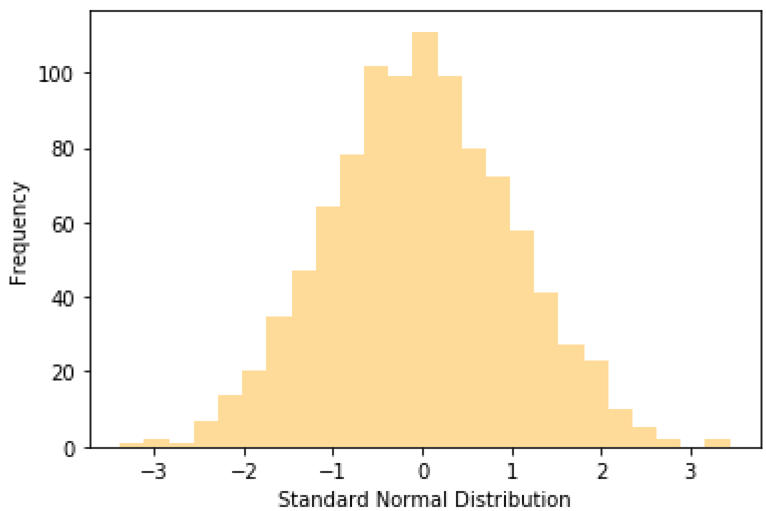
In [1]:
```
#Standard normal distribution arrays with size 10, 100, 1000, and 100000
0
from scipy.stats import norm

array1_standard_norm = norm.rvs(size=10)
array2_standard_norm = norm.rvs(size=100)
array3_standard_norm = norm.rvs(size=1000)
array4_standard_norm = norm.rvs(size=1000000)

#Calculates the mean of each arrays
print("The mean of a standard normal distribution with size 10 is " + st
r(array1_standard_norm.mean()))
print("The mean of a standard normal distribution with size 100 is " + s
tr(array2_standard_norm.mean()))
print("The mean of a standard normal distribution with size 1000 is " +
str(array3_standard_norm.mean()))
print("The mean of a standard normal distribution with size 1000000 is "
+ str(array4_standard_norm.mean()))
print("\n")

#Calculates the standard deviation of the arrays
print("The mean of a standard normal distribution with size 10 is " + st
r(array1_standard_norm.std()))
print("The mean of a standard normal distribution with size 100 is " + s
tr(array2_standard_norm.std()))
print("The mean of a standard normal distribution with size 1000 is " +
str(array3_standard_norm.std()))
print("The mean of a standard normal distribution with size 1000000 is "
+ str(array4_standard_norm.std()))
```

```
The mean of a standard normal distribution with size 10 is 0.4638811738
00191
The mean of a standard normal distribution with size 100 is -0.17496660
75907089
The mean of a standard normal distribution with size 1000 is 0.04512516
934582229
The mean of a standard normal distribution with size 1000000 is 0.00120
96017435044033


The mean of a standard normal distribution with size 10 is 1.4795291484
789874
The mean of a standard normal distribution with size 100 is 0.937721588
15736
The mean of a standard normal distribution with size 1000 is 0.97902110
39114353
The mean of a standard normal distribution with size 1000000 is 1.00078
90829384505
```

The expected value E(X) for a standard normal distribution is the mean, μ, which equals to 0. The standard deviation is 1. Based on the generated arrays, we can see that as bigger the size of the array, the more accurate the mean and standard deviation are. Therefore, the generated arrays correspond to the standard normal distribution.

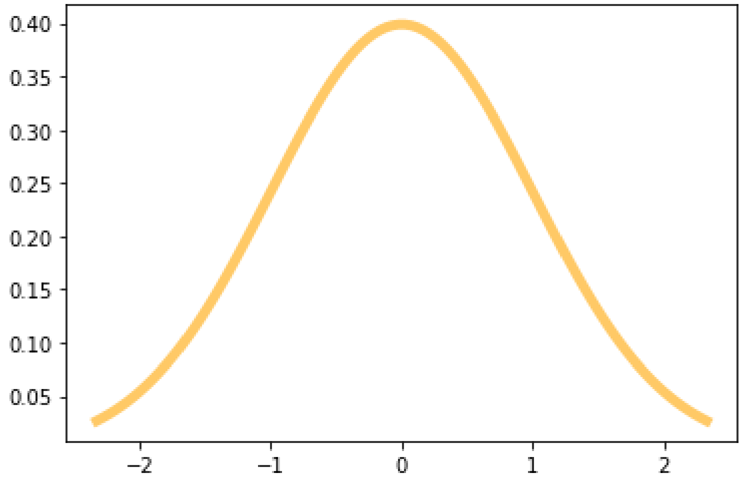
In [69]:
```
#Histogram for Standard Normal Distribution
import seaborn as sns

data_standard_norm1 = norm.rvs(size=1000)
ax= sns.distplot(data_standard_norm1,
                 kde=False,
                 color="orange",)
ax.set(xlabel='Standard Normal Distribution', ylabel='Frequency')
```

Out[69]:
```
[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Standard Normal Distributio
n')]
```



In [58]:
```
#PDF plot of Standard Normal Distribution
from scipy.stats import norm
import matplotlib.pyplot as plt
fig, ax = plt.subplots(1, 1)
x = np.linspace(norm.ppf(0.01),
                norm.ppf(0.99), 100)
ax.plot(x, norm.pdf(x),
          'orange', lw=5, alpha=0.6, label='norm pdf')
```

Out[58]:
```
[<matplotlib.lines.Line2D at 0x1a265c87b8>]
```



Based on the generated histogram and the PDF plot of a standard normal distribution, we can see that the histogram looks pretty similar to the bell curve shown in the PDF with the highest point being the mean=0. This means that the frequency of values closest to 0 are higher than the values away from 0. Therefore, the generated histogram corresponds to the PDF of the distribution.

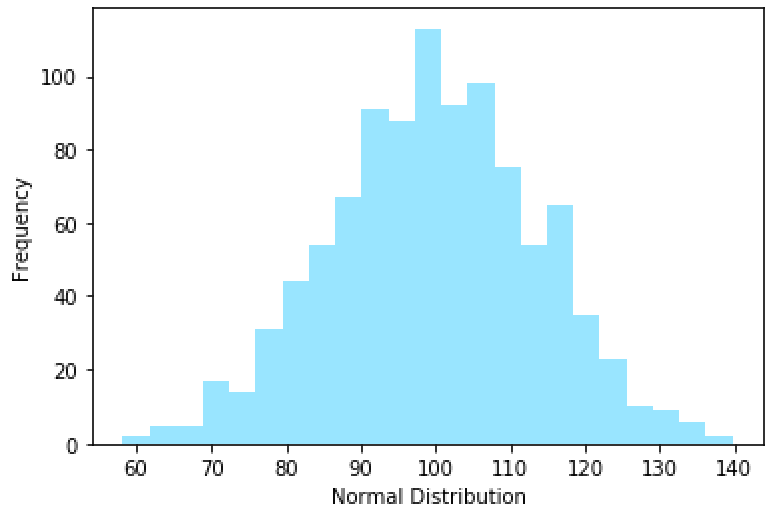
In [2]:
```
#Normal distribution arrays with size 10, 100, 1000, and 1000000
from scipy.stats import norm

array1_norm = norm.rvs(loc=100,scale=14, size=10)
array2_norm = norm.rvs(loc=100,scale=14, size=100)
array3_norm = norm.rvs(loc=100,scale=14, size=1000)
array4_norm = norm.rvs(loc=100,scale=14, size=1000000)

#Calculates the mean of each arrays
print("The mean of a normal distribution with given parameters 100 and 1
4 with size 10 is " + str(array1_norm.mean()))
print("The mean of a normal distribution with given parameters 100 and 1
4 with size 100 is " + str(array2_norm.mean()))
print("The mean of a normal distribution with given parameters 100 and 1
4with size 1000 is " + str(array3_norm.mean()))
print("The mean of a normal distribution with given parameters 100 and 1
4 with size 1000000 is " + str(array4_norm.mean()))
print("\n")

#Calculates the standard deviation of the arrays
print("The standard deviation of a normal distribution with given parame
ters 100 and 14 with size 10 is " + str(array1_norm.std()))
print("The standard deviation of a normal distribution with given parame
ters 100 and 14 with size 100 is " + str(array2_norm.std()))
print("The standard deviation of a normal distribution with given parame
ters 100 and 14 with size 1000 is " + str(array3_norm.std()))
print("The standard deviation of a normal distribution with given parame
ters 100 and 14 with size 1000000 is " + str(array4_norm.std()))
```

```
The mean of a normal distribution with given parameters 100 and 14 with
size 10 is 103.84252871830054
The mean of a normal distribution with given parameters 100 and 14 with
size 100 is 98.48897963036453
The mean of a normal distribution with given parameters 100 and 14with
size 1000 is 99.79483937298532
The mean of a normal distribution with given parameters 100 and 14 with
size 1000000 is 99.99918489383764


The standard deviation of a normal distribution with given parameters 1
00 and 14 with size 10 is 17.19799413532391
The standard deviation of a normal distribution with given parameters 1
00 and 14 with size 100 is 14.473579274410191
The standard deviation of a normal distribution with given parameters 1
00 and 14 with size 1000 is 13.964233541609088
The standard deviation of a normal distribution with given parameters 1
00 and 14 with size 1000000 is 13.996242803799179
```

The expected value of a normal distribution is the mean, μ, which is a given paramater of a value 100. The standard deviation, σ, is also a given parameter which equals to 14. From the generated arrays, we can see that the bigger the size, the closer the mean and the standard deviation are to the ideal number corresponding to the formulas given in class.

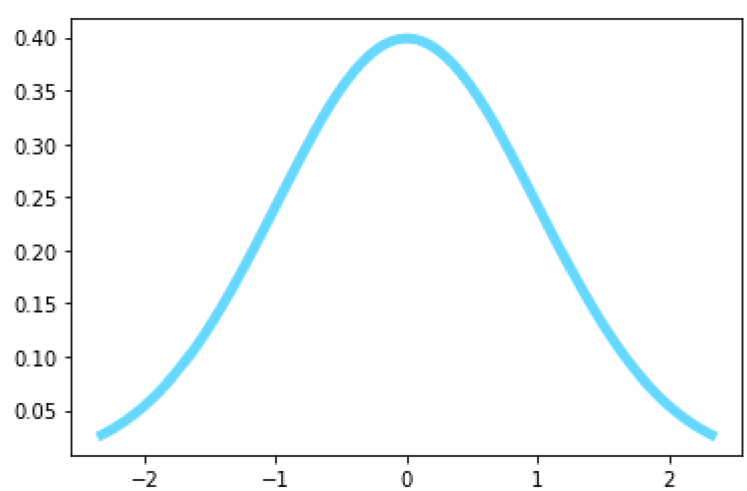
In [22]:
```
# Histogram for Normal Distribution
import seaborn as sns

data_norm1 = norm.rvs(size=1000,loc=100,scale=14)
ax= sns.distplot(data_norm1,
                  kde=False,
                  color="deepskyblue",)
ax.set(xlabel='Normal Distribution', ylabel='Frequency')
```

Out[22]:
```
[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Normal Distribution')]
```



In [54]:
```
#PDF plot of Normal Distribution
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt

fig, ax = plt.subplots(1, 1)
x = np.linspace(norm.ppf(0.01),
                norm.ppf(0.99), 100)
ax.plot(x, norm.pdf(x),
          'deepskyblue', lw=5, alpha=0.6, label='norm pdf')
```

Out[54]:
```
[<matplotlib.lines.Line2D at 0x1a25f8a630>]
```



Based on the generated histogram and the PDF plot of a normal distribution, we can see that the histogram looks pretty similar to the bell curve shown in the PDF with the highest point being the mean. This means that the frequency of values closest to 100 are higher than the values away from 100 since the mean is 100. Since the histogram was based on a randomly generated array, it may not always look the same as the pdf, rather it should look similar. Therefore, the generated histogram corresponds to the PDF of the distribution.

In [3]:

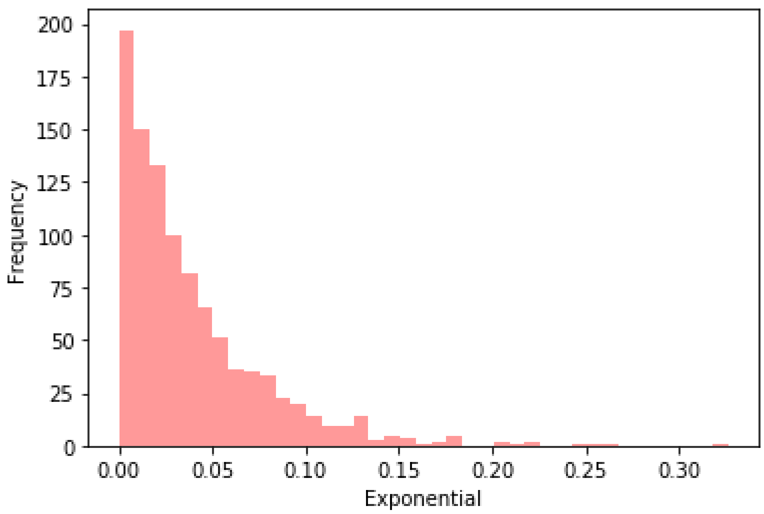
```
#Exponential distribution arrays with size 10, 100, 1000, and 1000000
from scipy.stats import expon

array1_expon = expon.rvs(scale=1/25, size=10)
array2_expon = expon.rvs(scale=1/25, size=100)
array3_expon = expon.rvs(scale=1/25, size=1000)
array4_expon = expon.rvs(scale=1/25, size=1000000)

print("The mean of an exponential distribution with a given lambba of 25 with size 10 is " + str(array1_expon.mean()))
print("The mean of an exponential distribution with a given lambba of 25 with size 100 is " + str(array2_expon.mean()))
print("The mean of an exponential distribution with a given lambba of 25 with size 1000 is " + str(array3_expon.mean()))
print("The mean of an exponential distribution with a given lambba of 25 with size 1000000 is " + str(array4_expon.mean()))
print("\n")

print("The standard deviation of an exponential distribution with a given lambba of 25 with size 10 is " + str(array1_expon.std()))
print("The standard deviation of an exponential distribution with a given lambba of 25 with size 100 is " + str(array2_expon.std()))
print("The standard deviation of an exponential distribution with a given lambba of 25 with size 1000 is " + str(array3_expon.std()))
print("The standard deviation of an exponential distribution with a given lambba of 25 with size 1000000 is " + str(array4_expon.std()))
```

```
The mean of an exponential distribution with a given lambba of 25 with size 10 is 0.017934727819409966
The mean of an exponential distribution with a given lambba of 25 with size 100 is 0.04569572612282163
The mean of an exponential distribution with a given lambba of 25 with size 1000 is 0.03866765476214938
The mean of an exponential distribution with a given lambba of 25 with size 1000000 is 0.040017323488955626



The standard deviation of an exponential distribution with a given lambba of 25 with size 10 is 0.013862932347685916
The standard deviation of an exponential distribution with a given lambba of 25 with size 100 is 0.05224305406767039
The standard deviation of an exponential distribution with a given lambba of 25 with size 1000 is 0.03908338883266795
The standard deviation of an exponential distribution with a given lambba of 25 with size 1000000 is 0.03998292214281732
```

The expected value E(X) of the exponential distribution is 1/lambda. The lambda is a given parameter of 25, which means that the expected value is 1/25 or 0.04. In an exponential distribution, the standard deviation has the same value of the mean, so the standard deviation is also 1/25 or 0.04. From the generated arrays, we can see that the bigger the size of the array, the closer the value to the ideal mean and standard deviation.

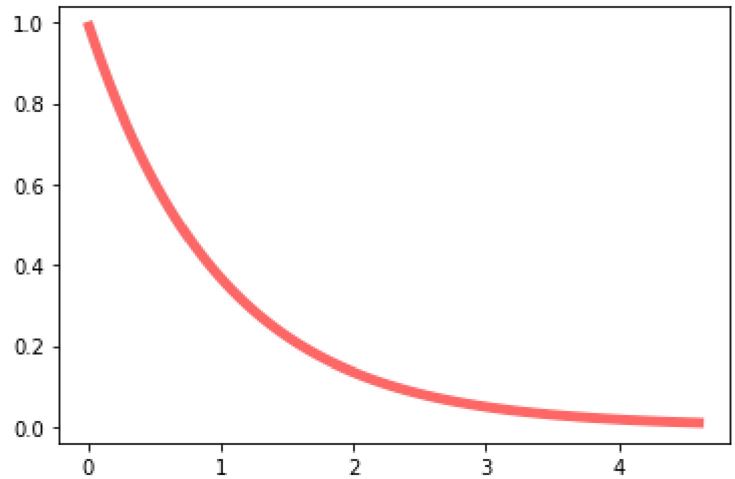
In [14]:
```
#Histogram for Exponential Distribution
import seaborn as sns

data_expon1 = expon.rvs(size=1000,scale=1/25)
ax= sns.distplot(data_expon1,
                 kde=False,
                 color="red",)
ax.set(xlabel='Exponential', ylabel='Frequency')
```

Out[14]:
```
[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Exponential')]
```



In [20]:
```
#PDF of Exponential Distribution
import numpy as np
from scipy.stats import expon
import matplotlib.pyplot as plt
fig, ax = plt.subplots(1, 1)
x = np.linspace(expon.ppf(0.01),
                expon.ppf(0.99), 100)
ax.plot(x, expon.pdf(x),
          'red', lw=5, alpha=0.6, label='expon pdf')
```

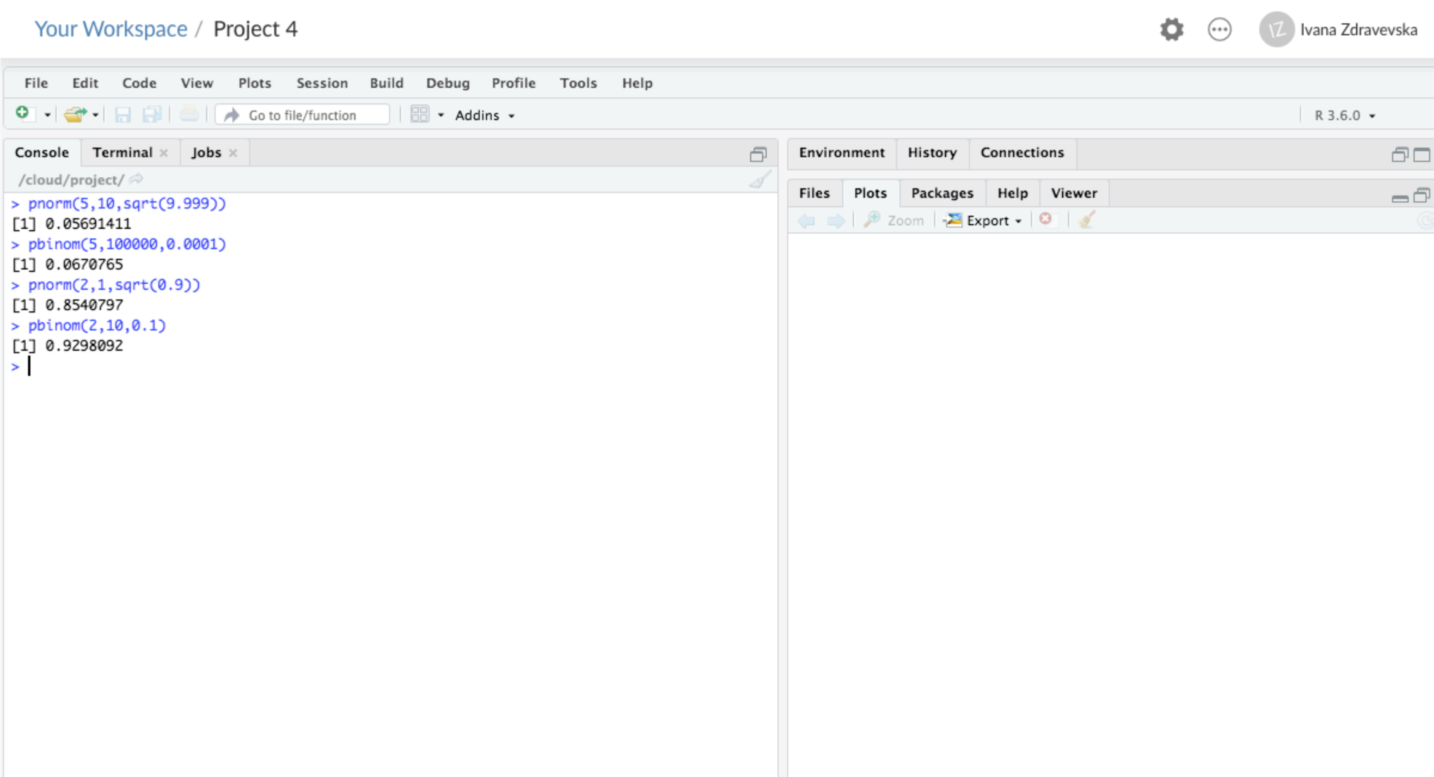
Out[20]:
```
[<matplotlib.lines.Line2D at 0x1a1d6a4ba8>]
```



Based on the generated histogram and the PDF plot of an exponential distribution, we can see that the histogram looks pretty similar to curve shown in the PDF with a constant rate. With lambda being 25, the expected value or the mean is 1/lambda or 1/25. This means that the frequency of values closet to 0.04 get higher as they reach the mean than the values away from the mean and this is shown in the histogram. Since the histogram was based on a randomly generated array, it may not always look the same as the pdf, rather it should look similar. Therefore, the generated histogram corresponds to the PDF of the distribution.

Normal Approximation

The normal approximation probability function takes 3 parameters: the probability value, mean, and standard deviation. The mean can be derived by multiplying n and p, which are both given in the problem. The standard deviation is the square root of np(1-p). Having this information, the normal approximation can be calculated. The binomial distribution calculation takes 3 parameters: the probability value, n, and p. All of these are already given in the problem. Based on the results, we can see that the normal approximation is close to the binomial distribution results, but not exact. This is because it is only a close approximation to the binomial distribution. The following code was done in RStudio.


Your Workspace / Project 4
Ivana Zdravevska
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
R 3.6.0
Console Terminal Jobs
/cloud/project/
> pnorm(5,10,sqrt(9.999))
[1] 0.05691411
> pbinom(5,100000,0.0001)
[1] 0.0670765
> pnorm(2,1,sqrt(0.9))
[1] 0.8540797
> pbinom(2,10,0.1)
[1] 0.9298092
>
Environment History Connections
Files Plots Packages Help Viewer
Zoom
Export