Data analysis with pandas

Numpy and matplotlib in one convenient package!

T. Cragnolini - 14/02/2024

Introduction

- Pandas is one of the most popular python library
  - To load, filter, analyse, and display data
- As a first approximation, you can think of it as Excel for Python!
- It uses, and plays nicely with, numpy, and other scientific libraries
- Deal well with heterogeneous datasets: texts and numbers
- Provides many handy functions for data processing
- Relatively fast

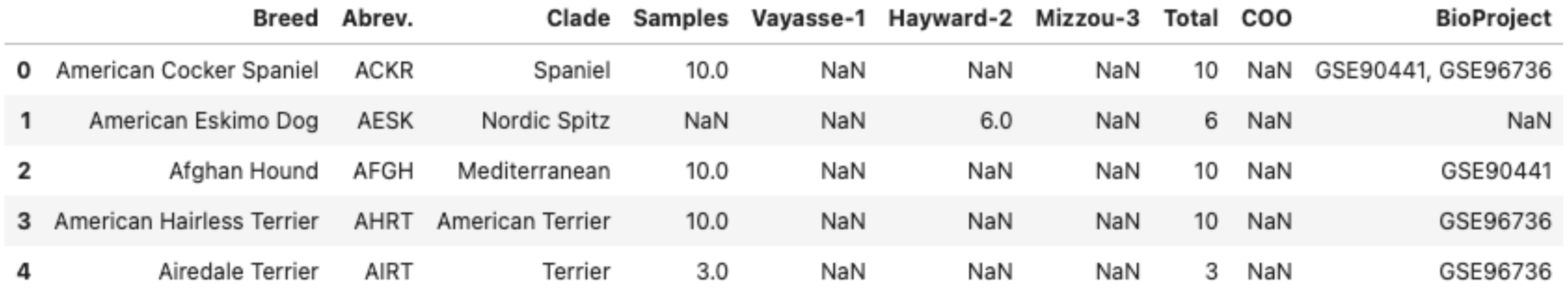
Installing pandas

- Pandas is a third-party library
- It needs to be installed (already done on jupyterhub)
- Pip or conda are common tools to manage third-party packages
- If you have installed python locally, you should have pip, then:
  ```
  pip install pandas
  ```
  should add Pandas to your local python install!

Reading data

- Pandas needs to first be imported, like all packages.
- The data is then stored in a variable (called “df” here):

```
import pandas as pd
df = pd.read_excel("dog_summary.xls", skipfooter=9)
df.head(5)
```



	Breed	Abrev.	Clade	Samples	Vayasse-1	Hayward-2	Mizzou-3	Total	COO	BioProject
0	American Cocker Spaniel	ACKR	Spaniel	10.0	NaN	NaN	NaN	10	NaN	GSE90441, GSE96736
1	American Eskimo Dog	AESK	Nordic Spitz	NaN	NaN	6.0	NaN	6	NaN	NaN
2	Afghan Hound	AFGH	Mediterranean	10.0	NaN	NaN	NaN	10	NaN	GSE90441
3	American Hairless Terrier	AHRT	American Terrier	10.0	NaN	NaN	NaN	10	NaN	GSE96736
4	Airedale Terrier	AIRT	Terrier	3.0	NaN	NaN	NaN	3	NaN	GSE96736

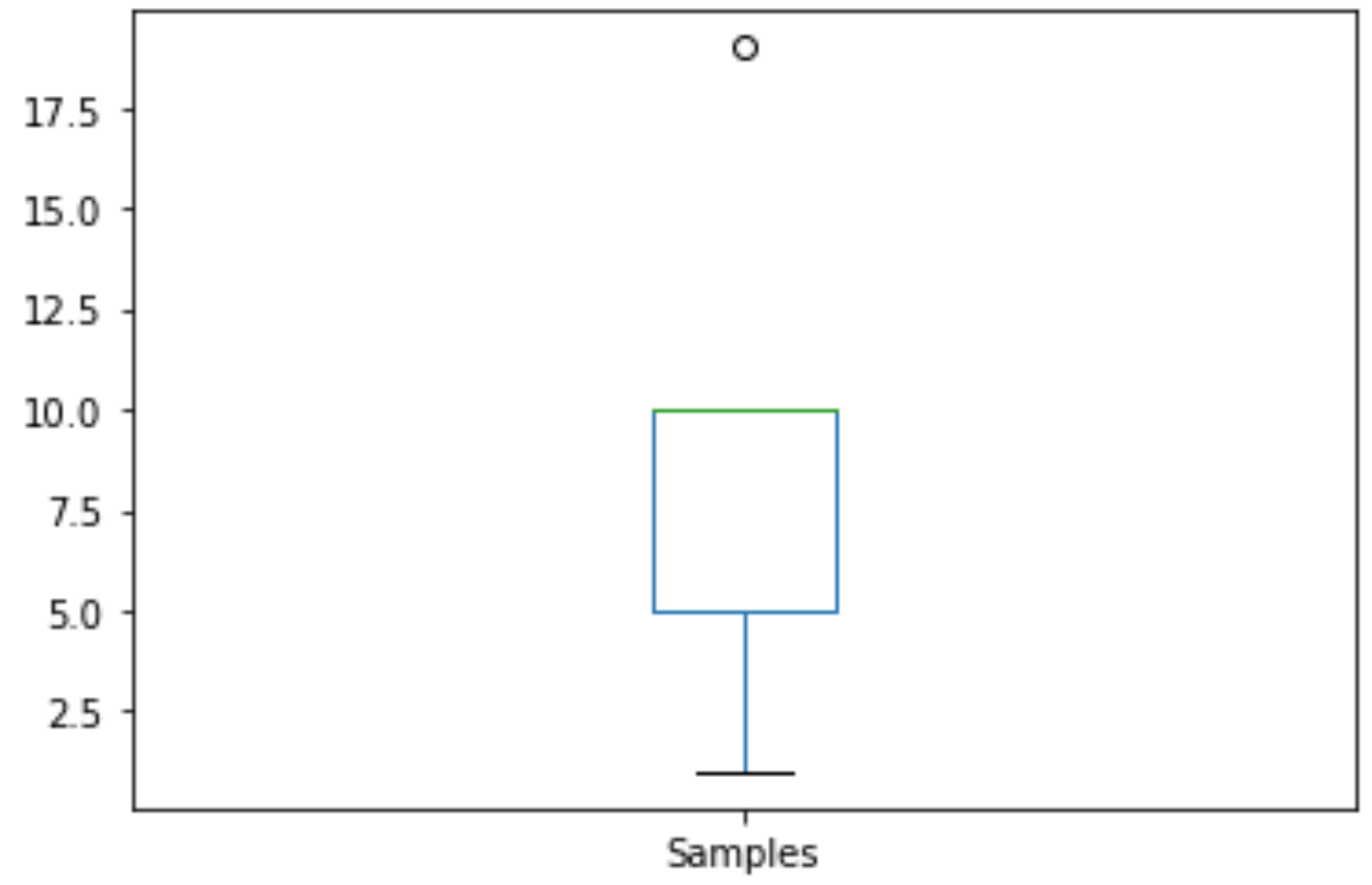
Pandas DataFrame

- DataFrames are similar to numpy array
    - in fact, you can get a numpy array out of any pandas dataframe, with `.values`: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.values.html
- You can think of them as spreadsheets

Getting statistics and plots

- Pandas dataframes let you get summary statistics and plot their content quite easily:

```
df["Samples"].median()
> 10.0
df[“Samples"].plot(kind="box
```



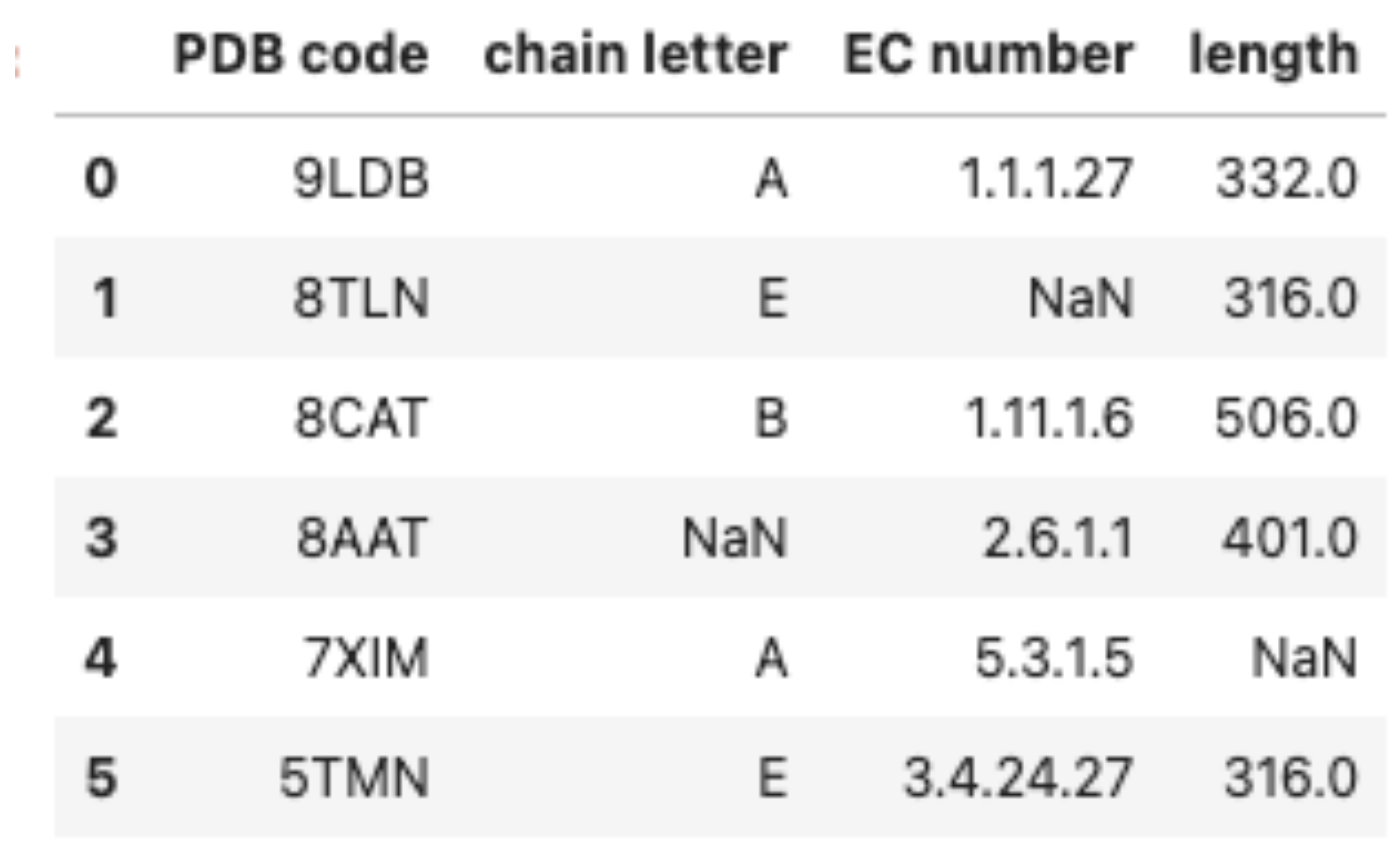
Accessing data

Accessing data in DataFrames

- Similar to dictionaries: `df['length']`
    - The ‘key’ is the column name
- This returns a Series object, **not** a DataFrame!
- Multiple columns can be accessed by passing a list:
    - `df[['length', 'PDB code']]`

Accessing data in DataFrames

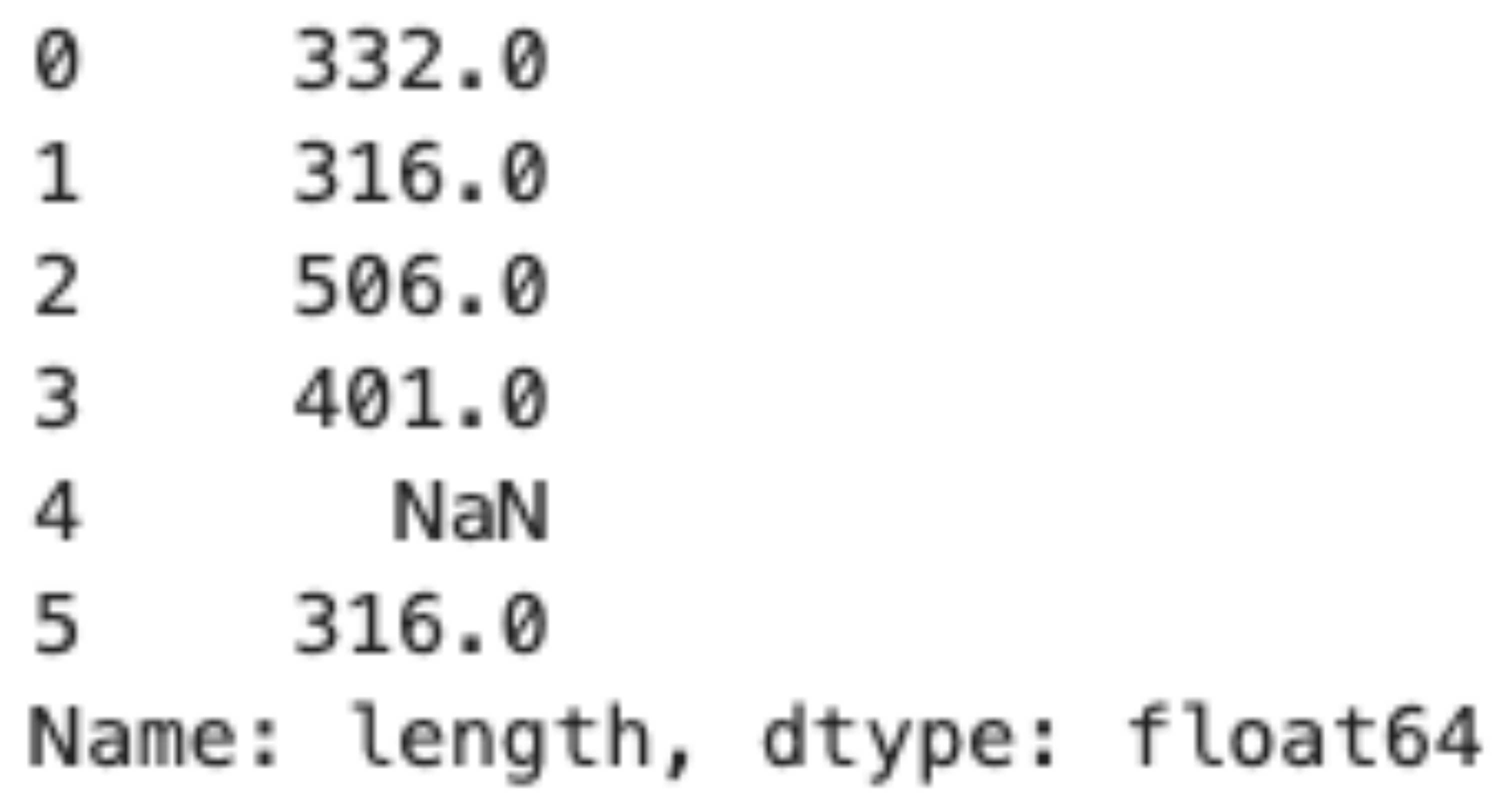
df



	PDB code	chain letter	EC number	length
0	9LDB	A	1.1.1.27	332.0
1	8TLN	E	NaN	316.0
2	8CAT	B	1.11.1.6	506.0
3	8AAT	NaN	2.6.1.1	401.0
4	7XIM	A	5.3.1.5	NaN
5	5TMN	E	3.4.24.27	316.0

DataFrame

df[‘length’]



```
0    332.0
1    316.0
2    506.0
3    401.0
4      NaN
5    316.0
Name: length, dtype: float64
```

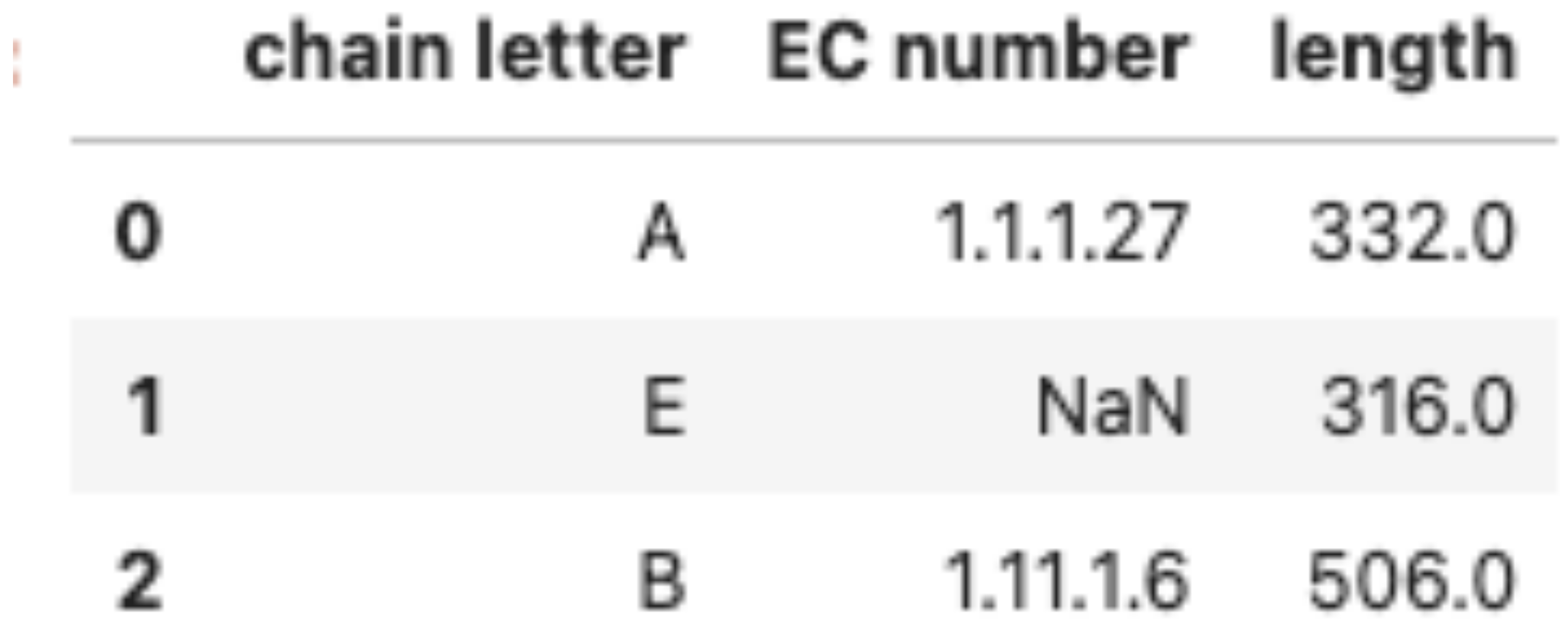
Series (aka a column)

Accessing data in DataFrames

- df.loc[1] # access the row with index 1
- df.iloc[1] # access the second row
- What is the difference?
- Pandas DataFrames can have arbitrary index values (even letters!)
    - loc will look for the matching index
    - iloc will access the data by position

Accessing data in DataFrames

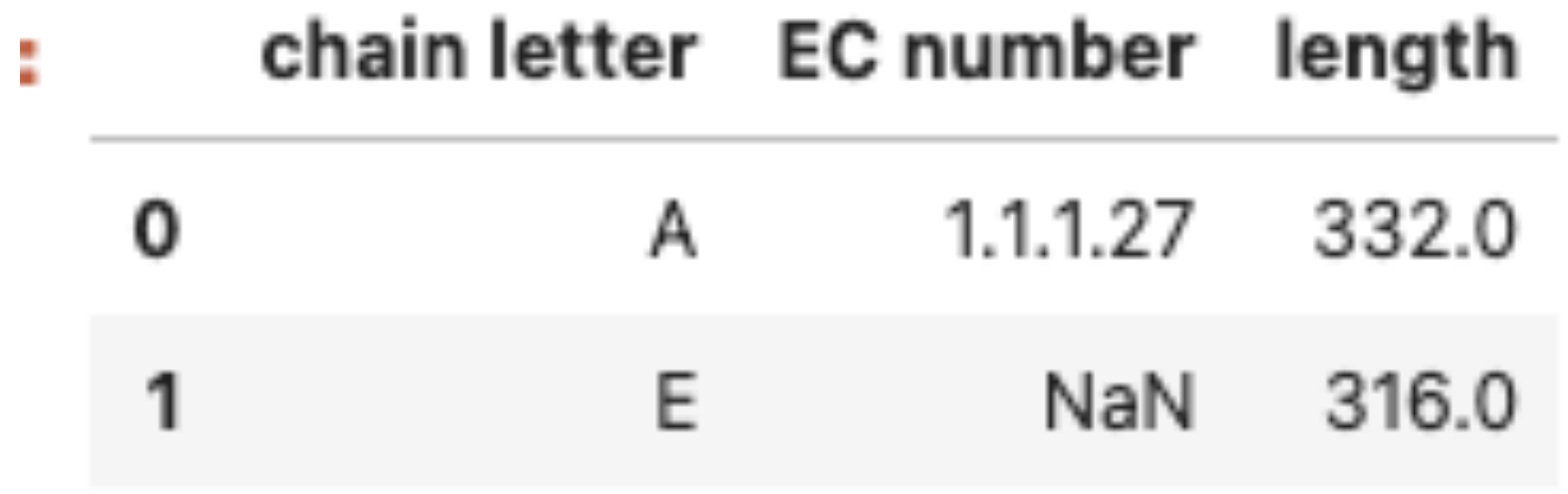
- Slicing works on Pandas dataframes:
- `df.loc[0:2, 'chain letter':'length']`



	chain letter	EC number	length
0	A	1.1.1.27	332.0
1	E	NaN	316.0
2	B	1.11.1.6	506.0

Accessing data in DataFrames

- Similarly with iloc:
- `df.iloc[0:2, 1:4]`



	chain letter	EC number	length
0	A	1.1.1.27	332.0
1	E	NaN	316.0

- Note that iloc uses indices in the same way as lists and numpy arrays!

Adding new data

- Works just like a dictionary:
  df["four"] = 4 #All the values in this new column will be ‘4’!
- A more useful example:
  df["normalised_length"] = df["length"] / df[“length”].sum()

- The `df.append` method lets you add rows, and do more complex operations.

DataFrame methods, attributes

Useful DataFrame methods

- df.head(N) # show the first N rows
- df.tail(N) # show the last N rows
- df.describe() # compute statistics on the DataFrame
- df.to_numpy() # export a numpy array
- df.to_csv("file.csv") # save the DataFrame to a csv file
- There are many `df.to_` methods!
  - To clipboard, dict, pdf, html, json, latex, SQL, etc...

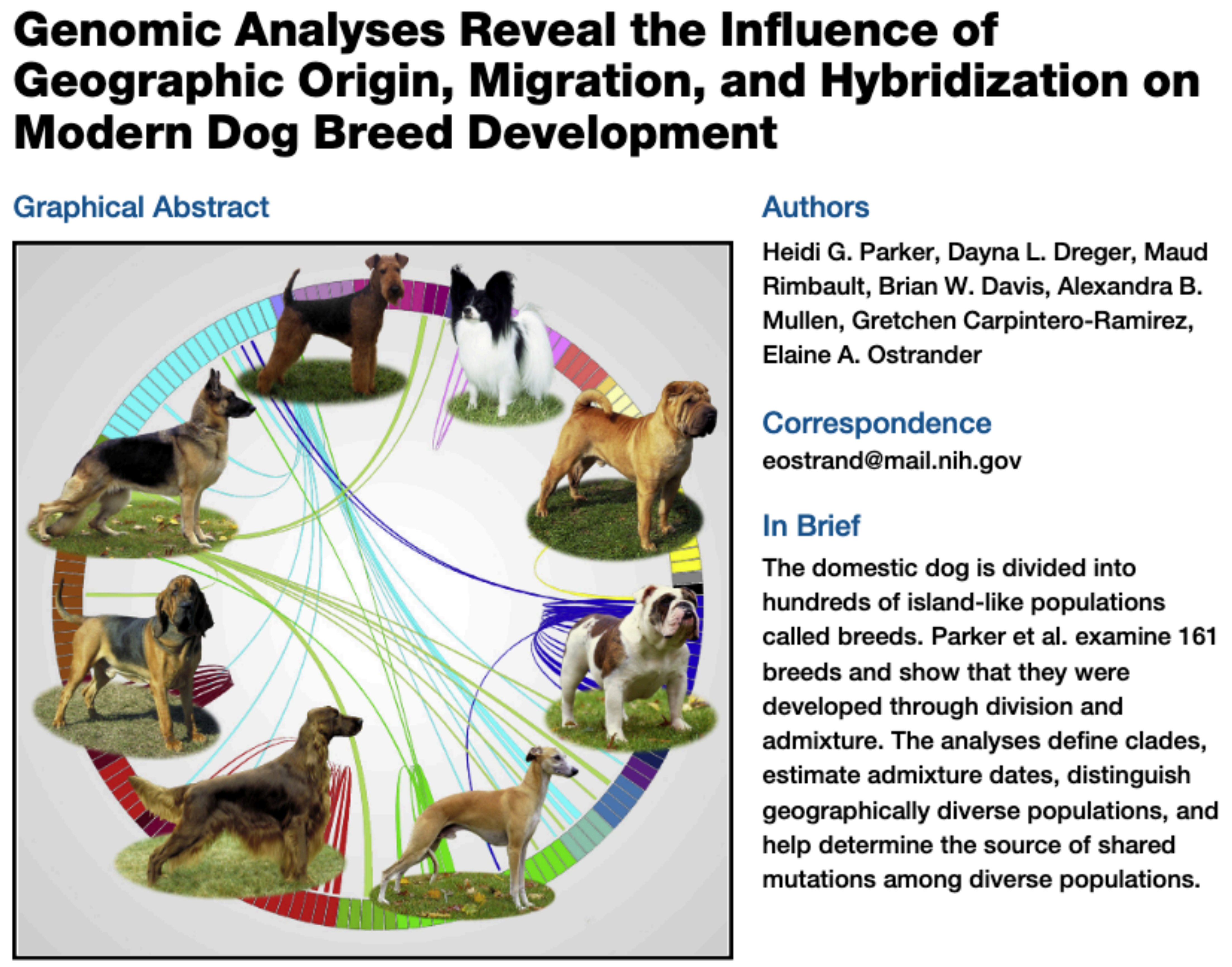
Useful DataFrame attributes

- df.columns # list of column names
- df.index # list of indices (or some other index object!)
- df.dtypes # list of column **types**
- df.describe() # compute statistics on the DataFrame

Practical

Analysing data: dogs

- We use spreadsheets from this paper
- We can get some summary data using pandas



Genomic Analyses Reveal the Influence of Geographic Origin, Migration, and Hybridization on Modern Dog Breed Development

Graphical Abstract

Authors

Heidi G. Parker, Dayna L. Dreger, Maud Rimbault, Brian W. Davis, Alexandra B. Mullen, Gretchen Carpintero-Ramirez, Elaine A. Ostrander

Correspondence

eostrand@mail.nih.gov

In Brief

The domestic dog is divided into hundreds of island-like populations called breeds. Parker et al. examine 161 breeds and show that they were developed through division and admixture. The analyses define clades, estimate admixture dates, distinguish geographically diverse populations, and help determine the source of shared mutations among diverse populations.

https://doi.org/10.1016/j.celrep.2017.03.079

The cutting edge? Polars

- Polars is newer, faster, and written in Rust (with Python bindings), and becoming quite popular: https://www.pola.rs
- New packages pop up all the time, they may appear better
  - But sometimes their flaws are simply not apparent!
  - Will it overtake pandas? Time will tell

Why not start with pandas?

- With pandas available, is there a use for matplotlib+numpy?
  - **YES!**
- Pandas is built on top of bumpy and matplotlib
  - Knowing how they work will help you use pandas better
- In general, more “advanced” libraries are built on top of simpler libraries, more basic code
  - Python interpreter is written in C! https://github.com/python/cpython
- Knowing the basics will help with advanced uses.