Algorithmic Methods of Data Mining

Initial Data Exploration & Visualization

Ioannis Chatzigiannakis

Sapienza University of Rome

Laboratory 3

Kaggle: Steam Reviews Dataset 2021

- Kaggle is probably world's largest data science community.
- We will work with data on the Steam Reviews Dataset 2021.
  - around 21 million user reviews of around 300 different games on Steam
  - `https://www.kaggle.com/najzeko/steam-reviews-2021`
- The dataset is formated using CSV
  - Each row in the file represents a review.
  - All reviews are related to products (games) and users.
  - Each row is like many-to-many relation between products and users.

CSV + Python + Pandas – Example

- Download the small version of the CSV dataset from: `https://www.kaggle.com/najzeko/steam-reviews-2021`
- The dataset contains 1 file.
- Create a folder data under your jupyter central folder.
- Unzip the archive and place the dataset files inside the data folder.

Download Kaggle Dataset from SageMaker/EC2 (1)

- Download the command line tool:

```
pip3 install --user --upgrade kaggle
```

- Get a Kaggle API Token from the website:
  1. (Top-Right) → Account
  2. API → Create New API Token
  3. Store `kaggle.json` under the correct path
  4. Upload `kaggle.json` to SageMaker/EC2
  5. Move `kaggle.json` to path `/root/.kaggle`
- View the dataset using the tool.

```
~/.local/bin/kaggle datasets files najzeko/steam-reviews-2021
```

Download Kaggle Dataset from SageMaker/EC2 (2)

- Download the dataset using the tool.

```
~/.local/bin/kaggle datasets download najzeko/steam-reviews-2021
```

- Unzip the file.

```
unzip steam-reviews-2021.zip
```

Check version of Pandas and Numpy

```
import pandas as pd
import numpy as np
pd.__version__
np.__version__
```

To upgrade to latest versions

```
pip3 install --upgrade pandas
pip3 install --upgrade numpy
```

Read dataset in CSV format

```
import pandas as pd
dataset = pd.read_csv('./steam_reviews.csv', header='infer', nrows=2000
```

Get help on command (due to comments)

```
pd.read_csv?
```

Shape: Rows × Columns

```
dataset.shape
(48296896, 9)
```

Dataset Columns

```
dataset.columns
Index(['Unnamed: 0', 'app_id', 'app_name', 'review_id', 'language',
       'review', 'timestamp_created', 'timestamp_updated',
       'recommended', 'votes_helpful', 'votes_funny',
       'weighted_vote_score', 'comment_count', 'steam_purchase',
       'received_for_free', 'written_during_early_access',
       'author.steamid', 'author.num_games_owned',
       'author.num_reviews', 'author.playtime_forever',
       'author.playtime_last_two_weeks',
       'author.playtime_at_review', 'author.last_played'],
      dtype='object')
```

```
dataset.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000000 entries, 0 to 1999999
Data columns (total 23 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Unnamed: 0                      100 non-null    int64
 1   app_id                          100 non-null    int64
 2   app_name                        100 non-null    object
 3   review_id                       100 non-null    int64
 4   language                        100 non-null    object
 5   review                          99 non-null     object
 6   timestamp_created               100 non-null    int64
 7   timestamp_updated               100 non-null    int64
 8   recommended                     100 non-null    bool
 9   votes_helpful                   100 non-null    int64
 10  votes_funny                     100 non-null    int64
 11  weighted_vote_score             100 non-null    float64
 12  comment_count                   100 non-null    int64
 13  steam_purchase                  100 non-null    bool
 14  received_for_free               100 non-null    bool
 15  written_during_early_access     100 non-null    bool
 16  author.steamid                  100 non-null    int64
 17  author.num_games_owned          100 non-null    int64
 18  author.num_reviews              100 non-null    int64
 19  author.playtime_forever         100 non-null    float64
 20  author.playtime_last_two_weeks  100 non-null    float64
 21  author.playtime_at_review       100 non-null    float64
 22  author.last_played              100 non-null    float64
dtypes: bool(4), float64(5), int64(11), object(3)
memory usage: 297.5+ MB
```

File structure (1)

1. `app_id` – the application unique identification number
2. `app_name` – the name of the game
3. `review_id` – unique identity of review
4. `language` – language used in review
5. `review` – the actual review
6. `timestamp_created` – date the review was created (unix timestamp)
7. `timestamp_updated` – date the review was last updated (unix timestamp)
8. `recommended` – `true` means it was a positive recommendation
9. `votes_helpful` – the number of users that found this review helpful
10. `votes_funny` – the number of users that found this review funny
11. `weighted_vote_score` – helpfulness score
12. `comment_count` – number of comments posted on this review

File structure (2)

13. `steam_purchase` – `true` if the user purchased the game on Steam
14. `received_for_free` – `true` if the user checked a box saying they got the app for free
15. `written_during_early_access` – `true` if the user posted this review while the game was in Early Access
16. Author fields:
    - 16.1 `steamid` – the user's SteamID
    - 16.2 `num_games_owned` – number of games owned by the user
    - 16.3 `num_reviews` – number of reviews written by the user
    - 16.4 `playtime_forever` – lifetime playtime tracked in this app
    - 16.5 `playtime_last_two_weeks` – playtime tracked in the past two weeks for this app
    - 16.6 `playtime_at_review` – playtime when the review was written
    - 16.7 `last_played` – time for when the user last played

Handling Date + Time

We need to parse dates only for columns `timestamp_created`, `timestamp_updated` and `author.last_played`

We need a function that converts unix timestamps into datetime.

```
def dateparse(time_in_secs):
    return pd.to_datetime(time_in_secs, unit='s')
```

Read CSV and parse dates

```
dataset = pd.read_csv('./steam_reviews.csv', header='infer',
                      nrows=2000000,
                      parse_dates=['timestamp_created',
                      'timestamp_updated', 'author.last_played'],
                      date_parser=dateparse)
```

Examine column data types

```
dataset.dtypes
```

```
Unnamed: 0                          int64
app_id                              int64
app_name                           object
review_id                           int64
language                           object
review                             object
timestamp_created          datetime64[ns]
timestamp_updated          datetime64[ns]
recommended                          bool
votes_helpful                       int64
votes_funny                         int64
weighted_vote_score               float64
comment_count                       int64
steam_purchase                       bool
received_for_free                    bool
written_during_early_access          bool
author.steamid                      int64
author.num_games_owned              int64
author.num_reviews                  int64
author.playtime_forever           float64
author.playtime_last_two_weeks    float64
author.playtime_at_review         float64
author.last_played         datetime64[ns]
dtype: object
```

```
dataset.timestamp_created.min()
```

```
Timestamp('2020-05-27 02:59:50')
```

```
dataset.timestamp_created.max()
```

```
Timestamp('2021-01-23 06:00:29')
```

```
dataset.timestamp_created.dt.dayofweek
```

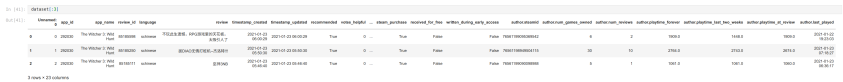
```
0        5
1        5
2        5
3        5
4        5
        ..
99995    2
99996    2
99997    2
99998    2
99999    2
Name: timestamp_created, Length: 100000, dtype: int64
```

Select a row

```
dataset.loc[3]
```

```
Unnamed: 0                                                  3
app_id                                                 292030
app_name                            The Witcher 3: Wild Hunt
review_id                                           85184605
language                                             english
review                  One of the best RPG's of all time, worthy of a...
timestamp_created                       2021-01-23 05:32:50
timestamp_updated                       2021-01-23 05:32:50
recommended                                            True
votes_helpful                                             0
votes_funny                                               0
weighted_vote_score                                     0.0
comment_count                                             0
steam_purchase                                         True
received_for_free                                     False
written_during_early_access                           False
author.steamid                            76561199054755373
author.num_games_owned                                    5
author.num_reviews                                        3
author.playtime_forever                              5587.0
author.playtime_last_two_weeks                       3200.0
author.playtime_at_review                            5524.0
author.last_played                      2021-01-23 06:35:44
Name: 3, dtype: object
```

```
dataset[:3]
```



Select a column

```
dataset.app_name (or dataset['app_name'])
```

```
0         The Witcher 3: Wild Hunt
1         The Witcher 3: Wild Hunt
2         The Witcher 3: Wild Hunt
3         The Witcher 3: Wild Hunt
4         The Witcher 3: Wild Hunt
                   ...
99995     The Witcher 3: Wild Hunt
99996     The Witcher 3: Wild Hunt
99997     The Witcher 3: Wild Hunt
99998     The Witcher 3: Wild Hunt
99999     The Witcher 3: Wild Hunt
Name: app_name, Length: 100000, dtype: object
```

```
dataset[dataset['author.steamid'] == 76561199095369542]
```

Shard – horizontal partition

```
only_english = dataset[dataset.language == 'english']

len(only_english)
```

```
24956
```

Unique languages

```
dataset.language.nunique()
```

```
28
```

```
dataset.language.unique()
```

```
array(['schinese', 'english', 'turkish', 'spanish', 'russian',
       'koreana', 'latam', 'brazilian', 'portuguese',
       'vietnamese', 'polish', 'french', 'german', 'hungarian',
       'ukrainian', 'tchinese', 'bulgarian', 'czech', 'italian',
       'thai', 'greek', 'dutch', 'finnish', 'romanian',
       'japanese', 'swedish', 'danish', 'norwegian'],
      dtype=object)
```

Examine entries based on Language

Use GroupBy

```
dataset.groupby('language').review_id.count()
```

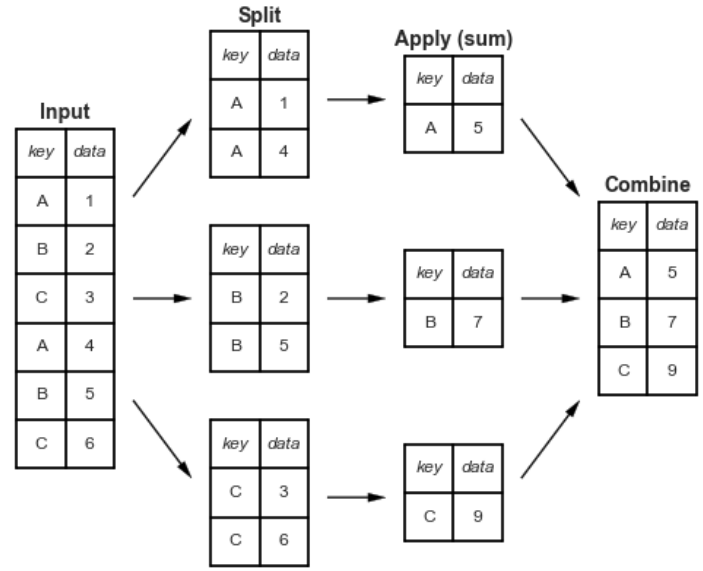
```
language
brazilian       6000
bulgarian         11
czech            925
danish            80
dutch            146
english        24956
finnish          116
french          1346
german          2230
greek             80
hungarian        228
italian          391
japanese         112
koreana         1810
latam            456
norwegian         69
polish          5811
portuguese       291
romanian          39
russian        17970
schinese       24110
spanish         2866
swedish          201
tchinese         934
thai             462
turkish         8051
ukrainian        231
vietnamese        78
Name: review_id, dtype: int64
```

GroupBy Type

```
dataset.groupby('language')
<pandas.core.groupby.generic.DataFrameGroupBy object at ...>
```

- GroupBy is Lazy – no operations are done until instructed.
- GroupBy is split-apply-combine:
  1. Split a table into groups
  2. Apply some operations to each of those smaller tables
  3. Combine the results
- So groupby('event_type') does none of these steps – not until we invoke a method.
- For every Group, we get 1 Dataframe.

```
for language, frame in dataset.groupby('language'):
    print(f'Number of reviews for entry {language!r}: ',
          frame.review_id.nunique())
```



GroupBy with more than 1 fields

```
dataset.groupby(['language', dataset.timestamp_created.dt.hour])
       .review_id.count()
```

```
language    timestamp_created
brazilian   0                   409
            1                   391
            2                   335
            3                   289
            4                   229
                               ...
vietnamese  15                    6
            16                    5
            17                    5
            18                    4
            20                    2
Name: review_id, Length: 623, dtype: int64
```

Generate descriptive statistics

```
dataset.groupby([dataset.timestamp_created.dt.hour,
                 dataset.language]).describe()
```

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html

Unstack a level of the table

```
dataset.groupby([dataset.timestamp_created.dt.hour,
                 dataset.language]).describe().unstack()
```

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.unstack.html

Visualization using MatPlotLib

- Install MatPlotLib

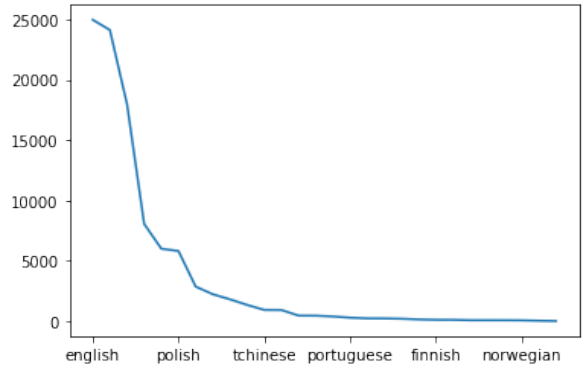
```
pip3 install matplotlib
```

- We can use MatPlotLib as a standalone package.
- We can use MatPlotLib via Pandas.
- or both.

```
import matplotlib.pyplot as plt
```

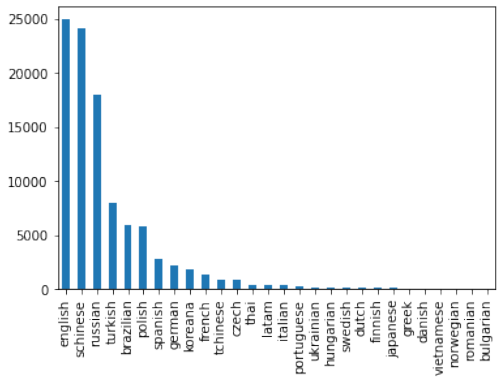
Using MatPlotLib via Pandas

```
dataset['language'].value_counts().plot()
```



Change Graph Type

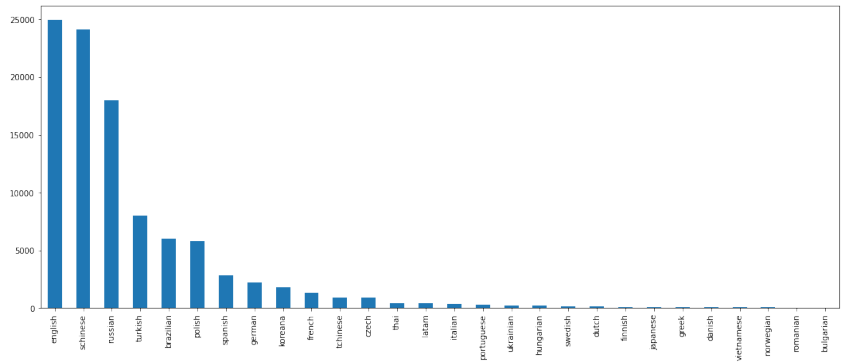
```
dataset['language'].value_counts().plot.bar()
```



Other types: hist for histogram, box for boxplot, kde or density for density plots, area for area plots, scatter for scatter plots, hexbin for hexagonal bin plots, pie for pie plots.

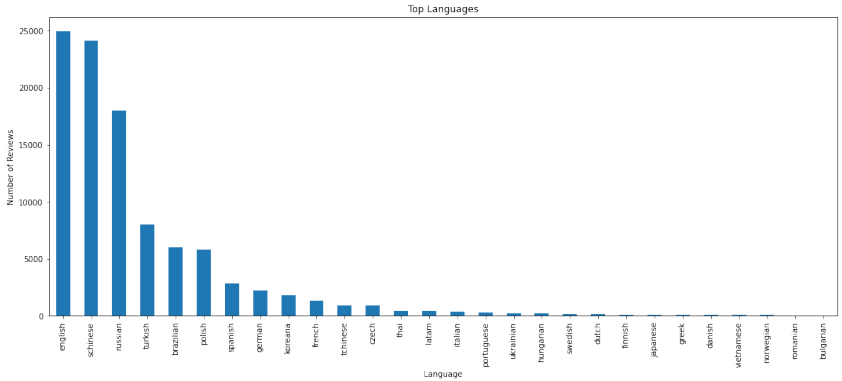
Change the dimensions of the figure

```
dataset['language'].value_counts().plot.bar(figsize = (18, 7))
```



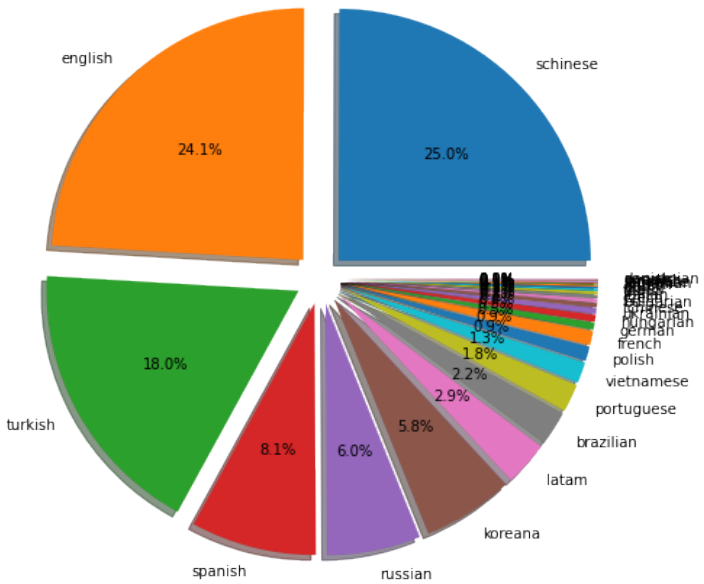
Add title and Axis Captions

```
dataset['language'].value_counts().plot.bar(\
    figsize = (18, 7),\
    title='Top Languages', \
    xlabel='Language',\
    ylabel='Number of Reviews')
```



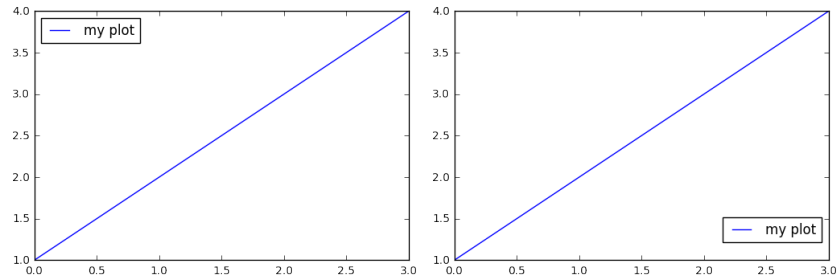
Using MatPlotLib directly

```
plt.rcParams['figure.figsize'] = (8, 8)
plt.pie(dataset['language'].value_counts(),
        labels = dataset.language.unique(),
        explode = [0.1 for value in range(0, dataset.language.nunique()
        shadow = True, autopct = '%.1f%%')
plt.title('Languages', fontsize = 20)
plt.axis('off')
plt.show()
```



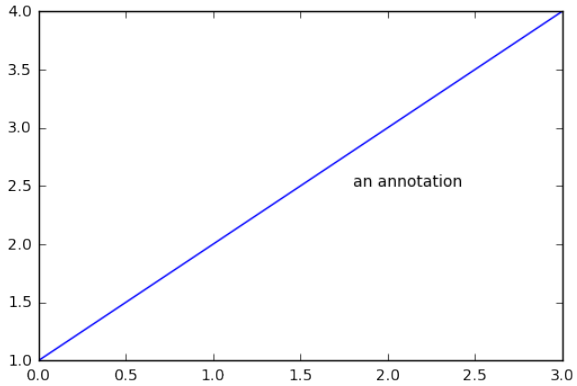
Legend Location

```
plt.plot([1,2,3,4], label='my plot')
plt.legend(bbox_to_anchor=(0.3, 1))
plt.show()

plt.plot([1,2,3,4], label='my plot')
plt.legend(bbox_to_anchor=(1, 0.2))
plt.show()
```



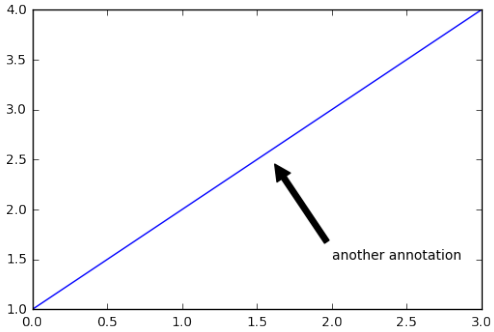
Simple Text

```
plt.plot([1,2,3,4], label='my plot')
plt.text(1.8, 2.5, 'an annotation')
plt.show()
```

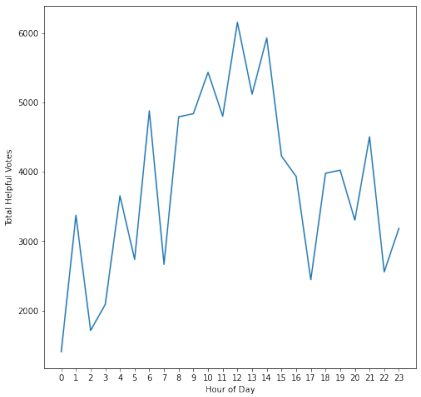


Text Annotations

```
plt.plot([1,2,3,4], label='my plot')
plt.annotate('another annotation', xy=(1.6, 2.5),
             xytext=(2, 1.5),
             arrowprops=dict(facecolor='black',shrink=0.05))
plt.show()
```

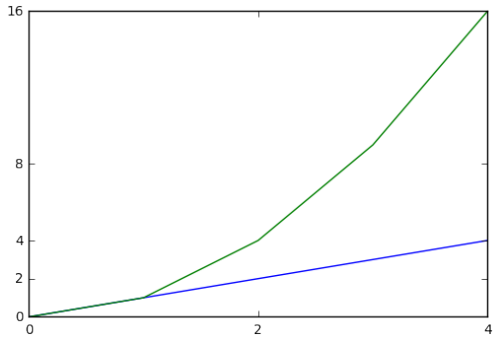



```
hour_reviews = dataset.groupby(dataset.timestamp_created.dt.hour)

plt.figure()
hour_reviews.votes_helpful.sum().plot()
plt.xlabel("Hour of Day")
plt.ylabel("Total Helpful Votes")
plt.xticks(range(0,24))
plt.show()
```



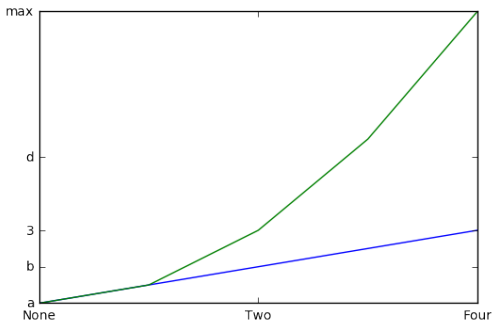
Axis Tick Positioning

```
plt.plot([0,1,2,3,4], [0,1,2,3,4])
plt.plot([0,1,2,3,4], [0,1,4,9,16])
plt.xticks([0,2,4])
plt.yticks([0,2,4,8,16])
plt.show()
```



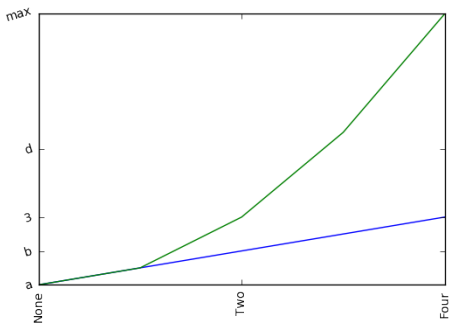
Axis Tick Positioning & Labels

```
plt.plot([0,1,2,3,4], [0,1,2,3,4])
plt.plot([0,1,2,3,4], [0,1,4,9,16])
plt.xticks([0,2,4],['None','Two','Four'])
plt.yticks([0,2,4,8,16],['a','b',3,'d','max'])
plt.show()
```



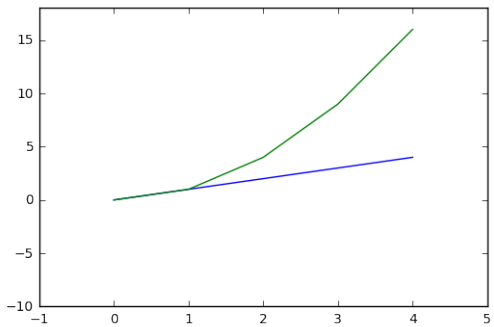
Axis Tick Positioning, Labels & Orientation

```
plt.plot([0,1,2,3,4], [0,1,2,3,4])
plt.plot([0,1,2,3,4], [0,1,4,9,16])
plt.xticks([0,2,4],['None','Two','Four'],rotation='vertical')
plt.yticks([0,2,4,8,16],['a','b',3,'d','max'], rotation=70)
plt.show()
```



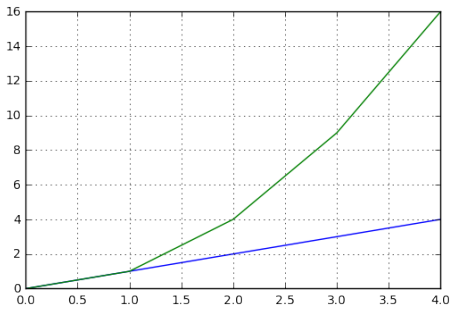
Controlling Axis Values

```
plt.plot([0,1,2,3,4], [0,1,2,3,4])
plt.plot([0,1,2,3,4], [0,1,4,9,16])
plt.xlim(-1,5)
plt.ylim(-10,18)
plt.show()
```



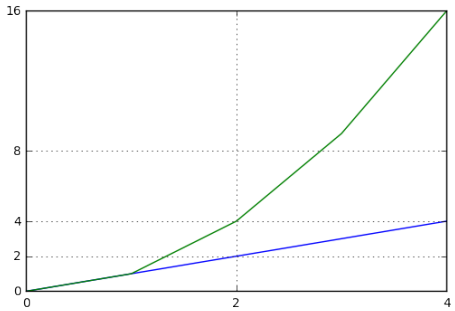
Simple Grid

```
plt.plot([0,1,2,3,4], [0,1,2,3,4])
plt.plot([0,1,2,3,4], [0,1,4,9,16])
plt.grid()
plt.show()
```



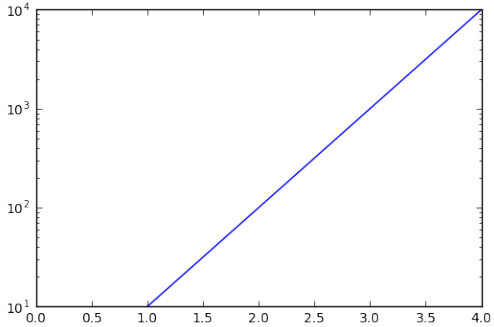
Ticks & Grid

```
plt.plot([0,1,2,3,4], [0,1,2,3,4])
plt.plot([0,1,2,3,4], [0,1,4,9,16])
plt.xticks([0,2,4])
plt.yticks([0,2,4,8,16])
plt.grid()
plt.show()
```



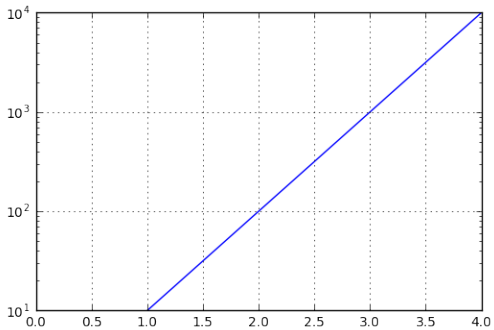
Logarithmic Scale

```
plt.plot([0,10,100,1000,10000])
plt.yscale('log')
plt.show()
```



Logarithmic Scale & Grid

```
plt.plot([0,10,100,1000,10000])
plt.yscale('log')
plt.grid()
plt.show()
```



Visualization using Other Libraries

- Many different libraries available, e.g., squarify

```
pip3 install squarify
```

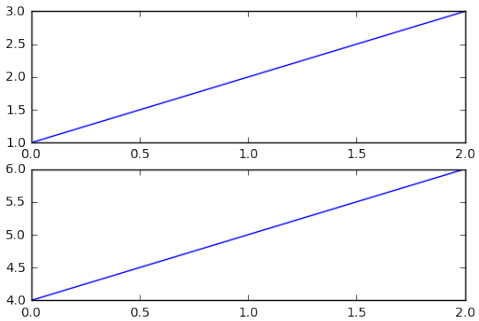
```
import squarify

top_language_n = 5
top_language = dataset.loc[:,'language'].value_counts()\
[:top_language_n].sort_values(ascending=False)

squarify.plot(sizes=top_language, label=top_language.index.array,\
color=["red","green","orange","blue","grey"], alpha=.7  )

plt.axis('off')
plt.show()
```

Producing Sub Plots

```
fig=plt.figure(figsize=(18,9))

ax1=fig.add_subplot(2,1,1)
sns.distplot(customer_analysis['total_sales'],
             ax=ax1)

ax2=fig.add_subplot(2,1,2)
sns.distplot(sales_filtered['total_sales'],
             ax=ax2)

fig.tight_layout(pad=3.0);
```

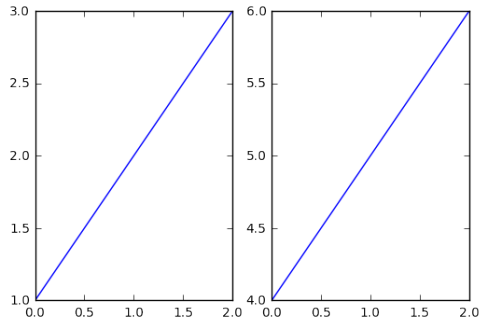
Multiple Subplots – Vertical

```
plt.figure()
plt.subplot(211)
plt.plot([1, 2, 3])
plt.subplot(212)
plt.plot([4, 5, 6])
plt.show()
```



Multiple Subplots – Horizontal

```
plt.figure()
plt.subplot(121)
plt.plot([1, 2, 3])
plt.subplot(122)
plt.plot([4, 5, 6])
plt.show()
```



Multiple Subplots – Grid

```
plt.figure()
plt.subplot(221)
plt.plot([1, 2, 3])
plt.subplot(222)
plt.plot([4, 5, 6])
plt.subplot(223)
plt.plot([10, 20, 30])
plt.subplot(224)
plt.plot([40, 50, 60])
plt.show()
```