Quantitative Analysis

Ross Bennett

December 2, 2013

Abstract

The goal of this vignette is to demonstrate key concepts in Financial Risk Manager (FRM ®) 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

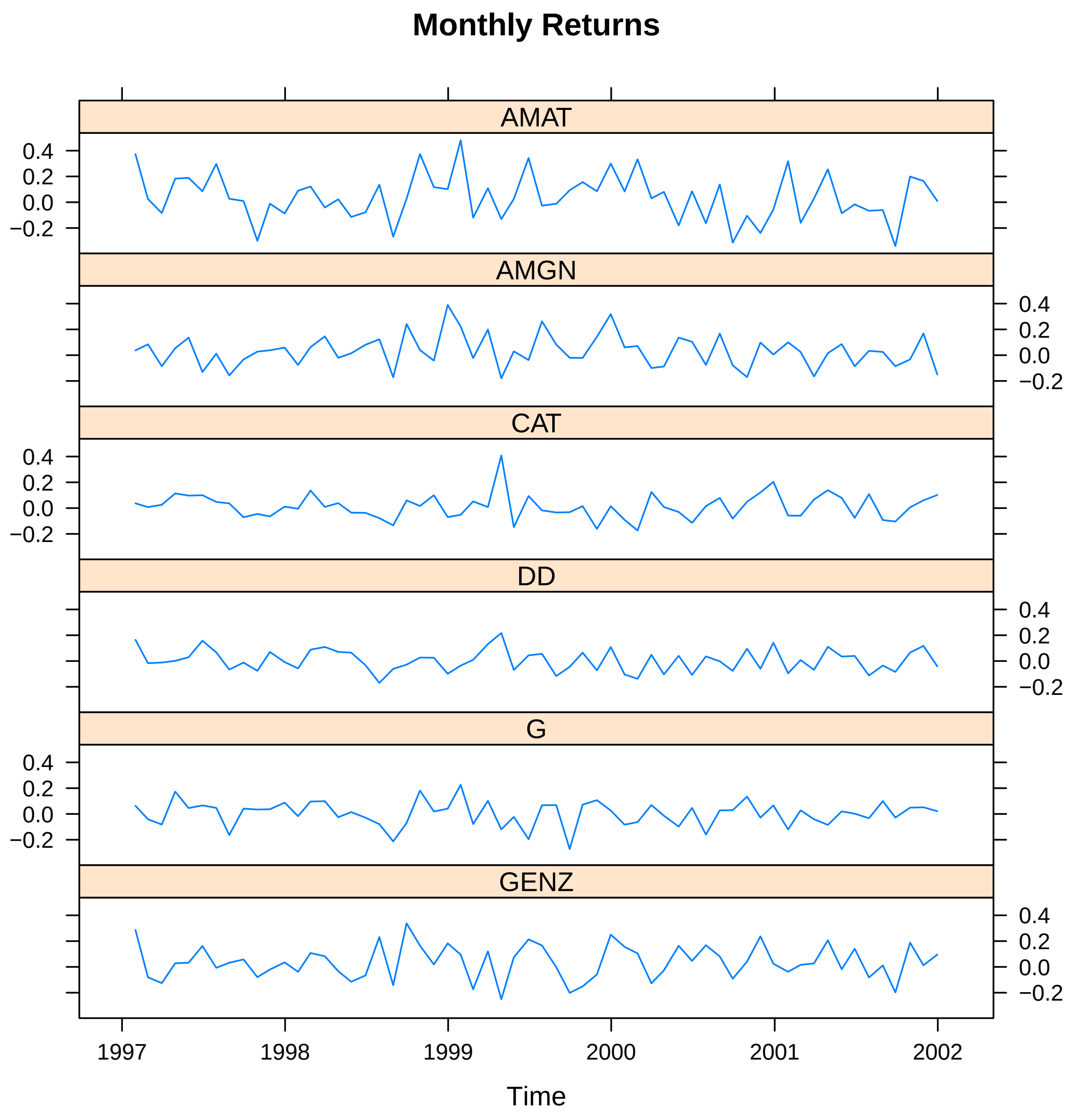
1 Exploratory Data Analysis

Load the GARPFRM package and the `crsp.short` dataset. Other packages are also loaded for plotting functions. The `crsp.short` dataset contains monthly returns from 1997-01-31 to 2001-12-31 of stocks in micro, small, mid, and large market capitalizations as well as market returns and the risk free rate. The market is defined as the value weighted NYSE and NYSE Amex, the latter formerly being the American Stock Exchange and the NASDAQ composite. The risk free rate comes from the 90 day Treasury Bill.

```r
suppressMessages(library(GARPFRM))
suppressMessages(library(lattice))
suppressMessages(library(pcaPP))
suppressMessages(library(hexbin))
data(crsp.short)

# Use the first 6 stocks in largecap.ts
crsp.returns <- largecap.ts[, 1:6]

# Get the names of the stocks and view the first 5 rows of crsp.returns
colnames(crsp.returns)

## [1] "AMAT" "AMGN" "CAT"  "DD"   "G"    "GENZ"

head(crsp.returns, 5)

##                AMAT     AMGN      CAT        DD        G     GENZ
## 1997-01-31  0.37391  0.03678 0.036877  0.164675  0.06341  0.28736
## 1997-02-28  0.02532  0.08426 0.008052 -0.016465 -0.04091 -0.08036
## 1997-03-31 -0.08395 -0.08589 0.025559 -0.011655 -0.08215 -0.12621
## 1997-04-30  0.18329  0.05369 0.114019  0.001179  0.17336  0.02778
## 1997-05-30  0.18907  0.13588 0.096910  0.029494  0.04559  0.03243
```

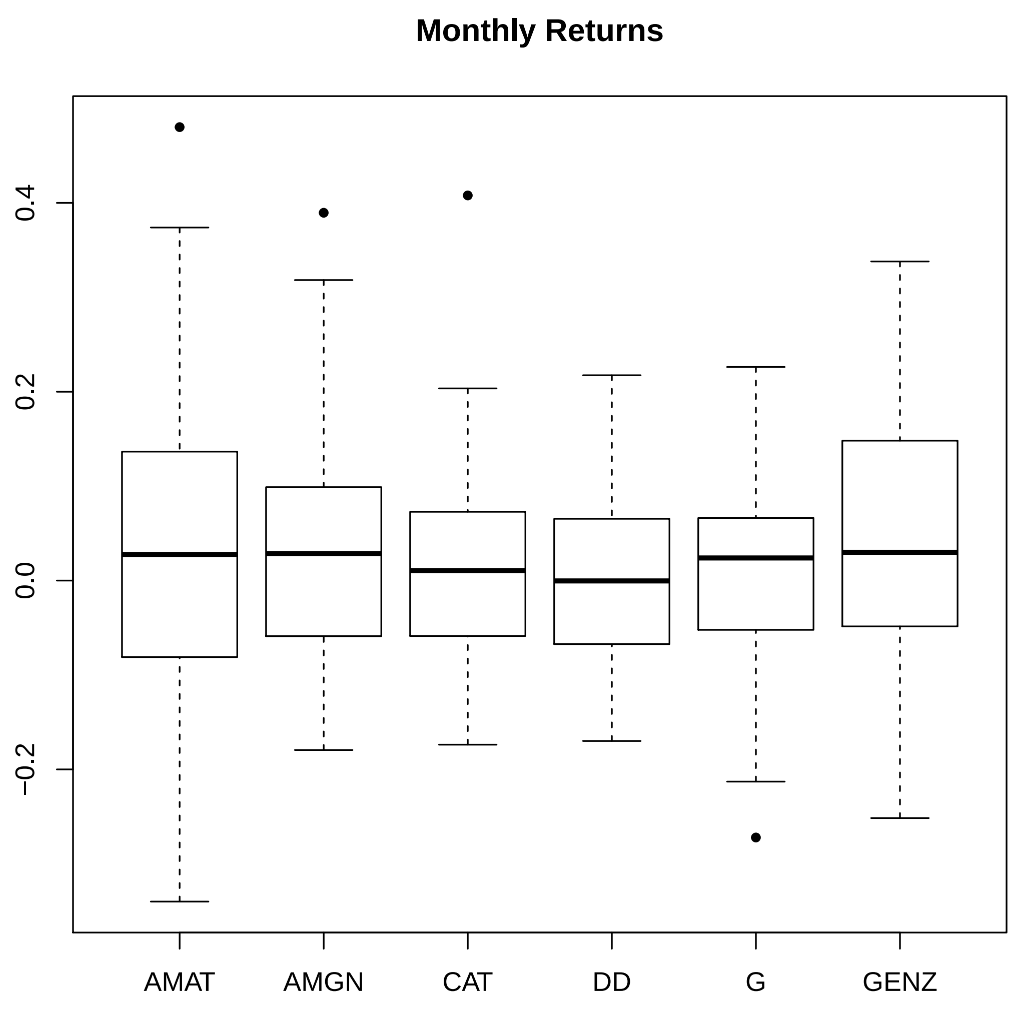
Plot of the returns of each asset.

```r
xyplot(crsp.returns, scale = list(y = "same"), main = "Monthly Returns")
```



Another way to compare returns of several assets is with a boxplot.

```
boxplot(coredata(crsp.returns), pch = 20, main = "Monthly Returns")
```

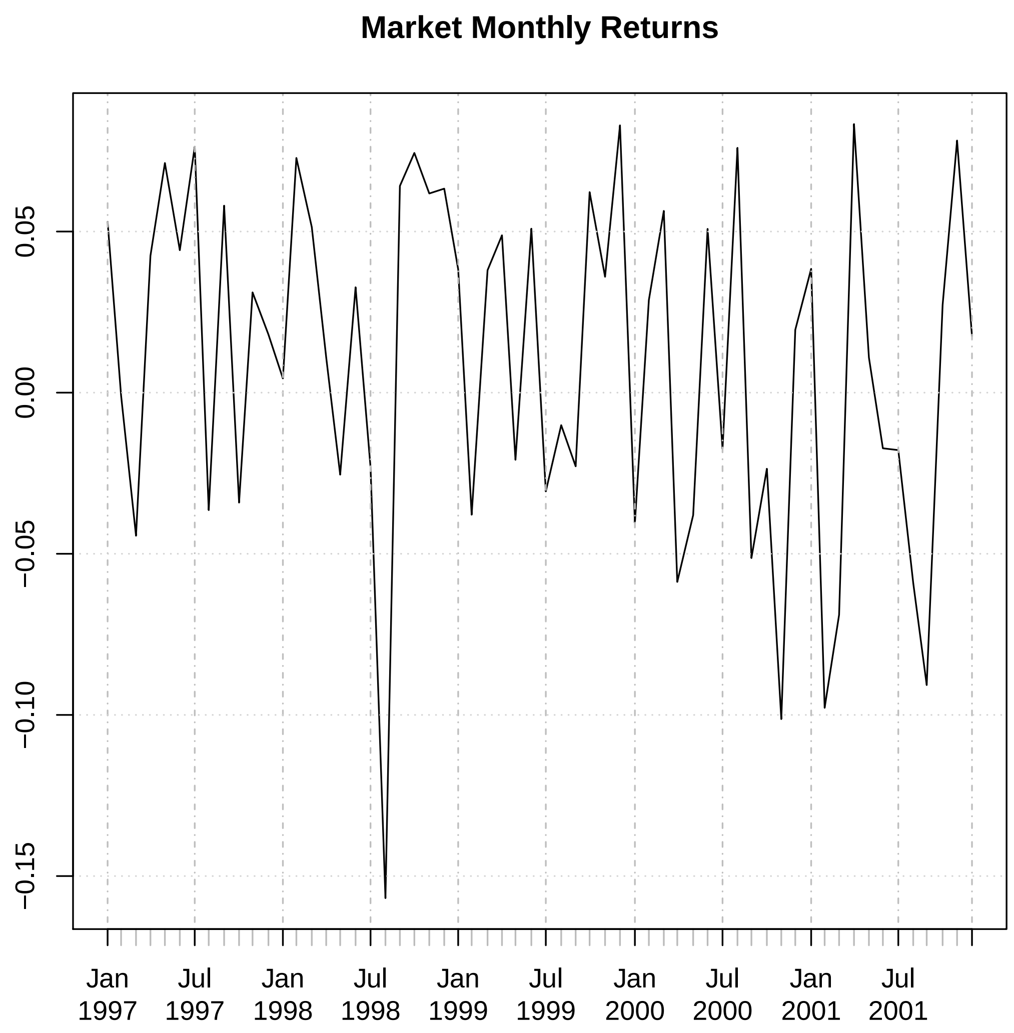


The exploratory data analysis, basic probability and statistics will use the market returns.

```r
# Extract the column labeled 'market' to get the market returns
MKT.ret <- largecap.ts[, "market"]
```

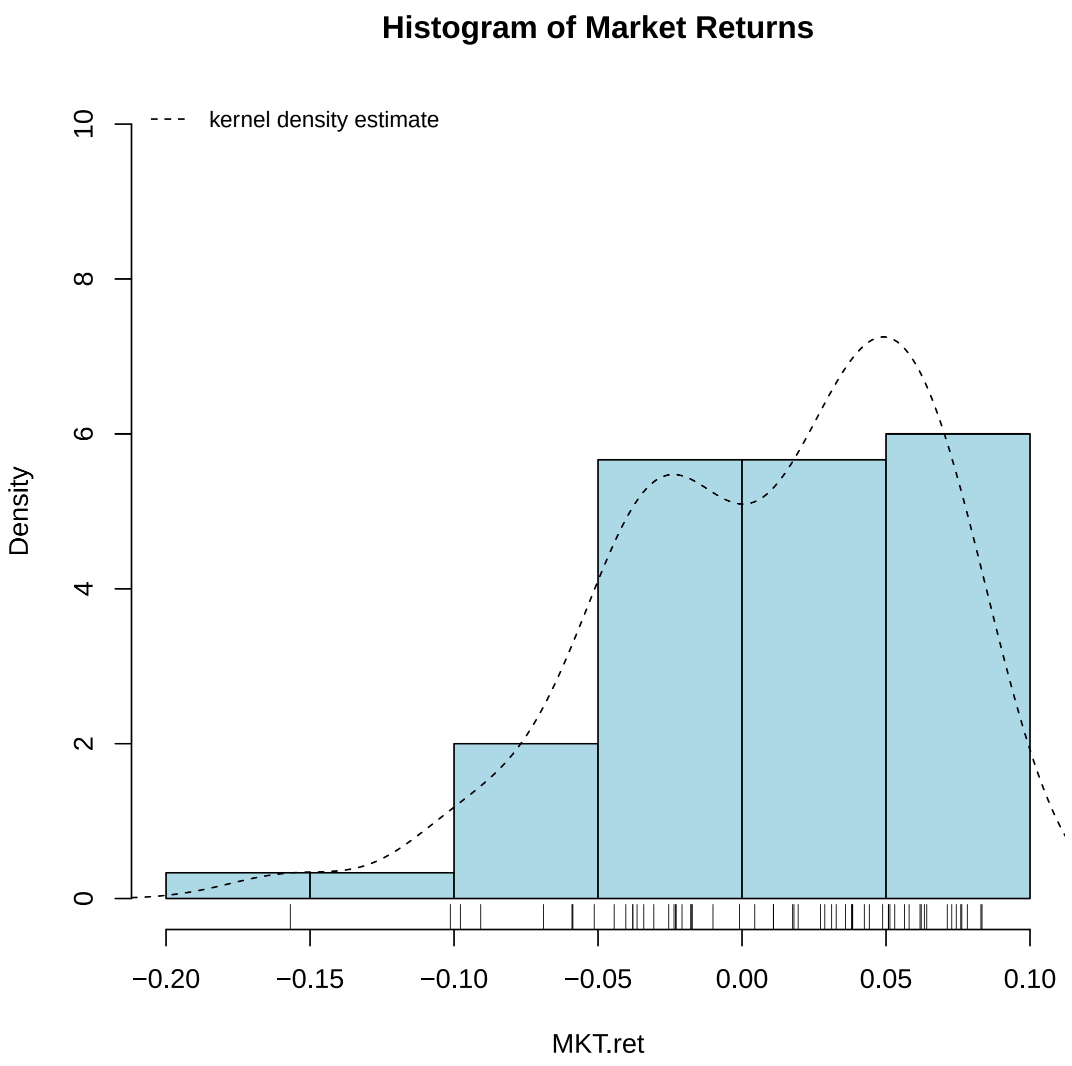
Plot of the MKT weekly returns.

```r
plot(MKT.ret, main = "Market Monthly Returns")
```



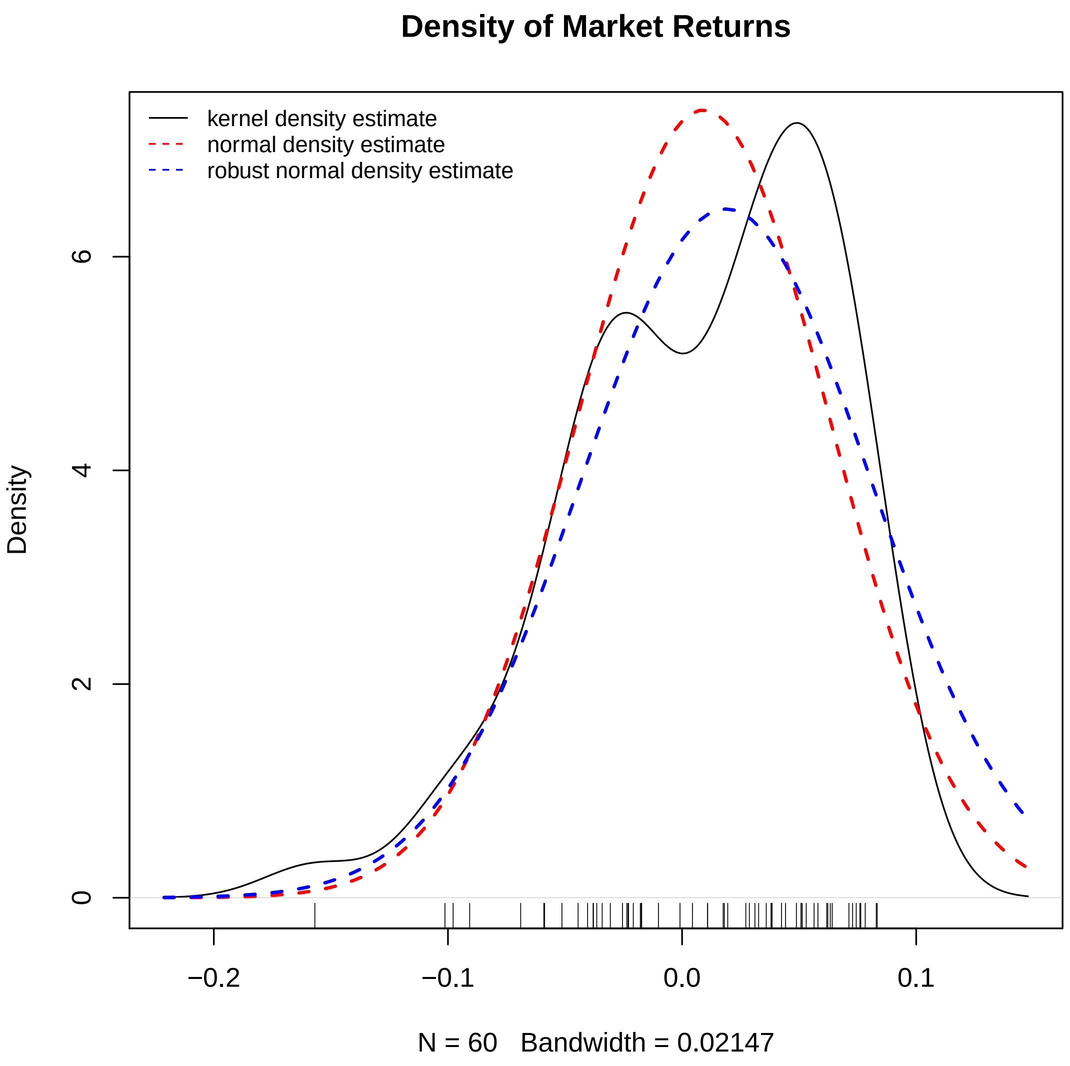
A histogram and kernel density estimate of the market returns is plotted to better understand its distribution.

```r
hist(MKT.ret, probability=TRUE, main="Histogram of Market Returns",
     col="lightblue", ylim=c(0, 10))
lines(density(MKT.ret), lty=2)
rug(MKT.ret)
legend("topleft", legend="kernel density estimate", lty=2,
       cex=0.8, bty="n")
```



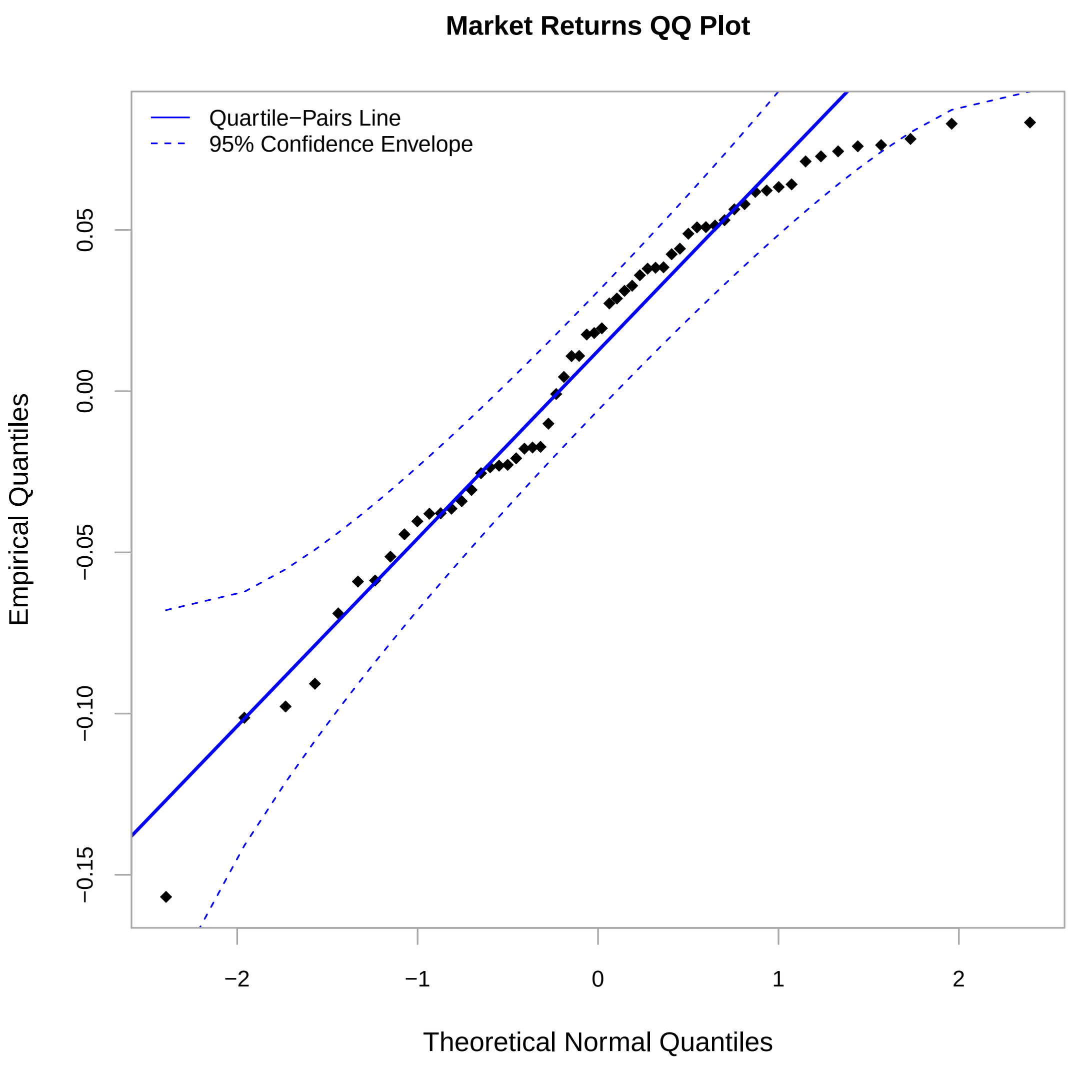
A normal density is overlayed on the plot of the kernel density estimate with standard estimates of the sample mean and standard deviation. Another normal density is overlayed using robust estimates. It is clear from the chart that neither the robust estimates nor the standard estimates of the sample mean and sample standard deviation provide a visually goot fit. It appears that the kernel density estimate of the market returns is bimodal.

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



Quantile-Quantile plot of market returns with a 95% confidence envelope. It can be seen from the Normal Q-Q plot that the tails of the market returns are well outside of the 95% confidence envelope.

```r
chart.QQPlot(MKT.ret, envelope=0.95, pch=18, main="Market Returns QQ Plot",
             xlab="Theoretical Normal Quantiles")
legend("topleft", legend=c("Quartile-Pairs Line", "95% Confidence Envelope"),
       col=c("blue", "blue"), lty=c(1, 2), cex=0.8, bty="n")
```



We can test if the market 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 less than 0.05 and the null hypothesis can be rejected at a 95% confidence level.

```r
shapiro.test(coredata(MKT.ret))

##
```

```
##  Shapiro-Wilk normality test
##
## data:  coredata(MKT.ret)
## W = 0.941, p-value = 0.005999
```

1.1 Basic Statistics

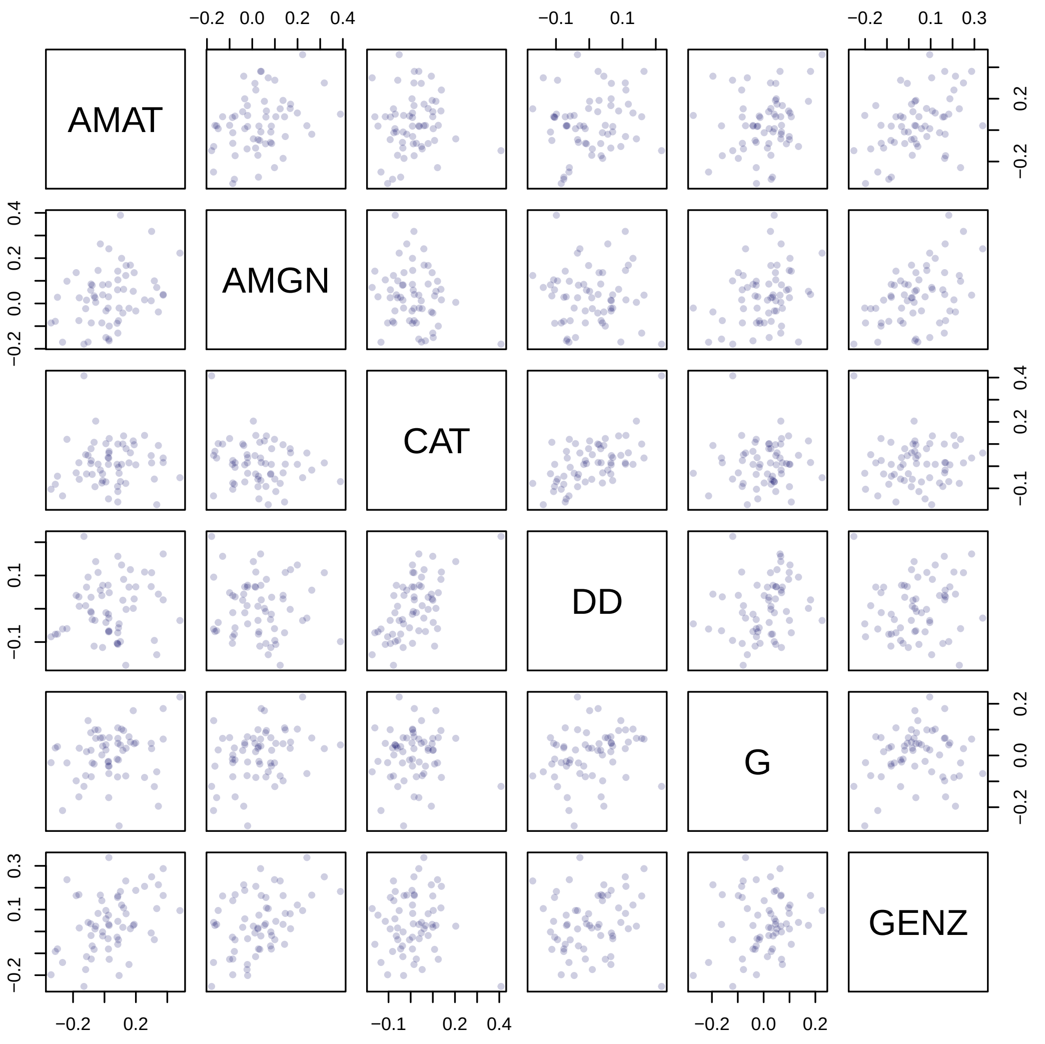
Here we calculate some basic statisitics on the market returns.

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

## [1] 0.009145


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

##          market
## market 0.002928


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

## [1] 0.05411


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

## [1] 0.006985


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

## [1] -0.7359
```

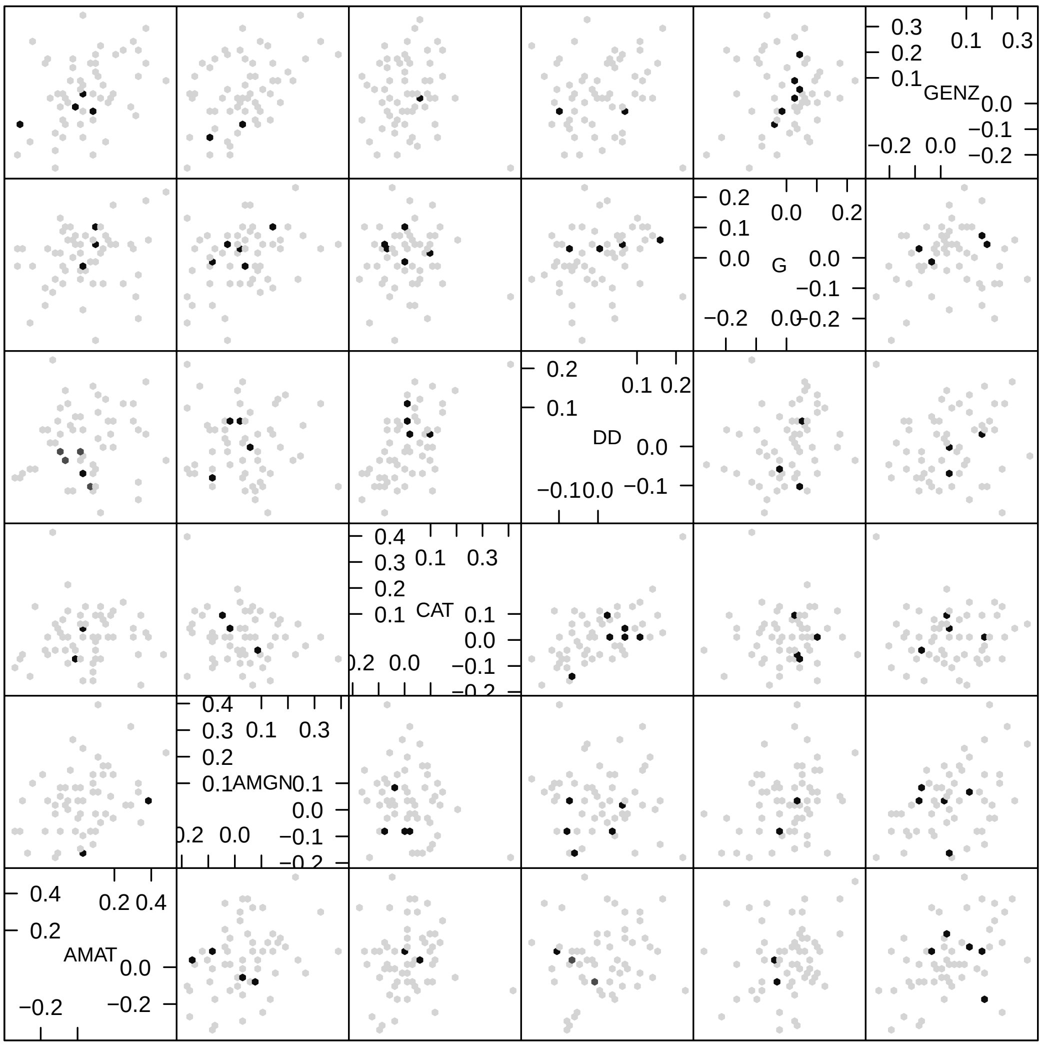
```
# Sample kurtosis of SPY returns.  See ?kurtosis for additional methods
# for calculating kurtosis
kurtosis(MKT.ret, method = "sample")

## [1] 3.309


# Summary statistics of SPY returns
summary(MKT.ret)

##      Index                market
##  Min.   :1997-01-31   Min.   :-0.15685
##  1st Qu.:1998-04-22   1st Qu.:-0.02675
##  Median :1999-07-15   Median : 0.01877
##  Mean   :1999-07-15   Mean   : 0.00914
##  3rd Qu.:2000-10-07   3rd Qu.: 0.05178
##  Max.   :2001-12-31   Max.   : 0.08335


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

##       0%      25%      50%      75%     100%
## -0.15685 -0.02675  0.01877  0.05178  0.08335
```

Scatter plot of each pair of assets. This can be usefule to visually look for relationships among the assets.

```
pairs(coredata(crsp.returns), pch=20, col=rgb(0,0,100,50,maxColorValue=255))
```



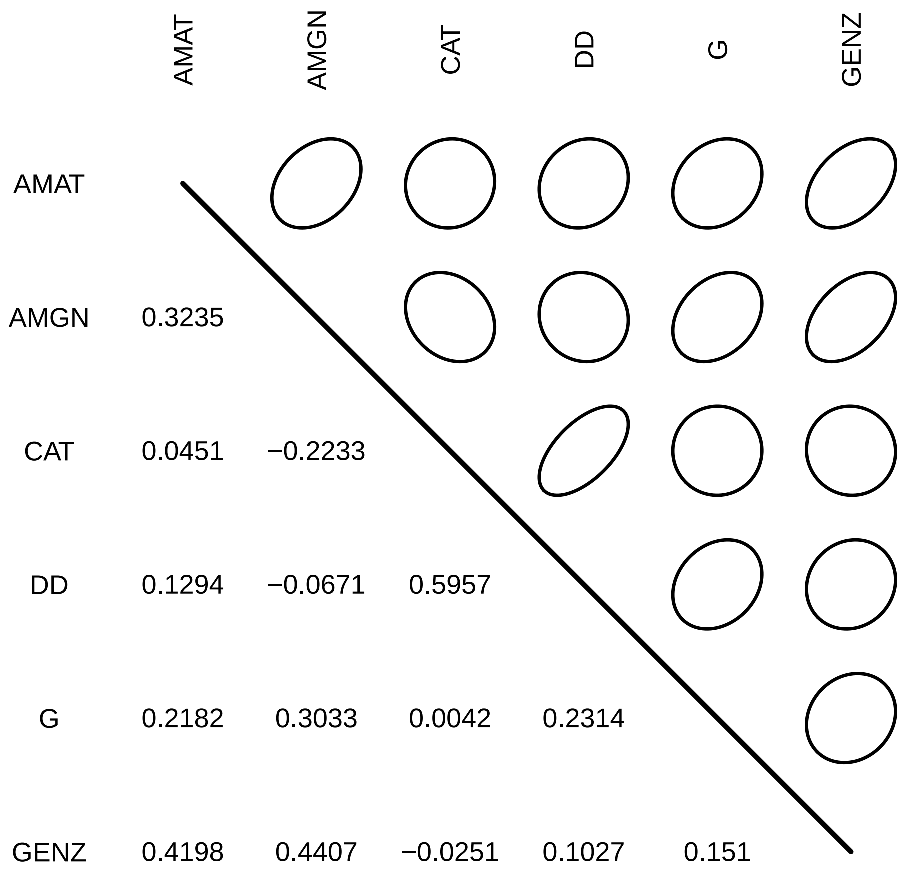
```
hexplom(coredata(crsp.returns), varname.cex=0.75)
```



Scatter Plot Matrix

Correlation and covariance matrices of the assets.

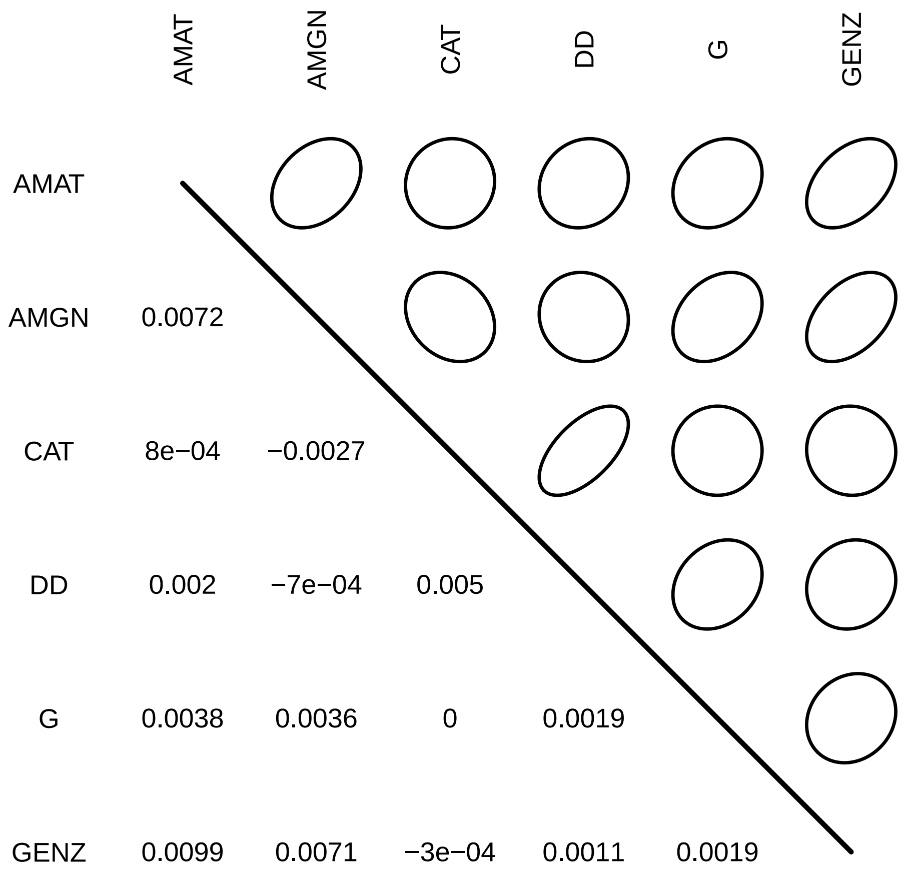
```
# Sample correlation of returns
cor(crsp.returns)

##         AMAT     AMGN       CAT       DD        G     GENZ
## AMAT 1.0000  0.32350  0.045105  0.12941 0.218250  0.41977
## AMGN 0.3235  1.00000 -0.223329 -0.06707 0.303333  0.44068
## CAT  0.0451 -0.22333  1.000000  0.59569 0.004176 -0.02505
## DD   0.1294 -0.06707  0.595690  1.00000 0.231427  0.10274
## G    0.2182  0.30333  0.004176  0.23143 1.000000  0.15096
## GENZ 0.4198  0.44068 -0.025055  0.10274 0.150958  1.00000
```



Sample Correlation Estimate

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

```
##           AMAT       AMGN        CAT         DD         G      GENZ
## AMAT 0.0322460  0.0071537  7.927e-04  0.0019791 3.761e-03  0.009867
## AMGN 0.0071537  0.0151647 -2.692e-03 -0.0007034 3.584e-03  0.007103
## CAT  0.0007927 -0.0026916  9.578e-03  0.0049652 3.921e-05 -0.000321
## DD   0.0019791 -0.0007034  4.965e-03  0.0072534 1.891e-03  0.001145
## G    0.0037607  0.0035843  3.921e-05  0.0018913 9.207e-03  0.001896
## GENZ 0.0098667  0.0071035 -3.210e-04  0.0011454 1.896e-03  0.017134
```



Sample Covariance Estimate

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:

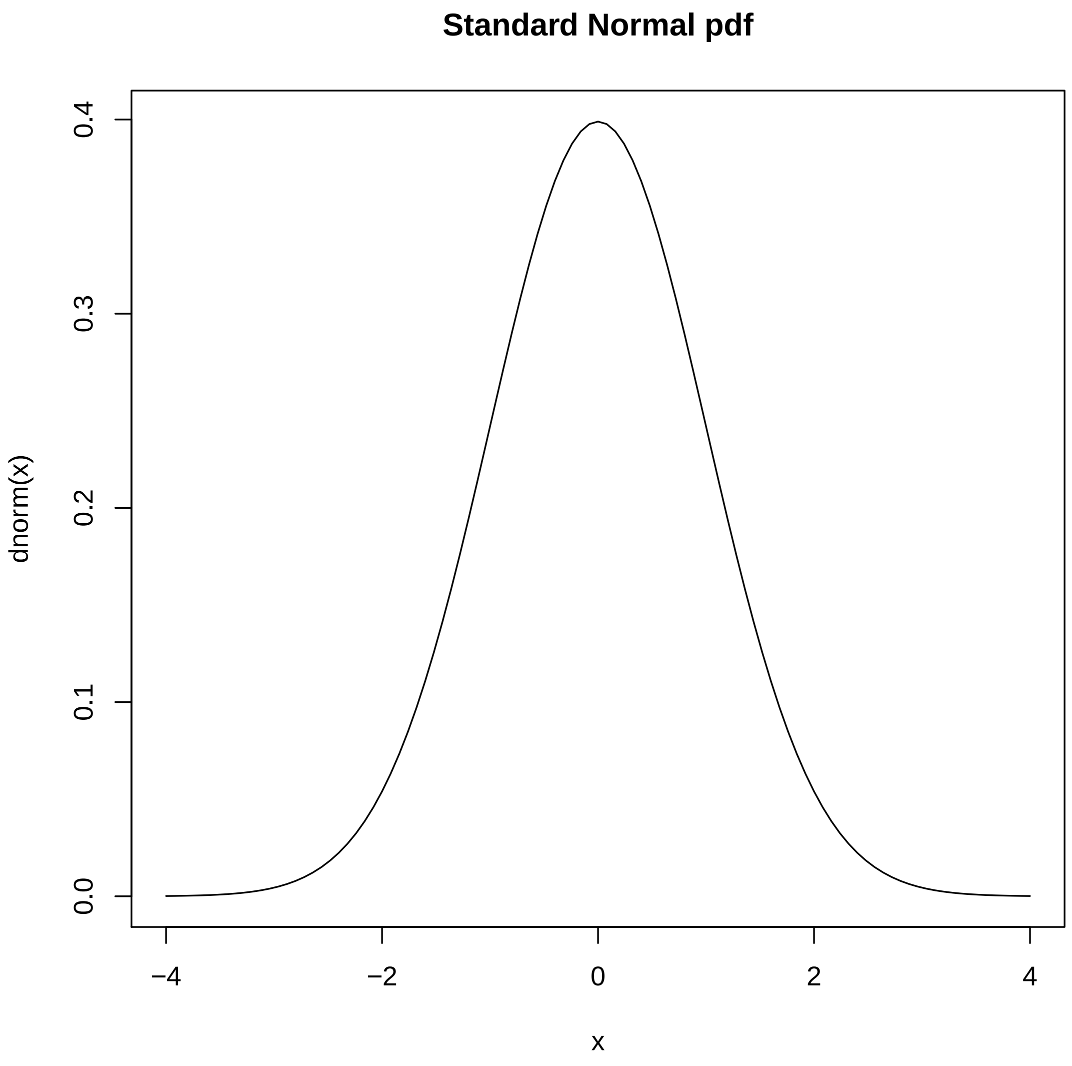
- d*: density
- p*: distribution function (probability)
- q*: quantile function
- r*: random generation

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

```
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 the standard normal distribution at probability 0.975.

```
qnorm(p = 0.975)

## [1] 1.96
```

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

```
# 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 the market is equal to 0.

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

##
##  One Sample t-test
##
## data:  MKT.ret
## t = 1.309, df = 59, p-value = 0.1956
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.004833  0.023122
## sample estimates:
## mean of x
##  0.009145
```

The p-value is greater than 0.05 so the null hypothesis cannot be rejected at a 95% confidence level.

```
# Replicate the results of t.test using the method outlined in the book
t_stat <- (mean(MKT.ret) - 0)/(sd(MKT.ret)/sqrt(nrow(MKT.ret)))
p_value <- 2 * pt(q = -abs(t_stat), df = nrow(MKT.ret))
df <- nrow(MKT.ret) - 1
ci <- mean(MKT.ret) + c(-1, 1) * 1.96 * sd(MKT.ret)/sqrt(nrow(MKT.ret))
paste("t = ", round(t_stat, 4), ", df = ", df, ", p-value = ", round(p_value,
    4), sep = "")

## [1] "t = 1.3091, df = 59, p-value = 0.1955"

print("95% Confidence Interval")

## [1] "95% Confidence Interval"

print(ci)

## [1] -0.004546  0.022836
```

Note that the confidence interval is different. This is because the `t.test` function calculates the exact confidence interval. Using a value of 1.96 is an approximation.

2 Regression

2.1 Regression with a single regressor

The general form of a linear model is:

$$Y_i = \beta_0 + \beta_i X_i + u_i \tag{1}$$

where:

Y_i is the dependent variable

X_i is the independent variable

β_0 is the intercept of the population regression line

β_1 is the slope of the population regression line

u_i is the error term

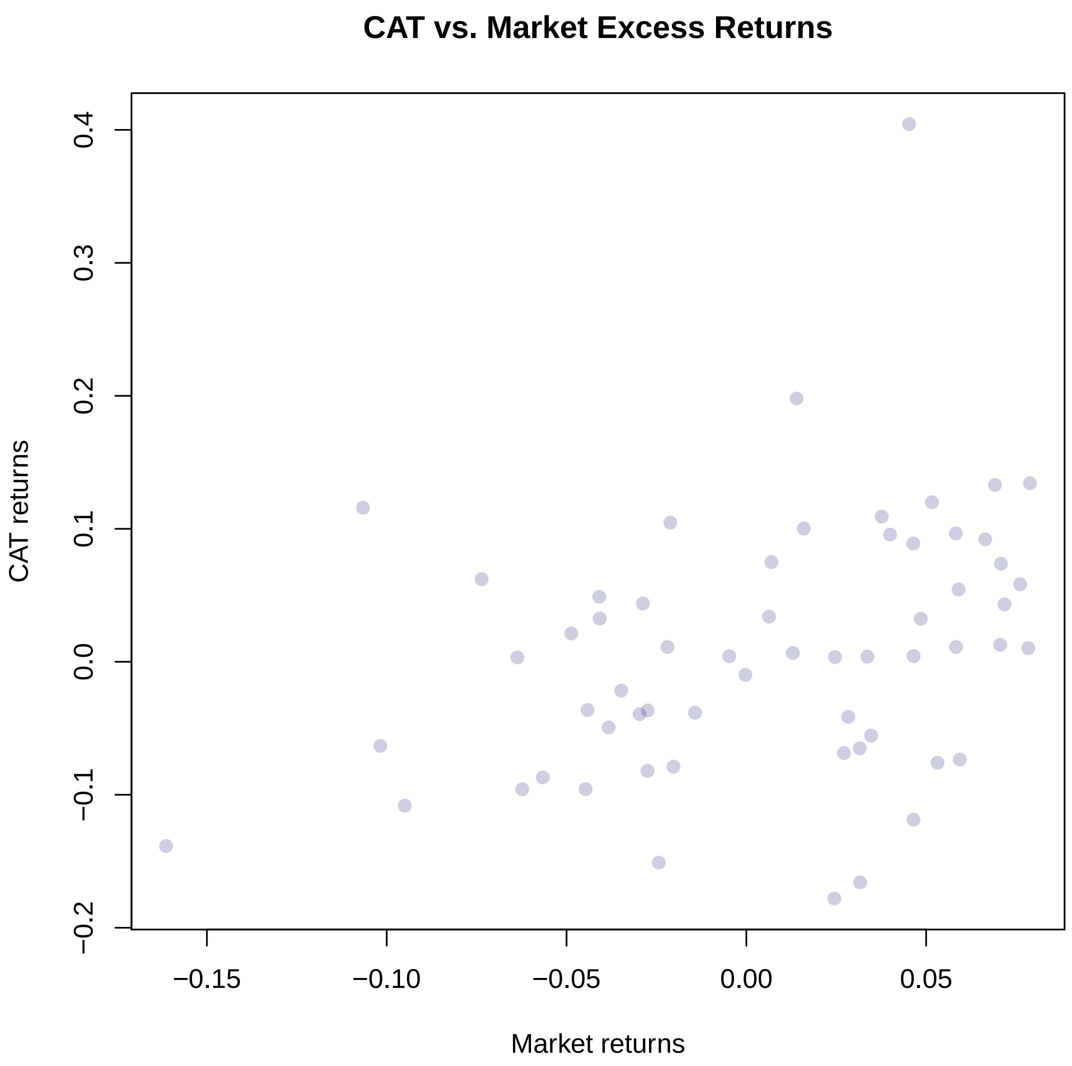
In the following linear model, CAT excess returns is the dependent variable and market excess returns is the independent variable. This model will be used to demonstrate linear regression in R.

Extract the weekly returns of CAT from the returns object.

```r
CAT.ret <- crsp.returns[, "CAT"] - largecap.ts[, "t90"]
MKT.ret <- largecap.ts[, "market"] - largecap.ts[, "t90"]

# 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(CAT.ret, MKT.ret))
```

Scatterplot of CAT and market excess returns.

```r
plot(coredata(MKT.ret), coredata(CAT.ret), pch=19,
     col=rgb(0,0,100,50,maxColorValue=255),
     xlab="Market returns", ylab="CAT returns",
     main="CAT vs. Market Excess Returns")
```



Fit the linear regression model. `CAT.ret` is the response variable and `MKT.ret` is the explanatory variable. That is, a linear model will be fit using market excesss returns to describe CAT excess returns.

```
model.fit <- lm(CAT ~ market, data = ret.data)
```

The `print` and `summary` methods for `lm` objects are very useful and provide several of the statistics covered in the book. Note that `summary(model.fit)` will print the summary statisitcs, but it is often useful to assign the summary object to a variable so that elements from the summary object can be extracted as shown below.

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

##
## Call:
## lm(formula = CAT ~ market, data = ret.data)
##
## Coefficients:
## (Intercept)       market
##     0.00489      0.60240


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

##
## Call:
## lm(formula = CAT ~ market, data = ret.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.1977 -0.0564 -0.0064  0.0561  0.3722
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.00489    0.01207    0.41   0.6867
## market       0.60240    0.22379    2.69   0.0093 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0931 on 58 degrees of freedom
## Multiple R-squared:  0.111,Adjusted R-squared:  0.0957
## F-statistic: 7.25 on 1 and 58 DF,  p-value: 0.00927
```

The results of the fitted model show that the equation for the linear model can be written as

$$C\hat{AT}.ret_i = 0.004892 + 0.602402 * MKT.ret_i + 0.09310604 \tag{2}$$

Examples of useful methods to access elements of the `lm` object

```r
# Note that some are commented out


# Coefficients
coef(model.fit)

## (Intercept)      market
##    0.004892    0.602402

# 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

```r
# Coefficients
model.coef <- coef(model.summary)
model.coef

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.004892    0.01207  0.4054 0.686706
## market      0.602402    0.22379  2.6918 0.009272


# Sigma
model.summary$sigma

## [1] 0.09311

# R squared
model.summary$r.squared

## [1] 0.1111

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

## [1] 0.09573
```

Accessing elements of the models can be ued to compute measures of fit. Some measures of fit, such as the R^2, adjusted R^2, and standard error of regression are computed by the `summary` function and just need to be extracted.

2.1.1 Measures of Fit

Explained sum of squares (ESS)

$$ESS = \sum_{i=1}^{n} (\hat{Y}_i - \bar{Y})^2 \tag{3}$$

```
ESS <- sum((fitted(model.fit) - mean(CAT.ret))^2)
```

Total sum of squares (TSS)

$$TSS = \sum_{i=1}^{n} (Y_i - \bar{Y})^2 \tag{4}$$

```
TSS <- sum((CAT.ret - mean(CAT.ret))^2)
```

Sum of squared residuals (SSR)

$$SSR = \sum_{i=1}^{n} (\hat{u}_i)^2 \tag{5}$$

```
SSR <- sum(resid(model.fit)^2)
```

The regression R^2 is the fraction of the sample variance of Y_i explained by X_i. The R^2 can be extracted from the `model.summary` object.

```
r_squared <- model.summary$r.squared
r_squared

## [1] 0.1111
```

The R^2 can also be computed using the ESS, TSS, and SSR.

```
ESS/TSS

## [1] 0.1111

1 - SSR/TSS

## [1] 0.1111
```

The standard error of the regression (SER) is computed as

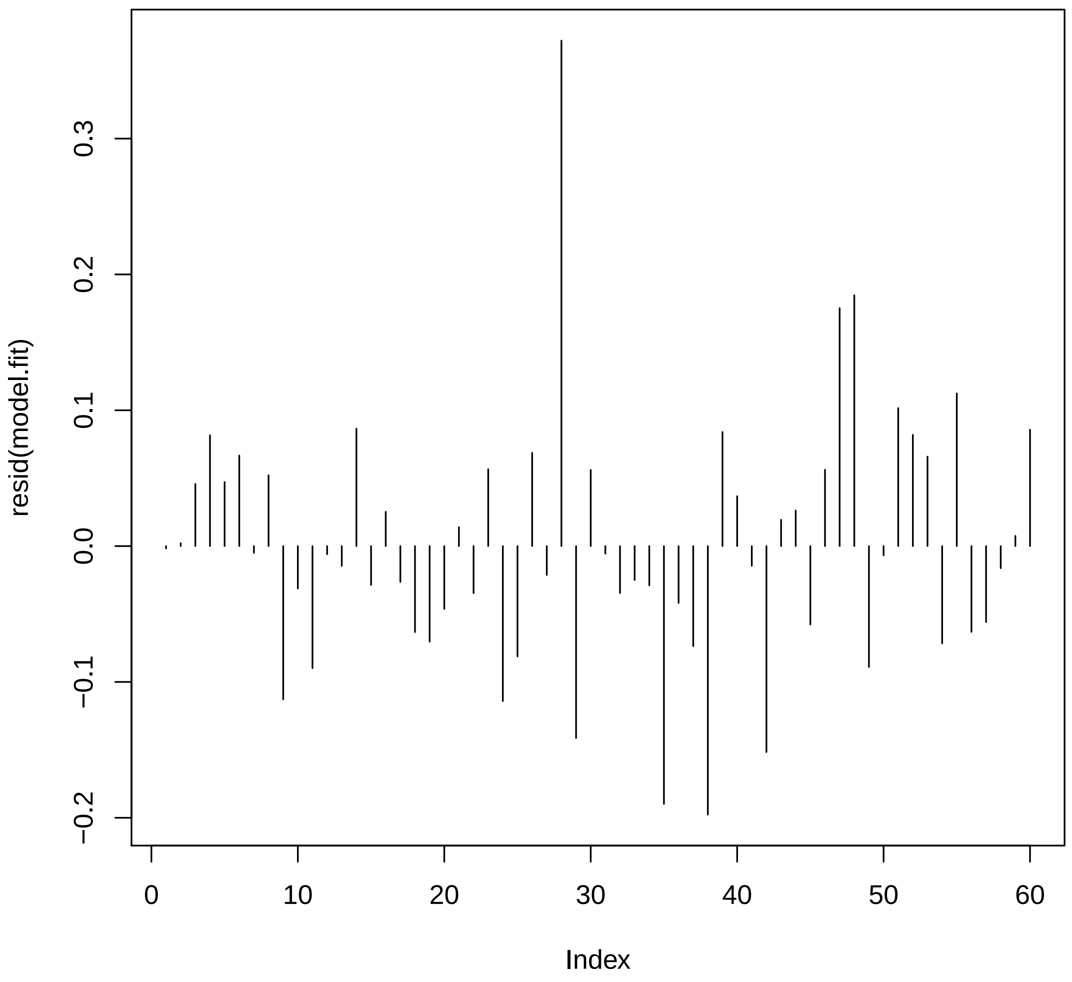
$$SER = s_{\hat{u}} \tag{6}$$

$$s_{\hat{u}}^2 = \frac{1}{n-2}\sum_{i=1}^{n}(\hat{u}_i)^2 = \frac{SSR}{n-2} \tag{7}$$

```
n <- nrow(CAT.ret)
SER <- sqrt(SSR/(n - 2))
SER

## [1] 0.09311
```

Plot the residuals of the model.

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



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

```
new <- data.frame(market = 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")
```

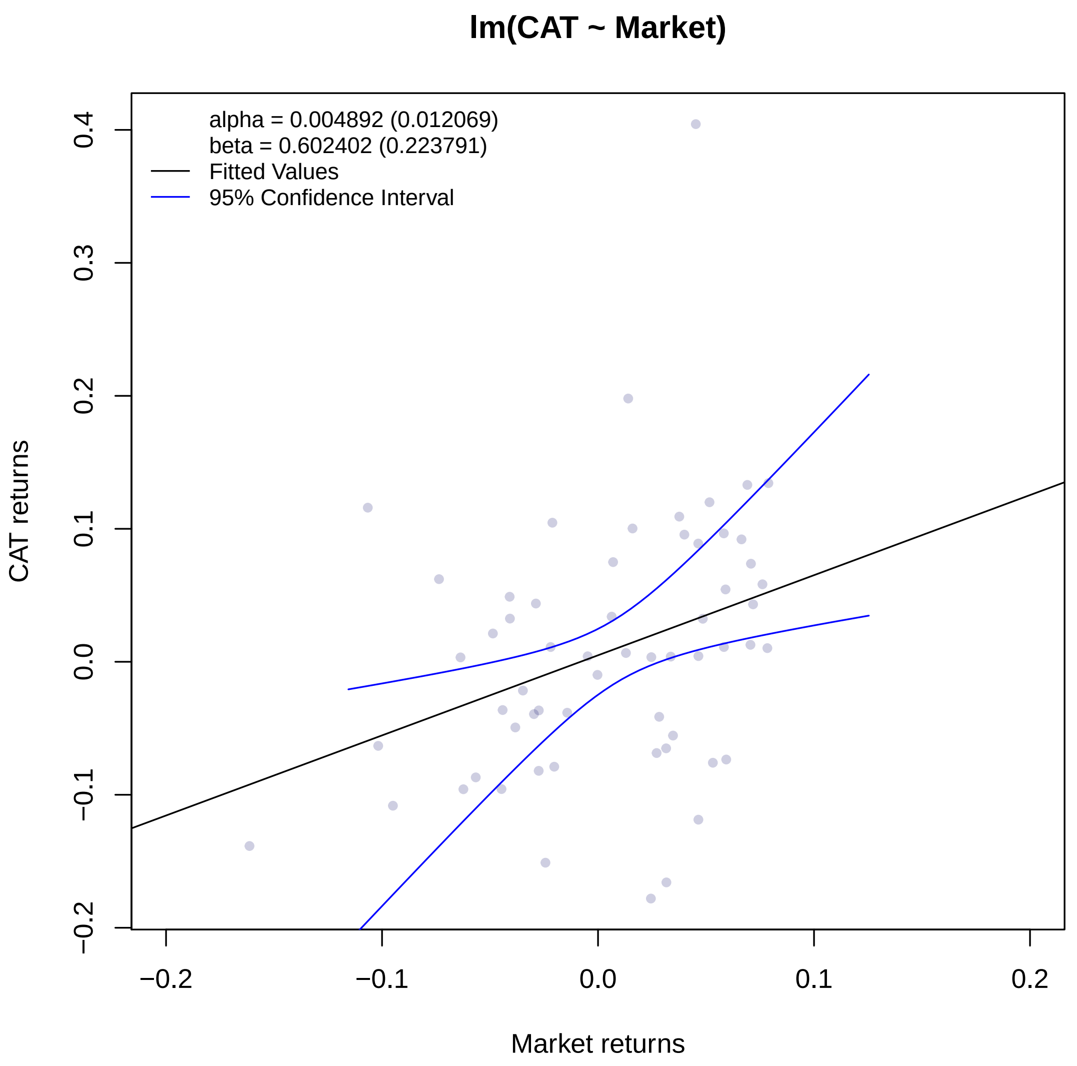
Access the betas and corresponding standard errors of the model.

```
# Get the betas and the standard errors
alpha <- round(model.coef[1, 1], 6)
alpha.se <- round(model.coef[1, 2], 6)
```

```r
beta <- round(model.coef[2, 1], 6)
beta.se <- round(model.coef[2, 2], 6)
```

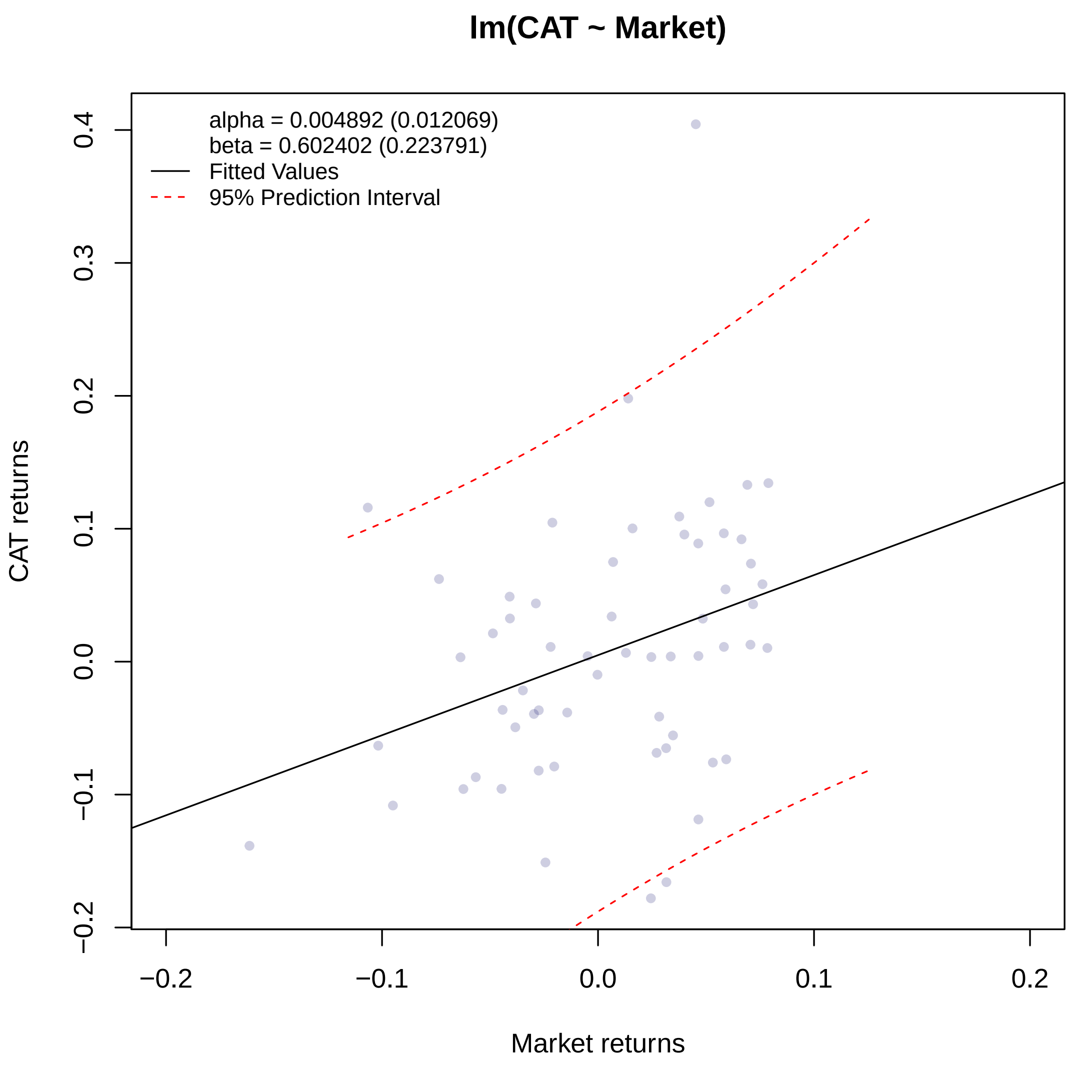
Plot the fitted model with the confidence interval.

```r
plot(coredata(MKT.ret), coredata(CAT.ret),
     col=rgb(0,0,100,50,maxColorValue=255), pch=20,
     xlab="Market returns", ylab="CAT returns", xlim=c(-0.2, 0.2),
     main="lm(CAT ~ Market)")
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)
legend("topleft", legend=c(paste("alpha = ", alpha, " (", alpha.se, ")", sep=""),
                           paste("beta = ", beta, " (", beta.se, ")", sep=""),
                           "Fitted Values", "95% Confidence Interval"),
       col=c("black", "blue"), lty=c(0, 0, 1, 1), cex=0.8, bty="n")
```



Plot the fitted model with the and prediction interval.

```r
plot(coredata(MKT.ret), coredata(CAT.ret),
     col=rgb(0,0,100,50,maxColorValue=255), pch=20,
     xlab="Market returns", ylab="CAT returns", xlim=c(-0.2, 0.2),
     main="lm(CAT ~ Market)")
abline(model.fit, col="black")
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(paste("alpha = ", alpha, " (", alpha.se, ")", sep=""),
                           paste("beta = ", beta, " (", beta.se, ")", sep=""),
                    "Fitted Values", "95% Prediction Interval"),
       col=c("black", "red"), lty=c(0, 0, 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]
head(ff_factors, 5)

##            Mkt-RF   SMB  HML
## 2005-01-14   0.00  0.44 1.10
## 2005-01-21  -1.33  0.17 0.58
## 2005-01-28   0.26  0.19 0.42
## 2005-02-04   2.90  0.94 0.39
## 2005-02-11   0.08 -0.68 0.50
```

Mkt-RF is the market excess return. SMB is **S**mall **M**inus **B**ig in terms of market capitalization. HML is **H**igh **M**inus **L**ow in terms of book-to-market.

```r
data(returns)
# 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
```

```r
# Fit the model
ff.fit <- lm(AAPL.e ~ ff_factors)
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
```

```
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.

```r
# 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
```

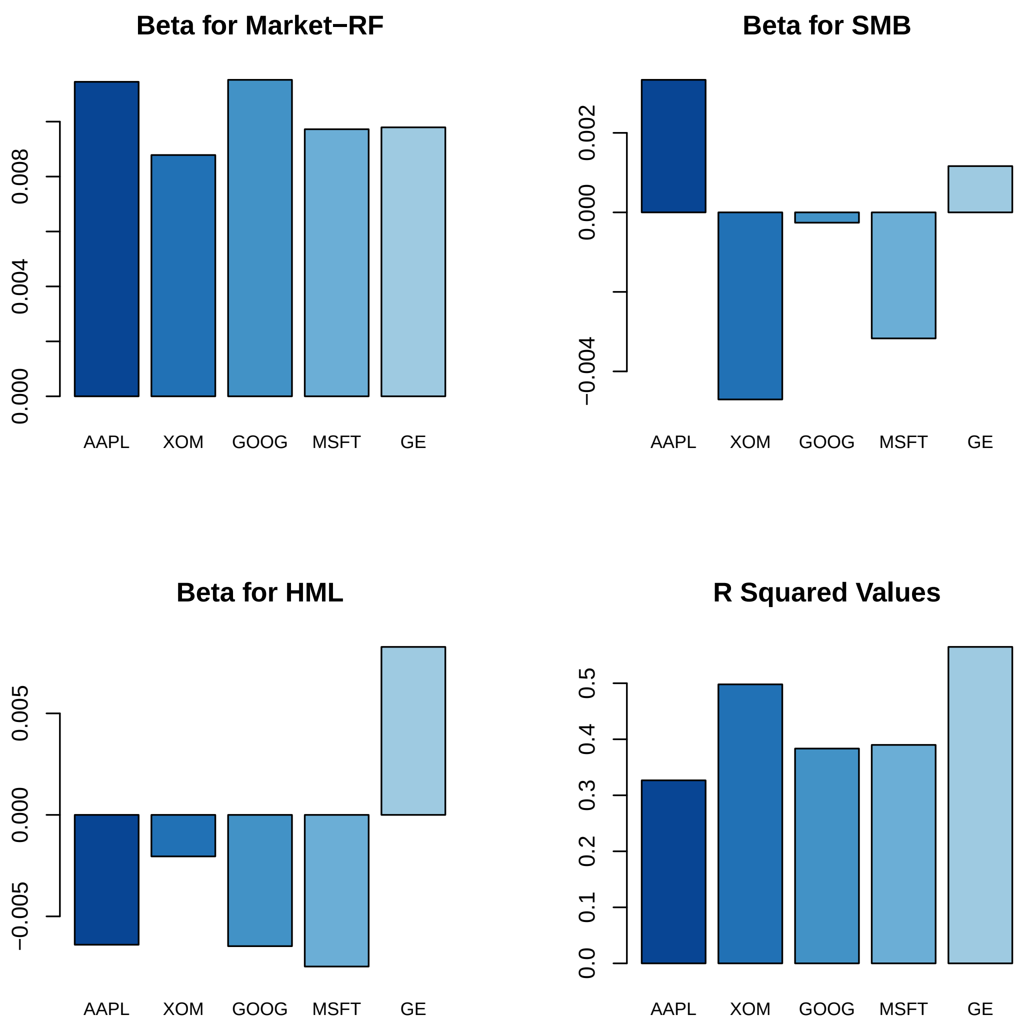
Here we fit the Fama French 3 Factor model to each asset in `ret.e`. This fits 5 models, 1 for each asset, and stores results of each model in the `ff.fit` object as a multiple linear model (mlm) object.

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

Extract and plot the beta values and the R squared values for each asset from the Fama French 3 Factor model.

```r
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)
```

```r
names(rsq) <- colnames(ret.e)


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



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

Display the summary object for each model

```
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
```