Time Series Models

Rob Leonard (robleonard@tamu.edu (mailto:robleonard@tamu.edu))

Preliminary Setup

Download return data for Texas Instruments

```
library("quantmod")
```

```
## Loading required package: xts
```

```
## Warning: package 'xts' was built under R version 4.0.5
```

```
## 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
```

```
library("forecast")
```

```
## Warning: package 'forecast' was built under R version 4.0.5
```

```
library(rugarch)
```

```
## Loading required package: parallel
```

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

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

```
library(patchwork)
```

```
## Warning: package 'patchwork' was built under R version 4.0.5
```

```
source("ARMAroots_RFunctions.R")
options(scipen=999)

getSymbols("TXN", from = "2007-01-03", to = "2021-04-01")
```

```
## '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] "TXN"
```

```
sum(is.na(TXN))   # check for missing values
```

```
## [1] 0
```

```
Yt = dailyReturn(Ad(TXN), type = "log")[-1]
head(Yt, 2)
```

```
##            daily.returns
## 2007-01-04    0.01873063
## 2007-01-05   -0.01175241
```

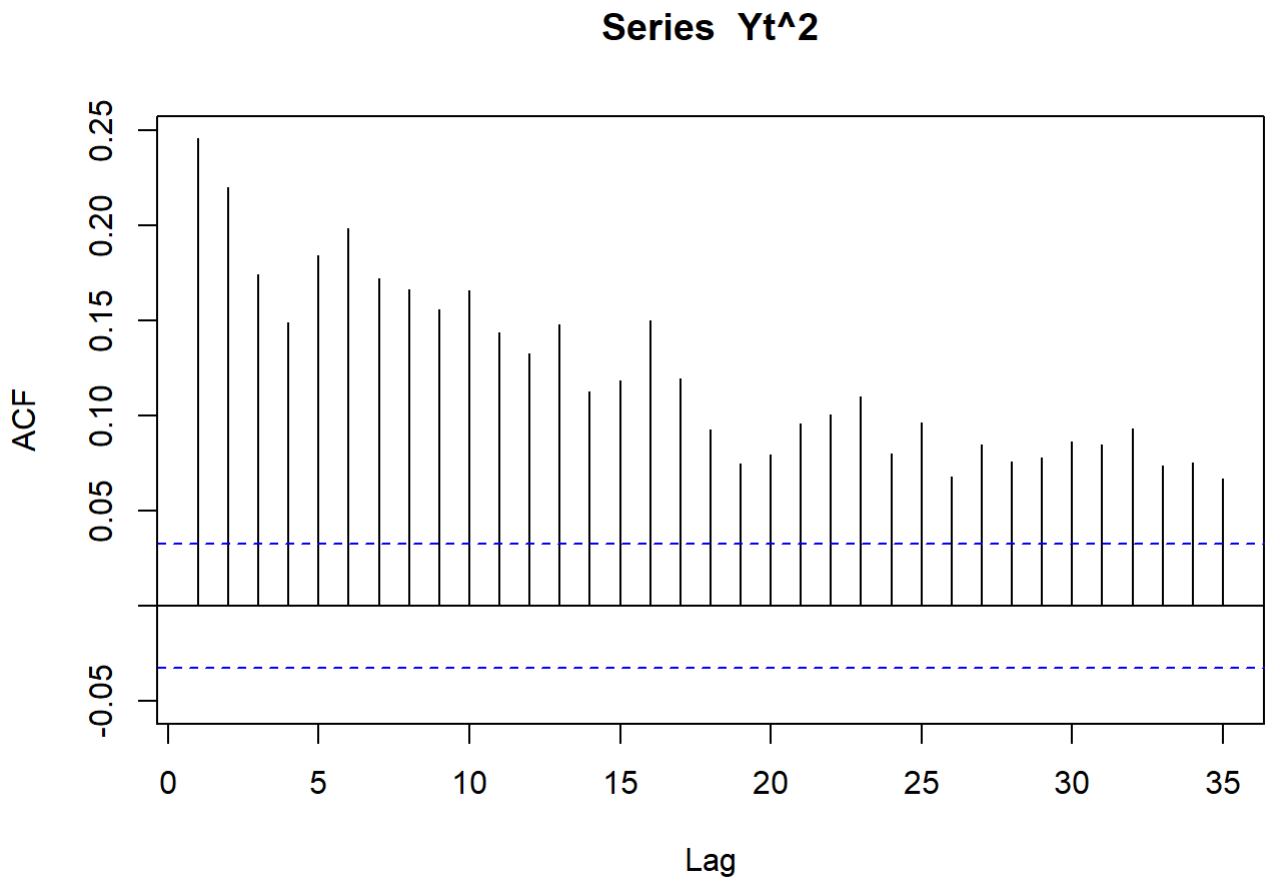
```
tail(Yt, 2)
```

```
##            daily.returns
## 2021-03-30  -0.001289968
## 2021-03-31   0.016270147
```

```
dim(Yt)
```

```
## [1] 3585    1
```

```
Acf(Yt^2)         # chech for ARCH/GARCH
```




```
ind = which(time(Yt) == "2016-12-30")
Yn = Yt[1:ind,]
dim(Yn)
```

```
## [1] 2517    1
```

```
save.image("garch.RData")
```

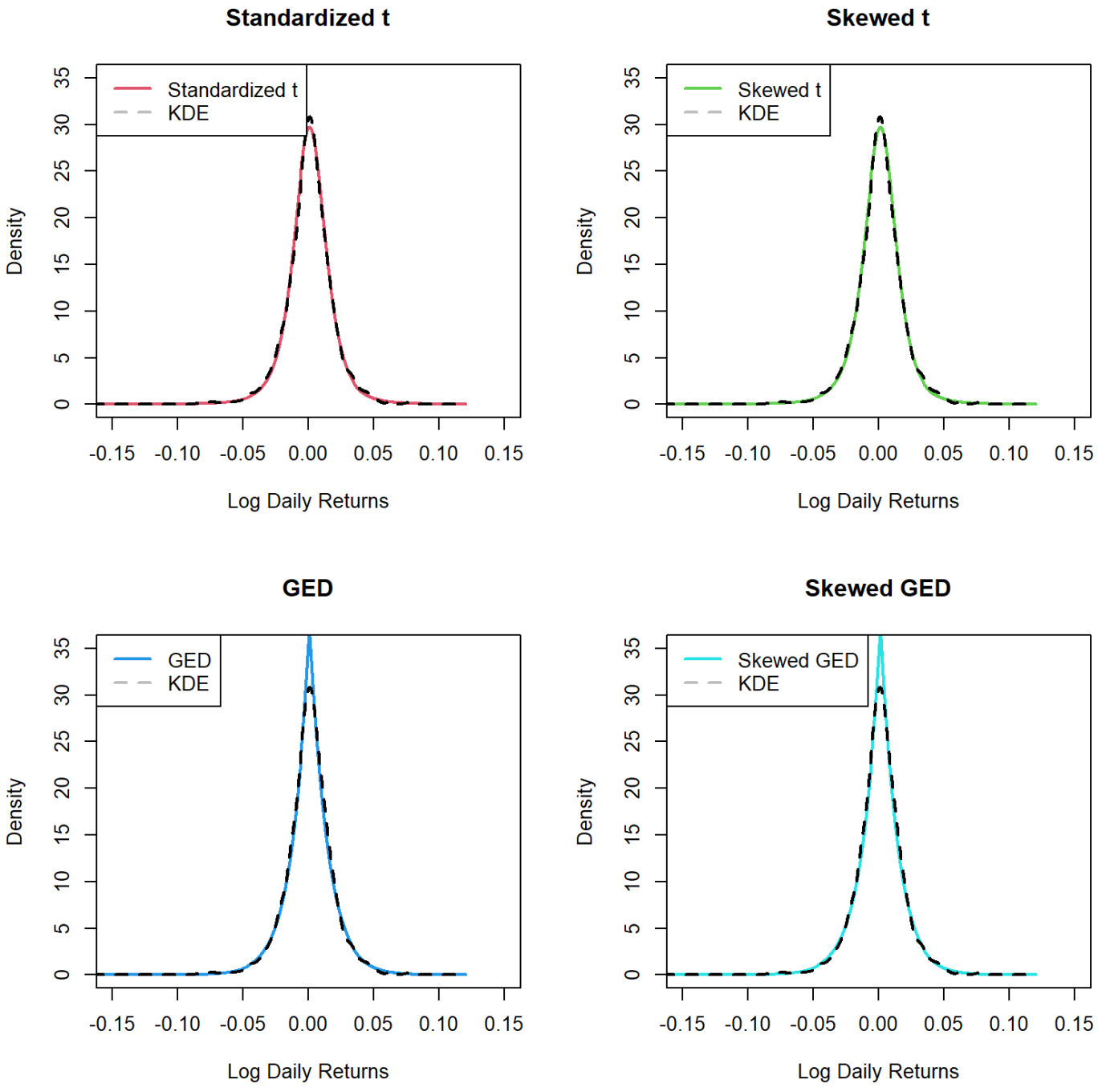
Questions 1

1a) Find a distribution that is the best fit.

```
dists = c("std", "sstd", "ged", "sged")
fits = vector("list", 4)
for(i in 1:4) fits[[i]] = fitdist(dists[i], Yn)
```

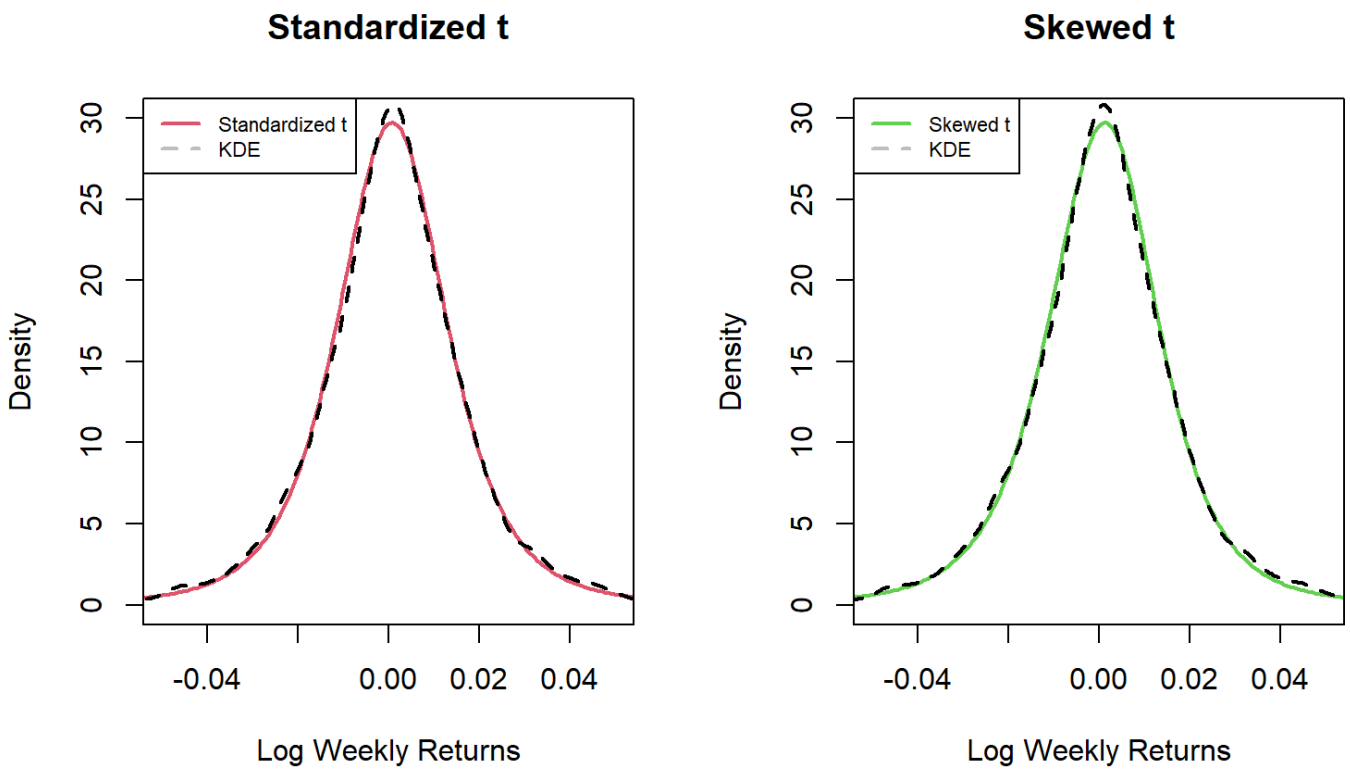
Plot densities.

```
par(mfrow = c(2, 2))
den = density(Yn)  # 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$x, yvals, type="l", lwd=2, col=i+1, xlab="Log Daily Returns", ylab="Density", main=la
bels[i], xlim = c(-.15,.15), ylim = c(0,35))
  lines(den$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"))
}
```



Either the stardized or skewed t fit the best based on these plots. The GED and skewed GED are too peaked. Zoom in on the first 2 plots:

```
# 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$x, yvals, type="l", lwd=2, col=i+1, xlab="Log Weekly Returns", ylab="Density", main=l
abels[i], xlim = c(-.05,.05), ylim = c(0,30))
  lines(den$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"), cex=.7)
}
```



Compute AIC and BIC:

```
AIC_dist  = BIC_dist = rep(0,2)  # set up results vectors
BIC_dist  = rep(0,2)
for (i in 1:2) {
  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(Yn))  # formulas on p. 55, fix this not length
(rt)
}
aicStt = round(AIC_dist[1],0)
bicStt = round(BIC_dist[1],0)
aicSkt = round(AIC_dist[2],0)
bicSkt = round(BIC_dist[2],0)
```

	Standardized t	Skewed t
AIC	-13422	-13422
BIC	-13405	-13398

AIC has no preference, but BIC ever so slightly prefers the skewed t distribution. But the skew might not be significant. Check that skewness is significant.

```
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]
```

The 95% CI for skewness for the Skewed t distribution is (0.92,1.019).
Since the CI contains 1, we don't have evidence to reject the null hypothesis that the distribution is symmetric. Thus, the **standardized t distribution** will be used. The paramters are as follows:

```
cat("The estimated standarized t distribution parameters are: \n", "mu \t", "sigma \t", "shape
 \n", round(fits[[1]]$pars,5))
```

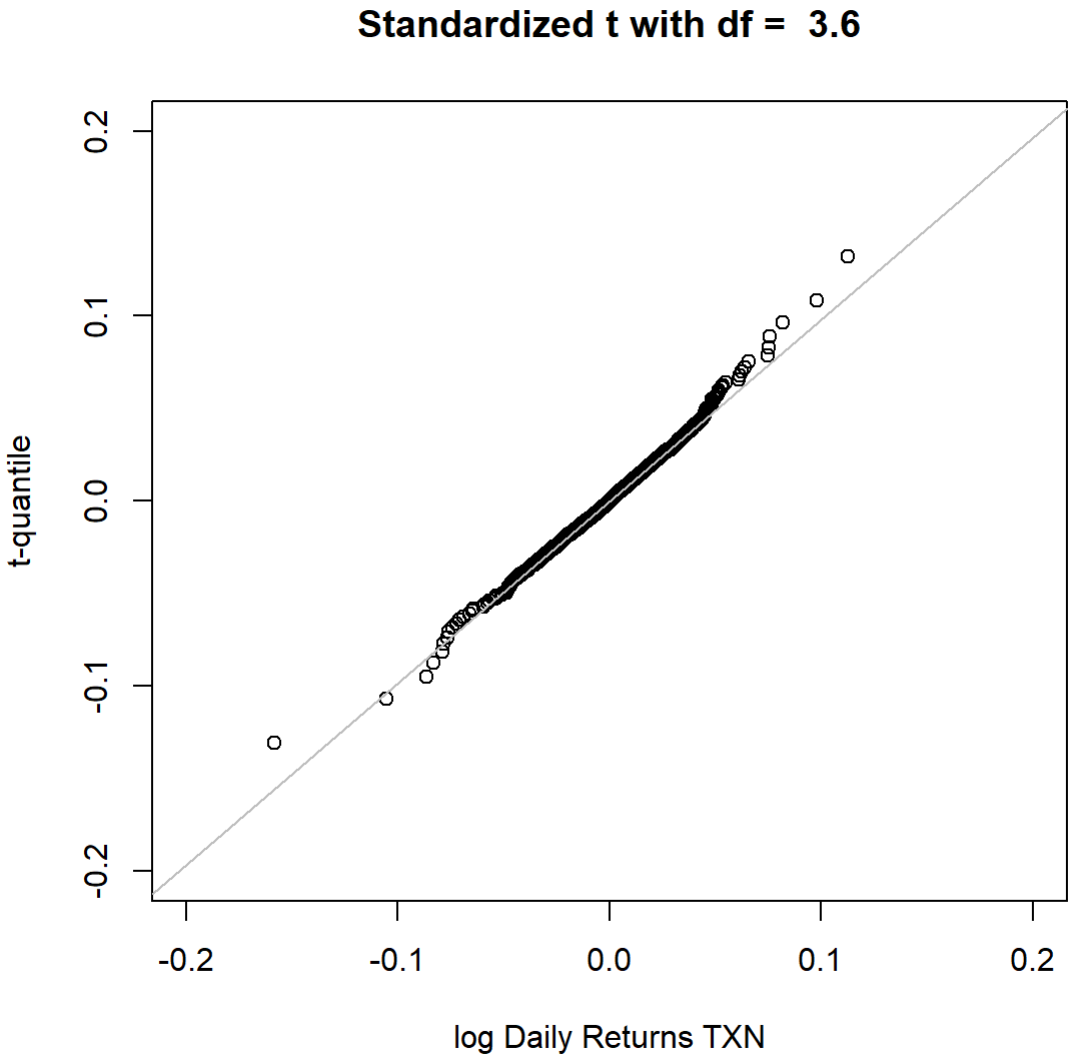
```
## The estimated standarized t distribution parameters are:
##  mu   sigma   shape
##  0.00075 0.01879 3.60028
```

1b) Plot qq plot.

```
par(mfrow = c(1, 1))
n         = length(Yn)
q_grid    = (1:n) / (n + 1)
df = fits[[1]]$pars[3]
returns = as.vector(Yn)

# calc abline for quantile
temp1 = quantile(qdist(distribution = "std", p=q_grid, mu=fits[[1]]$pars[1], sigma = fits[[1]]$p
ars[2], shape = fits[[1]]$pars[3]), c(0.25,0.75))
temp2 = quantile(Yn, c(0.25, 0.75))
temp3 = lsfit(temp1,temp2)

#qq plot
qqplot(returns, qdist(distribution = "std", p=q_grid, mu=fits[[1]]$pars[1], sigma = fits[[1]]$pa
rs[2], shape = fits[[1]]$pars[3]), main = paste("Standardized t with df = ", round(df,2)), ylab=
"t-quantile", xlab="log Daily Returns TXN", xlim = c(-.2,.2), ylim = c(-.2,.2) )
abline(temp3$coef, col = "gray")
```

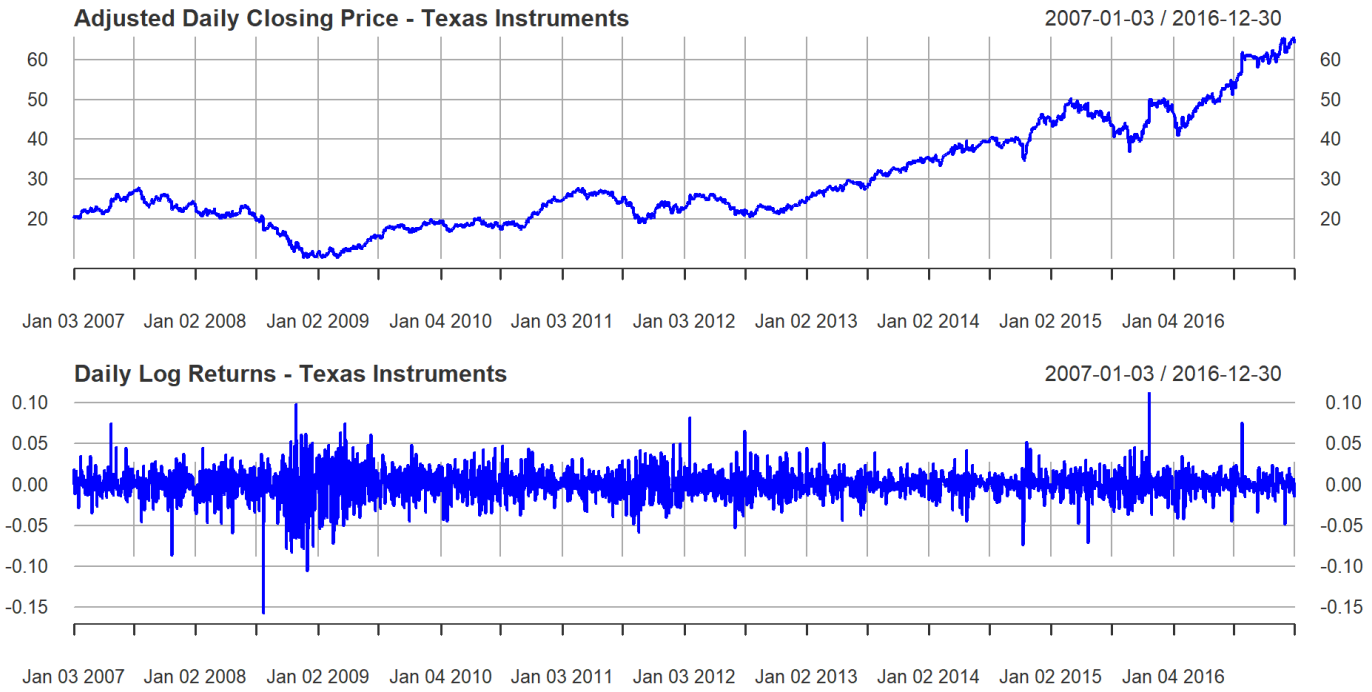


The QQ plot shows that the standardized t-distribution with 3.6 degrees of freedom fits fairly well to the daily log return data for Texas Instruments, with only very slight deviation in the tails.

Question 2: Fit an ARMA Model

Data Descritption

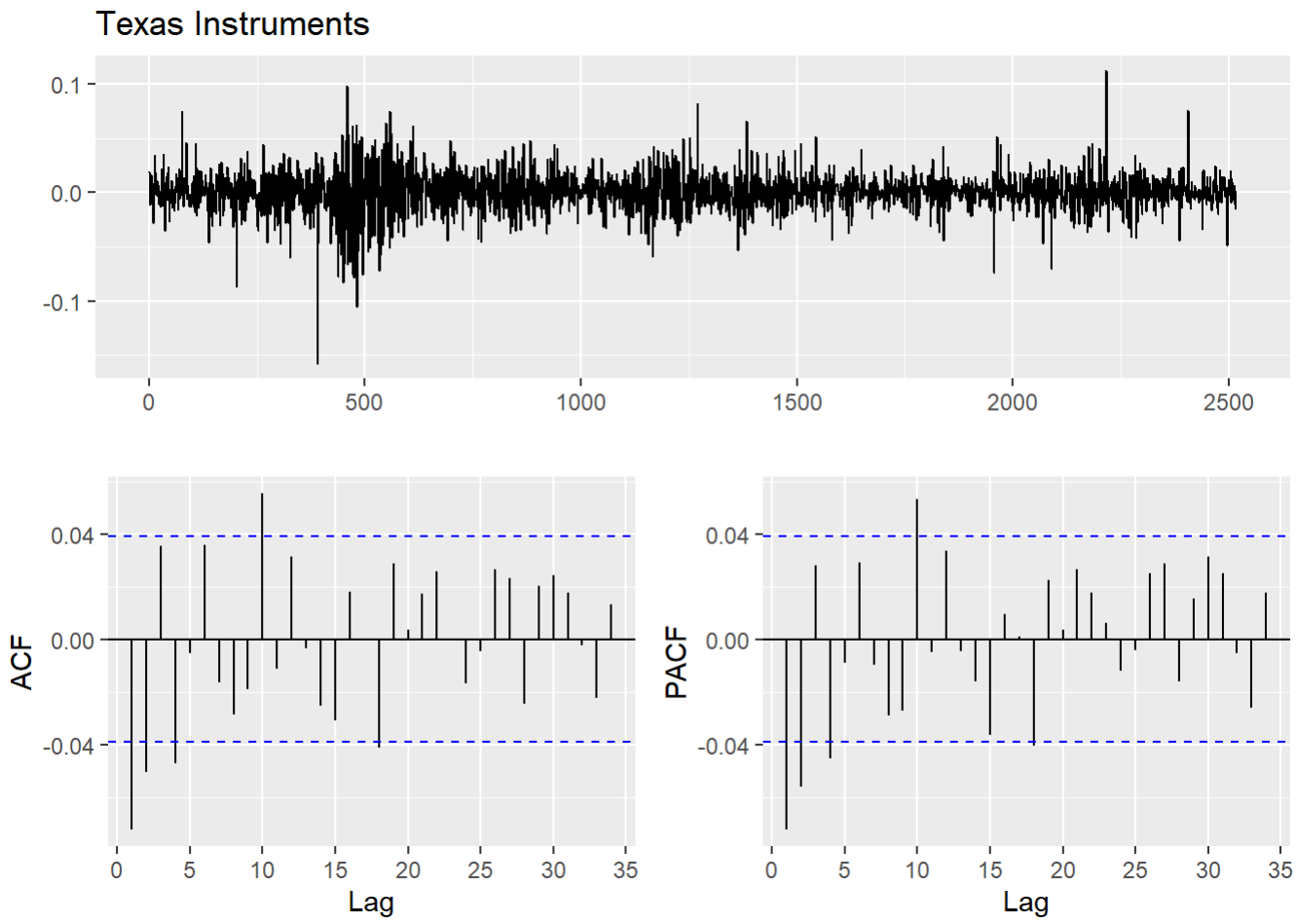
```
par(mfrow   = c(2, 1))
plot(Ad(TXN["2007-01-03::2016-12-30"]), main="Adjusted Daily Closing Price - Texas Instruments",
col="blue")
plot(dailyReturn(Ad(TXN["2007-01-03::2016-12-30"]), type="log"), main="Daily Log Returns - Texas
Instruments", col="blue")
```



The log daily returns are stationary (about 0), but do not exhibit constant variance. The return variance is clustered during certain time periods and there seems to be occasional correlation. TXN's returns were hit hard during the 2007-on financial crisis, but not as much during the pandemic.

Data Exploration

```
ggtsdisplay(Yn, points = F, main = "Texas Instruments")
```



Lags in periods 1 and 2 are generally the strongest. AR and MA models with a maximum of 2 would likely work best. For the autosearch, 4 will be used for conservatism, but models with 3 or 4 are unlikely.

Find Candidate Models

```
auto.arima(Yn, max.p = 4, max.q = 4, seasonal = F, ic ="bic" )
```

```
## Series: Yn
## ARIMA(0,0,1) with zero mean
##
## Coefficients:
##           ma1
##       -0.0794
## s.e.   0.0208
##
## sigma^2 estimated as 0.0003275:  log likelihood=6527.41
## AIC=-13050.81   AICc=-13050.81   BIC=-13039.15
```

```
auto.arima(Yn, max.p = 4, max.q = 4, seasonal = F, ic ="aic" )
```

```
## Series: Yn
## ARIMA(1,0,2) with zero mean
##
## Coefficients:
##           ar1     ma1      ma2
##       -0.4850  0.4125  -0.0898
## s.e.   0.1623  0.1620   0.0202
##
## sigma^2 estimated as 0.0003265:  log likelihood=6532.04
## AIC=-13056.07   AICc=-13056.06   BIC=-13032.75
```

The AIC chosen model isn't reasonable. The coefficients are simply way too high for the low amount of correlation exhibited in the prior plots. The BIC model looks more appealing and likely. The ma2 coefficient from the AIC chosen model also matches up well with the BIC chosen model and is much more reasonable. Multiple models will be tested further with a maximum of 2 for either/both p and q.

Model Selection

Explore 8 different model fits, although some are already fairly unlikely to work.
AR models with p = 1,2. MA models with q = 1,2. ARMA models with all 4 combinations of p,q.

```
# MA(1)
fit.MA1     = Arima(Yn, order = c(0,0,1), seasonal = FALSE, include.mean = FALSE)
plot.MA1    = autoplot_roots(fit.MA1)
MA1c1       = as.numeric(round(fit.MA1$coef[1],3))
MA1AIC      = round(fit.MA1$aicc,0)
MA1BIC      = round(fit.MA1$bic,0)

# MA(2)
fit.MA2     = Arima(Yn, order = c(0,0,2), seasonal = FALSE, include.mean = FALSE)
plot.MA2    = autoplot_roots(fit.MA2)
MA2c1       = as.numeric(round(fit.MA2$coef[1],3))
MA2c2       = as.numeric(round(fit.MA2$coef[2],3))
MA2AIC      = round(fit.MA2$aicc,0)
MA2BIC      = round(fit.MA2$bic,0)

# AR(1)
fit.AR1     = Arima(Yn, order = c(1,0,0), seasonal = FALSE, include.mean = FALSE)
plot.AR1    = autoplot_roots(fit.AR1)
AR1c1       = as.numeric(round(fit.AR1$coef[1],3))
AR1AIC      = round(fit.AR1$aicc,0)
AR1BIC      = round(fit.AR1$bic,0)

# AR(2)
fit.AR2     = Arima(Yn, order = c(2,0,0), seasonal = FALSE, include.mean = FALSE)
plot.AR2    = autoplot_roots(fit.AR2)
AR2c1       = as.numeric(round(fit.AR2$coef[1],3))
AR2c2       = as.numeric(round(fit.AR2$coef[2],3))
AR2AIC      = round(fit.AR2$aicc,0)
AR2BIC      = round(fit.AR2$bic,0)

# ARMA(1,1)
fit.ARMA11  = Arima(Yn, order = c(1,0,1), seasonal = FALSE, include.mean = FALSE)
plot.ARMA11 = autoplot_roots(fit.ARMA11)
ARMA11c1       = as.numeric(round(fit.ARMA11$coef[1],3))
ARMA11c2       = as.numeric(round(fit.ARMA11$coef[2],3))
ARMA11AIC      = round(fit.ARMA11$aicc,0)
ARMA11BIC      = round(fit.ARMA11$bic,0)

# ARMA(2,2)
fit.ARMA22  = Arima(Yn, order = c(2,0,2), seasonal = FALSE, include.mean = FALSE)
plot.ARMA22 = autoplot_roots(fit.ARMA22)
ARMA22c1       = as.numeric(round(fit.ARMA22$coef[1],3))
ARMA22c2       = as.numeric(round(fit.ARMA22$coef[2],3))
ARMA22c3       = as.numeric(round(fit.ARMA22$coef[3],3))
ARMA22c4       = as.numeric(round(fit.ARMA22$coef[4],3))
ARMA22AIC      = round(fit.ARMA22$aicc,0)
ARMA22BIC      = round(fit.ARMA22$bic,0)

# ARMA(1,2)
fit.ARMA12  = Arima(Yn, order = c(1,0,2), seasonal = FALSE, include.mean = FALSE)
plot.ARMA12 = autoplot_roots(fit.ARMA12)
ARMA12c1       = as.numeric(round(fit.ARMA12$coef[1],3))
```

```
ARMA12c2       = as.numeric(round(fit.ARMA12$coef[2],3))
ARMA12c3       = as.numeric(round(fit.ARMA12$coef[3],3))
ARMA12AIC      = round(fit.ARMA12$aicc,0)
ARMA12BIC      = round(fit.ARMA12$bic,0)

# ARMA(2,1)
fit.ARMA21  = Arima(Yn, order = c(2,0,1), seasonal = FALSE, include.mean = FALSE)
plot.ARMA21 = autoplot_roots(fit.ARMA21)
ARMA21c1       = as.numeric(round(fit.ARMA21$coef[1],3))
ARMA21c2       = as.numeric(round(fit.ARMA21$coef[2],3))
ARMA21c3       = as.numeric(round(fit.ARMA21$coef[3],3))
ARMA21AIC      = round(fit.ARMA21$aicc,0)
ARMA21BIC      = round(fit.ARMA21$bic,0)
```

Plots of the polynomial roots for each model are shown below and are labeled as figures A through H. Also shown below is a table of the relevant model coefficients and associated corrected AIC and BIC values for model comparison.

```
(plot.MA1|plot.MA2|plot.AR1) / (plot.AR2|plot.ARMA11|plot.ARMA22) /(plot.ARMA12|plot.ARMA21)  +
plot_annotation(
  title = 'Polynomial Root Plots for Each Model',
  subtitle = 'Models E, F and G all have similar/repeated roots.  All roots are within the unit
 circle.',
  tag_levels = "A", tag_prefix = 'Model ')
```

Polynomial Root Plots for Each Model

Models E, F and G all have similar/repeated roots. All roots are within the unit circle.

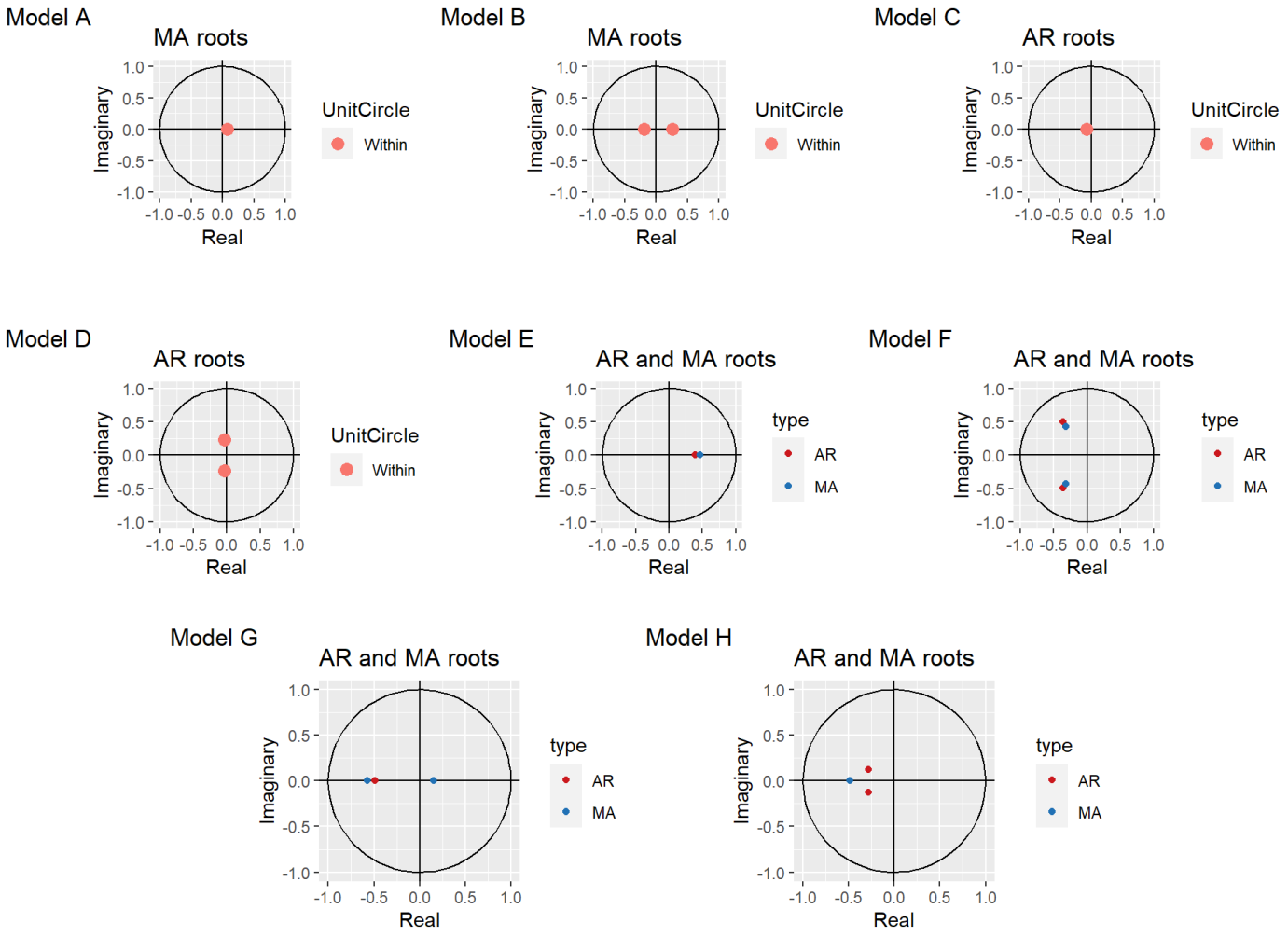


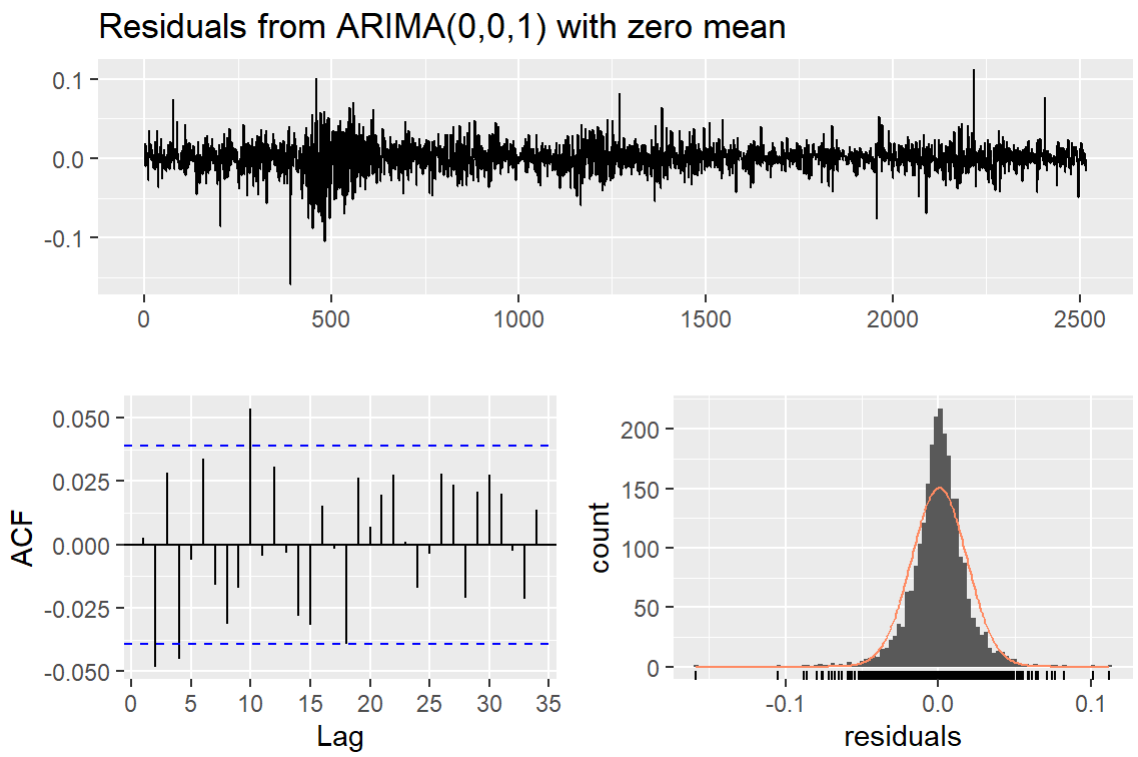
Table 1: Model coefficients, corrected AIC & BIC scores

Model	ar(1)	ar(2)	ma(1)	ma(2)	AICc	BIC
A: MA(1)	NA	NA	-0.079	NA	-13051	-13039
B: MA(2)	NA	NA	-0.073	-0.05	-13055	-13037
C: AR(1)	-0.072	NA	NA	NA	-13049	-13038
D: AR(2)	-0.076	-0.055	NA	NA	-13055	-13038
E: ARMA(1,1)	0.387	NA	-0.465	NA	-13052	-13035
F: ARMA(2,2)	-0.707	-0.374	0.636	0.285	-13056	-13027
G: ARMA(2,1)	-0.559	-0.093	0.486	NA	-13057	-13033
H: ARMA(1,2)	-0.485	NA	0.412	-0.09	-13056	-13033

Models E through H are simply not feasible. Most of these have duplicate/similar roots so information is being repeated. Models A through D all have very similar AIC and BIC scores and each has unique roots within the unit circle. These 4 models will be explored further by examining residuals.

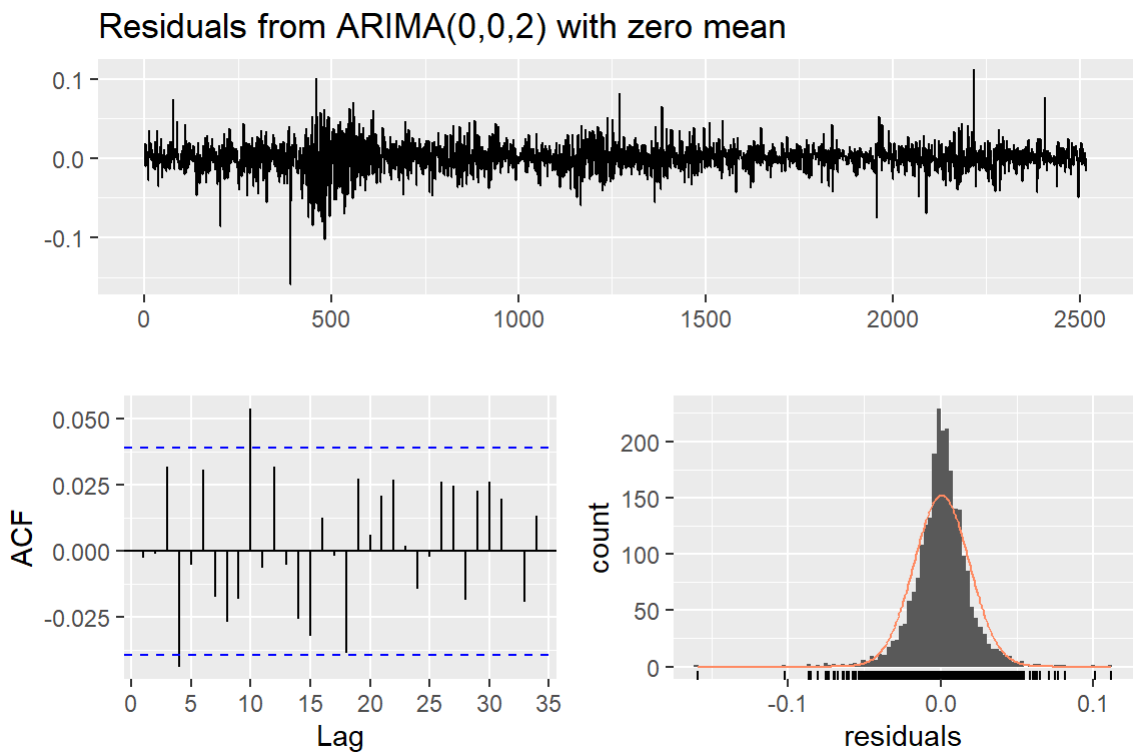
Check Model Residuals

```
plot.resA = checkresiduals(fit.MA1, points = FALSE)
```



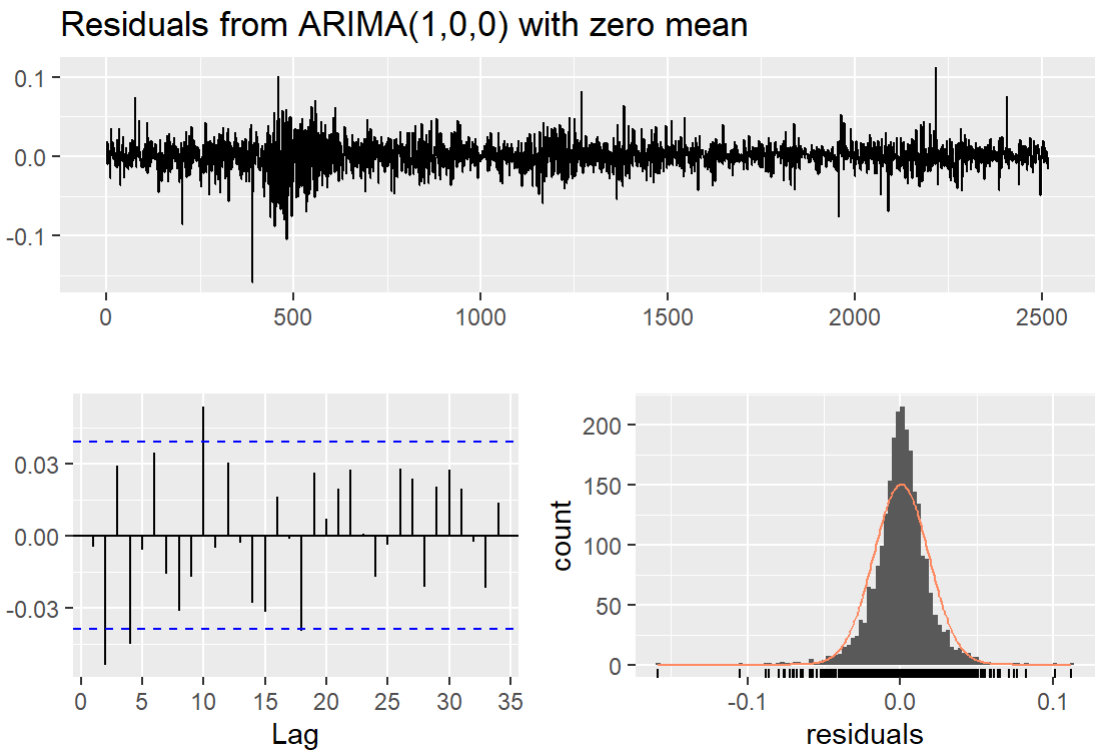

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,1) with zero mean
## Q* = 27.096, df = 9, p-value = 0.001349
##
## Model df: 1.   Total lags used: 10
```

```
plot.resB = checkresiduals(fit.MA2, points = FALSE)
```



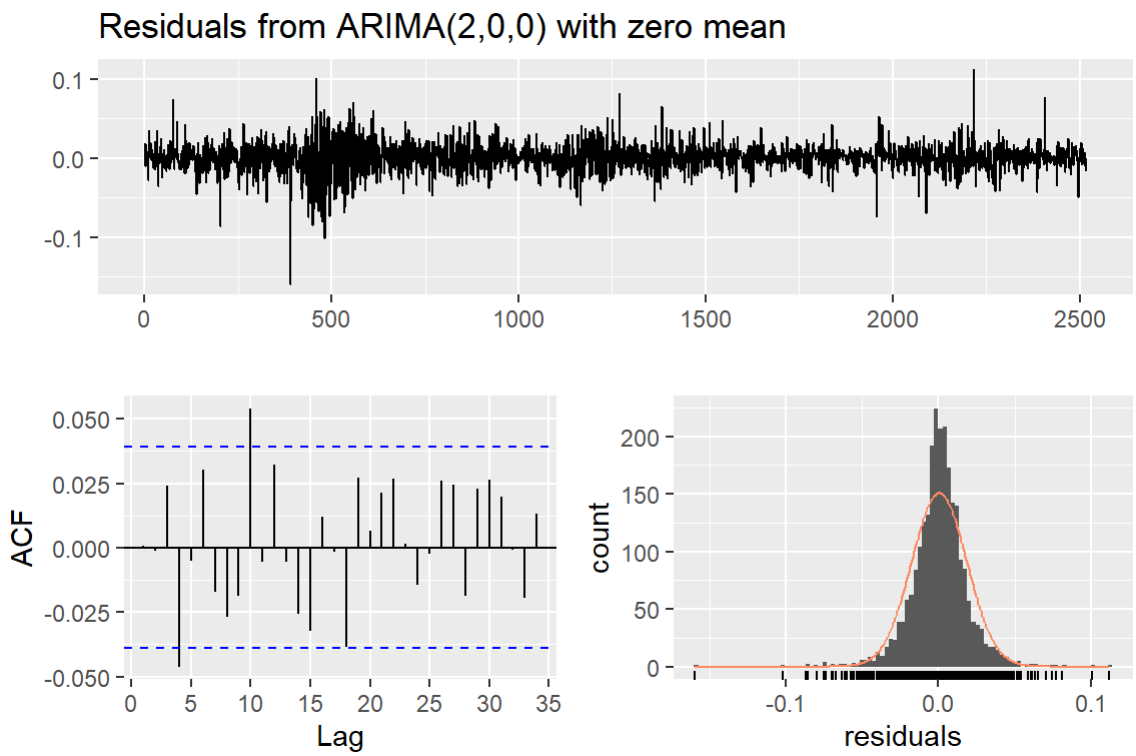

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,2) with zero mean
## Q* = 20.606, df = 8, p-value = 0.00827
##
## Model df: 2.   Total lags used: 10
```

```
plot.resC = checkresiduals(fit.AR1, points = FALSE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0) with zero mean
## Q* = 28.871, df = 9, p-value = 0.0006813
##
## Model df: 1.   Total lags used: 10
```

```
plot.resD = checkresiduals(fit.AR2, points = FALSE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,0) with zero mean
## Q* = 20.036, df = 8, p-value = 0.0102
##
## Model df: 2.   Total lags used: 10
```

From the residual plots and from the table above, it appears that the MA(2) model fits our data the best, as the only somewhat significant lag (of the first 5) is the 4th, and it's just barely significant. This model will be used in forecasting.

Question 3: Forecasting

3) Give the 1-, 2- and 3-step-ahead point forecasts and the corresponding 95% prediction intervals.

```
fore3.out = predict(fit.MA2, n.ahead=3)
PIlow = PIhi = rep(0,3)
q3 = quantile(qdist(distribution = "std", p=q_grid, mu=fits[[1]]$pars[1], sigma = fits[[1]]$pars
[2], shape = fits[[1]]$pars[3]),.975)
for (i in 1:3){
  PIlow[i] = fore3.out$pred[i] - q3 * fore3.out$se[i]
  PIhi[i] = fore3.out$pred[i] + q3 * fore3.out$se[i]
}
```

Step Ahead	Point Estimate - Log Return	95% Prediction Interval
1	0.00115	from 0.00049 to 0.00182

Step Ahead	Point Estimate - Log Return	95% Prediction Interval
2	0.00083	from 0.00016 to 0.0015
3	0	from -0.00067 to 0.00067

3b) Use Calculate the k-step-ahead point forecasts and the corresponding 90% and 95% prediction intervals for k = 1, 2, . . . , 100 with forecast(). Assign an object for the return value, please do not display them. Please plot the forecast.

```
pred3b = forecast(fit.MA2, h=100, level = c(90,95))
autoplot(pred3b, xlim=c(2517,2617), ylim = c(-.05,.05), xlab = "step ahead")
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```

```
## Warning: Removed 2516 row(s) containing missing values (geom_path).
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```