付録. データ分析概要

- 概要
  - Pythonでのデータ分析について学びます。

- 学習内容
  - 1. Numpy
  - 2. テスト結果を集計する
  - 3. ネットワーク機能
  - 4. ネットワークとデータ分析
  - 5. 統計計算
  - 6. 金融データ

1. Numpy

- Numpy
  - Pythonでデータ分析を行う時に最も使用されるパッケージの代表はNumPyです。
  - NumPyは、データ分析に欠かせない配列や行列の演算を高速に行うライブラリです。
  - 大量のデータを扱う場合は、Python標準のリストだと処理の実行速度が問題になることがあります。
  - NumPyは、C言語の処理性能とほぼ同等の速度で処理できるように開発されました。 また、データ分析に欠かせない、統計計算の関数も多数備えています。

2. テスト結果を集計する

- テストの結果を集計してみましょう。
  - テストの点数をNumpyを使って、平均や最高点、最低点、標準偏差を計算してみます。

```
import numpy as np

#テストのデータをNumPyを使い準備する。
x = np.array([44,10,30,23,23,35,35,40,41,42,50,51,52,55, ¥
              60,61,62,63,64,65,70,70,65,80,34,65,76,85,92])

print('受験者数=',np.count_nonzero(x))
print('最高点=',np.max(x))
print('最低点=',np.min(x))
print('平均点',np.average(x))
print('標準偏差',np.std(x))
```

2. テスト結果を集計する

- 偏差値の計算
  - 偏差値を求めてみます。 各点数をx,平均をx 標準偏差をSとして、偏差値yを求める式は 以下のようになります。

$$y = \frac{10(x - \bar{x})}{S} + 50$$

```
def std_score(a):
    return np.round_(50+10*(a-np.average(x))/np.std(x))

print('得点,偏差値')
print('--------------')
for data in x:

    print('{0}, {1}'.format(data, std_score(data)))
```

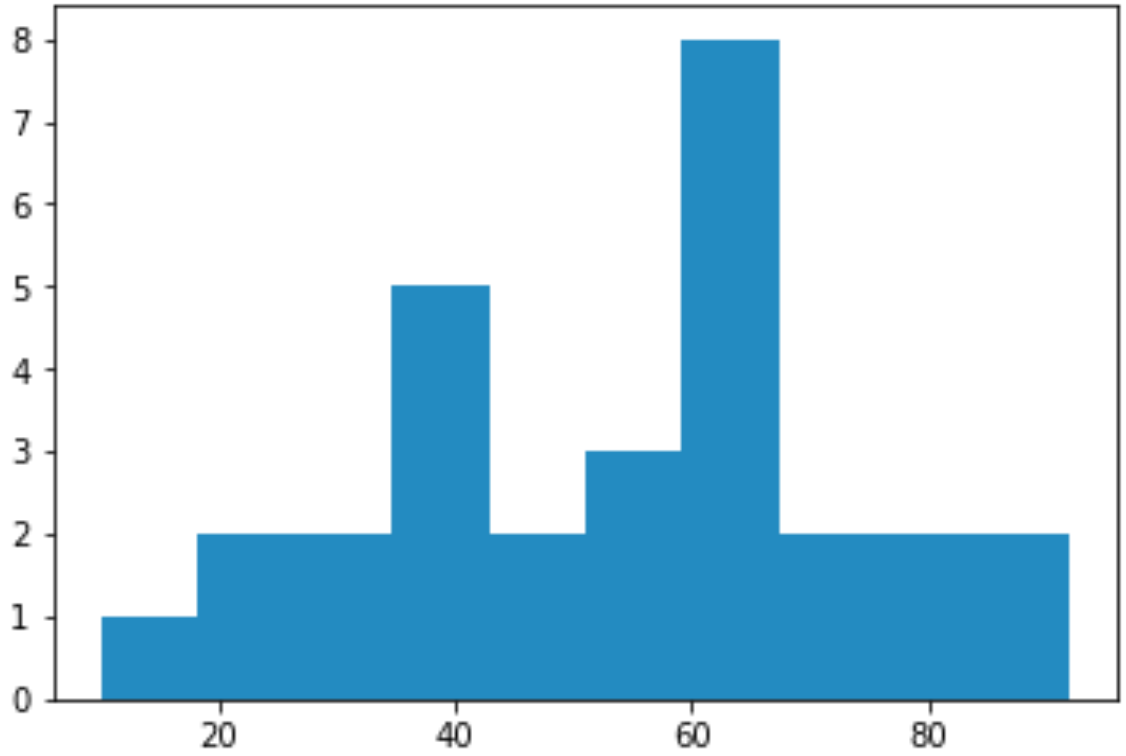
2. テスト結果を集計する

- テストの結果を可視化してみます。
  - グラフ化するためのパッケージは matplotlib と呼ばれるパッケージを利用します。テストの得点の度数分布表を作ってみます。

```
import numpy as np

#テストのデータをNumPyを使い準備する。
x = np.array([44,10,30,23,23,35,35,40,41,42,50,51,52,55, ¥
              60,61,62,63,64,65,70,70,65,80,34,65,76,85,92])


#度数分布表の出力 binsは度数としてまとめる数
plt.hist(x,bins=10)
```

2. テスト結果を集計する

- テストの結果を可視化してみます。
  - 以下のようなヒストグラム（度数分布表）が表示されます。



2. テスト結果をシュミレートする

- テストの結果をシュミレーションします。
  - Numpyには指定された平均、標準偏差に従った乱数データの発生機能があります。

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
#平均50, 標準偏差10で乱数データ10000個発生させる
X = np.random.normal(50, 10, 10000)
#小数点以下切り捨て
X = np.trunc(X)

print('count:',np.count_nonzero(X))
print('max:',np.max(X))
print('min:',np.min(X))
print('average:',np.average(X))
print('std:',np.std(X))

plt.hist(X,bins=10)
```

4. 統計計算

- 簡単な統計計算を行ってみます。
  - つい最近発表された日本の人口を分析してみましょう。 国勢調査のデータをグラフ化してみます。
  - 人口の推移は、国政調査の結果から、CSVファイルとして用意します。

```
17,127768
18,127901
19,128033
20,128084
21,128032
22,128057
23,127799
24,127515
25,127298
26,127083
27,127110
28,126930
```

4. 統計計算

- 簡単な統計計算を行ってみます。
  - ＣＳＶファイルをダウンロードします。

```
from urllib.request import urlopen

url  = 'http://www.lighthouse-w5.com/p/data/pop.csv'
html = urlopen(url)

encode = html.info().get_content_charset(failobj='utf-8')

text_byte = html.read()
text = text_byte.decode(encode)

with open('pop.csv','w') as f:
    f.write(text)
```

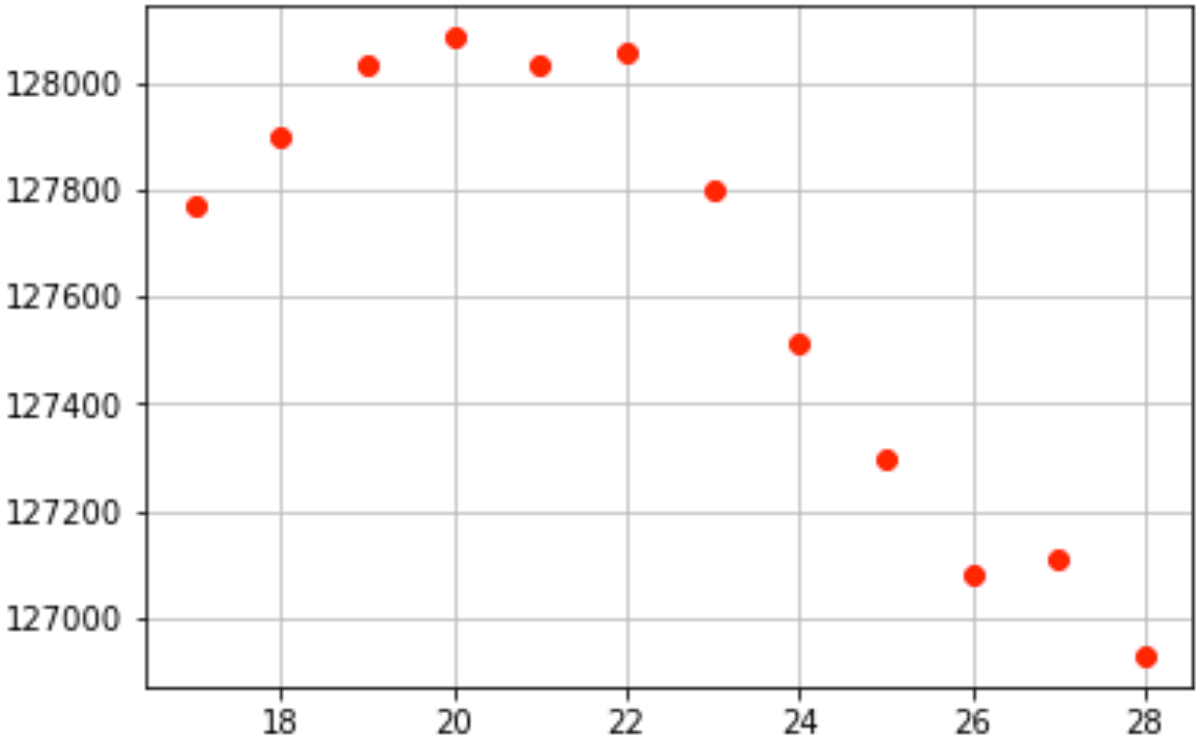
4. 統計計算

- 簡単な統計計算を行ってみます。
  - 可視化します。散布図を描いてみます。

```
import numpy as np
import matplotlib.pyplot as plt

data = np.genfromtxt('pop.csv',delimiter=',’¥
       ,dtype={'names':('year','population')¥
       ,'formats':('f','f')})

print(data)

plt.grid()
plt.plot(data['year'],data['population'],'ro')
```

4. 統計計算

- 簡単な統計計算を行ってみます。
  - 可視化します。散布図を描いてみます。



4 統計計算

- 簡単な統計計算を行ってみます。
  - 2つのデータ間での関係を分析する手法、回帰分析という 手法で人口の減少を予測してみます。
  - 今回は回帰分析の中でも 単純な、線形回帰分析手法を使ってみます。
  - pythonには、統計計算を行うライブラリscipyが用意されているので回帰分析も容易に行うことができます。

4. 統計計算

- 簡単な統計計算を行ってみます。
  - 可視化します。散布図を描いてみます。

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
data = np.genfromtxt('pop.csv',delimiter=','¥
            ,dtype={'names':('year','population'),¥
                    'formats':('f','f')})

#相関係数を求める。回帰直線の傾きがslope,切片がintercept
slope,intercept,r_value,p_value,std_err =
    stats.linregress(data['year'],data['population'])

print(slope,intercept)
x = np.arange(20,40,1)
plt.grid()
plt.plot(x, slope * x + intercept)

plt.plot(data['year'],data['population'],'ro')
```

4 統計計算

- 簡単な統計計算を行ってみます。
  - 可視化します。散布図と回帰直線を描画した例