es21btech11025-assign3

February 5, 2024

EP 4130/PH 6130 Assignment 3
Ranveer Sahu <ES21BTECH11025>

[53]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[54]:
```
from astroML.stats import median_sigmaG
from scipy.stats import chi2
from scipy.optimize import minimize, curve_fit
```

Q1 In class, we showed histograms of standard deviation and G of bootstrap samples drawn from a Gaussian distribution with mean equal to 0 and standard deviation equal to 1. Draw a similar histogram of median of 10,000 bootstrap samples drawn from the same Gaussian distribution. According to http://tinyurl.com/h6p43o8, the standard deviation of the sample median of a Gaussian distribution is equal to p 2n . Overlay a Gaussian distribution on top of the histogram with mean equal to the mean of the generated data sample and standard deviation equal to the standard deviation of the median (Hint: Look up astroML.stats.median sigmaG. Also note that you don't have to draw 10,000 histograms, but only one histogram consisting of 10,000 bootstrap resamples.)

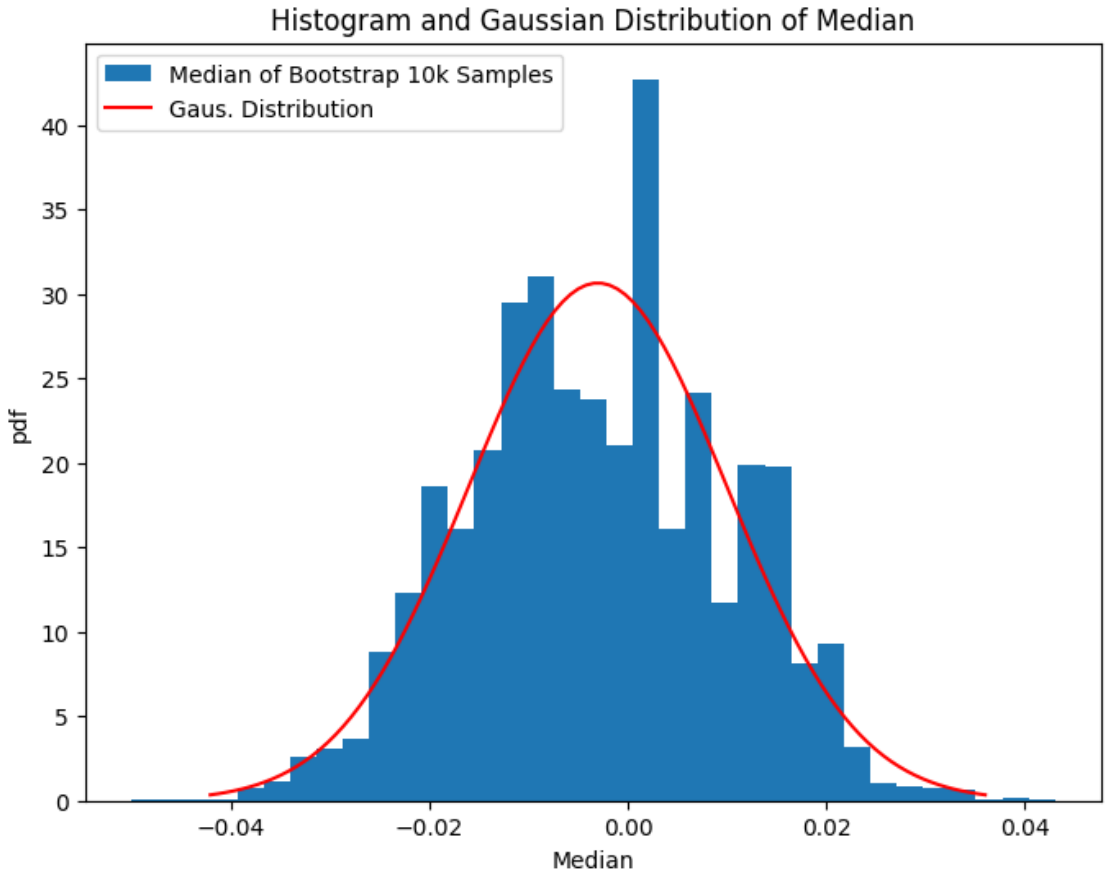
[55]:
```
!pip install astroML
```

```
Requirement already satisfied: astroML in /usr/local/lib/python3.10/dist-
packages (1.0.2.post1)
Requirement already satisfied: scikit-learn>=0.18 in
/usr/local/lib/python3.10/dist-packages (from astroML) (1.2.2)
Requirement already satisfied: numpy>=1.13 in /usr/local/lib/python3.10/dist-
packages (from astroML) (1.23.5)
Requirement already satisfied: scipy>=0.18 in /usr/local/lib/python3.10/dist-
packages (from astroML) (1.11.4)
Requirement already satisfied: matplotlib>=3.0 in
/usr/local/lib/python3.10/dist-packages (from astroML) (3.7.1)
Requirement already satisfied: astropy>=3.0 in /usr/local/lib/python3.10/dist-
packages (from astroML) (5.3.4)
Requirement already satisfied: pyerfa>=2.0 in /usr/local/lib/python3.10/dist-
packages (from astropy>=3.0->astroML) (2.0.1.1)
Requirement already satisfied: PyYAML>=3.13 in /usr/local/lib/python3.10/dist-
packages (from astropy>=3.0->astroML) (6.0.1)
Requirement already satisfied: packaging>=19.0 in
/usr/local/lib/python3.10/dist-packages (from astropy>=3.0->astroML) (23.2)
```

```
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->astroML) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib>=3.0->astroML) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->astroML) (4.47.2)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->astroML) (1.4.5)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib>=3.0->astroML) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->astroML) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->astroML) (2.8.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn>=0.18->astroML) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.18->astroML)
(3.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib>=3.0->astroML) (1.16.0)
```

[56]:
```
np.random.seed(42)

mn = 0
sd = 1
total_sample = 10000

data_sample = np.random.normal(mn, sd, total_sample)   #---- Generate a Gaus.␣
 ↪dist.
total_boot_samples = 10000          #----- resampling data
boot_samples = np.random.choice(data_sample, (total_sample,␣
 ↪total_boot_samples), replace=True)

# Calculate median for each above bootstrap samples
median_boot = np.median(boot_samples, axis=0)
fig, ax = plt.subplots(figsize=(8, 6)) #----- ploting guas. on bootstrap data
ax.hist(median_boot, bins=35, density=True, alpha=1, label='Median of Bootstrap␣
 ↪10k Samples')

# Overlay Gaus. distribution
mean_med = np.mean(median_boot)
std_dev_med = np.std(median_boot)
x = np.linspace(mean_med - 3 * std_dev_med, mean_med + 3 * std_dev_med, 100)
y = 1 / (std_dev_med * np.sqrt(2 * np.pi)) * np.exp(-(x - mean_med)**2 / (2 *␣
 ↪std_dev_med**2))
```

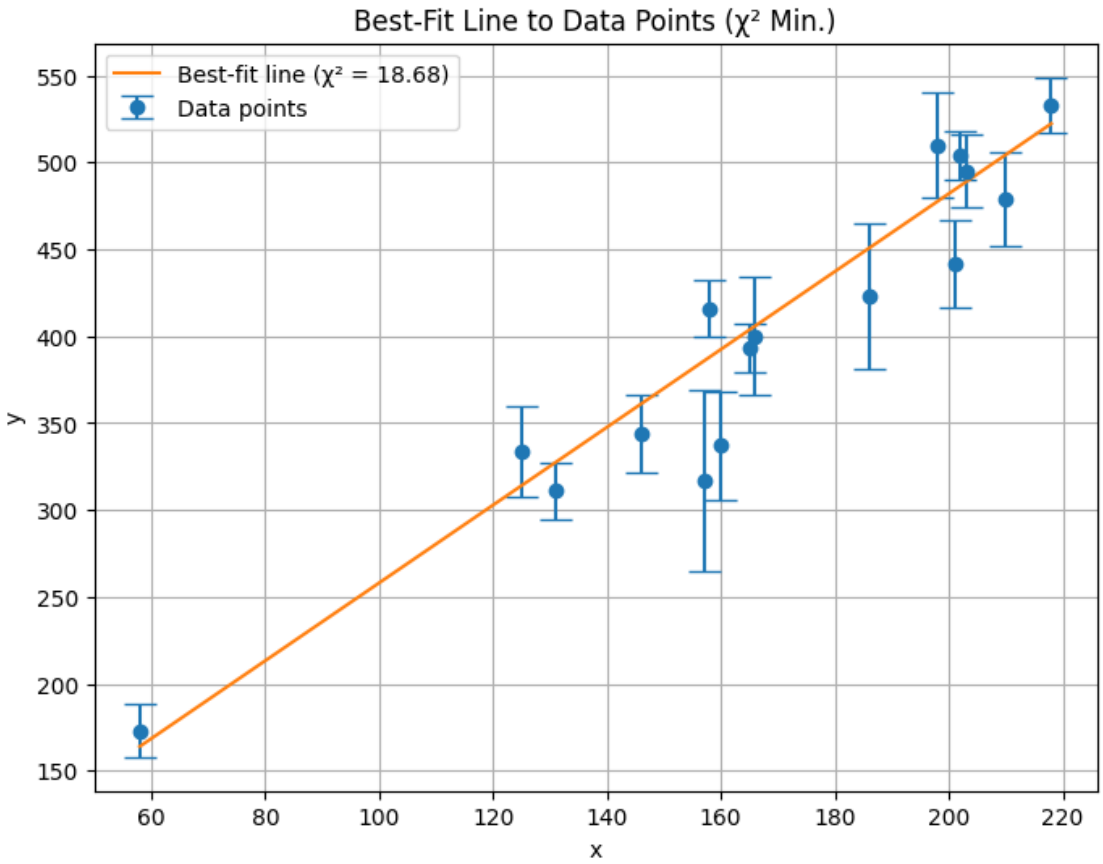
```
ax.plot(x, y, label='Gaus. Distribution', color='red')
ax.set_title('Histogram and Gaussian Distribution of Median')
ax.set_xlabel('Median')
ax.set_ylabel('pdf')
ax.legend()
plt.show()
```



Q2 arXiv:1008.4686, Exercise 1 on Page 5, except the last sentence of the question related to ^2*m. (Hint : Use 2 minimization to obtain best-fit values of b and m, instead of linear algebra. You can look up curve fit function in scipy.)

[62]:
```
x = np.array([203, 58, 210, 202, 198, 158, 165, 201, 157, 131, 166, 160, 186,␣
↪125, 218, 146])
y = np.array([495, 173, 479, 504, 510, 416, 393, 442, 317, 311, 400, 337, 423,␣
↪334, 533, 344])
sigma_y = np.array([21, 15, 27, 14, 30, 16, 14, 25, 52, 16, 34, 31, 42, 26, 16,␣
↪22])
```

```
# function to fit
def func(x, m, b):
    return m * x + b

popt, _ = curve_fit(func, x, y, sigma=sigma_y, absolute_sigma=True) # --- 2␣
 ↪minimization to find the best-fit param

m, b = popt           # ------ slope and intercept of best fit line respectively

chi2 = np.sum(((y - func(x, m, b)) / sigma_y)**2) #---- Calculate the  2 value

# best-fit line
x_fit = np.linspace(min(x), max(x), 100)
y_fit = func(x_fit, m, b)

plt.figure(figsize=(8, 6))
plt.errorbar(x, y, yerr=sigma_y, fmt='o', capsize=7, label='Data points') #----␣
 ↪error bar
plt.plot(x_fit, y_fit, label='Best-fit line ( ² = {:.2f})'.format(chi2)) #----␣
 ↪best fit line plot
plt.xlabel('x')
plt.ylabel('y')
plt.title('Best-Fit Line to Data Points ( ² Min.)')
plt.legend()
plt.grid(True)
plt.show()

print("slope(m) =", m)
print("intercept(b) = ", b)
print(" 2 value =", chi2)
```




```
slope(m) = 2.2399208553933314
intercept(b) =  34.047723577096654
 2 value = 18.680769911240873
```

Q3 Calculate the p-value for the four chi-square values for the plot shown in class from astroMl book which can be found at https://www.astroml. org/book_figures_1ed/chapter4/fig_chi2_eval.html. (Hint : You can read off the 2 values from the graph by multiplying by D.O.F.)

[72]:
```
def calculate_chi2_and_pvalue(y_vals, y_errs, N):
    objective = lambda params: np.sum(((y_vals - params[0]) / y_errs) ** 2)

    # Perform minimization to find best-fit mean
    result = minimize(objective, [np.mean(y_vals)], method='Powell')
    mu_fit = result.x[0]
    chi2_val = np.sum(((y_vals - mu_fit) / y_errs) ** 2) #----- chi-square␣
↪value calculation

    dof = N - 1 #------ degree of freedom
```

```
    p_value = 1 - chi2.pdf(chi2_val, dof) #--- Calculate the p-value using chi2.
 ↪pdf

    return chi2_val, p_value

np.random.seed(1)
N = 50
L0 = 10
dL = 0.2
t = np.linspace(0, 1, N)
L_obs = np.random.normal(L0, dL, N)

y_vals_list = [L_obs, L_obs, L_obs, L_obs + 0.5 - t ** 2]
y_errs_list = [dL, dL * 2, dL / 2, dL]
titles = ['correct errors', 'overestimated errors', 'underestimated errors',␣
 ↪'incorrect model']

for i in range(4):
    chi2_val, p_val = calculate_chi2_and_pvalue(y_vals_list[i], y_errs_list[i],␣
 ↪N)
    print(f'for Chi-sqr value = {chi2_val} p-value is = {p_val}\n')
```

```
for Chi-sqr value = 47.005168881363566 p-value is = 0.9589978982667309

for Chi-sqr value = 11.751292220340892 p-value is = 0.9999999868271917

for Chi-sqr value = 188.02067552545427 p-value is = 1.0

for Chi-sqr value = 139.5360944877475 p-value is = 0.9999999999578612
```