Estimation of Groupon's Gross Billings

——— ***Xiaoxuan Ma***

Summary

I used Excel and Python for analysis and calculation. The historical data showed the ***seasonality*** *of Groupon's business. I cleaned the raw data by deleting the deals with unit prices higher than $100,000. And I used ordinary least squares regression to adjust the missing local deals on Oct 20-30, 2013. My estimate of Groupon's 4Q13 North America* ***total gross billing is $805.3 million****. Pretending it is January 2014, before Groupon reports 4Q13 earnings in February 2014, based on the bad performance of Groupon stock in January and the lower estimate of Groupon's 4Q13 North America gross billings, my recommendation is to* ***sell*** *Groupon stock.*

Agenda

Business Overview

Business

Groupon is an American global e-commerce marketplace connecting subscribers with local merchants by offering activities, travel, goods and services in 48 countries. They distribute their deals to customers primarily through three channels: email; mobile platform; and websites. By bringing the brick and mortar world of local commerce onto the Internet and mobile devices, Groupon is creating a new way for local merchant partners to attract customers and sell goods and services. In 2012, they made some significant successes, particularly regarding customer demand: Gross billings grew 35 percent to $5.4 billion.

Gross billings

In this project, we will focus on estimating Groupon's 4Q13 North America gross billings by segment. This metric represents the total dollar value of customer purchases of goods and services, excluding applicable taxes and net of estimated refunds.

Categories

There are three main categories of Groupon: **Local**, **Travel**, and **Goods**.

Within Local, they offer deals for local merchant partners across multiple categories, including food and drink, events and activities, beauty and spa, fitness, health, home and auto, shopping, and education.

Through Travel, they feature personally curated offers from travel partners, including hotels, airfare and package deals covering both domestic and international travel.

Goods segment offers customers the consistent ability to find deals on a rotating selection of well-known brands across multiple product lines, including electronics, sports, outdoors & fitness, toys, home, and clothing.

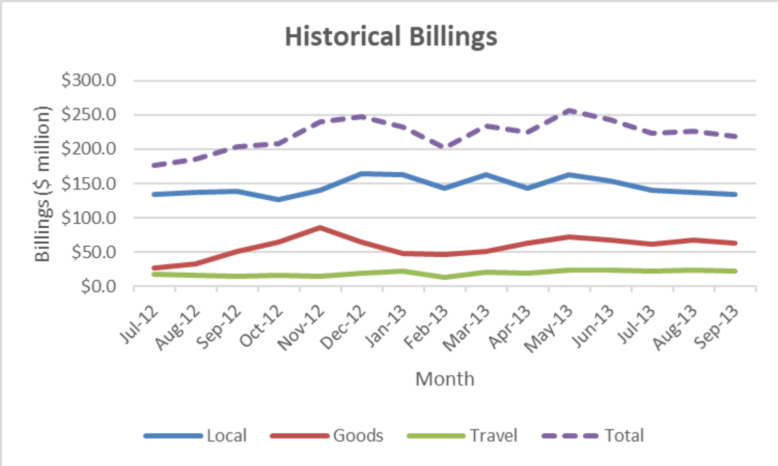
Seasonality

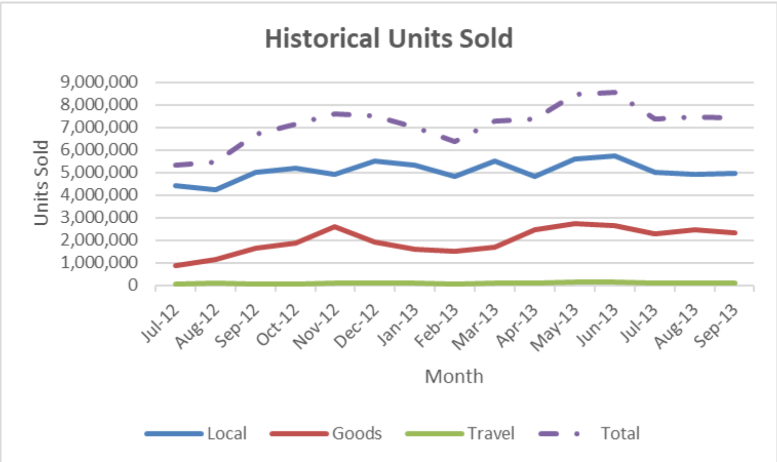
Some of Groupon's offerings experience seasonal buying patterns mirroring that of the larger consumer and e-commerce markets, where demand declines during customary summer vacation periods and increases during the fourth quarter holiday season.

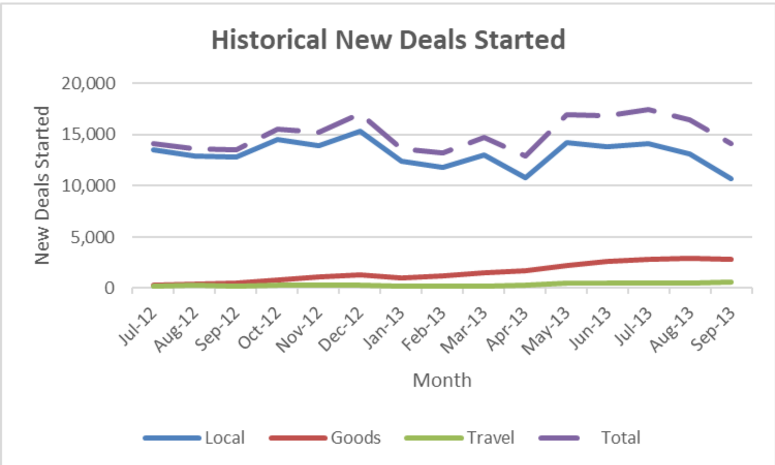
Raw Data Analysis

Historical Data

Based on the historical estimates (Jul 2012 to Sep 2013) provided by YipitData, I made three diagrams.







From the diagrams, I found there is no continuous growth in the three metrics, showing a certain seasonal effect. Around December and May, Groupon's business is booming. This may be caused by the effect of the holiday season. And the Local category is the most important part, which took the majority of billings, units sold and new deals. The Travel category took the minority.

Q4 2013 Raw data

This dataset is YipitData's proprietary estimate of gross billings and units sold for each deal that was active in Groupon's North America segment in Q4 2013.

Each row represents a Groupon deal that was active for some or all of Q4 2013. For each row they've provided data on units sold during the period, gross billings during the period, the date that the deal started, the URL of the deal page, the product segment of the deal (Local, Travel, or Goods), and the inventory type of the good (first-party means Groupon owns the inventory).

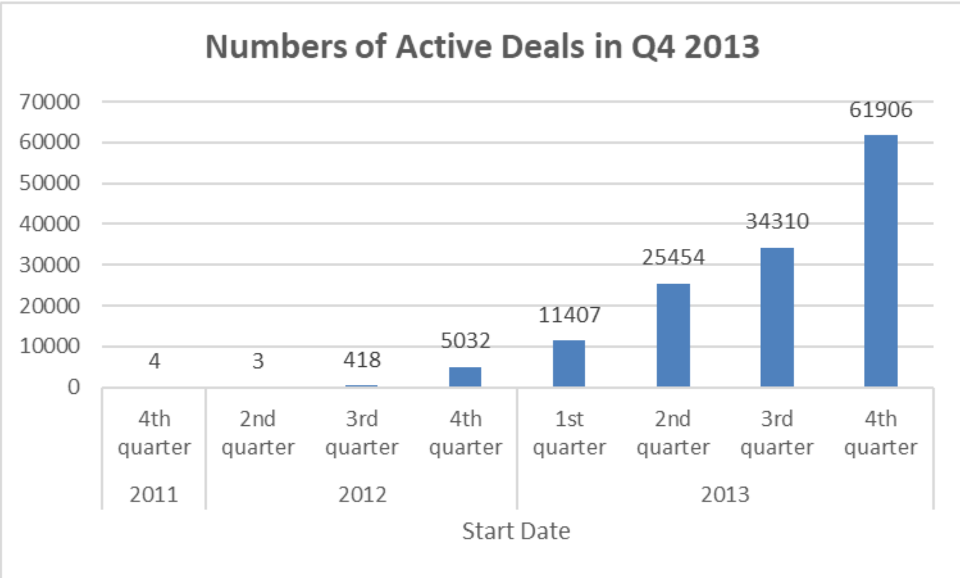
You'll notice that units sold are often in decimals. This is because they are employing estimation techniques behind the scenes - we can ignore the methodology behind these estimations and just take the data as given.

Ignoring the missing deals started on Oct 20th – 30th, I checked some basic irregularities in the data:

irregularities	number
NULL values	0
Units Sold == 0 & Billings != 0	0
Units Sold > 0 & Billings <= 0	0
Units Sold < 0 & Billings >= 0	0

There aren't any above irregularities in the raw data.

Then, let's check the distribution of the start dates of the active deals.



The deals can be active for many days, the oldest deals launched in the 4th quarter of 2011. The majority of active deals launched in the 4th quarter of 2013. And the later the start date, the more deals we can find.

As we can ignore the methodology behind these estimations of **Units sold** and just take the data as given, I'll focus on the **Billings**. Using the pivot table in Excel, I got some basic descriptive statistics, which can help us understand the characteristics of samples.

Billings	count	sum	mean	max	min	Standard Deviation
Goods	15234	282245671	18527	2874885	-147360	66017
Local	120576	409222658	3394	1371875	-218063	13623
Travel	2724	70552062	25900	1552777	-75024	70758
overall	138534	762020391	5501	2874885	-218063	27747

Only part of deals in Goods segment are provided by First-Party, all deals in the Local and Travel segments are provided by Third-Party. The Local category takes the majority of total billing. And it has the lowest average billing and standard deviation. Therefore, for estimating the missing deals in the Local category, I won't take samples from other categories.

Besides, we can find the Billings variable has a very high variation, it's hard to find irregularities directly. To better estimate it, I decided to create some related metrics.

Data Adjustments

Creating New Metrics and Cleaning Outliers

As I mentioned before, the Billings variable has a very high variation, because the deals have different active days, unit prices and sold units. I created three new metrics based on the raw data to better understand the data feature. I assumed the deals stay active till 2014-01-01.

Active Days = the number of days between the Start Date and 2014-01-01
Daily Billings = Billings / Active Days
Unit Prices = Billings / Units Sold

The dataframe looks like this:

	Deal ID	Units Sold	Billings	Unit Prices	Start Date	Active Days	Daily Billings	Deal URL	Segment	Inventory Type
0	gr-millevois-tire-service-center	0.0	0.0	NaN	2011-11-21	772 days	0.000000	http://www.groupon.com/deals/gr-millevois-tire...	Local	Third - Party
1	gr-manakeesh-cafe-bakery	0.0	0.0	NaN	2011-11-21	772 days	0.000000	http://www.groupon.com/deals/gr-manakeesh-cafe...	Local	Third - Party
2	gr-phoenix-salon-and-spa	0.0	0.0	NaN	2011-11-21	772 days	0.000000	http://www.groupon.com/deals/gr-phoenix-salon-...	Local	Third - Party
3	gr-hands-in-motion	0.0	0.0	NaN	2011-11-21	772 days	0.000000	http://www.groupon.com/deals/gr-hands-in-motion	Local	Third - Party
4	dc-fd2-bartending-college-allentown-reading	86.8	4253.2	49.0	2012-06-06	574 days	7.409756	http://www.groupon.com/deals/dc-fd2-bartending...	Local	Third - Party

Now, let's have a look at some basic descriptive statistics of new metrics:

	count	mean	std	min	25%	50%	75%	max
Unit Prices(without NaN)	111760.00	1.12E+14	1.37E+16	0.01	22.08	39.00	65.00	3.46E+18
Unit Prices Adjusted(without NaN)	111706.00	67.91	358.38	0.01	22.04	39.00	65.00	99119.50
Daily Billings	138534.00	159.82	1284.71	-1503.68	0.51	8.05	56.40	229885.00

The statistics of Daily Billings look fine. But we can find the mean, standard deviation and maximum of Unit Prices are extremely high. This is abnormal for online deals.

If we use boxplot to find outliers in Unit Prices, the lower boundary is ***Q1 -1.5*IQR = -42.31***, the upper boundary is ***Q3 + 1.5*IQR = 129.38***. The upper boundary is too strict for our data set, it's pretty normal that the price of a product is higher than $130.

Having a closer look at the data, I found there is a gap in Unit Prices.

Index	Deal ID	Units Sold	Billings	Unit Prices	Start Date	Active Days	Daily Billings	Deal URL	Segment	Inventory Type
19205	gateway-fun-p…	-2.27374e-13	-272.5	1.19847e+15	2013-05-31 00…	215 days 00:0…	-1.26744	http://www.gr…	Local	Third - Party
26964	gg-1-jessica-…	-2.12452e-14	-13.34	6.27906e+14	2013-07-11 00…	174 days 00:0…	-0.0766667	http://www.gr…	Goods	First - Party
33450	gg-rlc-ultima…	-1.04592e-13	-50.14	4.79387e+14	2013-08-13 00…	141 days 00:0…	-0.355603	http://www.gr…	Goods	First - Party
15565	icon-parking-…	-2.2	-218063	99119.5	2013-04-22 00…	254 days 00:0…	-858.515	http://www.gr…	Local	Third - Party
15567	icon-parking-…	-3.65	-140086	38379.8	2013-04-22 00…	254 days 00:0…	-551.521	http://www.gr…	Local	Third - Party
77318	ga-bk-the-wes…	22.55	225500	10000	2013-11-14 00…	48 days 00:00…	4697.92	http://www.gr…	Travel	Third - Party

Therefore, I set 100,000 as the upper boundary for Unit Prices, which means if the unit price of a deal is greater than 100,000, the deal will be regarded as an outlier.

As we can take the Units Sold as given, the only part we can adjust is Billings. For the outliers, we can use the URL to find the true Unit Prices. But the outliers' sold units are extremely small. If I adjust the outliers' Billings by multiplying their Units Sold and true Unit Price, I'll get some very small Billings, which won't have a big impact on the Total Gross Billings. So, I directly deleted the outliers.

After I cleaned the outliers, the descriptive statistics of Unit Prices look normal **(see the row of Unit Prices Adjusted(without NaN))**.

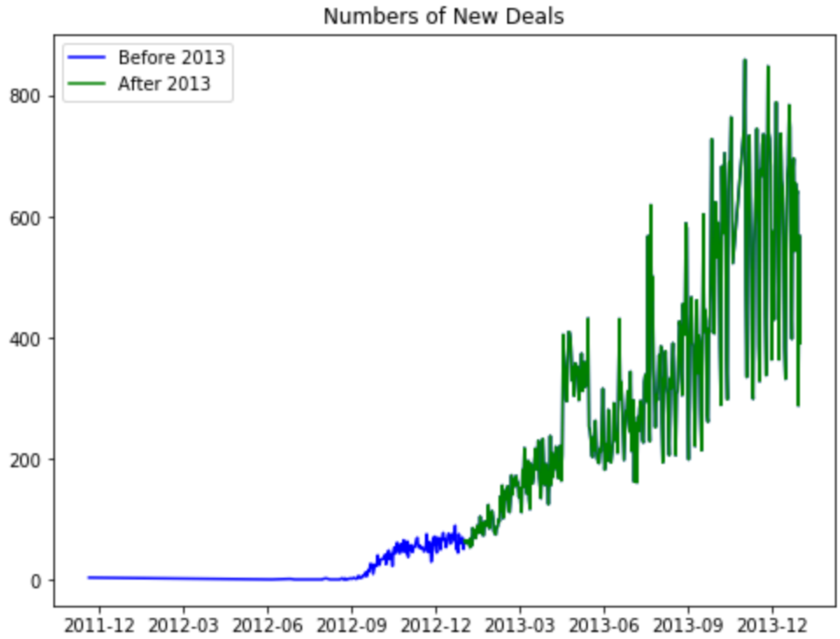
Adjustment for Missing Deals on Oct 20-30, 2013 of the Local Segment

The dataset includes zero Local deals that started from October 20 to October 30, 2013(inclusive). I estimated the total Billings of the missing deals as below:

Total Billings of the Missing Deals on Oct 20-30, 2013
= sum(Total Billings of the Missing Deals on Each Day)
= sum(<u>Numbers of New Deals Launched on Each Day</u> * <u>Active Days</u> * <u>Average Daily Billings</u>)

As I mentioned before, the Billings of the Local category has a big difference with other categories. Therefore, for estimating the missing deals, **I only took samples from the Local category**.

Numbers of New Deals (Active in 4Q13) Launched on Each Day
By grouping deals of the Local category by Start Dates, we can get the number of new deals of each recorded date.



From this diagram, we can find that generally, the number of new deals kept going up. And the trend is clearer after 2013. So, I only took the numbers of new deals after 2013 from the Local category as samples.

To capture the features of **periodicity** and **seasonality**, which are shown in the diagram, I built the following model:

$$Numbers\ of\ \ New\ Deals\ = \sum_{i=1}^{12}(\alpha_i * Month_i) + \sum_{j=0}^{6}(\beta_j * Weekday_j)$$

$\alpha_i s,\ \beta_j s$ are coefficients.

$Month_i s,\ Weekday_j s$ are dummy variables.

For example, for 2013-01-01, Tuesday, $Month_1$, $Weekday_1 = 1$, other dummy variables = 0.

I used **ordinary least squares regression (OLS)** to fit the model. Here is the result:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Count   R-squared:                       0.804
Model:                            OLS   Adj. R-squared:                  0.794
Method:                 Least Squares   F-statistic:                     80.85
Date:                Mon, 06 Apr 2020   Prob (F-statistic):          1.51e-107
Time:                        22:51:47   Log-Likelihood:                -2071.2
No. Observations:                 354   AIC:                             4178.
Df Residuals:                     336   BIC:                             4248.
Df Model:                          17
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Month_1     -130.7687     14.959     -8.742      0.000    -160.194    -101.344
Month_2      -82.1272     15.702     -5.230      0.000    -113.014     -51.240
Month_3      -34.4121     14.959     -2.300      0.022     -63.838      -4.986
Month_4       49.1255     15.196      3.233      0.001      19.233      79.018
Month_5       73.2103     14.959      4.894      0.000      43.785     102.635
Month_6       52.1212     15.196      3.430      0.001      22.229      82.013
Month_7       98.2431     14.960      6.567      0.000      68.816     127.670
Month_8      135.7584     14.958      9.076      0.000     106.335     165.182
Month_9      196.1321     15.197     12.906      0.000     166.238     226.026
Month_10     355.7012     18.514     19.213      0.000     319.284     392.119
Month_11     380.6263     15.196     25.049      0.000     350.736     410.517
Month_12     369.7807     14.960     24.717      0.000     340.353     399.208
Weekday_0    216.9112     11.698     18.542      0.000     193.900     239.922
Weekday_1    226.7600     11.570     19.600      0.000     204.002     249.518
Weekday_2    240.2291     11.680     20.567      0.000     217.254     263.204
Weekday_3    261.2792     11.565     22.593      0.000     238.531     284.028
Weekday_4    239.4109     11.570     20.693      0.000     216.653     262.169
Weekday_5    159.2362     11.570     13.763      0.000     136.477     181.995
Weekday_6    119.5644     11.698     10.221      0.000      96.553     142.575
==============================================================================
```

Supposing a significance level of 0.05, the P-value of the model is smaller than 0.05, which means the model is significant. And the P-values of coefficients (αs, βs) are also smaller than 0.05. They are also significant.

The R-squared (coefficient of determination) is 0.8, which means 80% of the variance in the dependent variable is predictable from the independent variables. This indicates the model fitted the data pretty well.

Using this model, I predicted the numbers of new deals launched on Oct 20th – 30th.

Active Days

The precondition is assuming the deals stay active till 2014-01-01. I'll multiply the Active Days and Average Daily Billings to get Billings, the first two variables are based on the same precondition. Therefore, this assumption won't affect the final result.

Active Days = the number of days between the Start Date and 2014-01-01

Here are the results of the above two steps:

	Date	Days to End	New Deals
0	2013-10-20	73	475.265556
1	2013-10-21	72	572.612366
2	2013-10-22	71	582.461159
3	2013-10-23	70	595.930277
4	2013-10-24	69	616.980364
5	2013-10-25	68	595.112094
6	2013-10-26	67	514.937369
7	2013-10-27	66	475.265556
8	2013-10-28	65	572.612366
9	2013-10-29	64	582.461159
10	2013-10-30	63	595.930277

Average Daily Billings

From the former analysis, I found seasonality affected Billings, and during December, Groupon's business was booming. This is caused by the effect of the holiday season. I assume the deals launched in December may have different features with deals launched in October. Therefore, I used the ***Average Daily Billings of deals from the Local category launched between 2013-10-01 and 2013-11-30*** to estimate the Average Daily Billings of the missing deals. It equals 103.13.

Combing the results of three steps, I got:

Total Billings of the Missing Deals on Oct 20-30, 2013

= ***sum(Numbers of New Deals Launched on Each Day * Active Days * Average Daily Billings)***

= $**43310413.68**

Gross Billings Estimates by Segment

By aggregating the Billings by segment, I got the estimates of Groupon's 4Q13 North America gross billings.

Segment	Gross Billings ($ million)
Local	$452.5
Goods	$282.2
Travel	$70.5
Total	**$805.3**

Recommendation for Groupon Stock

Other Companies' Recommendations

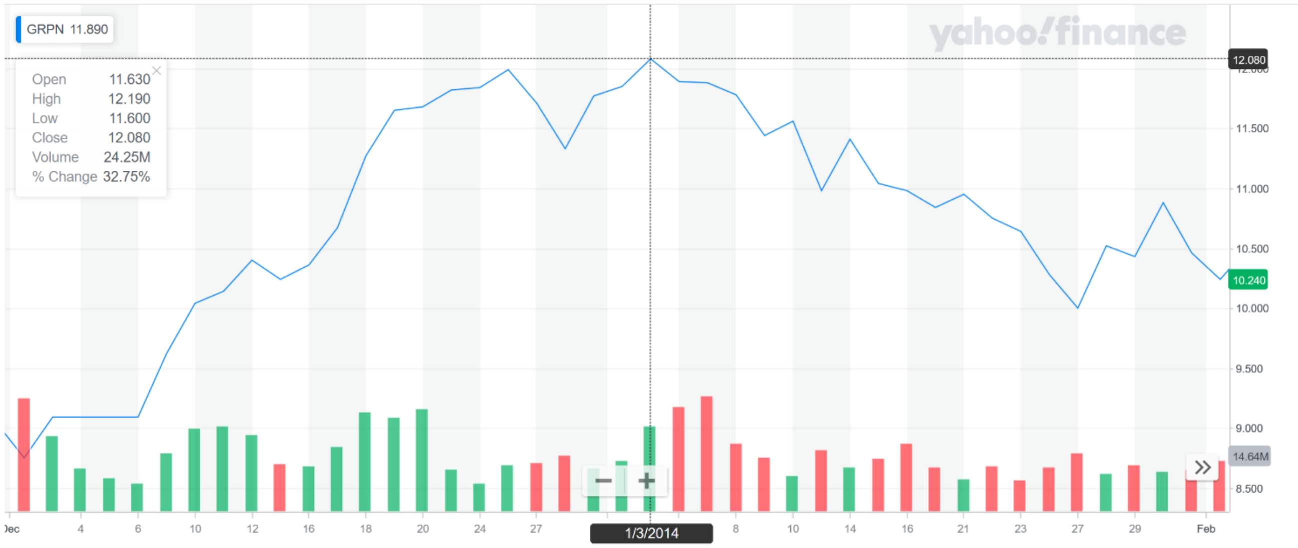
Company	Report Date	Rating	Previous Groupon Stock Price	Price target	Estimate of GRPN 4Q13 North America Gross Billings.
Deutsche Bank	2013/11/8	Buy	$9.50(2013/11/07)	$16.00(New) $17.00 (Old)	$803.2 million (New) $884.3 million (Old)
J.P.Morgan	2013/11/8	Neutral	$9.50(2013/11/07)	$11.00(New) $10.00(Old)	$836.9 million
Morgan Stanley	2013/12/10	Overweight	$9.62(2013/12/09)	$15.00	$870.1 million

Deutsche Bank's rating for Groupon on 8th November 2013 is **Buy**. Their estimate of Groupon's 4Q13 North America gross billing is $803.2 million, which is lower than their previous estimate. Their $16 price target is based on 25x 2014 EBITDA which represents a significant discount to the fast-growing internet peer universe multiple range at ~40x 2014 EBITDA. Key downside risks include Google mail changes, less favorable payment terms in Europe, and increased competition.

J.P.Morgan's rating for Groupon on 8th November 2013 is **Neutral**. Their estimate of Groupon's 4Q13 North America gross billing is $836.9 million. Their year-end 2014 price target of $11 is based on ~9.5x their 2015E EBITDA of $479M, roughly in line with industry peers such as Google and eBay. They mentioned downside risks include that users are likely feeling some degree of email and deal fatigue, thereby slowing growth in the local deals space.

Morgan Stanley's rating for Groupon on 10th December 2013 is **Overweight**. Their estimate of Groupon's 4Q13 North America gross billing is $870.1 million. Their price target of $15 is based on DCF, WACC of 14.5% and a perpetual growth rate of 4.5%. They mentioned downside risks include that Groupon's core growth continues to be sluggish. Take rates decline and both deals and goods businesses don't accelerate as the company's initiatives to attract better merchants to attract more customers fails to resonate. EBITDA margins decline to 8.5%.

My Recommendation: Sell



As we can see from the above diagram, the general trend of Groupon stock price is upward in December 2013, is downward in January 2014. Groupon stock didn't perform well in January 2014.

My estimate of Groupon's 4Q13 North America gross billing is $805.3 million, which is lower than J.P.Morgan and Morgan Stanley's estimates, and is slightly higher than Deutsche Bank's estimate. The average of estimates made by three companies is $836.7 million. If we regard this value as Wall Street consensus, we can say that Groupon's 4Q13 North America gross billing doesn't meet the expectation of Wall Street companies.

As I mentioned before, because of the seasonality, the fourth quarter should be the most profitable period. And the North American market takes the majority of Groupon's business. A lower 4Q gross billing should have a heavier negative impact on investors.

Based on the bad performance of Groupon stock in January and the lower estimate of Groupon's 4Q13 North America gross billings, my recommendation is to **sell** Groupon stock.

Python Codes

Groupon Exercise

Raw Data Analysis

Q4 2013 Raw data

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
```

In [2]:
```
raw_data = pd.read_excel('Q4_2013_Groupon_North_America_Data_XLSX.xlsx',sheet_name='Q4 2013 Raw Data')
```

In [3]:
```
print(raw_data.shape)
raw_data.head()
```

```
(138534, 7)
```

Out[3]:

	Deal ID	Units Sold	Billings	Start Date	Deal URL	Segment	Inventory Type
0	gr-millevois-tire-service-center	0.0	0.0	2011-11-21	http://www.groupon.com/deals/gr-millevois-tire...	Local	Third - Party
1	gr-manakeesh-cafe-bakery	0.0	0.0	2011-11-21	http://www.groupon.com/deals/gr-manakeesh-cafe...	Local	Third - Party
2	gr-phoenix-salon-and-spa	0.0	0.0	2011-11-21	http://www.groupon.com/deals/gr-phoenix-salon-...	Local	Third - Party
3	gr-hands-in-motion	0.0	0.0	2011-11-21	http://www.groupon.com/deals/gr-hands-in-motion	Local	Third - Party
4	dc-fd2-bartending-college-allentown-reading	86.8	4253.2	2012-06-06	http://www.groupon.com/deals/dc-fd2-bartending...	Local	Third - Party

In [4]:
```
df = raw_data.copy(deep=True)
df['Start Date'] = pd.to_datetime(df['Start Date'])
```

In [5]:
```
df.isnull().any(axis = 0)
```

Out[5]:
```
Deal ID           False
Units Sold        False
Billings          False
Start Date        False
Deal URL          False
Segment           False
Inventory Type    False
dtype: bool
```

In [6]:
```
u_0 = df[(df['Units Sold'] == 0) & (df['Billings'] != 0)]
u_p = df[(df['Units Sold'] > 0) & (df['Billings'] <= 0)]
u_n = df[(df['Units Sold'] < 0) & (df['Billings'] >= 0)]

print(u_0 .shape)
print(u_p.shape)
print(u_n.shape)
```

```
(0, 7)
(0, 7)
(0, 7)
```

Data Adjustments

Creating New Metrics and Cleaning Outliers

Active Days = the number of days between the Start Date and 2014-01-01
Daily Billings = Billings / Active Days
Unit Prices = Billings / Unit Sold

In [7]:
```
d1 = pd.to_datetime('2014-01-01')
active_days = (d1 - df['Start Date'])
df.insert(4,'Active Days',active_days)

daily_billing = df['Billings'] / df['Active Days'].dt.days
df.insert(5,'Daily Billings',daily_billing)

unit_price = df['Billings'] / df['Units Sold']
df.insert(3,'Unit Prices',unit_price)
```

In [8]:
```
df.head()
```

Out[8]:

	Deal ID	Units Sold	Billings	Unit Prices	Start Date	Active Days	Daily Billings	Deal URL	Segment	Inventory Type
0	gr-millevois-tire-service-center	0.0	0.0	NaN	2011-11-21	772 days	0.000000	http://www.groupon.com/deals/gr-millevois-tire...	Local	Third - Party
1	gr-manakeesh-cafe-bakery	0.0	0.0	NaN	2011-11-21	772 days	0.000000	http://www.groupon.com/deals/gr-manakeesh-cafe...	Local	Third - Party
2	gr-phoenix-salon-and-spa	0.0	0.0	NaN	2011-11-21	772 days	0.000000	http://www.groupon.com/deals/gr-phoenix-salon-...	Local	Third - Party
3	gr-hands-in-motion	0.0	0.0	NaN	2011-11-21	772 days	0.000000	http://www.groupon.com/deals/gr-hands-in-motion	Local	Third - Party
4	dc-fd2-bartending-college-allentown-reading	86.8	4253.2	49.0	2012-06-06	574 days	7.409756	http://www.groupon.com/deals/dc-fd2-bartending...	Local	Third - Party

let's have a look at some basic descriptive statistics of new metrics:

In [9]:
```
unit_price_nonNaN = unit_price.dropna()
unit_price_nonNaN.describe()
```

Out[9]:
```
count    1.117600e+05
mean     1.120983e+14
std      1.365227e+16
min      1.000000e-02
25%      2.207685e+01
50%      3.900000e+01
75%      6.500000e+01
max      3.461016e+18
dtype: float64
```

In [10]:
```
daily_billing.describe()
```

Out[10]:
```
count    138534.000000
mean        159.823401
std        1284.711307
min       -1503.677299
25%           0.507850
50%           8.046202
75%          56.402327
max      229885.000000
dtype: float64
```

The statistics of Daily Billings look fine. But we can find the mean, standard deviation and maximum of Unit Prices are extremely high. This is abnormal for online deals.

In [11]:
```
q1 = unit_price_nonNaN.quantile(0.25)
q3 = unit_price_nonNaN.quantile(0.75)
iqr = q3 - q1
min_unit_price = q1 - 1.5*iqr
max_unit_price = q3 + 1.5*iqr
```

In [12]:
```
min_unit_price
```

Out[12]:
```
-42.307882837110036
```

In [13]:
```
max_unit_price
```

Out[13]:
```
129.384729702266
```

In [14]:
```
unit_price_nonNaN.quantile(0.999)
```

Out[14]:
```
2898.9999999999995
```

If we use boxplot to find outliers in Unit Prices, the lower boundary is Q1 -1.5*IQR = -42.31, the upper boundary is Q3 + 1.5*IQR = 129.38. The upper boundary is too strict for our data set, it's pretty normal that the price of a product is higher than $130.

I set 100,000 as the upper boundary for Unit Prices, which means if the unit price of a deal is greater than 100,000, the deal will be regarded as an outlier.

As we can take the Units Sold as given, the only part we can adjust is Billings. For the outliers, we can use the URL to find the true Unit Prices. But the outliers' sold units are extremely small. If I adjust the outliers' Billings by multiplying their Units Sold and true Unit Price, I'll get some very small Billings, which won't have a big impact on the Total Gross Billings. So, I directly deleted the outliers.

In [15]:
```
df_outliers = df[(df['Unit Prices'] > 100000)]
df_cleaned = df.append(df_outliers)
df_cleaned = df_cleaned.drop_duplicates(keep=False)
```

Adjustment for Missing Deals on Oct 20-30, 2013 of the Local Segment

Total Billings of the Missing Deals on Oct 20-30, 2013
= sum(Total Billings of the Missing Deals on Each Day)
= sum(Numbers of New Deals Launched on Each Day *Active Days* Average Daily Billings)

In [16]:
```
local = df_cleaned[(df_cleaned['Segment'] == 'Local')]
```

In [17]:
```
local['Daily Billings'].describe()
```

Out[17]:
```
count    120532.000000
mean         72.654963
std         510.974406
min        -858.515358
25%           0.237636
50%           5.773019
75%          34.153603
max       75000.000000
Name: Daily Billings, dtype: float64
```

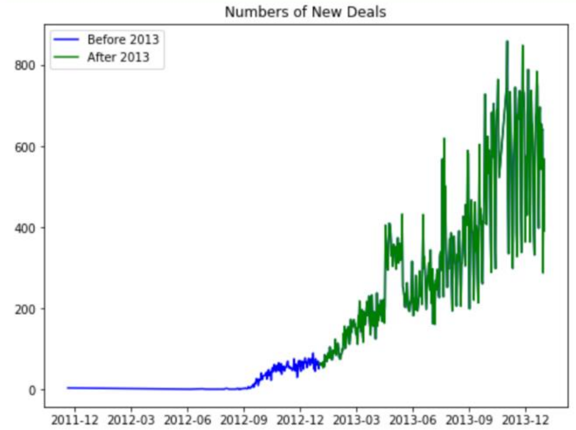
Numbers of New Deals (Active in 4Q13) Launched on Each Day

In [18]:
```
number_new_deals = local.groupby(['Start Date']).count()
number_new_deals = number_new_deals['Deal ID']
number_new_deals_2013 = number_new_deals[150:]
```

In [19]:
```
plt.figure(figsize=(8,6))
plt.plot(number_new_deals, color='blue')
plt.plot(number_new_deals_2013, color='green')
#count.plot()
#count2013.plot()
plt.legend(["Before 2013","After 2013"],loc='upper left')
plt.title('Numbers of New Deals' )
plt.show()
```

```
C:\Users\MXX\Anaconda3\lib\site-packages\pandas\plotting\_converter.py:129: FutureWarning: Using an implicitly registered datetime converter
for a matplotlib plotting method. The converter was registered by pandas on import. Future versions of pandas will require you to explicitly
register matplotlib converters.

To register the converters:
	>>> from pandas.plotting import register_matplotlib_converters
	>>> register_matplotlib_converters()
  warnings.warn(msg, FutureWarning)
```



In [20]:
```
df_number_new_deals_2013 = {'Date':number_new_deals_2013.index,'Count':number_new_deals_2013.values}
df_number_new_deals_2013 = pd.DataFrame(df_number_new_deals_2013)
```

In [21]:
```
month = df_number_new_deals_2013['Date'].dt.month
weekday = df_number_new_deals_2013['Date'].dt.weekday
df_number_new_deals_2013.insert(2,'Month', month)
df_number_new_deals_2013.insert(2,'Weekday', weekday)
```

In [22]:
```
dummy_month = pd.get_dummies(df_number_new_deals_2013['Month'],prefix='Month' )
dummy_weekday = pd.get_dummies(df_number_new_deals_2013['Weekday'],prefix='Weekday')
df_number_new_deals_2013 = pd.concat([df_number_new_deals_2013,dummy_month,dummy_weekday],axis = 1)
df_number_new_deals_2013 = df_number_new_deals_2013.drop(columns=['Date','Month','Weekday'])
```

In [23]:
```
df_number_new_deals_2013.head()
```

Out[23]:

	Count	Month_1	Month_2	Month_3	Month_4	Month_5	Month_6	Month_7	Month_8	Month_9	Month_10	Month_11	Month_12	Weekday_0	Weekday_1	W
0	65	1	0	0	0	0	0	0	0	0	0	0	0	0	1	
1	61	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	65	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	65	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	60	1	0	0	0	0	0	0	0	0	0	0	0	0	0	

In [24]:
```
y = df_number_new_deals_2013.iloc[:,0]
x = df_number_new_deals_2013.iloc[:,1:]
```

In [25]:
```
model = sm.OLS(y, x).fit()
print (model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Count   R-squared:                       0.804
Model:                            OLS   Adj. R-squared:                  0.794
Method:                 Least Squares   F-statistic:                     80.85
Date:                Mon, 06 Apr 2020   Prob (F-statistic):          1.51e-107
Time:                        22:51:47   Log-Likelihood:                -2071.2
No. Observations:                 354   AIC:                             4178.
Df Residuals:                     336   BIC:                             4248.
Df Model:                          17
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Month_1     -130.7687     14.959     -8.742      0.000    -160.194    -101.344
Month_2      -82.1272     15.702     -5.230      0.000    -113.014     -51.240
Month_3      -34.4121     14.959     -2.300      0.022     -63.838      -4.986
Month_4       49.1255     15.196      3.233      0.001      19.233      79.018
Month_5       73.2103     14.959      4.894      0.000      43.785     102.635
Month_6       52.1212     15.196      3.430      0.001      22.229      82.013
Month_7       98.2431     14.960      6.567      0.000      68.816     127.670
Month_8      135.7584     14.958      9.076      0.000     106.335     165.182
Month_9      196.1321     15.197     12.906      0.000     166.238     226.026
Month_10     355.7012     18.514     19.213      0.000     319.284     392.119
Month_11     380.6263     15.196     25.049      0.000     350.736     410.517
Month_12     369.7807     14.960     24.717      0.000     340.353     399.208
Weekday_0    216.9112     11.698     18.542      0.000     193.900     239.922
Weekday_1    226.7600     11.570     19.600      0.000     204.002     249.518
Weekday_2    240.2291     11.680     20.567      0.000     217.254     263.204
Weekday_3    261.2792     11.565     22.593      0.000     238.531     284.028
Weekday_4    239.4109     11.570     20.693      0.000     216.653     262.169
Weekday_5    159.2362     11.570     13.763      0.000     136.477     181.995
Weekday_6    119.5644     11.698     10.221      0.000      96.553     142.575
==============================================================================
Omnibus:                       11.674   Durbin-Watson:                   1.140
Prob(Omnibus):                  0.003   Jarque-Bera (JB):               12.863
Skew:                           0.361   Prob(JB):                      0.00161
Kurtosis:                       3.593   Cond. No.                     4.60e+15
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 3.81e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [26]:
```
predict_days = pd.date_range(start='20131020', end='20131030')
predict_x = {'Date':predict_days}
predict_x = pd.DataFrame(predict_x)
predict_month = predict_x['Date'].dt.month
predict_weekday = predict_x['Date'].dt.weekday
predict_x.insert(1,'Month', predict_month)
predict_x.insert(1,'Weekday', predict_weekday)
```

In [27]:
```
predict_dummy_month = pd.get_dummies(predict_x['Month'],prefix='Month' )
predict_dummy_weekday = pd.get_dummies(predict_x['Weekday'],prefix='Weekday')
predict_x = pd.concat([predict_x,predict_dummy_month,predict_dummy_weekday],axis = 1)
predict_x = predict_x.drop(columns=['Date','Month','Weekday'])
```

In [28]:
```
predict_x.head()
```

Out[28]:

	Month_10	Weekday_0	Weekday_1	Weekday_2	Weekday_3	Weekday_4	Weekday_5	Weekday_6
0	1	0	0	0	0	0	0	1
1	1	1	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0
3	1	0	0	1	0	0	0	0
4	1	0	0	0	1	0	0	0

In [29]:
```
zero = np.zeros((11,1)).astype(np.uint8)
predict_x.insert(1,'Month_12', zero)
predict_x.insert(1,'Month_11', zero)
predict_x.insert(0,'Month_9', zero)
predict_x.insert(0,'Month_8', zero)
predict_x.insert(0,'Month_7', zero)
predict_x.insert(0,'Month_6', zero)
predict_x.insert(0,'Month_5', zero)
predict_x.insert(0,'Month_4', zero)
predict_x.insert(0,'Month_3', zero)
predict_x.insert(0,'Month_2', zero)
predict_x.insert(0,'Month_1', zero)
```

In [30]:
```
predict_x.head()
```

Out[30]:

	Month_1	Month_2	Month_3	Month_4	Month_5	Month_6	Month_7	Month_8	Month_9	Month_10	Month_11	Month_12	Weekday_0	Weekday_1	Weekday
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	
2	0	0	0	0	0	0	0	0	0	1	0	0	0	1	
3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	1	0	0	0	0	

In [31]:
```
y_predict = model.predict(predict_x)
```

Active Days

In [32]:
```
start_2014 = pd.to_datetime('2014-01-01')
days_to_end = (start_2014 - predict_days)
missing = {'Date':predict_days, 'Days to End':days_to_end.days, 'New Deals':y_predict}
missing = pd.DataFrame(missing)
```

In [33]:
```
missing
```

Out[33]:

	Date	Days to End	New Deals
0	2013-10-20	73	475.265556
1	2013-10-21	72	572.612366
2	2013-10-22	71	582.461159
3	2013-10-23	70	595.930277
4	2013-10-24	69	616.980364
5	2013-10-25	68	595.112094
6	2013-10-26	67	514.937369
7	2013-10-27	66	475.265556
8	2013-10-28	65	572.612366
9	2013-10-29	64	582.461159
10	2013-10-30	63	595.930277

Average Daily Billings

In [34]:
```
daily_billing_10_11 = local[(local['Start Date'] >= '2013-10-01')&(local['Start Date'] < '2013-12-01')]
daily_billing_10_11_mean = daily_billing_10_11['Daily Billings'].mean()
```

In [35]:
```
daily_billing_10_11_mean
```

Out[35]: 103.13455463038575

In [36]:
```
misssing_billings = missing['Days to End']*missing['New Deals']*daily_billing_10_11_mean
```

Billings Estimate by Segment

In [37]:
```
Local_billings = local['Billings'].sum() + misssing_billings.sum()

Goods = df_cleaned[(df_cleaned['Segment'] == 'Goods')]
Goods_billings = Goods['Billings'].sum()

Travel = df_cleaned[(df_cleaned['Segment'] == 'Travel')]
Travel_billings = Travel['Billings'].sum()

Total_billings = df_cleaned['Billings'].sum() + misssing_billings.sum()
```

In [38]:
```
print('Gross billings of Local Segment: ', Local_billings)
print('Gross billings of Goods Segment: ', Goods_billings)
print('Gross billings of Travel Segment: ', Travel_billings)
print('Total gross billings: ', Total_billings)
```

```
Gross billings of Local Segment:  452532531.1526092
Gross billings of Goods Segment:  282245469.10132
Gross billings of Travel Segment:  70547395.7245
Total gross billings:  805325395.9784293
```

In [39]:
```
Local_billings + Goods_billings + Travel_billings
```

Out[39]:
```
805325395.9784293
```

Thanks for reading!