AST1100 - Week 1

Gunnar Lange

August 24, 2016

Exercise 1

A.1

Using a simple python script:

```
import math
import random
import numpy as np

def Gauss(x,sigma, mu):
    prefac=1/(sigma*math.sqrt(2*math.pi))
    exponent=-((x-mu)**2)/(2.0*sigma**2)
    return prefac*math.exp(exponent)


def next_step(x, delta_x, sigma, mu):
    return (Gauss(x,sigma,mu)+Gauss(x+delta_x,sigma,mu))/2.0

sigma=2
mu=29
start=19
end=39
npoints=10000
x=np.linspace(start, end, npoints)
delta_x=(end-start)/float(npoints)
integrated=0
for k in x:
    integrated+=next_step(k, delta_x, sigma, mu)*delta_x
print integrated
```

1) 0.3085

2) Symmetrical around σ so 0.68

3) $(0.3085)^7 = 2.659 \times 10^{-4}$

4) Binomial experiment so:

$$P(4, 7, 0, .5) = \binom{7}{4}(0.3085)^4(1 - 0.3085)^3 = 0.105$$

A.2

1)

Again, a simple Python Script solves 1a)-1e):

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

infile=np.loadtxt('Rainfall.dat')
january=infile[:,1]
nbins=20
largest=january.max()
smallest=january.min()
bins=np.linspace(smallest, largest, nbins+1)
values=[]
for k in range(len(bins)-1):
    relevant_values=np.logical_and(january>=bins[k], january<=bins[k+1])
    values.append(np.count_nonzero(relevant_values))

values=np.array(values)/(float(len(january)))
midpoints=[(a+b)/2.0 for a,b in zip(bins, bins[1::])]
plt.plot(midpoints, values)
plt.title('Histogram for January')
plt.xlabel('Amount of rainfall')
plt.ylabel('Probability')
plt.show()
```

This creates this figure:

f) The rainfall distribution is fairly clearly non-Gaussian.

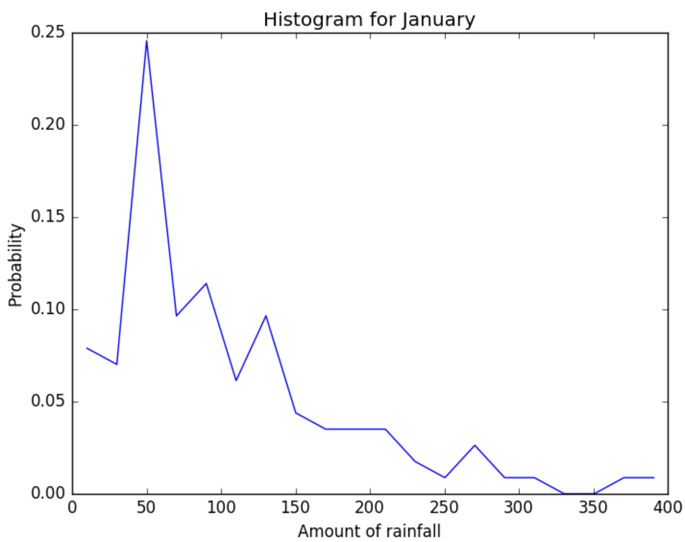

Figure 1: Histogram for January

g) Trying February gives the following figure which again is clearly non-Gaussian.

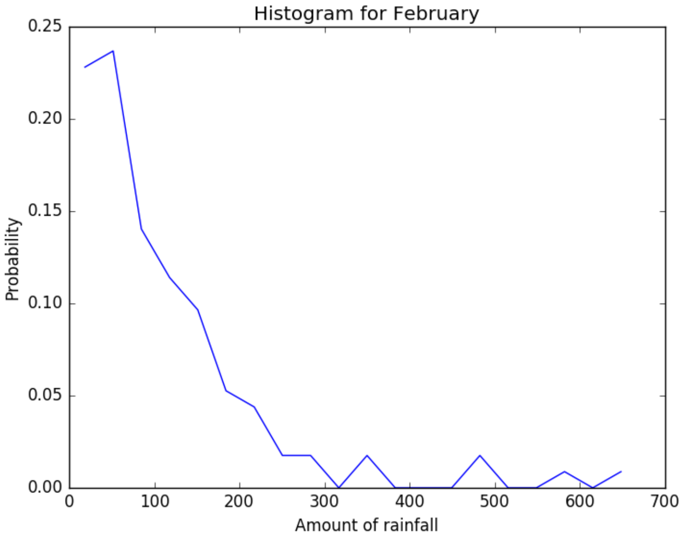

Figure 2: Histogram for February

2)

Once again a straightforward Python script:

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

infile=np.loadtxt('Rainfall.dat')
##Method 1
monthly_mean_1=[]
for k in range(infile.shape[1]-2):
     monthly_mean_1.append(np.mean(infile[:,k+1]))
print monthly_mean_1
##Method 2
monthly_mean_2=[]
nbins=20
for k in range(infile.shape[1]-2):
    largest=infile[:, k+1].max()
    smallest=infile[:, k+1].min()
    bins=np.linspace(smallest, largest, nbins+1)
    values=[]
    for l in range(len(bins)-1):
        relevant_values=np.logical_and(infile[:, k+1]>=bins[l], infile[:, k+1]<=bins[l+1])
        values.append(np.count_nonzero(relevant_values))
    probabilities=np.array(values)/(float(len(infile[:, k+1])))
    midpoints=[(a+b)/2.0 for a,b in zip(bins, bins[1::])]
    mean=np.sum(probabilities*midpoints)
    monthly_mean_2.append(mean)
print monthly_mean_2
```

I.e. the probability is the probability of getting a specified bin.

3)

Again a straightforward Python script:

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

infile=np.loadtxt('Rainfall.dat')
##Method 1
monthly_mean_1=[]
for k in range(infile.shape[1]-2):
     monthly_mean_1.append(np.mean(infile[:,k+1]))
monthly_std_1=[]
for k in range(infile.shape[1]-2):
    month_std=[]
    for l in range(infile.shape[0]):
        month_std.append((infile[l][k+1]-monthly_mean_1[k])**2)
    monthly_std_1.append(np.sqrt(np.mean(month_std)))
print monthly_std_1

##Method 2
monthly_mean_2=[]
monthly_std_2=[]
nbins=20
for k in range(infile.shape[1]-2):
    largest=infile[:, k+1].max()
    smallest=infile[:, k+1].min()
    bins=np.linspace(smallest, largest, nbins+1)
    values=[]
    for l in range(len(bins)-1):
        relevant_values=np.logical_and(infile[:, k+1]>=bins[l], infile[:, k+1]<=bins[l+1])
        values.append(np.count_nonzero(relevant_values))
    probabilities=np.array(values)/(float(len(infile[:, k+1])))
    midpoints=[(a+b)/2.0 for a,b in zip(bins, bins[1::])]
    mean=np.sum(probabilities*midpoints)
    monthly_mean_2.append(mean)
    monthly_std_2.append(np.sqrt(np.sum(((midpoints-mean)**2*probabilities))))
print monthly_std_2
```

4)

Again a simply Python script gives:

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

def Gauss(x,sigma, mu):
    prefac=1/(sigma*np.sqrt(2*np.pi))
    exponent=-((x-mu)**2)/(2.0*sigma**2)
    return prefac*np.exp(exponent)

infile=np.loadtxt('Rainfall.dat')
yearly_means=[]
for l in range(infile.shape[0]):
    current_means=[]
    for k in range(infile.shape[1]-2):
        current_means.append(infile[l][k+1])
    yearly_means.append(np.mean(current_means))

yearly_means=np.array(yearly_means)

mean_of_means=np.mean(yearly_means)
```

```
std_of_means=np.std(yearly_means)


nbins=20
largest=yearly_means.max()
smallest=yearly_means.min()
x=np.linspace(smallest, largest, 10000)
bins=np.linspace(smallest, largest, nbins+1)
values=[]
for k in range(len(bins)-1):
    relevant_values=np.logical_and(yearly_means>=bins[k], yearly_means<=bins[k+1])
    values.append(np.count_nonzero(relevant_values))

probabilities=np.array(values)/(float(len(yearly_means)))
midpoints=[(a+b)/2.0 for a,b in zip(bins, bins[1::])]

figure1=plt.figure()
plt.plot(x, Gauss(x, std_of_means, mean_of_means))
plt.plot(midpoints, probabilities)
plt.title('Histogram for yearly means')
plt.xlabel('Amount of rainfall')
plt.ylabel('Probability')
plt.show()
```

With the following figure, which certainly looks more Gaussian:

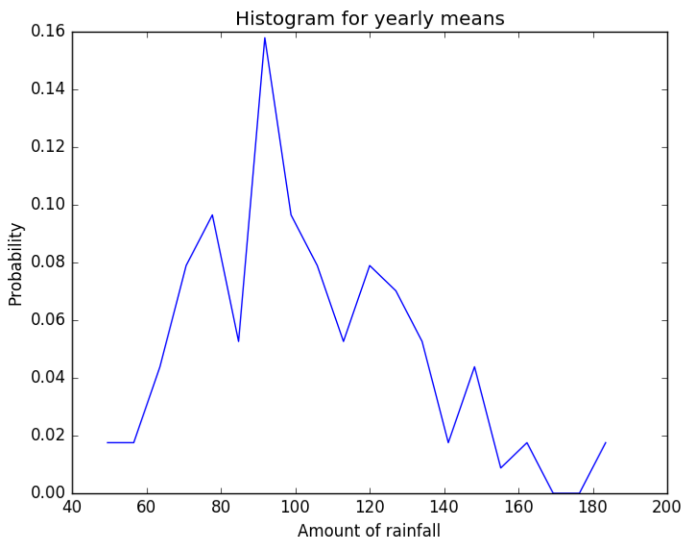

Figure 3: Histogram for February

5)

A Python script:

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

def Gauss(x,sigma, mu):
    prefac=1/(sigma*np.sqrt(2*np.pi))
```

```
    exponent=-((x-mu)**2)/(2.0*sigma**2)
    return prefac*np.exp(exponent)

infile=np.loadtxt('Rainfall.dat')
yearly_means=[]
for l in range(infile.shape[0]):
    current_means=[]
    for k in range(infile.shape[1]-2):
        current_means.append(infile[l][k+1])
    yearly_means.append(np.mean(current_means))

yearly_means=np.array(yearly_means)

mean_of_means=np.mean(yearly_means)
std_of_means=np.std(yearly_means)


nbins=20
largest=yearly_means.max()
smallest=yearly_means.min()
x=np.linspace(smallest, largest, 10000)
bins=np.linspace(smallest, largest, nbins+1)
values=[]
for k in range(len(bins)-1):
    relevant_values=np.logical_and(yearly_means>=bins[k], yearly_means<=bins[k+1])
    values.append(np.count_nonzero(relevant_values))

probabilities=np.array(values)/(float(len(yearly_means)))
midpoints=[(a+b)/2.0 for a,b in zip(bins, bins[1::])]

figure1=plt.figure()
plt.plot(x, Gauss(x, std_of_means, mean_of_means))
plt.plot(midpoints, probabilities)
plt.title('Histogram for yearly means')
plt.xlabel('Amount of rainfall')
plt.ylabel('Probability')
plt.show()
```

Producing the following plot: Something is wrong. I do not have time to figure that out now.

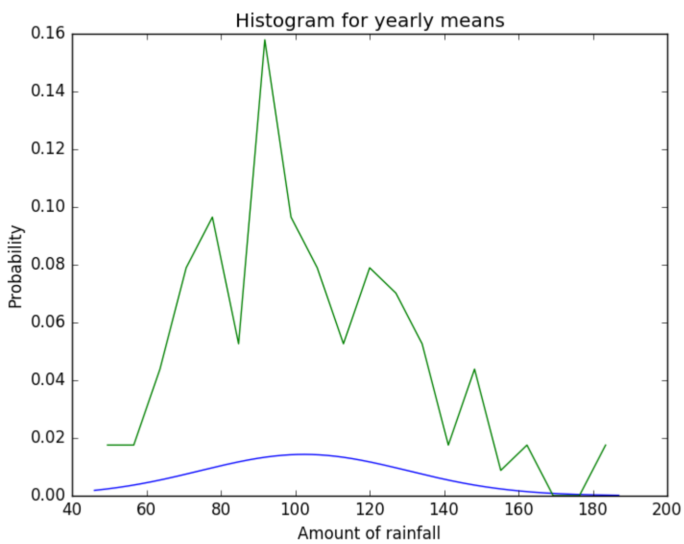

Figure 4: Histogram for yearly means

A.3

1)

This is a direct application of the probability calculator, giving:

$$\sigma = 0.682 \quad 2\sigma = 0.954, \quad 3\sigma = 0.997, \quad 4\sigma = 0.9998, \quad 5\sigma = 0.9998$$

Presumable numerical errors were too large in the last one.

2)

Definition of FWHM: width at $f(x) = \pm\max/2$. Recalling the definition of the Gaussian:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

The max is at $x = \mu$, where $f(x) = 1/\sigma\sqrt{2\pi}$, so I must investigate where the Gaussian is half that value:

$$\frac{1}{2\sigma\sqrt{2\pi}} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$\ln 2 = \frac{(x-\mu)^2}{2\sigma^2}$$

$$x = \mu \pm \sqrt{2\sigma^2 \ln 2}$$

To find the width, I must subtract the larger of these values from the smaller one:

$$x_2 - x_1 = FWHM = \mu + \sqrt{2\sigma^2 \ln 2} - (\mu - \sqrt{2\sigma^2 \ln 2}) = 2\sqrt{2\sigma^2 \ln 2} = \sigma\sqrt{8 \ln 2}$$

From which it follows that:

$$\sigma = \frac{FWHM}{\sqrt{8 \ln 2}}$$

As was to be shown.

A.4

1)

A simple Python script, implementing the Maxwell-Boltzmann distribution, gives the answer quickly:

```python
import numpy as np

def Maxwell_Boltzmann(m,T, v):
    k=1.380649e-23
    prefac=(m/(2*np.pi*k*T))**(3/2.0)
    exponent=-0.5*(m*v**2)/(k*T)
    return prefac*np.exp(exponent)*4*np.pi*v**2

```

```
def next_step(m, T, v, delta_v):
    return (Maxwell_Boltzmann(m,T,v)+Maxwell_Boltzmann(m,T,v+delta_v))/2.0


T=15e6
n=(150e3)*1000*6.02e23
start=100
end=1000
npoints=10000
v=np.linspace(start, end, npoints)
delta_v=(end-start)/float(npoints)
integrated=0
m=1.6737e-27
for k in v:
    integrated+=next_step(m,T,k, delta_v)*delta_v
print integrated*n
```

Which gives: $n(v) = 5.51 \times 10^{23}$.

2)

This is of course exactly the same calculation, only substituting the function. I will not do this now.

A.5

1)

Starting from equation 6 as suggested:

$$\langle v \rangle = \int_0^\infty v \left(\frac{m}{2\pi kT}\right)^{3/2} e^{-\frac{mv^2}{2kT}} 4\pi v^2 dv$$

I will now attempt to comptute this integral. A nautral way to begin is to make the substitution:

$$u = v^2, \quad \frac{du}{dv} = 2v \implies dv = \frac{du}{2v}$$

Giving:

$$\langle v \rangle = \left(\frac{m}{2\pi kT}\right)^{3/2} \int_0^\infty 2\pi u e^{-\frac{mu}{2kT}} du$$

Now let:

$$\beta = \frac{m}{2kT}, \quad v = \beta u, \quad du = \frac{dv}{\beta}$$

Transforming the integral into:

$$\langle v \rangle = \left(\frac{m}{2\pi kT}\right)^{3/2} 2\pi \int_0^\infty \frac{ve^{-v}}{\beta^2} dv = \frac{1}{\beta^2} \left(\frac{m}{2\pi kT}\right)^{3/2} = \sqrt{\frac{8kT}{m\pi}}$$

2)

Starting from equation 11 as suggested:

$$P = \frac{n}{3} \int_0^\infty \frac{p^2}{m} \left(\frac{1}{2\pi mkT}\right)^{3/2} e^{-\frac{1}{2}\frac{p^2}{mkT}} 4\pi p^2 dp$$

I want to arrive at:

$$P = nkT$$

I begin by taking all constants outside:

$$P = \frac{4\pi n}{3m}\left(\frac{1}{2\pi mkT}\right)^{3/2}\int_0^\infty p^4 e^{-\frac{1}{2}\frac{p^2}{mkT}}\,dp$$

Let $u = p^2$. Then:

$$\frac{du}{dp} = 2p, \quad dp = \frac{du}{2p}$$

Then the integral becomes:

$$\int_0^\infty \frac{u^{3/2}}{2} e^{-\frac{1}{2}\frac{u}{mkT}}\,du$$

Let:

$$\beta = \frac{1}{2mkT}, \quad v = \beta u, \quad du = \frac{dv}{\beta}$$

Then the integral transforms to:

$$\int_0^\infty \frac{v^{3/2}}{2\beta^{5/2}} e^{-v} dv = \frac{3\sqrt{\pi}}{8\beta^{5/2}}$$

Giving, finally:

$$P = \frac{4\pi n}{3m}\left(\frac{1}{2\pi mkT}\right)^{3/2}\frac{3\sqrt{\pi}(2mkT)^{5/2}}{8}$$

$$P = \frac{n}{2m}(2mkT) = nkT$$

As was to be shown.

3)

I need to use the RMS of the speed squared. To find the average of the square of the speed, I thus need to compute:

$$\langle E_k\rangle = \langle \frac{1}{2}mv^2\rangle = \int_0^\infty \frac{1}{2}mv^2 P(v)dv = \int_0^\infty v^4\left(\frac{m}{2\pi kT}\right)^{3/2} e^{-\frac{mv^2}{2kT}}2m\pi dv$$

$$\langle E_k\rangle = \frac{2\pi m^{5/2}}{(2\pi kT)^{3/2}}\int_0^\infty v^4 e^{-\frac{mv^2}{2kT}}\,dv$$

Using a standard strategy for the integral:

$$u = v^2, \quad \frac{du}{dv} = 2v, \quad dv = \frac{du}{2v}$$

$$\int_0^\infty \frac{u^{3/2}}{2} e^{-\frac{mu}{2kT}}\,du$$

Let:

$$\beta = \frac{m}{2kT}, \quad x = \beta u, \quad du = \frac{dx}{\beta}$$

$$\int_0^\infty \frac{u^{3/2}}{2} e^{-\frac{mu}{2kT}} = \int_0^\infty \frac{x^{3/2}}{2\beta^{5/2}} e^x dx = \frac{3\sqrt{\pi}}{8\beta^{5/2}}$$

Putting this all together then:

$$\langle E_k \rangle = \frac{2\pi m^{5/2}}{(2\pi kT)^{3/2}} \frac{3\sqrt{\pi}}{8\beta^{5/2}} = \frac{2\pi m^{5/2}}{(2\pi kT)^{3/2}} \frac{3\sqrt{\pi}(2kT)^{5/2}}{8m^{5/2}} = \frac{3}{2}kT$$

As was to be shown.

A.6

I do not have a home planet yet, so I will do this for the earth. Equating potential and kinetic energy gives:

$$\frac{1}{2} m v_{esc} = \frac{GmM}{R^2}$$

Giving:

$$v_{esc} = \sqrt{\frac{2GM}{R}}$$