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

Stefan McCabe

August 2018

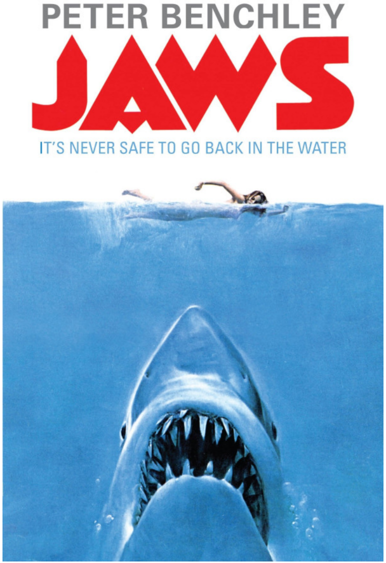
What are we doing here?

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.

Why sharks?



A representative article


EXPLAINERS POLITICS & POLICY WORLD CULTURE SCIENCE & HEALTH IDENTITIES ENERGY & ENVIRONMENT MORE

Two eminent political scientists: The problem with democracy is voters

Why almost everything you think about democracy is wrong.

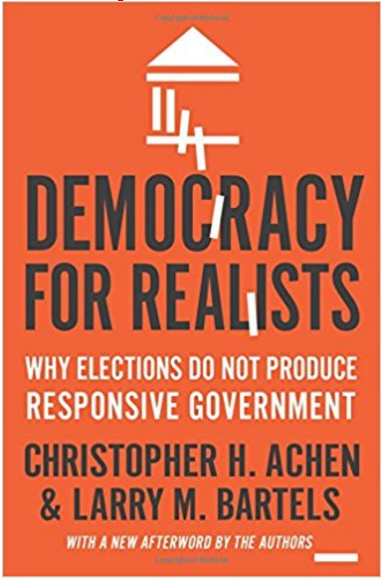
By Sean Illing | @seanilling | sean.illing@vox.com | Updated Jun 24, 2017, 12:12pm EDT

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



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

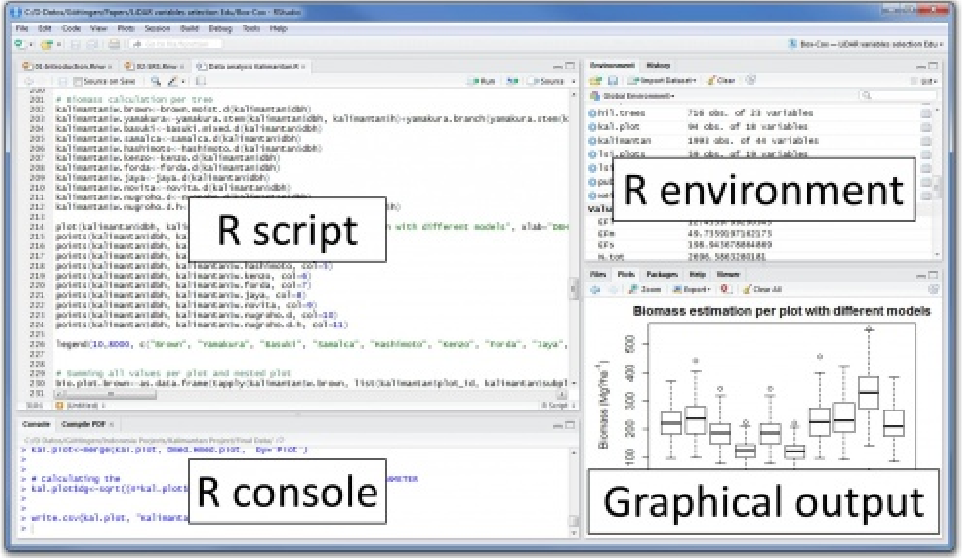
You're going to want the following libraries (hopefully already installed):

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

Preliminaries: RStudio

RStudio is *the* IDE for R. Accept no substitutes.

Preliminaries: RStudio



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_f18/resources/shark.csv")
sharks <- read_csv("https://sdmccabe.github.io/r_tutorial_f18/resources/shark.
```

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

Note that `read_csv` treats URLs and file paths the same when reading in a file.

Examining the sharks data

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

```
## [1] 21  8
## # A tibble: 6 x 8
##   county     wilson1912 wilson1916 beach machine mayhew attack coastal
##   <chr>           <dbl>      <dbl> <int>   <int>  <int>  <int>   <int>
## 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?

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:

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

```
## # A tibble: 6 x 1
##   beach
##   <int>
## 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 coastal
##   <chr>           <dbl>      <dbl> <int>   <int>  <int>  <int>   <int>
## 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:

```r
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:

```r
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.:0.0000
##  Mode  :character   Median :0.4203   Median :0.4331   Median :0.0000
##                     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
```

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 your namespace
                               # but this makes it explicit
                               # note that :: pulls something from a library
                               # compare to, say, the . in np.zeros()
```

A cautionary tale II

```r
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)
```

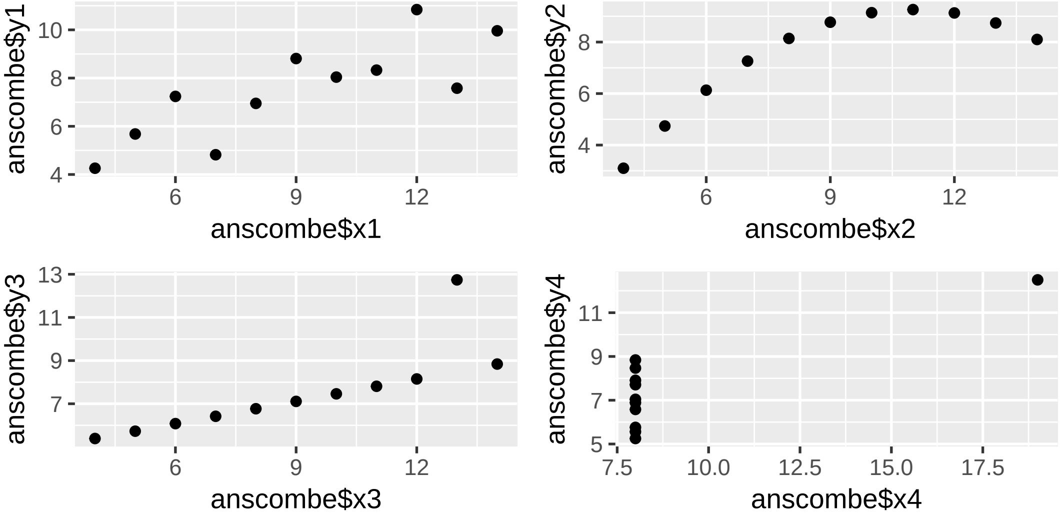
```
##    y1    y2    y3    y4
## 7.501 7.501 7.500 7.501
##    y1    y2    y3    y4
## 2.032 2.032 2.030 2.031
##    x1    x2    x3    x4
## 0.816 0.816 0.816 0.817
```

These summary statistics are quite similar…

Anscombe visualized I

```r
p1 <- qplot(anscombe$x1, anscombe$y1)
p2 <- qplot(anscombe$x2, anscombe$y2)
p3 <- qplot(anscombe$x3, anscombe$y3)
p4 <- qplot(anscombe$x4, anscombe$y4)
gridExtra::grid.arrange(p1, p2, p3, p4)
```

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 last year'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

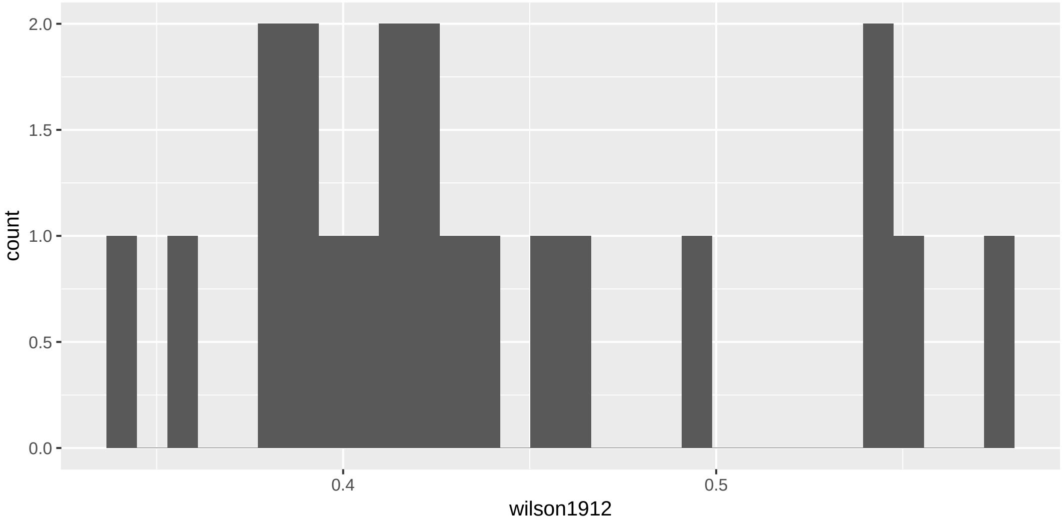
This sounds intimidating at first, but is relatively straightforward in practice. We already have the data, so once we figure out our mappings and geoms, we can make some plots.

Histograms I

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

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

Histograms II

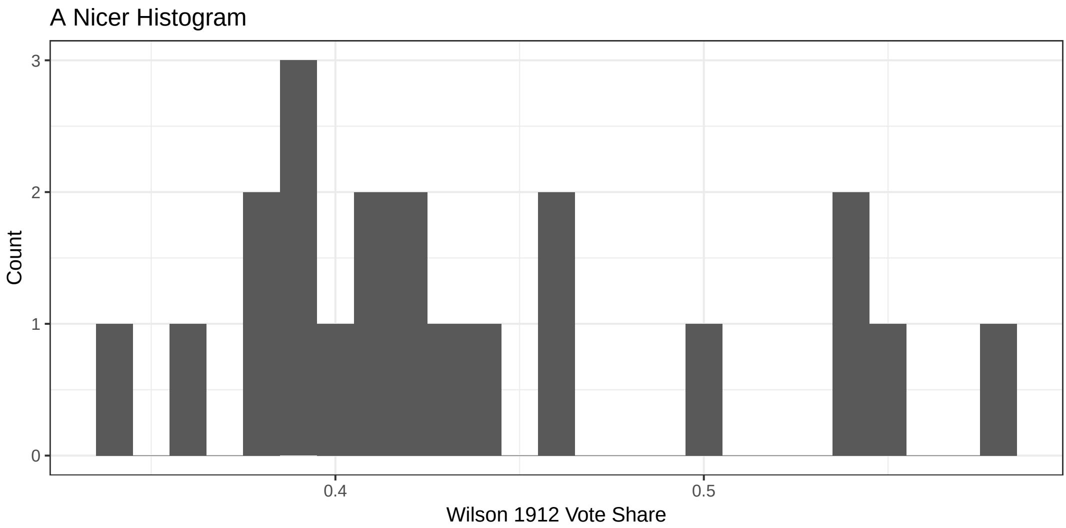


Fancier histograms I

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:

```r
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



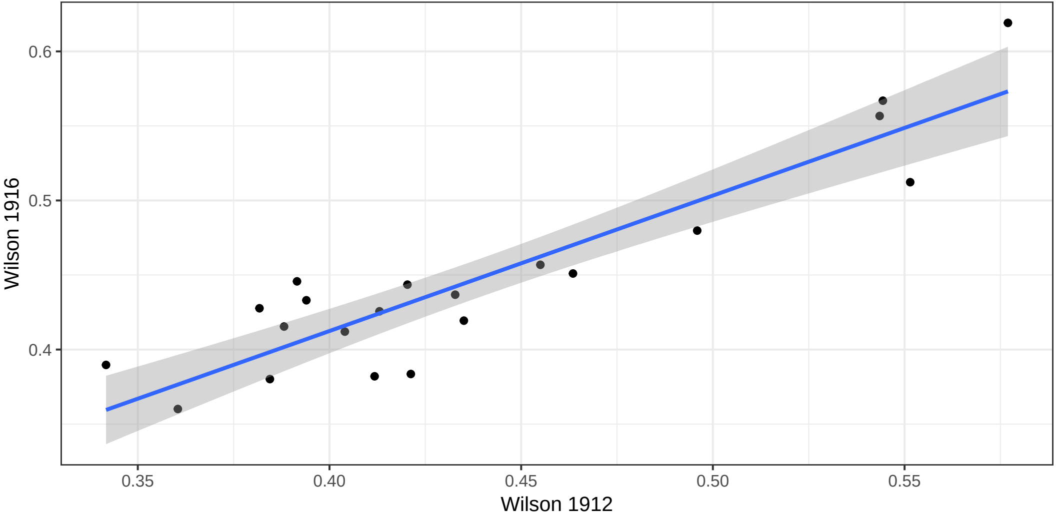
Bivariate geoms

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

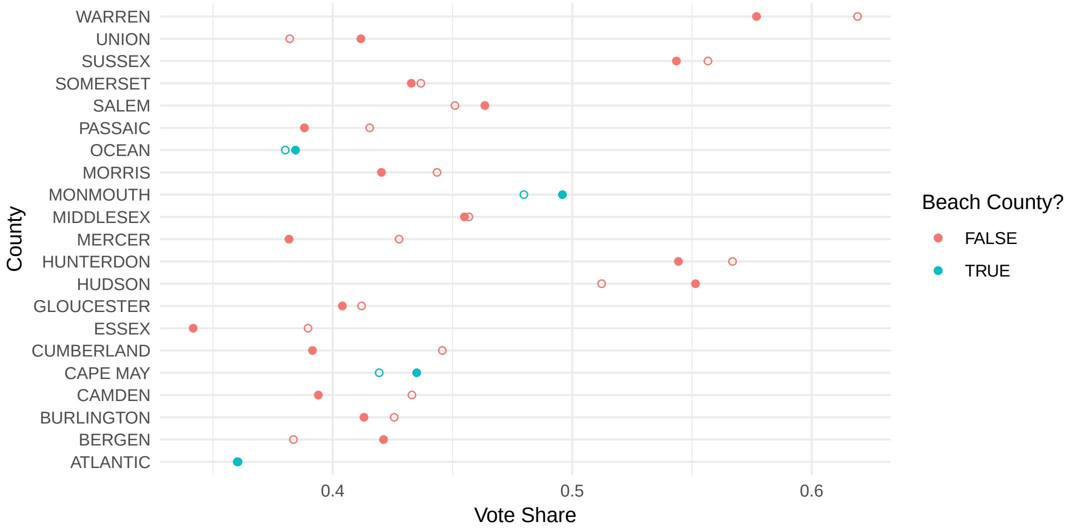


A more complicated example I

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

```r
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.

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 linear model.

An introduction to formula syntax II

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

```r
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|)
```

An introduction to formula syntax III

```
## (Intercept)  0.04908    0.04151   1.182    0.252
## 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
```

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?

Achen and Bartels's model I

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

```
## [1] 20  8
##
## Call:
## lm(formula = wilson1916 ~ wilson1912 + machine + beach, data = ab_sharks)
##
## 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 this summer.

- *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`.