Data Analytics for Microsoft Stock

> Mini Project
>
> **Surbhi Sonkiya** (EIT Digital Master School)

Description

To perform data analytics on the historical stock market dataset for Microsoft and to visualize price variation over several years.

This document presents how to generate Spark dataframe from the dataset which is stored in a csv file. It also presents queries used on spark dataframe to perform data analytics and extract some of the interesting facts from the dataset using spark libraries. In the end, it also shows various visualizations of the price variation of the stock over years.

Import libraries and create Spark Session

```
import findspark
findspark.init('/home/surbhi/Downloads/spark-2.2.0-bin-hadoop2.7')

from pyspark.sql import SparkSession
from pyspark.sql.functions import format_number
from pyspark.sql.functions import (dayofmonth, hour,
                                   dayofyear, month,
                                   year, weekofyear,
                                  format_number, date_format)

spark =
SparkSession.builder.master("local").appName("microsoft").config("spark.executor.memory
", "3g").getOrCreate()
```

Create spark dataframe and import dataset

`spark.read.csv` has been used to read the dataset which is present in the csv file named 'microsoftNew'.

```
df = spark.read.csv('microsoftNew.csv', inferSchema=True, header=True)
```

Use **`SparkSQL`** to view the dataset from the dataframe. To do so, create a temporary view named 'stock'.

```
df.createOrReplaceTempView("stock")
sqlStock = spark.sql("SELECT * FROM stock")
sqlStock.show()
```

Output

```
+-------------------+----------+----------+----------+----------+--------+----------+
|               Date|      Open|      High|       Low|     Close|  Volume| Adj Close|
+-------------------+----------+----------+----------+----------+--------+----------+
|2018-10-30 00:00:00|103.660004|104.379997|100.110001|103.730003|65350900|103.730003|
|2018-10-29 00:00:00|108.110001|108.699997|101.629997|103.849998|55162000|103.849998|
|2018-10-26 00:00:00|105.690002|    108.75|104.760002|106.959999|55523100|106.959999|
|2018-10-25 00:00:00|106.550003|109.269997|106.150002|108.300003|61646800|108.300003|
|2018-10-24 00:00:00|108.410004|108.489998|101.589996|    102.32|63897800|    102.32|
|2018-10-23 00:00:00|107.769997|108.970001|105.110001|108.099998|43770400|108.099998|
|2018-10-22 00:00:00|    109.32|110.540001|108.239998|109.629997|26545600|109.629997|
|2018-10-19 00:00:00|    108.93|110.860001|108.209999|108.660004|32785500|108.660004|
|2018-10-18 00:00:00|110.099998|110.529999|107.830002|     108.5|32506200|     108.5|
|2018-10-17 00:00:00|    111.68|111.809998|109.550003|110.709999|26548200|110.709999|
|2018-10-16 00:00:00|109.540001|111.410004|108.949997|     111.0|31610200|     111.0|
|2018-10-15 00:00:00|108.910004|109.480003|106.949997|107.599998|32068100|107.599998|
|2018-10-12 00:00:00|109.010002|111.239998|107.120003|    109.57|47742100|    109.57|
|2018-10-11 00:00:00|105.349998|    108.93|104.199997|105.910004|63904300|105.910004|
|2018-10-10 00:00:00|111.239998|     111.5|105.790001|106.160004|61376300|106.160004|
|2018-10-09 00:00:00|111.139999|113.080002|110.800003|112.260002|26198600|112.260002|
|2018-10-08 00:00:00|111.660004|112.029999|109.339996|110.849998|29640600|110.849998|
|2018-10-05 00:00:00|112.629997|113.169998|110.639999|112.129997|29068900|112.129997|
|2018-10-04 00:00:00|114.610001|114.760002|111.629997|112.790001|34821700|112.790001|
|2018-10-03 00:00:00|115.419998|    116.18|    114.93|115.169998|16648000|115.169998|
+-------------------+----------+----------+----------+----------+--------+----------+
only showing top 20 rows
```

What are the column names?

To extract the names of all the columns present in the dataframe created from the dataset.

```
df.columns
```

Output

```
['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close']
```

What does the dataframe schema look like?

To learn more about each column present in the dataframe, i.e. to explore the schema of it. Schema shows the name of the every column, their respective datatype and if the column values could be null.

```
df.printSchema()
```

OR use `SparkSQL` to achieve the similar results.

```
spark.sql("SELECT * FROM stock")
```

Output

```
root
 |-- Date: timestamp (nullable = true)
 |-- Open: double (nullable = true)
 |-- High: double (nullable = true)
 |-- Low: double (nullable = true)
 |-- Close: double (nullable = true)
 |-- Volume: integer (nullable = true)
 |-- Adj Close: double (nullable = true)
```

Extract the first 5 columns.

This code extracts the top five records from the dataframe. Output shows the column name and their respective values being printed for the first five rows.

```
for line in df.head(5):
    print(line, '\n')
```

Output

```
Row(Date=datetime.datetime(2018, 10, 30, 0, 0), Open=103.660004, High=104.379997,
Low=100.110001, Close=103.730003, Volume=65350900, Adj Close=103.730003)

Row(Date=datetime.datetime(2018, 10, 29, 0, 0), Open=108.110001, High=108.699997,
Low=101.629997, Close=103.849998, Volume=55162000, Adj Close=103.849998)

Row(Date=datetime.datetime(2018, 10, 26, 0, 0), Open=105.690002, High=108.75,
Low=104.760002, Close=106.959999, Volume=55523100, Adj Close=106.959999)

Row(Date=datetime.datetime(2018, 10, 25, 0, 0), Open=106.550003, High=109.269997,
Low=106.150002, Close=108.300003, Volume=61646800, Adj Close=108.300003)

Row(Date=datetime.datetime(2018, 10, 24, 0, 0), Open=108.410004, High=108.489998,
Low=101.589996, Close=102.32, Volume=63897800, Adj Close=102.32)
```

Describe function

Describe function of dataframe is used to learn more about the dataframe.

```
df.describe().show()
```

Output

```
+-------+------------------+-----------------+-----------------+------------------+----
----------------+------------------+
|summary|              Open|             High|              Low|             Close|
         Volume|         Adj Close|
+-------+------------------+-----------------+-----------------+------------------+----
----------------+------------------+
|  count|              4738|             4738|             4738|              4738|
          4738|              4738|
|   mean|37.026357026593516|37.41019501477407|36.64333786196699|
37.02922352195016|5.7082246707471505E7|31.650932395947663|
| stddev|18.635902942222035|18.75697105849408|
18.4835945869743|18.632471980918663|3.1630636929271787E7|20.173438768548856|
|    min|              15.2|            15.62|            14.87|             15.15|
        7425600|         11.956082|
|    max|        115.419998|           116.18|           114.93|        115.610001|
      591052200|        115.610001|
+-------+------------------+-----------------+-----------------+------------------+----
----------------+------------------+
```

Format the output for Describe function

As the above output is not very user friendly to understand, presented below are the formatted values for the output of the `describe` function. Column values are now limited to show only up to two decimal places.

```
summary = df.describe()
summary.select(summary['summary'],
                 format_number(summary['Open'].cast('float'), 2).alias('Open'),
                 format_number(summary['High'].cast('float'), 2).alias('High'),
                 format_number(summary['Low'].cast('float'), 2).alias('Low'),
                 format_number(summary['Close'].cast('float'), 2).alias('Close'),
                 format_number(summary['Volume'].cast('int'),0).alias('Volume')
                ).show()
```

Output

```
+-------+--------+--------+--------+--------+-----------+
|summary|    Open|    High|     Low|   Close|     Volume|
+-------+--------+--------+--------+--------+-----------+
|  count|4,738.00|4,738.00|4,738.00|4,738.00|      4,738|
|   mean|   37.03|   37.41|   36.64|   37.03|       null|
| stddev|   18.64|   18.76|   18.48|   18.63|       null|
|    min|   15.20|   15.62|   14.87|   15.15|  7,425,600|
|    max|  115.42|  116.18|  114.93|  115.61|591,052,200|
+-------+--------+--------+--------+--------+-----------+
```

What has been the peak 'High'?

Use `SparkSQL` to find out what has been the highest value of the stock so far since the year 2000. The output shows the **maximum price** of the Microsoft stock **over past 18 years** was **116.18**. The price of the Microsoft stock **dropped to 101.93** as of **05-January-2019**.

```
spark.sql("SELECT max(High) FROM stock").show()
```

Output

```
+---------+
|max(High)|
+---------+
|   116.18|
+---------+
```

Which date had the peak 'High' in price?

This is to know which date was the *highest value of the stock* so far since the year 2000. The output shows it reached it's highest value so far on **03-October-2018**.

```
df.orderBy(df['High'].desc()).select(['Date']).head(1)[0]['Date']
```

Output

```
datetime.datetime(2018, 10, 3, 0, 0)
```

What is the mean of the 'Close' column?

This is to learn about the average closing price of the stock calculated as the average of the closing price from past 18 years. The output shows the **average closing price** for this stock is **37.0292 USD**.

```
from pyspark.sql.functions import mean
df.select(mean('Close')).show()
```

Output

```
+-----------------+
|       avg(Close)|
+-----------------+
|37.02922352195016|
+-----------------+
```

What is the max and min of the 'Volume' column?

Use `max` and `min` `SQL` functions to find out the maximum and minimum volume of stock traded so far.

As seen from the output, this stock is a popular choice among the investors. Even the minimum volume of the stock that has been traded in a day is a seven figure number.

```
from pyspark.sql.functions import min, max
df.select(max('Volume'),min('Volume')).show()
```

Output

```
+-----------+-----------+
|max(Volume)|min(Volume)|
+-----------+-----------+
|  591052200|    7425600|
+-----------+-----------+
```

How many days was the closing lower than avg. value of 'Close'?

The average closing value has been obtained above. Use it to find out the total number days the closing price of the stock has been lower than the average closing price of the stock in past ~6570 days.

```
df.filter(df['Close'] < 37.02922352195016).count()
```

Output

```
3414
```

What percentage of the time was the 'High' greater than threshold?

The threshold is the average closing price of the stock. The output shows that around 27% of time the price of the stock was higher than the average closing price of the stock.

```
df.filter('High > 38').count() * 100/df.count()
```

Output

```
27.226677923174336
```

What is the max 'High' per year?

This presents the maximum price of the stock each year from 2000 to 2018. The output shows that the stock price crossed 50 USD in the year 2014 and has ever since reached higher and higher in terms of price in the following years.

```
year_df = df.withColumn('Year', year(df['Date']))
year_new_df = year_df.groupBy('Year').max()['Year', 'max(High)']
year_new_df.show()

#all the years in the dataset
#to save the output in the csv file
year_new_df.coalesce(1).write.option("header", "false").csv("year.csv")
```

Output

```
+----+---------+
|Year|max(High)|
+----+---------+
|2003|     30.0|
|2007|     37.5|
|2018|   116.18|
|2015|56.849998|
|2006|    30.26|
|2013|    38.98|
|2014|50.049999|
|2004|30.200001|
|2012|32.950001|
|2009|     31.5|
|2016|64.099998|
|2001|38.075001|
|2005|    28.25|
|2000|  59.3125|
|2010|    31.58|
|2011|29.459999|
|2008|35.959999|
|2017|     87.5|
|2002|35.310001|
+----+---------+
```

What is the average Close for each Calendar Month? i.e. across all the years, what is the average Close price for Jan, Feb, Mar, etc.

This presents the average 'Close' price of the stock for each month in a year from 2000 to 2018. In the output, Month value '1' stands for 'January', '2' for 'February' and so on. The average closing price of the stock across all months is around 38 USD.

```
import pandas as pd

#All the months from all the years in the dataset
#Create a new column Month from existing Date column
month_df = df.withColumn('Month', month(df['Date']))

#month_df.show(n=500)
#Group by month and take average of all other columns
month1_df = month_df.groupBy('Month').mean()
```

```
#Sort by month
month1_df = month1_df.orderBy('Month')

#Display only month and avg(Close), the desired columns
month_new_df = month1_df['Month', 'avg(Close)']
month_new_df.show()
```

Output

```
+-----+------------------+
|Month|        avg(Close)|
+-----+------------------+
|    1|37.235892825974005|
|    2|36.358448646575326|
|    3|36.361752183132545|
|    4| 36.40331635714283|
|    5| 36.83652537376236|
|    6| 37.03310339901478|
|    7| 37.53589384999999|
|    8| 38.04809403073286|
|    9| 37.73058580628271|
|   10|38.300537064439126|
|   11| 36.30581183423913|
|   12|35.962892548812654|
+-----+------------------+
```

What is the max High for each Calendar Month? i.e. across all the years, what is the average Close price for Jan, Feb, Mar, etc.

This presents the maximum 'High' price of the stock for each month in a year from 2000 to 2018. It can be seen that the highest price of the stock reached in the month of October, followed by the months September, August and July respectively. It can also be noted that month of November is a low price month.

```
#Max high in a month
month_new1_df = month_df.groupBy('Month').max()['Month', 'max(High)']
month_new1_df.show()

#to save the output in the csv file
month_new1_df.coalesce(1).write.option("header", "false").csv("month.csv")
```

Output

```
+-----+----------+
|Month| max(High)|
+-----+----------+
|   12|      87.5|
|    1| 95.449997|
|    6|102.690002|
|    3| 97.239998|
```

```
|    5| 99.989998|
|    9|115.290001|
|    4| 97.900002|
|    8|112.779999|
|    7|111.150002|
|   10|    116.18|
|   11| 85.059998|
|    2|     96.07|
+-----+----------+
```

Feature Engineering

HV Ratio: HV Ratio is the ratio of the 'High' price versus 'Volume' of the stock traded for a day. This is a new feature that has been added and a new dataframe has been created from the existing dataframe. This feature is only to understand the concept of Feature Engineering.

```
df_hv = df.withColumn('HV Ratio', df['High']/df['Volume']).select(['HV Ratio'])
df_hv.show()
```

Output

```
+--------------------+
|            HV Ratio|
+--------------------+
|1.597223557747483...|
|1.970559388709618...|
|1.958644239964987...|
|1.772516935185605...|
|1.697867500915524...|
|2.489582023467914...|
|4.164155302573685E-6|
|3.381372893504750...|
|3.400274378426269...|
|4.211584890877724E-6|
|3.524495384401237...|
|3.413984707544257E-6|
|2.330018956015759...|
|1.704580129975604...|
|1.816662131800059...|
|4.316261250601177E-6|
|3.779613064512864E-6|
|3.893164103216841E-6|
|3.295646163168369E-6|
|6.978616049975974E-6|
+--------------------+
only showing top 20 rows
```

Visualization using matplotlib

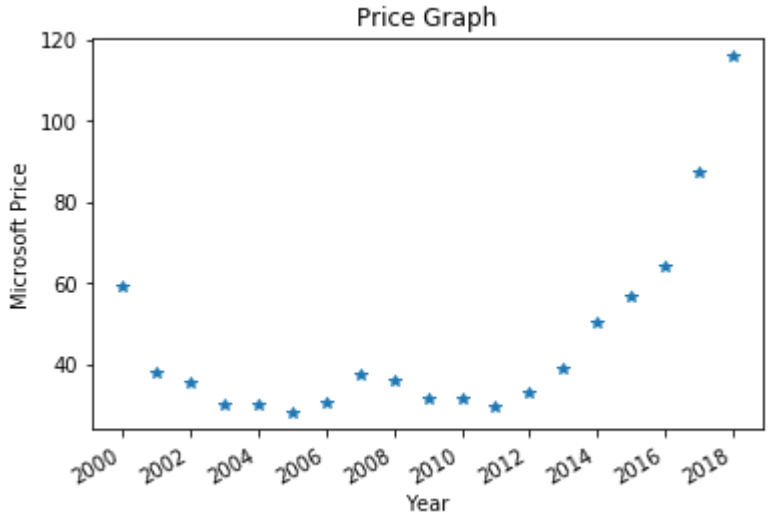
Below code shows the use of matplotlib library to plot the **graph** of **price vs year** to visualize the variation in price over several years.

Please note that the maximum 'High' price of the stock in a year is taken as the price of the stock for that particular year to plot in the graph.

```
from __future__ import print_function
from matplotlib.dates import strpdate2num
import numpy as np
from pylab import figure, show
import matplotlib.cbook as cbook

def bytespdate2num(fmt, encoding='utf-8'):
    strconverter = strpdate2num(fmt)
    def bytesconverter(b):
        s = b.decode(encoding)
        return strconverter(s)
    return bytesconverter


datafile = cbook.get_sample_data('/home/surbhi/Downloads/sitc-
master/year.csv/yearFloat.csv', asfileobj=False)
#print('loading', datafile)

dates, closes = np.loadtxt(
    datafile, delimiter=',',
    converters={0:bytespdate2num('%Y')},
    skiprows=0, usecols=(0,1), unpack=True)

fig = figure()
ax = fig.add_subplot(111)
ax.plot_date(dates, closes, '*')
ax.set_xlabel('Year')
ax.set_ylabel('Microsoft Price')
ax.set_title('Price Graph')
fig.autofmt_xdate()
```

Output

It is clearly visible that the price of Microsoft stock took a rise from the year 2014 and that the price only reached new higher values in the following years. Interestingly, the price for the stock was comparatively high enough even in the year 2000. However, the max 'High' price dropped tremendously between years 2001 to 2013.



Visualization using D3

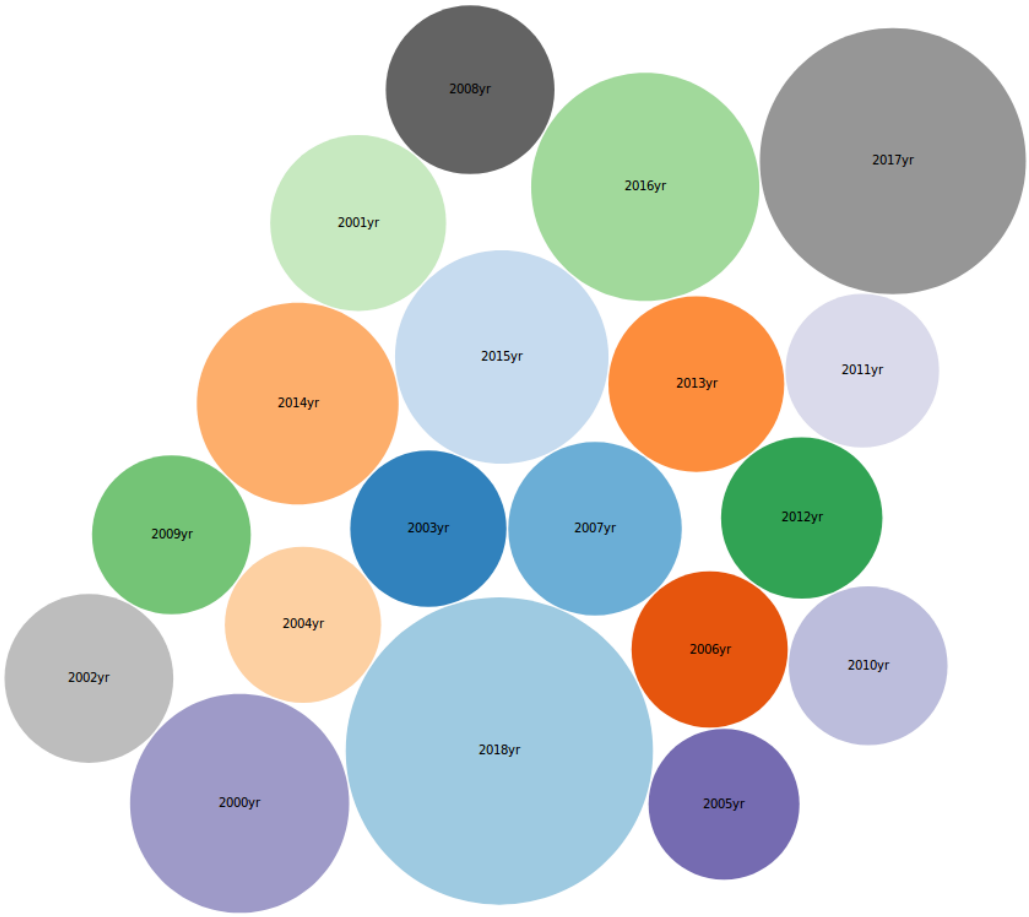
Bubble Output

The bubble plot shows the variation of price over years. The bigger the size of the bubble, the higher the max 'High' price of the stock for that year comparing to the other years.

As observed, for the year 2000, the price was higher than the price from years 2001-2013. Therefore, the size of the bubble for the year 2000 is bigger than other bubbles for the years 2001-2013. Gradually, the max'High' price kept increasing since year 2014 and so is the size of the bubble.

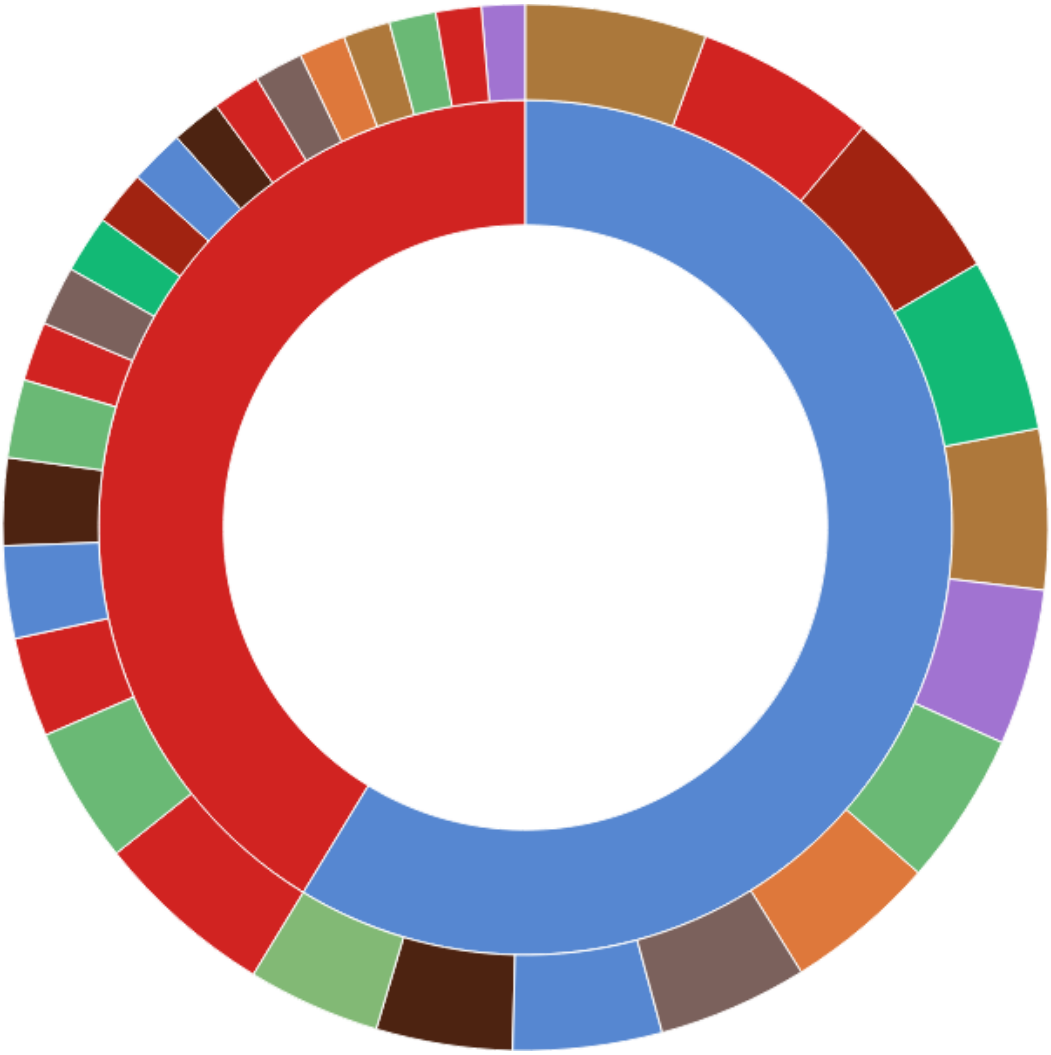
The highest max 'High' price was in the year 2018 and hence, the biggest bubble.

The lowest max 'High' price was in the year 2011 and hence, the smallest bubble.



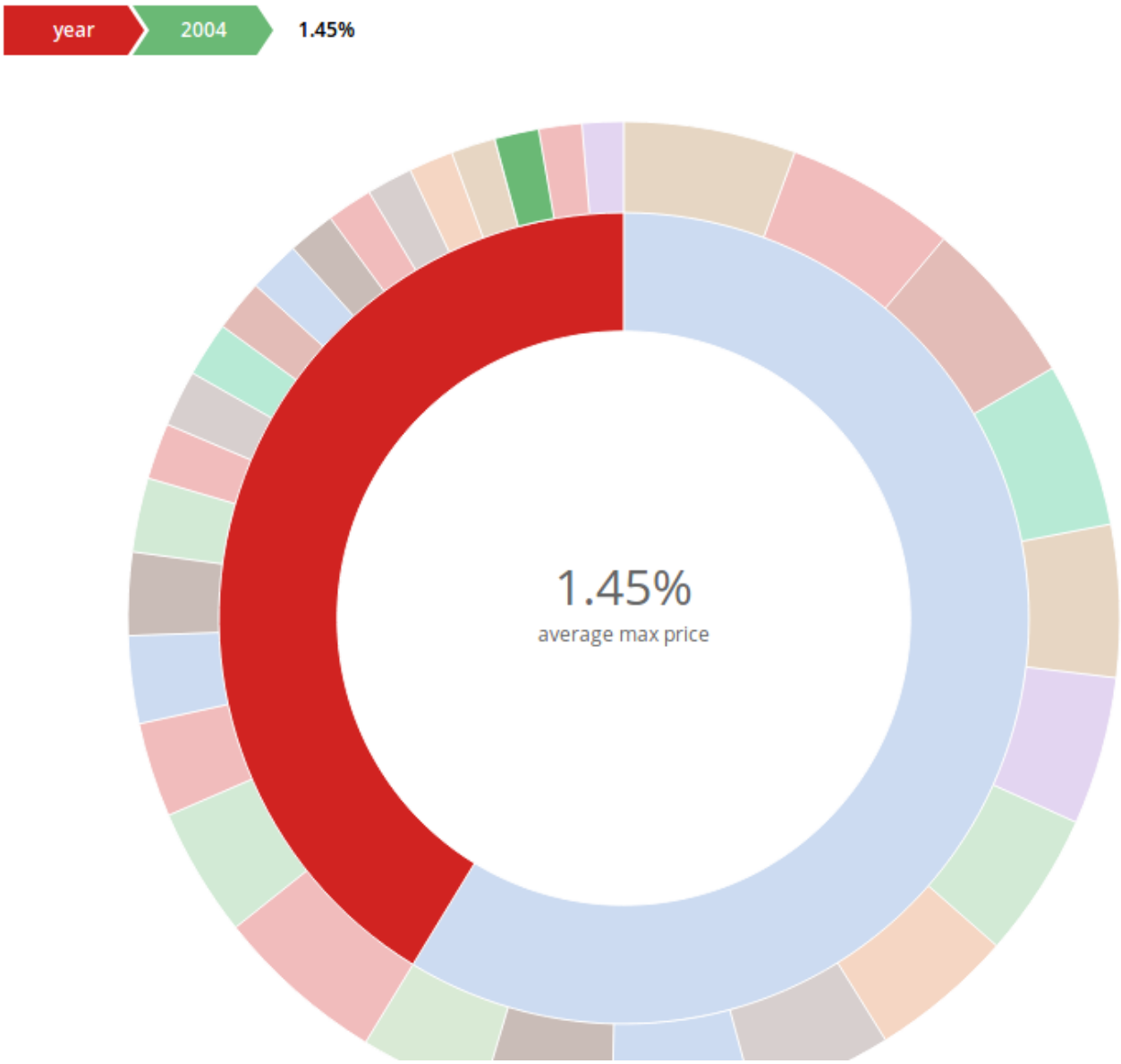
Sunburst Output

Presented below is another representation, a combination of the variation of price over years and variation of price over months. In the inner most circle, red part represents variation of price over years and blue part represents variations of price over months. The outer circle, above year (red colour from inner circle) represents average of price across 18 years where as above month (blue colour from inner circle) represents average of price across 12 months of a year.



Sunburst Output for Year

The below graph shows the average percentage of max 'High' price for a particular year from the dataset of max 'High' price of past 18 years. The selected part of the graph shows this value 1.45% for the year 2004.



Sunburst Output for Month

The below graph shows the average percentage of max 'High' price for a particular month from the dataset of max 'High' price of past 216 months. The selected part of the graph shows this value 5.44% for the month of August.

Month Aug 5.44%