Lab 3. Modeling the Market

Lab in Brief:
The Purpose of this assignment is to introduce you to finding solutions using panel data – i.e., data that measures things that change over time (as opposed to our last lab, where we were measuring things over space). In this case, you'll be learning how to process and use stock market data to identify a stock investment strategy, and use a wide set of data to defend your choices. Further, you'll be adding to your toolbox of data visualization (which now includes infographics and maps) by learning how to parse data to create a number of new visualizations. Finally, this lab is the first lab in which you will begin to examine the concept of data mining.

Data and Materials:
We'll be using a combination of online resources and R (with Rstudio) for this lab. Data will be downloaded in real-time from public sources (yahoo and google).

Objectives:
In this lab, you will (a) answer a series of questions regarding the stock market using basic strategies similar to those employed by High Frequency Traders (HFTs), and (b) select stock investment strategies. Finally, you will produce a brief written description defending your choices, emphasizing the data visualizations produced in this lab.

Deliverables:
By Friday, October 9th at 11:59PM EST, you will submit the lab deliverables described on the final page of the lab to blackboard.

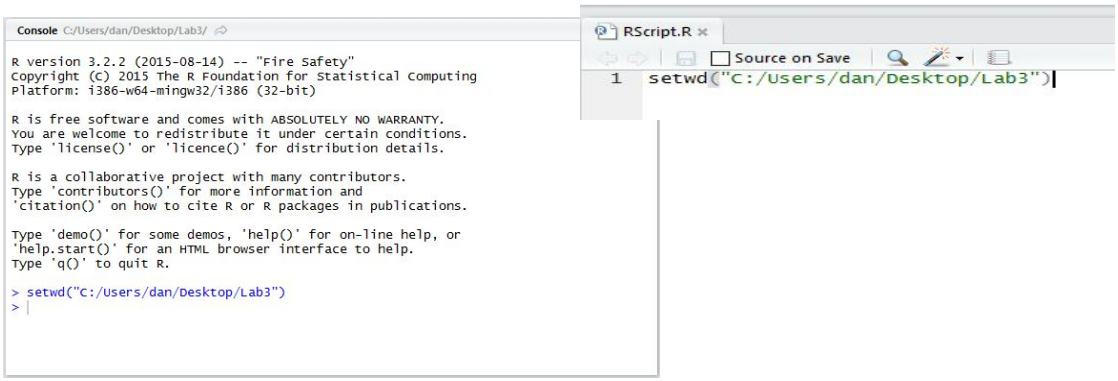
Grading:
This lab will be graded according to the same rubric used for Lab 1, which can be found on blackboard. The five key criteria are answering questions correctly, creativity, clarity, your ability to define the problem, and to use pertinent information to defend your site selections. **It is important that you spend time working on your short defense of the stocks you choose so that it is visually appealing – you may even consider mixing strategies from your infographic creation with the brief to prove your points.** What interesting statistics can you pull from your data to make a better product for “System 1”?

Part 1: Getting Setup

1. First, create a new folder on your H drive for this lab, and make sure to note where it is – just like last time in R, you're going to have to type that directory in periodically.

Warning*: Don't go investing all your money after doing this lab! The point of this lab is not to make you all millionaires, but rather to give a real-world example of how "panel data" (data that measures things over time) is used. While all of the data you're using will be real, without much more extensive training it is unlikely you'll succeed using these strategies on the market (after all, Dan isn't making millions on wall street!). Luckily for you, if you love this lab William and Mary has a Business Analytics program, which you'll be hearing about* next week.

2. Open up Rstudio and create a new document (File → New File → R Script), and save it on your H-Drive in your new folder (File → Save As).
3. Set your working directory (Session → Set Working Directory → Choose Directory). This is the folder R will look for all of your data in. When you do this, you'll also want to copy what pops up in your console below into your Rscript (that way, you won't have to do this again next lab). Note – your working directory will be different from the one shown here (as it will be on your H drive!).



4. Let's download some stock data. Lucky for us, lots of very rich people on wall street need to do this all the time, so ready-built tools already exist. Just like in lab 1 we're going to install a "Library" (also sometimes called a "Package") - last time, the library helped us make simple maps of our hometowns; this time the library will download stock information for us. The name of the package we will install is called "quantmod." In your R Script, type in the following:

```
install.packages("quantmod", dependencies = TRUE)

library(quantmod)
```

You only have to install a package one time in order for it to be accessible to your `R` framework. You need to load that library into your `R` session each time you plan to use it (this is what "library(quantmod)" does). Once you have installed the package and loaded the library, you can also inspect the complete contents by typing.

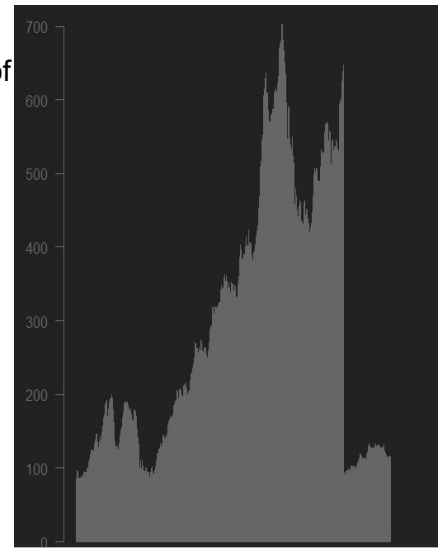
```
?quantmod
```

Which will present you with general information about the package, its author(s), its website and at the bottom of the page a link to the index that includes all of the functions contained within that particular library.

5. After installing the package and loading the library, download the stock data - this will create a new object called "AAPL":

   ```
   getSymbols("AAPL")
   ```

   If you recall from lab1 you can also type `str(AAPL)` and R will return basic information about the object, informing you it is an 'xts' object, which essentially means it is a time series of data (in fact, that's what the "ts" stands for!).



6. Now, take a look at that data to make sure it worked – you should see stock market open and closing dates from the past (head) and up to yesterday (tail). "Head" and "Tail" literally refer to the start (the first 6 time steps within the time series) and end (the last six time steps in the time series) of the dataset:

   ```
   head(AAPL)
   tail(AAPL)
   ```

7. Now, we're going to use the `as.vector()` command to translate a part of the data into something called a vector. This will let us isolate and look at one part of the data, which is important for us to do for certain kinds of visualizations. After creating your vector you can type in "head(AAPL_Close)" to see what you did, and you should see a series of numbers:

   ```
   AAPL_Close <- as.vector(AAPL[,"AAPL.Close"])
   head(AAPL_Close)
   ```

   Viola - now you have a new vector that only contains information on the closing price of Apple.

8. Alright, time for our first visualization! Let's start very simple, type in the code below (using the barplot command) and you should see a graphic that looks similar to the first image on the right:

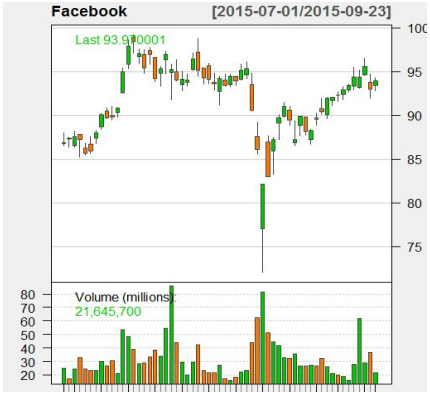
   ```
   barplot(AAPL_Close)
   ```

9. That isn't particularly pretty, so we're going to use something from the quantmod library to improve its appearance. Try this out:

   ```
   chartSeries(AAPL)
   ```

10. Quite a bit nicer, right? We can even do things like zoom in on just the last few months:

    ```
    chartSeries(AAPL, subset='last 3 months')
    ```



You can also review the help page for individual functions within the package by using the `?` operator followed by the name of the command.

```
?chartSeries
```

This will present you with the `R` help page for the function `chartSeries()`, and has some hints as to how you might further modify your visualization.

If you scroll to the bottom, the help pages are nearly always concluded with several **Examples** that provide the user with a blueprint for specific application.

11. This lab will introduce you to methods used to analyze a ton of data at once. For now, we'll grab 5 stocks, but you'll be able to get as many as you want later on.

```
getSymbols(c("TWTR", "IBM", "FB", "AMZN", "NFLX"))
```

12. There are a *ton* of indicators that stock traders use to choose if they will buy a stock or not. Let's make a new chart with one of our new stocks, and add a line representing the Moving Average. We will use the `addSMA()` command (add Simple Moving Average) to accomplish applying a line to our Facebook chart of the stock's performance during the last 3 months.

```
chartSeries(FB, subset='last 3 months')
addSMA()
```

13. We can even improve on how this looks by changing themes:

```
chartSeries(FB, theme="white", subset='last 3 months')
```

14. And, changing titles:

```
chartSeries(FB, theme="white", subset='last 3 months', name="Facebook")
```

15. Now you should have a chart that looks much better, and very similar to the one on the right.

Lab Question: Based on this chart, what date would have been the best to purchase Facebook stock?

Part 2: Predicting the Future (i.e., how folk are getting rich on Wall Street)

Now, we're ready to dive into how some of the algorithms on Wall Street currently perform. Remember that big warning about how you shouldn't use this for real investing? Some of you might find real patterns in the data, but keep in mind the folk doing this professionally code in R (or even more powerful languages) all day, every day – the algorithms we're showing here are just a very simple example of what they can do. But, the basic tools and concepts are the same. Remember, Dan isn't a millionaire, so it's unlikely you'll be from this one lab (but if you are, remember Dan!).

16. In the first step, we're going to do something a little complicated. What we want to do is predict what the value of a stock will be *tomorrow,* based on the value of another stock *today*. In this very simple example, we're going to predict Facebook's price tomorrow based on Twitter's closing price today. First, we're going to create our data – the part before the "~" is the data we're trying to predict (tomorrow's `Next()` closing price `Cl()` of Facebook `(FB)`. The part after the `~` is what we're using to predict the closing price `Cl()` of Twitter `(TWTR)`.

    ```
    dataIn <- specifyModel(Next(Cl(FB)) ~ Cl(TWTR))

    myData <- modelData(dataIn)

    View(myData)
    ```

    Everything within the parenthesis following the function `specifyModel()` is a `formula`. This `formula` follows the format of first defining the Predicted variable, a tilde `~`, which is then followed by one or more Predictor variables. Once you have specified your model that will be used to predict specific future values of Facebook, you can then use the `modelData()` function, which extracts the data for the model you want to specify (see the next step).

17. When you type in `View(myData)`, you're looking at three different columns: Next.Cl.FB is the predicted closing price of Facebook for the day *after* the date indicated in the first column. Cl.TWTR is the closing price for that day. So, we can answer a simple question: does the closing price of twitter today predict the closing price of Facebook tomorrow?

	Next.Cl.FB	Cl.TWTR
2013-11-07	47.53	44.90
2013-11-08	46.20	41.65
2013-11-11	46.61	42.90
2013-11-12	48.71	41.90

18. Let's start our simple analysis by using a scatterplot:

    ```
    plot(as.matrix(myData))
    ```

    There is a fairly clear pattern here - as TWTR's closing price today increases, it is likely that Facebook's closing price tomorrow will be lower (and, vice versa).

19. We can also create this type of visualization across all of our stocks, to see if any have strong correlations with Facebook's closing price tomorrow:

```
moreDataIn <- specifyModel(Next(Cl(FB)) ~ Cl(TWTR) + Cl(IBM) + Cl(AMZN)+
Cl(NFLX))

myBiggerData <- modelData(moreDataIn)

pairs(as.matrix(myBiggerData))
```

    At first glance, this chart is scary to look at, but all you really need is the top row. The y-axis in every one of those charts is the closing price "tomorrow" of Facebook, and each X-axis represents the other 4 stocks we're looking at. You can quickly visually see that all four stocks are related to Facebook, but in different ways (i.e., when Amazon closes high, Facebook also tends to close at a high value the next day).

    Lab Question: What is the relationship between Facebook and IBM – for example, if IBM closes at a high price today, do we expect a high or low price for Facebook tomorrow?

20. Alright, now let's start digging in a little deeper. Pick one stock you think is the closest correlated with Facebook, and we're going to use that one relationship to build a formal model. All this model is going to do is mathematically represent the relationship you see in the scatterplots. First, build a new dataset with the ONE stock you think will predict. In this example, I'm using Amazon to predict Facebook, but you can use any stock (you can even use something like "getSymbols("AAPL")" to get a stock I didn't include here):

```
myPredictiveModel <- specifyModel(Next(Cl(FB)) ~ Cl(AMZN))
```

21. Now, we're actually going to build the model itself – this line of code, while complicated looking, is pretty easy. The first part, "myPredictiveModel", is just telling the code which model to use (which you already made). The second part, "method=lm", is just telling it to use a particular statistical approach for modeling (here, we're using the absolute simplest approach). Finally, training.per stands for "Training Period". In this example, I'm building a prediction based on the historic period from July 1, 2015 to September 25, 2015. This is important, as in this simple modeling approach you want to choose the time period that is the most representative of what you think will happen in the future – i.e., are the last three months the most relevant in predicting tomorrow? Or the last three years? Last week?:

```
predicting_FB <- buildModel(myPredictiveModel, method="lm",
training.per=c('2015-07-01','2015-09-25'))
```

22. Let's take a look and see how well the model really works:

```
summary(predicting_FB)
```

    While there are many ways to interpret this, for this course we're going to stick with a basic and very naive interpretation of the R-squared (because I don't expect ya'll to have had statistics yet!). Values approaching one indicate a higher degree of predictive power - i.e., if you have a R squared of 1, you would be able to predict perfectly, and a R-squared of 0.2 you'd be able to predict around 20% of the stock changes. **As you will learn in statistics, there are a huge number of caveats and assumptions around these interpretations, but we won't be going into them in this class.**

Lab Question: In this model, how well are you able to predict Facebook tomorrow based on Amazon today? Can you achieve a higher R squared by changing the training period, and if so what is the highest value you were able to find?

23. Let's think a little bit about our model itself - while it's nice to know if Facebook will generally move up or down the next day based on Amazon, it would be even better if we could calculate the percent change - or "delta" – between today and the end of the next day. That would let us calculate how much money we could make by investing in Facebook today and selling tomorrow, just based on Amazon today! As you'll see, though, this can be very challenging. Type this in, and think through the differences between this and our earlier model:

```
myDeltaModel <- specifyModel(Delt(Next(Cl(FB))) ~ Cl(AMZN))

predictingFBChange <- buildModel(myDeltaModel, method="lm",
training.per=c('2015-07-01','2015-09-25'))

summary(predictingFBChange)
```

You'll see one new command here - “Delt”. This gives us the “Delta”, or change in Facebook’s closing cost between today and the next closing day for Facebook. In this model, we're predicting that change based on the closing cost of Amazon stock today.

Lab Question: Which model is more effective in prediction, the model you designed that predicts the *closing cost* of Facebook, or the one that you just designed that predicts the *change of value* in Facebook?

24. Try predicting the change in Facebook tomorrow based on the percent change of Amazon today to see if that improves the model:

```
myNewDeltaModel <- specifyModel(Delt(Next(Cl(FB))) ~ Delt(Cl(AMZN)))

predictingFBChange_2 <- buildModel(myNewDeltaModel, method="lm",
training.per=c('2015-07-01','2015-09-25'))

summary(predictingFBChange_2)
```

Lab Question: Was this model any better? How did you arrive at that answer?

25. To this point, we've only been using information from one stock to do our predictions. Now, we're going to explore if including more stocks helps us make a better prediction. For example, does predicting change in Facebook based on change in multiple other stocks help?

```
multiStockModel <- specifyModel(Delt(Next(Cl(FB))) ~ Delt(Cl(AMZN))
+ Delt(Cl(TWTR)) + Delt(Cl(IBM)) + Delt(Cl(NFLX)))

FB_multiStock <- buildModel(multiStockModel, method="lm",
training.per=c('2015-07-01','2015-09-25'))

summary(FB_multiStock)
```

Lab Question: On your own, *try other stocks and training periods* to see how strong of a prediction you can achieve. Report your best R-squared and the three lines of code (`specifyModel()`, `buildModel()`, and `summary()`) you used to reach that R-squared. How strong is your model? This isn't easy, or we'd all be rich!

Part 3: Some real-world Data Mining

To date, you've had to manually guess what time spans and stocks might be correlated with one another. The big advantage wallstreet brokers have is that they have computer algorithms that can explore all of these (and many other!) connections automatically – without a human having to spend time guessing. We're going to create a very simple example of data mining by automatically checking the fit of a wide set of models, and visualize those fits relative to the number of trades each stock has had. This is the first real example of "data mining" so far in this course.

26. Key to data mining is a new programming idea, something called a "while" loop. Basically, a "while" loop tells a computer to keep doing something until some number comes along that tells it to stop. A very simple example is as follows (go ahead and type this in - we're going to be building on it in the next few steps):

```
myVar = 1

while (myVar <= 10)
{
print(myVar)
myVar = myVar + 1
}
```

As you can see, that made R print out the numbers one to ten.

Lab Question: Edit the above code to make it print out 2 to 12 instead of 1 to 10, and provide your new code.

27. We're going to do something very similar, but we're going to loop through all of our stocks to make a lot of "guesses", just like you manually did in Part 2. First, let's define all the stocks we want to "guess" with. Note this list of stocks can be as small or big as you want, as the computer will automatically search for relationships between all of them and your stock of interest.

```
testStocks = c("TWTR", "IBM", "FB", "AMZN", "NFLX")
```

28. Now, let's tell the computer the stock we're interested in predicting:

```
predStock = "DAL" #Delta Air Lines
```

29. Just like before, let's download all that data:

```
getSymbols(testStocks)
getSymbols(predStock)
```

30. Now, we're going to count the number of stocks you put into "testStocks". This is important later on, as we need to tell the computer how many stocks to test.

```
numStocks = length(testStocks)
```

Double check you got it right by typing in "numStocks" (without quotes!).

31. Now, we're going to pull it all together in a while loop, looking at every stock. First, we tell the while loop to keep going until it runs out of stocks:

```
i = 1
while(i <= numStocks)
{
  print(i)
  i = i + 1
}
```

That should only count up to the number of stocks you had.

32. Now, let's replace print(i) with the visualization we learned above - chartSeries. This is a little bit tricky. First, we have to get the data from every stock. To do this, we use the command:

```
stockOfInterest = eval(parse(text=testStocks[1]))
```

which will give us the data for the first named stock in testStocks (in my example, Twitter). To see what happened here, just type in “stockOfInterest”.

33. Now, we want to make a chart of this data:

```
chartSeries(stockOfInterest, subset='last 3 months', name=testStocks[1])
```

which will give us the chart for the first named stock in testStocks (it should show up on the right). Basically, all you did in the above two lines is the equivalent to:

```
chartSeries(TWTR, subset='last 3 months', name="TWTR")
```

The only difference is, the two line version above uses the testStocks[1], which we will modify to let the computer go through each of your stocks in the while loop.

34. Put this while loop in. If you're confused about what's going on, it's important to go back through steps 31-33 above, or raise your hand. Loops like this are one of the most important tools in data mining, and used by big data folk all around the world:

```
i = 1
while(i <= numStocks)
{
  stockOfInterest = eval(parse(text=testStocks[i]))

  chartSeries(stockOfInterest, subset='last 3 months',
  name=testStocks[i])

  i = i + 1
}
```

When you run that, it should automatically produce (in my example) 5 charts - one for every stock. In your case it may be more or less, depending on how many stocks you put in step 27 above.

35. Now, lets add a little bit of modeling - we're going to see which stock predicts the stock we're interested in predicting the best (in this example, Delta Airlines), but automatically this

time. To do that, we're going to add a few lines of code to our while loop:

```
modelOfInterest <- paste("specifyModel(Delt(Next(Cl(",predStock,"))) ~ Cl(",testStocks[1],"))")

modelOfInterest
```

You'll see when you type in modelOfInterest, an actual line of R code comes out. If you change the "1" in testStocks to a "2", you'll see a slightly different line come out:

```
modelOfInterest <- paste("specifyModel(Delt(Next(Cl(",predStock,"))) ~ Cl(",testStocks[2],"))")

modelOfInterest
```

Lab Question: Copy what appears when you type “modelOfInterest” the second time, and explain in 1-2 sentences why it's different than the first time you typed it in at the beginning of step 35.

36. This is exactly what the while loop below will be doing - every time it runs the code, it tells R to run the code in "modelOfInterest" slightly differently, as the “testStocks” value goes up – from 1, to 2, to 3 and so on until all of your stocks are modeled:

```
i = 1

while(i <= numStocks)
{
  modelOfInterest <-
  paste("specifyModel(Delt(Next(Cl(",predStock,"))) ~
  Cl(",testStocks[i],"))")

  modelOfInterest_eval <- eval(parse(text=modelOfInterest))

  fitModel <- buildModel(modelOfInterest_eval, method="lm",
  training.per=c('2015-06-01','2015-09-25'))

  summary(fitModel)

  stockOfInterest = eval(parse(text=testStocks[i]))

  chartSeries(stockOfInterest, subset='last 3 months',
  name=testStocks[i])

  i = i + 1
}
```

37. Now, we want to save the information about how good the prediction was. To do this, we're just going to extract the "R squared" value out of each fit. This one is easier than it sounds - we're going to "subset" the model summary to only get the info we want, like this:

```
summary(fitModel)

rSquared <- summary(fitModel)$r.squared

rSquared
```

38. ***Finally – the most complicated step of the lab! Woohoo! It's all downhill after this.***
We're going to do the same thing as before, and save the rSquared for every model automatically. To do this, we put it in a while loop, and also create a new variable to save our stock information in (stockFit):

```
i = 1
stockFit = vector()

while(i <= numStocks)
{

  modelOfInterest <-
  paste("specifyModel(Delt(Next(Cl(",predStock,"))) ~
  Cl(",testStocks[i],"))")

  modelOfInterest_eval <- eval(parse(text=modelOfInterest))

  fitModel <- buildModel(modelOfInterest_eval, method="lm",
  training.per=c('2015-06-01','2015-09-25'))

  stockFit[testStocks[i]] <- as.numeric(summary(fitModel)$r.squared)

  stockOfInterest = eval(parse(text=testStocks[i]))

  chartSeries(stockOfInterest, subset='last 3 months',
  name=testStocks[i])

  i = i + 1

}
```

Lab Question: Type in "stockFit". Copy what R returns when you type that in, and in response to this question also write down the name of the stock that had the best predictive power, and how you arrived at that decision. Explain in 1-4 sentences how the above algorithm produced that variable (without worrying about the math, just discuss the order of what happened!).

Part 4: Data Visualization

Now, we're going to visualize the stocks using two dimensions of data: First, the volume (i.e., the number of trades that occurred) of the stock, and second, the strength of that stock in predicting (in this example) Delta. Stocks with a higher volume of trading are more likely to be better predictors, as there is more information on what the "correct" price for those stocks are. The best stock to use for prediction would be one with a relatively high volume, and also a high predictive fit (i.e., the R Squared closest to 1.)

39. We already have the data on how well each stock predicts Delta (StockFit), but we still need the volume of each stock. Luckily, that's very easy to get - remember how we did something similar with AAPL? (look all the way back to step 7):

```
AAPL_Volume <- as.vector(AAPL[,"AAPL.Volume"])
```

40. We want the average volume of the stock, so let's calculate that:

```
AAPL_avg_Volume <- mean(AAPL_Volume)
```

41. Now, we're going to use a while loop to calculate this for all of our stocks - just like before, but this time we're just calculating volume (so it's even easier!). Note the “tabs” don’t matter - I just moved “VolumeOfInterest” over and made it smaller so you could see everything on one line:

```
i = 1

stockVolume = vector()

while(i <= numStocks)
{

VolumeOfInterest <- paste("as.vector(",testStocks[i],"[,'",testStocks[i],".Volume'])", sep="")

  VolumeCalc <- eval(parse(text=VolumeOfInterest))

  stockVolume[i] <- mean(VolumeCalc)

  i = i + 1
}
```

42. Now we have both volume and predictive power – type in the two below lines to verify:

```
stockVolume
stockFit
```

43. Let’s make one more visualization with these two datasets. We're going to use two new R libraries to make this work – RcolorBrewer and portfolio:

```
install.packages("RColorBrewer")
install.packages("portfolio")
library(RColorBrewer)
library(portfolio)
```

44. Finally, we're going to produce Tree Map – this graphic will allow you to argue for which stocks you should base your purchases on. Tree Maps are well known for being capable of

visualizing huge number of data points at once. In our case, we want bigger squares to indicate more trading happened for a stocks, and lighter greens mean they have more predictive power (based on our naive interpretation of R-squared). So, a big, green square means that it's likely a good pick; a big red square means it's not; and a small green square means it might be, but we're not quite as sure (because fewer people are trading that stock):

```
map.market(id=names(stockFit),area=stockVolume,color=stockFit,
group=names(stockFit),main = "Pred. Pwr. & Vol. (Delta)")
```

45. The **final deliverable** for this lab is a readout of what stock you chose to try and purchase, and the best stocks you found to predict it with. Full details are found on the deliverables list below.

Lab Deliverables

The final version of this lab should be turned in on a single word document. It contain the following:

1. Answers to the following questions in "section 1" of your word document:
    1. **(1-15)** Around what date would have been the best to purchase Facebook stock, based on this chart?
    2. **(2-19)** What is the relationship between Facebook and IBM – for example, if IBM closes at a high price today, do we expect a high or low price for Facebook tomorrow?
    3. **(2-22)** In this model, how well are you able to predict Facebook tomorrow based on Twitter today? Can you achieve a higher R squared by changing the training period, and if so what is the highest value you were able to find?
    4. **(2-23)** Which model is more effective in prediction, the model you designed that predicts the closing cost of Facebook, or the one that you just designed that predicts the change of value in facebook?
    5. **(2-24)** Was this model any better? How did you arrive at that answer?
    6. **(2-25)** On your own, try other stocks and training periods to see how strong of a prediction you can achieve. Report your best R-squared and the three lines of code (specifyModel(), buildModel(), and summary()) you used to reach that R-squared. How strong is your model?This isn't easy, or we'd all be rich!
    7. **(3-26)** Edit the above code to make it print out 2 to 12 instead of 1 to 10, and provide your new code.
    8. **(3-35)** Copy what appears when you type "modelOfInterest" the second time, and explain in 1-2 sentences why it's different than the first time you typed it in at the beginning of step 35.
    9. **(3-38)** Type in "stockFit". Copy what R returns when you type that in – what do those values mean? Explain in 1-4 sentences how the above algorithm produced that variable (without worrying about the math, just discuss the order of what happened!).

2. Data Visualizations (which you will turn in as a part of #3 below).
    1. A Tree Map that summarizes all of the stocks you examined based on your data that helps to highlight the stocks you chose.
    2. Other graphics you produce in the lab (i.e., the plots of stock change over time; scatterplots) you think will support your argument for any given stock.

3. As a part of section 2 of your lab documents, you will write a one to two-page (including graphics) description describing the optimal stock purchasing strategy you found. This will include information on which stock you chose to predict, what stocks you chose to predict it, and how well it compares to other possible strategies. Your goal is to convince your audience (in this case, Dan!) that your buying strategy is the best, given the available information. Try to provide compelling evidence that you yourself would believe; feel free to use outside sources in conjunction with the data you produce in this lab. This step should include the visualizations you created in step 2, and should rely heavily (but not necessarily exclusively) on your actual quantitative analysis.

Stretch Goals

1. Produce an algorithm which automatically explores at least 20 different stocks for the best fit, create and provide the Tree Map from these, and explain which stock was best in its predictive power in no more than 1 paragraph.
2. Conduct a similar analysis, but using the day's opening prices as what you're predicting – i.e., does the closing price of Twitter yesterday predict the opening price of Google today?
3. Instead of a "while" loop, write the final step of the code (4-41) using a "for" loop. This will likely require some googling!