Quantitative Analysis

Ross Bennett

November 27, 2013

Abstract

The goal of this vignette is to demonstrate key concepts in Financial Risk Manager (FRM (R)) Part 1: Quantitative Analysis using R and the GARPFRM package. This vignette will cover exploratory data analysis, basic probability and statistics, and linear regression.

Contents

1 Exploratory Data Analysis

Load the GARPFRM package and the `returns` dataset. The `returns` dataset includes weekly returns for SPY, AAPL, XOM, GOOG, MSFT, and GE from 2005-01-14 to 2013-11-22.

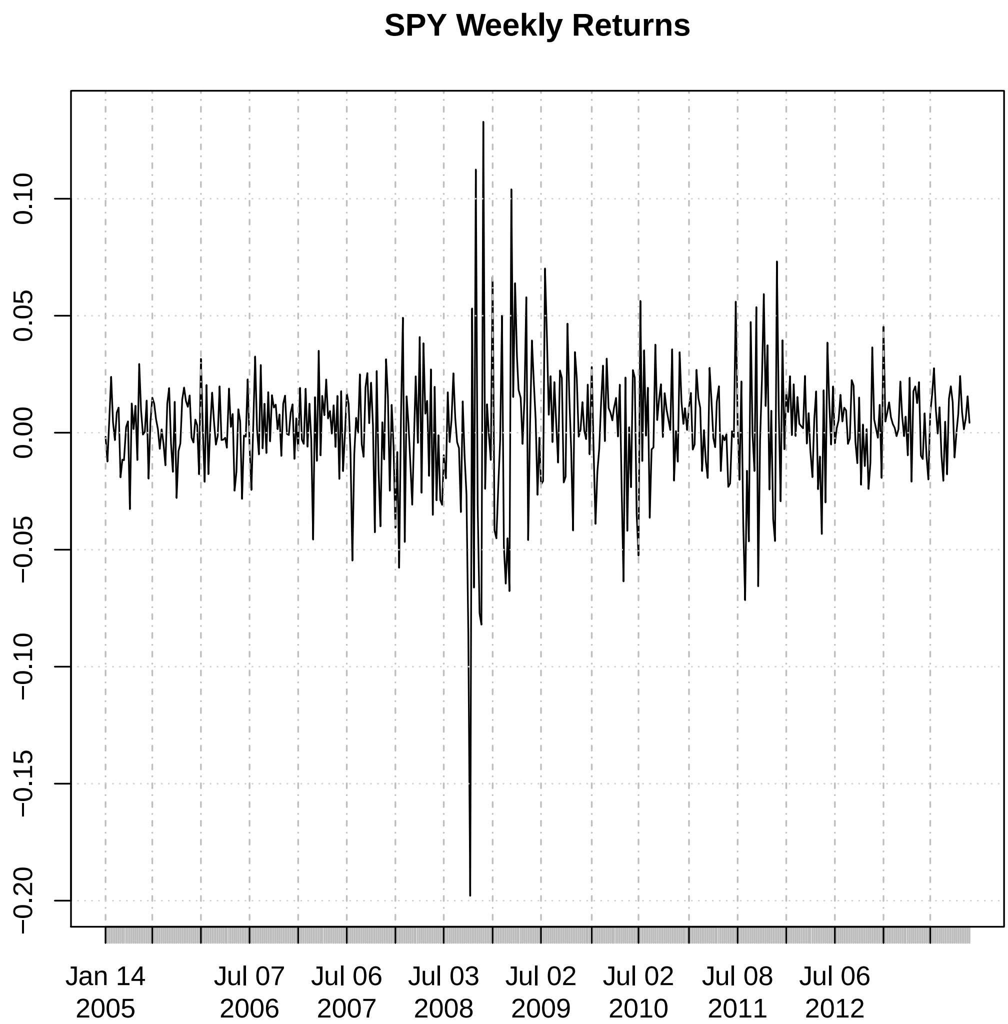
```
suppressMessages(library(GARPFRM))
data(returns)
```

The exploratory data analysis, basic probability and statistics will use the SPY weekly returns.

```r
SPY.ret <- returns[, "SPY"]
```

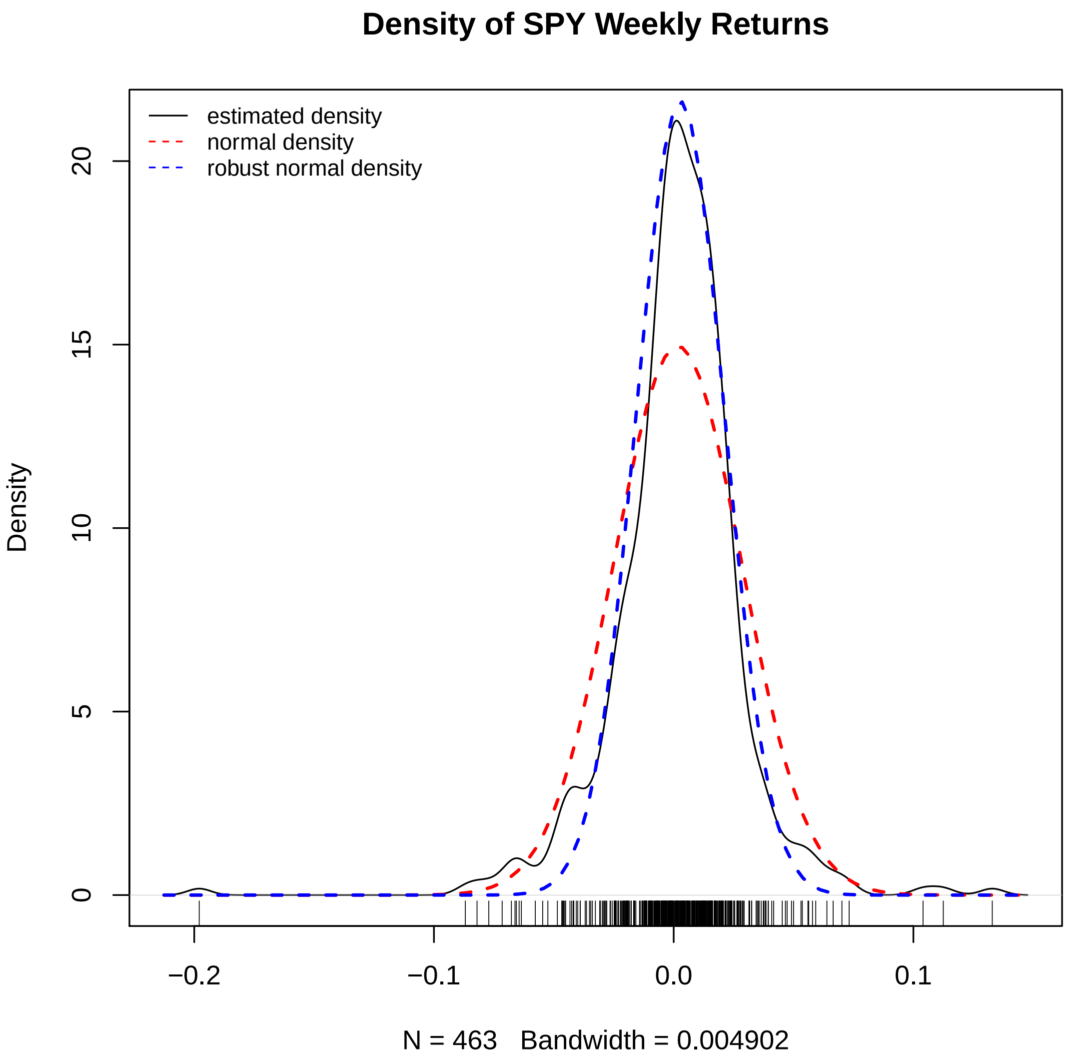
Plot of the SPY weekly returns.

```r
plot(SPY.ret, main = "SPY Weekly Returns")
```



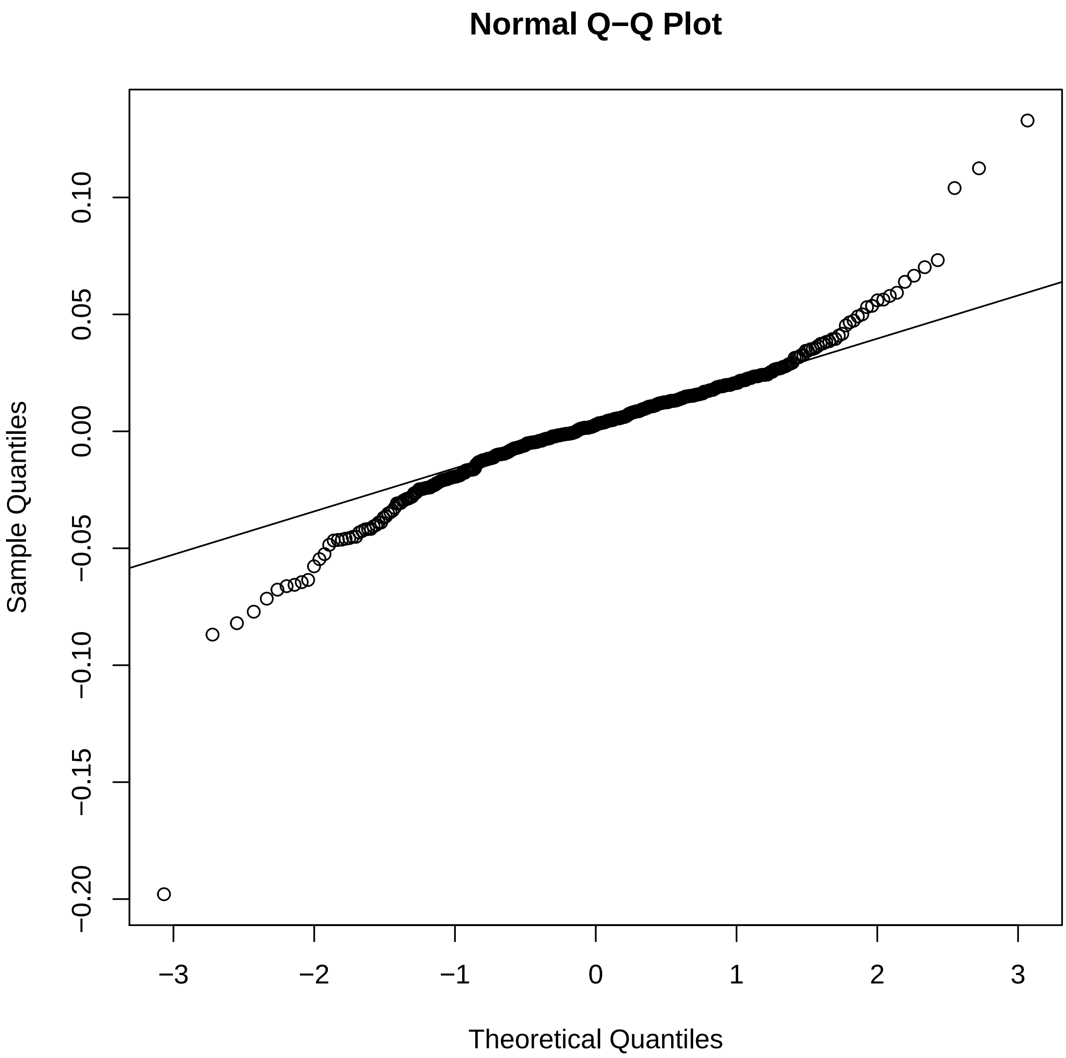
The density of the SPY weekly returns is plotted to better understand its distribution. A normal density is overlayed on the plot with standard estimates of the sample mean and standard deviation. Another normal density is overlayed using robust estimates. It is clear from the chart that the robust estimates provide a better fit than the standard estimates of the sample mean and sample standard deviation, but it is not clear if the SPY returns are normally distributed.

```r
# Plot the density of SPY Weekly Returns
plot(density(SPY.ret), main = "Density of SPY Weekly Returns")
rug(SPY.ret)
# sample estimates
curve(dnorm(x, mean = mean(SPY.ret), sd = sd(SPY.ret)), add = TRUE, col = "red",
    lty = 2, lwd = 2)
# robust estimates
curve(dnorm(x, mean = median(SPY.ret), sd = mad(SPY.ret)), add = TRUE, col = "blue",
    lty = 2, lwd = 2)
legend("topleft", legend = c("estimated density", "normal density", "robust normal density"),
    col = c("black", "red", "blue"), lty = c(1, 2, 2), bty = "n", cex = 0.8)
```



Quantile-Quantile plot of SPY weekly returns. It can be seen from the Normal Q-Q plot that the SPY returns have "fat tails".

```
qqnorm(SPY.ret)
qqline(SPY.ret)
```



We can test if the SPY weekly returns came from a normal distribution using the Shapiro-Wilk test of normality. The null hypothesis is that the data came from a normal distribution. The p-value is very small and we can reject the null hypothesis.

```
shapiro.test(coredata(SPY.ret))

##
##  Shapiro-Wilk normality test
##
## data:  coredata(SPY.ret)
## W = 0.9096, p-value = 5.567e-16
```

1.1 Basic Statistics

Here we calculate some basic statisitics on the SPY weekly returns.

```
# Sample mean of SPY return
mean(SPY.ret)

## [1] 0.001658

# Sample Variance of SPY returns
var(SPY.ret)

##           SPY
## SPY 0.0007114

# Sample standard deviation of SPY returns
sd(SPY.ret)

## [1] 0.02667

# Standard error of SPY returns
sd(SPY.ret)/sqrt(nrow(SPY.ret))

## [1] 0.00124

# Sample skewness of SPY returns.  See ?skewness for additional methods
# for calculating skewness
skewness(SPY.ret, method = "sample")

## [1] -0.7003

# Sample kurtosis of SPY returns.  See ?kurtosis for additional methods
# for calculating kurtosis
kurtosis(SPY.ret, method = "sample")

## [1] 11.96
```

```r
# Summary statistics of SPY returns
summary(SPY.ret)
```

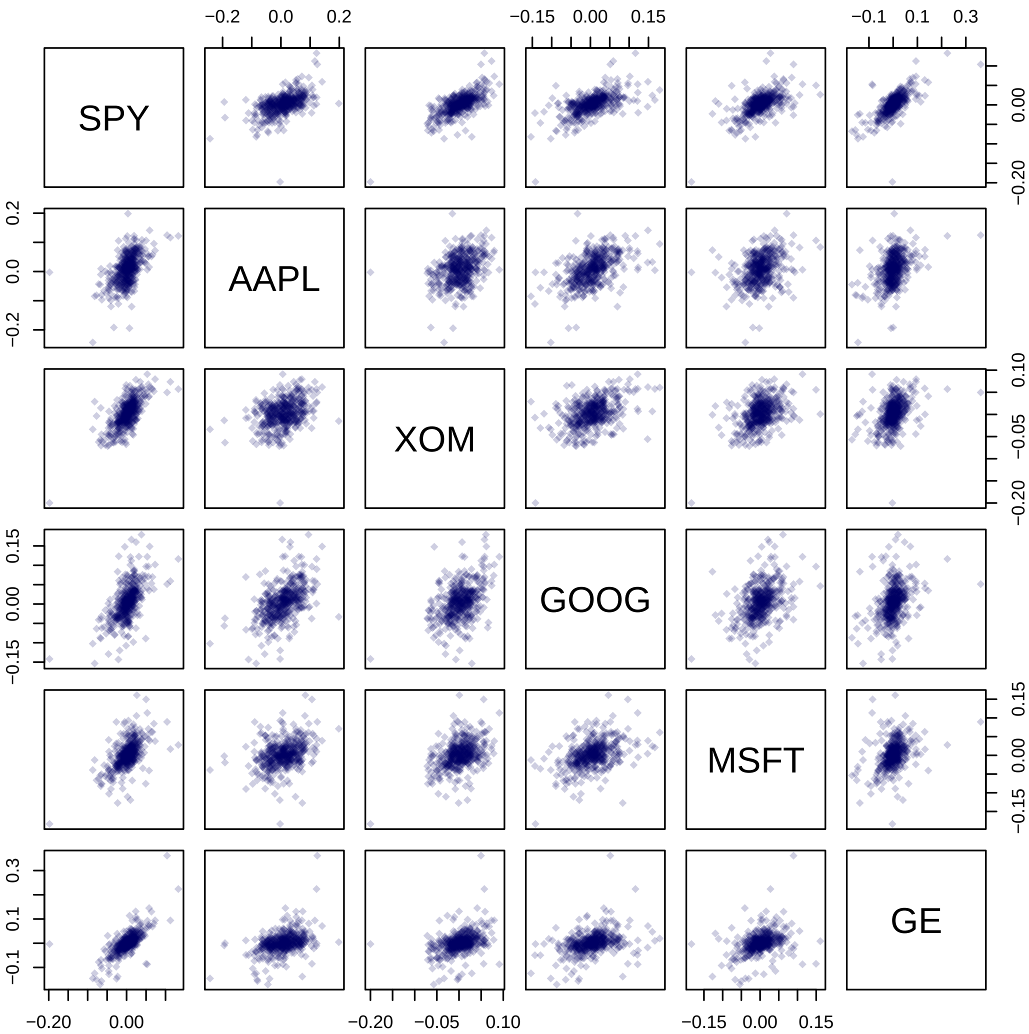
```
##      Index                 SPY
##  Min.   :2005-01-14   Min.   :-0.19792
##  1st Qu.:2007-04-02   1st Qu.:-0.00974
##  Median :2009-06-19   Median : 0.00280
##  Mean   :2009-06-18   Mean   : 0.00166
##  3rd Qu.:2011-09-05   3rd Qu.: 0.01516
##  Max.   :2013-11-22   Max.   : 0.13291
```

```r
# Sample quantiles of SPY returns
quantile(SPY.ret, probs = c(0, 0.25, 0.5, 0.75, 1))
```

```
##        0%       25%       50%       75%      100%
## -0.197921 -0.009744  0.002803  0.015164  0.132913
```

Scatter plot of each pair of assets in the returns dataset.

```r
pairs(coredata(returns), pch = 18, col = rgb(0, 0, 100, 50, maxColorValue = 255))
```



Correlation and covariance matrices of assets in the returns dataset.

```r
# Sample correlation of returns
cor(returns)
```

```
##          SPY   AAPL    XOM   GOOG   MSFT     GE
## SPY  1.0000 0.5301 0.6987 0.5891 0.5837 0.7105
## AAPL 0.5301 1.0000 0.3427 0.5011 0.3271 0.4143
## XOM  0.6987 0.3427 1.0000 0.4317 0.4229 0.3786
## GOOG 0.5891 0.5011 0.4317 1.0000 0.4048 0.3526
## MSFT 0.5837 0.3271 0.4229 0.4048 1.0000 0.3447
## GE   0.7105 0.4143 0.3786 0.3526 0.3447 1.0000
```

```r
# Sample covariance of returns
cov(returns)
```

```
##            SPY      AAPL       XOM      GOOG      MSFT        GE
## SPY  0.0007114 0.0007224 0.0005693 0.0007086 0.0005597 0.0008329
## AAPL 0.0007224 0.0026103 0.0005349 0.0011545 0.0006009 0.0009303
## XOM  0.0005693 0.0005349 0.0009331 0.0005947 0.0004645 0.0005083
## GOOG 0.0007086 0.0011545 0.0005947 0.0020335 0.0006562 0.0006989
## MSFT 0.0005597 0.0006009 0.0004645 0.0006562 0.0012926 0.0005447
## GE   0.0008329 0.0009303 0.0005083 0.0006989 0.0005447 0.0019318
```

1.2 Distributions

R has functions to compute the density, distribution function, quantile, and random number generation for several distributions. The continuous distributions covered in chapter 1 are listed here.

- Normal Distribution: `dnorm`, `pnorm`, `qnorm`, `rnorm`
- Chi-Squared Distribution: `dchisq`, `pchisq`, `qchisq`, `rchisq`
- Student t Distribution: `dt`, `pt`, `qt`, `rt`
- F Distribution: `df`, `pf`, `qf`, `rf`

In general, the functions are as follows:

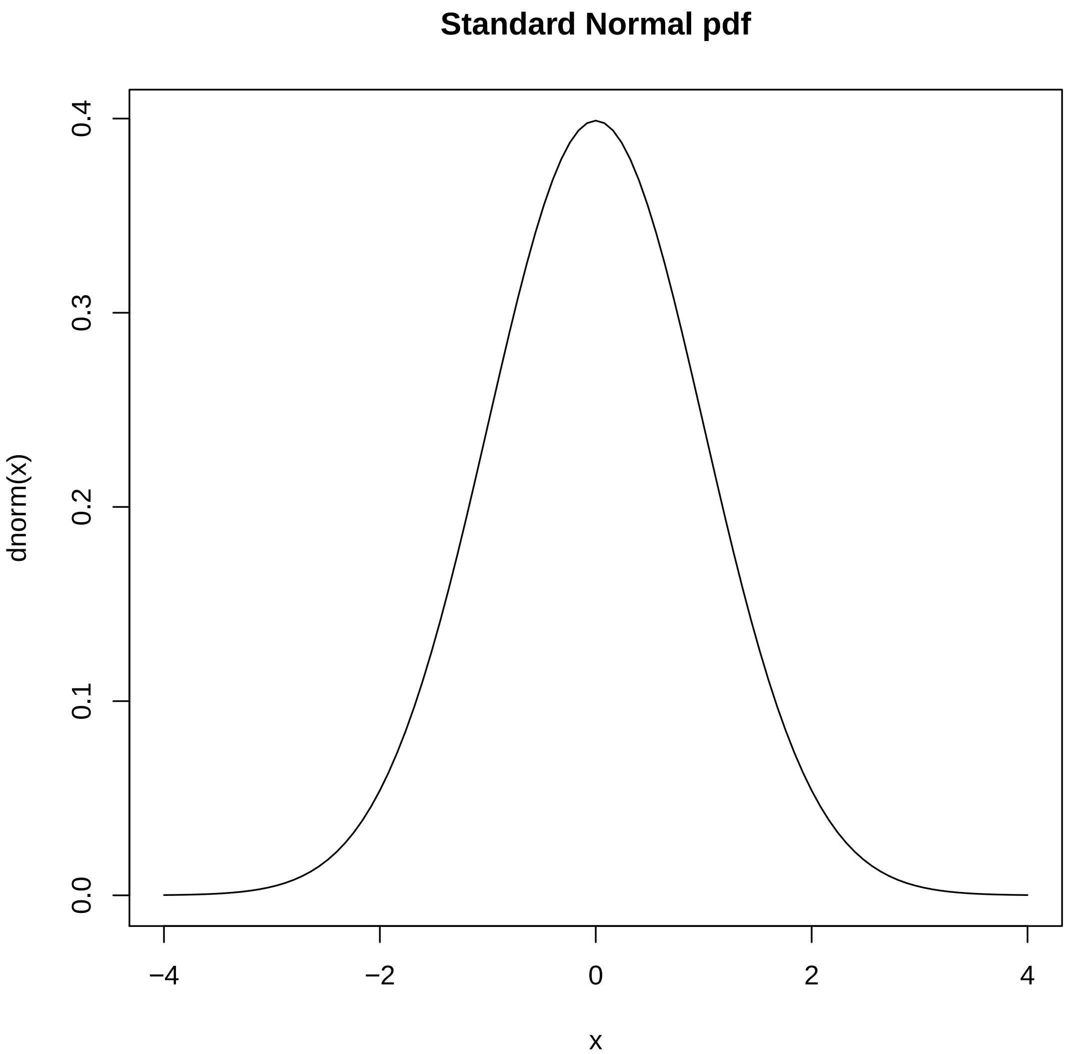
- d*: density
- p*: distribution function (probability)
- q*: quantile function
- r*: random generation

where * is the appropriate distribution.

Here we demonstrate these functions for the normal distribution.

Use dnorm to plot the pdf of a standard normal distribution

```r
curve(dnorm(x), from = -4, to = 4, main = "Standard Normal pdf")
```



Calculate the probability that $Y \leq 2$ when Y is distributed $N(1, 4)$ with mean of 1 and variance of 4.

```
pnorm(q = 2, mean = 1, sd = 2)

## [1] 0.6915

# Normalize as is done in the book
pnorm(q = 0.5)

## [1] 0.6915
```

Quantile function of a standard normal at probability 0.975.

```r
qnorm(p = 0.975)

## [1] 1.96
```

Generate 10 random numbers from a normal distribution with mean 0.0015 and standard deviation 0.025.

```r
# Set the seed for reproducible results
set.seed(123)
rnorm(n = 10, mean = 0.0015, sd = 0.025)

##  [1] -0.012512 -0.004254  0.040468  0.003263  0.004732  0.044377  0.013023
##  [8] -0.030127 -0.015671 -0.009642
```

1.3 Hypothesis Test

The null hypothesis is that the true mean return of SPY is equal to 0

```r
t.test(x = SPY.ret, alternative = "two.sided", mu = 0)

##
##  One Sample t-test
##
## data:  SPY.ret
## t = 1.338, df = 462, p-value = 0.1817
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.0007778  0.0040940
## sample estimates:
## mean of x
##  0.001658


# Replicate the results of t.test using the method outlined in the book
t_stat <- (mean(SPY.ret) - 0)/(sd(SPY.ret)/sqrt(nrow(SPY.ret)))
p_value <- 2 * pt(q = -abs(t_stat), df = 462)
df <- nrow(SPY.ret) - 1
ci <- mean(SPY.ret) + c(-1, 1) * 1.96 * sd(SPY.ret)/sqrt(nrow(SPY.ret))
```

```r
paste("t = ", round(t_stat, 4), ", df = ", df, ", p-value = ", round(p_value,
    4), sep = "")
```

```
## [1] "t = 1.3377, df = 462, p-value = 0.1817"
```

```r
print("95% Confidence Interval")
```

```
## [1] "95% Confidence Interval"
```

```r
print(ci)
```

```
## [1] -0.0007714  0.0040877
```

2 Regression

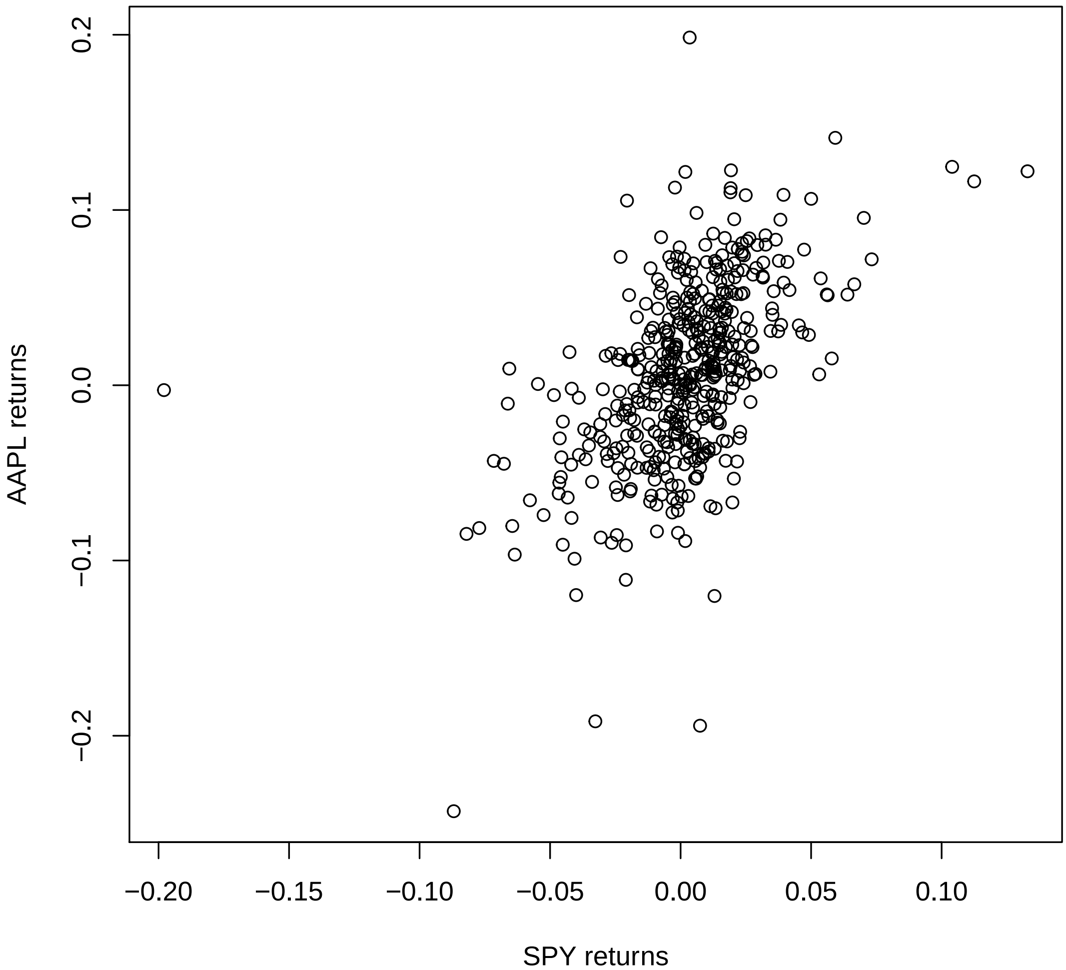
2.1 Regression with a single regressor

Extract the weekly returns of AAPL and SPY from the returns object. The returns of AAPL and SPY will be used to demonstrate linear regression in R.

```r
AAPL.ret <- returns[, "AAPL"]
SPY.ret <- returns[, "SPY"]


# Fitting linear models works with xts objects, but works better with
# data.frame objects. This is especially true with the predict method for
# linear models.
ret.data <- as.data.frame(cbind(AAPL.ret, SPY.ret))
```

Scatterplot of AAPL and SPY returns.

```r
plot(x = ret.data[, "SPY"], y = ret.data[, "AAPL"], xlab = "SPY returns", ylab = "AAPL returns")
```



Fit the linear regression model. `AAPL.ret` is the response variable and `SPY.ret` is the explanatory variable.

```
model.fit <- lm(AAPL ~ SPY, data = ret.data)
```

The `print` and `summary` methods for `lm` objects are very useful and provide several of the statistics covered in the book.

```
# The print method displays the call and the coefficients of the linear
# model
print(model.fit)

##
```

```
## Call:
## lm(formula = AAPL ~ SPY, data = ret.data)
##
## Coefficients:
## (Intercept)          SPY
##     0.00557      1.01539
```

```
# The summary method displays additional information for the linear model
model.summary <- summary(model.fit)
print(model.summary)
```

```
##
## Call:
## lm(formula = AAPL ~ SPY, data = ret.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.20741 -0.02582 -0.00002  0.02736  0.19262
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.00557    0.00202    2.76    0.006 **
## SPY          1.01539    0.07565   13.42   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0434 on 461 degrees of freedom
## Multiple R-squared:  0.281,Adjusted R-squared:  0.279
## F-statistic:  180 on 1 and 461 DF,  p-value: <2e-16
```

Access elements of the `lm` object

```
# Coefficients
coef(model.fit)
```

```
## (Intercept)         SPY
```

```
##    0.005573    1.015387

# Extract the fitted values fitted(model.fit) Extract the residuals
# resid(model.fit) Exctract the standardized residuals
# rstandard(model.fit)
```

Access elements of the `lm.summary` object

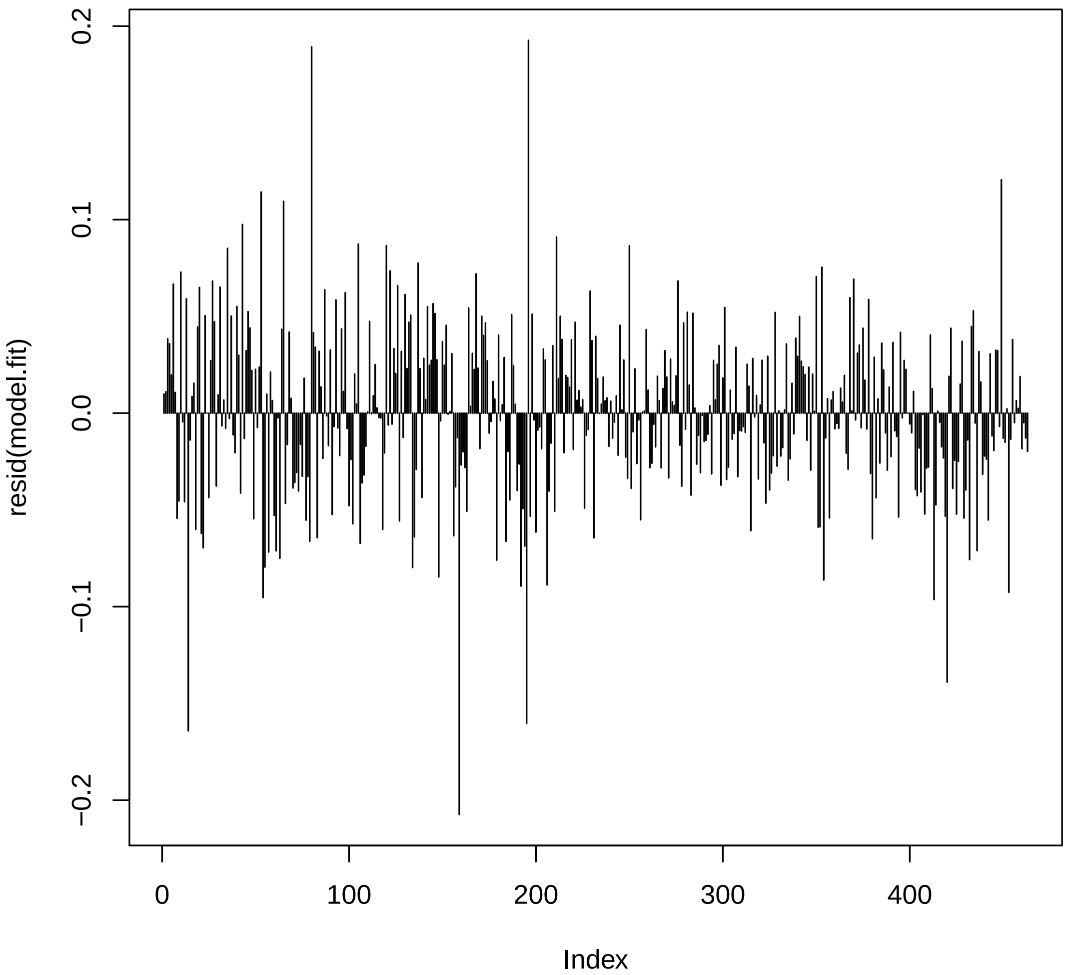
```
# Coefficients
coef(model.summary)

##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 0.005573   0.002019    2.76 6.017e-03
## SPY         1.015387   0.075648   13.42 6.607e-35

# Sigma
model.summary$sigma

## [1] 0.04337

# R squared
model.summary$r.squared

## [1] 0.281

# Adjusted R squared
model.summary$adj.r.squared

## [1] 0.2794
```

Use the `predict` method to calculate the confidence and prediction intervals of the fitted model.

```
new <- data.frame(SPY = seq(from = -0.2, to = 0.2, length.out = nrow(ret.data)))
model.ci <- predict(object = model.fit, newdata = new, interval = "confidence")
model.pi <- predict(object = model.fit, newdata = new, interval = "prediction")
```

Plot the residuals of the model.

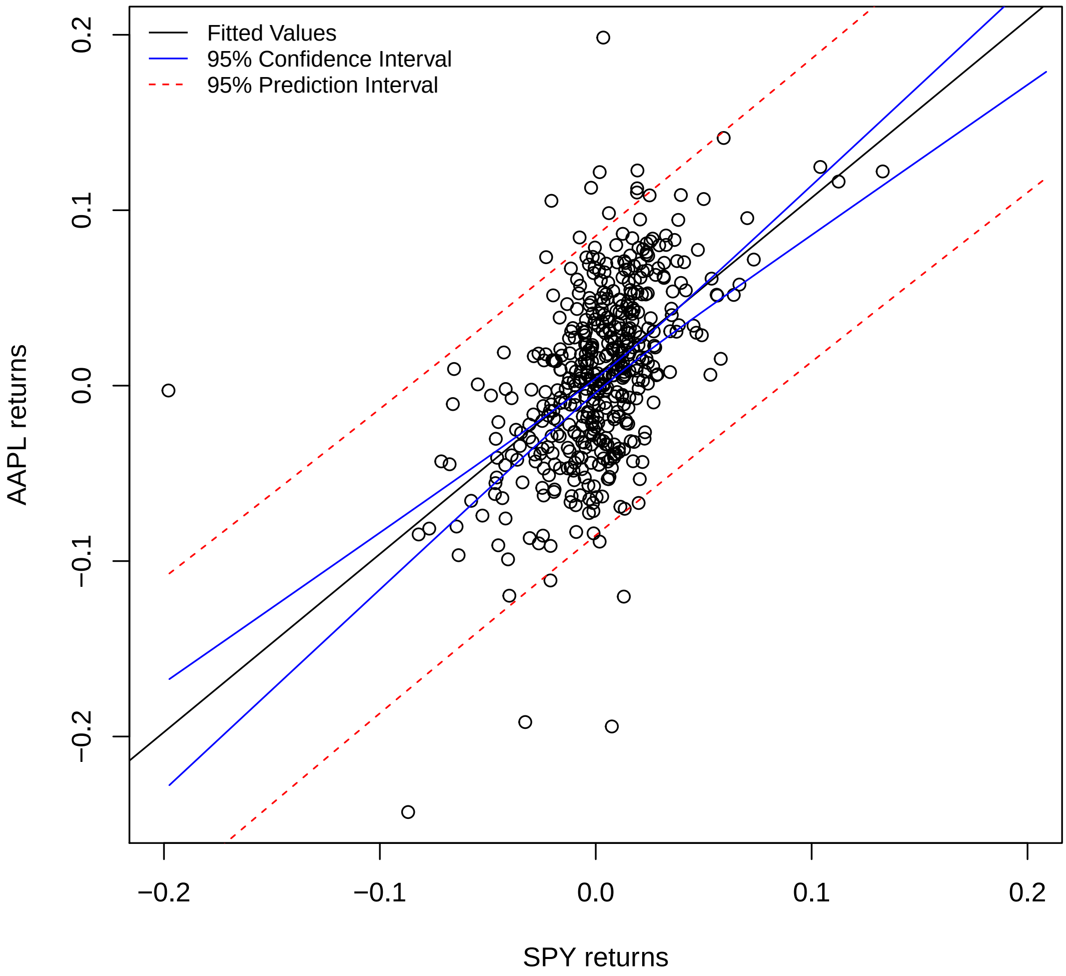
```r
plot(resid(model.fit), type = "h")
```



Plot the fitted model with the confidence and prediction intervals.

```r
plot(x=coredata(SPY.ret), y=coredata(AAPL.ret),
     xlab="SPY returns", ylab="AAPL returns", xlim=c(-0.2, 0.2))
abline(model.fit, col="black")
lines(x=model.ci[, "fit"], y=model.ci[, "upr"], col="blue", lty=1)
lines(x=model.ci[, "fit"], y=model.ci[, "lwr"], col="blue", lty=1)
lines(x=model.pi[, "fit"], y=model.pi[, "upr"], col="red", lty=2)
lines(x=model.pi[, "fit"], y=model.pi[, "lwr"], col="red", lty=2)
legend("topleft", legend=c("Fitted Values",
```

```
                    "95% Confidence Interval",
                    "95% Prediction Interval"),
       col=c("black", "blue", "red"), lty=c(1, 1, 2), cex=0.8, bty="n")
```



2.2 Regression with multiple regressors

The Fama French 3 Factor model is used to demonstrate regression with multiple regressors. The first example will use AAPL weekly returns and the Fama French factors from 2005-01-14 to 2013-10-25. The premise of the model is that AAPL returns can be explained by the 3 factors of the Fama French model.

```r
data(fama_french_factors)

# The first 3 columns are the factors, the 4th column is the risk free
# rate.
ff_factors <- fama_french_factors[, 1:3]
```

Prepare the data for the model.

```r
# Align the dates of the Fama-French Factors and the returns
returns <- returns["/2013-10-25"]
AAPL.ret <- returns[, "AAPL"]

# AAPL excess returns
AAPL.e <- AAPL.ret - fama_french_factors[, "RF"]/100
```

Fit the model.

```r
ff.fit <- lm(AAPL.e ~ ff_factors)
print(ff.fit)

##
## Call:
## lm(formula = AAPL.e ~ ff_factors)
##
## Coefficients:
##      (Intercept)  ff_factorsMkt-RF     ff_factorsSMB     ff_factorsHML
##          0.00540           0.01145           0.00333          -0.00640

print(summary(ff.fit))

##
## Call:
## lm(formula = AAPL.e ~ ff_factors)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.19963 -0.02661 -0.00039  0.02387  0.20464
##
```

```
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.00540    0.00197    2.74   0.0065 **
## ff_factorsMkt-RF   0.01145    0.00087   13.16   <2e-16 ***
## ff_factorsSMB      0.00333    0.00186    1.79   0.0735 .
## ff_factorsHML     -0.00640    0.00171   -3.75   0.0002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0422 on 455 degrees of freedom
## Multiple R-squared:  0.327,Adjusted R-squared:  0.322
## F-statistic: 73.6 on 3 and 455 DF,  p-value: <2e-16
```

If we wanted to fit the model to more assets, we could manually fit the model with different assets as the response variable. However, we can automatically fit several models very easily with R.

```
# Omit the first column of returns because it is the SPY weekly returns,
# which is a proxy for the market.
returns <- returns[, -1]

# Calculate the excess returns of all assets in the returns object
ret.e <- returns - (fama_french_factors[, "RF"]/100) %*% rep(1, ncol(returns))
```

The `ret.e` object contains the excess returns for AAPL, XOM, GOOG, MSFT, and GE.

```
# Show the first 5 rows of ret.e
head(ret.e, 5)
```

```
##                AAPL      XOM     GOOG     MSFT       GE
## 2005-01-14 0.013340  0.02512  0.03117 -0.02073 -0.01400
## 2005-01-21 0.003725 -0.01260 -0.05886 -0.01838 -0.01112
## 2005-01-28 0.049189  0.01607  0.01054  0.02026  0.01702
## 2005-02-04 0.065298  0.07799  0.07326  0.00466  0.01368
## 2005-02-11 0.029506  0.01926 -0.08339 -0.01367 -0.00115
```

Here we fit the Fama French 3 Factor model to each asset in `ret.e`. This fits 5 models, 1 for each asset, and stores results of each model in the `ff.fit` object as a multiple linear model (mlm) object.

```
ff.fit <- lm(ret.e ~ ff_factors)
# Display the coefficients of each model
print(ff.fit)
```

```
##
## Call:
## lm(formula = ret.e ~ ff_factors)
##
## Coefficients:
##                   AAPL       XOM        GOOG       MSFT       GE
## (Intercept)        0.005401   0.000809   0.002845   0.000300  -0.001109
## ff_factorsMkt-RF   0.011450   0.008784   0.011521   0.009724   0.009792
## ff_factorsSMB      0.003334  -0.004706  -0.000259  -0.003170   0.001164
## ff_factorsHML     -0.006396  -0.002046  -0.006467  -0.007470   0.008276
```

```
# Display the summary object for each model
print(summary(ff.fit))
```

```
## Response AAPL :
##
## Call:
## lm(formula = AAPL ~ ff_factors)
##
## Residuals:
##             AAPL
## Min    -0.199630
## 1Q     -0.026608
## Median -0.000385
## 3Q      0.023870
## Max     0.204637
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.00540    0.00197    2.74   0.0065 **
```

```
## ff_factorsMkt-RF  0.01145    0.00087   13.16   <2e-16 ***
## ff_factorsSMB     0.00333    0.00186    1.79   0.0735 .
## ff_factorsHML    -0.00640    0.00171   -3.75   0.0002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0422 on 455 degrees of freedom
## Multiple R-squared:  0.327,Adjusted R-squared:  0.322
## F-statistic: 73.6 on 3 and 455 DF,  p-value: <2e-16
##
##
## Response XOM :
##
## Call:
## lm(formula = XOM ~ ff_factors)
##
## Residuals:
##              XOM
## Min    -0.071559
## 1Q     -0.011856
## Median -0.000349
## 3Q      0.012376
## Max     0.092450
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.000809   0.001016    0.80     0.43
## ff_factorsMkt-RF  0.008784   0.000448   19.61  < 2e-16 ***
## ff_factorsSMB    -0.004706   0.000957   -4.92  1.2e-06 ***
## ff_factorsHML    -0.002046   0.000878   -2.33     0.02 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0217 on 455 degrees of freedom
```

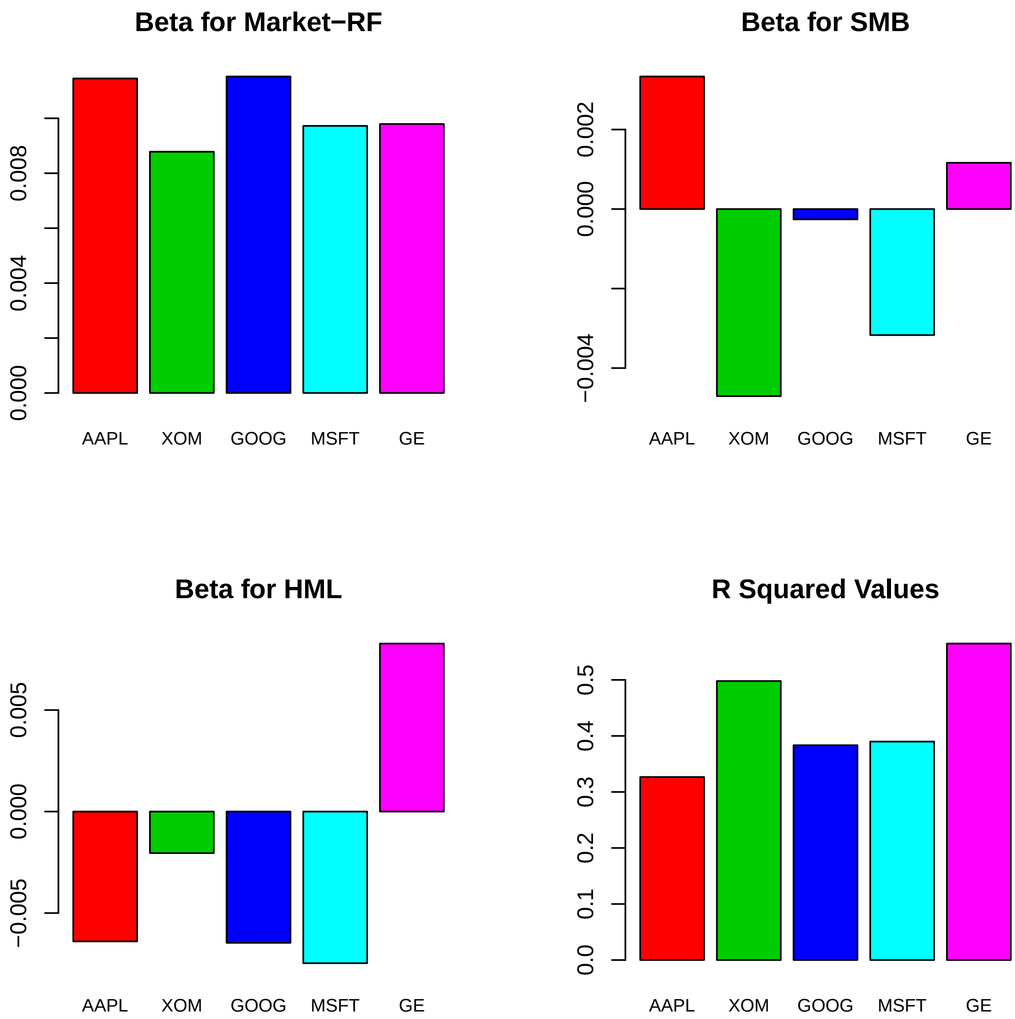
```
## Multiple R-squared:  0.498,Adjusted R-squared:  0.495
## F-statistic:  150 on 3 and 455 DF,  p-value: <2e-16
##
##
## Response GOOG :
##
## Call:
## lm(formula = GOOG ~ ff_factors)
##
## Residuals:
##             GOOG
## Min    -0.13693
## 1Q     -0.01741
## Median -0.00232
## 3Q      0.01527
## Max     0.15804
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.002845   0.001668    1.71    0.089 .
## ff_factorsMkt-RF  0.011521   0.000735   15.68  < 2e-16 ***
## ff_factorsSMB    -0.000259   0.001570   -0.17    0.869
## ff_factorsHML    -0.006467   0.001441   -4.49  9.1e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0357 on 455 degrees of freedom
## Multiple R-squared:  0.383,Adjusted R-squared:  0.379
## F-statistic: 94.3 on 3 and 455 DF,  p-value: <2e-16
##
##
## Response MSFT :
##
## Call:
```

```
## lm(formula = MSFT ~ ff_factors)
##
## Residuals:
##             MSFT
## Min    -0.125627
## 1Q     -0.013886
## Median -0.000495
## 3Q      0.012635
## Max     0.137178
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.000300   0.001319    0.23    0.820
## ff_factorsMkt-RF  0.009724   0.000581   16.73  < 2e-16 ***
## ff_factorsSMB    -0.003170   0.001241   -2.55    0.011 *
## ff_factorsHML    -0.007470   0.001140   -6.56  1.5e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0282 on 455 degrees of freedom
## Multiple R-squared:  0.39,Adjusted R-squared:  0.386
## F-statistic: 96.9 on 3 and 455 DF,  p-value: <2e-16
##
##
## Response GE :
##
## Call:
## lm(formula = GE ~ ff_factors)
##
## Residuals:
##              GE
## Min    -0.13291
## 1Q     -0.01460
## Median -0.00156
```

```
## 3Q      0.01266
## Max     0.21088
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -0.001109   0.001366   -0.81     0.42
## ff_factorsMkt-RF  0.009792   0.000602   16.27   < 2e-16 ***
## ff_factorsSMB     0.001164   0.001285    0.91     0.37
## ff_factorsHML     0.008276   0.001180    7.01   8.4e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0292 on 455 degrees of freedom
## Multiple R-squared:  0.565,Adjusted R-squared:  0.562
## F-statistic:  197 on 3 and 455 DF,  p-value: <2e-16
```

Extract and plot the beta values and the R squared values for each asset.

```
beta0 <- coef(ff.fit)[1, ]
beta1 <- coef(ff.fit)[2, ]
beta2 <- coef(ff.fit)[3, ]
beta3 <- coef(ff.fit)[4, ]
rsq <- sapply(X = summary(ff.fit), FUN = function(x) x$r.squared)
names(rsq) <- colnames(ret.e)

par(mfrow = c(2, 2))
barplot(beta1, main = "Beta for Market-RF", col = c(2:6), cex.names = 0.8)
barplot(beta2, main = "Beta for SMB", col = c(2:6), cex.names = 0.8)
barplot(beta3, main = "Beta for HML", col = c(2:6), cex.names = 0.8)
barplot(rsq, main = "R Squared Values", col = c(2:6), cex.names = 0.8)
```




```
par(mfrow = c(1, 1))
```