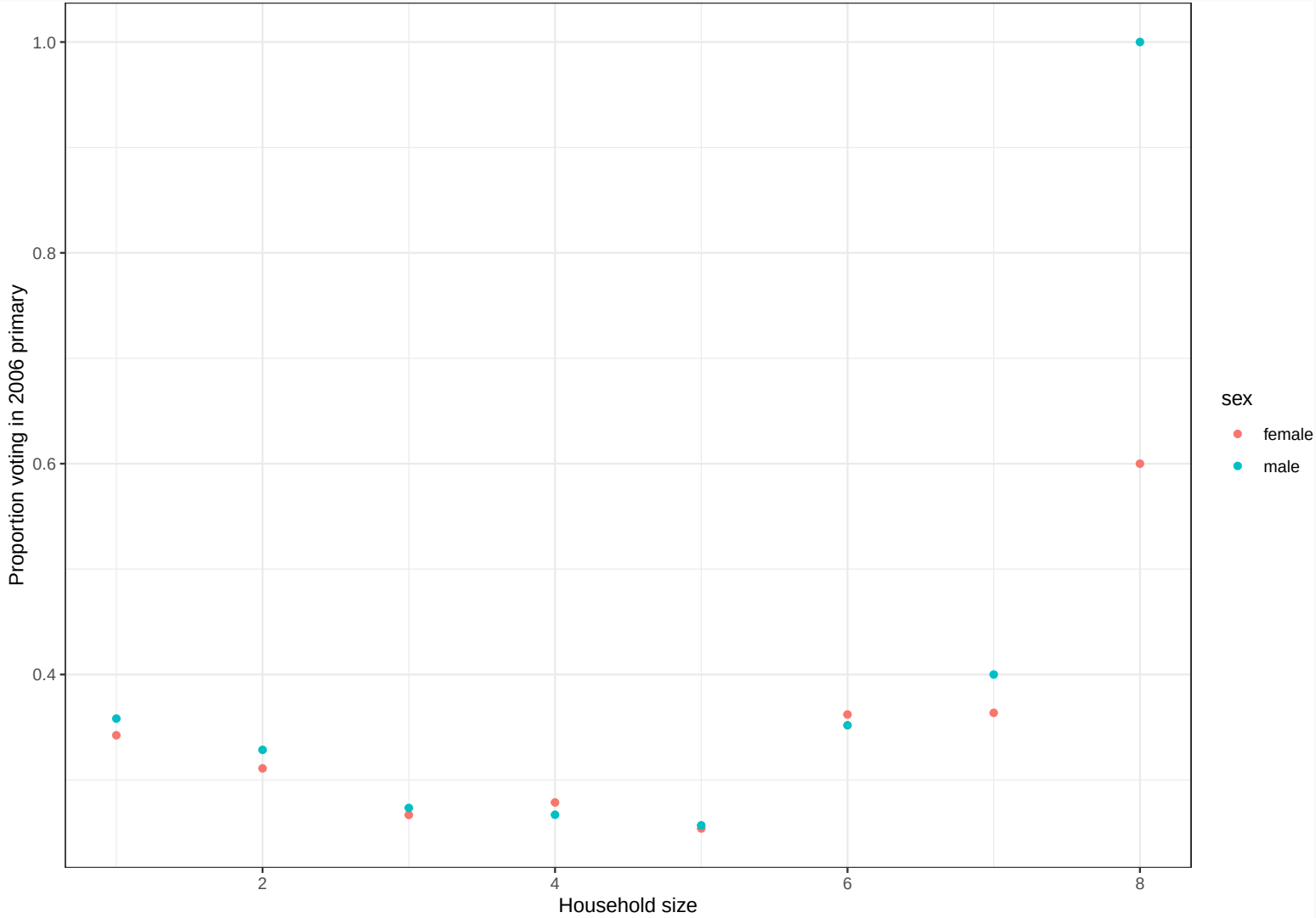
```
effect
```

```
## [1] 0.04164458
```

Why randomization matters



Why randomization matters (continued)



Randomization matters

- Because certain kinds of people are more likely to vote in primaries than others

Randomization matters

- Because certain kinds of people are more likely to vote in primaries than others
- We note these differences between observed variables and our outcome: primary2006

Randomization matters

- Because certain kinds of people are more likely to vote in primaries than others
- We note these differences between observed variables and our outcome: primary2006
- We didn't measure very much here. They could also differ across unobserved or unobservable variables!

Randomization matters

- Because certain kinds of people are more likely to vote in primaries than others
- We note these differences between observed variables and our outcome: primary2006
- We didn't measure very much here. They could also differ across unobserved or unobservable variables!
- Randomization (given a large enough n) ensures that treatment and control groups are *identical* across all observed and unobserved/unobservable differences prior to treatment

Randomization matters

- Because certain kinds of people are more likely to vote in primaries than others
- We note these differences between observed variables and our outcome: primary2006
- We didn't measure very much here. They could also differ across unobserved or unobservable variables!
- Randomization (given a large enough n) ensures that treatment and control groups are *identical* across all observed and unobserved/unobservable differences prior to treatment
- This condition – statistically identical treatment and control groups – is a necessary condition for causal inference. Randomization is the most straightforward way to achieve this condition.

Causal inference in observational data

Estimating the impact of a minimum wage increase

In 1992, New Jersey raised it's minimum wage from $4.25 to $5.05.

Pennsylvania did not.

```
data(minwage)
head(minwage)
```

```
##        chain location wageBefore wageAfter fullBefore fullAfter partBefore
## 1     wendys       PA       5.00      5.25         20         0         20
## 2     wendys       PA       5.50      4.75          6        28         26
## 3 burgerking       PA       5.00      4.75         50        15         35
## 4 burgerking       PA       5.00      5.00         10        26         17
## 5        kfc       PA       5.25      5.00          2         3          8
## 6        kfc       PA       5.00      5.00          2         2         10
##   partAfter
## 1        36
## 2         3
## 3        18
## 4         9
## 5        12
## 6         9
```

Describing the data, categoricals

```
table(minwage$chain)
```

```
##
## burgerking        kfc       roys     wendys
##        149         75         88         46
```

```
table(minwage$location)
```

```
##
## centralNJ   northNJ         PA   shoreNJ   southNJ
##        45        146         67         33         67
```

Did NJ minimum wage increase the wages paid to employees?

```
minwage %>%
  group_by(location) %>%
  summarise(wageBefore_mn = mean(wageBefore),
            wageAfter_mn = mean(wageAfter))
```

```
## # A tibble: 5 x 3
##   location  wageBefore_mn wageAfter_mn
##   <chr>             <dbl>        <dbl>
## 1 centralNJ          4.63         5.09
## 2 northNJ            4.63         5.09
## 3 PA                 4.65         4.61
## 4 shoreNJ            4.64         5.07
## 5 southNJ            4.54         5.06
```

Another way to look at change in wages

```
minwage %>%
  group_by(location) %>%
  summarise(prop_below_before = mean(wageBefore>=5.05),
            prop_below_after = mean(wageAfter>=5.05))
```

```
## # A tibble: 5 x 3
##   location  prop_below_before prop_below_after
##   <chr>                 <dbl>             <dbl>
## 1 centralNJ             0.133             0.978
## 2 northNJ               0.0753            1
## 3 PA                    0.0597            0.0448
## 4 shoreNJ               0.121             1
## 5 southNJ               0.0746            1
```

Look at our outcome variable

```
###Compute proportion full time before
###And after

minwage$prop_ft_pre<- minwage$fullBefore /
  (minwage$fullBefore +minwage$partBefore)

minwage$prop_ft_post <- minwage$fullAfter /
  (minwage$fullAfter + minwage$partAfter)
```

Look at our outcome variable

```
minwage %>%
  group_by(location) %>%
  summarise(prop_ft_pre = mean(prop_ft_pre),
            prop_ft_post = mean(prop_ft_post))
```

```
## # A tibble: 5 x 3
##   location  prop_ft_pre prop_ft_post
##   <chr>           <dbl>        <dbl>
## 1 centralNJ       0.311        0.251
## 2 northNJ         0.321        0.375
## 3 PA              0.310        0.272
## 4 shoreNJ         0.286        0.345
## 5 southNJ         0.239        0.236
```

Assumption: PA is a no-treatment counterfactual

Estimate the causal effect

```
control<-mean(
  minwage[minwage$location=="PA", "prop_ft_post"])

treatment<-mean(
  minwage[minwage$location!="PA", "prop_ft_post"])

treatment - control
```

```
## [1] 0.04811886
```

Is this a valid estimate of the causal effect?

Confounding jeopardizes causal inference

- Confounding bias: a third variable is associated with both the treatment and the outcome

Confounding jeopardizes causal inference

- Confounding bias: a third variable is associated with both the treatment and the outcome
- Selection bias: a unit may choose to participate in a treatment for reasons that are correlated with the outcome

Confounding jeopardizes causal inference

- Confounding bias: a third variable is associated with both the treatment and the outcome
- Selection bias: a unit may choose to participate in a treatment for reasons that are correlated with the outcome

Correlation != Causation

Dealing with confounding

- Randomize treatment!

Dealing with confounding

- Randomize treatment!
- When we can't...

Dealing with confounding

- Randomize treatment!
- When we can't...
- Statistical control: within-subgroup analysis based on confounder values

Are NJ and PA the same (at least when it comes to fast food jobs?)?

```
minwage %>%
  group_by(location) %>%
  summarise(prop_wendys = mean(chain=="wendys"),
            prop_bk = mean(chain=="burgerking"),
            prop_kfc = mean(chain=="kfc"),
            prop_roys = mean(chain=="roys"))
```

```
## # A tibble: 5 x 5
##   location  prop_wendys prop_bk prop_kfc prop_roys
##   <chr>           <dbl>   <dbl>    <dbl>     <dbl>
## 1 centralNJ      0.0889   0.378    0.244     0.289
## 2 northNJ        0.130    0.459    0.158     0.253
## 3 PA             0.164    0.463    0.149     0.224
## 4 shoreNJ        0.152    0.364    0.303     0.182
## 5 southNJ        0.104    0.328    0.313     0.254
```

Maybe restaurant chain matters? Let's control for it!

```
control<-minwage %>%
  filter(location=="PA") %>%
  group_by(chain) %>%
  summarise(prop_ft_post = mean(prop_ft_post))
```

Maybe restaurant chain matters? Let's control for it!

```
treatment<-minwage %>%
  filter(location!="PA") %>%
  group_by(chain) %>%
  summarise(prop_ft_post = mean(prop_ft_post))
```

Maybe restaurant chain matters? Let's control for it!

```
treatment$effect<-treatment$prop_ft_post -
  control$prop_ft_post

treatment
```

```
## # A tibble: 4 x 3
##   chain      prop_ft_post effect
##   <chr>             <dbl>  <dbl>
## 1 burgerking        0.358 0.0364
## 2 kfc               0.328 0.0918
## 3 roys              0.283 0.0697
## 4 wendys            0.260 0.0117
```

Maybe region matters: central and south vs north and shore

```
control<-minwage %>%
  filter(location=="PA") %>%
  summarise(prop_ft_post = mean(prop_ft_post))

treatment<-minwage %>%
  filter(location!="PA") %>%
  group_by(location) %>%
  summarise(prop_ft_post = mean(prop_ft_post))

control
```

```
##   prop_ft_post
## 1    0.2722821
```

```
treatment
```

```
## # A tibble: 4 x 2
##   location  prop_ft_post
##   <chr>            <dbl>
## 1 centralNJ        0.251
## 2 northNJ          0.375
## 3 shoreNJ          0.345
## 4 southNJ          0.236
```

Maybe region matters?

```
treatment$effect<-treatment$prop_ft_post -
  control$prop_ft_post

treatment
```

```
## # A tibble: 4 x 3
##   location  prop_ft_post  effect
##   <chr>            <dbl>   <dbl>
## 1 centralNJ        0.251 -0.0210
## 2 northNJ          0.375  0.103
## 3 shoreNJ          0.345  0.0728
## 4 southNJ          0.236 -0.0366
```

Cross-sections and time series

- Longitudinal data: repeated measurements of the same unit on the same variables over time

Cross-sections and time series

- Longitudinal data: repeated measurements of the same unit on the same variables over time
- Cross-sectional data: one measurement of many units

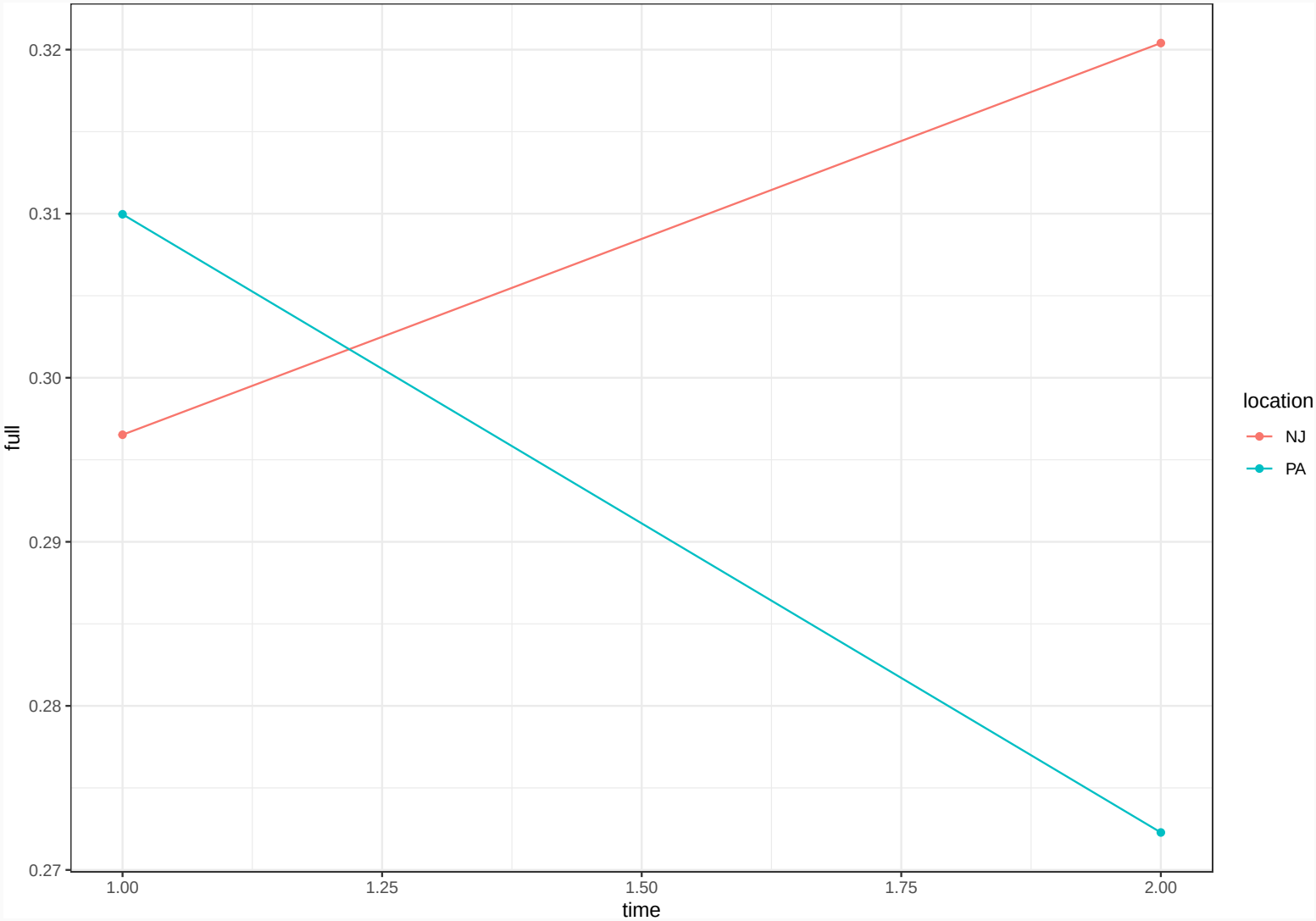
Cross-sections and time series

- Longitudinal data: repeated measurements of the same unit on the same variables over time
- Cross-sectional data: one measurement of many units
- Panel data (or time series cross-sectional data): repeated measurements of many units on the same variables over time

Cross-sections and time series

- Longitudinal data: repeated measurements of the same unit on the same variables over time
- Cross-sectional data: one measurement of many units
- Panel data (or time series cross-sectional data): repeated measurements of many units on the same variables over time
- Key advantages to panel data: variables may differ across units and within-units over time (trends).

Before and after design (longitudinal)



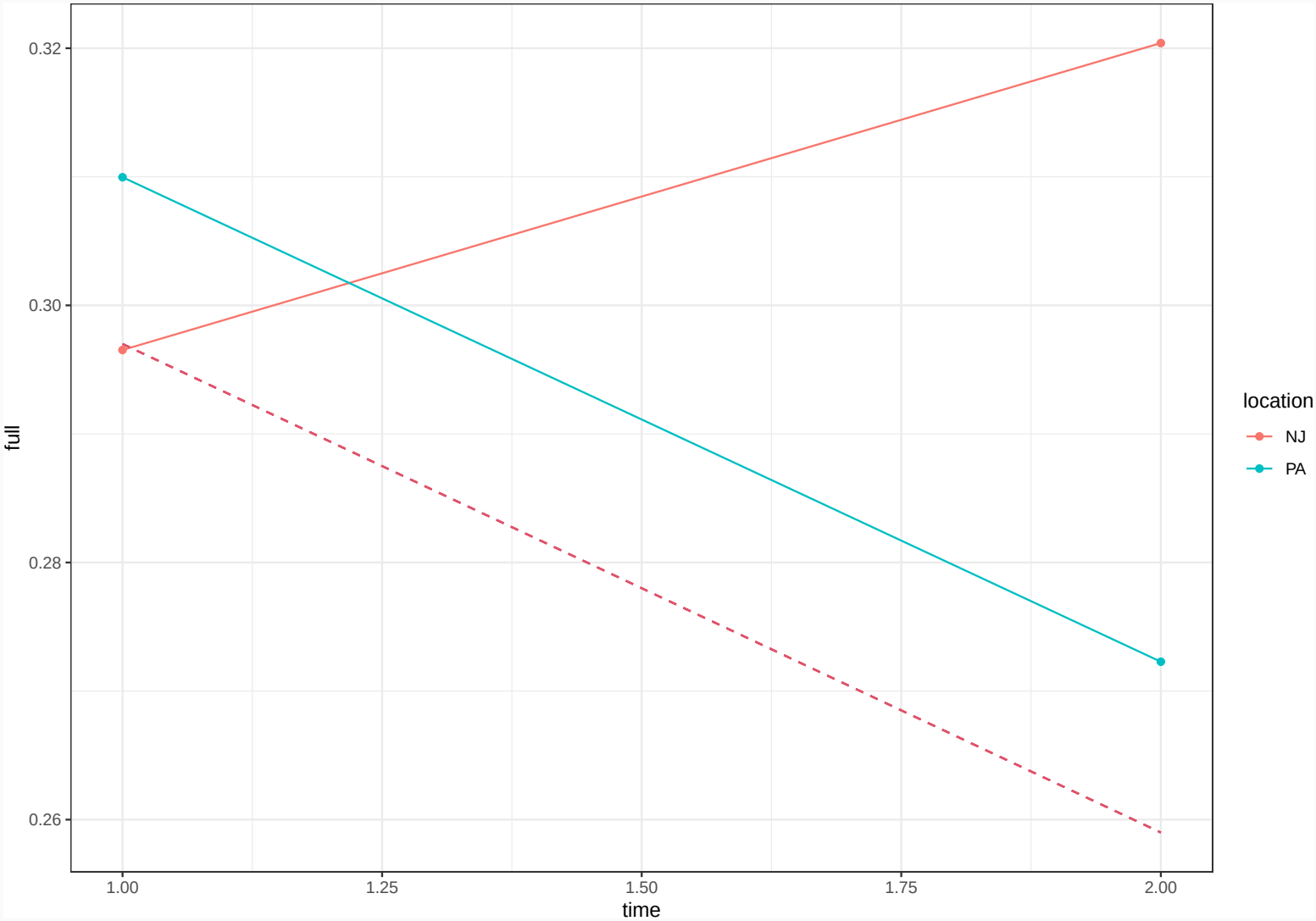
Difference in Differences

- What if we treated PA as the counterfactual, and used information about it's trend in employment to estimate the effect of NJ's minimum wage increase?

Difference in Differences

- What if we treated PA as the counterfactual, and used information about it's trend in employment to estimate the effect of NJ's minimum wage increase?
- Assumption: The trend in the outcome over time would have been identical across all units if the treatment had never been imposed (parallel trends)

Difference in Differences (visual)



Estimating the causal effect: Differenc in Differences

Where y_{ij} is the outcome for treatment group $i = 1$ and post-treatment time $j = 1$

$$\text{DiD} = (\bar{y}_{1,1} - \bar{y}_{1,0}) - (\bar{y}_{2,1} - \bar{y}_{2,0})$$

Assuming that the counterfactual outcome for the treatment group has a parallel time trend to that observed for the control group.

Compute the DiD estimator

```
## # A tibble: 2 x 3
##   location prop_ft_pre prop_ft_post
##   <chr>          <dbl>        <dbl>
## 1 NJ             0.297        0.320
## 2 PA             0.310        0.272
```

Compute the DiD estimator

```
## # A tibble: 2 x 3
##   location prop_ft_pre prop_ft_post
##   <chr>          <dbl>        <dbl>
## 1 NJ             0.297        0.320
## 2 PA             0.310        0.272
```

$$\text{DiD} = (\bar{y}_{1,1} - \bar{y}_{1,0}) - (\bar{y}_{2,1} - \bar{y}_{2,0})$$

Compute the DiD estimator

```
## # A tibble: 2 x 3
##   location prop_ft_pre prop_ft_post
##   <chr>          <dbl>        <dbl>
## 1 NJ             0.297        0.320
## 2 PA             0.310        0.272
```

$$\text{DiD} = (\bar{y}_{1,1} - \bar{y}_{1,0}) - (\bar{y}_{2,1} - \bar{y}_{2,0})$$

```
### the DiD Estimator
(0.320 - 0.297) - (0.272 - 0.310)
```

```
## [1] 0.061
```

Descriptive Statistics

Summarizing a variable

Reduce a vector to a single or smaller set of values that tell us something useful

Examples we've already used: - minimum: min() - maximum: max() - median: median() - mean: mean()

Quantiles

- The median is the 0.5 quantile (50th percentile)
- Quantiles are less sensitive to outliers than are other measures (like the mean)
- Quantiles tell you the proportion of a data that falls below some cutpoint

Quantiles: example

```r
quantile(minwage$wageBefore, 0.25)
```

```
##  25%
## 4.25
```

Quantiles: example

```r
quantile(minwage$wageBefore, 0.75)
```

```
##    75%
## 4.9875
```

Quantiles: example

```r
quantile(minwage$wageBefore, c(0.05, 0.25, 0.5, .75, 0.95))
```

```
##      5%     25%     50%     75%     95%
## 4.2500 4.2500 4.5000 4.9875 5.2500
```

Standard deviation

- The standard deviation (SD, σ) is a measure of the spread of a variable

Standard deviation

- The standard deviation (SD, σ) is a measure of the spread of a variable
- It provides a measure of how much each observation of a variable differs from the mean of the variable

Standard deviation

- The standard deviation (SD, σ) is a measure of the spread of a variable
- It provides a measure of how much each observation of a variable differs from the mean of the variable
- You can use the sd() function in R

Standard deviation

- The standard deviation (SD, σ) is a measure of the spread of a variable
- It provides a measure of how much each observation of a variable differs from the mean of the variable
- You can use the sd() function in R
- The variance (var() function) is the square of the standard deviation

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Standard deviation

- The standard deviation (SD, σ) is a measure of the spread of a variable
- It provides a measure of how much each observation of a variable differs from the mean of the variable
- You can use the sd() function in R
- The variance (var() function) is the square of the standard deviation

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$variance = \sigma^2$$

Compute an SD for these variables

```
minwage$wageBefore[1:10]
```

```
##  [1] 5.00 5.50 5.00 5.00 5.25 5.00 5.00 5.00 5.00 5.50
```

```
minwage$fullBefore[1:10]
```

```
##  [1] 20.0  6.0 50.0 10.0  2.0  2.0  2.5 40.0  8.0 10.5
```

Homework

- HW3 posted to Slack
- For part 1, load the data from `library(nycflights13)`
- For part 2, load the data with `data(STAR)` from `library(qss)`
- make sure to use `na.rm = TRUE` for mean(), quantile() and other functions
- `group_by()` and `summarize()` are very helpful on this one

Lab: Introducing the tidyverse: data transformation with dplyr

What is tidyverse?

A collection of packages that share an underlying philosophy of ‘tidy’ data.

Each variable is a column, each row is an observation.

```
library(tidyverse)
```

A valuable resource: QSS translation

Jeff Arnold has translated all relevant code from QSS into tidyverse syntax:

`https://jrnold.github.io/qss-tidy/introduction.html`

A valuable resource: Wickham's R for Data Science

`https://r4ds.had.co.nz/`

Today, covering chapter 5

Getting ready

```
library(gapminder)

head(gapminder)
```

```
## # A tibble: 6 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Afghanistan Asia       1957    30.3  9240934      821.
## 3 Afghanistan Asia       1962    32.0 10267083      853.
## 4 Afghanistan Asia       1967    34.0 11537966      836.
## 5 Afghanistan Asia       1972    36.1 13079460      740.
## 6 Afghanistan Asia       1977    38.4 14880372      786.
```

Dplyr basics (verbs)

- `filter()` chooses observations by their value

Dplyr basics (verbs)

- `filter()` chooses observations by their value
- `arrange()` reorders rows by value

Dplyr basics (verbs)

- `filter()` chooses observations by their value
- `arrange()` reorders rows by value
- `select()` chooses variables by name

Dplyr basics (verbs)

- `filter()` chooses observations by their value
- `arrange()` reorders rows by value
- `select()` chooses variables by name
- `mutate()` creates new variables

Dplyr basics (verbs)

- `filter()` chooses observations by their value
- `arrange()` reorders rows by value
- `select()` chooses variables by name
- `mutate()` creates new variables
- `summarize()` collapses variables to a single summary

Each of these can be paired with `group_by()` to operate on a subsset of the data with a grouping variable.

Dplyr input and output

We feed dplyr data frames, it returns data frames.

filter() to subset

The first argument is a data frame, the subsequent argument(s) are logical conditions to filter the data on

```
filter(gapminder, year<1957)
```

```
## # A tibble: 142 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Albania     Europe     1952    55.2  1282697     1601.
##  3 Algeria     Africa     1952    43.1  9279525     2449.
##  4 Angola      Africa     1952    30.0  4232095     3521.
##  5 Argentina   Americas   1952    62.5 17876956     5911.
##  6 Australia   Oceania    1952    69.1  8691212    10040.
##  7 Austria     Europe     1952    66.8  6927772     6137.
##  8 Bahrain     Asia       1952    50.9   120447     9867.
##  9 Bangladesh  Asia       1952    37.5 46886859      684.
## 10 Belgium     Europe     1952    68    8730405     8343.
## # ... with 132 more rows
```

filter() to subset

```
filter(gapminder, lifeExp > 80)
```

```
## # A tibble: 21 x 6
##    country          continent  year lifeExp      pop gdpPercap
##    <fct>            <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Australia        Oceania    2002    80.4 19546792    30688.
##  2 Australia        Oceania    2007    81.2 20434176    34435.
##  3 Canada           Americas   2007    80.7 33390141    36319.
##  4 France           Europe     2007    80.7 61083916    30470.
##  5 Hong Kong, China Asia       2002    81.5  6762476    30209.
##  6 Hong Kong, China Asia       2007    82.2  6980412    39725.
##  7 Iceland          Europe     2002    80.5   288030    31163.
##  8 Iceland          Europe     2007    81.8   301931    36181.
##  9 Israel           Asia       2007    80.7  6426679    25523.
## 10 Italy            Europe     2002    80.2 57926999    27968.
## # ... with 11 more rows
```

filter() to subset

```
filter(gapminder, country == "Algeria",
       year > 1990)
```

```
## # A tibble: 4 x 6
##   country continent  year lifeExp      pop gdpPercap
##   <fct>   <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Algeria Africa     1992    67.7 26298373     5023.
## 2 Algeria Africa     1997    69.2 29072015     4797.
## 3 Algeria Africa     2002    71.0 31287142     5288.
## 4 Algeria Africa     2007    72.3 33333216     6223.
```

filter() to subset

```
filter(gapminder, (country == "Algeria" | country == "Tunisia")
       & year > 1994)
```

```
## # A tibble: 6 x 6
##   country continent  year lifeExp      pop gdpPercap
##   <fct>   <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Algeria Africa     1997    69.2 29072015     4797.
## 2 Algeria Africa     2002    71.0 31287142     5288.
## 3 Algeria Africa     2007    72.3 33333216     6223.
## 4 Tunisia Africa     1997    72.0  9231669     4877.
## 5 Tunisia Africa     2002    73.0  9770575     5723.
## 6 Tunisia Africa     2007    73.9 10276158     7093.
```

Practice

- Find all country-years in South America with life expectancy greater than 75 years
- Find the country-year with the smallest life expectancy
- Find the country-year with the highest life expectancy

arrange() rows

Reorders rows by the values in one (or more) columns

```
arrange(gapminder, pop)
```

```
## # A tibble: 1,704 x 6
##    country               continent  year lifeExp   pop gdpPercap
##    <fct>                 <fct>     <int>   <dbl> <int>     <dbl>
##  1 Sao Tome and Principe Africa     1952    46.5 60011      880.
##  2 Sao Tome and Principe Africa     1957    48.9 61325      861.
##  3 Djibouti              Africa     1952    34.8 63149     2670.
##  4 Sao Tome and Principe Africa     1962    51.9 65345     1072.
##  5 Sao Tome and Principe Africa     1967    54.4 70787     1385.
##  6 Djibouti              Africa     1957    37.3 71851     2865.
##  7 Sao Tome and Principe Africa     1972    56.5 76595     1533.
##  8 Sao Tome and Principe Africa     1977    58.6 86796     1738.
##  9 Djibouti              Africa     1962    39.7 89898     3021.
## 10 Sao Tome and Principe Africa     1982    60.4 98593     1890.
## # ... with 1,694 more rows
```

arrange() rows

`desc()` puts them in descending order

```
arrange(gapminder, desc(pop))
```

```
## # A tibble: 1,704 x 6
##    country continent  year lifeExp        pop gdpPercap
##    <fct>   <fct>     <int>   <dbl>      <int>     <dbl>
##  1 China   Asia       2007    73.0 1318683096     4959.
##  2 China   Asia       2002    72.0 1280400000     3119.
##  3 China   Asia       1997    70.4 1230075000     2289.
##  4 China   Asia       1992    68.7 1164970000     1656.
##  5 India   Asia       2007    64.7 1110396331     2452.
##  6 China   Asia       1987    67.3 1084035000     1379.
##  7 India   Asia       2002    62.9 1034172547     1747.
##  8 China   Asia       1982    65.5 1000281000      962.
##  9 India   Asia       1997    61.8  959000000     1459.
## 10 China   Asia       1977    64.0  943455000      741.
## # ... with 1,694 more rows
```

arrange() rows

We can arrange by more than one variable

```
arrange(gapminder, year, pop)
```

```
## # A tibble: 1,704 x 6
##    country               continent  year lifeExp    pop gdpPercap
##    <fct>                 <fct>     <int>   <dbl>  <int>     <dbl>
##  1 Sao Tome and Principe Africa     1952    46.5  60011      880.
##  2 Djibouti              Africa     1952    34.8  63149     2670.
##  3 Bahrain               Asia       1952    50.9 120447     9867.
##  4 Iceland               Europe     1952    72.5 147962     7268.
##  5 Comoros               Africa     1952    40.7 153936     1103.
##  6 Kuwait                Asia       1952    55.6 160000   108382.
##  7 Equatorial Guinea     Africa     1952    34.5 216964      376.
##  8 Reunion               Africa     1952    52.7 257700     2719.
##  9 Gambia                Africa     1952    30   284320      485.
## 10 Swaziland             Africa     1952    41.4 290243     1148.
## # ... with 1,694 more rows
```

Practice

- Arrange countries by life expectancy from lowest to highest
- Arrange countries by life expectancy from highest to lowest

select() columns

`select()` chooses columns to return by name.

```
select(gapminder, country, year)
```

```
## # A tibble: 1,704 x 2
##    country      year
##    <fct>       <int>
##  1 Afghanistan  1952
##  2 Afghanistan  1957
##  3 Afghanistan  1962
##  4 Afghanistan  1967
##  5 Afghanistan  1972
##  6 Afghanistan  1977
##  7 Afghanistan  1982
##  8 Afghanistan  1987
##  9 Afghanistan  1992
## 10 Afghanistan  1997
## # ... with 1,694 more rows
```

select() columns

You can use - to remove a column from the selection

```
select(gapminder, -gdpPercap, -pop)
```

```
## # A tibble: 1,704 x 4
##    country     continent  year lifeExp
##    <fct>       <fct>     <int>   <dbl>
##  1 Afghanistan Asia       1952    28.8
##  2 Afghanistan Asia       1957    30.3
##  3 Afghanistan Asia       1962    32.0
##  4 Afghanistan Asia       1967    34.0
##  5 Afghanistan Asia       1972    36.1
##  6 Afghanistan Asia       1977    38.4
##  7 Afghanistan Asia       1982    39.9
##  8 Afghanistan Asia       1987    40.8
##  9 Afghanistan Asia       1992    41.7
## 10 Afghanistan Asia       1997    41.8
## # ... with 1,694 more rows
```

Practice

- Return a data frame containing continent, country, and year
- Return a data frame containing everything except population

Create new variables with mutate()

```
mutate(gapminder,
       pop_100k = pop / 100000)
```

```
## # A tibble: 1,704 x 7
##    country     continent  year lifeExp      pop gdpPercap pop_100k
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>    <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.     84.3
##  2 Afghanistan Asia       1957    30.3  9240934      821.     92.4
##  3 Afghanistan Asia       1962    32.0 10267083      853.    103.
##  4 Afghanistan Asia       1967    34.0 11537966      836.    115.
##  5 Afghanistan Asia       1972    36.1 13079460      740.    131.
##  6 Afghanistan Asia       1977    38.4 14880372      786.    149.
##  7 Afghanistan Asia       1982    39.9 12881816      978.    129.
##  8 Afghanistan Asia       1987    40.8 13867957      852.    139.
##  9 Afghanistan Asia       1992    41.7 16317921      649.    163.
## 10 Afghanistan Asia       1997    41.8 22227415      635.    222.
## # ... with 1,694 more rows
```

Create new variables with mutate()

```
mutate(gapminder,
       pop_100k = pop / 100000,
       gdp = gdpPercap * pop)
```

```
## # A tibble: 1,704 x 8
##    country     continent  year lifeExp      pop gdpPercap pop_100k          gdp
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>    <dbl>        <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.     84.3  6567086330.
##  2 Afghanistan Asia       1957    30.3  9240934      821.     92.4  7585448670.
##  3 Afghanistan Asia       1962    32.0 10267083      853.    103.   8758855797.
##  4 Afghanistan Asia       1967    34.0 11537966      836.    115.   9648014150.
##  5 Afghanistan Asia       1972    36.1 13079460      740.    131.   9678553274.
##  6 Afghanistan Asia       1977    38.4 14880372      786.    149.  11697659231.
##  7 Afghanistan Asia       1982    39.9 12881816      978.    129.  12598563401.
##  8 Afghanistan Asia       1987    40.8 13867957      852.    139.  11820990309.
##  9 Afghanistan Asia       1992    41.7 16317921      649.    163.  10595901589.
## 10 Afghanistan Asia       1997    41.8 22227415      635.    222.  14121995875.
## # ... with 1,694 more rows
```

Create new variables with mutate()

```
mutate(gapminder,
       year_c = year-1952,
       period = year_c / 5)
```

```
## # A tibble: 1,704 x 8
##    country     continent  year lifeExp      pop gdpPercap year_c period
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>  <dbl>  <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.      0      0
##  2 Afghanistan Asia       1957    30.3  9240934      821.      5      1
##  3 Afghanistan Asia       1962    32.0 10267083      853.     10      2
##  4 Afghanistan Asia       1967    34.0 11537966      836.     15      3
##  5 Afghanistan Asia       1972    36.1 13079460      740.     20      4
##  6 Afghanistan Asia       1977    38.4 14880372      786.     25      5
##  7 Afghanistan Asia       1982    39.9 12881816      978.     30      6
##  8 Afghanistan Asia       1987    40.8 13867957      852.     35      7
##  9 Afghanistan Asia       1992    41.7 16317921      649.     40      8
## 10 Afghanistan Asia       1997    41.8 22227415      635.     45      9
## # ... with 1,694 more rows
```

Practice

- Create a new variable equal to the log() of lifeExp

Summarize the data

`summarize()` collapses a data frame to one row

```
summarize(gapminder,
          lifeExp_mn = mean(lifeExp),
          lifeExp_md = median(lifeExp))
```

```
## # A tibble: 1 x 2
##   lifeExp_mn lifeExp_md
##        <dbl>      <dbl>
## 1       59.5       60.7
```

Summarizing by group

`group_by()` performs subsequent operations over specified groups (usually a factor). We use the 'pipe' operator `%>%` to string together commands

```
gapminder %>%
  group_by(continent) %>%
  summarise(lifeExp_mn = mean(lifeExp),
            lifeExp_sd = sd(lifeExp))
```

```
## # A tibble: 5 x 3
##   continent lifeExp_mn lifeExp_sd
##   <fct>          <dbl>      <dbl>
## 1 Africa          48.9       9.15
## 2 Americas        64.7       9.35
## 3 Asia            60.1      11.9
## 4 Europe          71.9       5.43
## 5 Oceania         74.3       3.80
```

Summarizing by group

```
gapminder %>%
  group_by(continent, year) %>%
  summarise(lifeExp_mn = mean(lifeExp))
```

```
## # A tibble: 60 x 3
## # Groups:   continent [5]
##    continent  year lifeExp_mn
##    <fct>     <int>      <dbl>
##  1 Africa     1952       39.1
##  2 Africa     1957       41.3
##  3 Africa     1962       43.3
##  4 Africa     1967       45.3
##  5 Africa     1972       47.5
##  6 Africa     1977       49.6
##  7 Africa     1982       51.6
##  8 Africa     1987       53.3
##  9 Africa     1992       53.6
## 10 Africa     1997       53.6
## # ... with 50 more rows
```

Advanced dplyr: piping

We can use the pipe to string together any number of dplyr commands.

We start with the data frame we are working from

```
gapminder %>%
  filter(continent=="Americas") %>%
  select(country, year, gdpPercap) %>%
  arrange(year)
```

```
## # A tibble: 300 x 3
##    country             year gdpPercap
##    <fct>              <int>     <dbl>
##  1 Argentina           1952     5911.
##  2 Bolivia             1952     2677.
##  3 Brazil              1952     2109.
##  4 Canada              1952    11367.
##  5 Chile               1952     3940.
##  6 Colombia            1952     2144.
##  7 Costa Rica          1952     2627.
##  8 Cuba                1952     5587.
##  9 Dominican Republic  1952     1398.
## 10 Ecuador             1952     3522.
## # ... with 290 more rows
```