Data analysis in Astronomy

Homework 2 due 11/5 11:59 pm

Name:

1. (Monte Carlo Exercise) The variance of the sample variance

A sample of N data points drawn from a normal distribution has a variance. This variance also has a variance.

Based on theoretical calculation, the variance of variance with N data points is —— where is the standard deviation of the normal distribution.

To do:

1. write a code and use Monte Carlo method to validate the theoretical expection. (15 points)

2. Produce a plot with x-axis (N data point) and y-axis (variance of variance) with two curves showing a. your Monte Carlo simulation and b. theoretical calculation. (10 points)

(You can just use a normal distribution with mean=0 and sigma=10)

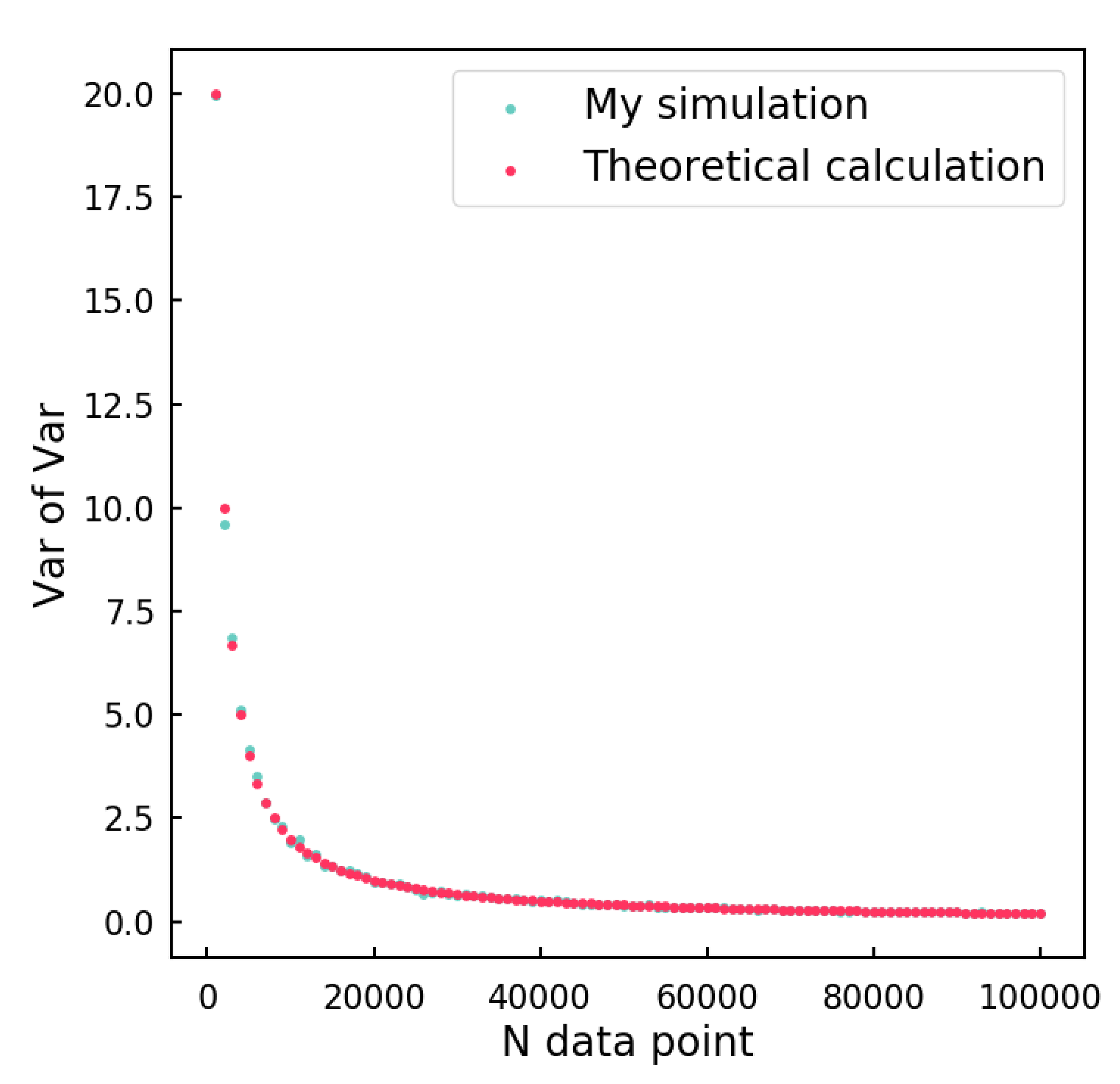
In [ ]:

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

def my_plot_style():

    params = {'legend.fontsize': 20,
             'axes.labelsize': 20,
             'axes.titlesize':20,
             'xtick.labelsize':16,
             'ytick.labelsize':16,
             'xtick.major.size':5,
              'xtick.minor.size':2.5,
             'ytick.major.size':5,
              'ytick.minor.size':2.5,
             'figure.facecolor':'w',
             #'lines.linewidth' : 1.5,
              'xtick.major.width':1.5,
              'ytick.major.width':1.5,
              'xtick.minor.width':1.5,
              'ytick.minor.width':1.5,
              'xtick.major.pad': 12,
              'ytick.major.pad': 8,
              'axes.linewidth':1.5,
              'xtick.direction':'in',
              'ytick.direction':'in',
             'ytick.labelleft':True,
              'text.usetex' : False,
             'font.family': 'sans-serif'}

    plt.rcParams.update(params)

mean = 0
sigma = 10

nsim = 1000
simvar = np.zeros(nsim)

npoints = 100
xaxis = np.zeros(npoints)
var2 = np.zeros(npoints)
theo_var2 = np.zeros(npoints)

for ipoint in range(0, npoints):
    ndata = (ipoint+1)*1000
    xaxis[ipoint] = ndata

    for isim in range(0, nsim):
        randomdata = np.random.normal(mean, sigma, ndata)
        simvar[isim] = np.var(randomdata)

    var2[ipoint] = np.var(simvar)
    theo_var2[ipoint] = 2*pow(sigma, 4)/ndata

my_plot_style()
plt.figure(figsize = (8, 8))
plt.scatter(xaxis, var2, s=15, c="#67ccc1", label="My simulation")
plt.scatter(xaxis, theo_var2, s=15, c="#ff3360", label="Theoretical calculation")
plt.legend()
plt.xlabel("N data point")
plt.ylabel("Var of Var")
plt.show()
```

In [3]:

```python
from IPython import display
display.Image("/home/judy/Astrophys/Week5/HW2/Monte_Carlo_1.png")
```

Out[3]:



2. When the central limit theorem breaks down

The central limit theorem states that given a sample with N data points drawn from some distributions with mean and standard deviation , the precision of the mean of the sample will scale with sigma/sqrt(N).

In other words, if you have a larger sample, you can obtain a more precise estimation of the mean value.

However, this is not always true.

To do:

a. Find a distribution that will break the central limit theorem and write a code using Monte Carlo method to demonstrate that. (10 points)

b. Make a plot showing your result with that distribution and the expected trend based on the central limit theorem. (10 points)

Cauchy distribution will break the central limit theorem since it's variance cannot be defined.

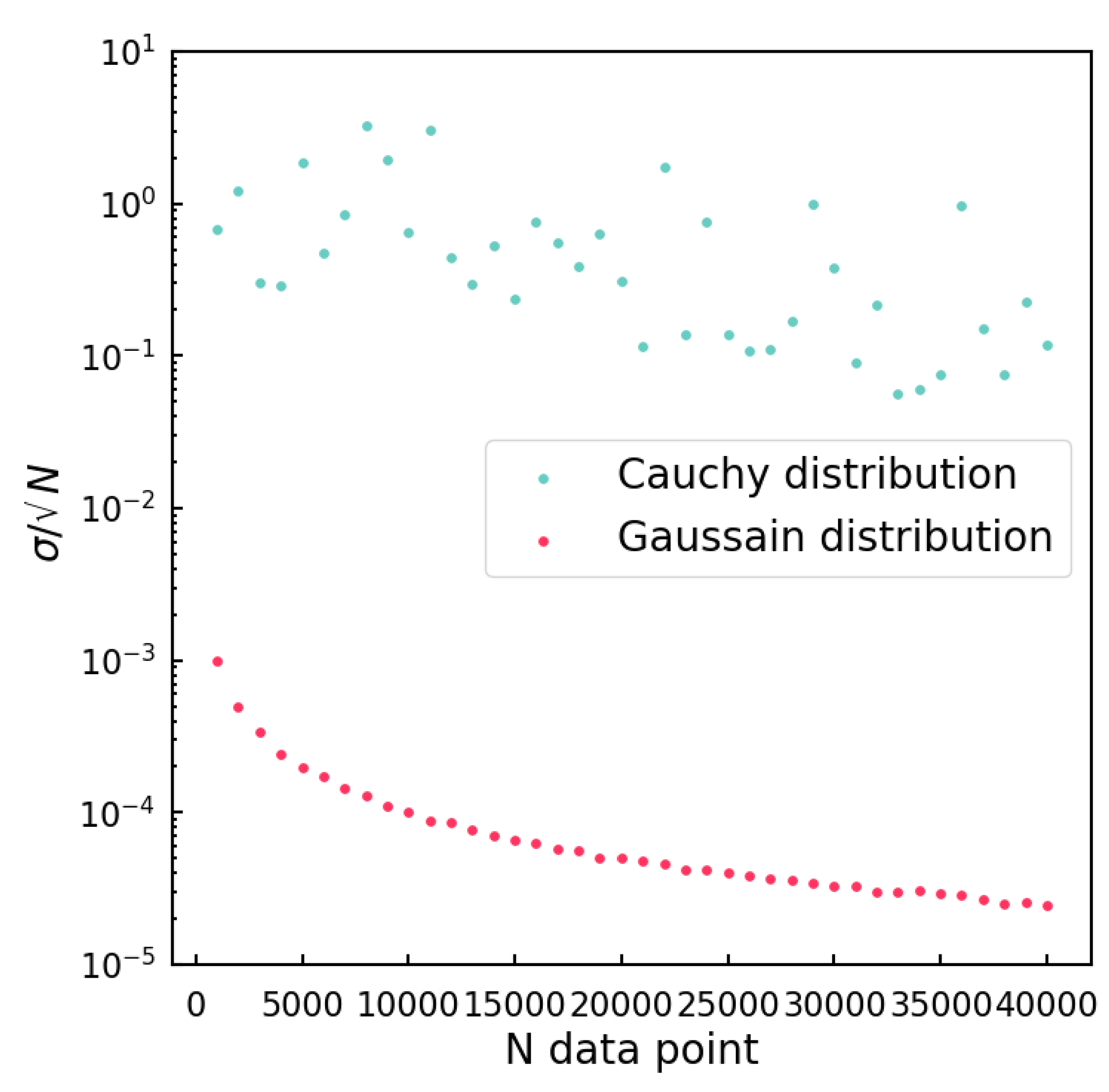
In [ ]:

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

def my_plot_style():

    params = {'legend.fontsize': 20,
             'axes.labelsize': 20,
             'axes.titlesize':20,
             'xtick.labelsize':16,
             'ytick.labelsize':16,
             'xtick.major.size':5,
              'xtick.minor.size':2.5,
             'ytick.major.size':5,
              'ytick.minor.size':2.5,
             'figure.facecolor':'w',
             #'lines.linewidth' : 1.5,
              'xtick.major.width':1.5,
              'ytick.major.width':1.5,
              'xtick.minor.width':1.5,
              'ytick.minor.width':1.5,
              'xtick.major.pad': 12,
              'ytick.major.pad': 8,
              'axes.linewidth':1.5,
              'xtick.direction':'in',
              'ytick.direction':'in',
             'ytick.labelleft':True,
              'text.usetex' : False,
             'font.family': 'sans-serif'}

    plt.rcParams.update(params)

nsim = 1000
simmean = np.zeros(nsim)
simgauss = np.zeros(nsim)

npoints = 40
xaxis = np.zeros(npoints)
std = np.zeros(npoints)
gstd = np.zeros(npoints)
mean_precision = np.zeros(npoints)
gmean_precision = np.zeros(npoints)

for ipoint in range(0, npoints):
    ndata = (ipoint+1)*1000
    xaxis[ipoint] = ndata

    for isim in range(0, nsim):
        randomdata = np.random.standard_cauchy(ndata)
        simmean[isim] = np.mean(randomdata)

        randomgauss = np.random.normal(0, 1, ndata)
        simgauss[isim] = np.mean(randomgauss)

    std[ipoint] = np.std(simmean)
    mean_precision[ipoint] = std[ipoint]/math.sqrt(ndata)
    gstd[ipoint] = np.std(simgauss)
    gmean_precision[ipoint] = gstd[ipoint]/math.sqrt(ndata)

my_plot_style()
plt.figure(figsize = (8, 8))
plt.scatter(xaxis, mean_precision, s=15, c="#67ccc1", label="Cauchy distribution")
plt.scatter(xaxis, gmean_precision, s=15, c="#ff3360", label="Gaussain distribution")
plt.legend()
plt.yscale("log")
ax = plt.gca()
ax.set_ylim([0.00001, 10])
plt.xlabel("N data point")
plt.ylabel(r'$\sigma$/$\sqrt{N}$')
plt.show()
```

In [4]:

```python
from IPython import display
display.Image("/home/judy/Astrophys/Week5/HW2/Central_limit_2.png")
```

Out[4]:



----

3. Finishing your Pearson and Spearman correlation coefficients calculation

In class, you have learned how to calculate Pearson and Spearman correlation coefficientts.

https://www.dropbox.com/s/h7545q0vzcqhi38/sky_maps_new_64_v6.fits?dl=0

To do:

a. Finishing your Pearson and Spearmn correlation coefficients (10 points)

b. Write a code to estimate the uncertainty of the Pearson and Spearman correlation coefficients for (EBV, HI) and (EBV, H2) (15 points)

In [10]:

```python
import numpy as np
import matplotlib.pyplot as plt
import astropy.io.fits as pf
import scipy.stats as ss

data = pf.open('sky_maps_new_64_v6.fits')
ISM = data[1].data
EBV = ISM['SFD']
HI = ISM['HI']/1e21

conversion_factor = 2*1e20/1e21
H2 = ISM['CO10']*conversion_factor

def pearson_correlation(par1, par2):
    std1 = np.std(par1)
    std2 = np.std(par2)
    mean1 = np.mean(par1)
    mean2 = np.mean(par2)

    ndata = len(par1)
    cov = (1/ndata)*(np.sum((par1-mean1)*(par2-mean2)))
    rho = cov/(std1*std2)
    return rho

def spearman_correlation(par1, par2):
    rank_par1 = len(par1) - rankdata(par1).astype(int) +1
    rank_par2 = len(par2) - rankdata(par2).astype(int) +1

    std1 = np.std(rank_par1)
    std2 = np.std(rank_par2)
    mean1 = np.mean(rank_par1)
    mean2 = np.mean(rank_par2)

    ndata = len(rank_par1)
    cov = (1/ndata)*(np.sum((rank_par1-mean1)*(rank_par2-mean2)))
    rho = cov/(std1*std2)
    return rho

bootstrap_time = 500
Pbootstrap_EBV_HI = np.zeros(bootstrap_time)
Sbootstrap_EBV_HI = np.zeros(bootstrap_time)
Pbootstrap_EBV_H2 = np.zeros(bootstrap_time)
Sbootstrap_EBV_H2 = np.zeros(bootstrap_time)

for iboot in range(0, bootstrap_time):
    random_EBV = np.random.randint(0, len(EBV), len(EBV))
    random_HI = np.random.randint(0, len(HI), len(HI))
    random_H2 = np.random.randint(0, len(H2), len(H2))

    new_random_EBV = EBV[random_EBV]
    new_random_HI = HI[random_HI]
    new_random_H2 = H2[random_H2]

    Pbootstrap_EBV_HI[iboot] = ss.pearsonr(new_random_EBV, new_random_HI)[0]
    Sbootstrap_EBV_HI[iboot] = ss.spearmanr(new_random_EBV, new_random_HI)[0]
    Pbootstrap_EBV_H2[iboot] = ss.pearsonr(new_random_EBV, new_random_H2)[0]
    Sbootstrap_EBV_H2[iboot] = ss.spearmanr(new_random_EBV, new_random_H2)[0]

Pbperr_EBV_HI = np.std(Pbootstrap_EBV_HI)
Sbperr_EBV_HI = np.std(Sbootstrap_EBV_HI)
Pbperr_EBV_H2 = np.std(Pbootstrap_EBV_H2)
Sbperr_EBV_H2 = np.std(Sbootstrap_EBV_H2)

print(f'Pearson(EBV, HI) uncertainty : {Pbperr_EBV_HI:.4f}')
print(f'Spearman(EBV, HI) uncertainty : {Sbperr_EBV_HI:.4f}')
print(f'Pearson(EBV, H2) uncertainty : {Pbperr_EBV_H2:.4f}')
print(f'Spearman(EBV, H2) uncertainty : {Sbperr_EBV_H2:.4f}')
```

```
Pearson(EBV, HI) uncertainty : 0.0043
Spearman(EBV, HI) uncertainty : 0.0044
Pearson(EBV, H2) uncertainty : 0.0041
Spearman(EBV, H2) uncertainty : 0.0046
```