HW 08

This is a set of problems which focus on goodness-of-fit tests for various models of data.

This homework is due Tuesday 6/26 at midnight. Hand it in with Lab 08 by the due date and time.

Be sure to select Run All from the Cell menu before submitting this notebook as your solution!

In [469]:
```
# Jupyter notebook specific
from IPython.display import Image
from IPython.core.display import HTML
from IPython.display import display_html
from IPython.display import display
from IPython.display import Math
from IPython.display import Latex
from IPython.display import HTML

# General useful imports
import numpy as np
from numpy import log,arange,linspace,mean, var, std,exp
from scipy.special import comb
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
from numpy.random import seed,random, randint, uniform, choice, binomial, geometric, poisson,
import math
from collections import Counter
import pandas as pd
%matplotlib inline


# Numpy basic stats functions

# https://docs.scipy.org/doc/numpy-1.13.0/reference/routines.statistics.html

X = [1,2,3]

# mean of a list
mean(X)

# population variance
var(X)

# sample variance
var(X,ddof=1)

# population standard deviation
std(X)

# sample standard deviation
std(X,ddof=1)

# Scipy statistical functions

from scipy.stats import norm,expon,chi2

# https://docs.scipy.org/doc/scipy/reference/stats.html

# The following will work with norm replaced by expon or by chi2

# given random variable X (e.g., housing prices) normally distributed with  mu = 60, sigma = 4


# pdf of the distribution
norm.pdf(x=50,loc=60,scale=40)
chi2.pdf(x=10,df=5)

#a. Find P(X<50)
norm.cdf(x=50,loc=60,scale=40) # 0.4012936743170763
chi2.cdf(x=10,df=5)

#a. Find P(X>50)
norm.sf(x=50,loc=60,scale=40) # 0.4012936743170763
chi2.sf(x=10,df=5)

#c. Find P(60<X<80)
norm.cdf(x=80,loc=60,scale=40) - norm.cdf(x=60,loc=60,scale=40)

#d. how much top most 5% expensive house cost at least? or find x where P(X>x) = 0.05
```

```
norm.isf(q=0.05,loc=60,scale=40)

#e. how much top most 5% cheapest house cost at least? or find x where P(X<x) = 0.05
norm.ppf(q=0.05,loc=60,scale=40)

#f give the endpoints of the range for the central alpha percent of the distribution
norm.interval(alpha=0.3, loc=60, scale=140)

# Same for exponential distribution
beta = 5    # mean of distribution



# Utility functions

# Round to 4 decimal places
def round4(x):
    return round(float(x)+0.00000000001,4)

def round4List(X):
    return [round(float(x)+0.00000000001,4) for x in X]

def show_histogram(freqs,bins,title="Data Histogram"):
    plt.bar(bins[:-1],freqs,width=(bins[1]-bins[0]),align='edge',tick_label=round4List(bins[:-
    plt.title(title)
    plt.ylabel("Frequency")
    plt.xlabel("Bins [Lo .. Hi)")
    plt.show()
```

Problem Zero (Optional)

I suggest that you consolidate the code from lab so that you can take a set of observations and a theoretical distribution (given as a list of probabilies) and print out the results of the test. Use this in problems 1 and 2.

In [470]:

```
# Obs = set of observations; Probs = matching set of probabilities for the distribution to tes
# los = level of significance

def do_chi2_hypothesis_test(Obs,Probs,los, H0 = "The data follows the given distribution."):
    pass


```

Now test your function on Example B.1 on p.285 of Schaum's (same as we went over in class) and confirm that you implemented it correctly.

In [471]:

```
Obs = [80,65,70,85]
n = len(Obs)
Uniform = [1/n for k in range(n)]
do_chi2_hypothesis_test(Obs,Uniform,0.05)
```

```
H0 = The data follows the given distribution.
p-value = 0.343030146138
Fail to reject H0 at the 0.05 level of significance.
```

Problem One (Chi^2 Hypothesis Testing -- Normal Distribution)

For this problem, we would like you to apply the χ^2 test to data following the normal distribution. The basic ideas of the test are shown in Schaum's Appendix B, Example B.5 on p.288. Before starting this problem, please look at that example.

The χ^2 test is designed for discrete distributions, so to test a continuous distribution we need to convert it into a discrete one by "binning" the data into some number of intervals. The first part of this problem shows how to do this easily using the numpy `histogram` function.

Part A

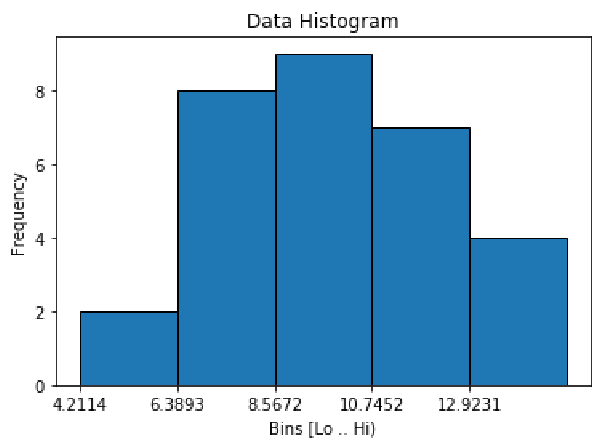
In order to bin the observed and theoretical values, we will use the numpy function `histogram`, which takes a list of outcomes and counts the frequencies in `num_bins` bins spread evenly over the range; it returns a list of frequencies for each bin (= how many values from the data set ended up in that bin) and a list of the bin boundaries.

For example, we can separate a set of observed data points into 4 bins as follows:

In [472]:

```

X1a = [ 11.69690176,   6.94784341,   4.21136807,   9.21746662,   6.84108737,
  13.08583604 , 12.22008482 ,  7.37578756 ,  7.19696017,  11.58233656,
   8.79227342 ,  9.77640417 , 13.29730199 ,  9.92861583 ,  6.82766429,
  10.55806493 , 12.74073126 ,  8.13821793 , 13.4815449,    9.67843858,
  12.49634904 , 12.834129   ,  9.66832643 ,  6.30537823,   9.78097709,
  15.10102417 ,  7.77790787 ,  8.17102049 , 10.51004163,  10.89602475]

obs,bins = np.histogram(X1a,bins=5)

print(obs)
print(bins)

show_histogram(obs,bins)

```

```
[2 8 9 7 4]
[  4.21136807   6.38929929   8.56723051  10.74516173  12.92309295
  15.10102417]
```



The bins are the ranges (values rounded to 4 decimal places):

```
[4.2114 .. 6.3893), [6.3893 .. 8.5672), [8.5672 .. 10.7452),  [ 10.7452 .. 12.9231),  [12.9231 .. 15.101].
```

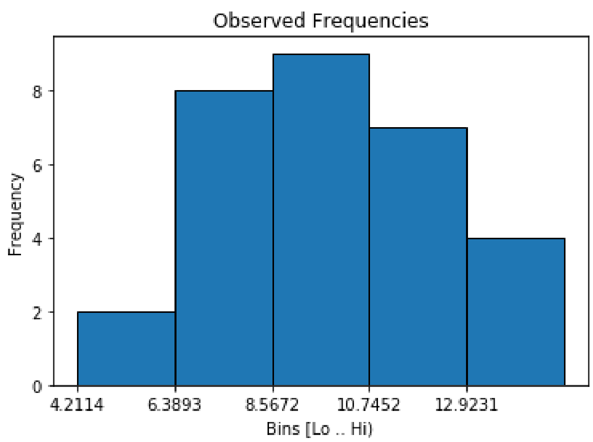
Note that the variable `bins` is the list of boundaries between each of the bins, so its length is `num_bins + 1`.

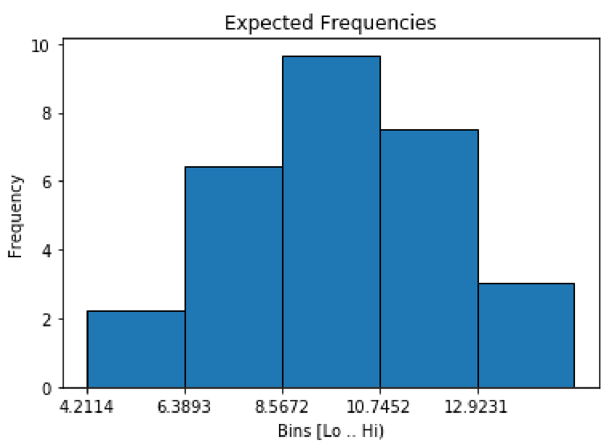
Now complete the following function stub to take a data set (which is potentially from a normal distribution with mean mu and standard deviation sigma) and bin the data into a set Obs of observed frequencies, and also produce a list Probs of expected probabilities for these bins if the data followed a normal distribution with the same mean and standard deviation as the data set.

In [473]:

```
def get_obs_and_normal_probs(X,num_bins):
    pass

    return (obs,probs,bins)

```

In [474]:
```
# Test your function on the data in X1a

Obs,Probs,bins = get_obs_and_normal_probs(X1a,5)

show_histogram(Obs,bins,"Observed Frequencies")
# convert probabilities into expected frequencies
Exp = [p*len(X1a) for p in Probs]
show_histogram(Exp,bins,"Expected Frequencies")

# Also: try 10 bins or 20 and see what happens!

# As a rough rule-of-thumb, you want at least approximately 5 values in each bin, so
# the largest number of bins would be # of data points / 5. In this case, that would give
# us an upper bound of 30/5 = 6 bins.  That is why we chose 5.

```



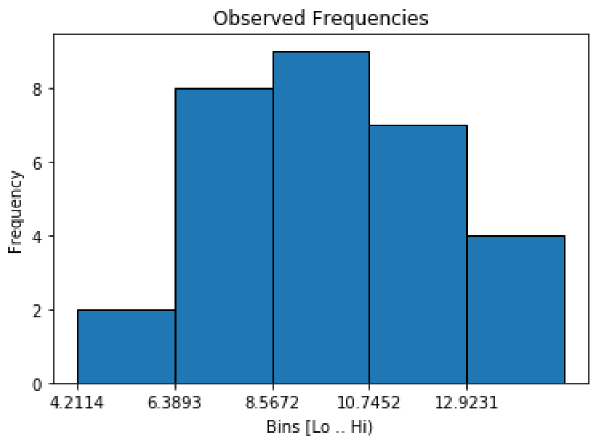


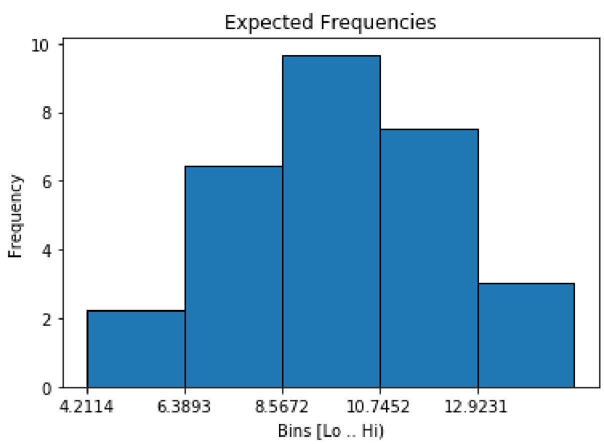
Part B

Now complete the following stub to do the normal distribution test on the data in X1a, as in the lab, using the code from Problem 0 (if you wish).

In [475]:
```
def do_chi2_normal_test(X,num_bins,los):
    pass

```

In [476]:
```
X1b = X1a
do_chi2_normal_test(X1b,5,0.1)

# You should get a large p-value, which means that the data is very, very likely to be normal
```



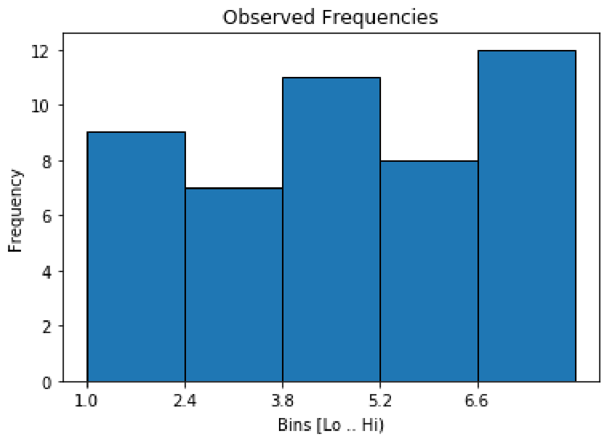


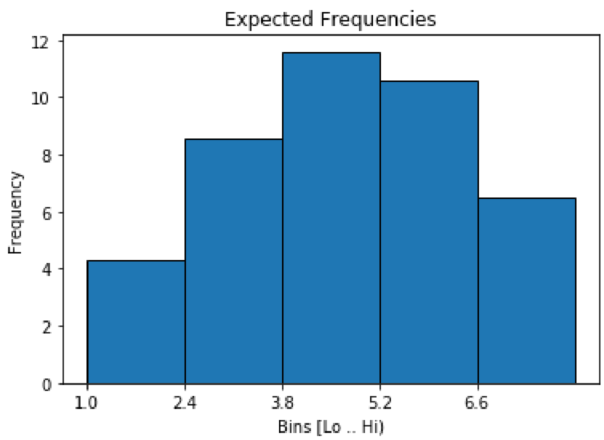

```
H0 = The data follows a normal distribution.
p-value = 0.938696801217
Fail to reject H0 at the 0.1 level of significance.
```

Part C

Now test the following data set using 5 bins and los = 0.05

In [477]:
```
X1c = [2,5,4,3,6,5,4,2,3,2,3,2,3,2,3,2,7,3,2,6,6,6,6,6,7,4,5,6,8,7,1,2,8,8,8,8,8,8,8,5,4,3,6,5
do_chi2_normal_test(X1c,5,0.05)

```






```
H0 = The data follows a normal distribution.
p-value = 0.0290361830785
Reject H0 at the 0.05 level of significance.
```

Problem Two (Chi^2 Normal Test for Real Data)

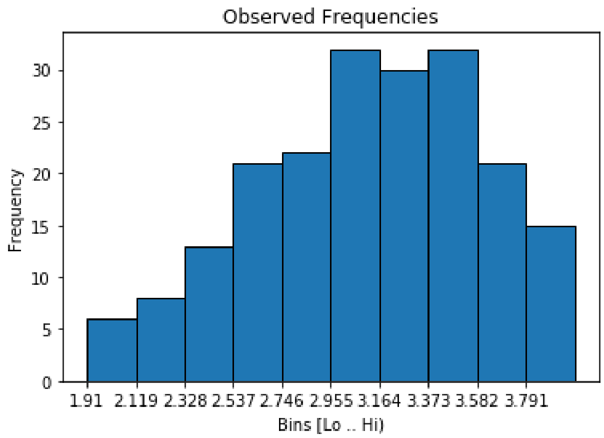
In this problem you will test a number of real data sets to see how well a normal distribution fits the data.

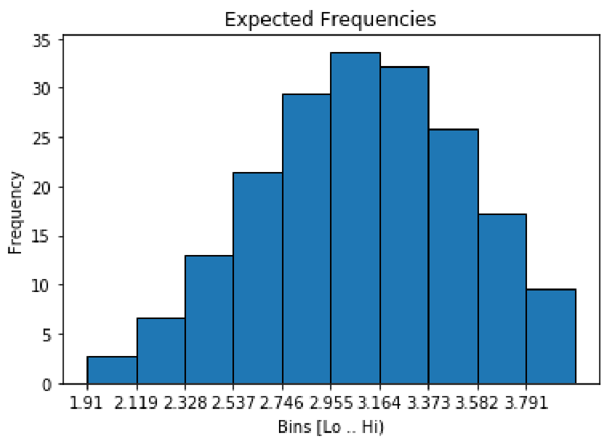
Part A

First we will test a small fraction of the BU GPA data to determine if the GPAs follow a normal distribution to a 0.05 level of significance.

In [478]:

```
# Test the file "BUDataSmall.csv" to see if the GPA data fits a normal distribution with los =
# Use 10 bins.
```



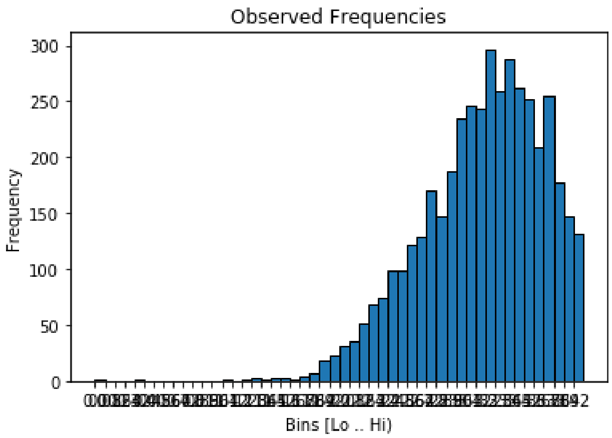


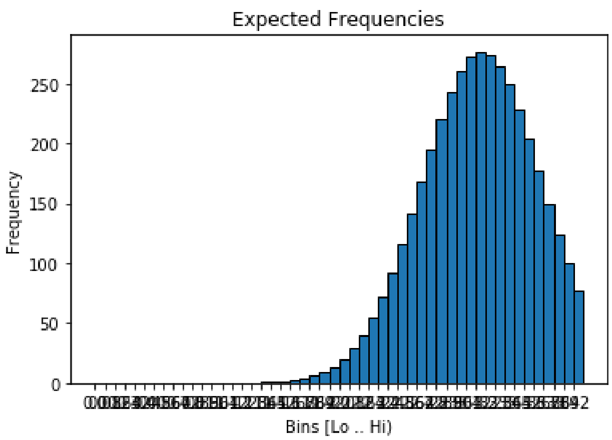

```
H0 = The data follows a normal distribution.
p-value = 0.241844264514
Fail to reject H0 at the 0.05 level of significance.
```

Part B

In [479]:

```
# Test the file "BUData.csv" to see if the whole data set of GPAs fits a normal distribution v
# Use 50 bins.



```

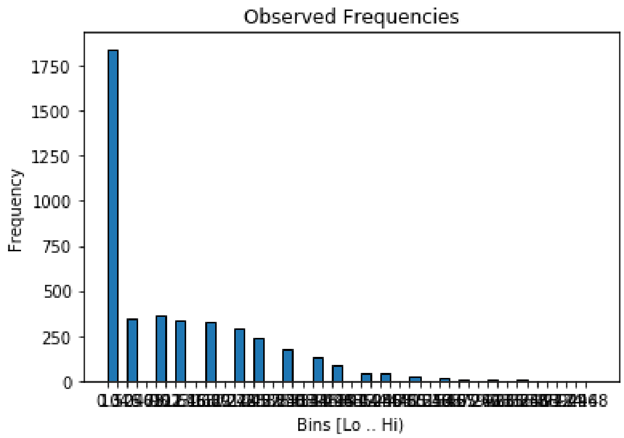


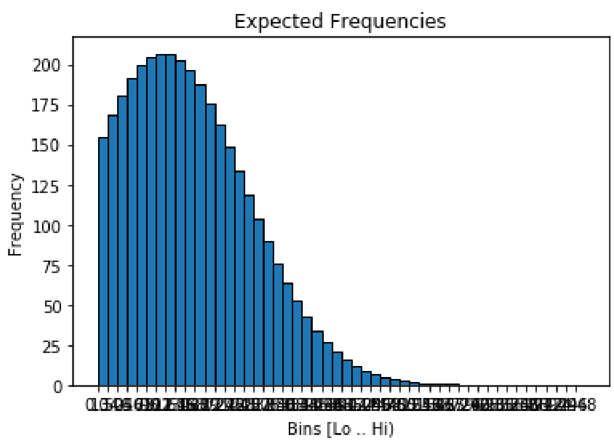



```
H0 = The data follows a normal distribution.
p-value = 0.0
Reject H0 at the 0.05 level of significance.
```

Part C

In [480]:
```
# Test the file "BUData.csv" to see if the whole data set of GPAs fits a normal distribution v
# Use 50 bins.



```

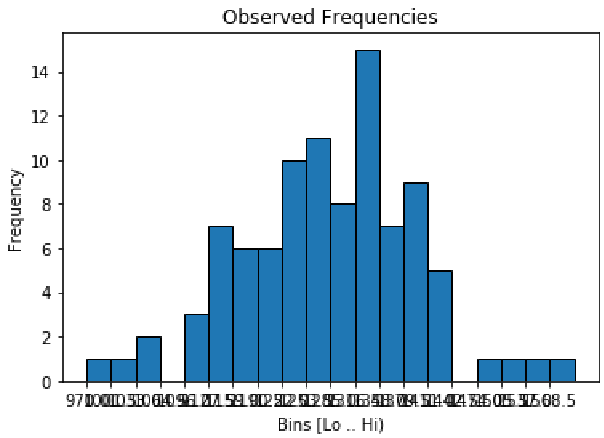


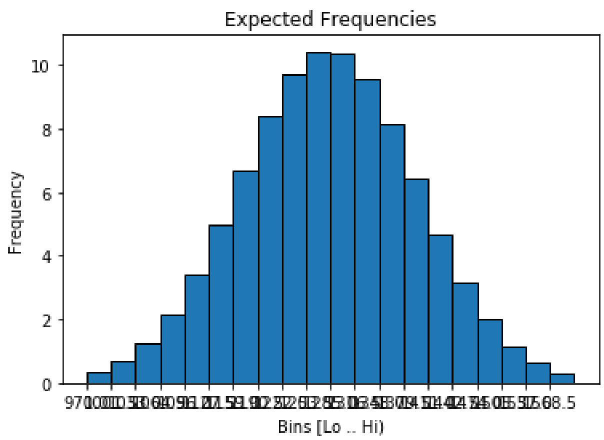



```
H0 = The data follows a normal distribution.
p-value = 0.0
Reject H0 at the 0.05 level of significance.
```

Part D

In [481]:
```
# Test the file "StudentData3.csv" to see if the SAT_TOTAL data fits a normal distribution wit
# Use 20 bins.



```






```
H0 = The data follows a normal distribution.
p-value = 0.660030575093
Fail to reject H0 at the 0.05 level of significance.
```

Hypothesis Testing using Normal Probability Plots

Basic reference: http://www.itl.nist.gov/div898/handbook/eda/section3/normprpl.htm (http://www.itl.nist.gov/div898/handbook/eda/section3/normprpl.htm)

From the link http://www.itl.nist.gov/div898/handbook/eda/section3/eda3676.htm (http://www.itl.nist.gov/div898/handbook/eda/section3/eda3676.htm):

"The test statistic is the correlation coefficient ρ of the points that make up a normal probability plot. This test statistic is compared with the critical value below. If the test statistic is less than the tabulated value, the null hypothesis that the data came from a population with a normal distribution is rejected."

NOTE: This framework uses an inverse table (compared with the textbook definition of LOS), since you reject when your statistic ρ is less than the value in the table.

In [482]:

```
# Critical Values for the normal probability plot correlation coefficient test for
# normality of a data set, from http://www.itl.nist.gov/div898/handbook/eda/section3/eda3676.

# First column is N = number of samples, second is for 0.01 LOS, third is for 0.05 LOS

CV = [[   3   ,      0.8687        ,  0.8790],
[   4   ,      0.8234       ,  0.8666],
[   5   ,      0.8240       ,  0.8786],
[   6   ,      0.8351       ,  0.8880],
[   7   ,      0.8474       ,  0.8970],
[   8   ,      0.8590       ,  0.9043],
[   9   ,      0.8689       ,  0.9115],
[  10   ,      0.8765       ,  0.9173],
[  11   ,      0.8838       ,  0.9223],
[  12   ,      0.8918       ,  0.9267],
[  13   ,      0.8974       ,  0.9310],
[  14   ,      0.9029       ,  0.9343],
[  15   ,      0.9080       ,  0.9376],
[  16   ,      0.9121       ,  0.9405],
[  17   ,      0.9160       ,  0.9433],
[  18   ,      0.9196       ,  0.9452],
[  19   ,      0.9230       ,  0.9479],
[  20   ,      0.9256       ,  0.9498],
[  21   ,      0.9285       ,  0.9515],
[  22   ,      0.9308       ,  0.9535],
[  23   ,      0.9334       ,  0.9548],
[  24   ,      0.9356       ,  0.9564],
[  25   ,      0.9370       ,  0.9575],
[  26   ,      0.9393       ,  0.9590],
[  27   ,      0.9413       ,  0.9600],
[  28   ,      0.9428       ,  0.9615],
[  29   ,      0.9441       ,  0.9622],
[  30   ,      0.9462       ,  0.9634],
[  31   ,      0.9476       ,  0.9644],
[  32   ,      0.9490       ,  0.9652],
[  33   ,      0.9505       ,  0.9661],
[  34   ,      0.9521       ,  0.9671],
[  35   ,      0.9530       ,  0.9678],
[  36   ,      0.9540       ,  0.9686],
[  37   ,      0.9551       ,  0.9693],
[  38   ,      0.9555       ,  0.9700],
[  39   ,      0.9568       ,  0.9704],
[  40   ,      0.9576       ,  0.9712],
[  41   ,      0.9589       ,  0.9719],
[  42   ,      0.9593       ,  0.9723],
[  43   ,      0.9609       ,  0.9730],
[  44   ,      0.9611       ,  0.9734],
[  45   ,      0.9620       ,  0.9739],
[  46   ,      0.9629       ,  0.9744],
[  47   ,      0.9637       ,  0.9748],
[  48   ,      0.9640       ,  0.9753],
[  49   ,      0.9643       ,  0.9758],
[  50   ,      0.9654       ,  0.9761],
[  55   ,      0.9683       ,  0.9781],
[  60   ,      0.9706       ,  0.9797],
[  65   ,      0.9723       ,  0.9809],
[  70   ,      0.9742       ,  0.9822],
[  75   ,      0.9758       ,  0.9831],
[  80   ,      0.9771       ,  0.9841],
[  85   ,      0.9784       ,  0.9850],
[  90   ,      0.9797       ,  0.9857],
[  95   ,      0.9804       ,  0.9864],
[ 100   ,      0.9814       ,  0.9869],
[ 110   ,      0.9830       ,  0.9881],
[ 120   ,      0.9841       ,  0.9889],
[ 130   ,      0.9854       ,  0.9897],
[ 140   ,      0.9865       ,  0.9904],
[ 150   ,      0.9871       ,  0.9909],
[ 160   ,      0.9879       ,  0.9915],
[ 170   ,      0.9887       ,  0.9919],
[ 180   ,      0.9891       ,  0.9923],
```

```
[ 190   ,      0.9897       ,  0.9927],
[ 200   ,      0.9903       ,  0.9930],
[ 210   ,      0.9907       ,  0.9933],
[ 220   ,      0.9910       ,  0.9936],
[ 230   ,      0.9914       ,  0.9939],
[ 240   ,      0.9917       ,  0.9941],
[ 250   ,      0.9921       ,  0.9943],
[ 260   ,      0.9924       ,  0.9945],
[ 270   ,      0.9926       ,  0.9947],
[ 280   ,      0.9929       ,  0.9949],
[ 290   ,      0.9931       ,  0.9951],
[ 300   ,      0.9933       ,  0.9952],
[ 310   ,      0.9936       ,  0.9954],
[ 320   ,      0.9937       ,  0.9955],
[ 330   ,      0.9939       ,  0.9956],
[ 340   ,      0.9941       ,  0.9957],
[ 350   ,      0.9942       ,  0.9958],
[ 360   ,      0.9944       ,  0.9959],
[ 370   ,      0.9945       ,  0.9960],
[ 380   ,      0.9947       ,  0.9961],
[ 390   ,      0.9948       ,  0.9962],
[ 400   ,      0.9949       ,  0.9963],
[ 410   ,      0.9950       ,  0.9964],
[ 420   ,      0.9951       ,  0.9965],
[ 430   ,      0.9953       ,  0.9966],
[ 440   ,      0.9954       ,  0.9966],
[ 450   ,      0.9954       ,  0.9967],
[ 460   ,      0.9955       ,  0.9968],
[ 470   ,      0.9956       ,  0.9968],
[ 480   ,      0.9957       ,  0.9969],
[ 490   ,      0.9958       ,  0.9969],
[ 500   ,      0.9959       ,  0.9970],
[ 525   ,      0.9961       ,  0.9972],
[ 550   ,      0.9963       ,  0.9973],
[ 575   ,      0.9964       ,  0.9974],
[ 600   ,      0.9965       ,  0.9975],
[ 625   ,      0.9967       ,  0.9976],
[ 650   ,      0.9968       ,  0.9977],
[ 675   ,      0.9969       ,  0.9977],
[ 700   ,      0.9970       ,  0.9978],
[ 725   ,      0.9971       ,  0.9979],
[ 750   ,      0.9972       ,  0.9980],
[ 775   ,      0.9973       ,  0.9980],
[ 800   ,      0.9974       ,  0.9981],
[ 825   ,      0.9975       ,  0.9981],
[ 850   ,      0.9975       ,  0.9982],
[ 875   ,      0.9976       ,  0.9982],
[ 900   ,      0.9977       ,  0.9983],
[ 925   ,      0.9977       ,  0.9983],
[ 950   ,      0.9978       ,  0.9984],
[ 975   ,      0.9978       ,  0.9984],
[1000   ,      0.9979       ,  0.9984]]


# Approximation to uniform order statistic medians due to Fillben:
#  http://www1.cmc.edu/pages/faculty/MONeill/Math152/Handouts/filliben.pdf

def U(i,n):
    if(i == n):
        return 0.5**(1/n)
    elif(i == 1):
        return 1 - 0.5**(1/n)
    else:
        return (i - 0.3175)/(n + 0.365)

# Normal Order Statistic Medians

def NOSM(i,n,mu,sigma):
    return norm.ppf(q=U(i,n),loc=mu,scale=sigma)

def getCVIndex(n,lo,hi):
    if(lo >= hi):
```

```
        return lo
    mid = (lo + hi)//2
    if(CV[mid][0] == n):
        return mid
    elif(CV[mid][0] < n):
        return getCVIndex(n,mid+1,hi)
    else:
        return getCVIndex(n,lo,mid-1)

# returns the critical value or the next lower value if not in the table

def getCriticalValue(N,los):
    N = min(N,1000)
    i = getCVIndex(N,0,len(CV))
    if(los == 0.01):
        return CV[i][1]
    else:
        return CV[i][2]

```

Problem Three (Tests using Normal Probability Plots)

In this problem you will write a function similar to the one above, but this time performing a test for fittness of the normal distribution using a normal probability plot. A big difference from the χ^2 test is that we can do this directly on the data and no "binning" of values is necessary. Thus, it is usually a more sensitive test than χ^2 for normal data.

The cell above contains all the code you need to do this test, supposing that your test data is in the list Y, here the outline of what you need to do in filling in the stub in the next cell:

```
N = len(Y)

los = level of significance of test.
```

The predicted values on the x-axis are produced as follows:

```
X = [NOSM(i,N,mean(X),std(X,ddof=1)) for i in range(1,N+1)]          # we use the sample s
td dev
```

and the test is performed as follows:

```
If rho(X,Y) < getCriticalValue(N,los) then reject, else fail to reject.
```

To display the probability plot, sort Y and draw a scatter plot of X against the sorted Y. To draw the linear prediction, plot X against X.

Part One

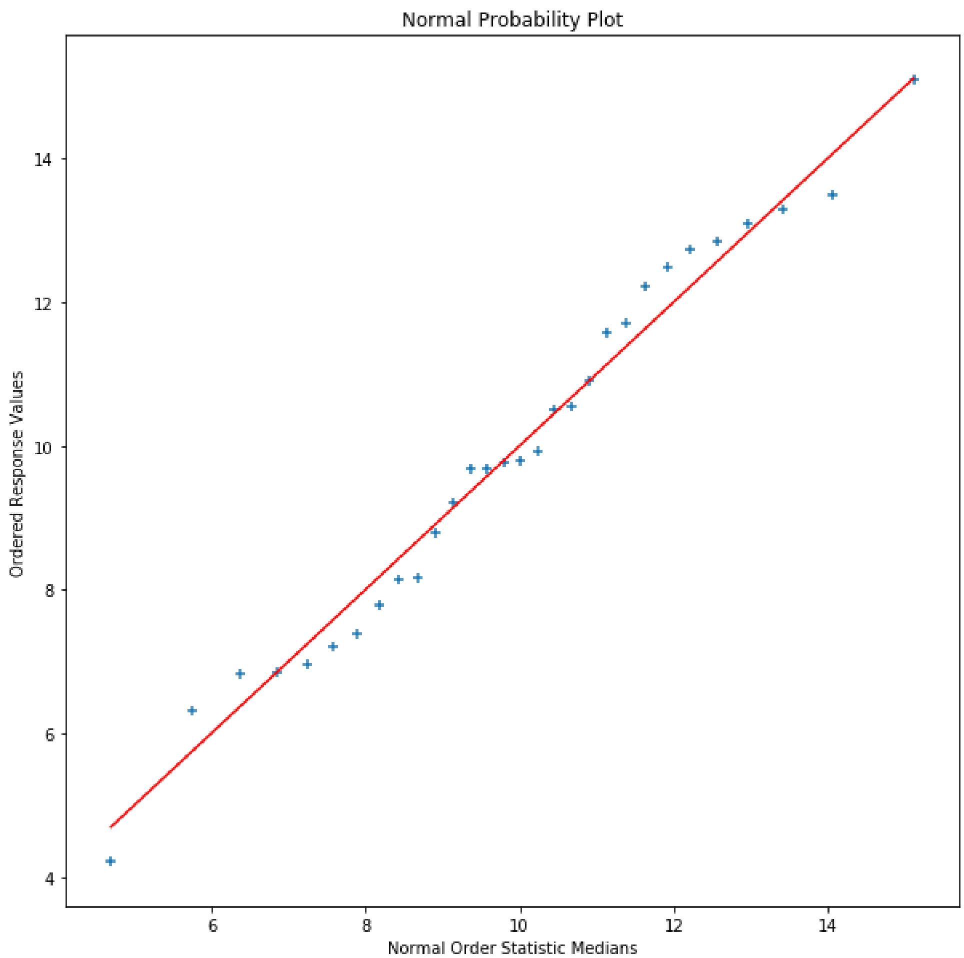
Complete the following function stub and test it on the data X1a with los = 0.01.

You must draw the probability plot and then print out the result of the test.

In [483]:

```
def do_pplot_normal_test(Y,los):
    pass



```

In [484]:
```
# Test X1a with los = 0.01

do_pplot_normal_test(X1a,0.01)
```

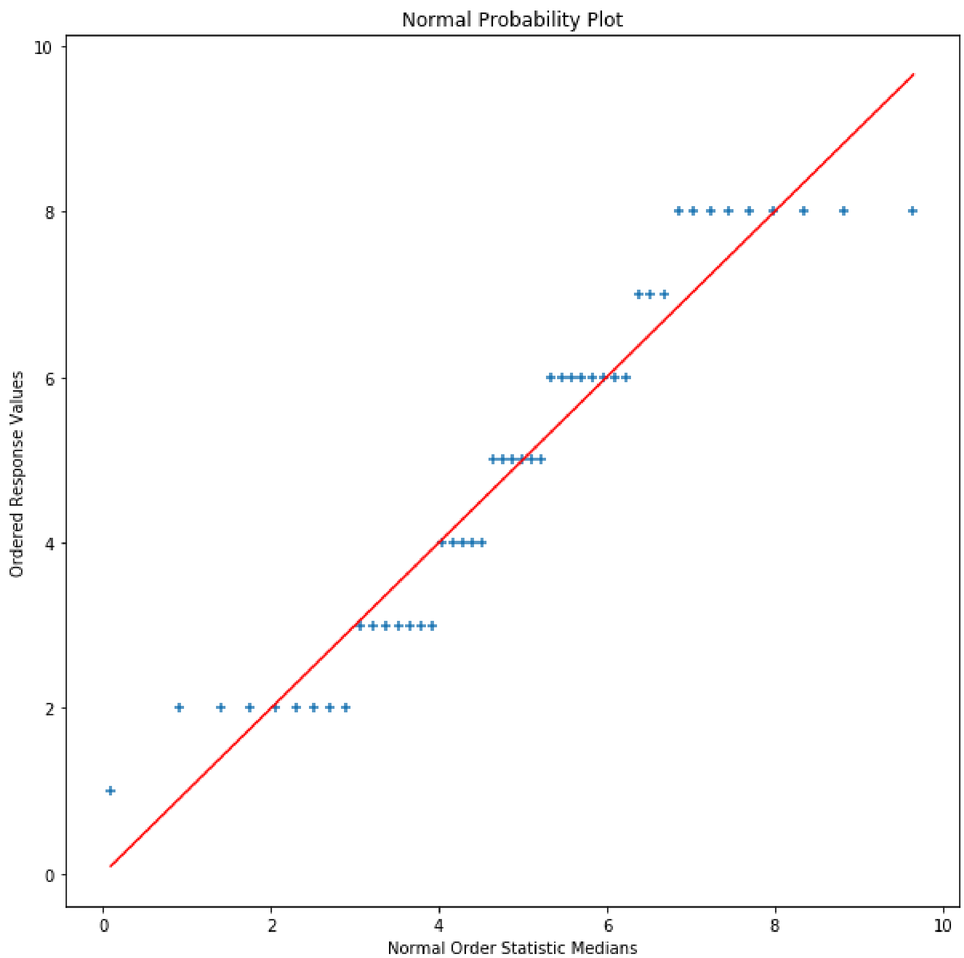



```
rho            = 0.9909
critical value = 0.9462

H0 = The samples come from a normal distribution.

Fail to reject at a 0.01 level of significance.
```

Part B

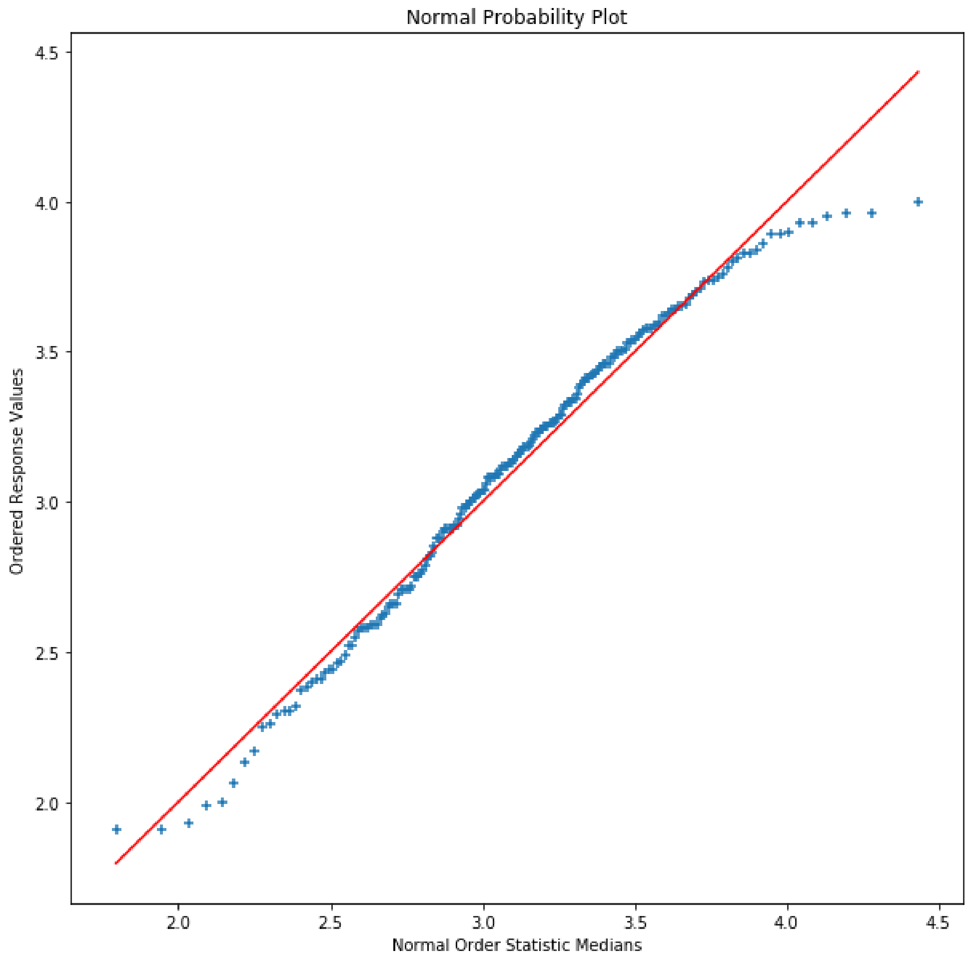
Now we would like you to test the data in X1c. It will fail, even at a 0.1 level of significance, because it is fundamentally a discrete distribution (the outcomes are all integers), and the probability plot expects a continuous distribution. Hence, in this case the χ^2 test is more appropriate.

In [485]:
```
# Test X1c with los = 0.1
```



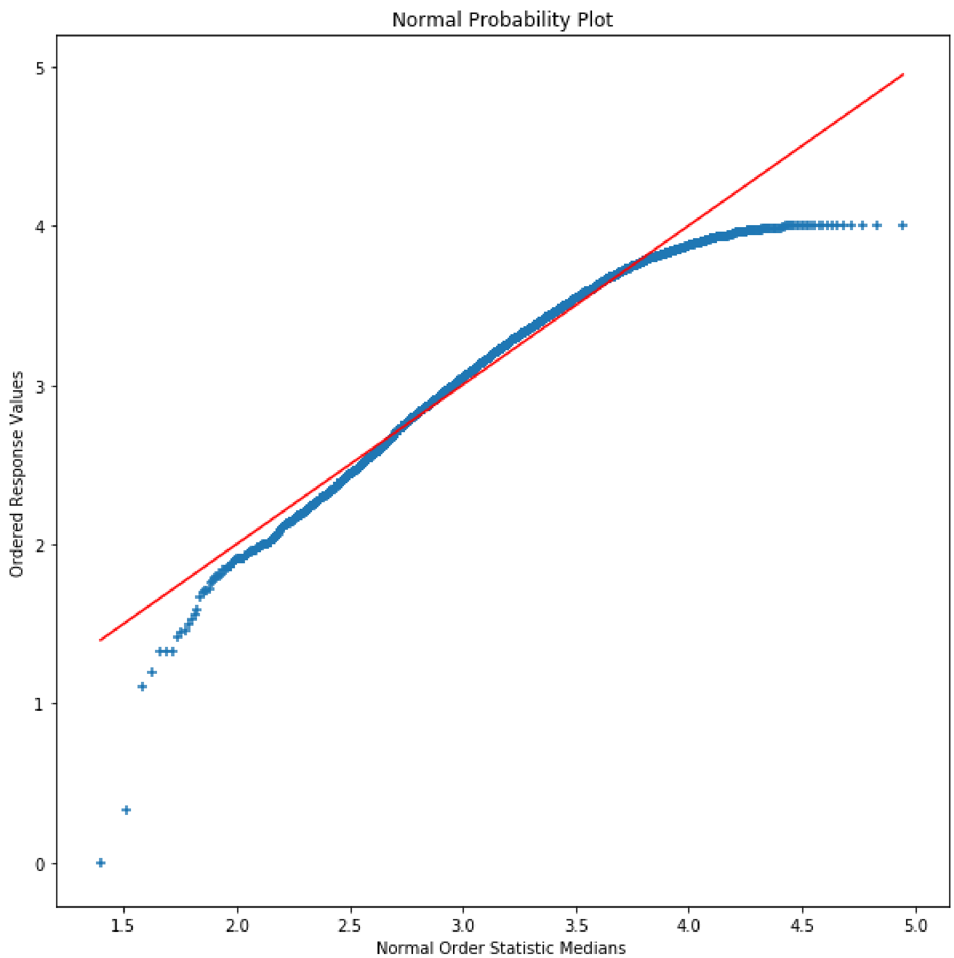

```
rho            = 0.9637
critical value = 0.9748

H0 = The samples come from a normal distribution.

Reject at a 0.1 level of significance.
```

In [486]:
```
# Test X2a with los = 0.05


```



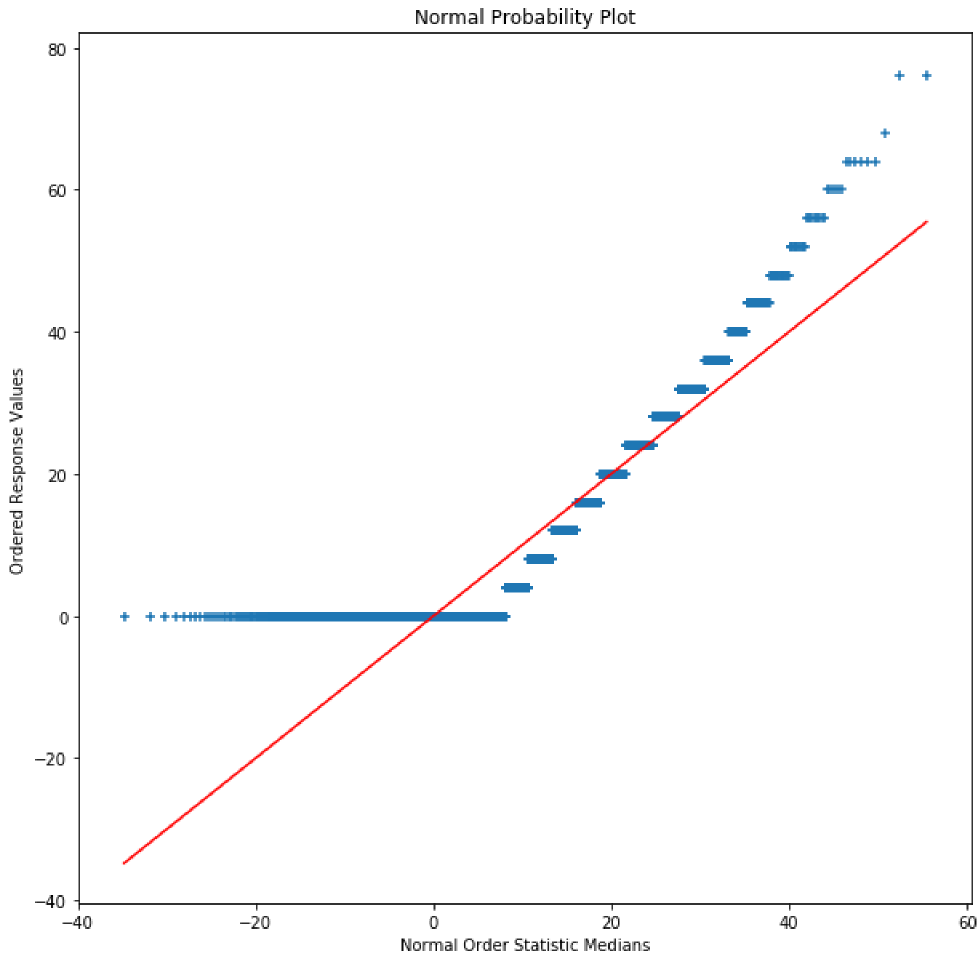

```
rho            = 0.9911
critical value = 0.993

H0 = The samples come from a normal distribution.

Reject at a 0.05 level of significance.
```

In [487]:
```
# Test X2b with los = 0.05


```



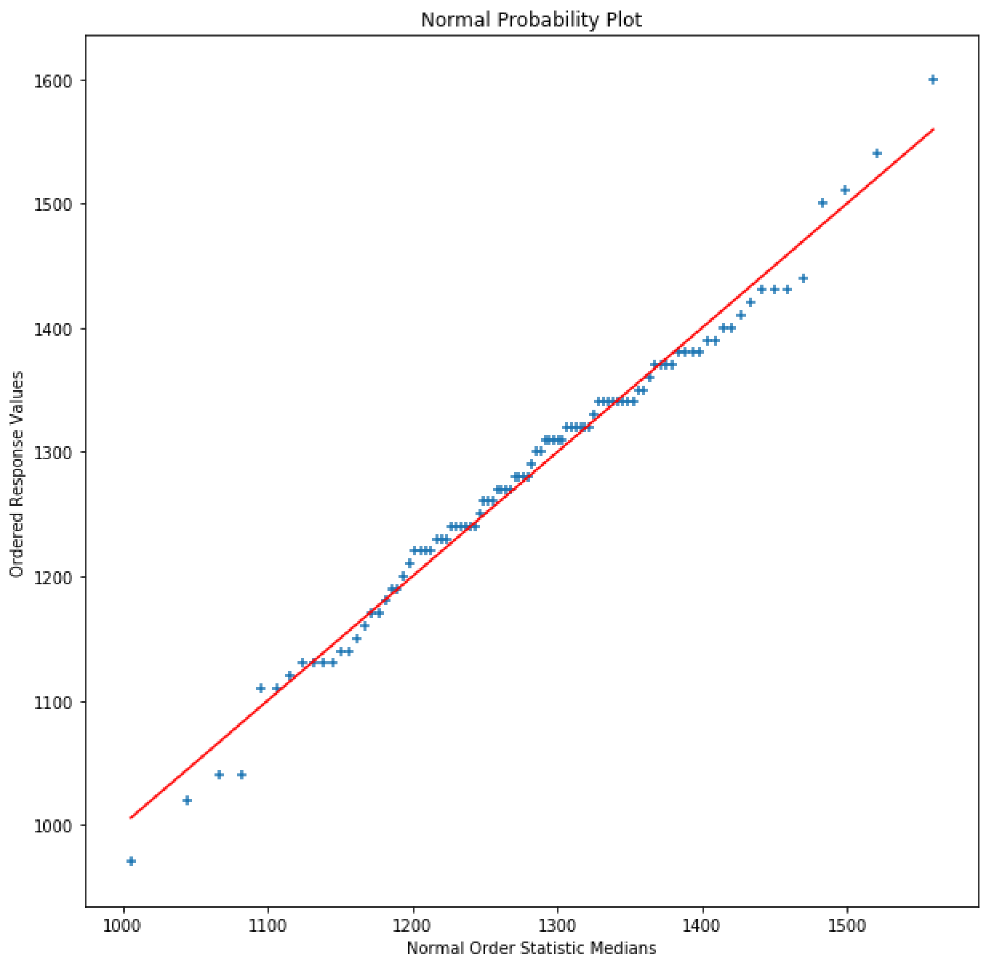

```
rho            = 0.9833
critical value = 0.9984

H0 = The samples come from a normal distribution.

Reject at a 0.05 level of significance.
```

In [488]:
```
# Test X2c with los = 0.05
```




```
rho            = 0.9005
critical value = 0.9984

H0 = The samples come from a normal distribution.

Reject at a 0.5 level of significance.
```

In [489]:
```
# Test X2d with los = 0.05


```




```
rho            = 0.993
critical value = 0.9864

H0 = The samples come from a normal distribution.

Fail to reject at a 0.05 level of significance.
```