Intro to Python

What *is* R?

- R is a programming language specifically designed for statistics and data analysis
  - Objects for storing data, and functions for interacting with data, are fundamental
  - R is very good at graphics and visualization
  - R is easily extended. Users can write and share their own functions and packages
- We can interact with R through IDEs like RStudio

What other options exist?

- SAS
- Stata
- SPSS
- Excel
- Python
- Julia
- Matlab
- Many others...

What is Python

- Python is a general-purpose programming language
- Like R, python has a wide range of packages to extend functionality
- Certain Python packages allow for sophisticated data analysis and modeling
  - `SciPy`, `NumPy`
  - `scikit-learn`, `statsmodels`, `pytorch`
  - `pandas`
  - `matplotlib`

R vs. Python

My own, *personal*, preferences:

R is good for

- Data visualization and wrangling
- Classical statistics
- Statistical inference
- New statistical methods

Python is good for

- General-purpose programming
- Challenging data types (e.g. images)
- Prediction and machine learning

Warmup

Work on the warmup activity (handout).

A taste of Python

```
1  import numpy as np
2
3  M = 10
4  hats = np.arange(M)
5  nsim = 10000
6  results = np.zeros(nsim)
7
8  for i in range(nsim):
9      randomized_hats = np.random.choice(hats, M, replace = False)
10     results[i] = np.sum(randomized_hats == hats) > 0
11
12 np.mean(results)
```

- What is this code doing?
- What similarities and differences do you notice, compared to R?

A taste of Python

Recall our code from a previous class:

```r
M <- 10 # number of people at the party
hats <- 1:M # numbered hats
nsim <- 10000 # number of simulations
results <- rep(0, nsim) # vector to store the results

for(i in 1:nsim){
  randomized_hats <- sample(hats, M, replace = FALSE)
  results[i] <- sum(randomized_hats == hats) > 0
}

mean(results)
```

A taste of Python

Here is the same code, written in Python

```
import numpy as np

M = 10 # number of people at the party
hats = np.arange(M) # numbered hats
nsim = 10000 # number of simulations
results = np.zeros(nsim) # to store the results

for i in range(nsim):
    randomized_hats = np.random.choice(hats, M, replace = False)
    results[i] = np.sum(randomized_hats == hats) > 0

np.mean(results)
```

Step 1: representing the hats

```
1 import numpy as np
2
3 M = 10 # number of people at the party
4 hats = np.arange(M) # numbered hats
5
6 hats
```

array containing 0, 1, 2, ..., M−1 (length M)

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
1 hats[0]
```

```
np.int64(0)
```

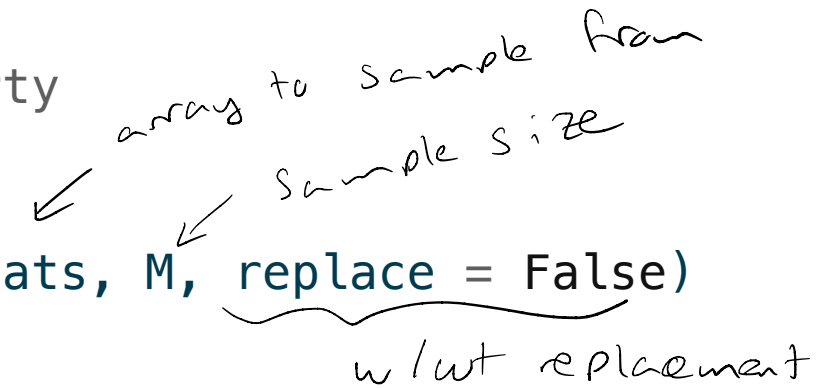
```
1 hats[1]
```

```
np.int64(1)
```

- `hats` is a 1-dimensional array (similar to a vector in R)
- Python is *0-indexed*: the first entry is `hats[0]`

Step 2: everyone draws a random hat

```
1 import numpy as np
2
3 M = 10 # number of people at the party
4 hats = np.arange(M) # numbered hats
5
6 randomized_hats = np.random.choice(hats, M, replace = False)
7
8 randomized_hats
```




```
array([7, 6, 1, 0, 9, 2, 8, 5, 4, 3])
```

- `np.random.choice` works like R's `sample` function
- Booleans in Python are `True` and `False` (as opposed to `TRUE` and `FALSE`, or `T` and `F`)

Step 3: check who got their original hat

```
1  import numpy as np
2
3  M = 10 # number of people at the party
4  hats = np.arange(M) # numbered hats
5
6  randomized_hats = np.random.choice(hats, M, replace = False)
7  randomized_hats
```

```
array([0, 3, 1, 5, 8, 7, 6, 4, 9, 2])
```

```
1  randomized_hats == hats
```

test for equality

```
array([ True, False, False, False, False, False,  True, False, False,
       False])
```

```
1  np.sum(randomized_hats == hats)
```

← get number of "True"s

```
np.int64(2)
```

- NumPy arrays allow for “vectorized” operations, like in R

Step 4: iteration

```
 1 import numpy as np
 2
 3 M = 10 # number of people at the party
 4 hats = np.arange(M) # numbered hats
 5 nsim = 10000 # number of simulations
 6 results = np.zeros(nsim) # to store the results
 7
 8 for i in range(nsim):
 9     randomized_hats = np.random.choice(hats, M, replace = False)
10     results[i] = np.sum(randomized_hats == hats) > 0
11
12 np.mean(results)
```

0 0 0 ... 0

0, 1, 2, ..., nsim - 1

- we don't use curly braces { }

  Instead we use whitespace (four spaces is standard, but you just have to be consistent)

Using Python through RStudio

- You can make Python chunks in Quarto documents, just like R chunks:

````
```{python}

```
````

Class activity

Work on the class activity on the course website. You do not need to submit anything for this activity.