Data Analysis&Visualization

Data analysis applications

Objective of the Course is to introduce:

- Analysis on unstructured data
- Data Cleaning, transformation and integration
- Data visualization techniques and applications

Course Plan

Week	Topics
1	Data analysis applications
2	Data Structures in Python
3	String objects and regular expressions for pattern search
4	Class, file management and important modules in Python
5	Case-study on unstructured data
6	Numpy array operations
7	Data Cleaning and Filtering with PANDAS
8	Data merging, joining and reshaping with PANDAS
9	Data aggregations and group operations with PANDAS
10	Time series analysis with PANDAS
11	Storytelling and visual communication
12	Application of common plot types
13	Plotting Multiple Graphs, Subplots and Figures
14	Customizing color and styles

Grading

Faaliyetler (Activities)	Adedi* (Quantity)	Değerlendirmedeki Katkısı, % (Effects on Grading, %)
Yıl İçi Sınavları (Midterm Exams)	1	30
Ödevler (Homework)	2	30
Final Sınavı (Final Exam)	1	40

Homeworks and exams will be take-home exam

In case of cheating etc., in class exams will be performed !!!

References

- **References**
- Wes McKinney, Python for Data Analysis, O'Reilly Media, Inc., 2nd Edition 2017
- Srinivasa Rao Poladi, Matplotlib 3.0 Cookbook, Pact Publishing, 2018
- Mark Lutz, Learning Python, O'Reilly Media, Inc., 5th Edition 2013
- Brian Heinold, A Practical Introduction toPython Programming, 2012

Audience for this class

- The target audience is scientists, educators, statisticians, financial analysts, and the rest of the non-programmers who want to effectively perform data analysis in Python.

- Prior knowledge on python is not required.

- This is a hands-on class: Most of the lectures will be on computers

Data types

- Structured data:
- Multidimensional arrays (matrices)
- Tabular or spreadsheet-like data in which each column may be a different type (string, numeric, date, or otherwise).
- Evenly or unevenly spaced time series
- Unstructured data can be transformed into a structured form that is more suitable for analysis and modeling.

Why Do People Use Python for data analysis?

- Among interpreted languages Python is distinguished by its large and active scientific computing community.
- Adoption of Python for scientific computing in both industry applications and academic research has increased significantly since the early 2000s.
- In recent years, Python's improved library support (scikit-learn, matplotlib, pandas etc.) has made it a strong alternative for data manipulation task.
- Python code is designed to be readable, and hence reusable and maintainable—much more so than traditional scripting languages.
- Python makes text manipulation extremely easy.
- A large number of people and organizations use Python, so there's ample development and documentation.

http://www.datasciencecentral.com/profiles/blogs/python-overtakes-r-for-data-science-and-machine-learning

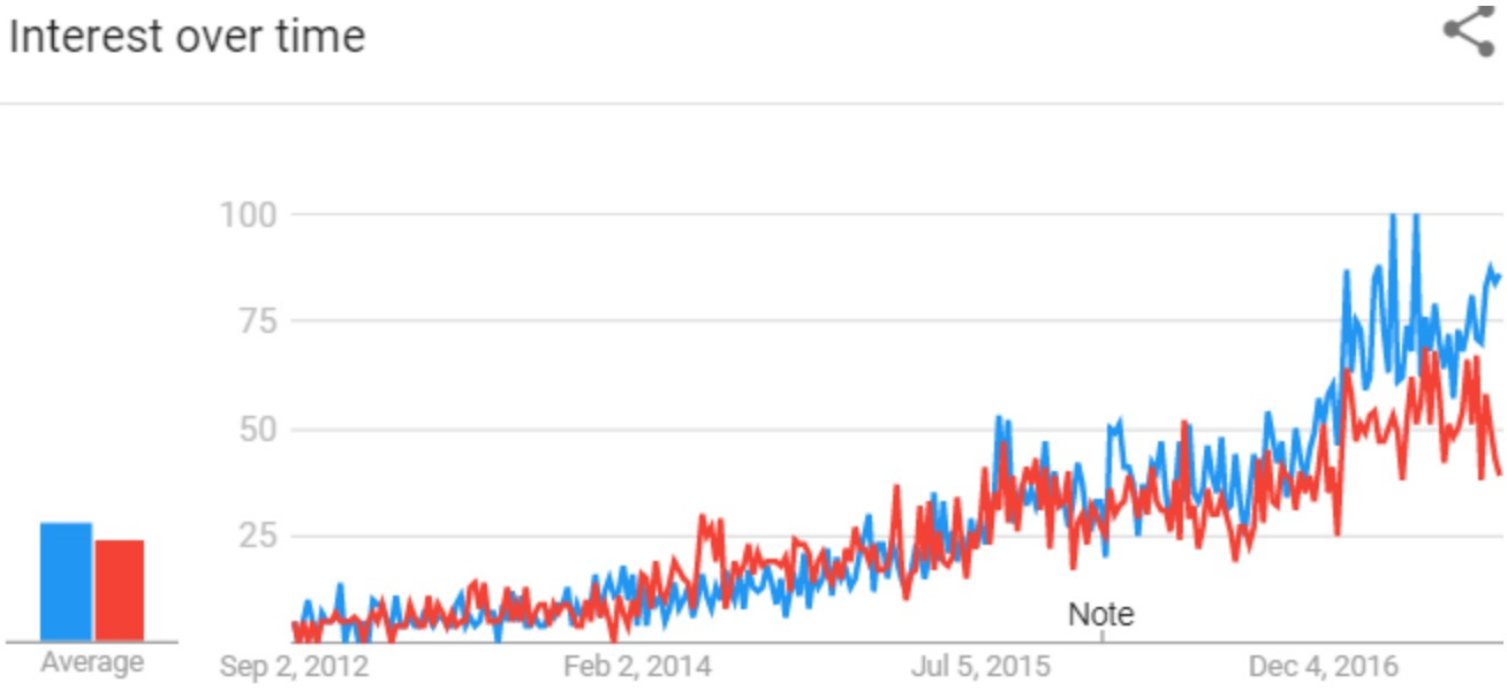
Python Overtakes R for Data Science and Machine Learning

Posted by Vincent Granville on August 30, 2017 at 10:00am View Blog

This article summarizes a trend in programming languages usage, based on a number of proxy metrics. This change started to be more pronounced in early 2017: Python became the language of choice, over R, for data science and machine learning applications.

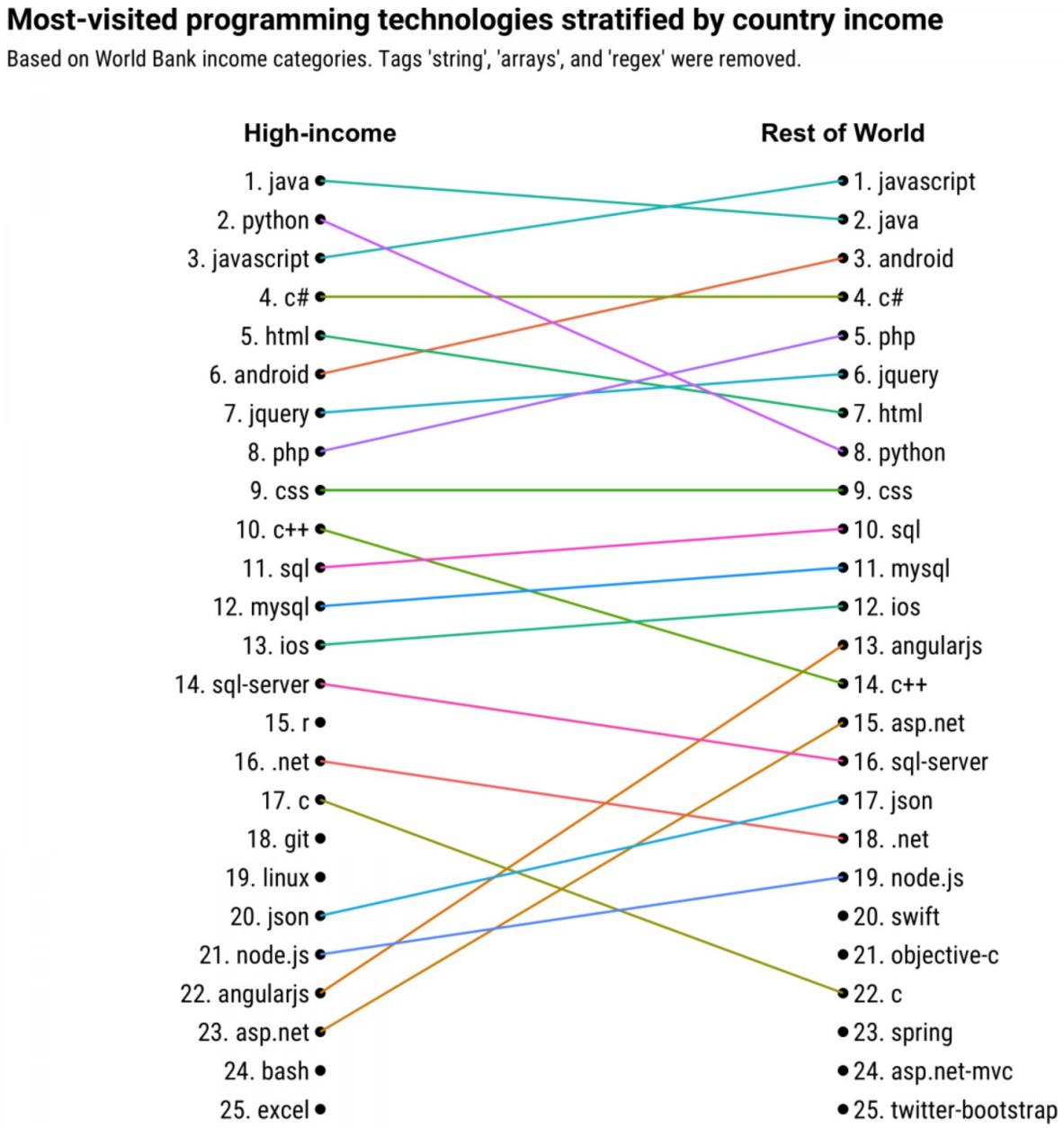
Statistics from Google

Google has one app called Google Trend to find out trends about specific subjects, to compare interest for a number of search topics, broken down by region or time period.

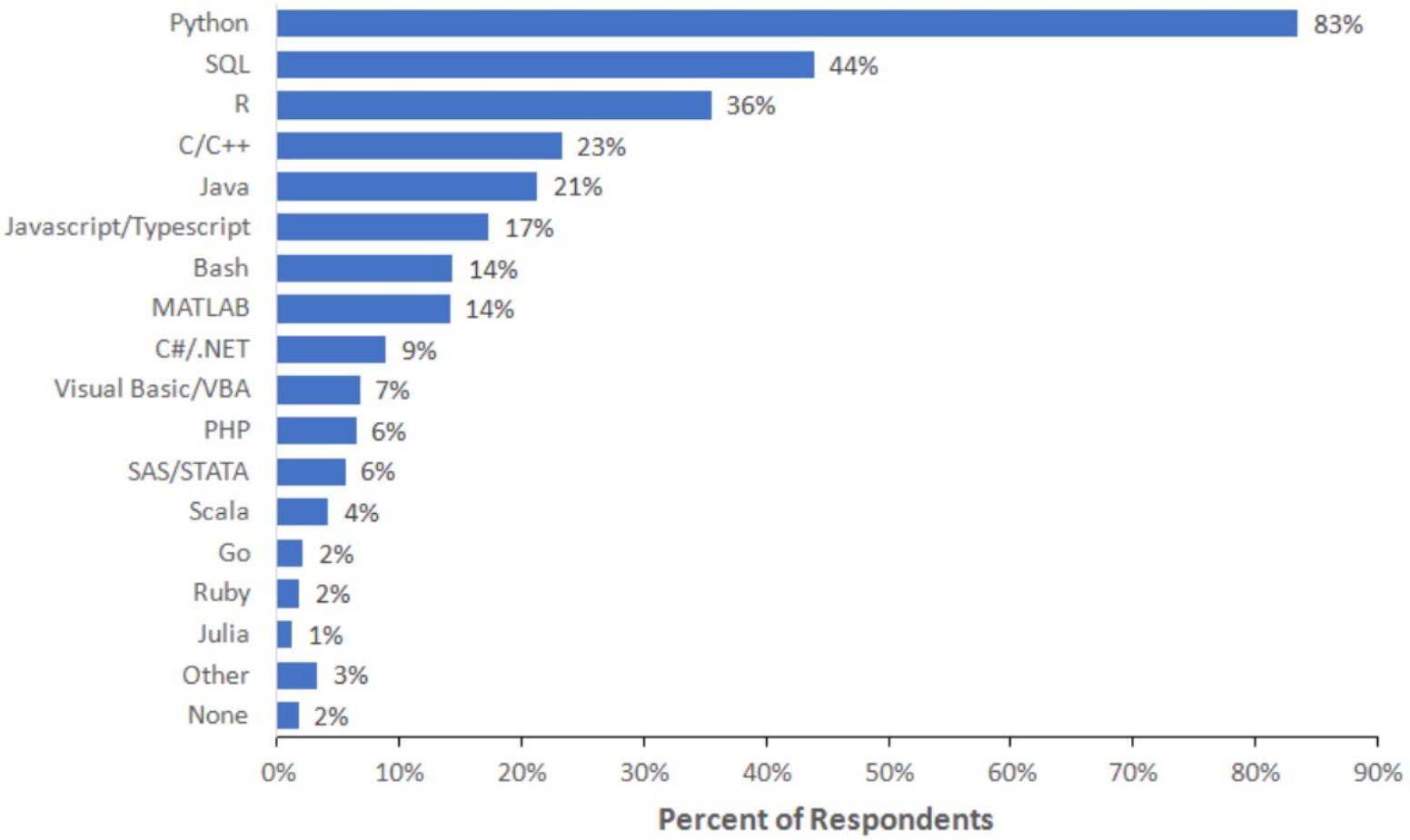


Search index for Python Data Science *(blue) versus* R Data Science *(red) over the last 5 years, in US*

https://stackoverflow.blog/2017/08/29/tale-two-industries-programming-languages-differ-wealthy-developing-countries/



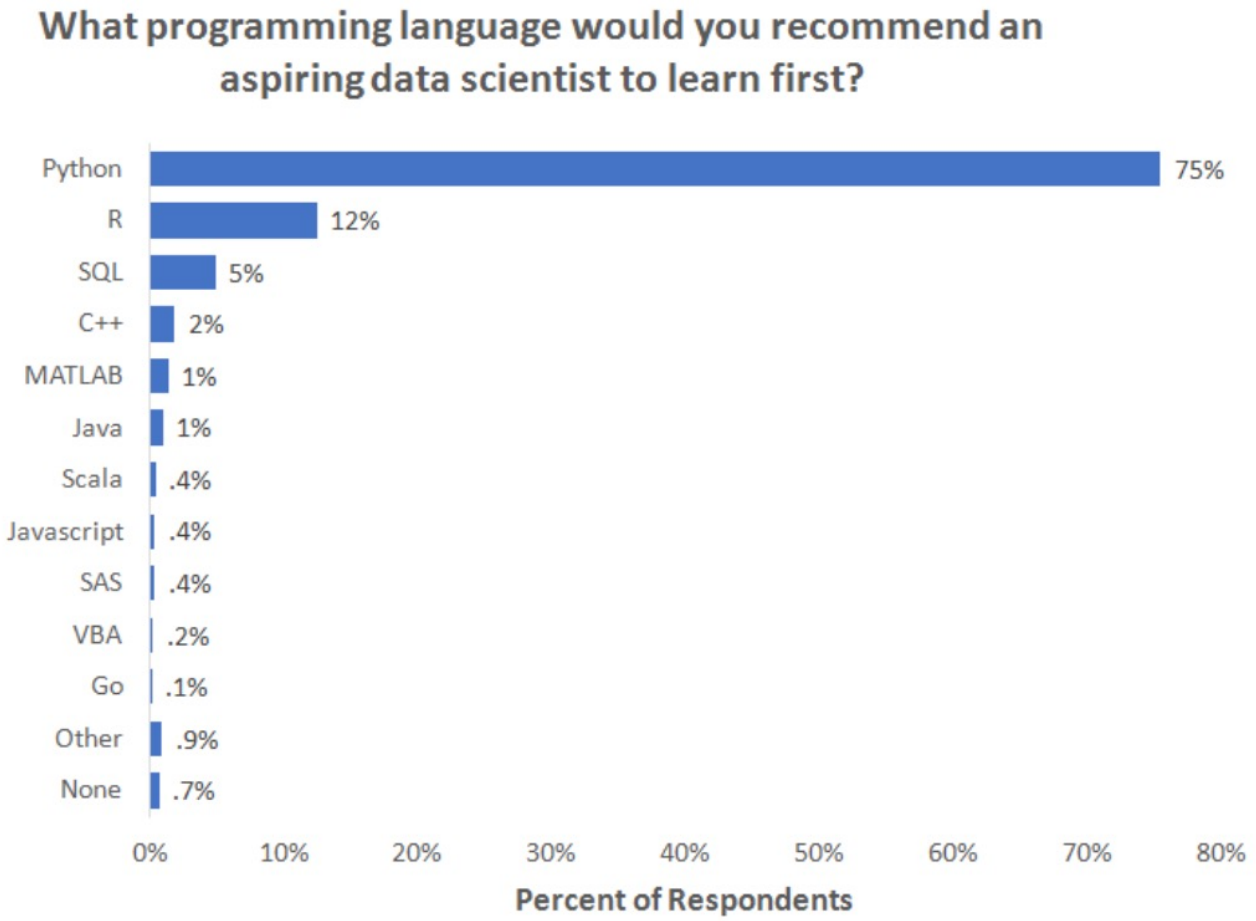
What programming language do you use on a regular basis?



Note: Data are from the 2018 Kaggle Machine Learning and Data Science Survey. You can learn more about the study here: http://www.kaggle.com/kaggle/kaggle-survey-2018. A total of 18827 respondents answered the question.

BUSINESS BROADWAY

https://www.kaggle.com/kaggle/kaggle-survey-2018



Note: Data are from the 2018 Kaggle ML and Data Science Survey. You can learn more about the study here: http://www.kaggle.com/kaggle/kaggle-survey-2018.
A total of 23859 respondents completed the survey; the percentages in the graph are based on a total of 18788 respondents who provided an answer to this question.

BUSINESS BROADWAY
DATA SCIENCE | CUSTOMER ANALYTICS | MACHINE LEARNING

Essential Python libraries

- **NumPy**
- NumPy, short for Numerical Python, is the foundational package for scientific computing in Python. It provides, among other things:
- A fast and efficient multidimensional array object ndarray
- Functions for performing element-wise computations with arrays or mathematical operations between arrays
- Tools for reading and writing array-based data sets to disk
- Linear algebra operations, Fourier transform, and random number generation
- Tools for integrating connecting C, C++, and Fortran code to Python
- For numerical data, NumPy arrays are a much more efficient way of storing and manipulating data than the other built-in Python data structures.

Essential Python libraries

- **Pandas**
- Pandas provides high-performance data structures and data analysis tools.
- Fast and efficient DataFrame object for data manipulation with integrated indexing
- The DataFrame structure can be seen as a spreadsheet which offers very flexible ways of working with it.
- You can easily transform any dataset in the way you want, by reshaping it and adding or removing columns or rows.
- Pandas also has tools for importing and exporting data from different formats: comma-separated value (CSV), text files, Microsoft Excel, SQL databases, and the fast HDF5 formats.
- In many situations, the data you have in such formats will not be complete or totally structured.
- For such cases, Pandas offers handling of **missing data** and intelligent data alignments.

Essential Python libraries

- **Matplotlib**
- matplotlib is the most popular Python library for producing plots and other 2D data visualizations.

- **Scikit-learn**
- Scikit-learn offers quite powerful tools for machine learning.

Scikit-learn: **Machine learning** in Python
F Pedregosa, G Varoquaux, A Gramfort… - … of **Machine Learning** …, 2011 - jmlr.org
... This answers the growing need for statistical data analysis by non-specialists in the software and web industries, as well as in fields outside of computer-science, such as **biology** or physics. Scikit-learn differs from other **machine learning** toolboxes in Python for various reasons: i **...**
Alıntılanma sayısı: 6831 İlgili makaleler 28 sürümün hepsi Web of Science: 2359 Alıntı yap Kaydet Diğer

Integrated Development Environments (IDEs)

- For any programmer, and by extension, for any data scientist, the integrated development environment (IDE) is an essential tool.
- IDEs are designed to maximize programmer productivity.
- Here are some:
- Eclipse with PyDev Plugin
- Python Tools for Visual Studio (for Windows users)
- PyCharm
- **Spyder**
- Komodo IDE

Web Integrated Development Environment (WIDE): Jupyter

- With the advent of web applications, a new generation of IDEs for interactive languages such as Python has been developed.
- Jupyter (for Julia, Python and R) aims to reuse the same WIDE for all these interpreted languages and not just Python
- http://jupyter.org/

jupyter Welcome to Python (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Python 3

Markdown CellToolbar

In [ ]:
```
%matplotlib notebook

import pandas as pd
import numpy as np
import matplotlib

from matplotlib import pyplot as plt
import seaborn as sns

ts = pd.Series(np.random.randn(1000), index=pd.date_range('1/1/2000', periods=1000))
ts = ts.cumsum()

df = pd.DataFrame(np.random.randn(1000, 4), index=ts.index,
                  columns=['A', 'B', 'C', 'D'])
df = df.cumsum()
df.plot(); plt.legend(loc='best')
```

Feel free to open new cells using the plus button (+), or hitting shift-enter while this cell is selected.

Behind the scenes, the software that powers this is tmpnb, a Tornado application that spawns pre-built Docker containers and then uses the jupyter/configurable-http-proxy to put your notebook server on a unique path.

jupyter Welcome to Python (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Python 3

Markdown CellToolbar

https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1007007

PLOS COMPUTATIONAL BIOLOGY

EDITORIAL

Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks

Adam Rule[1], Amanda Birmingham[2], Cristal Zuniga[3], Ilkay Altintas[4], Shih-Cheng Huang[4¤], Rob Knight[3,5], Niema Moshiri[6], Mai H. Nguyen[4], Sara Brin Rosenthal[2], Fernando Pérez[7], Peter W. Rose[4]*

Spyder – Documentation

Spyder is the Scientific PYthon Development EnviRonment:

- a powerful interactive development environment for the Python language with advanced editing, interactive testing, debugging and introspection features
- and a numerical computing environment thanks to the support of *IPython* (enhanced interactive Python interpreter) and popular Python libraries such as *NumPy* (linear algebra), *SciPy* (signal and image processing) or *matplotlib* (interactive 2D/3D plotting).

Installation

Spyder is quite easy to install on Windows, Linux and MacOS X. Just the read the following instructions with care.

Installing on Windows Vista/7/8/10

The easy way

Spyder is already included in these *Python Scientific Distributions*:

1. Anaconda
2. WinPython
3. Python(x,y)

Installing spyder through Anaconda is suggested

https://www.anaconda.com/distribution/

Download python3.7

Ipython

- A powerful interactive Python interpreter

```
In [9]: def myfunc(x):
   ...:     print(x*2)
   ...:

In [10]: myfunc(4)
8
```

```
In [3]: print("hello Ipython")
hello Ipython

In [4]: 30*2
Out[4]: 60
```

- Depending on the exact command you are typing you might realize that sometimes Enter will add a new line, and sometimes it will execute the current statement.
- IPython tries to guess what you are doing, so most of the time you should not have to care.

Ipython

- ? Command provides Ipython features
- %quickref

```
In [20]: %quickref

IPython -- An enhanced Interactive Python - Quick Reference Card
================================================================

obj?, obj??      : Get help, or more help for object (also works as
                   ?obj, ??obj).
?foo.*abc*       : List names in 'foo' containing 'abc' in them.
%magic           : Information about IPython's 'magic' % functions.

Magic functions are prefixed by % or %%, and typically take their arguments
without parentheses, quotes or even commas for convenience.  Line magics take a
single % and cell magics are prefixed with two %%.

Example magic function calls:

%alias d ls -F   : 'd' is now an alias for 'ls -F'
alias d ls -F    : Works if 'alias' not a python name
alist = %alias   : Get list of aliases to 'alist'
cd /usr/share    : Obvious. cd -<tab> to choose from visited dirs.
%cd??            : See help AND source for magic %cd
%timeit x=10     : time the 'x=10' statement with high precision.
%%timeit x=2**100
x**100           : time 'x**100' with a setup of 'x=2**100'; setup code is not
                   counted.  This is an example of a cell magic.
```

Ipython: Introspection

- Using a question mark (?) before or after a variable will display some general information about the object:

```
In [30]: b=[1,2,4,5]

In [31]: b?
Type:        list
String form: [1, 2, 4, 5]
Length:      4
Docstring:
list() -> new empty list
list(iterable) -> new list initialized from iterable's items

In [32]: names1880?
Type:        DataFrame
String form:
name sex  births
           0          Mary   F    7065
           1          Anna   F    2604
           2          Em <...> ight   M       5
           1998       York   M       5
           1999  Zachariah   M       5

           [2000 rows x 3 columns]
Length:      2000
File:        /opt/local/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/pandas/core/frame.py
Docstring:
Two-dimensional size-mutable, potentially heterogeneous tabular data
structure with labeled axes (rows and columns). Arithmetic operations
align on both row and column labels. Can be thought of as a dict-like
container for Series objects. The primary pandas data structure
```

Ipython: The %run command

- Any file can be run as a Python program inside the environment of your IPython session using the %run command.
- The %run magic command allows you to run any python script and load all of its data directly into the interactive namespace.

Let's create a small script and give name test_script1.py

```
# -*- coding: utf-8 -*-
"""
Created on Fri Sep 22 11:34:20 2017

@author: baday_imac
"""

def myfunc(x):
    print(x*2)

```

```
In [33]: %run test_script1.py

In [34]: myfunc(5)
10
```

Using –t with %run command shows the execution time of the script

```

def myfunc(x):
    print(x*2)

for i in range(0,1000):
    myfunc(i)
    print(i)

```

```
In [41]: %run -t test_script1.py

IPython CPU timings (estimated):
  User   :       0.12 s.
  System :       0.27 s.
Wall time:       0.36 s.
```

Ipython: Magic Commands

- IPython has many special commands, known as "magic" commands, which are designed to facilitate common tasks and enable you to easily control the behavior of the IPython system.

```
In [43]: %timeit range(1000)
255 ns ± 3.24 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)

In [49]: alpha="124"

In [50]: beta=4

In [51]: who
alpha    beta    k
```

```
In [52]: whos
Variable   Type    Data/Info
-----------------------------
alpha      str     124
beta       int     4
k          int     4
```

```
In [88]: %reset

Once deleted, variables cannot be recovered. Proceed (y/[n])? y
```

Ipython: Magic Commands

- To run any command at the system shell, simply prefix it with !, e
- For example: ls command in linux shows the files in a directory. (dir command does the same on windows)
- Mkdir command creates a new folder both in linux and windows command line

```
In [112]: !ls
iris-example.py          names                    test_script1.py
ml-1m                    names-plot.png           untitled0.py
movie-rating-example.py  python_scripts

In [117]: !mkdir newfolder
```

Ipython: Keyboard shortcuts

Command	Description
Ctrl-P or up-arrow	Search backward in command history for commands starting with currently-entered text
Ctrl-N or down-arrow	Search forward in command history for commands starting with currently-entered text
Ctrl-R	Readline-style reverse history search (partial matching)
Ctrl-Shift-V	Paste text from clipboard
Ctrl-C	Interrupt currently-executing code
Ctrl-A	Move cursor to beginning of line
Ctrl-E	Move cursor to end of line
Ctrl-K	Delete text from cursor until end of line
Ctrl-U	Discard all text on current line
Ctrl-F	Move cursor forward one character
Ctrl-B	Move cursor back one character
Ctrl-L	Clear screen

Ipython: Input and Output variables

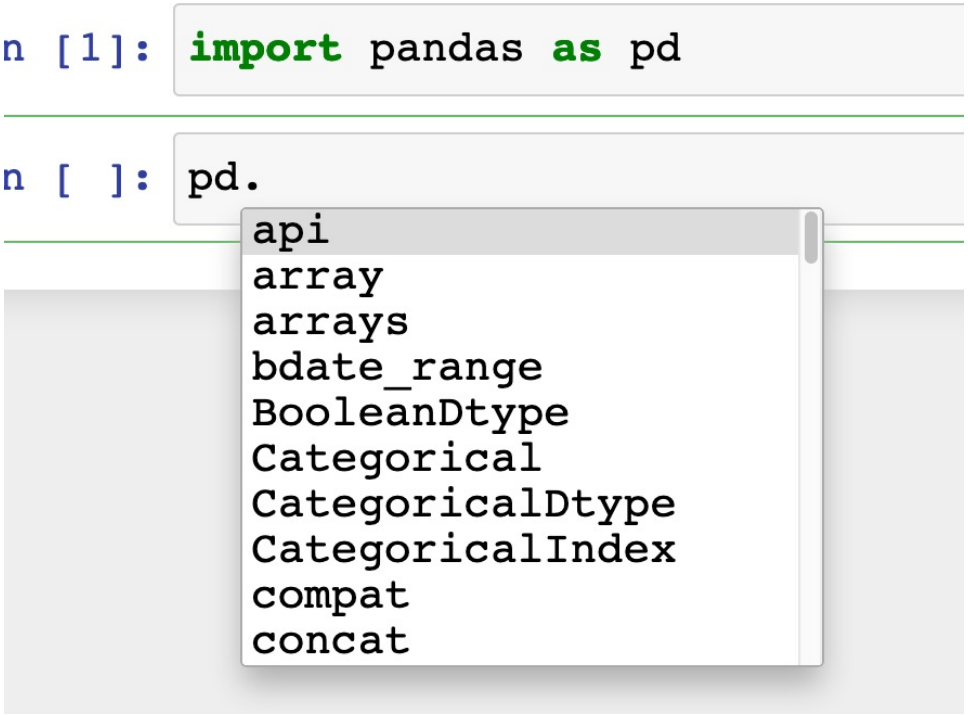
- The previous three outputs are stored in the _ (one underscore) , __ (two underscores) , and ___(three underscore) variables
- Input variables are stored in variables named like _iX, where X is the input line number.

```
In [103]: x=2

In [104]: x*2
Out[104]: 4

In [105]: x*3
Out[105]: 6

In [106]: x*4
Out[106]: 8

In [107]: ___
Out[107]: 4

In [108]: __
Out[108]: 8

In [109]: _
Out[109]: 8

In [110]: a=_

In [111]: print(a)
8
```

```
In [78]: _i72
Out[78]: 'x**2'

In [79]: _i73
Out[79]: 'x*2'
```

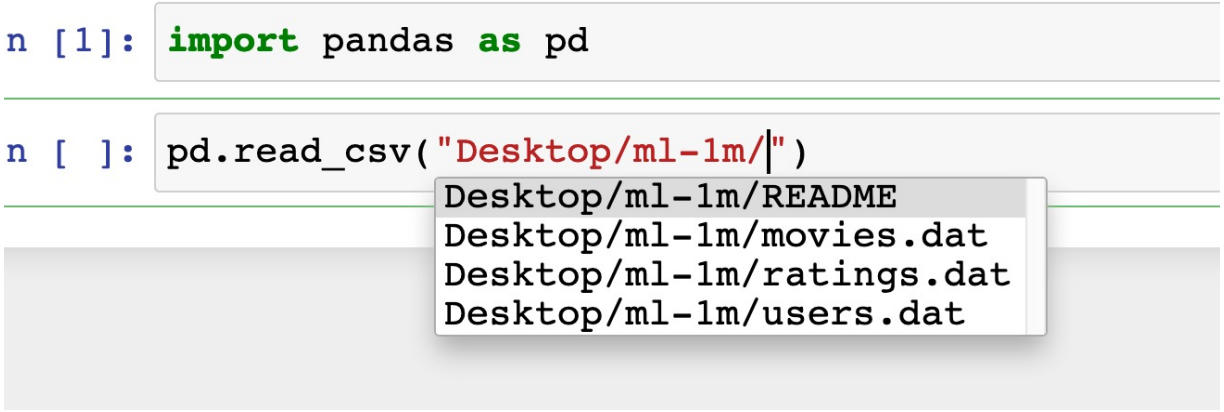
Ipython: Tab Completion

- While entering expressions in the shell, pressing the Tab key will search the namespace for any variables (objects, functions, etc.) matching the characters you have typed so far:



Ipython: Tab Completion

- When typing anything that looks like a file path (even in a Python string), pressing the Tab key will complete anything on your computer's filesystem matching what you've typed:



Ipython: Tab Completion

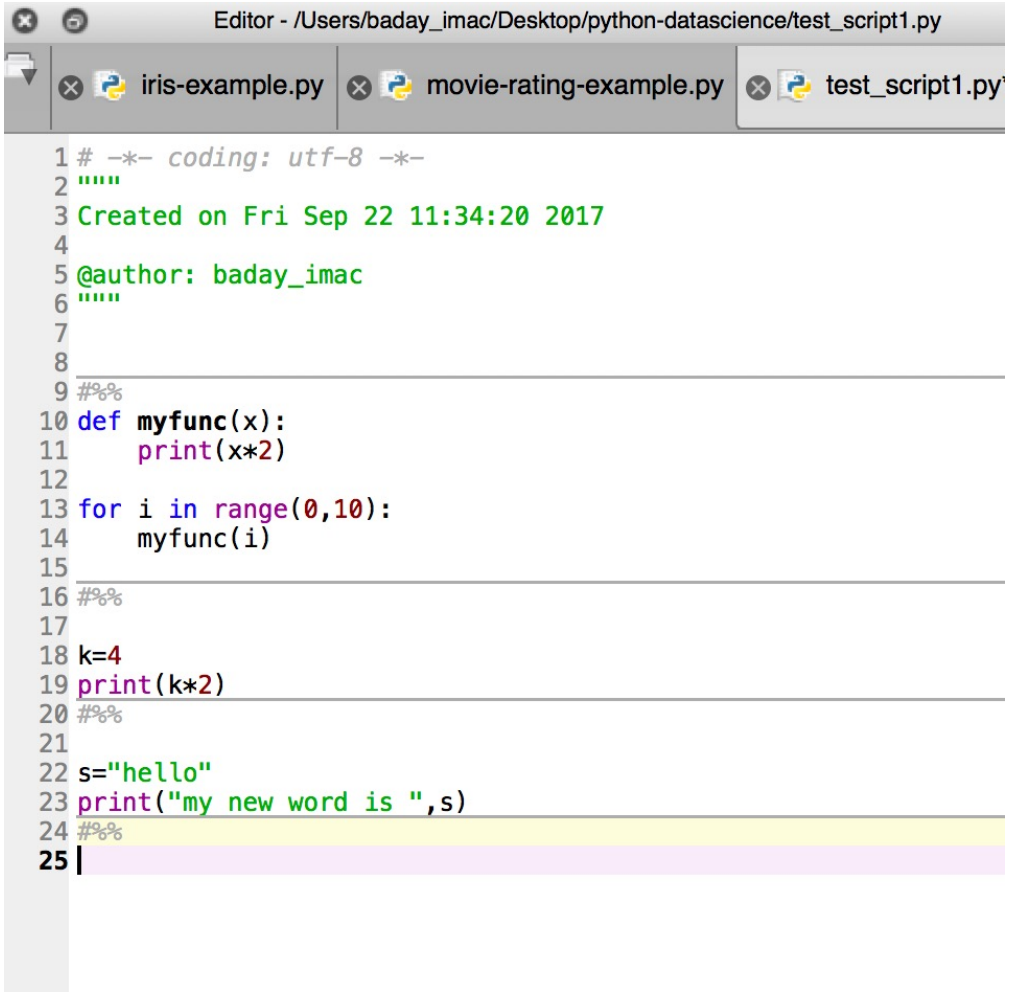
- Inside a function if you press both shit-tab you can see what input arguments you can specify with that function

```
In [1]: import pandas as pd

In [ ]: pd.read_csv("Desktop/ml-1m/")
```

```
Signature:
pd.read_csv(
    filepath_or_buffer: Union[str, pathlib.Path, IO[~AnyStr]],
    sep=',',
    delimiter=None,
    header='infer',
    names=None,
    index_col=None,
    usecols=None,
    squeeze=False,
```

Spyder:Code Cell

- A "code cell" is a concept that block of lines to be executed at once in the current interpreter (Python or IPython).
- Every script may be divided in as many cells as needed.
- Cells are separated by lines starting with: #%%



Spyder

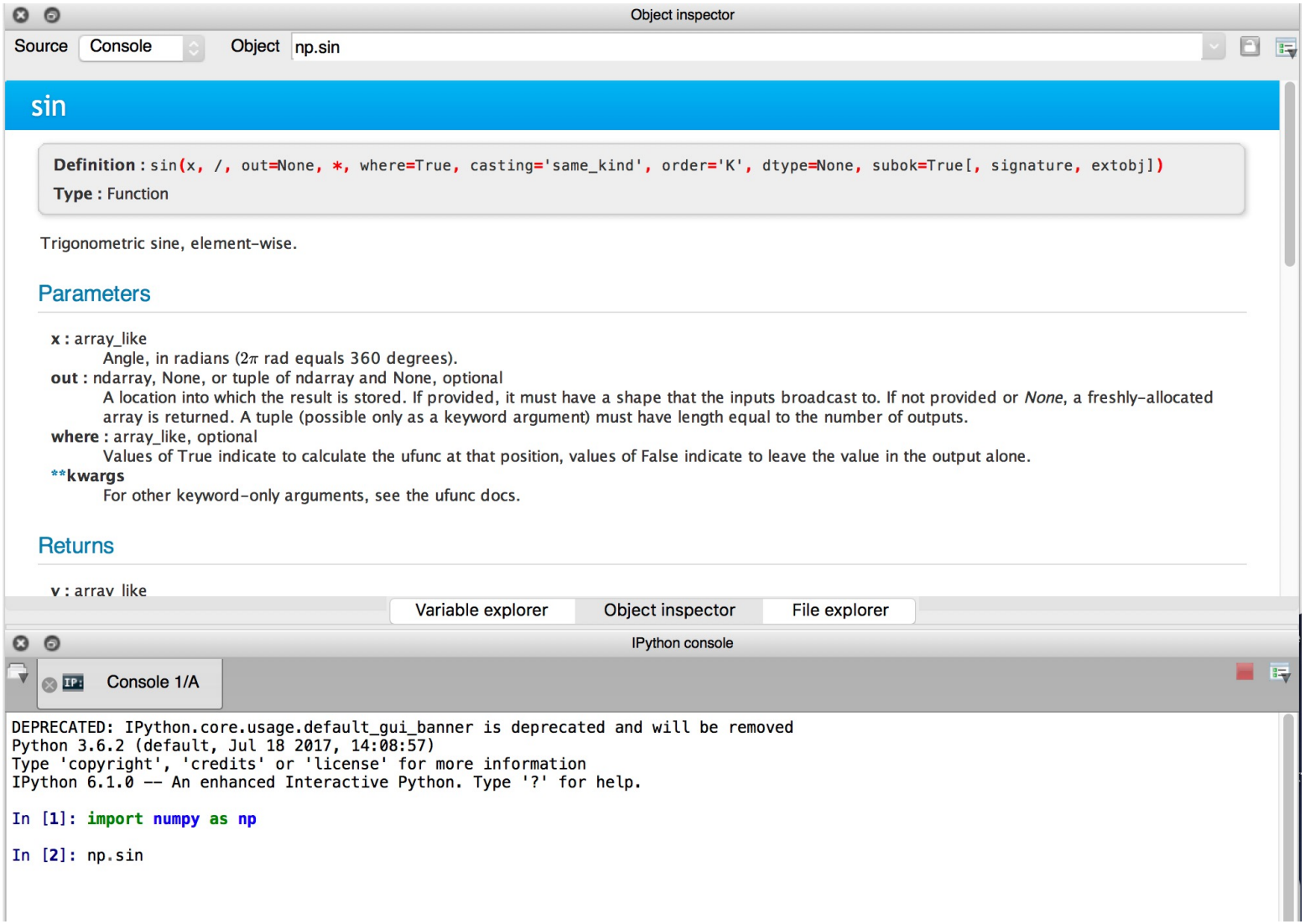
- With variable explorer window you can see the content ant type of variables

Variable explorer

Name	Type	Size	Value
i	int	1	9
k	int	1	4
mnames	list	3	['movie_id', 'title', 'genres']
movies	DataFrame	(3883, 3)	Column names: movie_id, title, genres
ratings	DataFrame	(1000209, 4)	Column names: user_id, movie_id, rating, timestamp
rnames	list	4	['user_id', 'movie_id', 'rating', 'timestamp']
s	str	1	hello
unames	list	5	['user_id', 'gender', 'age', 'occupation', 'zip']
users	DataFrame	(6040, 5)	Column names: user_id, gender, age, occupation, zip

Spyder:Object Inspector

- Press control + I to get documentation for an object



Spyder:Editor code analysis

- Warns the user if any errors in the script

```
24 print( 'my new word is  ',s)
25 #%%
26 import numpy as np
27 x=np.arange(10)
28
29 print x
30
31
32
33
34
```

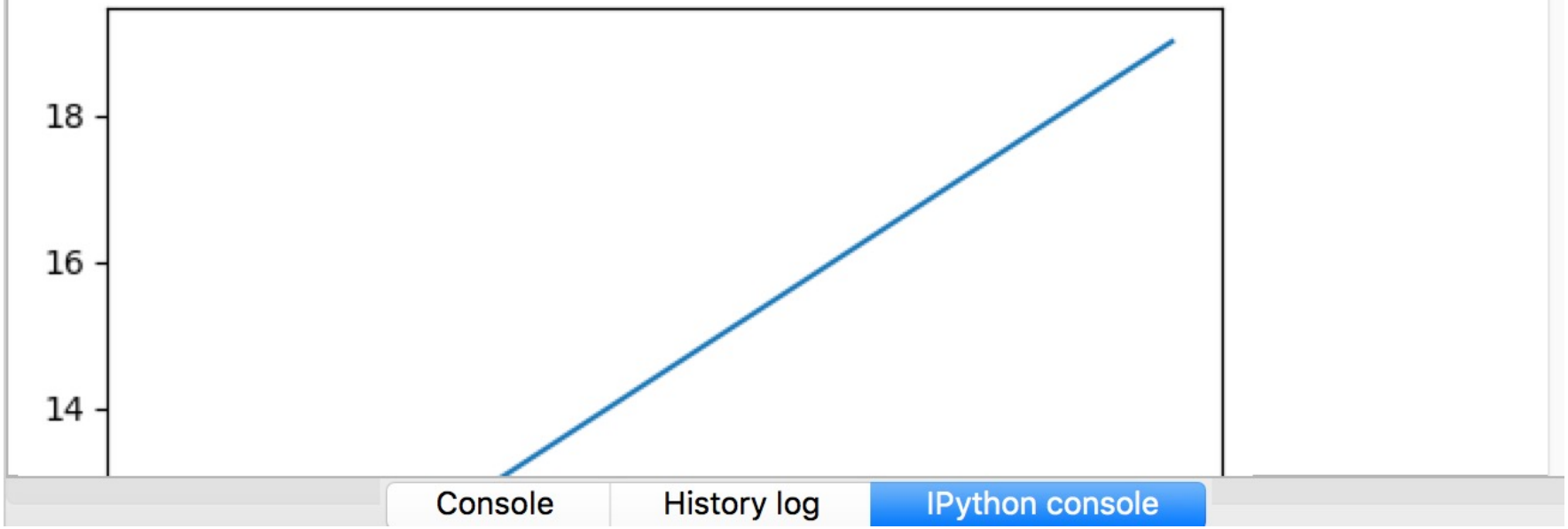
Code analysis

invalid syntax

Spyder:pyplot

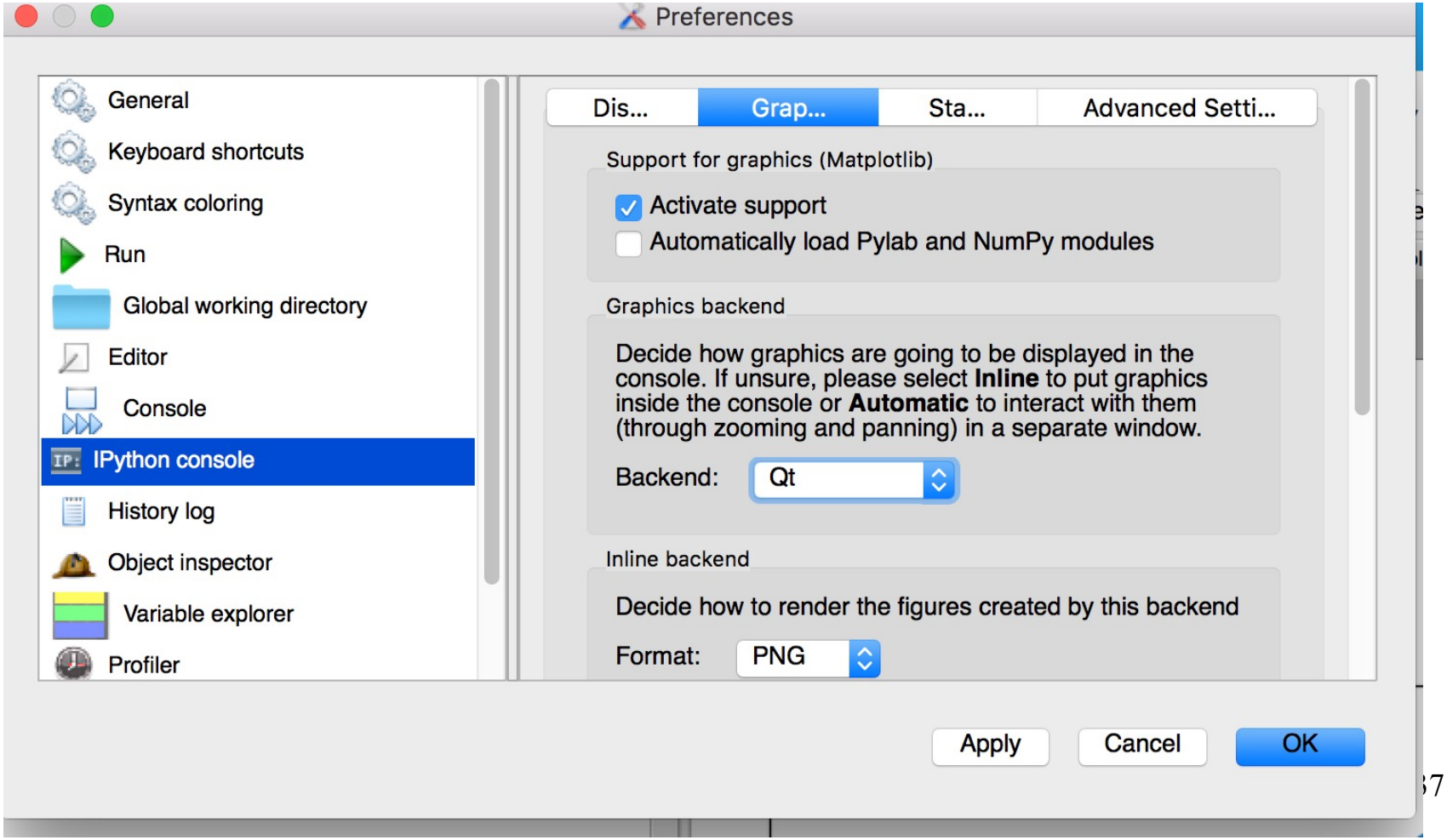
- Plotting inside the console

```
In [25]: import matplotlib.pyplot as plt

In [26]: import numpy as np

In [27]: x=np.arange(0,10)

In [28]: y=np.arange(10,20)

In [29]: plt.plot(x,y)
Out[29]: [<matplotlib.lines.Line2D at 0x1135b2e10>]
```



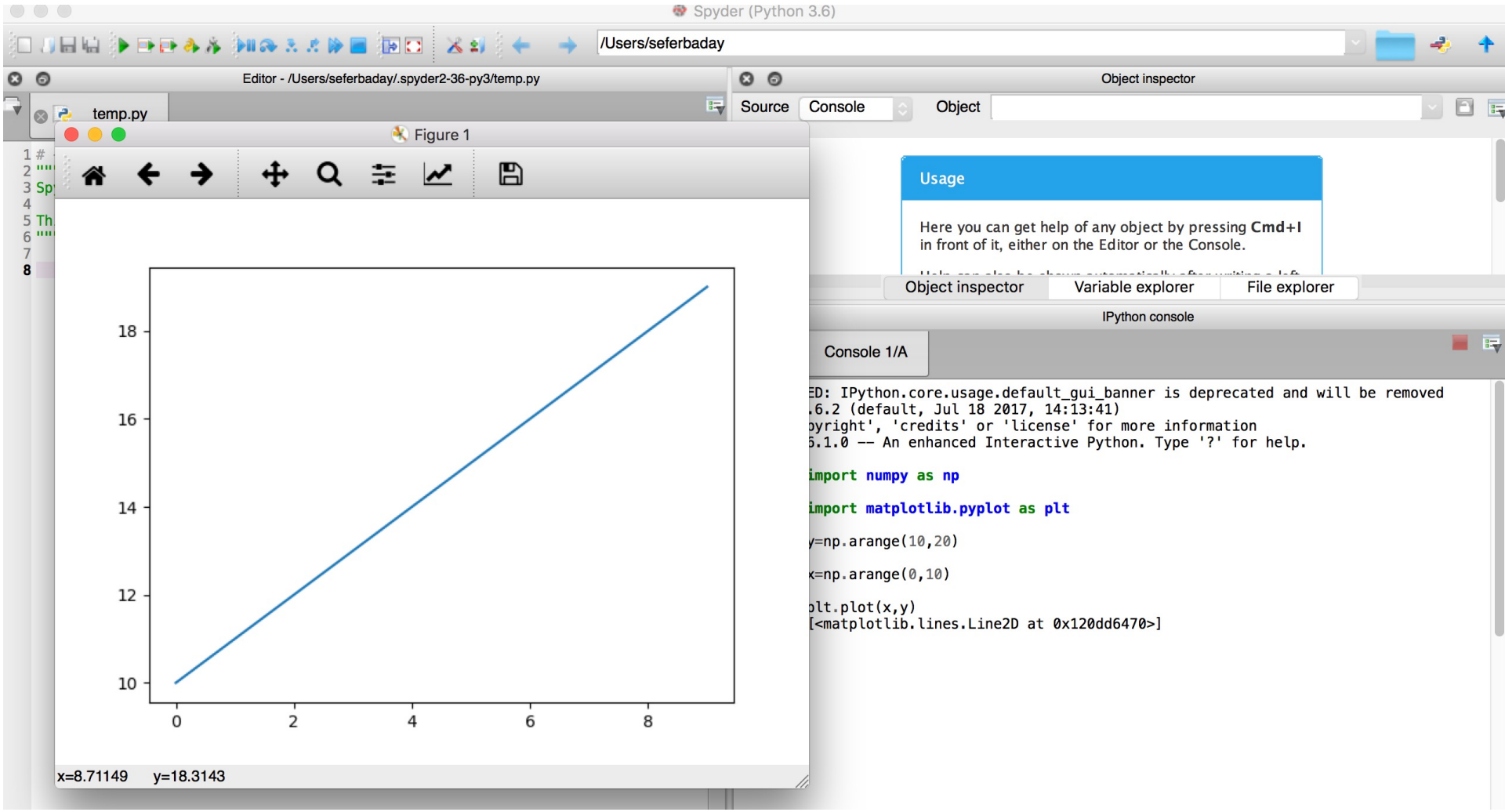
Spyder:pyplot

- To plot in a separate window, go to prefeferences-Ipython console-Graph,
- Change backend option to QT than restart the kernel



Spyder:pyplot

- If you plot the graph again, you will have it in separate window



Python2 vs Python3

- There are currently two different versions of Python: Python 2.X and Python 3.X.
- The differences between the versions are important, so there is no compatibility between the codes, i.e., code written in Python 2.X does not work in Python 3.X and vice versa.
- Much of the scientific community did not change to Python 3.0 immediately and they were stuck with Python 2.7.
- By now, almost all libraries have been ported to Python 3.0; but Python 2.7 is still maintained, so one or another version can be chosen.
- However, those who already have a large amount of code in 2.X rarely change to Python 3.X.
- Through the course we will use Python 3.

Installing or Updating Python Packages

- You can install or update a library using :
  - Anaconda Navigator
  - Command prompt
  - Ipython/Jupyter notebook
- Two major package manager: Conda and Pip
  - In general, these can be installed with the following command:
  - conda install package_name
    If this does not work, you may also be able to install the package using the pip
  - pip install package_name
    You can update packages by using the conda update command:
  - conda update package_name
    pip also supports upgrades using the --upgrade flag:
  - pip install --upgrade package_name

Virtual Environments

- https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html

1. To create an environment:

```
conda create --name myenv
```

> **Note**
>
> Replace `myenv` with the environment name.

2. When conda asks you to proceed, type `y`:

```
proceed ([y]/n)?
```

This creates the myenv environment in `/envs/`. No packages will be installed in this environment.

3. To create an environment with a specific version of Python:

```
conda create -n myenv python=3.6
```

4. To create an environment with a specific package:

```
conda create -n myenv scipy
```

Introductory Example: MovieLens 1M Data Set analysis using PANDAs

- GroupLens Research has collected and made available rating data sets from the MovieLens web site (https://grouplens.org/datasets/movielens/).
- The data sets were collected over various periods of time, depending on the size of the set.
- MovieLens 1M Dataset: collected from users of MovieLens in the late 1990s and early 2000s.
- Stable benchmark dataset. 1 million ratings from 6000 users on 4000 movies. Released 2/2003.
- It's spread across 3 tables: ratings, user information, and movie information. After extracting the data from the zip file, each table can be loaded into a pandas.
- If you download ml-1m.zip file and extract folders you will see:
- Movies.dat, ratings.dat, users.dat and README files.

Introductory Example: MovieLens 1M Data Set analysis using PANDAs

The content of users.dat file

```
1::F::1::10::48067
2::M::56::16::70072
3::M::25::15::55117
4::M::45::7::02460
5::M::25::20::55455
6::F::50::9::55117
7::M::35::1::06810
8::M::25::12::11413
9::M::25::17::61614
10::F::35::1::95370
11::F::25::1::04093
12::M::25::12::32793
13::M::45::1::93304
14::M::35::0::60126
15::M::25::7::22903
16::F::35::0::20670
17::M::50::1::95350
```

```
UserID::Gender::Age::Occupation::Zip-code

All demographic information is provided voluntarily by the users and is
not checked for accuracy.  Only users who have provided some demographic
information are included in this data set.

- Gender is denoted by a "M" for male and "F" for female
- Age is chosen from the following ranges:

        *  1:  "Under 18"
        * 18:  "18-24"
        * 25:  "25-34"
        * 35:  "35-44"
        * 45:  "45-49"
        * 50:  "50-55"
        * 56:  "56+"

- Occupation is chosen from the following choices:

        *  0:  "other" or not specified
        *  1:  "academic/educator"
        *  2:  "artist"
        *  3:  "clerical/admin"
        *  4:  "college/grad student"
        *  5:  "customer service"
        *  6:  "doctor/health care"
        *  7:  "executive/managerial"
        *  8:  "farmer"
        *  9:  "homemaker"
        * 10:  "K-12 student"
        * 11:  "lawyer"
        * 12:  "programmer"
        * 13:  "retired"
        * 14:  "sales/marketing"
        * 15:  "scientist"
        * 16:  "self-employed"
        * 17:  "technician/engineer"
        * 18:  "tradesman/craftsman"
        * 19:  "unemployed"
        * 20:  "writer"
```

Introductory Example: MovieLens 1M Data Set analysis using PANDAs

The content of movies.dat file

```
1::Toy Story (1995)::Animation|Children's|Come
2::Jumanji (1995)::Adventure|Children's|Fantas
3::Grumpier Old Men (1995)::Comedy|Romance
4::Waiting to Exhale (1995)::Comedy|Drama
5::Father of the Bride Part II (1995)::Comedy
6::Heat (1995)::Action|Crime|Thriller
7::Sabrina (1995)::Comedy|Romance
8::Tom and Huck (1995)::Adventure|Children's
9::Sudden Death (1995)::Action
10::GoldenEye (1995)::Action|Adventure|Thrille
11::American President, The (1995)::Comedy|Dra
12::Dracula: Dead and Loving It (1995)::Comedy
13::Balto (1995)::Animation|Children's
14::Nixon (1995)::Drama
15::Cutthroat Island (1995)::Action|Adventure|
16::Casino (1995)::Drama|Thriller
17::Sense and Sensibility (1995)::Drama|Romanc
18::Four Rooms (1995)::Thriller
19::Ace Ventura: When Nature Calls (1995)::Com
20::Money Train (1995)::Action
21::Get Shorty (1995)::Action|Comedy|Drama
22::Copycat (1995)::Crime|Drama|Thriller
23::Assassins (1995)::Thriller
24::Powder (1995)::Drama|Sci-Fi
25::Leaving Las Vegas (1995)::Drama|Romance
26::Othello (1995)::Drama
27::Now and Then (1995)::Drama
```

```
MOVIES FILE DESCRIPTION
================================================================================

Movie information is in the file "movies.dat" and is in the following
format:

MovieID::Title::Genres

- Titles are identical to titles provided by the IMDB (including
year of release)
- Genres are pipe-separated and are selected from the following genres:

        * Action
        * Adventure
        * Animation
        * Children's
        * Comedy
        * Crime
        * Documentary
        * Drama
        * Fantasy
        * Film-Noir
        * Horror
        * Musical
        * Mystery
        * Romance
        * Sci-Fi
        * Thriller
        * War
        * Western

- Some MovieIDs do not correspond to a movie due to accidental duplicate
entries and/or test entries
- Movies are mostly entered by hand, so errors and inconsistencies may exist
```

Introductory Example: MovieLens 1M Data Set analysis using PANDAs

The content of ratings.dat file

```
1::1193::5::978300760
1::661::3::978302109
1::914::3::978301968
1::3408::4::978300275
1::2355::5::978824291
1::1197::3::978302268
1::1287::5::978302039
1::2804::5::978300719
1::594::4::978302268
1::919::4::978301368
1::595::5::978824268
1::938::4::978301752
1::2398::4::978302281
1::2918::4::978302124
1::1035::5::978301753
1::2791::4::978302188
1::2687::3::978824268
1::2018::4::978301777
1::3105::5::978301713
1::2797::4::978302039
1::2321::3::978302205
```

```
RATINGS FILE DESCRIPTION
================================================================================

All ratings are contained in the file "ratings.dat" and are in the
following format:

UserID::MovieID::Rating::Timestamp

- UserIDs range between 1 and 6040
- MovieIDs range between 1 and 3952
- Ratings are made on a 5-star scale (whole-star ratings only)
- Timestamp is represented in seconds since the epoch as returned by time(2)
- Each user has at least 20 ratings
```

Introductory Example: MovieLens 1M Data Set analysis using PANDAs

İmporting the data using pandas

```
import pandas as pd

unames = ['user_id', 'gender', 'age', 'occupation', 'zip']
users = pd.read_table('ml-1m/users.dat', sep='::', header=None,
                      names=unames)

rnames = ['user_id', 'movie_id', 'rating', 'timestamp']
ratings = pd.read_table('ml-1m/ratings.dat', sep='::', header=None,
                        names=rnames)

mnames = ['movie_id', 'title', 'genres']
movies = pd.read_table('ml-1m/movies.dat', sep='::', header=None,
                       names=mnames)
```

You can verify that everything succeeded by looking at the first few rows of each Data Frame with Python's slice syntax:

```
In [334]: users[:5]
Out[334]:
   user_id gender  age  occupation    zip
0        1      F    1          10  48067
1        2      M   56          16  70072
2        3      M   25          15  55117
3        4      M   45           7  02460
4        5      M   25          20  55455

In [335]: ratings[:5]
Out[335]:
   user_id  movie_id  rating  timestamp
0        1      1193       5  978300760
1        1       661       3  978302109
2        1       914       3  978301968
3        1      3408       4  978300275
4        1      2355       5  978824291

In [336]: movies[:5]
Out[336]:
   movie_id                               title                        genres
0         1                    Toy Story (1995)   Animation|Children's|Comedy
1         2                      Jumanji (1995)  Adventure|Children's|Fantasy
2         3             Grumpier Old Men (1995)                Comedy|Romance
3         4            Waiting to Exhale (1995)                  Comedy|Drama
```

Analyzing the data spread across three tables is not a simple task;
for example, suppose you wanted to compute mean ratings for a particular movie by sex and age.
This is much easier to do with all of the data merged together into a single table.
Using pandas's merge function, we first merge ratings with users then merging that result with the movies data.

In [338]: data = pd.merge(pd.merge(ratings, users), movies)

```
In [340]: data.ix[0]
Out[340]:
user_id                                   1
movie_id                                  1
rating                                    5
timestamp                         978824268
gender                                    F
age                                       1
occupation                               10
zip                                   48067
title                      Toy Story (1995)
genres        Animation|Children's|Comedy
Name: 0
```

To get mean movie ratings for each film grouped by gender, we can use the pivot_table method:

```
>>> mean_ratings = data.pivot_table('rating', index='title', columns='gender', aggfunc='mean')
```

This produced another DataFrame containing mean ratings with movie totals as row labels and gender as column labels.

```
>>> mean_ratings[:5]
>>> gender                                F         M
title
$1,000,000 Duck (1971)              3.375000  2.761905
'Night Mother (1986)                3.388889  3.352941
'Til There Was You (1997)           2.675676  2.733333
'burbs, The (1989)                  2.793478  2.962085
...And Justice for All (1979)       3.828571  3.689024
```

Let's filter down to movies that received at least 250 ratings (a completely arbitrary number);
to do this, we group the data by title and use size() to get a Series of group sizes for each title:

```
>>> ratings_by_title = data.groupby('title').size()
>>>
>>> ratings_by_title[:10]
>>> title
$1,000,000 Duck (1971)                37
'Night Mother (1986)                  70
'Til There Was You (1997)             52
'burbs, The (1989)                   303
...And Justice for All (1979)        199
1-900 (1994)                           2
10 Things I Hate About You (1999)    700
101 Dalmatians (1961)                565
101 Dalmatians (1996)                364
12 Angry Men (1957)                  616
dtype: int64
```

More up to date data on movie ratings can be found on grouplens web site.
https://grouplens.org/datasets/movielens/

MovieLens

GroupLens Research has collected and made available rating data sets from the MovieLens web site (http://movielens.org). The data sets were collected over various periods of time, depending on the size of the set. Before using these data sets, please review their README files for the usage licenses and other details.

Help our research lab: Please take a short survey about the MovieLens datasets

recommended for new research

MovieLens 20M Dataset

Stable benchmark dataset. 20 million ratings and 465,000 tag applications applied to 27,000 movies by 138,000 users. Includes tag genome data with 12 million relevance scores across 1,100 tags. Released 4/2015; updated 10/2016 to update links.csv and add tag genome data.

- README.html
- ml-20m.zip (size: 190 MB, checksum)

Permalink: http://grouplens.org/datasets/movielens/20m/

recommended for education and development

MovieLens Latest Datasets

These datasets will change over time, and are not appropriate for reporting research results. We will keep the download links stable for automated downloads. We will not archive or make available previously released versions.

Small: 100,000 ratings and 1,300 tag applications applied to 9,000 movies by 700 users. Last updated 10/2016.

- README.html
- ml-latest-small.zip (size: 1 MB)

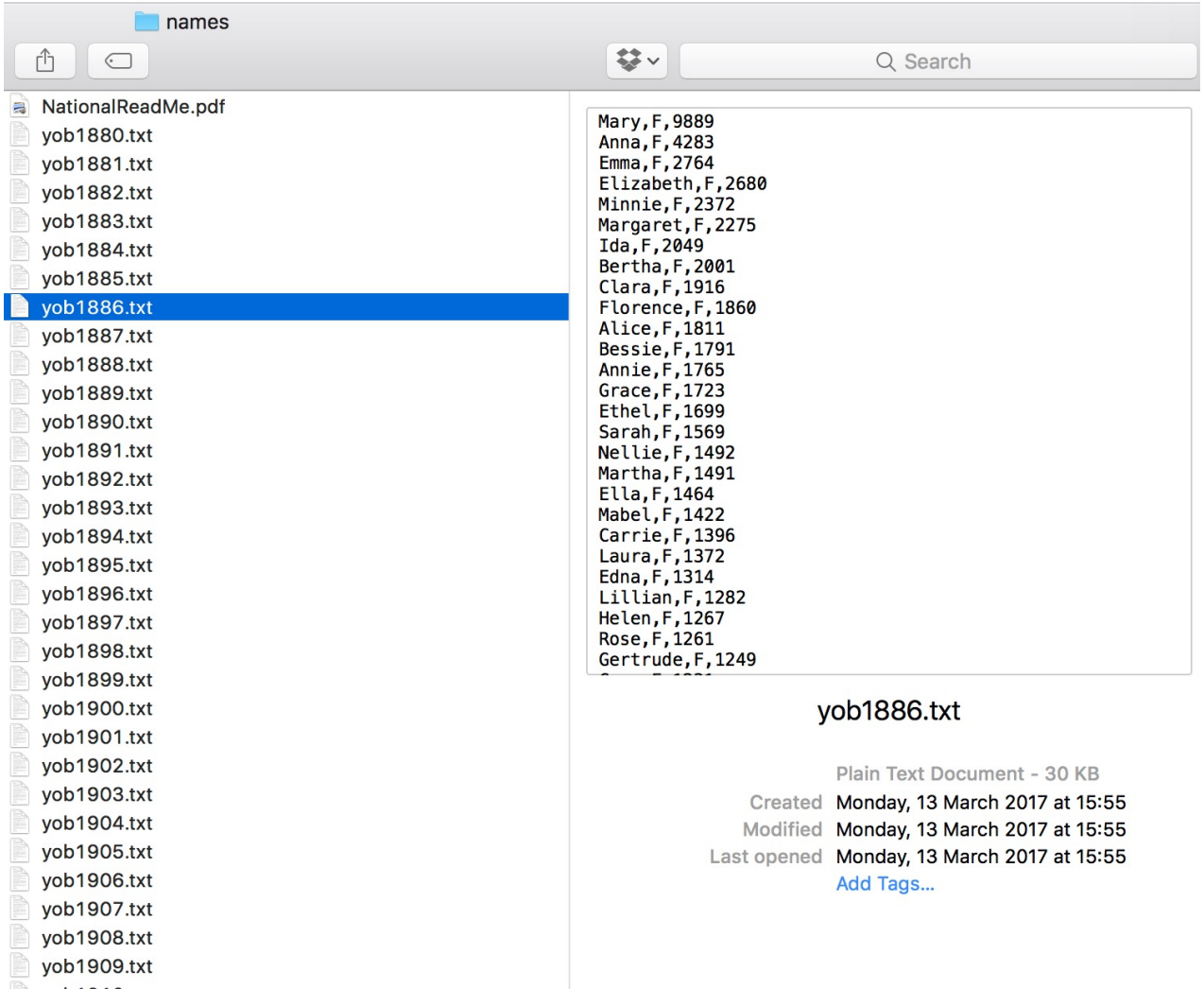
Full: 26,000,000 ratings and 750,000 tag applications applied to 45,000 movies by 270,000 users. Includes tag

Datasets

- MovieLens
- HetRec 2011
- WikiLens
- Book-Crossing
- Jester
- EachMovie

Introductory Example: US Baby names analysis

- The United States Social Security Administration (SSA) has made available data on the frequency of baby names from 1880 through the present.
- http://www.ssa.gov/oact/babynames/limits.html



Introductory Example: US Baby names analysis

- The United States Social Security Administration (SSA) has made available data on the frequency of baby names from 1880 through the present.
- http://www.ssa.gov/oact/babynames/limits.html
- As this is a nicely comma-separated form, it can be loaded into a DataFrame with pandas.read_csv:

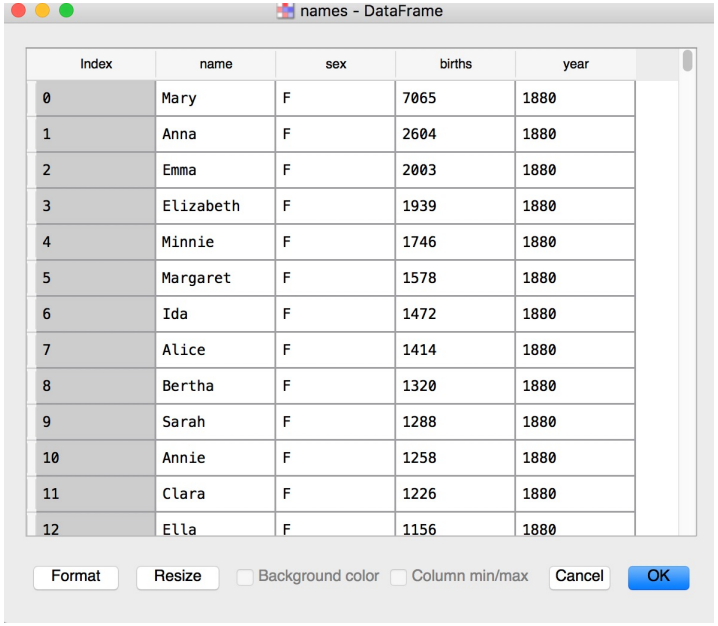
```
In [2]: import pandas as pd

In [3]: names1880=pd.read_csv('./names/yob1880.txt',names = [ 'name','sex', 'births' ])

In [4]: names1880
Out[4]:
            name sex  births
0           Mary   F    7065
1           Anna   F    2604
2           Emma   F    2003
3      Elizabeth   F    1939
4         Minnie   F    1746
5       Margaret   F    1578
6            Ida   F    1472
7          Alice   F    1414
8         Bertha   F    1320
9          Sarah   F    1288
10         Annie   F    1258
11         Clara   F    1226
12          Ella   F    1156
13      Florence   F    1063
```

- To calculate total number of births for each gender:

```
In [5]: names1880.groupby('sex').births.sum()
Out[5]:
sex
F     90992
M    110491
Name: births, dtype: int64
```

- Since the data set is split into files by year, one of the first things to do is to assemble all of the data into a single DataFrame and further to add a year field. This is easy to do using pandas.concat:

```
In [6]: #2016 is the lat availabe year right now

In [7]: years = range(1880,2017)

In [8]: pieces = []

In [9]: columns = [ 'name', 'sex', 'births' ]

In [10]: for year in years:
    ...:     path = 'names/yob%d.txt' % year
    ...:     frame = pd.read_csv(path,names=columns)
    ...:     frame['year']= year
    ...:     pieces.append(frame)
    ...:

In [11]: #concatenate everything into a single DataFrame

In [12]: names=pd.concat(pieces, ignore_index=True)
```

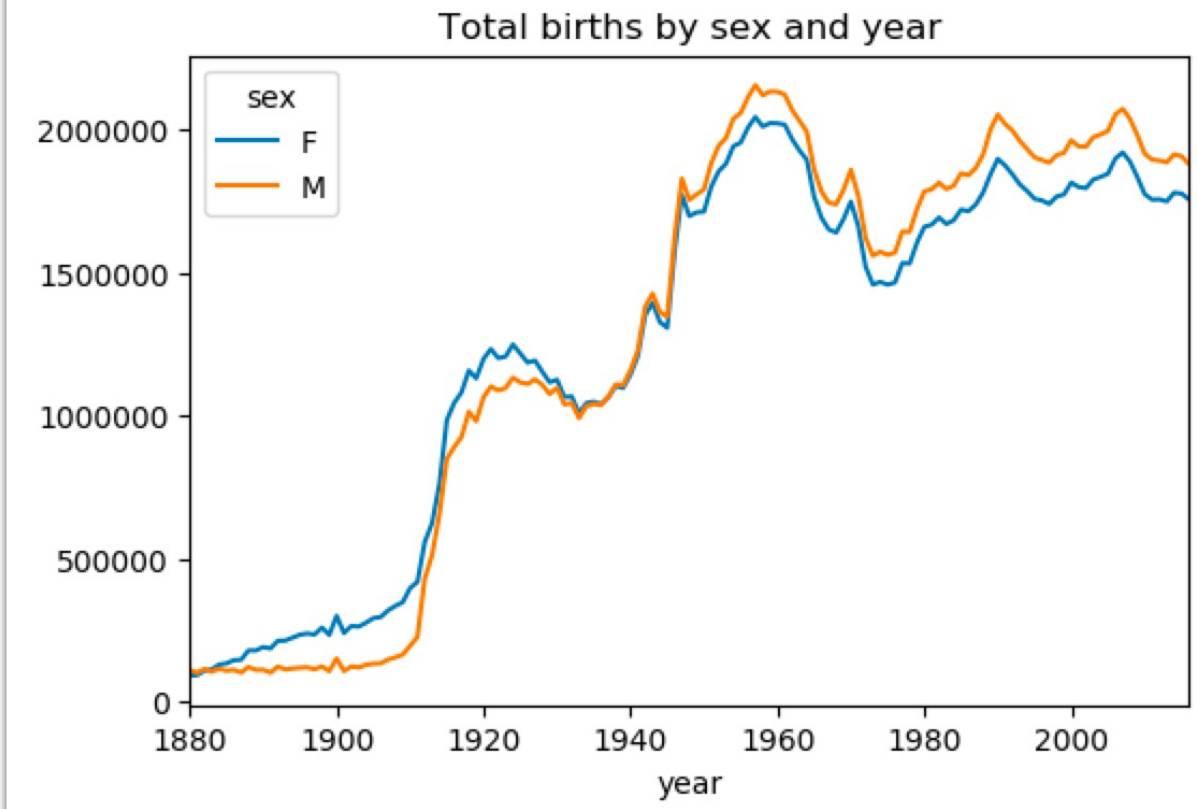


- There are a couple things to note here.
- First, remember that concat glues the DataFrame objects together row-wise by default.
- Secondly, you have to pass ignore_index=True because we're not interested in preserving the original row numbers returned from read_csv.

- We can aggregate the data at the year and sex level using pivot_table feature

```
In [13]: total_births = names.pivot_table('births', index='year', columns='sex', aggfunc=sum)

In [14]: total_births.tail()
Out[14]:
sex          F        M
year
2012   1756347  1892094
2013   1749061  1885683
2014   1779496  1913434
2015   1776538  1907211
2016   1756647  1880674
```

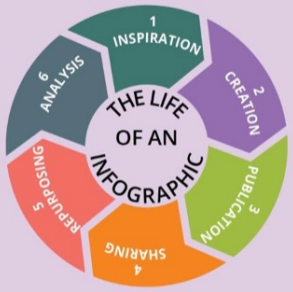
```
In [15]: total_births.plot(title='Total births by sex and year')
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x114e68c50>
```



Data Visualization

- Data storytelling
- General Design concepts
- Things that should be avoided when prouducing visuals

IMPACT OF INFOGRAPHICS ON SOCIAL MEDIA MARKETING



DID YOU KNOW?

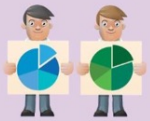
Only **20%** of text is remembered

90% of information transmitted to the brain is visual

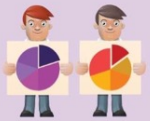
Color images increase willingness to read by **80%**

THE HUMAN BRAIN PROCESSES IMAGES FASTER THAN TEXT **3X**

90% OF ALL INFORMATION THAT COMES TO THE BRAIN IS VISUAL



40% OF PEOPLE WILL RESPOND BETTER TO VISUAL INFORMATION THAN TEXT



50% AMOUNT OF CONTENT A PERSON RETAINS FROM A PRESENTATION WHEN IT INCLUDES VISUALS AND WORDS

10% AMOUNT OF CONTENT A PERSON RETAINS FROM A PRESENTATION IF PRESENTED ORALLY

HOW IMAGES AFFECT SOCIAL ENGAGEMENT

Social media posts also benefit from adding photos. According to an analysis by Web Liquid (conducted per-Timeline), Facebook posts with photos have the highest user engagement than any other post.

HOW IMAGES AFFECT ARTICLES

Using images in articles can give a boost to that content's page views. But according to data from Skyword, certain content categories benefit more from having images than others.

www.medialabs.in

MediaLabs
THE DIGITAL NATIVE

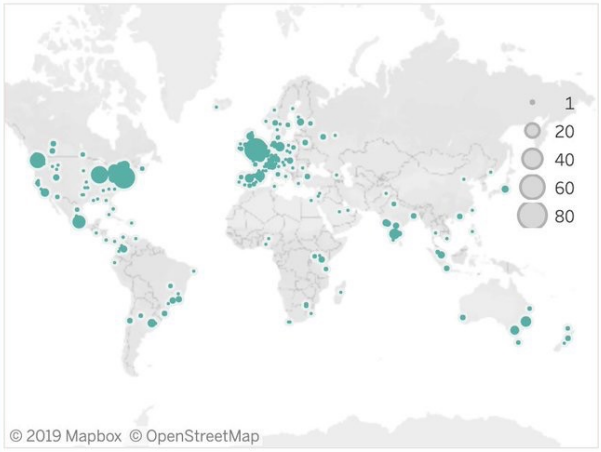
DATA VISUALIZATION SOCIETY

"Getting to Know You"

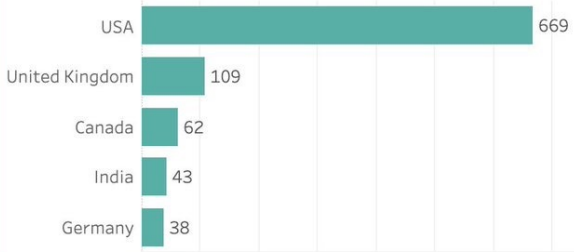
Survey Responses from 1,359 Members

WHERE THEY LIVE

Members Participate from Cities Across the World in 77 Countries



Top 5 Countries Where Members Live

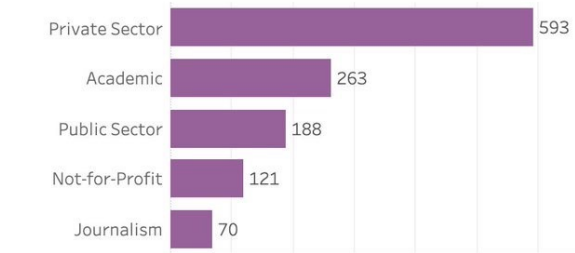


WHO THEY ARE

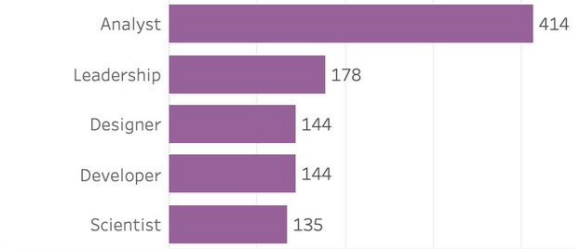
63% identify as men

32% identify as women

Top 5 Fields Where Members Work



Top 5 Professional Roles

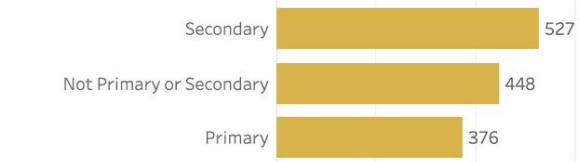


HOW THEY USE DATA

5.5 avg years of professional data visualization experience

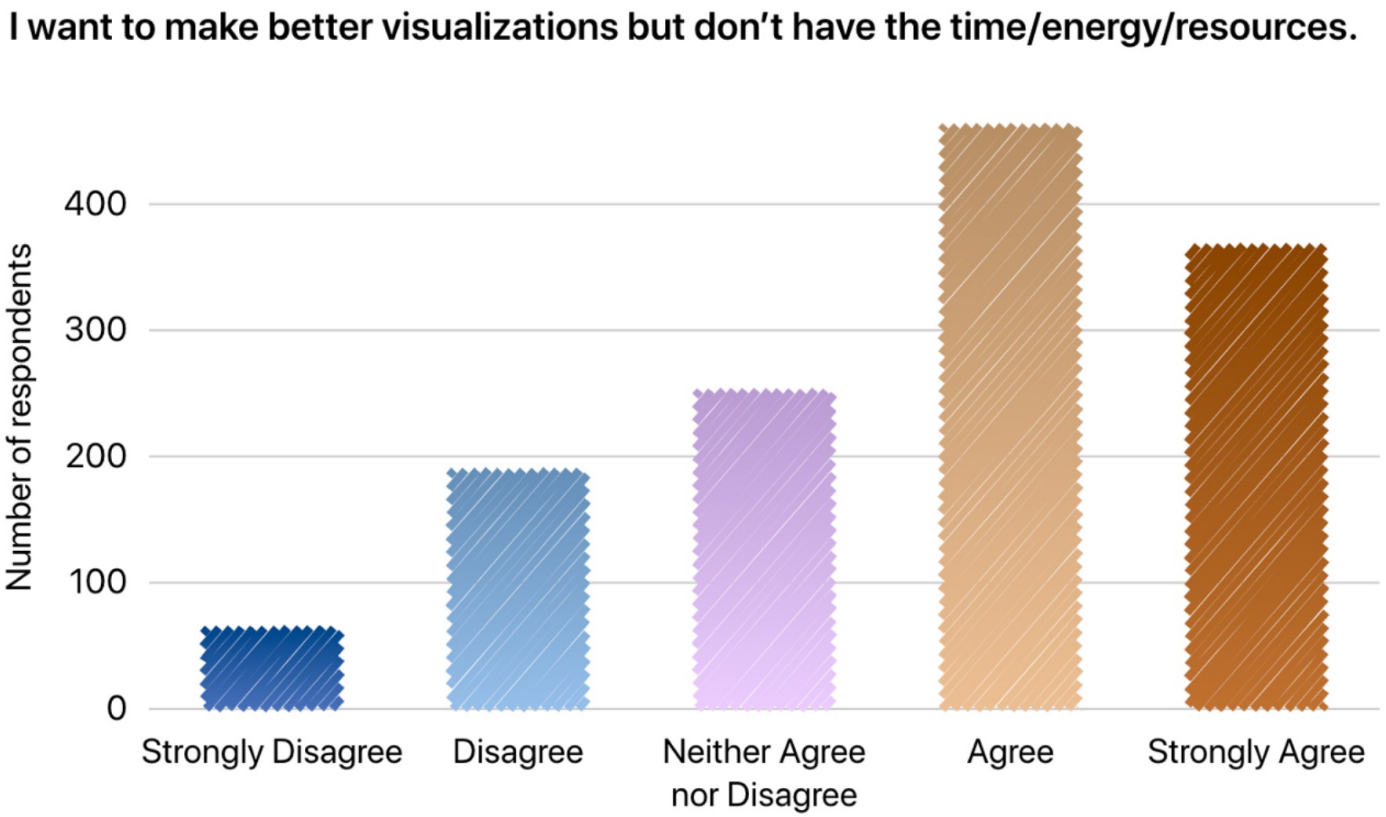
79% are mostly self-taught data visualization users

Data Visualization is a Secondary Responsibility or Less in Member Roles

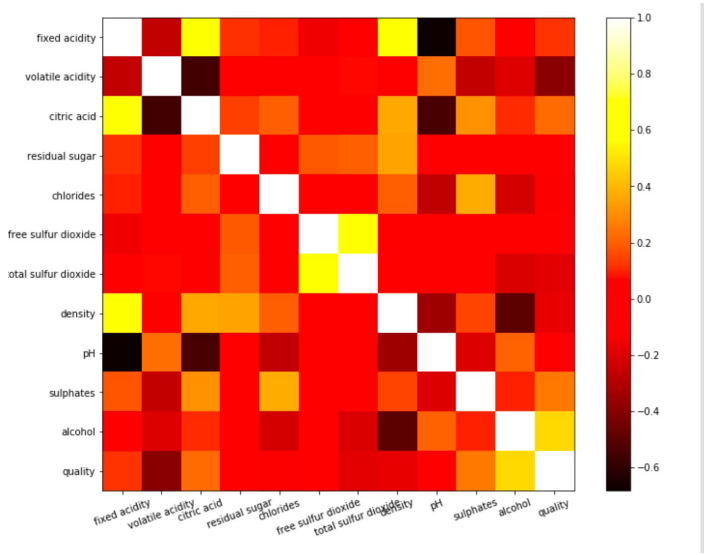


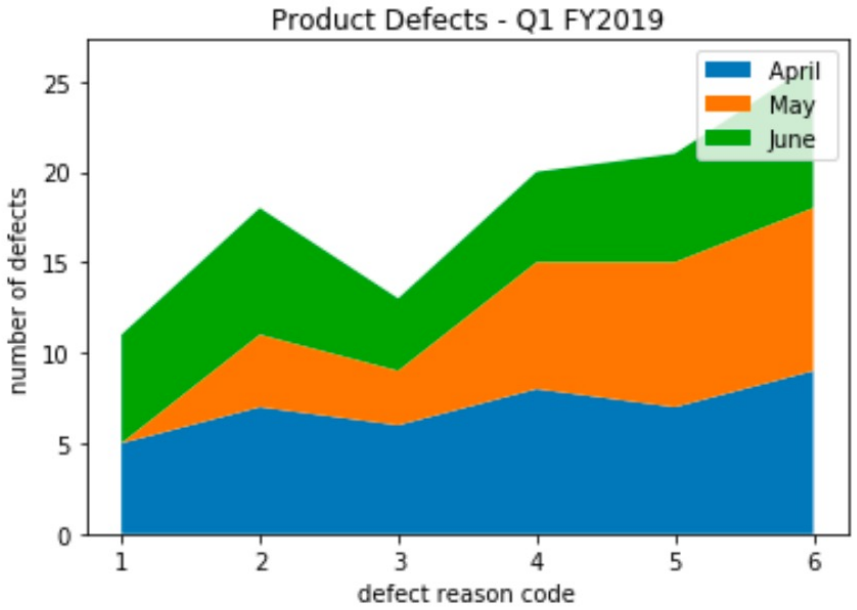
Top 5 Priorities for Improving Data Visualizations

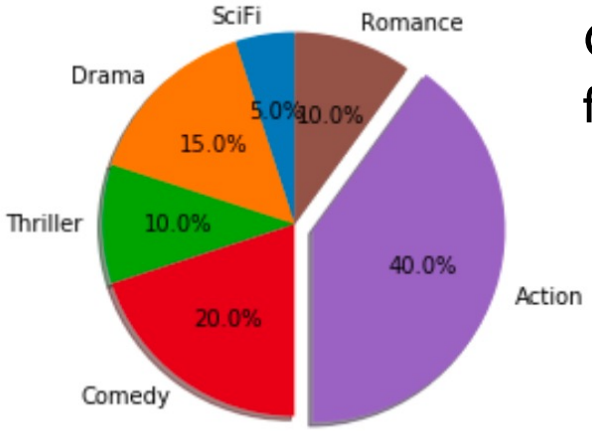
DataViz Society 2019 Survey results



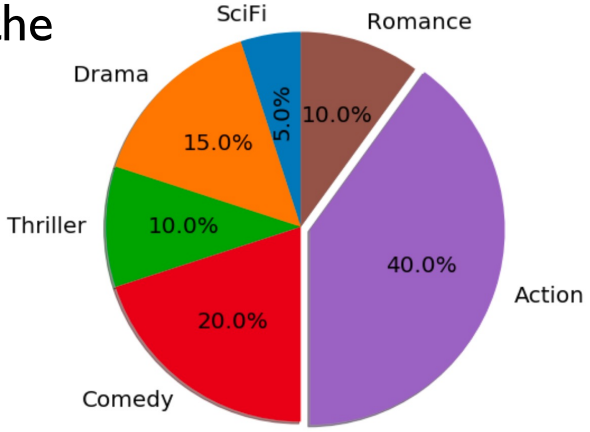
Basic Plot Types



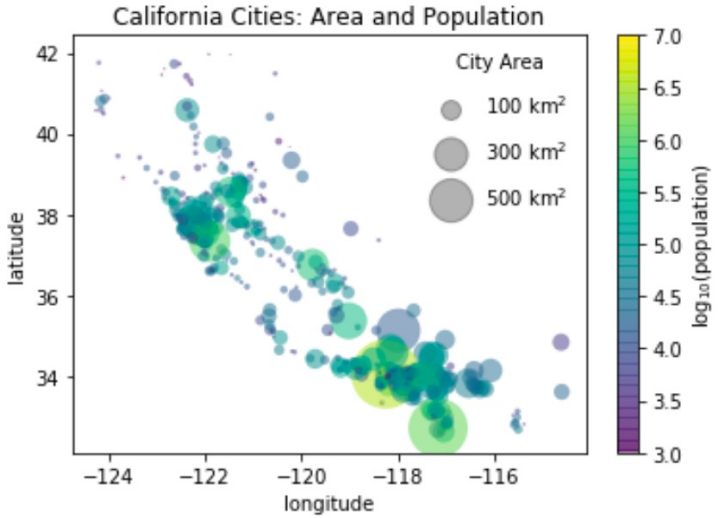


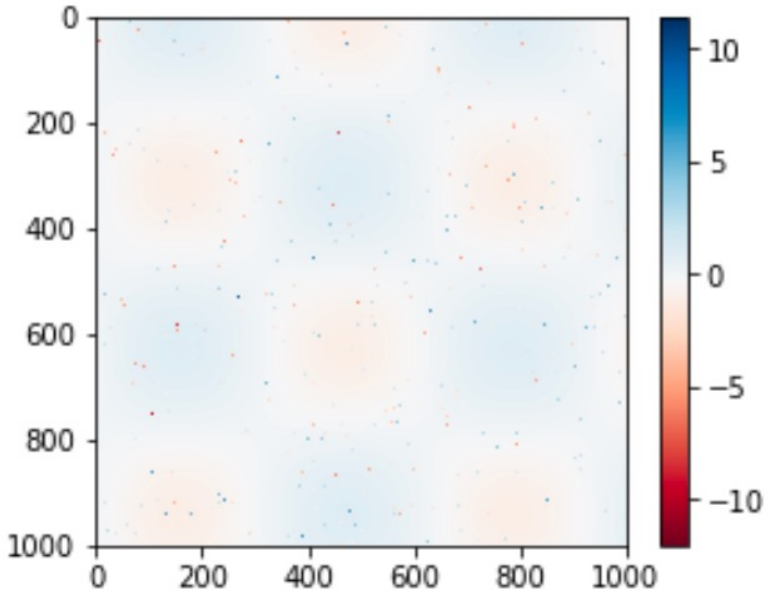


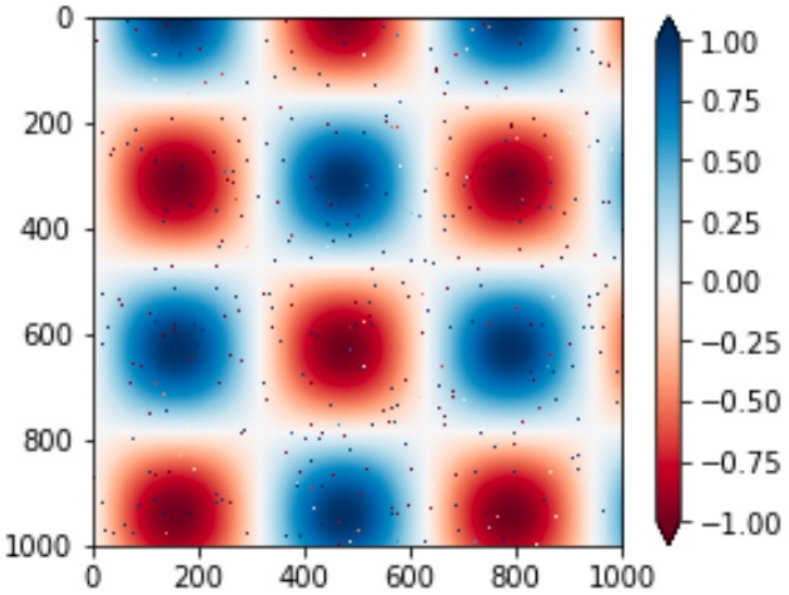
Customizing the figures



Customizing colors and styles







Text annotation