Stock Project

Gina Norato

November 13, 2015

Load packages

```
if(!("ggplot2" %in% rownames(installed.packages()))){
  install.packages("ggplot2")
}
if(!("quantmod" %in% rownames(installed.packages()))){
  install.packages("quantmod")
}
if(!("rvest" %in% rownames(installed.packages()))){
  install.packages("rvest")
}
if(!("lubridate" %in% rownames(installed.packages()))){
  install.packages("lubridate")
}
if(!("caret" %in% rownames(installed.packages()))){
  install.packages("caret")
}
if(!("plyr" %in% rownames(installed.packages()))){
  install.packages("plyr")
}
if(!("dplyr" %in% rownames(installed.packages()))){
  install.packages("dplyr")
}
if(!("scales" %in% rownames(installed.packages()))){
  install.packages("scales")
}
if(!("randomForest" %in% rownames(installed.packages()))){
  install.packages("randomForest")
}
if(!("mgcv" %in% rownames(installed.packages()))){
  install.packages("mgcv")
}

library(ggplot2)
library(quantmod)
```

```
## Loading required package: xts
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
```

```
## Loading required package: TTR
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```
library(rvest)
library(lubridate)
library(caret)
```

```
## Loading required package: lattice
```

```
library(plyr)
```

```
##
## Attaching package: 'plyr'
##
## The following object is masked from 'package:lubridate':
##
##     here
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
##
## The following objects are masked from 'package:lubridate':
##
##     intersect, setdiff, union
##
## The following objects are masked from 'package:xts':
##
##     first, last
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(scales)
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
```

```
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(mgcv)
```

```
## Loading required package: nlme
##
## Attaching package: 'nlme'
##
## The following object is masked from 'package:dplyr':
##
##     collapse
##
## This is mgcv 1.8-7. For overview type 'help("mgcv-package")'.
```

Data Acquisition and Management

Stock symbols and stock information were acquired on 11/14/2015. The Wikipedia page used is indicated below, and the stock names and data are saved in the attached documentation. This code will not be re-evaluated.

```
url <- "https://en.wikipedia.org/w/index.php?title=List_of_S%26P_500_companies&oldid=689057793"
symb <- url %>%
  html() %>%
  html_nodes(xpath='//*[@id="mw-content-text"]/table[1]') %>%
  html_table()
symb <- as.data.frame(symb)
symb <- subset(symb,select=c("Ticker.symbol","GICS.Sector"))
names(symb) <- c("symb", "sector")
save(symb,file="symb.Rda")

# create environment
data <- new.env()
getSymbols(symb[,1],env=data)
symb[,1]<-gsub("-", "_", symb[,1])


# save acquired data in the data environment
# data that I used during the project
save(data, file="~/Education/Hopkins/2015 Term 2/Data Science/Project 1/data/mydata.Rdata")
load("~/Education/Hopkins/2015 Term 2/Data Science/Project 1/data/mydata.Rdata")
```

The data were then processed from list form into a long dataframe, retaining Open, Close, and Volume information. The data were saved at this step as well.

```
# make the environment into a list
dat.list <- as.list(data) # dat.list is a list of time series objects

# remove columns from list that are unwanted
dat.list = lapply(dat.list, function(x){
```

```
  #### subsets just open, closed, and volume columns
  cn = colnames(x)
  res = grep("Open|Close|Volume", cn, value = TRUE)
  x = x[, res]
  x
})

# reduce list into wide dataframe
alldf = Reduce(function(...)
        merge(..., all = TRUE),
    dat.list)

alldf = as.data.frame(alldf)
alldf$time = rownames(alldf)
colnames(alldf) = gsub("(.*)[.](Open|Close|Volume)", "\\2.\\1", colnames(alldf))

# fix strange naming (stock BF.B -> BF_B)
colnames(alldf) = gsub("(.*)[.](.*)[.](.*)", "\\1.\\2_\\3", colnames(alldf))

# create super-long dataframe
long = stats::reshape(alldf, direction = "long",
                idvar = "time",
                timevar = "stock",
                varying = grep("[.]", colnames(alldf), value = TRUE))
long = merge(long,symb,all=T,by.x="stock", by.y="symb")
long = long[ order(long[,1], long[,2]), ]
rownames(long) = NULL

# save data
save(long,file="~/Education/Hopkins/2015 Term 2/Data Science/Project 1/data/long.Rda")
```

Load in data:

```
load("../data/long.Rda")
load("../data/symb.Rda")
```

Feature Creation

The data were then cleaned and properly formatted, and also new features were added in order to do data exploration/model development.

```
# proper date formatting
long$time = ymd(long$time)
long$weekday = wday(long$time, label=T)
long$month = month(long$time, label=T)
long$year = year(long$time)

# number of days within stock- useful for lag variables
long$numday<-ave(long$stock, long$stock, FUN=seq_along)

# reverse dates, reversed numbering column
long = long[order(long[,1], rev(long[,2])), ]
```

```
long$numdayrev <- ave(long$stock, long$stock, FUN=seq_along)

# return to original numbering
long = long[order(long[,1],long[,2]), ]

####################
# Create Variables
####################
# current day return (also what we will want to be predicting)
long$return0 <- (long$Close-long$Open) / long$Open

# lagged returns (returns yesterday, two, three days ago and 1 week ago)
long$return1minus <- c(NA,long$return0[1:(nrow(long)-1)])
    long$return1minus[which(long$numday %in% c(1))]<-NA
long$return2minus <- c(NA,NA,long$return0[1:(nrow(long)-2)])
    long$return2minus[which(long$numday %in% c(1:2))]<-NA
long$return3minus <- c(NA,NA,NA,long$return0[1:(nrow(long)-3)])
    long$return3minus[which(long$numday %in% c(1:3))]<-NA
long$returnwkminus <-c(NA,NA,NA,NA,NA,long$return0[1:(nrow(long)-5)])
    long$returnwkminus[which(long$numday %in% c(1:5))]<-NA

# lagged volumes (volumes from yesterday and two days ago)
long$volume1minus <- c(NA,long$Volume[1:(nrow(long)-1)])
    long$volume1minus[which(long$numday %in% c(1))] <- NA

long$volume2minus <- c(NA,NA,long$Volume[1:(nrow(long)-2)])
    long$volume2minus[which(long$numday %in% c(1:2))] <- NA

# lagged close (close yesterday)
long$close1minus <- c(NA,long$Close[1:(nrow(long)-1)])
    long$close1minus[which(long$numday %in% c(1))] <- NA

# future return
# return 5 days in the future
long$close5plus <- c(long$Close[5:nrow(long)],NA,NA,NA,NA)
  long$close5plus[which(long$numdayrev %in% c(1:4))] <- NA
long$return5plus <- (long$close5plus-long$Close)/long$Close

# edit variable types
long$sector <- factor(long$sector)
long$time <- as.Date(long$time)
```

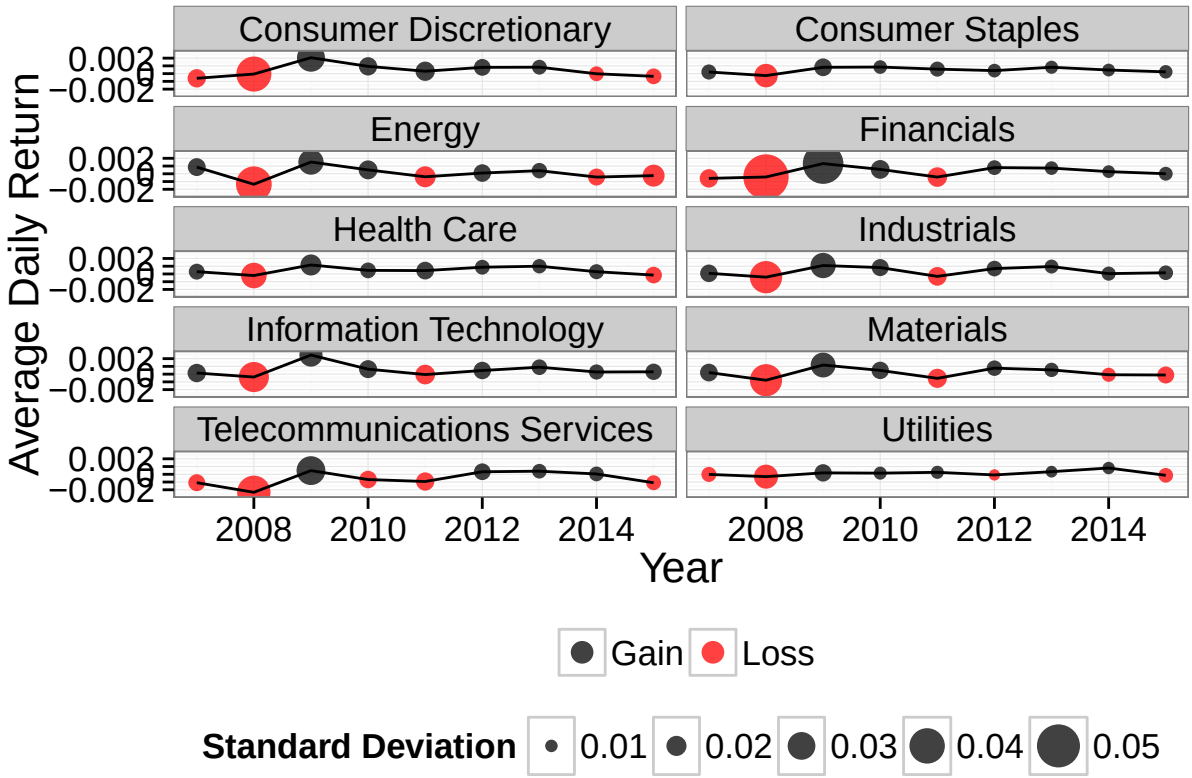
Data Exploration

```
# mean and sd of returns by sector
ddply(long,~sector,summarise,mean=mean(return0, na.rm=T),sd=sd(return0, na.rm=T))
```

```
##                         sector          mean         sd
## 1       Consumer Discretionary  0.0004451174 0.02158409
## 2             Consumer Staples  0.0004652010 0.01436957
## 3                       Energy  0.0001064044 0.02258729
```

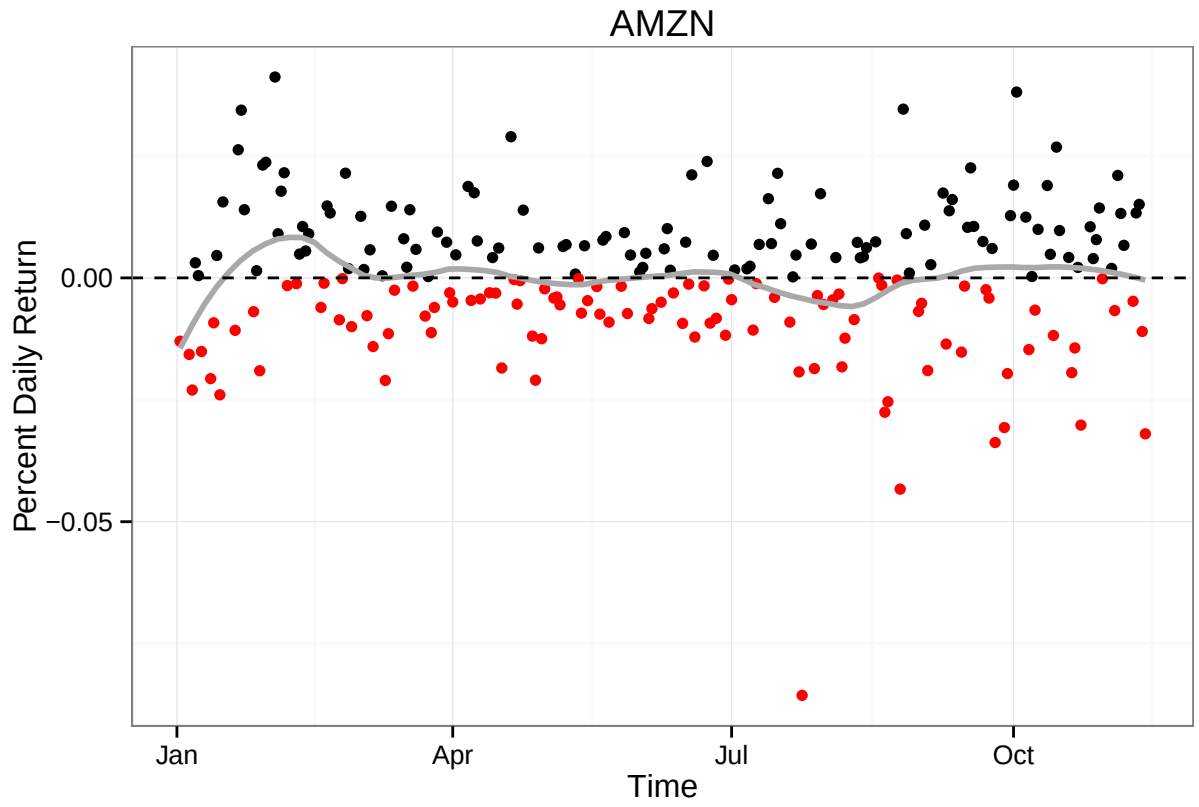
```
## 4                    Financials  0.0002571929 0.02718295
## 5                   Health Care  0.0004697640 0.01702927
## 6                   Industrials  0.0003509647 0.01968007
## 7       Information Technology  0.0005322514 0.01981940
## 8                     Materials  0.0001865671 0.02001025
## 9  Telecommunications Services -0.0005329906 0.02108755
## 10                    Utilities  0.0001375421 0.01386932
```

```
# mean and sd of returns by sector and year
sec.year <- ddply(long, ~sector + year,summarise,mean=mean(return0, na.rm=T),sd=sd(return0, na.rm=T))

#tiff(file="Figure1.tiff", width=7, height=7, res=600, units="in")
ggplot(data=subset(sec.year), aes(x=year,y=mean)) +
  geom_point(aes(size=sd,colour=ifelse(mean>0,"Gain","Loss")), alpha=0.75) +
  scale_size("Standard Deviation", range=c(2,8)) +
  geom_line() + facet_wrap(~sector, nrow=5) +
    theme_bw() +
  ylab("Average Daily Return") + xlab("Year") +
  #geom_hline(yintercept=0, linetype="dashed") +
    coord_cartesian(ylim=c(-0.003,0.003)) +
  scale_color_manual("",values=c("black", "red")) +
  scale_y_continuous(breaks=c(-0.002,-0.001,0,0.001,0.002),
                     labels=c(-0.002,"",0,"",0.002)) +
  theme(legend.position="bottom", text=element_text(size=16)) +
  guides(colour=guide_legend(override.aes=list(size=4)))
```



```
#dev.off()


# plotting an individual stock over some easily visualized span
plotstockname <- "AMZN"
ggplot(aes(x = time, y = return0), data = subset(long,stock==plotstockname& time >= "2015-01-01")) +
  geom_point(aes(colour=ifelse(return0>0,"Gain","Loss"))) + xlab("Time") + ylab("Percent Daily Return") +
  stat_smooth(span=0.3, se=F, colour="dark grey", size=1) +
  scale_color_manual(guide=FALSE, values=c("black", "red")) +
  theme_bw() + geom_hline(yintercept=0, linetype="dashed") +
  ggtitle(plotstockname)
```

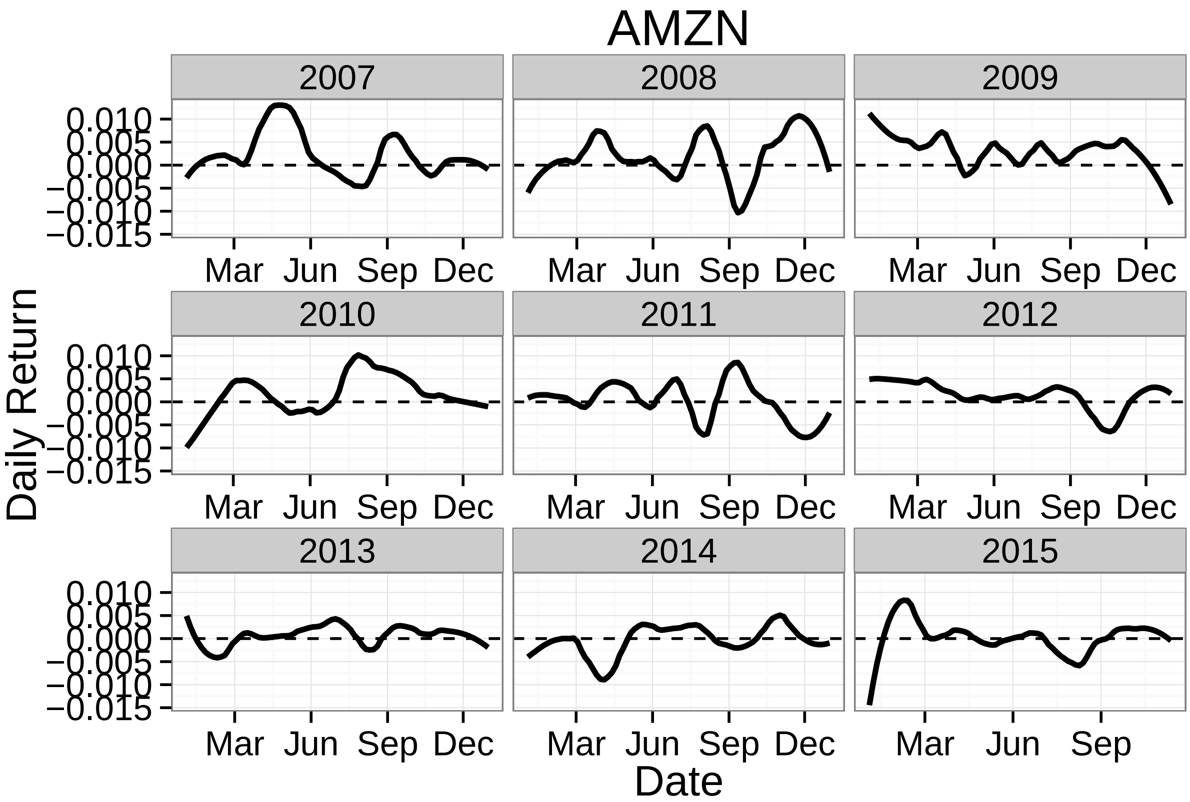
```
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method = x' to ch
```




```
# plotting one stock faceted by year
#tiff(file="Figure2.tiff", width=7, height=7, res=600, units="in")
ggplot(aes(x = time, y = return0),
       data = subset(long,stock==plotstockname & !is.na(Open))) +
  stat_smooth(span=0.3, se=F, colour="black",size=1) +
  theme_bw() + geom_hline(yintercept=0, linetype="dashed") +
  ggtitle(plotstockname) + facet_wrap(~year, scales="free_x") +
  xlab("Date") +scale_x_date(breaks=date_breaks("3 months"),labels = date_format("%b")) + ylab("Daily Re
  theme(text=element_text(size=16))
```

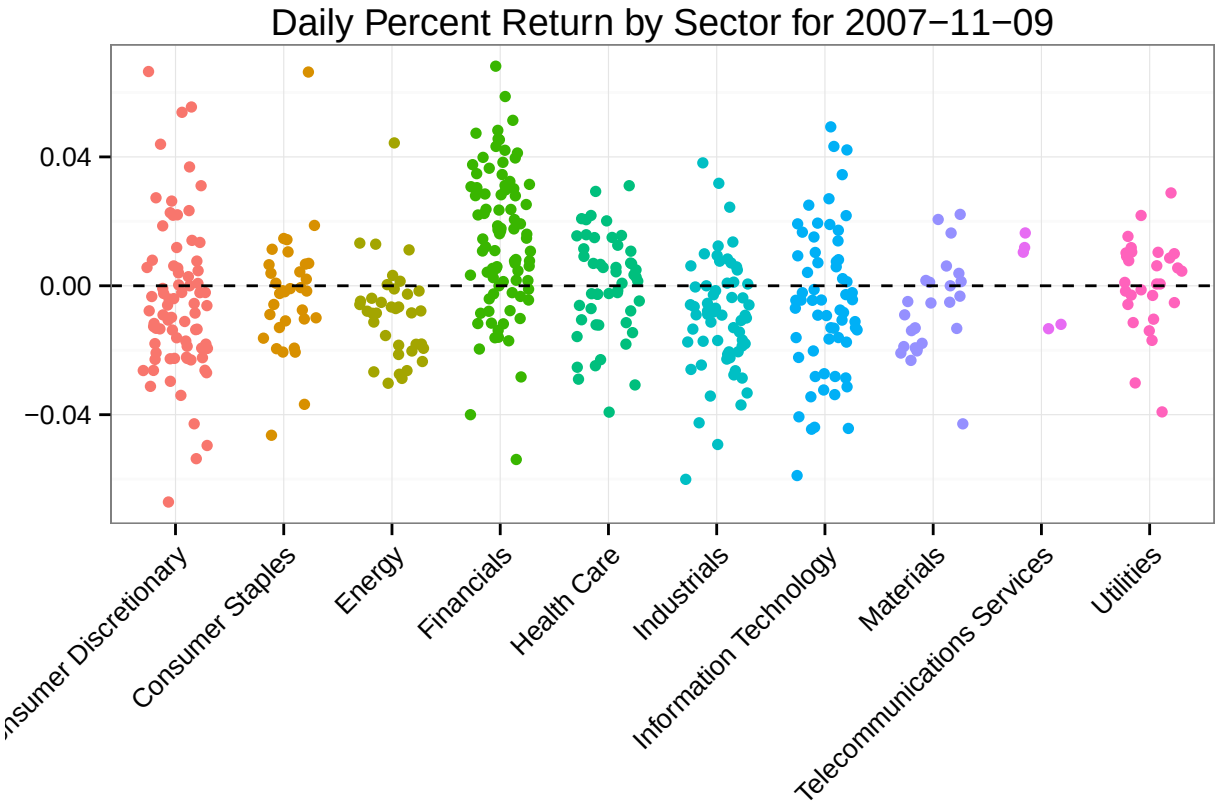
```
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method = x' to ch
```

```
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method = x' to cl
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method = x' to cl
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method = x' to cl
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method = x' to cl
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method = x' to cl
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method = x' to cl
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method = x' to cl
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method = x' to cl
```




```
#dev.off()


# plot daily return for all stocks on a certain day
plotdate <- "2007-11-09"
ggplot(aes(x = sector, y = return0, colour=sector), data =subset(long,time==plotdate)) +
  geom_point(position=position_jitter(0.3)) +
  xlab(NULL) + ylab(NULL) +
  ggtitle(paste0("Daily Percent Return by Sector for ",plotdate)) +
  geom_hline(yintercept=0, linetype="dashed") +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust=1)) +
  guides(colour=F)
```

```
## Warning: Removed 37 rows containing missing values (geom_point).
```

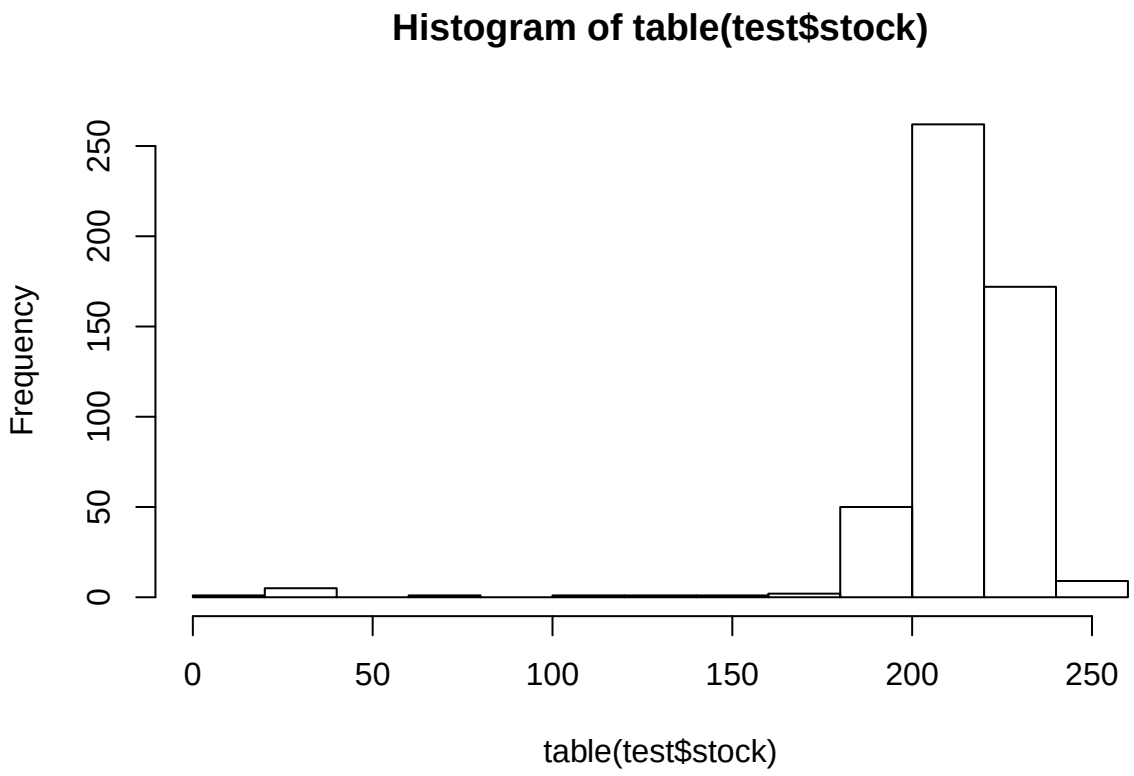



```
# number of stocks in each year of data
stocksdf <- subset(long,select=c("stock","year", "Open"))
stocksdf <- stocksdf[complete.cases(stocksdf),]
stocksum <- ddply(stocksdf,~stock,summarise,year=min(year))
stocksum$y2007 <- ifelse(stocksum$year>=2007,1,0)
stocksum <- ddply(stocksum, ~ year, summarize,n=n())
stocksum$availstock <- cumsum(stocksum$n)
```

Modeling

Split Data

```
# create testing and training sets, from 2013 on, complete cases
model.long <- subset(long,time>="2013-01-01")
comp.long <- model.long[complete.cases(model.long),] # this gets used in CV as well

# partition into training/CV and test- set aside test until end
set.seed(129031)
partition <- createDataPartition(comp.long$return0,p=0.7,list=F)
train <- comp.long[partition,]
test <- comp.long[-partition,]

# tabulate test to see if we have good splits
hist(table(test$stock))
```




```
head(sort(table(test$stock)),10)
```

```
##
##  HPE  WRK  KHC PYPL CPGX BXLT QRVO NAVI GOOG ALLE
##    2   21   24   25   28   31   71  118  124  147
```

```
length(unique(test$stock)) # all stocks included
```

```
## [1] 505
```

Modeling

1 Day Return

```
############### Cross Validation on Training Set
# create folds
k = 3
folds <- createFolds(train$return0, k=k)

# create blank cvpred and blank rmse
train$cvpred0 <- rep(NA,nrow(train))
train$cvpredlwr0 <- rep(NA,nrow(train))
```

```
train$cvpredupr0 <- rep(NA,nrow(train))
rmse0 <- rmse.glm0 <- rmse.rf0 <- rep(NA,k)

for (i in 1:k){
  print(i)
  # take subsets of comp.long
  cvtrain <- train[-folds[[i]],]
  cvtest <- train[folds[[i]],]

  # GLM
  glm <- glm(return0 ~ month + weekday + return1minus + return2minus + return3minus +
                 returnwkminus + volume1minus + volume2minus + close1minus +
                 sector + year + year*month + year*weekday, data=cvtrain)
  glmpred <- predict(glm,cvtest)
  rmse.glm0[i] <- sqrt(sum((glmpred - cvtest$return0)^2) / nrow(cvtest))

  # Random Forest
  rf <- randomForest(return0 ~ weekday + month + return1minus + return2minus + return3minus +
                 returnwkminus + volume1minus + volume2minus + close1minus + sector + year,
                 data=cvtrain,
                 ntree=30,
                 importance=T,
                 nodesize=nrow(cvtrain)*0.01)
  rfpred <- predict(rf,cvtest)
  rmse.rf0[i] <- sqrt(sum((rfpred - cvtest$return0)^2) / nrow(cvtest))
  # node size of 1% of training data increases speed without losing substantial MSE (0.01265 .1% vs 0.0
  # test tree sizes of 30 (0.012985), 100 (0.012988)

  # model averaging
  predDF <- data.frame(glmpred,rfpred,return0=cvtest$return0)
  combModFit <- lm(return0 ~ glmpred + rfpred,data=predDF)
  combPred <- predict(combModFit,predDF, interval="prediction")

  # get predictions from model averaging, fill into dataframe
  train[folds[[i]], ]$cvpred0 <- combPred[,1]
  train[folds[[i]], ]$cvpredlwr0 <- combPred[,2]
  train[folds[[i]], ]$cvpredupr0 <- combPred[,3]

  # calculate rMSE for this fold, store
  rmse0[i] <- sqrt(sum((combPred[,1]- cvtest$return0)^2) / nrow(cvtest))
}
```

```
## [1] 1
## [1] 2
## [1] 3
```

```
rmse0
```

```
## [1] 0.01273923 0.01269325 0.01282772
```

```
mean(rmse0)
```

```
## [1] 0.0127534
```

```
######### Fit on Training Once, Predict Test

# GLM
glm0 <- glm(return0 ~ month + weekday + return1minus + return2minus + return3minus +
                  returnwkminus + volume1minus + volume2minus + close1minus +
                  sector + year + year*month + year*weekday, data=train)
glmpred <- predict(glm,test)
rmse.glmtest0 <- sqrt(sum((glmpred - test$return0)^2) / nrow(test))

# Random Forest
rf0 <- randomForest(return0 ~ weekday + month + return1minus + return2minus + return3minus +
                  returnwkminus + volume1minus + volume2minus + close1minus + sector + year,
                  data=train,
                  ntree=30,
                  importance=T,
                  nodesize=nrow(train)*0.01)
rfpred <- predict(rf,test)
rmse.rftest0 <- sqrt(sum((rfpred - test$return0)^2) / nrow(test))

# model averaging
predDF <- data.frame(glmpred,rfpred,return0=test$return0)
combModFit0 <- lm(return0 ~ glmpred + rfpred,data=predDF) # this is our final model
combPred <- predict(combModFit0,predDF, interval="prediction")

  # get predictions from model averaging, fill into dataframe
test$pred0 <- combPred[,1]
test$predlwr0 <- combPred[,2]
test$predupr0 <- combPred[,3]

  # calculate rMSE for this fold, store
rmse.final0 <- sqrt(sum((combPred[[1]]- test$return0)^2) / nrow(test))
rmse.final0
```

```
## [1] 0.01363435
```

5-day Return

```
############### Cross Validation on Training Set
# create folds
k = 3
folds <- createFolds(train$return5plus, k=k)

# create blank cvpred and blank rmse
train$cvpred5 <- rep(NA,nrow(train))
train$cvpredlwr5 <- rep(NA,nrow(train))
train$cvpredupr5 <- rep(NA,nrow(train))
rmse5 <- rmse.glm5 <- rmse.rf5 <- rep(NA,k)

for (i in 1:k){
  print(i)
  # take subsets of comp.long
```

```
  cvtrain <- train[-folds[[i]],]
  cvtest <- train[folds[[i]],]

  # GLM
  glm <- glm(return5plus ~ month + weekday + return1minus + return2minus + return3minus +
                  returnwkminus + volume1minus + volume2minus + close1minus +
                  sector + year + year*month + year*weekday, data=cvtrain)
  glmpred <- predict(glm,cvtest)
  rmse.glm5[i] <- sqrt(sum((glmpred - cvtest$return5plus)^2) / nrow(cvtest))

  # Random Forest
  rf <- randomForest(return5plus ~ weekday + month + return1minus + return2minus + return3minus +
                  returnwkminus + volume1minus + volume2minus + close1minus + sector + year,
                  data=cvtrain,
                  ntree=30,
                  importance=T,
                  nodesize=nrow(cvtrain)*0.01)
  rfpred <- predict(rf,cvtest)
  rmse.rf5[i] <- sqrt(sum((rfpred - cvtest$return5plus)^2) / nrow(cvtest))

  # model averaging
  predDF <- data.frame(glmpred,rfpred,return5plus=cvtest$return5plus)
  combModFit <- lm(return5plus ~ glmpred + rfpred,data=predDF)
  combPred <- predict(combModFit,predDF, interval="prediction")

  # get predictions from model averaging, fill into dataframe
  train[folds[[i]], ]$cvpred5 <- combPred[,1]
  train[folds[[i]], ]$cvpredlwr5 <- combPred[,2]
  train[folds[[i]], ]$cvpredupr5 <- combPred[,3]

  # calculate rMSE for this fold, store
  rmse5[i] <- sqrt(sum((combPred[,1]- cvtest$return5plus)^2) / nrow(cvtest))
}
```

```
## [1] 1
## [1] 2
## [1] 3
```

```
rmse5
```

```
## [1] 0.03315894 0.03237609 0.03292636
```

```
mean(rmse5)
```

```
## [1] 0.03282046
```

```
######### Fit on Training Once, Predict Test
# GLM
glm5 <- glm(return5plus ~ month + weekday + return1minus + return2minus + return3minus +
                  returnwkminus + volume1minus + volume2minus + close1minus +
                  sector + year + year*month + year*weekday, data=train)
glmpred <- predict(glm,test)
```

```
rmse.glmtest5 <- sqrt(sum((glmpred - test$return5plus)^2) / nrow(test))

# Random Forest
rf5 <- randomForest(return5plus ~ weekday + month + return1minus + return2minus + return3minus +
                    returnwkminus + volume1minus + volume2minus + close1minus + sector + year,
                    data=train,
                    ntree=30,
                    importance=T,
                    nodesize=nrow(train)*0.01)
rfpred <- predict(rf,test)
rmse.rftest5 <- sqrt(sum((rfpred - test$return5plus)^2) / nrow(test))

# model averaging
predDF <- data.frame(glmpred,rfpred,return5plus=test$return5plus)
combModFit5 <- lm(return5plus ~ glmpred + rfpred,data=predDF) # this is our final model
combPred <- predict(combModFit5,predDF,interval="prediction")

  # get predictions from model averaging, fill into dataframe
test$pred5 <- combPred[,1]
test$predlwr5 <- combPred[,2]
test$predupr5 <- combPred[,3]

  # calculate rMSE for this fold, store
rmse.final5 <- sqrt(sum((combPred[,1]- test$return5plus)^2) / nrow(test))
rmse.final5
```

```
## [1] 0.03249893
```

Data were saved at this step for reference and for use with the attached prediction function ("Function Script.R")

```
#################
# save test df
save(test,file="test.Rda")
# save train df
save(train,file="train.Rda")
# save models
save(glm0,rf0,combModFit0,glm5, rf5, combModFit5, file="models.Rda")
save(rf5)
save(combModFit5)
save(combModFit0)
```

Investigate rank predictions by pulling new days, predicting and ranking (11/16-11/20)

For efficiency, this code will not be re-evaluated. The outputs were saved and loaded in below

```
########## reload data from 11/16-11/20 for my stocks
data <- new.env()
getSymbols(symb[,1],env=data,from="2015-11-16",to="2015-11-20")
symb[,1]<-gsub("-", "_", symb[,1])
```

```
########## process data
dat.list <- as.list(data) # dat.list is a list of time series objects

# remove columns from list that are unwanted
dat.list = lapply(dat.list, function(x){
  #### subsets just open, closed, and volume columns
  cn = colnames(x)
  res = grep("Open|Close|Volume", cn, value = TRUE)
  x = x[, res]
  x
})

# reduce list into wide dataframe
alldf = Reduce(function(...)
        merge(..., all = TRUE),
    dat.list)

alldf = as.data.frame(alldf)
alldf$time = rownames(alldf)
colnames(alldf) = gsub("(.*)[.](Open|Close|Volume)", "\\2.\\1", colnames(alldf))

# fix strange naming (stock BF.B -> BF_B)
colnames(alldf) = gsub("(.*)[.](.*)[.](.*)", "\\1.\\2_\\3", colnames(alldf))

# create super-long dataframe
new.long = stats::reshape(alldf, direction = "long",
               idvar = "time",
               timevar = "stock",
               varying = grep("[.]", colnames(alldf), value = TRUE))
new.long = merge(new.long,symb,all=T,by.x="stock", by.y="symb")
new.long = new.long[ order(new.long[,1], new.long[,2]), ]
rownames(new.long) = NULL

############### add variables
# first- rbind additional days from my data
new.long<-rbind(subset(long,select=c("stock","time","Open","Close","Volume","sector"),
                           time>="2015-11-09"),
                     new.long)
new.long <- new.long[order(new.long[,1], new.long[,2]),]

# proper date formatting
new.long$time = ymd(new.long$time)
new.long$weekday = wday(new.long$time, label=T)
new.long$month = month(new.long$time, label=T)
new.long$year = year(new.long$time)

# number of days within stock- useful for lag variables
new.long$numday<-ave(new.long$stock, new.long$stock, FUN=seq_along)

# reverse dates, reversed numbering column
new.long = new.long[order(new.long[,1], rev(new.long[,2])), ]
new.long$numdayrev <- ave(new.long$stock, new.long$stock, FUN=seq_along)
```

```
# return to original numbering
new.long = new.long[order(new.long[,1],new.long[,2]), ]

####################
# Create Variables
####################
# current day return (also what we will want to be predicting)
new.long$return0 <- (new.long$Close-new.long$Open) / new.long$Open

# lagged returns (returns yesterday, two, three days ago and 1 week ago)
new.long$return1minus <- c(NA,new.long$return0[1:(nrow(new.long)-1)])
    new.long$return1minus[which(new.long$numday %in% c(1))]<-NA
new.long$return2minus <- c(NA,NA,new.long$return0[1:(nrow(new.long)-2)])
    new.long$return2minus[which(new.long$numday %in% c(1:2))]<-NA
new.long$return3minus <- c(NA,NA,NA,new.long$return0[1:(nrow(new.long)-3)])
    new.long$return3minus[which(new.long$numday %in% c(1:3))]<-NA
new.long$returnwkminus <-c(NA,NA,NA,NA,NA,new.long$return0[1:(nrow(new.long)-5)])
    new.long$returnwkminus[which(new.long$numday %in% c(1:5))]<-NA

# lagged volumes (volumes from yesterday and two days ago)
new.long$volume1minus <- c(NA,new.long$Volume[1:(nrow(new.long)-1)])
    new.long$volume1minus[which(new.long$numday %in% c(1))] <- NA

new.long$volume2minus <- c(NA,NA,new.long$Volume[1:(nrow(new.long)-2)])
    new.long$volume2minus[which(new.long$numday %in% c(1:2))] <- NA

# lagged close (close yesterday)
new.long$close1minus <- c(NA,new.long$Close[1:(nrow(new.long)-1)])
    new.long$close1minus[which(new.long$numday %in% c(1))] <- NA

# future return
# return 5 days in the future
new.long$close5plus <- c(new.long$Close[5:nrow(new.long)],NA,NA,NA,NA)
  new.long$close5plus[which(new.long$numdayrev %in% c(1:4))] <- NA
new.long$return5plus <- (new.long$close5plus-new.long$Close)/new.long$Close

# edit variable types
new.long$sector <- factor(new.long$sector)
new.long$time <- as.Date(new.long$time)

############### run models and add predictions onto dataframe
comp.new.long.0<- new.long[complete.cases(new.long[,1:18]),]
comp.new.long.5<- new.long[complete.cases(new.long),]
## 0
glmpred <- predict(glm0,comp.new.long.0)
rfpred <- predict(rf0,comp.new.long.0)
pred0 <- predict(combModFit0,data.frame(glmpred=glmpred,rfpred=rfpred), interval="prediction")

comp.new.long.0$pred0 <- pred0[,1]
comp.new.long.0$predlwr0 <- pred0[,2]
comp.new.long.0$predupr0 <- pred0[,3]

## 5
```

```
glmpred <- predict(glm5,comp.new.long.5)
rfpred <- predict(rf5,comp.new.long.5)
pred5 <- predict(combModFit5,data.frame(glmpred=glmpred,rfpred=rfpred), interval="prediction")

comp.new.long.5$pred5 <- pred5[,1]
comp.new.long.5$predlwr5 <- pred5[,2]
comp.new.long.5$predupr5 <- pred5[,3]

### get predicted and true ranks by day
# return0
comp.new.long.0<-comp.new.long.0[order(comp.new.long.0[,'time'],-comp.new.long.0[,'return0']), ]
comp.new.long.0$truerank<- ave(as.character(comp.new.long.0$time), as.character(comp.new.long.0$time), I

comp.new.long.0<-comp.new.long.0[order(comp.new.long.0[,'time'],-comp.new.long.0[,'predlwr0']), ]
comp.new.long.0$predrank<- ave(as.character(comp.new.long.0$time), as.character(comp.new.long.0$time), I

# return5
comp.new.long.5<-comp.new.long.5[order(comp.new.long.5[,'time'],-comp.new.long.5[,'return5plus']), ]
comp.new.long.5$truerank<- ave(as.character(comp.new.long.5$time), as.character(comp.new.long.5$time), I

comp.new.long.5<-comp.new.long.5[order(comp.new.long.5[,'time'],-comp.new.long.5[,'predlwr5']), ]
comp.new.long.5$predrank<- ave(as.character(comp.new.long.5$time), as.character(comp.new.long.5$time), I

save(comp.new.long.0,comp.new.long.5,file="newpred.Rda")
```

```
load("../data/newpred.Rda")
cleandf <- function(df){
  dat <- subset(df,truerank %in% 1:10)
  dat <- dat[order(as.numeric(dat[,'truerank'])),]
  dat <- subset(dat,select=c("stock","truerank","predrank"))
  return(dat)
}

# same-day return predictions
nov16<- subset(comp.new.long.0,time=="2015-11-16")
nov16 <- cleandf(nov16)
nov16
```

```
##      stock truerank predrank
## 3441  FLIR        1      344
## 4511   RRC        2       40
## 3051   COG        3      146
## 4151  NFLX        4       77
## 3091   CSC        5      276
## 4951   WMB        6      233
## 4656   SWN        7       96
## 4131   NBL        8       92
## 3541   GNW        9      481
## 3066   COP       10      343
```

```
nov17<- subset(comp.new.long.0,time=="2015-11-17")
nov17 <- cleandf(nov17)
nov17
```

```
##      stock truerank predrank
## 2752   ARG        1      406
## 4832  URBN        2      105
## 3092   CSC        3      143
## 3462  FOSL        4       21
## 4447  QRVO        5      135
## 2527     A        6      431
## 3262    EA        7      147
## 4152  NFLX        8      326
## 3977   MAT        9      344
## 4132   NBL       10      379
```

```
nov18<- subset(comp.new.long.0,time=="2015-11-18")
nov18 <- cleandf(nov18)
nov18
```

```
##      stock truerank predrank
## 3043   CNX        1       10
## 4738  TRIP        2      287
## 4688   TDC        3      232
## 4418   PVH        4      292
## 4868  VIAB        5      180
## 4898  VRTX        6      219
## 4963   WRK        7      279
## 2843  BIIB        8      473
## 2958  CELG        9       90
## 2903    CA       10      246
```

```
nov19<- subset(comp.new.long.0,time=="2015-11-19")
nov19 <- cleandf(nov19)
nov19
```

```
##      stock truerank predrank
## 4194   NSC        1      106
## 2814   BBY        2       13
## 4054   MNK        3       36
## 3724  INTC        4      143
## 3264    EA        5      346
## 3104   CSX        6      129
## 3124  CTXS        7      218
## 2534    AA        8       37
## 4494    RL        9      177
## 4569   SJM       10      182
```

```
nov20<- subset(comp.new.long.0,time=="2015-11-20")
nov20 <- cleandf(nov20)
nov20
```

```
##      stock truerank predrank
## 3545   GNW        1      191
## 3565   GPS        2      335
## 3325  EQIX        3      153
```

```
## 4895  VRSN        4      326
## 3860   KSS        5      160
## 3665   HPE        6      438
## 2630   AET        7       61
## 2985    CI        8      427
## 4725   TJX        9      314
## 2820   BCR       10      333
```

```
# 5 day return prediction
nov16.5 <- cleandf(comp.new.long.5)
nov16.5
```

```
##      stock truerank predrank
## 2751   ARG        1      170
## 3461  FOSL        2        7
## 4151  NFLX        3       11
## 4506  ROST        4      105
## 4191   NSC        5      485
## 3101   CSX        6      172
## 3206  DLTR        7       46
## 4891  VRSN        8      345
## 4736  TRIP        9      445
## 4566   SJM       10      295
```

‘‘‘