Time-Series Analysis and Forecasting of the S&P 500 Index

Please download the file "`sp500_full.csv`". It provides the data on daily prices and returns (among other things) of the S&P 500 index from 1957-01-02 to 2021-12-31.

Distribution Fitting and Parameter Estimation

The goal is to appropriately model the distribution of S&P 500 Index and estimate the according parameters.

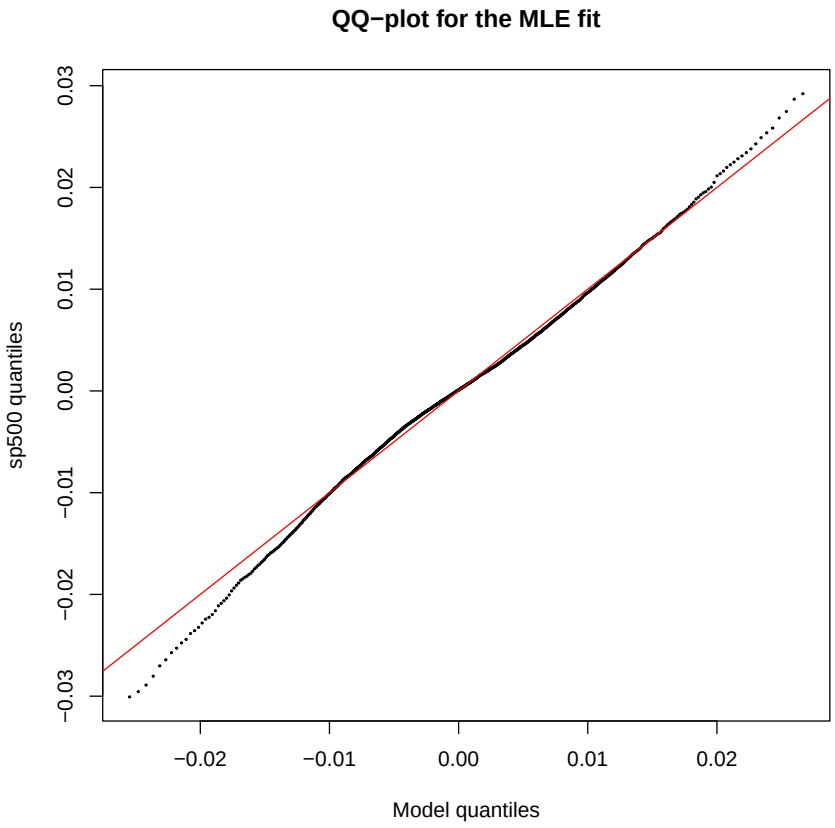
1. Using maximum likelihood, fit the t-distribution to the daily returns from 1980 to 2021-12-31 and check goodness-of-fit.

```
library(knitr)
sp = read.csv("sp500_full.csv", header = T)
sp$caldt = as.Date(as.character(sp$caldt),"%Y%m%d");
D0 = as.Date("1980/01/01","%Y/%m/%d")
D1 = as.Date("2021/12/31","%Y/%m/%d")
idx = which((sp$caldt>=D0)&(sp$caldt<=D1))
sp = sp[idx,]

n = length(sp$sprtrn)
mu = mean(sp$sprtrn)
# initialization of parameters: mu, sig, nu
start = c(mu, sd(sp$sprtrn), 5)

# define the negative log likelihood function
loglik.t = function(theta)(
  sum(-dt((sp$sprtrn-theta[1])/theta[2],
          theta[3],log=TRUE)+log(theta[2])))

# optimize the function to find the best parameters
fit = optim(start, loglik.t, hessian=T,
              method="L-BFGS-B",lower=c(-1,0.001,1))

res = fit$par;
Cov = solve(fit$hessian)
se = sqrt(diag(Cov))
# compute 95% confidence interval
```

```
res = rbind(res,fit$par-1.96*se,fit$par+1.96*se)
colnames(res) = c("mu","sig","nu")
rownames(res) = c("est","lower","upper")
kable(res)
```

	mu	sig	nu
est	0.0005942	0.0077463	5.000001
lower	0.0004249	0.0074066	3.976390
upper	0.0007635	0.0080860	6.023612

```
# generate a quantile-quantile (QQ) plot
p = seq(from=0.01,to=0.99,by=0.001)
model.quant = fit$par[1] + fit$par[2]*qt(p, df=fit$par[3]);
sp500.quant = quantile(sp$sprtrn, p);
plot(model.quant, sp500.quant,cex=0.2,
      xlab="Model quantiles",
      ylab="sp500 quantiles",
      main="QQ-plot for the MLE fit");
abline(a=0,b=1,col="red")
```



2. Using maximum likelihood, fit the skewed t-distribution to the daily returns (variable `sprtrn`) of the sp500 time series. Namely, consider the range of years 1962, 1963, ..., 2021. For the samples of daily returns corresponding to each year y in this range, obtain $\hat{\theta}_{MLE}(y)$, $y = 1962, \ldots, 2021$. Do so by using the `optim`function in R in two differnt ways (1) using the method "BFGS" and (2) via "Nelder-Mead".

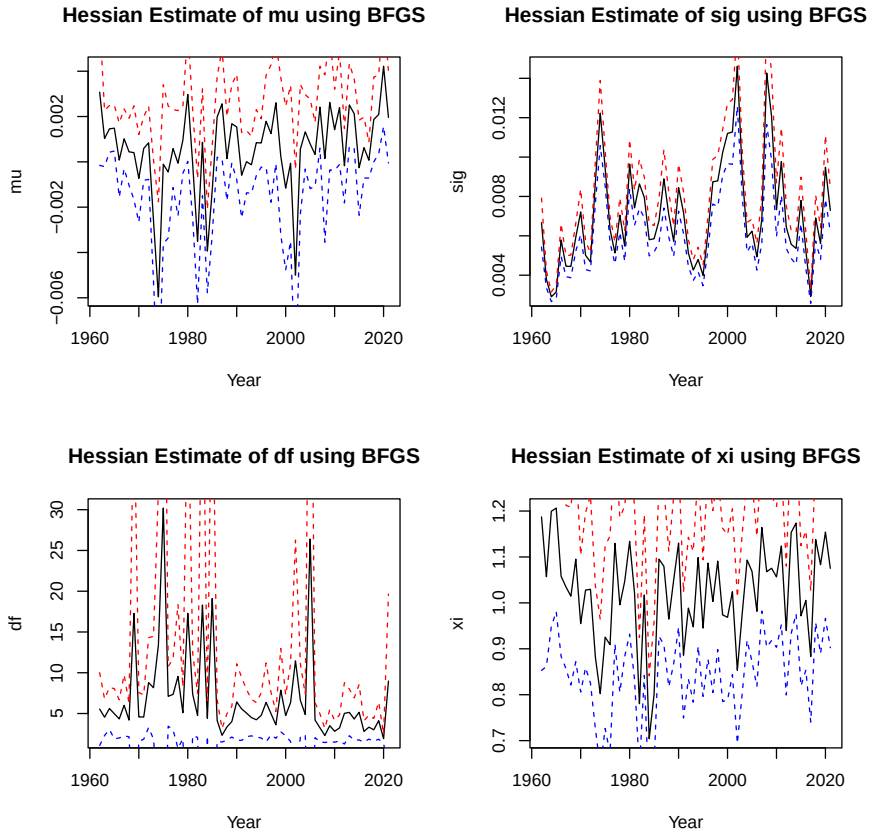
   Finally, using the Hessian estimate from the optimization routine, for each of the 4 parameters μ, σ, ν, and ξ, produce a plot of the point estimate along with point-wise 95% confidence intervals as a function of time (years).

```
sp = read.csv("sp500_full.csv", header = T)
# remove the N/A values in sp£sprtrn
```

```r
sp = sp[sp$caldt >= "19620703",]
t.sp = as.Date(as.character(sp$caldt), "%Y%m%d")
# Extract the year component of the date
t.year = format(t.sp, "%Y")
sp.split = split(sp, t.year)

# Define the skew-t density
dskew.t = function(x,xi=1,df=5){
 (dt(x*xi^(sign(x)),df=df)*2*xi/(1+xi^2))}

# The negative log-likelihood
nlog_lik_skew_t = function(theta){
  -sum(log(
        dskew.t((x-theta[1])/theta[2], df=theta[3],xi=theta[4])
        ) -log(theta[2]))
}

# Compute the Moore-Penrose generalized inverse
ginv <- function(A){
  out = svd(A)
  d = out$d; d[d>0] = 1/d[d>0]; d[d<0]=0;
  return(out$v %*% diag(d) %*% t(out$u) )
}

# theta_MLE obtained using "BFGS" method
th_BFGS = matrix(nrow=60, ncol=4)
colnames(th_BFGS) <- c("mu","sig","df","xi")
row.names(th_BFGS) <- as.character(1962:2021)
upper_BFGS = matrix(nrow=60, ncol=4)
colnames(upper_BFGS) <- c("mu","sig","df","xi")
row.names(upper_BFGS) <- as.character(1962:2021)
lower_BFGS = matrix(nrow=60, ncol=4)
colnames(lower_BFGS) <- c("mu","sig","df","xi")
row.names(lower_BFGS) <- as.character(1962:2021)

# calculate theta_MLE and the confidence level for each year
for (i in 1:length(sp.split)) {
  sp.year = sp.split[[i]]
  x = sp.year$sprtrn
```

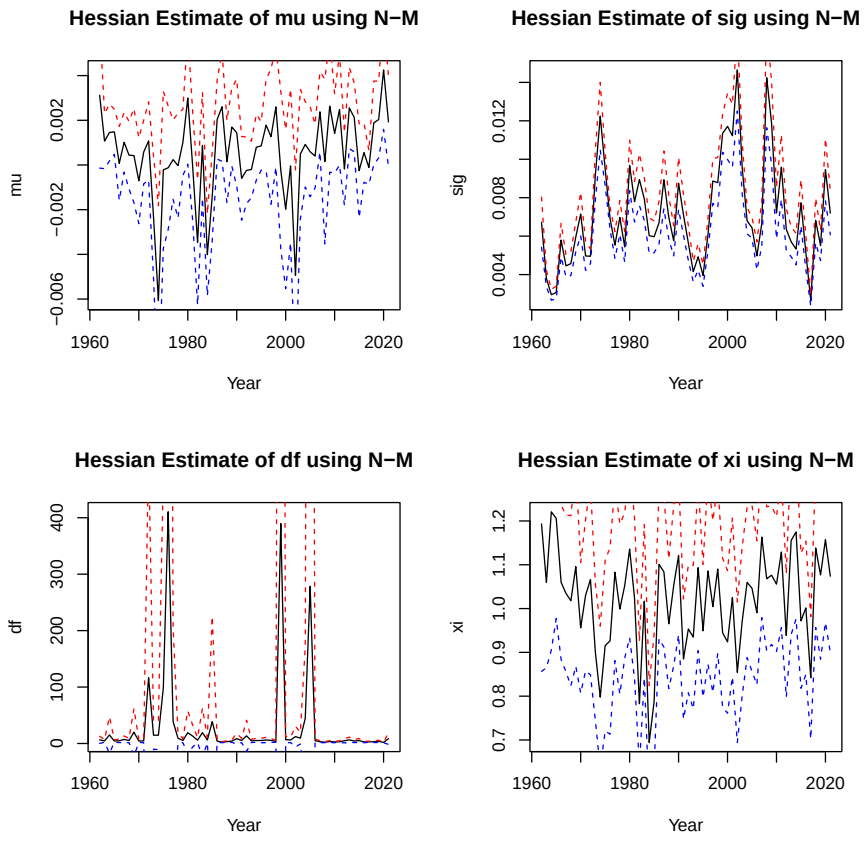
```
  # the initial values are important for convergence!
  start = c(mean(x),sd(x), 1.5, 1)
  fit_st = optim(start, nlog_lik_skew_t, hessian=T,
                 method="BFGS")
  th = fit_st$par
  th_BFGS[i,] = th

  Cov = ginv(fit_st$hessian)
  se = sqrt(abs(diag(Cov)))
  z.alpha.2 = qnorm(1-0.05/2)
  upper_BFGS[i,] = fit_st$par + z.alpha.2*se
  lower_BFGS[i,] = fit_st$par - z.alpha.2*se
}

# theta_MLE obtained using "Nelder-Mead" method
th_NM = matrix(nrow=60, ncol=4)
colnames(th_NM) <- c("mu","sig","df","xi")
row.names(th_NM) <- as.character(1962:2021)
upper_NM = matrix(nrow=60, ncol=4)
colnames(upper_NM) <- c("mu","sig","df","xi")
row.names(upper_NM) <- as.character(1962:2021)
lower_NM = matrix(nrow=60, ncol=4)
colnames(lower_NM) <- c("mu","sig","df","xi")
row.names(lower_NM) <- as.character(1962:2021)
for (i in 1:length(sp.split)) {
  sp.year = sp.split[[i]]
  x = sp.year$sprtrn
  start = c(mean(x),sd(x),5, 1)
  fit_st = optim(start, nlog_lik_skew_t, hessian=T,
                 method="Nelder-Mead")
  th = fit_st$par
  th_NM[i,] = th

  Cov = ginv(fit_st$hessian)
  se = sqrt(abs(diag(Cov)))
  z.alpha.2 = qnorm(1-0.05/2)
  upper_NM[i,] = fit_st$par + z.alpha.2*se
  lower_NM[i,] = fit_st$par - z.alpha.2*se
}
```

```
year = 1962:2021
## 1. plot the hessian estimate using BFGS method
par(mfrow=c(2, 2))
mu = data.frame(time=year, value=th_BFGS[,1])
plot(mu$time, mu$value, type="l", xlab="Year", ylab="mu",
     main='Hessian Estimate of mu using BFGS')
lines(mu$time, upper_BFGS[,1], col = "red", lty = 2)
lines(mu$time, lower_BFGS[,1], col = "blue", lty = 2)

sig = data.frame(time=year, value=th_BFGS[,2])
plot(sig$time, sig$value, type="l", xlab="Year", ylab="sig",
     main='Hessian Estimate of sig using BFGS')
lines(sig$time, upper_BFGS[,2], col = "red", lty = 2)
lines(sig$time, lower_BFGS[,2], col = "blue", lty = 2)

df = data.frame(time=year, value=th_BFGS[,3])
plot(df$time, df$value, type="l", xlab="Year", ylab="df",
     main='Hessian Estimate of df using BFGS')
lines(df$time, upper_BFGS[,3], col = "red", lty = 2)
lines(df$time, lower_BFGS[,3], col = "blue", lty = 2)

xi = data.frame(time=year, value=th_BFGS[,4])
plot(xi$time, xi$value, type="l", xlab="Year", ylab="xi",
     main='Hessian Estimate of xi using BFGS')
lines(xi$time, upper_BFGS[,4], col = "red", lty = 2)
lines(xi$time, lower_BFGS[,4], col = "blue", lty = 2)
```




```r
## 2. plot the hessian estimate using Nelder-Mead method
par(mfrow=c(2, 2))
mu = data.frame(time=year, value=th_NM[,1])
plot(mu$time, mu$value, type="l", xlab="Year", ylab="mu",
     main='Hessian Estimate of mu using N-M')
lines(mu$time, upper_NM[,1], col = "red", lty = 2)
lines(mu$time, lower_NM[,1], col = "blue", lty = 2)

sig = data.frame(time=year, value=th_NM[,2])
plot(sig$time, sig$value, type="l", xlab="Year", ylab="sig",
     main='Hessian Estimate of sig using N-M')
lines(sig$time, upper_NM[,2], col = "red", lty = 2)
lines(sig$time, lower_NM[,2], col = "blue", lty = 2)
```

```
df = data.frame(time=year, value=th_NM[,3])
plot(df$time, df$value, type="l", xlab="Year", ylab="df",
     main='Hessian Estimate of df using N-M')
lines(df$time, upper_NM[,3], col = "red", lty = 2)
lines(df$time, lower_NM[,3], col = "blue", lty = 2)

xi = data.frame(time=year, value=th_NM[,4])
plot(xi$time, xi$value, type="l", xlab="Year", ylab="xi",
     main='Hessian Estimate of xi using N-M')
lines(xi$time, upper_NM[,4], col = "red", lty = 2)
lines(xi$time, lower_NM[,4], col = "blue", lty = 2)
```



From the plots of the estimated degrees of freedom generated from

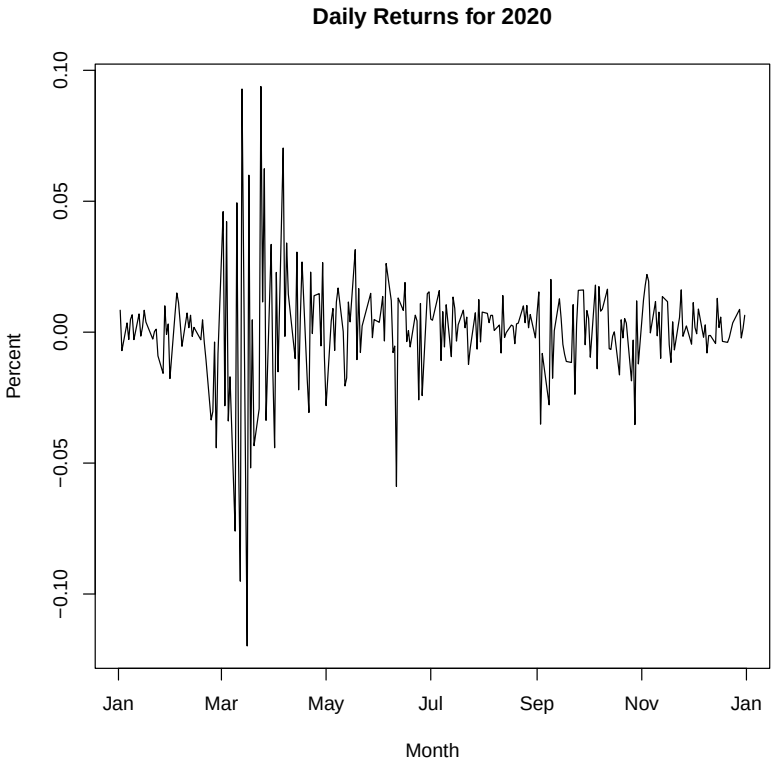
the two methods, it can be observed that the result of Nelder-Mead method is more stable though there are some spikes. Therefore in this case, Nelder-Mead method is more stable than BFGS method.

3. Consider the plot the estimated degrees of freedom parameter $\hat{\nu}(y)$ as a function of time y (in years) from the previous part.

   The below analysis is based on the results obtained using Nelder-Mead method as it's more stable than the results obtained using Nelder-Mead method.

   - A smaller degree of freedom corresponds to a heavier-tailed returns. Therefore, from the plot of estimated degrees of freedom parameter with BFGS method, there are several years when the degree of freedom is lower than 3 (which is the degree of freedom of normal distribution): 1987, 2008, 2010, 2016-2018 and 2020. Therefore, these years seem to have relatively heavy-tailed returns.
   - For values of degree of freedom less than or equal to 2, the estimated model has infinite variance and the tails of the distribution will be extremely heavy. From the results, in 2020, the value is around 1.9, which is smaller than 2. Thus, the estimated model has infinite variance in 2020.

```
t.sp = as.Date(x=as.character(sp.split[[59]]$caldt),
               format="%Y%m%d")
par(mfrow=c(1,1))
plot(t.sp, sp.split[[59]]$sprtrn, type="l",xlab="Month",
     ylab="Percent", main="Daily Returns for 2020")
```



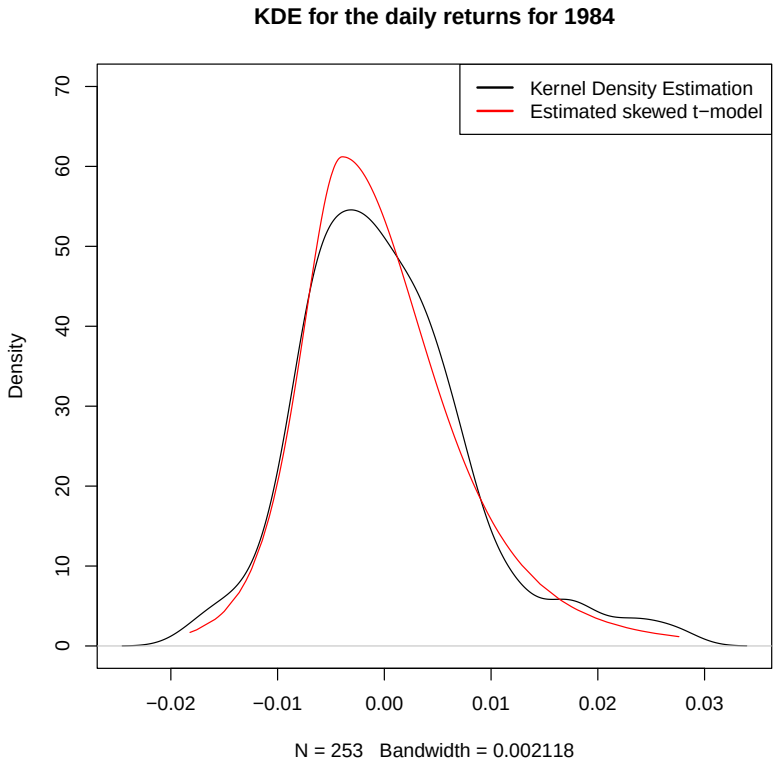
4. Consider the confidence intervals for the skewness parameter estimates obtained in the first part. Identify during which years (if any) the skewness parameter ξ is likely to be significantly different from 1. Ignoring multiple testing issues, test the corresponding (two-sided) hypothesis at a level of 5%.

   Plot the kernel density estimator for the daily returns for one of these "skewed" years and on the same plot display (in different line-style/color) the density of the estimated skewed t-model.

   - Observing the skewness parameter obtained in the first part, it can be found that in 1974, 1982, 1984 and 1985, the point-wise 95% confidence intervals do not contain the value $\xi = 1$. The hypothesis is that if the 95% confidence interval of ξ does not contain the value 1, the skewness parameter ξ is likely to be significantly different from 1.

- I choose to plot the kernel density estimator for the daily returns for 1984.

```
# Find the indices where 1 is not included in the
# confidence interval
xi_years=xi$time[which(!(lower_BFGS[,4]<=1 &
upper_BFGS[,4]>=1))]

# use daily returns for 1984
sp.1984 = sp.split[[23]]
x = sp.1984$sprtrn
start = c(mean(x),sd(x),1.5,1)
fit_st = optim(start, nlog_lik_skew_t, hessian=T,
               method="BFGS")

th = fit_st$par
x = sort(x);
plot(density(x),
     main="KDE for the daily returns for 1984",
     ylim=c(0,70));
lines(x,dskew.t((x-th[1])/th[2], df=th[3],
                xi=th[4])*(1/th[2]), cex=0.2,col="red");
legend("topright",
       legend=c("Kernel Density Estimation",
                "Estimated skewed t-model"),
       col=c("black","red"),lwd=c(2,2))
```


KDE for the daily returns for 1984
Kernel Density Estimation
Estimated skewed t−model
Density
0
10
20
30
40
50
60
70
−0.02
−0.01
0.00
0.01
0.02
0.03
N = 253 Bandwidth = 0.002118