Data science for everyone

Prof. Jones-Rooy & Prof. Policastro

March 4, 2020

6.2: Visualizations & Functions

ANNOUNCEMENTS

- Lab 3 due today, March 4, 8p
- Lab 4 out today, March 4, 8p; due Monday, March 9, 8p
- HW 2 due Monday, March 9
- Project description available on JupyterHub shared folder
- **Midterm**
  - Bring questions to section this week (in addition to lab 4, review HW1)
  - Review in lecture Monday, March 9
  - TA office hours review session: Tuesday, March 10, 2-5p CDS rooms 660 & 665
  - I will hold extra office hours on Wednesday 12-2p in CDS 640
  - Exam in lecture Wednesday, March 11 (60 minutes)

Outline

1. Data visualizations in Python
2. An example from start → visualization
3. Functions & other handy code

FOUR MAJOR TYPES OF VISUALIZATIONS

Histogram

More reading on histograms (esp. v. bar chars) here

Optional!

```
1 plt.hist(data['temperament'], bins=10
2 plt.show()
```

Bar chart

```
1 data['weight'].value_counts().plot(kind='bar')
2 plt.show()
```

Scatterplot

```
1 plt.scatter(data['temperament'], data['height'])
2 plt.show()
```

Line graph

```
1 plt.plot(data['temperament'])
2 plt.show()
```

CREATING VISUALIZATIONS IN PYTHON

See Lecture 6.2 – Visualizations Code
On Classes (under Resources) and JupyterHub!

Outline

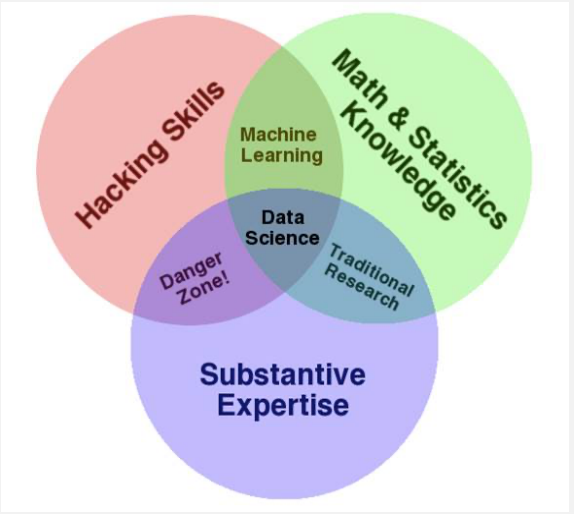
1. Data visualizations in Python
2. An example from start → visualization
3. Functions & other handy code

A NEW, REAL DATASET

- A.K.A. Please for the love of all things good in this world, no more horses.csv!
- Your wish is my command:
- **Polity IV dataset!**
  - By the Center for Systemic Peace
- A dataset about democracy & democratization
  - (note some possible invisibility bias/error of exclusion already)

Polity = A society with organized government

- Authority patterns based on how states (countries) are organized
- Authority patterns = asymmetric relations between groups of humans



Center for Systemic Peace

HOME | ABOUT | PUBLICATIONS | ANALYSIS | INSCR DATA | VIDEO BOOK

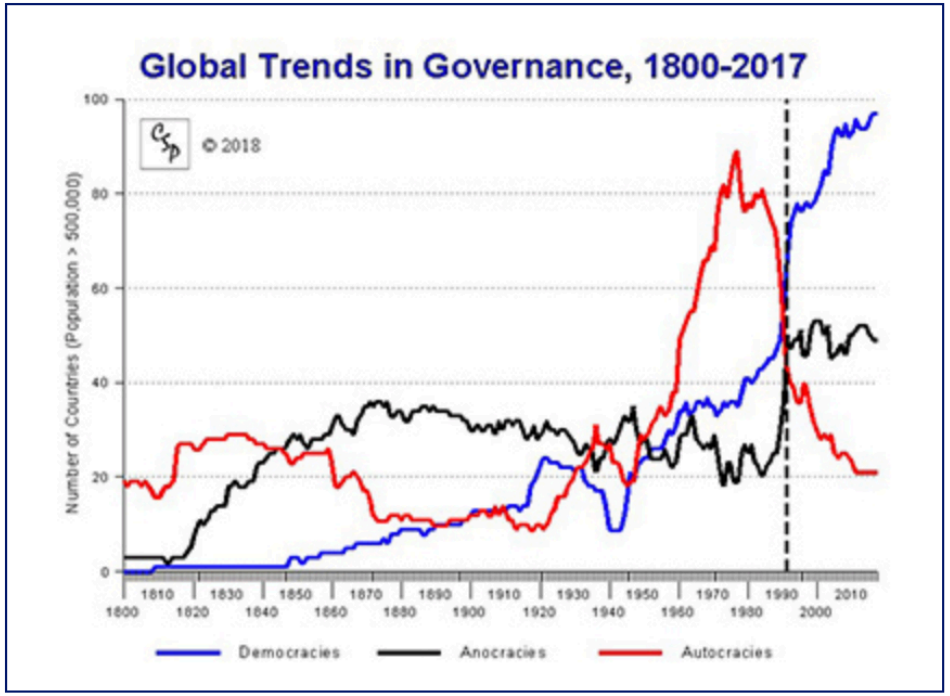
Note
Two kinds of datasets (generally):
Ongoing and simple
Occasional and processed/coded

About Polity

The Polity IV Project continues the Polity research tradition of coding authority characteristics of states in the world system for purposes of comparative, quantitative analysis. An improved and enhanced Polity 5 version in the series is currently in development.

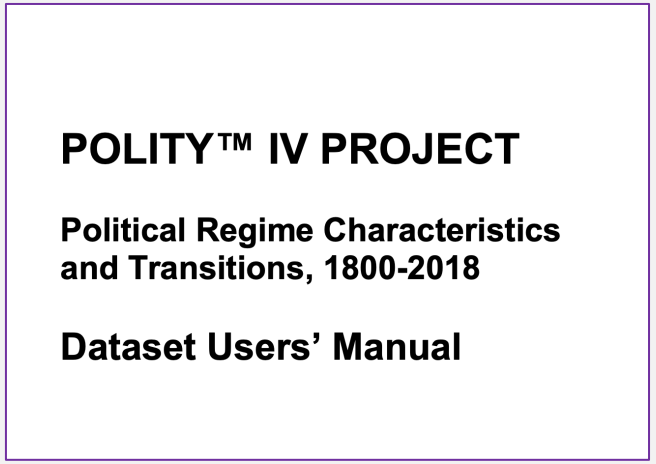
The Polity IV dataset covers all major, independent states in the global system over the period 1800-2017 (i.e., states with a total population of 500,000 or more in the most recent year; currently 167 countries). With the support of the Political Instability Task Force, the Polity IV Project has been transformed into a living data collection effort, meaning that it constantly monitors regime changes in all major countries and provides annual assessments of regime authority characteristics, changes and data updates.

The Polity conceptual scheme is unique in that it examines concomitant qualities of democratic and autocratic authority in governing institutions, rather than discreet and mutually exclusive forms of governance. This perspective envisions a spectrum of governing authority that spans from fully institutionalized



FIND THE DATA

- http://www.systemicpeace.org/inscrdata.html
  - Polity IV Annual Time-Series, 1800-2017
  - Original .xls file and codebook ("Users Manual PDF")
- I'm going to create "polity4.csv" and upload/move it to the right directory so I can import into Jupyter notebook


POLITY™ IV PROJECT

Political Regime Characteristics and Transitions, 1800-2018

Dataset Users' Manual

POLITY SCORES

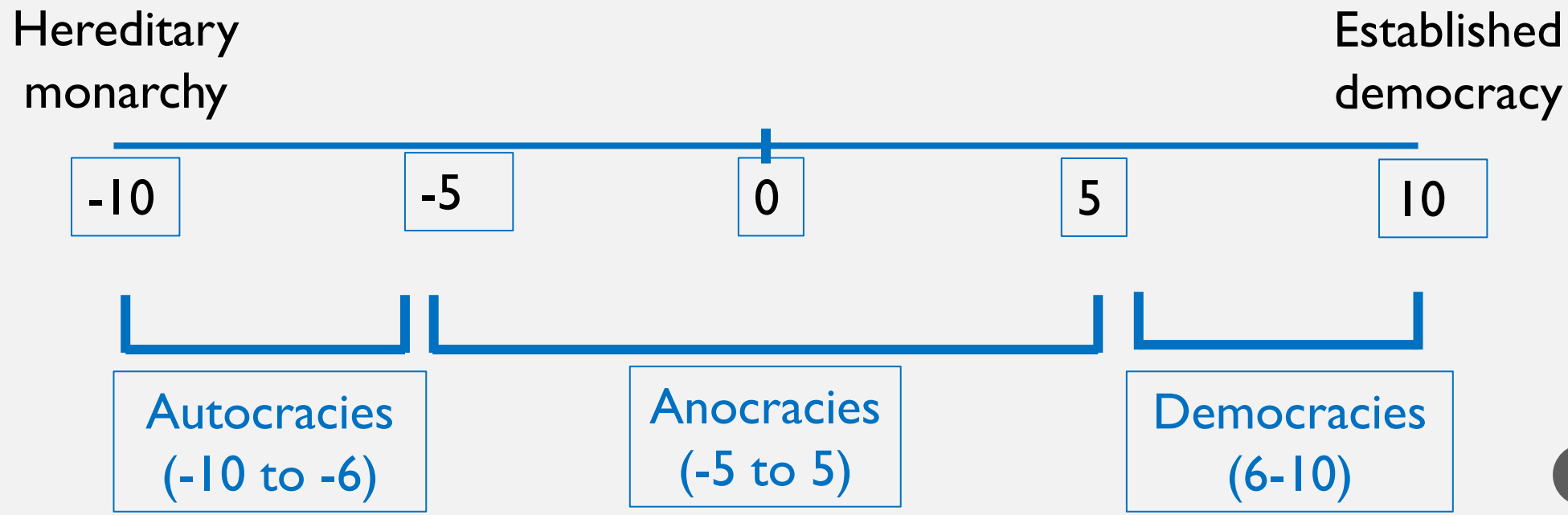
- DEMOCRACY score (0 to 10)
- AUTOCRACY score (0 to 10)

- Competitiveness of executive recruitment
- Openness of executive recruitment
- Constraints on chief executive
- Regulation of participation
- Competitiveness of political participation

POLITY = DEMOCRACY – AUTOCRACY

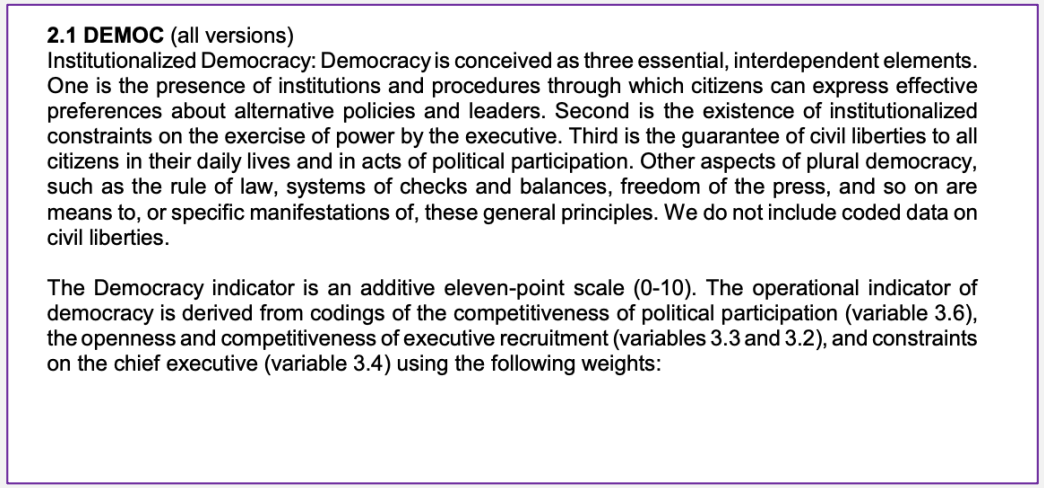
POLITY SCORE

- Regime authority spectrum
- Measures regime type on a 21-point scale from -10 to 10



EVALUATE MEASUREMENT & DATA QUALITY

- Conceptualization & Operationalization
- Random, systematic/selection, validity, exclusion/invisibility
- You can get all this from the codebook, and of course by exploring the data to look for irregularities or issues
  - e.g., tons of missing data, esp. if not accounted for in the codebook (not that that makes it stop being worth thinking about, but can increase confidence in the dataset)



2.1 DEMOC (all versions)
Institutionalized Democracy: Democracy is conceived as three essential, interdependent elements. One is the presence of institutions and procedures through which citizens can express effective preferences about alternative policies and leaders. Second is the existence of institutionalized constraints on the exercise of power by the executive. Third is the guarantee of civil liberties to all citizens in their daily lives and in acts of political participation. Other aspects of plural democracy, such as the rule of law, systems of checks and balances, freedom of the press, and so on are means to, or specific manifestations of, these general principles. We do not include coded data on civil liberties.

The Democracy indicator is an additive eleven-point scale (0-10). The operational indicator of democracy is derived from codings of the competitiveness of political participation (variable 3.6), the openness and competitiveness of executive recruitment (variables 3.3 and 3.2), and constraints on the chief executive (variable 3.4) using the following weights:

Most codebooks provide measurement information overall and also by variable

(See, e.g., this excerpt on p. 14 of the polity4 codebook.)

WORKING WITH THE DATA IN PYTHON

- Import
- Inspect – make sense of (what are the variables, how many observations, what types of data are we working with)
- Clean – if necessary to make it easier to use for our purposes
- Explore descriptive statistics for the whole thing
- Explore descriptive statistics for subsets we might be interested in
- Create visualizations for the whole thing and/or subsets of interest

This dataset is both **time series** and **cross-sectional**

This means we have observations for: **specific cases over time** and **over many cases in one time**

So: we will likely we working with subsets quite a bit

WORKING WITH THE DATA IN PYTHON

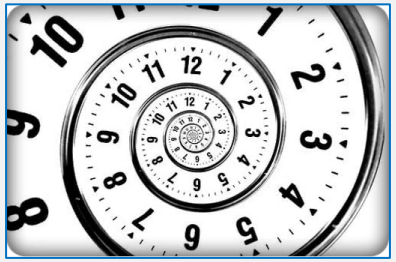
See Lecture 6.2 – Example Code
On Classes (under Resources) and JupyterHub!

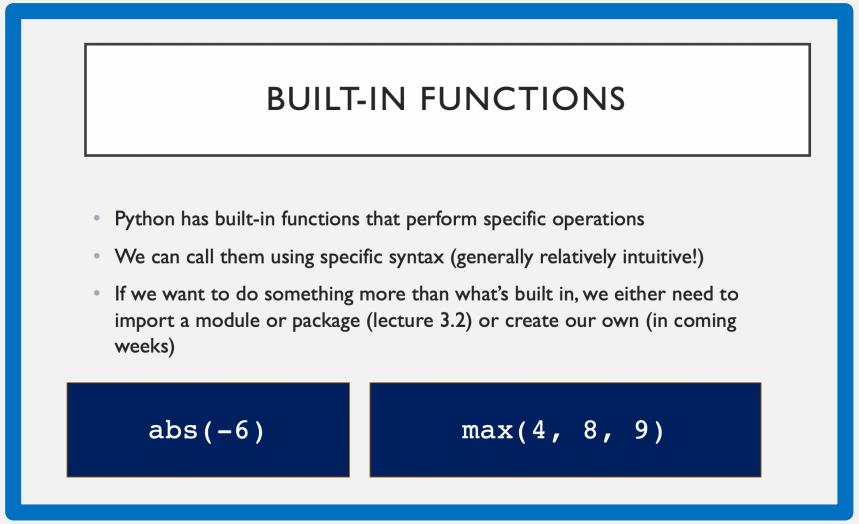
Outline

1. Data visualizations in Python
2. An example from start → visualization
3. Functions & other handy code

FUNCTIONS

- We learned in lecture 3.1 that Python has built-in functions
  - E.g., max(1, 4, 7, 3) or abs(-200)
- We discussed that if we want to do something more complicated than the default built-in functions we can import a module or package (which have their own built-in functions)
  - E.g., plt.hist(data['variable'])
- And we said we can also just **write our own**!
  - And that we would do so in a few weeks
  - That few weeks is now!





CREATING A FUNCTION

- Function = blocks of organized, reusable code that performs a specific operation
  - In Python, packages, or we write ourselves
- Format for writing ourselves:

```
def function_name(input argument):
    'Description of function'
    return expression or other outcome (e.g., print)
```

```
def triple(x):
    'This triples the amount in x'
    return 3*x
```

```
triple(7)
```

```
21
```

FUNCTIONS CAN WORK WITH MORE THAN ONE VARIABLE

```
def sum_numbers(x, y):
    'Add the numbers'
    return x + y
```

```
sum_numbers(11, 6)
```

```
17
```

```
result = sum_numbers(11, 6)
result
```

```
17
```

We also don't have to write out the function description

```
1 def double(x):
2     return 2*x
```

```
1 double(14)
```

```
28
```

See Lecture 6.2 – Example Code On Classes (under Resources) and JupyterHub!

RANGES

- A kind of sequence that we can automate (creates an array)
- Syntax is np.arange(start, end, interval)
  - The start is inclusive
  - The end is not
  - We don't have to have all three elements (e.g., we could leave off interval and the default is by 1)

```
1  myrange = np.arange(0, 20, 3)
2  myrange
```

```
array([ 0,  3,  6,  9, 12, 15, 18])
```

See Lecture 6.2 – Example Code On Classes (under Resources) and JupyterHub!

CONDITIONAL STATEMENTS

- Create decisions in Python
- Decision making is required if we want to execute a code only if certain conditions are satisfied

if

```
if test expression:
    statement(s)
```

if…else

```
if test expression:
    Body of if
else:
    Body of else
```

if…else if…else

```
if test expression:
    Body of if
elif test expression:
    Body of elif
else:
    Body of else
```

EXAMPLES OF CONDITIONAL STATEMENTS

if

```
x = 5
if x > 0:
    print(x, 'is positive')
```

```
5 is positive
```

```
y = -1
if y > 0:
    print(y, "is negative")
```

(nothing!)

```
y = -1
if y < 0:
    print(y, "is negative")
```

```
-1 is negative
```

if…else

```
x = -4
if x >= 0:
    print(x, 'is positive or zero')
else:
    print(x, 'is negative')
```

```
-4 is negative
```

if…else if…else

```
x = -2

if x > 0:
    print(x, 'is positive')
elif x == 0:
    print(x, 'is zero')
else:
    print(x, 'is negative')
```

```
-2 is negative
```

See Lecture 6.2 – Example Code On Classes (under Resources) and JupyterHub!

Outline

1. Data visualizations in Python
2. An example from start → visualization
3. Functions & other handy code

The end!