Assignment 2: Applied data science

**PROBLEMS:**

1. In words, the “standard normal distribution” is a normal distribution with mean zero and variance one, denoted here as N(0,1). For a random variable X that is distributed as a standard normal, mathematically we write . [5 points]
   1. Using R or Python, write code to draw at random 10 observations from a N(0,1) random variable. Instruct the machine to calculate the mean, variance and standard deviation of your draws.
   2. Repeat this exercise using 10,000 draws from a N(0,1), instructing again the machine to calculate the mean, variance and standard deviation of your draws.
   3. Repeat this exercise with 1,000,000 draws from a N(0,1), instructing again the machine to calculate the mean, variance and standard deviation of your draws.
   4. What conclusions, if any, do you draw from increasing the sample size?
   5. Submit your code and results.

**ANSWERS:**

1. Output:

Mean = 0.123408348806

Variance = 1.34614414078

Standard Deviation = 1.16023451973

1. Output:

Mean = 0.00445184390789

Variance = 0.987748960347

Standard Deviation = 0.993855603369

1. Output:

Mean = 0.00106250329193

Variance = 0.999348641121

Standard Deviation = 0.99967426751

1. Random values that are normally distributed shows more accurate distribution in larger number of sample.
2. Source code:

Source Code (with num\_samples being adjusted to number of samples a-c):

##############################################

##############################################

# Applied Data Science GX5004 #

# Assignment 2 #

# Dimas Rinarso Putro | drp354@nyu.edu #

##############################################

**import** argparse**,**csv**,** sys**,** os

**import** numpy **as** np

**from** matplotlib **import** pyplot **as** plt

**import** pylab

###No.1a-1c, just by changing the num\_samples###

###########################################

#variable initialization

mu**,** sigma **=** 0**,** 1

num\_samples **=** 10 #change to the desired sample numbers in 1a-1c

#generate random numbers

nor **=** np**.**random**.**normal**(**mu**,** sigma**,** num\_samples**)**

#calculate mean

**print** 'Mean = '**,**str**(**np**.**mean**(**nor**))**

#verify the standard deviation

**print** 'Variance = '**,**str**(**np**.**var**(**nor**,** ddof**=**1**))**

#verify the standard deviation

**print** 'Standard Deviation = '**,**str**(**np**.**std**(**nor**,** ddof**=**1**))**

#plot histogram

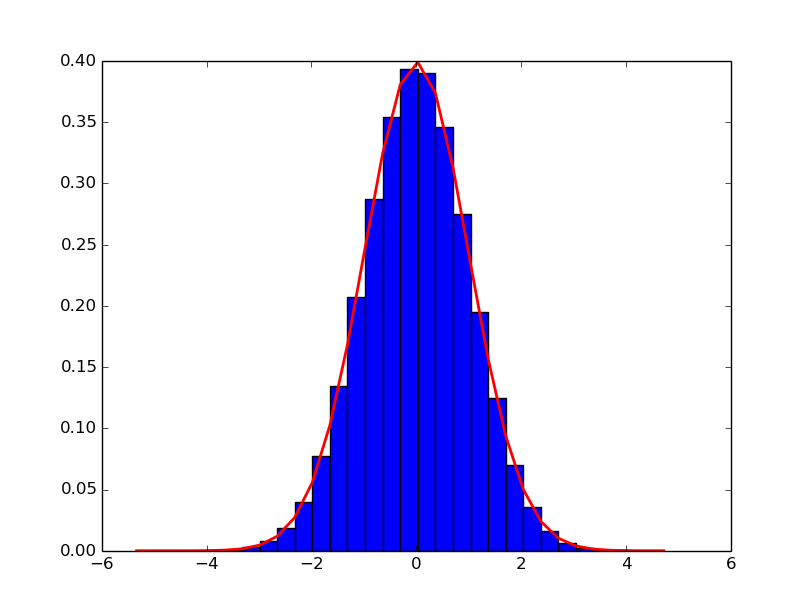
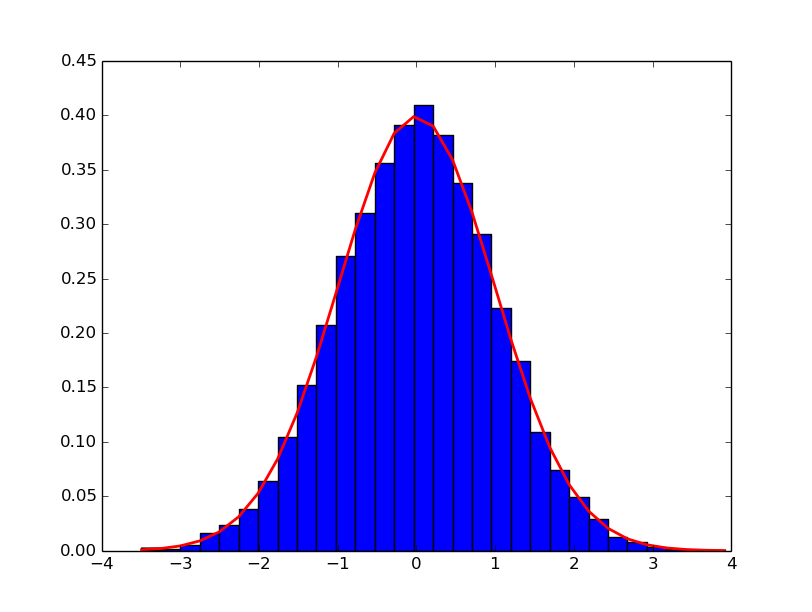
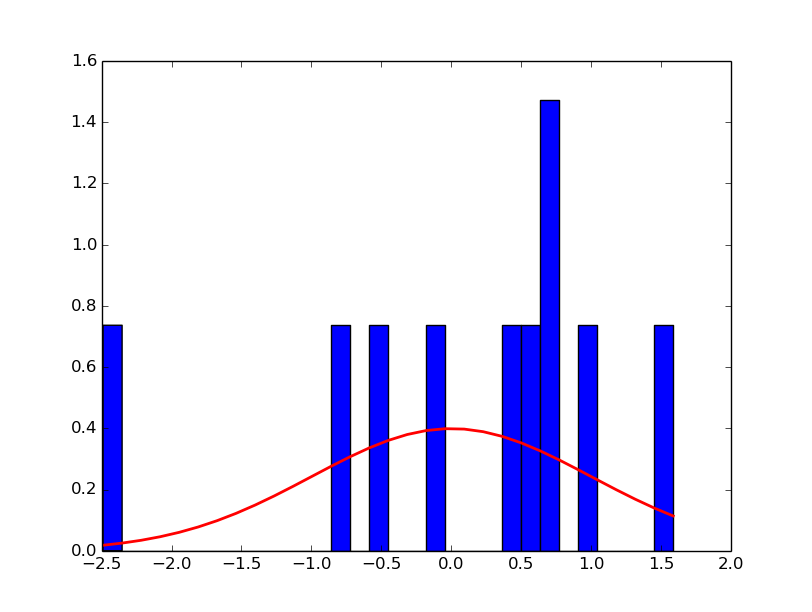
count**,** bins**,** ignored **=** plt**.**hist**(**nor**,** 30**,** normed**=True)**

plt**.**plot**(**bins**,** 1**/(**sigma **\*** np**.**sqrt**(**2 **\*** np**.**pi**))** **\***

np**.**exp**(** **-** **(**bins **-** mu**)\*\***2 **/** **(**2 **\*** sigma**\*\***2**)** **),**

linewidth**=**2**,** color**=**'r'**)**

plt**.**show**()**



**PROBLEMS:**

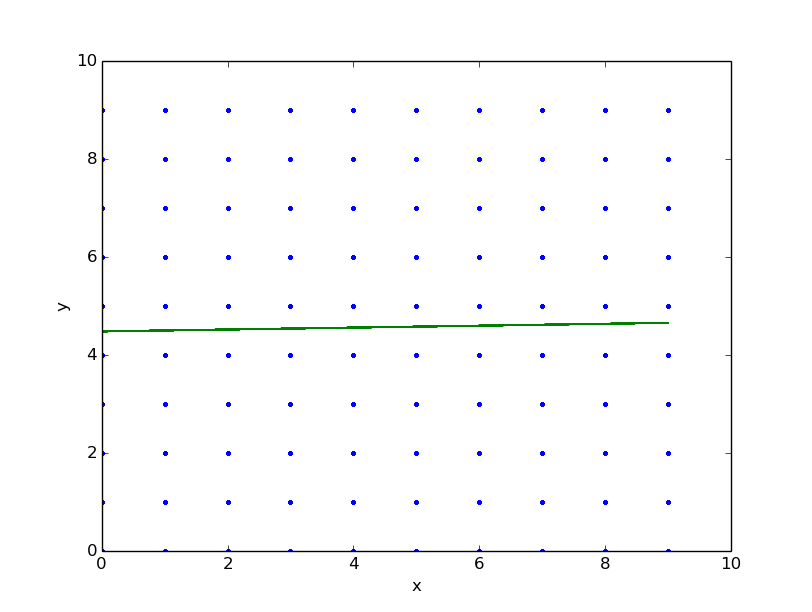
1. We discussed at some length the bivariate linear regression model, .

[5 points]

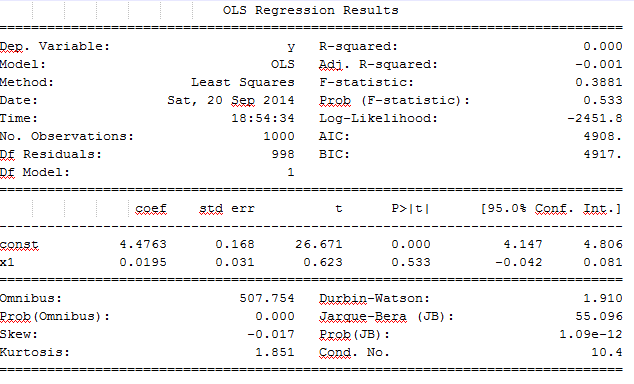
* 1. Go to <http://www.random.org/integers/> and generate two series of 1,000 random integers with values between 0 and 9. Call one series y and the other x.
  2. Using Python or R, fit the bivariate linear regression model.
  3. Examine your t-statistic to evaluate whether it is greater than two in absolute value. Would you reject or fail to reject that there is *any relationship* between these two series?
  4. Submit your series, your code, and your results.

**ANSWERS**

2.b. Plotting bivariate linear regression model:



2.c. t-statistic of the x and y

As shown on the OLS regression result, the value of *t* is 0.623. This means that the absolute value is less than 2 (|0.623|<2) so we should reject that there is a relation between two variables.

2.d. Source code and output:

##############################################

##############################################

# Applied Data Science GX5004 #

# Assignment 2 #

# Dimas Rinarso Putro | drp354@nyu.edu #

##############################################

**import** argparse**,**csv**,** sys**,** os

**import** numpy **as** np

**from** matplotlib **import** pyplot **as** plt

**import** pylab

**import** statsmodels**.**api **as** sm

###No.2###

###########################################

#variable initialization

mu**,** sigma **=** 0**,** 1

num\_samples **=** 1000000

filename**=**'no-2.csv'

x **=** **[]**

y **=** **[]**

**with** open**(**filename**,** 'rb'**)** **as** f**:**

csvReader **=** csv**.**reader**(**f**)**

headers **=** next**(**csvReader**)**

**for** row **in** csvReader**:**

x**.**append**(**int**(**row**[**0**]))**

y**.**append**(**int**(**row**[**1**]))**

**print** x

**print** y

##t-statistic

# Fit regression model

x **=** sm**.**add\_constant**(**x**)**

results **=** sm**.**OLS**(**y**,** x**).**fit**()**

# Inspect the results

**print** results**.**summary**()**

Output:

X = [8, 9, 9, 1, 6, 8, 8, 4, 7, 6, 9, 6, 1, 4, 1, 5, 9, 0, 7, 7, 2, 2, 1, 1, 4, 2, 7, 5, 3, 0, 2, 1, 8, 4, 0, 4, 7, 6, 4, 6, 4, 5, 1, 3, 1, 9, 7, 3, 0, 0, 9, 7, 5, 9, 2, 6, 6, 9, 9, 8, 5, 7, 5, 6, 5, 3, 8, 8, 0, 0, 0, 2, 0, 2, 4, 6, 6, 3, 1, 7, 5, 9, 5, 8, 2, 2, 1, 2, 6, 6, 0, 1, 9, 6, 3, 6, 1, 7, 8, 6, 5, 3, 5, 0, 5, 7, 2, 5, 7, 1, 5, 5, 7, 4, 2, 7, 9, 1, 0, 0, 2, 7, 3, 6, 9, 5, 6, 8, 9, 0, 8, 1, 2, 0, 3, 5, 2, 9, 4, 9, 8, 0, 8, 1, 3, 0, 2, 0, 1, 7, 3, 2, 7, 2, 3, 9, 7, 5, 3, 2, 2, 4, 2, 1, 1, 2, 3, 9, 4, 7, 5, 0, 5, 6, 3, 5, 9, 1, 9, 1, 0, 4, 9, 9, 9, 5, 1, 6, 9, 2, 2, 4, 4, 4, 4, 4, 1, 2, 5, 8, 2, 1, 5, 1, 2, 6, 0, 1, 6, 0, 8, 1, 6, 5, 8, 4, 7, 3, 1, 0, 9, 9, 0, 9, 6, 2, 3, 7, 4, 8, 4, 5, 7, 4, 8, 5, 8, 3, 6, 8, 7, 0, 8, 7, 2, 1, 5, 1, 6, 4, 0, 2, 2, 6, 2, 6, 8, 2, 2, 2, 5, 4, 7, 9, 5, 0, 6, 8, 6, 1, 3, 1, 1, 1, 5, 8, 7, 2, 3, 6, 9, 8, 9, 6, 3, 4, 6, 6, 3, 7, 4, 9, 3, 2, 4, 3, 1, 4, 4, 5, 1, 0, 5, 7, 6, 9, 0, 2, 0, 4, 8, 6, 5, 7, 4, 6, 5, 6, 7, 9, 4, 0, 6, 2, 1, 7, 7, 5, 5, 8, 2, 8, 0, 5, 4, 9, 1, 4, 4, 0, 3, 7, 5, 1, 5, 6, 8, 5, 2, 3, 2, 6, 9, 2, 1, 9, 4, 1, 6, 3, 6, 5, 7, 2, 1, 7, 2, 2, 7, 6, 0, 3, 4, 3, 7, 6, 7, 8, 6, 8, 3, 0, 5, 6, 6, 9, 4, 5, 0, 4, 4, 0, 3, 8, 8, 3, 6, 4, 9, 7, 0, 4, 9, 7, 4, 9, 0, 2, 4, 4, 2, 2, 9, 8, 4, 3, 4, 2, 7, 8, 2, 5, 6, 5, 0, 9, 1, 9, 8, 2, 1, 4, 1, 9, 3, 4, 6, 7, 1, 8, 1, 8, 9, 3, 5, 1, 7, 9, 5, 5, 9, 5, 0, 9, 0, 0, 3, 7, 9, 5, 0, 1, 5, 0, 8, 2, 4, 5, 8, 5, 3, 4, 9, 0, 1, 9, 0, 6, 2, 5, 7, 8, 1, 9, 8, 2, 6, 0, 8, 6, 2, 5, 2, 0, 0, 6, 2, 8, 4, 6, 0, 8, 4, 0, 1, 7, 6, 4, 9, 3, 0, 4, 6, 6, 5, 8, 3, 2, 7, 4, 8, 1, 1, 3, 3, 3, 2, 8, 8, 4, 9, 7, 6, 0, 9, 8, 2, 5, 7, 1, 8, 4, 0, 1, 8, 8, 2, 5, 9, 0, 4, 6, 0, 6, 9, 8, 5, 1, 3, 0, 0, 9, 5, 8, 8, 5, 8, 9, 5, 9, 0, 4, 8, 9, 2, 4, 4, 6, 7, 9, 6, 3, 5, 9, 2, 1, 0, 0, 9, 6, 4, 2, 6, 7, 7, 4, 9, 5, 8, 0, 5, 0, 9, 6, 0, 7, 7, 3, 4, 4, 1, 3, 7, 4, 5, 4, 3, 5, 7, 9, 9, 1, 2, 1, 3, 8, 4, 0, 4, 0, 3, 5, 3, 4, 1, 9, 1, 3, 2, 1, 4, 4, 1, 8, 9, 6, 9, 5, 3, 2, 3, 5, 8, 4, 2, 8, 5, 7, 3, 5, 2, 3, 0, 1, 4, 1, 3, 8, 0, 6, 3, 4, 6, 4, 5, 8, 1, 8, 1, 5, 6, 6, 6, 7, 9, 8, 0, 8, 2, 2, 4, 5, 2, 4, 6, 1, 7, 0, 4, 5, 0, 2, 3, 1, 6, 9, 6, 9, 3, 0, 1, 3, 9, 3, 1, 2, 4, 9, 3, 5, 5, 2, 5, 4, 7, 5, 9, 3, 1, 3, 7, 0, 6, 2, 3, 9, 5, 2, 6, 5, 2, 4, 8, 0, 3, 2, 8, 3, 6, 8, 6, 0, 7, 6, 6, 7, 8, 3, 2, 4, 8, 3, 9, 3, 5, 0, 9, 4, 3, 4, 4, 6, 6, 4, 6, 5, 5, 1, 6, 3, 4, 4, 4, 8, 7, 5, 6, 0, 6, 1, 6, 3, 6, 2, 8, 3, 5, 7, 6, 4, 8, 5, 2, 6, 4, 2, 0, 1, 5, 9, 1, 9, 9, 8, 9, 9, 8, 1, 4, 5, 9, 1, 8, 0, 2, 5, 8, 6, 2, 7, 3, 5, 6, 6, 8, 9, 5, 6, 8, 9, 9, 8, 7, 5, 1, 6, 5, 8, 1, 5, 0, 8, 0, 7, 8, 7, 4, 0, 6, 8, 3, 0, 7, 5, 3, 8, 8, 0, 9, 7, 1, 1, 8, 9, 0, 8, 2, 5, 1, 3, 6, 2, 8, 5, 8, 2, 6, 2, 0, 7, 4, 4, 7, 4, 2, 6, 7, 3, 7, 0, 1, 0, 7, 1, 6, 1, 8, 8, 1, 2, 1, 8, 6, 0, 2, 3, 8, 8, 5, 2, 0, 3, 2, 9, 1, 7, 2, 4, 6, 3, 7, 7, 9, 9, 2, 4, 6, 7, 4, 0, 0, 4, 5, 7, 3, 2, 1, 3, 1, 3, 8, 7, 5, 7, 6, 9, 7, 1, 3, 5, 3, 2, 6, 1, 5, 3, 5, 7, 9, 8, 8, 7, 8, 2, 7, 3, 8, 3, 9, 7, 9, 2, 6, 5, 1, 3, 3, 9, 0, 7, 8, 6, 3, 6, 3, 8, 3, 3, 6, 5]

Y = [1, 7, 8, 4, 9, 5, 6, 6, 1, 3, 1, 7, 0, 8, 8, 6, 5, 5, 0, 7, 1, 0, 5, 2, 2, 5, 2, 0, 7, 5, 3, 6, 7, 6, 4, 0, 1, 5, 4, 6, 3, 4, 6, 4, 8, 4, 5, 0, 2, 3, 3, 6, 3, 0, 2, 4, 3, 6, 2, 2, 0, 8, 2, 3, 4, 1, 6, 8, 9, 6, 4, 4, 5, 1, 9, 3, 5, 5, 3, 8, 9, 8, 9, 1, 4, 5, 2, 6, 5, 4, 4, 0, 1, 1, 1, 9, 4, 3, 8, 1, 6, 0, 3, 4, 0, 4, 5, 2, 4, 5, 5, 9, 2, 9, 5, 0, 6, 6, 1, 1, 7, 3, 0, 6, 9, 3, 5, 9, 0, 0, 1, 4, 2, 2, 7, 0, 6, 7, 5, 5, 6, 9, 0, 5, 1, 2, 3, 9, 0, 7, 1, 4, 0, 5, 8, 1, 4, 5, 0, 3, 3, 4, 3, 0, 8, 9, 1, 7, 8, 4, 6, 7, 8, 2, 6, 1, 2, 8, 7, 9, 6, 1, 5, 6, 9, 9, 7, 4, 1, 2, 2, 7, 7, 1, 2, 3, 3, 2, 5, 3, 7, 7, 4, 5, 5, 9, 1, 5, 4, 2, 7, 4, 0, 3, 9, 2, 5, 4, 9, 4, 6, 7, 5, 8, 6, 2, 1, 8, 1, 9, 5, 3, 7, 1, 5, 6, 4, 7, 9, 8, 3, 9, 6, 6, 7, 9, 5, 4, 6, 4, 8, 9, 9, 1, 8, 3, 5, 8, 3, 7, 5, 6, 2, 9, 9, 7, 1, 0, 3, 4, 2, 5, 7, 9, 2, 0, 2, 7, 0, 6, 5, 8, 6, 7, 1, 3, 4, 5, 3, 1, 9, 2, 3, 6, 2, 5, 4, 4, 8, 3, 4, 5, 9, 4, 2, 4, 5, 7, 2, 7, 7, 1, 4, 7, 3, 9, 6, 0, 1, 1, 5, 6, 2, 6, 1, 0, 1, 8, 9, 7, 8, 3, 7, 1, 2, 2, 2, 4, 8, 6, 7, 1, 8, 9, 0, 5, 1, 5, 8, 1, 1, 7, 5, 5, 3, 6, 6, 6, 2, 0, 3, 5, 4, 5, 2, 5, 8, 7, 3, 7, 0, 9, 7, 0, 8, 5, 1, 3, 7, 5, 9, 6, 8, 7, 8, 2, 8, 2, 2, 2, 6, 4, 2, 0, 8, 7, 7, 2, 2, 5, 2, 6, 3, 1, 8, 9, 9, 4, 7, 6, 1, 6, 0, 9, 7, 5, 2, 9, 1, 9, 5, 5, 6, 0, 9, 4, 5, 3, 3, 8, 3, 6, 9, 8, 7, 4, 8, 3, 1, 2, 1, 0, 7, 9, 5, 1, 9, 8, 0, 6, 5, 3, 5, 3, 7, 4, 1, 6, 8, 4, 3, 8, 8, 2, 4, 1, 4, 1, 6, 8, 1, 1, 6, 9, 9, 8, 6, 3, 3, 6, 4, 4, 0, 5, 3, 7, 9, 1, 9, 9, 1, 2, 9, 9, 9, 8, 5, 9, 3, 1, 1, 7, 2, 0, 4, 1, 3, 9, 5, 8, 6, 1, 6, 6, 6, 2, 4, 3, 8, 4, 2, 5, 2, 7, 5, 0, 0, 5, 4, 8, 6, 2, 8, 3, 0, 3, 3, 0, 7, 4, 4, 5, 7, 8, 9, 7, 7, 9, 9, 7, 1, 3, 4, 8, 6, 9, 5, 8, 4, 3, 3, 1, 4, 7, 0, 5, 1, 9, 7, 9, 3, 8, 9, 6, 9, 3, 4, 0, 4, 2, 0, 6, 4, 2, 9, 9, 3, 6, 2, 2, 8, 3, 5, 9, 9, 0, 0, 7, 5, 2, 5, 6, 9, 2, 5, 7, 0, 4, 5, 0, 3, 4, 0, 9, 6, 2, 2, 6, 0, 9, 6, 4, 9, 4, 4, 9, 0, 5, 0, 8, 8, 2, 0, 7, 1, 8, 4, 5, 6, 5, 1, 7, 1, 9, 4, 1, 6, 9, 2, 3, 7, 1, 9, 6, 5, 1, 3, 2, 2, 5, 1, 9, 9, 6, 0, 5, 0, 5, 4, 0, 1, 3, 0, 8, 5, 8, 3, 0, 6, 1, 8, 5, 7, 5, 7, 7, 2, 4, 6, 3, 1, 6, 2, 9, 9, 9, 3, 4, 3, 0, 2, 2, 3, 7, 7, 0, 3, 0, 4, 7, 8, 9, 0, 3, 4, 8, 5, 9, 3, 5, 7, 0, 0, 7, 6, 4, 7, 7, 3, 6, 4, 0, 7, 0, 5, 0, 0, 1, 1, 2, 2, 2, 7, 8, 5, 4, 5, 9, 2, 7, 5, 2, 3, 3, 2, 4, 9, 8, 9, 2, 9, 8, 7, 6, 1, 7, 3, 9, 2, 8, 4, 2, 0, 0, 2, 5, 2, 5, 4, 8, 1, 1, 7, 4, 4, 0, 9, 1, 6, 7, 2, 5, 5, 7, 5, 9, 7, 5, 4, 4, 2, 5, 4, 6, 0, 6, 2, 7, 4, 6, 4, 5, 8, 8, 3, 3, 1, 7, 0, 4, 9, 8, 7, 4, 6, 7, 6, 4, 2, 8, 5, 9, 9, 4, 5, 3, 2, 1, 3, 9, 9, 5, 3, 1, 2, 8, 1, 9, 5, 8, 5, 5, 8, 0, 1, 3, 0, 0, 2, 4, 7, 8, 7, 5, 1, 1, 8, 7, 8, 9, 2, 8, 4, 8, 6, 0, 7, 9, 5, 4, 3, 0, 0, 0, 0, 1, 3, 3, 6, 7, 3, 1, 8, 8, 3, 3, 4, 7, 9, 6, 4, 3, 8, 8, 0, 4, 0, 5, 3, 3, 4, 3, 7, 6, 2, 6, 3, 1, 7, 7, 4, 8, 6, 4, 6, 6, 5, 8, 5, 3, 2, 5, 3, 6, 6, 8, 0, 3, 1, 3, 1, 7, 9, 3, 5, 7, 1, 0, 4, 8, 5, 5, 3, 8, 7, 2, 5, 6, 8, 5, 2, 0, 5, 9, 4, 3, 8, 7, 4, 6, 9, 4, 7, 9, 6, 7, 8, 4, 5, 2, 9, 1, 5, 0, 4, 2, 7, 5, 0, 4, 5, 6, 7, 1, 7, 8, 2, 1, 3, 9]

**PROBLEMS**

1. Go to Yahoo!Finance (or the source of your choice, such as Bloomberg) and download a daily price series for a particular publicly-traded stock of your choice for a ten-year time period (don’t use Apple), as well as the daily price series on the exchange on which it trades. [5 points]
2. Using R or Python, calculate the log returns of each series as the natural log of the ratio of (price today/price yesterday). Use the reported closing price for this exercise.
3. Using R or Python, generate a histogram of log returns of the stock of your choice.
4. Using R or Python, generate a scatterplot that relates the log returns of your stock of choice to the log returns of the exchange on which it is traded.
5. Finally, using R or Python, fit a linear model to obtain estimates of what finance folks call the “alpha” and the “beta”. Is “alpha” significantly different than zero at a 95% level? Does a 95% confidence level for “beta” include one?
6. Submit all code and results.

**ANSWERS**

3.a. Plotting bivariate linear regression model:

**Microsoft:**

Open High Low Close Volume Adj Close Return

Date

2004-09-20 27.44 27.65 27.33 27.51 51513600 20.07 NaN

2004-09-21 27.45 27.53 27.25 27.26 73874400 19.89 -0.009129

2004-09-22 27.28 27.74 27.07 27.12 68409000 19.79 -0.005149

2004-09-23 27.19 27.39 27.17 27.35 52155800 19.96 0.008445

2004-09-24 27.39 27.46 27.19 27.29 49859800 19.91 -0.002196

2004-09-27 27.17 27.32 27.13 27.19 47813600 19.84 -0.003671

2004-09-28 27.21 27.36 27.04 27.27 62055100 19.90 0.002938

2004-09-29 27.26 27.69 27.23 27.58 61529300 20.12 0.011304

2004-09-30 27.59 27.79 27.52 27.65 71218000 20.18 0.002535

2004-10-01 27.82 28.32 27.78 28.25 66302800 20.61 0.021468

**Nasdaq**

Open High Low Close Volume Adj Close Return

Date

………………………..(shortened to oldest records)

………………………..

2004-09-20 1903.02 1921.50 1900.24 1908.07 1565540000 1908.07 NaN

2004-09-21 1913.13 1925.85 1909.43 1921.18 1531560000 1921.18 0.006847

2004-09-22 1910.23 1910.23 1884.85 1885.71 1588290000 1885.71 -0.018635

2004-09-23 1887.02 1894.67 1883.32 1886.43 1396810000 1886.43 0.000382

2004-09-24 1888.89 1897.42 1879.48 1879.48 1360090000 1879.48 -0.003691

2004-09-27 1871.16 1871.94 1858.88 1859.88 1316790000 1859.88 -0.010483

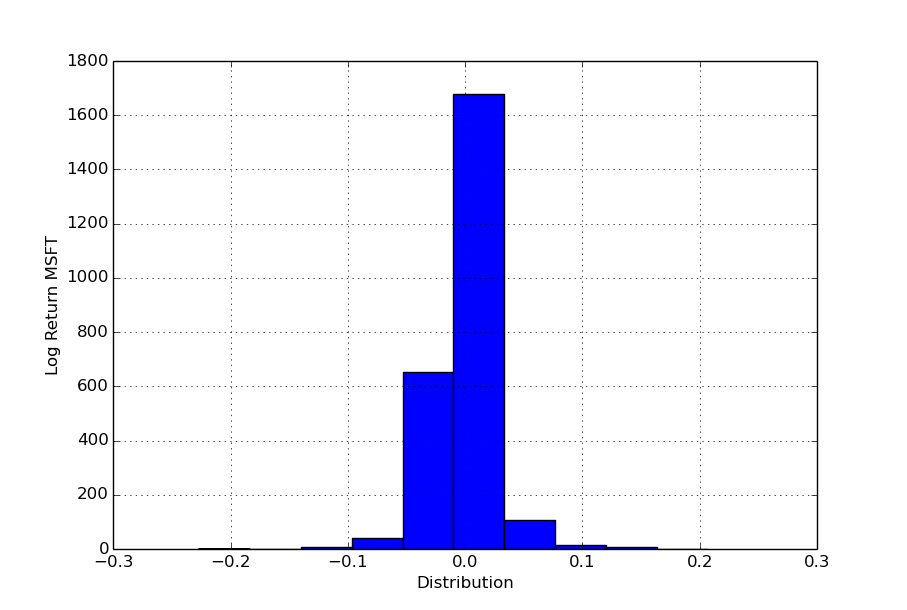
2004-09-28 1865.88 1873.86 1852.59 1869.87 1536980000 1869.87 0.005357

2004-09-29 1870.61 1894.06 1869.95 1893.94 1637280000 1893.94 0.012790

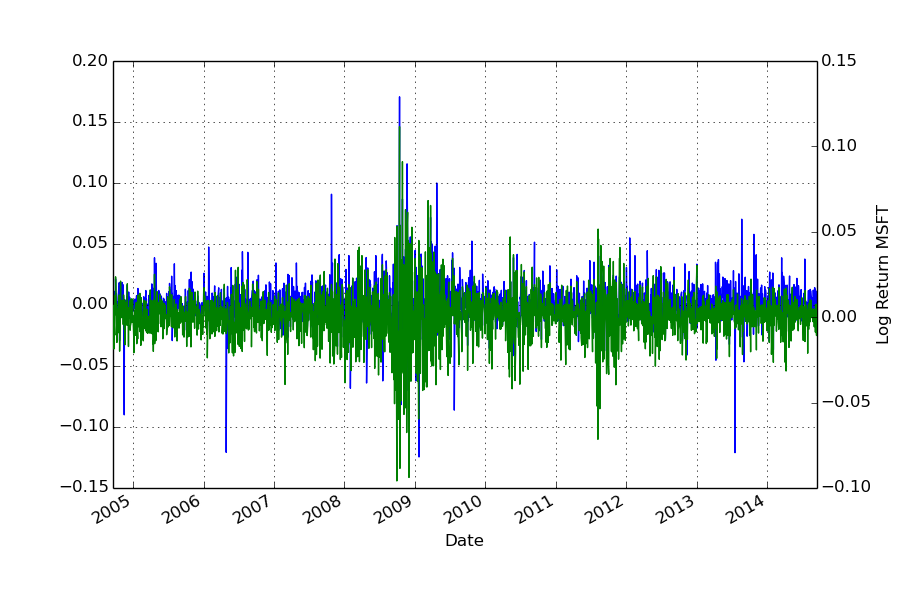
2004-09-30 1892.60 1902.25 1887.68 1896.84 1656620000 1896.84 0.001530

2004-10-01 1909.59 1942.23 1908.57 1942.20 1820300000 1942.20 0.023632

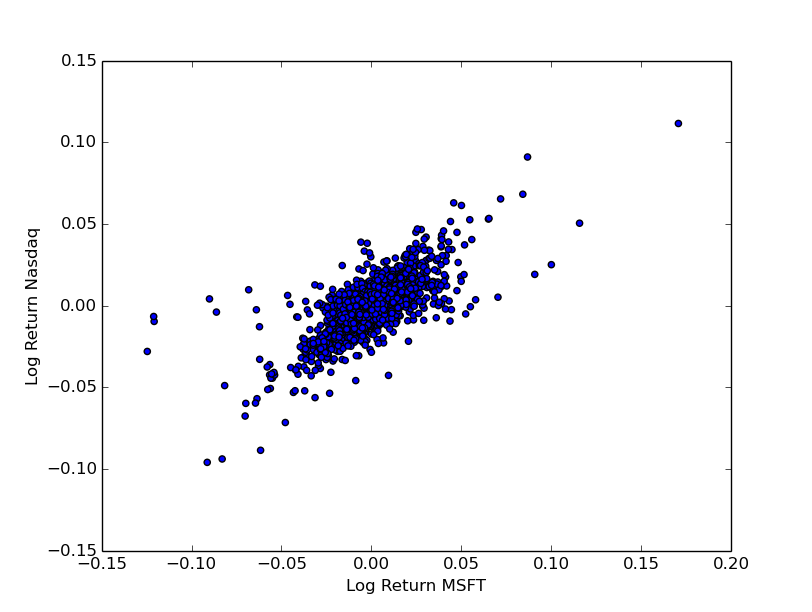
3.b. Histogram of the log returns of MSFT stocks:



3.c. Scatterplot that relates the log returns of MSFT stock to the log returns of the exchange on which it is plotted agains the time :

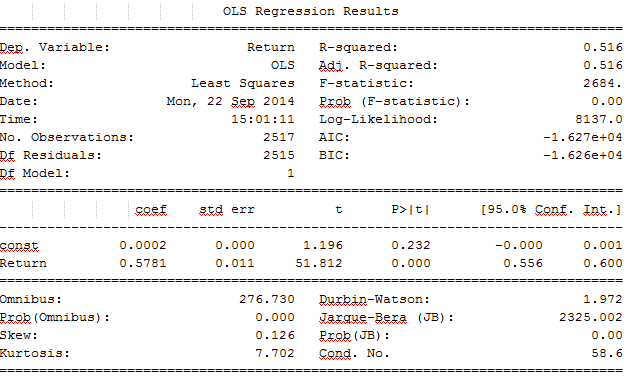


And consequently Microsoft stock value log return VS Nasdaq stock value log return:



3.d. a linear model to obtain “Alpha” and “Beta”:

With the summary of t statistic of both log returns:



Here we could observe that “Beta” is 0.5871 and “Alpha” is 0.011.

3.e source code

##############################################

##############################################

# Applied Data Science GX5004 #

# Assignment 2 #

# Dimas Rinarso Putro | drp354@nyu.edu #

##############################################

**import** argparse**,**csv**,**sys**,** os

**import** numpy **as** np

**from** matplotlib **import** pyplot **as** plt

**import** pylab

**from** pandas**.**io**.**data **import** DataReader

**import** pandas **as** pd

**from** pandas**.**tools**.**plotting **import** scatter\_matrix **as** scatter

**from** datetime **import** date**,**datetime

**from** matplotlib**.**dates **import** date2num

**import** statsmodels**.**api **as** sm

###No.3###

###########################################

#Read data

df\_mic **=** DataReader**(**'MSFT'**,**'yahoo'**,**start**=**'09/18/2004'**,**end**=**'09/18/2014'**)**

df\_nas **=** DataReader**(**'^IXIC'**,**'yahoo'**,**start**=**'09/18/2004'**,**end**=**'09/18/2014'**)**

#calculate log returns

df\_mic**[**'Return'**]** **=** np**.**log**(**df\_mic**[**'Close'**]/**df\_mic**[**'Close'**].**shift**(**1**))**

df\_nas**[**'Return'**]** **=** np**.**log**(**df\_nas**[**'Close'**]/**df\_nas**[**'Close'**].**shift**(**1**))**

#print to screen

**print** df\_mic**[**'Return'**].**describe**()**

**print** df\_nas**[**'Return'**].**describe**()**

#histogram of Microsoft stock

plt**.**figure**();**

df\_mic**[**'Return'**].**diff**().**hist**()**

plt**.**xlabel**(**'Distribution'**)**

plt**.**ylabel**(**'Log Return MSFT'**)**

plt**.**show**()**

#scatterplot of both log returns

df\_mic**[**'Return'**].**plot**()**

df\_nas**[**'Return'**].**plot**(**secondary\_y**=True,** style**=**'g'**)**

plt**.**xlabel**(**'distribution'**)**

plt**.**ylabel**(**'Log Return MSFT'**)**

plt**.**show**()**

##calculate alpha

X**=**df\_mic**.**Return**[**1**:]**

y**=**df\_nas**.**Return**[**1**:]**

X **=** sm**.**add\_constant**(**X**)**

model **=** sm**.**OLS**(**y**,** X**)**

results **=** model**.**fit**()**

**print(**results**.**summary**())**

#scatterplot of both log returns

plt**.**scatter**(**df\_mic**[**'Return'**],**df\_nas**[**'Return'**])**

plt**.**xlabel**(**'Log Return MSFT'**)**

plt**.**ylabel**(**'Log Return Nasdaq'**)**

plt**.**show**()**

**PROBLEMS**

1. Download the file train.dta from the course website. These data are formatted as a Stata dataset. [5 points]
   1. Read this dataset into R or Python. (For R, you may find the “foreign” library of use. For Python, check out Pandas. The goal here is to get you familiar with reading datasets with alternative formats.)
   2. Generate summary statistics for the following variables in the data:
      * d, which is an indicator for whether a particular email is spam
      * x1, which is an attribute of the email
   3. Using least squares, regress d on x1. (For R, check out lm. For Python, check out StatsModels.) Congratulations, you have created a support vector machine that you will use to forecast whether an incoming email with a different attribute is spam.
   4. Suppose you set the threshold that an email is spam if the predicted value exceeds 1.[[1]](#footnote-1) I give you a new email with an attribute value 0.65. Would you classify it as spam or not spam?
   5. I give you another new email with an attribute value of 0.99. Would you classify it as spam or not spam?

**ANSWERS**

1. Read the dataset using pandas (pandas.io.stata) in python (see source code)
2. The summary of both the d and x1 respectively:

d=

count 1000.000000

mean 0.477000

std 0.499721

min 0.000000

25% 0.000000

50% 0.000000

max 1.000000

dtype: float64

x1=

count 1000.000000

mean 0.487376

std 0.283345

min 0.000286

25% 0.249382

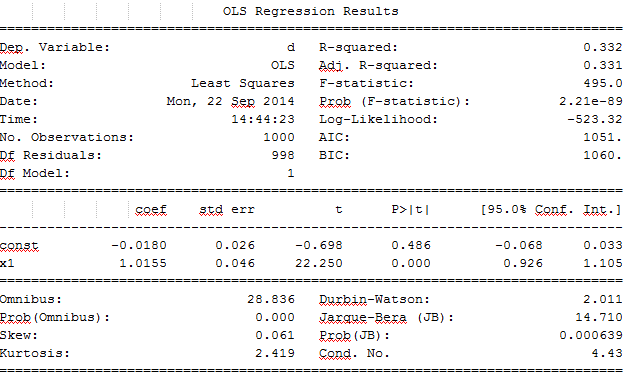
50% 0.484645

75% 0.732494

max 0.999006

dtype: float64

4.c. The summary of the regression d on x1:



4.d. From the summary it is given the coef, that is the estimated coefficient is 1.0155 so if the there is an additional email attribute with the value of 0.65, then the predicted value is Mx which is 1.0155(0.65) which is equal to 0.66. Since it’s smaller than the threshold, it should be classified as **non-spam email**.

4.e. if the there is an additional email attribute with the value of 0.99, then the predicted value is should be classified **as spam email** because at 1.0155(0.99) is 1.0053 which is greater than the threshold.

4.f. source code:

##############################################

##############################################

# Applied Data Science GX5004 #

# Assignment 2 #

# Dimas Rinarso Putro | drp354@nyu.edu #

# No.4 #

##############################################

**import** argparse**,**csv**,**sys**,**os

**import** numpy **as** np

**from** matplotlib **import** pyplot **as** plt

**import** pandas **as** pd

**from** pandas**.**tools**.**plotting **import** scatter\_matrix **as** scatter

**from** pandas**.**io**.**stata **import** read\_stata **as** rd\_stata

**import** statsmodels**.**api **as** sm

###No.4###

##############################################

#Read data

filename **=** 'train.dta'

df **=** rd\_stata**(**filename**)**

#print the summary for each column

**print** 'd='

**print** df**[**'d'**].**describe**()**

**print** 'x1='

**print** df**[**'x1'**].**describe**()**

#sorting

**print** df**.**head**()**

result **=**df**.**sort**([**'x1'**],** ascending**=[**1**])**

X **=** df**.**x1

y **=** df**.**d

#linear regression d on x1

X **=** sm**.**add\_constant**(**X**)**

model **=** sm**.**OLS**(**y**,** X**)**

results **=** model**.**fit**()**

**print(**results**.**summary**())**

#plot sorted column into scatterplot

#for better understanding

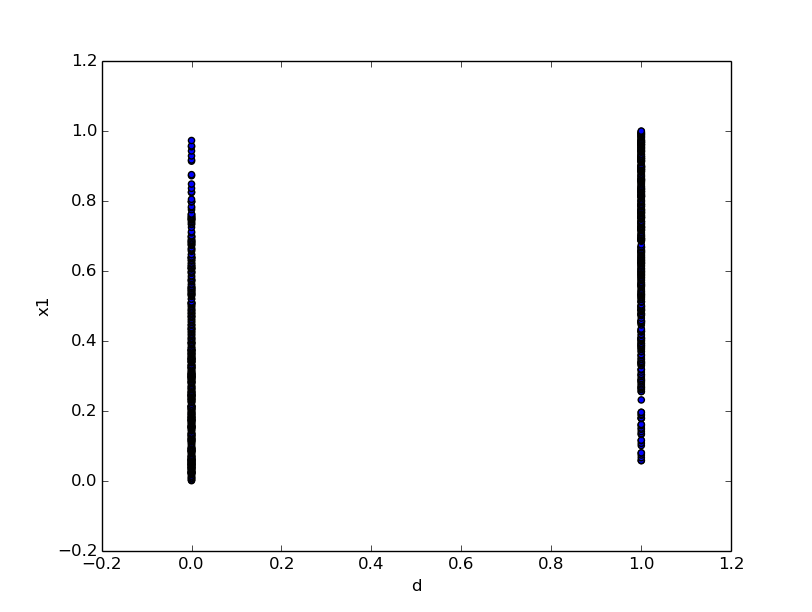
plt**.**scatter**(**result**[**'d'**],**result**[**'x1'**])**

plt**.**xlabel**(**'d'**)**

plt**.**ylabel**(**'x1'**)**

plt**.**show**()**

Output:



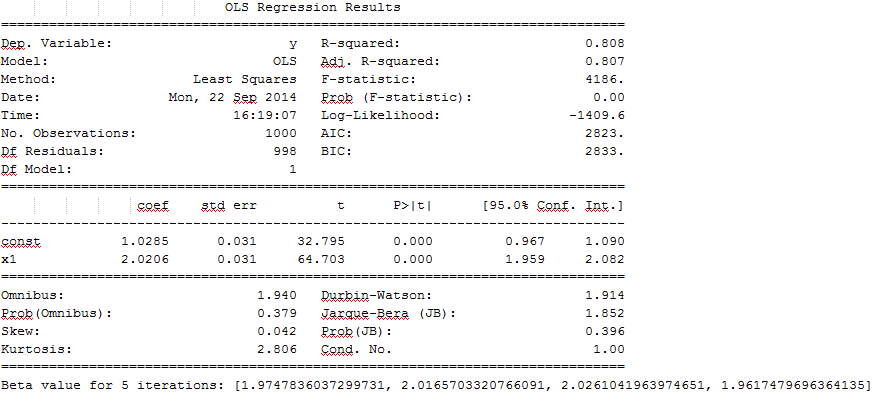
**PROBLEMS**

1. This is a very challenging question, but it addresses several key topics in data analytics. You should work collectively on a solution with the recognition that you may not complete it. The phrase “data generating process” (or DGP) is often used to describe the hypothetical process by which observations arise in the real world. We discussed at some length the bivariate linear regression model, . In this problem, we will work with a specific DGP and evaluate features of . [5 points]
   1. Suppose your DGP is , where .
   2. Using R or Python, write code to generate 1,000 draws for . Use these draws to generate in accordance with the DGP in a.
   3. Using R or Python, write code to estimate the bivariate model, and summarize the findings.
   4. Repeat b. and c. above five times for a new set of random draws for each replication. (This effort is called Monte Carlo simulation.)
   5. Given what you’ve done in d., Suppose you wrote code to repeat b. and c. above 1,000 times, each time recording the estimated value of . What do you think a histogram of these 1,000 replications of the estimate value of would show?
   6. Suppose that you were not interested in the estimate of per se, but instead in some functional transformation, such as the estimate of . What might you do with your 1,000 replications from e. above to inform you about the distribution of the estimate of ?
   7. Submit code and results.

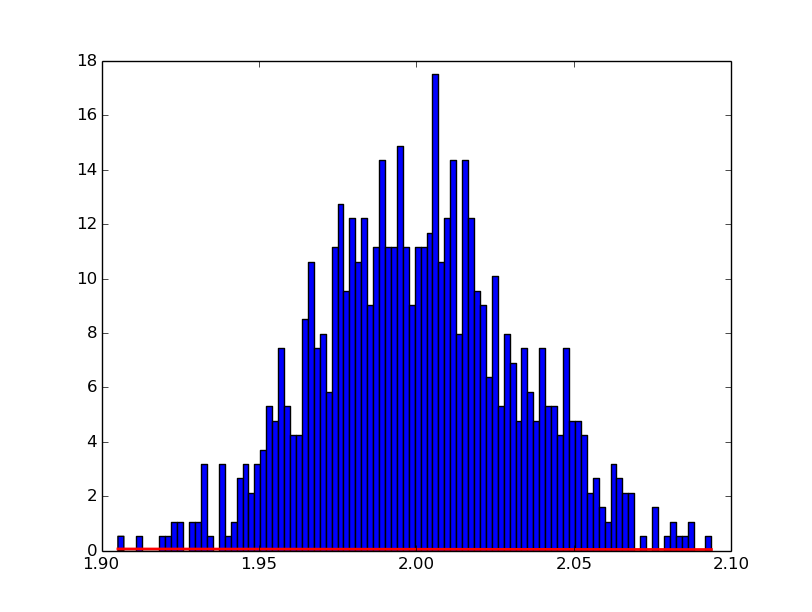
**ANSWERS:**

For no.5a-b please refer to the source code.

c.-d. the summary and the result of the 5 times iterations:

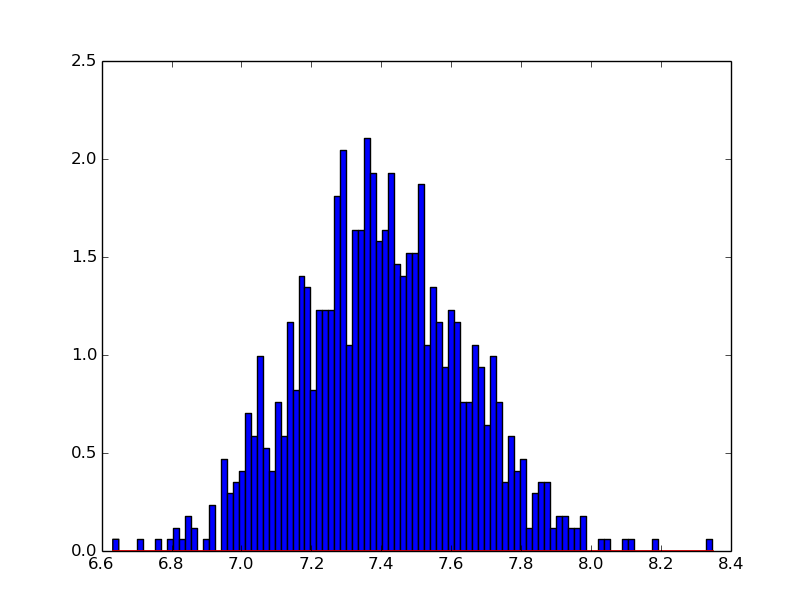


5.e. the histogram of 1000 iterations for Beta estimation:



**Conclusion:** it can be observed that by 1 time iteration the value of beta is 2.02 and by 1000 times iteration and its histogram plot the value of beta can vary from 1.90 to 2.1.

5.f. The histogram of 1,000 iterations of :



5.g. source code and result

##############################################

##############################################

# Applied Data Science GX5004 #

# Assignment 2 #

# Dimas Rinarso Putro | drp354@nyu.edu #

##############################################

**import** argparse**,**csv**,** sys**,** os

**import** numpy **as** np

**from** matplotlib **import** pyplot **as** plt

**import** pylab

**import** statsmodels**.**api **as** sm

###No.5###

###########################################

#variable initialization

mu**,** sigma **=** 0**,** 1

num\_samples **=** 1000

#generate random numbers

x **=** np**.**random**.**normal**(**mu**,** sigma**,** num\_samples**)**

e **=** np**.**random**.**normal**(**mu**,** sigma**,** num\_samples**)**

y **=** 1**+**2**\***x**+**e

#regression

x **=** sm**.**add\_constant**(**x**)**

model **=** sm**.**OLS**(**y**,** x**)**

results **=** model**.**fit**()**

**print(**results**.**summary**())**

###########################################

###########NOW WITH MONTE CARLO############

#---------5 iterations

m **=** **[]**

**for** i **in** xrange **(**0**,**4**):**

#generate random numbers

x **=** np**.**random**.**normal**(**mu**,** sigma**,** num\_samples**)**

e **=** np**.**random**.**normal**(**mu**,** sigma**,** num\_samples**)**

y **=** 1**+**2**\***x**+**e

#regression

regression **=** np**.**polyfit**(**x**,** y**,** 1**)**

m**.**append**(**regression**[**0**])** #adding beta into list

**print** 'Beta value for 5 iterations: '**+**str**(**m**)**

###########################################

###########NOW WITH MONTE CARLO############

#---------1000 iterations

m\_1000 **=** **[]**

**for** i **in** xrange **(**0**,**999**):**

#generate random numbers

x **=** np**.**random**.**normal**(**mu**,** sigma**,** num\_samples**)**

e **=** np**.**random**.**normal**(**mu**,** sigma**,** num\_samples**)**

y **=** 1**+**2**\***x**+**e

#regression

regression **=** np**.**polyfit**(**x**,** y**,** 1**)**

m\_1000**.**append**(**regression**[**0**])** #adding beta into list

#plot histogram

acount**,** bins**,** ignored **=** plt**.**hist**(**m\_1000**,** 100**,** normed**=True)**

plt**.**plot**(**bins**,** 1**/(**sigma **\*** np**.**sqrt**(**2 **\*** np**.**pi**))** **\***

np**.**exp**(** **-** **(**bins **-** mu**)\*\***2 **/** **(**2 **\*** sigma**\*\***2**)** **),**

linewidth**=**2**,** color**=**'r'**)**

plt**.**show**()**

###########################################

###########NOW WITH MONTE CARLO############

#---------1000 iterations on exp(Beta)

m\_1000 **=** **[]**

**for** i **in** xrange **(**0**,**999**):**

#generate random numbers

x **=** np**.**random**.**normal**(**mu**,** sigma**,** num\_samples**)**

e **=** np**.**random**.**normal**(**mu**,** sigma**,** num\_samples**)**

y **=** 1**+**2**\***x**+**e

#regression

regression **=** np**.**polyfit**(**x**,** y**,** 1**)**

m\_1000**.**append**(**np**.**exp**(**regression**[**0**]))** #adding beta into list

#plot histogram

acount**,** bins**,** ignored **=** plt**.**hist**(**m\_1000**,** 100**,** normed**=True)**

plt**.**plot**(**bins**,** 1**/(**sigma **\*** np**.**sqrt**(**2 **\*** np**.**pi**))** **\***

np**.**exp**(** **-** **(**bins **-** mu**)\*\***2 **/** **(**2 **\*** sigma**\*\***2**)** **),**

linewidth**=**2**,** color**=**'r'**)**

plt**.**show**()**

1. [↑](#footnote-ref-1)