Tutorial #2 - Explore and Visualize

October 22, 2018

1 Tutorial #2 - Explore and Visualize

Welcome to Cognitive Class Labs. This notebook is the **second** in a series of "getting started" tutorials that is designed to introduce some basic concepts and help get you familiar with using the workbench.

In this notebook, we will explore and visualize the olympic medals data that you added to your workbench in **Tutorial #1 - Get Data**. Specifically, this tutorial covers:

1. Load data in memory using a `pandas` DataFrame
2. Explore and manipulate data using DataFrame functions
3. Group data by columns
4. Calculate statistics over grouped data
5. Plot data using the pre-installed `matplotlib` package
6. Plot data using a third-party library

We will do this in the context of answering the following questions:

1. Which discipline and event has awarded the most gold medals?
2. Which country has won the most gold, silver, and bronze medals?

1.1 Pre-requisites

- A basic familiarity with the Python Programming Language and the IPython Notebook.
- A basic understanding of common graphical techniques used for exploratory data analysis.
- A basic understanding of `matplotlib`

1.2 Load Data

Our first step is to load the olympic medal data into a pandas DataFrame in memory. As demonstrated in **Tutorial #1 - Get Data**, we can do this by following steps:

1. Download the olympic medal data in CSV format. Click this Box link to open the document in a new browser window.
2. Save the CSV file to your computer by clicking on the Download button.
3. Drag the CSV file from your desktop onto the workbench (Note that the CSV file appears under your **Recent Data** panel in the sidebar.)
4. Click the arrow button (**>**) next to the CSV file you just uploaded.
5. In the section that appears below the item, click "Rename"

6. Change the name of the file to "medals.csv" and press Enter or click outside the name.
7. Execute the code cell by clicking the () play button on the notebook toolbar, or by pressing Ctrl-Enter.

```
In [1]: import pandas
        medals_df = pandas.read_csv('/resources/medals.csv')
        # Prune non-data rows
        medals_df = medals_df.dropna()
```

```
In [8]: medals_df.tail()
```

```
Out[8]:       Year   City   Sport Discipline  NOC            Event Event gender  \
        2306  2006  Turin  Skiing  Snowboard  USA         Half-pipe            M
        2307  2006  Turin  Skiing  Snowboard  USA         Half-pipe            W
        2308  2006  Turin  Skiing  Snowboard  USA         Half-pipe            W
        2309  2006  Turin  Skiing  Snowboard  USA   Snowboard Cross            M
        2310  2006  Turin  Skiing  Snowboard  USA   Snowboard Cross            W

               Medal
        2306  Silver
        2307    Gold
        2308  Silver
        2309    Gold
        2310  Silver
```

```
In [9]: medals_df.head(6)
```

```
Out[9]:    Year       City       Sport      Discipline  NOC           Event  \
        0  1924  Chamonix     Skating  Figure skating  AUT       individual
        1  1924  Chamonix     Skating  Figure skating  AUT       individual
        2  1924  Chamonix     Skating  Figure skating  AUT            pairs
        3  1924  Chamonix   Bobsleigh       Bobsleigh  BEL         four-man
        4  1924  Chamonix  Ice Hockey      Ice Hockey  CAN       ice hockey
        5  1924  Chamonix    Biathlon        Biathlon  FIN  military patrol

          Event gender   Medal
        0            M  Silver
        1            W    Gold
        2            X    Gold
        3            M  Bronze
        4            M    Gold
        5            M  Silver
```

1.3 Explore

So what does the olympic medal data look like? We can peek at the data and its structure by looking at the first few rows. The DataFrame's `head()` method exists for this purpose.

```
In [10]: medals_df.head()
```

```
Out[10]:    Year      City       Sport      Discipline  NOC       Event Event gender  \
         0  1924  Chamonix     Skating  Figure skating  AUT  individual            M   
         1  1924  Chamonix     Skating  Figure skating  AUT  individual            W   
         2  1924  Chamonix     Skating  Figure skating  AUT       pairs            X   
         3  1924  Chamonix   Bobsleigh       Bobsleigh  BEL    four-man            M   
         4  1924  Chamonix  Ice Hockey      Ice Hockey  CAN  ice hockey            M   

             Medal  
         0  Silver  
         1    Gold  
         2    Gold  
         3  Bronze  
         4    Gold  
```

Each row provides:

- The year the medal was awarded
- The city where the games took place
- The sport
- The discipline
- THe nationality of the medal winner
- The specific event
- The gender of the medal winner
- The type of medal

A first logical question might be, for what time period does the data apply? To find out, we turn our attention to the 'Year' column, which we can access directly from the DataFrame object. We can invoke the built-in `min()` and `max()` functions on the column (which is a pandas Series).

```
In [11]: medals_df.Year.min(), medals_df.Year.max()
```

```
Out[11]: ('1924', '2006')
```

To be more precise, we can list all years for which we have medal data.

```
In [14]: medals_df.Year.unique()
```

```
Out[14]: array(['1924', '1928', '1932', '1936', '1948', '1952', '1956', '1960',
                '1964', '1968', '1972', '1976', '1980', '1984', '1988', '1992',
                '1994', '1998', '2002', '2006'], dtype=object)
```

OK. So it looks like we have medal data for all winter olympics from 1924 through 2006.

Next question: How many medals have been awarded? The easiest way to find out is to count the rows using Python's built-in `len()` function, which returns the number of rows.

```
In [15]: len(medals_df)
```

```
Out[15]: 2311
```

ow about a sanity check? What are the distinct medal colors for all medals awarded? And how many of each color were awarded? We can answer these questions without much effort. Here we access the DataFrame's Medal column and invoke built-in functions on the resulting Series.

```
In [16]: medals_df.Medal.unique()

Out[16]: array(['Silver', 'Gold', 'Bronze'], dtype=object)

In [17]: medals_df.Medal.value_counts()

Out[17]: Gold      774
         Silver    773
         Bronze    764
         Name: Medal, dtype: int64
```

Well, these results make sense.

Let us now list all the winter olympic sports and disciplines that have awarded medals. To do this, we first use the DataFrame `grouby()` function to group all medal data by sport and discipline. We then list the keys used to identify each group.

```
In [18]: disciplines = medals_df.groupby(['Sport','Discipline'])

In [22]: disciplines.groups.keys()

Out[22]: dict_keys([('Biathlon', 'Biathlon'), ('Bobsleigh', 'Bobsleigh'), ('Bobsleigh', 'Skeleto
```

1.4 Question 1: Which discipline and event has awarded the most gold medals?

The first step we must take to answer this question is to filter our data. We want to ensure we only consider gold medals. We can apply a filter to our DataFrame by selecting records using a boolean indicator. We store the result in a new DataFrame.

```
In [23]: gold_df = medals_df[medals_df.Medal == 'Gold']

In [24]: gold_df.head()

Out[24]:    Year      City       Sport       Discipline  NOC        Event Event gender  \
         1  1924  Chamonix     Skating  Figure skating  AUT  individual            W
         2  1924  Chamonix     Skating  Figure skating  AUT       pairs            X
         4  1924  Chamonix  Ice Hockey      Ice Hockey  CAN  ice hockey            M
         7  1924  Chamonix     Skating   Speed skating  FIN      10000m            M
         9  1924  Chamonix     Skating   Speed skating  FIN       1500m            M

           Medal
         1  Gold
         2  Gold
         4  Gold
         7  Gold
         9  Gold
```

```
In [25]: gold_df.tail()
```

```
Out[25]:       Year   City   Sport     Discipline  NOC            Event Event gender  \
         2301  2006  Turin  Skiing  Alpine Skiing  USA  Alpine combined            M
         2302  2006  Turin  Skiing  Alpine Skiing  USA     giant slalom            W
         2305  2006  Turin  Skiing      Snowboard  USA         Half-pipe           M
         2307  2006  Turin  Skiing      Snowboard  USA         Half-pipe           W
         2309  2006  Turin  Skiing      Snowboard  USA  Snowboard Cross            M

              Medal
         2301  Gold
         2302  Gold
         2305  Gold
         2307  Gold
         2309  Gold
```

Next, we want to group the gold medals by the `Discipline` and `Event` columns. The result gives us the gold medals awarded by event.

```
In [26]: by_event = gold_df.groupby(['Discipline','Event'])
```

We can get a glimpse of the grouped data using the DataFrame `head()` function.

```
In [29]: by_event.head()
```

```
Out[29]:       Year        City       Sport       Discipline  NOC  \
         1     1924    Chamonix     Skating   Figure skating  AUT
         2     1924    Chamonix     Skating   Figure skating  AUT
         4     1924    Chamonix  Ice Hockey       Ice Hockey  CAN
         7     1924    Chamonix     Skating    Speed skating  FIN
         9     1924    Chamonix     Skating    Speed skating  FIN
         10    1924    Chamonix     Skating    Speed skating  FIN
         14    1924    Chamonix     Skating    Speed skating  FIN
         20    1924    Chamonix     Curling          Curling  GBR
         30    1924    Chamonix      Skiing  Cross Country S  NOR
         33    1924    Chamonix      Skiing  Cross Country S  NOR
         36    1924    Chamonix      Skiing  Nordic Combined  NOR
         38    1924    Chamonix      Skiing      Ski Jumping  NOR
         40    1924    Chamonix    Biathlon         Biathlon  SUI
         41    1924    Chamonix   Bobsleigh        Bobsleigh  SUI
         44    1924    Chamonix     Skating   Figure skating  SWE
         47    1924    Chamonix     Skating    Speed skating  USA
         54    1928  St. Moritz  Ice Hockey       Ice Hockey  CAN
         55    1928  St. Moritz     Skating    Speed skating  FIN
         58    1928  St. Moritz     Skating    Speed skating  FIN
         59    1928  St. Moritz     Skating   Figure skating  FRA
         62    1928  St. Moritz     Skating   Figure skating  NOR
         66    1928  St. Moritz     Skating    Speed skating  NOR
         68    1928  St. Moritz     Skating    Speed skating  NOR
```

```
70    1928  St. Moritz      Skiing  Cross Country S  NOR
73    1928  St. Moritz      Skiing  Nordic Combined  NOR
75    1928  St. Moritz      Skiing      Ski Jumping  NOR
79    1928  St. Moritz     Skating   Figure skating  SWE
81    1928  St. Moritz      Skiing  Cross Country S  SWE
84    1928  St. Moritz   Bobsleigh        Bobsleigh  USA
86    1928  St. Moritz   Bobsleigh         Skeleton  USA
...    ...        ...         ...              ...  ...
2078  2006       Turin      Skiing  Nordic Combined  AUT
2087  2006       Turin   Bobsleigh         Skeleton  CAN
2106  2006       Turin      Skiing  Cross Country S  CAN
2122  2006       Turin      Skiing    Alpine Skiing  CRO
2126  2006       Turin      Skiing  Cross Country S  CZE
2130  2006       Turin      Skiing  Cross Country S  EST
2140  2006       Turin    Biathlon         Biathlon  FRA
2143  2006       Turin    Biathlon         Biathlon  FRA
2151  2006       Turin    Biathlon         Biathlon  GER
2157  2006       Turin    Biathlon         Biathlon  GER
2170  2006       Turin     Skating    Speed skating  GER
2175  2006       Turin      Skiing  Nordic Combined  GER
2185  2006       Turin     Skating    Speed skating  ITA
2194  2006       Turin     Skating   Short Track S.  KOR
2196  2006       Turin     Skating   Short Track S.  KOR
2198  2006       Turin     Skating   Short Track S.  KOR
2199  2006       Turin     Skating   Short Track S.  KOR
2235  2006       Turin    Biathlon         Biathlon  RUS
2236  2006       Turin    Biathlon         Biathlon  RUS
2250  2006       Turin      Skiing  Cross Country S  RUS
2265  2006       Turin      Skiing        Snowboard  SUI
2267  2006       Turin      Skiing        Snowboard  SUI
2268  2006       Turin      Skiing        Snowboard  SUI
2270  2006       Turin    Biathlon         Biathlon  SWE
2281  2006       Turin      Skiing  Cross Country S  SWE
2282  2006       Turin      Skiing  Cross Country S  SWE
2283  2006       Turin      Skiing  Cross Country S  SWE
2301  2006       Turin      Skiing    Alpine Skiing  USA
2305  2006       Turin      Skiing        Snowboard  USA
2309  2006       Turin      Skiing        Snowboard  USA

                                Event Event gender Medal
1                          individual            W  Gold
2                               pairs            X  Gold
4                          ice hockey            M  Gold
7                              10000m            M  Gold
9                               1500m            M  Gold
10                              5000m            M  Gold
14                combined (4 events)            M  Gold
20                            curling            M  Gold
```

```
30                                18km            M  Gold
33                                50km            M  Gold
36                          individual            M  Gold
38                K90 individual (70m)            M  Gold
40                     military patrol            M  Gold
41                            four-man            M  Gold
44                          individual            M  Gold
47                                500m            M  Gold
54                          ice hockey            M  Gold
55                               1500m            M  Gold
58                                500m            M  Gold
59                               pairs            X  Gold
62                          individual            W  Gold
66                               5000m            M  Gold
68                                500m            M  Gold
70                                18km            M  Gold
73                          individual            M  Gold
75                K90 individual (70m)            M  Gold
79                          individual            M  Gold
81                                50km            M  Gold
84                            five-man            M  Gold
86                          individual            M  Gold
...                                ...          ...   ...
2078                 Individual sprint            M  Gold
2087                        individual            M  Gold
2106                      sprint 1.5km            W  Gold
2122                   Alpine combined            W  Gold
2126                              30km            W  Gold
2130  Combined 7.5 + 7.5km mass start            W  Gold
2140                    12.5km pursuit            M  Gold
2143                             7.5km            W  Gold
2151                      10km pursuit            W  Gold
2157                    15km mass start            M  Gold
2170                      Team pursuit            W  Gold
2175                        Individual            M  Gold
2185                      Team pursuit            M  Gold
2194                             1500m            M  Gold
2196                             1500m            W  Gold
2198                       3000m relay            W  Gold
2199                       5000m relay            M  Gold
2235                              15km            W  Gold
2236                       4x6km relay            W  Gold
2250    Combined 15 + 15km mass start            M  Gold
2265             Giant parallel slalom            M  Gold
2267             Giant parallel slalom            W  Gold
2268                   Snowboard Cross            W  Gold
2270                 12,5km mass start            W  Gold
2281                      Sprint 1,5km            M  Gold
```

```
2282                     Team sprint            M  Gold
2283                     Team sprint            W  Gold
2301                 Alpine combined            M  Gold
2305                       Half-pipe            M  Gold
2309                 Snowboard Cross            M  Gold

[296 rows x 8 columns]
```

We can easily tally the number of gold medals per event using the pandas DataFrame `count()` function.

```
In [30]: golds_by_event = by_event.Medal.count()

In [31]: print (golds_by_event)
```

```
Discipline       Event
Alpine Skiing    Alpine combined           2
                 alpine combined          14
                 downhill                 32
                 giant slalom             30
                 slalom                   32
                 super-G                  12
Biathlon         10km                      8
                 10km pursuit              2
                 12,5km mass start         1
                 12.5km pursuit            2
                 15km                      5
                 15km mass start           1
                 20km                     13
                 3x7.5km relay             1
                 4x6km relay               1
                 4x7.5km relay            14
                 7.5km                     5
                 military patrol           1
Bobsleigh        five-man                  1
                 four-man                 18
                 two-man                  20
Cross Country S  10km                     12
                 10km pursuit              5
                 15km                     11
                 15km mass start           4
                 18km                      6
                 20km                      2
                 30km                      5
                 30km mass start          13
                 3x5km relay               5
                                          ..
Freestyle Ski.   moguls                   10
Ice Hockey       ice hockey               23
```

```
Luge              doubles                  13
                  singles                  24
Nordic Combined   Individual                1
                  Individual sprint         1
                  Team                      6
                  individual               19
                  sprint                    1
Short Track S.    1000m                     9
                  1500m                     4
                  3000m relay               5
                  5000m relay               5
                  500m                      9
Skeleton          individual                6
Ski Jumping       K120 individual (90m)    12
                  K120 team (90m)           6
                  K90 individual (70m)     20
Snowboard         Giant parallel slalom     4
                  Half-pipe                 6
                  Snowboard Cross           2
                  giant-slalom              2
Speed skating     10000m                   19
                  1000m                    22
                  1500m                    35
                  3000m                    13
                  5000m                    26
                  500m                     34
                  Team pursuit              2
                  combined (4 events)       1
Name: Medal, Length: 77, dtype: int64
```

Finally, we sort the results. Here are the 10 events with the most gold medals awarded.

```
In [34]: golds_by_event.sort_values(ascending=False)
         golds_by_event.head(10)
```

```
Out[34]: Discipline     Event
         Alpine Skiing  Alpine combined       2
                        alpine combined      14
                        downhill             32
                        giant slalom         30
                        slalom               32
                        super-G              12
         Biathlon       10km                  8
                        10km pursuit          2
                        12,5km mass start     1
                        12.5km pursuit        2
         Name: Medal, dtype: int64
```

Answer: Individual figure skating has awarded the most gold medals to olympians with **40**.

1.5 Question 2: Which country has won the most gold, silver, and bronze medals?

For this question, we need to group and count medals awarded by country code. Because we need sub-totals by medal color, we must group the data by both the NOC and Medal columns. We calculate the medal counts for each group using the resulting DataFrame's size() function, which gives us the number of rows in each group.

```
In [35]: medals_by_country = medals_df.groupby(['NOC','Medal']).size()
```

```
In [36]: print (medals_by_country)
```

```
NOC  Medal
AUS  Bronze     3
     Gold       3
AUT  Bronze    70
     Gold      51
     Silver    64
BEL  Bronze     3
     Gold       1
     Silver     1
BLR  Bronze     3
     Silver     3
BUL  Bronze     3
     Gold       1
     Silver     2
CAN  Bronze    43
     Gold      38
     Silver    38
CHN  Bronze    13
     Gold       4
     Silver    16
CRO  Gold       4
     Silver     3
CZE  Bronze     2
     Gold       3
     Silver     5
DEN  Silver     1
ESP  Bronze     1
     Gold       1
EST  Bronze     1
     Gold       4
     Silver     1
               ..
POL  Silver     3
PRK  Bronze     1
     Silver     1
ROU  Bronze     1
RUS  Bronze    19
     Gold      33
     Silver    24
SLO  Bronze     4
SUI  Bronze    43
     Gold      38
     Silver    37
SVK  Silver     1
SWE  Bronze    44
     Gold      43
     Silver    31
TCH  Bronze    15
     Gold       2
     Silver     8
UKR  Bronze     3
     Gold       1
     Silver     1
URS  Bronze    59
     Gold      78
     Silver    57
USA  Bronze    58
     Gold      78
     Silver    80
UZB  Gold       1
YUG  Bronze     1
     Silver     3
Length: 112, dtype: int64
```

The result is a pandas Series object containing the medal counts by country.

```
In [37]: medals_by_country.head(10)
```

```
Out[37]: NOC  Medal
         AUS  Bronze     3
              Gold       3
         AUT  Bronze    70
              Gold      51
              Silver    64
         BEL  Bronze     3
              Gold       1
              Silver     1
         BLR  Bronze     3
              Silver     3
         dtype: int64
```

We want to convert this Series to a DataFrame containing a column for each medal color. Fortunately, this is easy to do using the Series' unstack() function, which pivots the data for us by creating a column for each medal color.

```
In [38]: medals_by_country_df = medals_by_country.unstack()
         medals_by_country_df.head()
```

```
Out[38]: Medal  Bronze  Gold  Silver
         NOC
         AUS       3.0   3.0     NaN
         AUT      70.0  51.0    64.0
         BEL       3.0   1.0     1.0
         BLR       3.0   NaN     3.0
         BUL       3.0   1.0     2.0
```

Many countries do not have medals of every color, so we replace any missing data with a zero value.

```
In [39]: medals_by_country_df.fillna(0, inplace=True)
```

```
In [40]: medals_by_country_df.head()
```

```
Out[40]: Medal  Bronze  Gold  Silver
         NOC
         AUS       3.0   3.0     0.0
         AUT      70.0  51.0    64.0
         BEL       3.0   1.0     1.0
         BLR       3.0   0.0     3.0
         BUL       3.0   1.0     2.0
```

Now we can answer our question using yet another DataFrame function, `idxmax()`, which gives us the index (in this case, the country code) corresponding to the maximum count for each medal color (column in our DataFrame).

```
In [41]: medals_by_country_df.idxmax()
```

```
Out[41]: Medal
         Bronze    NOR
         Gold      NOR
         Silver    NOR
         dtype: object
```

Norway (NOR) appears to be the winner.

1.5.1 Plot

We can use the popular `matplotlib` package to produce a plot of the results.

Note: Cognitive Class Labs pre-installs many third-party Python libraries and packages. To see a list of these packages, run `!pip freeze` in a code cell.

First, tell the notebook server to render charts inline:

```
In [42]: !pip freeze
```

```
absl-py==0.5.0
alabaster==0.7.10
anaconda-client==1.6.14
anaconda-navigator==1.8.7
```

```
anaconda-project==0.8.2
appdirs==1.4.3
asn1crypto==0.24.0
astor==0.7.1
astroid==1.6.3
astropy==3.0.2
attrs==18.1.0
autobahn==18.9.2
Automat==0.7.0
Babel==2.5.3
backcall==0.1.0
backports.shutil-get-terminal-size==1.0.0
basemap==1.1.0
beautifulsoup4==4.6.0
bitarray==0.8.1
bkcharts==0.2
blaze==0.11.3
bleach==1.5.0
bokeh==0.12.16
boto==2.48.0
boto3==1.7.12
botocore==1.10.84
Bottleneck==1.2.1
certifi==2018.8.24
cffi==1.11.5
chardet==3.0.4
click==6.7
cloudpickle==0.5.3
clyent==1.2.2
colorama==0.3.9
conda==4.5.11
conda-build==3.10.5
conda-verify==2.0.0
constantly==15.1.0
contextlib2==0.5.5
cryptography==2.2.2
cycler==0.10.0
Cython==0.28.2
cytoolz==0.9.0.1
dask==0.17.5
datashape==0.5.4
decorator==4.3.0
distributed==1.21.8
docutils==0.14
entrypoints==0.2.3
et-xmlfile==1.0.1
fastcache==1.0.2
filelock==3.0.4
```

```
Flask==1.0.2
Flask-Cors==3.0.4
future==0.16.0
gast==0.2.0
gevent==1.3.0
gitdb2==2.0.4
GitPython==2.1.11
glob2==0.6
gmpy2==2.0.8
greenlet==0.4.13
grpcio==1.14.1
h5py==2.8.0
heapdict==1.0.0
html5lib==0.9999999
hyperlink==18.0.0
ibm-cos-sdk==2.1.1
ibm-cos-sdk-core==2.3.0
ibm-cos-sdk-s3transfer==2.3.0
ibm-db==2.0.8a0
ibm-db-sa==0.3.3
idna==2.6
imageio==2.3.0
imagesize==1.0.0
incremental==17.5.0
ipykernel==4.8.2
ipython==6.4.0
ipython-genutils==0.2.0
ipython-sql==0.3.9
ipywidgets==7.2.1
isort==4.3.4
itsdangerous==0.24
jdcal==1.4
jedi==0.12.0
Jinja2==2.10
jmespath==0.9.3
jsonschema==2.6.0
jupyter==1.0.0
jupyter-client==5.2.3
jupyter-console==5.2.0
jupyter-core==4.4.0
jupyterlab==0.34.7
jupyterlab-cognos-dashboard-embedded==0.1.0
jupyterlab-github==0.6.1
jupyterlab-launcher==0.13.1
jupyterlab-tutorials==0.2.0
Keras==2.1.5
kiwisolver==1.0.1
lazy-object-proxy==1.3.1
```

```
llvmlite==0.23.1
locket==0.2.0
lxml==4.2.1
Mako==1.0.7
Markdown==2.6.11
MarkupSafe==1.0
matplotlib==2.2.2
mccabe==0.6.1
mistune==0.8.3
mkl-fft==1.0.0
mkl-random==1.0.1
more-itertools==4.1.0
mpmath==1.0.0
msgpack-python==0.5.6
multipledispatch==0.5.0
navigator-updater==0.2.1
nbconvert==5.3.1
nbformat==4.4.0
networkx==2.1
nltk==3.3
nose==1.3.7
notebook==5.5.0
numba==0.38.0
numexpr==2.6.5
numpy==1.14.3
numpydoc==0.8.0
odo==0.5.1
olefile==0.45.1
openpyxl==2.5.3
packaging==17.1
pandas==0.23.0
pandocfilters==1.4.2
parso==0.2.0
partd==0.3.8
path.py==11.0.1
pathlib2==2.3.2
patsy==0.5.0
pep8==1.7.1
pexpect==4.5.0
pickleshare==0.7.4
Pillow==5.2.0
pkginfo==1.4.2
pluggy==0.6.0
ply==3.11
prettytable==0.7.2
prompt-toolkit==1.0.15
protobuf==3.5.2
psutil==5.4.5
```

```
ptyprocess==0.5.2
py==1.5.3
pyarrow==0.7.1
pyasn1==0.4.4
pyasn1-modules==0.2.2
pycodestyle==2.4.0
pycosat==0.6.3
pycparser==2.18
pycrypto==2.6.1
pycurl==7.43.0.1
pydotplus==2.0.2
pyflakes==1.6.0
Pygments==2.2.0
pygpu==0.7.6
PyHamcrest==1.9.0
pylint==1.8.4
pyodbc==4.0.23
pyOpenSSL==18.0.0
pyparsing==2.2.0
pyproj==1.9.5.1
pyshp==1.2.12
PySocks==1.6.8
pytest==3.5.1
pytest-arraydiff==0.2
pytest-astropy==0.3.0
pytest-doctestplus==0.1.3
pytest-openfiles==0.3.0
pytest-remotedata==0.2.1
python-dateutil==2.7.3
pytz==2018.4
PyWavelets==0.5.2
PyYAML==3.12
pyzmq==17.0.0
QtAwesome==0.4.4
qtconsole==4.3.1
QtPy==1.4.1
quilt==2.9.4
raven==6.9.0
requests==2.18.4
rope==0.10.7
ruamel-yaml==0.15.35
s3transfer==0.1.13
scikit-image==0.13.1
scikit-learn==0.19.1
scipy==1.1.0
seaborn==0.8.1
Send2Trash==1.5.0
service-identity==17.0.0
```

```
simplegeneric==0.8.1
singledispatch==3.4.0.3
six==1.11.0
smmap2==2.0.4
snowballstemmer==1.2.1
sortedcollections==0.6.1
sortedcontainers==1.5.10
Sphinx==1.7.4
sphinxcontrib-websupport==1.0.1
spyder==3.2.8
SQLAlchemy==1.2.7
sqlparse==0.2.4
statsmodels==0.9.0
sympy==1.1.1
tables==3.4.3
tblib==1.3.2
tensorboard==1.8.0
tensorflow==1.8.0
termcolor==1.1.0
terminado==0.8.1
testpath==0.3.1
Theano==1.0.3
toolz==0.9.0
torch==0.4.1
torchvision==0.2.1
tornado==5.0.2
tqdm==4.26.0
traitlets==4.3.2
Twisted==18.7.0
txaio==18.8.1
typing==3.6.4
unicodecsv==0.14.1
urllib3==1.22
watson-developer-cloud==1.4.1
wcwidth==0.1.7
webencodings==0.5.1
Werkzeug==0.14.1
widgetsnbextension==3.2.1
wrapt==1.10.11
xlrd==1.1.0
XlsxWriter==1.0.4
xlwt==1.3.0
zict==0.1.3
zope.interface==4.5.0
```

telling the notebook server to render charts inline:

```
In [43]: %matplotlib inline
```

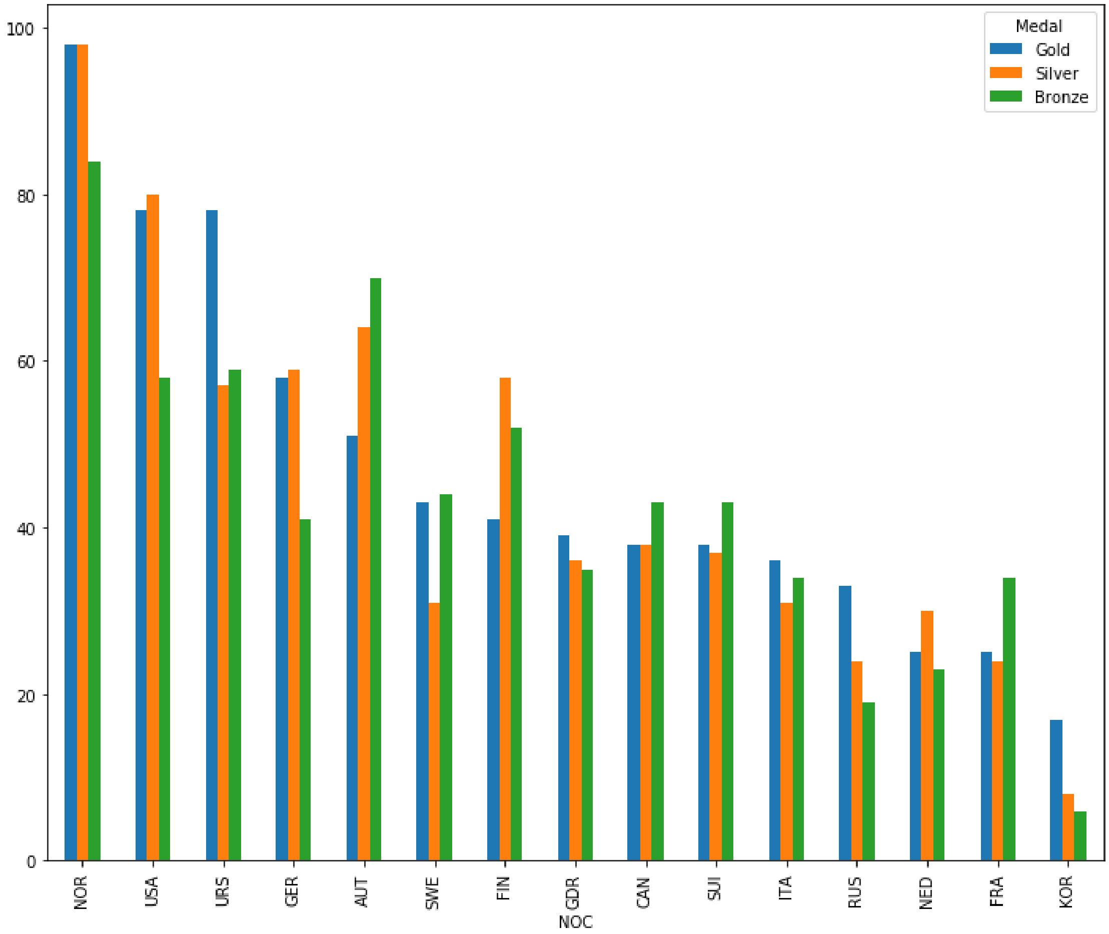
We sort the results by highest gold medal count.

```
In [44]: medals_by_country_df.sort_values('Gold', ascending=False, inplace=True)
```

Now we can use the DataFrame `plot()` function to produce our plot. We plot individual medal counts for the top 15 countries.

```
In [45]: medals_by_country_df[['Gold','Silver','Bronze']][:15]\
         .plot(kind='bar', figsize=(12,10))
```

```
Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff63e6d3f98>
```



The plot confirms that Norway has been on every step of the winter olympic podium more than any other country.

Let's take a look at another plot: the olympic medals won by Norway over time. To do this, we must filter our original medal data (containing all medal colors) to just those won by Norway, and group by Year. We then count the number of medals for each year. Amazingly, we can do all this in a single line of code.

```
In [46]: nor_medals_year = medals_df[medals_df.NOC == 'NOR'].groupby('Year').size()
In [47]: print (nor_medals_year)
```

```
Year
1924    17
1928    15
1932    10
1936    15
1948    10
1952    16
1956     4
1960     6
1964    15
1968    14
1972    12
1976     7
1980    10
1984     9
1988     5
1992    20
1994    26
1998    25
2002    25
2006    19
dtype: int64
```

```
In [48]: nor_medals_year
```

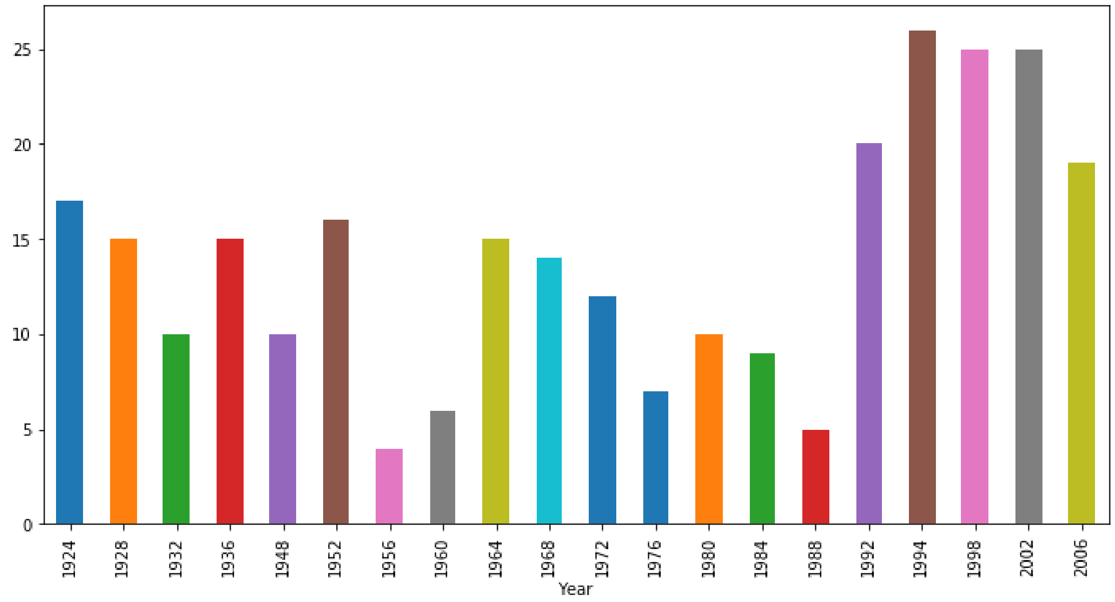
```
Out[48]: Year
         1924    17
         1928    15
         1932    10
         1936    15
         1948    10
         1952    16
         1956     4
         1960     6
         1964    15
         1968    14
         1972    12
         1976     7
         1980    10
         1984     9
         1988     5
         1992    20
         1994    26
         1998    25
         2002    25
```

```
2006    19
dtype: int64
```

We can plot the resulting pandas Series using it's built-in `plot()` function, which is just a convenience function that wraps `matplotlib`.

```
In [49]: nor_medals_year.plot(kind='bar', figsize=(12,6))
```

```
Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff63e6d36d8>
```



1.6 Third-Party Visualization Libraries

It is relatively easy to create plots quickly using `pandas` and `matplotlib`. If you want to take your plots to the next level in terms of enhanced presentation, it is possible to install third-party libraries to help you.

One such library is prettyplotlib, which enhances `matplotlib` plots with better default colors, etc. This can be extremely valuable if your notebook is to be used for presentations.

Note: You have the ability to install Python packages in your Cognitive Class Labs environment. Just use the `pip` package installer.

Here we use the shorthand IPython cell magic (!) to invoke `pip` to install `prettyplotlib`.

```
In [50]: # Install third party color palette
         !pip install prettyplotlib
```

```
Collecting prettyplotlib
  Downloading https://files.pythonhosted.org/packages/f2/89/35079781fe5f8c4e5258b88bb0a80d4a2028
    100% || 706kB 18.5MB/s
Requirement already satisfied: matplotlib>=1.2.1 in /home/jupyterlab/conda/lib/python3.6/site-pa
```

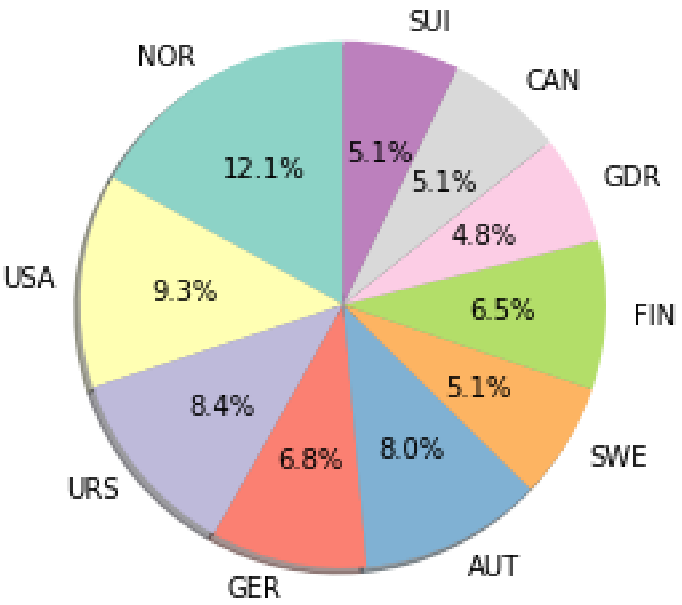
```
Collecting brewer2mpl>=1.3.1 (from prettyplotlib)
  Downloading https://files.pythonhosted.org/packages/84/57/00c45a199719e617db0875181134fcb3aeef
Requirement already satisfied: numpy>=1.7.1 in /home/jupyterlab/conda/lib/python3.6/site-package
Requirement already satisfied: cycler>=0.10 in /home/jupyterlab/conda/lib/python3.6/site-package
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /home/jupyterlab/cond
Requirement already satisfied: python-dateutil>=2.1 in /home/jupyterlab/conda/lib/python3.6/site
Requirement already satisfied: pytz in /home/jupyterlab/conda/lib/python3.6/site-packages (from
Requirement already satisfied: six>=1.10 in /home/jupyterlab/conda/lib/python3.6/site-packages (
Requirement already satisfied: kiwisolver>=1.0.1 in /home/jupyterlab/conda/lib/python3.6/site-pa
Requirement already satisfied: setuptools in /home/jupyterlab/conda/lib/python3.6/site-packages
Building wheels for collected packages: prettyplotlib
  Running setup.py bdist_wheel for prettyplotlib ... done
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/76/ad/45/9fcfb9e97ecccc850d8b2fb20b8d6
Successfully built prettyplotlib
distributed 1.21.8 requires msgpack, which is not installed.
Installing collected packages: brewer2mpl, prettyplotlib
Successfully installed brewer2mpl-1.4.1 prettyplotlib-0.1.7
```

Now we create a color palette...

```
In [51]: # Use color palette "Set3" from http://bl.ocks.org/mbostock/5577023
         import brewer2mpl
         set3 = brewer2mpl.get_map('Set3', 'qualitative', 10).mpl_colors
```

...and use it to generate a pretty pie chart showing the percent of all medals awarded to the top 10 countries.

```
In [52]: # Plot the top 10 countries based on podium appearances.
         import pylab as plt
         # Calculate the total medals for each country
         t = medals_by_country_df.sum(axis=1)
         # Limit to top 10 medal-winning countries
         t.sort_values(ascending=False)
         awards = t[:10]
         countries = awards.index.values
         total = float(t.sum())
         # Create a pie chart
         pct = lambda x: '{p:1.1f}%'.format(p=(x*sum(awards)/100)/total*100)
         plt.pie(awards, labels=countries, shadow=True,
                 autopct=pct, colors=set3, startangle=90)
         # Set aspect ratio to be equal so that pie is drawn as a circle.
         plt.axis('equal')
         plt.show()
```



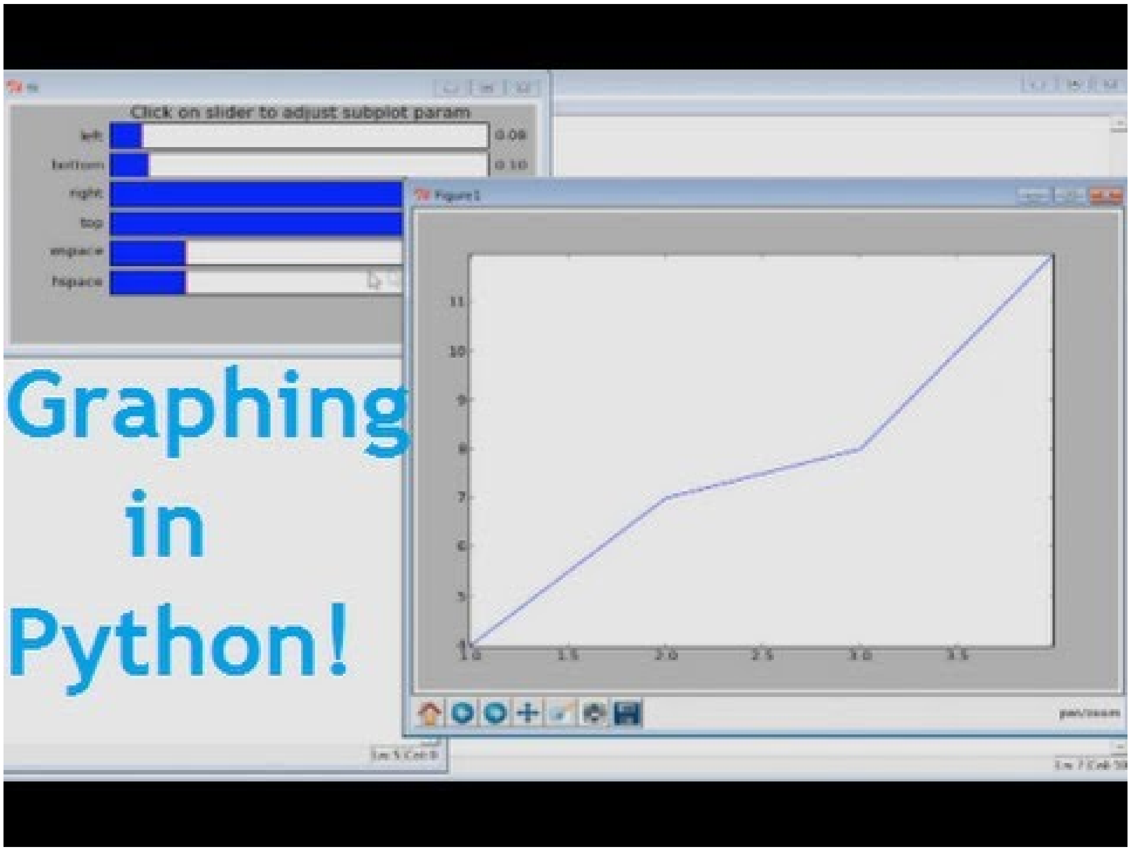
1.7 Online Tutorials

If you need further instruction on how to create visualizations in IPython notebook, there is content available online. Here is an example. The video below represents a resource for visual programming using `matplotlib`.

Note: The code cell below shows one way you can embed a video in your notebook.

```
In [53]: from IPython.display import YouTubeVideo
         # Matplotlib Python Tutorial Part 1: Basics and your first Graph!
         # Tutorial series by Sentdex - http://sentdex.com/about-us/
         YouTubeVideo('wAwQ-noyB98')

   Out[53]:
```



1.8 Next: Organize

Our next tutorial topic will focus on how Cognitive Class Labs can help you organize your work. Visit the Welcome page to download **Tutorial #3 - Organize**.