Single-Stock Circuit Breakers

William David Brinda

Practical Work

Spring 2013

Contents

1 The Research Question

On May 6, 2010, the U.S. stock market experienced what has come to be called the "flash crash," in which the prices of various stocks fluctuated violently. Overall, the market rapidly plunged about nine percent and recovered minutes later. In respone, the SEC and stock exchanges devised the Single-Stock Circuit Breaker (SSCB) rules, which were implemented gradually. According to Traders Magazine,[1]

> The single-stock circuit breakers will pause trading in any component stock of the Russell 1000 or S&P 500 Index in the event that the price of that stock has moved 10 percent or more in the preceding five minutes. The pause generally will last five minutes, and is intended to give the markets a hiatus to attract trading interest at the last price, as well as to give traders time to think rationally.

It is important to note that the rules do not apply to the first fifteen minutes or the last fifteen minutes of each trading day.

Stock price fluctuations are not steady over the course of a day. They tend to be much stronger toward the beginning and the end of the trading day. We refer to this daily pattern as the "volatility profile."

Under the direction of Dr. Michael Kane, I am analyzing recent stock data. The aim of our research is to understand the effects of the SSCB rules. It is possible that they could "distort" the natural daily volatility patterns by suppressing volatility during the middle of the trading day. In general, this report addresses the research question: *do the SSCB rules have any effect on the daily volatility profiles of stocks?*

To answer the question, we must first define the concept of a volatility profile more carefully and decide how to estimate it from data, as described in Section 2. Then in Section 3, we retrieve the needed data estimate volatility profiles for a period before the SSCB rules were implemented and a period afterward. Finally, Section 4 attempts to answer our research question by using the fact that many stocks were not subject to the SSCB rules; they will act as a control group of sorts.

Much of the data referenced in this report is available for download from my blog.[2]

[1] http://www.tradersmagazine.com/news/single-stock-circuit-breakers-sec-flash-crash-trading-106018-1.html

[2] http://blog.quantitations.com/tutorial/2013/01/14/single-stock-circuit-breakers/

2 Volatility Profiles

2.1 Definition

Stock prices are generally modeled as Geometric Brownian Motion. Each stock is assumed to have a volatility parameter that is roughly stable over time frames on the order of, say, a year. In practice, stock prices tend to change much more rapidly at the beginning and end of each trading day than they do in the middle. To analyze intra-day volatilities, we need to use a more general diffusion model that allows the volatility σ to depend on t. I will refer to a stock's daily volatility function σ as its "volatility profile." A priori, we will not assume any parametric structure to σ other than a general "smoothness." As a proxy for a stock's σ, we will estimate the volatilities at a set of times spread along the trading day.

2.2 Estimation

2.2.1 Assuming Constant Volatility over Small Intervals

In theory, one could compute volatility estimates over arbitrarily small time intervals. However, the more you zoom in, the less "GBM-like" stock prices are. For our analysis, we broke each trading day ($T = 6.5$ hours) into seventy-eight five-minute ($l = 5$ minutes) intervals. On day i, represent a stock price in terms of a standard Brownian Motion W_i by

$$U_i(t) = \exp\left(b_{i,0} + t\mu + \sigma(t)W_i(t)\right)$$

We define $n = T/l = 78$ random variables, one for each interval:

$$\begin{aligned} X_{i,j} &:= \log\left(\frac{U_i(jl)}{U_i((j-1)l)}\right) \\ &= \log U_i(jl) - \log U_i((j-1)l) \\ &= (b_{i,0} + jl\mu + \sigma(jl)W_i(jl)) - (b_{i,0} + (j-1)l\mu + \sigma((j-1)l)W_i((j-1)l)) \\ &\approx l\mu + \sigma(jl - l/2)\left(W_i(jl) - W_i((j-1)l)\right) \qquad \text{by assumption explained below} \end{aligned}$$

As a first attempt, we are assuming $\sigma(jl) \approx \sigma((j-1)l) \approx \sigma(jl - l/2)$, that is, volatility is nearly constant over small intervals. For a given stock, each $X_{i,j}$ is normal with mean $l\mu$ and variance approximately l times $\sigma^2(jl - l/2)$ (henceforth abbreviated to σ_j^2), and they are independent of each other. Notice that σ_j^2 represents the squared volatility at the midpoint of the jth interval.

Assume we have data for m trading days. For a given stock, the σ function should be the same every day. The random variable

$$\sum_{i=1}^{m} (X_{i,j} - \bar{X}_{\cdot,j})^2 / (l\sigma_j^2)$$

has an approximately χ^2_{m-1} distribution, so the standard unbiased estimator of σ_j^2 is

$$Y_j := \frac{\sum (X_{i,j} - \bar{X}_{\cdot,j})^2}{l(m-1)} \tag{1}$$

The mean-squared error of this estimator is equal to its variance

$$\begin{aligned}
\mathrm{Var}\frac{\sum(X_{i,j}-\bar{X}_{\cdot,j})^2}{l(m-1)} &\approx \frac{\sigma_j^4}{(m-1)^2}\mathrm{Var}\frac{\sum(X_{i,j}-\bar{X}_{\cdot,j})^2}{l\sigma_j^2} \\
&= \frac{\sigma_j^4}{(m-1)^2}\mathrm{Var}Y \\
&= \frac{\sigma_j^4}{(m-1)^2}2(m-1) \qquad \text{because } \chi_r^2 \text{ has variance } 2r \\
&= \frac{2\sigma_j^4}{m-1}
\end{aligned}$$

Unfortunately, after glancing at some plots (such as Figure 7), the assumption of nearly constant volatility over five-minute intervals seems entirely untenable. Often, a volatility seems to change by a large proportion over the course of five minutes. In fact, we would have to use extremely small intervals to make this assumption remotely plausible. But small intervals of stock data are not very GBM-like, and can be problematic.

2.2.2 Assuming Linear Volatility over Small Intervals

After realizing this, I devised a more plausible assumption: *the change in volatility over any five-minute interval is approximately linear.* Let $V(t)$ be a process whose natural logarithm is a diffusion with linearly changing $\sigma(t)$. That is,

$$\begin{aligned}
\log V(t) &= v_0 + \mu t + \int_0^t \sigma(\tau)dW(\tau) \\
&= v_0 + \mu t + \int_0^t (\sigma_0 + m\tau)dW(\tau)
\end{aligned}$$

Let us zero in on the complicated part of this expression by defining $R(t) := \int_0^t(\sigma_0 + m\tau)dW(\tau)$. At any given time t, the behavior of the diffusion $R(t)$ is locally approximated by a Brownian Motion with volatility $\sigma_0 + mt$. In other words, $R(t+\delta) - R(t)$ should approach a $N(0, (\sigma_0 + mt)^2\delta)$ distribution for small δ.

By transforming the time axis of a standard Brownian Motion in just the right way, we can produce a much more familiar-looking process with this same behavior. We need to find a transformation D such that $W(D(t))$ has a "volatility" of $\sigma_0 + mt$ at any time t. A change in time of δ from time t must produce a change in D by $(\sigma_0 + mt)^2\delta$. That is, D satisfies $D(t+\delta) = D(t) + (\sigma_0 + mt)^2\delta$ in the limit. Rearranging, we find that a solution is $D(t) = \sigma_0^2 t + \sigma_0 mt^2 + m^2t^3/3$. So our final result is

$$W(\sigma_0^2 t + \sigma_0 mt^2 + m^2t^3/3)$$

This diffusion has the same infinitesimal behavior as $R(t)$ at all times, so they are identical processes. The following simulations support this result. We will use each method, in turn, to plot a diffusion on $(0, 1)$ with $\sigma(t) = 1 - t$ (i.e. linear with $\sigma_0 = 1$ and $m = -1$). First, we generate the desired diffusion sequentially. Simulated diffusions of this type are shown in Figure 1.

```
LinearVolDiffusion <- function(end=1, initial=0, mu=0,
                               sigma0=1, m=0, n=1000) {
  delta <- 1/n
  x <- delta*(0:n)
  y <- c(initial, rep(NA,n))
  for(i in 1:n) {
    t <- x[i] + delta/2
    y[i+1] <- y[i] + mu*delta + rnorm(1, sd=(sigma0 + m*t)*sqrt(delta))
  }
  return(cbind(x, y))
}

set.seed(1)
plot(c(0, 1),c(-1.25, 1.25), type="n", main="Volatility Changes by Step",
     xlab="Time", ylab="Diffusion")
for(i in 1:4) {
  lines(LinearVolDiffusion(m=-1), col=i+1)
}
```

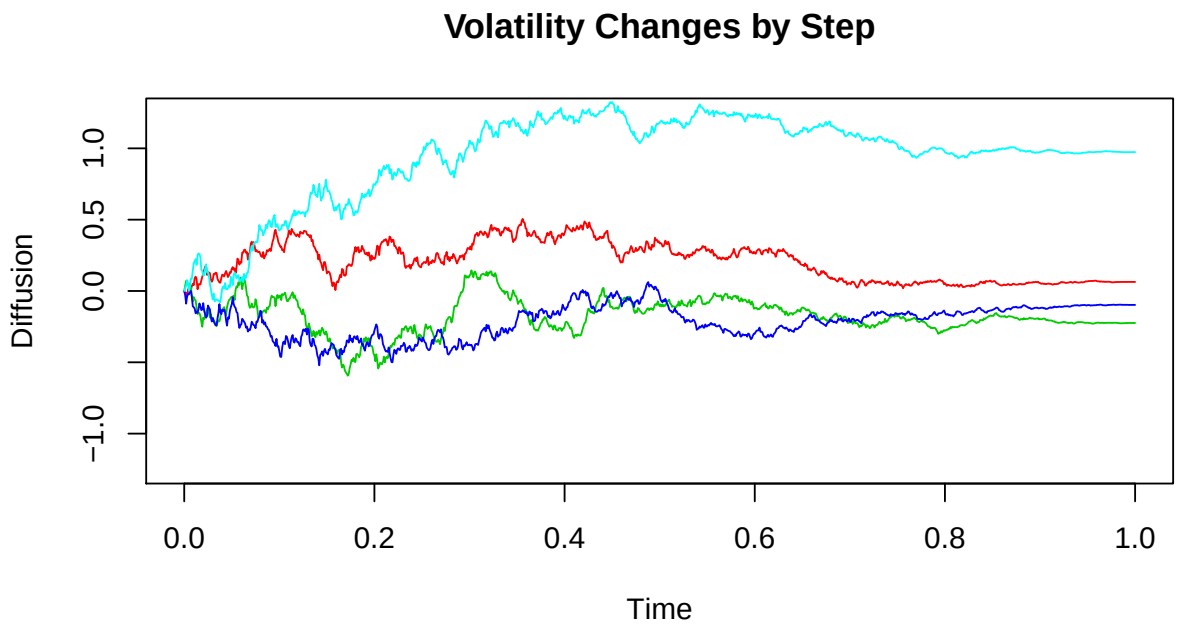

Figure 1: Each simulated diffusion is generated by concatenating a sequence of small Brownian Motions with different (linearly decreasing) volatilities.

Next, we generate the same diffusion using standard Brownian Motion and the D transformation discovered earlier. Simulated results are in Figure 2.

```
BMrecur <- function(x, y, sigma) {
  m <- length(x)
  mid <- (m + 1)/2
  delta <- x[m] - x[1]
  y[mid] <- (y[1] + y[m])/2 + rnorm(1, sd=sigma*sqrt(delta)/2)
  if (m <= 3) {
    return(y[mid])
  } else {
    return(c(BMrecur(x[1:mid], y[1:mid], sigma), y[mid],
             BMrecur(x[mid:m], y[mid:m], sigma)))
  }
}

GenerateBM <- function(end=1, initial=0, mu=0, sigma=1, log2n=10,
                       geometric=F) {
  # Based on code written in Spring 2012 for a Stochastic Processes project
  n <- 2^log2n
  x <- end*(0:n)/n
  final <- initial + mu * end + rnorm(1, sd=sigma*sqrt(end))
  y <- c(initial, rep(NA, n - 1), final)
  y[2:n] <- c(BMrecur(x, y, sigma))
  if (geometric) y <- exp(y)
  return(cbind(x, y))
}

right <- function(f, b=10) {
  while (f(b) < 0) b <- 10*b
  return(b)
}

D <- function(t, sigma0=1, m=-1) {
  return(sigma0^2*t + sigma0*m*t^2 + m^2*t^3/3)
}

Dinv <- function(x, ...) {
  sub <- function(t) D(t, ...) - x
  return(uniroot(sub, lower=0, upper=right(sub))$root)
}

plot(c(0,1),c(-1.25,1.25), type="n", xlab="Time", ylab="Diffusion",
     main="Standard Brownian Motion with Transformed Time Axis")
for(i in 1:4) {
  b <- GenerateBM(end=D(1))
  lines(sapply(b[,1], Dinv), b[,2], col=i+1)
}
```

The resemblance between Figures 1 and 2 gives us some reassurance that our results are correct. Now consider the distribution of $\log V(l)/V(0)$.

$$\begin{aligned}
\log V(l)/V(0) &= \log V(l) - \log V(0) \\
&= [v_0 + \mu l + W(\sigma_0^2 l + \sigma_0 m l^2 + m^2 l^3/3)] - [v_0] \\
&= \mu l + W(l[(\sigma_0 + ml/2)^2 + (ml)^2/12]) \\
&\approx \mu l + W(l(\sigma_0 + ml/2)^2) \qquad \text{if } ml \text{ is small}
\end{aligned}$$

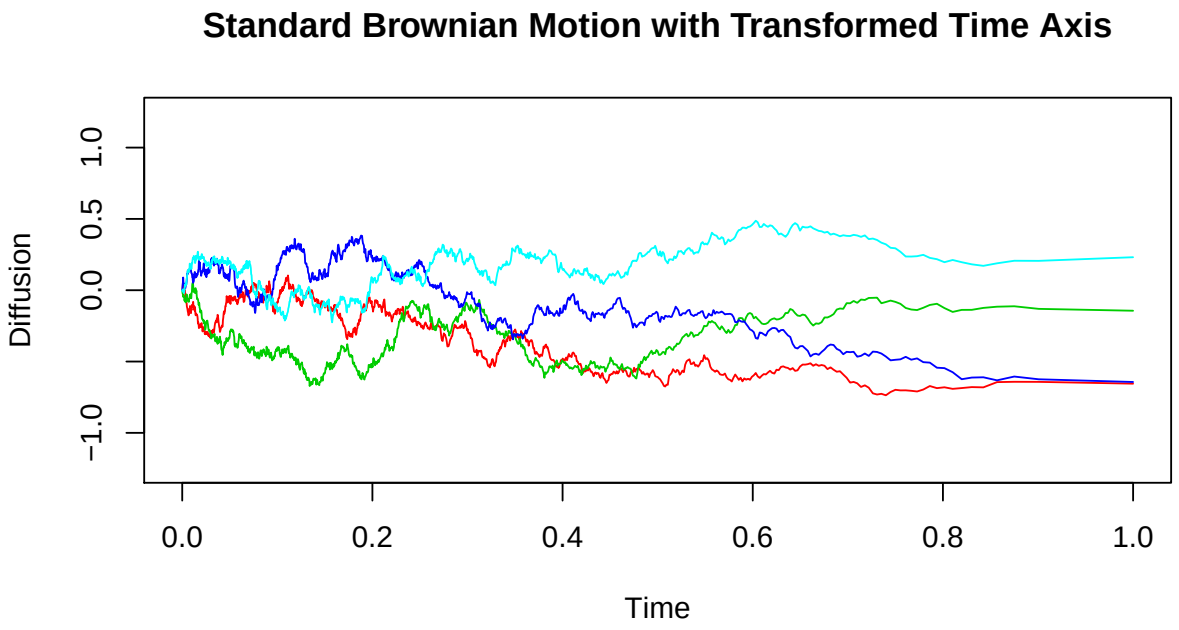

Figure 2: Each simulated diffusion is generated by transforming the time axis of a standard Brownian Motion.

It is normally distributed with variance approximately $l(\sigma_0 + ml/2)^2$, assuming the product of m and l, which is the change in σ over the interval, is much smaller than one. Plots indicate that this assumption holds up quite well overall (though not perfectly). Therefore it is easy to see that estimating the variance from a sample of random variables with this distribution allows us to estimate $(\sigma_0 + ml/2)^2$. Conveniently, this is the squared volatility at time $l/2$, the midpoint of the first interval $(0, l)$. Earlier, we called this quantity σ_1^2. Likewise, it can be shown that the same process provides an estimate for the squared volatilities at the midpoints of the n consecutive intervals of length l from $(0, T)$.

The upshot is, we can still use the same estimator Y_j defined in Equation (1) to estimate the same quantity σ_j^2, the squared volatility at the midpoint of the jth interval However, now we have justified ourselves with a more believable model.

For simplicity, our analysis will use these estimated values of the *squared* volatility, rather that estimates of the volatility itself. They are just as good for answering our research question. After all, squared-volatility profiles have a one-to-one correspondence with volatility profiles.

3 Gathering and Processing the Data

3.1 Gathering Stock Trades

To estimate a stock's volatility profile in the way described in Section 2, we need to know its price at the beginning and end of every 5-minute interval for a set of trading days. This information, and more, can be found in Trade and Quote (TAQ) data, which records the trades that occur every day.

The SSCB rules began taking effect on December 10th, 2010.[3] We decided to select a period of time before the SSCB rules were enacted and a period after the rules were in place to look for differences. The two periods that we chose have about the same level of overall volatility, based on the Volatility Index (VIX), which had a value of around 30 during both periods. This level of volatility is extraordinarily high, which should make the SSCB rules more likely to have an effect.

Our Pre-SSCB period is May 17, 2010 through June 10, 2010; our Post-SSCB period is August 4, 2011 through August 29, 2011. Each period has eighteen trading days. For each of the trading days, we downloaded the full TAQ record from the Wharton Research Data Services website, using the Yale Center for Analytical Sciences subscription.

To demonstrate this data, here are the first few rows of a TAQ file:

```
SYMBOL,DATE,TIME,PRICE,SIZE,CORR,COND,EX
A,20100104,9:30:02,31.32,98,0,Q,T
A,20100104,9:30:50,31.39,100,0,F,T
A,20100104,9:30:50,31.4,300,0,F,T
A,20100104,9:30:50,31.41,100,0,Q,P
```

"SYMBOL" refers to the stock's ticker symbol. "DATE" is a string comprising the year, month, and day of the transaction; it is the same in every row of a given TAQ file. "TIME" is the hour, minute, and second of the transaction, in Eastern Standard Time. "PRICE" is the amount of money exchanged per share of stock in the transaction. "SIZE" is the number of shares traded in the transaction. The other columns are unimportant for my analysis.

This data will allow me to compute a price at any given time. It also tells me how frequently each stock is traded, which is information that I will make use of in my analysis.

Trading frequency is likely to be a confounding variable. This is because the set of stocks that were subjected to the SSCB rules are systematically different from rest. The rules only applied to any stock in the S&P 500 Index. These stock indices comprise only stocks with very large market capitalizations, and they are traded more frequently than most.

3.2 Gathering a List of Stocks

It is easy to find a list of the current S&P 500 stocks, but the set of stocks changes over time. It is not as simple to get the stock list for a particular date in history. Fortunately, I found a list of the recent changes that have been made, along with the dates of those changes. Using this information, we can backtrack and find the set of stocks that were in the S&P during the periods of interest. The code below finds the 485 stocks that were in the S&P 500 Index during both of our periods.

```
html <- function(url) {
  # Attempts to retrieve html from url twice
  x <- NA
  try(x <- htmlParse(url), silent = T)
  if (is.na(x)) {
    Sys.sleep(5)  # Pause and try again
    try(x <- htmlParse(url), silent = T)
    if (is.na(x))
      print(paste("Error: Could not connect to", url))
```

[3] http://www.tradersmagazine.com/news/single-stock-circuit-breakers-sec-flash-crash-trading-106018-1.html

```
  }
  return(x)
}
url <- 'http://en.wikipedia.org/wiki/List_of_S%26P_500_companies'
x <- html(url)
# List of current S&P 500 stocks
pattern <- "//table[@class='wikitable sortable'][1]/tr/td[1]/a/text()"
index <- xpathSApply(x, pattern, xmlValue)
# List of changes and dates
pattern <- "//table[@class='wikitable sortable'][2]/tr"
changes <- xpathSApply(x, pattern, xmlValue)
changes <- changes[-(1:2)]
n <- length(changes)
new <- rep(NA, n)
old <- rep(NA, n)
dates <- rep(NA, n)
for(i in 1:n) {
  v <- strsplit(changes[i], "\n")[[1]]
  if(nchar(v[1]) > 10) {
    dates[i] <- v[1]
    new[i] <- v[2]
    old[i] <- v[4]
  } else {
    dates[i] <- dates[i-1]
    new[i] <- v[1]
    old[i] <- v[3]
  }
}
dates <- strptime(dates, "%B %d, %Y")
historicSP <- function(date) {
  # Returns the 500 S&P stocks on the date given
  stocks <- index
  date <- strptime(date, "%Y-%m-%d")
  for(i in 1:n) {
    if(dates[i] < date) break
    stocks[stocks==new[i]] <- old[i]
  }
  return(stocks)
}
# No stocks changed during the periods themselves,
# but several changes happened between the two periods.
SP2010 <- historicSP("2010-05-17")
SP2011 <- historicSP("2011-08-04")
SP <- intersect(SP2010, SP2011)
writeLines(SP, "SP.txt")
```

For comparison, we need a "control" group of stocks. I decided to use the Russell 1000 Index stocks[4] as the control. In fact, nearly all of the S&P 500 stocks are in the Russell 1000, so our control group will actually be the set difference, which ends up leaving a total of 507 stocks. These stocks have a large market capitalization and are actively traded, so they are likely to be comparable. Incidentally, the Russell 1000 stocks came under the SSCB rules as well, but that started after the time periods of our analysis.[5]

[4] www.russell.com/indexes/documents/Membership/Russell1000_Membership_list.pdf
[5] http://www.sec.gov/investor/alerts/circuitbreakers.htm

```
R <- readLines("Russell.txt")
# Remove any stocks from SP2010 or SP2011
R <- setdiff(R, SP2011)
R <- setdiff(R, SP2010)
writeLines(R, "Russell.txt")
```

3.3 Gathering Market Capitalizations

To attempt to control for any systematic differences between the two groups, we decided to take market capitalization into account. Market capitalization "is the total value of the issued shares of a publicly traded company; it is equal to the share price times the number of shares outstanding."[6] We need to determine the market cap of stocks at particular past dates. Our TAQ data gives us the share prices for these dates, so we only needed to find the number of shares outstanding during the periods. I wrote code to scrape the web and estimate past shares outstanding.

My code estimates the number of shares outstanding on any historical date by considering the current number of shares outstanding and the splits that have occurred since that date. This procedure is definitely not perfect! Splits have a big impact on the number of shares outstanding, but other factors[7] can affect the number as well. Hopefully, the fluctuations caused by the other factors are relatively small. Regardless, the other factors are much harder to get historical data for.

The current number of shares outstanding can be retrieved from Yahoo! Finance.[8] Unfortunately, this database is incomplete; in particular, many small stocks are missing. Split histories can be found at GetSplitHistory;[9] it was also missing many stocks.

My code uses the `XML` package and the XPath specification for parsing XML. Also, because web scraping can take a long time and is particularly vulnerable to failure, I created an `append.csv` function. It behaves much like `write.csv`, but it builds the csv file one line at a time, in order to save your progress as you go.

3.3.1 Scraping Yahoo! Finance and GetSplitHistory

The functions that perform the scraping are posted below. It relies on the `html` function defined in Section 3.2.

```
outstanding <- function(symbol, dates) {
# Scrapes finance.yahoo.com and getsplithistory.com to
# determine the number of outstanding shares of a stock
# on any given date. Works as of early January 2013, but
# may stop working if either site changes its html.

  # First get current number of shares from Yahoo
  yahooURL <- 'http://finance.yahoo.com/q/ks?s=XXX'
  thisURL <- gsub('XXX', symbol, yahooURL)
  x <- html(thisURL)
  if(x$fail) return(rep(NA, length(dates)))
  pattern <- "//td[preceding-sibling::td[text()='Shares Outstanding']]/text()"
  y <- xpathSApply(x$html, pattern, xmlValue)
  if(length(y)==0) {
    print(paste(symbol, "Error: Yahoo has no data"))
    return(rep(NA, length(dates)))
  }
  power <- switch(substr(y, nchar(y), nchar(y)), M=6, B=9)
  currentshares <- as.numeric(substr(y, 1, nchar(y)-1))*10^power
```

[6] http://en.wikipedia.org/wiki/Market_capitalization
[7] http://answers.yahoo.com/question/index?qid=20061026104715AAJUVbe
[8] http://finance.yahoo.com/
[9] http://getsplithistory.com/

```
  # Next, use split history to determine past shares
  gshURL <- 'http://getsplithistory.com/XXX'
  thisURL <- gsub('XXX', symbol, gshURL)
  x <- html(thisURL)
  if(x$fail) return(rep(NA, length(dates)))
  pattern <- "//table[@id='table-splits']/tbody/tr[@class='2']"
  y <- xpathSApply(x$html, pattern, xmlValue)
  if(length(y)==0) {
    pattern <- "//b[text()='404 error.']"
    y <- xpathSApply(x$html, pattern, xmlValue)
    if(length(y)>0) {
      print(paste(symbol, "Error: GetSplitHistory has no data"))
      return(rep(NA, length(dates)))
    }
    # stock has never split
    return(rep(currentshares, length(dates)))
  }
  splitdates <- strptime(substr(y, 1, 12), "%b %d, %Y")
  ratios <- as.numeric(substr(y, 13, 13)) /
            as.numeric(substr(y, 17, 17))
  results <- rep(currentshares, length(dates))
  for(i in 1:length(dates)) {
    j <- 1
    while(dates[i] <= splitdates[j]) {
      results[i] <- results[i]/ratios[j]
      if(j==length(splitdates)) break
      j <- j+1
    }
  }
  return(results)
}
append.csv <- function(row, rowname="", file="myfile.csv") {
  # Adds one row to a csv file
  if(file.exists(file)) {
    cat(rowname, row, file=file, sep=",", append=T)
  } else { # row is assumed to be the header
    cat(row, file=file, sep=",")
  }
  cat("\n", file=file, append=T)
}
```

For any given stock symbol, I can call the `outstanding` function, passing in the desired dates, to get an estimate of that stock's number of shares outstanding on those dates.

```
library(XML)
symbols <- c(readLines("SP.txt"), readLines("Russell.txt"))
dates <- c("20100517", "20100610", "20110804", "20110829")
append.csv(dates, "outstanding.csv")
results <- matrix(NA, length(symbols), length(dates))
colnames(results) <- dates
rownames(results) <- symbols
dates <- strptime(dates, "%Y%m%d")
for(i in 1:length(symbols)) {
  results[i,] <- outstanding(symbols[i], dates)
```

```
  append.csv(results[i,], "outstanding.csv", rowname = symbols[i])
  Sys.sleep(5)
}
```

This creates a file "outstanding.csv" with the number of shares outstanding on each date for each stock. For example,

```
,20100517,20100610,20110804,20110829
A,347930000,347930000,347930000,347930000
AA,1.07e+09,1.07e+09,1.07e+09,1.07e+09
AAP,73170000,73170000,73170000,73170000
AAPL,940690000,940690000,940690000,940690000
```

For any past date, I will multiply a stock price by the number of shares outstanding to estimate a stocks market capitalization on that date.

3.4 Processing the Data

3.4.1 Splitting up the TAQ Files

Each raw TAQ data file is too big to open all at once using, for example, `read.csv`. Attempting to open one on our machine (with 8 GB of RAM) results in memory swapping, slowing the processing to a practical standstill.

Instead, I processed the TAQ files by reading in one line at a time, which uses a negligible amount of memory. I wrote a Python script "TAQprocess.py" which is shown below. The code is object-oriented, which slows it down somewhat. But in this case, making the code easier to read and manage seemed worth the added computation time.

```
# Based on code written for a Statistical Computing assignment
import sys
import os
import string
def toSeconds(time):
  timev = time.split(":")
  return(3600*int(timev[0]) + 60*int(timev[1]) + int(timev[2]))
class Line:
  def __init__(self, ifile):
    self.ifile = ifile
    skip = self.ifile.readline()
    skip = self.next()
  def next(self):
    linev = self.ifile.readline().split(",", 5)
    if len(linev) == 1:
      return False
    self.symbol = linev[0]
    self.time = toSeconds(linev[2])
    self.price = float(linev[3])
    self.size = int(linev[4])
    return True
class Stock:
  def __init__(self, symbol):
    self.symbol = symbol
    self.ofile = open("Processed/%s/%s.csv" % (sys.argv[1], symbol), "w")
    self.write("time,price\n")
  def next(self, symbol):
```

```
    self.close()
    self.__init__(symbol)
  def write(self, text):
    self.ofile.write(text)
  def close(self):
    self.ofile.close()
class Second:
  def __init__(self, line):
    self.time = line.time
    self.spent = line.price*line.size
    self.volume = line.size
  def add(self, line):
    self.spent += line.price*line.size
    self.volume += line.size
  def next(self, stock, line):
    self.write(stock)
    self.__init__(line)
  def write(self, stock):
    stock.write("%s,%s\n" % (self.time, round(self.spent/self.volume, 2)))
print "Processing %s." % sys.argv[1]
os.makedirs("Processed/%s" % sys.argv[1])
os.system("gunzip -c TAQ/taq_%s_trades_all.csv.gz > %s" % (sys.argv[1], sys.argv[1]))
ifile = open(sys.argv[1])
line = Line(ifile)
stock = Stock(line.symbol)
second = Second(line)
while line.next():
  if line.symbol != stock.symbol:
    second.next(stock, line)
    stock.next(line.symbol)
  elif line.time == second.time:
    second.add(line)
  else:
    second.next(stock, line)
second.write(stock)
stock.close()
ifile.close()
os.remove(sys.argv[1])
```

The following R code was used to run the script on each date of interest. It processes the files in parallel, making use of all twelve cores on Dr. Kane's machine.

```
require(foreach)
require(doMC)
registerDoMC()
taqfiles <- list.files("TAQ")
taqfiles <- taqfiles[grep("20.*_trades", taqfiles)]
taqdates <- substring(taqfiles, 5, 12)
taqdates <- c(taqdates[taqdates >= "20100517" & taqdates <= "20100610"],
              taqdates[taqdates >= "20110804" & taqdates <= "20110829"])
a <- foreach(taqdate=taqdates, .combine=c) %dopar% {
  command <- paste("python TAQprocess.py", taqdate)
  system(command)
}
```

Now we have one folder for each date of interest. Within that folder, there is a file for each stock that was represented on that day's TAQ file. This stock's file has a row for each second of the day during which trades occurred. Each row consists of two columns: time of day (in seconds) and weighted average trade price. For example,

```
time,price
34201,52.41
34205,52.4
34210,52.41
34222,52.4
```

3.4.2 Calculating Market Capitalization

For a handful of stocks, I was unable to retrieve Yahoo! Finance or GetSplitHistory data, so they were thrown out. Also, one stock split during the period of study; it was tossed out for simplicity. This left 473 S&P stocks and 488 Russell stocks. Next, the stocks that were not traded every day during the periods of interest were discarded. This leaves 464 S&P stocks and 430 Russell stocks. The list of all remaining stocks was stored in the file "symbols.txt."

```
# Discard stocks based on failure to acquire shares outstanding data
shares <- read.csv("outstanding.csv", row.names=1)
shares <- shares[complete.cases(shares), ]
shares <- shares[-which(shares[, 1] != shares[, 4]), ]
stocks <- rownames(shares)
# Discard stocks based on insufficient TAQ data
setwd("Processed")
dates <- list.files()
for (date in dates[1:length(dates)]) {
  files <- list.files(date)
  datestocks <- substr(files, 1, nchar(files) - 4)
  stocks <- intersect(stocks, datestocks)
}
o <- order(stocks)
stocks <- stocks[o]
shares <- shares[o, ]
writeLines(stocks, "symbols.txt")
shares <- shares[which(rownames(shares) %in% stocks), ]
write.csv(shares, "outstanding.csv")
```

Finally, multiplying the number of shares outstanding by the share price tells us the market capitalization of each stock for each period. Of course, each stock does not have a single price for each period. Instead, I compute the average price.

```
require(foreach)
require(doMC)
registerDoMC()
DayAverage <- function(stock, date) {
  filename <- paste0(date, "/", stock, ".csv")
  x <- read.csv(filename)
  n <- length(x$time)
  if(n==1) return(x$price[1])
  total <- 0
  for(i in 1:(n-1)) {
    total <- total + x$price[i]*(x$time[i+1]-x$time[i])
```

```
  }
  total <- total + x$price[n]*(57600-x$time[n])
  avg <- total/(57600-x$time[1])
  return(avg)
}
# Detemine Average Price for each Stock in each Period
stocks <- readLines("symbols.txt")
n <- length(stocks)
setwd("Processed")
dates <- list.files()
years <- c("2010", "2011")
prices <- matrix(NA, n, length(years),
                 dimnames=list(stocks, years))
for(i in 1:length(years)) {
  yeardates <- dates[grep(years[i], dates)]
  m <- length(yeardates)
  prices[,i] <- foreach(stock=stocks, .combine=c) %dopar% {
    print(paste(years[i], stock))
    avgs <- rep(NA, m)
    for(j in 1:m) {
      avgs[j] <- DayAverage(stock, yeardates[j])
    }
    return(mean(avgs))
  }
}
# Multiply Price by Shares Outstanding to get Market Cap
setwd("..")
shares <- read.csv("outstanding.csv", row.names=1)
cap <- cbind(shares[,1]*prices[,1], shares[,3]*prices[,2])
colnames(cap) <- years
write.csv(cap, "cap.csv")
```

Figure 3 gives a glance at the spread of market caps and the relationships among market caps for the two periods.

```
par(mfrow=c(1, 2))
boxplot(log(cap), main="(a) Market Capitalizations", ylab="Natural Log of Dollars")
plot(log(cap), main="(b) Natural Log of Market Caps")
```

Because the market caps of the two years are so highly correlated (R^2 is over .94), I will simply take the means of the two years to be the single market cap for each stock.

```
cap.mean <- apply(cap, 1, mean)
write.csv(cap.mean, "capmean.csv")
```

3.5 Cleaning the Data

On some holidays the stock market is only open for a shortened trading day. If any partial trading days are present in our data set we should throw them out. I would guess that any partial trading days should have noticeably less trading activity than ordinary trading days. Figure 4 shows boxplot of trading activity. Because we see no lower outliers, I assume there are no partial trading days in my data set.

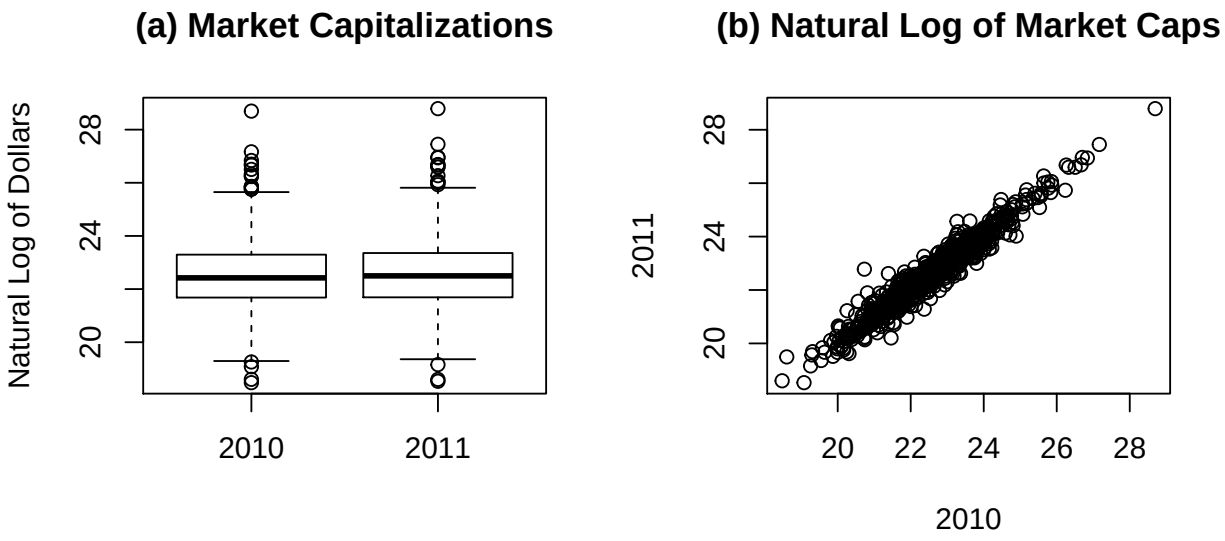

Figure 3: (a) Market capitalizations of all stocks during the 2010 period and the 2011 period. (b) Scatterplot showing how closely related the market caps are between the two periods.

```
stocks <- readLines("symbols.txt")
n <- length(stocks)
setwd("Processed")
dates <- list.files()
# For each stock, record the number of seconds in which trades
# occurred on each date of interest.
activity <- matrix(NA, n, length(dates),
                   dimnames=list(stocks, dates))
for(i in 1:length(dates)) {
  files <- paste0(dates[i], "/", stocks, ".csv")
  for(j in 1:n) {
    command <- paste("wc -l", files[j])
    r <- system(command, intern=T)
    activity[j,i] <- as.integer(unlist(strsplit(r, " "))[1])-1
  }
}
write.csv(activity, "activity.csv")
```

```
act.daymeans <- apply(activity, 2, mean)
boxplot(log(act.daymeans), main="Trading Activity Each Day",
        ylab="Log of Avg Num Seconds in which Trades Occurred")
```

Ultimately, we want to compare the SSCB stocks to the others, in hopes of seeing differences caused by the SSCB rules. To that end, we will try to control for some systematic differences, including trading activity. As a first step, I want to remove the control stocks whose trading activity is far below that of the SSCB stocks in order to make the groups more similar. Figure 5 shows how dissimilar the groups are.

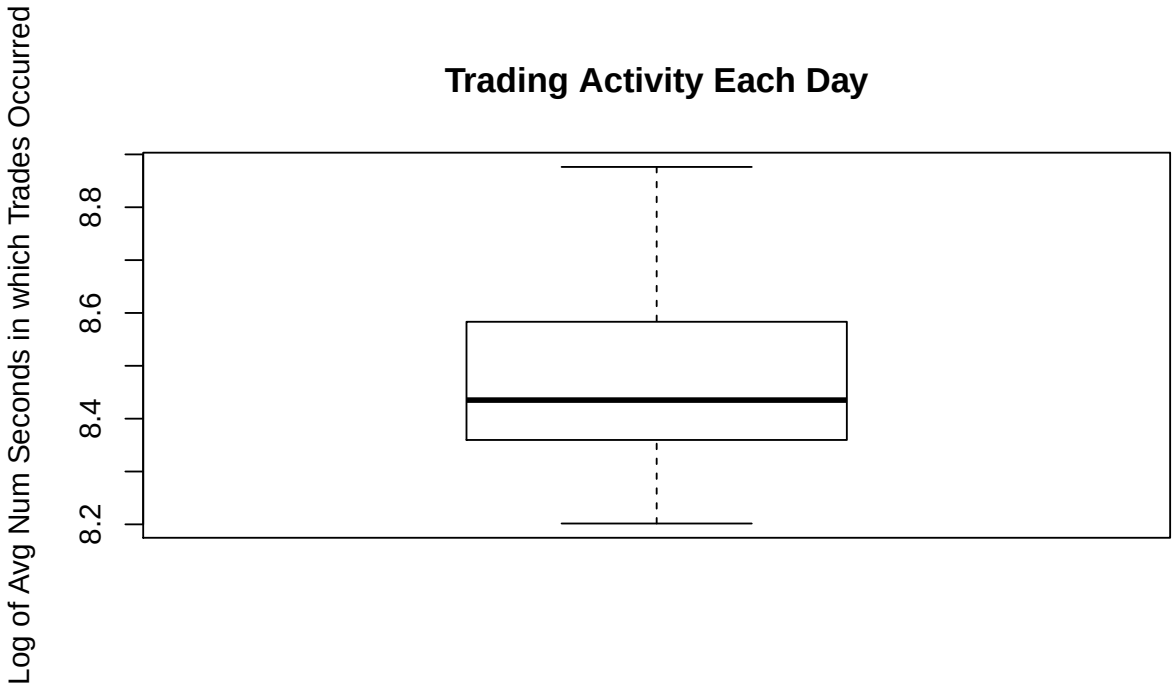

Figure 4: Boxplot of average number of seconds in which trades occurred over all stocks.

```
SSCBstocks <- readLines("SP.txt")

mins <- apply(activity, 1, min)
sscb <- names(mins) %in% SSCBstocks
boxplot(log(mins) ~ sscb, names=c("Control", "SSCB"), main="Trading Activity Each Stock",
        ylab="Log of Min Num Seconds in which Trades Occurred")
```

I decided to discard any stock that had a day of fewer than 400 seconds of trading. The control group still skews lower, but at least they are in the same ballpark now, as Figure 6 demonstrates. Another benefit of discarding stocks whose activity is below this minimum threshold is that their data is less reliable.

```
mins <- mins[mins >= 400]
sscb <- names(mins) %in% SSCBstocks
boxplot(log(mins) ~ sscb, names=c("Control", "SSCB"), main="Trading Activity Each Stock",
        ylab="Log of Min Num Seconds in which Trades Occurred")
writeLines(names(mins), "symbols_final.txt")
```

At this point, we have 463 SSCB stocks and 404 control stocks remaining.

As we did with the market cap data, we will average the activity levels of the two periods into a single overall measure of trading frequency for each stock. We will control for these values in Section 4.3.2.

```
activity <- activity[rownames(activity) %in% names(mins), ]
act.mean <- apply(activity, 1, mean)
write.csv(act.mean, "actmean.csv")
# Also, synchronize mean market caps with final list of stocks
cap.mean <- cap.mean[names(cap.mean) %in% names(mins)]
write.csv(cap.mean, "capmean.csv")
```

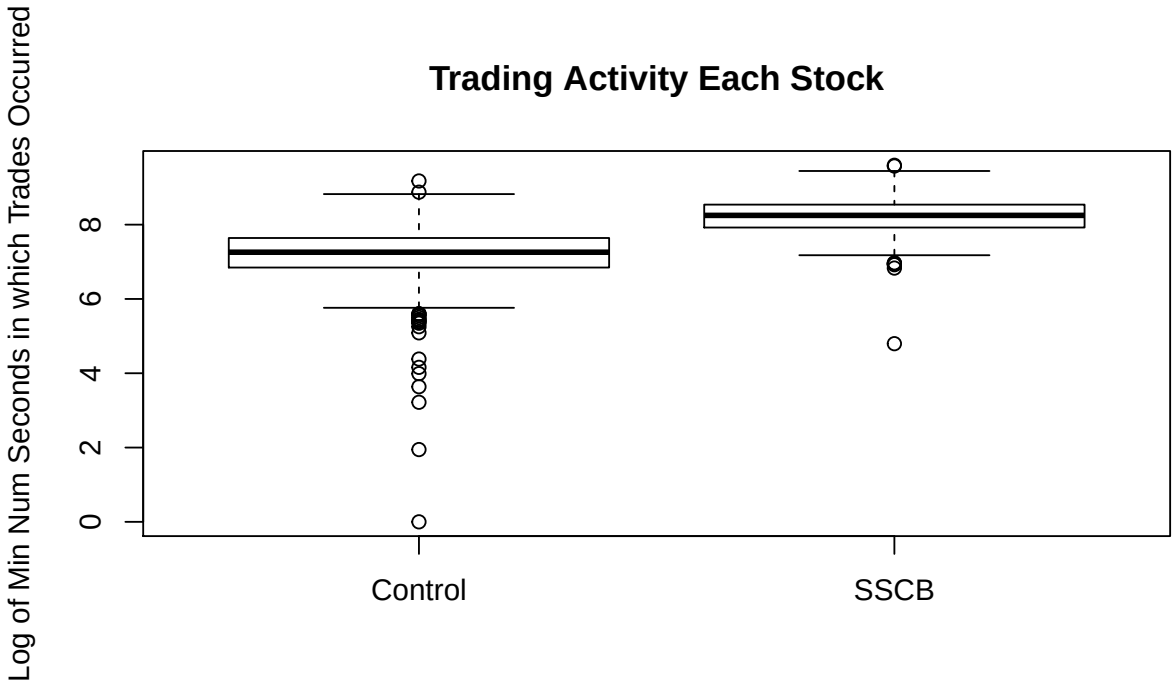

Figure 5: For each stock, we have 36 trading days worth of activity. These boxplots correspond to each stock's least active day.

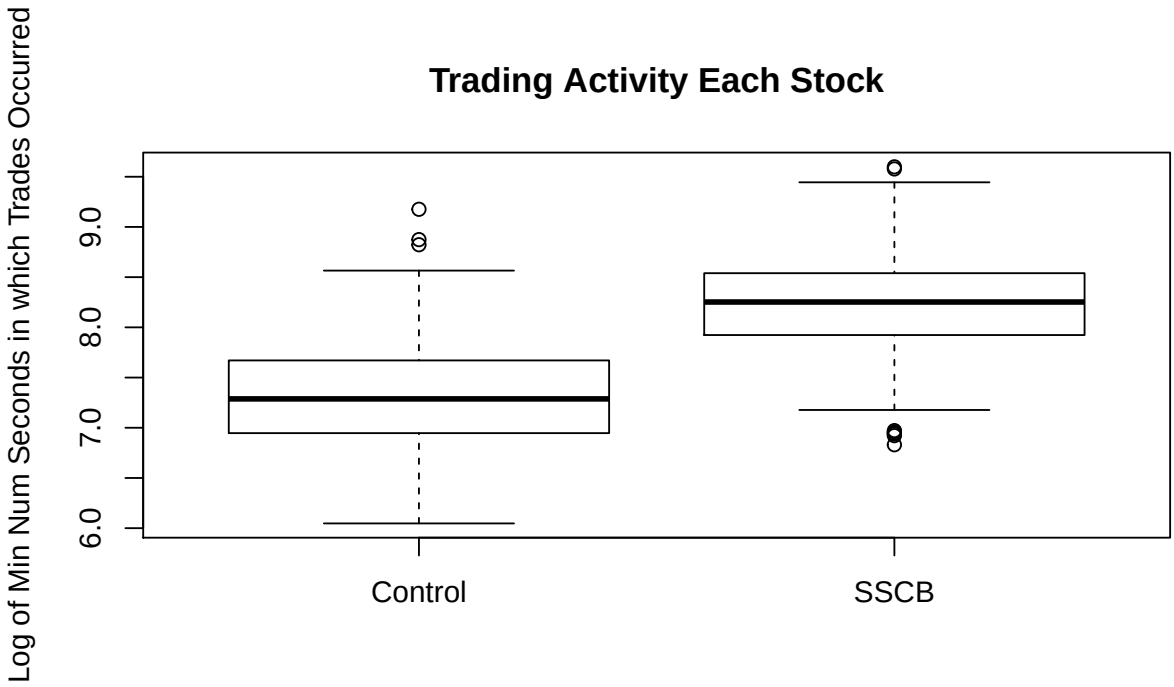

Figure 6: Boxplots of minimum trading activity after removing all stocks below the threshold of 400 seconds.

3.6 Computing Volatility Estimates

Now that the stock data has been simplified, cleaned, and organized into manageable chunks, we can estimate the volatility profiles. Notice that far outliers are discarded by this code. Otherwise, they tend to dominate the results. There are certainly interesting things we could learn by analyzing the outliers, but our present research question is about the ordinary volatility profiles of stocks rather than the extraordinary events.

```
require(foreach)
require(doMC)
registerDoMC()
Prices <- function(S, times) {
  m <- length(S$price)
  n <- length(times)
  p <- rep(NA, n)
  j <- 1
  for(i in 1:n) {
    while(j < m) {
      if(S$time[j+1] > times[i]) break
      j <- j+1
    }
    p[i] <- S$price[j]
  }
  return(p)
}
X <- function(S, l=5) {
  times <- seq(34200, 57600, by=60*l)
  p <- Prices(S, times)
  return(p[-1]/p[-length(times)])
}
out.discard <- function(x, threshold = 3) {
    scores <- (x - mean(x)) / sd(x)
    return(x[abs(scores) <= threshold])
}
Y <- function(x, l=5) {
  x <- out.discard(x)
  num <- sum((x-mean(x))^2)
  denom <- (length(x)-1)*l/60/6.5/252
  return(num/denom)
}
Profile <- function(stock, dates, period, l=5) {
  if(390%%l) stop("Error: Interval length should divide 390.")
  stockfiles <- paste0("Processed/", dates, "/", stock, ".csv")
  print(paste("Profiling", stock))
  M <- matrix(NA, length(dates), 390/l)
  for(i in 1:length(dates)) {
    S <- read.csv(stockfiles[i])
    M[i,] <- X(S, l)
  }
  filename <- paste0("Profiles/", period, "/", stock, ".csv")
  print(paste("Creating", filename))
  write.csv(M, filename)
  result <- rep(NA, 390/l)
  try(result <- apply(M, 2, Y), silent=T)
  return(result)
```

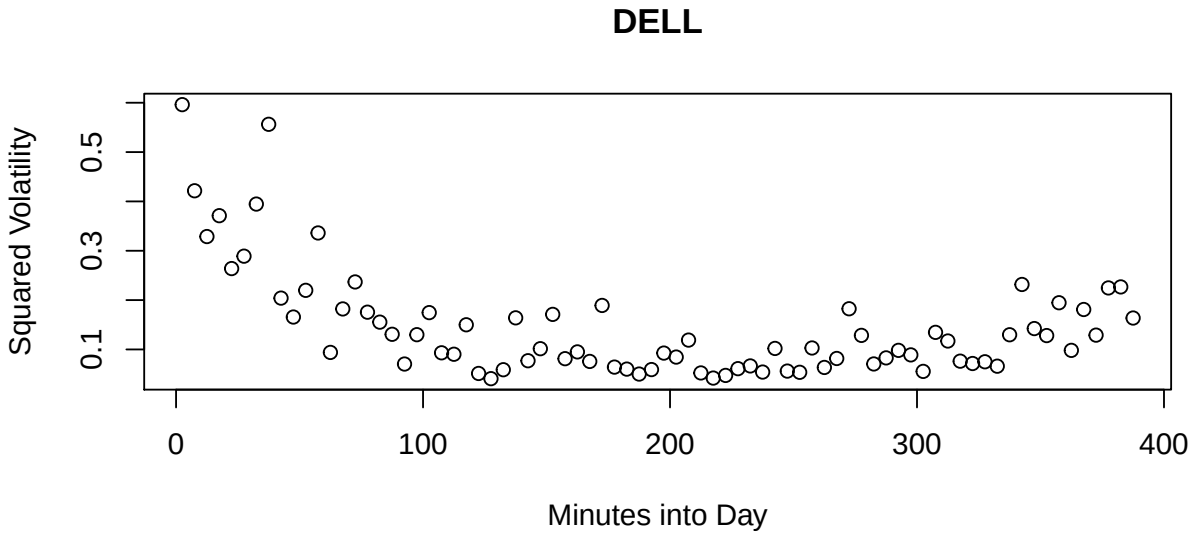

Figure 7: 78 estimated points along the squared volatility profile of a random stock.

```
}
stocks <- readLines("symbols_final.txt")
dates <- list.files("Processed")
for(year in c("2010", "2011")) {
  yeardates <- dates[grep(year, dates)]
  a <- matrix(unlist(mclapply(stocks, Profile, yeardates, year)),
              nrow=length(stocks), byrow=T)
  rownames(a) <- stocks
  write.csv(a, paste0(year, ".csv"))
}
```

Now, we have two files: "2010.csv" and "2011.csv." Each file has 867 rows, one for each stock that we will analyze. There are 78 columns corresponding to the midpoints of the 78 five-minute intervals of the trading day. Cell $[i, j]$ of the matrix contains the estimated volatility of stock i at the midpoint of interval j.

Finally, here is a typical example of a squared volatility profile estimate.

```
s2010 <- read.csv("2010.csv", row.names=1)
```

```
set.seed(1)
i <- runif(1, 1, nrow(s2010))
t <- 2.5 + 5 * (0:77)
plot(t, s2010[i,], main=rownames(s2010)[i],
     xlab="Minutes into Day", ylab="Squared Volatility")
```

4 Analysis

4.1 Transformation

It is clear to me from browsing many more plots like Figure 7 that the first and last points of the day tend to be the highest. They also seem to have the most variability. In fact, the spread versus level plot in Figure 8a confirms that a strong relationship exists between spread and level. Times of the day with higher volatilities also have higher variation in their volatilities. Generally, analyses go more smoothly if this relationship can be transformed away. A log transform does the trick, as Figure 8b shows.

```
s2010 <- read.csv("2010.csv", row.names=1)
s2011 <- read.csv("2011.csv", row.names=1)
```

```
par(mfrow=c(1, 2))
means <- apply(s2010, 2, mean)
sds <- apply(s2010, 2, sd)
plot(means, sds, xlab = "Level", ylab = "Spread", main = "(a) Original Data")

trans <- log(s2010)
means <- apply(trans, 2, mean)
sds <- apply(trans, 2, sd)
plot(means, sds, xlab = "Level", ylab = "Spread", main = "(b) Log Transformed Data")
```

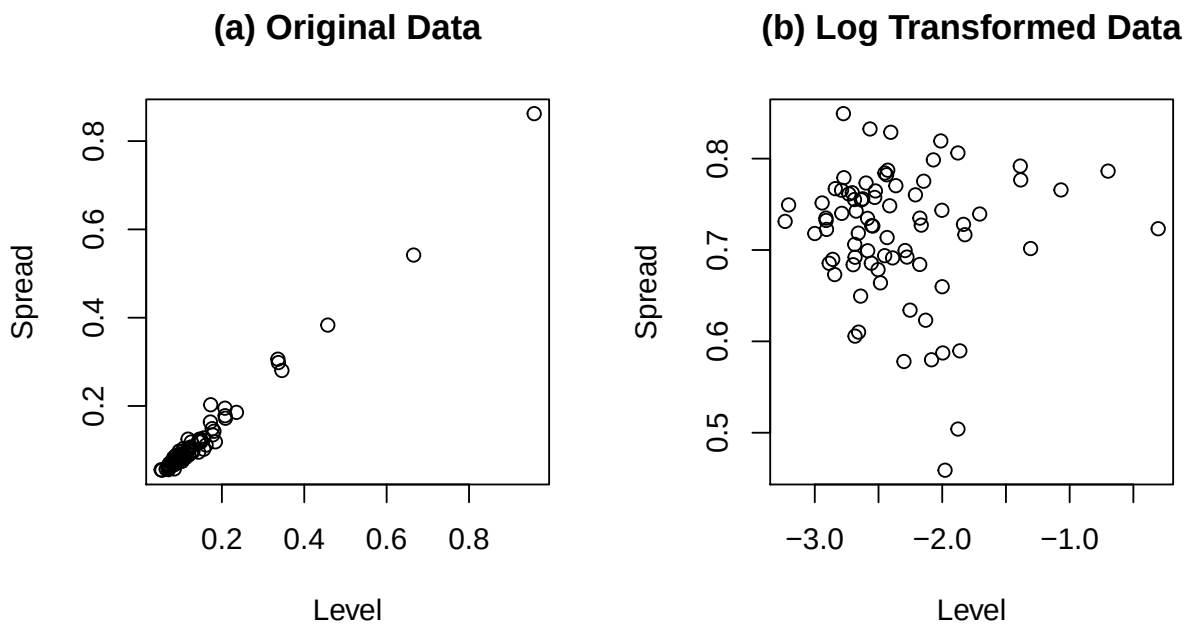

Figure 8: (a) In the original spread versus level plot, the spreads are strongly correlated with level. (b) After log-transforming, this relationship basically vanishes.

Let us transform both the 2010 and 2011 data.

```
# One stock has a zero in 2010, which would cause problems
# We remove it from the analysis: SIRI (control group)
mins <- apply(s2010, 1, min)
problem <- which.min(mins)
s2010 <- s2010[-problem, ]
s2011 <- s2011[-problem, ]

s2010 <- log(s2010)
s2011 <- log(s2011)
```

4.2 Basis Expansion

Clearly, we expect neighboring data points of a volatility profile to be very close to each other. In such cases as this, one can often improve the quality of one's data by letting neighboring points "inform" each other. By trying various polynomial fits, I found that a fifth-degree is the smallest polynomial fit that leaves essentially no discernible structure in the residuals. It follows the basic shape of the curves remarkably well as shown in Figure 9.

```
# Initialize
par(mfrow=c(3, 2))
stocks <- rownames(s2010)
n <- nrow(s2010)
m <- ncol(s2010)
t <- 2.5 + 5 * (0:77)
t.mat <- cbind(t, t^2, t^3, t^4, t^5)

# Demonstrate fit on first few stocks
for(i in 1:3) {
  v <- unlist(s2010[i, ])
  L <- lm(v ~ t.mat)
  plot(t, v, main=stocks[i], xlab="Minutes into Day",
       ylab="Log of Squared Volatility")
  lines(t, L$fit, col=2)
  plot(t, L$res, main=paste(stocks[i], "Residuals"), xlab="Minutes into Day",
       ylab="Log of Squared Volatlity")
  abline(h=0, col=2, lty=2)
}
```

For the remainder of our analysis I will assume that *the 78 data points are well summarized by a fifth degree polynomial*. I believe that switching to this polynomial basis will introduce very little bias, while decreasing variance appreciably. Each stock's volatility profile will be represented by the six coefficients of this fit.

```
p <- ncol(t.mat) + 1
coef2010 <- matrix(NA, n, p, dimnames=list(stocks, 0:(p-1)))
coef2011 <- matrix(NA, n, p, dimnames=list(stocks, 0:(p-1)))
for (i in 1:n) {
  # 2010 coefficients
  v <- unlist(s2010[i, ])
  L <- lm(v ~ t.mat)
  coef2010[i, ] <- L$coef
  # 2011 coefficients
  v <- unlist(s2011[i, ])
```

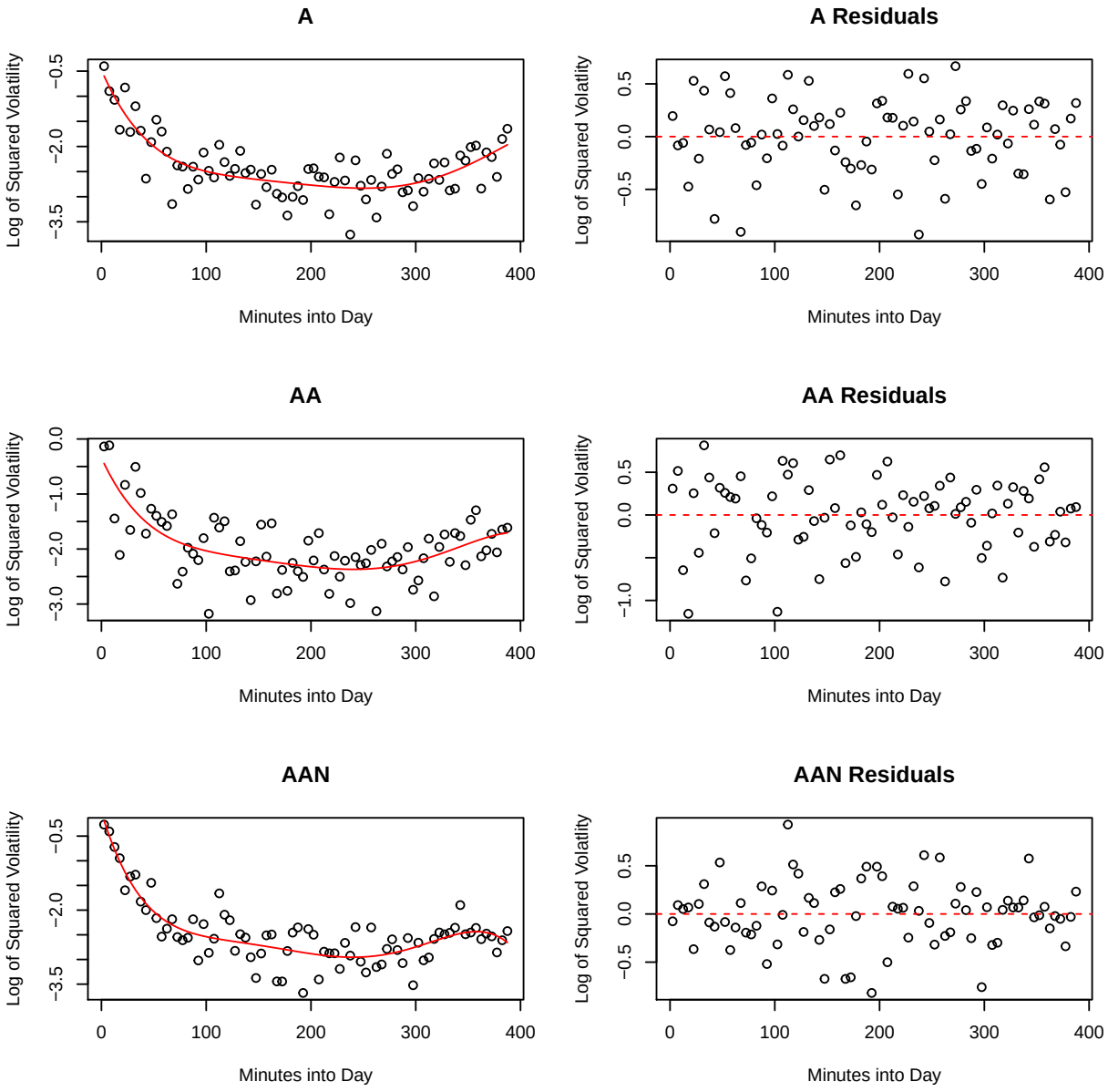

Figure 9: A fifth-degree polynomial fit captures the shape of the data remarkably well and leaves no discernible pattern in the residuals.

```
  L <- lm(v ~ t.mat)
  coef2011[i, ] <- L$coef
}
```

4.3 Controlling for Other Factors

Because our groups are systematically different, we should try to control for those differences the best we can. We have gathered market cap and trading frequency data so we will control for those variables.

4.3.1 Market Capitalization

As shown in Figures 10 and 11, the relationships between market cap and coefficient estimates is extremely weak. Still, I used a quadratic fit across the board to remove any slight effects. This will not do any harm in cases where there is no effect, because it will leave those data points essentially unchanged.

```
cap.mean <- read.csv("capmean.csv", row.names=1)$x[-problem]
```

```
cap.logmean <- log(cap.mean)
cap.mat <- cbind(cap.logmean, cap.logmean^2)
coef.means2010 <- apply(coef2010, 2, mean)
coef.means2011 <- apply(coef2011, 2, mean)
grid <- seq(20, 26, by = 0.1)

# 2010 coefficients
par(mfrow=c(3, 2))
for(i in 1:6) {
  plot(cap.logmean, coef2010[, i], main="2010 Coefficient Estimates",
       xlab="Log of Market Cap", ylab=bquote("Coefficient for" ~ t^.(i-1) ~ "term"))
  L <- lm(coef2010[, i] ~ cap.mat)
  lines(grid, cbind(1, grid, grid^2) %*% L$coef, col = 2)
  coef2010[, i] <- L$res + coef.means2010[i]
}
```

```
# 2011 coefficients
par(mfrow=c(3, 2))
for(i in 1:6) {
  plot(cap.logmean, coef2011[, i], main="2011 Coefficient Estimates",
       xlab="Log of Market Cap", ylab=bquote("Coefficient for" ~ t^.(i-1) ~ "term"))
  L <- lm(coef2011[, i] ~ cap.mat)
  lines(grid, cbind(1, grid, grid^2) %*% L$coef, col = 2)
  coef2011[, i] <- L$res + coef.means2011[i]
}
```

4.3.2 Trading Frequency

Before we control for trading frequency, let us take this opportunity to glance at the group differences in market cap and trading frequency. Figure 12a shows that the groups have about the same average market cap, but the S&P 500 stocks are traded much more frequently on average.

Figure 12b shows an unexpected relationship between market cap and trading frequency. The slight trend shown does not hold up more generally, but it is only a peculiarity of how our data was selected: S&P 500 stocks are among the most frequently traded, while Russell 1000 stocks are among the largest.

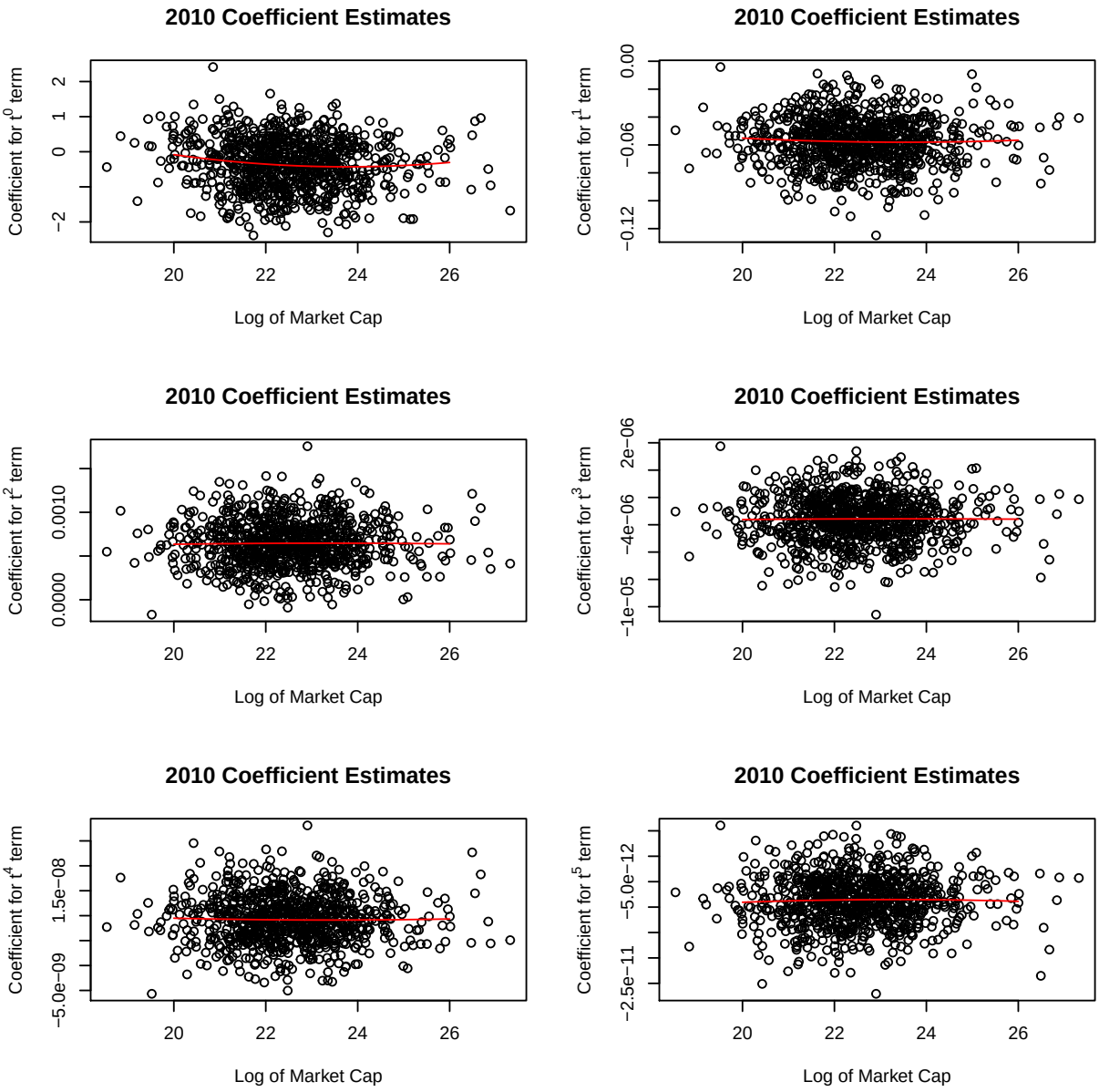

Figure 10: Relationships between market cap and 2010 coefficient estimates.

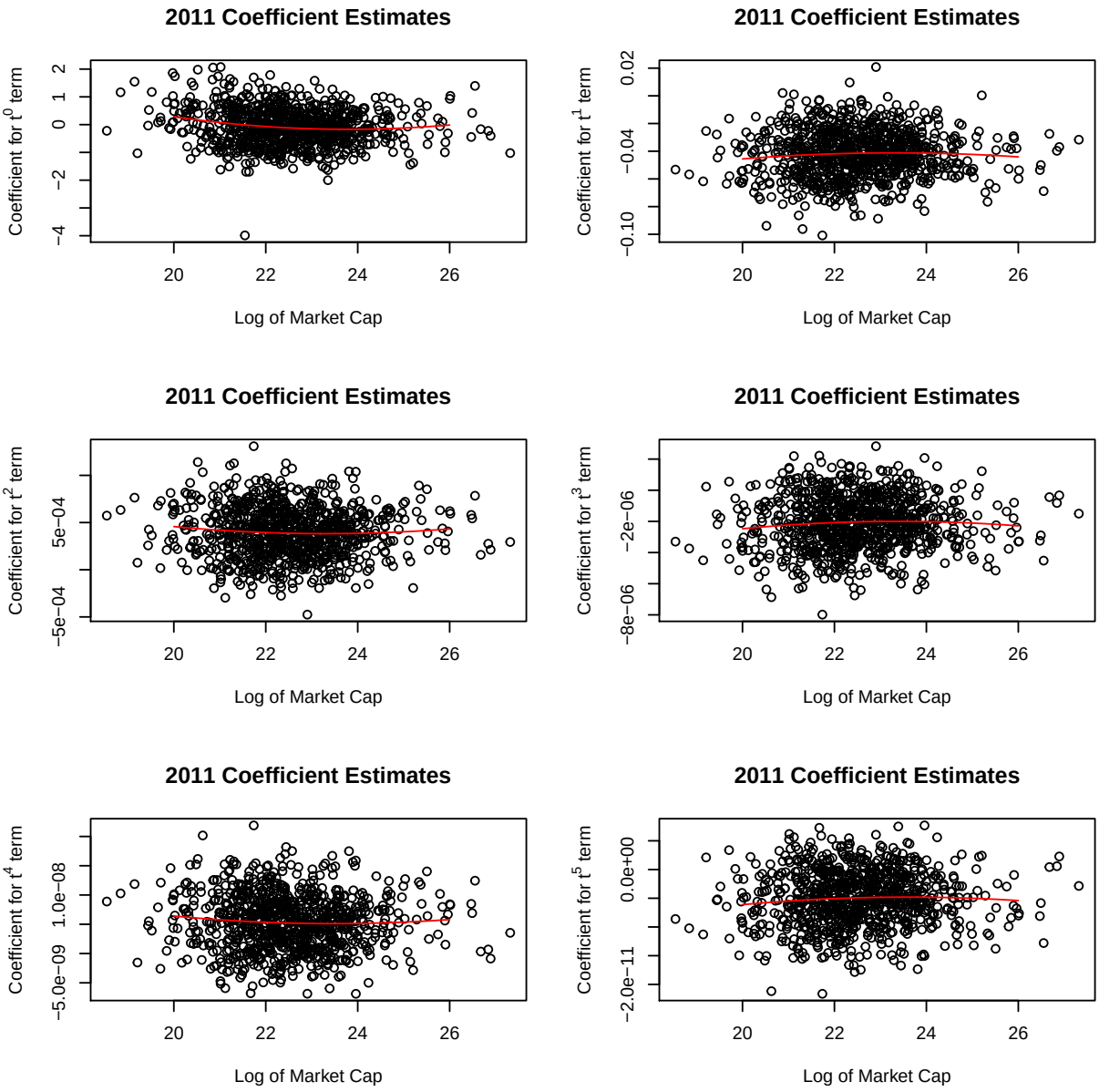

Figure 11: Relationships between market cap and 2011 coefficient estimates.

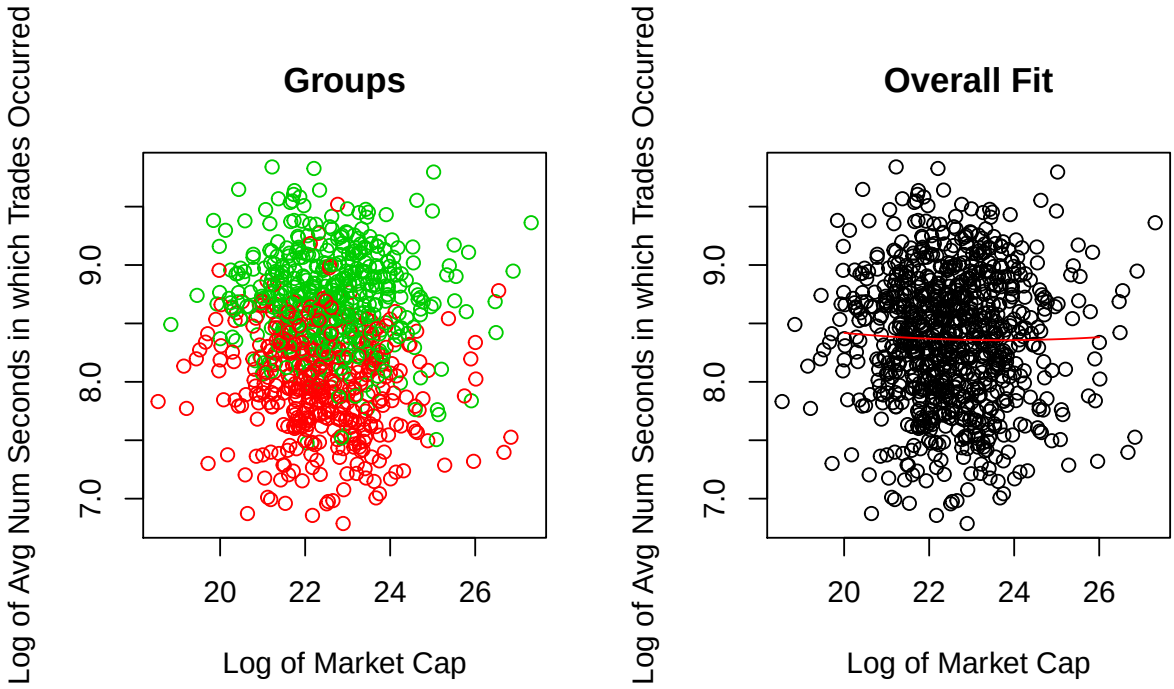

Figure 12: (a) Green points indicate SSCB stocks. Although they have about the same market caps, they are traded more frequently than the control stocks. (b) Relationship between market cap and trading frequency.

```
act.mean <- read.csv("actmean.csv", row.names=1)$x[-problem]
SSCBstocks <- readLines("SP.txt")
```

```
act.logmean <- log(act.mean)

par(mfrow=c(1, 2))
sscb <- stocks %in% SSCBstocks
plot(cap.logmean, act.logmean, col=sscb+2, main="Groups", xlab="Log of Market Cap",
     ylab="Log of Avg Num Seconds in which Trades Occurred")

plot(cap.logmean, act.logmean, main="Overall Fit", xlab="Log of Market Cap",
     ylab="Log of Avg Num Seconds in which Trades Occurred")
L <- lm(act.logmean ~ cap.mat)
lines(grid, cbind(1, grid, grid^2) %*% L$coef, col = 2)
act.logmean <- L$res + mean(act.logmean)
```

As shown in Figures 13 and 14, the trading frequency has a somewhat stronger effect on the coefficient estimates than we saw with market cap. Again, I used a quadratic fit across the board to control.

```
act.mat <- cbind(act.logmean, act.logmean^2)
grid <- seq(7, 9.5, by = 0.1)

# 2010 coefficients
par(mfrow=c(3, 2))
for(i in 1:6) {
  plot(act.logmean, coef2010[, i], main="2010 Coefficient Estimates",
```

```
       xlab="Log of Avg Num Seconds in which Trades Occurred",
       ylab=bquote("Coefficient for" ~ t^.(i-1) ~ "term"))
  L <- lm(coef2010[, i] ~ act.mat)
  lines(grid, cbind(1, grid, grid^2) %*% L$coef, col = 2)
  coef2010[, i] <- L$res + coef.means2010[i]
}
```

```
# 2011 coefficients
par(mfrow=c(3, 2))
for(i in 1:6) {
  plot(act.logmean, coef2011[, i], main="2011 Coefficient Estimates",
       xlab="Log of Avg Num Seconds in which Trades Occurred",
       ylab=bquote("Coefficient for" ~ t^.(i-1) ~ "term"))
  L <- lm(coef2011[, i] ~ act.mat)
  lines(grid, cbind(1, grid, grid^2) %*% L$coef, col = 2)
  coef2011[, i] <- L$res + coef.means2011[i]
}
```

4.4 Comparing Groups

First, we will compare the groups based on the 2010 data. Then, we will compare them based on the 2011 data. Finally, we will compare the changes that the two groups experienced from 2010 to 2011.

4.4.1 Before the SSCB Rules

Figure 15 gives a glimpse at the average volatility profiles of the SSCB and control groups in 2010. They have a similar basic shape, and the sample average of the control group is higher overall.

```
Tmat <- cbind(1, t.mat)

sscb.avg2010 <- apply(coef2010[sscb, ], 2, mean)
control.avg2010 <- apply(coef2010[!sscb, ], 2, mean)
sscb.fit <- Tmat %*% sscb.avg2010
control.fit <- Tmat %*% control.avg2010
plot(t, sscb.fit, type="l", col=3, main="2010 Average Log of Squared Volatility Profiles",
     ylab="Log of Squared Volatility", xlab="Minutes into Day")
lines(t, control.fit, col=2)
```

In order to get a sense of the shape of the 2010 data, let us consider the pairs plot of the coefficient estimates in Figure 16. The estimates of the different coefficients have an extremely large correlation in many cases. This means that the data could probably be summarized well by a lower-dimensional representation. Furthermore, each plot looks reasonably bivariate normal.

```
pairs(coef2010, main="Estimated Coefficients")
```

In fact, a plot of the first two principal components (Figure 17) also shows a basically ellipsoidal data cloud. And the covariance structures of the groups appear to be similar to each other.

```
pc <- princomp(coef2010, cor=T)
plot(pc$scores[, 1], pc$scores[, 2], col=sscb+2, xlab="First Principal Component",
     ylab="Second Principal Component", main="2010 Principal Component Projection")
vars <- pc$sd^2
```

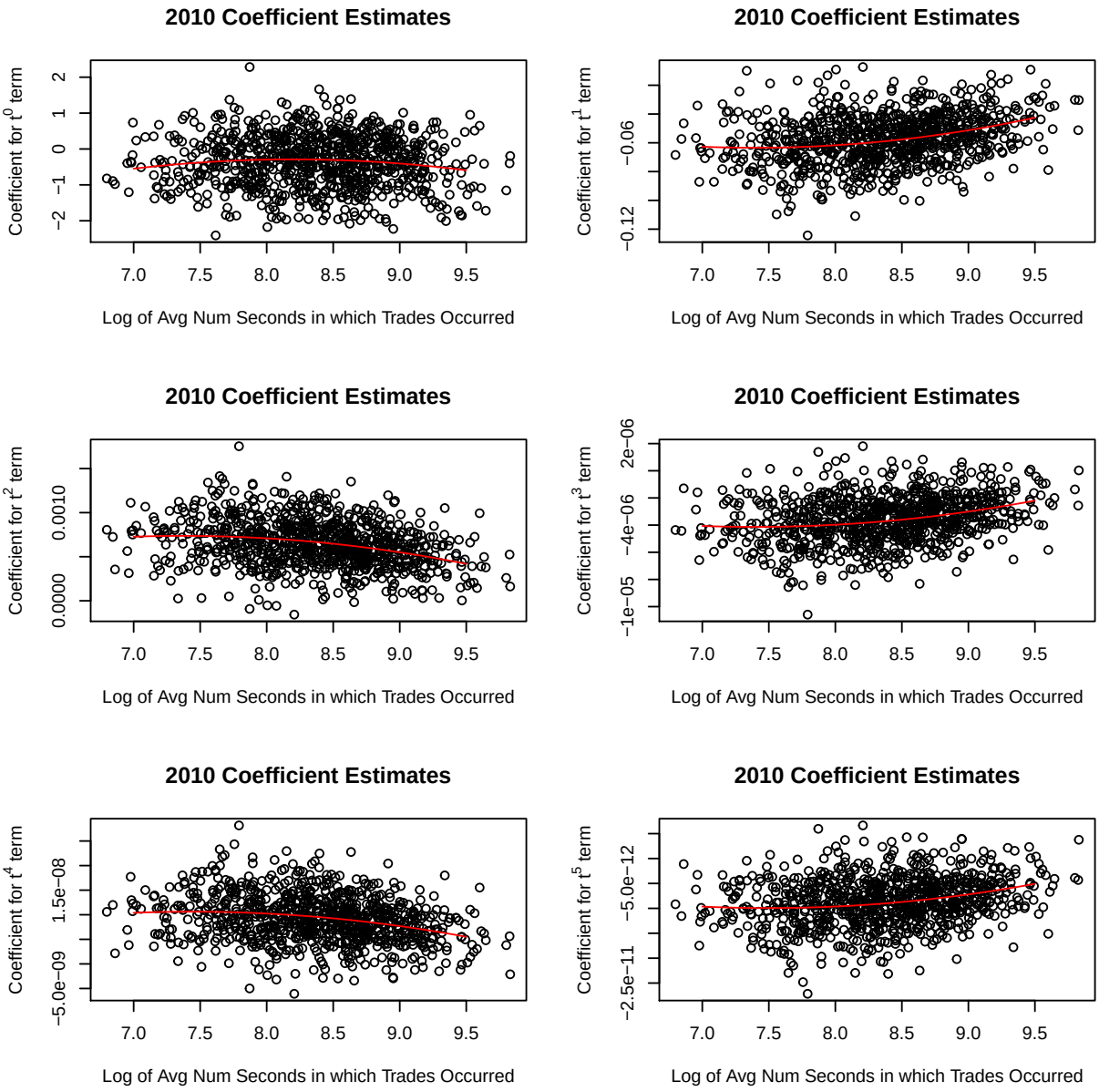

Figure 13: Relationships between trading frequency and 2010 coefficient estimates.

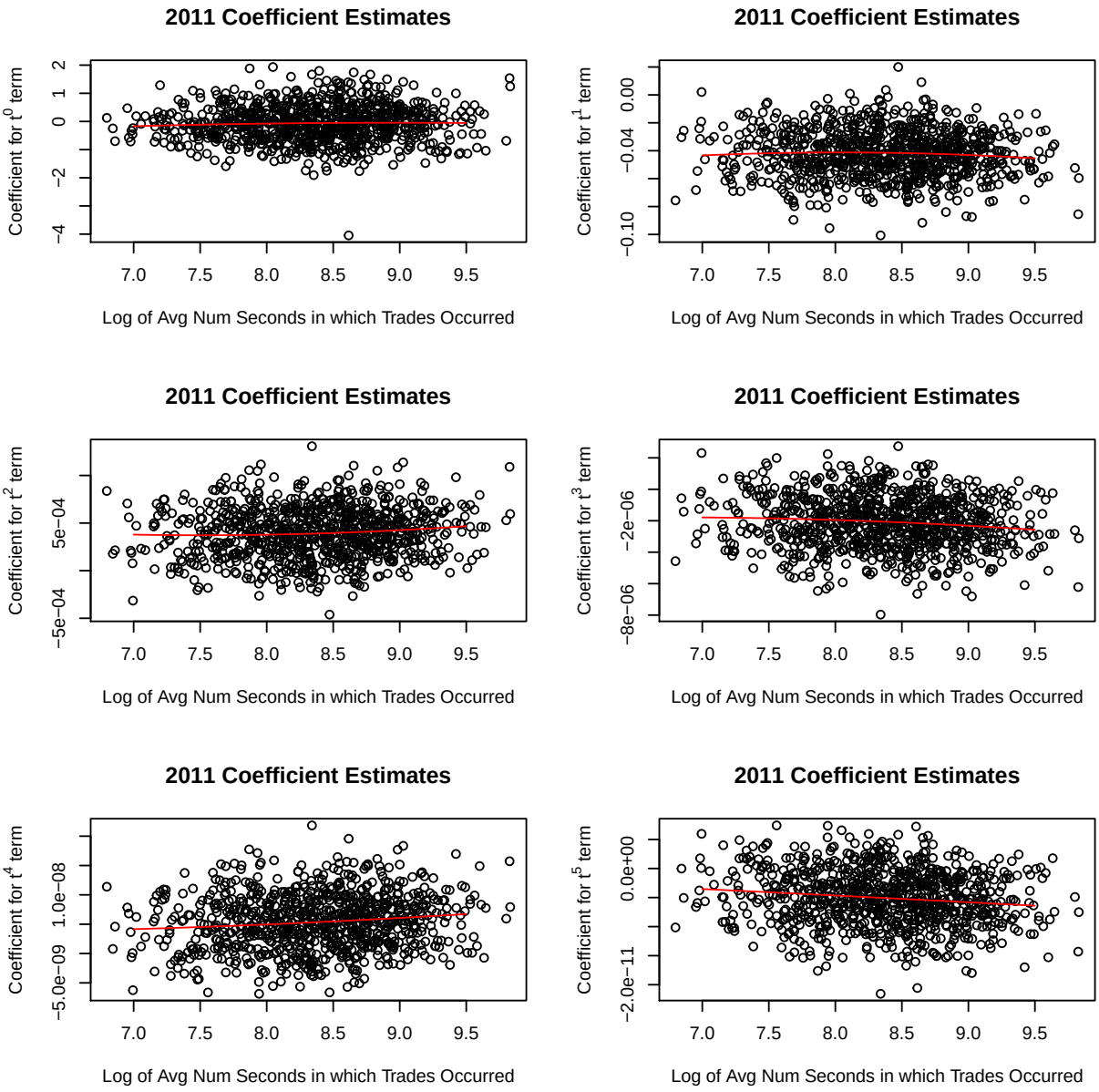

Figure 14: Relationships between trading frequency and 2011 coefficient estimates.

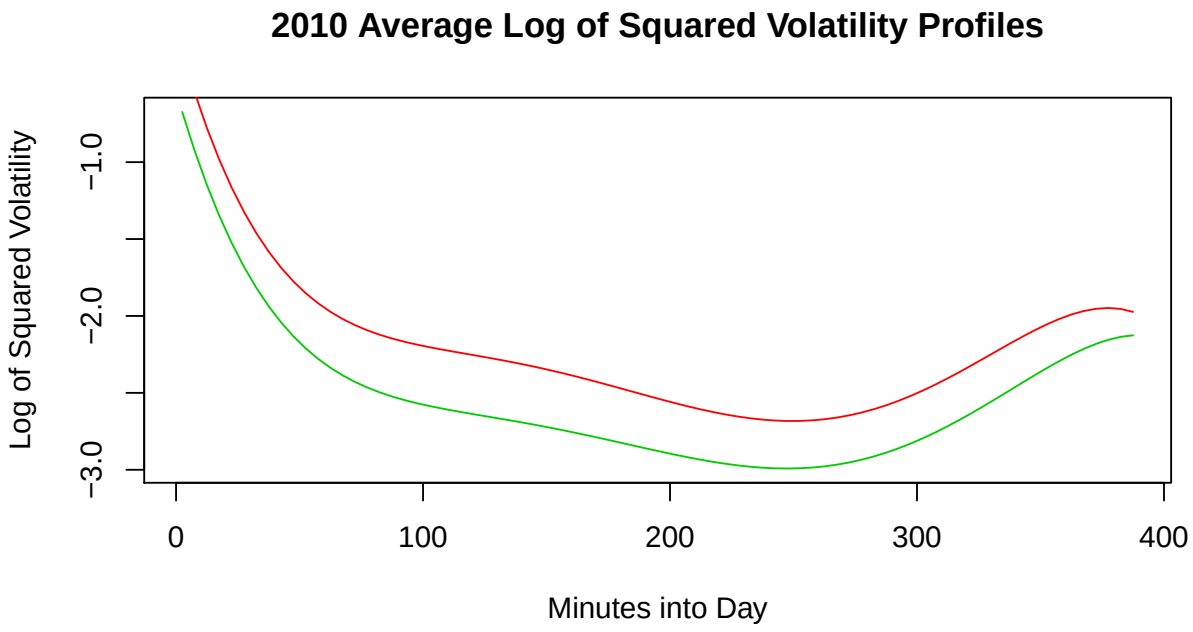

Figure 15: Average log of squared volatility profiles for each group in 2010. The red curve represents the control group average, while the green curve represents the SSCB group average.

```
# See proportion of variability in each component
vars/sum(vars)

##    Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6
## 8.607e-01 1.015e-01 3.660e-02 1.213e-03 3.044e-05 3.143e-07
```

Based on approximate normality and similar covariance structures, a linear discriminant analysis could can help us visualize the separation between the groups. Figure 18 shows boxplots of the two groups scores in the direction of maximum discrimination. Although they are different on average, there is not a strong separation between them.

```
library(MASS)

# Scale to avoid R's error with small numbers.
# (This is fine because LDA is scale-invariant.)
scale2010 <- apply(coef2010, 2, scale)
z <- lda(sscb ~ scale2010)

# Plot the LDA scalings
r <- scale2010 %*% z$scaling
SSCB <- factor(sscb, c(TRUE, FALSE), c("SSCB", "Control"))
boxplot(r ~ SSCB, main="2010 Coefficients Discrimination",
        ylab="Linear Discriminant Score")
```

MANOVA confirms groups' volatility profiles were significantly different on average in 2010.

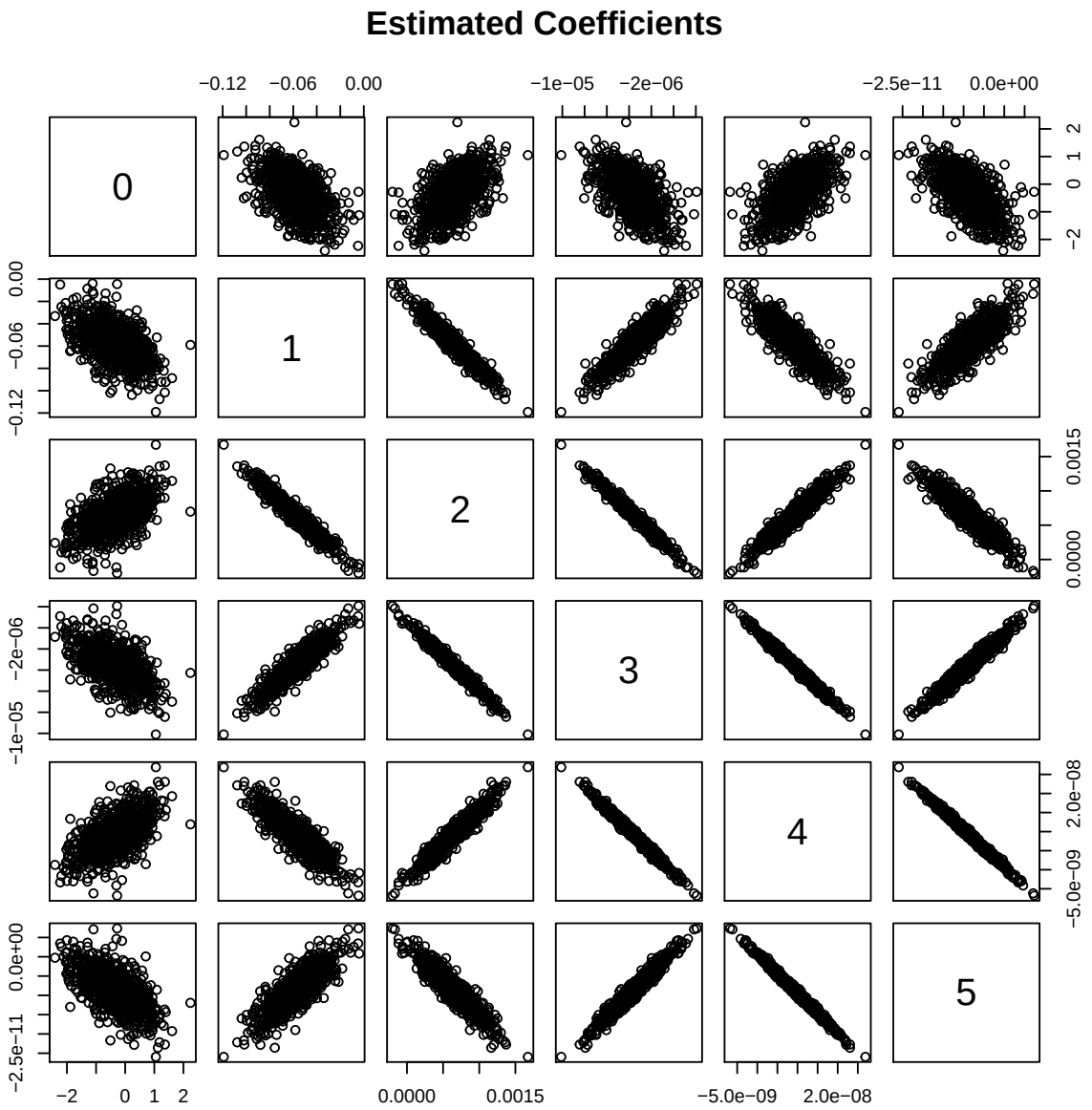

Figure 16: Pairs plot showing the relationships between the coefficient estimates of the 2010 data.

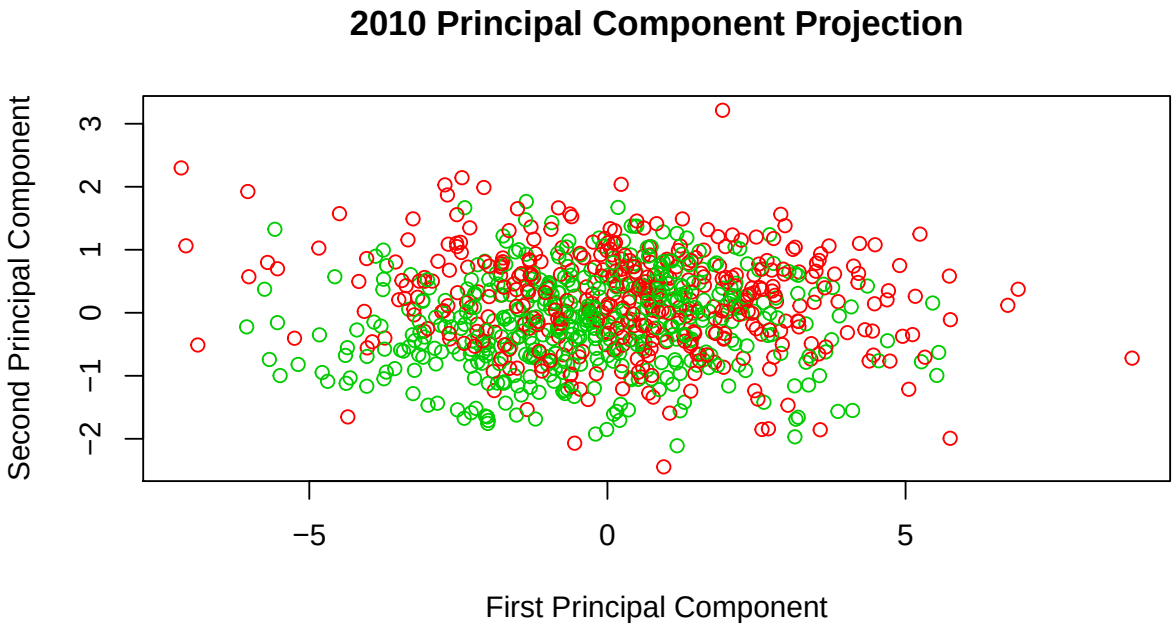

Figure 17: Projection of 2010 coefficient estimates into first two principal component scores. The red points indicate control stocks, while the green points are SSCB stocks.

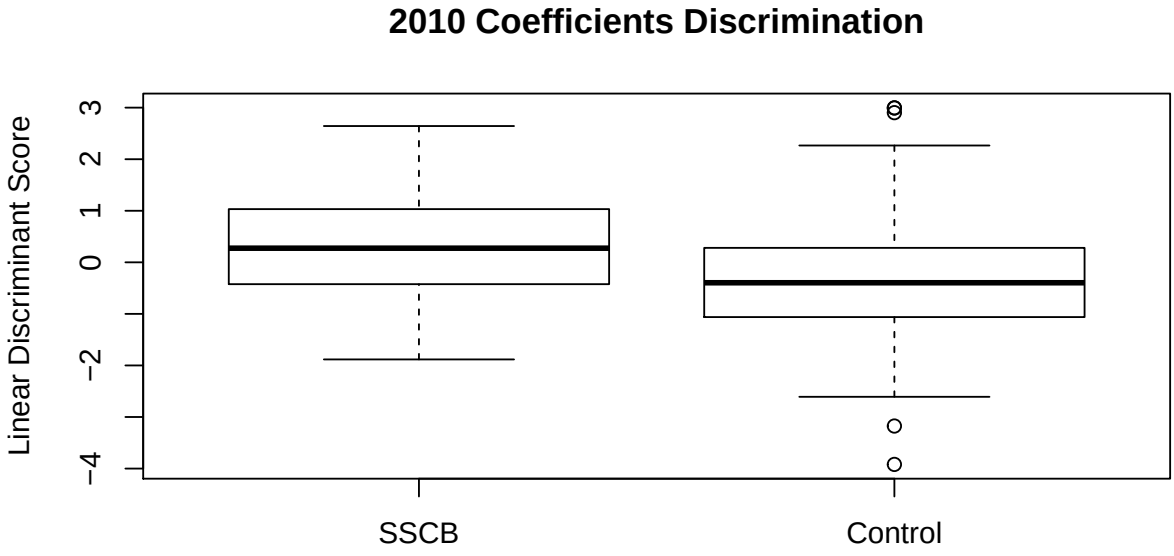

Figure 18: Boxplots of the best linear discriminator score of the stocks' 2010 coefficient estimates.

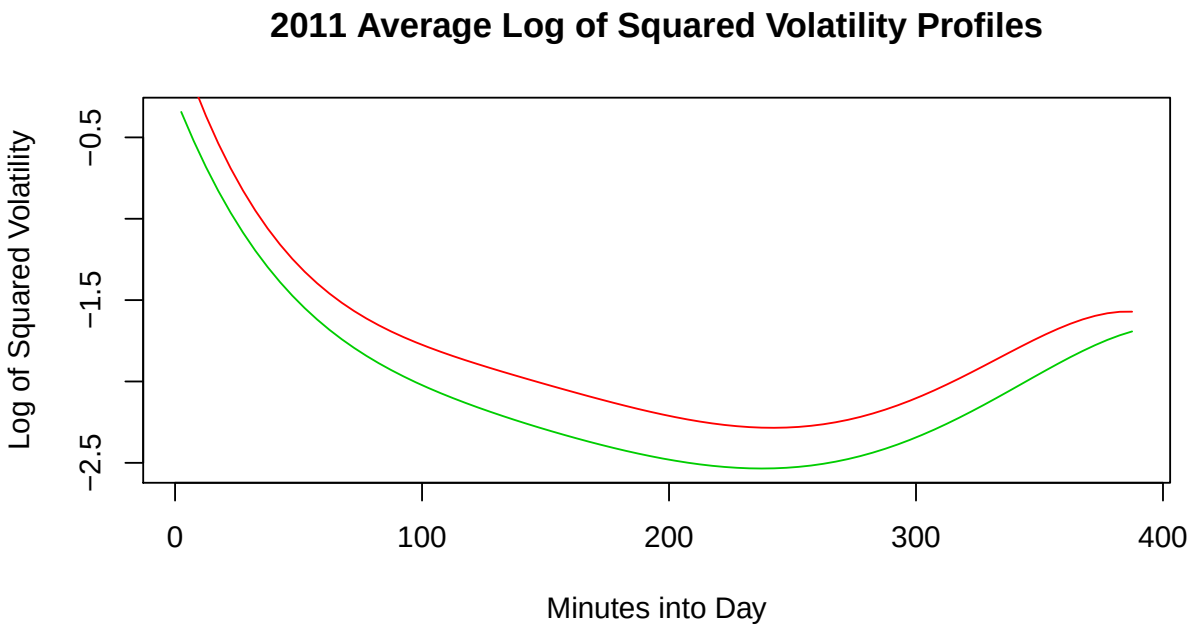

Figure 19: Average log of squared volatility profiles for each group in 2011. The red curve represents the control group average, while the green curve represents the SSCB group average.

```
m <- manova(coef2010 ~ sscb)
summary.manova(m, test="Wilks")

##            Df Wilks approx F num Df den Df Pr(>F)
## sscb        1 0.903     15.4      6    859 <2e-16 ***
## Residuals 864
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.4.2 After the SSCB Rules

Next, we repeat all of the same procedures on the 2011 coefficient estimates. Figure 19 shows that the volatility profiles are similar to their 2010 shapes, although they are somewhat smoother now. The control group is still higher than the SSCB group.

```
sscb.avg2011 <- apply(coef2011[sscb, ], 2, mean)
control.avg2011 <- apply(coef2011[!sscb, ], 2, mean)
sscb.fit <- Tmat %*% sscb.avg2011
control.fit <- Tmat %*% control.avg2011
plot(t, sscb.fit, type="l", col=3, main="2011 Average Log of Squared Volatility Profiles",
     ylab="Log of Squared Volatility", xlab="Minutes into Day")
lines(t, control.fit, col=2)
```

The pairs plot (Figure 20) is basically the same as that of the 2010 data.

```
pairs(coef2011, main="Estimated Coefficients")
```

Again, the principal component scatterplot (Figure 21 shows normal-looking data clouds with similar covariance structures. The plot shows a far outlier, but we have so many data points that its impact is too small to worry about. Furthermore, it may be of interest on its own, but it does not seem relevant to our particular research question.

```
pc <- princomp(coef2011, cor=T)
plot(pc$scores[, 1], pc$scores[, 2], col=sscb+2, xlab="First Principal Component",
     ylab="Second Principal Component", main="2011 Principal Component Projection")
vars <- pc$sd^2
vars/sum(vars)

##    Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6
## 8.623e-01 1.039e-01 3.267e-02 1.162e-03 2.762e-05 3.895e-07
```

The linear discriminant projection (Figure 22) shows a very similar picture to the 2010 data. There is an average difference, but there is not a strong separation.

```
scale2011 <- apply(coef2011, 2, scale)
z <- lda(sscb ~ scale2011)
r <- scale2011 %*% z$scaling
boxplot(r ~ SSCB, main="2011 Coefficients Discrimination",
        ylab="Linear Discriminant Score")
```

In 2011, the groups' volatility profiles are still significantly different.

```
m <- manova(coef2011 ~ sscb)
summary.manova(m, test="Wilks")

##            Df Wilks approx F num Df den Df Pr(>F)
## sscb        1 0.905     15.1      6    859 <2e-16 ***
## Residuals 864
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.4.3 Changes

Next, let us consider the change in the coefficients for each group. This will be the key in our investigation of the effects of the SSCB rules. For each group, the matrix of coefficient changes is defined to be the 2011 matrix of estimated coefficients minus the 2010 matrix. We repeat all of the same techniques once more.

First, Figure 23 shows us that the two groups' changes had very similar shapes, and that the SSCB changes were more sizable.

```
sscb.avgChange <- sscb.avg2011 - sscb.avg2010
control.avgChange <- control.avg2011 - control.avg2010
sscb.fit <- Tmat %*% sscb.avgChange
control.fit <- Tmat %*% control.avgChange
plot(t, sscb.fit, type="l", col=3, ylab="Log of Squared Volatility",
     xlab="Minutes into Day", main="Change in Average Log of Squared Volatility Profiles")
lines(t, control.fit, col=2)
```

The pairs plot (Figure 24) shows very strong correlations between the changes in the coefficient estimates.

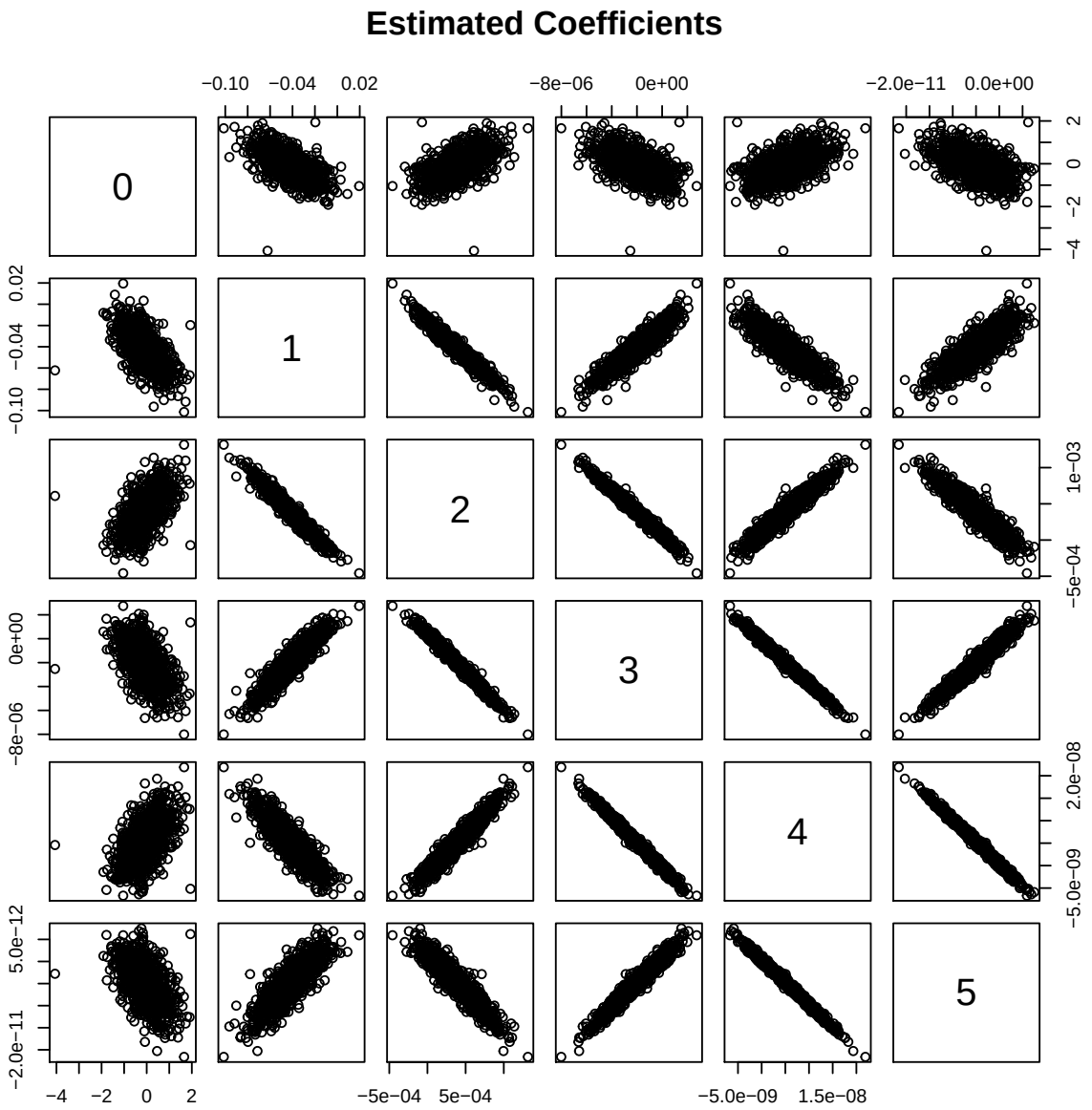

Figure 20: Pairs plot showing the relationships between the coefficient estimates of the 2011 data.

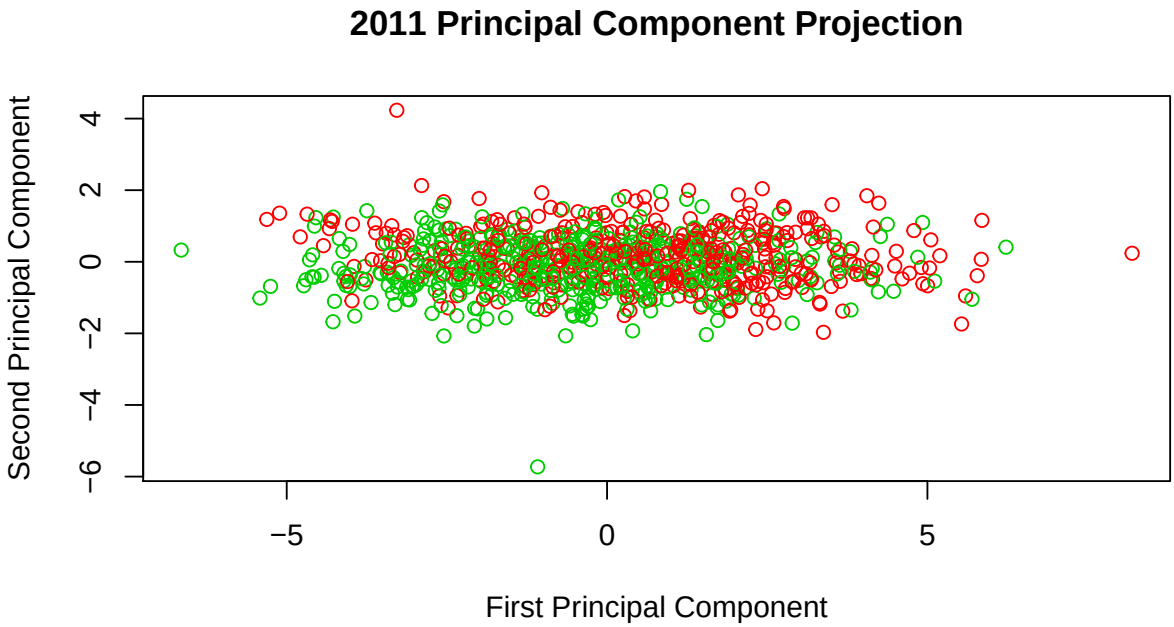

Figure 21: Projection of 2011 coefficient estimates into first two principal component scores. The red points indicate control stocks, while the green points are SSCB stocks.

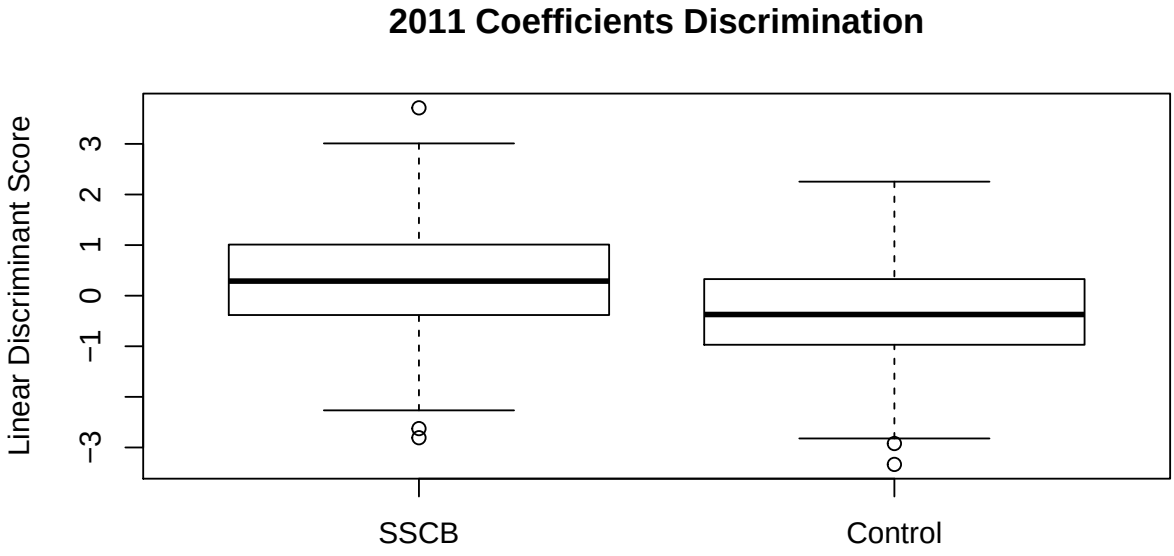

Figure 22: Boxplots of the best linear discriminator score of the stocks' 2011 coefficient estimates.

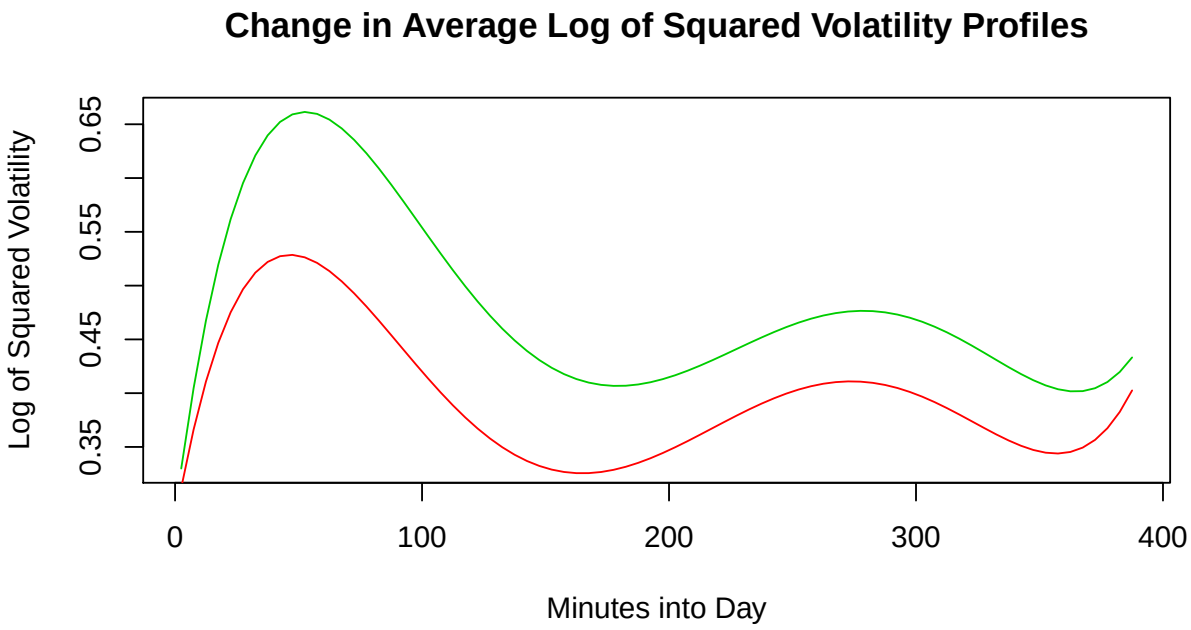

Figure 23: Average change in log of squared volatility profiles for each group in 2010. The red curve represents the control group average, while the green curve represents the SSCB group average.

```
change <- coef2011 - coef2010
pairs(change, main="Estimated Coefficients")
```

The principal component projection (Figure 25) shows two ellipsoidal data clouds with similar covariance structures.

```
pc <- princomp(change, cor=T)
plot(pc$scores[, 1], pc$scores[, 2], col=sscb+2, xlab="First Principal Component",
      ylab="Second Principal Component", main="Change Principal Component Projection")
vars <- pc$sd^2
vars/sum(vars)

##    Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6
## 8.585e-01 1.120e-01 2.850e-02 1.002e-03 2.625e-05 2.027e-07
```

The linear discriminant projection (Figure 26) shows somewhat less separation than we saw on the other plots. But there does seem to be an average difference. The plot shows a far outlier, but we have so many data points that its impact is not very large.

```
scalechange <- apply(change, 2, scale)
z <- lda(sscb ~ scalechange)
r <- scalechange %*% z$scaling
boxplot(r ~ SSCB, main="Change of Coefficients Discrimination",
        ylab="Linear Discriminant Score")
```

MANOVA tells us that the changes in volatility profiles were significantly different between the two groups on average.

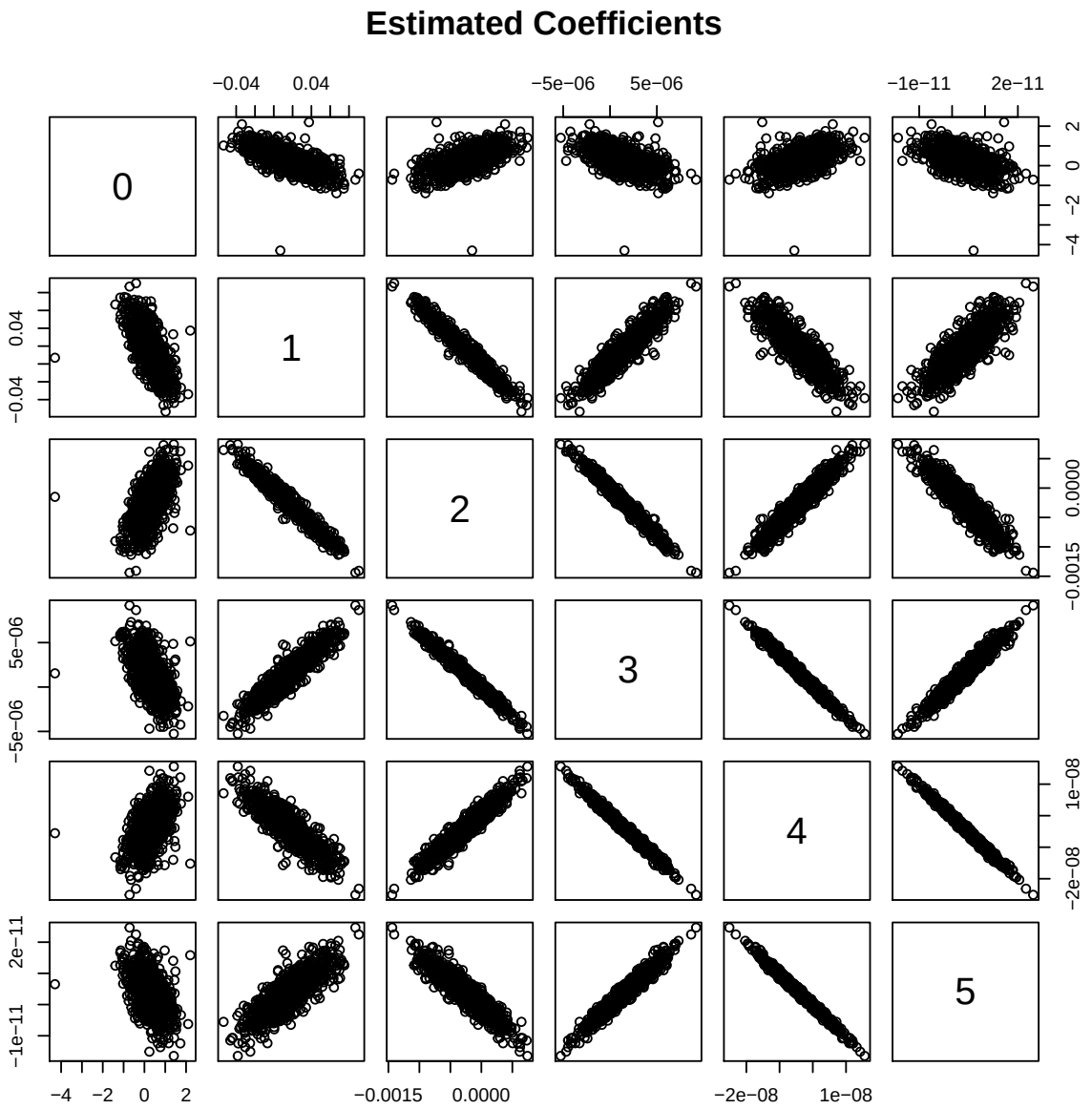

Figure 24: Pairs plot showing the relationships between changes in the coefficient estimates.

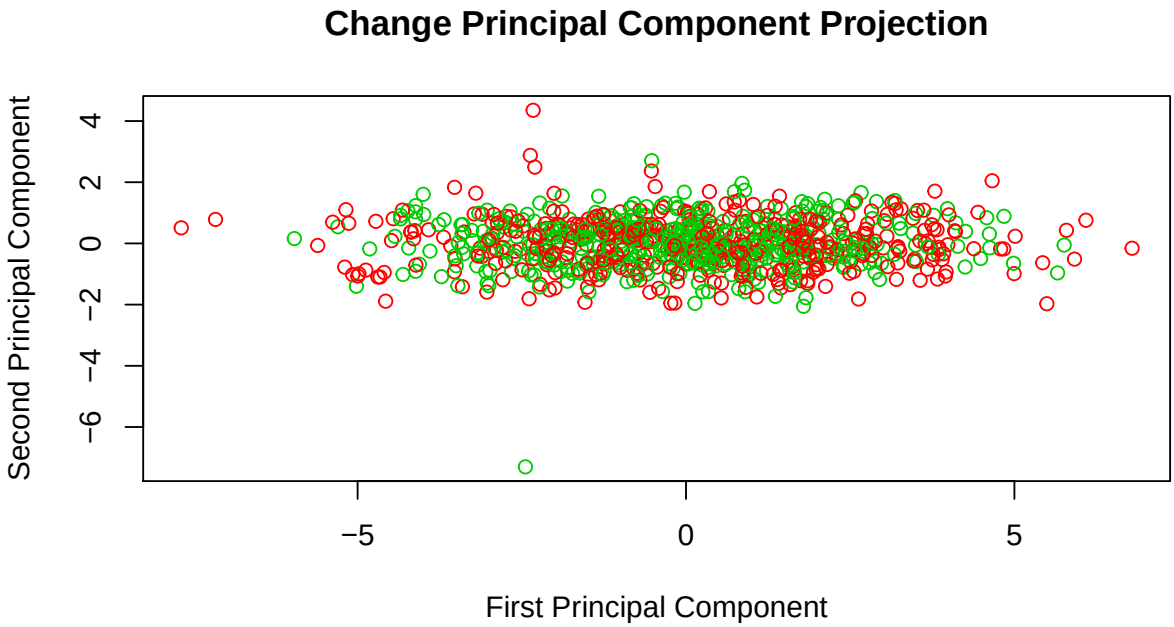

Figure 25: Projection of changes in coefficient estimates into first two principal component scores. The red points indicate control stocks, while the green points are SSCB stocks.

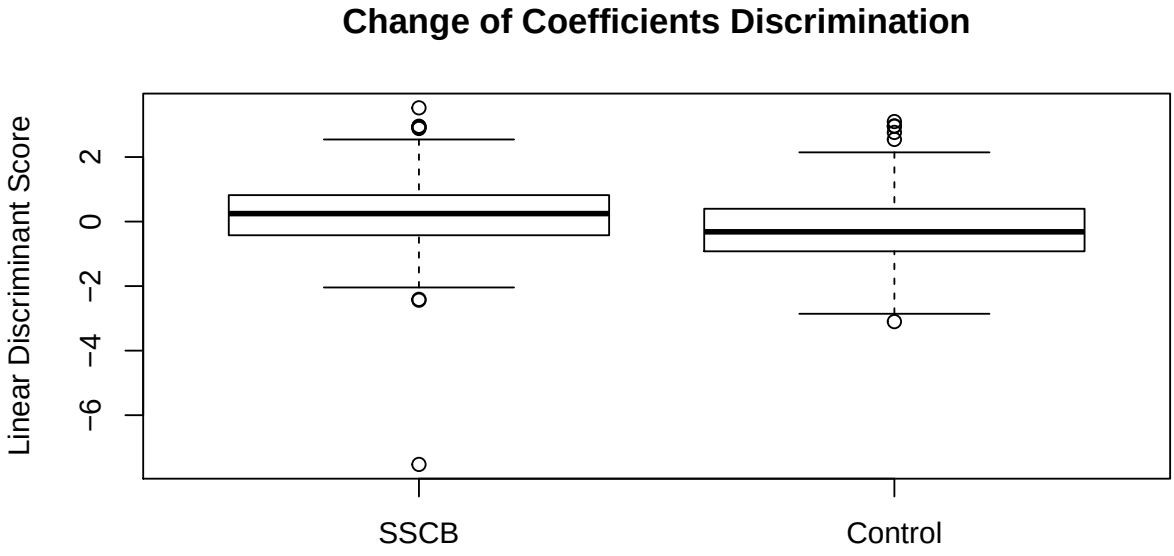

Figure 26: Boxplots of the best linear discriminator score of the stocks' changes coefficient estimates.

```
m <- manova(change ~ sscb)
summary.manova(m, test="Wilks")

##            Df Wilks approx F num Df den Df  Pr(>F)
## sscb        1 0.954     6.88      6    859 3.8e-07 ***
## Residuals 864
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.5 Conclusions

Unfortunately, controlling for market cap and trading frequency were not enough to make the two groups' 2010 volatility profiles the same. Therefore, we have no hope of saying that they were comparable before the SSCB rules. Then, if one group changed differently from the other, it could be due to other systematic differences rather than the SSCB rules. With that in mind, let us recall the main results.

Although the changes in the two groups' volatility profiles were *significantly* different on average, they were not *substantially* different. They overlap to a large extent (Figure 26) and they were in the same basic direction (Figure 23). In other words, one would have very little power in classifying a stock into one of the groups based on its volatility profile change.

The analysis was unable to determine whether the SSCB rules had a significant impact on volatility profiles. However, I do think that it establishes that the rules most likely *did not have a dramatic effect on the overall volatility profiles of the affected stocks as of late 2011*. Of course, they may have affected the stocks in some other way that a different analysis could uncover.

While our SSCB questions are not entirely answered, looking at our estimated volatility profiles brings other interesting questions to light. Why were are the two groups' volatility profiles different, even after taking market cap and trading frequency into account? And more interestingly, why did volatility profiles change in the way that they did (see Figure 23) between our two periods of study?