Statistical Analysis of Financial Data with Examples in R

Solutions and Comments for Selected Exercises

1 The Nature of Financial Data

1.2a **Coupon bonds.**

Using equation (1.14) in *SAFD*, the bond's fair market price is

$$p = 20/0.03 + (1000 - 20/.03)(1 + 0.03)^{-4} = 962.829.$$

For the zero-coupon bond, it is

$$p_0 = 1000(1 + 0.03)^{-4} = 888.487.$$

1.4 **Returns.**

We have $R_1 = (P_1 - P_0)/P_0$ and $R_2 = (P_2 - P_1)/P_1$, so

$$\begin{aligned} R_1 + R_2 &= (P_1 - P_0)/P_0 + (P_2 - P_1)/P_1 \\ &= \left(P_1^2 - 2P_1P_0 + P_0P_2\right)/P_0P_1. \end{aligned}$$

(For comparison, the total log return over the two periods is $\log(P_2) - \log(P_0) = r_1 + r_2$.)

1.6 **Moving averages.**

Moving averages are based on past prices, so the longer the moving average window, the more influence past prices will have compared to the current price.

If the price of a stock is in a general upward trend, past prices are lower than current or more recent prices. A moving average with a larger window will have more of those lower prices from the past than a moving average with a smaller window. So for a stock whose daily prices are generally increasing, the price (the one-day moving average) will be the highest, followed by the 20-day moving average, and then by the 50-day moving average.

1.11a **Portfolio construction.**

Finding the value of w_1 that minimizes σ_P^2 is equivalent to finding the value of w_1 that minimizes the risk σ_P.

Substituting $1 - w_1$ for w_2 and rearranging, we have

$$\sigma_\mathrm{P}^2(w_1) = w_1^2(\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2) + w_1(2\rho\sigma_1\sigma_2 - 2\sigma_2^2) + \sigma_2^2.$$

The risk is a continuous differentiable function of the weight w_1 over the closed interval $[0, 1]$; hence, a minimum in that domain is either an end point or a stationary point. We find a stationary point by differentiation.

Taking the first derivative with respect to w_1 and setting it equal to 0, we get

$$2w_1(\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2) + 2\rho\sigma_1\sigma_2 - 2\sigma_2^2 = 0,$$

or

$$w_1 = \frac{\sigma_2^2 - \rho\sigma_1\sigma_2}{\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2}.$$

This is a stationary point. The second derivative with respect to w_1 is

$$2(\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2).$$

If $\rho < 1$, this second derivative is positive; hence, the stationary point is a minimum in that case.

In the case that $\rho = 1$, there is no curvature in the function $\sigma_{\mathrm{P}}^2(w_1)$; see the graph in Figure 1.18 in *SAFD*. In that case, the minimum risk portfolio is $w_1 = 1$ if $\sigma_1 < \sigma_2$, and $w_1 = 0$ if $\sigma_1 > \sigma_2$. If $\sigma_1 = \sigma_2$, the risk is the same for any value of $w_1 \in [0, 1]$.

Appendix A1: Exercises and Solutions in R

A1.3a **Yield curves.**

We use Quandl to obtain yield curves for 1990 through the present date. Quandl's dataset is in reverse chronological order, so we reverse it, so that is it ordered chronologically. (This is not necessary, but the data frame then is in the order that we have used in other examples and exercises.)

```
require(Quandl)
ycAll_Original <- Quandl("USTREASURY/YIELD")
ind <- rev(1:dim(ycAll_Original)[1])
ycAll <- ycAll_Original[ind,]
head(ycAll, n=2)
```

which yields

```
           Date 1 MO 2 MO 3 MO 6 MO 1 YR 2 YR 3 YR 5 YR
7379 1990-01-02   NA   NA 7.83 7.89 7.81 7.87 7.90 7.87
7378 1990-01-03   NA   NA 7.89 7.94 7.85 7.94 7.96 7.92
     7 YR 10 YR 20 YR 30 YR
7379 7.98  7.94    NA  8.00
7378 8.04  7.99    NA  8.04
```

Note the missing values; the corresponding Treasury series were not offered during that time period.

A1.6 The code below computes the simple daily returns and the histogram.

```
require(quantmod)
z <- getSymbols("MSFT", env=NULL, from="2017-1-1",
     to="2018-01-01", periodicity="daily")
z <- as.numeric(z[,6])
num <- length(z)
zret <- diff(z)/z[-length(z)]
hist(zret, main="Histogram of MSFT Returns",
     xlab="Return")
```

A1.9a **Returns over different time intervals;** changing frequency in `xts` objects.

We first obtain the daily data, and compute the log differences.

```
require(quantmod)
MSFTd <- getSymbols("MSFT", env=NULL, from="1988-1-1",
      to="2018-01-01", periodicity="daily")
xdr <- diff(log(as.numeric(MSFTd[,4])))
```

We next compute some simple statistics for the daily returns:

```
> require(e1071)
> mean(xdr)
[1] 0.0007134187
> sd(xdr)
[1] 0.02040061
> skewness(xdr)
[1] -0.08236978
> kurtosis(xdr)
[1] 5.907261
```

Finally, we produce a q-q plot with a normal reference distribution.

```
qqnorm(xdr, main="MSFT Daily Returns",
       xlab="Normal Quantiles")
qqline(xdr, col="green4")
```

The distribution of the returns is very heavy-tailed. We see this in the computed kurtosis and in the q-q plot with a normal reference distribution.

The distribution appears rather symmetric, however, both from the computed skewness and from the q-q plot.

A1.12a **Option chains.**

The following code uses `getSymbols` and `getOptionChain` in the `quantmod` package. The code and the plot use standard notation in options analyses. The current stock price is shown as “S”.

When the work below was done for this exercise, the next expiry date was June 21, 2019.

```
require(quantmod)
require(lubridate)
expiry <- as.Date("2019-06-21")
today <- today()
S <- as.numeric(getSymbols("MSFT",  env=NULL,
                from=today, to=today))[4]
Kcp <- getOptionChain("MSFT", Exp=expiry)
ind1 <- which(Kcp$calls$Strike<=S)
ind <- ind1[(length(ind1)-4):length(ind1)]
ind1 <- which(Kcp$calls$Strike>=S)
ind <- c(ind,ind1[1:5])
K <- Kcp$calls$Strike[ind]
c <- (Kcp$calls$Bid[ind]+Kcp$calls$Ask[ind])/2
p <- (Kcp$puts$Bid[ind]+Kcp$puts$Ask[ind])/2
lim <- max(c,p)
```

```
plot(K,c,typ="b", xlab="Strike", ylab="Option Price",
    ylim=c(0,lim), main=paste(
    "Option Chain for MSFT, T=",expiry,", on ", today))
iv <- ifelse(S-K>0,S-K,0)
lines(K, iv, lty=2)
lines(K, p, typ="b")
iv <- ifelse(K-S>0,K-S,0)
lines(K, iv, lty=2)
text((K[1]+K[2])/2,max(c)/2,"Calls")
text((K[length(K)-1]+K[length(K)])/2,max(c)/2,"Puts")
legend("top", lty=c(1,2),
       legend=c("option price", "intrinsic value"))
text(S,0,"S")
```

The lines shown in the graph are in the typical shape of option prices. The intrinsic values are never less than 0. The time values are the differences between the prices and the intrinsic values. In this case, they are all positive. For options deep into the money, they may occasionally be negative.

A1.15 **VaR with a normal distribution.**

"Critical values" are quantiles.

Following the discussion of R functions for probability distributions on page 144 and from Table 3.4 in *SAFD*, we know that the R function to compute critical values is `qnorm`. Hence, we have

```
> qnorm(0.01, 0, 0.00892)
[1] -0.02075102
> qnorm(0.10, 0, 0.00892)
[1] -0.01143144
```

So a 1% critical value using that distribution is -0.0208, and a 10% critical value using that distribution is -0.0114.

The 5% critical value in Figure 1.38 in *SAFD* is -0.0147.

A1.22a **Working with `xts` objects;** indexing and graphing.

The code below obtains the STZ daily closes for 2017 and 2018 and plots them as an xts object.

```
require(quantmod)
z <- getSymbols("STZ", env=NULL, from="2017-1-1",
              to="2019-01-01", periodicity="daily")
plot.xts(z[,6], main="STZ Closing Prices")
```

We cannot add a support line with this type of plot; the usual commands such as `abline` and line do not work with an `xts` object.

2 Exploratory Data Analysis

2.2 **Simple summary statistics.**

The MAD is given in equation (1.62) in *SAFD*:

$$\text{MAD} = \text{median}(|x_i - \text{median}(x)|).$$

For a symmetric distribution such as the normal, it is clear that the median of the absolute values after subtraction of the median of the underlying distribution is the 75th percentile of the underlying distribution. The `qnorm` function will give this value for a theoretical normal distribution with a specified mean and standard deviation.

The code below compares the ratio of the standard deviation to MAD for five different normal distributions. The standard deviations of the distributions are different, but they all have the same means. Both the MAD and the standard deviation are invariant to the mean.

```
#  Initialize for standard deviations 1,5,10,50 and 100
sd <- c(1,5,10,50,100)
ratio <- numeric(length(sd))
for(i in 1:5) ratio[i] <- sd[i]/qnorm(.75,0,sd[i])

> ratio
[1] 1.482602 1.482602 1.482602 1.482602 1.482602
```

In all cases, the ratio is 1.482602, confirming the normal approximate relationship

$$\sigma \approx 1.48\text{MAD},$$

equation (1.64) in *SAFD*.

2.6a **The time component.**

We obtain the dataset and convert it convert it to a numeric object so as to be able to manipulate more easily.

```
require(quantmod)
NFLXd20181Q<-getSymbols("NFLX",env=NULL,from="2018-1-1",
                     to="2018-4-1", periodicity="daily")
x <- as.numeric(NFLXd20181Q[,4])
n <- length(x)
plot(x[1:(n-1)],x[2:n],xlab=expression(NFLX[t]),
    ylab=expression(NFLX[t-1]),
    main="Netflix Prices on Successive Days")
```

The graph looks like a random walk similar to Figure 2.3 in *SAFD*. One day's price is close to the previous day's price. We do not see any time-dependent patterns in the price changes, however.

2.12b **Q-q plots and S&P returns.**

In this case we bring in the data in separate files. It would be better to bring in one file and separate it.

```
require(quantmod)
SP500d1 <- getSymbols("^GSPC",env=NULL,from="1992-1-1",
            to="1996-1-1", periodicity="daily")
SP500ret1 <- diff(log(as.numeric(SP500d1[,4])))
SP500d2 <- getSymbols("^GSPC",env=NULL,from="1997-1-1",
            to="2003-1-1", periodicity="daily")
SP500ret2 <- diff(log(as.numeric(SP500d2[,4])))
SP500d3 <- getSymbols("^GSPC",env=NULL,from="2003-1-1",
            to="2006-1-1", periodicity="daily")
SP500ret3 <- diff(log(as.numeric(SP500d3[,4])))
SP500d4 <- getSymbols("^GSPC",env=NULL,from="2006-1-1",
            to="2010-1-1", periodicity="daily")
SP500ret4 <- diff(log(as.numeric(SP500d4[,4])))
```

We next compute the historical volatility for the four periods.

```
> sd(SP500ret1)
[1] 0.005697352
> sd(SP500ret2)
[1] 0.01337986
> sd(SP500ret3)
[1] 0.008206338
> sd(SP500ret4)
[1] 0.01665585
```

The historical volatility varies widely between the four periods. It is relatively low in the 1992-1995 and 2003-2005 periods (0.00570 and 0.00821 respectively) but more than doubles in the 1997-2002 and 2006-2009 periods (0.0134 and 0.0167 respectively).

Now we produce a two-by-two display of q-q plots for those four sets of returns.

```
oldparreset <- par(no.readonly=T)
par(mfrow=c(2,2))
qqnorm(SP500ret1,main="Returns 1992-1995",
       xlab="Normal Quantiles",
       ylab="S&P 500 Log Return Quantiles")
qqline(SP500ret1, col="green4")
qqnorm(SP500ret2,main="Returns 1997-2002",
       xlab="Normal Quantiles",
       ylab="S&P 500 Log Return Quantiles")
```

```
qqline(SP500ret2, col="green4")
qqnorm(SP500ret3,main="Returns 2003-2005",
       xlab="Normal Quantiles",
       ylab="S&P 500 Log Return Quantiles")
qqline(SP500ret3, col="green4")
qqnorm(SP500ret4,main="Returns 2006-2009",
       xlab="Normal Quantiles",
       ylab="S&P 500 Log Return Quantiles")
qqline(SP500ret4, col="green4")
par(oldparreset)
```

In the q-q plots, the period 2006-2009 shows the heaviest tails and thus the largest deviation from normality. The periods 1997-2002 and 2003-2005 show the next heaviest tails, with 1997-2002 showing a larger lower tail and 2003-2005 showing a larger upper tail. The period 1992-1995 has the smallest tails.

The 2006-2009 period contains the major financial difficulties of 2008, and the 1997-2002 contains the recession of 2000. The other two periods were more stable financially. Although normality and statistical volatility (standard deviation) are unrelated (see Exercise 2.12a), heavy tails, and hence, major departures from normality, tend to occur during periods of high volatility.

2.17a **Q-q plots of exceedance distributions.**

We use the FSLR returns from previous exercises.

Because we will be dealing with various subsets of the order statistics, we first just sort the data. We also create a file of quantiles corresponding to the sorted data for use in making the q-q plots.

```
FSLRretsorted <- sort(FSLRret)
n <- length(FSLRretsorted)
quants <- 1:n/n-1/(2*n)
```

We use the upper 10% of the data. The code below creates the plots.

```
p <- 0.9
FSLRtop <- FSLRretsorted[(round(p*n)):n]
quantstop <- quants[(round(p*n)):n]
oldparreset <- par(no.readonly=T)
par(mfrow=c(2,2))
qqplot(FSLRtop,qt(quantstop,df=100),main="100 df",
        xlab="FSLR top 10%",ylab="t top 10%")
 qqplot(FSLRtop,qt(quantstop,df=20),main="20 df",
        xlab="FSLR top 10%",ylab="t top 10%")
 qqplot(FSLRtop,qt(quantstop,df=5),main="5 df",
        xlab="FSLR top 10%",ylab="t top 10%")
 qqplot(FSLRtop,qt(quantstop,df=3),main="3 df",
        xlab="FSLR top 10%",ylab="t top 10%")
par(oldparreset)
```

3 Probability Distributions in Models of Observable Events

3.1a **Transformations.**

By direct substitution, we get the probability functions

$$f_Y(y) = \begin{cases} 1/4 & \text{for } y = -1 \\ 1/2 & \text{for } y = 1 \\ 1/4 & \text{for } y = 3 \\ 0 & \text{otherwise.} \end{cases}$$

$$f_Z(z) = \begin{cases} 1/2 & \text{for } z = 1 \\ 1/2 & \text{for } z = 2 \\ 0 & \text{otherwise.} \end{cases}$$

Hence, we have

$\mathrm{E}(X) = 1/4(-1) + 1/2(0) + 1/4(1) = 0.$
$\mathrm{E}(X^2) = 1/4(1) + 1/2(0) + 1/4(1) = 1/2$
hence, $\mathrm{V}(X) = 1/2 - 0 = 1/2.$

$\mathrm{E}(Y) = 1/4(-1) + 1/2(1) + 1/4(3) = 1.$
$\mathrm{E}(Y^2) = 1/4(1) + 1/2(1) + 1/4(9) = 3$
hence, $\mathrm{V}(Y) = 3 - 1 = 2.$

$\mathrm{E}(Z) = 1/2(1) + 1/2(2) = 1\frac{1}{2}.$
$\mathrm{E}(Z^2) = 1/2(1) + 1/2(4) = 2\frac{1}{2}$
hence, $\mathrm{V}(Z) = \frac{5}{2} - \frac{9}{4} = \frac{1}{4}.$

3.7a **Heavy-tailed distributions.**

The following code creates the plots with a legend in the upper left corner for the different line types and distributions.

```
curve(dnorm(x,mean=0,sd=1), from=-5, to=5,
      ylab="", ylim=c(0,0.4), lty=1, col="red")
curve(dt(x,df=2),
      from=-5, to=5, add=TRUE, lty=2, col="green4")
curve(dt(x,df=10),
      from=-5, to=5, add=TRUE, lty=3, col="blue")
curve(dcauchy(x),
      from=-5, to=5, add=TRUE, lty=4, col="black")
qend <- c("Normal","t 2 df","t 10 df", "Cauchy")
legend("topleft",legend=qend,
        col=c("red","green4","blue","black"), lty=1:4)
```

The normal curve has the highest peak and the lightest tails. The t distribution with 10 df has a slightly lower peak and slightly heavier tails. The t distribution with 2 df has a substantially lower peak and substantially heavier tails, and the Cauchy distribution, which is the same as a t with 1 df, has an even lower peak and even heavier tails.

3.11a **Extreme value distributions.**

The CDF of the max order statistic is just the CDF of the elements of the sample raised to the n^{th} power:

$$F_{X_{(n:n)}}(x) = \left(1 - \frac{1}{x}\right)^n$$

3.14a **Linear multivariate transformations.**

```
rmulvnorm <-
   function(n, d=1, mu=0, Sigma=diag(rep(1,d))) {
       x <- NA
       if ((dim(as.matrix(Sigma))[1] != d) |
           (dim(as.matrix(Sigma))[2] != d)) return(x)
       cholSig <- chol(Sigma)
       if (n ==1){
          x <- rnorm(d)%*%t(cholSig)+mu
       }else{
          x <- matrix(rnorm(n*d),nrow=n)%*%t(cholSig)
          if (length(mu)==1) x=x+mu
          if (length(mu)==d)
              x=x+matrix(rep(mu,n),nrow=n,byrow=TRUE)
       }
       return(x)
}
```

This function returns the variates in the rows of a two-dimensional array.

3.22a **Monte Carlo simulation of two stock prices using a copula.**

Note that we require only the ending prices.

```
Finv <-
   function(p, df, mu=0, lambda=1) {
          x <- qt(p, df)*sqrt((df-2)/df)*lambda + mu
}
n <- 252
df1 <- 6
tscale1 <- sqrt((df1-2)/df1)
lambda1 <- 0.002
df2 <- 3
tscale2 <- sqrt((df2-2)/df2)
lambda2 <- 0.005
mu <- 0
```

```
rho <- 0.75
rhoc <- sqrt(1-rho^2)
XYZ1 <- 100
ABC1 <- 100
p1 <- 0.5
p2 <- 1 - p1

m <- 1000
portend <- numeric(m)

set.seed (12346)
for (i in 1:m){
   x1 <- rnorm(n)
   x2 <- rho*x1 +rhoc*rnorm(n)
   q1 <- pnorm(x1)
   q2 <- pnorm(x2)
   XYZlogret <- Finv(q1, df1, mu, lambda1)
   ABClogret <- Finv(q2, df2, mu, lambda2)
   XYZend <- XYZ1*exp(diffinv(XYZlogret)[253])
   ABCend <- ABC1*exp(diffinv(ABClogret)[253])
   portend[i] <- p1*XYZend + p2*ABCend
}
```

Now we compute the standard deviation of the ending prices of the portfolio.

```
> sd(portend)
[1] 5.061779
```

4 Statistical Models and Inference

4.5 **MLE in the "standardized" t distribution.**

```
require(quantmod)
require(MASS)
getSymbols("INTC", from="2017-1-1", to="2017-10-1")
returns <- diff(log(as.numeric(INTC[,6])))

fit <- fitdistr(returns, "t")
```

This code yielded warnings that NAs were generated during the MLE optimization computations. This is to be expected in many cases. A program to perform optimization will often attempt to evaluate a function outside of its range.

The results of the fit are now stored in `fit`, which is an R list.

The three parameter estimates for the location-scale t distribution are stored in the `estimate` component of the list in the order mean, scale, and degrees of freedom (shape).

```
> fit$estimate
[1] 0.0007450371 0.0069762364 4.9716842277
```

The shape of the frequency distribution of these returns is similar to that of a t distribution with 4.97 df.

The MLEs of the mean and scale of the location-scale t distribution should be close to the actual sample mean and scale, and we see them to be so.

```
> mean(returns)
[1] 0.0003295066
> sd(returns)
[1] 0.008889629
```

4.12 **Bayesian analysis.**

We first get the raw data and compute the daily returns,

```
require(quantmod)
INTCd <- getSymbols("INTC", env=NULL, from="2018-1-1",
            to="2019-1-1", periodicity="daily")
INTCdret   <- diff(log(as.numeric(INTCd[,4])))
```

and we compute the sample mean and standard deviation,

```
> mean(INTCdret)
[1] -8.581166e-06
> sd(INTCdret)
[1] 0.0214177
```

The first step in the Bayesian analysis is decide on the prior distributions and then to use `rjags`, we make an external text file to specify the model.

```
model{
#  observables
   for(i in 1:N) {
      x[i] ~ dnorm(mu,tau)
   }
#  priors on parameters
   mu ~ dnorm(0, 1.0e-2)
   tau ~ dchisq(10)
   sigma <- 1/sqrt(tau)  #  the standard deviation
}
```

We write this file in any text editor and store it in a text file. We name the file `INTC18.bug` and store it in the directory `c:/Books/StatFinBk/BUGS/bugs_files`.

We next load the `rjags` library and use the `jags.model` function to compute the analysis. For MCMC this function requres initial values. We specify how to obtain random starting values using the R functions `rnorm` and `rchisq`.

```
require(rjags)
setwd("c:/Books/StatFinBk/BUGS/bugs_files/")
INTCRet <- list(x = INTCdret, N=length(INTCdret))
inits <- function(){list(mu = rnorm(1,0,1),
                         tau = rchisq(1,5))}
INTC18 <- jags.model("INTC18.bug", data=INTCRet)
INTC18.coda <- coda.samples(INTC18,
                            c("mu", "tau", "sigma"),
                            n.iter = 1000)
```

The results are shown below. Because of the Monte Carlo methods used in the computations. these results will vary from one run to another. Note that the sample mean lies within the central quantiles estimated on this run, but the sample standard deviation lies below the 0.025 estimated quantile.

```
> summary(INTC18.coda)
1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:
```

```
            Mean        SD  Naive SE Time-series SE
mu    -4.909e-05  0.004260 1.347e-04      0.0001347
sigma  6.595e-02  0.003098 9.795e-05      0.0001307
tau    2.314e+02 21.404894 6.769e-01      0.9059729

2. Quantiles for each variable:

            2.5%        25%        50%       75%     97.5%
mu     -0.008269  -0.002909 -9.042e-05 2.714e-03 8.278e-03
sigma   0.060620   0.063710  6.573e-02 6.803e-02 7.253e-02
tau   190.095205 216.081138  2.314e+02 2.464e+02 2.721e+02
```

4.13a **Robust fitting.**

We bring in the data and compute the returns.

```
require(quantmod)
GSPCd <- getSymbols("^GSPC", env=NULL, from="2017-1-1",
            to="2018-1-1", periodicity="daily")
GSPCdret <- diff(log(as.numeric(GSPCd[,4])))
vol <- sd(GSPCdret)
madev <- mad(GSPCdret)
iqr <- IQR(GSPCdret)

> vol
[1] 0.004192348
> madev
[1] 0.002742426
> iqr
[1] 0.003673249
```

4.17a **Tail dependencies.**

We first bring in the data, compute the returns and determine the 0.05 univariate quantiles for each set of returns.

```
require(quantmod)
INTCd <- getSymbols("INTC",env=NULL,from="2017-1-1",
            to="2018-1-1", periodicity="daily")
INTCdret <- diff(log(as.numeric(INTCd[,4])))
GLDd <- getSymbols("GLD",env=NULL,from="2017-1-1",
            to="2018-1-1", periodicity="daily")
GLDdret <- diff(log(as.numeric(GLDd[,4])))
IXICd <- getSymbols("^IXIC",env=NULL,from="2017-1-1",
            to="2018-1-1", periodicity="daily")
IXICdret <- diff(log(as.numeric(IXICd[,4])))
```

```
p <- 0.05
INTC05 <- quantile(INTCdret, p)
GLD05 <- quantile(GLDdret, p)
IXIC05 <- quantile(IXICdret, p)
```

We next produce the two bivariate scatterplots with the cutoff lines defining the lower bivariate tails.

```
oldparreset <- par(no.readonly=T)
par(mfrow=c(1,2))
plot(IXICdret, INTCdret, ylab="INTC Returns",
      xlab="Nasdaq Composite Returns",
      main="Lower Tail of Bivariate Returns")
abline(v=IXIC05, col="red")
abline(h=INTC05, col="red")
plot(IXICdret, GLDdret, ylab="GLD Returns",
      xlab="Nasdaq Composite Returns",
      main="Lower Tail of Bivariate Returns")
abline(v=IXIC05, col="red")
abline(h=GLD05, col="red")
par(oldparreset)
```

4.27b **Regression**

```
require(quantmod)
getSymbols("WBAA", src = "FRED")
WBAA1 <- WBAA["20080101/20101231"]
getSymbols("FF", src = "FRED")
FF1 <- FF["20080101/20101231"]
# Check for same variable length
length(WBAA1) #[1] 157
length(FF1)  #[1] 157
# Convert weekly prices to weekly differences
WBAAdiff <- diff(WBAA1)
FFdiff <- diff(FF1)
```

Then we regress the two weekly differences.

```
>reg <- lm(WBAAdiff~FFdiff)
>summary(reg)
Call:
lm(formula = WBAAdiff ~ FFdiff)

Residuals:
     Min       1Q   Median       3Q      Max
-0.40748 -0.07435 -0.00363  0.08044  0.70385

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.007941   0.010879  -0.730   0.4665
FFdiff      -0.228720   0.077825  -2.939   0.0038 **
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.134 on 154 df
  (1 observation deleted due to missingness)
Multiple R-squared:  0.05311,
Adjusted R-squared:  0.04696
F-statistic: 8.637 on 1 and 154 DF,  p-value: 0.003801
```

The low p-value for the FFdiff coefficient seems to indicate a linear relationship between the two variables. We plot the model using the code below.

The student should provide further discussion...

Next, residuals should be plotted in various ways, and any interesting properties noted.

4.30 **Tests for normality of returns.**

The code below will obtain the closing prices and compute the simple returns.

```
require(quantmod)
GSPCd <- getSymbols("^GSPC", env=NULL, from="2017-1-1",
                    to="2018-1-1", periodicity="daily")
z <- as.numeric(GSPCd[,4])
zret <- diff(z)/z[-length(z)]
```

The code below will test normality using the three tests.

```
require(nortest)
require(tseries)
> ad.test(zret)
        Anderson-Darling normality test
data:  zret
A = 3.7314, p-value = 2.484e-09
> jarque.bera.test(zret)
        Jarque Bera Test
data:  zret
X-squared = 98.011, df = 2, p-value < 2.2e-16
```

```
> shapiro.test(zret)
        Shapiro-Wilk normality test
data:  zret
W = 0.94384, p-value = 3.333e-08
```

All three tests have extremely small p-values, indicating nonnormal returns.

We note that the p-values for the three tests are in the second, third and second positions, respectively, of the test objects. This information will be useful in Exercise 4.31.

5 Discrete Time Series Models and Analysis

5.2 **Linear operators.**

Let $x = (x_1, x_2)$ and $y = (y_1, y_2)$, where x_1, x_2, y_1, and y_2 are real numbers, and let a be a real number. If h is linear, $h(ax+y) = ah(x)+h(y)$.

$$\begin{aligned} h(ax+y) &= h((ax_1+y_1,\, ax_2+y_2) \\ &= (ax_1+y_1)^2 + ax_2 + y_2 \\ &= a^2x_1^2 + 2ax_1y_1 + y_1^2 + ax_2 + y_2 \end{aligned}$$

and

$$ah(x) + h(y) = ax_1^2 + ax_2 + y_1^2 + y_2.$$

It is clear that these two expressions are not equal for arbitrary values of x_1, x_2, y_1, y_2, and a. The second expression lacks the cross-product term, $2ax_1y_1$, for example. Hence, $h(ax+y) \neq ah(x) + h(y)$ in general, and so h is not linear.

5.5 **Random walk; detrending.**

```
set.seed(123456)
n <- 100
df <- 4
scale <- 4
w <- rt(n, df)*sqrt((df-2)/df)*scale + 2
x <- cumsum(w)+10
plot(x, xlab="Time")
ind<-1:n
xfit <- lm (x~ind)
xfit
abline(xfit$coef[1], xfit$coef[2], col="red")
```

We use the data in `x` and the residuals from the least squares line.

```
res <- xfit$residuals
plot(res, xlab="Time", ylab="Least Squares Residuals")
```

These data appear to follow a random walk (no drift).

5.15a **ACFs and CCFs of returns.**

```
library(quantmod)
library(forecast)
z <- getSymbols("MSFT", env=NULL, from="2017-1-1",
                to="2018-01-01")
MSFTdlog <- log(as.numeric(z[,4]))
```

```
MSFTdReturns <- diff(MSFTdlog)
z <- getSymbols("^GSPC", env=NULL, from="2017-1-1",
                to="2018-01-01")
GSPCdlog <- log(as.numeric(z[,4]))
GSPCdReturns <- diff(GSPCdlog)
oldparreset <- par(no.readonly=T)
par(mfrow=c(1,2))
acfMSFT <- Acf(MSFTdReturns,
               main="MSFT Daily Log Returns")
acfGSPC <- Acf(GSPCdReturns,
               main="S&P 500 Daily Log Returns")
par(oldparreset)
```

5.18 **MA(2) models.**

We bring in the data and do the fit.

```
library(quantmod)
library(forecast)
z <- getSymbols("^GSPC", env=NULL, from="1987-1-1",
                to="2018-01-01")
GSPCdlog <- log(as.numeric(z[,4]))
GSPCdReturns <- diff(GSPCdlog)
ma2fit <- arima(GSPCdReturns, order=c(0,0,2))
```

We now inspect the fit and the residuals.

```
> ma2fit
Call:
arima(x = GSPCdReturns, order = c(0, 0, 2))

Coefficients:
          ma1      ma2  intercept
      -0.0477  -0.0534      3e-04
s.e.   0.0113   0.0116      1e-04

sigma^2 estimated as 0.0001309:
log likelihood = 23843.13,  aic = -47678.25
> Box.test(ma2fit$residuals, lag=20, type="Ljung")

        Box-Ljung test

data:  ma2fit$residuals
X-squared = 58.389, df = 20, p-value = 1.261e-05
```

The Box-Ljung test indicates that there are significant autocorrelations among the residuals from the MA(2) fit.

We may do various additional analyses and compare the ACF and the PACF, for example.

```
oldparreset <- par(no.readonly=T)
par(mfrow=c(1,2))
  Acf(GSPCdReturns, main="MA(2) Residuals")
  pacf(GSPCdReturns, main="MA(2) Residuals")
par(oldparreset)
```

We now insect the ACF and PACF of the residuals.

```
oldparreset <- par(no.readonly=T)
par(mfrow=c(1,2))
  Acf(ma2fit$residuals, main="MA(2) Residuals")
  pacf(ma2fit$residuals, main="MA(2) Residuals")
par(oldparreset)
```

We will now briefly consider other ARMA models. We will use `auto.arima{forecast}` to select an ARMA model.

```
aicbest <- auto.arima(GSPCdReturns, max.p = 5, max.q = 5,
                      ic="aic")
bicbest <- auto.arima(GSPCdReturns, max.p = 5, max.q = 5,
                      ic="bic")
```

Both the AIC and BIC suggest an ARMA(1,1) model. We now the ARMA(1,1) fit and the residuals.

```
aicbest
Box.test(aicbest$residuals, lag=20, type="Ljung")
oldparreset <- par(no.readonly=T)
par(mfrow=c(1,2))
  Acf(aicbest$residuals, main="ARMA(1,1) Residuals")
  pacf(aicbest$residuals, main="ARMA(1,1) Residuals")
par(oldparreset)
```

The Box-Ljung test indicates that there ar significant autocorrelations among the residuals from the ARMA(1,1) fit.

They do not indicate that the ARMA(1,1) model provides a good fit.

5.21a **ACF of AR(2) models.**

```
> ARMAacf(ar=c(sqrt(2), -1/2), lag.max=15)
         0          1          2          3          4
1.00000000 0.94280904 0.83333333 0.70710678 0.58333333
```

```
         5          6          7          8          9 
0.47140452 0.37500000 0.29462783 0.22916667 0.17677670 
        10         11         12         13         14 
0.13541667 0.10311974 0.07812500 0.05892557 0.04427083 
        15 
0.03314563 
```

There is a steady monotonic decrease in the autocorrelations, all of which are positive.

5.26a **ARIMA models with innovations with heavy tails.**

```
n <- 500
set.seed(12345)
xt1 <- arima.sim(n, model=list(ar=c(1/4,-1/2),
                order=c(2,0,0)), rand.gen=rt,df=5)
set.seed(12345)
xt2 <- arima.sim(n, model=list(ma=c(1/4,1/2),
                order=c(0,0,2)), rand.gen=rt,df=5)
set.seed(12345)
xt3 <- arima.sim(n, model=list(ar=c(1/4,-1/2),
      ma=c(1/4,1/2),order=c(2,0,2)), rand.gen=rt,df=5)
set.seed(12345)
xt4 <- arima.sim(n, model=list(ar=c(1/4,-1/2),
      ma=c(1/4,1/2),order=c(2,1,2)), rand.gen=rt,df=5)
oldparreset <- par(no.readonly=T)
par(mfrow=c(2,2))
plot.ts(xt1, main=expression(paste("ARIMA(2,0,0); ",
                             phi, " = (1/4,-1/2) ")))
plot.ts(xt2, main=expression(paste("ARIMA(0,0,2); ",
                             theta," = (1/4,1/2) ")))
plot.ts(xt3, main=expression(paste("ARIMA(2,0,2); ",
        phi, " = (1/4,-1/2) ", theta," = (1/4,1/2) ")))
plot.ts(xt4, main=expression(paste("ARIMA(2,1,2); ",
        phi, " = (1/4,-1/2) ", theta," = (1/4,1/2) ")))
par(oldparreset)
```

The student should look carefully at the plots generated and comment on differences and interesting properties.

5.31a **Unit roots.**

i. We bring in the data and use `adf.test` in the `tseries` library to perform the augmented Dickey-Fuller test.

```
> library(quantmod)
> library(tseries)
> z <- getSymbols("MSFT", env=NULL, from="2007-1-1",
+                 to="2008-01-01")
> MSFTd <- as.numeric(z[,6])
> adf.test(MSFTd)

        Augmented Dickey-Fuller Test

data:  MSFTd
Dickey-Fuller = -2.2096, Lag order = 6, p-value = 0.4875
alternative hypothesis: stationary
```

The test indicates the existence of a unit root.

ii. We now compute the returns and again perform the augmented Dickey-Fuller test.

```
> MSFTdreturns <- diff(log(MSFTd))
> adf.test(MSFTdreturns)

        Augmented Dickey-Fuller Test

data:  MSFTdreturns
Dickey-Fuller = -6.5643, Lag order = 6, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(MSFTdreturns) : p-value smaller than printed p-value
```

The test does not indicate the existence of a unit root.