

BIG DATA ANALYSIS OF FUTURES: PENALIZED REGRESSION SPLINES  
FOR TRADE VOLUME PREDICTION AND PRICE VOLATILITY VS TRADE  
VOLUME RELATIONSHIP

by

Aniver Oluwatobi Bosede

A thesis submitted to Johns Hopkins University in conformity with the  
requirements for the degree of Master of Science in Engineering

Baltimore, Maryland

© 2017 Aniver Oluwatobi Bosede

All Rights Reserved

# Abstract

# Acknowledgments

# Contents

<b>Abstract</b>	<b>ii</b>
<b>Acknowledgments</b>	<b>iii</b>
<b>List of Tables</b>	<b>v</b>
<b>List of Figures</b>	<b>vi</b>
<b>Introduction</b>	<b>1</b>
<b>Methods</b>	<b>3</b>
<b>Results</b>	<b>11</b>
<b>Discussion</b>	<b>17</b>
<b>Appendices</b>	<b>21</b>
A: Spark/python code for raw data generation . . . . .	21
B: Spark/python code for creating datasets to be analyzed . . . . .	23
C: Python code for exploratory data analysis . . . . .	26
D: R code for spline regression for trade volume . . . . .	28
E: R code for price volatility and trade volume relationship . . . . .	32
<b>Bibliography</b>	<b>35</b>
<b>Curriculum Vitae</b>	<b>36</b>

## List of Tables

Predictors used in models . . . . .	8
Linear models comparison at market level . . . . .	11
Multiple linear models comparison at instrument level . . . . .	12
Significant p-values at FDR=.2 . . . . .	13

## List of Figures

Histograms of trade volume before and after Box-Cox transform . . . . .	4
Plot of trade volume as days pass and as time to maturity gets further. . .	4
Histograms for instrument types . . . . .	5
Plot of trade volume as days pass for instrument types . . . . .	6
Plot of trade volume as time to maturity gets further for instrument types	7
Correlation (top) and trade volume (bottom) for NQ maturing 2016-12-16	9
Correlation (top) and trade volume (bottom) for ZC maturing 2016-07-14 .	10
Spline curves for best model . . . . .	12
Uniform Q-Q plot . . . . .	13
Correlation regression plots for most significant products . . . . .	14
Price volatility vs. trade volume shows positive correlation . . . . .	15
Price volatility vs. trade volume may show two clusters . . . . .	15
Price volatility vs. trade volume shows two clusters . . . . .	15
Price volatility vs. trade volume hourly & daily show different correlations	16
Price volatility vs. trade volume hourly & daily show different clusters . .	16
Price volatility vs. trade volume daily may show negative correlation . . .	16
Price volatility vs. trade volume hourly shows three clusters . . . . .	17
Volume plots for significant products, dashes denote 60th-90th percentiles .	20

# Introduction

Technology has advanced so far that in our society today we are constantly collecting data [1]. This has created the issue of how to feasibly analyze such overwhelming amounts of data [1]. We can then consider how to efficiently store it or how to best carry out statistical computations [9]. The aforementioned together form the relatively new and evolving field of big data. The research here focuses on the analysis of hundreds of million of rows of futures trading data with the aid of Apache Spark (Spark).

Spark is a fault-tolerant and general-purpose cluster computing system providing APIs in Java, Scala, Python, and R [5]. Spark was chosen for the analysis over the similarly popular Hadoop MapReduce (MapReduce) because of Spark's performance advantages as well as its greater computational capabilities [8]. Not only does Spark cache data resulting in persisted in-memory manipulations [7], it also includes Structured Query Language (SQL) and MLlib, a machine learning library [5]. Conversely, MapReduce only does manipulations via disk reads and thus does not allow for data sharing [9]. Furthermore for a typical pipeline, external systems would have to be combined with MapReduce to provide querying and machine learning functionality [9]. In this case, A stand alone set-up was employed, meaning that analysis was done using a one node cluster or one machine. Out of convenience the local file system was used for storage as opposed to a database like Cassandra or HDFS (Hadoop Distributed File System). However, for long term data analysis, it would be worthwhile to invest the time needed to set up a more robust data storage system.

The futures trading data come from the Chicago Mercantile Exchange (CME) and were collected from May 2, 2016 to November 18, 2016. Raw data from the CME included extended hours trading and was collected via the Trading Technologies X\_TRADER® API RTD (Real Time Data) server. The server returned raw records with instrument name, maturity, date, time stamp, price, and quantity fields. The

futures were comprised of 21 financial instruments spanning six markets - foreign exchange, metal, energy, index, bond, and agriculture recording roughly a trade every half second. First, this work uses spline regression to predict the volume of trading for any given day. Volume during a particular time period is taken to mean the number of units traded. A spline regression was chosen due to lack of knowledge regarding the likely non-linear function underlying the response to covariates, in particular time to maturity. Predicting trade volume is of interest because many trading algorithms depend on volume [6]. Additionally, accurate volume predictions over a given interval allows traders to be more effective [6]. In general, volume prediction increases trading strategy capacity, controls trading risk, and manages slippage [6].

Second, the relationship between price volatility and trade volume is explored using standard deviation as a measure of volatility. In particular, volatility vs daily volume and volatility versus hourly volume were plotted to see whether or not the correlation remained the same with the passing of days and hours. It should be noted that unpredictable volume shocks have been known to be more predictive of change in volatility than predictable volume changes [2]. This volatility-volume relationship is of importance due to the notion that hedgers are motivated to trade futures to stabilize their future income flows or costs, wherein the volume of their trading is based on their expectation of price variability [3]. Likewise, speculators are motivated to trade futures based on expectations of price variability [3]. Due to the fact that new information on the market causes agents such as hedgers and speculators to trade until prices reach a revised equilibrium which then changes price and trading volume, we expect a positive correlation between volatility and volume [3]. Indeed past research indicates that there is a positive relationship between volume and price volatility [3]. This sort of exploration provides information on the efficiency of futures markets which regulators can then use to decide upon market restrictions [3].



## Methods

A cubic regression spline was thought to be appropriate for modeling trade volume. Spline regression derives its name from a draftsman’s spline which is a flexible strip of metal or rubber used to draw curves [4]. Similarly, spline basis functions are piecewise polynomials used in fitting curves which are linear in terms of the basis function. Splines have been used, principally in the physical sciences as well as in biomedicine, to approximate a wide variety of functions [4]. Cubic splines in particular have been found to have nice properties with good ability to fit nonlinear curves. Cubic splines are made to be smooth at the knots, endpoints of intervals on the x-axis, by forcing the first and second derivatives of the function to agree at the knot [4].

Holidays were removed from the raw data. Then the day of the month, day of the week, and hour of the trade were extracted from the time stamp. An aggregation was then done to sum the number of trades per hour for each product, where product is defined as an instrument-maturity pair. There were 139 such products. Aggregation reduced the data from 105 million records to 8,826 records. Day, time to maturity, and market fields were created and total trade volume for each day was calculated.

Exploratory analysis was then done on the reduced data set. To ensure that a spline regression was appropriate for modeling trade volume, the first thing done was to create histograms of the trade volumes. One of the assumptions behind regression is that the response conditioned on the predictors is normally distributed. Even if normality fails, regression can be an effective tool, but under normality least squares is optimal. Thus transforming to normality is desirable. The histogram of the raw trade volume was skewed as shown in left of Figure 1. Therefore the volume was Box-Cox transformed, after which the data became normal as shown in right of Figure 1.

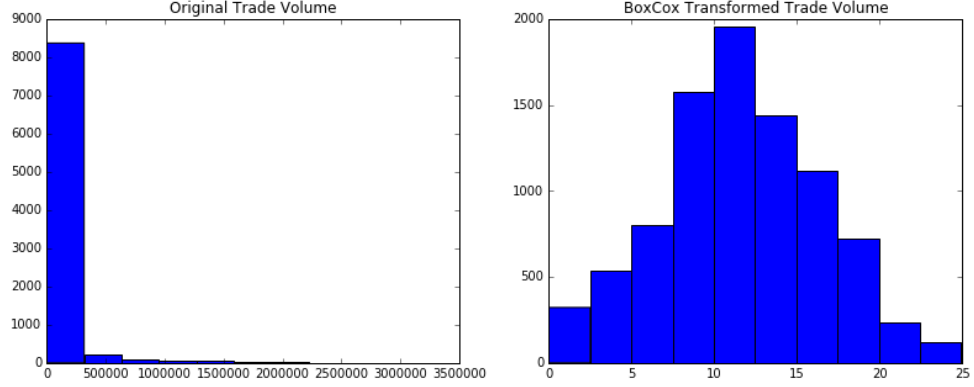


Figure 1: Histograms of trade volume before and after Box-Cox transform

There was curiosity regarding how trade volume changed as time passed. Thus a plot of the transformed trade volume versus day (ignoring weekends) was made. Referring to the left plot in Figure 2, the trade volume appears constant across the days with a cluster of high volume trades above 20, which untransformed is 412,823 trades. This is almost half a million trades of a single product in a single day.

Also intuitively, it makes sense that less trading occurs far from maturity and near maturity. Far from maturity speculators might not have any information that would move them to purchase a future and most hedgers may only seek to minimize risk in the short term. Then near maturity traders are closing their positions. To confirm

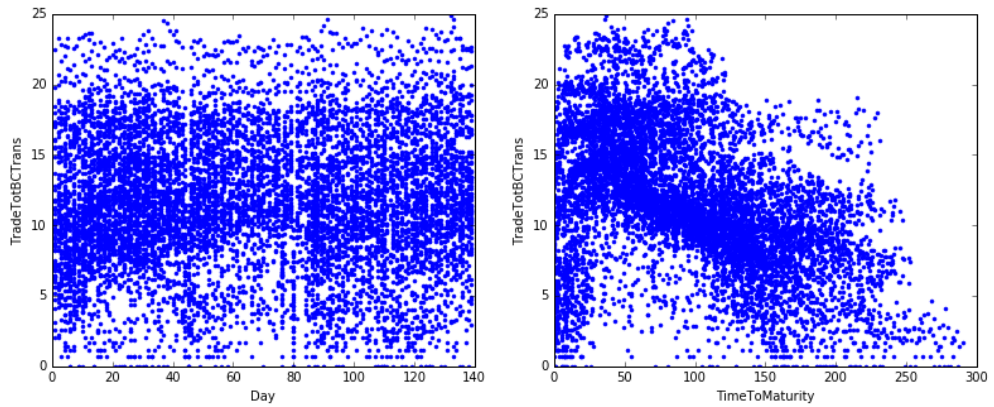


Figure 2: Plot of trade volume as days pass and as time to maturity gets further.

this theory a plot was made of the transformed trade volume versus time to maturity, right in Figure 2. Looking at the plot it appears that the data follows intuition.

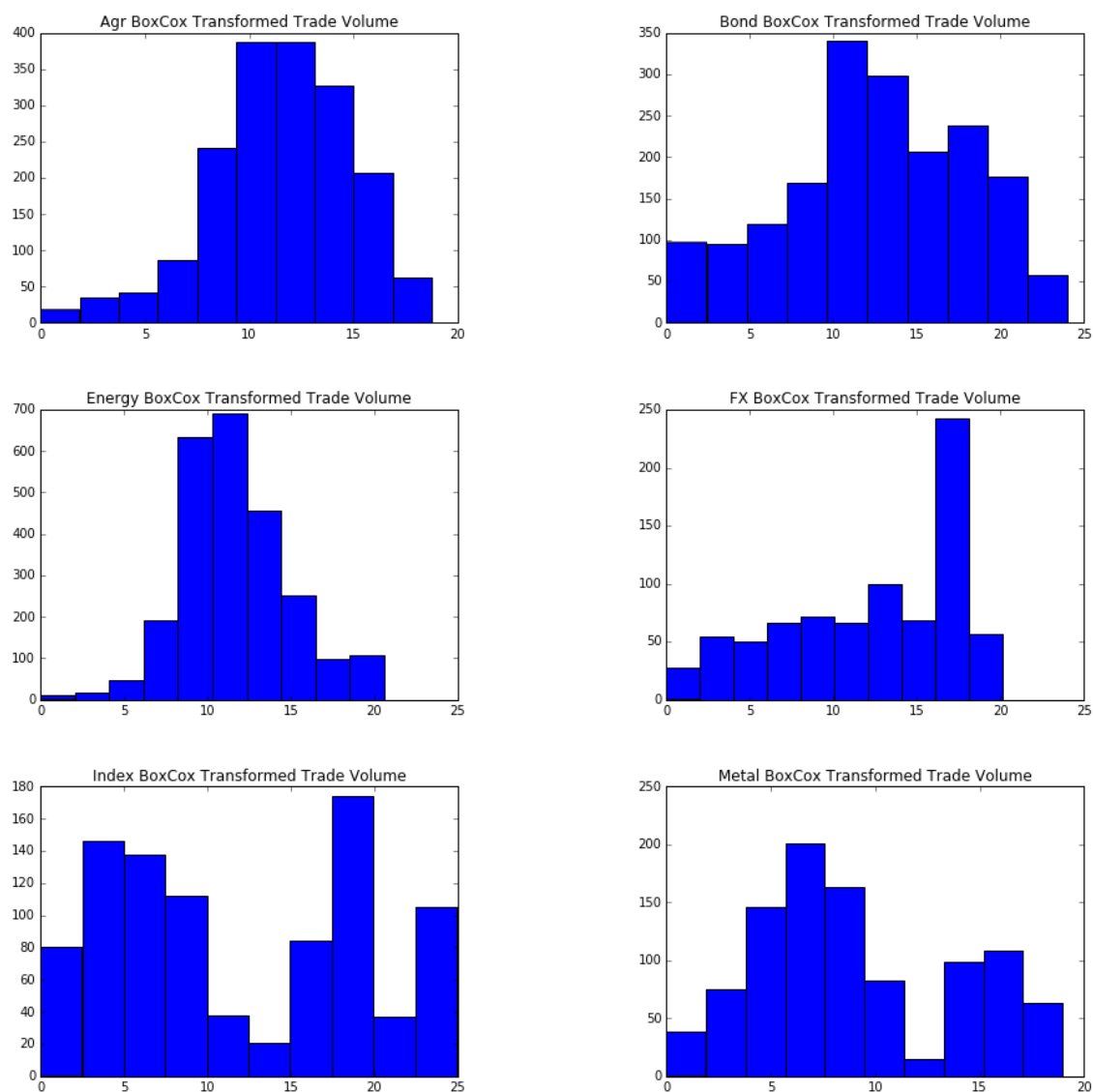


Figure 3: Histograms for instrument types

Whether it made sense to make one model or several models for each instrument market: foreign exchange, metal, energy, index, bond, and agriculture was also considered. This would mean that each of the markets need to have near normal distributions. In Figure 3 it can be seen that agriculture, bond, and energy are approximately normally distributed after Box-Cox transformation, but foreign exchange, index, and

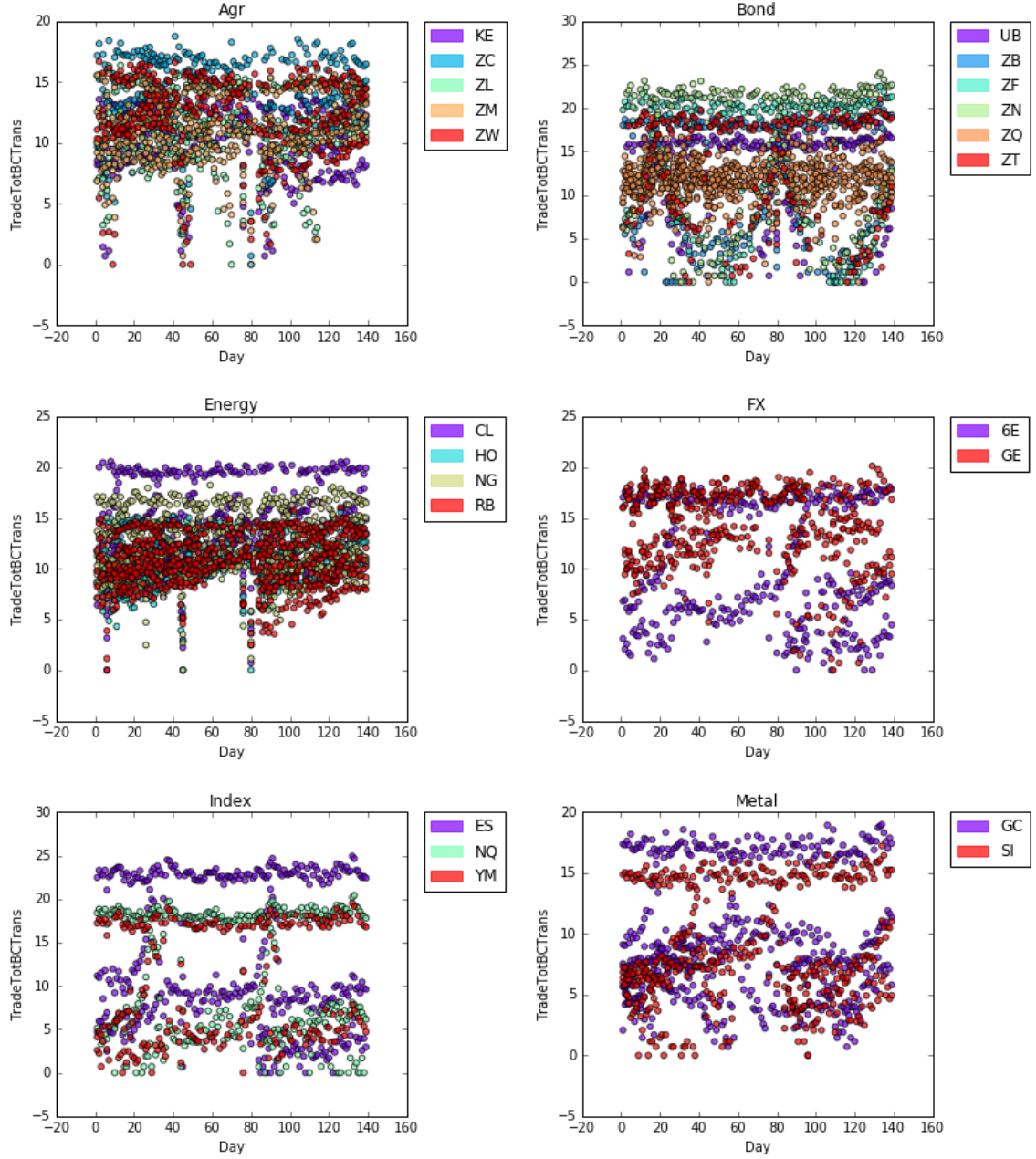


Figure 4: Plot of trade volume as days pass for instrument types

metal are not. Therefore we expect higher error in the spline model of the latter compared to the former group of instruments. The trend in daily trade volume as time passed was also explored for each instrument market in Figure 4. These points were then color-coded to understand which instruments comprised the various clusters. It is clear that low volumes of trading occur for metal and agriculture markets compared

with bond and index markets which appear to be more liquid.

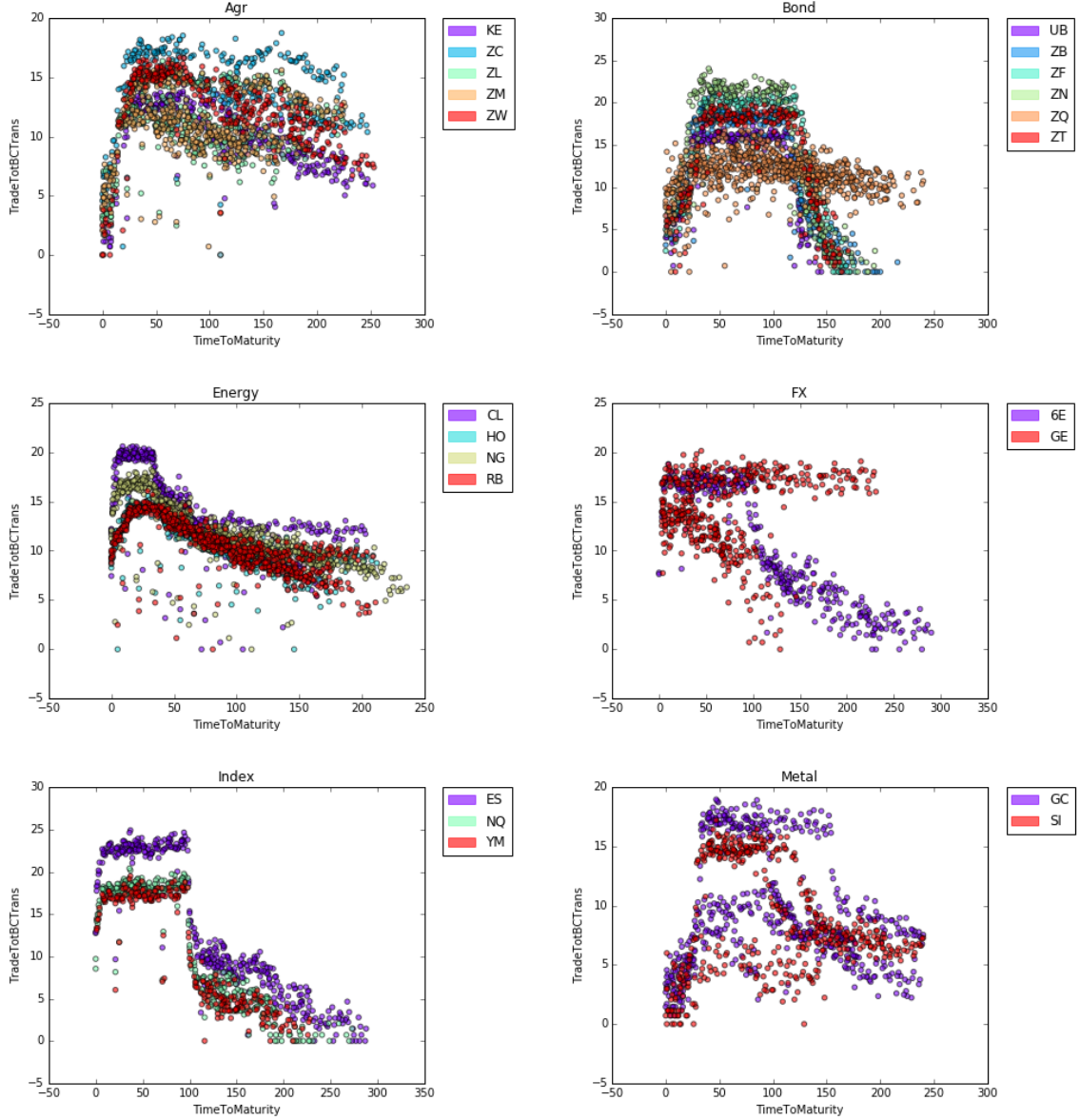


Figure 5: Plot of trade volume as time to maturity gets further for instrument types

Finally, in Figure 5, the trend in daily trade volume as time to maturity increased was visualized for each market. Like the histograms, agriculture, bond, and energy plots follow the general trend of low trade volume far from the maturity and close to maturity, but foreign exchange, index, and metal do not. There is some clustering in the index and metal plots as well which seems in line with the trimodal and bimodal

histograms respectively in Figure 3. This clustering is also consistent with plots in Figure 4.

The above described exploratory analysis deemed spline regression reasonable for trade volume prediction. Thus, both one linear model and multiple linear models paradigms were considered, two versions for each, making four models in total as shown in Table 1. All models under consideration were linear. One model paradigm employed a single model for all markets, whereas in the multiple models paradigm each market had its own model. Spline variables are denoted by “s”. The models were fit on data from May 2, 2016 to August 2, 2016 and tested or forecasted on data from August 2, 2016 through November 18, 2016. The mean absolute deviation (MAD) was then calculated for each model to compare the errors of forecasted volumes in a robust manner. MAD is defined as:  $\sum_{i=1}^N \frac{|\hat{Y}_i - Y_i|}{N}$ . The models were penalized with an integrated square second derivative cubic spline. This amounted to a natural spline and so generalized cross validation was employed to find an optimal smoothing parameter. The knots were placed at fixed intervals.

Table 1: Predictors used in models

One Linear Model		Multiple Linear Models	
Model 1'	Model 1	Model 2'	Model 2
s(TimeToMaturity) s(DayofMonth) Market DayOfTheWeek	s(TimeToMaturity) s(DayofMonth) Market DayOfTheWeek Instrument	s(TimeToMaturity) s(DayofMonth) DayOfTheWeek	s(TimeToMaturity) s(DayofMonth) DayOfTheWeek Instrument

An additional goal of this study was to understand the relationship between price volatility and trade volume. To this end, hourly aggregation similar to that which was done for trade volume was done for price volatility. Volatility was measured using standard deviation. Data was filtered for products with more 30 or more days of data. Next each product’s daily price was plotted against daily trade volume for the full

period over which the data was collected; that is from May to November. Hourly price volatility was plotted against its hourly trade volume, ~~was also plotted~~. Given past research, we expect a positive correlation between price and volume. Consistency of the correlation between price and volume over the days was of interest as well. This was investigated for each product by plotting daily correlations between hourly price volatility and hourly trade volume.

Observations with trade counts within the 60th to 90th percentiles were used to identify periods of normal trade volumes. In other words, to avoid the nascent and near maturity periods of low trading which would confound results, only records rep-

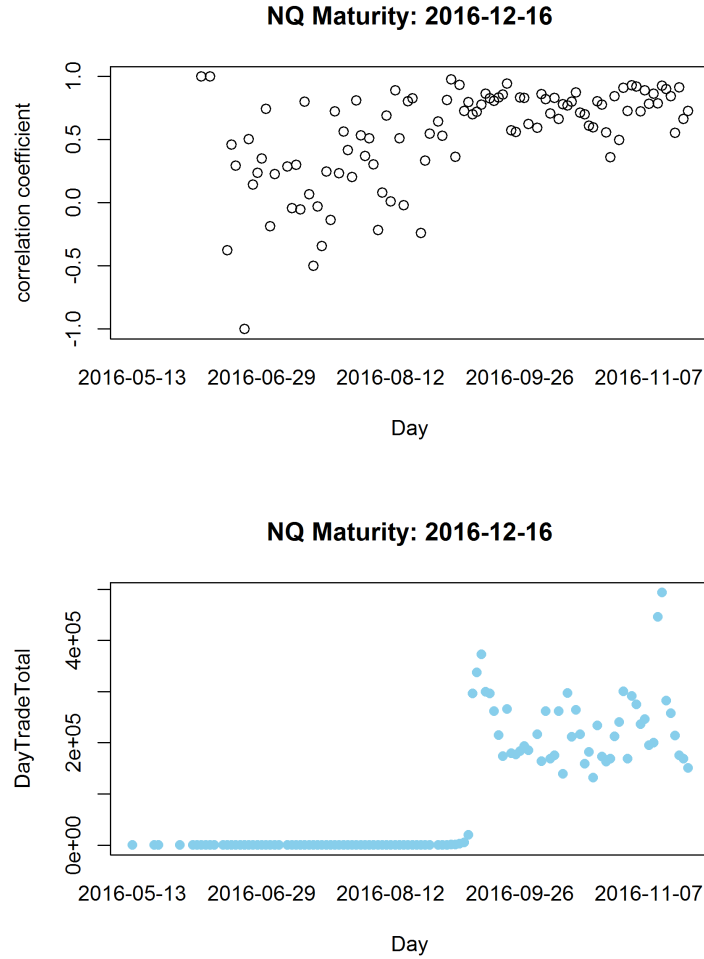


Figure 6: Correlation (top) and trade volume (bottom) for NQ maturing 2016-12-16

representing a product's active period were selected. The E-Mini Nasdaq 100 (NQ) with maturity 2016-12-16 is a product whose nascent period is captured in the time frame data was recorded, as demonstrated in Figure 6. From the top of Figure 6, it is clear that the correlation between price volatility and trade volume stabilizes around 2016-09-15. Looking at the bottom of Figure 6, 2016-09-15 is also the time that the daily trade volume makes a sharp increase from zero to around 100,000.

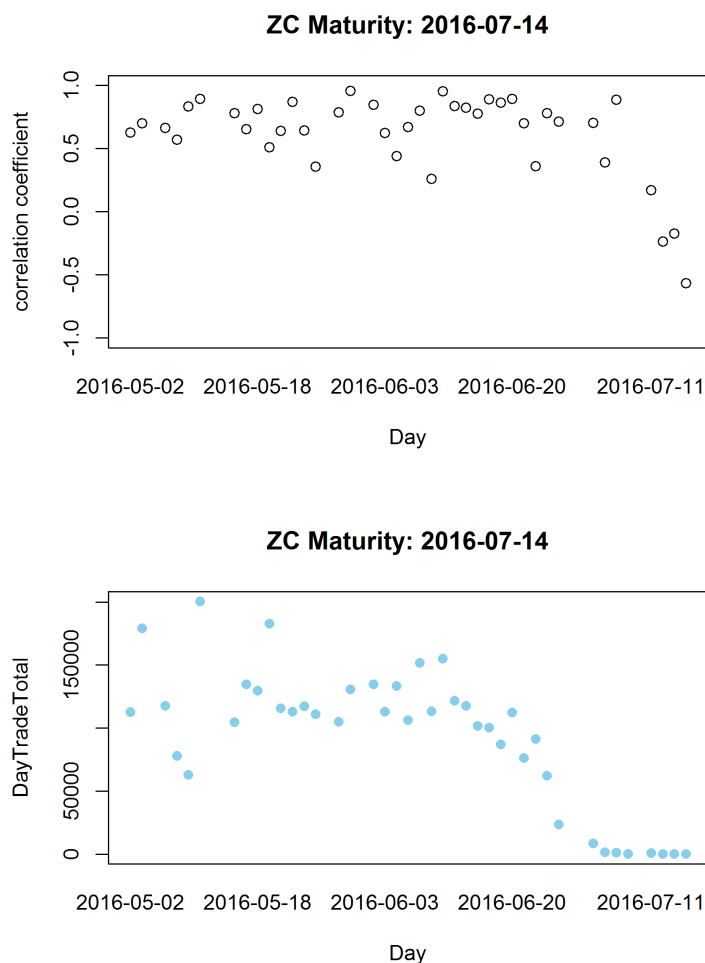


Figure 7: Correlation (top) and trade volume (bottom) for ZC maturing 2016-07-14

Corn (ZC) with maturity 2016-07-14 is a product whose near maturity period is captured in the time frame data was recorded, as demonstrated in Figure 7. Like



NQ, the top of Figure 7 shows that the correlation for ZC is constant until around 2016-07-01. Looking at the bottom of Figure 7, 2016-07-01 is also the time that the daily trade volume makes a sharp decrease from 65,000 to zero.

After records in a product’s active period were selected, linear regression was done on the correlation coefficient between the price volatility and trade volume to test whether the best fit line was constant or flat. A flat regression line would mean that the expected correlation between price volatility did not change with time.

## Results

The best linear regression model is Model 2 based on Table 2 since Model 2 has the lowest MAD across markets. Thus the multiple models paradigm is better than the one model paradigm.

Table 2: Linear models comparison at market level

	Market	Metal	Bond	Agr	Energy	Index	FX
	Median	7.98	12.512	11.579	11.751	9.705	12.478
One Model	Model 1’ MAD	3.537	3.742	2.261	1.958	5.103	4.223
	Model 1 MAD	3.603	3.615	1.809	1.818	5.455	4.08
Multiple Models	Model 2’ MAD	3.966	4.463	3.966	4.463	3.966	4.463
	Model 2 MAD	3.023	3.174	2.103	1.327	1.591	3.753

Referring to Table 3, MAD is lower for Model 2 than Model 2’ for the most part. Given its superiority at the market level, it is not surprising that model 2 is also superior at the instrument level.

Figure 8 shows that the time to maturity influences trade volume more significantly than day of the month since all the day of the month cubic spline curves hover around zero.

Table 3: Multiple linear models comparison at instrument level

Multiple Linear Models			
Instrument	Median	Model 2' MAD	Model 2 MAD
GC	8.1	3.417	2.914
SI	7.807	3.685	3.158
ZQ	11.944	2.159	3.066
ZT	16.55	4.229	2.5
ZF	19.266	5.222	3.095
ZB	12.466	4.665	3.313
UB	15.629	2.762	2.642
ZN	19.061	5.473	4.335
ZW	11.917	1.963	2.269
ZM	10.891	2.302	2.071
KE	10.514	1.76	2.009
ZC	13.935	3.303	2.202
ZL	11.099	2.139	2.004
CL	14.407	2.744	1.556
NG	11.549	1.929	1.26
HO	11.083	1.912	1.251
RB	10.907	1.497	1.299
NQ	16.341	4.997	1.607
ES	9.624	5.664	1.765
YM	9.087	4.271	1.262
GE	13.956	3.966	3.724
6E	8.244	4.463	3.78

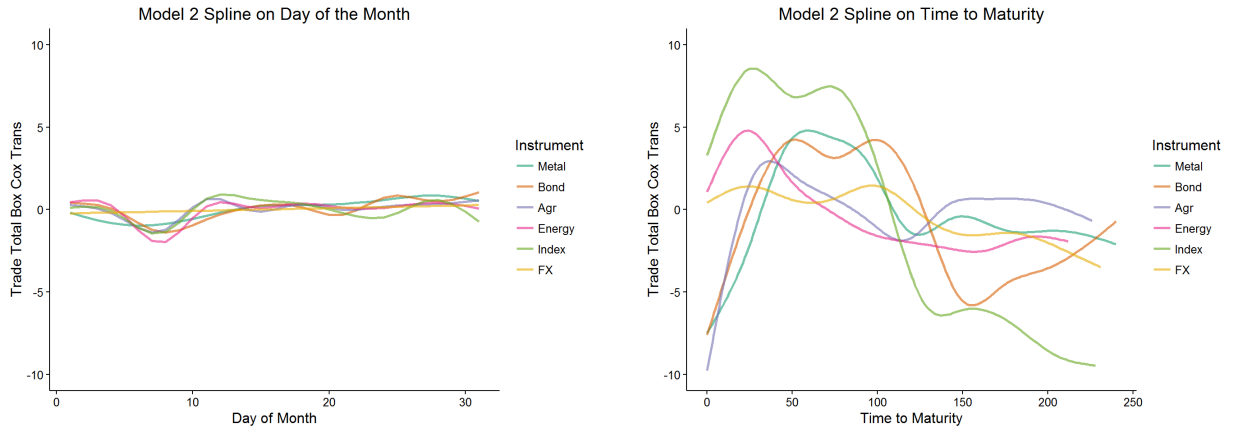


Figure 8: Spline curves for best model

Figure 9 shows that ~~more products had a low probability of having a constant~~ correlation between price volatility and trade volume than if all probability values were equally likely. ~~Consequently, the hypothesis of constant correlations can be rejected in some cases.~~

*p-values for testing constancy of correlations between price volatility & trade volume across products tend to be smaller than expected under the null hypothesis of constant correlations (under which the p-values should appear uniform in (0,1))*

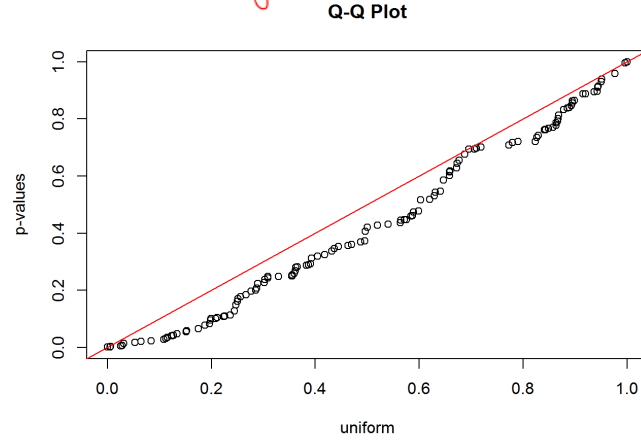


Figure 9: Uniform Q-Q plot

Table 4 shows the five products were deemed as not having constant correlation (i.e. null hypothesis  $\beta_1 = 0$  was rejected) using the Benjamini-Hochberg(B-H) procedure to control false discovery rate(FDR) at 20%.

Table 4: Significant p-values at FDR=.2

Product	P-value
ZQ 2016-08-31	0.00132
YM 2017-03-17	0.00222
HO 2016-11-30	0.00281
ZF 2016-12-30	0.00449
RB 2016-07-29	0.00693

The correlation plots for these five products are shown in Figure 10 with lines of best fit in red.

Figures 11 to 17 show trends in price volatility versus trade volume plots daily and hourly for the entire data collection period. Most products manifested the positive

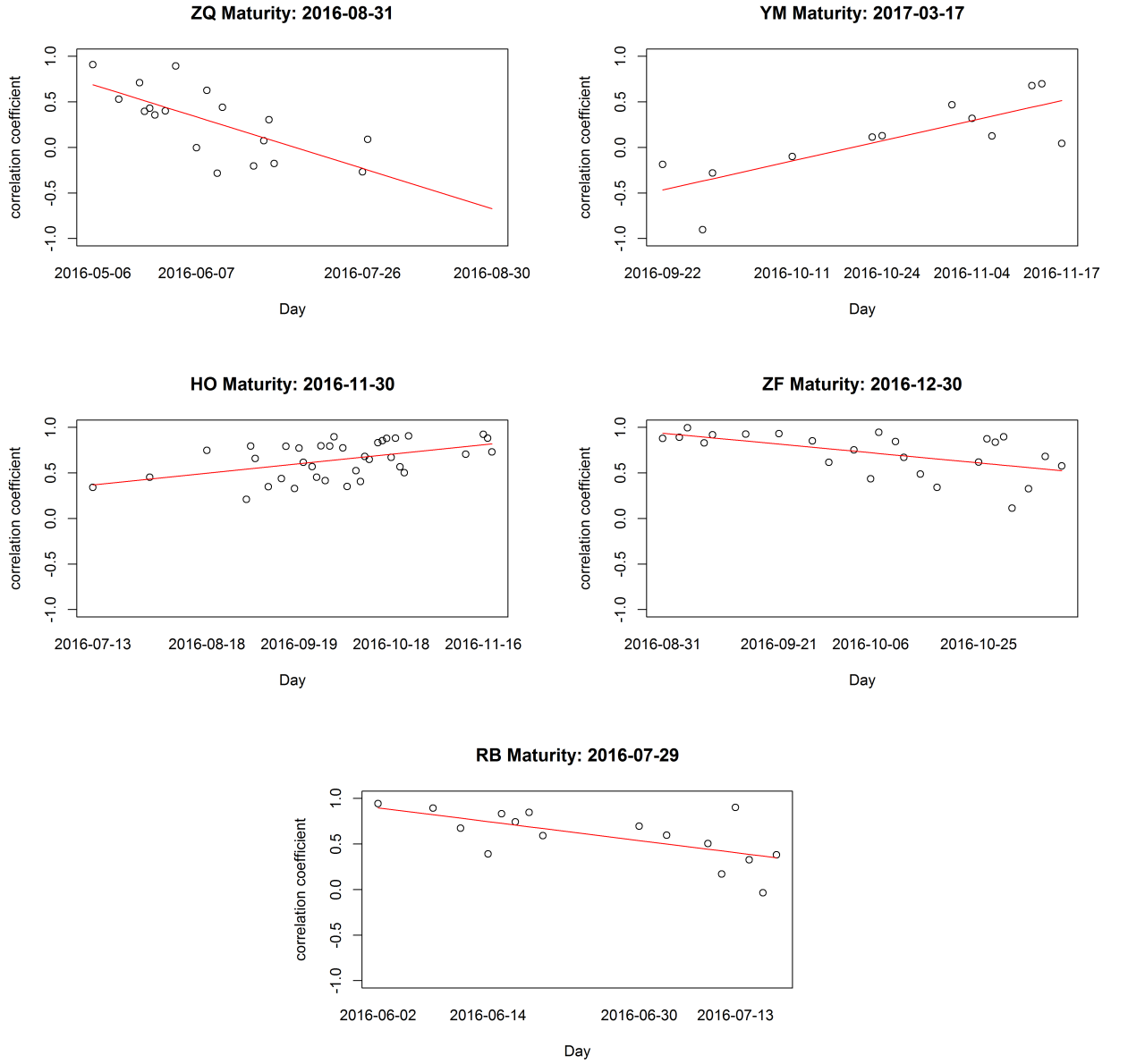


Figure 10: Correlation regression plots for most significant products

correlation expected between price volatility and trade volume as in Figure 11. The relationship, whether measurements were made hourly or daily, was also consistent for the most part. However, some plots (Figures 12 to 17) either deviated from each other when daily and hourly measurements were compared or did not exhibit the expectation of a positive correlation.

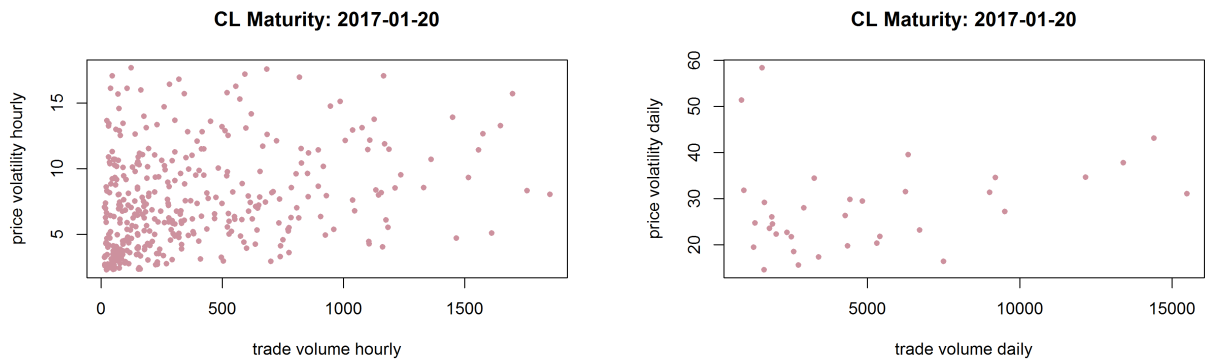


Figure 11: Price volatility vs. trade volume shows positive correlation

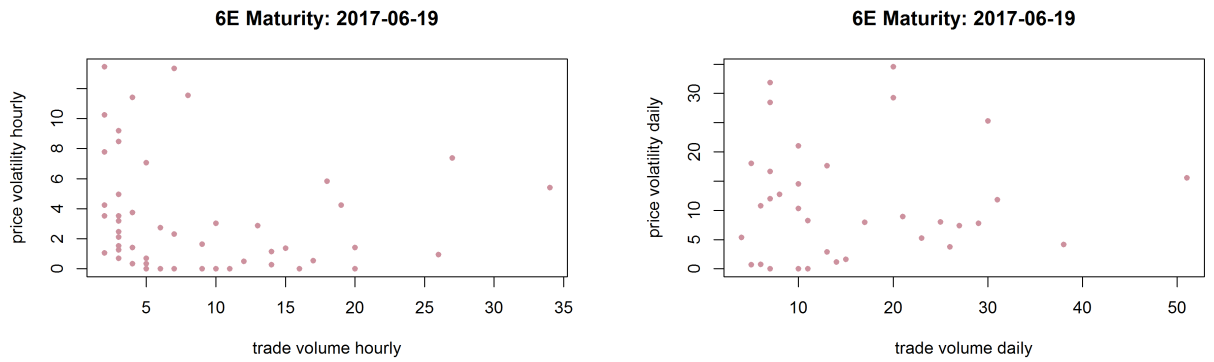


Figure 12: Price volatility vs. trade volume may show two clusters

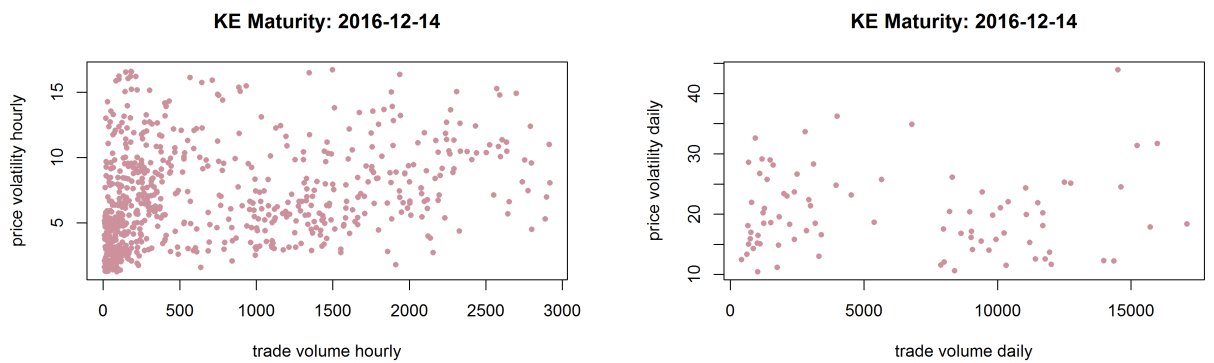


Figure 13: Price volatility vs. trade volume shows two clusters

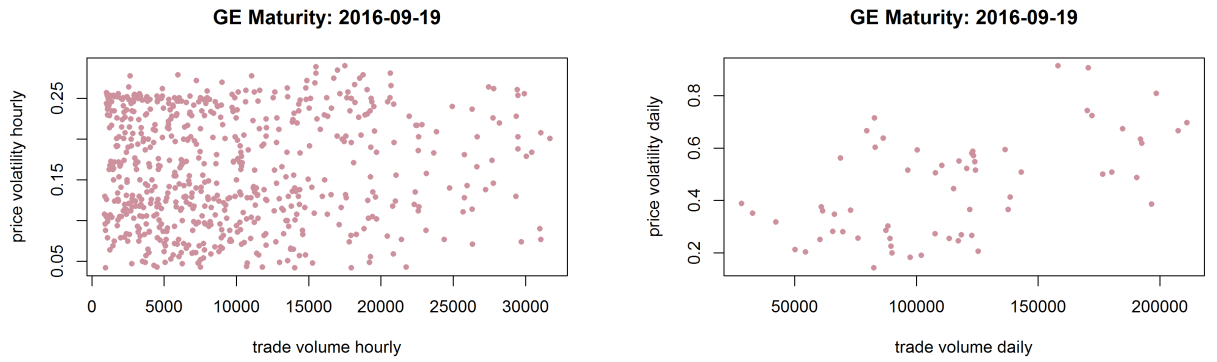


Figure 14: Price volatility vs. trade volume hourly & daily show different correlations

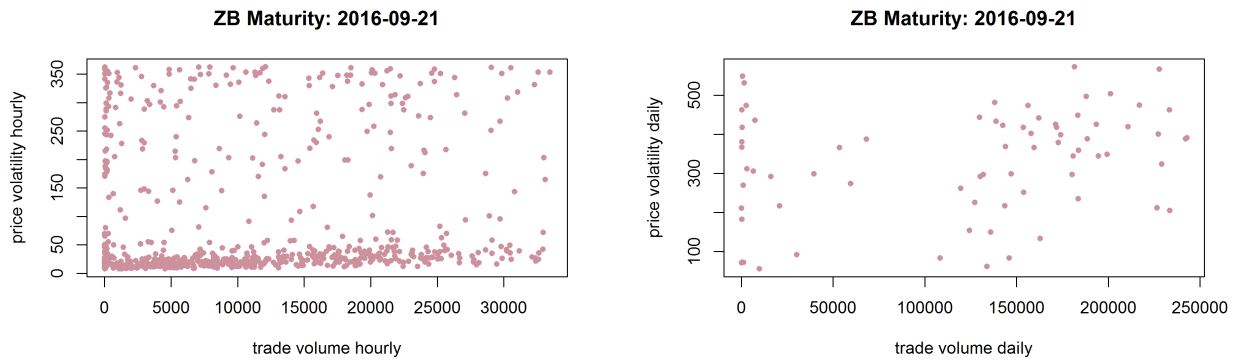


Figure 15: Price volatility vs. trade volume hourly & daily show different clusters

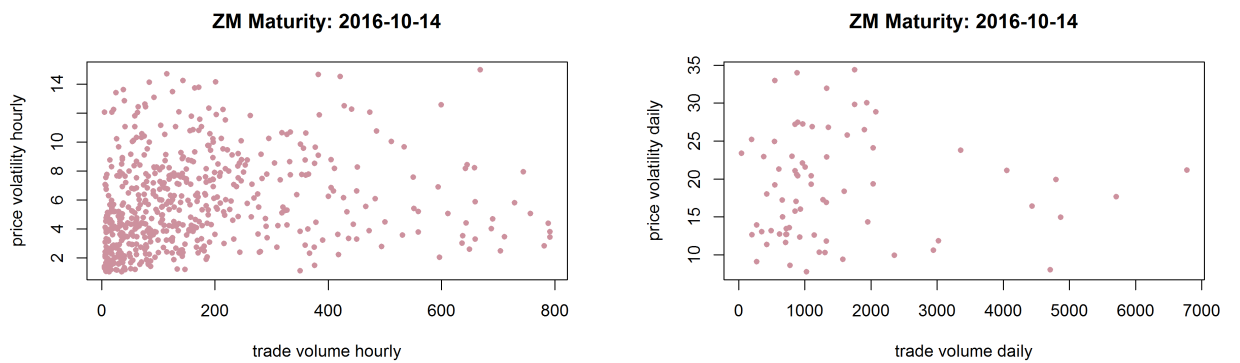


Figure 16: Price volatility vs. trade volume daily may show negative correlation

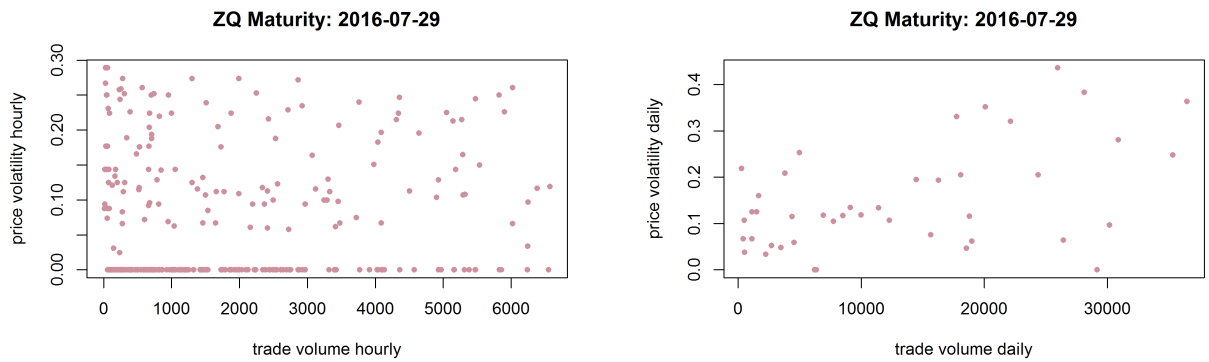


Figure 17: Price volatility vs. trade volume hourly shows three clusters

## Discussion

Considering the best model, and comparing MAD to the median, agriculture, energy, and index have relative errors of 0.182, 0.113, and 0.164 respectively but metal, bond, and FX have relative errors of 0.379, 0.254, and 0.301 respectively. This indicates that the regression performs well on agriculture and energy, which was expected since their trade volumes are normally distributed. Surprisingly, the regression does not perform well on the bond market, but performs well on index even though bond trade volume is more normally distributed than index trade volume. Referring to Figure 5, regression is better for index because all the index instruments are similar but the bond instruments are not.

For instance, in the bond market, the 30 Day Fed Fund (ZQ) is much different than the other instruments. This is because the rest are long term <sup>with maturities</sup> bonds of 2 or more years. It is reasonable that a 30 day bond should behave differently than a long term bond simply because the time frames are so different. The time to maturity spline curve for bond serves as reinforcement of the differences given that the curve begins to rise sharply around 150 instead of staying low as anticipated. The upward trend is due to the effect of ZQ at 150 days is where it begins to deviate from the other

instruments. Proper modeling should separate ZQ from other bonds.

Regression also performs poorly on FX which was expected due to lack of normality. In addition, Figure 5 shows heterogeneity within the FX market. Euro (6E) and Eurodollar (GE) behave completely differently. Upon closer inspection, it seems 6E has the shape of an index, as the shape of the curve is in line with the index market. So improvement to this work may be to model 6E as an index.

Finally, regression <sup>performs the</sup> ~~is~~ worst for metal. In Figure 5, silver and gold appear to behave very differently. In particular, the bottom cluster seems to follow a periodic trend whereas <sup>for</sup> ~~the~~ gold <sup>the trade volume curve</sup> has in inverse parabolic shape.

When we look at the regression results at the instrument level, the relative error is under .20 for most instruments. The exceptions are both metals (GC and SI) as on the market level, both FX instruments (6E and GE) as on the market level, and three bonds: 30 Day Fed Fund (ZQ), 30 Yr U.S. Treasury Bond (ZB), and 10Y Treasury Note (ZN). Their relative errors are 0.360, 0.405, 0.2676, 0.459, 0.257, 0.266, and 0.227 respectively. The clustering observed in FX and Metals might be another reason these markets' trade volume are not predicted well. It is reasonable to separate trades into small and large volumes before doing regression. Looking at the bond instruments ZQ is not shocking for aforementioned reasons. Neither is ZN since it has the highest trading volume. ZB is mysterious, as it appears to behave similarly to ZT. Any future work should analyze this further.

Thus it can be concluded that similarity of instruments being modeled is more important for prediction than normality of distribution for a large number of observations. Therefore instruments similar to one another should be modeled jointly. Models should not be created simply based on market.

The model coefficients are also in line with the volumes depicted in visualizations. For instance the coefficient for silver is smaller than gold. The model coefficients also indicate that day of week is not significant. Day of month does seem significant,



although it is not as pronounced as time to maturity in most cases. Furthermore, the day of month spline curves reinforce patterns seen in the exploratory analysis visualizations of essentially constant daily trade volume as well as anticipated periodicity, but show a dip around day 8 of 30 for all markets with the exception of FX. Future work should investigate what happens to a product after the first week of trading and why FX is not affected.

Regarding the significant products which do not show constant daily correlation, it is possible that the selection of high volume trading by including observations within the 60th to 90th percentiles was not sufficient and so effects of low volume trading in the nascent and near maturity periods may be present. To find out if this theory is true, the trade volumes were plotted in Figure 18.

All five products have high volume regions selected with ZQ and 5 year treasury note (ZF) showing idealized pattern of dense population above zero, as in Figures 6 and 7. Moreover, none of the products manifest a sharp increase, which typically delineates low and high volumes. We can safely exclude the effects of nascent and near maturity periods from all but Rbob gasoline (RB) since the selected area of this product is still very close to maturity. So non-constant correlation significance is legitimate for ZQ, mini Dow Jones index (YM), heating oil (HO), and ZF implying that these products prices will not be well predicted by trade volume over time because their correlation does not remain constant. Despite this, it would be better to analytically determine the volume inflection point rather than using percentiles. Due to time considerations, the latter was done.

Aside from constant correlation, strong correlation between trade volume and price volatility is also needed. That is, preferably above .5 in absolute value. Returning to Figures 11 through 17, for most products the daily correlations are stronger and less noisy than the hourly correlations. In some cases where there is clustering, such as Kc Hrw wheat (KE), ZB, and soybean meal (ZM), observations should be partitioned

what do you  
mean by needed?

into low and high volumes before taking correlations into account for price prediction. For agriculture instruments seasonality and temperature most likely play a part in the trade volume clustering observed.

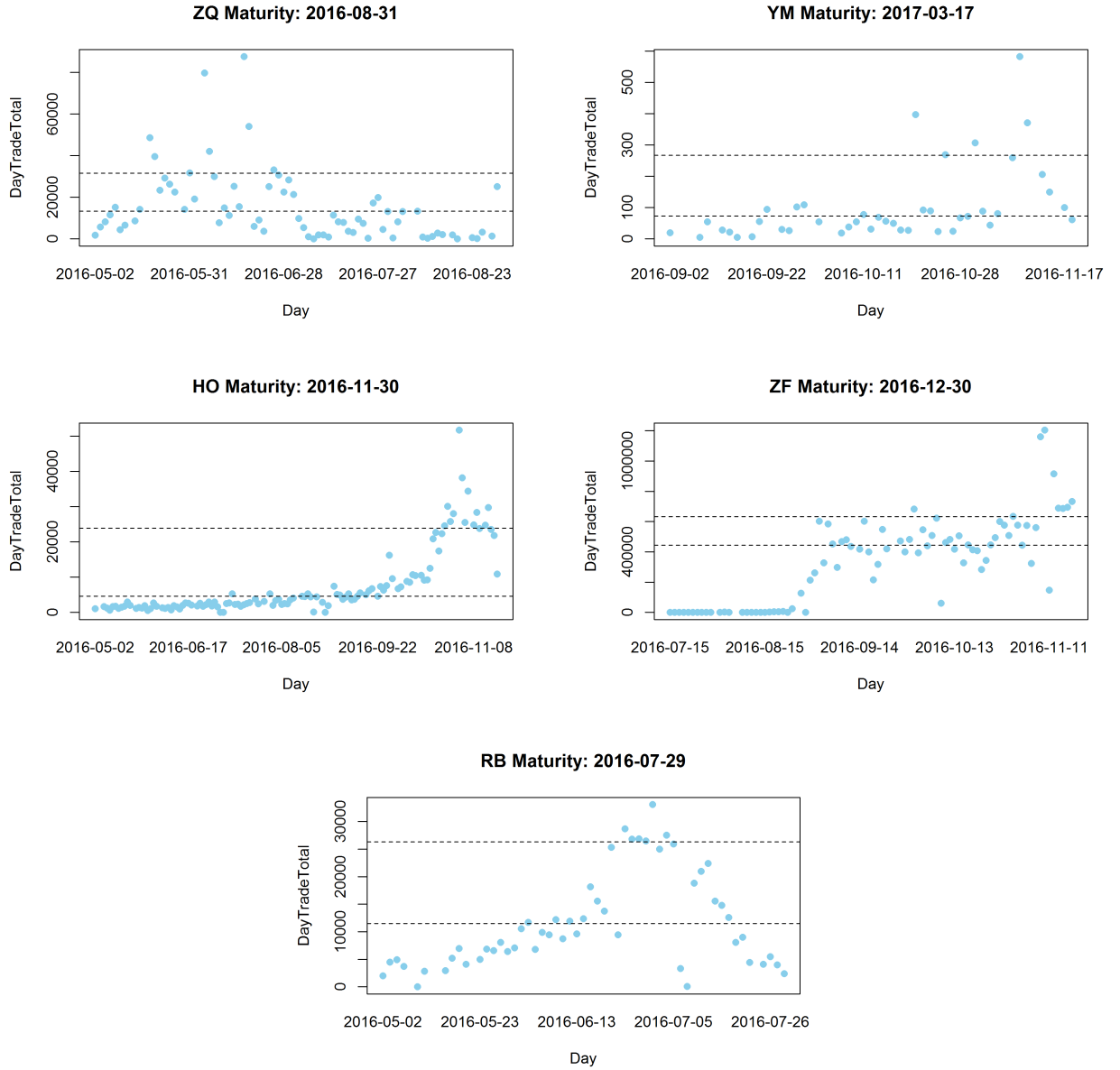


Figure 18: Volume plots for significant products, dashes denote 60th-90th percentiles

# Appendices

## A: Spark/python code for raw data generation

```
1 #Jupyter Notebook Code: NOT MEANT TO BE EXECUTABLE
2
3 import matplotlib.pyplot as plt
4 %matplotlib inline
5 %pylab inline
6 import numpy as np
7 import pandas as pd
8 from os import listdir
9 from pyspark import SparkContext, SparkConf
10 from pyspark.sql import SQLContext, DataFrame, HiveContext
11 from pyspark.sql.functions import *
12 from pyspark.sql.types import *
13 from pyspark.ml.feature import VectorAssembler
14 from pyspark.ml.clustering import KMeans
15
16 sconf = SparkConf().setMaster("local[32]").setAppName("TradeDataCount")\
17 .set("spark.driver.maxResultSize", "16g").set("spark.shuffle.
    consolidateFiles", "true") #spark://10.160.5.48:7077 or local[*] to
    use as many threads as cores
18 sc = SparkContext(conf=sconf)
19 #sqlContext = SQLContext(sc)
20 sqlContext = HiveContext(sc) #really hc
21
22 files = [f for f in listdir('/home/dnaiman1/finddata/big_harvest') if '
    Trades' in f]
23 paths=['/home/dnaiman1/finddata/big_harvest/%s' % i for i in files]
24
25 def parseData(file_path):
26     customSchema = StructType([ \
27         StructField("ProductName", StringType(), True), \
28         StructField("Maturity", StringType(), True), \
29         StructField("Date", StringType(), True), \
30         StructField("Time", StringType(), True), \
31         StructField("Price", DoubleType(), True), \
32         StructField("Quantity", IntegerType(), True)])
33
34     df = (sqlContext
35         .read.format('com.databricks.spark.csv')
36         .options(header='false')
37         .load(file_path, schema=customSchema))
38
39     def maturityConv(date):
40         elements=date.split("/")
41         return "%s-%s-%s" % (elements[2], elements[0], elements[1])
42
43     maturityConvUDF=udf(maturityConv, StringType())
44
45     def dateConv(date):
46         elements=date.split(":")
```

```

47         return "%s-%s-%s" % (elements[0], elements[1], elements[2])
48
49     dateConvUDF=udf(dateConv, StringType())
50
51     def timeConv(time):
52         return time[0:-4]
53
54     timeConvUDF=udf(timeConv, StringType())
55
56     df2=(df
57         .withColumn("MaturityConv", maturityConvUDF(df.Maturity))
58         .withColumn("DateConv", dateConvUDF(df.Date))
59         .withColumn("TimeConv", timeConvUDF(df.Time)))
60
61     df3=(df2
62         .select(['ProductName', col('MaturityConv').cast('date').alias('
Maturity'), col('DateConv').cast('date').alias('Date'), concat_ws('
', df2.DateConv, df2.TimeConv)
63         .alias('TimeStamp').cast('timestamp'), 'Price', 'Quantity'])) #can
also use to_date('MaturityConv').alias('Maturity')
64
65     return df3
66
67 parsed_data=[]
68 for file in paths[8:]:
69     parsed_data.append(parseData(file))
70
71 def union(data_list):
72     unionDF=data_list[0].unionAll(data_list[1])
73     for df in data_list[2:]:
74         unionDF=unionDF.unionAll(df)
75     return unionDF
76
77 data=union(parsed_data).cache()
78
79 print "Raw data is %d rows." % data.count() #takes long b/c count action
initiates caching process
80
81 max_date=data.select(max('Date')).first()
82 print "Max date is %s." % max_date
83
84 data.show()
85
86 data.rdd.saveAsPickleFile('raw_dataset_rdd.pickle') #python equivalent
of saveAsObjectFile()
87 sc.stop()

```

## B: Spark/python code for creating datasets to be analyzed

```
1 #Jupyter Notebook Code: NOT MEANT TO BE EXECUTABLE
2
3 import numpy as np
4 import pandas as pd
5 from scipy import stats
6 from pyspark import SparkContext, SparkConf
7 from pyspark.sql import DataFrame, HiveContext #SQLContext
8 from pyspark.sql.functions import *
9 from pyspark.sql.types import *
10
11 pd.set_option('display.max_columns', None)
12 pd.set_option('display.max_rows', None)
13
14 sconf = SparkConf().setMaster("local[32]").setAppName("TradeDataCount").
    set("spark.driver.maxResultSize", "32g").set("spark.shuffle.
    consolidateFiles", "true") #spark://10.160.5.48:7077 or local[*] to
    use as many threads as cores
15 sc = SparkContext(conf=sconf)
16 hc = HiveContext (sc)
17
18 dataset=sc.pickleFile('raw_dataset_rdd.pickle')
19 dataset=dataset.toDF()
20
21 holidays=["2016-05-30", "2016-07-04", "2016-09-05"] #better to exclude
    files when doing intial parsing?
22 data=(dataset
23 .select(['ProductName', 'Maturity', 'Date', 'TimeStamp', hour("TimeStamp")
    ).alias("Hour"), 'Price', 'Quantity'])
24 .where((dataset.Date.isin(holidays)==False)).cache())
25 print "Raw data is %d rows." % data.count()
26
27 max_date=data.select(max('Date')).first()
28 print "Max date is %s." % max_date
29
30 data.registerTempTable("RawData")
31 #check holidays and weather index gone
32 hc.sql('select * from RawData where Date in ("2016-05-30", "2016-07-04",
    "2016-09-05")').show()
33
34 uniqueDates=hc.sql("select distinct Date from RawData order by Date ASC"
    )
35 uniqueDatesDF=uniqueDates.toPandas()
36 uniqueDatesDF.head(10)
37
38 n=len(uniqueDatesDF)
39 n
40
41 uniqueDatesDF['Day']=range(1, n+1)
42 uniqueDatesDF.tail(10)
43
44 def productConv(product):
45     if product in ['6E', 'GE']:
```

```

46         return 'FX'
47     elif product in ['SI', 'GC']:
48         return 'Metal'
49     elif product in ['CL', 'NG', 'RB', 'HO']:
50         return 'Energy'
51     elif product in ['ES', 'NQ', 'YM']:
52         return 'Index'
53     elif product in ['KE', 'ZC', 'ZL', 'ZM', 'ZW']:
54         return 'Agr'
55     else:
56         return 'Bond'
57
58 productConvUDF=udf(productConv, StringType())
59
60 def dateConv(date):
61     if date in list(uniqueDatesDF["Date"]):
62         return int(uniqueDatesDF.loc[uniqueDatesDF["Date"]==date]['Day'
63 ])
64     else:
65         return 0
66
67 dateConvUDF=udf(dateConv, IntegerType())
68
69 #Creating the price volatility data set.
70 pivotVol=(data
71 .groupBy("ProductName", "Maturity", "Date")
72 .pivot("Hour")
73 .agg(round(stddev_samp("Price"), 3))
74 .orderBy("ProductName", "Maturity", "Date")
75 .cache())
76
77 print "Pivoted data is %d rows." % pivotVol.count()
78
79 pivotVol.show(5)
80
81 dayVol= hc.sql('select ProductName, Maturity, Date, round(stddev_samp(
82     Price), 3) DayVolatility from RawData group by
83     ProductName, Maturity, Date order by ProductName, Maturity, Date')
84
85 dayVol.show(5)
86
87 pivotVolDF=pivotVol.toPandas()
88 dayVolDF=dayVol.toPandas()
89 pivotVolDF["DayVolatility"]=dayVolDF["DayVolatility"]
90 pivotVolDF.head()
91
92 pivotVolDF.to_csv("trades_pivot_vol_%s.txt" % max_date, sep='\t', index=
93     False)
94
95 #Creating the trade volume data set.
96 pivot=(data
97 .groupBy("ProductName", "Maturity", "Date")
98 .pivot("Hour")
99 .agg(sum("Quantity"))
100 .orderBy("ProductName", "Maturity", "Date"))

```

```

96     .na.fill(0)
97     .cache())
98
99 print "Pivoted data is %d rows." % pivot.count()
100
101 pivot.show(5)
102
103 pivotDF=pivot.toPandas()
104 pivotDF['DayTradeTotal'] = pivotDF[pivotDF.columns[3:]].sum(axis=1)
105 pivotDF.head()
106
107 pivotDF.to_csv("trades_pivot_%s.txt" % max_date, sep='\t', index=False)
108
109 #creating regression data set
110 #adding Time to Maturity, Day of week, Day of Month, Product Type/Market
    , and Day columns
111 pivot=pivot.select(["*", datediff(pivot.Maturity, pivot.Date).alias("
    TimeToMaturity"), date_format(pivot.Date, 'E').alias('DayOfTheWeek')
    , dayofmonth(pivot.Date).alias('DayofMonth')])
112 pivot=pivot.withColumn("ProductType", productConvUDF(pivot.ProductName))
    .withColumn("Day", dateConvUDF(pivot.Date)).withColumn("DayofWeek",
    dayConvUDF(pivot.DayOfTheWeek))
113 pivot.select(["ProductName", "Date", "Day"]).where(pivot.Day==0).show()
    #check all dates matched, should get null result
114
115 pivot.columns
116
117 pivot.columns[0:3]+pivot.columns[-5:]
118
119 pivotDF=pivot.toPandas()
120 pivotDF['DayTradeTotal'] = pivotDF[pivotDF.columns[3:-5]].sum(axis=1)
121 pivotDF.head()
122
123 pivotDF['TradeTotBCTrans']=stats.boxcox(pivotDF['DayTradeTotal'])[0]
124 pivotDF['TradeTotLogTrans']=np.log(pivotDF['DayTradeTotal'])
125 dataset=pivotDF[list(pivotDF.columns[0:3])+list(pivotDF.columns[-8:])]
126 dataset.head()
127
128 dataset.to_csv("trades_count_regression_%s.txt" % max_date, sep='\t',
    index=False)
129
130 sc.stop()

```

## C: Python code for exploratory data analysis

```
1 #Jupyter Notebook Code: NOT MEANT TO BE EXECUTABLE
2
3 import matplotlib.pyplot as plt
4 import matplotlib.patches as mpatches
5 get_ipython().magic(u'matplotlib inline')
6 import numpy as np
7 import pandas as pd
8 import matplotlib.cm as cm
9 from scipy import stats
10
11 pd.set_option('display.max_columns', None)
12 pd.set_option('display.max_rows', None)
13
14 dataset=pd.read_csv('trades_count_regression_2016-11-18.txt', sep='\t')
15 dataset.head()
16
17 #making raw (not Box-Cox transformed) trade volume histogram for entire
    population
18 plt.figure(figsize=(6, 5))
19 plt.hist(dataset.DayTradeTotal); plt.title("Original Trade Volume");
20 plt.savefig('hist_orig.png', bbox_inches='tight')
21
22 #making trade volume histogram for entire population
23 plt.figure(figsize=(6, 5))
24 plt.hist(dataset.TradeTotBCTrans); plt.title("BoxCox Transformed Trade
    Volume");
25 plt.savefig('hist_bc.png', bbox_inches='tight')
26
27 #making trade volume histograms by market
28 plt.figure(figsize=(15, 15))
29 for i, j in enumerate(np.unique(dataset.ProductType)):
30     subset=dataset.loc[dataset.ProductType==j]
31     plt.subplot(3,2,i+1);
32     plt.subplots_adjust(hspace=.3);
33     plt.subplots_adjust(wspace=.5);
34     plt.hist(subset['TradeTotBCTrans']); plt.title(j+" BoxCox
        Transformed Trade Volume");
35
36 plt.savefig('types_bc_hist.png', bbox_inches='tight')
37
38 #making trade volume vs time to maturity plot for entire population
39 plt.figure(figsize=(6, 5))
40 plt.plot(dataset[['TimeToMaturity']], dataset[['TradeTotBCTrans']], '.');
    ; plt.xlabel('TimeToMaturity');
41 plt.ylabel('TradeTotBCTrans');
42 plt.savefig('time_to_mat.png', bbox_inches='tight')
43
44 #making trade volume vs day plot for entire population
45 plt.figure(figsize=(6, 5))
46 plt.plot(dataset[['Day']], dataset[['TradeTotBCTrans']], '.'); plt.
    xlabel('Day'); plt.ylabel('TradeTotBCTrans');
47 plt.ylabel('TradeTotBCTrans');
```



```

48 plt.savefig('bc_sample_day_wMat.png', bbox_inches='tight')
49
50 #making trade volume vs day plots by market
51 trans=.7 #edgecolor='none'
52 plt.figure(figsize=(12, 15))
53 for j, i in enumerate(np.unique(dataset.ProductType)):
54     subset=dataset.loc[dataset.ProductType==i]
55     uCols=np.unique(subset.ProductName)
56     labs=[np.unique(subset.ProductName).tolist().index(k) for k in
subset.ProductName]
57     plt.subplot(3,2,j+1);
58     plt.subplots_adjust(hspace=.3);
59     plt.subplots_adjust(wspace=.5);
60     plt.scatter(subset[['Day']], subset[['TradeTotBCTrans']], c=labs,
cmap=cm.rainbow, alpha=trans); plt.xlabel('Day'); plt.ylabel('
TradeTotBCTrans');
61     plt.title(i);
62
63     cust_hand=[]
64     for j in range(len(uCols)):
65         cust_hand.append(mpatches.Patch(color=cm.rainbow(np.linspace(0,
1, len(uCols)))[j], alpha=trans), label=uCols[j]))
66     plt.legend(handles=cust_hand, bbox_to_anchor=(1.05, 1), loc=2,
borderaxespad=0.)
67
68 plt.savefig('agr_scatter_transMat.png', bbox_inches='tight')
69
70 #making trade volume vs time to maturity plots by market
71 trans=.6 #edgecolor='none'
72 plt.figure(figsize=(15, 17))
73 for i, j in enumerate(np.unique(dataset.ProductType)):
74     subset=dataset.loc[dataset.ProductType==j]
75     uCols=np.unique(subset.ProductName)
76     labs=[np.unique(subset.ProductName).tolist().index(k) for k in
subset.ProductName]
77     plt.subplot(3,2,i+1);
78     plt.subplots_adjust(hspace=.3);
79     plt.subplots_adjust(wspace=.5);
80     plt.scatter(subset[['TimeToMaturity']], subset[['TradeTotBCTrans']],
c=labs, cmap=cm.rainbow, alpha=trans); plt.xlabel('TimeToMaturity')
;
81     plt.ylabel('TradeTotBCTrans'); plt.title(j);
82
83     cust_hand=[]
84     for j in range(len(uCols)):
85         cust_hand.append(mpatches.Patch(color=cm.rainbow(np.linspace(0,
1, len(uCols)))[j], alpha=trans), label=uCols[j]))
86     plt.legend(handles=cust_hand, bbox_to_anchor=(1.05, 1), loc=2,
borderaxespad=0.)
87
88 plt.savefig('types_time_to_mat.png', bbox_inches='tight')

```

## D: R code for spline regression for trade volume

```
1 library(mgcv)
2 library(ggplot2)
3 setwd("your\\computer\\directory")
4 trade_data=read.table("trades_count_regression_2016-11-18.txt", sep="\t"
5 , header=T)
6 trade_data=trade_data[order(trade_data$Date),]
7 head(trade_data)
8
9 #use only half of data randomly split into train and test
10 n=nrow(trade_data)
11 lastDate=as.vector(trade_data$Date[n/2])
12 nRem=nrow(subset(trade_data[(n/2+1):n,], Date==lastDate))
13 train=trade_data[1:(n/2+nRem), ] #past data, ends with 8/2 data
14 set.seed(12-1-16)
15 test_id=sample((n/2+nRem+1):n, (n/2-nRem)[1:floor((n/2-nRem)/2)])
16 test1=trade_data[test_id, ] #future data
17 test2=trade_data[-test_id, ] #reserve data
18
19 # Build linear regression model with penalized cubic spline. Spline
20 parameters found by cross validation.
21
22 mod1 <- gam(TradeTotBCTrans ~ s(TimeToMaturity, bs="cr") + s(
23 DayofMonth, bs="cr") + factor(ProductType) + factor(ProductName) +
24 factor(DayOfTheWeek), family=gaussian(), data = train, method="GCV.
25 Cp", select=TRUE)
26
27 summary(mod1)
28
29 plot(mod1)
30
31 p <- predict(mod1, type="lpmatrix")
32
33
34 beta <- coef(mod1)[grepl("TimeToMaturity", names(coef(mod1)))]
35 s <- p[,grepl("TimeToMaturity", colnames(p))] %*% beta
36
37 ggplot(data=cbind.data.frame(s, train$TimeToMaturity), aes(x=train$
38 TimeToMaturity, y=s, ymin = -5, ymax = 3)) + geom_line()
39
40 beta <- coef(mod1)[grepl("DayofMonth", names(coef(mod1)))]
41 s <- p[,grepl("DayofMonth", colnames(p))] %*% beta
42
43 ggplot(data=cbind.data.frame(s, train$DayofMonth), aes(x=train$
44 DayofMonth, y=s, ymin = -5, ymax = 3)) + geom_line()
45
46
47
48
49 mod1$coefficients
50 mod1$sp #spline parameters
51
52 res=mod1$residuals
53 plot(res, pch=20)
54 hist(res)
55
56
57 # Actual values are black, predicted values are red
58 trainPred=mod1$fitted.values
59 plot(train$TimeToMaturity, train$TradeTotBCTrans, pch=20, ylim=c(min(
60 train$TradeTotBCTrans, trainPred), max(train$TradeTotBCTrans,
61 trainPred)))
62
63 points(train$TimeToMaturity, trainPred, col="red", pch=20)
64 legend("topright", c("actual", "predicted"), lwd=c(3,3), col=c("black", "
```

```

    red"))
44
45 plot(train$Day, train$TradeTotBCTrans, pch=20, ylim=c(min(train$
    TradeTotBCTrans, trainPred), max(train$TradeTotBCTrans, trainPred)))
46 points(train$Day, trainPred, col="red", pch=20)
47
48 # Model trained on 1/2 of past data so only testing on 1/4 of randomized
    future data.
49 new_data1=test1
50 head(new_data1)
51 new_data1Pred <- predict(modell, new_data1)
52 new_data1["Prediction"]=new_data1Pred
53 plot(new_data1$TimeToMaturity, new_data1$TradeTotBCTrans, pch=20, ylim=c
    (min(new_data1$TradeTotBCTrans, new_data1Pred), max(new_data1$
    TradeTotBCTrans, new_data1Pred)))
54 points(new_data1$TimeToMaturity, new_data1$Prediction, col="red", pch
    =20)
55 legend("topright", c("actual", "predicted"),lwd=c(3,3),col=c("black", "
    red"))
56
57 plot(new_data1$Day, new_data1$TradeTotBCTrans, pch=20, ylim=c(min(new_
    data1$TradeTotBCTrans, new_data1Pred), max(new_data1$TradeTotBCTrans
    , new_data1Pred)))
58 points(new_data1$Day, new_data1$Prediction, col="red", pch=20)
59
60 modellprime <- gam(TradeTotBCTrans ~ s(TimeToMaturity, bs = "cr") + s(
    DayofMonth, bs = "cr") + factor(ProductType) + factor(DayOfTheWeek
    ), family=gaussian(), data = train, method="GCV.Cp", select=TRUE)
61 summary(modellprime)
62 new_data1PredP <- predict(modellprime, new_data1)
63 new_data1["PredictionPrime"]=new_data1PredP
64
65 # Calculating mean absolute error (MAE/MAD) and means for each product
    type.
66 types=c('Metal', 'Bond', 'Agr', 'Energy', 'Index', 'FX')
67 MAEone_mod=c()
68 MAEone_modP=c()
69 medians=c()
70 count=0
71 for (type in types){
72   count=count+1
73   typsub=subset(new_data1, ProductType==type)
74   MAEone_mod[count]= mean(abs(typsub$TradeTotBCTrans-typsub$Prediction))
75   MAEone_modP[count]= mean(abs(typsub$TradeTotBCTrans-typsub$
    PredictionPrime))
76   medians[count]=median(typsub$TradeTotBCTrans)
77 }
78
79 # Creating models for each product type.
80 placeH=data.frame(Instrument=NULL, STTM=NULL, TimeToMat=NULL)
81 placeHD=data.frame(Instrument=NULL, sDoM=NULL, DoM=NULL)
82 MAEmult_mod=c()
83 MAEmult_modPrime=c()
84 all_products=c()

```

```

85 all_medians=c()
86 all_MAE=c()
87 all_MAEprime=c()
88 count=0
89 for (type in types){
90   count=count+1
91   typsub=subset(train, ProductType==type)
92   model <- gam(TradeTotBCTrans ~ s(TimeToMaturity, bs="cr") + s(
     DayOfMonth, bs="cr") + factor(ProductName) + factor(DayOfTheWeek),
     family=gaussian(), data = typsub, method="GCV.Cp", select=TRUE)
93   print(summary(model))
94   plot(model)
95
96   #superimposed spline plots
97   p <- predict(model, type="lpmatrix")
98
99   beta <- coef(model)[grepl("TimeToMaturity", names(coef(model)))]
100   sTTM <- p[,grepl("TimeToMaturity", colnames(p))] %*% beta
101   placeH= rbind(placeH, data.frame(Instrument=rep(type, nrow(typsub)),
     STTM=sTTM, TTM=typsub$TimeToMaturity))
102
103   beta <- coef(model)[grepl("DayOfMonth", names(coef(model)))]
104   sDoM <- p[,grepl("DayOfMonth", colnames(p))] %*% beta
105   placeHD = rbind(placeHD, data.frame(Instrument=rep(type, nrow(typsub)),
     sDoM=sDoM, DoM=typsub$DayOfMonth))
106
107   modelPrime <- gam(TradeTotBCTrans ~ s(TimeToMaturity, bs = "cr") + s(
     DayOfMonth, bs = "cr") + factor(DayOfTheWeek), family=gaussian(),
     data = typsub, method="GCV.Cp", select=TRUE)
108   print(summary(modelPrime))
109   plot(modelPrime)
110
111   typsub_new=subset(new_data1, ProductType==type)
112   new_data1Pred <- predict(model, typsub_new)
113   typsub_new["Prediction"]=new_data1Pred
114   cat(type)
115   plot(typsub_new$TimeToMaturity, typsub_new$TradeTotBCTrans, pch=20,
     ylim=c(min(typsub_new$TradeTotBCTrans, typsub_new$Prediction), max(
     typsub_new$TradeTotBCTrans, typsub_new$Prediction)))
116   points(typsub_new$TimeToMaturity, typsub_new$Prediction, col="red",
     pch=20)
117   legend("topright", c("actual", "predicted"),lwd=c(3,3),col=c("black",
     "red"))
118   plot(typsub_new$Day, typsub_new$TradeTotBCTrans, pch=20, ylim=c(min(
     typsub_new$TradeTotBCTrans, typsub_new$Prediction), max(typsub_new$
     TradeTotBCTrans, typsub_new$Prediction)))
119   points(typsub_new$Day, typsub_new$Prediction, col="red", pch=20)
120   MAEmult_mod[count]= mean(abs(typsub_new$TradeTotBCTrans-typsub_new$
     Prediction))
121
122   typsub_newT=typsub_new[rep(1,2200),]
123   true=typsub_newT$Day
124   set.seed(1-13-17)
125   typsub_newT['Day']=sample(rep(1:140, 16)[1:2200], 2200)

```

```

126 print(head(tybsub_newT))
127 new_data1PredT <- predict(model, tybsub_newT)
128 tybsub_newT["Prediction"]=new_data1PredT
129
130 new_data1PredTP <- predict(modelPrime, tybsub_newT)
131 tybsub_newT["PredictionPrime"]=new_data1PredTP
132
133 legend("topright", c("actual", "predicted"), lwd=c(3,3), col=c("black"
, "red"))
134 plot(tybsub_newT$Day, tybsub_newT$TradeTotBCTrans, pch=20, ylim=c(min(
tybsub_newT$TradeTotBCTrans, tybsub_newT$Prediction), max(tybsub_
newT$TradeTotBCTrans, tybsub_newT$Prediction)))
135 points(tybsub_newT$Day, tybsub_newT$Prediction, col="red", pch=20)
136 cat("True Day:", true[1], "\n", "\n")
137
138 MAEmult_modP=c()
139 MAEmult_modPprime=c()
140 medianP=c()
141 countP=0
142 products=as.vector(unique(tybsub_new$ProductName))
143 all_products=append(all_products,products,length(all_products))
144 for (product in products){
145     countP=countP+1
146     probsub=subset(tybsub_new, ProductName==product)
147     MAEmult_modP[countP]= mean(abs(probsub$TradeTotBCTrans-probsub$
Prediction))
148     MAEmult_modPprime[countP]= mean(abs(probsub$TradeTotBCTrans-
probsub$PredictionPrime))
149     medianP[countP]=median(probsub$TradeTotBCTrans)
150 }
151 all_MAE=append(all_MAE,MAEmult_modP,length(all_MAE))
152 all_MAEprime=append(all_MAEprime,MAEmult_modPprime,length(all_MAE))
153 all_medians=append(all_medians, medianP, length(all_medians))
154 }
155 p<-ggplot(placeH, aes(x=TTM, y=STTM, group=Instrument, ymin = -10, ymax
= 10)) + geom_line(aes(color=Instrument), size=1, alpha = 0.6)+ ylab
("Trade Total Box Cox Trans")+xlab("Time to Maturity") + labs(title=
"Model 2 Spline on Time to Maturity")+ theme_classic()+scale_color_
brewer(palette="Dark2")
156 p
157
158 pD<-ggplot(placeHD, aes(x=DoM, y=sDoM, group=Instrument, ymin = -10,
ymax = 10)) + geom_line(aes(color=Instrument), size=1, alpha = 0.6)+
ylab("Trade Total Box Cox Trans")+xlab("Day of Month") + labs(title
="Model 2 Spline on Day of the Month")+ theme_classic()+scale_color_
brewer(palette="Dark2")
159 pD
160
161 data.frame(types, Model2=MAEmult_mod, Model2Prime=MAEmult_modPprime,
Model1=MAEone_mod, Model1Prime=MAEone_modP, Median=medians) #
separate models for each product type
162 #comparing all 4 models
163 data.frame(all_products, Model2=all_MAE, Model2Prime=all_MAEprime,
Median=all_medians) #comparing just separate models

```

## E: R code for price volatility and trade volume relationship

```
1 setwd("your\\computer\\directory")
2 tv=read.table("trades_pivot_2016-11-18.txt", sep="\t", header=T)
3 pv=read.table("trades_pivot_vol_2016-11-18.txt", sep="\t", header = T)
4 trade_data=read.table("trades_count_regression_2016-11-18.txt", sep="\t"
, header=T)
5 tv_pv_comb=cbind(tv[,1:16], pv[,4:16], tv[17], pv[17])
6 tv_pv_comb["Day"]=trade_data["Day"]
7 head(tv_pv_comb)
8
9 pvals=c()
10 prod=c()
11 products=as.vector(unique(tv_pv_comb$ProductName))
12 significant=list(c("ZQ", "2016-08-31"), c("YM", "2017-03-17"), c("HO", "
2016-11-30"), c("ZF", "2016-12-30"), c("RB", "2016-07-29"))
13 dfInt=data.frame(Products=c("HO", "6E", "CL", "ES", "KE", "ZL", "CL", "
GC", "RB", "GE", "GE", "GE", "UB", "ZB", "ZM", "ZQ"), Maturities=c(
"2016-09-30", "2017-06-19", "2016-07-20", "2016-09-16", "2016-12-14",
"2016-12-14", "2017-01-20", "2016-10-27", "2016-07-29", "2016-09-19"
, "2016-07-18", "2016-11-14", "2016-09-21", "2016-09-21", "
2016-10-14", "2016-07-29"))
14
15 inDF=function(x, product, maturity){
16   if ((x[1]==product) & (x[2]==maturity)){
17     return(TRUE)
18   }else{
19     return(FALSE)
20   }
21 }
22
23 for (product in products){ #in results, put 3 plots on 1 line
24   tv_pv_combS=subset(tv_pv_comb, ProductName==product)
25   maturities=as.vector(unique(tv_pv_combS$Maturity))
26   for (maturity in maturities){
27     tv_pv_combM=subset(tv_pv_combS, Maturity==maturity)
28     if (nrow(tv_pv_combM)>=30){
29       cors=apply(tv_pv_combM, 1, function(x){cor(as.numeric(x[4:16]),
as.numeric(x[17:29]), use="na.or.complete")})
30       dfCors=data.frame(tv_pv_combM[c("Date", "Day", "DayTradeTotal")
], cors=as.vector(cors))
31       tradeQuant=quantile(dfCors$DayTradeTotal, c(.6, .9))
32       tv_pv_combMQ=dfCors[(dfCors$DayTradeTotal>tradeQuant[1]) & (
dfCors$DayTradeTotal<tradeQuant[2]),] #selecting high volume trades
33       mod=lm(tv_pv_combMQ$cors~tv_pv_combMQ$Day)
34       cor_hat=predict(mod, tv_pv_combMQ["Day"])
35       tv_pv_combMQ["cor_hat"]=cor_hat
36
37       #saving correlation plots of significance with line of best fit
38       if ((product==significant[[1]][1]) & (maturity==significant
[[1]][2]) | (product==significant[[2]][1]) & (maturity==significant
[[2]][2]) | (product==significant[[3]][1]) & (maturity==significant
[[3]][2]) | (product==significant[[4]][1]) & (maturity==significant
[[4]][2]) | (product==significant[[5]][1]) & (maturity==significant
```

```

[[5]][2])){
39     png(paste("Coeff_fit", product, maturity, ".png", sep=""),
width = 7, height = 7, units = 'in', res = 400)
40     plot(tv_pv_combMQ$Day, tv_pv_combMQ$cors, ylim=c(-1,1), xlab="
Day", ylab="correlation coefficient", main=paste(product, "Maturity:
", maturity), xaxt="n")
41     lines(tv_pv_combMQ$Day, tv_pv_combMQ$cor_hat, col="red")
42     at=seq(min(tv_pv_combMQ$Day), max(tv_pv_combMQ$Day), 1)
43     labs=rep(NA, length(at))
44     labs[which(at%in%tv_pv_combMQ$Day)]=as.vector(tv_pv_combMQ$
Date)
45     axis(1, at=at, labels=labs, tick=F)
46     dev.off()
47 }
48
49     plot(tv_pv_combMQ$Day, tv_pv_combMQ$cors, ylim=c(-1,1), xlab="
Day", ylab="correlation coefficient", main=paste(product, "Maturity:
", maturity), xaxt="n")
50     lines(tv_pv_combMQ$Day, tv_pv_combMQ$cor_hat, col="red")
51     at=seq(min(tv_pv_combMQ$Day), max(tv_pv_combMQ$Day), 1)
52     labs=rep(NA, length(at))
53     labs[which(at%in%tv_pv_combMQ$Day)]=as.vector(tv_pv_combMQ$Date)
54     axis(1, at=at, labels=labs, tick=F)
55
56     stat=summary(mod)
57     stat=summary(mod)
58     cof=stat$coefficients
59     pval=cof[,4][2]
60     pvals=append(pvals, pval, after=length(pvals))
61     prod=append(prod, paste(product, maturity), after=length(prod))
62
63     trades=as.vector(as.matrix(t(tv_pv_combM[,4:16])))
64     priceVol=as.vector(as.matrix(t(tv_pv_combM[,17:29])))
65     df=data.frame(trades, priceVol)[order(trades),]
66     df=na.omit(df)
67     n=nrow(df)
68     dfTrim=df[ceiling(.05*n):round(.95*n),]
69     dfTrimO=dfTrim[order(dfTrim$priceVol),]
70     n2=nrow(dfTrimO)
71     dfTrimP=dfTrimO[ceiling(.05*n2):round(.95*n2),]
72
73     plot(dfTrimP, xlab="trade volume hourly", ylab="price volatility
hourly", main=paste(product, "Maturity:", maturity), pch=20)
74
75     daily=tv_pv_combM[,30:31][order(tv_pv_combM$DayTradeTotal),]
76     daily=na.omit(daily)
77     nD=nrow(daily)
78     dailyTrim=daily[ceiling(.05*nD):round(.95*nD),]
79     dailyTrimO=dailyTrim[order(dailyTrim$DayVolatility),]
80     nD2=nrow(dailyTrimO)
81     dailyTrimP=dailyTrimO[ceiling(.05*nD2):round(.95*nD2),]
82     plot(dailyTrimP, xlab="trade volume daily", ylab="price
volatility daily", main=paste(product, "Maturity:", maturity), pch
=20)

```

```

83
84     t=apply(dfInt, 1, inDF, product, maturity)
85     if (TRUE %in% t){
86         png(paste(product, maturity, "hourly.png", sep=""), width = 6,
87             height = 4, units = 'in', res = 300)
88         plot(dfTrimP, xlab="trade volume hourly", ylab="price
89             volatility hourly", main=paste(product, "Maturity:", maturity), col=
90             "pink3", pch=20)
91         dev.off()
92         png(paste(product, maturity, "daily.png", sep=""), width = 6,
93             height = 4, units = 'in', res = 300)
94         plot(dailyTrimP, xlab="trade volume daily", ylab="price
95             volatility daily", main=paste(product, "Maturity:", maturity), col=
96             "pink3", pch=20)
97         dev.off()
98     }
99 }
100 }
101
102 pvalsO=order(pvals)
103 corPvals=data.frame(prod, pvals)[pvalsO,]
104 nPvals=nrow(corPvals)
105 corPvals["id"]=1:nPvals #B-H Procedure
106 corPvals["q"]=nPvals*corPvals$pvals/corPvals$id
107 f=c()
108 for(i in 1:nPvals){f=append(f, min(corPvals$q[i:nPvals]), length(f))}
109 corPvals["f"]=f
110 corSig=corPvals[which(corPvals$pvals<.05),]
111 nSig=nrow(corSig)
112 nSig
113 corSig #4 rejected at FDR=.20
114
115 #uniform Q-Q plot
116 set.seed(3-3-17)
117 u=sort(runif(length(pvals)))
118 #png("unifQQ.png")
119 plot(u, corPvals$pvals, ylab="p-values", xlab="uniform", main="Q-Q Plot"
120 )
121 abline(0,1, col="red")
122 #dev.off()

```



## Bibliography

- [1] The data deluge, Feb 2010.
- [2] Hendrik Bessembinder and Paul J Seguin. Price volatility, trading volume, and market depth: Evidence from futures markets. *Journal of financial and Quantitative Analysis*, 28(01):21–39, 1993.
- [3] Andrew J Foster. Volume-volatility relationships for crude oil futures markets. *Journal of Futures Markets*, 15(8):929–951, 1995.
- [4] Frank Harrell. *Regression modeling strategies: with applications to linear models, logistic and ordinal regression, and survival analysis*. Springer, 2015.
- [5] Xiangrui Meng, Joseph Bradley, B Yuvaz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, D Tsai, Manish Amde, Sean Owen, et al. Mllib: Machine learning in apache spark. *JMLR*, 17(34):1–7, 2016.
- [6] Venkatesh Satish, Abhay Saxena, and Max Palmer. Predicting intraday trading volume and volume percentages. *The Journal of Trading*, 9(3):15–25, 2014.
- [7] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association, 2012.
- [8] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets. *HotCloud*, 10:10–10, 2010.
- [9] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

# Curriculum Vitae