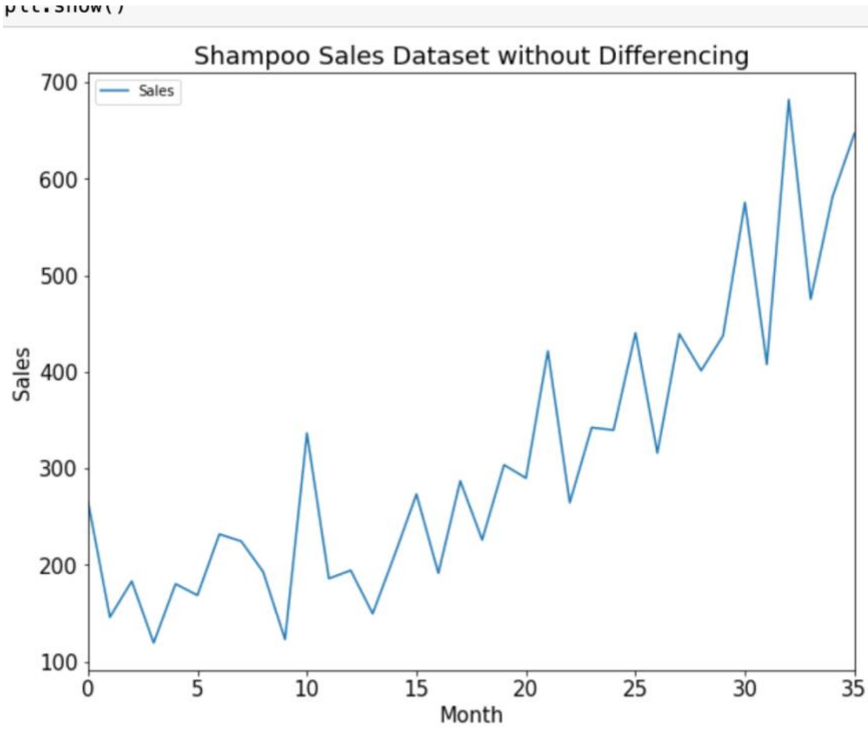
Fernando Zambrano
Multivariate Modeling
Dr. Jafari
5 February 2020

Homework 2

1. Load data. Plot data with proper labels and titles. What behaviors are seen in the data?



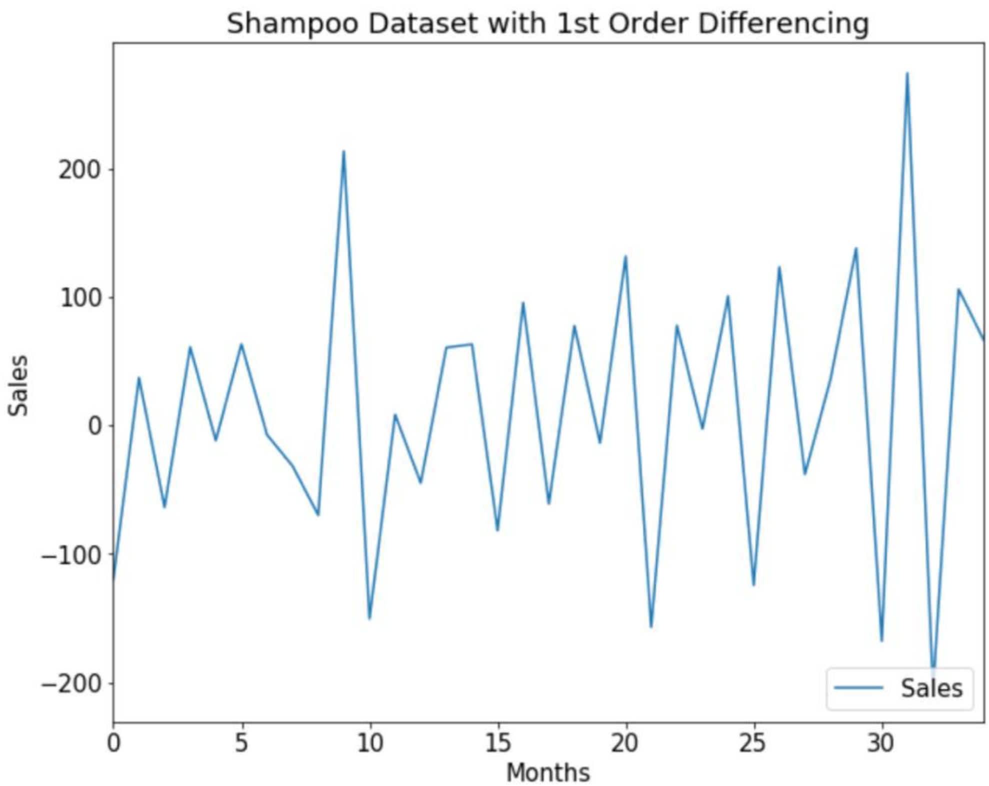
There is definitely an upwards positive trend. There might be some weak signs of seasonality. There is no obvious signs cyclical behavior. However, based on the decomposition in step 5, the data is definitively seasonal.

2. Is the data non-stationary? Perform ADF test to justify answer.

Yes, the data is non-stationary. The ADF test demonstrates that the data is non-stationary because the p-value, 1.00, is much larger than the critical value threshold of 0.05. This means that there is not enough evidence to reject the null hypothesis that the data is non-stationary in favor of the alternative, that it is stationary.

```
ADF Statistic: 3.060142
p-value: 1.000000
Critical Values:
	1%: -3.723863
	5%: -2.986489
	10%: -2.632800
```

3. Write a Python code that will detrend the data by first order difference. Plot the dataset.



4. Is the detrended dataset stationary? If not, try a logarithmic transformation and check if the data becomes stationary.

The detrended dataset does look stationary at this point. There might be a bit a varying variance. However once applying an ADF test on the differenced data, the results show that the data is stationary. The p-value, 0.000 is below the critical value threshold of 0.05. This means that there is enough evidence to reject the null hypothesis that the data is non-stationary in favor of the alternative, that it is stationary.

```
ADF Statistic: -7.249074
p-value: 0.000000
Critical Values:
	1%: -3.646135
	5%: -2.954127
	10%: -2.615968
```

5. Plot decomposed additive and multiplicative models of the raw data.
    a. Additive model

    b. Multiplicative model

The additive model has residuals that hover around 0, while the multiplicative model has all of its residuals around 1. The additive model should be used since it has less error than the multiplicative.

Phase II

1. Create random variables X and Y.

```
X = np.random.randn(N)
print(X)
```

```
[-0.56166447 -0.85728595  0.93963947 ... -0.43843385 -1.22960919
 -0.67425493]
```

```
Y = np.random.randn(N)
print(Y)
```

```
[ 2.38155571  0.62563904 -1.02650381 ...  0.10782283  1.56889438
 -1.23317434]
```

2. Perform ADF test on X and Y to test for stationarity. Are they stationary? Justify.

```
x_adf = ADF_Cal(X)
```

```
ADF Statistic: -100.257445
p-value: 0.000000
Critical Values:
	1%: -3.431004
	5%: -2.861829
	10%: -2.566924
```

```
y_adf = ADF_Cal(Y)
```

```
ADF Statistic: -72.089758
p-value: 0.000000
Critical Values:
	1%: -3.431004
	5%: -2.861829
	10%: -2.566924
```

Based upon the evidence provided by the ADF test, both X and Y are stationary datasets. This is because their p-values, 0.0000, are below the critical value threshold of 0.05. This means that there is enough evidence to reject the null hypothesis that the data is non-stationary in favor of the alternative, that it is stationary.

3. Create two more random variables G and Z. G = X, Z = -X

```
#3
# Create two more random variables G and Z
# G = X and Z = -X

G = X
Z = -X
```

4. Using the function from Lab 2, find the correlation between (X,Y), (X,Z), and (X,G).

```
r_xy = correlation_coefficent_cal(X,Y,"X","Y")
```

```
mean of X: 0.010467831151935458
mean of Y: -0.0012346955336879134
Cross variance: 36.810801051302974
standard devation of X: 100.02022416767
standard devation of Y: 99.99814815471808
The correlation coefficent between X and Y is: 0.0036804039432365224
```

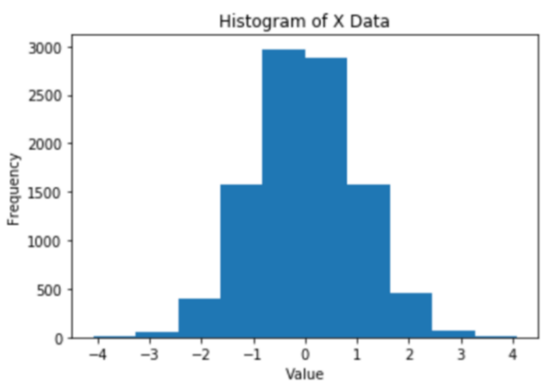
```
r_xz = correlation_coefficent_cal(X,Z,"X","Z")
```

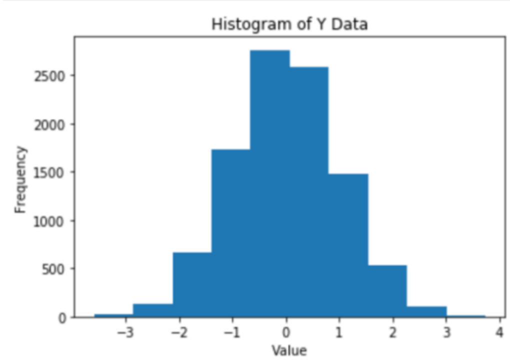
```
mean of X: 0.010467831151935458
mean of Z: -0.010467831151935458
Cross variance: -10004.045242550957
standard devation of X: 100.02022416767
standard devation of Z: 100.02022416767
The correlation coefficent between X and Z is: -1.0
```

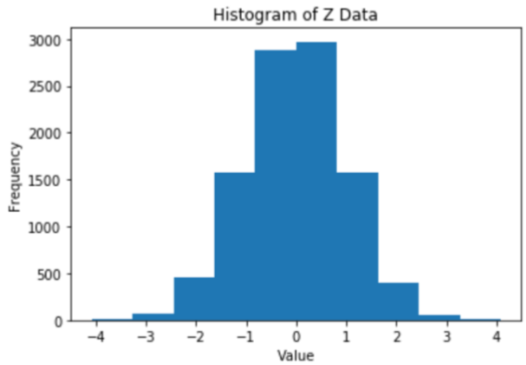
```
r_xg = correlation_coefficent_cal(X,G,"X","G")
```

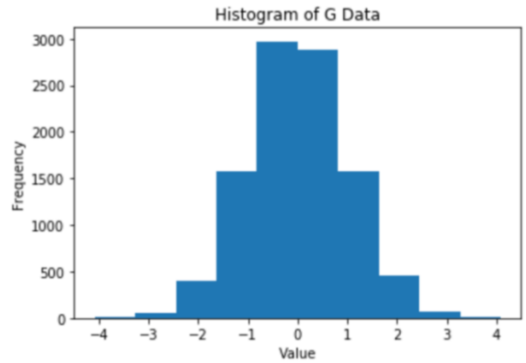
```
mean of X: 0.010467831151935458
mean of G: 0.010467831151935458
Cross variance: 10004.045242550957
standard devation of X: 100.02022416767
standard devation of G: 100.02022416767
The correlation coefficent between X and G is: 1.0
```

5. Graph the histogram of X, Y, G, and Z. Appropriate labels and title.

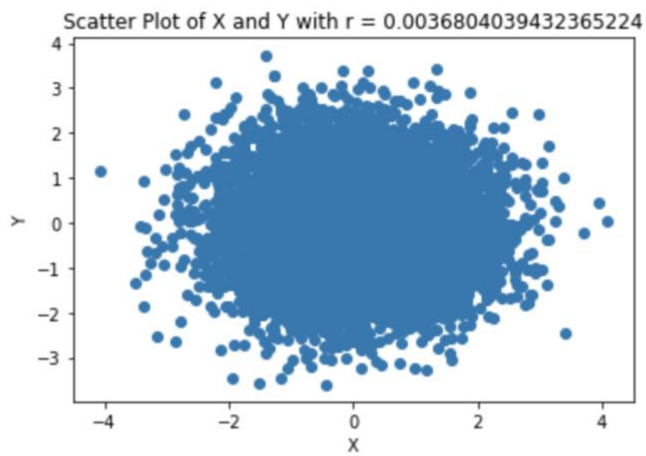




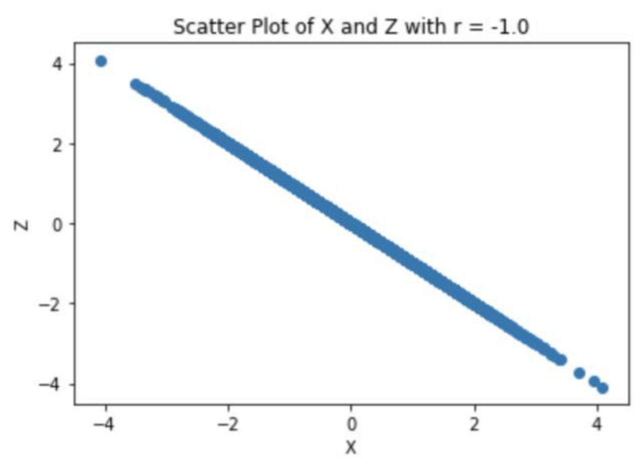




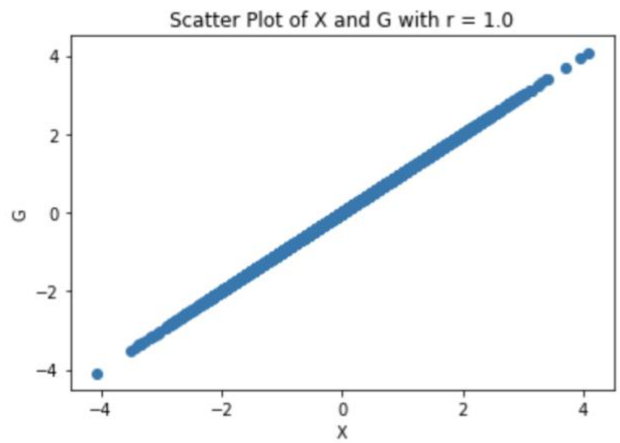
6. Make scatter plots between (X,Y)



7. Make a scatter plot between (X,Z).



8. Make a scatter plot between (X,G).



9. Calculate the correlation between X,Y,Z

```
# 9
# Calcualte the correlation coefficent between x,y,z
r_xy = correlation_coefficent_cal(X,Y,"X","Y")
```

```
mean of X: 0.010467831151935458
mean of Y: -0.0012346955336879134
Cross variance: 36.810801051302974
standard devation of X: 100.02022416767
standard devation of Y: 99.99814815471808
The correlation coefficent between X and Y is: 0.0036804039432365224
```

```
r_xz = correlation_coefficent_cal(X,Z,"X","Z")
```

```
mean of X: 0.010467831151935458
mean of Z: -0.010467831151935458
Cross variance: -10004.045242550957
standard devation of X: 100.02022416767
standard devation of Z: 100.02022416767
The correlation coefficent between X and Z is: -1.0
```

```
r_xg = correlation_coefficent_cal(X,G,"X","G")
```

```
mean of X: 0.010467831151935458
mean of G: 0.010467831151935458
Cross variance: 10004.045242550957
standard devation of X: 100.02022416767
standard devation of G: 100.02022416767
The correlation coefficent between X and G is: 1.0
```

Appendix:

```
import matplotlib.pyplot as plt
import pandas as pd
```

```python
import numpy as np
from statsmodels.tsa.stattools import adfuller
import os
import statistics
from random import randrange
from matplotlib import pyplot
from statsmodels.tsa.seasonal import seasonal_decompose

# ADF test function
def ADF_Cal(x):
    result = adfuller(x)
    print("ADF Statistic: %f" %result[0])
    print("p-value: %f" %result[1])
    print("Critical Values:")
    for key, value in result[4].items():
        print('\t%s: %3f' % (key,value))

# 1
# load shampoo data
os.listdir()
df = pd.read_csv("shampoo.csv")

# Plot Shampoo data
ax = df.plot(kind='line',figsize=(10,8), fontsize=15)
plt.legend(loc='upper left', fontsize=10)
ax.set_xlabel('Month', fontsize=15)
ax.set_ylabel('Sales', fontsize=15)
ax.set_title('Shampoo Sales Dataset with Differencing',fontsize=18)
plt.show()

# 2
# Is the data stationary or non-stationary
# Perform an ADF test on the data

shampoo_adf = ADF_Cal(df.Sales)

# 3
```

```python
# If the answer is yes, write a python code that will detrend it
# Detrend by the 1st differencing technique
# Plot the differenced data

# Convert shampoo sales into a numpy array
sales_diff_arr = np.array(df.Sales)
print("length of input",len(sales_diff_arr))
# apply first order differncing on the passenger array
sales_diff = np.diff(df.Sales)
print("length of differnce",len(sales_diff))
print("Input array  : ", sales_diff_arr)
print("First order difference  : ", sales_diff)

# Convert first order differencing array into a pandas dataframe
df_sales = pd.DataFrame(sales_diff)
# Rename columns
df_sales = df_sales.rename(columns={0: "Sales"})

# Plot the first order difference of passenger data
ax = df_sales.plot(kind='line',figsize=(10,8), fontsize=15)
plt.legend(loc='lower right', fontsize=15)
ax.set_xlabel('Months', fontsize=15)
ax.set_ylabel('Sales', fontsize=15)
ax.set_title('Shampoo Dataset with 1st Order Differencing',fontsize=18)
plt.show()

# 4
# Is the detrended dataset stationary?
# If the answer is no, try a logarithmic transformation
# Check if the new transformed dataset is stationary with an ADF test

# Visually the dataset does not look stationary because there is still some degree of of varying variance through time
sdiff = ADF_Cal(df_sales.Sales)
# Apply a logorithimic transformation on original data

df["LogSales"] = np.log(df.Sales)
slog_adf = ADF_Cal(df.LogSales)
```

```python
# Apply log transformation and differencing
log_diff = np.diff(df.LogSales)

# Convert first order differencing array into a pandas dataframe
log_diff = pd.DataFrame(log_diff)

# Rename columns
log_diff = log_diff.rename(columns={0: "Sales"})
log_adf = ADF_Cal(log_diff.Sales)

# 5
# Decompose the dataset into trend, seasonal, and residuals
# Use both additive and Multiplicative models
# Write down observation on which model should be used

# Decompose with Additive Model
result = seasonal_decompose(df.Sales, model ='additive', period =1)
result.plot()
pyplot.show()

# Decompose with Additive Model
result = seasonal_decompose(df.Sales, model ='multiplicative', period =1)
result.plot()
pyplot.show()

# PHASE II

# 1
# create two random variables X and Y with normal distribution zero mean u  = 0
# Standard deviation = 1
# N = 10,000

s = 0
u = 1
N = 10000
```

```
X = np.random.randn(N)
print(X)

Y = np.random.randn(N)
print(Y)

# 2
# Using ADF test to check if X and Y are stationary data sets
# Show results

x_adf = ADF_Cal(X)

y_adf = ADF_Cal(Y)

#3
# Create two more random variables G and Z
# G = X and Z = -X

G = X
Z = -X

# 4
# Use correlation_coefficent function from LAB2
# Calcualte (X,Y), (X,Z), (X,G)

# Create a function that calculates the correlation coefficent of x and y variables
def correlation_coefficent_cal(dat1,dat2,x,y):
    # calculate cross_variance between x and y
    # convert list data into numpy arrays
    dat1_mean = np.array(dat1).mean()
    print("mean of " + str(x) +":",dat1_mean)
    dat2_mean = np.array(dat2).mean()
    print("mean of " + str(y) + ":",dat2_mean)
    cross_v = sum((dat1 - dat1_mean)*(dat2-dat2_mean))
    print("Cross variance:", cross_v)
    dat1_sd = np.sqrt(sum(np.square(dat1 - dat1_mean)))
    print("standard devation of " + str(x) + ":",dat1_sd)
```

```python
    dat2_sd = np.sqrt(sum(np.square(dat2 - dat2_mean)))
    print("standard devation of "+ str(y) + ":",dat2_sd)
    r = cross_v/(dat1_sd * dat2_sd)
    print("The correlation coefficent between " + str(x) + " and " + str(y) + " is:",r)
    return r


r_xy = correlation_coefficent_cal(X,Y,"X","Y")
r_xz = correlation_coefficent_cal(X,Z,"X","Z")
r_xg = correlation_coefficent_cal(X,G,"X","G")

# 5
# Plot the histogram for X, Y, Z, G with x and y labels

# X histogram

ax = plt.hist(X)
plt.title("Histogram of X Data")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()

# Y histogram

ax = plt.hist(Y)
plt.title("Histogram of Y Data")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()

# Z histogram
ax = plt.hist(Z)
plt.title("Histogram of Z Data")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```

```python
# G histogram
ax = plt.hist(G)
plt.title("Histogram of G Data")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()

# 6
# Make scatter plots for X and Y

def scatter_plt_dft(X,Y,x,y,r):
    ax = plt.scatter(X,Y)
    plt.xlabel(str(x))
    plt.ylabel(str(y))
    plt.title("Scatter Plot of " + str(x) + " and " + str(y) + " with r = {}".format(r))
    plt.show()
    return ax

scat_xy = scatter_plt_dft(X,Y,"X","Y",r_xy)
plt.show()

# 7
# Make a scatter plot of X,Z

scat_xz = scatter_plt_dft(X,Z,"X","Z",r_xz)
plt.show()

# 8
# Make a scatter plot of X,G

scat_xg = scatter_plt_dft(X,G,"X","G",r_xg)
plt.show()

# 9
# Calcualte the correlation coefficent between x,y,z
r_xy = correlation_coefficent_cal(X,Y,"X","Y")
```

```
r_xz = correlation_coefficent_cal(X,Z,"X","Z")

r_xg = correlation_coefficent_cal(X,G,"X","G")
```