# POLI 30 D: Political Inquiry

## TA Sessions

## Lab 03 | R Basics II

# Before we start

**Announcements:**

- Github page: **https://github.com/umbertomig/POLI30Dpublic**
- Piazza forum: https://piazza.com/ucsd/winter2023/17221

# Before we start

**Recap:** In the Lab sessions you learned:

- How to install R and R Studio in your computer.
- How to do basic math operations in R.

**Great job!**

- Do you have any questions about these contents?

# Plan for Lab 03

- Learn R data types
- Learn how to create datasets
- Learn how to load a dataset
- Learn how to explore a dataset

# R Data Types

# R Numeric and R Character

You can create numeric and character variables in R easily:

```
# Numeric
x <- 20
# Character (or string, same thing!)
y <- 'POLI 30 D'
print(x)
```

```
## [1] 20
```

```
print(y)
```

```
## [1] "POLI 30 D"
```

# R Numeric and R Character

We can easily check the types with the `class(.)` function:

```
class(x)
```

```
## [1] "numeric"
```

```
class(y)
```

```
## [1] "character"
```

# R Numeric and R Character

The **str(.)** function also gives a neat description of what is going on.

- `str` is short for structure of the data:

```
str(x)
```

```
##  num 20
```

```
str(y)
```

```
##  chr "POLI 30 D"
```

# R Vectors

To create a vector, you use the function `c(.)`, and separate the values with a comma:

```
voted   <- c(1, 0, 0, 1) # Binary vector (1=yes; 0=no)
age     <- c(48, 23, 18, 33) # Numeric vector with ages
message <- c('yes', 'yes', 'no', 'no') # Got message?
place   <- c('La Jolla', 'Del Mar',
             'Del Mar', 'Poway') # Character with places
```

Each of the variables have four positions (like four observations).

# R Vectors

We can check the data for each of the four observations we have. We use square brackets ([and ]) to index.

First person in the `voted` variable:

```
voted[1]
```

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

Ages of first three people. Note `1:3` (try it on the console!). It creates sequences.

```
age[1:3]
```

```
## [1] 48 23 18
```

# R Vectors

Messages for all people but the second person:

```
message[-2]
```

```
## [1] "yes" "no"  "no"
```

Checking all the places vector:

```
place
```

```
## [1] "La Jolla" "Del Mar"  "Del Mar"  "Poway"
```

Changing the age of the second person:

```
age[2] <- 24
age
```

```
## [1] 48 24 18 33
```

# R Vectors

You can also create a numeric vector from a character vector. You should use the function **ifelse(.)**:

```
message
```

```
## [1] "yes" "yes" "no"  "no"
```

```
message_num <-
  ifelse(message == 'yes', # Check if message is yes
         1, # change to 1 if yes
         0) # change to 0 if no
message_num
```

```
## [1] 1 1 0 0
```

# R Vectors

You can also check the length of the vector:

```
length(message)
```

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

And if you don't need something anymore, you can remove it:

```
rm(message_num)
```

Now check the environment. `message_num` should have disappeared!

# R `data.frame`

# Creating a `data.frame` from scratch

To create a **`data.frame(.)`** from scratch, you can just add variables inside it:

```
dat <- data.frame(
  v1 = c(1,2,3),
  v2 = c('a', 'b', 'c'),
  v3 = c('Treatment', 'Control', 'Control')
)
dat
```

```
##   v1 v2        v3
## 1  1  a Treatment
## 2  2  b   Control
## 3  3  c   Control
```

# Creating a `data.frame` from scratch

**Your turn**: Create a data frame with the following info and call it `dat2`:

| | A | B | C | D |
|---|---|---|---|---|
| 1 | age | college | voted | work |
| 2 | 23 | Yes | 1 | FT |
| 3 | 33 | No | 0 | FT |
| 4 | 67 | No | 1 | PT |
| 5 | 81 | Yes | 1 | RT |
| 6 | 18 | No | 0 | UN |
| 7 | | | | |

# Creating a `data.frame` from scratch

If you are curious, these are the meanings:

| Variable | Meaning |
|---|---|
| age | Age in years |
| college | Yes means college complete |
| voted | 1 means voted |
| work | FT means full-time worker; PT means partial-time worker; UN means unemployed; RT means retired |

# Creating a `data.frame` from existing variables

If you recall, we created the following variables: `voted`, age, `message`, and `place`.

Here is how to create a data.frame with them:

```
dat3 <- data.frame(voted, age, message, place)
dat3
```

```
##   voted age message    place
## 1     1  48     yes La Jolla
## 2     0  24     yes  Del Mar
## 3     0  18      no  Del Mar
## 4     1  33      no    Poway
```

# Loading CSV data in R

# Loading a CSV dataset from locale

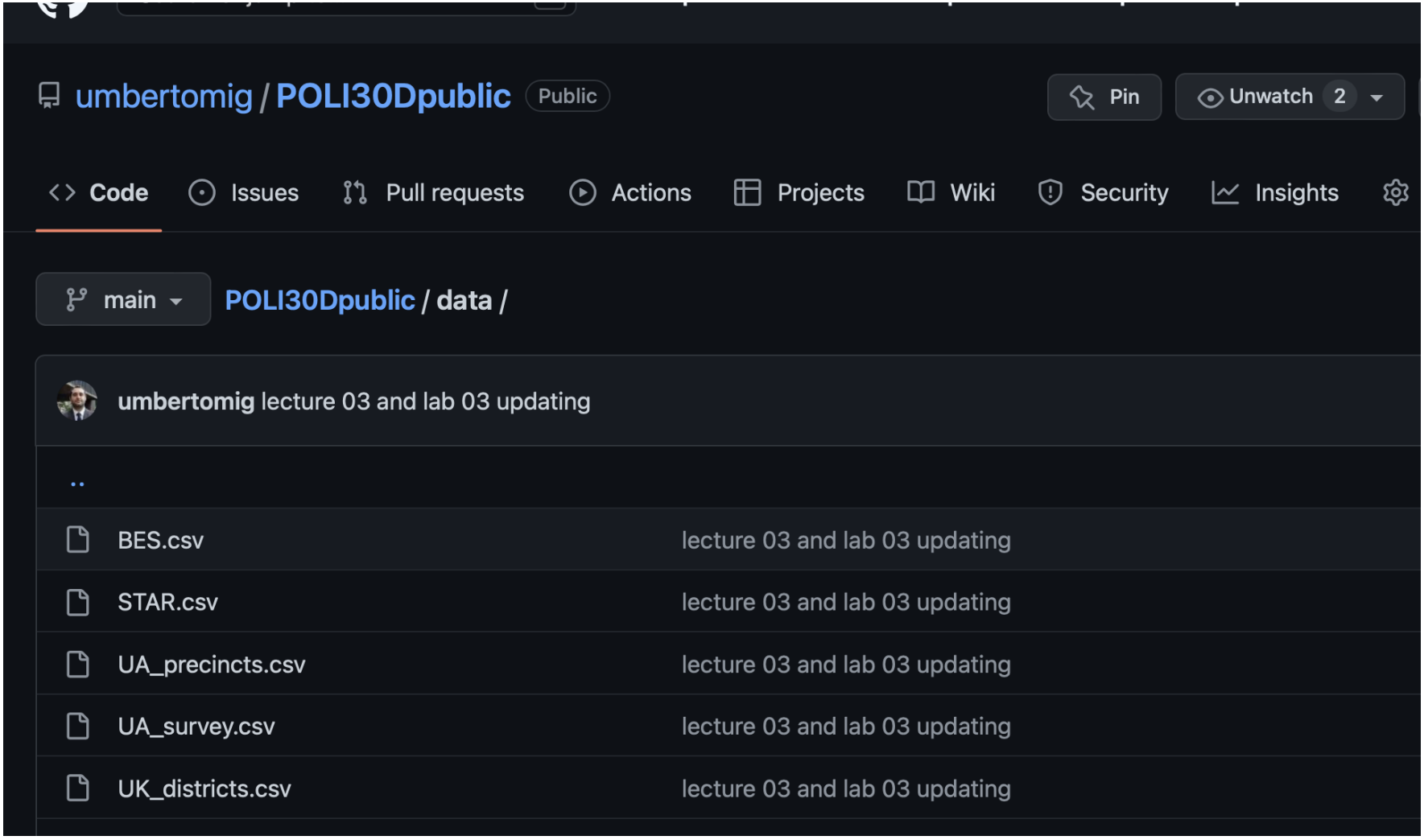
- CSV stands for Comma Separated Values. It is a special way to organize data:
  - Each line corresponds to one observation
  - Within lines, information for each variable is separated by a comma.
- To load data from the locale (i.e., your computer), you need to find and change your working directory.
- The book explains more on that. We are not going to deal with these cases in here.
- We will frequently analyze data from **GitHub**.

# Loading a CSV dataset from GitHub

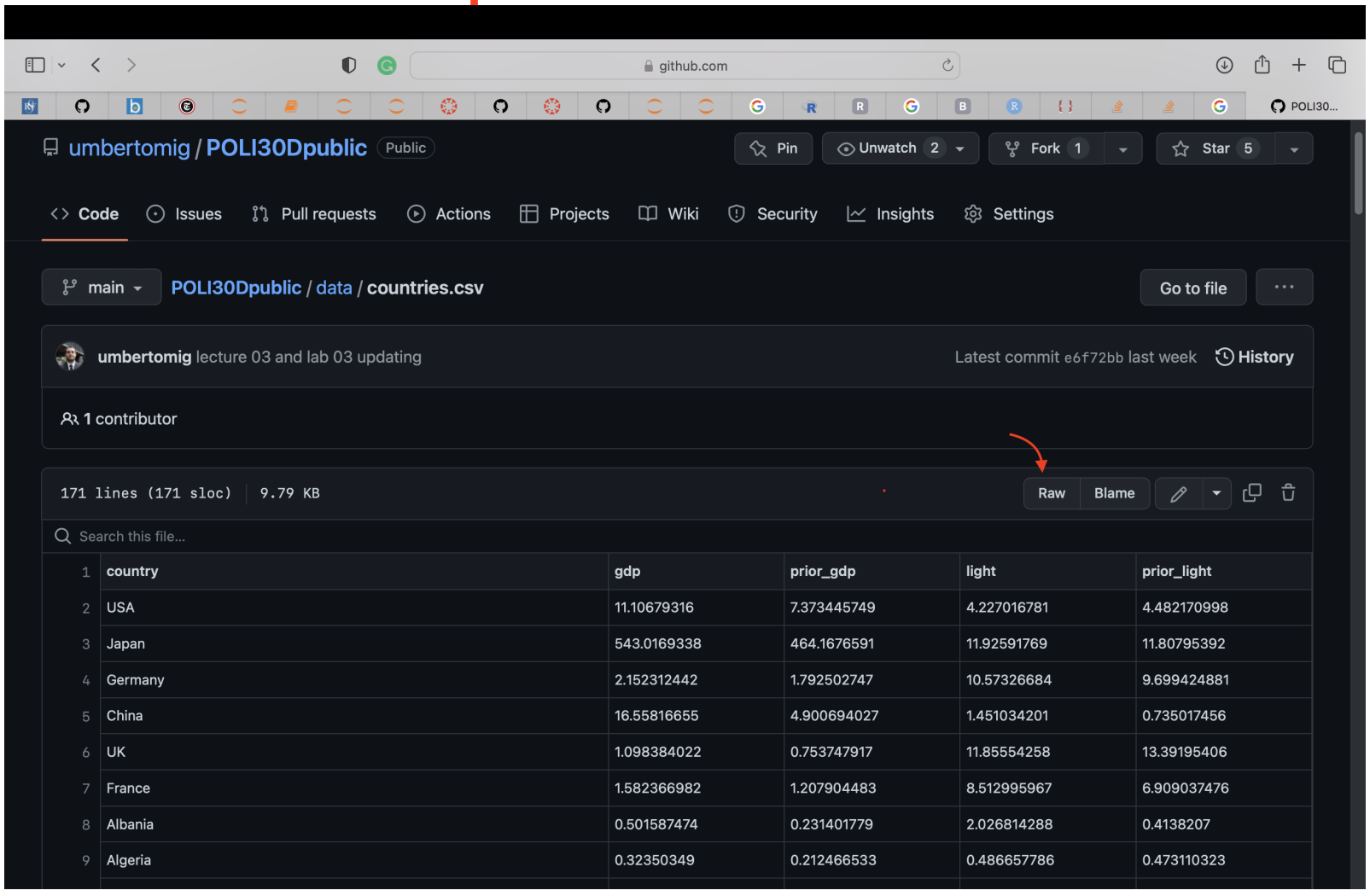
To load a CSV dataset from the GitHub:

1. Open the GitHub
2. Go to the GitHub data folder
3. Select the Dataset you want to open (in the case, `countries.csv`)
4. Find the *Raw* Version of it
5. Copy the URL (CMD + C / Ctrl + C). In the case of countries: https://raw.githubusercontent.com/umbertomig/POLI30Dpublic/main/data/countries.csv
6. Do `name_data_frame <- read.csv('paste_URL_in_here')`

# Example: `countries.csv`

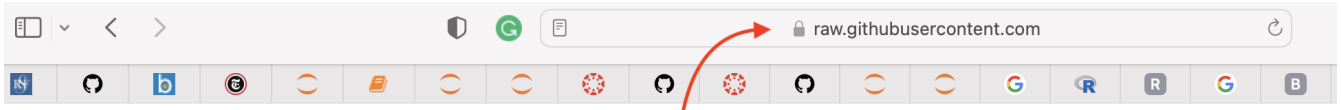


# Example: countries.csv

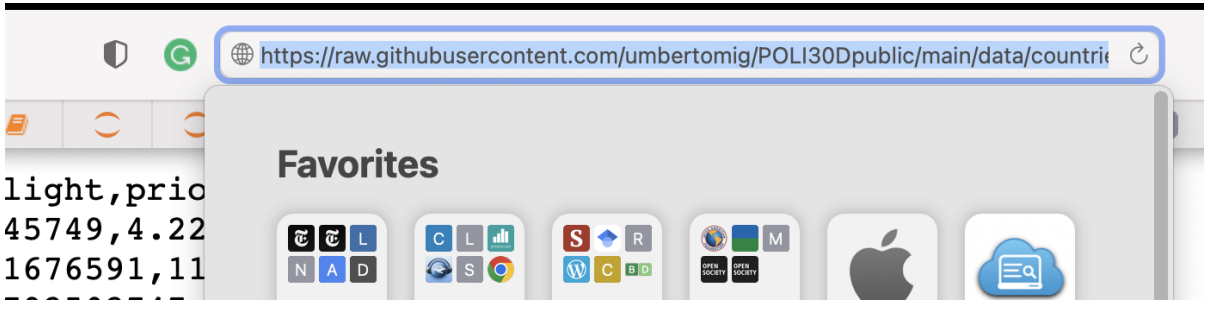



| | country | gdp | prior_gdp | light | prior_light |
|---|---|---|---|---|---|
| 2 | USA | 11.10679316 | 7.373445749 | 4.227016781 | 4.482170998 |
| 3 | Japan | 543.0169338 | 464.1676591 | 11.92591769 | 11.80795392 |
| 4 | Germany | 2.152312442 | 1.792502747 | 10.57326684 | 9.699424881 |
| 5 | China | 16.55816655 | 4.900694027 | 1.451034201 | 0.735017456 |
| 6 | UK | 1.098384022 | 0.753747917 | 11.85554258 | 13.39195406 |
| 7 | France | 1.582366982 | 1.207904483 | 8.512995967 | 6.909037476 |
| 8 | Albania | 0.501587474 | 0.231401779 | 2.026814288 | 0.4138207 |
| 9 | Algeria | 0.32350349 | 0.212466533 | 0.486657786 | 0.473110323 |

# Example: countries.csv




```
country,gdp,prior_gdp,light,prior_light
USA,11.10679316,7.373445749,4.227016781,4.482170998
Japan,543.0169338,464.1676591,11.92591769,11.80795392
Germany,2.152312442,1.792502747,10.57326684,9.699424881
China,16.55816655,4.900694027,1.451034201,0.735017456
UK,1.098384022,0.753747917,11.85554258,13.39195406
France,1.582366982,1.207904483,8.512995967,6.909037476
Albania,0.501587474,0.231401779,2.026814288,0.4138207
Algeria,0.32350349,0.212466533,0.486657786,0.473110323
Angola,0.003544499,0.001472475,0.05414023,0.032691038
Antigua and Barbuda,0.001905568,0.001116855,11.89651952,8.774662651
Azerbaijan,0.011544727,0.006051771,2.029455169,2.175190872
Argentina,0.31740252,0.229815244,0.641698249,0.420320367
Australia,0.997419262,0.61571937,0.158016218,0.142925014
Austria,0.229786753,0.173707733,6.780558816,4.731001868
```



# Example: countries.csv

And since the link is:
https://raw.githubusercontent.com/umbertomig/POLI30Dpub

```
countries <- read.csv('https://raw.githubusercontent.com/um
head(countries)
```

```
##   country         gdp   prior_gdp     light prior_light
## 1     USA  11.106793   7.3734457  4.227017   4.4821710
## 2   Japan 543.016934 464.1676591 11.925918  11.8079539
## 3 Germany   2.152312   1.7925027 10.573267   9.6994249
## 4   China  16.558167   4.9006940  1.451034   0.7350175
## 5      UK   1.098384   0.7537479 11.855543  13.3919541
## 6  France   1.582367   1.2079045  8.512996   6.9090375
```

# Exploring a dataset

# head and tail

`head(.)` shows the first six observations:

```
head(countries)
```

```
##   country         gdp   prior_gdp      light prior_light
## 1     USA  11.106793   7.3734457  4.227017   4.4821710
## 2   Japan 543.016934 464.1676591 11.925918  11.8079539
## 3 Germany   2.152312   1.7925027 10.573267   9.6994249
## 4   China  16.558167   4.9006940  1.451034   0.7350175
## 5      UK   1.098384   0.7537479 11.855543  13.3919541
## 6  France   1.582367   1.2079045  8.512996   6.9090375
```

# head and tail

`tail(.)` shows the last six observations:

```
tail(countries)
```

```
##         country          gdp   prior_gdp     light prior_light
## 165     Uruguay 0.434783819 0.341299579 0.5807991  0.48451899
## 166 Uzbekistan 1.486802454 0.983226379 1.6353821  1.83231197
## 167   Venezuela 0.048871355 0.038868261 1.4243972  1.21489199
## 168       Samoa 0.001036411 0.000620675 0.5527025  0.38445127
## 169       Yemen 0.270938873 0.142525797 0.6498414  0.34853987
## 170      Zambia 3.252155015 2.372258356 0.0968198  0.08652917
```

# str

Nice and neat description of the data.frame. You can try with other objects too!

```
str(countries)
```

```
## 'data.frame':    170 obs. of  5 variables:
##  $ country    : chr  "USA" "Japan" "Germany" "China" ...
##  $ gdp        : num  11.11 543.02 2.15 16.56 1.1 ...
##  $ prior_gdp  : num  7.373 464.168 1.793 4.901 0.754 ...
##  $ light      : num  4.23 11.93 10.57 1.45 11.86 ...
##  $ prior_light: num  4.482 11.808 9.699 0.735 13.392 ...
```

## summary, View, and dim

- `dim(.)` gives you the dimension of the data.frame.
  - The first number is the number of rows (observations).
  - The second number is the number of columns (variables).

```
dim(countries)
```

```
## [1] 170   5
```

- The `View(.)` function is also useful. In your console, type the following:

```
View(countries)
```

- Very cool, right?!

# summary, View, and dim

And for a quick summary of the variables in your dataset, you can use the function **summary(.)**:

```
summary(countries)
```

```
##    country                gdp               prior_gdp            light
##  Length:170         Min.   :   0.0000   Min.   :   0.0000   Min.   : 0.00494
##  Class :character   1st Qu.:   0.0117   1st Qu.:   0.0086   1st Qu.: 0.19885
##  Mode  :character   Median :   0.2588   Median :   0.1419   Median : 1.24459
##                     Mean   :  27.5893   Mean   :  16.6594   Mean   : 3.56543
##                     3rd Qu.:   1.5106   3rd Qu.:   0.9746   3rd Qu.: 3.64854
##                     Max.   :1798.3271   Max.   :1111.8674   Max.   :45.69160
##   prior_light
##  Min.   : 0.00299
##  1st Qu.: 0.16311
##  Median : 0.89487
##  Mean   : 2.95033
##  3rd Qu.: 3.37282
##  Max.   :44.18754
```

## Today's Lab

- Learn R data types
- Learn how to create datasets
- Learn how to load a dataset
- Learn how to explore a dataset

## Next Lab

- How to operate with data.frames
- Learn how to use R Markdown

Questions?

See you in the next lab!