An introduction to data analysis in R, and also to shark attacks

September 2019

Stefan McCabe

What are we doing here?

This is a quick and dirty introduction to data analysis in R. The goals are to:

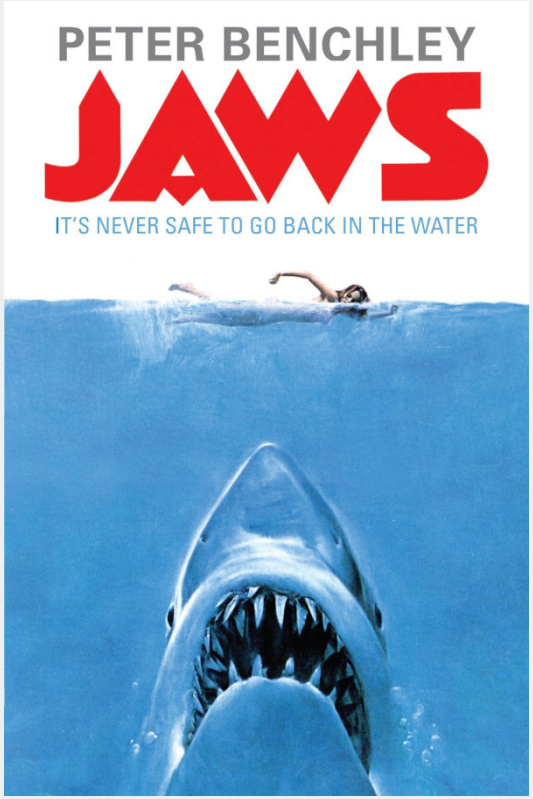
- introduce how to analyze data in R
- introduce how to visualize data in R

It is **not** anything resembling a course on statistics and data science.

I am a petty dictator, and have added some goals about things I care about:

- learn how to interpret a regression model
- learn why quantative social scientists answer questions they way they do

Why sharks?



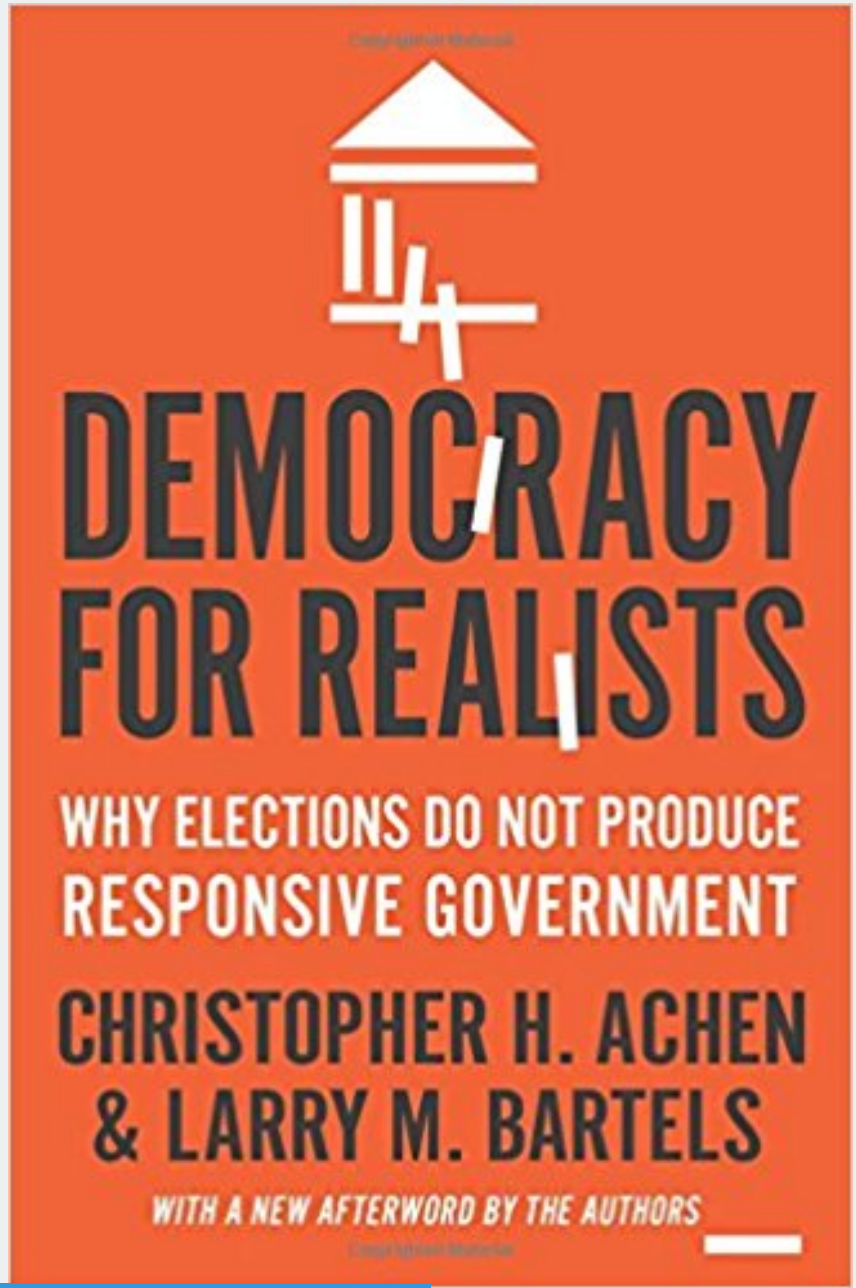
A representative article

A quote from said article

Consider the curious case of New Jersey in 1916: That summer, there was a string of deadly shark attacks along the Jersey Shore. As a result, Woodrow Wilson lost his home state in the presidential election.

Why, you ask? Because the beachfront towns (which rely on tourism) were negatively impacted by the attacks. Though Wilson wasn't responsible for the hungry sharks, he was the incumbent, and people vote against incumbents when things are bad.

Democracy for Realists



The Discourse I

Regardless of one's views on what the shark-attack result means for democracy, this debate is irrelevant if the empirical claim is invalid. Therefore, in this paper, we reassess the evidence that shark attacks influence presidential elections. There is virtually no compelling evidence that shark attacks influence elections, and any such effect—if it exists—must be substantively tiny. Whether or not voters are informed or competent in general, the evidence that shark attacks affect voter decisions is not nearly strong enough to generate dismay over the democratic process.

The Discourse II

including Essex makes little difference. But we have gone over this point in some detail because it illustrates clearly the pitfalls of Fowler and Hall's approach. In their critique, the historical context, both what we wrote and what others have written, goes unmentioned or even explicitly disregarded, as if qualitative evidence were irrelevant to quantitative researchers. That is not a recipe for good social science, and especially not for good historical research. Respect for qualitative evidence prevents many a quantitative blunder. As we will see, Fowler and Hall's arguments demonstrate over and over again how important it is to understand historical and cultural contexts and how badly quantitative researchers can go wrong when that step is skipped.

The Discourse III

Fowler and Hall proceed by ignoring or overriding this historical evidence. But even without attention to the history, careful consideration of the quantitative evidence would have provided ample warning not to proceed as Fowler and Hall do.

The Discourse IV

The only remaining point of disagreement is over the purported electoral effects of shark attacks in New Jersey in 1916. The evidence offered consists of two regressions with 20 and 14 non-independent observations, respectively, for which the results depend upon specification choices, a historical error, using the aberrant 1912 election to control for baseline preferences, and incorrect standard errors.

We agree with Achen and Bartels that historical context and substantive knowledge are vital for studying politics, but none of the historical facts and conjectures presented in their reply constitute persuasive evidence that shark attacks influenced elections even in this one case. Their claim that we ignore historical evidence is especially unpersuasive since it was our historical research that uncovered consequential errors in their analyses.

Why R?

- **Pragmatism**. It's another thing you can put on your resume.
- **Interdisciplinarity**. Social scientists love R.
- **Data cleaning**. The tidyverse provides a consistent and friendly interface for data cleaning and munging.
- **Visualization**. ggplot2 is R's killer app.
- **Libraries**. Living at the cutting edge of statistical modeling? You're probably going to want to know some R.

Preliminaries

I gave a similar but drier tutorial last year; a copy of that document is here. If you have questions about R's bizarre and terrible type system, or why there are so many `<-`s littered throughout the code, check it out.

Preliminaries: Libraries

You're going to want the following libraries (hopefully already installed):

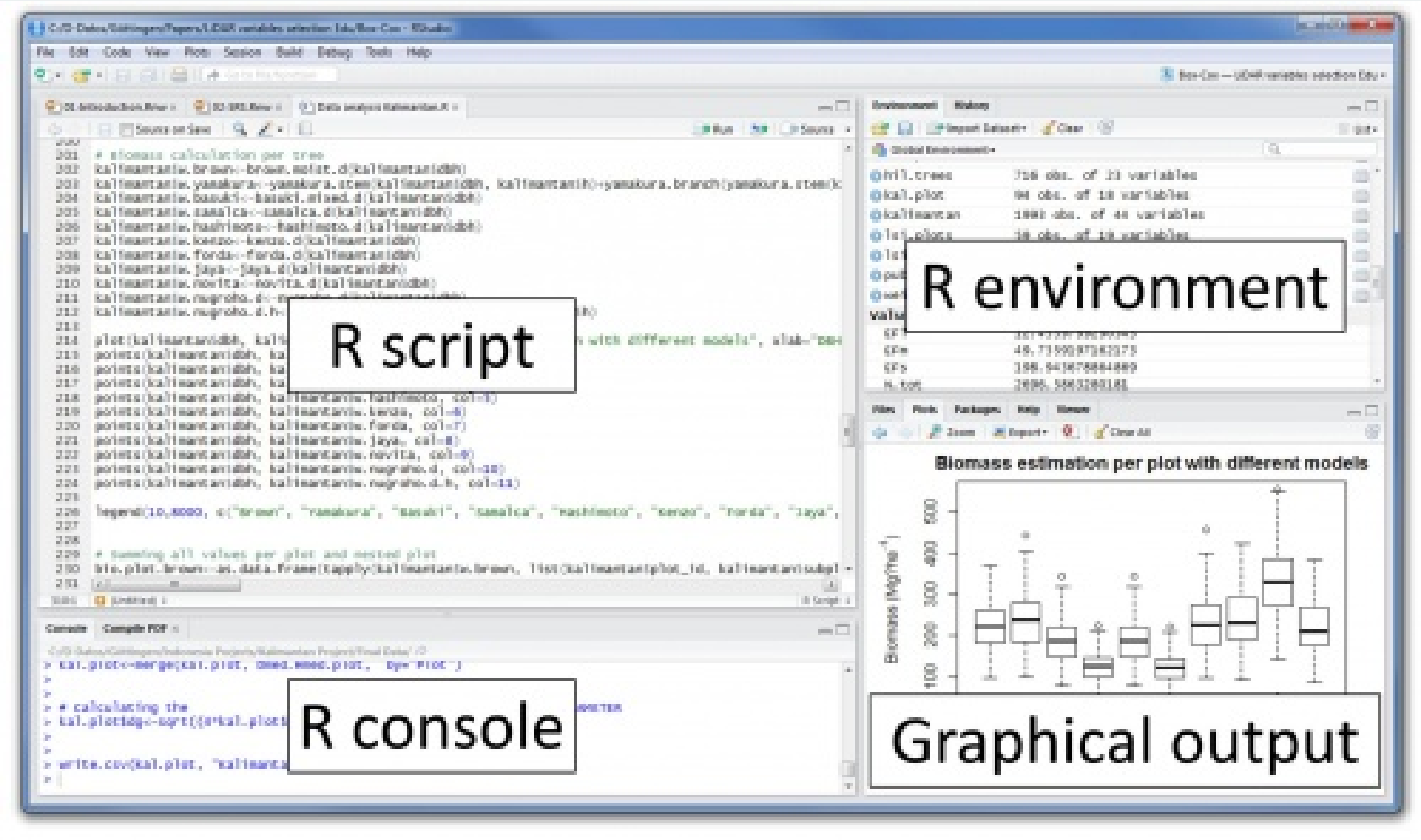
- ggplot2
- dplyr and tidyr
- readr and haven
- devtools

The `tidyverse` meta-package provides all the needed libraries, except for devtools.

Preliminaries: RStudio

RStudio is the IDE for R. Accept no substitutes.

Preliminaries: RStudio



Preliminaries: A quick R-Python Rosetta Stone

	R	Python
Assignment	`<-`	`=`
Import (i)	`library(x)`	`from x import *`
Import (ii)	`N/A`	`import x`
Calling Libraries (i)	`library(lib); func()`	`from lib import func; fu`
Calling Libraries (ii)	`lib::func()`	`import lib; lib.func()`
Concetenation	`c(x, y)`	`[x] + [y]`
String Concatenation	`paste('x', 'y')`	`'x' + ' y'`
Pipes	`x %>% f(y)`	`f(x, y)`
Conditionals	`ifelse(cond, 1, 2)`	`if cond: 1; else: 2`
Package Installation	`install.packages(x)`	`pip install x`
Vectorized Math	`c(1, 2) + c(1, 2)`	`np.array([1,2]) + np.arr`

Locating the data

The data is taken from Fowler and Hall's critique of an earlier paper on shark attacks. I've converted their Stata files to CSV for convenience; they're on the tutorial website.

readr

To load a data file, `readr` provides a consistent interface across formats (and if `readr` can't load it, try `haven`). Thus, we'll use the library's `read_csv` function instead of base R's `read.csv`.

(Note that `.` is valid in function and variable names in R; that is, `read.csv` is **not** a method of a class `read`.)

Loading the sharks data

```
# sharks <- read_csv("~/git/r_tutorial_f19/resources/shark.csv")
sharks <- read_csv("https://sdmccabe.github.io/r_tutorial_f19/re
```

```
## Parsed with column specification:
## cols(
##   county = col_character(),
##   wilson1912 = col_double(),
##   wilson1916 = col_double(),
##   beach = col_double(),
##   machine = col_double(),
##   mayhew = col_double(),
##   attack = col_double(),
##   coastal = col_double()
## )
```

Note that `read_csv` treats URLs and file paths the same when reading in a file. Storing a local copy of a file is almost always preferable, but using URLs can be

Examining the sharks data

```
dim(sharks)
head(sharks)
```

```
## [1] 21  8
## # A tibble: 6 x 8
##   county    wilson1912 wilson1916 beach machine mayhew attack coa
##   <chr>          <dbl>      <dbl> <dbl>   <dbl>  <dbl>  <dbl>  <dbl>
## 1 ATLANTIC       0.360      0.360     1       0      0      0      1
## 2 BERGEN         0.421      0.384     0       1      0      0      1
## 3 BURLINGTON     0.413      0.426     0       0      0      0      0
## 4 CAMDEN         0.394      0.433     0       0      1      0      1
## 5 CAPE MAY       0.435      0.419     1       0      0      0      1
## 6 CUMBERLAND     0.392      0.446     0       0      0      0      1
```

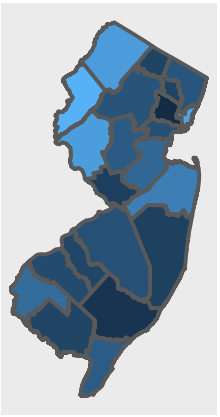
So, we have a data frame containing 21 observations of 8 attributes.

The columns

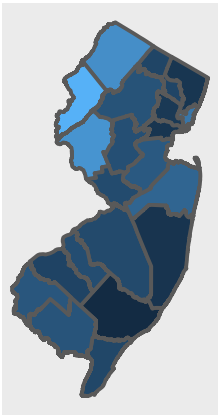
- **county**: the name of a county in New Jersey
- **wilson1912**: Woodrow Wilson's (three-party) share of the vote in 1912
- **wilson1916**: Woodrow Wilson's (two-party) share of the vote in 1916
- **beach**: does the county have substantial beach-related tourism?
- **machine**: were the politics of this county run by a political machine?
- **mayhew**: an alternative specification of machine
- **attack**: was there a shark attack in this county?
- **coastal**: is the county located on the coast?

The columns, visualized

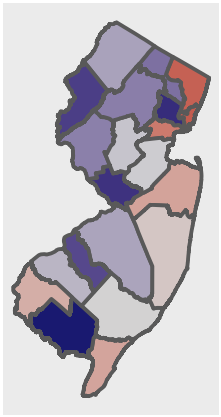
wilson1912



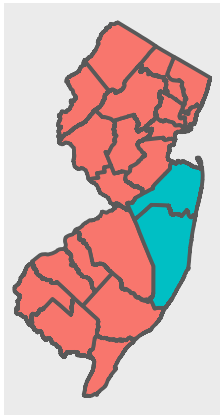
wilson1916



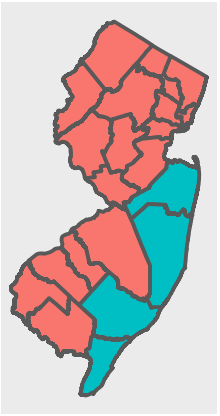
change



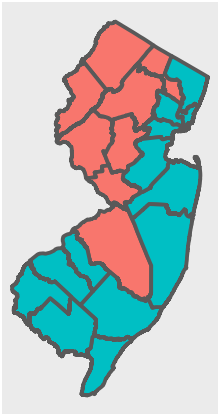
attack



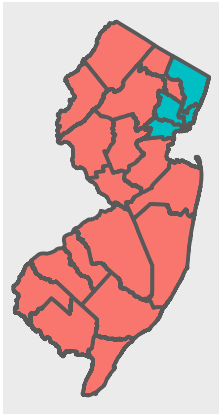
beach



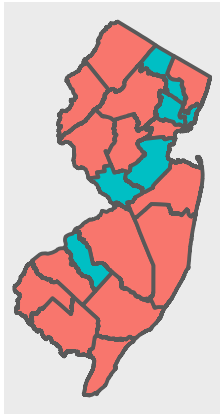
coastal



machine



mayhew



Indexing with $

Columns of a data frame are indexed with the $ operator, so we can pull out a single column like so:

```
sum(sharks$beach)
```

```
## [1] 4
```

There are four beach counties in New Jersey.

Numeric indexing I

We can also use numeric indices to pull out rows or columns:

```
head(sharks[,4]) # column indexing
```

```
## # A tibble: 6 x 1
##   beach
##   <dbl>
## 1     1
## 2     0
## 3     0
## 4     0
## 5     1
## 6     0
```

Numeric indexing II

```
sharks[1:4,] # row indexing, plus showing 1:4 to generate a
             # sequence from 1 to 4
```

```
## # A tibble: 4 x 8
##   county    wilson1912 wilson1916 beach machine mayhew attack coa
##   <chr>          <dbl>      <dbl> <dbl>   <dbl> <dbl>  <dbl>  <dbl>
## 1 ATLANTIC       0.360      0.360     1       0     0      0      1
## 2 BERGEN         0.421      0.384     0       1     0      0      1
## 3 BURLINGTON     0.413      0.426     0       0     0      0      0
## 4 CAMDEN         0.394      0.433     0       0     1      0      1
```

This can be handy for quick and dirty operations but is less explicit than the $ operator. Note also that **R is one-indexed**.

Summary statistics

R has most univariate and bivariate summary statistics built in, so they can be acessed rather simply:

```
mean(sharks$wilson1912)
cor(sharks$wilson1912, sharks$wilson1916)
```

```
## [1] 0.4386176
## [1] 0.9121978
```

The `summary()` function

A particularly useful function here is `summary`, which can be applied across an entire data frame:

```
summary(sharks)
```

```
##    county          wilson1912      wilson1916        beach
## Length:21        Min.   :0.3417  Min.   :0.3601  Min.   :0.0000
## Class :character  1st Qu.:0.3915  1st Qu.:0.4120  1st Qu.:0.00
## Mode  :character  Median :0.4203  Median :0.4331  Median :0.00
##                   Mean   :0.4386  Mean   :0.4475  Mean   :0.1905
##                   3rd Qu.:0.4635  3rd Qu.:0.4569  3rd Qu.:0.0000
##                   Max.   :0.5770  Max.   :0.6191  Max.   :1.0000
##    machine           mayhew           attack          coastal
## Min.   :0.0000  Min.   :0.0000  Min.   :0.00000  Min.   :0.000
## 1st Qu.:0.0000  1st Qu.:0.0000  1st Qu.:0.00000  1st Qu.:0.000
## Median :0.0000  Median :0.0000  Median :0.00000  Median :1.000
## Mean   :0.1905  Mean   :0.2857  Mean   :0.09524  Mean   :0.619
##                                                  3rd Qu.:1.000
```

A cautionary tale I

Summary statistics are helpful but are no substitute for a visual understanding of a dataset. To illustrate, consider Anscombe's Quartet.

```
anscombe <- datasets::anscombe # anscombe should already be in y
                               # but this makes it explicit
                               # note that :: pulls something fr
                               # compare to, say, the . in np.ze
```

A cautionary tale II

```
anscombe
```

```
##    x1 x2 x3 x4    y1   y2    y3    y4
## 1  10 10 10  8  8.04 9.14  7.46  6.58
## 2   8  8  8  8  6.95 8.14  6.77  5.76
## 3  13 13 13  8  7.58 8.74 12.74  7.71
## 4   9  9  9  8  8.81 8.77  7.11  8.84
## 5  11 11 11  8  8.33 9.26  7.81  8.47
## 6  14 14 14  8  9.96 8.10  8.84  7.04
## 7   6  6  6  8  7.24 6.13  6.08  5.25
## 8   4  4  4 19  4.26 3.10  5.39 12.50
## 9  12 12 12  8 10.84 9.13  8.15  5.56
## 10  7  7  7  8  4.82 7.26  6.42  7.91
## 11  5  5  5  8  5.68 4.74  5.73  6.89
```

Anscombe's summary statistics

```
round(map_dbl(anscombe[,5:8], mean), 3)
round(map_dbl(anscombe[,5:8], sd), 3)
round(map2_dbl(anscombe[,1:4], anscombe[,5:8], cor), 3)
```

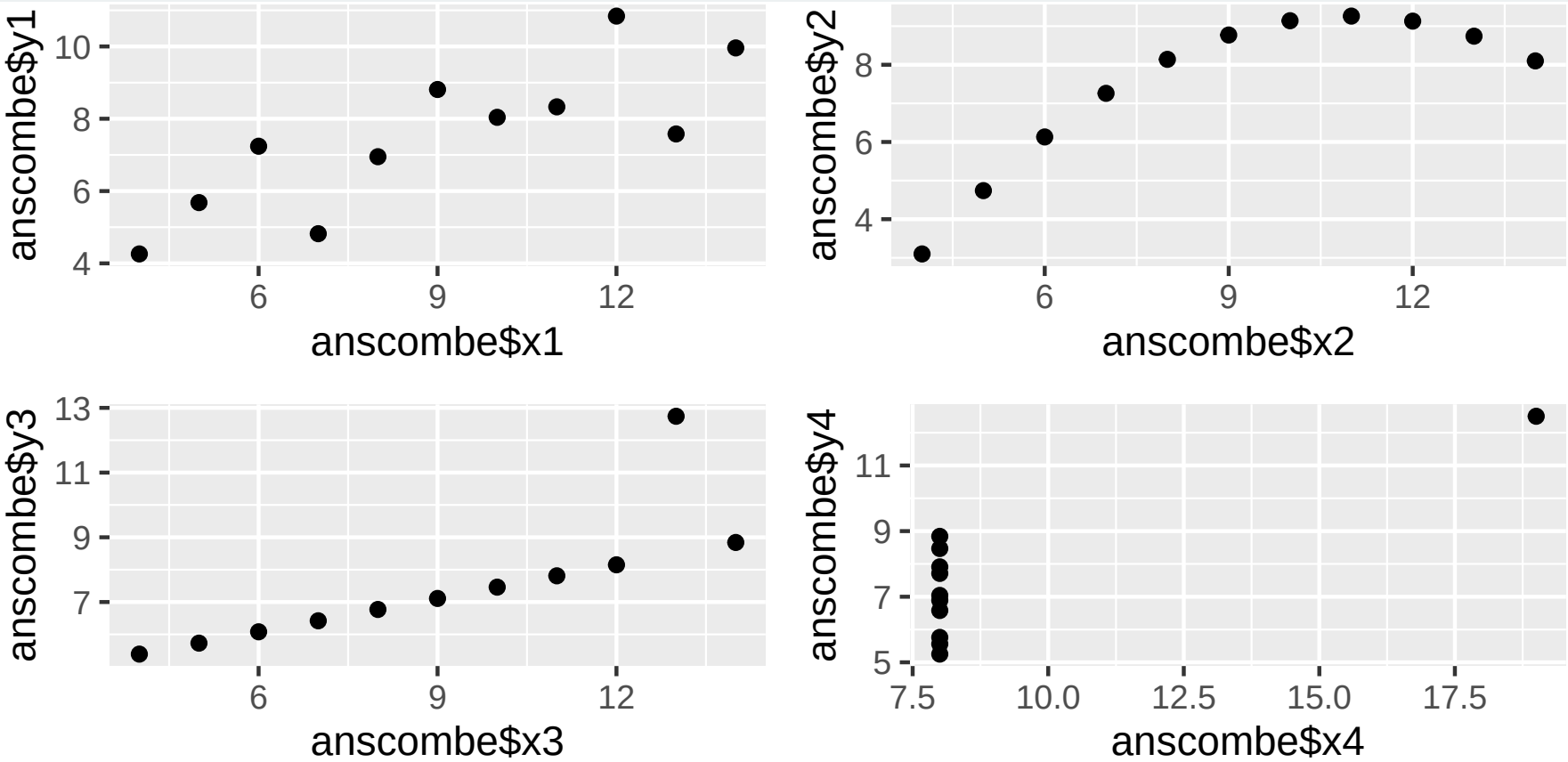
```
##    y1    y2    y3    y4
## 7.501 7.501 7.500 7.501
##    y1    y2    y3    y4
## 2.032 2.032 2.030 2.031
##    x1    x2    x3    x4
## 0.816 0.816 0.816 0.817
```

These summary statistics are quite similar. . .

Anscombe visualized I

```
a1 <- qplot(anscombe$x1, anscombe$y1)
a2 <- qplot(anscombe$x2, anscombe$y2)
a3 <- qplot(anscombe$x3, anscombe$y3)
a4 <- qplot(anscombe$x4, anscombe$y4)
gridExtra::grid.arrange(a1, a2, a3, a4)
```

Anscombe visualized II



Anscombe visualized III

. . . but the underlying data are quite different. Data visualization is your friend, and one of R's strengths.

`ggplot` versus "base R"

I'm deliberately omitting discussion of so-called "base R" plotting—although it is frequently useful—in favor of emphasizing `ggplot2`'s feature set. The analogy is slightly inapt, but think of `ggplot2` as playing a similar role relative to base R graphics as `seaborn` plays to `matplotlib`.

I provided some discussion of base R plotting in 2017's tutorial, so, again, check it out here.

The structure of a `ggplot()` call

Although simple plots—scatter plots and histograms, mostly—can be generated with the `qplot` function, most of the useful visualization features require wrapping your mind around `ggplot` and its associated functions. The goal of `ggplot2` is to implement a consistent grammar of graphics, and to that end most visualizations will have the same core elements:

- a **data** frame; that is, you want to have well-structured data ahead of time
- an **aes**thetic mapping telling R which columns to include, what your x-axis is, etc.
- various **geom** or **stat** functions to turn the mapped data into visualizations

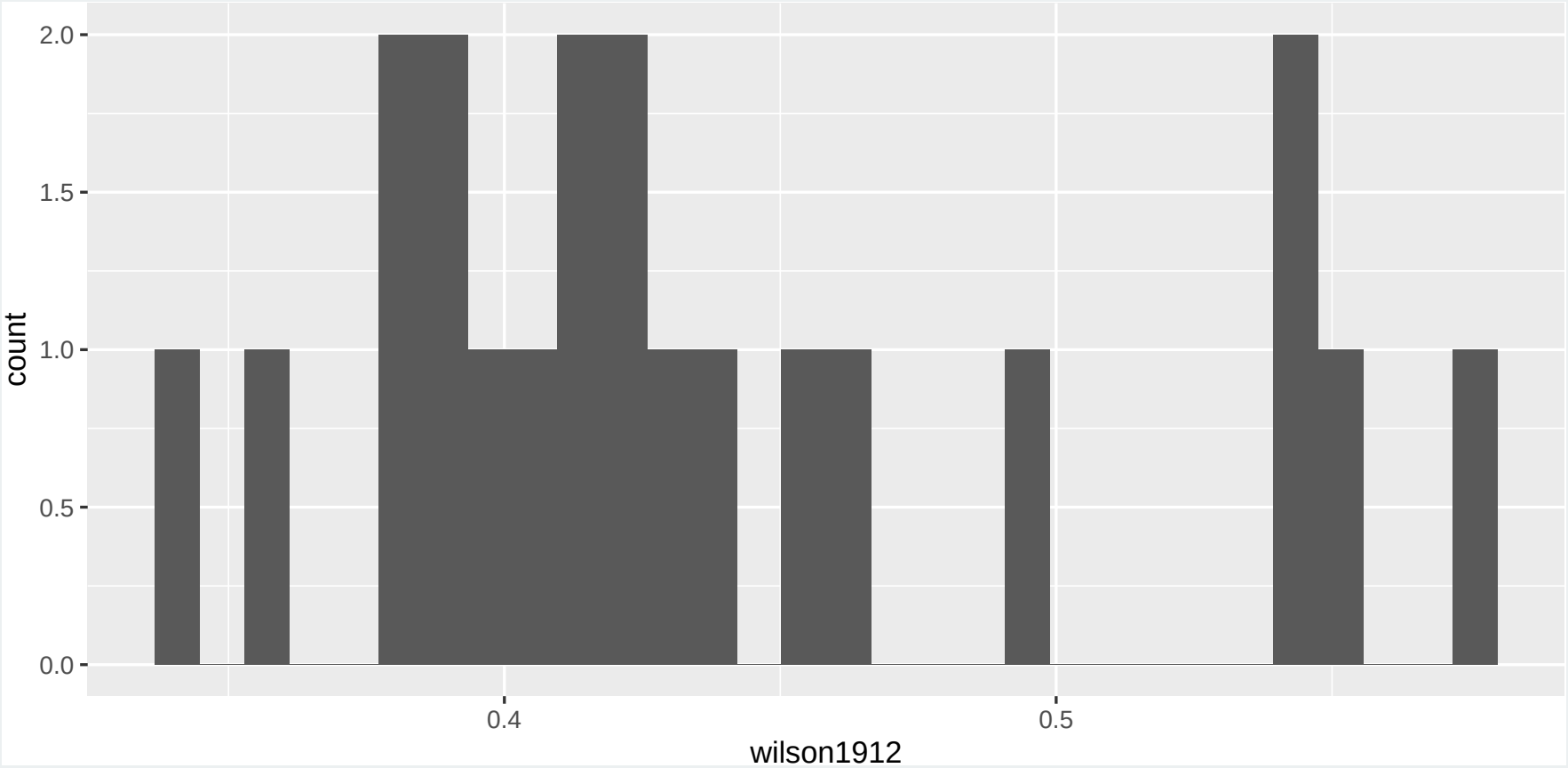
This sounds intimidating at first, but is relatively straightforward in practice. We already have the data, so once we figure out our mappings and geoms, we can make some plots.

Histograms I

What about a histogram of Wilson's 1912 vote share?

```
ggplot(data = sharks, mapping = aes(x = wilson1912)) +
  geom_histogram()
```

Histograms II

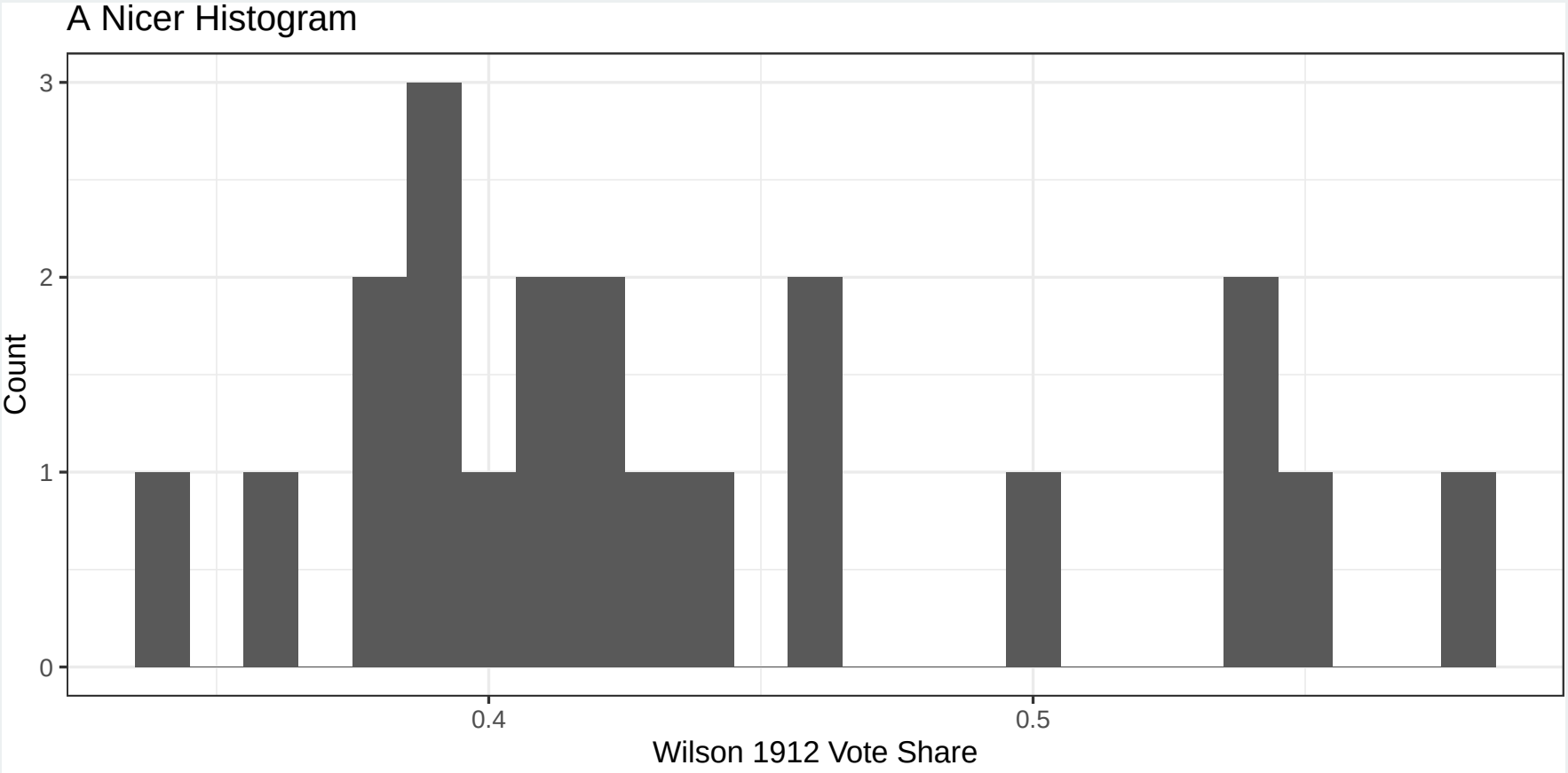


Fancier histograms I

Not so bad. Similarly to how, with `matplotlib`, we might build up a plot with multiple method calls, here we chain function calls with the `+` operator. So we can make the histogram slightly nicer:

```
ggplot(sharks, aes(x = wilson1912)) +
  geom_histogram(binwidth = 0.01) +
  theme_bw() +
  labs(x = "Wilson 1912 Vote Share",
       y = "Count",
       title = "A Nicer Histogram")
```

Fancier histograms II



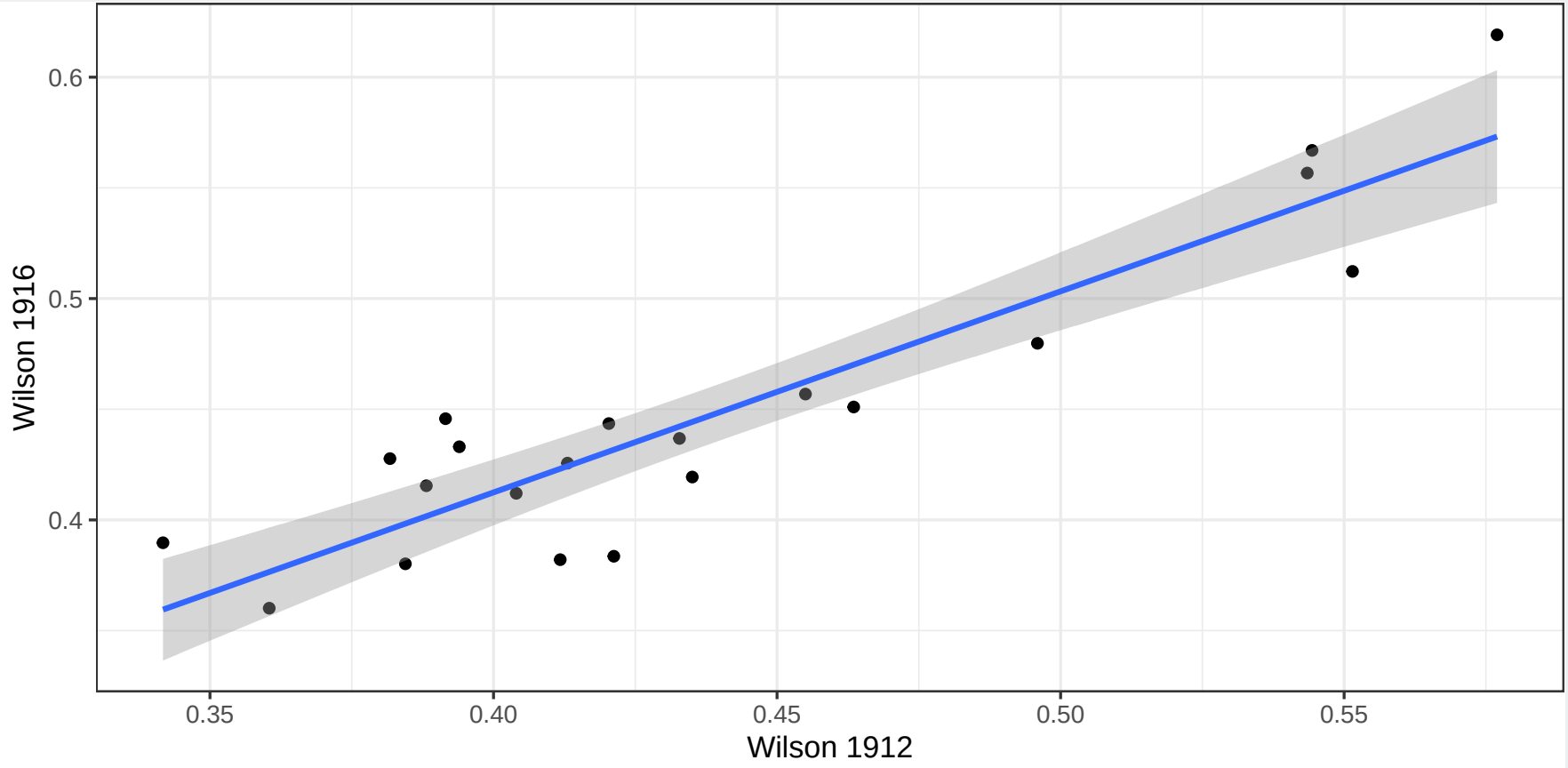
Bivariate geoms

We can also use geoms like `geom_point` or `geom_smooth` to plot a bivariate relationship, like that between Wilson's 1912 and 1916 vote shares. How consistent are election returns from cycle to cycle?

```
ggplot(sharks, aes(x = wilson1912, y = wilson1916)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_bw() +
  labs(x = "Wilson 1912",
       y = "Wilson 1916")
```

Here we are using two geoms in one plot; there's no limitation (except pragmatic ones) on the number you can use. So we used `geom_point` to draw a scatter plot and then `geom_smooth` to fit a straight line summarizing those data points (the additional parameter `method = "lm"` indicates to use a linear model instead of the potentially nonlinear method used by default).

Bivariate geoms II

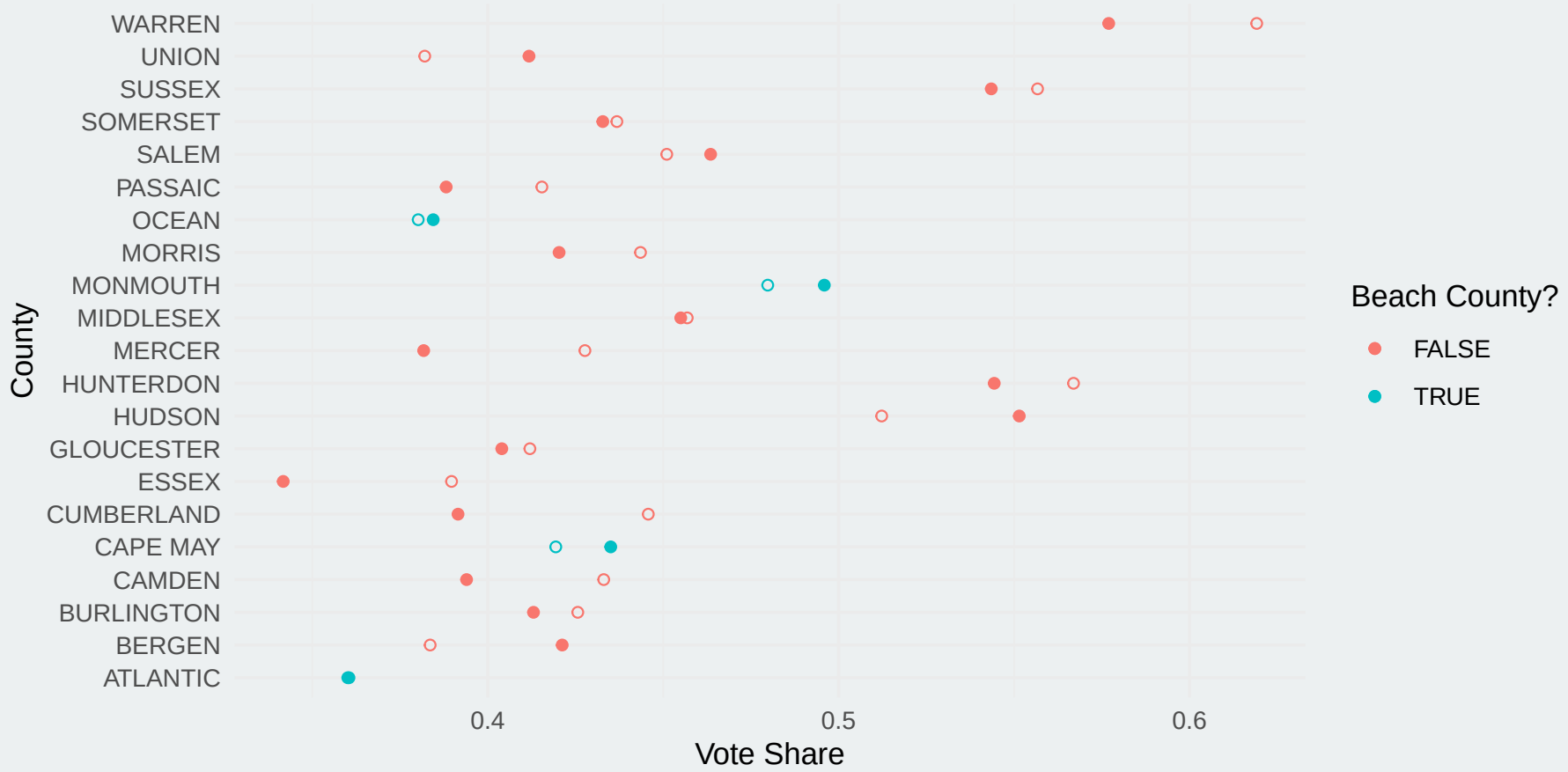


A more complicated example I

Another way to look at this is to go county-by-county and compare.

```
ggplot(sharks, aes(y = county, color = (beach == 1))) +
  geom_point(aes(x = wilson1912)) + # solid point
  geom_point(aes(x = wilson1916), shape = 1) + # hollow point
  labs(x = "Vote Share",
       y = "County",
       color = "Beach County?") +
  theme_minimal()
```

A more complicated example II



A more complicated example III

```
ggplot(sharks, aes(y = county, color = (beach == 1))) +
  geom_point(aes(x = wilson1912)) + # solid point
  geom_point(aes(x = wilson1916), shape = 1) + # hollow point
  labs(x = "Vote Share",
       y = "County",
       color = "Beach County?") +
  theme_minimal()
```

Aside from showing how easy it is to build up useful visualizations, this also starts to give us some substantive insight: all of the beach counties saw either no change or a decrease in Wilson vote share; no beach county saw a meaningful increase in Wilson support. That's interesting, at least, and some (weak) evidence for the claim that voters punished Wilson for the shark attacks.

Statistical models

What is a statistical model?

1. a probability model ($Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$)
2. a sampling model (independent samples from an infinite population)
3. a statistical generating mechanism (the error term is normally distributed)

Regression

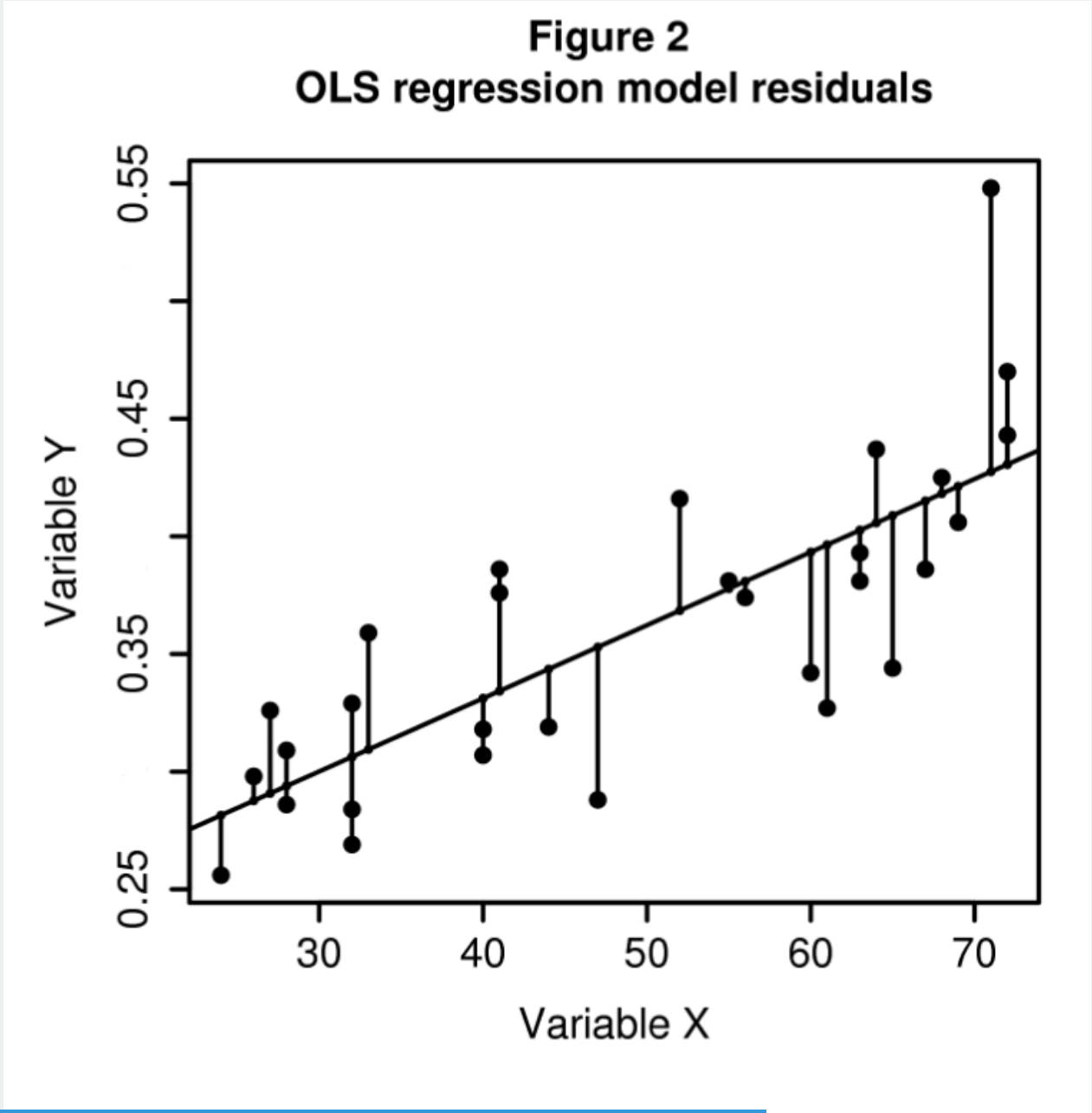
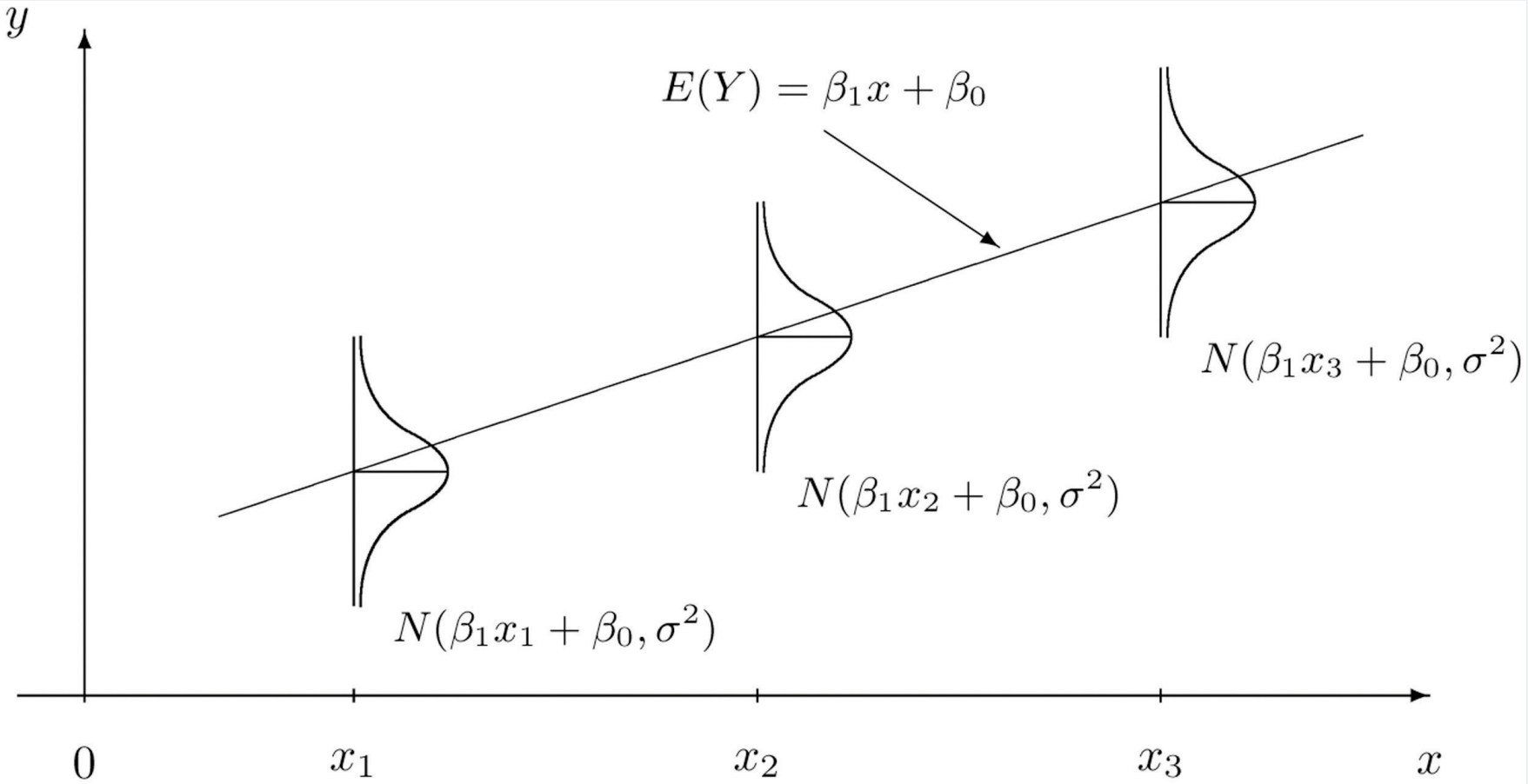


Figure 2
OLS regression model residuals
Variable Y
0.25
0.35
0.45
0.55
30
40
50
60
70
Variable X

Regression


y
$E(Y) = \beta_1 x + \beta_0$
$N(\beta_1 x_3 + \beta_0, \sigma^2)$
$N(\beta_1 x_2 + \beta_0, \sigma^2)$
$N(\beta_1 x_1 + \beta_0, \sigma^2)$
0
x_1
x_2
x_3
x

An introduction to formula syntax I

Most statistical modeling functions in R use some variant of formula syntax:

```
Y ~ X
```

Often, X will include multiple variables of interest, and potentially transformations of those variables or interactions bewteen variables.

```
Y ~ X1 + X2 + X1:X2 + I(X1^2) # transformations are nested in I(
Y ~ X1*X2 + I(X1^2) # this expresses the same equation
Y ~ X1*X2 + I(X1^2) - 1 # as above, but drop the intercept term
```

So, applying this to our example, we might want to regress Wilson's 1916 vote share on his 1912 vote share and see how much of the variance is explained by a simple **l**inear **m**odel.

An introduction to formula syntax II

```
m <- lm(wilson1916 ~ wilson1912, data = sharks)
```

```
summary(m)
```

```
##
## Call:
## lm(formula = wilson1916 ~ wilson1912, data = sharks)
##
## Residuals:
##       Min         1Q     Median         3Q        Max
## -0.048114 -0.019104 -0.004053  0.023390  0.045968
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.04908    0.04151   1.182    0.252
```

An introduction to formula syntax III

```
## wilson1912   0.90838    0.09361   9.704 8.52e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02807 on 19 degrees of freedom
## Multiple R-squared:  0.8321, Adjusted R-squared:  0.8233
## F-statistic: 94.17 on 1 and 19 DF,  p-value: 8.522e-09
```

```
stargazer(m, header = F, font.size = 'scriptsize')
```

An introduction to formula syntax IV

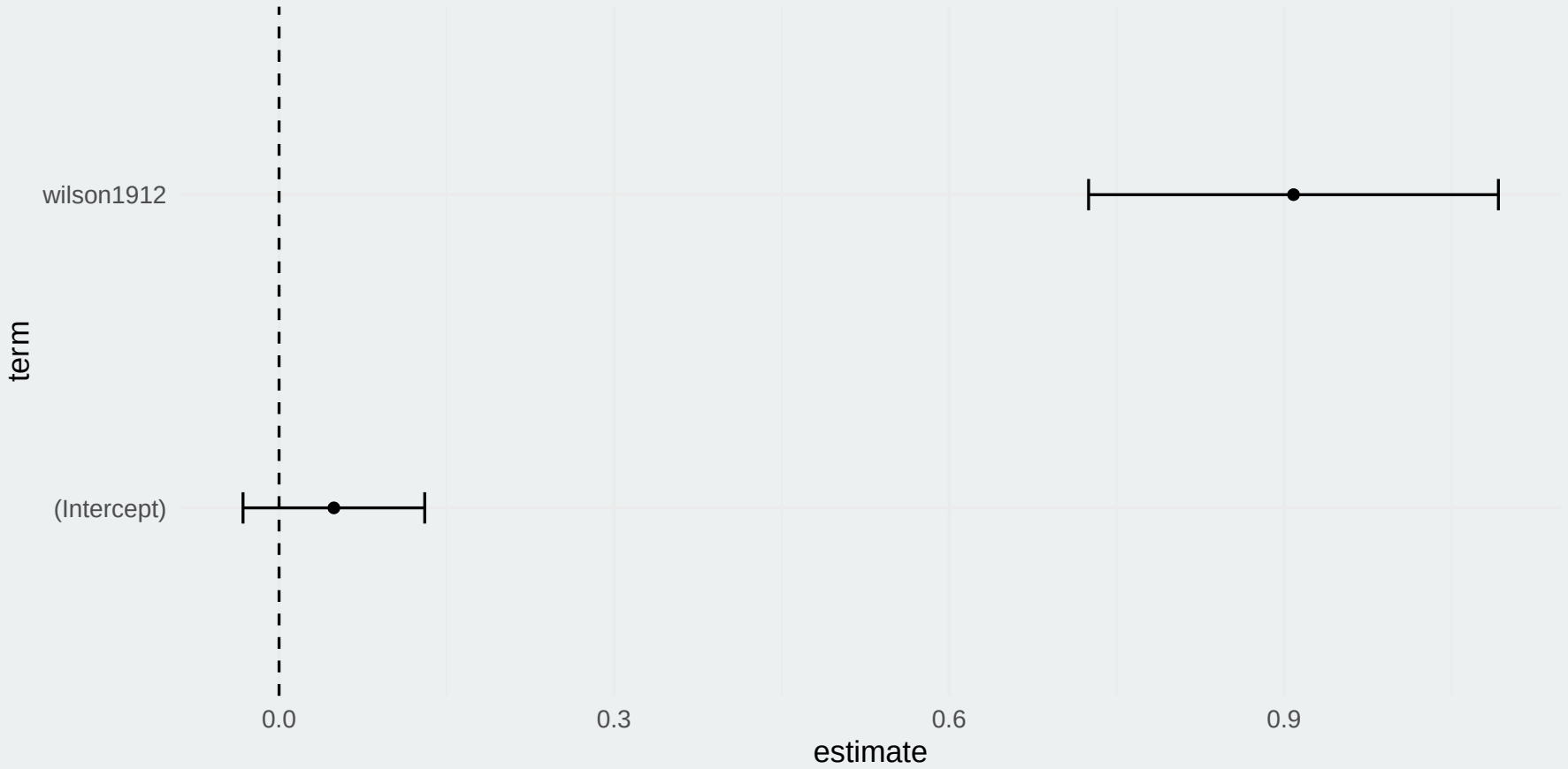
Table:

	Dependent variable:
	wilson1916
wilson1912	0.908^{***}
	(0.094)
Constant	0.049
	(0.042)
Observations	21
R^2	0.832
Adjusted R^2	0.823
Residual Std. Error	0.028 (df = 19)
F Statistic	94.166^{***} (df = 1; 19)
Note:	*p<0.1; **p<0.05; ***p<0.01

An introduction to formula syntax V

```
broom::tidy(m) %>%
  ggplot(aes(x = term)) +
  geom_point(aes(y = estimate)) +
  geom_errorbar(aes(ymin = estimate - 1.96 * std.error,
                    ymax = estimate + 1.96 * std.error),
                width = 0.1) +
  geom_hline(aes(yintercept = 0),
             linetype = 'dashed') +
  coord_flip() +
  theme_minimal()
```

An introduction to formula syntax VI



An exercise: build your own shark attack model!

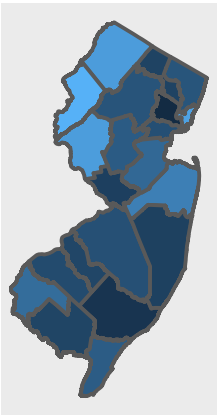
So, with all this in hand, we can start doing the sort of research that gets our name in Vox.

Ask yourself:

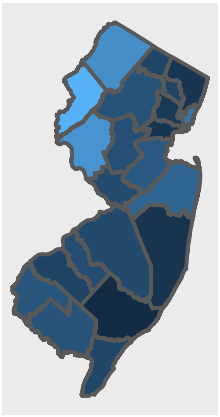
- What is the outcome of interest?
- What variables are appropriate to include?
- What would qualify as a substantively meaningful result?
- What is the interpretation of each of your coefficients?

The columns, visualized

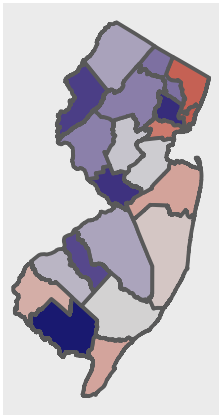
wilson1912



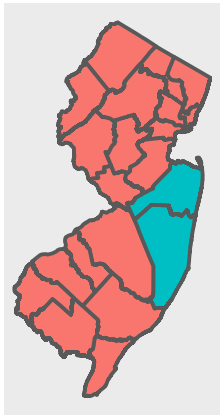
wilson1916



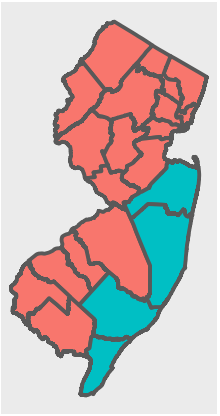
change



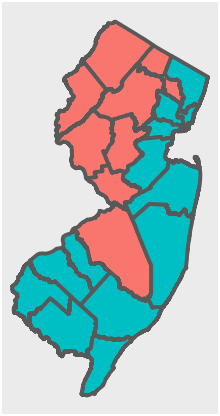
attack



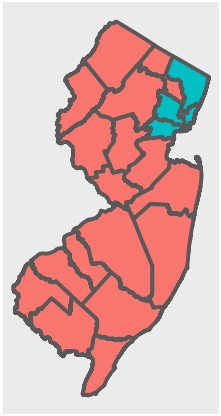
beach



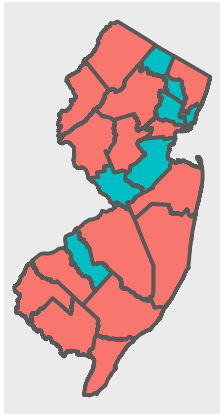
coastal



machine



mayhew



Achen and Bartels's model I

```
ab_sharks <- sharks[-7,]
dim(ab_sharks)
ab_model <- lm(wilson1916 ~ wilson1912 + machine + beach, data =
summary(ab_model)
```

```
## [1] 20  8
##
## Call:
## lm(formula = wilson1916 ~ wilson1912 + machine + beach, data = ab
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.033250 -0.006989  0.000657  0.005740  0.029520
##
```

Achen and Bartels's model II

```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.046093   0.027895   1.652  0.11794
## wilson1912   0.945336   0.061527  15.365 5.33e-11 ***
## machine     -0.056383   0.010939  -5.154 9.60e-05 ***
## beach       -0.032288   0.009882  -3.268  0.00484 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01702 on 16 degrees of freedom
## Multiple R-squared:  0.9459, Adjusted R-squared:  0.9357
## F-statistic: 93.21 on 3 and 16 DF,  p-value: 2.397e-10
```

References on shark attacks I

Shark attacks were a big story on Political Science Twitter in summer 2018.

- Democracy for Realists (Achen and Bartels 2016) is the canonical reference for the original shark attack claim, though it was first presented at a conference over a decade earlier.
- Fowler and Hall (2018a) present a critique of the result; the data used here is drawn from their critique.
- Achen and Bartels (2018) is a response to the critique.
- Fowler and Hall (2018b) get the last word. (For now, at least.)
- Lenz (2018) wonders if this has been a good use of anyone's time.

References on R I

Cheat sheets

The RStudio website has some terrific cheat sheets that I encourage everyone to bookmark (especially the data wrangling one, which I have to reference every time I use tidyr):

- Base R
- ggplot2
- RMarkdown
- RStudio IDE
- Data Transformation

Working in R

References on R II

If you're doing large-scale work in R, especially involving package development, here are some useful sources on development and R internals:

- The companion website to Wickham's Advanced R book.
- The companion website to Wickham's R Packages book.

Visualization

- Healy (2017) - Data Visualization: A Practical Introduction
- Ognyanova (2018) - Static and dynamic network visualization with R

Workflow

References on R III

Some good resources on structuring and approaching a data analysis project:

- **Healy (2016) - The Plain Person's Guide to Plain Text Social Science**
- Wilson et al (2017) - Good enough practices in scientific computing
- Wickham (2014) - Tidy data
- Leek (2015) - The elements of data analytic style
- The tidyverse style guide

Networks in R

Katya Ognyanova, one of David's former postdocs, has a good introduction to network analysis in R with `igraph`.

Code for NJ plots I

```
library(sf)
nj <- read_sf("~/Downloads/shape/New_Jersey_Counties.shp")
nj <- nj %>% left_join(sharks, by = c("COUNTY" = "county"))
```

```
p1 <- ggplot() +
  geom_sf(data = nj, aes(fill = wilson1912)) +
  coord_sf(ndiscr = 0) +
  scale_fill_continuous() +
  labs(title = "wilson1912") +
  theme(legend.position = "none")
```

Code for NJ plots II

```
p2 <- ggplot() +
  geom_sf(data = nj, aes(fill = wilson1916)) +
  coord_sf(ndiscr = 0) +
  scale_fill_continuous() +
  labs(title = "wilson1916") +
  theme(legend.position = "none")
```

```
p3 <- ggplot() +
  geom_sf(data = nj, aes(fill = wilson1916 - wilson1912)) +
  coord_sf(ndiscr = 0) +
  scale_fill_gradient2(low = "firebrick",
                       mid = "lightgray",
                       high = "midnightblue") +
  labs(title = "change") +
  theme(legend.position = "none")
```

Code for NJ plots III

```
p4 <- ggplot() +
  geom_sf(data = nj, aes(fill = as.factor(attack))) +
  coord_sf(ndiscr = 0) +
  scale_fill_discrete() +
  labs(title = "attack") +
  theme(legend.position = "none")
```

```
p5 <- ggplot() +
  geom_sf(data = nj, aes(fill = as.factor(beach))) +
  coord_sf(ndiscr = 0) +
  scale_fill_discrete() +
  labs(title = "beach") +
  theme(legend.position = "none")
```

Code for NJ plots IV

```
p7 <- ggplot() +
  geom_sf(data = nj, aes(fill = as.factor(machine))) +
  coord_sf(ndiscr = 0) +
  scale_fill_discrete() +
  labs(title = "machine") +
  theme(legend.position = "none")
```

```
p8 <- ggplot() +
  geom_sf(data = nj, aes(fill = as.factor(mayhew))) +
  coord_sf(ndiscr = 0) +
  scale_fill_discrete() +
  labs(title = "mayhew") +
  theme(legend.position = "none")
```

Code for NJ plots V

```
gridExtra::grid.arrange(p1,p2,p3,p4,
                        p5,p6,p7,p8,
                        ncol = 4)
```

Okay, but how do I do all this in Python? I

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
import seaborn as sns
```

```
py_sharks = pd.read_csv("https://sdmccabe.github.io/r_tutorial_f
py_ab_sharks = py_sharks.drop(py_sharks.index[6])
py_m = smf.  \
  ols("wilson1916 ~ wilson1912 + machine + beach", data = py_ab_
  fit()
print(py_m.summary())
```

Okay, but how do I do all this in Python? II

```
##                             OLS Regression Results
## ==============================================================================
##  Dep. Variable:                      wilson1916      R-
squared:                       0.946
## Model:                                 OLS    Adj. R-
squared:                  0.936
##  Method:                      Least Squares      F-
statistic:                     93.21
## Date:                    Mon, 19 Aug 2019    Prob (F-
statistic):           2.40e-10
##  Time:                            21:31:24    Log-
Likelihood:                 55.320
## No. Observations:                  20  AIC:                              -
102.6
## Df Residuals:                      16  BIC:                              -
98.66
```

Okay, but how do I do all this in Python? III

```
## Df Model:                           3
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
##  ------------------------------------------------------------------------------
----------------------
## Intercept      0.0461      0.028      1.652      0.118      -
0.013       0.105
## wilson1912    0.9453    0.062    15.365    0.000      0.815
## machine       -0.0564      0.011     -5.154      0.000      -
0.080      -0.033
## beach         -0.0323      0.010     -3.268      0.005      -
0.053      -0.011
## ==============================================================================
## Omnibus:                                   0.266    Durbin-
Watson:                   2.007
```

Okay, but how do I do all this in Python? IV

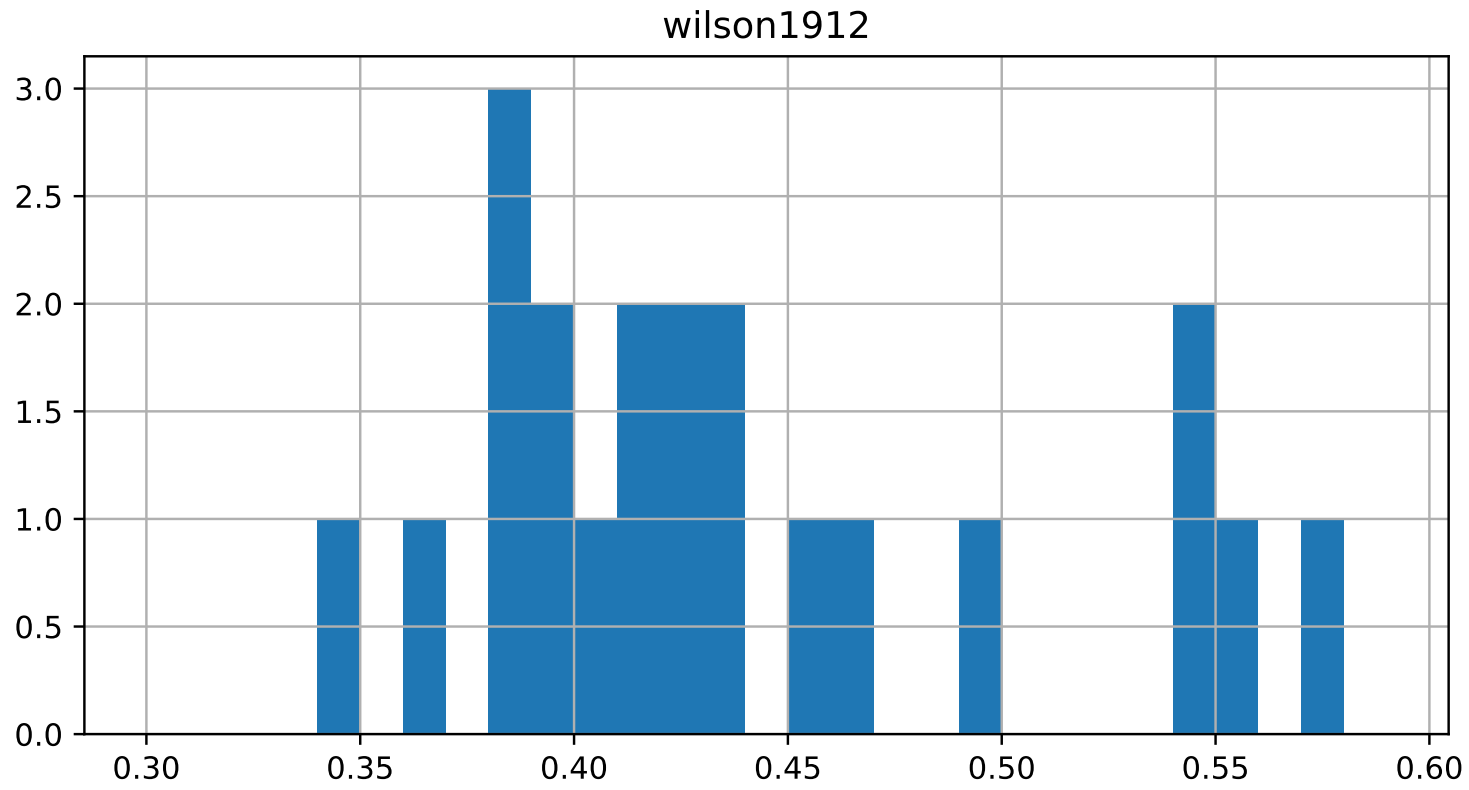
```
##  Prob(Omnibus):                          0.875    Jarque-
Bera (JB):                 0.013
## Skew:                          0.046   Prob(JB):                     0.994
## Kurtosis:                       2.917   Cond. No.                      19.9
## ==============================================================================
##
## Warnings:
## [1] Standard Errors assume that the covariance matrix of the erro
```

```
fig, ax = plt.subplots()
py_sharks.hist('wilson1912', bins = np.arange(0.3, 0.6, 0.01), a
```

```
## array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f50
##        dtype=object)
```

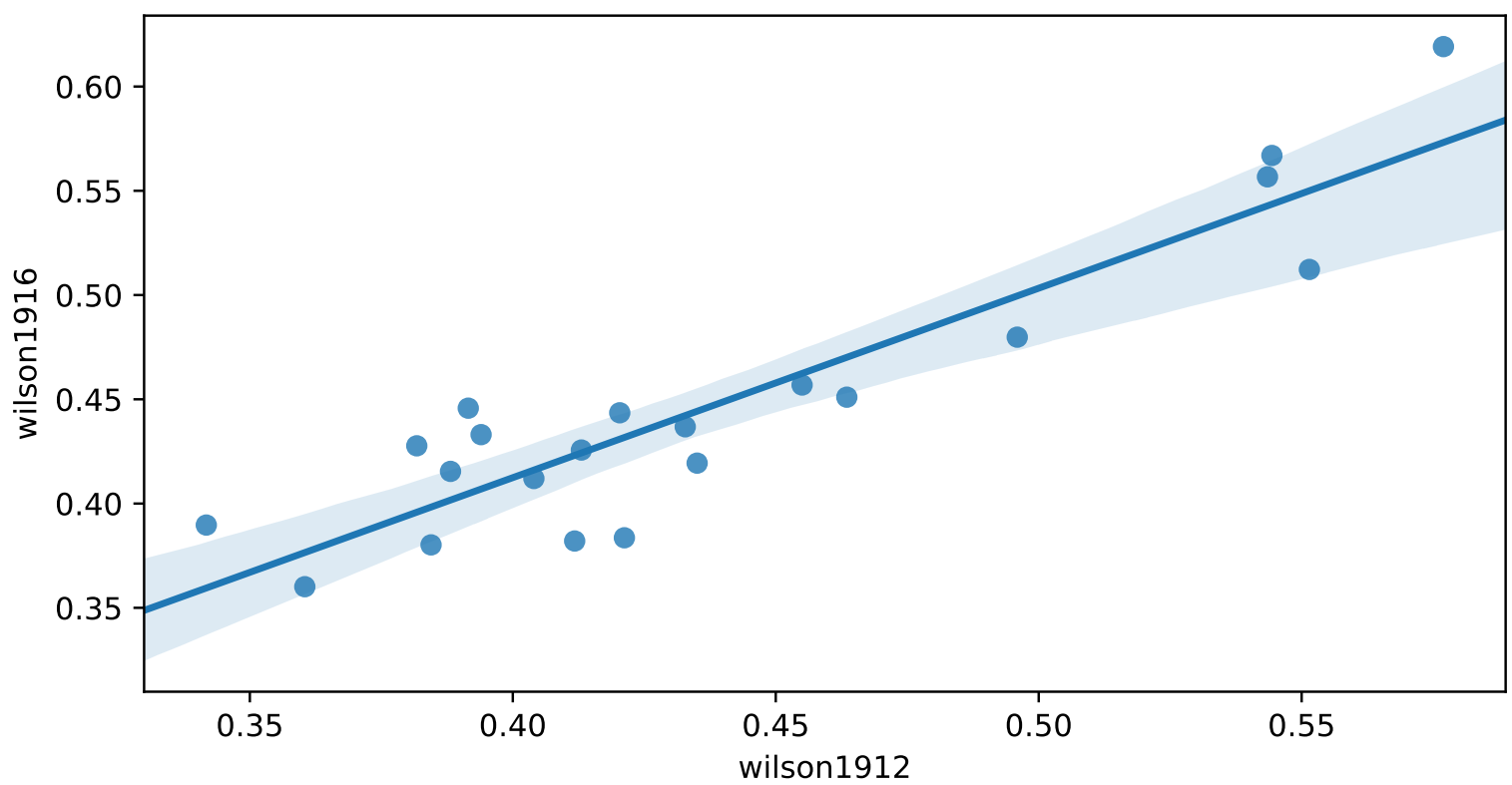
Okay, but how do I do all this in Python? V

```
plt.show()
```



Okay, but how do I do all this in Python? VI

```
fig2, ax2 = plt.subplots()
sns.regplot(py_sharks['wilson1912'], py_sharks['wilson1916'], ax
plt.show()
```



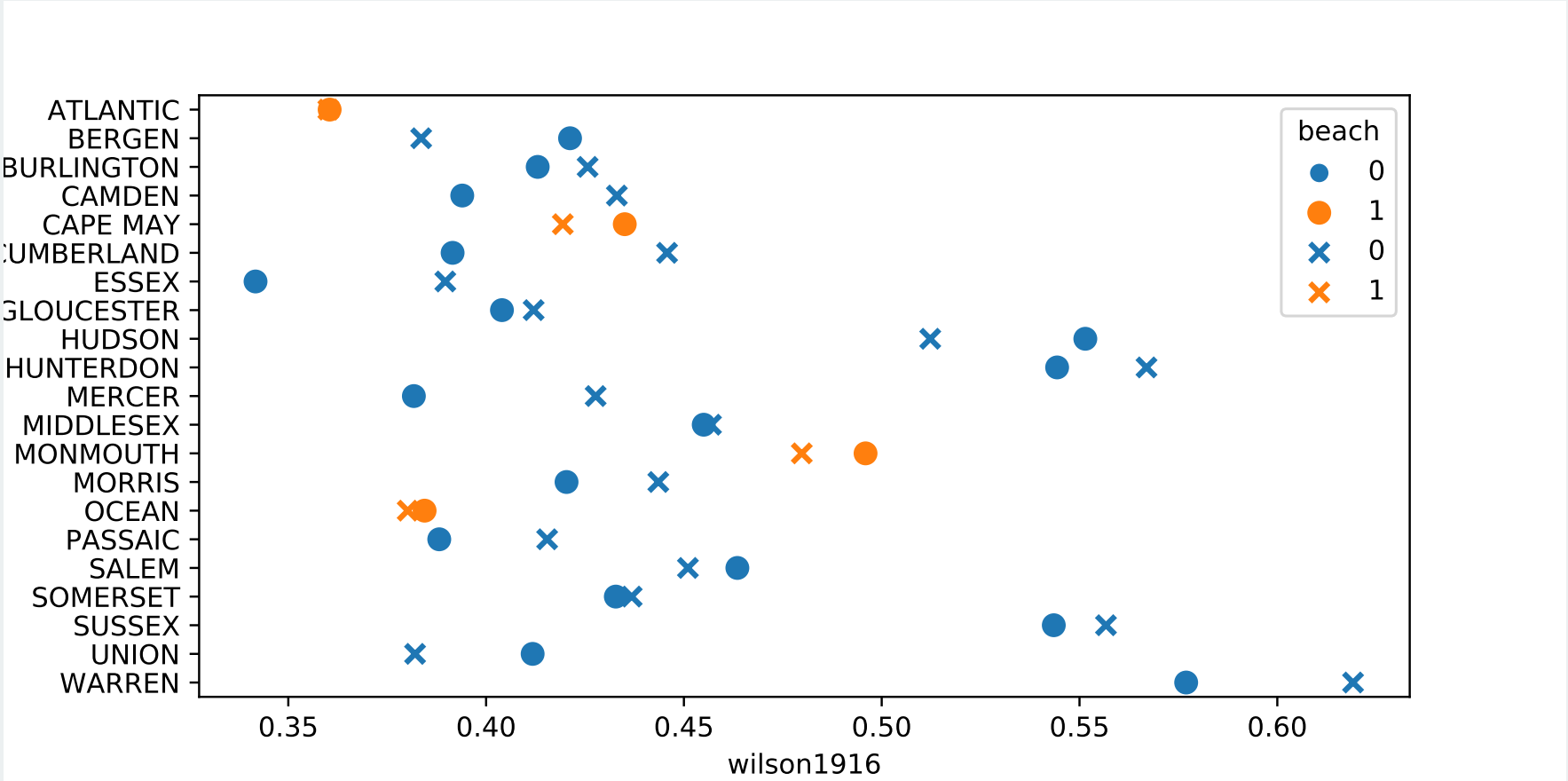
Okay, but how do I do all this in Python? VII

```
fig3, ax3 = plt.subplots()
sns.pointplot(y=py_sharks['county'], x=py_sharks['wilson1912'],
  hue = py_sharks['beach'], linestyles=' ', markers = 'o')
sns.pointplot(y=py_sharks['county'], x=py_sharks['wilson1916'],
  hue = py_sharks['beach'], linestyles=' ', markers = 'x')
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x7f50440599e
```

```
plt.show()
```

Okay, but how do I do all this in Python? VIII



I guess that made Python look simpler than R. . .