# assignment3

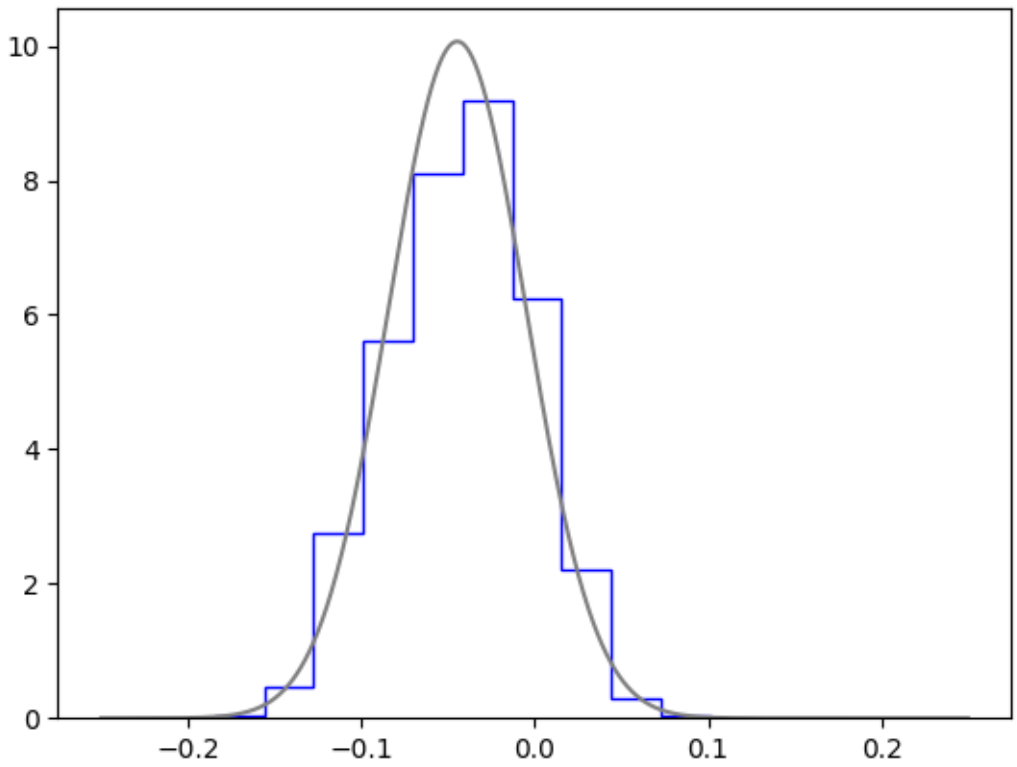
January 30, 2023

[2]:
```
import numpy as np
from scipy.stats import norm
from matplotlib import pyplot as plt
from astroML.resample import bootstrap
from astroML import stats
import pandas as pd
import scipy
from scipy.stats import chi2
```

## 0.1 Question 1

[3]:
```
m = 1000  # number of points
n = 10000  # number of bootstraps

# sample values from a normal distribution
np.random.seed(123)
data = norm(0, 1).rvs(m)

# Compute bootstrap resamplings of data
median ,sigmag = bootstrap(data, n, stats.median_sigmaG, kwargs=dict(axis=1))

# # Compute the theoretical expectations for the two distributions
x = np.linspace(-0.25, 0.25, 1000)

sigma1 = np.sqrt(np.pi/(2*m))
pdf1 = norm(np.mean(median), sigma1).pdf(x)
```

[4]:
```
plt.hist(median, bins=10, density=True, histtype='step', color='blue')
plt.plot(x, pdf1, color='gray')
plt.show()
```




### 0.2 Question 2

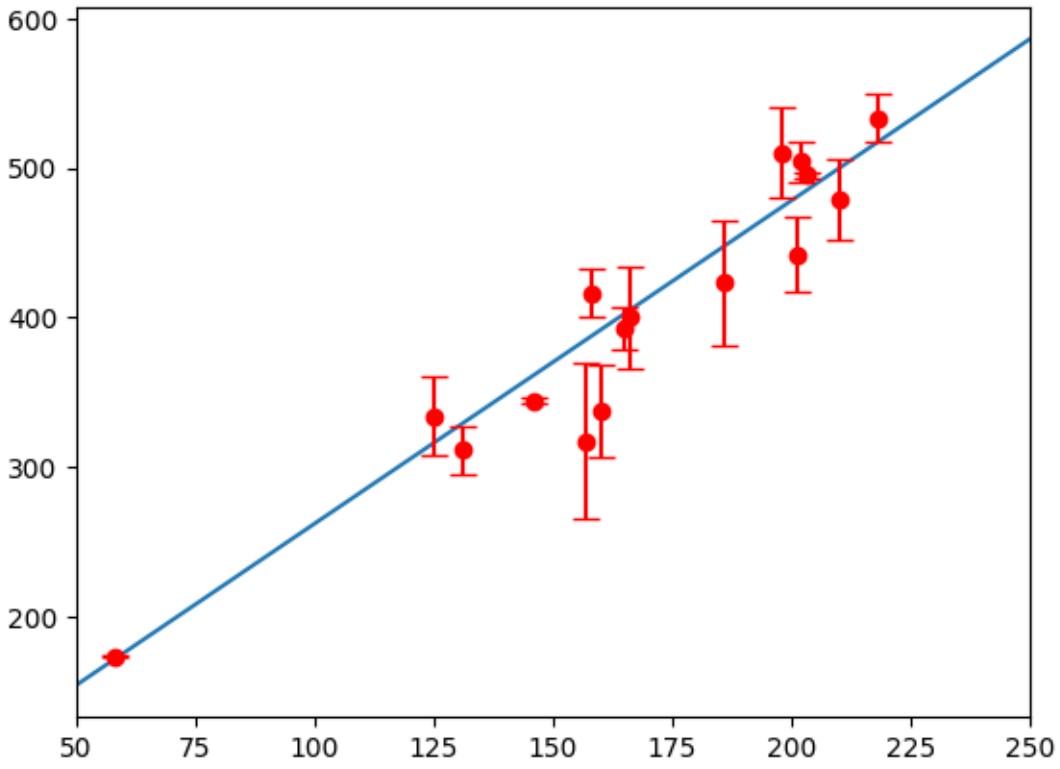
[5]:
```
dataset = pd.read_csv("input.txt" , sep=" ")
dataset = pd.DataFrame(dataset)
print(dataset)
```

```
      x    y  sigmay
0   203  495       2
1    58  173       1
2   210  479      27
3   202  504      14
4   198  510      30
5   158  416      16
6   165  393      14
7   201  442      25
8   157  317      52
9   131  311      16
10  166  400      34
11  160  337      31
12  186  423      42
13  125  334      26
14  218  533      16
```

```
15   146   344        2
```

[6]:
```
def model(x,m,b):
    return m*x+b
init_guess = [1,1]
fit = scipy.optimize.curve_fit(model,dataset['x'],dataset['y'],sigma =␣
 ↪dataset['sigmay'],p0=init_guess,absolute_sigma=True)
ans,cov = fit
fit_m,fit_b = ans
print(fit_m)
print(fit_b)

x = np.linspace(50, 250, 1000)
plt.plot(x,model(x, fit_m, fit_b))
plt.errorbar(dataset['x'],dataset['y'], dataset['sigmay'], fmt="o", color="r",␣
 ↪capsize=5)
plt.xlim(50,250)
plt.show()
```

```
2.1615664402882535
45.88201303034733
```



### 0.3 Question 3

[14]:
```
N=50
degree_of_freedom = N-1
chi_sq_dof = [0.96,0.24,3.84,2.85]
p_values = []
for i in range(0,len(chi_sq_dof)):
    p = chi2.sf(chi_sq_dof[i]*degree_of_freedom,degree_of_freedom)
    p_values.append(p)
    print("P{} = {}".format(i,p))
```

```
P0 = 0.5529264339960217
P1 = 0.9999999917009567
P2 = 3.477504685373815e-18
P3 = 1.2107295923765585e-10
```