Working with Data in R

Sam Frederick
sdf2128@columbia.edu

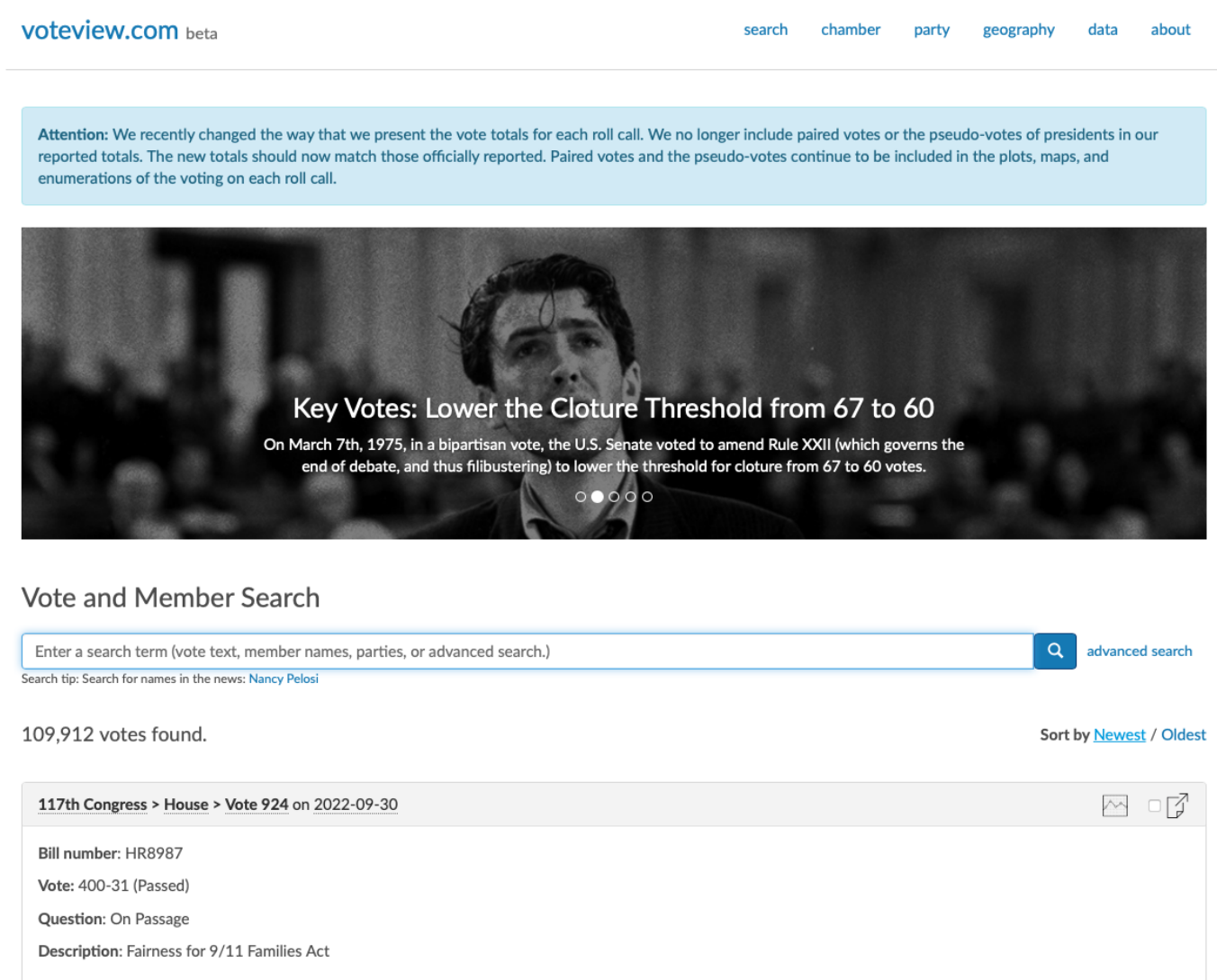
Data Sources for Final Projects

- Harvard Dataverse
- Cooperative Election Study
- American National Election Study
- Voteview (NOMINATE Scores for Congress)
- Center for Effective Lawmaking (Effective Lawmaking in Congress)
- Countless others
- Can also make your own

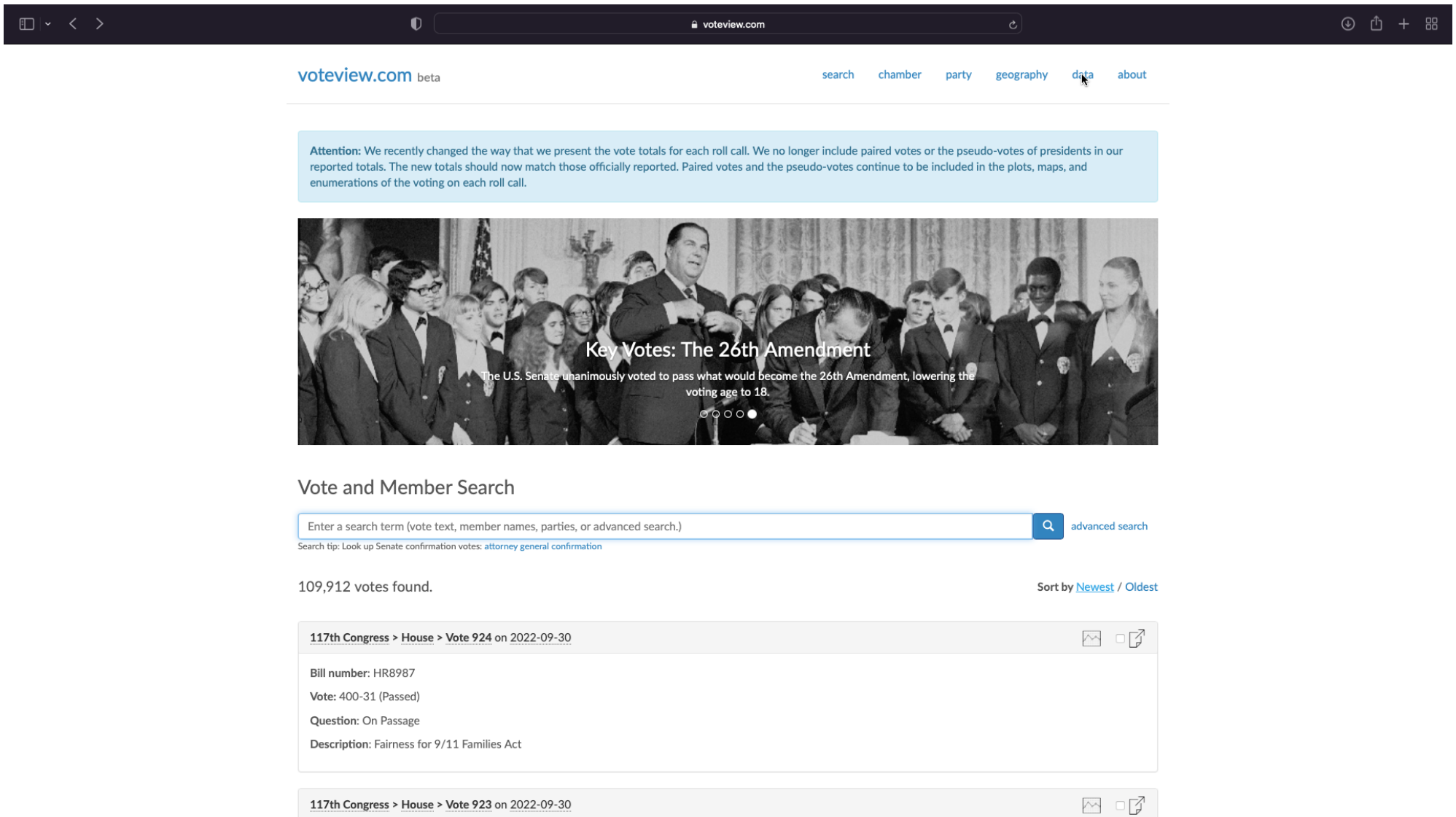
Reading Different File Types

File Type	R Function
.csv	`read.csv()`
.RData	`load()`
.txt	`read.delim()`
.dta	`read_dta()` from `haven` package
.xlsx	`read_excel()` from `readxl` package

Downloading Data



Downloading Data



Downloading Data

voteview.com

voteview.com beta

search chamber party geography data about

Attention: We recently changed the way that we present the vote totals for each roll call. We no longer include paired votes or the pseudo-votes of presidents in our reported totals. The new totals should now match those officially reported. Paired votes and the pseudo-votes continue to be included in the plots, maps, and enumerations of the voting on each roll call.

Realtime NOMINATE Ideology and Related Data

This section contains download links for NOMINATE scores and other data that we make available to the public, in addition to tutorial articles explaining how to generate popular ancillary data from our data exports. Please continue by choosing the data you wish to download.

Data is updated live, as new votes are taken. If you are an institutional user of our data and wish to be notified before major or breaking changes are made to the data, please see the About page.

Please cite the dataset as:

Lewis, Jeffrey B., Keith Poole, Howard Rosenthal, Adam Boche, Aaron Rudkin, and Luke Sonnet (2022). *Voteview: Congressional Roll-Call Votes Database*. https://voteview.com/

Data Type: Member Ideology

Chamber: House Only

Congress: 116th (2019 - 2021)

File Format: CSV (Recommended)

Download Data

Member Ideology

This data includes basic biographical information (state, district, party, name) and ideological scores for members of the selected congresses.

Click here for help using this data

Ancillary Data and Analyses

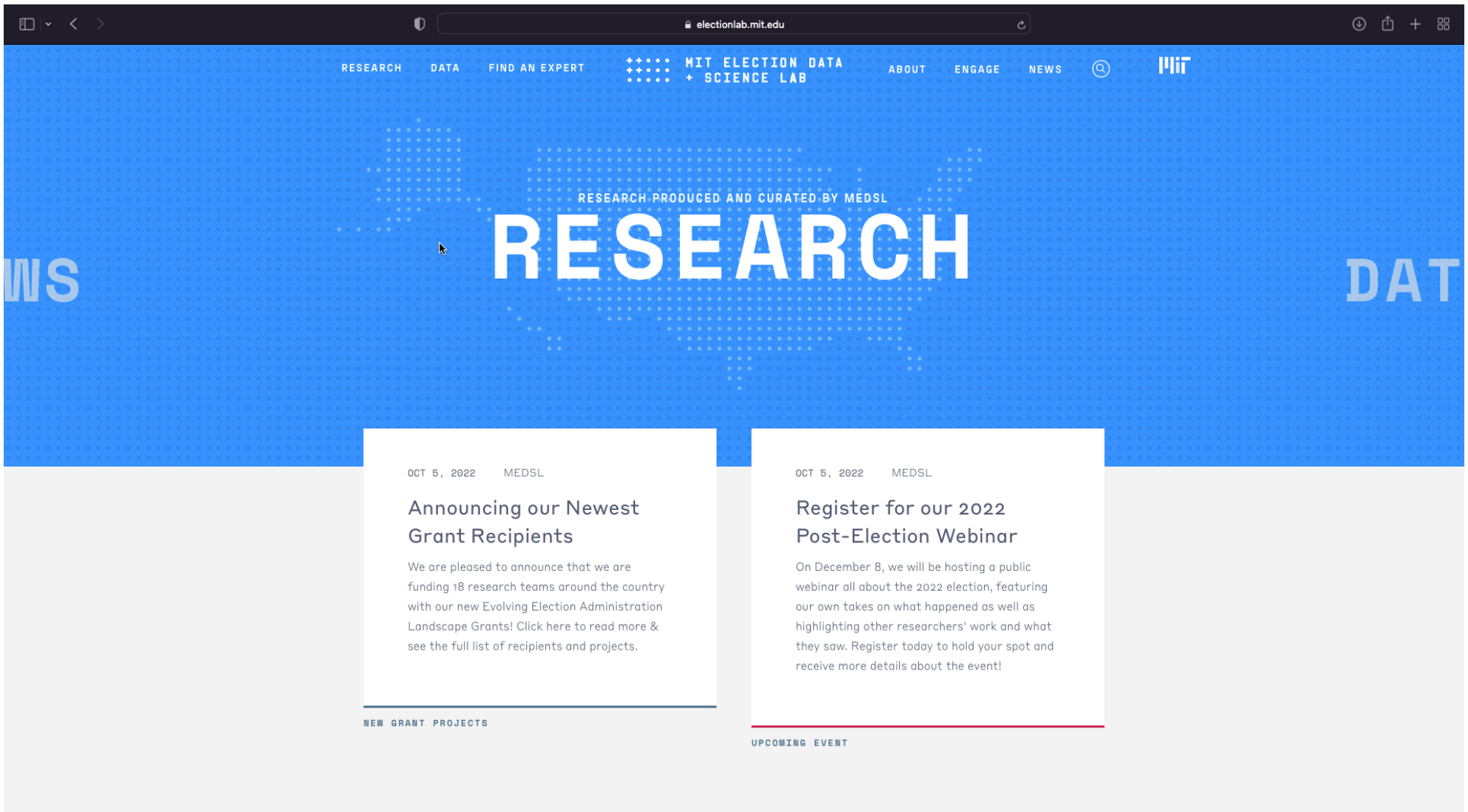
We are pleased to present a collection of articles discussing data and analyses that make use of NOMINATE / voteview.com, along with the source code used to produce the analyses. We hope these will be of use to scholars, journalists, and students interested in producing analysis using our data:

Attendance of Senators and House Members Running for President: Track the rate of participation in roll call votes of Members of Congress who are running for President in 2020

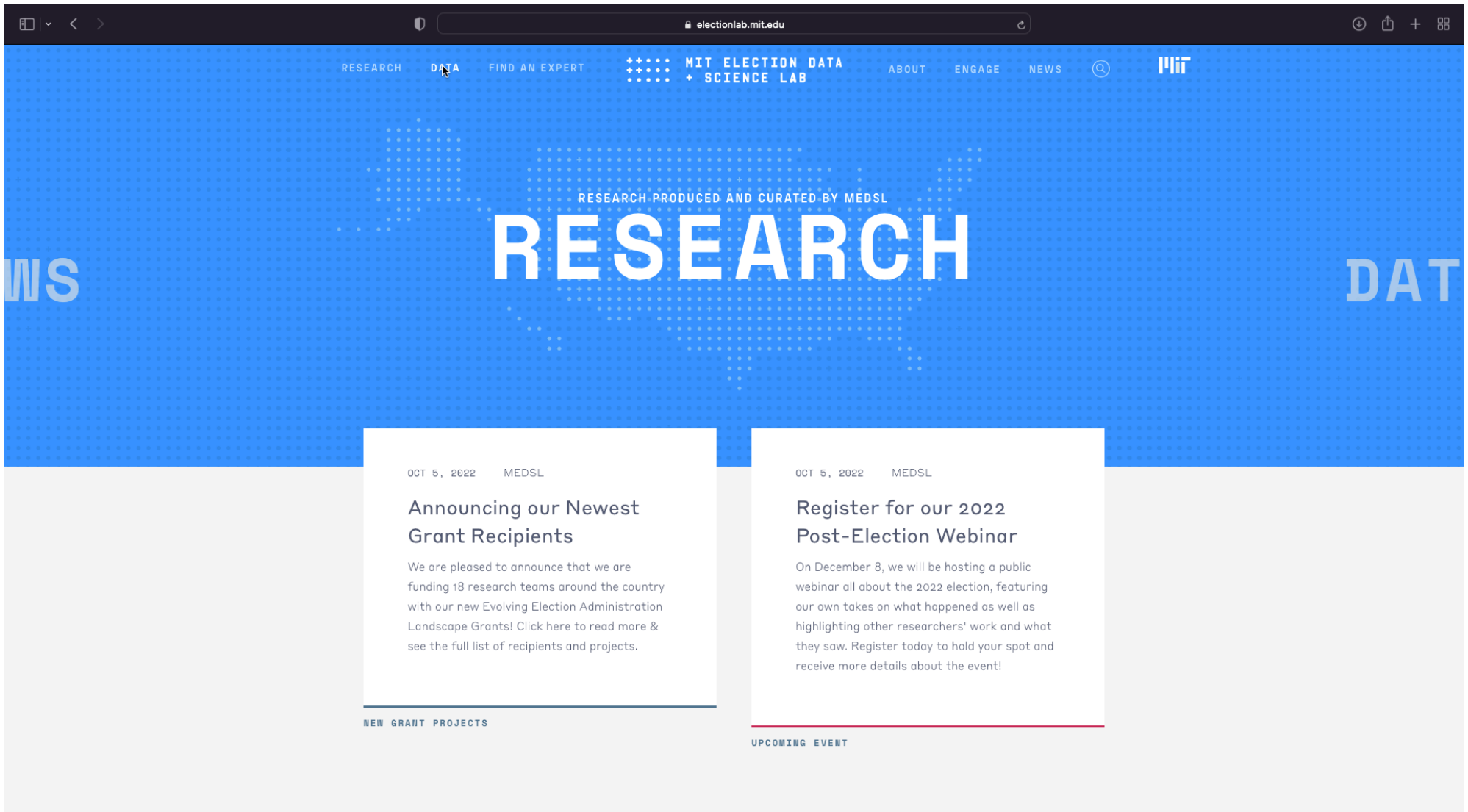
Clausen, Peltzman, and Issue codes for 1st to 113th Congresses: This short article provides descriptions for the numeric Clausen, Peltzman, and Issue codes supplied with our data.

Joe Biden's Party Loyalty in the Senate: This article calculates Joe Biden's career rate of party loyalty in the Senate and compares

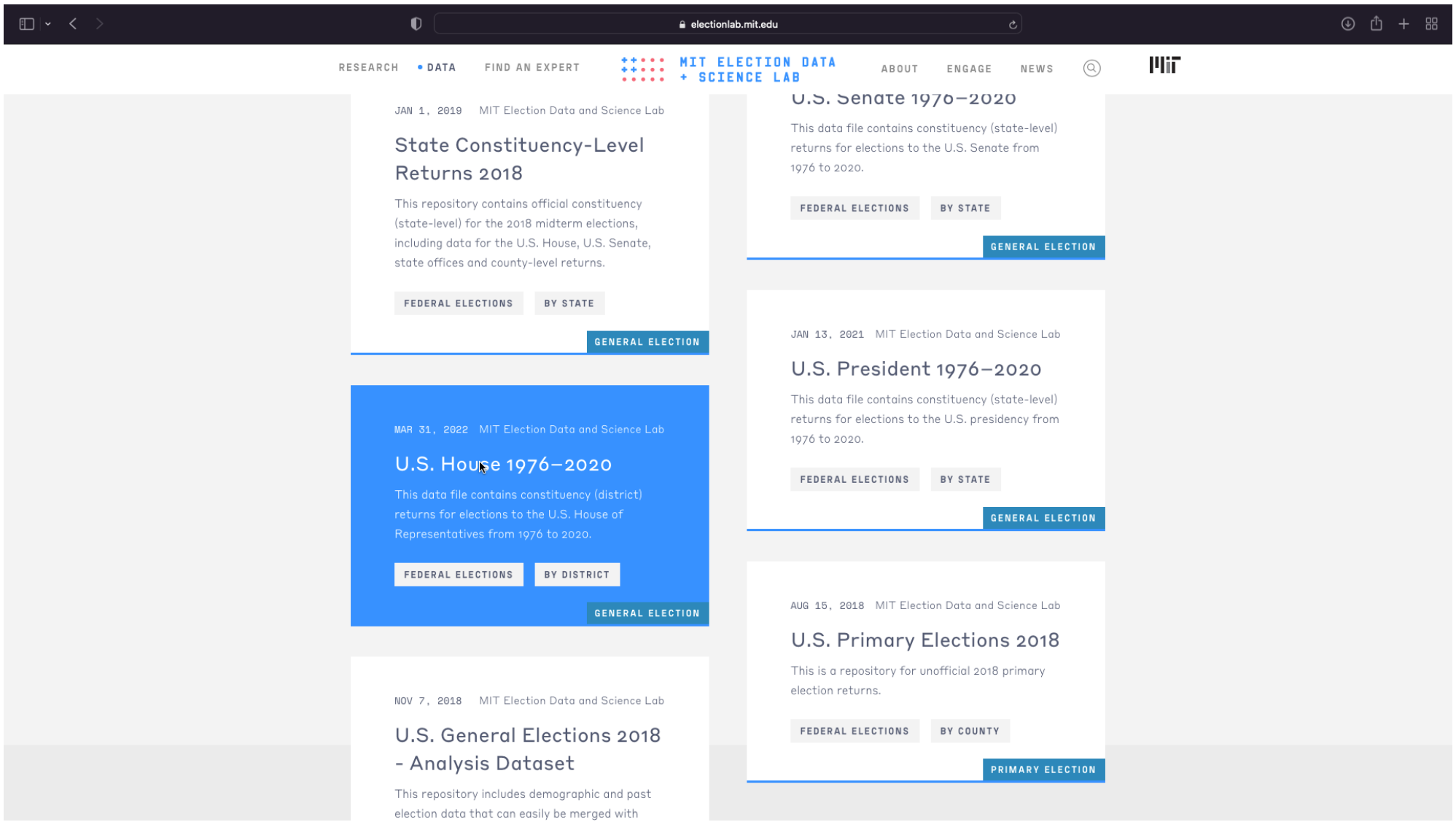
Downloading Data



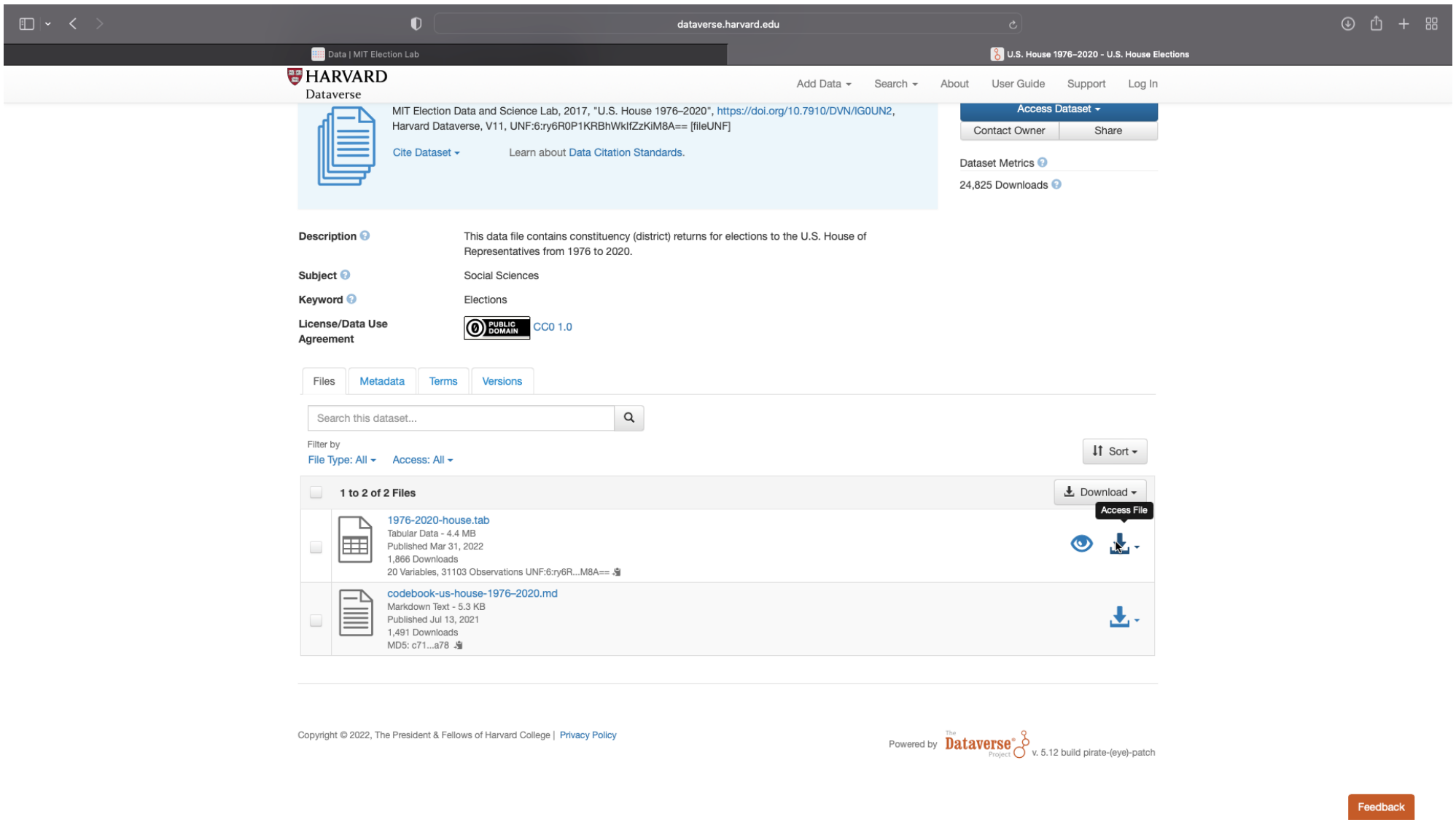
Downloading Data



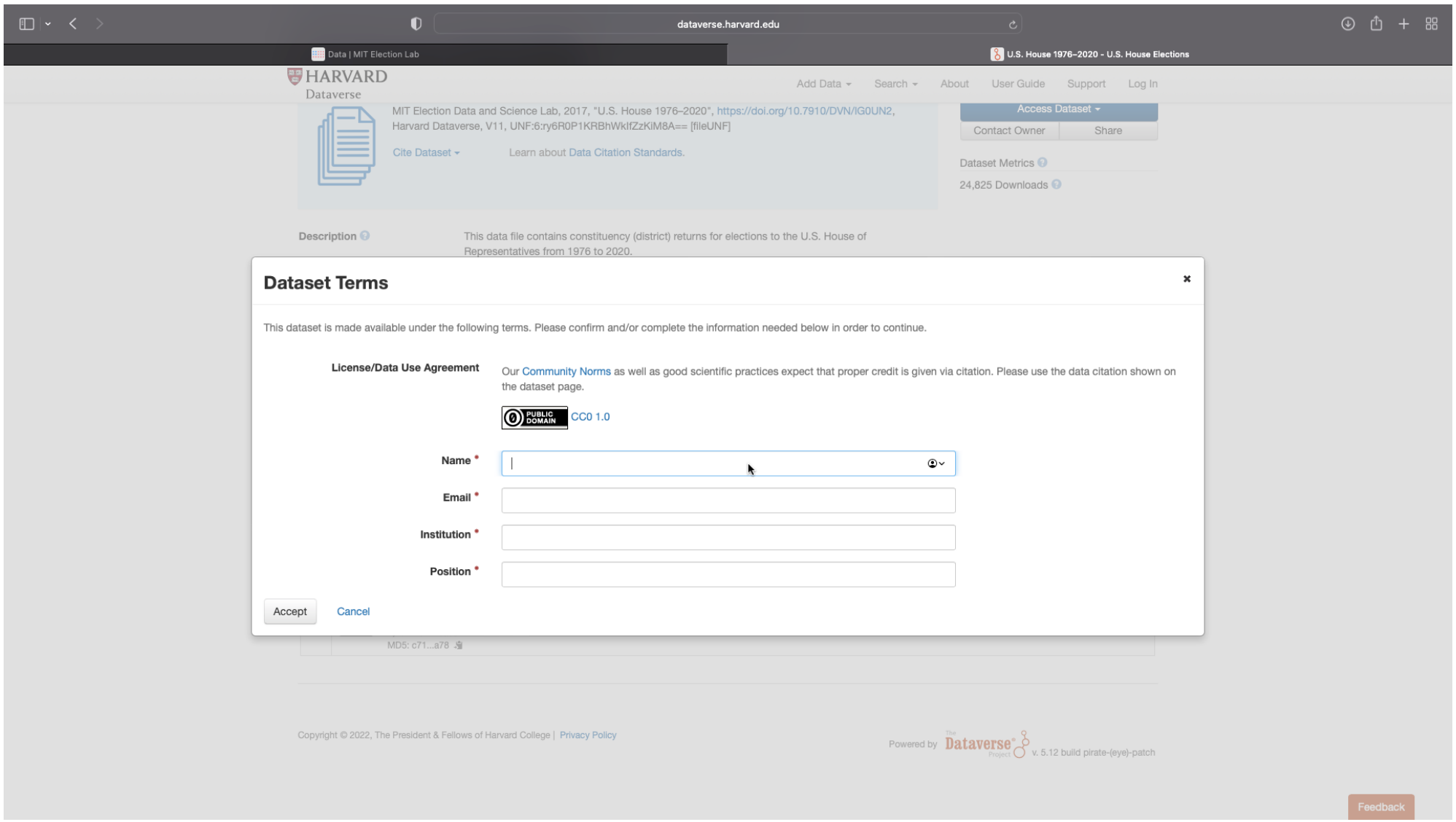
Downloading Data



Downloading Data



Downloading Data



Building a Dataset

1. Clean and Standardize Data
2. Merge Data
3. Check Merge
4. Fix Errors (if any)
5. Repeat until no errors remain

Cleaning Data

```r
nominate <- read.csv("H116_members.csv")
elections <- read.csv("1976-2020-house.csv")
```

```r
elections <- subset(elections, subset = year ==
    2018 & writein == FALSE & special ==
    FALSE & is.na(runoff) & stage == "GEN")
nominate <- subset(nominate, subset = chamber ==
    "House" & state_abbrev %in% state.abb)
```

Cleaning Data: Elections

```r
elections <- aggregate(candidatevotes ~ state_po +
    district + candidate + totalvotes, data = elections,
    FUN = sum)
elections$pct_vote <- (elections$candidatevotes/elections$totalvotes) *
    100
elections$candidate_lastname <- gsub(" $|\\'",
    "", gsub(" +", " ", gsub("JR$| I$| II$| III$| IV$",
        "", elections$candidate)))
elections$candidate_lastname <- unlist(lapply(elections$candidate_lastname,
    FUN = function(x) tail(unlist(strsplit(x,
        " ")), n = 1)))
```

Cleaning Data: Elections

What does this code do?

```r
elections <- aggregate(candidatevotes ~ state_po +
    district + candidate + totalvotes, data = elections,
    FUN = sum)
elections$pct_vote <- (elections$candidatevotes/elections$totalvotes) *
    100
elections$candidate_lastname <- gsub(" $|\\'",
    "", gsub(" +", " ", gsub("JR$| I$| II$| III$| IV$",
        "", elections$candidate)))
elections$candidate_lastname <- unlist(lapply(elections$candidate_lastname,
    FUN = function(x) tail(unlist(strsplit(x,
        " ")), n = 1)))
```

Cleaning Data: Elections

What does this code do?

```r
elections <- aggregate(candidatevotes ~ state_po +
    district + candidate + totalvotes, data = elections,
    FUN = sum)
elections$pct_vote <- (elections$candidatevotes/elections$totalvotes) *
    100
elections$candidate_lastname <- gsub(" $|\\'",
    "", gsub(" +", " ", gsub("JR$| I$| II$| III$| IV$",
        "", elections$candidate)))
elections$candidate_lastname <- unlist(lapply(elections$candidate_lastname,
    FUN = function(x) tail(unlist(strsplit(x,
        " ")), n = 1)))
```

Cleaning Data: Elections

```r
elections <- elections[, colnames(elections) %in%
    c("state_po", "district", "candidate_lastname",
        "pct_vote")]
elections$district <- ifelse(elections$district ==
    0, 1, elections$district)
colnames(elections)[1] <- "state_abbrev"
colnames(elections)[2] <- "district_code"
```

Cleaning Data: Elections

What does this code do?

```
elections <- elections[, colnames(elections) %in%
    c("state_po", "district", "candidate_lastname",
        "pct_vote")]
elections$district <- ifelse(elections$district ==
    0, 1, elections$district)
colnames(elections)[1] <- "state_abbrev"
colnames(elections)[2] <- "district_code"
```

Cleaning Data: Elections

What does this code do?

```r
elections <- elections[, colnames(elections) %in%
    c("state_po", "district", "candidate_lastname",
        "pct_vote")]
elections$district <- ifelse(elections$district ==
    0, 1, elections$district)
colnames(elections)[1] <- "state_abbrev"
colnames(elections)[2] <- "district_code"
```

Cleaning Data: Helpful Functions

```r
aggregate(y ~ g1 + g2 + g3..., data = data,
    FUN = ...)
```

- Groups y variable by g1, g2, g3,…
- data: Include your data
- FUN: apply this function to grouped y variable within each group

Cleaning Data: Helpful Functions

```r
x %in% y
```

- Checks whether x is in vector y

```r
1 %in% c(1, 2, 3)
```

```
[1] TRUE
```

```r
1 %in% c(2, 3, 4)
```

```
[1] FALSE
```

```r
"a" %in% c("a", "b", "c")
```

```
[1] TRUE
```

Cleaning Data: Helpful Functions

```r
ifelse(condition, x, y)
```

- If the condition is TRUE, then ifelse() returns x
- If the condition is FALSE, then ifelse() returns y

```r
ifelse(1 %in% c(1, 2, 3), "In Vector", "Not In Vector")
```

```
[1] "In Vector"
```

```r
ifelse(5 %in% c(1, 2, 3), "In Vector", "Not In Vector")
```

```
[1] "Not In Vector"
```

Cleaning Data: Helpful Functions

```r
lapply(x, FUN = ...)
```

- Applies function specified by FUN to every element of x
- Returns a list

```r
lapply(1:3, FUN = function(x) x * 2)
```

```
[[1]]
[1] 2

[[2]]
[1] 4

[[3]]
[1] 6
```

Cleaning Data: Helpful Functions

```r
unlist(x)
```

- Turns list into vector

```r
unlist(lapply(1:3, FUN = function(x) x *
    2))
```

```
[1] 2 4 6
```

Cleaning Data: Strings

```r
toupper(x)
tolower(x)
```

- Changes strings to all uppercase or all lowercase

```r
toupper("abc")
```

```
[1] "ABC"
```

```r
tolower("AbCD")
```

```
[1] "abcd"
```

Cleaning Data: Strings

```r
strsplit(x, split = s)
```

- Splits string at s into a vector within a list

```r
strsplit("BIDEN, Joseph R.", ", ")
```

```
[[1]]
[1] "BIDEN"     "Joseph R."
```

```r
unlist(strsplit("BIDEN, Joseph R.", ", "))
```

```
[1] "BIDEN"     "Joseph R."
```

Cleaning Data: Strings

```r
iconv(x, to = "ASCII//TRANSLIT")
```

- Converts text x to ASCII

```r
iconv("äé", to = "ASCII//TRANSLIT")
```

```
[1] "\"a'e"
```

Regular Expressions (Regex)

- Often working with texts that vary in content but have predictable formats
- Regex can help with this
- Regex matches patterns in text

```
grepl(pattern = patt, x = x)
```

- Is there a match for pattern patt in string x?

```
grepl(pattern = "A", x = "BCD")
```

```
[1] FALSE
```

```
grepl(pattern = "9", x = "2019")
```

```
[1] TRUE
```

Regular Expressions (Regex)

```r
gsub(pattern = patt, replacement = rep, x = x)
```

- Replaces occurrences of patt with rep in string x

```r
gsub(pattern = "A", replacement = "Z", x = "ABACAD")
```

```
[1] "ZBZCZD"
```

```r
gsub(pattern = "9", replacement = "X", x = "1999999999")
```

```
[1] "1XXXXXXXXX"
```

Regular Expressions (Regex)

- Brackets [] create groups of characters to be matched
  - “[A-Z]” matches all uppercase letters from A to Z
  - “[0-9]” matches all numbers
  - “[^0-9]” matches everything except numbers
  - Can customize content in brackets

```
gsub("[0-9]", "", "2019AL01")
```

```
[1] "AL"
```

Regular Expressions (Regex)

- How many matches do we want?
  - “s*” signifies 0 or more matches of s
  - “s+” signifies 1 or more matches of s
  - “s{num}” signifies num matches of s
  - “s{min,max}” signifies between min and max matches of s
- How could we get only state and district from the string “2019AL01”?

```
gsub(pattern, "", "2019AL01")
```

```
gsub("[0-9]{4}", "", "2019AL01")
```

```
[1] "AL01"
```

Regular Expressions (Regex)

- "^[A-Z]" matches only at the start of the string

```
grepl("^[A-Z]", "2019AL01")
```

```
[1] FALSE
```

- "[A-Z]$" matches only at the end of the string

```
grepl("[A-Z]$", "2019AL01")
```

```
[1] FALSE
```

```
grepl("[A-Z]$", "AL201901A")
```

```
[1] TRUE
```

Regular Expressions (Regex)

- Matching special characters
  - Need to escape with two “\”

```
grepl("\\?", "How's it going?")
```

```
[1] TRUE
```

Regular Expressions (Regex)

- Great reference at this link

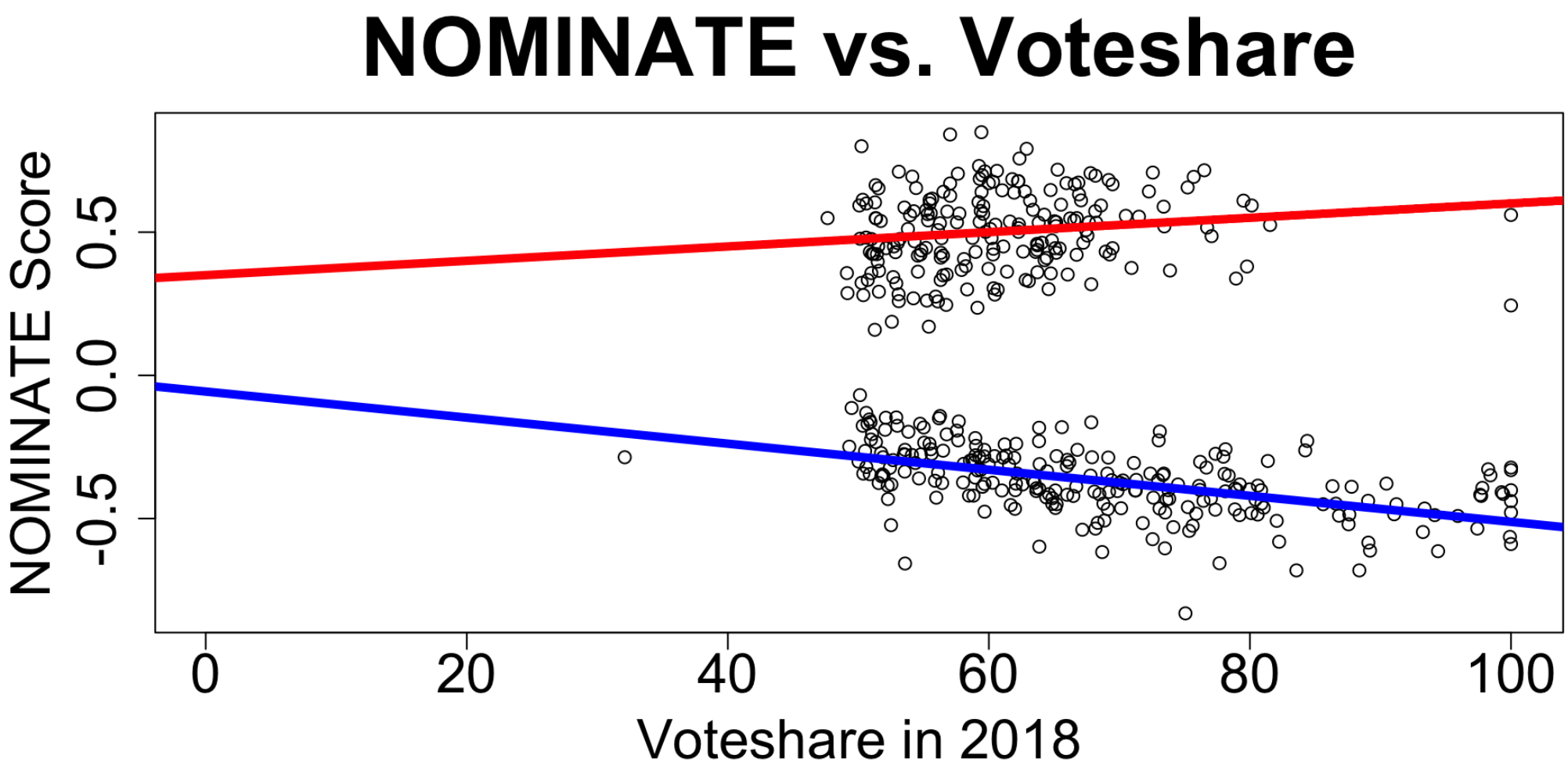
Cleaning Data: NOMINATE

```r
nominate <- nominate[, colnames(nominate) %in%
    c("district_code", "state_abbrev", "party_code",
        "nominate_dim1", "bioname")]
nominate$candidate_lastname <- unlist(lapply(nominate$bioname,
    function(x) head(unlist(strsplit(x, ",")),
        n = 1)))
nominate$candidate_lastname <- toupper(nominate$candidate_lastname)
nominate$candidate_lastname <- gsub("\\'",
    "", iconv(nominate$candidate_lastname,
        to = "ASCII//TRANSLIT"))
nominate$candidate_lastname <- unlist(lapply(nominate$candidate_lastname,
    function(x) tail(unlist(strsplit(x, " ")),
        n = 1)))
nominate$party <- ifelse(nominate$party_code ==
    200, "REPUBLICAN", ifelse(nominate$party_code ==
    100, "DEMOCRAT", "INDEPENDENT"))
nominate <- subset(nominate, subset = party !=
    "INDEPENDENT")
```

Merging Data

```r
full <- merge(elections, nominate, by = c("district_code",
    "state_abbrev", "candidate_lastname"),
    all = TRUE)
plot(full$pct_vote, full$nominate_dim1, xlab = "Voteshare in 2018",
    ylab = "NOMINATE Score", main = "NOMINATE vs. Voteshare")
fit <- lm(nominate_dim1 ~ pct_vote * party,
    data = full)
abline(a = coef(fit)[1], b = coef(fit)[2],
    col = "blue", lwd = 5)
abline(a = coef(fit)[1] + coef(fit)[3], b = coef(fit)[2] +
    coef(fit)[4], col = "red", lwd = 5)
```

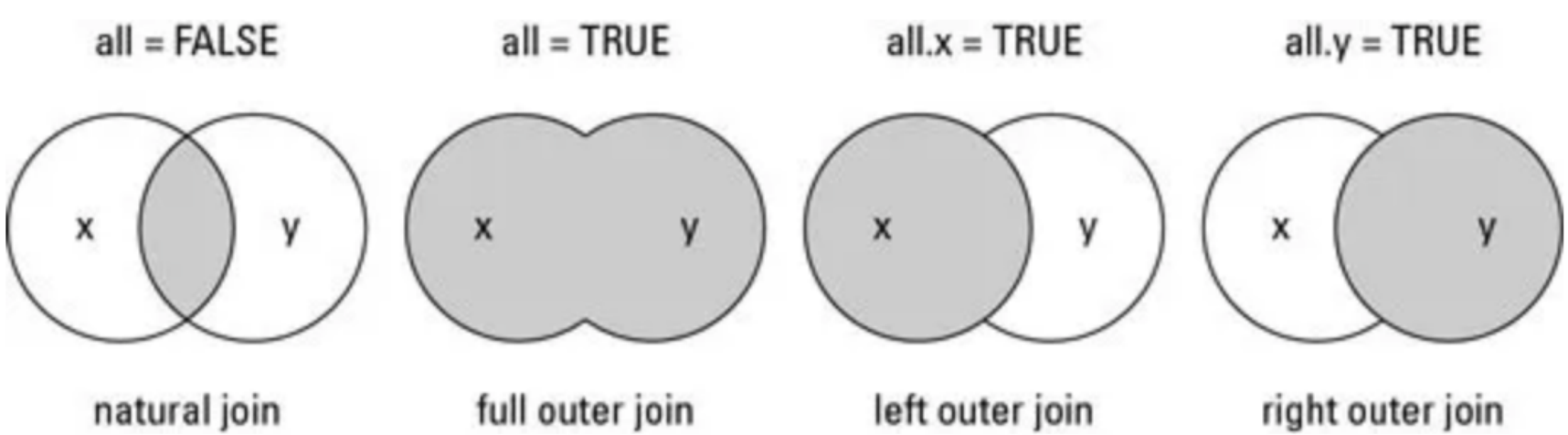
Merging Data



Merging Data

```
merge(x, y, by = c(colname1, colname2, ...),
      all = , all.x = , all.y = )
```

- Merge dataframe x with dataframe y by matches in specified columns



Source

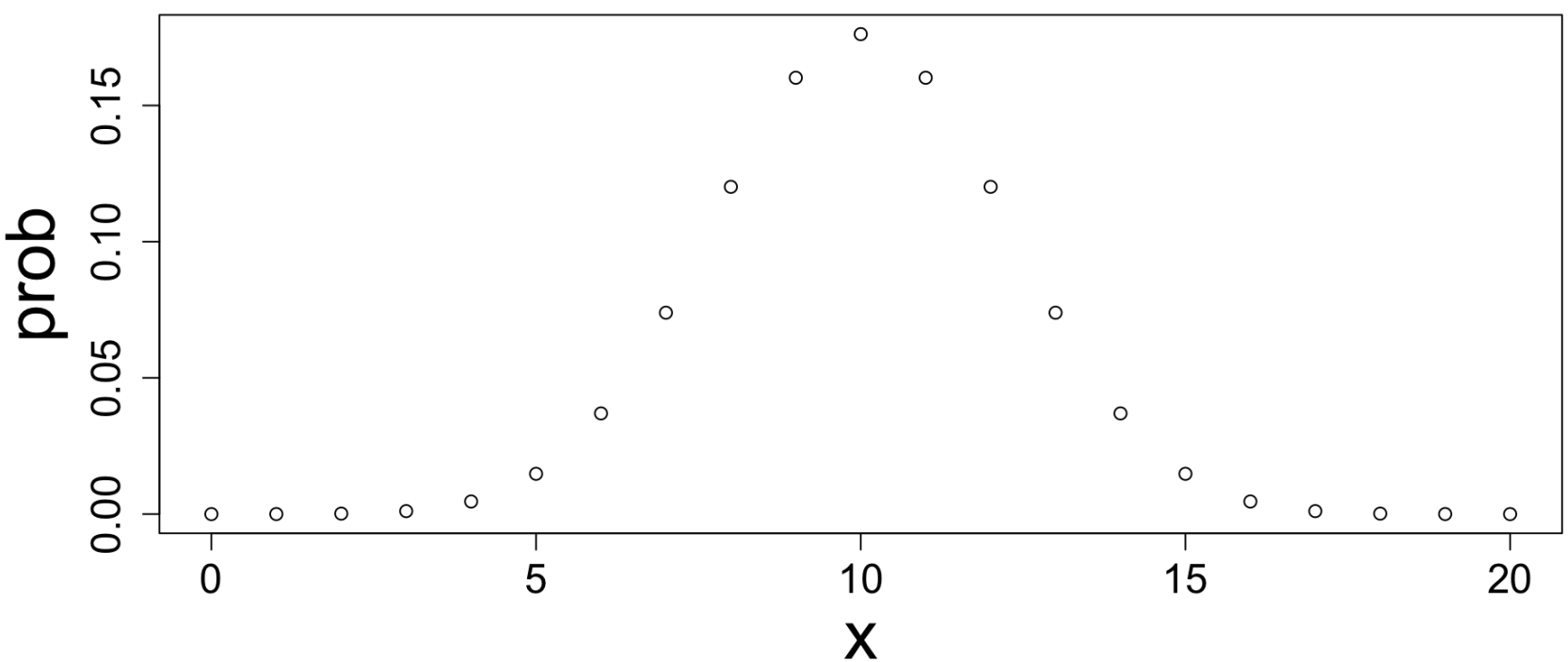
Probability: Binomial Distribution

Probability Mass Function

$$
\begin{aligned}
Pr(X = x) &= \binom{n}{x} p^x (1-p)^{n-x} \\
&= \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x}
\end{aligned}
$$

Probability: Binomial Distribution

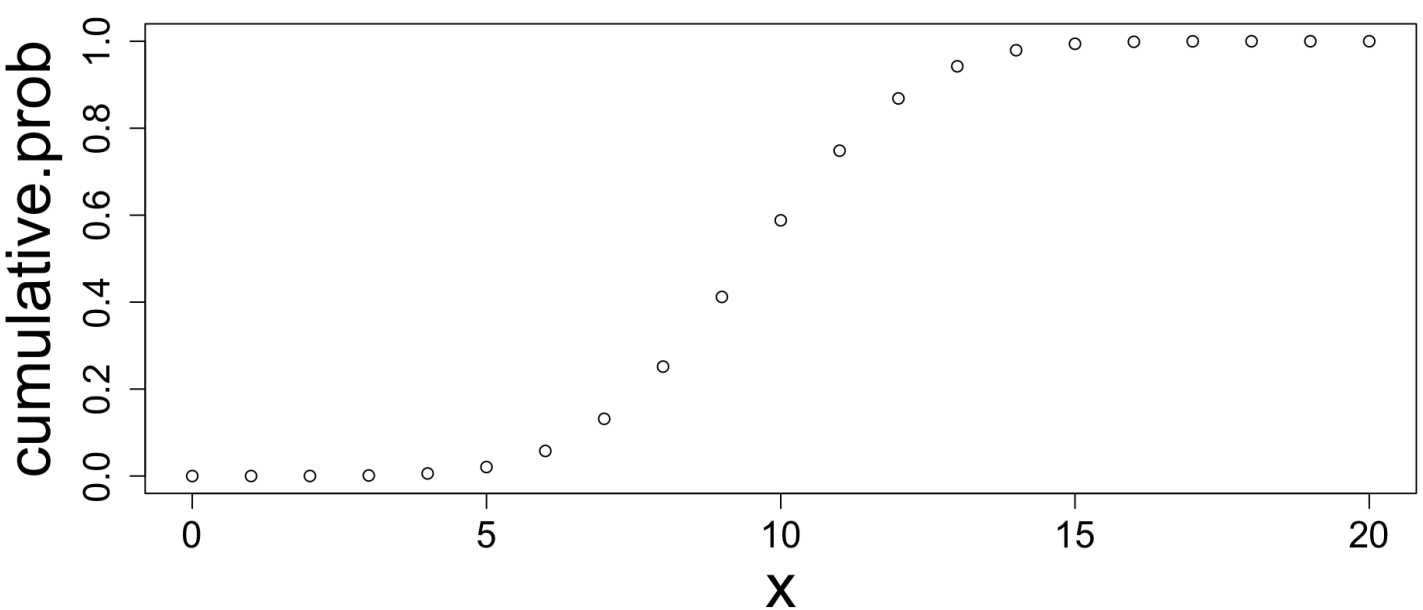
```r
n <- 20
x <- seq(0, 20, 1)
prob <- dbinom(x, size = n, prob = 0.5)
plot(x, prob)
```



Probability: Binomial Distribution

Cumulative Distribution Function

```r
n <- 20
x <- seq(0, 20, 1)
cumulative.prob <- pbinom(x, n, prob = 0.5)
plot(x, cumulative.prob)
```

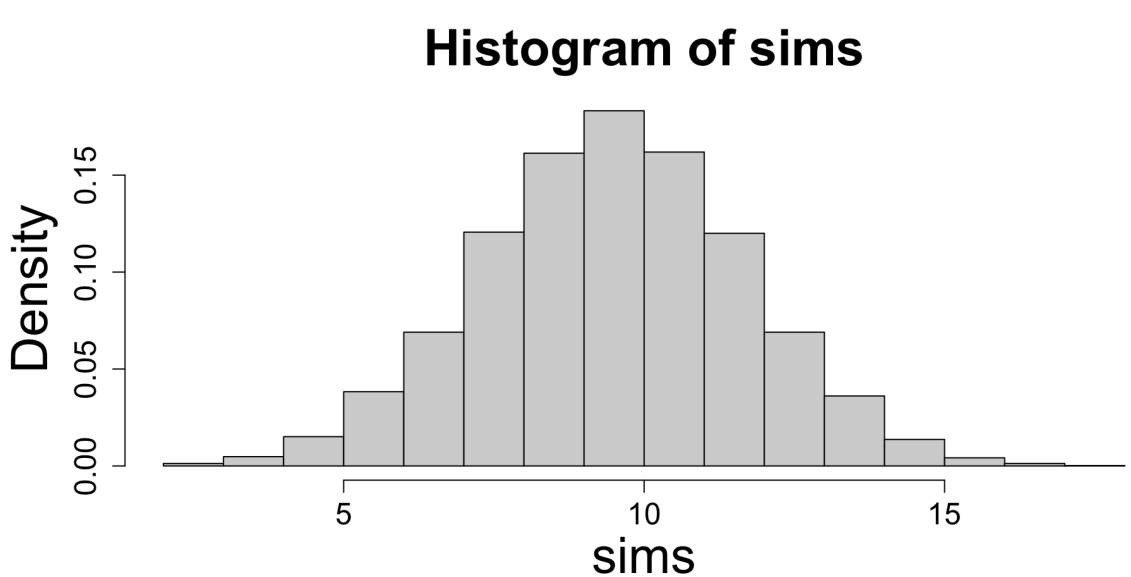


Probability: Binomial Distribution

Random Number Generation

- Simulate random numbers from distribution

```r
set.seed(123)
sims <- rbinom(10000, 20, 0.5)
hist(sims, freq = F)
```



Probability

- Why simulate random numbers?
  - Challenging computations
  - Fake data with known distribution to check our intuition