**#Q1**

import numpy as np

from scipy.stats import norm

from astroML.resample import bootstrap

from astroML.stats import median\_sigmaG

from matplotlib import pyplot as plt

points = 1000

bootstraps = 10000

np.random.seed(0)

gdsamples = norm(0, 1).rvs(points)

median, sigmaG = bootstrap(gdsamples, bootstraps, median\_sigmaG, kwargs=dict(axis=1))

x = np.linspace(-2, 2, 1000)

sd = np.sqrt(np.pi/(2\*points))

mean = np.mean(median)

pdf = norm(mean, sd).pdf(x)

fig, ax = plt.subplots(figsize=(5, 3.75))

ax.hist(median, bins = 20, density = True, histtype = 'step', color = 'orange',

label = r'$\sigma\ {\rm (median)}$')

ax.plot(x, pdf, color = 'blue', label = '$Gaussian\ Distribution$')

ax.set\_xlim(-0.5, 0.5)

ax.set\_xlabel(r'$\sigma$')

ax.set\_ylabel(r'$p(\sigma|x,I)$')

ax.legend()

plt.show()

Chart, line chart, histogram

Description automatically generated

#Q2

import numpy as np

import pandas as pd

from scipy.optimize import curve\_fit

from matplotlib import pyplot as plt

q2data = pd.read\_excel("C:\\Users\\Heera Baiju\\Downloads\\Q2Data.xlsx")

xq2data = q2data ['x']

yq2data = q2data ['y']

sigmaq2data = q2data ['sigmaY']

def function(x, m, c):

    return m \* x + c

arr = np.array([0, 0])

param, param\_cov = curve\_fit(function, xq2data, yq2data, arr, sigmaq2data)

print("Line function coefficients: {}" .format(param) )

print("Covariance of coefficients: {}" .format(param\_cov) )

x1 = np.linspace(0, 300, 1000)

c1 = param[1]

m1 = param[0]

print("Y-intercept = {}" .format(c1) )

print("Slope = {}" .format(m1) )

plt.errorbar(xq2data, yq2data, sigmaq2data , fmt='.k', ecolor='red', label='y vs x')

plt.plot(x1, m1\*x1+c1, '--', color ='green', label ="Optimized data")

plt.xlim(0, 300, 50)

plt.ylim(0, 700, 100)

plt.xlabel('x')

plt.ylabel('y')

plt.legend()

plt.show()

Chart, line chart

Description automatically generated

Line function coefficients: [ 2.2399208 34.04773403]

Covariance of coefficients: [[ 1.55005444e-02 -2.52129683e+00]

[-2.52129683e+00 4.44232207e+02]]

Y-intercept = 34.04773403259783

Slope = 2.2399207961186938

**#Q3**

from scipy import stats

import numpy as np

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 = [L\_obs, L\_obs, L\_obs, L\_obs + 0.5 - t \*\* 2]

y\_errs = [dL, dL \* 2, dL / 2, dL]

titles = ['correct errors', 'overestimated errors', 'underestimated errors',

'incorrect model']

for i in range(4):

    mu = np.mean(y\_vals[i])

    z = (y\_vals[i] - mu) / y\_errs[i]

    chi2 = np.sum(z \*\* 2)

    chi2dof = chi2 / (N - 1)

    pvalue = stats.chi2(N-1).sf(chi2dof\*chi2)

    print(" The p-value for the chi-square values of {} is: {}" .format(titles[i], pvalue))

**OUTPUT**

The p-value for the chi-square values of correct errors is: 0.6323042459089494

The p-value for the chi-square values of overestimated errors is: 1.0

The p-value for the chi-square values of underestimated errors is: 2.2834522158057202e-120

The p-value for the chi-square values of incorrect model is: 4.7450723075838625e-56