Stat 2332 Project

Andrew Goeden

December 2021

1. Read the Final.csv data from D2L to R and Python and denote this data by d1.

```
#Python code
print("-- No Console Output --")
d1 = pd.read_csv(open("final.csv","r"))
```

```
#R code
d1 <- read.csv("final.csv")
```

2. How many observations (number of rows) and Variables (columns) in the d1 data?

```
#Python code
print(d1.shape)
print(d1.columns)
```

```
#R code
nrow(d1)
ncol(d1)
```

```
answer: 17842 rows x 28 columns
```

3. How many variables are numerical/continuous and how many are them are integers/discrete?

```
#Python code
print(d1.info())
```

```
#R code
str(d1)
```

```
answer: 27 numerical & 1 discrete
```

4. Delete ID variable from the d1 data.

```
#Python code
del d1['ID'] # or -> d1 = d1.drop(["ID"], axis=1)
print(d1)
```

```
#R code
d1 <- subset(d1, select = -c(ID))
str(d1)
```

5. Report the number of missing values for the variables MOFB, YOB, and AOR

```
#Python code
print("MOFB missing: {}".format(d1['MOFB'].isnull().sum())) # number of missing values for e
print("YOB missing: {}".format(d1['YOB'].isnull().sum())) # number of missing values for eac
print("AOR missing: {}".format(d1['AOR'].isnull().sum())) # number of missing values for eac

#R code
sapply(d1[c("MOFB", "YOB", "AOR")], function(x) sum(is.na(x)))

answer:
MOFB  YOB  AOR
5507 5507 1817
```

6. Create d2 data from d1 data by selecting variables RMOB, WI, RCA, Religion, Region, AOR, HEL, DOBCMC, DOFBCMC, MTFBI, RW, RH, and RBMI variables.

```
#Python code
d2 = d1[["RMOB", "WI", "RCA", "Religion", "Region", "AOR", "HEL", "DOBCMC", "DOFBCMC", "MTFB
print(d2)

#R code
d2 <- d1[c("RMOB", "WI", "RCA", "Religion", "Region", "AOR", "HEL", "DOBCMC", "DOFBCMC", "MT
str(d2)
```

7. Delete rows that have missing values for any variable in the d2 data and denote this new data by d3.

```
#Python code
d3 = d2.dropna() # deleting rows that has missing values (deleting all missings)
print(d3)

#R code
d3 <- na.omit(d2)
str(d3)
```

8. Find the summary statistics of the d3 data.

```
#Python code
print(d3.describe().transpose())

#R code
summary(d3)
```

```
answer:
      RMOB              WI              RCA            Religion    
 Min.   : 1.000   Min.   :1.000   Min.   :13.00   Min.   :1.000  
 1st Qu.: 3.000   1st Qu.:2.000   1st Qu.:24.00   1st Qu.:1.000  
 Median : 6.000   Median :3.000   Median :31.00   Median :1.000  
 Mean   : 6.437   Mean   :3.129   Mean   :31.84   Mean   :1.121  
 3rd Qu.:10.000   3rd Qu.:4.000   3rd Qu.:39.00   3rd Qu.:1.000  
 Max.   :12.000   Max.   :5.000   Max.   :49.00   Max.   :4.000  

     Region            AOR             HEL            DOBCMC    
 Min.   :1.000   Min.   :11.0   Min.   :0.00   Min.   : 921  
 1st Qu.:2.000   1st Qu.:16.0   1st Qu.:0.00   1st Qu.:1212  
 Median :4.000   Median :17.0   Median :1.00   Median :1273  
 Mean   :3.935   Mean   :17.9   Mean   :1.21   Mean   :1253  
 3rd Qu.:6.000   3rd Qu.:19.0   3rd Qu.:2.00   3rd Qu.:1311  
 Max.   :7.000   Max.   :40.0   Max.   :3.00   Max.   :1344  

    DOFBCMC          MTFBI              RW                RH      
 Min.   : 893   Min.   :  0.00   Min.   : 229.0   Min.   :1044  
 1st Qu.:1091   1st Qu.: 12.00   1st Qu.: 423.0   1st Qu.:1474  
 Median :1188   Median : 22.00   Median : 482.0   Median :1510  
 Mean   :1174   Mean   : 35.44   Mean   : 702.5   Mean   :1701  
 3rd Qu.:1264   3rd Qu.: 38.00   3rd Qu.: 556.0   3rd Qu.:1547  
 Max.   :1344   Max.   :996.00   Max.   :9999.0   Max.   :9999  

      RBMI           AVG          Newreligion   
 Min.   :1245   Min.   : 605.7   Min.   :1.000  
 1st Qu.:1876   1st Qu.: 780.0   1st Qu.:1.000  
 Median :2114   Median : 830.0   Median :1.000  
 Mean   :2343   Mean   : 820.8   Mean   :1.113  
 3rd Qu.:2420   3rd Qu.: 865.0   3rd Qu.:1.000  
 Max.   :9999   Max.   :1217.0   Max.   :2.000  
```

9. Add a new variable in the d3 data by finding the average of DOBCMC, DOFBCMC and MTFBI.

```
#Python code
d3["AVG"] = d3[["DOBCMC","DOFBCMC","MTFBI"]].mean(axis=1)
print(d3)

#R code
d3$AVG <- rowMeans(d3[,c("DOBCMC", "DOFBCMC", "MTFBI")])
```

10. Create a new variable named "Newreligion" by recoding '1' as '1' and rest as '2' from the Religion Variable

```
#Python code
d3["Newreligion"] = d3[["Religion"]]
d3["Newreligion"].loc[d3.Newreligion != 1] = 2
print(d3)
```

```
#R code
d3$Newreligion[d3$Religion == 1] <- 1
d3$Newreligion[d3$Religion != 1] <- 2
```

11. Find the frequency table for the Region variable

```
#Python code
print(pd.crosstab(index=d3.Region,columns=["Region","Count"]))
```

```
#R code
table(d3$Region)
```

```
answer:
   1    2    3    4    5    6    7
1856 2604 2724 2379 2340 2252 1870
```

12. Find the joint frequency table for the variables Region and Religion.

```
#Python code
print(d3.melt(id_vars="Region", value_vars=["Religion"])
         .groupby([pd.Grouper(key='Region'),'value'])
         .size()
         .unstack(fill_value=0))
```

```
#R code
table(d3$Region, d3$Religion)
```

```
answer:
        1    2    3    4
  1  1686  170    0    0
  2  2332  239   31    2
  3  2564  159    0    1
  4  2070  274    1   34
  5  2184  151    0    5
  6  1829  421    0    2
  7  1550  319    0    1
```

13. Find the mean values of AOR variable corresponding to each label of Region variable.

```
#Python code
print(pd.crosstab(index=d3["Region"], columns=d3["AOR"]).mean(axis=1))
```

```
#R code
aggregate(AOR~Region, data = d3, mean)
```

```
answer:
Region
1     64.000000
2     89.793103
3     93.931034
4     82.034483
5     80.689655
6     77.655172
7     64.482759
```

14. Find the variances of AOR variable corresponding to each label of Religion variable.

```
#Python code
print(pd.crosstab(index=d3["Religion"], columns=d3["AOR"]).var(axis=1))
```

```
#R code
aggregate(AOR~Religion, data = d3, var)
```

```
answer:
Religion
1    487257.433498
2      5949.689655
3         3.096059
4         6.399015
```

– Needed for 15-18 (part of Question 19) –

```
#Python code
fig, ((ax1,ax2),(ax3,ax4)) = plt.subplots(2,2)
```

```
#R code
par(mfrow = c(2, 2))
```

15. Draw a boxplot for the MTFBI variable.

```
#Python code
plt.subplot(2,2,1)
plt.boxplot(d3["MTFBI"])
```

```
#R code
labels <- factor(d3$MTFBI)
boxplot(table(labels), main = "Boxplot of MTFBI")
```

16. Draw a histogram for the RCA variable.

```
#Python code
plt.subplot(2,2,2)
plt.hist(d3["RCA"])
```

```
#R code
labels <- factor(d3$RCA)
hist(table(labels), main = "Histogram of RCA")
```

17. Draw a bar chart for the Region variable

```
#Python code
regionCrosstab = pd.crosstab(index=d3["Region"],columns="Count")
regionCrosstab.plot.bar(rot=360,title="Total for Regions", layout=(2,2), ax=ax3)
```

```
#R code
labels <- factor(d3$Region)
barplot(table(labels), main = "Bar graph of Region")
```

18. Draw a pie chart for the Region variable

```
#Python code
pd.crosstab(index=d3["Region"],columns="Count").plot.pie(subplots=True,title="Pie chart of R
```

```
#R code
labels <- factor(d3$Region)
pie(table(labels), main = "Pie chart of Region")
```

19. Put above four figures (question 15 to question 18) in a 2 by 2 grid
20. Split the d3 data by WI variable and denote it by d4

```
#Python code
d4 = d3.groupby(d3['WI'])
```

```
#R code
d4 <- with(d3, split(d3, WI))
length(d4)
```

21. For each split data in d4 write a single loop to find the mean, minimum, maximum, standard deviation of MTFBI.

```
#Python code
printQuestion()
for key in d4.groups:
    group = d4.get_group(key)
    print("WI[\"{}\"]".format(key))
    print("Mean: {} | Min: {} | Max: {} | STD: {} | VAR: {}".format(
        group.mean()["MTFBI"],
        group.min()["MTFBI"],
        group.max()["MTFBI"],
        group.std()["MTFBI"],
        group.var()["MTFBI"]))
    print()
```

```
#R code
result <- matrix(NA, length(d4), 5)
colnames(result) <- c("Mean", "Min", "Max", "Var", "STD")
for(i in 1:length(d4)) { # nolint
    group <- d4[[i]]
    mean <- round(mean(group$MTFBI), 1)
    min <- round(min(group$MTFBI), 1)
    max <- round(max(group$MTFBI), 1)
    var <- round(var(group$MTFBI), 1)
    med <- round(median(group$MTFBI), 1)
    result[i, ] <- c(mean, min, max, var, med)
}
result
```

```
answer:
WI["1"]
Mean: 37.351323119777156 | Min: 0.0 | Max: 996.0 | STD: 88.04623107791824 | VAR: 7752.138807

WI["2"]
Mean: 36.69753497668221 | Min: 0.0 | Max: 996.0 | STD: 90.38456725157154 | VAR: 8169.3699972

WI["3"]
Mean: 33.77448747152619 | Min: 0.0 | Max: 996.0 | STD: 77.86868520372839 | VAR: 6063.5321353

WI["4"]
Mean: 34.51912731619844 | Min: 0.0 | Max: 996.0 | STD: 83.608657185177 | VAR: 6990.407556308

WI["5"]
Mean: 35.13960342979635 | Min: 0.0 | Max: 996.0 | STD: 79.76787260228429 | VAR: 6362.9134994
```

22. Conduct a one sample mean test of hypothesis to check whether MTFBI has a mean of 30 or not.

```
#Python code
print(stats.ttest_1samp(d3.MTFBI, 30))
```

```
#R code
t.test(d3$MTFBI, mu = 30)
```

```
answer:
Ttest_1sampResult(statistic=8.21168162823237, pvalue=2.345894356344911e-16)
```

23. Conduct a normality test of the MTFBI variable

```
#Python code
print(stats.shapiro(d3.MTFBI))
```

```
#R code
shapiro.test(d3[1:5000, "MTFBI"])
```

```
answer:
data:  d3[1:5000, "MTFBI"]
W = 0.22766, p-value < 2.2e-16
```

24. Check the equality of mean for MTFBI variable corresponding to two labels of “Newreligion” variable.

```
#Python code
print(stats.ttest_ind(d3[d3.Newreligion==1].MTFBI,d3[d3.Newreligion==2].MTFBI))
```

```
#R code
t.test(d3$MTFBI~d3$Newreligion)
```

```
answer:
Ttest_indResult(statistic=0.5372104777803612, pvalue=0.5911296942975692)
```

25. Find the correlation matrix of the variables DOBCMC, DOFBCMC, AOR, MTFBI, RW, RH and RBMI from the d3 data.

```
#Python code
correlation= d3[["DOBCMC", "DOFBCMC", "AOR", "MTFBI", "RW", "RH","RBMI"]].corr()
plt.matshow(correlation)
print(correlation)
```

```
#R code
cor(d3[, c("DOBCMC", "DOFBCMC", "AOR", "MTFBI", "RW", "RH", "RBMI")])
```

```
answer:
           DOBCMC   DOFBCMC       AOR     MTFBI        RW        RH      RBMI
DOBCMC   1.000000  0.778187  0.079339 -0.058359 -0.017914 -0.008280 -0.051947
DOFBCMC  0.778187  1.000000  0.198533 -0.060363 -0.008717 -0.001939 -0.032612
AOR      0.079339  0.198533  1.000000  0.112409  0.033758  0.026895  0.061365
MTFBI   -0.058359 -0.060363  0.112409  1.000000  0.002767  0.001257  0.006589
RW      -0.017914 -0.008717  0.033758  0.002767  1.000000  0.972281  0.943352
RH      -0.008280 -0.001939  0.026895  0.001257  0.972281  1.000000  0.936981
RBMI    -0.051947 -0.032612  0.061365  0.006589  0.943352  0.936981  1.000000
```

27. Fit a multiple regression model by considering MTFBI as dependent variable and AOR, RW, Region as independent variables

```
#Python code
y=d3.MTFBI
x=sm.add_constant(d3[["AOR", "RW", "Region"]])
model = OLS(y, x).fit()
print(model.summary())
```

```
#R code
summary(lm(MTFBI~AOR + RW + Religion, data = d3))
```

```
answer:
                            OLS Regression Results
==============================================================================
Dep. Variable:                  MTFBI   R-squared:                       0.013
Model:                            OLS   Adj. R-squared:                  0.013
Method:                 Least Squares   F-statistic:                     69.15
Date:                Mon, 06 Dec 2021   Prob (F-statistic):           1.99e-44
Time:                        22:57:17   Log-Likelihood:                -93602.
No. Observations:               16025   AIC:                         1.872e+05
Df Residuals:                   16021   BIC:                         1.872e+05
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        -17.3356      3.847     -4.506      0.000     -24.876      -9.795
AOR            2.8308      0.198     14.291      0.000       2.443       3.219
RW         -2.056e-05      0.000     -0.044      0.965      -0.001       0.001
Region         0.5356      0.347      1.546      0.122      -0.144       1.215
==============================================================================
Omnibus:                    25541.255   Durbin-Watson:                   1.982
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         10089108.291
Skew:                          10.691   Prob(JB):                         0.00
Kurtosis:                     124.049   Cond. No.                     9.16e+03
==============================================================================
```

28. Simulate one data from the following equation y=50+10X+20U+100N+E. Where X is binomial with n=20, p=.70. U is uniform between 15 and 30 (inclusive). N is normal with mean 0 and standard deviation 5. E is random uniform between -1 and 1. True mean is 640. True variance is 257920.

```
#Python code
trueMean = 640
trueVariance = 257920
def sim(n):
    X=np.random.binomial(20,.7,n)
    U=np.random.uniform(15,30,n)
    N=np.random.normal(0,5,n)
    E=np.random.uniform(-1,1,n)

    y= 50 + 10*X + 20*U + 100*N + E
    y1=pd.DataFrame(y)
    return y1
simulation = sim(100)

#R code
true_mean <- 640
```

```
true_variance <- 257920

simdata <- function(n) {

    x <- rbinom(n, 20, .7)
    u <- runif(n, min = 15, max = 30)
    n1 <- rnorm(n, 0, 5)
    e <- runif(n, min = -1, max = 1)

    y <- 50 + (10 * x) + (20 * u) + (100 * n1) + e
    y
}
simdata(100)
```

29 and 30. Repeat the procedure 100 times and check the true mean with the simulated mean check the true variance with the simulated variance.

```
#Python code
firstTest = pd.concat([sim(1000) for i in range(100)])
testMean= np.mean(firstTest)
print(abs(testMean-trueMean))

testVariance = np.var(firstTest)
print(abs(testVariance-trueVariance))
```

```
#R code
result_function <- function(n) {
    x <- simdata(n)
    mean.x <- mean(x)
    var.x <- var(x)
    mv <- c(mean.x, var.x)
    mv
}

a <- replicate(100, result_function(1000))

sim_mean <- mean(a[1, ])
sim_var <- var(a[2, ])

abs(sim_mean - true_mean)
abs(sim_var - true_variance)
```

```
answer:
mean: 0.03821546
variance: 110145184
```

31 and 32. Repeat the procedure 500 times and check the true mean with the simulated mean and check the true variance with the simulated variance.

```
#Python code
secondTest=pd.concat([sim(1000) for i in range(500)])
secondTestMean=np.mean(secondTest)
print(abs(secondTestMean-trueMean))

secondTestVariance=np.var(secondTest)
print(abs(secondTestVariance-trueVariance))

#R code
result_function <- function(n) {
    x <- simdata(n)
    mean.x <- mean(x)
    var.x <- var(x)
    mv <- c(mean.x, var.x)
    mv
}

b <- replicate(500, result_function(1000))

sim_mean <- mean(b[1, ])
sim_var <- var(b[2, ])

abs(sim_mean - true_mean)
abs(sim_var - true_variance)

answer:
mean: 0.3646633
variance: 151317263
```

33. For five values of x=1:5, y=2:6, and z=3:7, compute 5 values for

$$f(x) = (e^x - log(z^2))/((5 + y))$$

```
#Python code
for x in range(1,6):
    y = x + 1
    z = x + 2
    dividend = (np.e ** x)-np.log10(z**z)
    divisor = (5+y)
    result = dividend / divisor
    print(result)

#R code
for (x in 1:5) {
    y <- x + 1
    z <- y + 1
```

```
    result <-  (exp(1)^x - log10(z^z)) / (5 + y)
    print(result)
}
```

```
answer:
0.1838454377571511
0.62260201670235
1.8434096557230635
4.992924253084237
12.954315711134251
```

34. Solve the following system of linear equations: 70x+100y+40z=900; 120x+450y+340z=1000; 230x+230y+1230z=3000

```
#Python code
a = np.array([[70,100,40],[120,450,340],[230,230,1230]])
b = np.array([900,1000,3000])
print(np.linalg.solve(a,b))
```

```
#R code
x <- matrix(c(70, 100, 40,
              120, 450, 340,
              230, 230, 1230), 3, 3, byrow = TRUE)
y <- c(900, 1000, 1230)
sol <- solve(x, y)
print(sol)
```

```
answer:
[15.53852758 -1.8270015  -0.12491951]
```

35. Find the inverse of the following matrix A

```
#Python code
A = np.array([[20,30,30],[20,80,120],[40,90,360]])
print(np.linalg.inv(A))
```

```
#R code
A <- matrix(c(20, 30, 30,
              20, 80, 120,
              40, 90, 360), 3, 3, byrow = TRUE)
solve(A)
```

```
answer:
[[ 0.07317073 -0.03292683  0.00487805]
 [-0.0097561   0.02439024 -0.00731707]
 [-0.00569106 -0.00243902  0.00406504]]
```

36. Suppose b. Then find:

$$(A^{'A)^{(-1)A'b}}$$

```
#Python code
b = np.array([10,20,30])
result = np.invert(np.matrix.transpose(A) * A) * np.matrix.transpose(A) * b
print(result)

#R code
b <- c(10, 20, 30)
solve(t(A) * A) * t(A) * b
```

```
answer:
Question #36
[[     -80200     -240400    -1441200]
 [    -180300   -10241600   -29162700]
 [    -360300   -25922400 -1399690800]]
```

37. Draw the graph for the function

$$f(x) = e^x/x!$$

2x15

```
#Python code
x = range(2,16)
y = []
for num in x:
    y.append((np.e**num)/(factorial(num)))

plt.figure()
plt.plot(x,y)

#R code
curve((exp(1)^x) / factorial(x), from = 2, to = 15)
```

38. Draw the graph for the step functions Consider the continuous function

```
#Python code
x = np.linspace(-1000, 1000)
y = []
for num in x:
    if (num < 0):
        y.append((2*(num**2)) + np.e**num + 3)
    elif (num >= 0 or num < 10):
        y.append((9*num) + np.log10(20))
    else:
        y.append((7*(num**2)) + (5*num) - 17)

fig,ax= plt.subplots()
ax = sns.lineplot(x=x, y=y)
```

39. Find the areas of 10 circles, which have radii 10:19. The Area of a circle is given

$$r^2$$

```
#Python code
for x in range(10,20):
    print("Circle radius {} with area {}".format(x,np.pi * x**2))

#R code
radii <- 10:19
print(pi * radii^2)

answer:
Circle radius 10 with area 314.1592653589793
Circle radius 11 with area 380.132711084365
Circle radius 12 with area 452.3893421169302
Circle radius 13 with area 530.929158456675
Circle radius 14 with area 615.7521601035994
Circle radius 15 with area 706.8583470577034
Circle radius 16 with area 804.247719318987
Circle radius 17 with area 907.9202768874502
Circle radius 18 with area 1017.8760197630929
Circle radius 19 with area 1134.1149479459152
```

40. Find

$$\sum_{x=2}^{10000}(1/log(x))$$

```
#Python code
result = 0
for x in range(2, 10001):
    result += (1/np.log10(x))
print(result)

#R code
x <- 2:10000
fx <- 1 / log10(x)
print(sum(fx))

answer:
2868.901899946457
```

41. Compute

$$\sum_{i=1}^{30}\sum_{j=1}^{10}((i^{10})/(3+j))$$

```
#Python code
result = 0
for i in range(1, 31):
    for j in range(1, 11):
        result += (i**10) / (3+j)
print(result)
```

```
#R code
result <- 0
for (i in 1:30) {
    for (j in 1:10) {
        result <- result + ((i^10) / (3 + j))
    }
}
print(result)
```

```
answer:
2588621692824886.5
```

42. Compute the integral

$$\int_0^{\infty} x^{15} e^{(-40x)} dx$$

```
#Python code
x = sy.Symbol("x")
result = sy.integrate(((x**15) * (np.e**(-40 * x))), (x,0,np.Infinity))
print(result)
```

```
#R code
integrate(function(x) (x^15 * exp(1)^(-40 * x)), lower = 0, upper = Inf)
```

```
answer:
3.04466664791108e-14
```

43. Compute the integral

$$\int_0^{1} x^{150}(1-x)^{30} dx$$

```
#Python code
x = sy.Symbol("x")
result = sy.integrate(((x**150) * (1 - 30)**30), (x,0,1))
print(result)
```

```
#R code
integrate(function(x) (x^150 * (1 - x)^30), lower = 0, upper = 1)
```

```
answer:
74462898441675122902293018227199467668020601/151
```

44. For five values of x=1:5, y=2:6, and z=3:7, compute 5 values for

$$f(x) = (e^x - log(z^2))/((5+y)).$$

```
#Python code
for x in range(1,6):
    y = x + 1
    z = x + 2
    dividend = (np.e ** x)-np.log10(z**z)
    divisor = (5+y)
    result = dividend / divisor
    print(result)

#R code
for (x in 1:5) {
    y <- x + 1
    z <- y + 1

    result <-  (exp(1)^x - log10(z^z)) / (5 + y)
    print(result)
}

answer:
0.1838454377571511
0.62260201670235
1.8434096557230635
4.992924253084237
12.954315711134251
```

45. Solve the equation

$$x^2 - 33x + 1 = 0$$

```
#Python code
x = sy.Symbol("x")
print(sy.solve((x**2) - (33 * x) + 1), x)

#R code
quadratic <- function(a, b, c) {
  result <- c((-b + sqrt(b^2 - 4 * a * c)) / (2 * a),
              (-b - sqrt(b^2 - 4 * a * c)) / (2 * a))
  result
}
print(quadratic(1, -33, 1))

answer:
x = 32.96966909, 0.03033091
```

47. If 40 is invested today for 50 years with interest rate .10, the find the total amount of money in 50 years. p=40, t=50, and r=.10.

```
#Python code
p = 40
t = 50
r = 0.10
print(p * (1 + r)**t)
```

```
#R code
p <- 40
t <- 50
r <- 0.10
print(p * (1 + r)^t)
```

```
answer:
4695.634115187831
```

48. Fit a simple regression model by using MTFBI as dependent variable and AOR as independent variable.

```
#Python code
model = sm.OLS(d3.MTFBI, d3.AOR).fit()
model.predict(d3.AOR)
print(model.summary())
```

```
answer:
                                 OLS Regression Results
=======================================================================================
Dep. Variable:                  MTFBI   R-squared (uncentered):                   0.161
Model:                            OLS   Adj. R-squared (uncentered):              0.161
Method:                 Least Squares   F-statistic:                              3085.
Date:                Mon, 06 Dec 2021   Prob (F-statistic):                        0.00
Time:                        22:57:20   Log-Likelihood:                         -93612.
No. Observations:               16025   AIC:                                  1.872e+05
Df Residuals:                   16024   BIC:                                  1.872e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
AOR            2.0079      0.036     55.545      0.000       1.937       2.079
==============================================================================
Omnibus:                    25449.183   Durbin-Watson:                   1.981
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          9881485.045
Skew:                          10.619   Prob(JB):                         0.00
Kurtosis:                     122.783   Cond. No.                         1.00
==============================================================================
```

49. Check whether AOR and MTFBI are correlated or not.

```
#Python code
print(stats.pearsonr(d3.AOR, d3.MTFBI))
```

```
answer:
(0.11240865360850757, 3.1805625139858684e-46)
```

50. Check whether variance of AOR is 10 or not.

```
#Python code
dbp_variance = round(d3["AOR"].var(),2)
x_sq_stat, pval, dof = chi_sq_test_for_variance(d3["AOR"],h0=10)
print(dbp_variance)
```

```
answer:
11.05
```