data analysis with pandas

1. Series and DataFrames
   - pandas for data analysis
   - examples of the data structures
   - making DataFrames
2. An Application
   - analyzing reviews from video games
   - asking questions about the data
3. Visualization
   - making histograms with matplotlib in ipython

MCS 507 Lecture 25
Mathematical, Statistical and Scientific Software
Jan Verschelde, 18 October 2023

data analysis with pandas

1. Series and DataFrames
   - pandas for data analysis
   - examples of the data structures
   - making DataFrames
2. An Application
   - analyzing reviews from video games
   - asking questions about the data
3. Visualization
   - making histograms with matplotlib in ipython

background

The software `pandas` was built to satisfy a set of requirements:

- Data structures with labeled axes should support data alignment, both automatically and explictly.
- Functionality to integrate time series.
- The same data structures should handle both times series data and nontime series data.
- Arithmetic operations and reductions (like summing across an axis) should pass on the metadata (axis labels).
- Flexible handling of missing data.
- Support for merge and other relational operations as in databases.

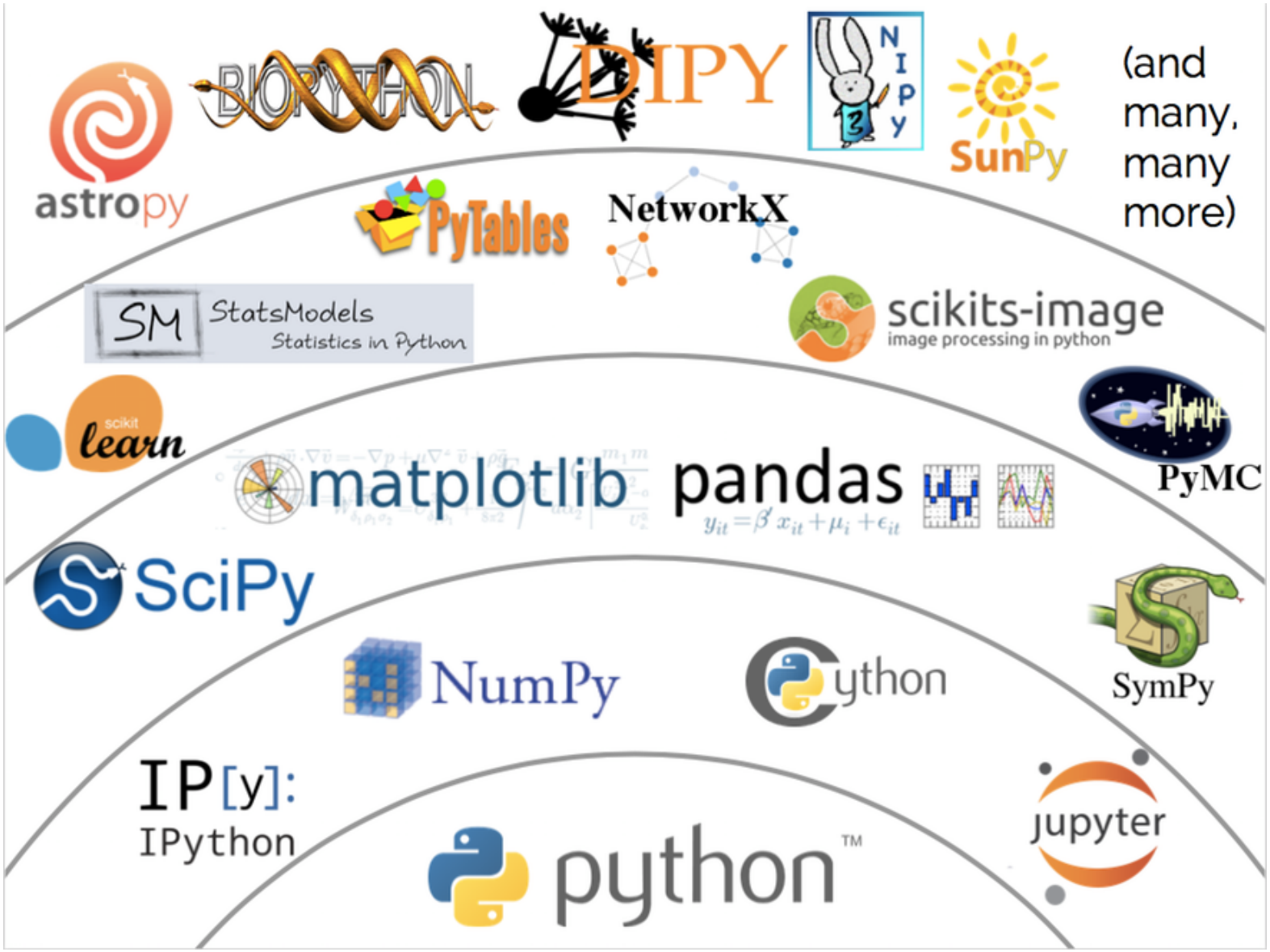
Wes McKinney: *Python for Data Analysis*, O'Reilly 2013.

about pandas

- open source Python library
- uses numpy for performance
- uses matplotlib for visualization
- SQL operations can be done with pandas
- installs with conda or pip
- widely used for data analysis

pandas in the stack

picture from the slides of Jake VanderPlas



data structures

We can organize the pandas data structures by dimension:

1. A **Series** is a one dimensional labeled array, capable of storing data of any type. The axis labels are called the index.
2. A **DataFrame** is a table with rows and colums.
   - columns may be of different type,
   - the size is mutable,
   - axes are labeled,
   - arithmetic can be performed on the data.
3. A **Panel** is a 3d container of data.

The name pandas is derived from Panel Data, as pan(el)-da(ta)-s.

```
>>> from pandas import Panel
__main__:1: FutureWarning: The Panel class is removed
from pandas.
```

data frames in Julia

The package `DataFrames.jl` is the Julia analogue to Pandas.

A recommended source:

- Jose Storopoli, Rik Huijzer, Lazaro Alonso: *Julia Data Science*.
  First edition published 2021. `https://juliadatascience.io`
  Creative Commons Attribution-Noncommercial-ShareAlike 4.0 International

data analysis with pandas

1. Series and DataFrames
   - pandas for data analysis
   - examples of the data structures
   - making DataFrames
2. An Application
   - analyzing reviews from video games
   - asking questions about the data
3. Visualization
   - making histograms with matplotlib in ipython

currency conversion rates

To illustrate pandas data types, consider three major currencies

USD = US dollar, EUR = euro, GBP = Pound Sterling,

and their conversion rates:

	USD	EUR	GBP
USD	1.0	0.91	0.79
EUR	1.1	1.0	0.87
GBP	1.26	1.14	1.0

Running through this example by dimension:

1. One column in this table we represent by a **Series**.
2. The 2-dimensional table is stored as a **DataFrame**.
3. The historical evolution of the rates gives a **Panel**.
   Instead of `Panel`, we use `xarray` for multi-dimensional arrays, on top of `numpy` and for its interaction with `pandas`.

making Series from a numpy array

	USD	EUR	GBP
USD	1.0	0.91	0.79
EUR	1.1	1.0	0.87
GBP	1.26	1.14	1.0

```
>>> import pandas as pd
>>> import numpy as np
>>> currencies = ['USD', 'EUR', 'GBP']
>>> usd_data = np.array([1.0, 1.1, 1.26])
>>> usd = pd.Series(usd_data, index=currencies)
>>> usd
USD    1.00
EUR    1.10
GBP    1.26
dtype: float64
```

session continued

```
>>> usd
USD    1.00
EUR    1.10
GBP    1.26
dtype: float64
>>> usd['EUR']
1.1
>>> usd.values
array([1.  , 1.1 , 1.26])
>>> usd.index
Index(['USD', 'EUR', 'GBP'], dtype='object')
>>> usd.size
3
>>> usd.iloc[1]
1.1
```

data analysis with pandas

1. Series and DataFrames
   - pandas for data analysis
   - examples of the data structures
   - making DataFrames
2. An Application
   - analyzing reviews from video games
   - asking questions about the data
3. Visualization
   - making histograms with matplotlib in ipython

a DataFrame from a dictionary of series

```
>>> import pandas as pd
>>> import numpy as np
>>> currencies = ['USD', 'EUR', 'GBP']
>>> usd_data = np.array([1.0, 1.1, 1.26])
>>> usd = pd.Series(usd_data, index=currencies)

>>> eur_data = np.array([0.91, 1.0, 1.14])
>>> gbp_data = np.array([0.79, 0.87, 1.0])
>>> eur = pd.Series(eur_data, index=currencies)
>>> gbp = pd.Series(gbp_data, index=currencies)
>>> dic = {'USD': usd, 'EUR': eur, 'GBP': gbp}
>>> rates = pd.DataFrame(dic)
>>> rates
      USD   EUR   GBP
USD  1.00  0.91  0.79
EUR  1.10  1.00  0.87
GBP  1.26  1.14  1.00
```

session continued

```
>>> rates.values
array([[1.  , 0.91, 0.79],
       [1.1 , 1.  , 0.87],
       [1.26, 1.14, 1.]])
>>> rates.index
Index(['USD', 'EUR', 'GBP'], dtype='object')
>>> rates['USD']
USD    1.00
EUR    1.10
GBP    1.26
Name: USD, dtype: float64
>>> rates['USD']['EUR']
1.1
>>> rates.size
9
>>> rates.iloc[1][0]
1.1
>>> rates.columns
Index(['USD', 'EUR', 'GBP'], dtype='object')
```

a DataFrame from a numpy array

```
>>> import pandas as pd
>>> import numpy as np
>>> currencies = ['USD', 'EUR', 'GBP']
>>> data = np.array([['', 'USD', 'EUR', 'GBP'], \
...                  ['USD', 1.0,  0.91, 0.79], \
...                  ['EUR', 1.1,  1.0,  0.87], \
...                  ['GBP', 1.26, 1.14, 1.0]])
>>> rates = pd.DataFrame(data[1:,1:], \
... columns=currencies, index=currencies)
>>> rates
      USD   EUR   GBP
USD   1.0  0.91  0.79
EUR   1.1   1.0  0.87
GBP  1.26  1.14   1.0
>>>
```

a DataFrame from a csv file

If the file `currencies.csv` contains

```
 ,USD,EUR,GBP
USD,1.0,0.91,0.79
EUR,1.1,1.0,0.87
GBP,1.26,1.14,1.0
```

then we can make a DataFrame as follows:

```
>>> import pandas as pd
>>> d = pd.read_csv('currencies.csv')
>>> d
          USD   EUR   GBP
0   USD  1.00  0.91  0.79
1   EUR  1.10  1.00  0.87
2   GBP  1.26  1.14  1.00
```

session continued

```
>>> d
          USD    EUR    GBP
0   USD   1.00   0.91   0.79
1   EUR   1.10   1.00   0.87
2   GBP   1.26   1.14   1.00
>>> d.values
array([['USD', 1.0, 0.91, 0.79],
       ['EUR', 1.1, 1.0, 0.87],
       ['GBP', 1.26, 1.14, 1.0]], dtype=object)
>>> d.columns
Index([' ', 'USD', 'EUR', 'GBP'], dtype='object')
>>> d.index
RangeIndex(start=0, stop=3, step=1)
```

adding columns and rows to a DataFrame

We want to extend the table

	USD	EUR	GBP
USD	1.0	0.91	0.79
EUR	1.1	1.0	0.87
GBP	1.26	1.14	1.0

with another currency, the Japanese Yen:

	USD	EUR	GBP	YEN
USD	1.0	0.91	0.79	108.37
EUR	1.1	1.0	0.87	119.49
GBP	1.26	1.14	1.0	136.59
YEN	0.0092	0.0084	0.0073	1.0

adding a column to a DataFrame

```
>>> currencies
['USD', 'EUR', 'GBP']
>>> rates
      USD   EUR   GBP
USD   1.0  0.91  0.79
EUR   1.1   1.0  0.87
GBP  1.26  1.14   1.0
>>> yen_data = np.array([108.37, 119.49, 136.59])
>>> rates['YEN'] = pd.Series(yen_data, \
...     index=currencies)
>>> rates
      USD   EUR   GBP      YEN
USD   1.0  0.91  0.79   108.37
EUR   1.1   1.0  0.87   119.49
GBP  1.26  1.14   1.0   136.59
>>>
```

adding a row to a DataFrame

```
>>> currencies = currencies + ['YEN']
>>> yen_row = pd.DataFrame([[0.0092, 0.0084, 0.0073, \
... 1.0]], columns=currencies, index=['YEN'])
>>> yen_row
        USD     EUR     GBP  YEN
YEN  0.0092  0.0084  0.0073  1.0
>>> rates = rates.append(yen_row)
>>> rates
        USD     EUR     GBP     YEN
USD     1.0    0.91    0.79  108.37
EUR     1.1     1.0    0.87  119.49
GBP    1.26    1.14     1.0  136.59
YEN  0.0092  0.0084  0.0073    1.00
>>>
```

data analysis with pandas

1. Series and DataFrames
   - pandas for data analysis
   - examples of the data structures
   - making DataFrames
2. An Application
   - analyzing reviews from video games
   - asking questions about the data
3. Visualization
   - making histograms with matplotlib in ipython

analyzing reviews from video games

We follow a tutorial at
`https://www.dataquest.io/blog/pandas-python-tutorial`

This tutorial uses pandas to analyze video game reviews from IGN,
a popular video game review site,
using data scraped by Eric Grinstein.

The data can be downloaded from
`https://www.dataquest.io/wp-content/uploads/2019/09/ign.csv`

importing the data

```
>>> import pandas as pd
>>> reviews = pd.read_csv("ign.csv")
>>> reviews.head()
   Unnamed: 0 score_phrase  ... release_month release_day
0           0      Amazing  ...             9          12
1           1      Amazing  ...             9          12
2           2        Great  ...             9          12
3           3        Great  ...             9          11
4           4        Great  ...             9          11

[5 rows x 11 columns]
```

It looks as we have some valid data.

examining the data

```
>>> reviews.size
204875
>>> reviews.shape
(18625, 11)
>>> reviews.columns
Index(['Unnamed: 0', 'score_phrase', 'title', 'url',
       'platform', 'score', 'genre', 'editors_choice',
       'release_year', 'release_month', 'release_day'],
      dtype='object')
>>> reviews.index
RangeIndex(start=0, stop=18625, step=1)
>>>
```

Similar to a spreadsheet, the data types of the columns differs.

data analysis with pandas

1. Series and DataFrames
   - pandas for data analysis
   - examples of the data structures
   - making DataFrames
2. An Application
   - analyzing reviews from video games
   - asking questions about the data
3. Visualization
   - making histograms with matplotlib in ipython

some math questions

What is the numerical range of the scores?

```
>>> scores = reviews['score']
>>> type(scores)
<class 'pandas.core.series.Series'>
>>> scores.head
<bound method NDFrame.head of 0          9.0
1          9.0
2          8.5
3          8.5
4          8.5
          ...
18620      7.6
18621      9.0
18622      5.8
18623     10.0
18624     10.0
Name: score, Length: 18625, dtype: float64>
```

minimum, maximum, and average

Let us see what is the range of the scores.

```
>>> min(scores)
0.5
>>> max(scores)
10.0
>>> sum(scores)
129452.29999999967
>>> len(scores)
18625
>>> sum(scores)/len(scores)
6.950459060402666
>>> scores.mean()
6.950459060402685
```

filtering scores

How many scores are higher than seven?

```
>>> above7 = scores > 7
>>> filtered = scores[above7]
>>> filtered.head()
0    9.0
1    9.0
2    8.5
3    8.5
4    8.5
Name: score, dtype: float64
>>> filtered.size
9800
```

filtering the reviews

Suppose we are only interested in positive reviews ...

```
>>> filteredreviews = reviews[above7]
>>> filteredreviews.head()
   Unnamed: 0 score_phrase  ... release_month release_day
0           0      Amazing  ...             9          12
1           1      Amazing  ...             9          12
2           2        Great  ...             9          12
3           3        Great  ...             9          11
4           4        Great  ...             9          11

[5 rows x 11 columns]
>>> filteredreviews.shape
(9800, 11)
```

the `mean()` on the DataFrame

We can ask for the mean on all scores,
but also on the entire DataFrame:

```
>>> reviews.mean()
Unnamed: 0        9312.000000
score                6.950459
release_year      2006.515329
release_month        7.138470
release_day         15.603866
dtype: float64
```

Here is an interesting question:
how do the scores correlate with the release year?

computing correlations

How do the scores correlate with the release year?

```
>>> reviews.corr()
              Unnamed: 0   score release_year release_month release_day
Unnamed: 0        1.0000  0.0356     0.893394     -0.096676    0.010068
score             0.0356  1.0000     0.062716      0.007632    0.020079
release_year      0.8934  0.0627     1.000000     -0.115515    0.016867
release_month    -0.0967  0.0076    -0.115515      1.000000   -0.067964
release_day       0.0100  0.0201     0.016867     -0.067964    1.000000
```

data analysis with pandas

1. Series and DataFrames
   - pandas for data analysis
   - examples of the data structures
   - making DataFrames
2. An Application
   - analyzing reviews from video games
   - asking questions about the data
3. Visualization
   - making histograms with matplotlib in ipython

matplotlib in ipython

The SciPy stack:
1. numpy, 2. scipy, 3. matplotlib, 4. ipython, 5. pandas.

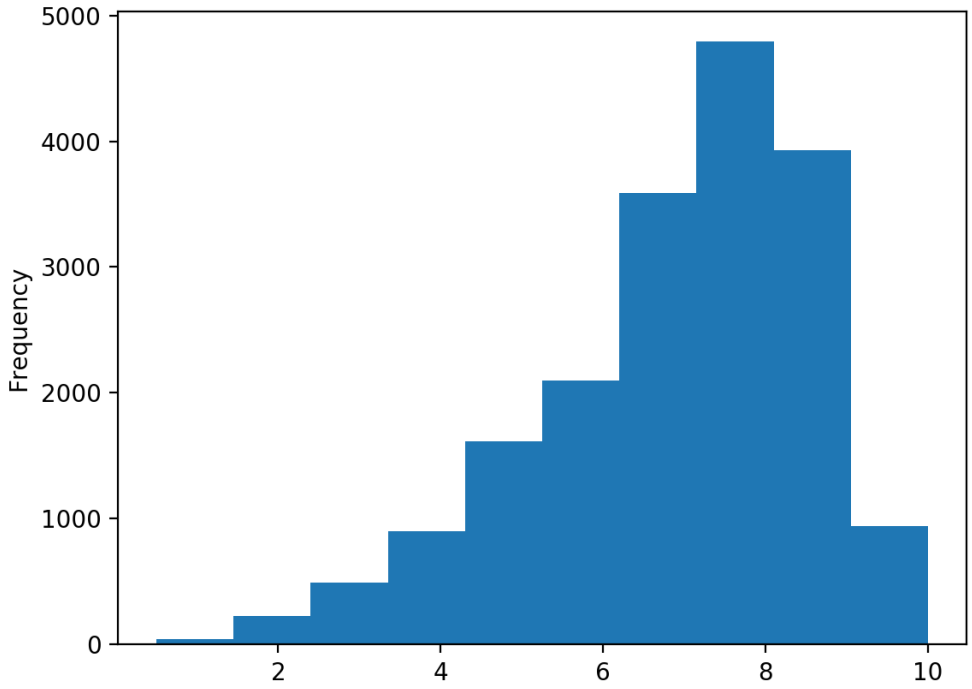
```
$ ipython --pylab
Python 3.7.3 (default, Mar 27 2019, 16:54:48)
Type 'copyright', 'credits' or 'license' for more info
IPython 7.8.0 -- An enhanced Interactive Python. Type
Using matplotlib backend: MacOSX

In [1]: x = linspace(0, 2*pi, 100)

In [2]: y = sin(x)

In [3]: plot(x,y)
Out[3]: [<matplotlib.lines.Line2D at 0x11b1c1278>]
```

pandas in ipython

```
In [4]: import pandas as pd

In [5]: reviews = pd.read_csv("ign.csv")

In [6]: reviews.columns
Out[6]:
Index(['Unnamed: 0', 'score_phrase', 'title', 'url',
       'platform', 'score', 'genre', 'editors_choice',
       'release_year', 'release_month', 'release_day'],
      dtype='object')

In [7]: scores = reviews['score']
```

plotting a histogram of the scores

```
In [8]: scores.plot(kind='hist')
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x11
```



an exercise

From `https://grouplens.org/datasets/movielens/latest/` download the MovieLens Latest Datasets, preferably the full dataset, but if 256MB is too large, you may work with the smaller 1MB dataset. Use pandas to explore the data.

1. Load the file `ratings.csv` into a DataFrame and select the column `ratings`. Use matplotlib to make a histogram of the ratings.
2. Load the the file `movies.csv` into a DataFrame and consider the column `genres`.
   For each movie, the genres are listed, separated by |. Write a Python script to traverse all rows in the DataFrame and to separate the genres. A movie like Toy Story that has five genres (Adventure, Animation, Children, Comedy, and Fantasy) will be counted five times in the frequency table of genres. The Python script prints the dictionary with the frequencies for each genre.
3. Combining movies and ratings,
   for each genre find the movie with the highest rating.