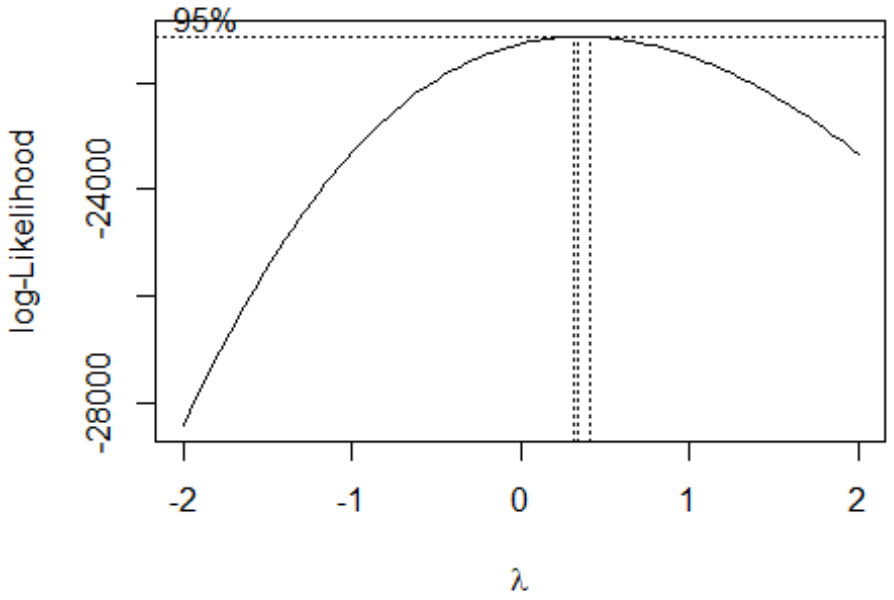
Univariate/Multivariate Models

Rob Leonard (robleonard@tamu.edu)

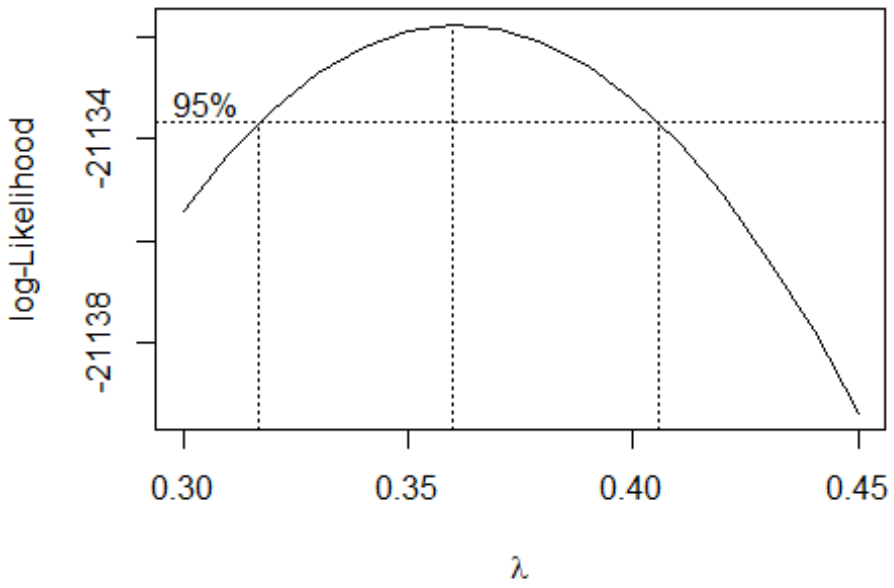
Question 1: R Lab 5.19 Problem 2

Initial setup.

```
library("MASS")
library("Ecdat")

## Loading required package: Ecfun

##
## Attaching package: 'Ecfun'

## The following object is masked from 'package:base':
##
##     sign

##
## Attaching package: 'Ecdat'

## The following object is masked from 'package:MASS':
##
##     SP500

## The following object is masked from 'package:datasets':
##
##     Orange

data(CPSch3)
male.earnings = CPSch3[CPSch3[ ,3] == "male", 2]
sqrt.male.earnings = sqrt(male.earnings)
log.male.earnings = log(male.earnings)
par(mfrow = c(1, 1))
boxcox(male.earnings ~ 1)
```




```
#boxcox(male.earnings ~ 1, lambda = seq(0.3, 0.45, 1 / 100))
bc = boxcox(male.earnings ~ 1, lambda = seq(0.3, 0.45, by = 1 / 100), interp
= FALSE)
```



```
ind = (bc$y == max(bc$y))
ind2 = (bc$y > max(bc$y) - qchisq(0.95, df = 1) / 2)
bc$x[ind]

## [1] 0.36

bc$x[ind2]

## [1] 0.32 0.33 0.34 0.35 0.36 0.37 0.38 0.39 0.40
```

2a: What are ind and ind2 and what purposes do they serve?
Ind is a logical vector, where the single "TRUE" indicates which value of the λ search sequence is the maximum likelihood estimator for the data transformation power. Ind2 is a similar logic vector where TRUE indicates the values of the sequence that are in a 95% confindence interval for the MLE.

2b: What is the effect of interp on the output from boxcox?
Interp is a logical argument, and when set to TRUE, it would turn on spline interpolation. This would theoretically make a less jagged/smoother plot, but we already have a fairly small interval. It would also increase the output vector of the 95% confidence interval values as we would no longer be using a .01 fixed interval, but values associated with the smoothing technique.

2c: What is the MLE of lambda?
The MLE of λ is 0.36.

2d: What is a 95% CI for lambda?
A 95% CI for lambda is (0.32 , 0.4).

2e: What is a 99% CI for lambda?

```
ind3 = (bc$y > max(bc$y) - qchisq(0.99, df = 1) / 2)
bc$x[ind3]

##  [1] 0.31 0.32 0.33 0.34 0.35 0.36 0.37 0.38 0.39 0.40 0.41
```

A 99% CI for lambda is (0.31 , 0.41).

Question 2: S&P 500

Part a) Fit standardized t, skewed t, GED, skewed GED.

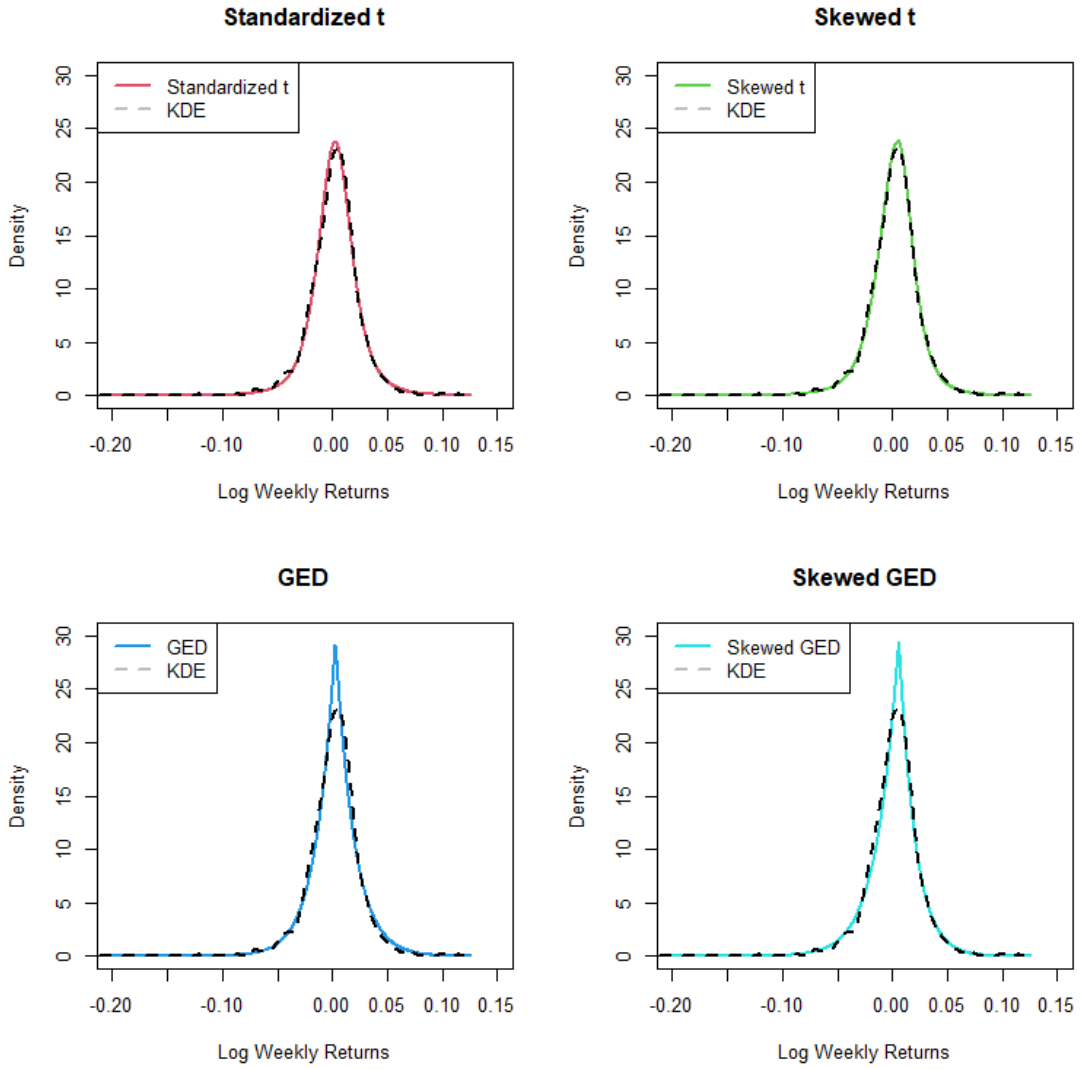
```
library(rugarch)

## Loading required package: parallel

##
## Attaching package: 'rugarch'
```

```
## The following object is masked from 'package:stats':
##
##     sigma

library(quantmod)

## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: TTR

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

getSymbols("^GSPC", from = "1991-01-01", to = "2021-02-01")

## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.

## [1] "^GSPC"

rt = weeklyReturn(Ad(GSPC), type = "log")
dists = c("std", "sstd", "ged", "sged")
fits = vector("list", 4)
for(i in 1:4) fits[[i]] = fitdist(dists[i], rt)
```

Part b) Plot density curve and overlay with kernel density estimate.

```
par(mfrow = c(2, 2))
den = density(rt)  # store kernel density estimates
labels = c("Standardized t", "Skewed t", "GED", "Skewed GED")
for (i in 1:4) {
  est = fits[[i]]$pars
  yvals = ddist(dists[i], den$x, mu = est["mu"], sigma = est["sigma"], skew =
est["skew"], shape = est["shape"])
  plot(den$x, yvals, type="l", lwd=2, col=i+1, xlab="Log Weekly Returns",
ylab="Density", main=labels[i], xlim = c(-.2,.15), ylim = c(0,30))
  lines(den$x, den$y, lwd=2, lty=2, col="black") # Kernel Density
```

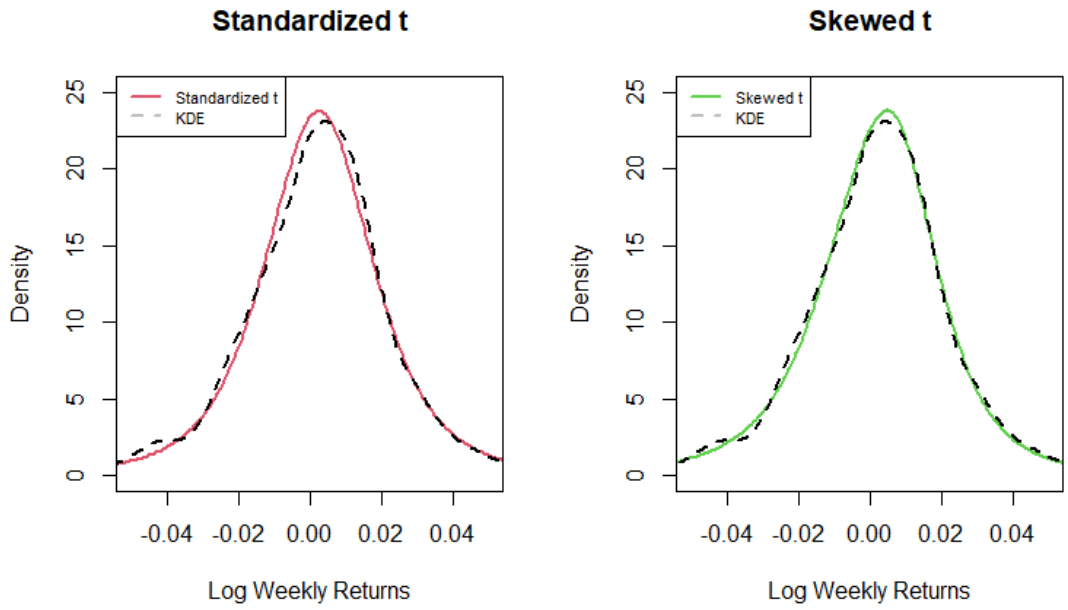
```
  legend("topleft", c(labels[i],"KDE"),  lwd=c(2,2), lty=c(1,2), col = c(i+1,
"gray"))
}
```



From the plots it appears that the standardized t and the skewed t distributions match up fairly well to the kernel density estimator. Both the GED and skewed GED have peaks that are too high.

```
# zoom in on standardized and skew t peaks
par(mfrow = c(1,2))
for (i in 1:2) {
  est = fits[[i]]$pars
  yvals = ddist(dists[i], den$x, mu = est["mu"], sigma = est["sigma"], skew =
est["skew"], shape = est["shape"])
  plot(den$x, yvals, type="l", lwd=2, col=i+1, xlab="Log Weekly Returns",
ylab="Density", main=labels[i], xlim = c(-.05,.05), ylim = c(0,25))
  lines(den$x, den$y, lwd=2, lty=2, col="black") # Kernel Density
```

```
  legend("topleft", c(labels[i],"KDE"),  lwd=c(2,2), lty=c(1,2), col = c(i+1,
"gray"), cex=.7)
}
```



From the zoomed in view, it looks like the skewed t distribution more closely matches the KDE as there is less deviation on the left side of the peak and both peaks are more closely aligned with each other. But either distribution would be a good match.

Part c) Compute the AIC and BIC criteria for all 4 models.

```
AIC_dist  = BIC_dist = rep(0,4)  # set up results vectors
BIC_dist  = rep(0,4)
for (i in 1:4){
  logLike      = min(fits[[i]]$values)
  numbParam    = length(fits[[i]]$pars) # count # parameters used for each
  AIC_dist[i]  = 2*logLike + 2*numbParam  # from p. 54 HO 3 fitdist() gives -
loglikelihood,  use last value
  BIC_dist[i]  = 2*logLike + numbParam*log(length(rt))  # formulas on p. 55
}
```

	Standardized t	Skewed t	GED	Skewed GED
AIC	-7639	-7647.1	-7606.6	-7619.1
BIC	-7622.9	-7625.7	-7590.5	-7597.6

Using AIC as a criteria, the model selects the Skewed t distribution.
Using BIC as a criteria, the model selects the Skewed t distribution.
I would also choose the Skewed t distribution since both AIC and BIC concur and the plots from part b tilt slightly towards using a Skewed t over a Standardized t.

Part d) For the 2 skewed distributions, construct 95% CI's of the skew parameter.

```
#names(fits[[2]]$pars)
sigmaDistb = sqrt(diag(solve(fits[[2]]$hessian)))  # skew is parameter 3 for
both distributions
lowerDistb = fits[[2]]$pars[3] - 1.96*sigmaDistb[3]
upperDistb = fits[[2]]$pars[3] + 1.96*sigmaDistb[3]
sigmaDistd = sqrt(diag(solve(fits[[4]]$hessian)))  # skew is parameter 3
lowerDistd = fits[[4]]$pars[3] - 1.96*sigmaDistd[3]
upperDistd = fits[[4]]$pars[3] + 1.96*sigmaDistd[3]
```

The 95% CI for skewness for the Skewed t distribution is (0.837,0.957). The 95% CI for skewness for the Skewed t distribution is (0.845,0.908).

Since neither CI contains 1, we have significant evidence to reject the null hypothesis of a symmetric distribution and conclude that both distributions are skewed. Since both skew intervals are below 1 and we are using F-S distributions, skew < 1 indicates left skew (HO 3 p. 43, symmetric when skew parameter = 1).

Part e).

```
library(fGarch)  # follow format on HO 3, page 59

## Loading required package: timeDate

## Loading required package: timeSeries

##
## Attaching package: 'timeSeries'

## The following object is masked from 'package:zoo':
##
##     time<-

## Loading required package: fBasics

##
## Attaching package: 'fBasics'

## The following object is masked from 'package:TTR':
##
##     volatility

logLik0_sstd = function(theta){  # 0 is the reduced model, fix skew to 1
which is symmetric
  -sum(dsstd(rt, mean = theta[1], sd = theta[2], nu = theta[3], xi = 1,
log=TRUE))
}
startValues = c(fits[[2]]$pars[1], fits[[2]]$pars[2], fits[[2]]$pars[4])  #
use estimates from before
fit0_sstd = nlminb(startValues, logLik0_sstd, lower=c(0, 0.001, 0))
(LRT = -2*(fits[[2]]$values[3]-fit0_sstd$objective))
```

```
## [1] 10.12049
```

```
(pval = 1-pchisq(LRT,1))
```

```
## [1] 0.0014663
```

H_0: Distribution is symmetric (skew = 1).
H_a: Distribution is not symmetric (skew ≠ 1).

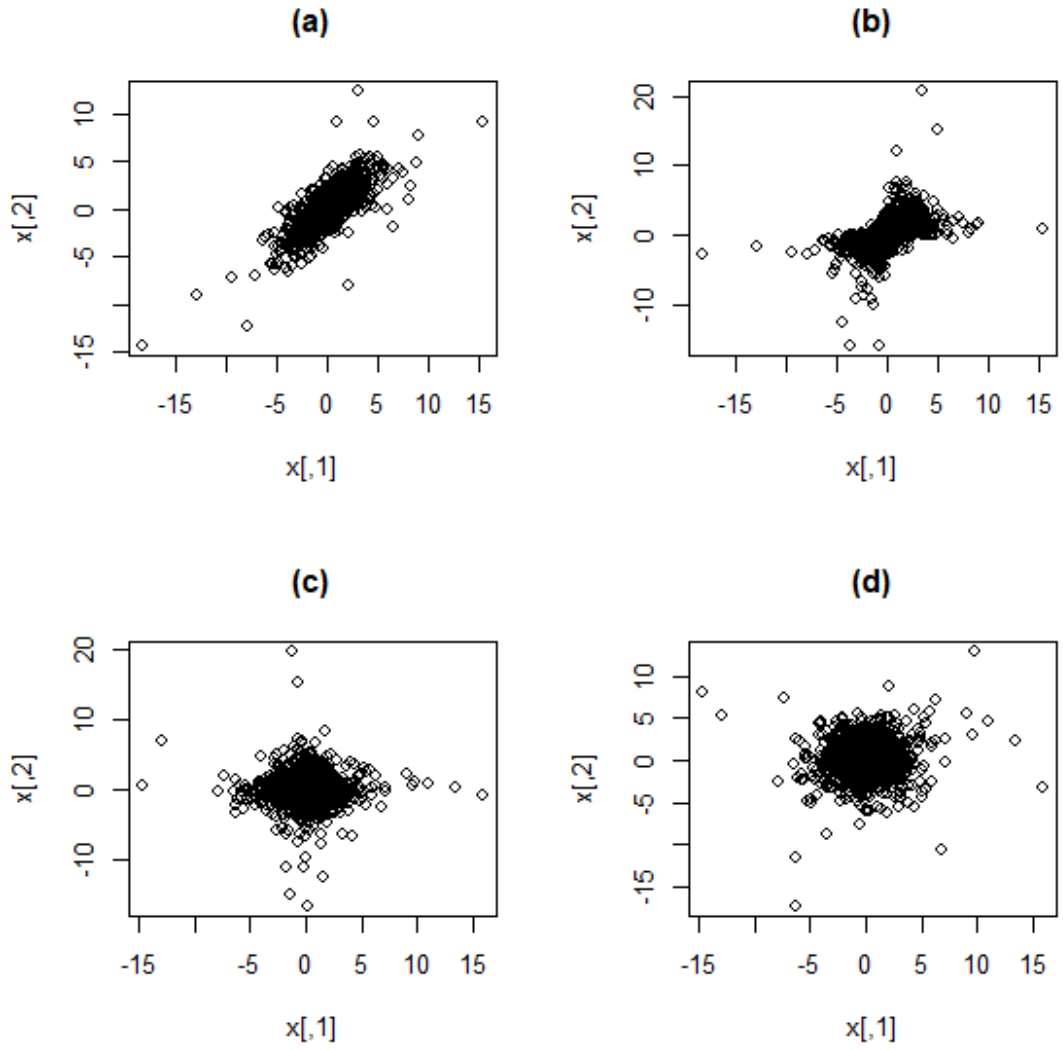
The Likelihood Ratio Test statistic is 10.12 and the corresponding p-value is 0.0015. Since the test statistic is high and the p-value if very low (less than an α=.05), there is significant evidence to reject the null hypothesis and conclude that the distribution is not symmetric. This answer corresponds to the answers in parts b and c which indicated that the Skewed t distribution was a better fit.

Question 3:

Set up.

```
par(mfrow=c(2,2))
N = 2500
nu = 3
# a
set.seed(5640)
cov=matrix(c(1, 0.8, 0.8, 1), nrow = 2)
x= mvrnorm(N, mu = c(0, 0), Sigma = cov)
w = sqrt(nu / rchisq(N, df = nu))
x = x * cbind(w, w)
plot(x, main = "(a)")
# b
set.seed(5640)
cov=matrix(c(1, 0.8, 0.8, 1),nrow = 2)
x= mvrnorm(N, mu = c(0, 0), Sigma = cov)
w1 = sqrt(nu / rchisq(N, df = nu))
w2 = sqrt(nu / rchisq(N, df = nu))
x = x * cbind(w1, w2)
plot(x, main = "(b)")
# c
set.seed(5640)
cov=matrix(c(1, 0, 0, 1), nrow = 2)
x= mvrnorm(N, mu = c(0, 0), Sigma = cov)
w1 = sqrt(nu / rchisq(N, df = nu))
w2 = sqrt(nu / rchisq(N, df = nu))
x = x * cbind(w1, w2)
plot(x, main = "(c)")
# d
set.seed(5640)
cov=matrix(c(1, 0, 0, 1), nrow = 2)
x= mvrnorm(N, mu = c(0, 0), Sigma = cov)
```

```
w = sqrt(nu / rchisq(N, df = nu))
x = x * cbind(w, w)
plot(x, main = "(d)")
```



Problem 3: Which sample has independent variates? Explain your answer.
Sample C has the independent variates as it is generated by two independent normal random variable draws combines with two separate, independent chi-square draws. The normal random variables have a correlation of 0 and for the multivariate normal case (only) this implies independence. There isn't any apparent tail dependence here as the outliers tend to cling closely to the axes.

Problem 4: Which sample has variates that are correlated but do not have tail dependence? Explain your answer.
Sample B has variates that are correlated but without tail dependence. There is no tail dependence since the sample is generated with two independent chi-square draws (rather than using the same single draw). There is correlation since the variance-covariance matrix

shows positive covariance of 0.8. The scatterplot shows that outliers again cling to the axes (so no tail-dependence).

Problem 5 Which sample has variates that are uncorrelated but with tail dependence? Explain your answer.
Sample D has uncorrelated variates but has tail dependence. The scatterplot shows that the outliers do not cling to the axes like in samples B and C. The sample is generated using the same chi-squared draw, which generates the tail dependence. The sample is uncorrelated since the multivariate normal part of the generation has a covariance value of 0 in the variance-covariance matrix.

Problem 6: Part a - What is the distribution of R?
From Handout pages 69-70
If $Y \sim t_v(\mu, \Lambda)$ *and w is a vector of weights* $w = \left(\frac{1}{2}, \frac{1}{2}\right)$, *then* $w^T Y \sim t_v(w^T\mu,\ w^T \Lambda w)$ *which is* ***univariate t***

$$Mean = w^T\mu = (.5 \quad .5)\begin{pmatrix}0.001\\0.002\end{pmatrix} = 0.0015$$

$$Variance = var(w^T Y) = w^T \Sigma w = (.5 \quad .5)\begin{pmatrix}0.10 & 0.03\\0.03 & 0.15\end{pmatrix}\begin{pmatrix}.5\\.5\end{pmatrix} = 0.0775$$

$$\boldsymbol{w^T Y \sim t_5^{std}(0.0015,\ 0.0775)} \rightarrow t_5\left(.0015, .0775 * \frac{5-2}{5}\right)$$

Problem 6: Part b - Generate a random sample from R. Compute the 0.01 upper quantile of this sample and the sample average of all returns that exceed this quantile.

```
set.seed(200128)
meanQ6    = 0.0015
varQ6     = 0.0775
sampleQ6  = 10000
vQ6       = 5
stdDevCl  = sqrt(((5-2)/5)*varQ6)  # convert to studentized/classical which
is used in rt()
Rvalues   = meanQ6 + rt(sampleQ6,vQ6)*stdDevCl
UpperQ    = round(quantile(Rvalues,.99),3)
AvgUpper  = round(mean(Rvalues[Rvalues > UpperQ]),3)
```

The upper quantile of the sample is 0.711. The average of all returns in the sample that exceed this quantile is 0.93.