Senior Project - Exploratory Data Analysis Using Python

April 29, 2018

1 e-Commerce Exploratory Data Analysis Using Python

1.0.1 Faculty Advisor: Dr. Patrick King

Date: Tuesday, April 3rd, 2018 This is a project of doing an exploratory data analysis (EDA) in Python Jupyter Notebook. This project is part II of the first phase, which is a comparison of the EDA process using R Markdown. We will run a parallel analysis and compare the two softwares while providing essential background information on the advantages and capabilities of the two softwares.

First, let's import the packages needed to create the plots we will use to describe our data as well as making hypptheses. We start with the **pandas**, **numpy**, and **matplotib** packages which are very useful for data exploration.

```
In [40]: # import python packages
         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         from pandas import ExcelWriter
         from pandas import ExcelFile
```

We will use the *read_excel* module from **pandas** package to import the Excel spreadsheet containing the Zappos cutomer transactions. We can take a look at the columns and the type of data each holds. We can also identify the number of missing values for each columns as well as their range if they are type *int*.

```
In [41]: mydata = pd.read_excel('C:/Users/Dorcas/OneDrive - University of Houston Downtown/Sen
         print("Column headings: ")
         # List all column names
         print(mydata.columns)

Column headings:
Index(['day', 'site', 'new_customer', 'platform', 'visits',
       'distinct_sessions', 'orders', 'gross_sales', 'bounces', 'add_to_cart',
       'product_page_views', 'search_page_views', 'conversion_rate',
       'bounce_rate', 'add_to_cart_rate'],
      dtype='object')
```

```
In [42]: mydata.head(25)
```

```
Out[42]:
           day      site  new_customer    platform  visits  distinct_sessions  \
0   2013-01-01      Acme           1.0     Android      24                 16
1   2013-01-01      Acme           1.0  BlackBerry       0                  0
2   2013-01-01    Sortly           1.0        iPad       0                  0
3   2013-01-01      Acme           1.0     Windows     922                520
4   2013-01-01     Botly           1.0     Android      11                 10
5   2013-01-01      Acme           1.0   Macintosh     384                214
6   2013-01-01    Sortly           1.0     Android      14                 10
7   2013-01-01    Sortly           1.0     Windows       1                  0
8   2013-01-01      Acme           0.0       Linux      41                 27
9   2013-01-01      Acme           0.0      iPhone     448                368
10  2013-01-01  Widgetry           1.0      iPhone      15                 14
11  2013-01-01      Acme           NaN     Windows   58192              46312
12  2013-01-01    Sortly           1.0       Other       0                  0
13  2013-01-01   Tabular           1.0        iPad      22                 21
14  2013-01-01    Sortly           1.0   Macintosh       0                  0
15  2013-01-01      Acme           NaN     Android    4942               4290
16  2013-01-01      Acme           NaN  BlackBerry     111                 94
17  2013-01-01  Pinnacle           NaN       Linux       7                  4
18  2013-01-01  Pinnacle           0.0   Macintosh      56                 45
19  2013-01-01     Botly           0.0     Android      20                 19
20  2013-01-01  Widgetry           0.0      iPhone     113                109
21  2013-01-01  Pinnacle           NaN  BlackBerry       4                  2
22  2013-01-01      Acme           NaN       Linux     984                886
23  2013-01-01  Pinnacle           NaN      iPhone     282                239
24  2013-01-01      Acme           NaN   SymbianOS       4                  3

    orders  gross_sales  bounces  add_to_cart  product_page_views  \
0       14       1287.0        4           16                 104
1        0         13.0        0            0                   1
2        0         98.0        0            0                   0
3      527      60753.0      149          610                3914
4       11       1090.0        0           11                   4
5      213      28129.0       65          245                1783
6        4        432.0        4            7                  33
7        0         31.0        0            0                   2
8        6        705.0        6           12                 130
9       36       4637.0       80           79                 722
10      15       1813.0        0           15                 230
11       0          NaN    23664         2285              104651
12       0          NaN        0            0                   4
13      22       3378.0        0           22                   0
14       0         13.0        0            0                   3
15       0          NaN     1751          110                5750
16       0          NaN       55            3                 111
17       0          NaN        5            0                   8
```

```
18       5    1563.0     17         11                 120
19      20    2405.0      0         20                   0
20     113   15320.0      0        113                1729
21       0       NaN      2          0                   4
22       0       NaN    777         10                1078
23       0       NaN    155          6                 292
24       0       NaN      2          0                   3

    search_page_views conversion_rate bounce_rate add_to_cart_rate
0                 192        0.583333    0.166667         0.666667
1                   0
2                   0
3                7367        0.571584    0.161605         0.661605
4                  19               1           0                1
5                3255        0.554688    0.169271         0.638021
6                  52        0.285714    0.285714              0.5
7                   2               0           0                0
8                 272        0.146341    0.146341         0.292683
9                1073       0.0803571    0.178571         0.176339
10                216               1           0                1
11             258511               0    0.406654        0.0392666
12                  2
13                  0               1           0                1
14                  1
15              14185               0     0.35431        0.0222582
16                198               0    0.495495         0.027027
17                 10               0    0.714286                0
18                232       0.0892857    0.303571         0.196429
19                  0               1           0                1
20               1466               1           0                1
21                  4               0         0.5                0
22               1840               0    0.789634        0.0101626
23                297               0    0.549645        0.0212766
24                  4               0         0.5                0
```

In [43]:
```
mydata.describe() # Summary of numerical variables
```

Out[43]:
```
       new_customer          visits  distinct_sessions         orders  \
count  12802.000000    21061.000000       21061.000000   21061.000000
mean       0.448055     1934.708039        1515.205024      62.378994
std        0.497314     7448.607191        5925.833287     260.279286
min        0.000000        0.000000           0.000000       0.000000
25%        0.000000        3.000000           2.000000       0.000000
50%        0.000000       24.000000          19.000000       0.000000
75%        1.000000      360.000000         274.000000       7.000000
max        1.000000   136057.000000      107104.000000    4916.000000

         gross_sales        bounces   add_to_cart  product_page_views  \
```

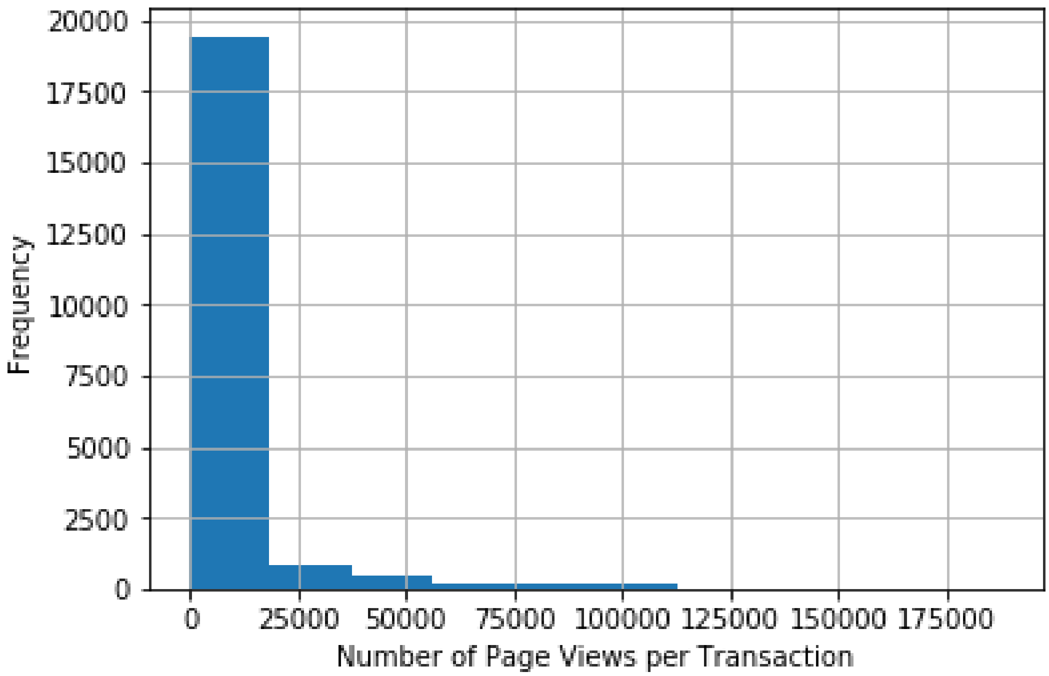
```
count    11485.000000  21061.000000  21061.000000        21061.000000
mean     16473.395821    743.282085    166.250890         4358.198234
std      51111.354605   3154.697787    505.186834        14327.287354
min          1.000000      0.000000      0.000000            0.000000
25%         79.000000      0.000000      0.000000            3.000000
50%        851.000000      5.000000      4.000000           53.000000
75%       3145.000000     97.000000     43.000000          708.000000
max     707642.000000  54512.000000   7924.000000       187601.000000

       search_page_views
count       21061.000000
mean         8584.187788
std         31120.321365
min             0.000000
25%             4.000000
50%            82.000000
75%          1229.000000
max        506629.000000
```

In [44]:
```
mydata['new_customer'].value_counts()
```

Out[44]:
```
0.0    7066
1.0    5736
Name: new_customer, dtype: int64
```

In [45]:
```
mydata['product_page_views'].hist()
plt.xlabel("Number of Page Views per Transaction")
plt.ylabel("Frequency")
```

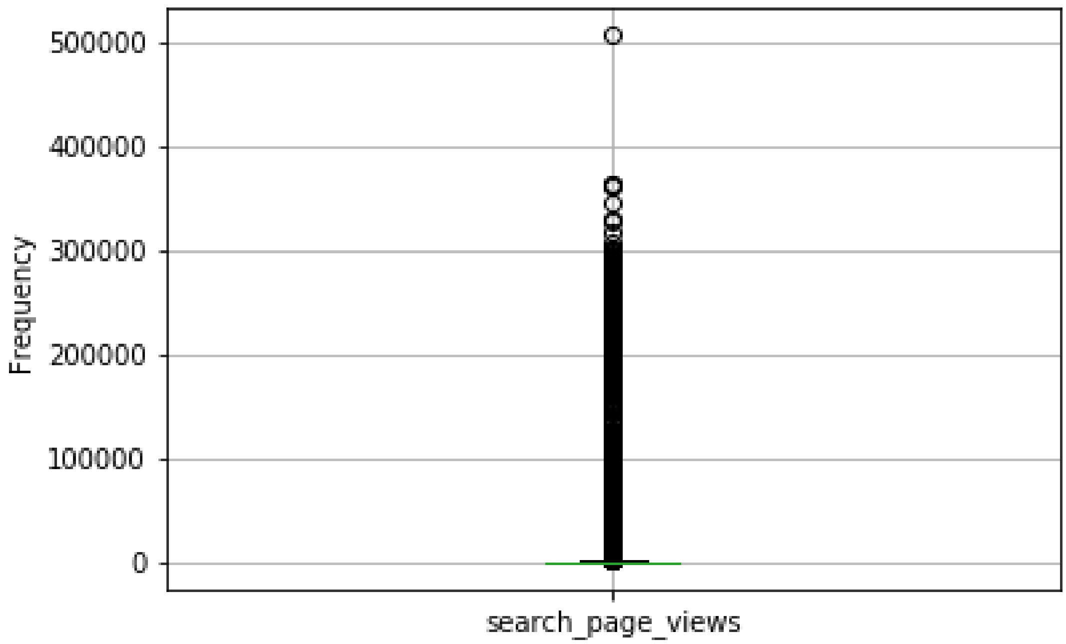
Out[45]:
```
Text(0,0.5,'Frequency')
```



As we can see in the above histogram there are a few extreme values or outliers. Plotting such distributions will help us in understanding how these outliers can affect the statistical assumptions and analysis of our data. Next, let us take a look at boxplots of some of columns in **mydata**.

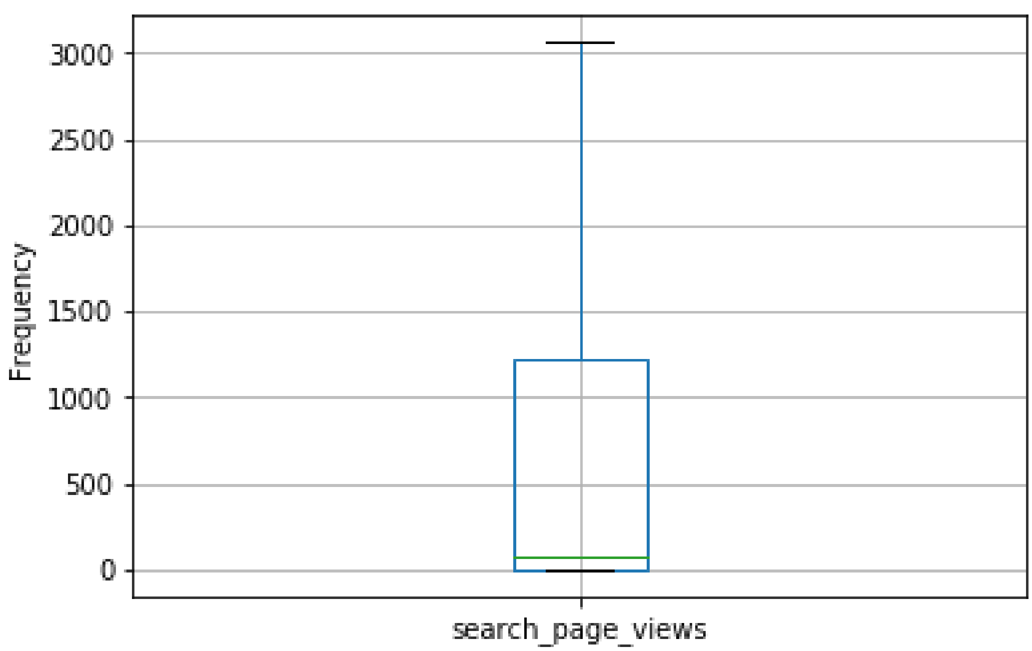
Below are two boxplots of the *search_page_views* column with extreme values and without extreme values. The first plot shows a heavy presence of outliers, which makes our distribution very skewed. From the second plot, we can see that the minimum value is the same as the 1st (lower) quartile.

```
In [46]: mydata.boxplot(column='search_page_views')
         plt.ylabel("Frequency")

Out[46]: Text(0,0.5,'Frequency')
```




```
In [47]: mydata.boxplot(column='search_page_views', showfliers=False)
         plt.ylabel("Frequency")
```

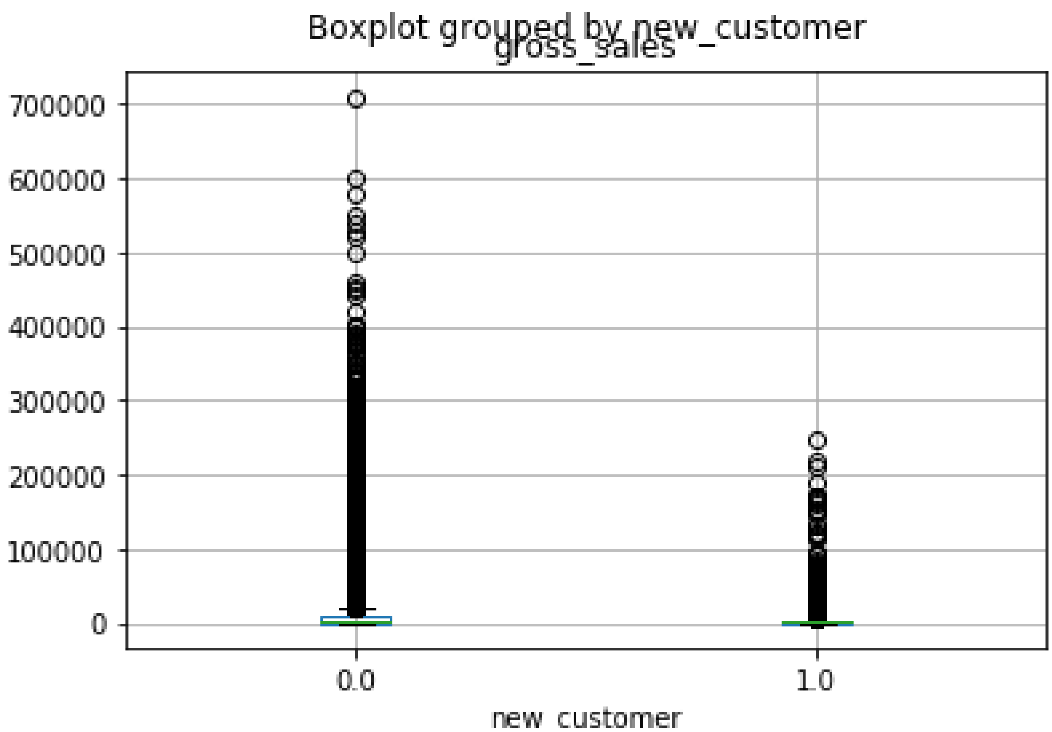
```
Out[47]: Text(0,0.5,'Frequency')
```



One question we may ask is what is the comparison between the gross sales made by new customers (1) versus returning customers (0). Using the below two boxplots, we are able to see that the median and range of gross sales for returning customers is quite bigger than those of new customers. However, both plots indicate that the distribution of the gross sales for each type of customer dimension is skewed right. The first boxplot shows countless outliers that make it nearly impossile to see the spread of the data column. By removing the outliers, we get a better look of the distribution (second plot).

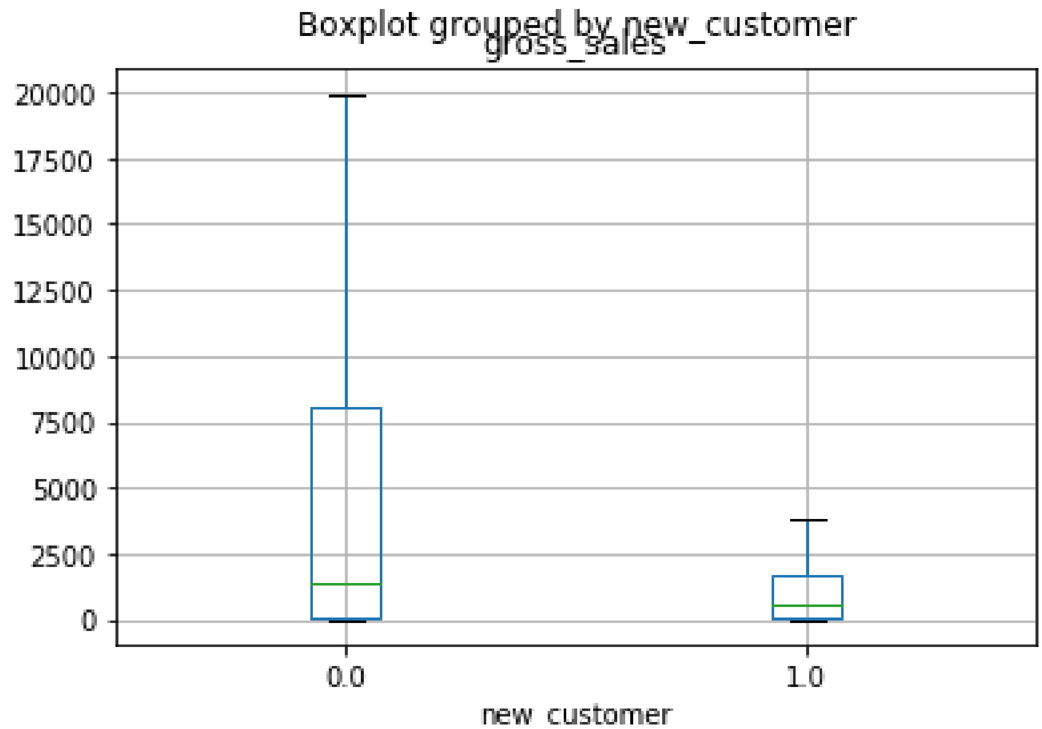
```
In [48]: mydata.boxplot(column='gross_sales', by= 'new_customer')
```

```
Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x13f7cef6668>
```




```
In [49]: mydata.boxplot(column='gross_sales', by= 'new_customer', showfliers=False)
```

```
Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0x13f7be0f588>
```



We can see how strong the relationship between the types of customers we've had and gross sales by finding the correlation coefficient, which will solidfy the evidence we saw in the distribution plots. The results shows that there isn't a strong nor positive relationship between the users and the gross sales.

```
In [50]: mydata['new_customer'].corr(mydata['gross_sales'])

Out[50]: -0.21188490786384043
```

We want to see the sales, orders, visits and bounces by month. First, we will have to convert *day* to be datetime format using the **datetime** package.

```
In [51]: import datetime

         mydata['year'] = pd.DatetimeIndex(mydata['day']).year
         mydata['month'] = pd.DatetimeIndex(mydata['day']).month
         mydata['dow'] = pd.DatetimeIndex(mydata['day']).weekday_name
         mydata['myday'] = pd.DatetimeIndex(mydata['day']).day

         print(mydata['year'])
         print(mydata['month'])
         print(mydata['dow'])
         print(mydata['myday'])
```

```
0        2013
1        2013
2        2013
3        2013
4        2013
5        2013
6        2013
7        2013
8        2013
9        2013
10       2013
11       2013
12       2013
13       2013
14       2013
15       2013
16       2013
17       2013
18       2013
19       2013
20       2013
21       2013
22       2013
23       2013
24       2013
25       2013
26       2013
27       2013
28       2013
29       2013
         ...
21031    2013
21032    2013
21033    2013
21034    2013
21035    2013
21036    2013
21037    2013
21038    2013
21039    2013
21040    2013
21041    2013
21042    2013
21043    2013
21044    2013
21045    2013
21046    2013
21047    2013
```

```
21048    2013
21049    2013
21050    2013
21051    2013
21052    2013
21053    2013
21054    2013
21055    2013
21056    2013
21057    2013
21058    2013
21059    2013
21060    2013
Name: year, Length: 21061, dtype: int64
0         1
1         1
2         1
3         1
4         1
5         1
6         1
7         1
8         1
9         1
10        1
11        1
12        1
13        1
14        1
15        1
16        1
17        1
18        1
19        1
20        1
21        1
22        1
23        1
24        1
25        1
26        1
27        1
28        1
29        1
         ..
21031    12
21032    12
21033    12
```

```
21034    12
21035    12
21036    12
21037    12
21038    12
21039    12
21040    12
21041    12
21042    12
21043    12
21044    12
21045    12
21046    12
21047    12
21048    12
21049    12
21050    12
21051    12
21052    12
21053    12
21054    12
21055    12
21056    12
21057    12
21058    12
21059    12
21060    12
Name: month, Length: 21061, dtype: int64
0        Tuesday
1        Tuesday
2        Tuesday
3        Tuesday
4        Tuesday
5        Tuesday
6        Tuesday
7        Tuesday
8        Tuesday
9        Tuesday
10       Tuesday
11       Tuesday
12       Tuesday
13       Tuesday
14       Tuesday
15       Tuesday
16       Tuesday
17       Tuesday
18       Tuesday
19       Tuesday
```

```
20       Tuesday
21       Tuesday
22       Tuesday
23       Tuesday
24       Tuesday
25       Tuesday
26       Tuesday
27       Tuesday
28       Tuesday
29       Tuesday
           ...
21031    Tuesday
21032    Tuesday
21033    Tuesday
21034    Tuesday
21035    Tuesday
21036    Tuesday
21037    Tuesday
21038    Tuesday
21039    Tuesday
21040    Tuesday
21041    Tuesday
21042    Tuesday
21043    Tuesday
21044    Tuesday
21045    Tuesday
21046    Tuesday
21047    Tuesday
21048    Tuesday
21049    Tuesday
21050    Tuesday
21051    Tuesday
21052    Tuesday
21053    Tuesday
21054    Tuesday
21055    Tuesday
21056    Tuesday
21057    Tuesday
21058    Tuesday
21059    Tuesday
21060    Tuesday
Name: dow, Length: 21061, dtype: object
0         1
1         1
2         1
3         1
4         1
5         1
```

6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1
16	1
17	1
18	1
19	1
20	1
21	1
22	1
23	1
24	1
25	1
26	1
27	1
28	1
29	1
	..
21031	31
21032	31
21033	31
21034	31
21035	31
21036	31
21037	31
21038	31
21039	31
21040	31
21041	31
21042	31
21043	31
21044	31
21045	31
21046	31
21047	31
21048	31
21049	31
21050	31
21051	31
21052	31
21053	31

```
21054    31
21055    31
21056    31
21057    31
21058    31
21059    31
21060    31
Name: myday, Length: 21061, dtype: int64
```

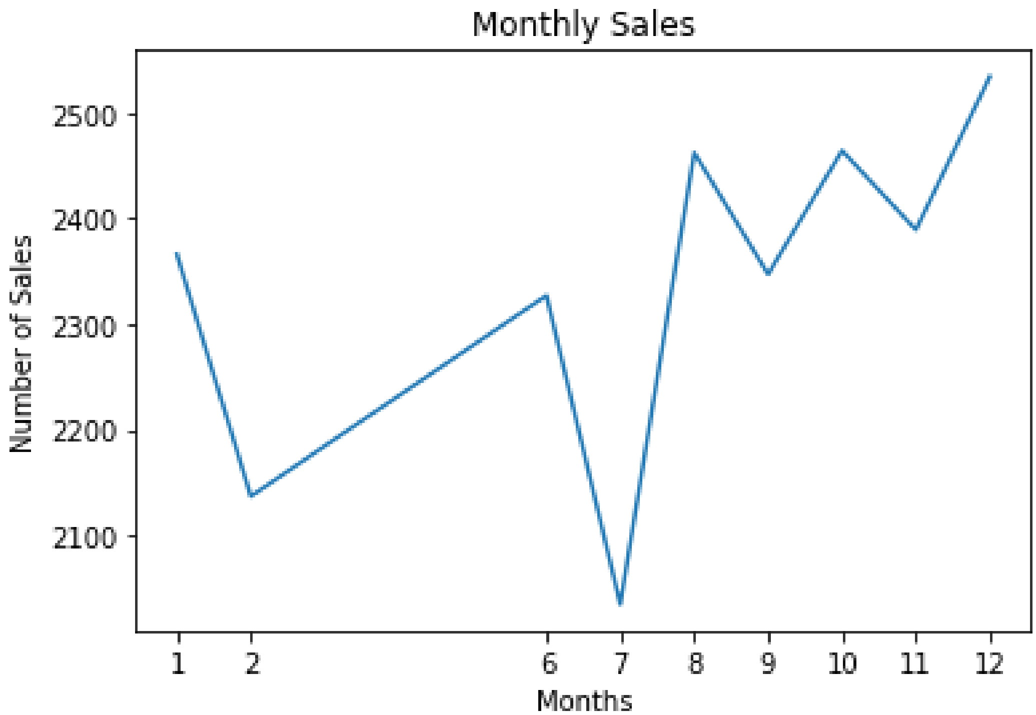
After creating new columns **year**, **month**, **dow** to hold the year, month and day of the week (Monday-Sunday), we can find the distributions grouped by these columns.

Sales by Month

```
In [52]: #Sales
         sales_by_month = mydata.groupby(mydata['month']).size()
         print(sales_by_month)

         #Plotting the graph
         plot_by_month = sales_by_month.plot(title='Monthly Sales', xticks=(1,2,6,7,8,9,10,11,1
         plot_by_month.set_xlabel('Months')
         plot_by_month.set_ylabel('Number of Sales')
```

```
month
1     2366
2     2137
6     2327
7     2035
8     2462
9     2347
10    2464
11    2389
12    2534
dtype: int64
```

```
Out[52]: Text(0,0.5,'Number of Sales')
```



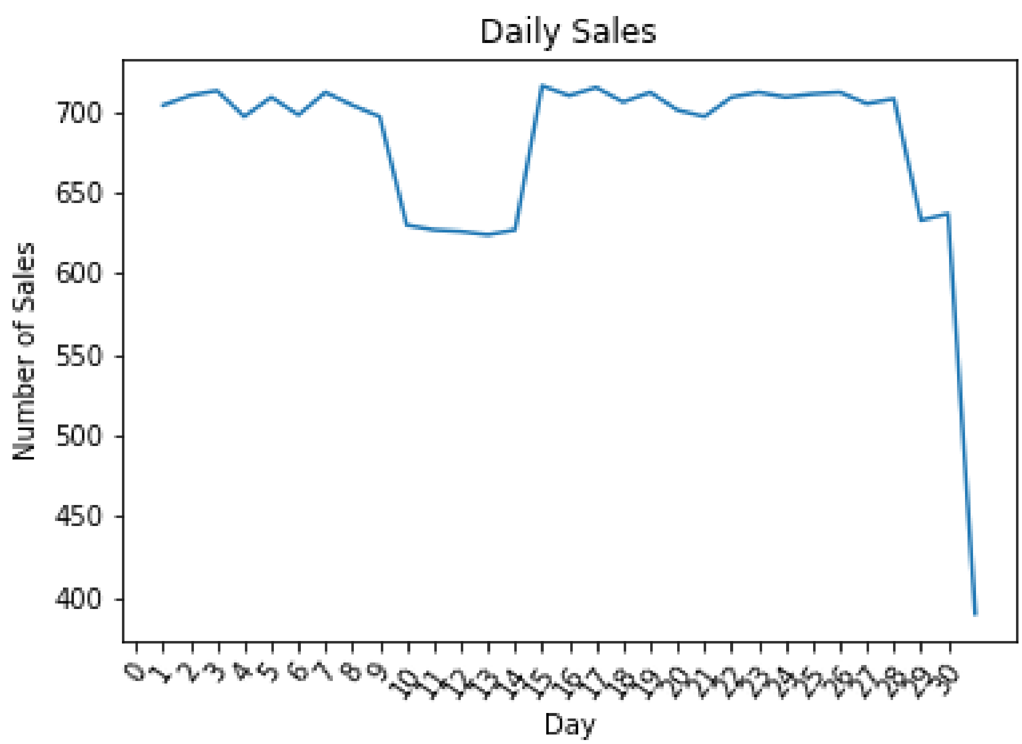
Sales by Day

```
In [53]: #Sales
         sales_by_day = mydata.groupby(mydata['myday']).size()
         print(sales_by_day)

         #Plotting the graph
         sales_by_day = sales_by_day.plot(title='Daily Sales', xticks=(range(0,31)), rot=55)
         sales_by_day.set_xlabel('Day')
         sales_by_day.set_ylabel('Number of Sales')
```

```
myday
1      704
2      710
3      713
4      697
5      709
6      698
7      712
8      704
9      697
10     630
11     627
12     626
13     624
```

```
14      627
15      716
16      710
17      715
18      706
19      712
20      701
21      697
22      709
23      712
24      709
25      711
26      712
27      705
28      708
29      633
30      637
31      390
dtype: int64
```

```
Out[53]: Text(0,0.5,'Number of Sales')
```



Sales by Day of the Week

In [54]:
```
#Sales
sales_by_dow = mydata.groupby(mydata['dow']).size()
print(sales_by_dow)

#Plotting the graph
sales_by_dow = sales_by_dow.plot(title='Day of Week Sales', xticks = range(0,7), rot=(
sales_by_dow.set_xlabel('Day of Week')
sales_by_dow.set_ylabel('Number of Sales')
```

```
dow
Friday       2895
Monday       3064
Saturday     2973
Sunday       2971
Thursday     3006
Tuesday      3151
Wednesday    3001
dtype: int64
```

Out[54]:
```
Text(0,0.5,'Number of Sales')
```