Data Cleaning

(Please grab your next assignment sheet!)

Understanding Political Numbers

March 13, 2019

Review

Multiple regression

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \epsilon$$

Evaluating coefficient estimates

What's the sign? (+ / -)

What's the magnitude? (how big)

Is it significant?

Example: voter registration in Wisconsin

Uniform statewide voter registration requirements → lower turnout?

Controlling for...

Municipal expenditures on election administration

Population size

Table 2. The Effect of Registration on Voter Turnout, 2000–2008

Explanatory variable	Baseline model	Simple model
New registration requirement	–0.017** (0.002)	–0.018** (0.002)
Administrative expenditures (logged)	—	0.007** (0.002)
Population (logged)	—	–0.095** (0.033)
Constant	0.663** (0.001)	1.298** (0.228)
Adjusted R^2	.88	.88
N	9,245	9,240
Municipality fixed effects?	Yes	Yes
Year fixed effects?	Yes	Yes

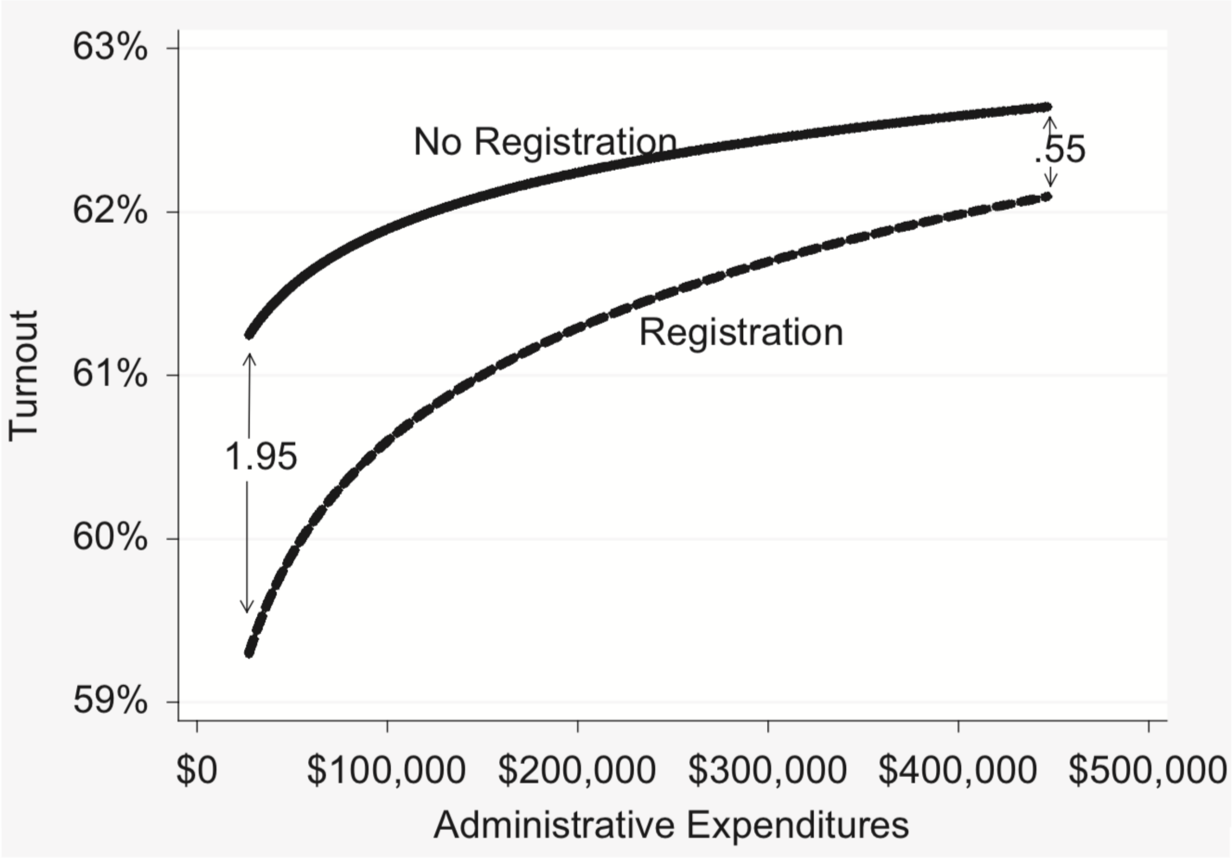
Clustered standard errors in parentheses in the first four models.
$^*p < .05$. $^{**}p < .01$.

$$\hat{y} = a + (b_1 \times \text{Requirement}) + (b_2 \times \text{logExpenditures}) + (b_3 \times \text{logPopulation})$$

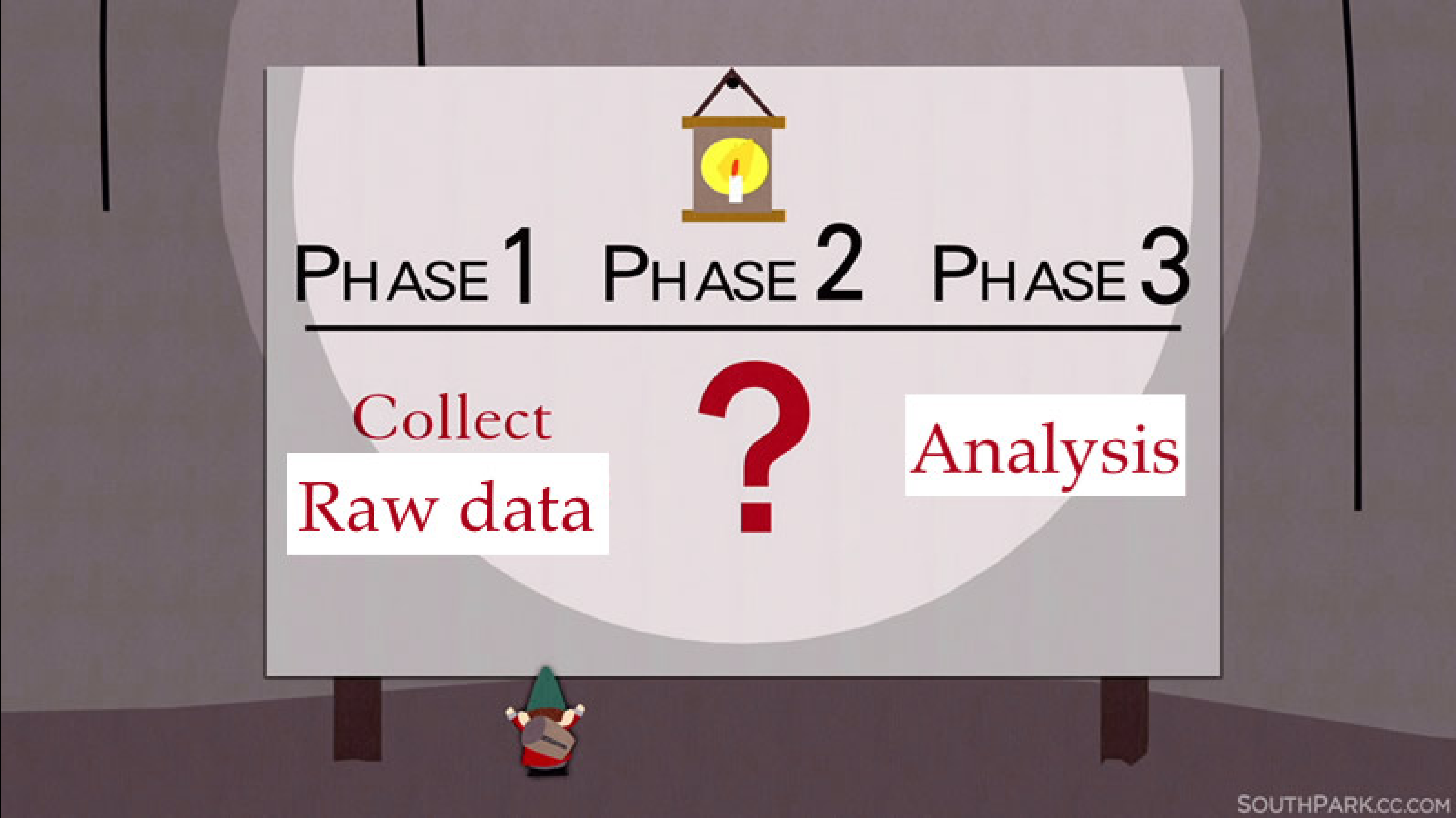
$$\hat{y} = 1.298 + (-0.018 \times \text{Requirement}) + (0.007 \times \text{logExpenditures}) + (-0.095 \times \text{logPopulation})$$

Predicted values

Predicted values



Data organization and cleaning

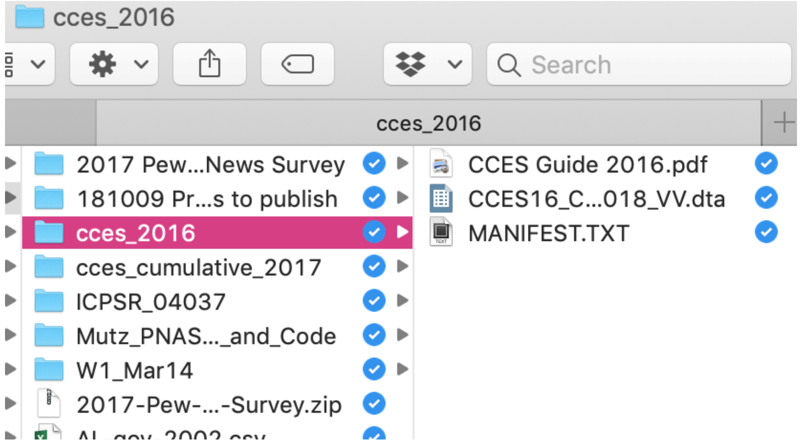

Phase 1
Phase 2
Phase 3
Collect
Raw data
?
Analysis
SOUTHPARK.CC.COM

1. Spreadsheets
2. Shaping / Merging
3. Recoding

Data in spreadsheets

Downloading pre-packaged data

Creating your own spreadsheet




```
# Package for reading .dta, .sav, .por files
# Don't listen to google when it tells you to use {foreign} pkg
library("haven")

my_data <- read_dta(here("data", "some-data.dta"))  # for .dta files
my_data <- read_spss(here("data", "some-data.sav")) # for .sav (or .por)

# for excel files
library("readxl")
my_data <- read_excel(here("data", "some-data.xlsx"))

# "Catch-all" data reading package
library("rio")
my_data <- import(here("data", "some-data.dta"))
```

Shaping data

Wide data: the same variable is split across multiple columns.

Shaping data

Wide data: the same variable is split across multiple columns.

```
AirPassengers
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1949 112 118 132 129 121 135 148 148 136 119 104 118
## 1950 115 126 141 135 125 149 170 170 158 133 114 140
## 1951 145 150 178 163 172 178 199 199 184 162 146 166
## 1952 171 180 193 181 183 218 230 242 209 191 172 194
## 1953 196 196 236 235 229 243 264 272 237 211 180 201
## 1954 204 188 235 227 234 264 302 293 259 229 203 229
## 1955 242 233 267 269 270 315 364 347 312 274 237 278
## 1956 284 277 317 313 318 374 413 405 355 306 271 306
## 1957 315 301 356 348 355 422 465 467 404 347 305 336
## 1958 340 318 362 348 363 435 491 505 404 359 310 337
## 1959 360 342 406 396 420 472 548 559 463 407 362 405
## 1960 417 391 419 461 472 535 622 606 508 461 390 432
```

Shaping data

Wide data: the same variable is split across multiple columns.

```
AirPassengers
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1949 112 118 132 129 121 135 148 148 136 119 104 118
## 1950 115 126 141 135 125 149 170 170 158 133 114 140
## 1951 145 150 178 163 172 178 199 199 184 162 146 166
## 1952 171 180 193 181 183 218 230 242 209 191 172 194
## 1953 196 196 236 235 229 243 264 272 237 211 180 201
## 1954 204 188 235 227 234 264 302 293 259 229 203 229
## 1955 242 233 267 269 270 315 364 347 312 274 237 278
## 1956 284 277 317 313 318 374 413 405 355 306 271 306
## 1957 315 301 356 348 355 422 465 467 404 347 305 336
## 1958 340 318 362 348 363 435 491 505 404 359 310 337
## 1959 360 342 406 396 420 472 548 559 463 407 362 405
## 1960 417 391 419 461 472 535 622 606 508 461 390 432
```

We don't want this

Shaping data

Long data: Each variable gets its own column

Shaping data

Long data: Each variable gets its own column

```
## # A tibble: 144 x 3
##     year month passengers
##    <int> <chr>      <dbl>
##  1  1949 Jan          112
##  2  1949 Feb          118
##  3  1949 Mar          132
##  4  1949 Apr          129
##  5  1949 May          121
##  6  1949 Jun          135
##  7  1949 Jul          148
##  8  1949 Aug          148
##  9  1949 Sep          136
## 10  1949 Oct          119
## # … with 134 more rows
```

Shaping data

Long data: Each variable gets its own column

```
## # A tibble: 144 x 3
##     year month passengers
##    <int> <chr>      <dbl>
##  1  1949 Jan          112
##  2  1949 Feb          118
##  3  1949 Mar          132
##  4  1949 Apr          129
##  5  1949 May          121
##  6  1949 Jun          135
##  7  1949 Jul          148
##  8  1949 Aug          148
##  9  1949 Sep          136
## 10  1949 Oct          119
## # … with 134 more rows
```

Much better

Shaping data

Go from wide to long with `gather()`

```
## [1] "WIDE"
```

```
## # A tibble: 12 x 13
##     year   Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov
##    <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1  1949   112   118   132   129   121   135   148   148   136   119   104
##  2  1950   115   126   141   135   125   149   170   170   158   133   114
##  3  1951   145   150   178   163   172   178   199   199   184   162   146
##  4  1952   171   180   193   181   183   218   230   242   209   191   172
##  5  1953   196   196   236   235   229   243   264   272   237   211   180
##  6  1954   204   188   235   227   234   264   302   293   259   229   203
##  7  1955   242   233   267   269   270   315   364   347   312   274   237
##  8  1956   284   277   317   313   318   374   413   405   355   306   271
##  9  1957   315   301   356   348   355   422   465   467   404   347   305
## 10  1958   340   318   362   348   363   435   491   505   404   359   310
## 11  1959   360   342   406   396   420   472   548   559   463   407   362
## 12  1960   417   391   419   461   472   535   622   606   508   461   390
## # … with 1 more variable: Dec <dbl>
```

Shaping data

Go from wide to long with `gather()`

```r
# how to use gather()
#   key = variable name for the labels
#   value = variable name for the data
#   variables you want to "stack" (comma sep'd)
#   can grab a range of variables using : colon
wide_data %>%
  gather(key = month, value = passengers,
         Jan, Feb, Mar, Apr:Dec)
```

```
## # A tibble: 144 x 3
##     year month passengers
##    <int> <chr>      <dbl>
##  1  1949 Jan          112
##  2  1950 Jan          115
##  3  1951 Jan          145
##  4  1952 Jan          171
##  5  1953 Jan          196
##  6  1954 Jan          204
##  7  1955 Jan          242
##  8  1956 Jan          284
##  9  1957 Jan          315
## 10  1958 Jan          340
## # … with 134 more rows
```

Merging

I have two data tables

```
## # A tibble: 50 x 2
##    state_name  Murder
##    <chr>        <dbl>
##  1 Alabama       13.2
##  2 Alaska        10
##  3 Arizona        8.1
##  4 Arkansas       8.8
##  5 California     9
##  6 Colorado       7.9
##  7 Connecticut    3.3
##  8 Delaware       5.9
##  9 Florida       15.4
## 10 Georgia       17.4
## # … with 40 more rows
```

```
## # A tibble: 5 x 3
##   state state_name mean_poverty
##   <chr> <chr>             <dbl>
## 1 IL    Illinois           11.9
## 2 IN    Indiana            10.7
## 3 MI    Michigan           13.1
## 4 OH    Ohio               12.5
## 5 WI    Wisconsin          10.7
```

Merging

Inner join: return only the rows that match

```
inner_join(arrests, midwest_poverty, by = "state_name")
```

```
## # A tibble: 5 x 4
##   state_name Murder state mean_poverty
##   <chr>       <dbl> <chr>        <dbl>
## 1 Illinois     10.4 IL            11.9
## 2 Indiana       7.2 IN            10.7
## 3 Michigan     12.1 MI            13.1
## 4 Ohio          7.3 OH            12.5
## 5 Wisconsin     2.6 WI            10.7
```

Merging

Left join: return the "left" dataset and anything that matches from the right

```
# arranging to show the effect
left_join(arrests, midwest_poverty, by = "state_name") %>%
  arrange(mean_poverty)
```

```
## # A tibble: 50 x 4
##    state_name Murder state mean_poverty
##    <chr>       <dbl> <chr>        <dbl>
##  1 Wisconsin     2.6 WI            10.7
##  2 Indiana       7.2 IN            10.7
##  3 Illinois     10.4 IL            11.9
##  4 Ohio          7.3 OH            12.5
##  5 Michigan     12.1 MI            13.1
##  6 Alabama      13.2 <NA>          NA
##  7 Alaska       10   <NA>          NA
##  8 Arizona       8.1 <NA>          NA
##  9 Arkansas      8.8 <NA>          NA
## 10 California    9   <NA>          NA
## # … with 40 more rows
```

Recoding / Cleaning

Selectively altering variables

```
# case_when(logical_test ~ result)
# works like: if [logical_test] then [result]
# unmatched cases default to NA but you can catch all with `TRUE ~ result`

midwest_poverty %>%
  mutate(is_great = case_when(state == "WI" ~ "Pretty great",
                              state == "IL" ~ "Medium",
                              TRUE ~ "Crappy"))
```

```
## # A tibble: 5 x 4
##   state state_name mean_poverty is_great
##   <chr> <chr>             <dbl> <chr>
## 1 IL    Illinois           11.9 Medium
## 2 IN    Indiana            10.7 Crappy
## 3 MI    Michigan           13.1 Crappy
## 4 OH    Ohio               12.5 Crappy
## 5 WI    Wisconsin          10.7 Pretty great
```

Recoding / Cleaning

Fun with strings ("character vectors")

```
my_string <- c("a", "b", "cdef")
my_string
```

```
## [1] "a"    "b"    "cdef"
```

```
str_detect(my_string, pattern = "b")  # detect a pattern
```

```
## [1] FALSE  TRUE FALSE
```

```
str_replace(my_string, pattern = "b", replace = "bee") # replace a pattern
```

```
## [1] "a"    "bee"  "cdef"
```

```
str_sub(my_string, start = 1, end = 2) # grab a substring
```

```
## [1] "a"  "b"  "cd"
```

Recoding / Cleaning

Indicator variables (aka dummy variables, aka binary variables): 0 or 1

```
# equals 1 if a condition is satisfied
arrests %>%
  mutate(
    placed_ive_lived = case_when(state_name == "Missouri" ~ 1,
                                 state_name == "California" ~ 1,
                                 state_name == "Wisconsin" ~ 1,
                                 TRUE ~ 0)
  ) %>%
  print()
```

```
## # A tibble: 50 x 3
##    state_name  Murder placed_ive_lived
##    <chr>        <dbl>            <dbl>
##  1 Alabama       13.2                0
##  2 Alaska        10                  0
##  3 Arizona        8.1                0
##  4 Arkansas       8.8                0
##  5 California     9                  1
##  6 Colorado       7.9                0
##  7 Connecticut    3.3                0
##  8 Delaware       5.9                0
```