Using Python & R in Harmony

PyOhio 2019

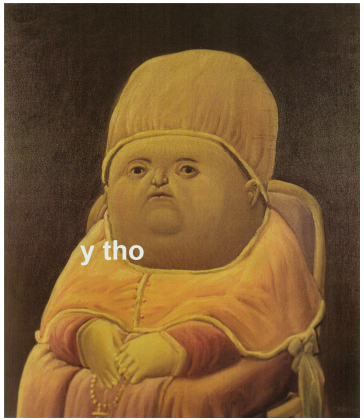
Krista Readout & Matthew Brower

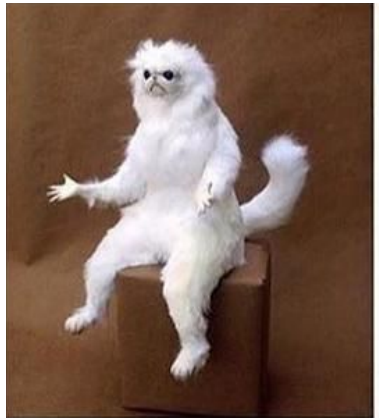
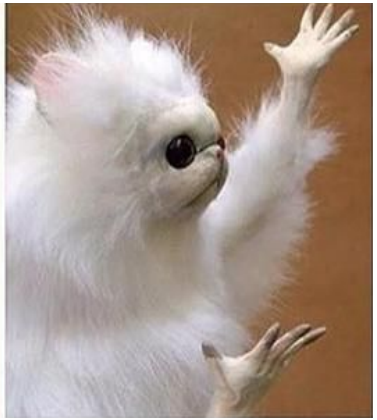
Goals for today’s 30-minute talk:

Build an awareness of rpy2 & reticulate and when they’re useful

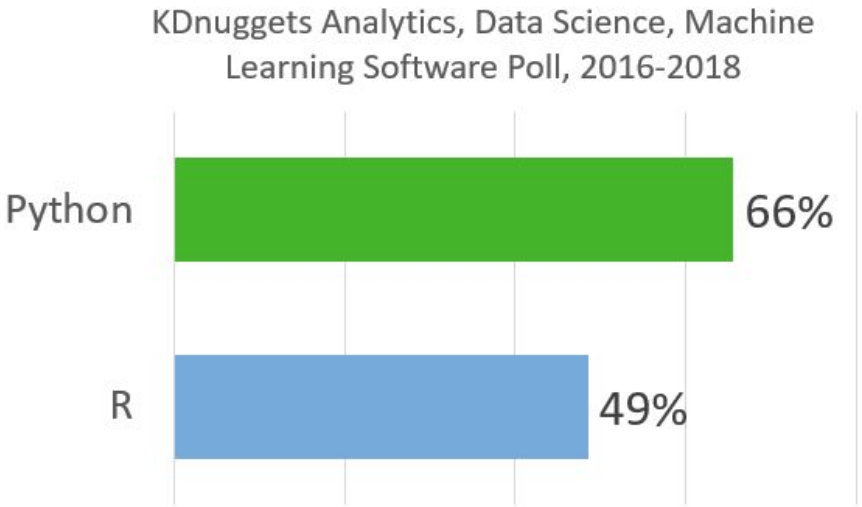
Understand some basic operations & syntax of each library

Review some applied examples & put this knowledge to work


y tho

???
???



Just like hard and soft tacos



Why would cross-pollinating languages be useful?

A package or module you want to use is exclusive to one language

You have an existing script & want to leverage that instead of reinventing the wheel

You can simply handle some workflows more efficiently with the other language

Img credit: https://twitter.com/rstudio/status/978293244390985728

Event: PyOhio 2019
Talk: Using Python & R in Harmony
Date: July 27th, 2019

Code + Slides

bit.ly/2LIzhbC

Using Python from within R with `reticulate`

What is reticulate?

The **reticulate** package provides a comprehensive set of tools for interoperability between Python and R.

The package includes facilities for:

- **Calling Python from R** in a variety of ways including **R Markdown**, **sourcing Python scripts**, **importing Python modules**, and using Python interactively within an R session.
- **Translation between R and Python objects** (for example, between R and Pandas data frames, or between R matrices and NumPy arrays).
- Flexible **binding to different versions of Python** including **virtual environments** and **Conda environments.**

TLDR: Methods for importing Python modules directly to R, converting objects back & forth between Python and R, tools for managing Python environments

Source: https://rstudio.github.io/reticulate/

A brief guide to using reticulate within R

Create & use Python virtual environments

```
virtualenv_create('your_env')
use_virtualenv('your_env')
```

Use raw strings as Python

```
py_eval('print("hello")')
```

Convert R objects to Python objects & vice versa

```
py_to_r([3,4,5])
r_to_py(c(3,4,5))
```

Install Python packages

```
py_install(c('pandas',
             'requests'),
           envname='your_env'))
```

Write Python chunks in RMD

````
```{python}
print('hello')
```
````

Call Python functions / scripts & supply R objects as inputs

```
py_run_file('cool_script.py')

requests = import('requests')
url = 'www.google.com'
requests$get(url)
```

Official RStudio reticulate cheat sheet

Spinning up a Python virtualenv within R

Python

```
$ mkvirtualenv r_py3_env --python=python3
```

R / Reticulate

```
virtualenv_create(
          'r_py3_venv',
          python='/usr/local/bin/python3'
          )

use_virtualenv('r_py3_venv')
```

!

You can also use existing environments or use `conda` instead of virtualenv

Installing Python modules to virtualenvs from R

Python

```
workon r_py3_env

pip install pandas
pip install requests
```

R / Reticulate

```
py_install(
      c('pandas',
      'requests',
      'pandas.io'),
      envname = 'r_py3_venv'
      )
```

!

reticulate will establish its own virtualenv named `r-reticulate` if one isn't specified upon installation

Importing Python modules within R

Python

```
import pandas
import requests
import pandas.io.json
```

R / Reticulate

```
pandas = import('pandas')
requests = import('requests')
pandas_io = import('pandas.io.json')
```

Leveraging Python module functions

Python

```
pandas.pivot()

requests.get()
```

R / Reticulate

```
pandas$pivot()

requests$get()
```

An example with a web API & SF Bay climates

"If you don't like the weather in ~~New England~~ [Ohio] now, just wait five minutes."

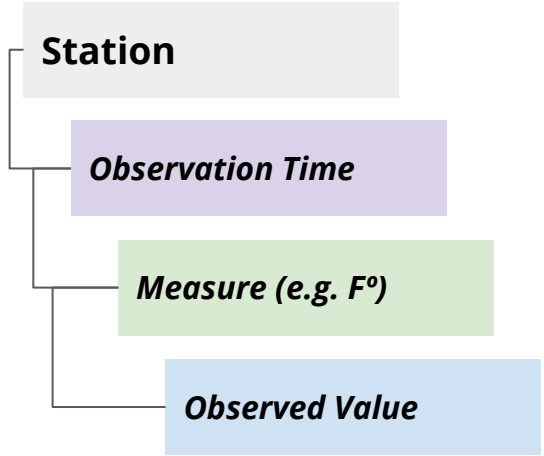
Mark Twain

Calling the weather.gov web API

https://api.weather.gov/stations/**{station_id}**/observations

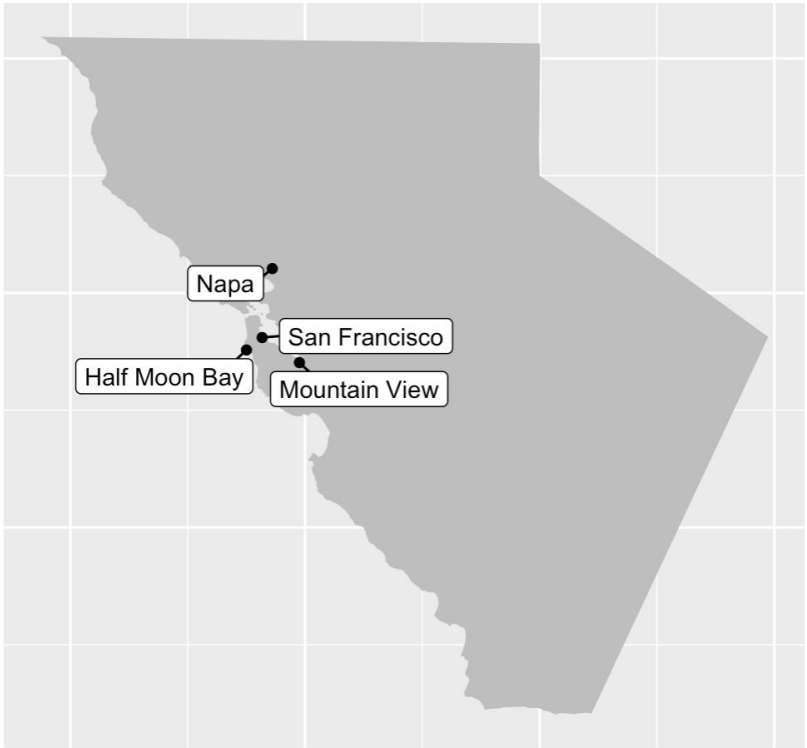
```
curl -s https://api.weather.gov/stations/KSFO/observations | tail -n 75
                    "qualityControl": "qc:V"
                },
                "windGust": {
                    "value": null,
                    "unitCode": "unit:m_s-1",
                    "qualityControl": "qc:Z"
                },
                "barometricPressure": {
                    "value": 101390,
                    "unitCode": "unit:Pa",
                    "qualityControl": "qc:V"
                },
                "seaLevelPressure": {
                    "value": 101390,
                    "unitCode": "unit:Pa",
                    "qualityControl": "qc:V"
                },
```

JSON Response



Some SF Bay Area stations...

Station Name	Station Identifier	Latitude	Longitude
Half Moon Bay	KHAF	37.51360	-122.4996
Mountain View	KNUQ	37.40583	-122.0481
Napa	KAPC	38.20750	-122.2794
San Francisco	KSFO	37.61961	-122.3656



Gathering JSON responses with help from Python

```
temperature_data = data.frame(NULL)

for (i in (1:nrow(station_list))){ ## begin parent loop

  ## create the URL we'll use to make the API request
  url = paste('https://api.weather.gov/stations/',
              station_list$station_identifier[i],
              '/observations',
              sep='')

  page = requests$get(url) ## use requests library to fetch the URL from the web

  content = page$content ## isolate to on-page content only

  ## use pandas.io to extract and normalize the JSON content from the request
  content_pandas_io = pandas_io$loads(content)


  [...] continued on next slide
```

1

2

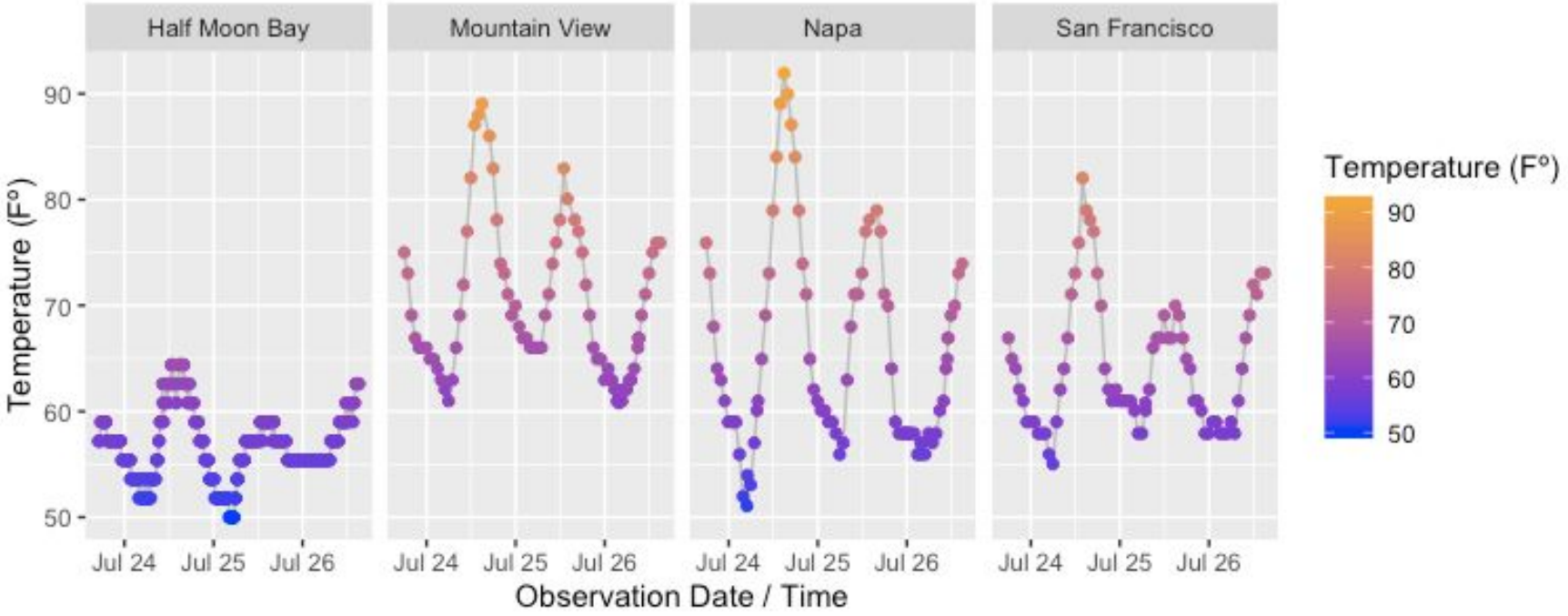
3

Wrangling & stacking observations

1

```
## Now for every leaf j in this normalized JSON, we need to extract the time
of observation observation was recorded and corresponding temperature

for (j in seq_along(content_pandas_io$features)){   ## begin child loop

  record = data.frame(
   date = content_pandas_io$features[[j]]$properties$timestamp[1],
   temperature=content_pandas_io$features[[j]]$properties$temperature[1]$value,
   station_name = station_list$name[i],
   station_id = station
   )

  ## Append this new row to a dataframe containing the final results
  temperature_data = bind_rows(record,temperature_data)

} ## close child loop

} ## close parent loop
```

Recent Bay Area temperatures, visualized!



Using R from within Python with `rpy2`

What is rpy2?

The **rpy2** provides simple and robust access to R from within Python.

- **Calling R from Python** in a variety of ways including **importing R packages, calling R functions**, and using R interactively within Python.
- **Translation between Python and R objects** (for example, between R and Pandas data frames, or between R matrices and NumPy arrays).

Source: https://rpy2.readthedocs.io/en/version_2.8.x

A brief guide to using rpy2 within Python

Install rpy2

- conda
- pip

Import rpy2 modules

```
import rpy2.robjects as robjects
```

Execute R Code

```
In [12]: robjects.r('paste0("he", "llo")')
Out[12]:
R object with classes: ('character',) mapped to:
<StrVector - Python:0x0000023E5C4931C8 / R:
0x0000023E62BD54F0>
['hello']
```

Install R Packages

```
robjects.r("install.packages('package')")
```

Import R Packages

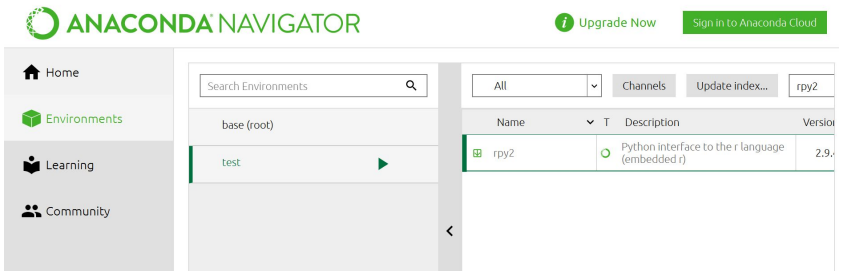
```
package=importr('package')
robjects.r('library(package)')
```

Use R Functions

```
function_name=robjects.r('fuction_name')
robjects.r('fuction_name(x,y)')
```

Installing and Importing rpy2

Installing rpy2




```
pip install rpy2
```

```
conda install rpy2
```

Importing rpy2

```
import rpy2.robjects as robjects
```

Gives the r() function

```
from rpy2.robjects import pandas2ri
```

Makes it easy to convert objects

```
from rpy2.robjects.packages import importr
```

Gives the ability to import functions

Executing an R Script

- Use r() in robjects
- Execute r function seq() on a date
  - Go back in time 4 periods, inclusive, by increments of 2 months

```
In [6]: date_list = robjects.r("as.character(seq(from = as.Date('2018-01-15'), by = ('-2 months'), length = 4))")

In [7]: date_list
Out[7]:
R object with classes: ('character',) mapped to:
<StrVector - Python:0x0000027D6B9BEA88 / R:0x0000027D6BF9B7A8>
['2018-01-15', '2017-11-15', '2017-09-15', '2017-07-15']
```

- pandas2ri.activate()

Import R Packages

- Some packages come installed when you install rpy2
- Some packages you have to install

```
robjects.r("install.packages('package')")
```

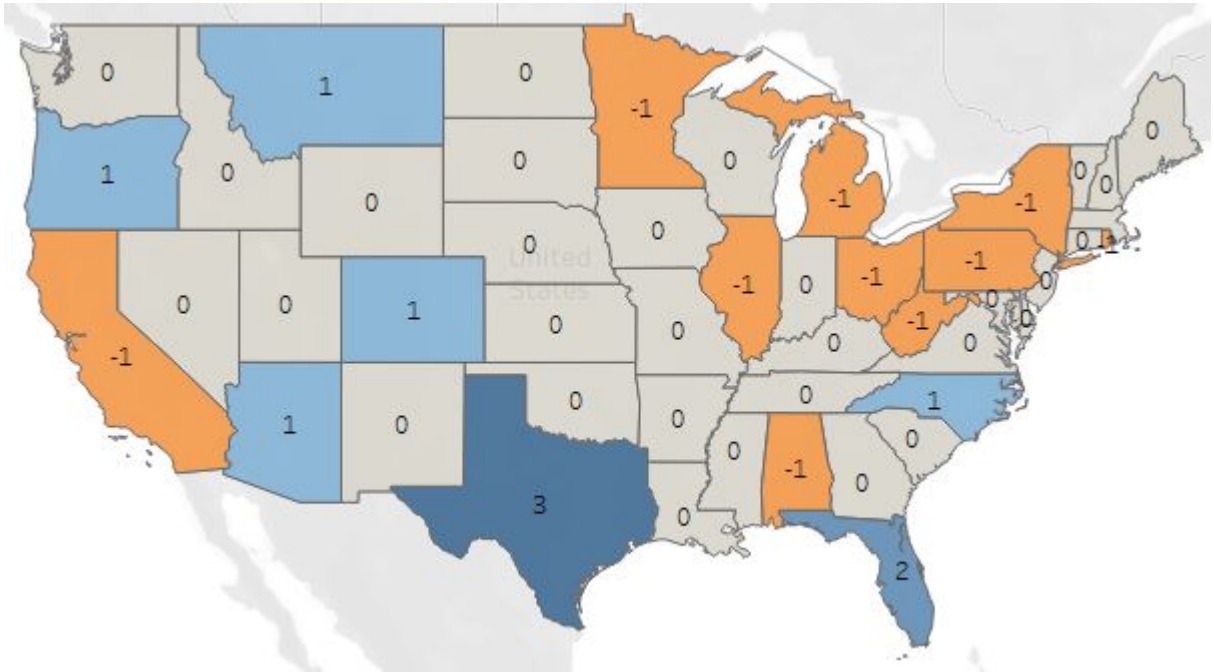
In r()

```
robjects.r('library(forecast)')
```

Import into Python

```
forecast=importr('forecast')
```

Estimating Electoral College Votes by State



Spoiler alert, Ohio and California are estimated to lose a vote

Apparently nice weather only helps so much

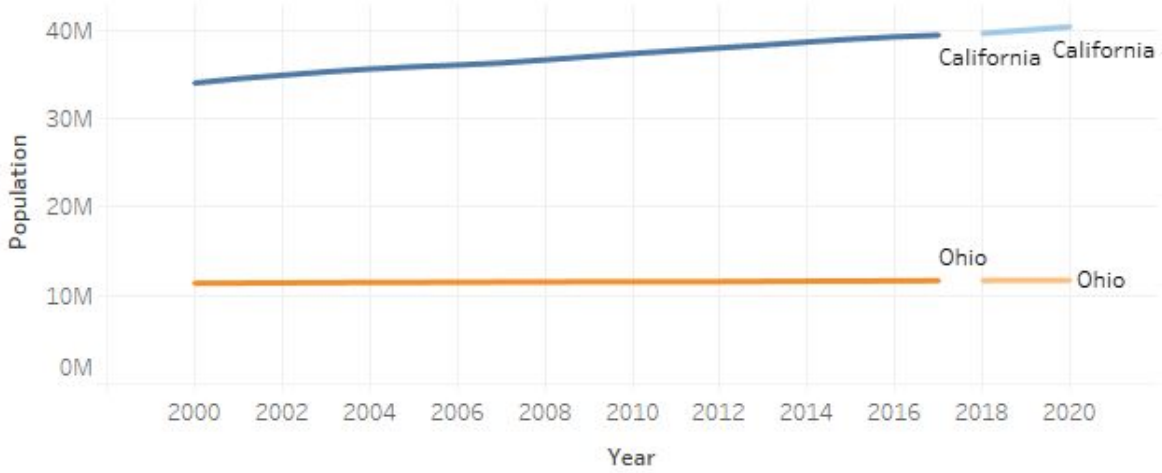
Scraping State Population Data

```
for year in range(2000, 2018):
    # Keeping track of the loop
    print(year)
    # This is the URL that has the data
    url = 'https://fred.stlouisfed.org/release/tables?rid=118&eid=259194&od='
    # Read in the url
    url_request = requests.get(url)
    # Make the soup
    pop_data = BeautifulSoup(url_request.content, features="lxml")
```

		Thousands of Persons		
Name	Period	Value	Preceding Period	Year Ago from Period
Alabama	2018	4,887.871	4,875.120	4,875.120
Alaska	2018	737.438	739.786	739.786
Arizona	2018	7,171.646	7,048.876	7,048.876
Arkansas	2018	3,013.825	3,002.997	3,002.997
California	2018	39,557.045	39,399.349	39,399.349

Using R Functions, Generate Forecasts

```
forecast_ts = ts(data_for_forecast)

fit = forecast.auto_arima(forecast_ts)

forecast_output=forecast.forecast(fit,h=2,level=(95.0))
```



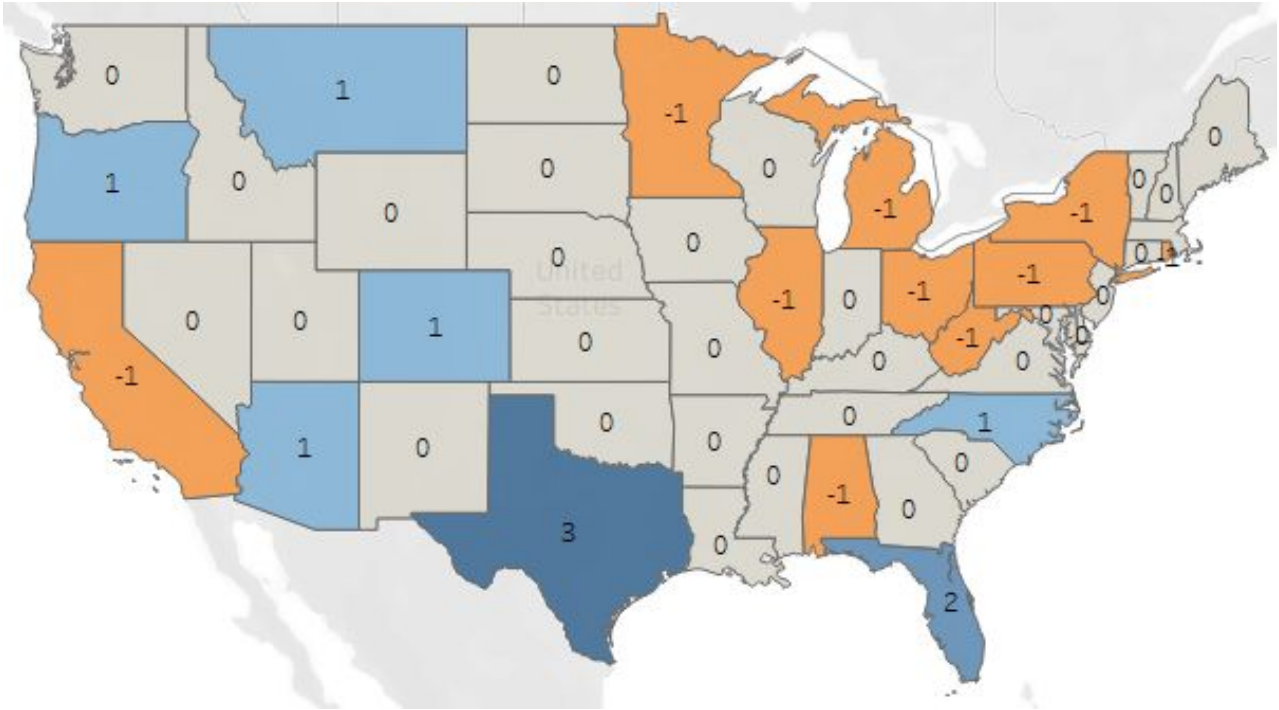
Assigning House Seats

- Electoral Votes = House Seats + Senate Seats

```
# While there are seats left to be allocated
while seats < 435:
    # Get the state with the Max priority
    max_state = state_forecasts[state_forecasts['Priority'] == max(state_forecasts['Priority'])]
    # Get the rest of the states
    not_max_state = state_forecasts[state_forecasts['Priority'] != max(state_forecasts['Priority'])]

    # Change the number, priority, and number of seats
    max_state.loc[:,'A_n'] = max_state['A_n'] + 1
    max_state.loc[:,'Priority'] = math.sqrt(max_state['Seats'].unique()/(max_state['Seats'].unique()+2))*state_forecasts['Priority']
    max_state.loc[:,'Seats'] = max_state['Seats'] + 1

    # Append all of the data back together
    state_forecasts = max_state.append(not_max_state)

    #Add up the number of seats
    seats = sum(state_forecasts['Seats'])
```

$$A_1 = \frac{P}{\sqrt{2}} \qquad A_{n+1} = \sqrt{\frac{n}{n+2}}\, A_n$$

Estimating Electoral College Votes by State



Alaska and Hawaii are not forgotten, just both at 0 change

Event: PyOhio 2019
Talk: Using Python & R in Harmony
Date: July 27th, 2019

Code + Slides

bit.ly/2LIzhbC