Introduction to Python

Subrata Paul

6/3/2020

Data types

```
l = [1,2,3,4]
t = (1,2,3,4)
s = {1,2,3,4}
d = {'a':'something', 'b':'something else'}
type(l)
```

```
## <class 'list'>
```

```
type(t)
```

```
## <class 'tuple'>
```

```
type(s)
```

```
## <class 'set'>
```

```
type(d)
```

```
## <class 'dict'>
```

The `range` function

```
range(1,10)
```

```
## range(1, 10)
```

```
type(range(1,10))
```

```
## <class 'range'>
```

```
list(range(1,10))
```

```
## [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
range(10)
```

```
## range(0, 10)
```

```
list(range(10))
```

```
## [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
list(range(1,10,2))
```

```
## [1, 3, 5, 7, 9]
```

Libraries

```
import numpy as np
np.linspace(2,3,5)
```

```
## array([2.  , 2.25, 2.5 , 2.75, 3.  ])
```

```
np.arange(1,10)
```

```
## array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
np.sqrt(100)
```

```
## 10.0
```

```
x = np.array([1,2,3,4])
np.sqrt(x)
```

```
## array([1.        , 1.41421356, 1.73205081, 2.        ])
```

```
np.sum(x)
```

```
## 10
```

The `rep` function in `R`

```
[1,2] * 3
```

```
## [1, 2, 1, 2, 1, 2]
```

```
[1,2,3] + [2,3,4]
#[1,2,3] + 1 Does not work
```

```
## [1, 2, 3, 2, 3, 4]
```

```
np.array([1,2,3]) + 1
```

```
## array([2, 3, 4])
```

Logical

```
x = np.random.normal(loc = 5, scale = 3, size = 10)
x
```

```
## array([-1.28484483, 10.33549025,  3.32330382,  5.1917064 ,  6.41168262,
##         9.7346145 ,  5.26552406,  3.00029804,  6.90064899, 10.31778038])
```

```
x > 5
```

```
## array([False,  True, False,  True,  True,  True,  True, False,  True,
##         True])
```

Commonly-used Functions

```
x = np.random.normal(loc = 5, scale = 3, size = 10)
np.sum(x)
```

```
## 50.76907661221843
```

```
np.std(x)
```

```
## 3.8053914459548155
```

```
np.abs(x)
```

```
## array([ 7.52754874,  7.29607673,  4.95793231,  5.85435887,  1.02213247,
##         0.56686051,  9.02324184, 11.66286083,  3.67282345,  1.2295058 ])
```

```
np.var(x)
```

```
## 14.481004056946082
```

```
np.log(x)
```

```
## array([ 2.01856946,  1.98733677,  1.60098878,  1.76718649,         nan,
##        -0.56764203,  2.19980368,  2.4564095 ,  1.3009607 ,  0.2066123 ])
##
## /Users/spaul/Library/r-miniconda/envs/r-reticulate/bin/python:1: RuntimeWarning: invalid val
```

Function

```
def five_point_summary(x):
  mini = np.min(x)
  maxi = np.max(x)
  q = np.quantile(x,[.25,.50,.75])
  return [mini] + list(q) + [maxi]
x = np.random.normal(loc = 5, scale = 3, size = 10)
five_point_summary(x)
```

```
## [-1.6936253268348889, 0.5257096814562654, 4.577801774829398, 6.725092755021153, 10.098419108
```

Functions related to statistical distribution

```
import scipy.stats as st
#loc = mean, scale = sd
st.norm.cdf(3, loc = 1, scale = 2) #pnorm
```

```
## 0.8413447460685429
```

```
st.norm.pdf(3, loc = 1, scale = 2) #dnorm
```

```
## 0.12098536225957168
```

```
st.norm.ppf(0.8413, loc = 1, scale = 2) # qnorm
```

```
## 2.9996301872294895
```

```
st.norm.rvs(loc =1, scale = 2, size = 10) # rnorm
```

```
## array([ 2.45493967,  4.10533824,  1.76931079,  0.78393315,  3.24838318,
##         0.8134111 ,  0.85946704, -0.55453708,  1.15296128,  0.24683208])
```

```
st.norm.rvs(10) # Standard normal
```

```
## 10.833291650203595
```

```
st.norm.rvs(size = 10)
```

```
## array([-0.92107289, -0.88847123, -0.83235645, -0.83131681,  1.29286413,
```

Data Frames

```
import pandas as pd
mydataframe = pd.DataFrame({'d':[1,2,3,4], 'e':['red','white','blue',np.NAN], 'f':[True, True,
mydataframe
```

```
##    d      e      f
## 0  1    red   True
## 1  2  white   True
## 2  3   blue  False
## 3  4    NaN   True
```

Rename columns

```
mydataframe.columns = ['ID','Color','Passed']
mydataframe
```

```
##    ID  Color  Passed
## 0   1    red    True
## 1   2  white    True
## 2   3   blue   False
## 3   4    NaN    True
```

Access columns

```
mydataframe.ID
```

```
## 0    1
## 1    2
## 2    3
## 3    4
## Name: ID, dtype: int64
```

```
mydataframe[['ID','Color']]
```

```
##    ID  Color
## 0   1    red
## 1   2  white
## 2   3   blue
## 3   4    NaN
```

Indexing and Slicing

```
mydataframe.loc[1:2,['ID','Passed']]
```

```
##    ID  Passed
## 1   2    True
## 2   3   False
```

```
mydataframe.loc[:,'ID']
```

```
## 0    1
## 1    2
## 2    3
## 3    4
## Name: ID, dtype: int64
```

Indexing and Slicing

```
mydataframe.iloc[1:2,0]
```

```
## 1    2
## Name: ID, dtype: int64
```

```
mydataframe.iloc[:,[0,1]]
```

```
##    ID  Color
## 0   1    red
## 1   2  white
## 2   3   blue
## 3   4    NaN
```

Importing Data

```
help(pd.read_csv)
```

```
## Help on function read_csv in module pandas.io.parsers:
##
## read_csv(filepath_or_buffer, sep=',', delimiter=None, header='infer', names=None, index_col=
##     Read a comma-separated values (csv) file into DataFrame.
##
##     Also supports optionally iterating or breaking of the file
##     into chunks.
##
##     Additional help can be found in the online docs for
##     `IO Tools <http://pandas.pydata.org/pandas-docs/stable/io.html>`_.
##
##     Parameters
##     ----------
##     filepath_or_buffer : str, path object, or file-like object
##         Any valid string path is acceptable. The string could be a URL. Valid
##         URL schemes include http, ftp, s3, and file. For file URLs, a host is
##         expected. A local file could be: file://localhost/path/to/table.csv.
##
##         If you want to pass in a path object, pandas accepts either
##         ``pathlib.Path`` or ``py._path.local.LocalPath``.
```

Sample data analysis

```
pima = pd.read_csv('./data/pima.txt', header = 0, sep = '\t')
pima.head()
```

```
##    pregnant  glucose  diastolic  triceps  insulin   bmi  diabetes  age  test
## 0         6      148         72       35        0  33.6     0.627   50     1
## 1         1       85         66       29        0  26.6     0.351   31     0
## 2         8      183         64        0        0  23.3     0.672   32     1
## 3         1       89         66       23       94  28.1     0.167   21     0
## 4         0      137         40       35      168  43.1     2.288   33     1
```

Assign NULL for missing data

```
pima.diastolic.sort_values().head()
```

```
## 347    0
## 494    0
## 222    0
## 81     0
## 78     0
## Name: diastolic, dtype: int64
```

```
pima.diastolic[pima.diastolic==0] = np.NaN
```

```
## /Users/spaul/Library/r-miniconda/envs/r-reticulate/bin/python:1: SettingWithCopyWarning:
## A value is trying to be set on a copy of a slice from a DataFrame
##
## See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.h
```

```
pima.diastolic.sort_values(na_position = 'first').head()
```

```
## 7    NaN
## 15   NaN
## 49   NaN
## 60   NaN
## 78   NaN
## Name: diastolic, dtype: float64
```

Assign NaN to missing data

```
pima.glucose[pima.glucose==0] = np.NaN
pima.triceps[pima.triceps==0] = np.NaN
pima.insulin[pima.insulin==0] = np.NaN
pima.bmi[pima.bmi==0] = np.NaN
```

Assign NaN to missing data

```
pima = pd.read_csv('./data/pima.txt', header = 0, sep = '\t')
pima.replace(0,np.nan, inplace = True)
pima.head()
```

```
##    pregnant  glucose  diastolic  triceps  insulin   bmi  diabetes  age  test
## 0       6.0    148.0       72.0     35.0      NaN  33.6     0.627   50   1.0
## 1       1.0     85.0       66.0     29.0      NaN  26.6     0.351   31   NaN
## 2       8.0    183.0       64.0      NaN      NaN  23.3     0.672   32   1.0
## 3       1.0     89.0       66.0     23.0     94.0  28.1     0.167   21   NaN
## 4       NaN    137.0       40.0     35.0    168.0  43.1     2.288   33   1.0
```

Plotting Histogram

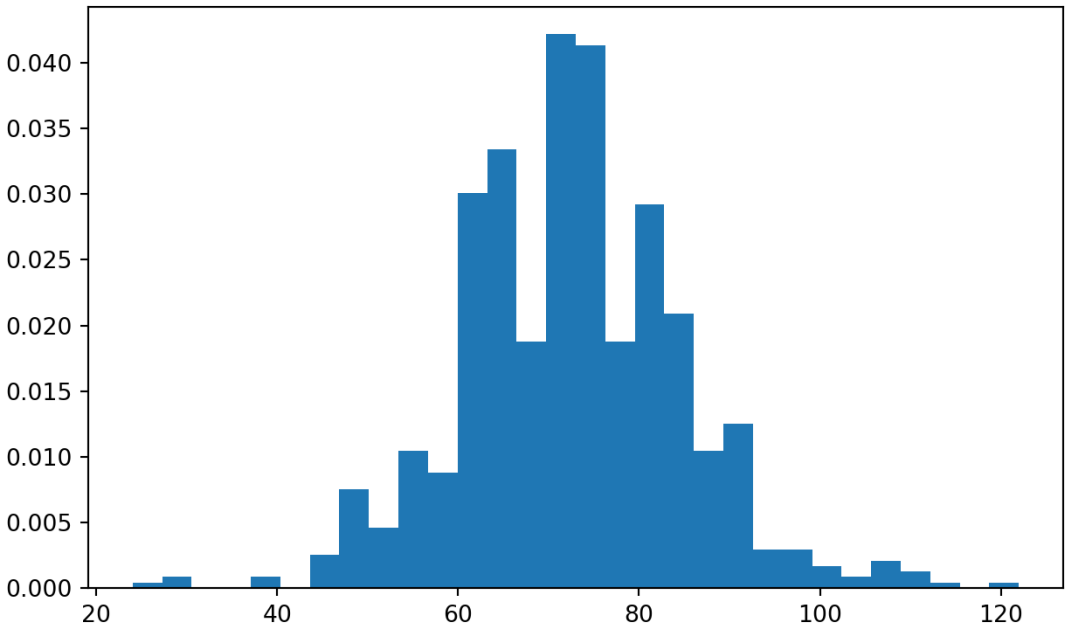
```
import matplotlib.pyplot as plt
_ = plt.hist(pima.diastolic)
```

```
## /Users/spaul/Library/r-miniconda/envs/r-reticulate/lib/python3.6/site-packages/numpy/lib/his
##   keep = (tmp_a >= first_edge)
## /Users/spaul/Library/r-miniconda/envs/r-reticulate/lib/python3.6/site-packages/numpy/lib/his
##   keep &= (tmp_a <= last_edge)
```

```
plt.show()
```

Plotting Histogram

```
import matplotlib.pyplot as plt
_ =plt.hist(pima.diastolic[~np.isnan(pima.diastolic)], bins = 30, density = True)
plt.show()
```



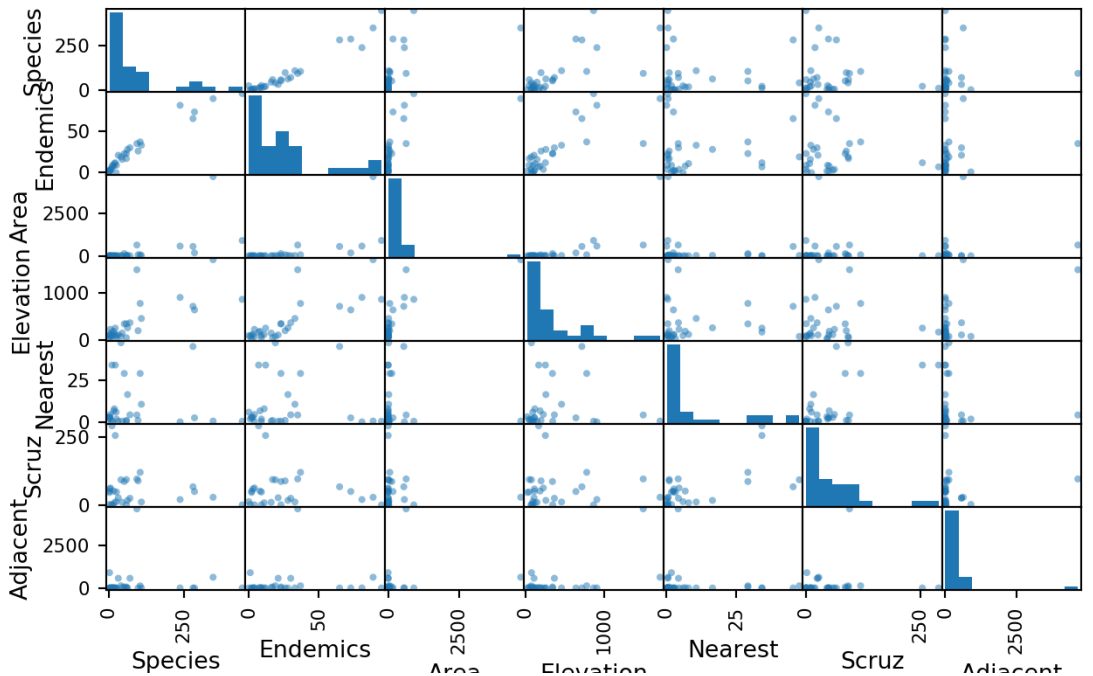
Galapagos Data

```
gala = pd.read_csv('./data/gala.txt',header = 0, sep = '\t')
gala.head()
```

```
##             Species  Endemics    Area  Elevation  Nearest  Scruz  Adjacent
## Baltra           58        23   25.09        346      0.6    0.6      1.84
## Bartolome        31        21    1.24        109      0.6   26.3    572.33
## Caldwell          3         3    0.21        114      2.8   58.7      0.78
## Champion         25         9    0.10         46      1.9   47.4      0.18
## Coamano           2         1    0.05         77      1.9    1.9    903.82
```

Plot data

```
_=pd.plotting.scatter_matrix(gala)
plt.show()
```



Fit multiple regression model

```
import statsmodels.api as sm
Y = gala.Species
X = gala.iloc[:,2:]
X['const'] = np.ones(X.shape[0])
lmod = sm.OLS(Y, X)
res = lmod.fit()
print(res.summary())
```

```
##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:                Species   R-squared:                       0.766
## Model:                            OLS   Adj. R-squared:                  0.717
## Method:                 Least Squares   F-statistic:                     15.70
## Date:                Fri, 17 Jul 2020   Prob (F-statistic):           6.84e-07
## Time:                        17:32:54   Log-Likelihood:                -162.54
## No. Observations:                  30   AIC:                             337.1
## Df Residuals:                      24   BIC:                             345.5
## Df Model:                           5
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
```

Fit multiple regression model

```
import statsmodels.api as sm
Y = gala.Species
X = gala.iloc[:,2:]
X['const'] = np.ones(X.shape[0])
lmod = sm.OLS(Y, X)
res = lmod.fit()
print(res.summary())
```

```
##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:                Species   R-squared:                       0.766
## Model:                            OLS   Adj. R-squared:                  0.717
## Method:                 Least Squares   F-statistic:                     15.70
## Date:                Fri, 17 Jul 2020   Prob (F-statistic):           6.84e-07
## Time:                        17:32:54   Log-Likelihood:                -162.54
## No. Observations:                  30   AIC:                             337.1
## Df Residuals:                      24   BIC:                             345.5
## Df Model:                           5
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
```

Parameters

```
res.params
```

```
## Area        -0.023938
## Elevation    0.319465
## Nearest      0.009144
## Scruz       -0.240524
## Adjacent    -0.074805
## const        7.068221
## dtype: float64
```

Standard Errors

```
res.bse
```

```
## Area          0.022422
## Elevation     0.053663
## Nearest       1.054136
## Scruz         0.215402
## Adjacent      0.017700
## const        19.154198
## dtype: float64
```

Fitted Values

```
res.fittedvalues
```

```
## Baltra          116.725946
## Bartolome        -7.273154
## Caldwell         29.330659
## Champion         10.364266
## Coamano         -36.383916
## Daphne.Major     43.087705
## Daphne.Minor     33.919668
## Darwin           -9.018992
## Eden             28.314202
## Enderby          30.785943
## Espanola         47.656487
## Fernandina       96.989598
## Gardner1         -4.033276
## Gardner2         64.633796
## Genovesa         -0.497176
## Isabela         386.403558
## Marchena         88.694540
## Onslow            4.037233
## Pinta           215.679486
## Pinzon          150.475375
```