PLS_GARCH

2023-01-23

Implied Volatility vs. Call Price Surface

```
rm(list=ls())
library(pls)
```

```
##
## Attaching package: 'pls'

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

```
library(forecast)
```

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

```
library(car)
```

```
## Loading required package: carData
```

```
library(tseries)
library(rugarch)
```

```
## Loading required package: parallel

##
## Attaching package: 'rugarch'

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

```
library(Dowd)
```

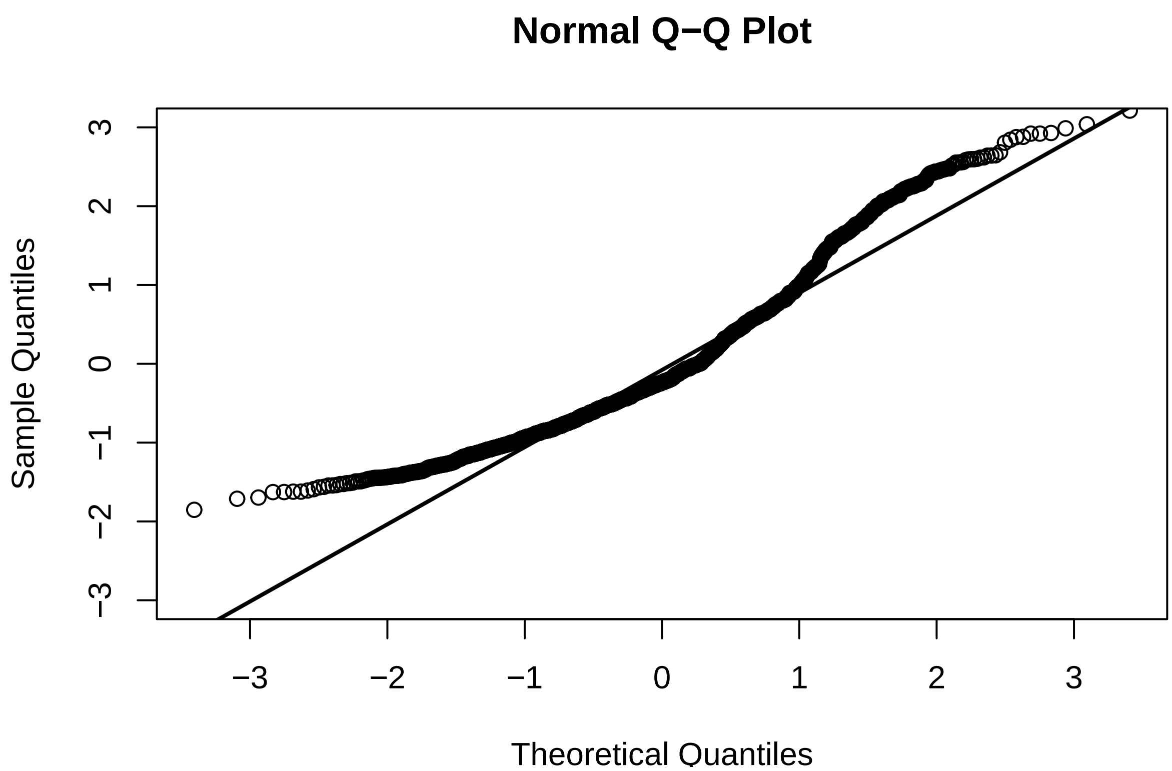
```
## Loading required package: bootstrap

##
## Attaching package: 'bootstrap'

## The following object is masked from 'package:pls':
##
##     crossval

## Loading required package: MASS
```

```
library(moments)

# Import data
Data_impvol <- read.csv("/Users/benjye/Dropbox/Pricing/Data_R/Data_impvol.csv",
```

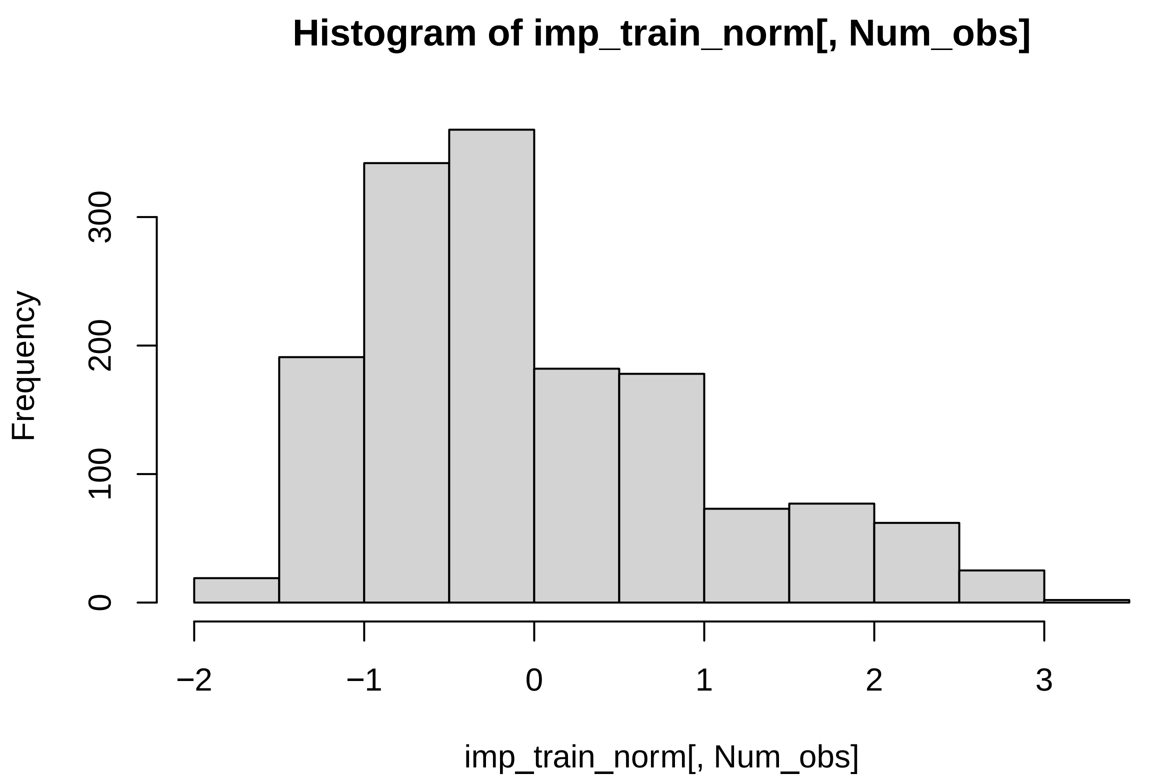
```r
                      header=TRUE, stringsAsFactors = FALSE)

S_all <- Data_impvol[,1]
imp_all <- Data_impvol[,-c(1)]

# Start from 2009
start <- 1767
R_all <- diff(log(S_all))
R_all <- R_all[-1]
R_all <- R_all[start:length(R_all)]
R_sq_all <- R_all^2
imp_all <- imp_all[-c(1),]
imp_all <- imp_all[start:nrow(imp_all),]

# Split data and normalize it
N_test <- 1000
R_train <- R_all[1:(length(R_all)-N_test)]
imp_train <- imp_all[1:(length(R_all)-N_test),]
R_test <- R_all[(length(R_all)-N_test+1):length(R_all)]
imp_test <- imp_all[(length(R_all)-N_test+1):length(R_all),]

train_data <- cbind(R_train,imp_train)
test_data <- cbind(R_test,imp_test)
train_data <- scale(train_data,center=TRUE,scale=TRUE)
```

The implied volatility is not normal, but the χ^2 test of Jarque Bera test, including the skewness and kurtosis is still better than the call price surface. Thus, IV is used for the following models.

```r
# IV data
imp_train_norm <- scale(imp_train,center=TRUE,scale=TRUE)

# Test whether the IV data is normal or not
Num_obs <- 10
qqnorm(imp_train_norm[,Num_obs],ylim=c(-3,3))
qqline(imp_train_norm[,Num_obs],lwd = 2)
```




```
hist(imp_train_norm[,Num_obs])
```




```
# Jarque Bera test for normality
jarque.bera.test(imp_train_norm[,Num_obs])
```

```
##
```

```
##  Jarque Bera Test
##
## data:  imp_train_norm[, Num_obs]
## X-squared = 179.05, df = 2, p-value < 2.2e-16
```

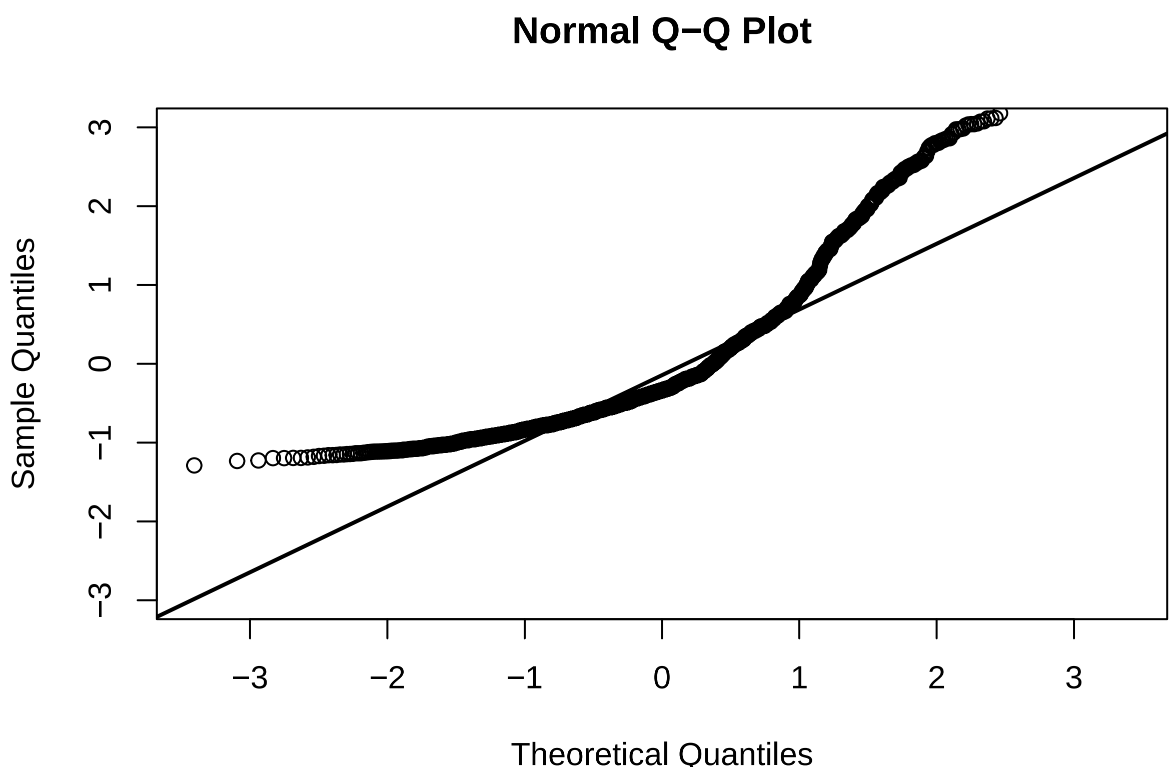
```r
# Find skewness and kurtosis
skew_imp <- apply(imp_train_norm,2,skewness)
kurt_imp <- apply(imp_train_norm,2,kurtosis)
sprintf('Mean of skewness (IV) is %f',mean(skew_imp))
```

```
## [1] "Mean of skewness (IV) is 1.013224"
```

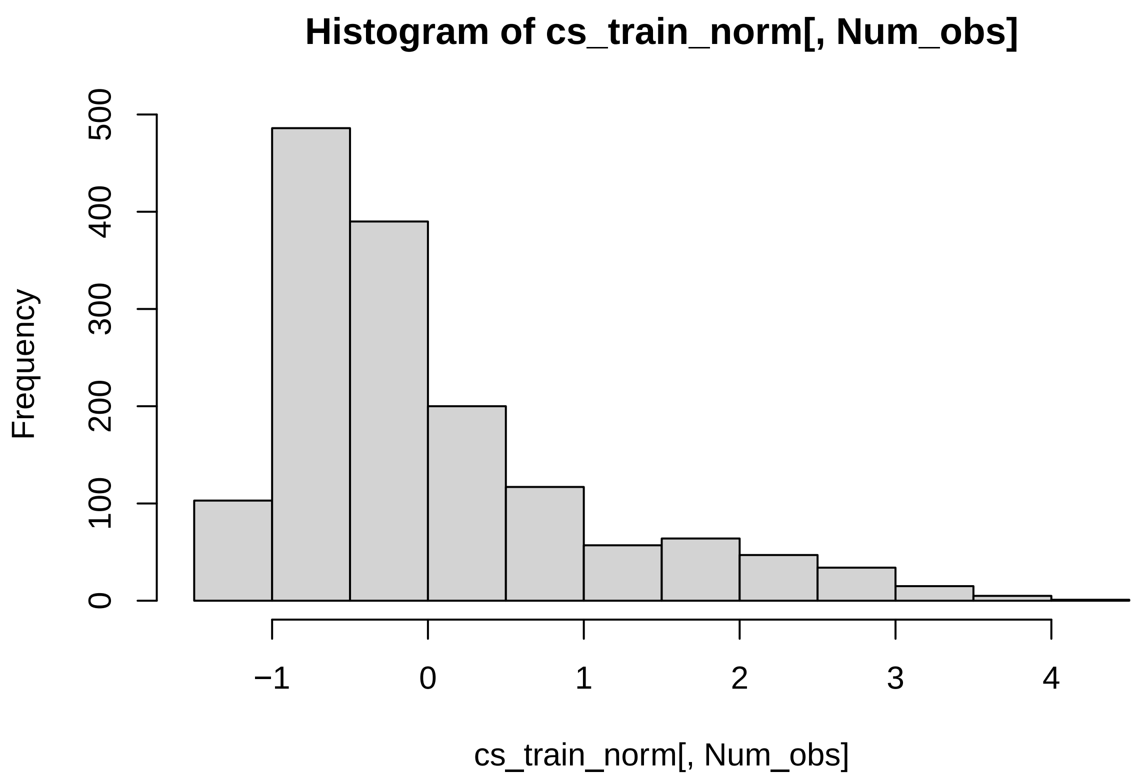
```r
sprintf('Mean of kurtosis (IV) is %f',mean(kurt_imp))
```

```
## [1] "Mean of kurtosis (IV) is 3.627883"
```

```r
# Import call price surface data
Data_all_cs <- read.csv("/Users/benjye/Dropbox/Pricing/Data_R/Date_all.csv",
                     header=FALSE, stringsAsFactors = FALSE)
start <- 1767
Date_all_cs <- Data_all_cs[start:nrow(Data_all_cs),1]
S_all_cs <- Data_all_cs[start:nrow(Data_all_cs),2]
cs_all <- Data_all_cs[start:nrow(Data_all_cs),-c(1,2)]

R_all_cs <- diff(log(S_all_cs))
R_all_cs <- R_all_cs[-1]
cs_all <- cs_all[2:nrow(cs_all),]

N_test <- 1000
R_train_cs <- R_all_cs[1:(length(R_all_cs)-N_test)]
cs_train <- cs_all[1:(length(R_all_cs)-N_test),]
R_test_cs <- R_all_cs[(length(R_all_cs)-N_test+1):length(R_all_cs)]
cs_test <- cs_all[(length(R_all_cs)-N_test+1):length(R_all_cs),]


cs_train_norm <- scale(cs_train,center=TRUE,scale=TRUE)
qqnorm(cs_train_norm[,Num_obs],ylim=c(-3,3))
qqline(cs_train_norm[,Num_obs],lwd = 2)
```




```
hist(cs_train_norm[,Num_obs])
```




```
# Jarque Bera test, skewness and kurtosis from call price surface
jarque.bera.test(cs_train_norm[,Num_obs])
```

```
##
```

```
##  Jarque Bera Test
##
## data:  cs_train_norm[, Num_obs]
## X-squared = 665.16, df = 2, p-value < 2.2e-16
```

```r
skew_cs <- apply(cs_train_norm,2,skewness)
kurt_cs <- apply(cs_train_norm,2,kurtosis)
sprintf('Mean of skewness (CS) is %f',mean(skew_cs))
```

```
## [1] "Mean of skewness (CS) is 1.292788"
```

```r
sprintf('Mean of kurtosis (CS) is %f',mean(kurt_cs))
```

```
## [1] "Mean of kurtosis (CS) is 4.370945"
```

```r
# Difference between chi^2 test of IV and CS
jb_imp <- c()
jb_cs <- c()
for(i in 1:130){
  jb_imp <- rbind(jb_imp,jarque.bera.test(imp_train_norm[,i])$statistic)
  jb_cs <- rbind(jb_cs,jarque.bera.test(cs_train_norm[,i])$statistic)
}
sprintf('Difference between chi^2 test of IV and CS is %f',mean(jb_imp-jb_cs))
```

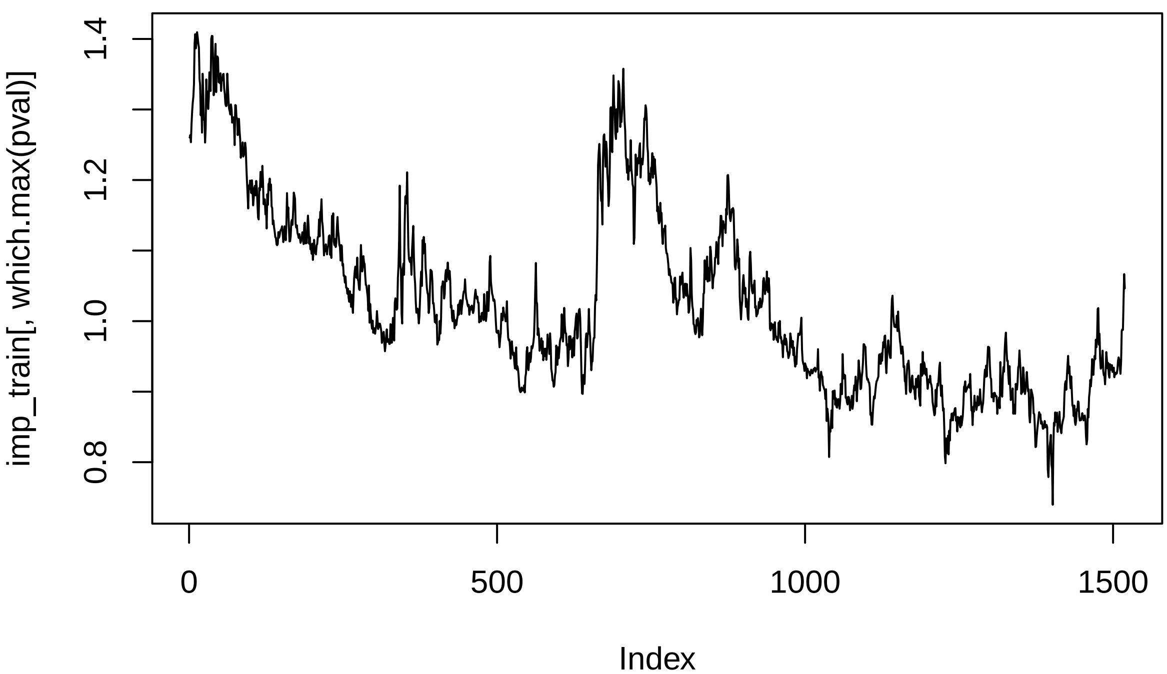
```
## [1] "Difference between chi^2 test of IV and CS is -248.533908"
```

Using Augmented Dickey Fuller test, we conclude that the series are stationary under 90% confidence.

```r
# ADF test for IV training data
pval <- sapply(1:130,function(x) adf.test(imp_train[,x])$p.value) # p-value from ADF
pval[which.max(pval)]
```

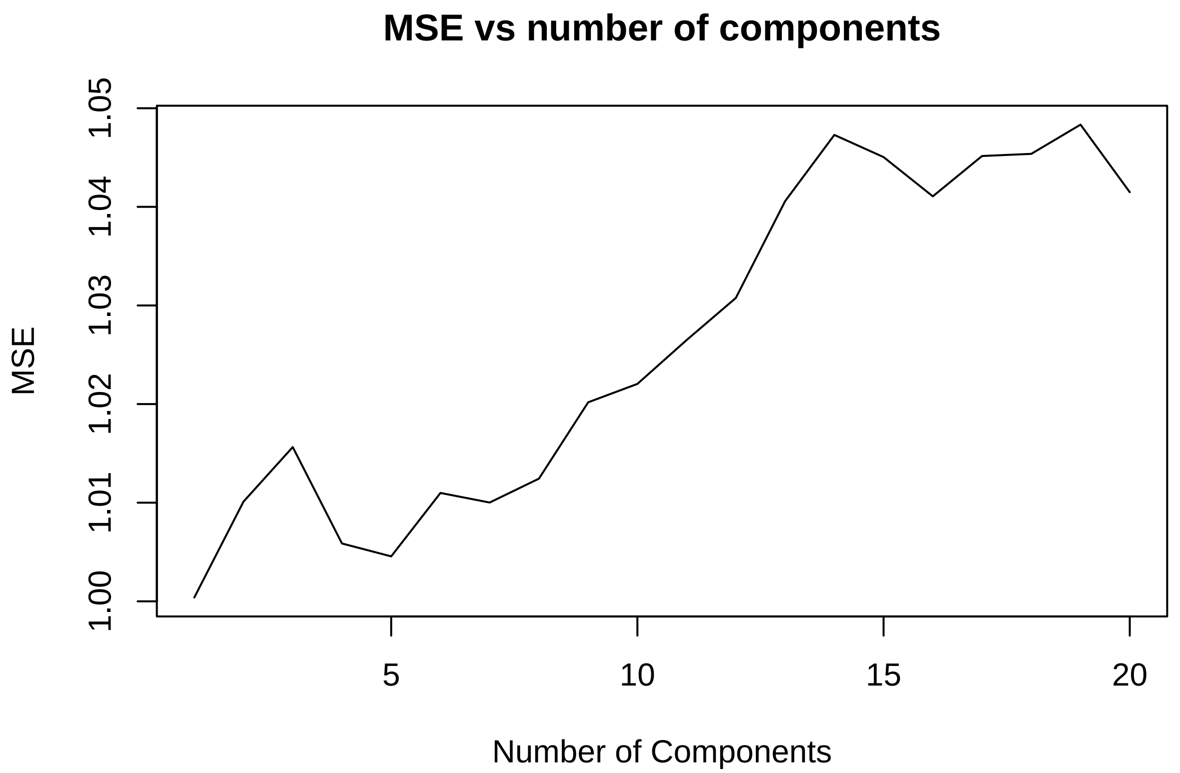
```
## [1] 0.09155252
```

```r
# Plot the most unstationary series
plot(imp_train[,which.max(pval)],type='l')
```



We fit PLS model for R, and by selecting the lowest MSE, 1 component is the best result. However, in scale factors, the third component in 3-component model contains the most information compared to PCR. Thus, we use the optimal component as 3.

```
# Fit PLS model on R
fit <- plsr(formula = R_train~., data=data.frame(train_data), rescale = F, validation="CV",segment.type=

fit.cv <- pls::crossval(fit, segments = 10,segment.type = c("consecutive"))
mse_pls <- MSEP(fit.cv)

plot(c(1:20),mse_pls$val[2,1,2:21],'l',xlab='Number of Components',ylab='MSE'
     ,main='MSE vs number of components')
```




```
# Find the number of components with minimum MSE
which.min(mse_pls$val[2,1,])
```

```
## 1 comps
##       2
```

```
# Plot scale factors for PLS of R
V <- t(as.matrix(train_data[,2:ncol(train_data)])) %*%
  as.matrix(train_data[,2:ncol(train_data)])
e <- eigen(V)
alpha <- apply(diag(train_data[,1]) %*%
                 as.matrix(train_data[,2:ncol(train_data)]) %*% e$vectors, 2, mean) / e$values
#K: component numbers
f_PLS_1 <- c()
for(K in 1:5){
  w <- sapply(1:K, function(k) sum(alpha^2*e$values^(k+1)))
  W <- matrix(0, K, K)
  for(l in 1:K)
```

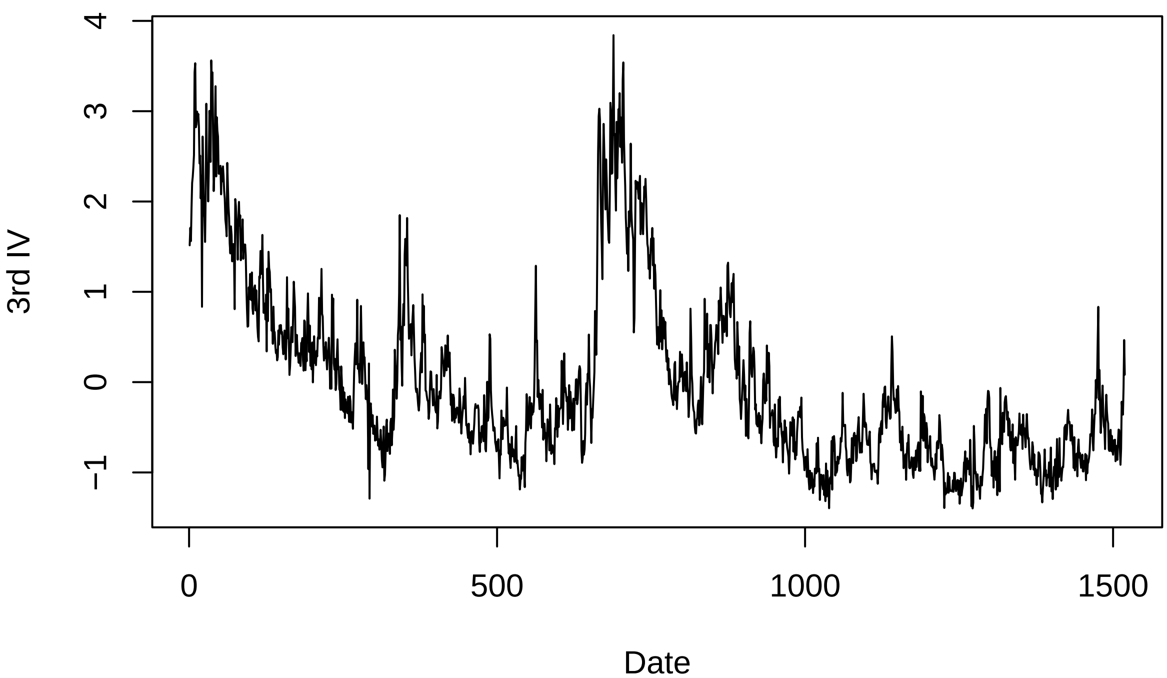
```r
  {
    W[,l] <- sapply(1:K, function(k) sum(alpha^2*e$values^(k+l+1)))
  }
  beta <- solve(W, w,tol = 1e-200)
  temp <- sapply(1:130, function(j) sum(beta*(e$values[j])^(1:K)))
  f_PLS_1 <- rbind(f_PLS_1,temp)
}

par(mfrow=c(2,2))
dev.off()
```

```
## null device
##           1
```

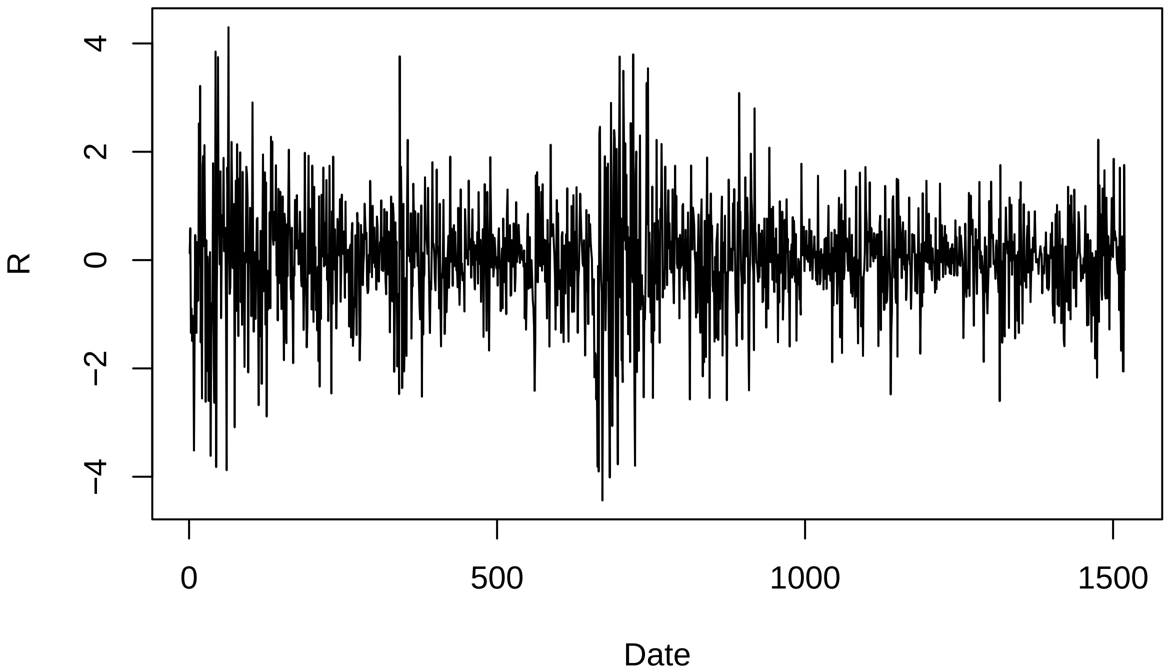
```r
ms <- c(0.947,0.960,0.971,0.979,0.987,0.995,1.001,1.007,1.014,1.021)
plot(1:10,f_PLS_1[1,1:10],'l',xlab='ms',ylab='scale factor',ylim=c(-5,5),main='scale factor (tau = 0.08
lines(f_PLS_1[2,1:10],col='blue')
lines(f_PLS_1[3,1:10],col='green')
lines(f_PLS_1[4,1:10],col='red')
lines(f_PLS_1[5,1:10],col='brown')
abline(h=1,lty='dashed')
legend('bottomright',lty = c(1,1,1,1,1),legend=c("1 component", "2 components","3 components","4 compone
axis(1,at=1:10,label=ms)
```

The 3rd component has large value when the return R has large volatility, and thus, a better capture of the return behavior.

```r
# Plot the 3rd component in the training IV
plot(train_data[,4],type='l',xlab='Date',ylab='3rd IV')
```




```r
# plot the return
plot(train_data[,1],type='l',xlab='Date',ylab='R')
```




```
# Fit PLS of 3 comp
N_comp <- 3
fit_new <- plsr(formula = R_train~., data=data.frame(train_data),ncomp=N_comp, rescale = F, validation=

# Find PLS factor of training data
V_pls_train <- as.matrix(fit_new$scores)%*%diag(N_comp)

# Find PLS factor of test data
P_pls <- fit_new$projection

## Normalize test data
imp_norm <- (imp_test - t(t(rep(1,nrow(imp_test))))%*%t(apply(imp_train,2,mean)))/(t(t(rep(1,nrow(imp_te
imp_norm <- imp_norm-rep(1,dim(imp_norm)[1])%*%t(fit_new$Xmeans)

V_pls_test <- as.matrix(imp_norm)%*%as.matrix(P_pls)
```

The PLS factor in training data is in general stationary, except factor 2.

```
# Decide whether the PLS factors are stationary or not
for(i in 1:N_comp){
  print(adf.test(V_pls_train[,i]))
}
```

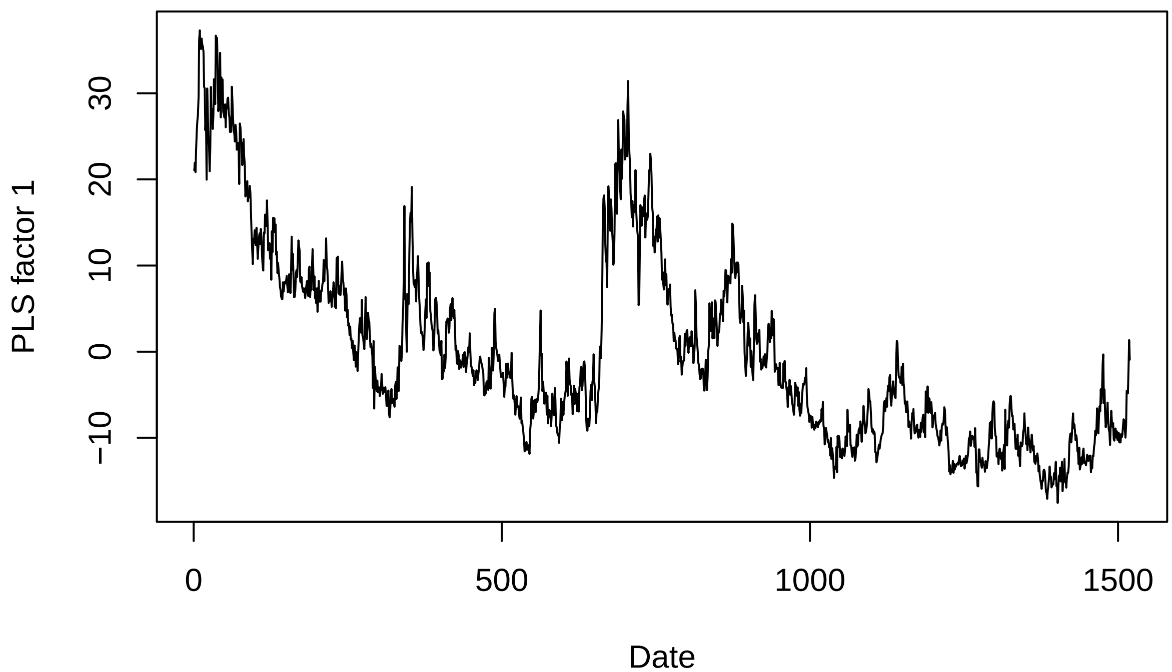
```
##
##  Augmented Dickey-Fuller Test
##
## data:  V_pls_train[, i]
## Dickey-Fuller = -3.4803, Lag order = 11, p-value = 0.04419
## alternative hypothesis: stationary
##
##
##  Augmented Dickey-Fuller Test
##
## data:  V_pls_train[, i]
```

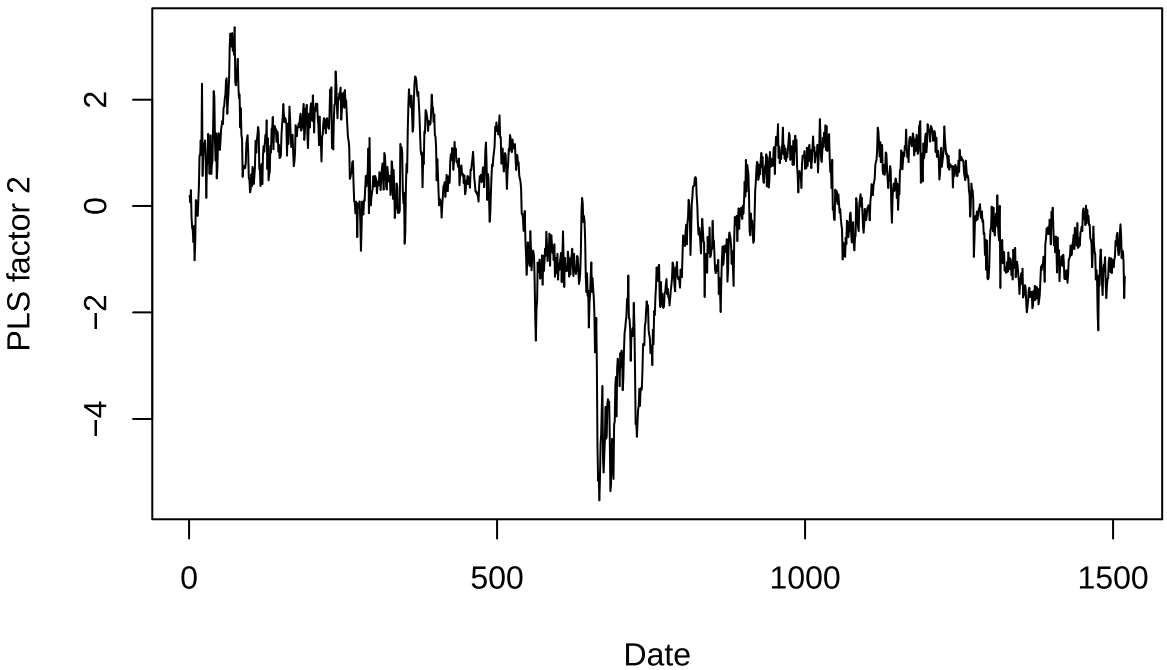
```
## Dickey-Fuller = -3.0895, Lag order = 11, p-value = 0.1171
## alternative hypothesis: stationary
```

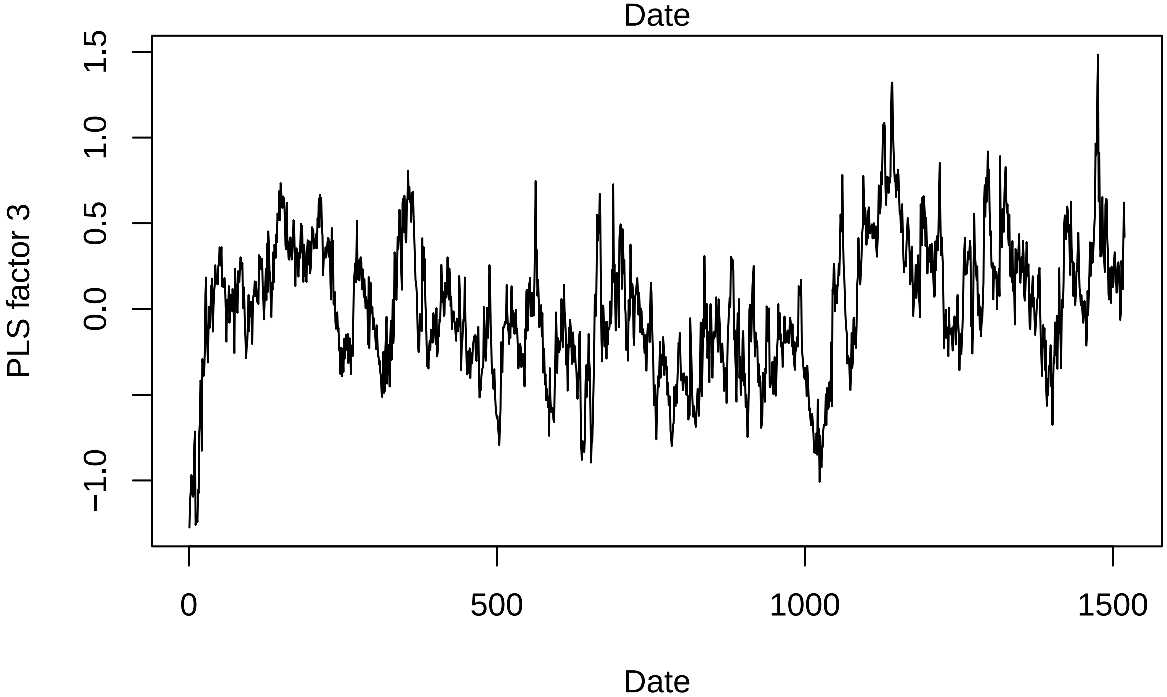
```
## Warning in adf.test(V_pls_train[, i]): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  V_pls_train[, i]
## Dickey-Fuller = -5.584, Lag order = 11, p-value = 0.01
## alternative hypothesis: stationary
```

```
# Plot the training PLS factor
for(i in 1:N_comp){
  plot(V_pls_train[,i],type='l',xlab='Date',ylab=paste0("PLS factor ",i))
}
```





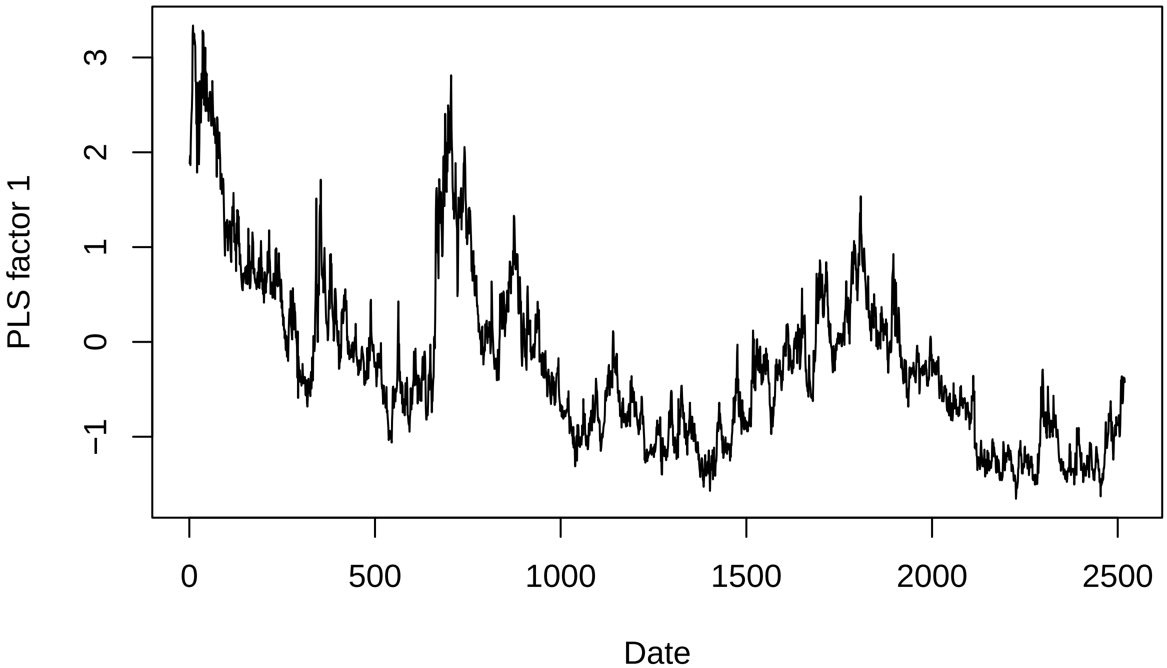


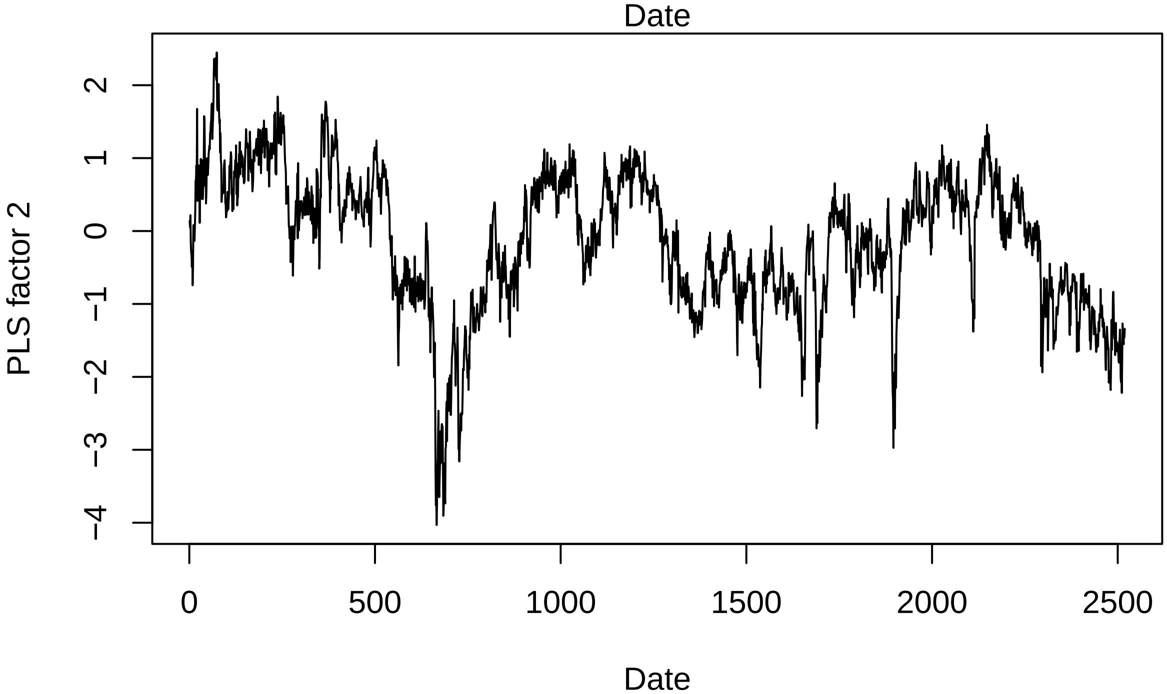

```r
# Normalize PLS factors in training and test data
V_train_norm <- scale(V_pls_train,center=TRUE,scale=TRUE)
V_test_norm <- (V_pls_test - t(t(rep(1,nrow(V_pls_test))))%*%t(apply(V_pls_train,2,mean)))/(t(t(rep(1,n

# Combine traing and test data
V_norm <- rbind(V_train_norm,V_test_norm)
```

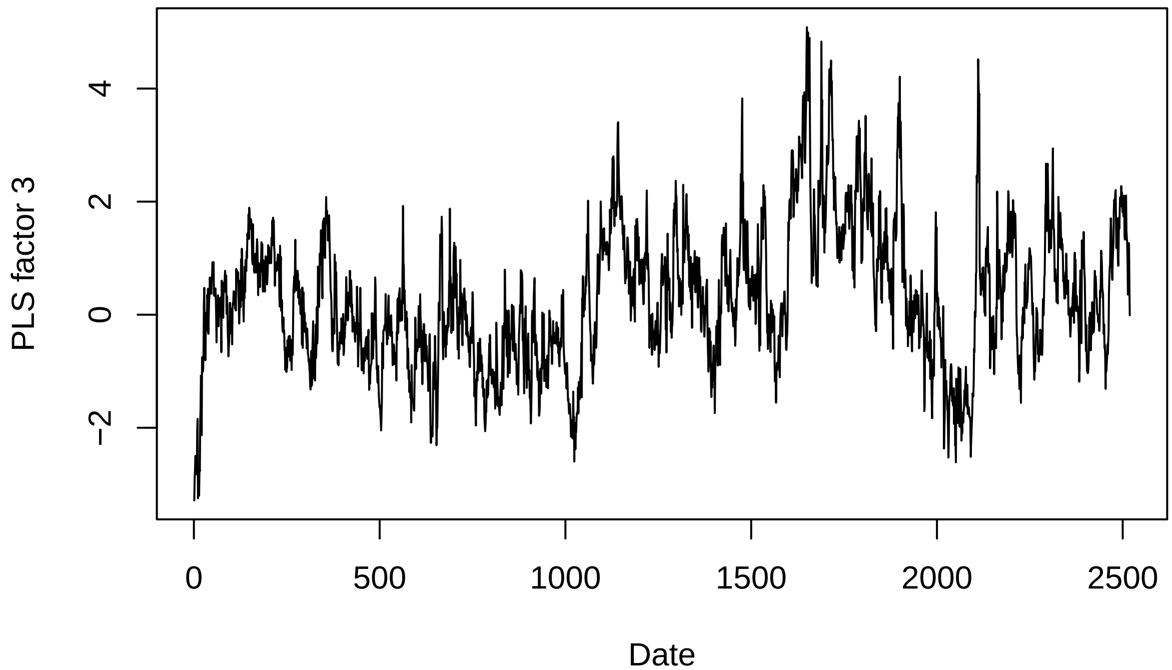
However, the normalized training + test data is stationary according to ADF test.

```r
# Plot the normalized PLS factor (training+test)
for(i in 1:N_comp){
  plot(V_norm[,i],type='l',xlab='Date',ylab=paste0("PLS factor ",i))
```

```
}
```


PLS factor 1
3
2
1
0
−1
0
500
1000
1500
2000
2500
Date


PLS factor 2
2
1
0
−1
−2
−3
−4
0
500
1000
1500
2000
2500
Date




```
# ADF test for training+test data
adf.test(V_norm[,1],k=10)
```

```
## Warning in adf.test(V_norm[, 1], k = 10): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  V_norm[, 1]
## Dickey-Fuller = -4.1158, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(V_norm[,2],k=10)
```

```
## Warning in adf.test(V_norm[, 2], k = 10): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  V_norm[, 2]
## Dickey-Fuller = -4.5175, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(V_norm[,3],k=10)
```

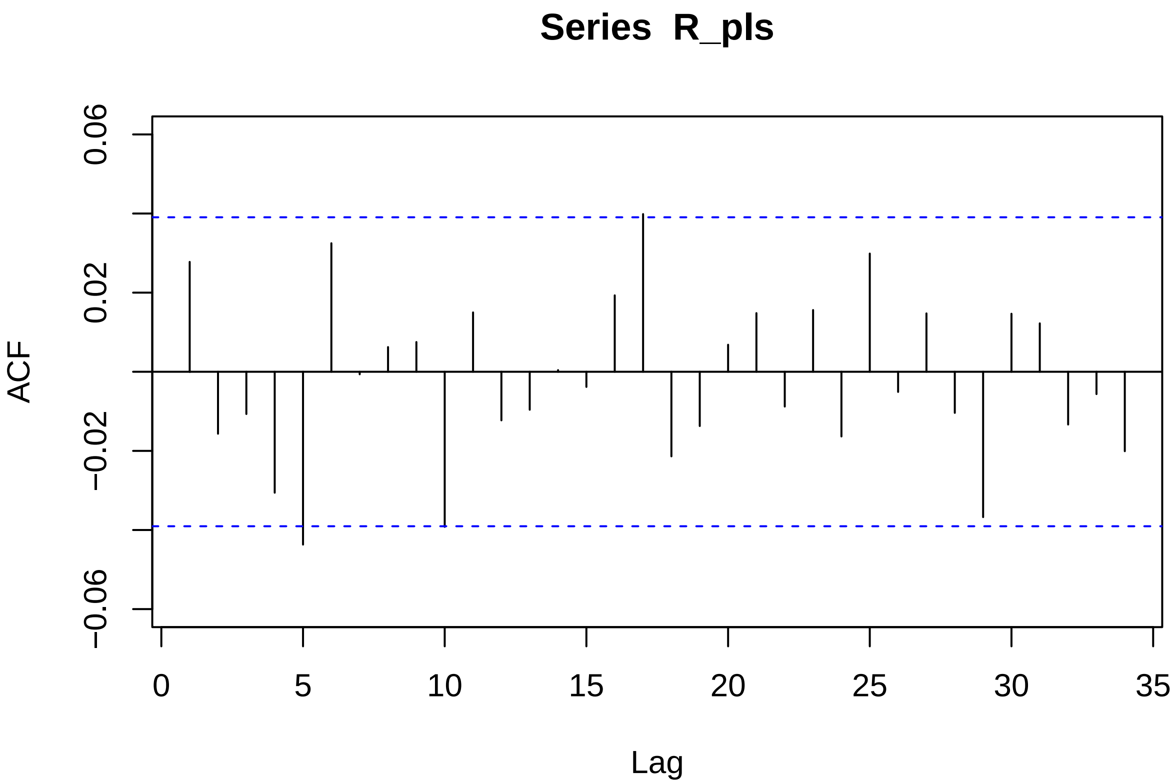
```
## Warning in adf.test(V_norm[, 3], k = 10): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  V_norm[, 3]
## Dickey-Fuller = -6.9637, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

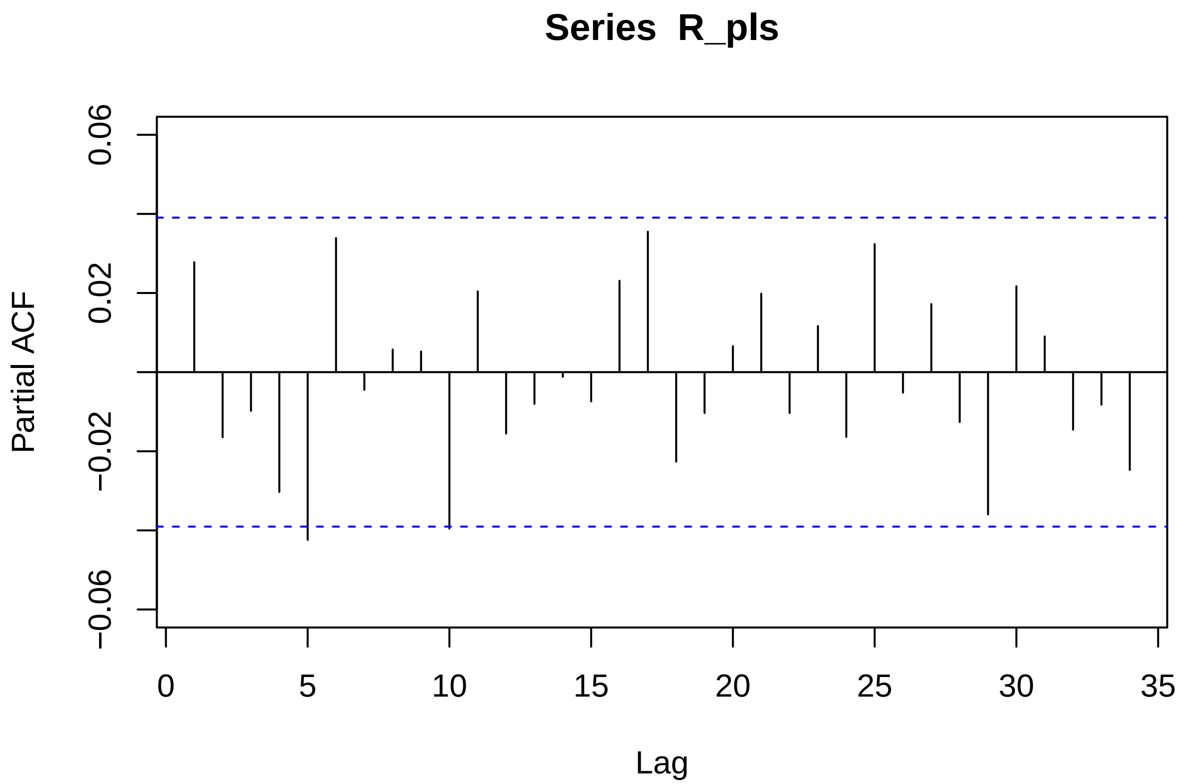
```
# Prepare R with normalized training + test data
R_train_norm <- scale(R_train,center=TRUE,scale=TRUE)
```

```
R_test_norm <- (R_test-mean(R_train))/sd(R_train)
R_pls <- c(R_train_norm,R_test_norm)
```

```
Acf(R_pls)
```




```
Pacf(R_pls)
```



First, the base model is ARMA(0,0)+GARCH(0,1) with external factors as the normalized PLS factors. The coefficients of PLS factors are all insignificant. In-sample MSE is 0.0001836, while out-of-sample MSE is 0.000137. The residuals are independent and heteroscedastic, but not normal. It passes the coverage test with the exceed of 6.2%.

```
# GARCH model with PLS factors of R
spec.sGARCH_pls <- ugarchspec(variance.model=list(model="sGARCH",
                              garchOrder=c(0,1),external.regressors = as.matrix(V_norm)),
                              mean.model=list(armaOrder = c(0,0),include.mean=TRUE,
                                              external.regressors = as.matrix(V_norm)),
                              distribution.model="norm")
sGARCH_pls <- ugarchfit(R_pls, spec=spec.sGARCH_pls,out.sample=N_test)
sGARCH_pls
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(0,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error    t value Pr(>|t|)
## mu      0.000000    0.025249 0.0000e+00 1.000000
## mxreg1  0.040878    0.025785 1.5853e+00 0.112890
```

```
## mxreg2  0.027749    0.025476 1.0892e+00 0.276068
## mxreg3  0.068417    0.025090 2.7268e+00 0.006394
## omega   0.000158    0.000055 2.8506e+00 0.004364
## beta1   0.999602    0.000001 1.7001e+06 0.000000
## vxreg1  0.000000    0.000196 5.2000e-05 0.999959
## vxreg2  0.000000    0.000136 7.3000e-05 0.999941
## vxreg3  0.000000    0.000432 2.3000e-05 0.999982
##
## Robust Standard Errors:
##         Estimate  Std. Error    t value Pr(>|t|)
## mu      0.000000    0.029995 0.0000e+00 1.000000
## mxreg1  0.040878    0.039379 1.0381e+00 0.299243
## mxreg2  0.027749    0.036951 7.5096e-01 0.452678
## mxreg3  0.068417    0.030036 2.2778e+00 0.022737
## omega   0.000158    0.000132 1.1924e+00 0.233094
## beta1   0.999602    0.000001 7.2720e+05 0.000000
## vxreg1  0.000000    0.000371 2.7000e-05 0.999978
## vxreg2  0.000000    0.000308 3.2000e-05 0.999974
## vxreg3  0.000000    0.000922 1.1000e-05 0.999991
##
## LogLikelihood : -2135.419
##
## Information Criteria
## ------------------------------------
##
## Akaike       2.8235
## Bayes        2.8550
## Shibata      2.8234
## Hannan-Quinn 2.8352
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      3.343 0.06748
## Lag[2*(p+q)+(p+q)-1][2]     3.633 0.09467
## Lag[4*(p+q)+(p+q)-1][5]     4.761 0.17320
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic   p-value
## Lag[1]                      22.32 2.305e-06
## Lag[2*(p+q)+(p+q)-1][2]     54.01 5.551e-15
## Lag[4*(p+q)+(p+q)-1][5]    158.73 0.000e+00
## d.o.f=1
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale   P-Value
## ARCH Lag[2]     63.22 0.500 2.000 1.887e-15
## ARCH Lag[4]    159.70 1.397 1.611 0.000e+00
## ARCH Lag[6]    213.47 2.222 1.500 0.000e+00
##
```

```
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  18.027
## Individual Statistics:
## mu     0.1384
## mxreg1 0.1010
## mxreg2 0.0489
## mxreg3 0.1116
## omega  2.4193
## beta1  2.5456
## vxreg1 4.3082
## vxreg2 1.1234
## vxreg3 4.7149
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.1 2.32 2.82
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                     t-value      prob sig
## Sign Bias             0.245 8.065e-01
## Negative Sign Bias    4.949 8.310e-07 ***
## Positive Sign Bias    1.621 1.053e-01
## Joint Effect         35.518 9.471e-08 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     99.18    7.527e-13
## 2    30    124.68    7.863e-14
## 3    40    132.80    3.707e-12
## 4    50    157.53    2.552e-13
##
##
## Elapsed time : 0.2007349
```

```r
# In-sample MSE
sprintf("In-sample MSE is %g",mean((sGARCH_pls@fit$residuals*sd(R_train))^2))
```

```
## [1] "In-sample MSE is 0.000183648"
```

```r
# Out-of-sample
forecast_sGARCH_pls<-ugarchforecast(sGARCH_pls, data = R_pls, n.ahead = 1, n.roll = N_test,out.sample =
sigma_sGARCH_pls<-sigma(forecast_sGARCH_pls)
fitted_sGARCH_pls<-fitted(forecast_sGARCH_pls)
sprintf("Out-of-sample MSE is %g",mean(((t(fitted_sGARCH_pls)-R_pls[(length(R_pls)-N_test):length(R_pls
```

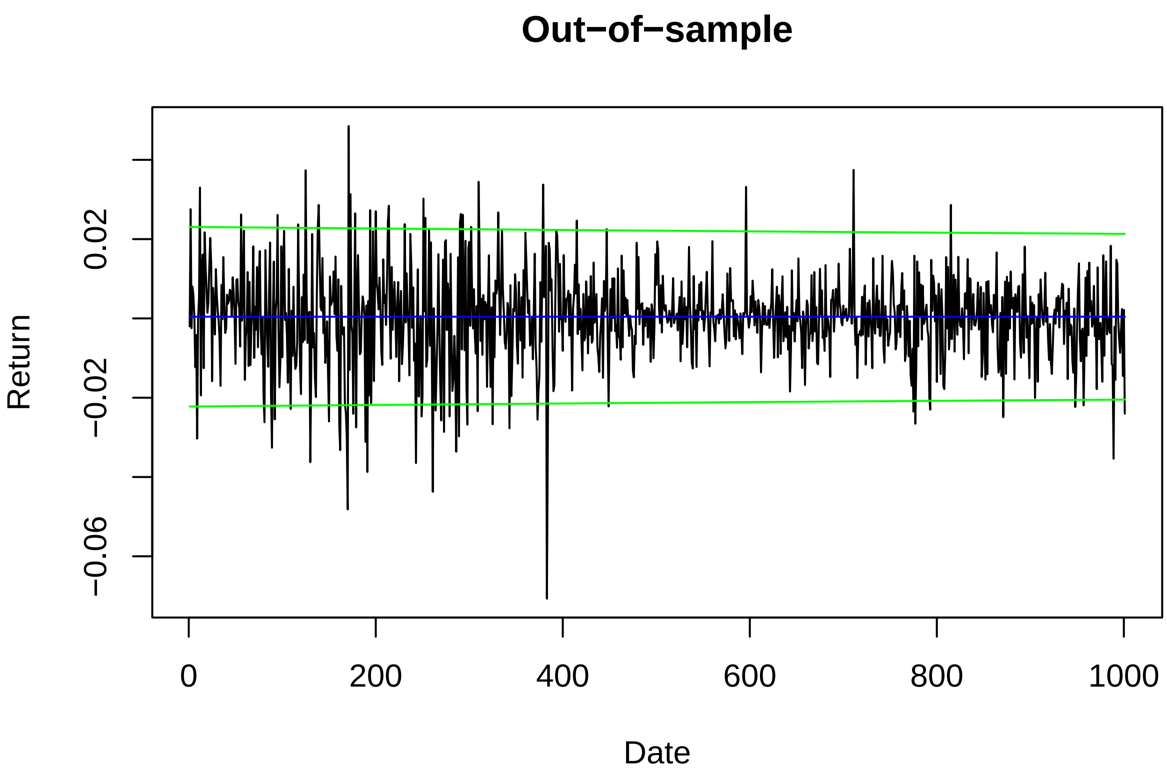
```
## [1] "Out-of-sample MSE is 0.000136734"
```

```r
sprintf("Out-of-sample mean of sd is %g",mean(sigma_sGARCH_pls))
```

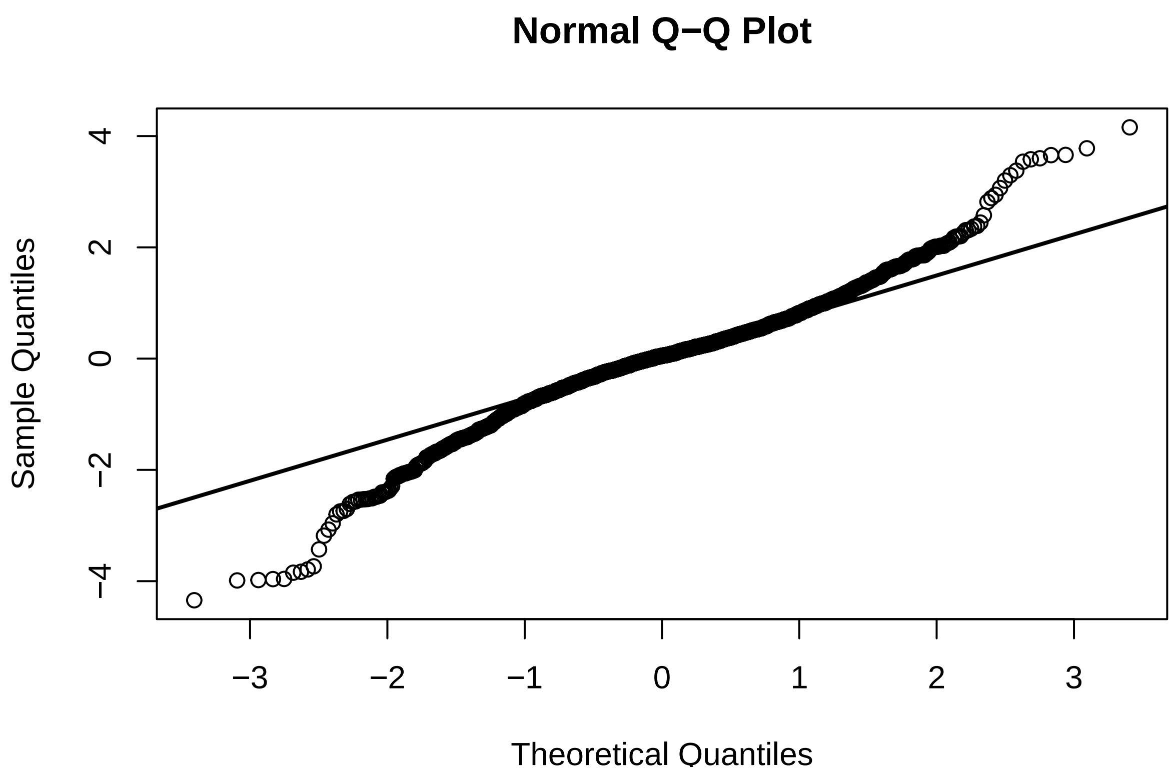
```
## [1] "Out-of-sample mean of sd is 0.814912"
```

```r
# 95% CI
plot(R_all[(length(R_all)-N_test):length(R_all)],type='l',xlab='Date',ylab='Return',main="Out-of-sample
```

```
lines(t(fitted_sGARCH_pls)*sd(R_train)+mean(R_train),col='blue')
lines(t(fitted_sGARCH_pls)*sd(R_train)+mean(R_train)+1.96*t(sigma_sGARCH_pls)*sd(R_train),col='green')
lines(t(fitted_sGARCH_pls)*sd(R_train)+mean(R_train)-1.96*t(sigma_sGARCH_pls)*sd(R_train),col='green')
```




```
# QQplot, Normality test
qqnorm(sGARCH_pls@fit$residuals)
qqline(sGARCH_pls@fit$residuals,lwd = 2)
```




```
jarque.bera.test(sGARCH_pls@fit$residuals)
```

```
##
##  Jarque Bera Test
##
## data:  sGARCH_pls@fit$residuals
## X-squared = 342.26, df = 2, p-value < 2.2e-16
```

```
# Autocorrelation test, heteroscedasticity
Box.test(sGARCH_pls@fit$residuals, lag = 10, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  sGARCH_pls@fit$residuals
## X-squared = 8.3178, df = 10, p-value = 0.5978
```

```
Box.test(sGARCH_pls@fit$residuals^2, lag = 10, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  sGARCH_pls@fit$residuals^2
## X-squared = 509.37, df = 10, p-value < 2.2e-16
```

```
# Coverage test
roll_sGARCH_pls<-ugarchroll(spec=spec.sGARCH_pls, data=R_pls, n.ahead=1, forecast.length=N_test, refit.
report(roll_sGARCH_pls, type="VaR", VaR.alpha = 0.05, conf.level = 0.95)
```

```
## VaR Backtest Report
## ===========================================
## Model:              sGARCH-norm
```

```
## Backtest Length: 1000
## Data:
##
## ==========================================
## alpha:               5%
## Expected Exceed: 50
## Actual VaR Exceed:   62
## Actual %:            6.2%
##
## Unconditional Coverage (Kupiec)
## Null-Hypothesis: Correct Exceedances
## LR.uc Statistic: 2.826
## LR.uc Critical:      3.841
## LR.uc p-value:       0.093
## Reject Null:     NO
##
## Conditional Coverage (Christoffersen)
## Null-Hypothesis: Correct Exceedances and
##                  Independence of Failures
## LR.cc Statistic: 13.807
## LR.cc Critical:      5.991
## LR.cc p-value:       0.001
## Reject Null:     YES
```

Next, the external regressors include the lagged terms of the latent factors.

```
# Create lag terms
N_train <- nrow(V_norm)
lags <- 2
X_train_new <- V_norm[(lags+1):N_train,]
for(i in 1:lags){
  if(lags==0){
    break
  }else{
    temp <- V_norm[(lags+1-i):(N_train-i),]
    X_train_new <- cbind(X_train_new,temp)
  }
}

R_pls1 <- R_pls[(lags+1):N_train]


# eGARCH
spec.eGARCH_pls <- ugarchspec(variance.model=list(model="eGARCH",
                            garchOrder=c(2,2),external.regressors = as.matrix(X_train_new)),
                            mean.model=list(armaOrder = c(1,1),include.mean=FALSE,
                                            external.regressors = as.matrix(X_train_new)),
                            distribution.model="ged")
eGARCH_pls <- ugarchfit(R_pls1, spec=spec.eGARCH_pls,out.sample=N_test)
eGARCH_pls
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
```

```
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : eGARCH(2,2)
## Mean Model   : ARFIMA(1,0,1)
## Distribution : ged
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    -0.472161    0.066559 -7.09385 0.000000
## ma1     0.520925    0.068948  7.55536 0.000000
## mxreg1  0.321313    0.062608  5.13217 0.000000
## mxreg2  0.057110    0.307573  0.18568 0.852697
## mxreg3  0.008014    0.069383  0.11550 0.908050
## mxreg4 -0.594749    0.068822 -8.64185 0.000000
## mxreg5 -0.234870    0.101400 -2.31628 0.020543
## mxreg6  0.017626    0.051854  0.33992 0.733921
## mxreg7  0.288246    0.055476  5.19585 0.000000
## mxreg8  0.184253    0.264258  0.69725 0.485648
## mxreg9  0.033365    0.040303  0.82785 0.407753
## omega  -0.045525    0.012675 -3.59180 0.000328
## alpha1 -0.305552    0.056337 -5.42363 0.000000
## alpha2  0.123210    0.055359  2.22567 0.026036
## beta1   0.997009    0.035048 28.44700 0.000000
## beta2  -0.132698    0.018273 -7.26185 0.000000
## gamma1 -0.156079    0.063321 -2.46488 0.013706
## gamma2  0.196193    0.064405  3.04626 0.002317
## vxreg1 -0.666534    0.175005 -3.80866 0.000140
## vxreg2 -0.387692    0.226746 -1.70981 0.087301
## vxreg3  0.129820    0.136908  0.94823 0.343012
## vxreg4  1.442508    0.202391  7.12732 0.000000
## vxreg5  0.734299    0.424851  1.72837 0.083922
## vxreg6 -0.118806    0.253164 -0.46929 0.638866
## vxreg7 -0.695695    0.159373 -4.36520 0.000013
## vxreg8 -0.372588    0.212619 -1.75238 0.079709
## vxreg9 -0.011228    0.136695 -0.08214 0.934535
## shape   1.511706    0.080429 18.79560 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error    t value Pr(>|t|)
## ar1    -0.472161    0.096131  -4.911657 0.000001
## ma1     0.520925    0.069189   7.528990 0.000000
## mxreg1  0.321313    0.066413   4.838088 0.000001
## mxreg2  0.057110    0.852465   0.066994 0.946587
## mxreg3  0.008014    0.258675   0.030980 0.975286
## mxreg4 -0.594749    0.027256 -21.821044 0.000000
## mxreg5 -0.234870    0.131294  -1.788889 0.073633
## mxreg6  0.017626    0.111485   0.158102 0.874376
## mxreg7  0.288246    0.094990   3.034479 0.002410
## mxreg8  0.184253    0.722132   0.255151 0.798606
## mxreg9  0.033365    0.179812   0.185556 0.852793
## omega  -0.045525    0.013453  -3.384064 0.000714
## alpha1 -0.305552    0.064451  -4.740851 0.000002
```

```
## alpha2  0.123210    0.059352   2.075914 0.037902
## beta1   0.997009    0.028259  35.280646 0.000000
## beta2  -0.132698    0.041830  -3.172320 0.001512
## gamma1 -0.156079    0.064336  -2.426003 0.015266
## gamma2  0.196193    0.063008   3.113780 0.001847
## vxreg1 -0.666534    0.216101  -3.084362 0.002040
## vxreg2 -0.387692    0.220282  -1.759980 0.078411
## vxreg3  0.129820    0.148798   0.872457 0.382959
## vxreg4  1.442508    0.065793  21.924846 0.000000
## vxreg5  0.734299    0.425276   1.726644 0.084232
## vxreg6 -0.118806    0.243847  -0.487215 0.626106
## vxreg7 -0.695695    0.192771  -3.608916 0.000307
## vxreg8 -0.372588    0.317309  -1.174212 0.240310
## vxreg9 -0.011228    0.123537  -0.090889 0.927581
## shape   1.511706    0.075779  19.948765 0.000000
##
## LogLikelihood : -1890.826
##
## Information Criteria
## ------------------------------------
##
## Akaike       2.5298
## Bayes        2.6280
## Shibata      2.5291
## Hannan-Quinn 2.5664
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.05729  0.8108
## Lag[2*(p+q)+(p+q)-1][5]   0.63508  1.0000
## Lag[4*(p+q)+(p+q)-1][9]   1.32856  0.9986
## d.o.f=2
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                     0.03779  0.8459
## Lag[2*(p+q)+(p+q)-1][11]   1.22167  0.9927
## Lag[4*(p+q)+(p+q)-1][19]   3.60374  0.9871
## d.o.f=4
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[5]   0.03714 0.500 2.000  0.8472
## ARCH Lag[7]   0.42513 1.473 1.746  0.9170
## ARCH Lag[9]   0.45914 2.402 1.619  0.9867
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  no.parameters>20 (not available)
## Individual Statistics:
```

```
## ar1    0.11461
## ma1    0.11251
## mxreg1 0.03180
## mxreg2 0.10312
## mxreg3 0.14934
## mxreg4 0.02965
## mxreg5 0.09742
## mxreg6 0.11891
## mxreg7 0.02994
## mxreg8 0.13046
## mxreg9 0.12105
## omega  0.06523
## alpha1 0.07432
## alpha2 0.03661
## beta1  0.02899
## beta2  0.02740
## gamma1 0.08285
## gamma2 0.13264
## vxreg1 0.04381
## vxreg2 0.10326
## vxreg3 0.03562
## vxreg4 0.04839
## vxreg5 0.09504
## vxreg6 0.03970
## vxreg7 0.04990
## vxreg8 0.09372
## vxreg9 0.04211
## shape  0.31617
##
## Asymptotic Critical Values (10% 5% 1%)
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           1.1413 0.2539
## Negative Sign Bias  0.3388 0.7348
## Positive Sign Bias  1.2585 0.2084
## Joint Effect        2.6511 0.4486
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     23.66      0.20955
## 2    30     46.74      0.01979
## 3    40     50.38      0.10471
## 4    50     61.68      0.10562
##
##
## Elapsed time : 2.458295
```

```r
# In-sample MSE
sprintf('In-sample MSE is %g',mean((eGARCH_pls@fit$residuals*sd(R_train))^2))
```

```
## [1] "In-sample MSE is 0.000183703"
```

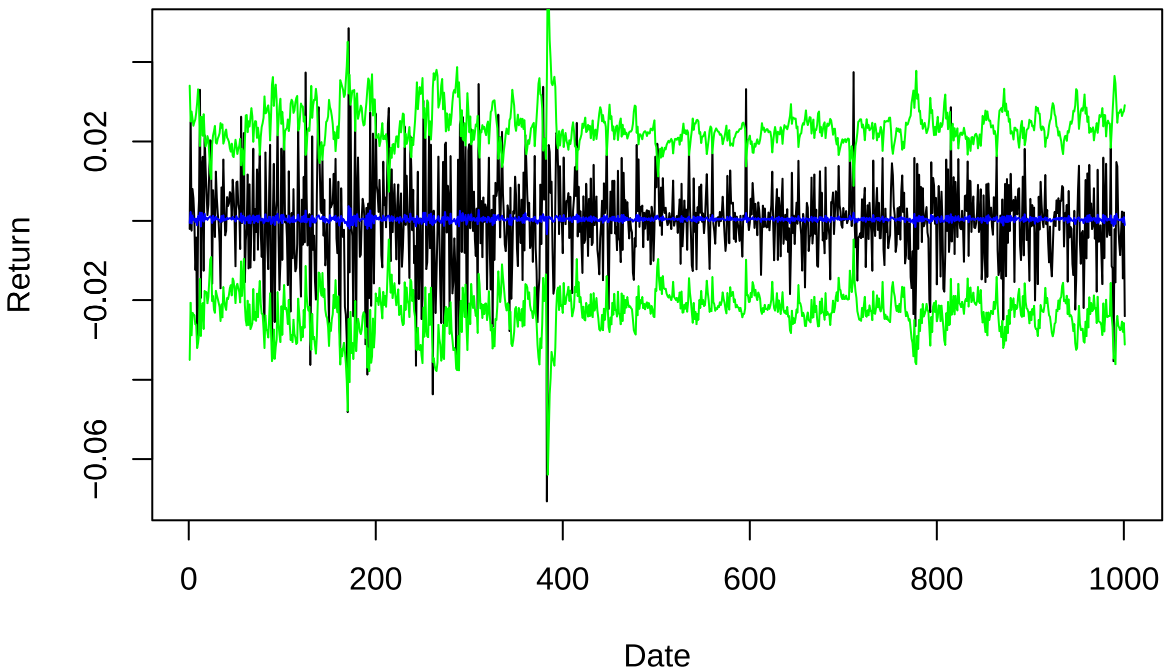
```r
# Out-of-sample
forecast_eGARCH_pls<-ugarchforecast(eGARCH_pls, data = R_pls1, n.ahead = 1, n.roll = N_test,out.sample =
sigma_eGARCH_pls<-sigma(forecast_eGARCH_pls)
fitted_eGARCH_pls<-fitted(forecast_eGARCH_pls)
sprintf('Out-of-sample MSE is %g',mean(((t(fitted_eGARCH_pls)-R_pls1[(length(R_pls1)-N_test):length(R_pl
```

```
## [1] "Out-of-sample MSE is 0.000123971"
```

```r
sprintf('Out-of-sample mean of sd is %g',mean(sigma_eGARCH_pls))
```

```
## [1] "Out-of-sample mean of sd is 0.881036"
```

```r
# 95% CI
plot(R_pls1[(length(R_pls1)-N_test):length(R_pls1)]*sd(R_train)+mean(R_train),type='l',ylab='Return',xla
lines(t(fitted_eGARCH_pls)*sd(R_train)+mean(R_train),col='blue')
lines(t(fitted_eGARCH_pls)*sd(R_train)+mean(R_train)+1.96*t(sigma_eGARCH_pls)*sd(R_train),col='green')
lines(t(fitted_eGARCH_pls)*sd(R_train)+mean(R_train)-1.96*t(sigma_eGARCH_pls)*sd(R_train),col='green')
```




```r
# Coverage Test
roll_GARCH_pls<-ugarchroll(spec=spec.eGARCH_pls, data=R_pls1, n.ahead=1, forecast.length=N_test, refit.
report(roll_GARCH_pls, type="VaR", VaR.alpha = 0.05, conf.level = 0.95)
```

```
## VaR Backtest Report
## ===========================================
## Model:              eGARCH-ged
## Backtest Length: 1000
## Data:
##
## ==========================================
## alpha:              5%
## Expected Exceed: 50
## Actual VaR Exceed:  57
## Actual %:           5.7%
```

```
##
## Unconditional Coverage (Kupiec)
## Null-Hypothesis: Correct Exceedances
## LR.uc Statistic: 0.989
## LR.uc Critical:      3.841
## LR.uc p-value:       0.32
## Reject Null:     NO
##
## Conditional Coverage (Christoffersen)
## Null-Hypothesis: Correct Exceedances and
##                  Independence of Failures
## LR.cc Statistic: 1.202
## LR.cc Critical:      5.991
## LR.cc p-value:       0.548
## Reject Null:     NO
```

```
# Autocorrelation and heteroscedasticity
Box.test(eGARCH_pls@fit$residuals, lag = 10, type = "Ljung")
```

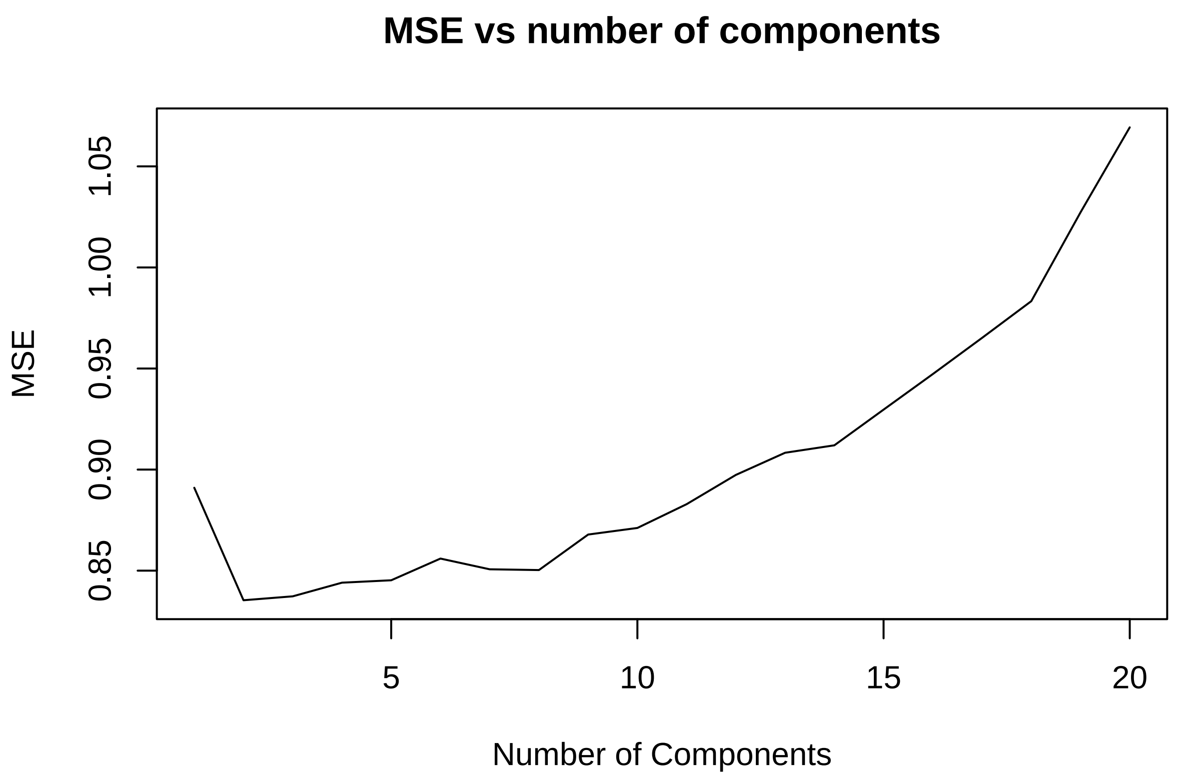
```
##
##  Box-Ljung test
##
## data:  eGARCH_pls@fit$residuals
## X-squared = 7.9468, df = 10, p-value = 0.634
```

```
Box.test(eGARCH_pls@fit$residuals^2, lag = 10, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  eGARCH_pls@fit$residuals^2
## X-squared = 502.52, df = 10, p-value < 2.2e-16
```

Based on MSE from cross-validation, the optimal number of PLS factors on R^2 is 2. However, the scale factors show that 5 components give more information than PCA.

```
# PLS on R^2
R_sq_train <- R_train^2
R_sq_test <- R_test^2
train_data_2 <- cbind(R_sq_train,imp_train)
test_data_2 <- cbind(R_sq_test,imp_test)
train_data_2 <- scale(train_data_2,center=TRUE,scale=TRUE)

fit2 <- plsr(formula = R_sq_train~., data=data.frame(train_data_2), rescale = F, validation="CV",segmen

fit2.cv <- pls::crossval(fit2, segments = 5,segment.type = c("consecutive"))
mse_pls2 <- MSEP(fit2.cv)
plot(c(1:20),mse_pls2$val[2,1,2:21],'l',xlab='Number of Components',ylab='MSE'
     ,main='MSE vs number of components')
```



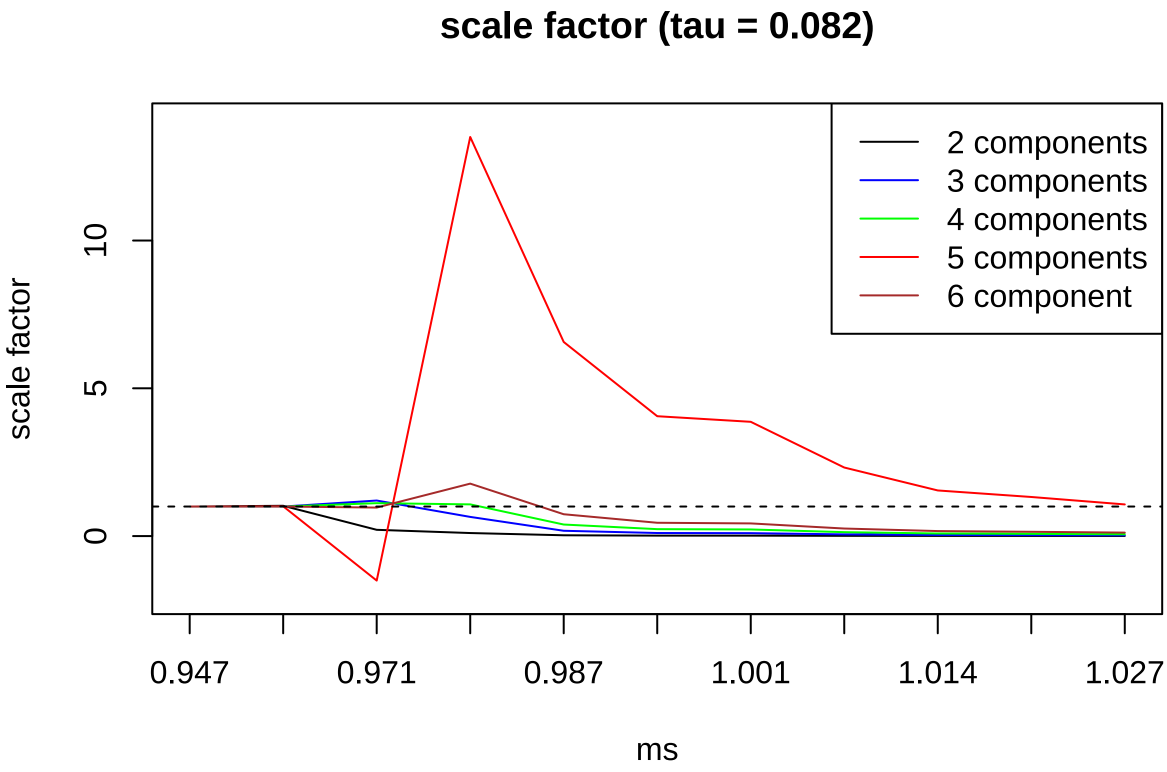

```
which.min(mse_pls2$val[2,1,])
```

```
## 2 comps
##       3
```

```
V <- t(as.matrix(train_data_2[,2:ncol(train_data_2)])) %*%
  as.matrix(train_data_2[,2:ncol(train_data_2)])
e <- eigen(V)
par(mfrow=c(2,2))
alpha <- apply(diag(train_data_2[,1]) %*%
                as.matrix(train_data_2[,2:ncol(train_data_2)]) %*% e$vectors, 2, mean) / e$values
#K: component numbers
f_PLS_2 <- c()
for(K in 2:6){
  w <- sapply(1:K, function(k) sum(alpha^2*e$values^(k+1)))
  W <- matrix(0, K, K)
  for(l in 1:K)
  {
    W[,l] <- sapply(1:K, function(k) sum(alpha^2*e$values^(k+l+1)))
  }
  beta <- solve(W, w,tol = 1e-200)
  temp <- sapply(1:130, function(j) sum(beta*(e$values[j])^(1:K)))
  f_PLS_2 <- rbind(f_PLS_2,temp)
}
```

```
ms <- c(0.947,0.960,0.971,0.979,0.987,0.995,1.001,1.007,1.014,1.021,1.027)
plot(1:11,f_PLS_2[1,1:11],'l',xlab='ms',ylab='scale factor',ylim=c(-2,14),main='scale factor (tau = 0.0
lines(f_PLS_2[2,1:11],col='blue')
lines(f_PLS_2[3,1:11],col='green')
lines(f_PLS_2[4,1:11],col='red')
```

```r
lines(f_PLS_2[5,1:11],col='brown')
abline(h=1,lty='dashed')
legend('topright',lty = c(1,1,1,1,1),legend=c("2 components","3 components","4 components","5 component
axis(1,at=1:11,label=ms)
```



PLS factors are all stationary for training + test data. First, we try the optimal component as 2.

```r
# PLS with R^2
N_comp <- 2
fit_new2 <- plsr(formula = R_sq_train~., data=data.frame(train_data_2),ncomp=N_comp, rescale = F, valid

V_pls_train2 <- as.matrix(fit_new2$scores)%*%diag(N_comp)

# Find test factors
P_pls2 <- fit_new2$projection
imp_norm <- (imp_test - t(t(rep(1,nrow(imp_test))))%*%t(apply(imp_train,2,mean)))/(t(t(rep(1,nrow(imp_t
imp_norm <- imp_norm-rep(1,dim(imp_norm)[1])%*%t(fit_new2$Xmeans)
V_pls_test2 <- as.matrix(imp_norm)%*%as.matrix(P_pls2)

# Normalize factors
V_norm_train2 <- scale(V_pls_train2,scale=TRUE,center=TRUE)
V_norm_test2 <- (V_pls_test2-t(t(rep(1,nrow(V_pls_test2))))%*%t(apply(V_pls_train2,2,mean)))/(t(t(rep(1

# Combine training and test
V_norm2 <- rbind(V_norm_train2,V_norm_test2)
```

```r
# Create lag terms for PLS with R^2
lags <- 2
N_train <- nrow(V_norm2)
```

```r
X_pls_2_all <- V_norm2[(lags+1):N_train,]
X_pls_1_all <- V_norm[(lags+1):N_train,]
for(i in 1:lags){
  if(lags==0){
    break
  }else{
    temp <- V_norm2[(lags+1-i):(N_train-i),]
    temp2 <- V_norm[(lags+1-i):(N_train-i),]
    X_pls_2_all <- cbind(X_pls_2_all,temp)
    X_pls_1_all <- cbind(X_pls_1_all,temp2)
  }
}

R_pls2 <- R_pls[(lags+1):N_train]


# GARCH
spec.eGARCH_pls2l <- ugarchspec(variance.model=list(model="eGARCH",
                    garchOrder=c(2,2),external.regressors = as.matrix(X_pls_2_all)),
                    mean.model=list(armaOrder = c(1,1),include.mean=FALSE,external.regressors =
                                   as.matrix(X_pls_2_all)), distribution.model="std")
eGARCH_pls2l <- ugarchfit(R_pls2, spec=spec.eGARCH_pls2l,out.sample=N_test)
eGARCH_pls2l
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : eGARCH(2,2)
## Mean Model   : ARFIMA(1,0,1)
## Distribution : std
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error    t value Pr(>|t|)
## ar1    -0.383336    0.041871  -9.155132 0.000000
## ma1     0.436389    0.041262  10.576104 0.000000
## mxreg1  0.313181    0.063093   4.963790 0.000001
## mxreg2 -0.052269    0.028924  -1.807143 0.070740
## mxreg3 -0.318449    0.060147  -5.294473 0.000000
## mxreg4  0.060370    0.026468   2.280842 0.022558
## mxreg5  0.023616    0.033893   0.696766 0.485949
## mxreg6  0.001816    0.035053   0.051812 0.958678
## omega  -0.037592    0.010593  -3.548754 0.000387
## alpha1 -0.332551    0.054331  -6.120811 0.000000
## alpha2  0.146070    0.053456   2.732539 0.006285
## beta1   0.999974    0.006622 151.009018 0.000000
## beta2  -0.116071    0.027393  -4.237249 0.000023
## gamma1 -0.190689    0.063380  -3.008644 0.002624
## gamma2  0.245324    0.063173   3.883343 0.000103
## vxreg1 -0.779410    0.125562  -6.207388 0.000000
```

```
## vxreg2  0.503541    0.165529   3.042015 0.002350
## vxreg3  1.645214    0.020306  81.020138 0.000000
## vxreg4 -0.840631    0.304365  -2.761920 0.005746
## vxreg5 -0.795400    0.140594  -5.657421 0.000000
## vxreg6  0.357805    0.166657   2.146953 0.031797
## shape   8.944991    1.964241   4.553917 0.000005
##
## Robust Standard Errors:
##         Estimate  Std. Error    t value Pr(>|t|)
## ar1    -0.383336    0.010291 -37.248982 0.000000
## ma1     0.436389    0.012211  35.738795 0.000000
## mxreg1  0.313181    0.023153  13.526491 0.000000
## mxreg2 -0.052269    0.016310  -3.204626 0.001352
## mxreg3 -0.318449    0.018890 -16.858450 0.000000
## mxreg4  0.060370    0.019358   3.118632 0.001817
## mxreg5  0.023616    0.009157   2.579009 0.009908
## mxreg6  0.001816    0.018417   0.098616 0.921443
## omega  -0.037592    0.009960  -3.774160 0.000161
## alpha1 -0.332551    0.062611  -5.311382 0.000000
## alpha2  0.146070    0.055714   2.621799 0.008747
## beta1   0.999974    0.002135 468.366429 0.000000
## beta2  -0.116071    0.030358  -3.823445 0.000132
## gamma1 -0.190689    0.061676  -3.091798 0.001989
## gamma2  0.245324    0.058170   4.217348 0.000025
## vxreg1 -0.779410    0.140177  -5.560179 0.000000
## vxreg2  0.503541    0.182352   2.761362 0.005756
## vxreg3  1.645214    0.022721  72.409385 0.000000
## vxreg4 -0.840631    0.333647  -2.519520 0.011751
## vxreg5 -0.795400    0.152222  -5.225267 0.000000
## vxreg6  0.357805    0.180518   1.982099 0.047468
## shape   8.944991    1.846299   4.844821 0.000001
##
## LogLikelihood : -1896.742
##
## Information Criteria
## ------------------------------------
##
## Akaike       2.5297
## Bayes        2.6069
## Shibata      2.5292
## Hannan-Quinn 2.5584
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.00286  0.9574
## Lag[2*(p+q)+(p+q)-1][5]   0.17853  1.0000
## Lag[4*(p+q)+(p+q)-1][9]   0.68975  1.0000
## d.o.f=2
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
```

```
## Lag[1]                         0.1326  0.7158
## Lag[2*(p+q)+(p+q)-1][11]    1.7260  0.9756
## Lag[4*(p+q)+(p+q)-1][19]    4.5858  0.9587
## d.o.f=4
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[5]    0.1506 0.500 2.000  0.6980
## ARCH Lag[7]    0.8372 1.473 1.746  0.8032
## ARCH Lag[9]    0.9359 2.402 1.619  0.9399
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  no.parameters>20 (not available)
## Individual Statistics:
## ar1    0.08985
## ma1    0.09231
## mxreg1 0.09935
## mxreg2 0.20560
## mxreg3 0.10343
## mxreg4 0.18210
## mxreg5 0.08310
## mxreg6 0.30533
## omega  0.09785
## alpha1 0.07982
## alpha2 0.05394
## beta1  0.03621
## beta2  0.03533
## gamma1 0.13418
## gamma2 0.10311
## vxreg1 0.07784
## vxreg2 0.15210
## vxreg3 0.08503
## vxreg4 0.13778
## vxreg5 0.08856
## vxreg6 0.13516
## shape  0.34967
##
## Asymptotic Critical Values (10% 5% 1%)
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           1.3465 0.1783
## Negative Sign Bias  0.2078 0.8354
## Positive Sign Bias  1.1244 0.2610
## Joint Effect        2.9592 0.3980
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
```

```
## 1    20     34.07     0.018009
## 2    30     46.70     0.019975
## 3    40     61.56     0.012095
## 4    50     75.52     0.008847
##
##
## Elapsed time : 1.330254
```

```
# In-sample MSE
sprintf('In-sample MSE is %g',mean((eGARCH_pls2l@fit$residuals*sd(R_train))^2))
```

```
## [1] "In-sample MSE is 0.000184657"
```

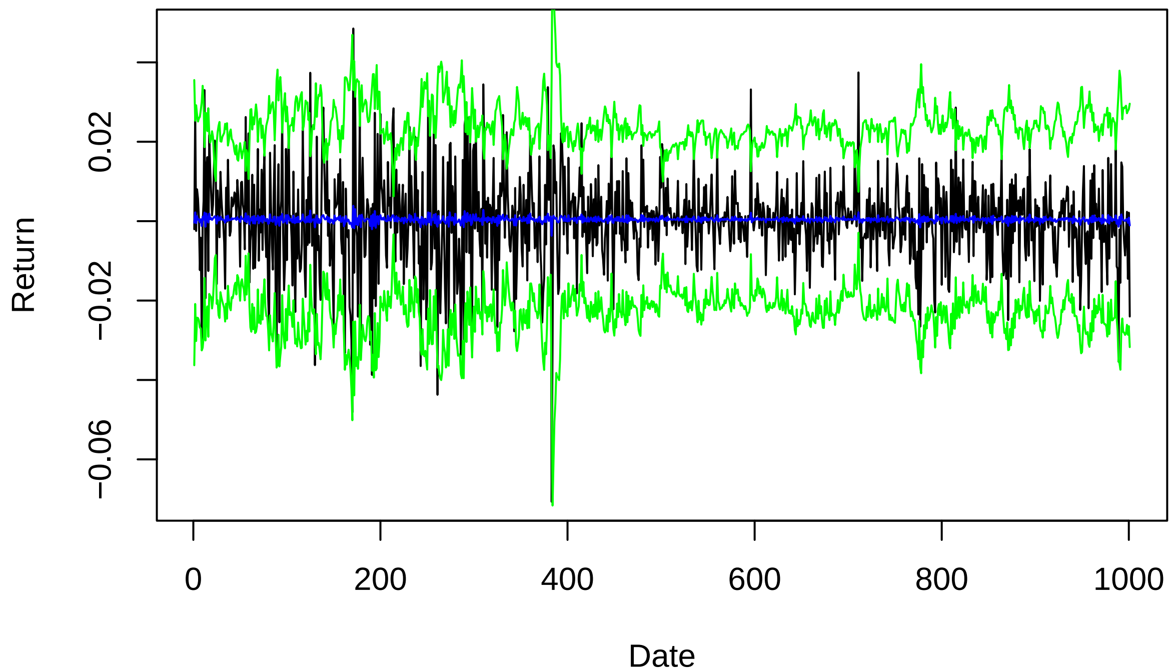
```
# Out-of-sample
forecast_eGARCH_pls2l <-ugarchforecast(eGARCH_pls2l, data = R_pls2, n.ahead = 1, n.roll = N_test,out.sa
sigma_eGARCH_pls2l<-sigma(forecast_eGARCH_pls2l)
fitted_eGARCH_pls2l<-fitted(forecast_eGARCH_pls2l)
sprintf('Out-of-sample MSE is %g',mean(((t(fitted_eGARCH_pls2l)-R_pls2[(length(R_pls2)-N_test):length(R_
```

```
## [1] "Out-of-sample MSE is 0.000122794"
```

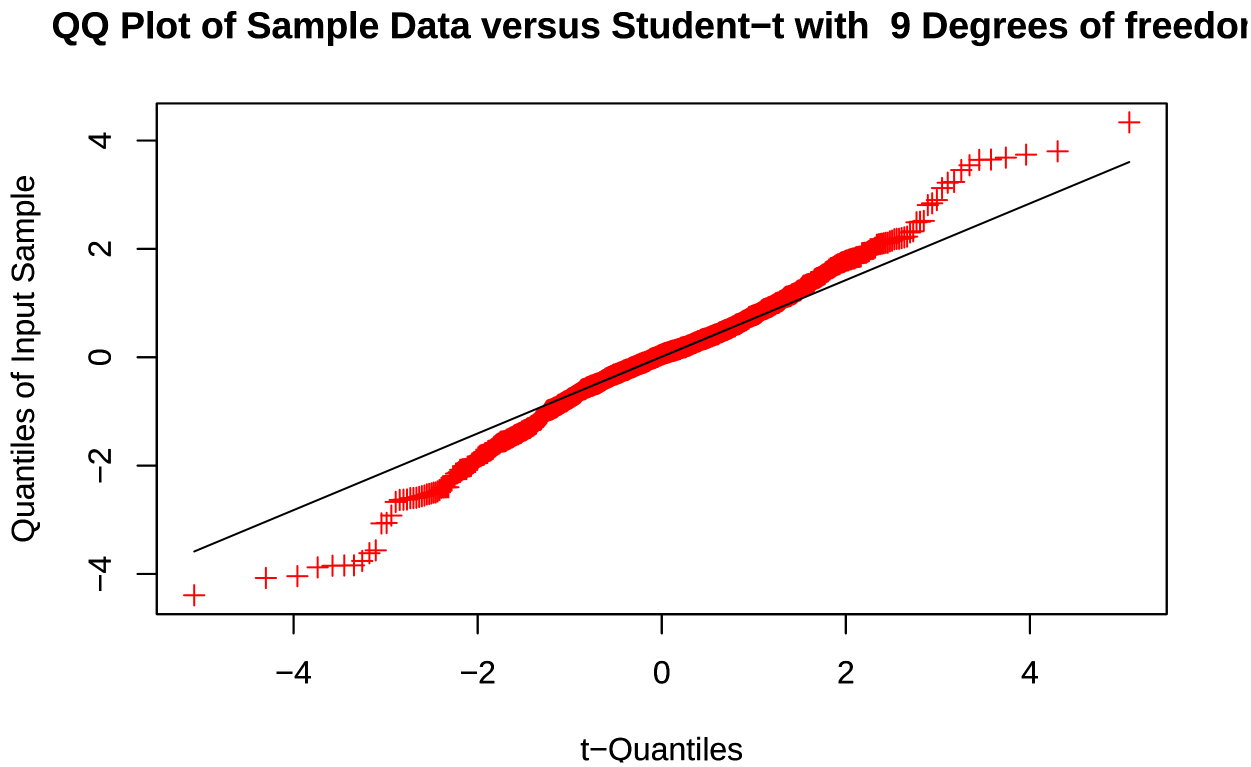
```
sprintf('Out-of-sample mean of sd is %g',mean(sigma_eGARCH_pls2l))
```

```
## [1] "Out-of-sample mean of sd is 0.890866"
```

```
plot(R_all[(length(R_all)-N_test):length(R_all)],type='l',xlab='Date',ylab='Return')
lines(t(fitted_eGARCH_pls2l)*sd(R_train)+mean(R_train),col='blue')
lines(t(fitted_eGARCH_pls2l)*sd(R_train)+mean(R_train)+1.96*t(sigma_eGARCH_pls2l)*sd(R_train),col='gree
lines(t(fitted_eGARCH_pls2l)*sd(R_train)+mean(R_train)-1.96*t(sigma_eGARCH_pls2l)*sd(R_train),col='gree
```




```
# Autocorrelaiton and heteroscedasticity for residuals
TQQPlot(eGARCH_pls2l@fit$residuals, 9)
```




```
Box.test(eGARCH_pls21@fit$residuals, lag = 10, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  eGARCH_pls21@fit$residuals
## X-squared = 6.5881, df = 10, p-value = 0.7637
```

```
Box.test(eGARCH_pls21@fit$residuals^2, lag = 10, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  eGARCH_pls21@fit$residuals^2
## X-squared = 497.48, df = 10, p-value < 2.2e-16
```

```
# Coverage test
roll_GARCH_pls21 <-ugarchroll(spec=spec.eGARCH_pls21, data=R_pls2, n.ahead=1, forecast.length=N_test, r
report(roll_GARCH_pls21, type="VaR", VaR.alpha = 0.05, conf.level = 0.95)
```

```
## VaR Backtest Report
## ===========================================
## Model:                  eGARCH-std
## Backtest Length: 1000
## Data:
##
## ==========================================
## alpha:                  5%
## Expected Exceed: 50
## Actual VaR Exceed:      57
## Actual %:               5.7%
```

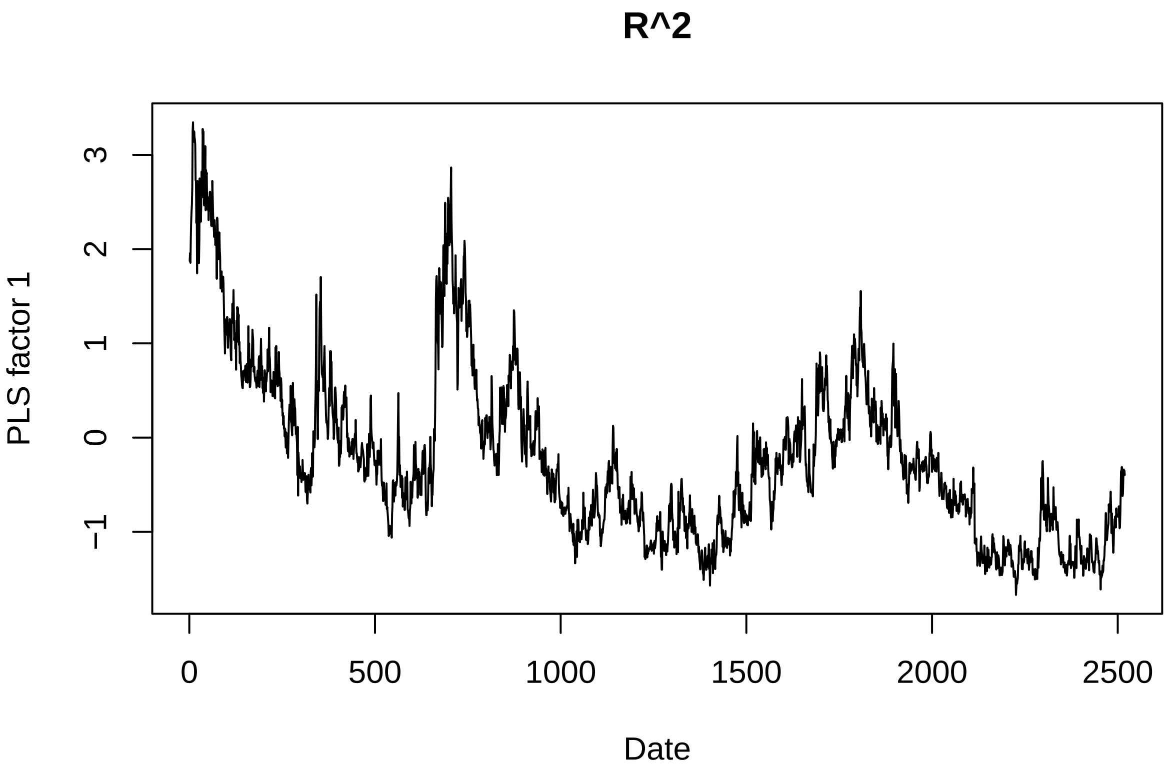
```
##
## Unconditional Coverage (Kupiec)
## Null-Hypothesis: Correct Exceedances
## LR.uc Statistic: 0.989
## LR.uc Critical:      3.841
## LR.uc p-value:       0.32
## Reject Null:     NO
##
## Conditional Coverage (Christoffersen)
## Null-Hypothesis: Correct Exceedances and
##                  Independence of Failures
## LR.cc Statistic: 1.985
## LR.cc Critical:      5.991
## LR.cc p-value:       0.371
## Reject Null:     NO
```

Next, we try the optimal number of components as 5.

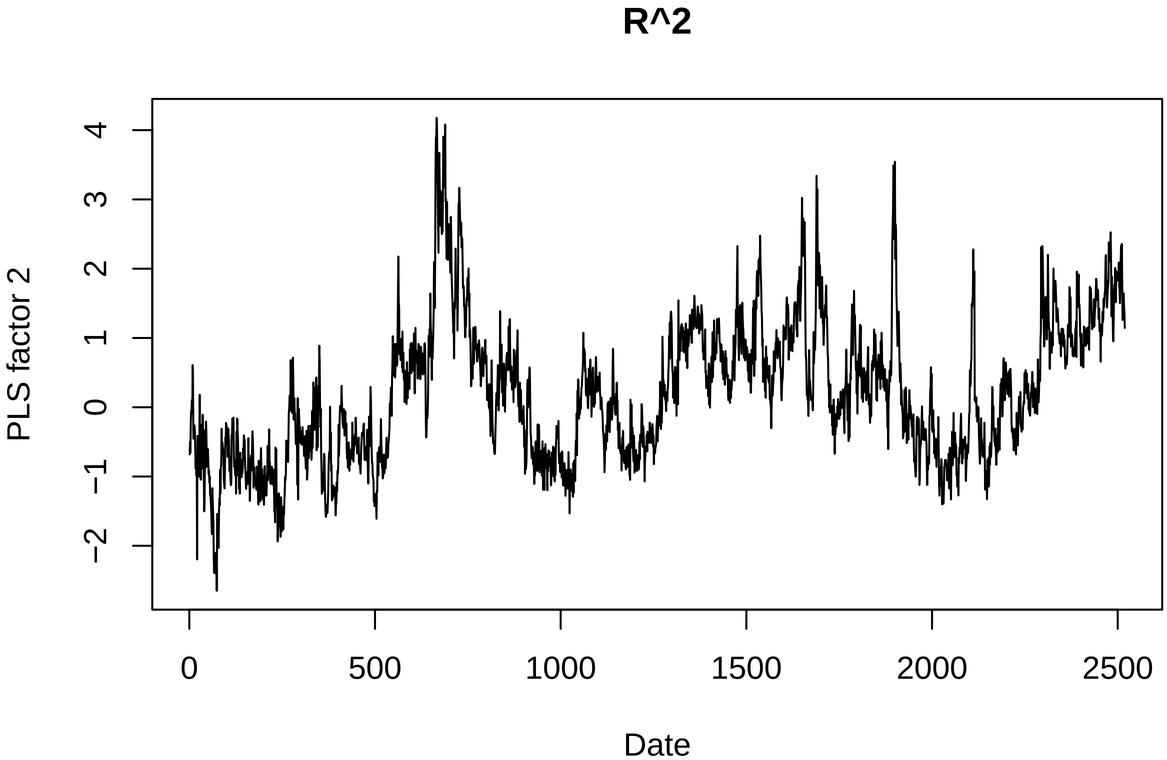
```
# PLS with R^2
N_comp <- 5
fit_new2 <- plsr(formula = R_sq_train~., data=data.frame(train_data_2),ncomp=N_comp, rescale = F, valid

V_pls_train2 <- as.matrix(fit_new2$scores)%*%diag(N_comp)

# Find test factors
P_pls2 <- fit_new2$projection
imp_norm <- (imp_test - t(t(rep(1,nrow(imp_test))))%*%t(apply(imp_train,2,mean)))/(t(t(rep(1,nrow(imp_te
imp_norm <- imp_norm-rep(1,dim(imp_norm)[1])%*%t(fit_new2$Xmeans)
V_pls_test2 <- as.matrix(imp_norm)%*%as.matrix(P_pls2)

# Normalize factors
V_norm_train2 <- scale(V_pls_train2,scale=TRUE,center=TRUE)
V_norm_test2 <- (V_pls_test2-t(t(rep(1,nrow(V_pls_test2))))%*%t(apply(V_pls_train2,2,mean)))/(t(t(rep(1

# Combine training and test
V_norm2 <- rbind(V_norm_train2,V_norm_test2)
```

```
# ADF test for training data
for(i in 1:N_comp){
  plot(V_norm2[,i],type='l',xlab='Date',ylab=paste0("PLS factor ",i),main='R^2')
  print(adf.test(V_norm2[,i],k=10))
}
```

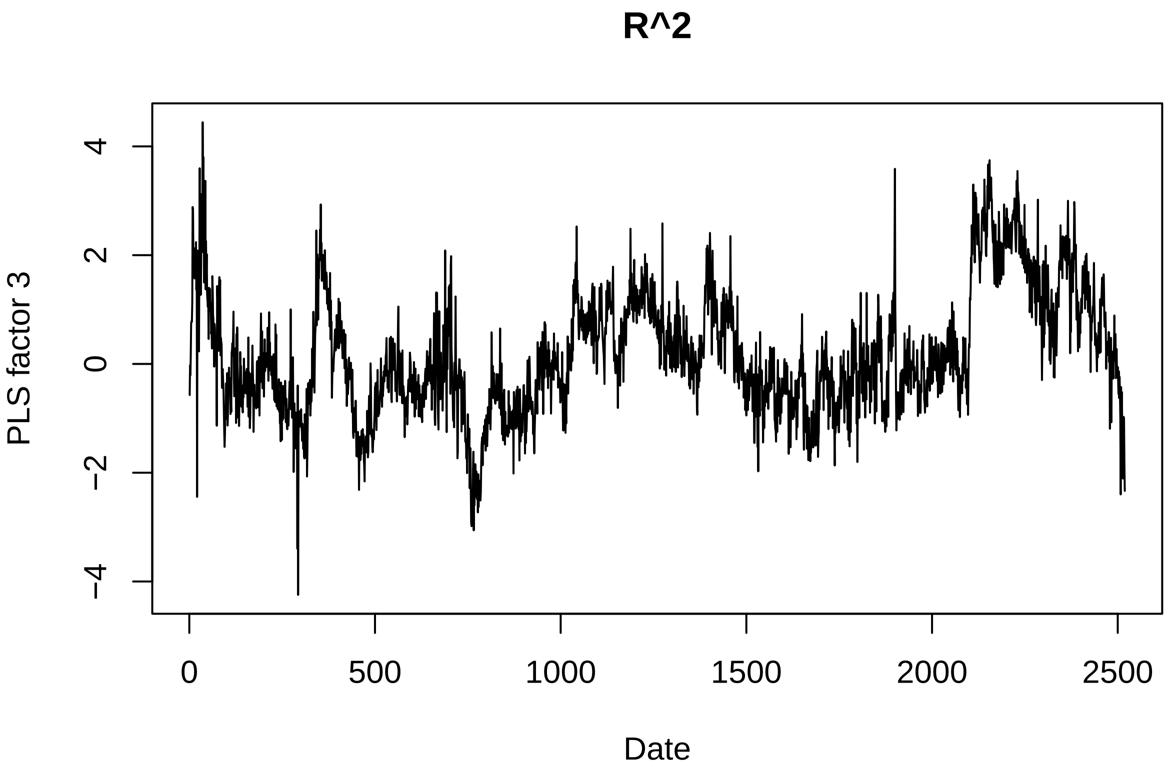
```
## Warning in adf.test(V_norm2[, i], k = 10): p-value smaller than printed p-value
```



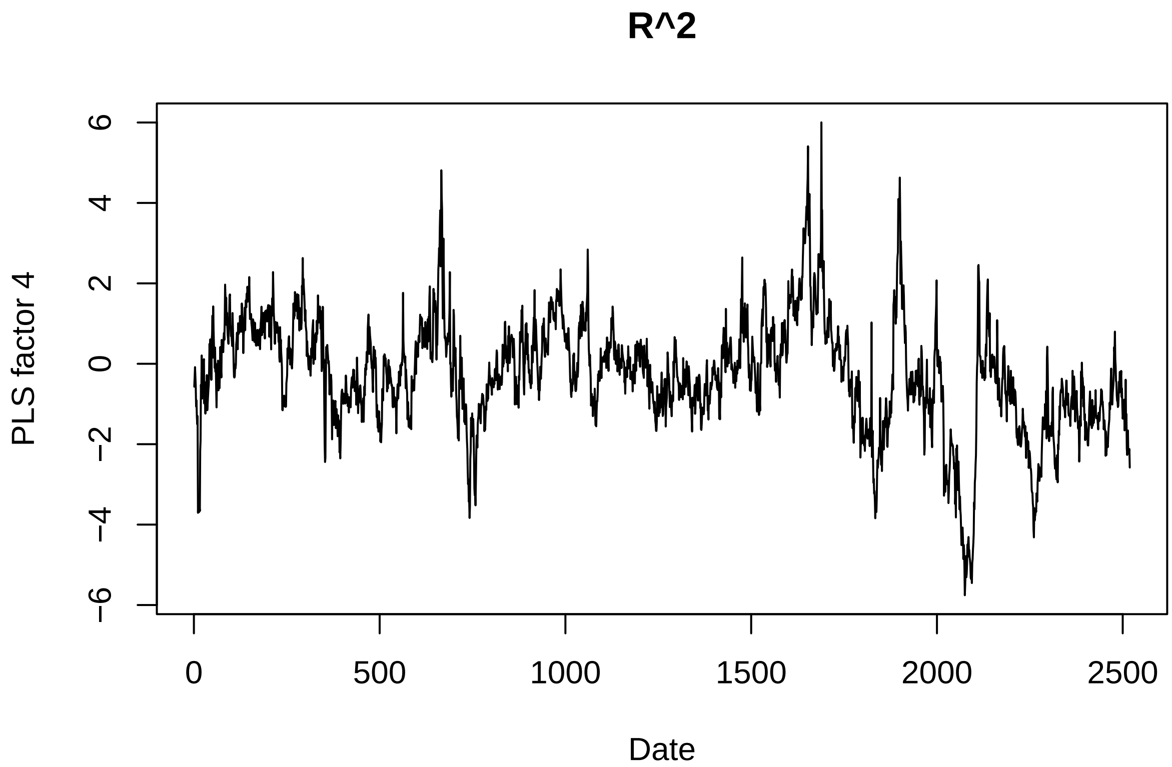

```
##
##  Augmented Dickey-Fuller Test
##
## data:  V_norm2[, i]
## Dickey-Fuller = -4.1464, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary

## Warning in adf.test(V_norm2[, i], k = 10): p-value smaller than printed p-value
```



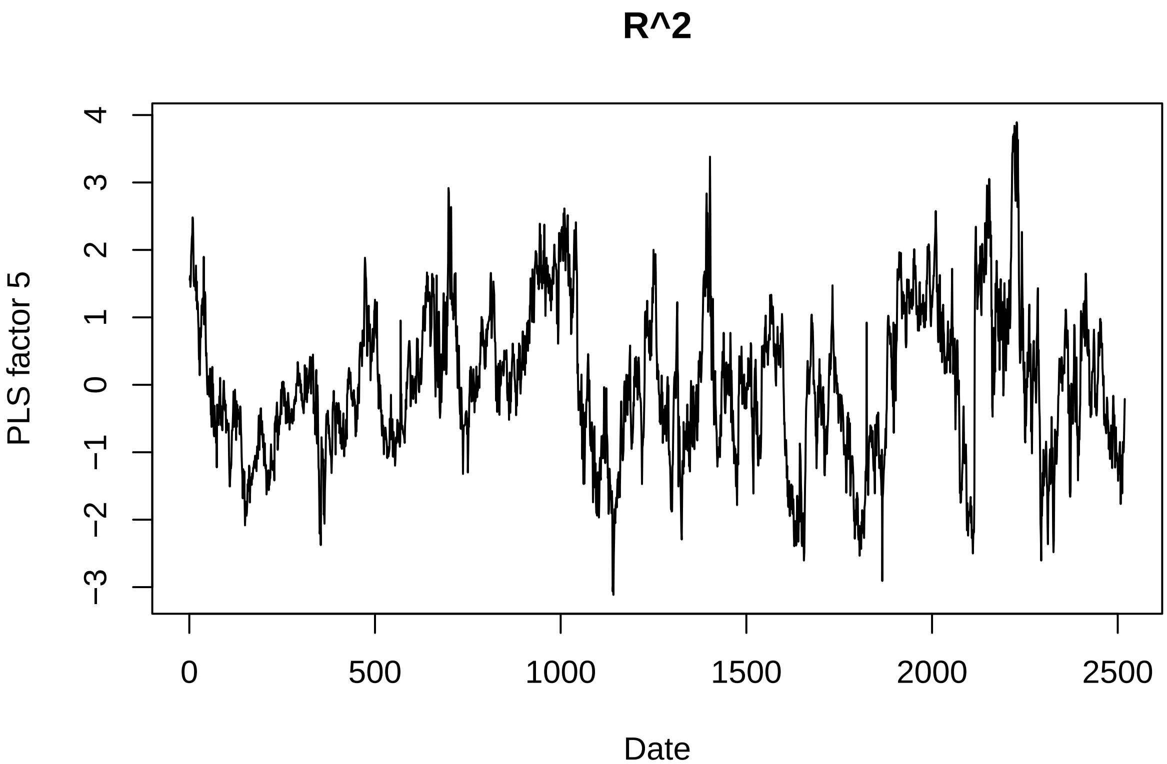

```
##
##  Augmented Dickey-Fuller Test
##
## data:  V_norm2[, i]
## Dickey-Fuller = -5.2968, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
## Warning in adf.test(V_norm2[, i], k = 10): p-value smaller than printed p-value
```




```
##
##  Augmented Dickey-Fuller Test
##
## data:  V_norm2[, i]
## Dickey-Fuller = -4.5606, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary

## Warning in adf.test(V_norm2[, i], k = 10): p-value smaller than printed p-value
```




```
##
##  Augmented Dickey-Fuller Test
##
## data:  V_norm2[, i]
## Dickey-Fuller = -6.1945, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary

## Warning in adf.test(V_norm2[, i], k = 10): p-value smaller than printed p-value
```




```
##
##  Augmented Dickey-Fuller Test
##
## data:  V_norm2[, i]
## Dickey-Fuller = -6.0346, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

The correlation between first and second component of R and R^2 are highly correlated. Thus, we use PLS factors from R^2 alone in the following model.

```
# Correlation between latent factors from R and R^2
for(i in 1:3){
  print(cor(cbind(V_norm[,i],V_norm2[,i])))
}
```

```
##           [,1]      [,2]
## [1,] 1.0000000 0.9997255
## [2,] 0.9997255 1.0000000
##            [,1]       [,2]
## [1,]  1.0000000 -0.9791016
## [2,] -0.9791016  1.0000000
##           [,1]      [,2]
## [1,] 1.0000000 0.1539187
## [2,] 0.1539187 1.0000000
```

Next, we try the eGARCH model with ged distribution. The in-sample MSE is 0.0001837, while the out-of-sample MSE is 0.0001286. The residuals are independent, and the model passes the coverage test.

```
# Create lag terms for PLS with R^2
lags <- 2
N_train <- nrow(V_norm2)
```

```r
X_pls_2_all <- V_norm2[(lags+1):N_train,]
X_pls_1_all <- V_norm[(lags+1):N_train,]
for(i in 1:lags){
  if(lags==0){
    break
  }else{
    temp <- V_norm2[(lags+1-i):(N_train-i),]
    temp2 <- V_norm[(lags+1-i):(N_train-i),]
    X_pls_2_all <- cbind(X_pls_2_all,temp)
    X_pls_1_all <- cbind(X_pls_1_all,temp2)
  }
}

R_pls2 <- R_pls[(lags+1):N_train]


# GARCH
spec.eGARCH_pls2 <- ugarchspec(variance.model=list(model="eGARCH",
                    garchOrder=c(2,2),external.regressors = as.matrix(X_pls_2_all[,1:5])),
                    mean.model=list(armaOrder = c(1,1),include.mean=FALSE,external.regressors =
                                  as.matrix(X_pls_1_all)), distribution.model="ged")
eGARCH_pls2 <- ugarchfit(R_pls2, spec=spec.eGARCH_pls2,out.sample=N_test)
eGARCH_pls2
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : eGARCH(2,2)
## Mean Model   : ARFIMA(1,0,1)
## Distribution : ged
##
## Optimal Parameters
## ------------------------------------
##          Estimate  Std. Error   t value Pr(>|t|)
## ar1     -0.475654    0.038562 -12.33465 0.000000
## ma1      0.521073    0.036594  14.23946 0.000000
## mxreg1   0.273050    0.097086   2.81246 0.004916
## mxreg2   0.034385    0.051770   0.66420 0.506563
## mxreg3   0.002583    0.008150   0.31699 0.751252
## mxreg4  -0.568752    0.259723  -2.18984 0.028536
## mxreg5  -0.209454    0.087887  -2.38322 0.017162
## mxreg6   0.036039    0.017448   2.06547 0.038878
## mxreg7   0.312639    0.180623   1.73089 0.083471
## mxreg8   0.177465    0.067690   2.62172 0.008749
## mxreg9   0.011431    0.011052   1.03427 0.301009
## omega   -0.061576    0.024072  -2.55800 0.010528
## alpha1  -0.257569    0.045913  -5.60997 0.000000
## alpha2   0.034776    0.081202   0.42826 0.668463
## beta1    0.839164    0.282329   2.97229 0.002956
## beta2   -0.026417    0.226188  -0.11679 0.907026
```

```
## gamma1 -0.144816    0.064219  -2.25504 0.024131
## gamma2  0.177284    0.066197   2.67811 0.007404
## vxreg1  0.116666    0.044882   2.59941 0.009338
## vxreg2  0.036708    0.019162   1.91566 0.055408
## vxreg3  0.004121    0.010371   0.39738 0.691085
## vxreg4  0.016772    0.012368   1.35611 0.175063
## vxreg5  0.002907    0.010383   0.27998 0.779490
## shape   1.509560    0.080879  18.66431 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error    t value Pr(>|t|)
## ar1    -0.475654    0.024686 -19.268049 0.000000
## ma1     0.521073    0.019911  26.170332 0.000000
## mxreg1  0.273050    0.117924   2.315480 0.020587
## mxreg2  0.034385    0.042008   0.818541 0.413048
## mxreg3  0.002583    0.029109   0.088752 0.929279
## mxreg4 -0.568752    0.347328  -1.637505 0.101525
## mxreg5 -0.209454    0.059034  -3.548001 0.000388
## mxreg6  0.036039    0.018511   1.946891 0.051548
## mxreg7  0.312639    0.234125   1.335348 0.181763
## mxreg8  0.177465    0.050716   3.499210 0.000467
## mxreg9  0.011431    0.004581   2.495431 0.012580
## omega  -0.061576    0.020858  -2.952245 0.003155
## alpha1 -0.257569    0.046334  -5.559012 0.000000
## alpha2  0.034776    0.072515   0.479565 0.631537
## beta1   0.839164    0.192966   4.348761 0.000014
## beta2  -0.026417    0.149620  -0.176559 0.859855
## gamma1 -0.144816    0.065223  -2.220333 0.026396
## gamma2  0.177284    0.065445   2.708887 0.006751
## vxreg1  0.116666    0.036094   3.232282 0.001228
## vxreg2  0.036708    0.018951   1.937004 0.052745
## vxreg3  0.004121    0.011710   0.351945 0.724880
## vxreg4  0.016772    0.011087   1.512723 0.130350
## vxreg5  0.002907    0.010679   0.272235 0.785442
## shape   1.509560    0.076274  19.791153 0.000000
##
## LogLikelihood : -1891.562
##
## Information Criteria
## ------------------------------------
##
## Akaike       2.5255
## Bayes        2.6097
## Shibata      2.5250
## Hannan-Quinn 2.5568
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.05877  0.8085
## Lag[2*(p+q)+(p+q)-1][5]   0.67104  1.0000
## Lag[4*(p+q)+(p+q)-1][9]   1.34303  0.9985
## d.o.f=2
## H0 : No serial correlation
```

```
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                    0.004342  0.9475
## Lag[2*(p+q)+(p+q)-1][11]  1.166193  0.9938
## Lag[4*(p+q)+(p+q)-1][19]  3.501836  0.9888
## d.o.f=4
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[5]    0.0103 0.500 2.000  0.9192
## ARCH Lag[7]    0.3080 1.473 1.746  0.9463
## ARCH Lag[9]    0.3525 2.402 1.619  0.9926
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  no.parameters>20 (not available)
## Individual Statistics:
## ar1    0.10629
## ma1    0.10593
## mxreg1 0.02475
## mxreg2 0.10065
## mxreg3 0.18066
## mxreg4 0.02496
## mxreg5 0.08706
## mxreg6 0.13147
## mxreg7 0.02391
## mxreg8 0.12325
## mxreg9 0.16477
## omega  0.05785
## alpha1 0.12379
## alpha2 0.05463
## beta1  0.02644
## beta2  0.02537
## gamma1 0.15482
## gamma2 0.06171
## vxreg1 0.02239
## vxreg2 0.08902
## vxreg3 0.05386
## vxreg4 0.07764
## vxreg5 0.07187
## shape  0.29441
##
## Asymptotic Critical Values (10% 5% 1%)
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.7570 0.4492
## Negative Sign Bias  0.6942 0.4877
## Positive Sign Bias  1.0758 0.2822
```

```
## Joint Effect        2.3172 0.5092
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     26.82      0.10884
## 2    30     35.03      0.20355
## 3    40     46.74      0.18429
## 4    50     68.20      0.03614
##
##
## Elapsed time : 1.767093
```

```r
# In-sample MSE
sprintf('In-sample MSE is %g',mean((eGARCH_pls2@fit$residuals*sd(R_train))^2))
```

```
## [1] "In-sample MSE is 0.000183728"
```

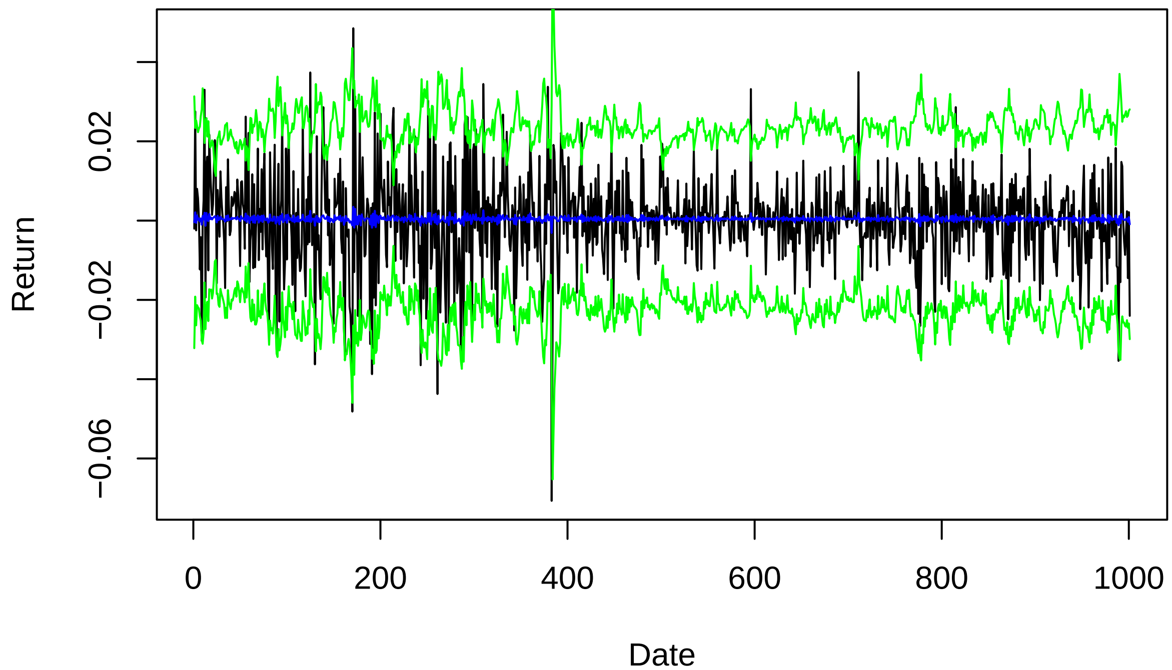
```r
# Out-of-sample
forecast_eGARCH_pls2<-ugarchforecast(eGARCH_pls2, data = R_pls2, n.ahead = 1, n.roll = N_test,out.sampl
sigma_eGARCH_pls2<-sigma(forecast_eGARCH_pls2)
fitted_eGARCH_pls2<-fitted(forecast_eGARCH_pls2)
sprintf('Out-of-sample MSE is %g',mean(((t(fitted_eGARCH_pls2)-R_pls2[(length(R_pls2)-N_test):length(R_
```

```
## [1] "Out-of-sample MSE is 0.000124819"
```

```r
sprintf('Out-of-sample mean of sd is %g',mean(sigma_eGARCH_pls2))
```

```
## [1] "Out-of-sample mean of sd is 0.880146"
```

```r
plot(R_all[(length(R_all)-N_test):length(R_all)],type='l',xlab='Date',ylab='Return')
lines(t(fitted_eGARCH_pls2)*sd(R_train)+mean(R_train),col='blue')
lines(t(fitted_eGARCH_pls2)*sd(R_train)+mean(R_train)+1.96*t(sigma_eGARCH_pls2)*sd(R_train),col='green')
lines(t(fitted_eGARCH_pls2)*sd(R_train)+mean(R_train)-1.96*t(sigma_eGARCH_pls2)*sd(R_train),col='green')
```



```
# Autocorrelaiton and heteroscedasticity for residuals
Box.test(eGARCH_pls2@fit$residuals, lag = 10, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  eGARCH_pls2@fit$residuals
## X-squared = 7.9792, df = 10, p-value = 0.6309
```

```
Box.test(eGARCH_pls2@fit$residuals^2, lag = 10, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  eGARCH_pls2@fit$residuals^2
## X-squared = 505.02, df = 10, p-value < 2.2e-16
```

```
# Coverage test
roll_GARCH_pls2<-ugarchroll(spec=spec.eGARCH_pls2, data=R_pls2, n.ahead=1, forecast.length=N_test, refi
report(roll_GARCH_pls2, type="VaR", VaR.alpha = 0.05, conf.level = 0.95)
```

```
## VaR Backtest Report
## ===========================================
## Model:               eGARCH-ged
## Backtest Length: 1000
## Data:
##
## ==========================================
## alpha:               5%
## Expected Exceed: 50
## Actual VaR Exceed:   61
## Actual %:            6.1%
##
## Unconditional Coverage (Kupiec)
## Null-Hypothesis: Correct Exceedances
## LR.uc Statistic: 2.388
## LR.uc Critical:      3.841
## LR.uc p-value:       0.122
## Reject Null:     NO
##
## Conditional Coverage (Christoffersen)
## Null-Hypothesis: Correct Exceedances and
##                  Independence of Failures
## LR.cc Statistic: 2.89
## LR.cc Critical:      5.991
## LR.cc p-value:       0.236
## Reject Null:     NO
```

Next, we use gjrGARCH with std distribution. The in-sample MSE is 0.0001839, while the out-of-sample MSE is 0.0001189. The residuals are independent, and the model passes the coverage test. The main problem of this model is that the coefficients of latent factors in variance model is insignificant.

```
lags <- 1
N_train <- nrow(V_norm2)

X_pls_2_all <- V_norm2[(lags+1):N_train,]
```

```r
X_pls_1_all <- V_norm[(lags+1):N_train,]
for(i in 1:lags){
  if(lags==0){
    break
  }else{
    temp <- V_norm2[(lags+1-i):(N_train-i),]
    temp2 <- V_norm[(lags+1-i):(N_train-i),]
    X_pls_2_all <- cbind(X_pls_2_all,temp)
    X_pls_1_all <- cbind(X_pls_1_all,temp2)
  }
}


R_pls2 <- R_pls[(lags+1):N_train]


# GARCH
spec.gjrGARCH_pls2 <- ugarchspec(variance.model=list(model="gjrGARCH",
                            garchOrder=c(0,1),external.regressors = as.matrix(X_pls_2_all)),
                            mean.model=list(armaOrder = c(1,1),include.mean=FALSE,
                                            external.regressors =
                                              as.matrix(X_pls_1_all)),
                            distribution.model="std")
gjrGARCH_pls2 <- ugarchfit(R_pls2, spec=spec.gjrGARCH_pls2,out.sample=N_test)
gjrGARCH_pls2
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : gjrGARCH(0,1)
## Mean Model   : ARFIMA(1,0,1)
## Distribution : std
##
## Optimal Parameters
## ------------------------------------
##          Estimate  Std. Error    t value Pr(>|t|)
## ar1     -0.198345    0.002694  -73.61272 0.000000
## ma1      0.272665    0.000488  559.27552 0.000000
## mxreg1   0.604155    0.002813  214.79134 0.000000
## mxreg2   0.018795    0.048721    0.38576 0.699672
## mxreg3  -0.065024    0.001191  -54.59152 0.000000
## mxreg4  -0.589399    0.005287 -111.47146 0.000000
## mxreg5  -0.009230    0.052444   -0.17600 0.860290
## mxreg6   0.127549    0.000800  159.41476 0.000000
## omega    0.657294    0.001852  354.87181 0.000000
## beta1    0.271716    0.000862  315.25119 0.000000
## vxreg1   0.373401    0.001120  333.43740 0.000000
## vxreg2   0.215562    0.000632  340.92087 0.000000
## vxreg3   0.000259    0.000002  137.83181 0.000000
## vxreg4   0.050376    0.000155  324.10107 0.000000
```

```
## vxreg5   0.000064    0.000000  221.06821 0.000000
## vxreg6   0.000000    0.000007    0.00650 0.994814
## vxreg7   0.017785    0.000060  295.28612 0.000000
## vxreg8   0.087172    0.000275  317.28401 0.000000
## vxreg9   0.000009    0.000005    1.86570 0.062083
## vxreg10  0.030941    0.000104  296.91118 0.000000
## shape   99.985706   53.744750    1.86038 0.062832
##
## Robust Standard Errors:
##          Estimate  Std. Error   t value Pr(>|t|)
## ar1     -0.198345    0.828805 -0.239314 0.810862
## ma1      0.272665    0.609637  0.447257 0.654689
## mxreg1   0.604155    0.109511  5.516828 0.000000
## mxreg2   0.018795    4.185792  0.004490 0.996417
## mxreg3  -0.065024    0.137940 -0.471395 0.637359
## mxreg4  -0.589399    0.154662 -3.810886 0.000138
## mxreg5  -0.009230    3.688271 -0.002503 0.998003
## mxreg6   0.127549    0.001455 87.661937 0.000000
## omega    0.657294    0.073302  8.966888 0.000000
## beta1    0.271716    0.195333  1.391039 0.164214
## vxreg1   0.373401    0.086424  4.320545 0.000016
## vxreg2   0.215562    0.013401 16.085351 0.000000
## vxreg3   0.000259    0.000168  1.538335 0.123967
## vxreg4   0.050376    0.033088  1.522501 0.127884
## vxreg5   0.000064    0.000054  1.178734 0.238504
## vxreg6   0.000000    0.000608  0.000073 0.999942
## vxreg7   0.017785    0.000445 39.946605 0.000000
## vxreg8   0.087172    0.073502  1.185991 0.235626
## vxreg9   0.000009    0.000337  0.025807 0.979412
## vxreg10  0.030941    0.014905  2.075933 0.037900
## shape   99.985706   42.666443  2.343427 0.019107
##
## LogLikelihood : -1926.725
##
## Information Criteria
## ------------------------------------
##
## Akaike       2.5662
## Bayes        2.6398
## Shibata      2.5658
## Hannan-Quinn 2.5936
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.3077  0.5791
## Lag[2*(p+q)+(p+q)-1][5]    0.6053  1.0000
## Lag[4*(p+q)+(p+q)-1][9]    1.5048  0.9970
## d.o.f=2
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
```

```
## Lag[1]                          4.080 0.04340
## Lag[2*(p+q)+(p+q)-1][2]     4.084 0.07157
## Lag[4*(p+q)+(p+q)-1][5]     7.235 0.04546
## d.o.f=1
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[2]  0.009447 0.500 2.000  0.9226
## ARCH Lag[4]  3.811707 1.397 1.611  0.1697
## ARCH Lag[6]  4.541470 2.222 1.500  0.2402
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  no.parameters>20 (not available)
## Individual Statistics:
## ar1     0.31402
## ma1     0.31248
## mxreg1  0.03776
## mxreg2  0.13503
## mxreg3  0.26325
## mxreg4  0.04011
## mxreg5  0.09780
## mxreg6  0.26836
## omega   0.65530
## beta1   0.66898
## vxreg1  0.67206
## vxreg2  0.66550
## vxreg3  0.66927
## vxreg4  0.66921
## vxreg5  0.66938
## vxreg6  0.68338
## vxreg7  0.66926
## vxreg8  0.66928
## vxreg9  0.63846
## vxreg10 0.66934
## shape   2.83218
##
## Asymptotic Critical Values (10% 5% 1%)
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value    prob sig
## Sign Bias           0.7676 0.44282
## Negative Sign Bias  0.4590 0.64627
## Positive Sign Bias  2.2334 0.02567  **
## Joint Effect        5.8942 0.11687
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     49.77    1.415e-04
```

```
## 2    30      69.23    3.858e-05
## 3    40      74.62    5.138e-04
## 4    50      79.96    3.433e-03
##
##
## Elapsed time : 2.335755
```

```
# In-sample MSE
sprintf('In-sample MSE is %g',mean((gjrGARCH_pls2@fit$residuals*sd(R_train))^2))
```

```
## [1] "In-sample MSE is 0.000183477"
```

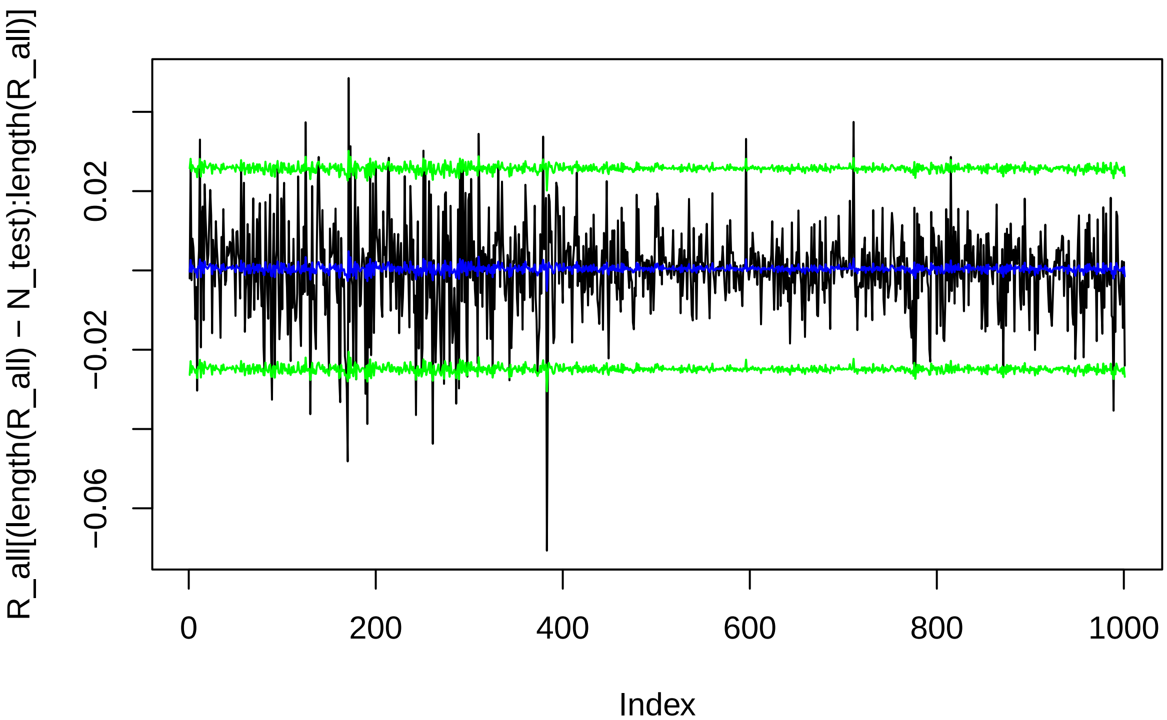
```
# Out-of-sample
forecast_gjrGARCH_pls2<-ugarchforecast(gjrGARCH_pls2, data = R_pls2, n.ahead = 1, n.roll = N_test,out.sa
sigma_gjrGARCH_pls2<-sigma(forecast_gjrGARCH_pls2)
fitted_gjrGARCH_pls2<-fitted(forecast_gjrGARCH_pls2)
sprintf('Out-of-sample MSE is %g',mean(((t(fitted_gjrGARCH_pls2)-R_pls2[(length(R_pls2)-N_test):length(R
```

```
## [1] "Out-of-sample MSE is 0.000117272"
```

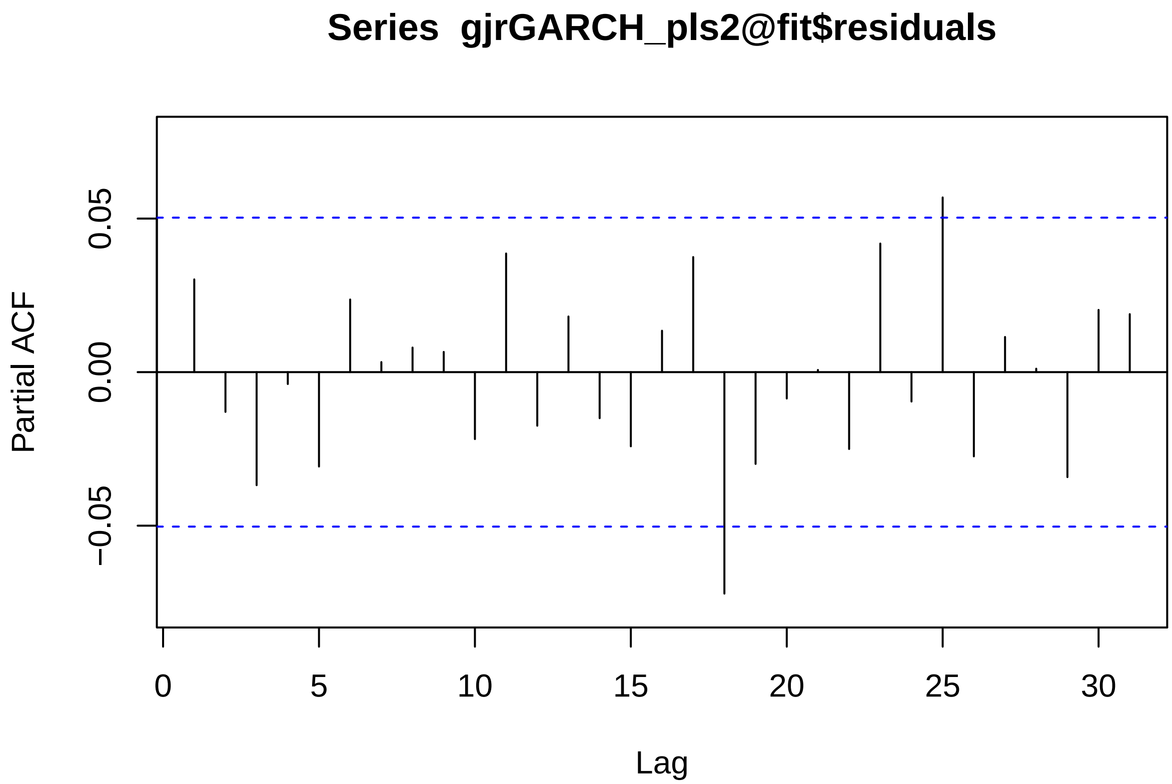
```
sprintf('Out-of-sample mean of sd is %g',mean(sigma_gjrGARCH_pls2))
```

```
## [1] "Out-of-sample mean of sd is 0.95005"
```

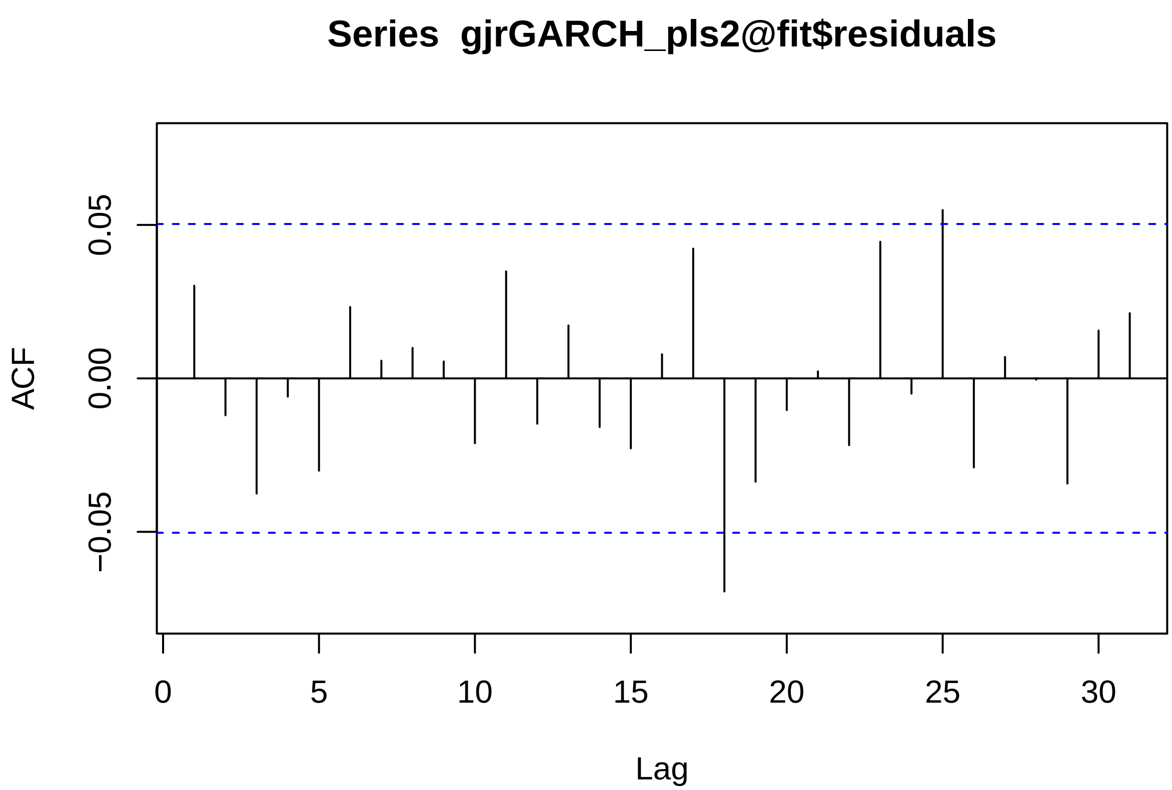
```
plot(R_all[(length(R_all)-N_test):length(R_all)],type='l')
lines(t(fitted_gjrGARCH_pls2)*sd(R_train)+mean(R_train),col='blue')
lines(t(fitted_gjrGARCH_pls2)*sd(R_train)+mean(R_train)+1.96*t(sigma_gjrGARCH_pls2)*sd(R_train),col='gre
lines(t(fitted_gjrGARCH_pls2)*sd(R_train)+mean(R_train)-1.96*t(sigma_gjrGARCH_pls2)*sd(R_train),col='gre
```




```
# QQplot, autocorrelation, and heteroscedasticity for residuals
Pacf(gjrGARCH_pls2@fit$residuals)
```




```
Acf(gjrGARCH_pls2@fit$residuals)
```




```
Box.test(gjrGARCH_pls2@fit$residuals, lag = 10, type = "Ljung")
```

```
##
```

```
##  Box-Ljung test
##
## data:  gjrGARCH_pls2@fit$residuals
## X-squared = 6.946, df = 10, p-value = 0.7305
```

```
Box.test(gjrGARCH_pls2@fit$residuals^2, lag = 10, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  gjrGARCH_pls2@fit$residuals^2
## X-squared = 490.96, df = 10, p-value < 2.2e-16
```

```
# Coverage test
roll_GARCH_pls2<-ugarchroll(spec=spec.gjrGARCH_pls2, data=R_pls2, n.ahead=1, forecast.length=N_test, re
report(roll_GARCH_pls2, type="VaR", VaR.alpha = 0.05, conf.level = 0.95)
```

```
## VaR Backtest Report
## ===========================================
## Model:               gjrGARCH-std
## Backtest Length: 1000
## Data:
##
## ==========================================
## alpha:               5%
## Expected Exceed: 50
## Actual VaR Exceed:   44
## Actual %:            4.4%
##
## Unconditional Coverage (Kupiec)
## Null-Hypothesis: Correct Exceedances
## LR.uc Statistic: 0.788
## LR.uc Critical:      3.841
## LR.uc p-value:       0.375
## Reject Null:     NO
##
## Conditional Coverage (Christoffersen)
## Null-Hypothesis: Correct Exceedances and
##                  Independence of Failures
## LR.cc Statistic: 4.771
## LR.cc Critical:      5.991
## LR.cc p-value:       0.092
## Reject Null:     NO
```

DCC VAR for PLS factors

From ACF and PACF, for PLS factors from R, each series follows AR(2). For factors from R^2, except factor 3, they follow AR(2) as well.

```
library(vars)
```

```
## Loading required package: strucchange
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

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

```
## Loading required package: sandwich
```

```
## Loading required package: urca
```

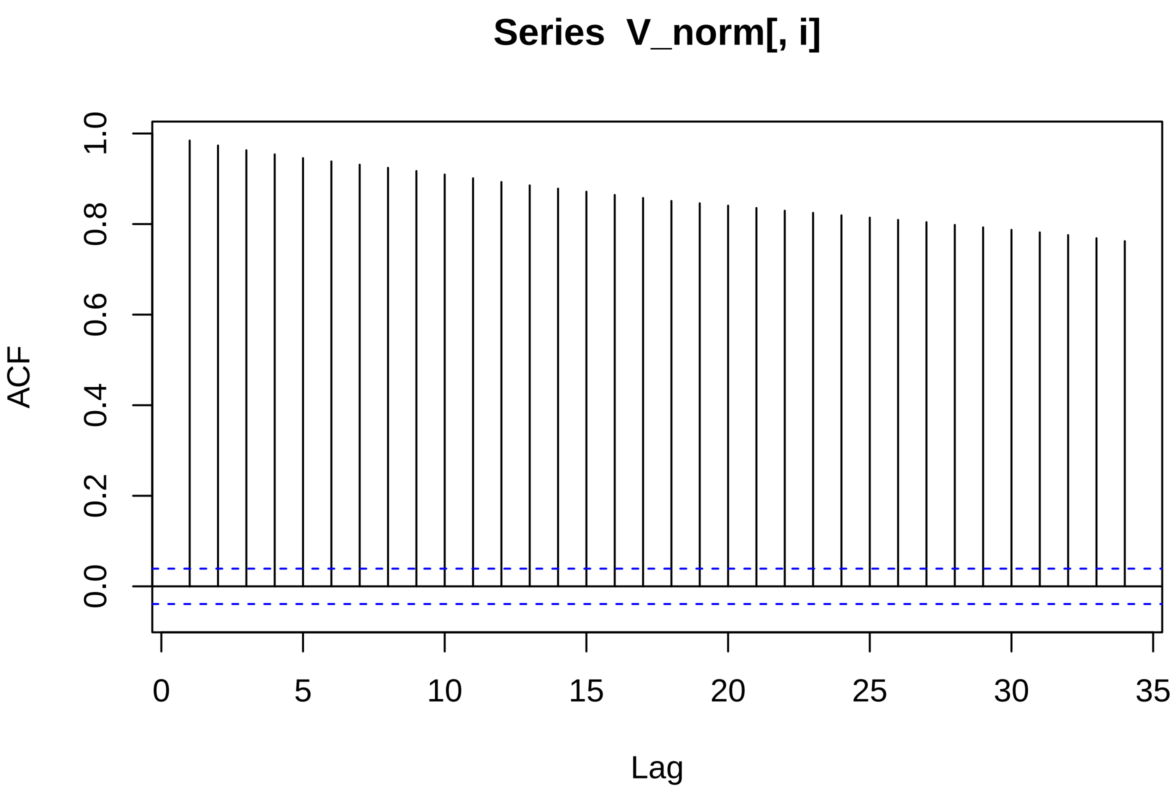
```
## Loading required package: lmtest
```

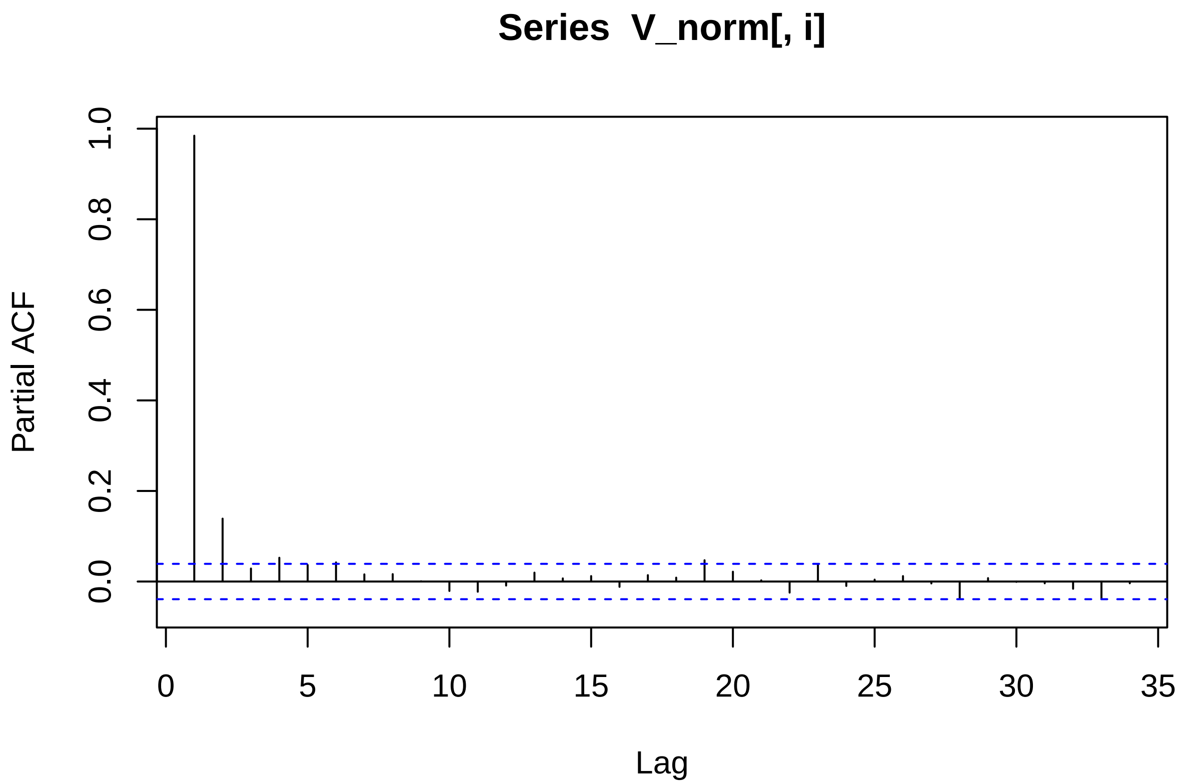
```
library(rmgarch)
library(MTS)
```

```
##
## Attaching package: 'MTS'
```

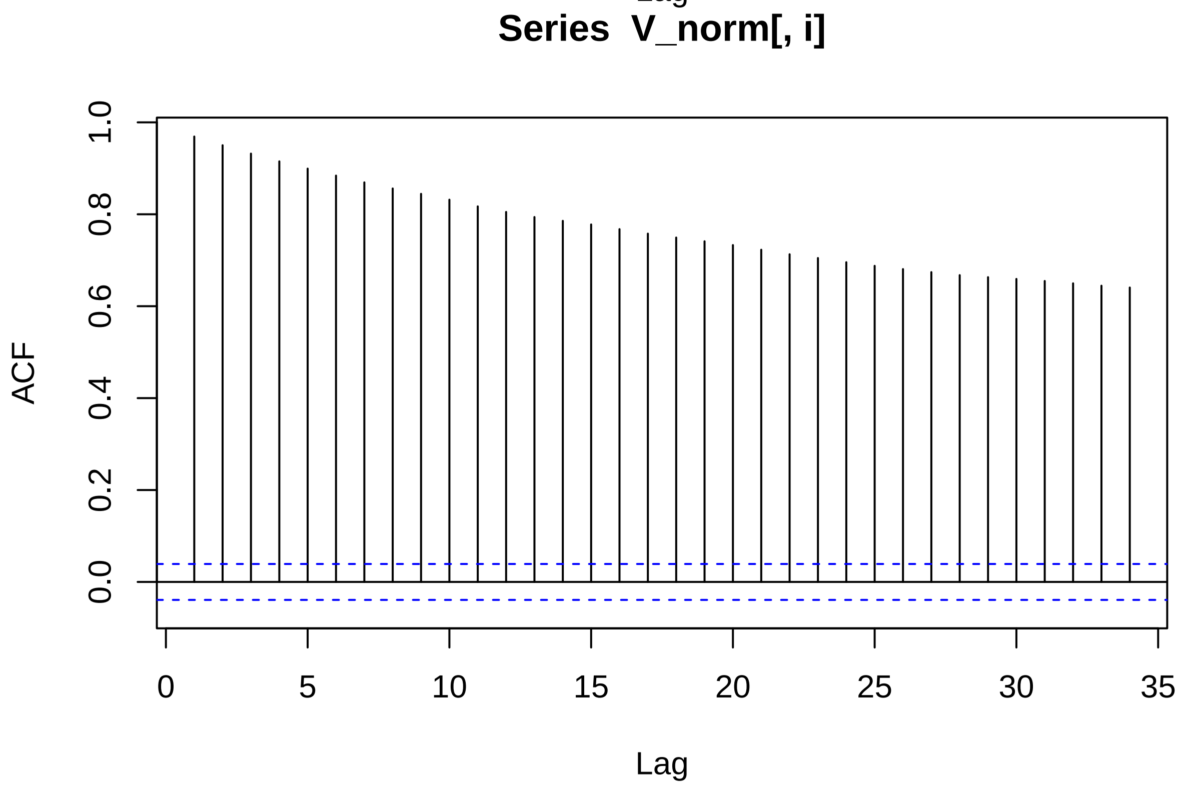
```
## The following object is masked from 'package:vars':
##
##     VAR
```

```
# Plot ACF and PACF for factors
for(i in 1:ncol(V_norm)){
  Acf(V_norm[,i])
  Pacf(V_norm[,i])
}
```

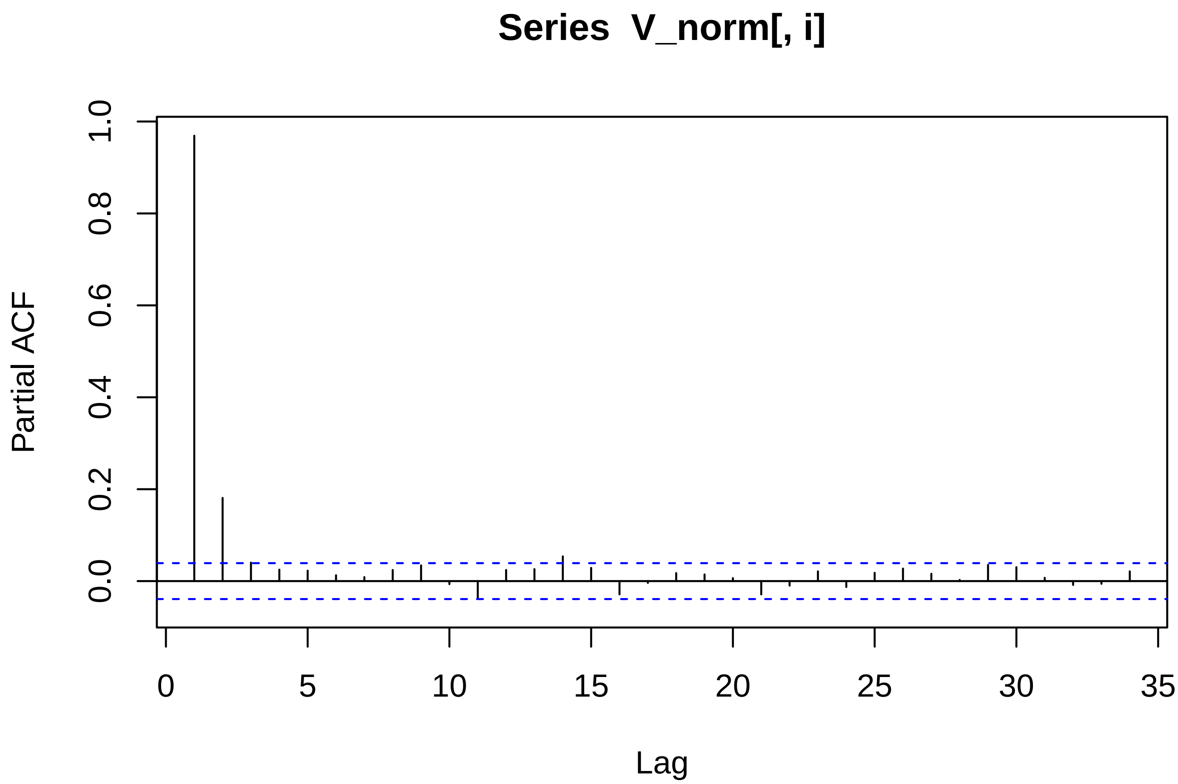


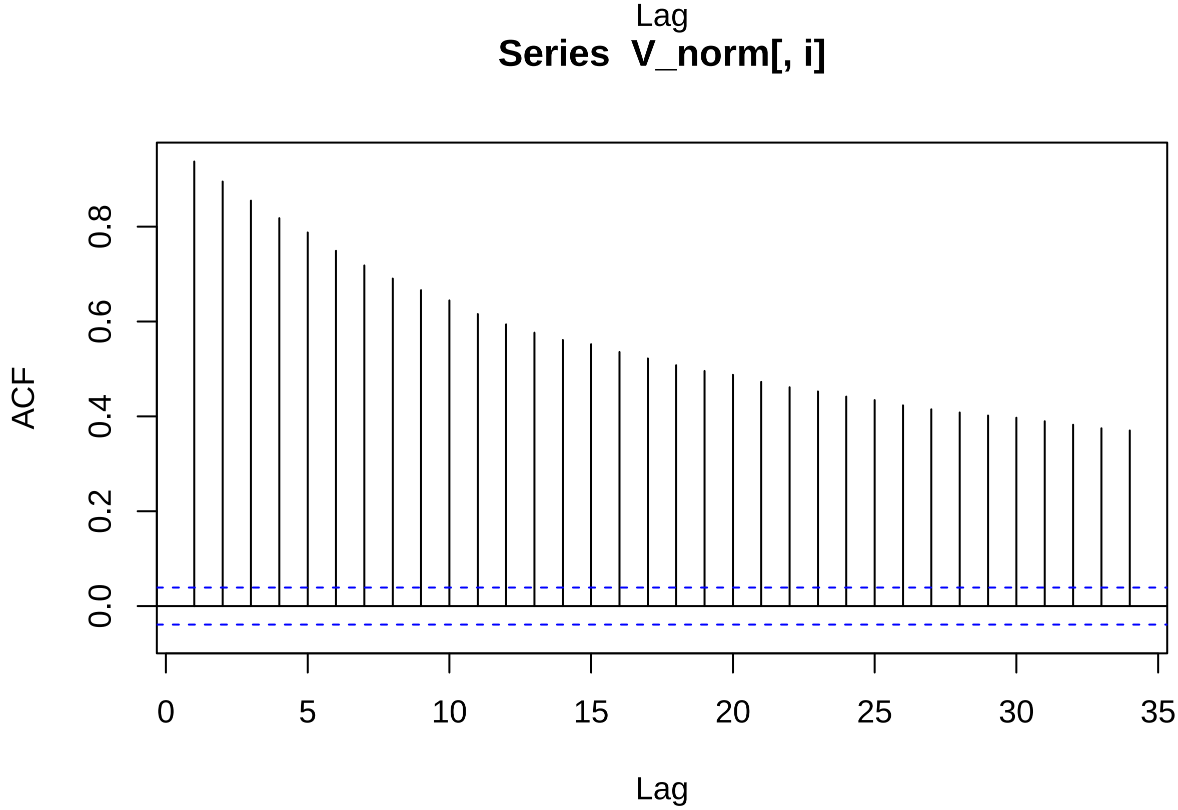


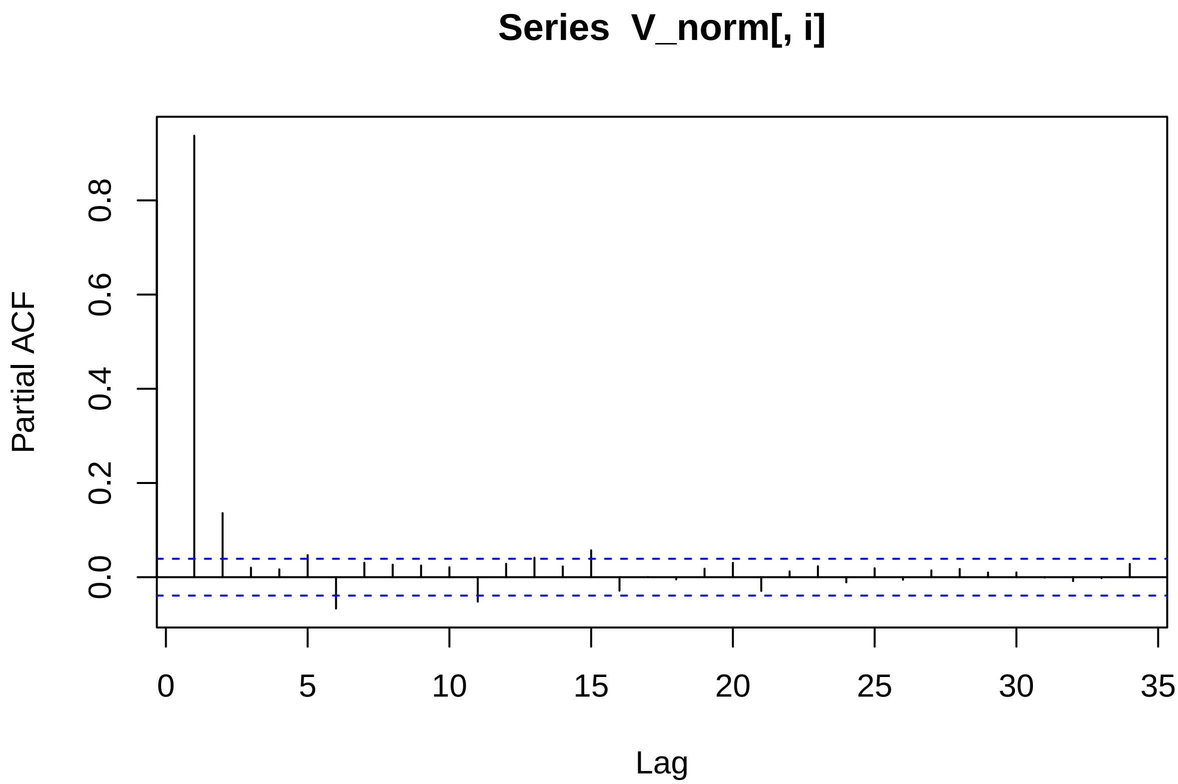
Series V_norm[, i]
Partial ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
35
Lag



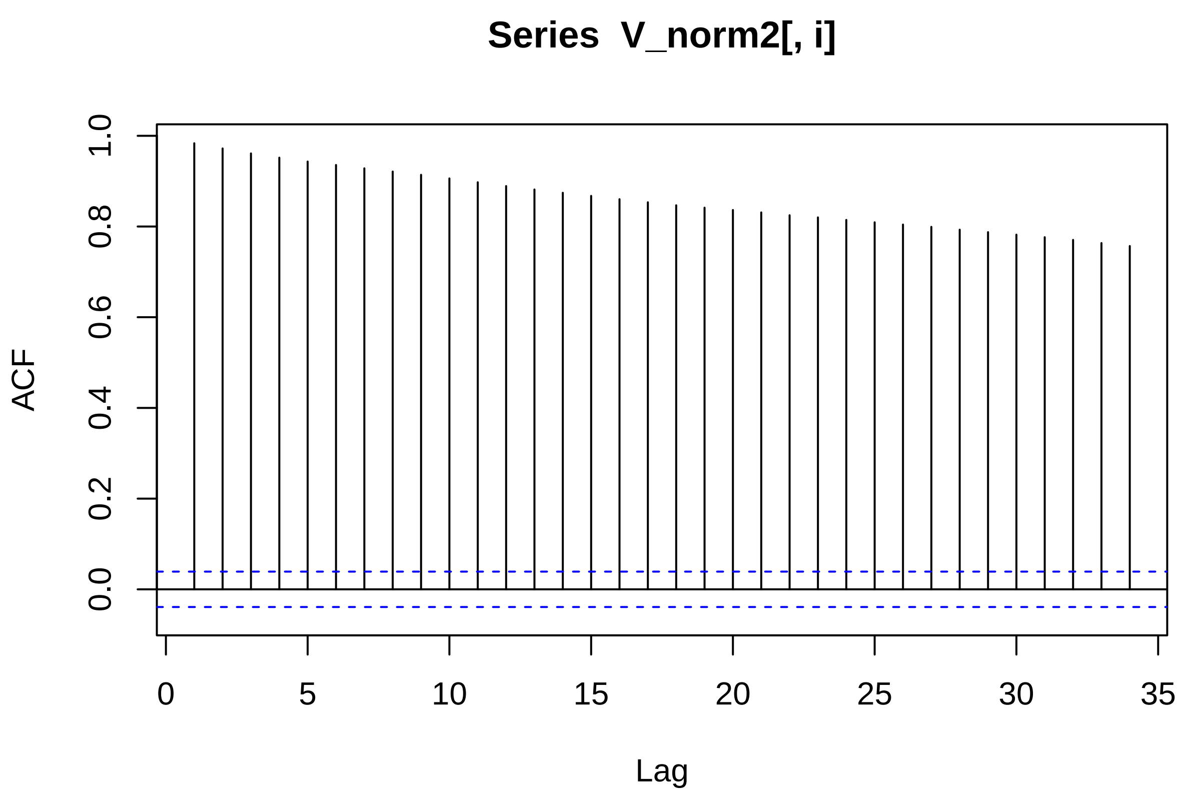
Series V_norm[, i]
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
35
Lag

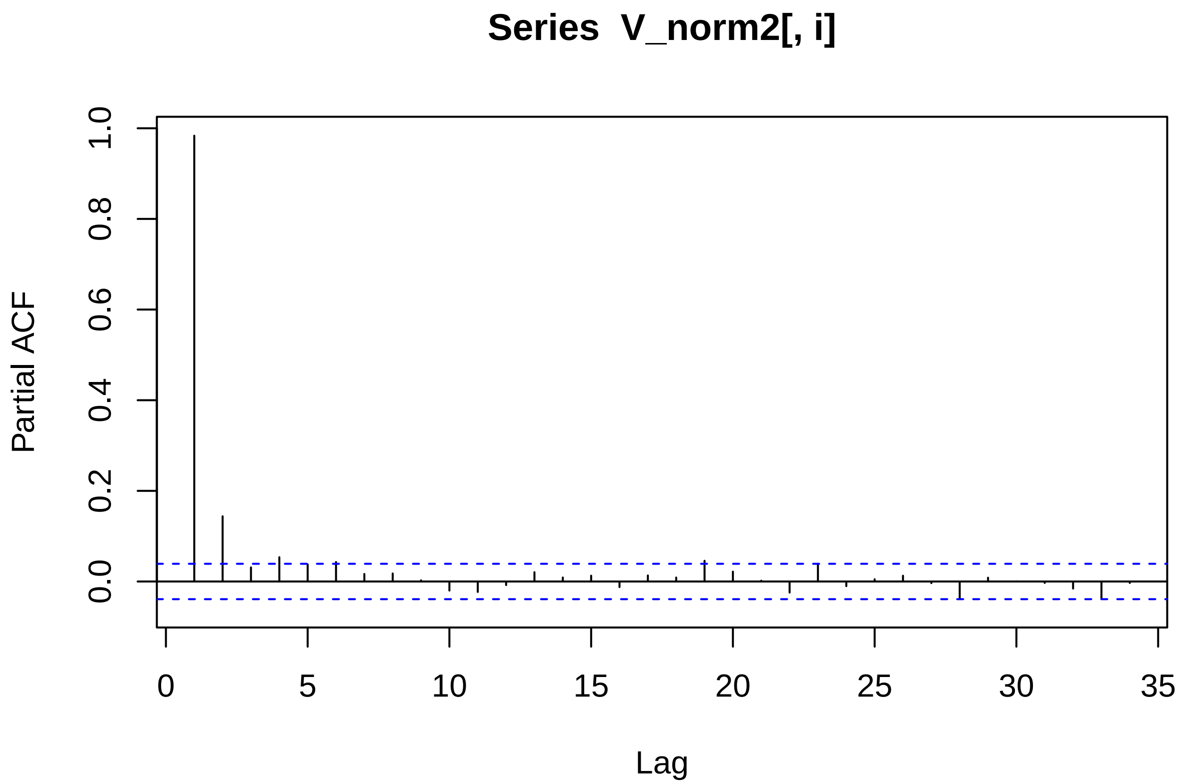

Series V_norm[, i]
Partial ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
35
Lag

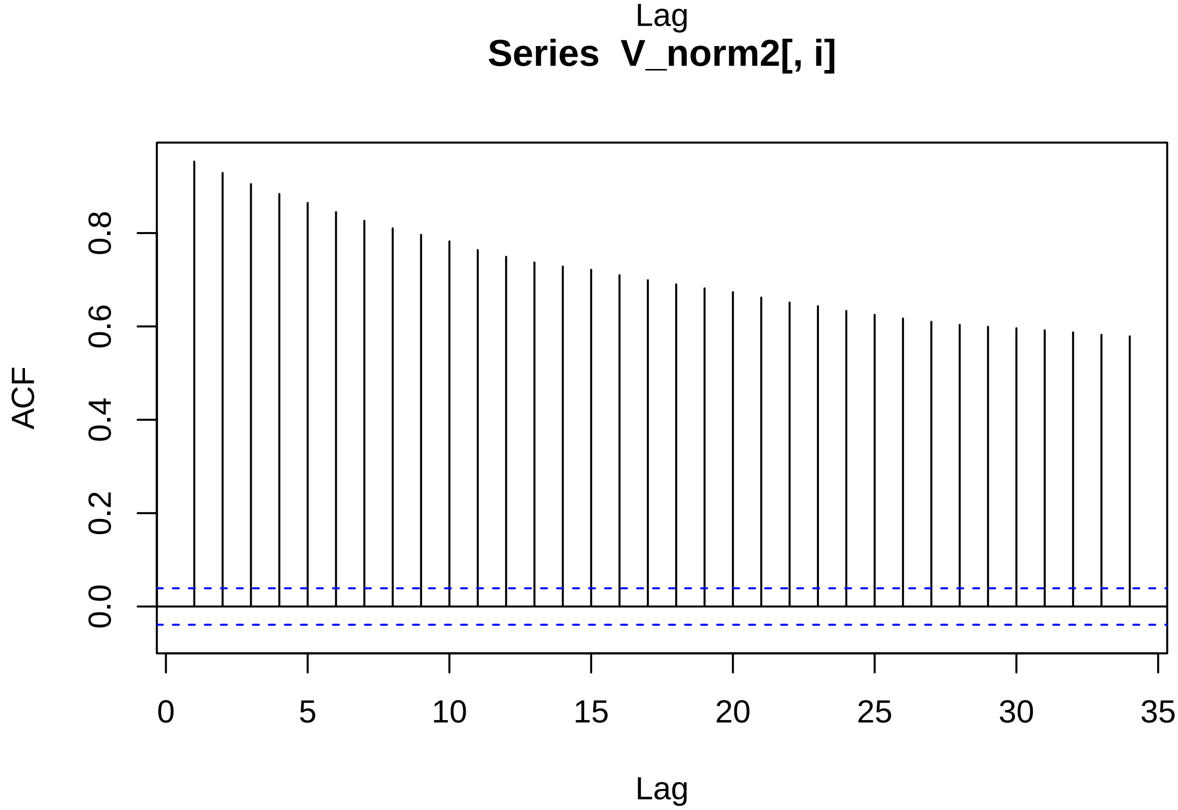

Series V_norm[, i]
ACF
0.0
0.2
0.4
0.6
0.8
0
5
10
15
20
25
30
35
Lag

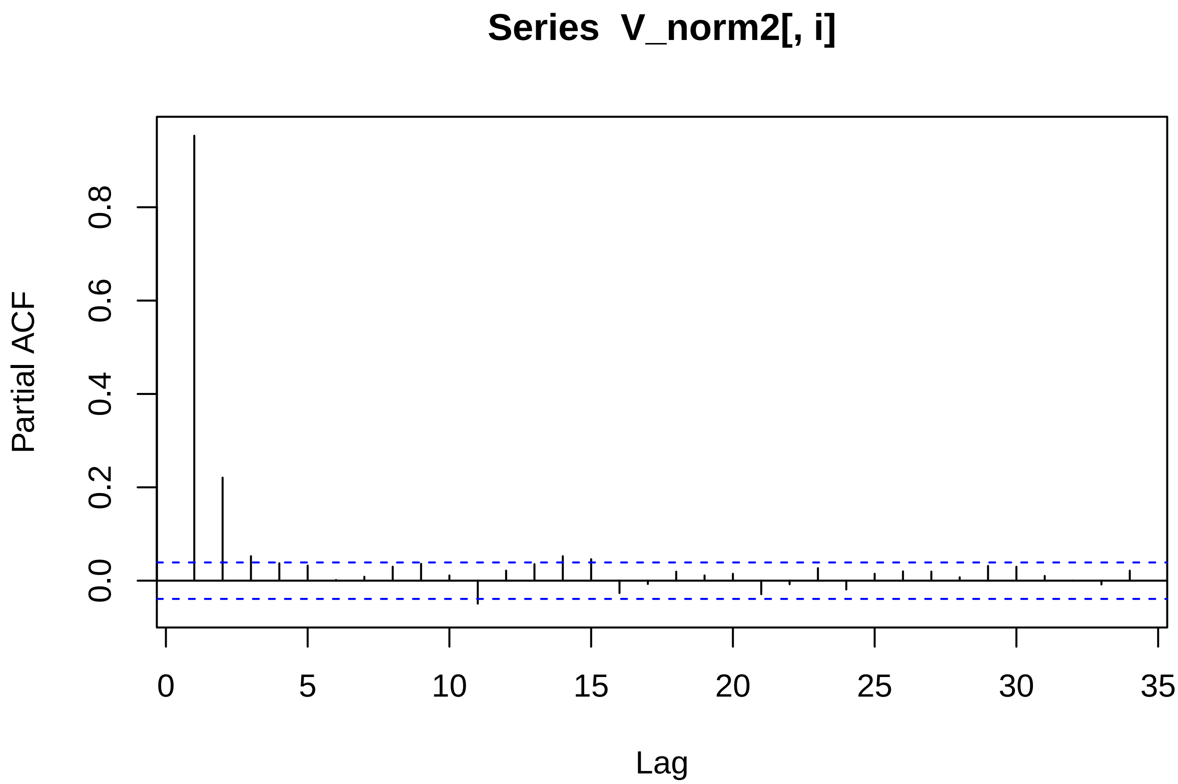


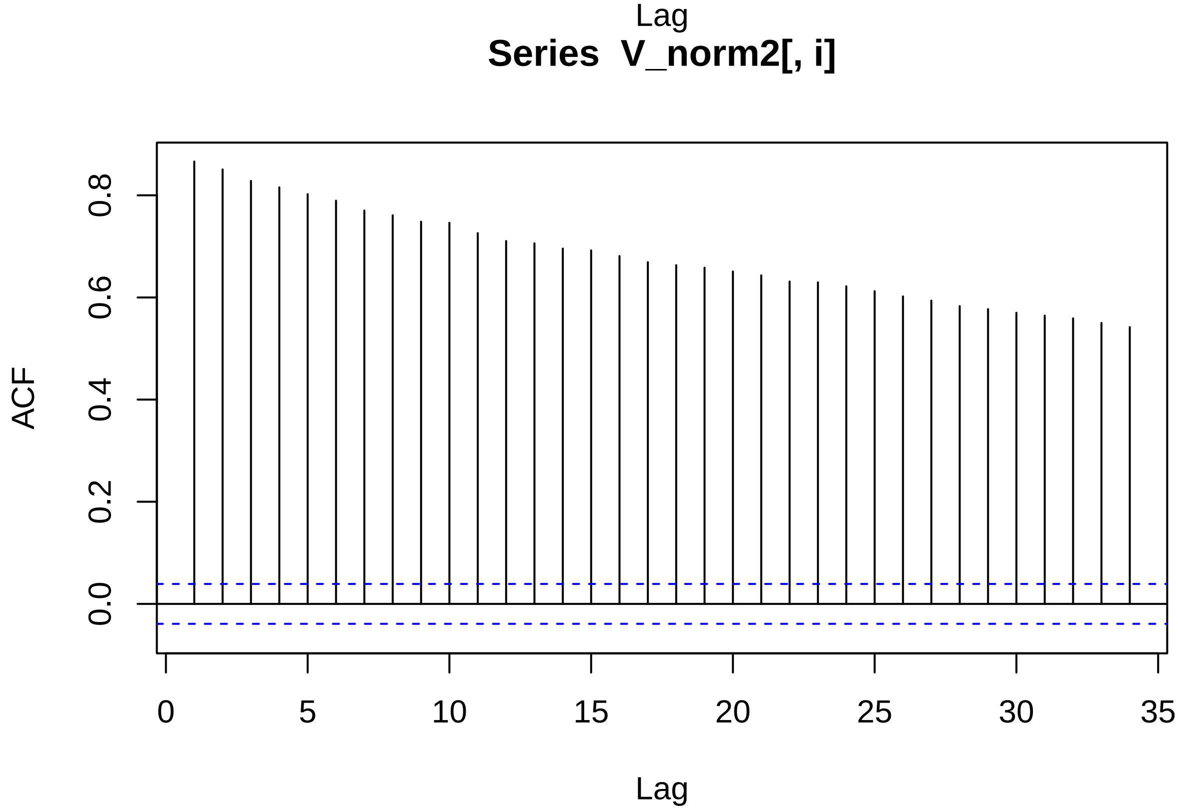

```r
for(i in 1:ncol(V_norm2)){
  Acf(V_norm2[,i])
  Pacf(V_norm2[,i])
}
```

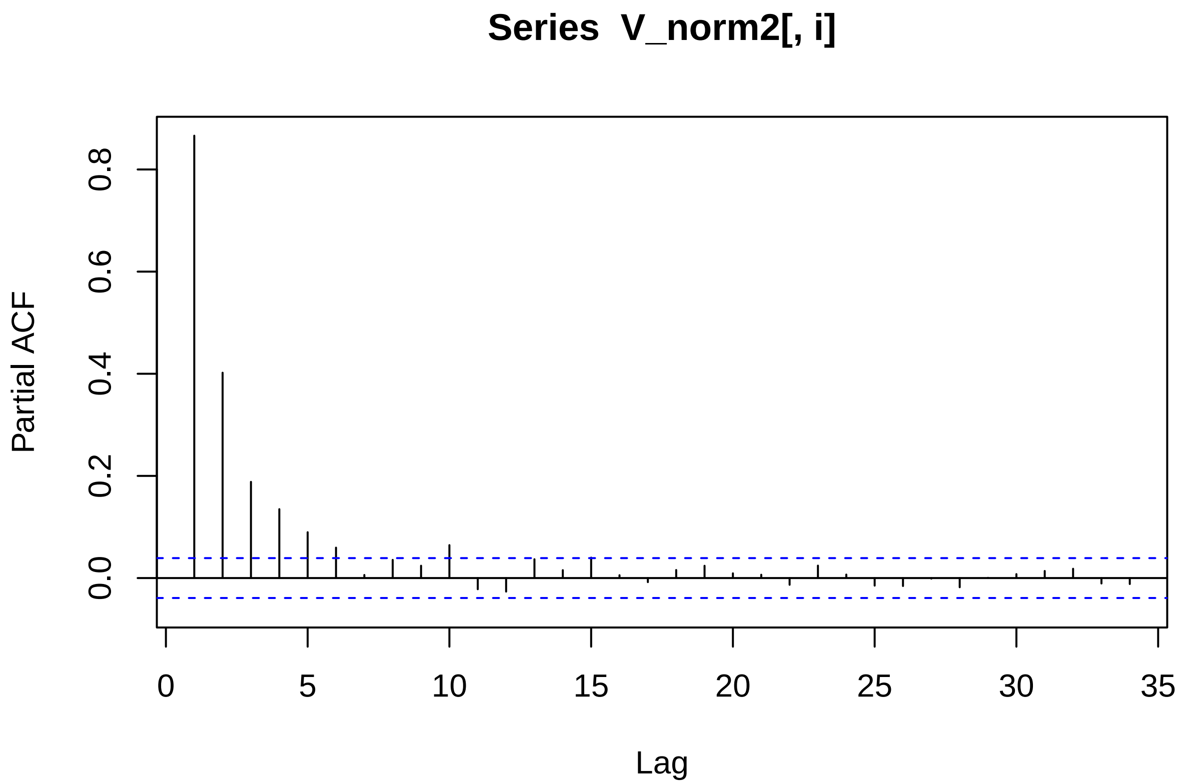


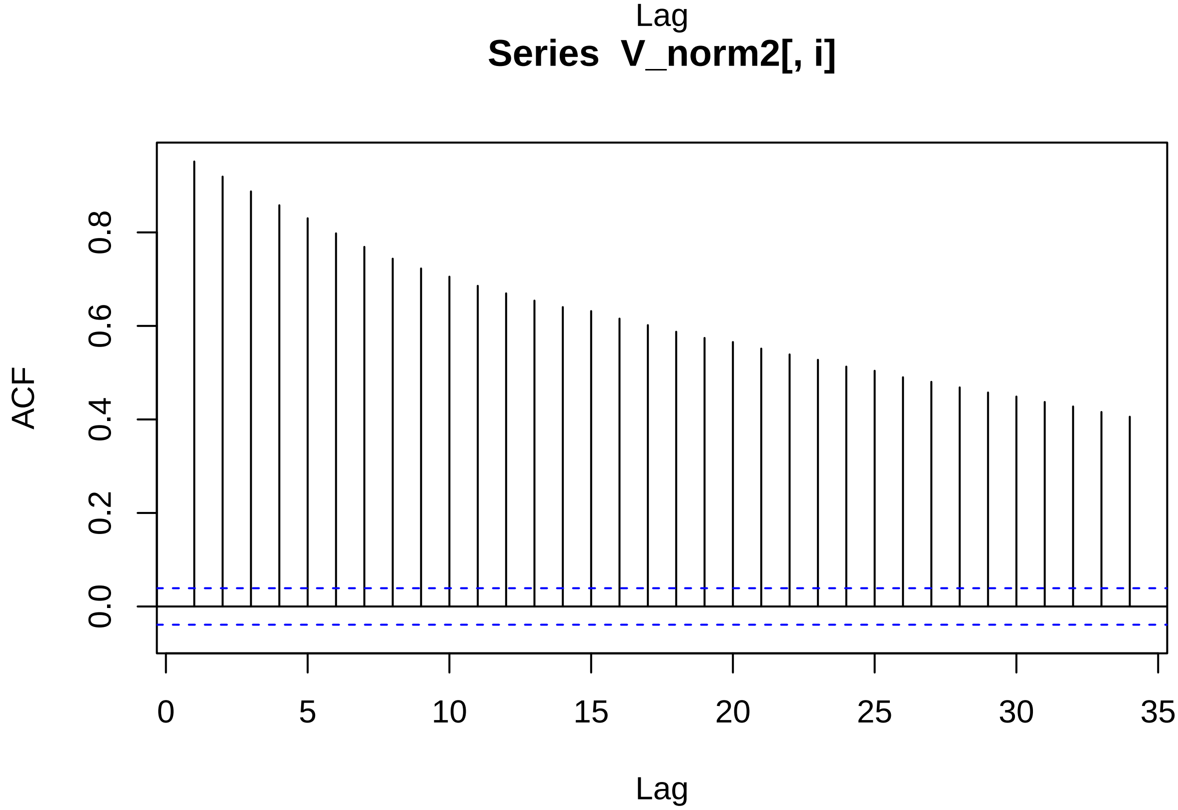

Series V_norm2[, i]
Partial ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
35
Lag

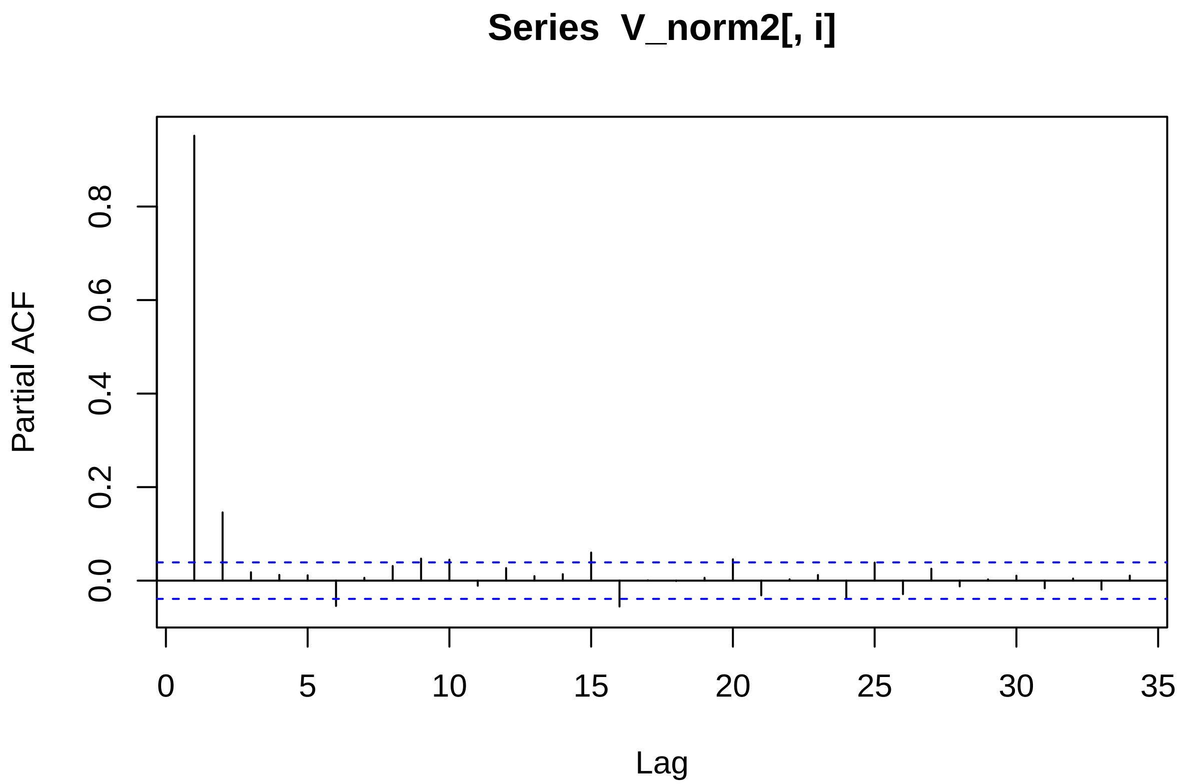

Series V_norm2[, i]
ACF
0.0
0.2
0.4
0.6
0.8
0
5
10
15
20
25
30
35
Lag

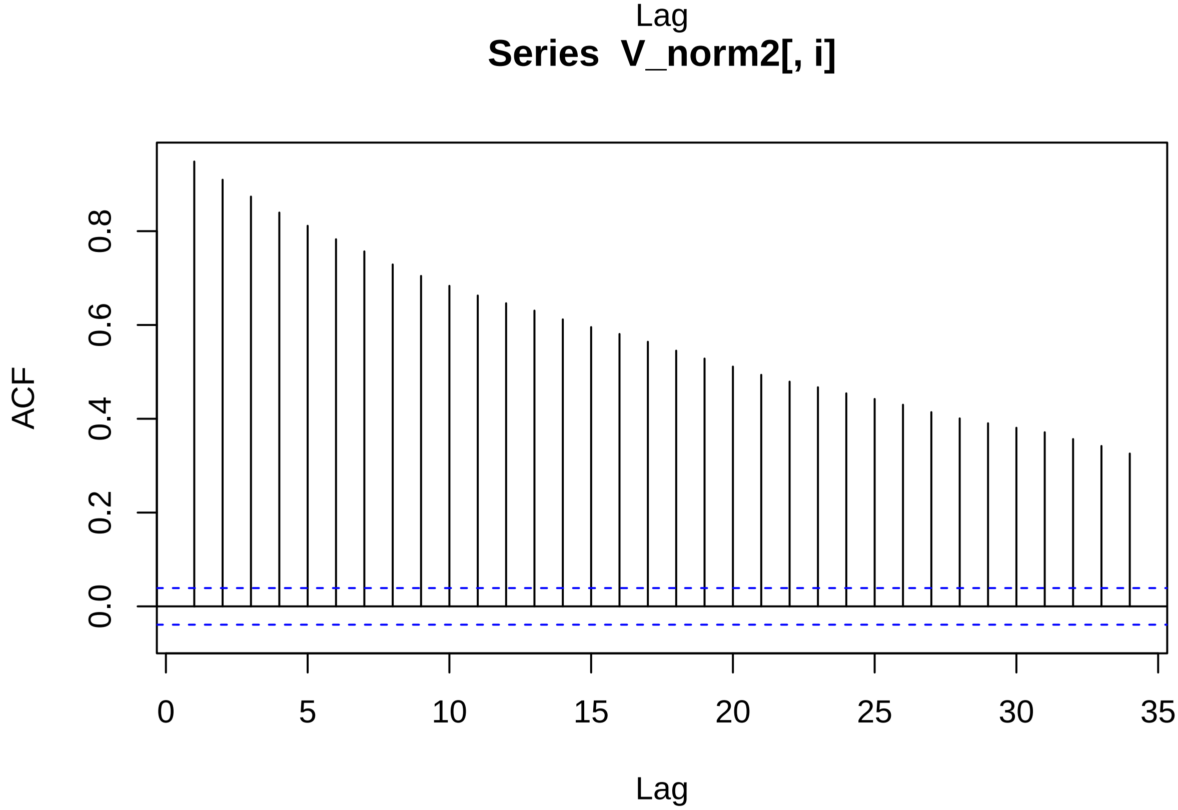

Series V_norm2[, i]
Partial ACF
0.0
0.2
0.4
0.6
0.8
0
5
10
15
20
25
30
35
Lag

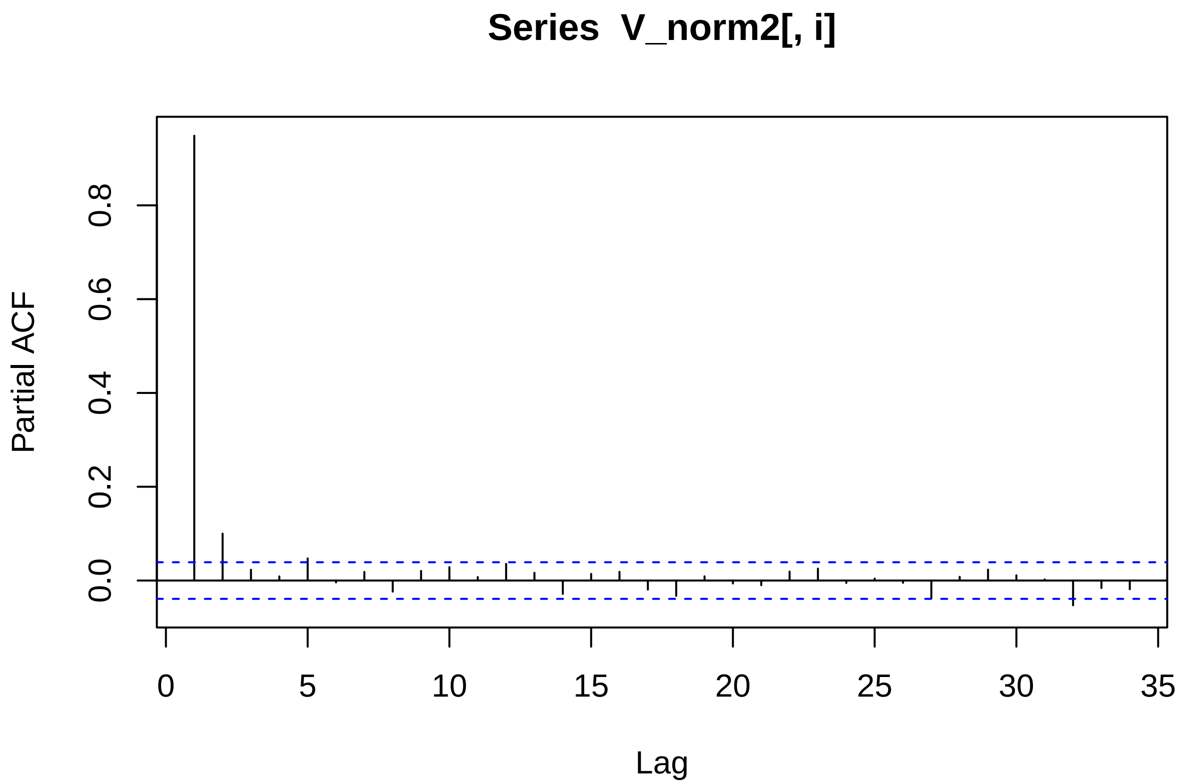

Series V_norm2[, i]
ACF
0.0
0.2
0.4
0.6
0.8
0
5
10
15
20
25
30
35
Lag


Series V_norm2[, i]
Partial ACF
0.0
0.2
0.4
0.6
0.8
0
5
10
15
20
25
30
35
Lag


Series V_norm2[, i]
ACF
0.0
0.2
0.4
0.6
0.8
0
5
10
15
20
25
30
35
Lag


Series V_norm2[, i]
Partial ACF
0.0
0.2
0.4
0.6
0.8
0
5
10
15
20
25
30
35
Lag


Series V_norm2[, i]
ACF
0.0
0.2
0.4
0.6
0.8
0
5
10
15
20
25
30
35
Lag



From VARselect, we conclude that the optimal lag for VAR model is 2 for both factors from R and R^2.

```
# Select optimal AR lag for factors from R
VARselect(V_norm,lag.max=10,type="none")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      6      4      2      6
##
## $criteria
##                    1             2             3             4             5
## AIC(n) -1.024225e+01 -1.031529e+01 -1.032422e+01 -1.033654e+01 -1.033863e+01
## HQ(n)  -1.023466e+01 -1.030011e+01 -1.030146e+01 -1.030618e+01 -1.030069e+01
## SC(n)  -1.022134e+01 -1.027348e+01 -1.026151e+01 -1.025292e+01 -1.023410e+01
## FPE(n)  3.563262e-05  3.312284e-05  3.282819e-05  3.242648e-05  3.235879e-05
##                    6             7             8             9            10
## AIC(n) -1.034946e+01 -1.034713e+01 -1.034364e+01 -1.034189e+01 -1.033658e+01
## HQ(n)  -1.030393e+01 -1.029402e+01 -1.028294e+01 -1.027360e+01 -1.026070e+01
## SC(n)  -1.022403e+01 -1.020080e+01 -1.017641e+01 -1.015375e+01 -1.012754e+01
## FPE(n)  3.201016e-05  3.208477e-05  3.219684e-05  3.225351e-05  3.242504e-05
```

```
# Select optimal AR lag for factors from R^2
VARselect(V_norm2,lag.max=10,type="none")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      6      4      2      6
##
## $criteria
##                    1             2             3             4             5
## AIC(n) -1.394237e+01 -1.421427e+01 -1.426164e+01 -1.428713e+01 -1.429598e+01
```

```
## HQ(n)  -1.392130e+01 -1.417212e+01 -1.419841e+01 -1.420282e+01 -1.419060e+01
## SC(n)  -1.388431e+01 -1.409814e+01 -1.408744e+01 -1.405486e+01 -1.400565e+01
## FPE(n)  8.808546e-07  6.711513e-07  6.401000e-07  6.239911e-07  6.184925e-07
##                   6             7             8             9            10
## AIC(n) -1.431275e+01 -1.430397e+01 -1.429329e+01 -1.428946e+01 -1.428457e+01
## HQ(n)  -1.418629e+01 -1.415643e+01 -1.412467e+01 -1.409976e+01 -1.407380e+01
## SC(n)  -1.396435e+01 -1.389750e+01 -1.382875e+01 -1.376685e+01 -1.370390e+01
## FPE(n)  6.082098e-07  6.135741e-07  6.201688e-07  6.225535e-07  6.256057e-07
```

However, the residuals of VAR(2) model do not pass the ARCH-LM test, so they have heteroscedasticity.

```r
# VAR model for R
VAR_pls <- vars::VAR(V_norm,p=2,type="none")
summary(VAR_pls)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: Comp.1, Comp.2, Comp.3
## Deterministic variables: none
## Sample size: 2517
## Log Likelihood: 2270.395
## Roots of the characteristic polynomial:
## 0.9906 0.9906 0.9498 0.1643 0.1336 0.03588
## Call:
## vars::VAR(y = V_norm, p = 2, type = "none")
##
##
## Estimation results for equation Comp.1:
## =======================================
## Comp.1 = Comp.1.l1 + Comp.2.l1 + Comp.3.l1 + Comp.1.l2 + Comp.2.l2 + Comp.3.l2
##
##            Estimate Std. Error t value Pr(>|t|)
## Comp.1.l1  0.844560   0.036493  23.143  < 2e-16 ***
## Comp.2.l1  0.006086   0.024277   0.251    0.802
## Comp.3.l1 -0.005230   0.010207  -0.512    0.608
## Comp.1.l2  0.143064   0.036648   3.904 9.72e-05 ***
## Comp.2.l2 -0.005786   0.024045  -0.241    0.810
## Comp.3.l2 -0.004181   0.010190  -0.410    0.682
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.1527 on 2511 degrees of freedom
## Multiple R-Squared: 0.9736,  Adjusted R-squared: 0.9735
## F-statistic: 1.544e+04 on 6 and 2511 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation Comp.2:
## =======================================
## Comp.2 = Comp.1.l1 + Comp.2.l1 + Comp.3.l1 + Comp.1.l2 + Comp.2.l2 + Comp.3.l2
##
##            Estimate Std. Error t value Pr(>|t|)
## Comp.1.l1  0.413865   0.053527   7.732 1.52e-14 ***
## Comp.2.l1  0.973852   0.035609  27.349  < 2e-16 ***
```

```
## Comp.3.l1 -0.024818   0.014971  -1.658   0.0975 .
## Comp.1.l2 -0.391018   0.053754  -7.274 4.63e-13 ***
## Comp.2.l2  0.009712   0.035268   0.275   0.7831
## Comp.3.l2  0.037787   0.014946   2.528   0.0115 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.224 on 2511 degrees of freedom
## Multiple R-Squared: 0.9455,  Adjusted R-squared: 0.9454
## F-statistic:  7259 on 6 and 2511 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation Comp.3:
## =======================================
## Comp.3 = Comp.1.l1 + Comp.2.l1 + Comp.3.l1 + Comp.1.l2 + Comp.2.l2 + Comp.3.l2
##
##           Estimate Std. Error t value Pr(>|t|)
## Comp.1.l1 -0.39058    0.09925  -3.935 8.53e-05 ***
## Comp.2.l1 -0.14403    0.06602  -2.181 0.029241 *
## Comp.3.l1  0.85046    0.02776  30.637  < 2e-16 ***
## Comp.1.l2  0.37090    0.09967   3.721 0.000202 ***
## Comp.2.l2  0.15357    0.06539   2.349 0.018924 *
## Comp.3.l2  0.10195    0.02771   3.679 0.000239 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.4154 on 2511 degrees of freedom
## Multiple R-Squared: 0.8907,  Adjusted R-squared: 0.8904
## F-statistic:  3411 on 6 and 2511 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##          Comp.1   Comp.2   Comp.3
## Comp.1  0.02332 -0.02812  0.04263
## Comp.2 -0.02812  0.05018 -0.06209
## Comp.3  0.04263 -0.06209  0.17238
##
## Correlation matrix of residuals:
##         Comp.1  Comp.2  Comp.3
## Comp.1  1.0000 -0.8219  0.6723
## Comp.2 -0.8219  1.0000 -0.6676
## Comp.3  0.6723 -0.6676  1.0000
```

```
# Residual check
arch.test(VAR_pls)
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object VAR_pls
## Chi-squared = 1076.1, df = 180, p-value < 2.2e-16
```

```r
# VAR model for R^2
VAR_pls2 <- vars::VAR(V_norm2,p=2,type="none")
summary(VAR_pls2)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: Comp.1, Comp.2, Comp.3, Comp.4, Comp.5
## Deterministic variables: none
## Sample size: 2517
## Log Likelihood: 62.31
## Roots of the characteristic polynomial:
## 0.9912 0.9912 0.9695 0.9572 0.9072 0.4375 0.1934 0.1336 0.08505 0.08505
## Call:
## vars::VAR(y = V_norm2, p = 2, type = "none")
##
##
## Estimation results for equation Comp.1:
## =======================================
## Comp.1 = Comp.1.l1 + Comp.2.l1 + Comp.3.l1 + Comp.4.l1 + Comp.5.l1 + Comp.1.l2 + Comp.2.l2 + Comp.3.l
##
##              Estimate Std. Error t value Pr(>|t|)
## Comp.1.l1  0.9533836  0.0393452  24.231  < 2e-16 ***
## Comp.2.l1  0.0437662  0.0266375   1.643   0.1005
## Comp.3.l1 -0.0915031  0.0100755  -9.082  < 2e-16 ***
## Comp.4.l1 -0.0001850  0.0089254  -0.021   0.9835
## Comp.5.l1  0.0195869  0.0107277   1.826   0.0680 .
## Comp.1.l2  0.0278910  0.0395797   0.705   0.4811
## Comp.2.l2 -0.0513389  0.0264192  -1.943   0.0521 .
## Comp.3.l2  0.0751201  0.0100784   7.454 1.24e-13 ***
## Comp.4.l2  0.0002245  0.0088809   0.025   0.9798
## Comp.5.l2 -0.0155048  0.0106695  -1.453   0.1463
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.154 on 2507 degrees of freedom
## Multiple R-Squared: 0.9731,  Adjusted R-squared: 0.973
## F-statistic:  9083 on 10 and 2507 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation Comp.2:
## =======================================
## Comp.2 = Comp.1.l1 + Comp.2.l1 + Comp.3.l1 + Comp.4.l1 + Comp.5.l1 + Comp.1.l2 + Comp.2.l2 + Comp.3.l
##
##             Estimate Std. Error t value Pr(>|t|)
## Comp.1.l1 -0.300836   0.071855  -4.187 2.93e-05 ***
## Comp.2.l1  1.063412   0.048648  21.859  < 2e-16 ***
## Comp.3.l1 -0.168621   0.018401  -9.164  < 2e-16 ***
## Comp.4.l1  0.014329   0.016300   0.879 0.379435
## Comp.5.l1 -0.008706   0.019592  -0.444 0.656803
## Comp.1.l2  0.260918   0.072284   3.610 0.000313 ***
## Comp.2.l2 -0.095925   0.048249  -1.988 0.046907 *
## Comp.3.l2  0.132554   0.018406   7.202 7.83e-13 ***
```

```
## Comp.4.l2 -0.019103   0.016219  -1.178 0.238977
## Comp.5.l2  0.012656   0.019486   0.650 0.516073
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.2813 on 2507 degrees of freedom
## Multiple R-Squared: 0.9243,  Adjusted R-squared: 0.924
## F-statistic:  3060 on 10 and 2507 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation Comp.3:
## ========================================
## Comp.3 = Comp.1.l1 + Comp.2.l1 + Comp.3.l1 + Comp.4.l1 + Comp.5.l1 + Comp.1.l2 + Comp.2.l2 + Comp.3.l
##
##           Estimate Std. Error t value Pr(>|t|)
## Comp.1.l1 -0.20294    0.12799  -1.586 0.112956
## Comp.2.l1 -0.01871    0.08665  -0.216 0.829019
## Comp.3.l1  0.52025    0.03278  15.873  < 2e-16 ***
## Comp.4.l1  0.10462    0.02903   3.603 0.000320 ***
## Comp.5.l1  0.07433    0.03490   2.130 0.033260 *
## Comp.1.l2  0.14598    0.12875   1.134 0.256991
## Comp.2.l2 -0.02110    0.08594  -0.246 0.806071
## Comp.3.l2  0.39071    0.03278  11.918  < 2e-16 ***
## Comp.4.l2 -0.10947    0.02889  -3.789 0.000155 ***
## Comp.5.l2 -0.07376    0.03471  -2.125 0.033668 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.501 on 2507 degrees of freedom
## Multiple R-Squared: 0.8034,  Adjusted R-squared: 0.8026
## F-statistic:  1025 on 10 and 2507 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation Comp.4:
## ========================================
## Comp.4 = Comp.1.l1 + Comp.2.l1 + Comp.3.l1 + Comp.4.l1 + Comp.5.l1 + Comp.1.l2 + Comp.2.l2 + Comp.3.l
##
##            Estimate Std. Error t value Pr(>|t|)
## Comp.1.l1 -0.070777   0.106868  -0.662    0.508
## Comp.2.l1  0.003044   0.072352   0.042    0.966
## Comp.3.l1  0.014390   0.027367   0.526    0.599
## Comp.4.l1  0.810398   0.024243  33.428  < 2e-16 ***
## Comp.5.l1 -0.034951   0.029138  -1.199    0.230
## Comp.1.l2  0.058917   0.107505   0.548    0.584
## Comp.2.l2 -0.032911   0.071759  -0.459    0.647
## Comp.3.l2 -0.036924   0.027375  -1.349    0.178
## Comp.4.l2  0.151750   0.024122   6.291 3.71e-10 ***
## Comp.5.l2  0.032171   0.028980   1.110    0.267
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
```

```
## Residual standard error: 0.4183 on 2507 degrees of freedom
## Multiple R-Squared: 0.9126,  Adjusted R-squared: 0.9123
## F-statistic:  2618 on 10 and 2507 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation Comp.5:
## =======================================
## Comp.5 = Comp.1.l1 + Comp.2.l1 + Comp.3.l1 + Comp.4.l1 + Comp.5.l1 + Comp.1.l2 + Comp.2.l2 + Comp.3.l
##
##           Estimate Std. Error t value Pr(>|t|)
## Comp.1.l1  0.12854    0.08798   1.461   0.1441
## Comp.2.l1 -0.28340    0.05956  -4.758 2.07e-06 ***
## Comp.3.l1  0.02057    0.02253   0.913   0.3613
## Comp.4.l1  0.01172    0.01996   0.587   0.5571
## Comp.5.l1  0.82541    0.02399  34.410  < 2e-16 ***
## Comp.1.l2 -0.14249    0.08850  -1.610   0.1075
## Comp.2.l2  0.28238    0.05907   4.780 1.85e-06 ***
## Comp.3.l2 -0.04570    0.02254  -2.028   0.0427 *
## Comp.4.l2 -0.01069    0.01986  -0.538   0.5905
## Comp.5.l2  0.13060    0.02386   5.474 4.83e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.3444 on 2507 degrees of freedom
## Multiple R-Squared: 0.9042,  Adjusted R-squared: 0.9038
## F-statistic:  2367 on 10 and 2507 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##           Comp.1   Comp.2  Comp.3    Comp.4     Comp.5
## Comp.1  0.023718  0.03685 0.04752 0.0151316 -0.0082680
## Comp.2  0.036847  0.07909 0.09496 0.0514839 -0.0114912
## Comp.3  0.047522  0.09496 0.25089 0.0215093  0.0509584
## Comp.4  0.015132  0.05148 0.02151 0.1749735  0.0005129
## Comp.5 -0.008268 -0.01149 0.05096 0.0005129  0.1185829
##
## Correlation matrix of residuals:
##         Comp.1  Comp.2 Comp.3   Comp.4    Comp.5
## Comp.1  1.0000  0.8507 0.6160 0.234886 -0.155902
## Comp.2  0.8507  1.0000 0.6741 0.437636 -0.118654
## Comp.3  0.6160  0.6741 1.0000 0.102659  0.295436
## Comp.4  0.2349  0.4376 0.1027 1.000000  0.003561
## Comp.5 -0.1559 -0.1187 0.2954 0.003561  1.000000
```

```
# Residual check
arch.test(VAR_pls2)
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object VAR_pls2
## Chi-squared = 4344.4, df = 1125, p-value < 2.2e-16
```

The optimal model for factors from both R and R^2 is DCC(1,1)+VAR(2). It passes the ARCH-LM test and Ljung-Box test for residuals.

```
# Model for R
xspec_pls_1 <- ugarchspec(mean.model = list(armaOrder = c(2, 0),include.mean=TRUE), variance.model = lis
xspec_pls_2 <- ugarchspec(mean.model = list(armaOrder = c(2, 0),include.mean=TRUE), variance.model = lis
xspec_pls_3 <- ugarchspec(mean.model = list(armaOrder = c(2, 0),include.mean=TRUE), variance.model = lis
uspec_pls <- multispec(c(xspec_pls_1, xspec_pls_2, xspec_pls_3))
spec1_pls <- dccspec(uspec = uspec_pls, VAR=TRUE, robust=TRUE,lag=2, dccOrder = c(1,1), model="DCC", di
fit_pls <- dccfit(spec1_pls, data = V_norm, fit.control = list(eval.se = TRUE), out.sample=N_test)
fit_pls
```

```
##
## *---------------------------------*
## *          DCC GARCH Fit          *
## *---------------------------------*
##
## Distribution         :  mvt
## Model                :  DCC(1,1)
## No. Parameters       :  42
## [VAR GARCH DCC UncQ] : [21+15+3+3]
## No. Series           :  3
## No. Obs.             :  1519
## Log-Likelihood       :  1769.586
## Av.Log-Likelihood    :  1.16
##
## Optimal Parameters
## -----------------------------------
##                  Estimate  Std. Error  t value Pr(>|t|)
## [Comp 1].omega  -0.210650    0.088715  -2.3745 0.017575
## [Comp 1].alpha1  0.122209    0.033651   3.6317 0.000282
## [Comp 1].beta1   0.946486    0.022929  41.2793 0.000000
## [Comp 1].gamma1  0.177985    0.051718   3.4415 0.000579
## [Comp 1].shape   5.527496    0.695499   7.9475 0.000000
## [Comp 2].omega  -0.212803    0.202720  -1.0497 0.293837
## [Comp 2].alpha1 -0.044229    0.037091  -1.1925 0.233079
## [Comp 2].beta1   0.932070    0.065070  14.3242 0.000000
## [Comp 2].gamma1  0.165429    0.076377   2.1660 0.030314
## [Comp 2].shape   7.209477    1.123224   6.4186 0.000000
## [Comp 3].omega  -0.231751    0.130895  -1.7705 0.076641
## [Comp 3].alpha1  0.048355    0.044840   1.0784 0.280854
## [Comp 3].beta1   0.879496    0.068122  12.9106 0.000000
## [Comp 3].gamma1  0.127301    0.054449   2.3380 0.019389
## [Comp 3].shape   6.292888    0.871201   7.2232 0.000000
## [Joint]dcca1     0.053327    0.011957   4.4598 0.000008
## [Joint]dccb1     0.928628    0.019729  47.0685 0.000000
## [Joint]mshape    6.096558    0.453523  13.4427 0.000000
##
## Information Criteria
## ---------------------
##
## Akaike       -2.2746
## Bayes        -2.1274
## Shibata      -2.2761
## Hannan-Quinn -2.2198
```

```
##
##
## Elapsed time : 3.739161
```

```r
fit_pls@model$varcoef
```

```
##         Comp.1.l1    Comp.2.l1    Comp.3.l1  Comp.1.l2   Comp.2.l2   Comp.3.l2
## Comp.1  0.8572383  0.052439548  0.002068752  0.1197067 -0.04186800 -0.01556747
## Comp.2  0.3946832  0.949479730 -0.006748666 -0.3630537  0.02284460  0.02786001
## Comp.3 -0.3307945 -0.004284883  0.848002078  0.2908205  0.04402008  0.09067868
##              const
## Comp.1 -0.01523372
## Comp.2  0.01622051
## Comp.3 -0.03035444
```

```r
# In-sample
print('In-sample MSEs are')
```

```
## [1] "In-sample MSEs are"
```

```r
apply(fit_pls@model$residuals^2,2,mean)
```

```
## [1] 0.03362863 0.04974834 0.16356429
```

```r
forcast_dcc_pls <- dccforecast(fit_pls,n.ahead=1,n.roll=N_test)
fitted_pls <- t(fitted(forcast_dcc_pls)[1,,])
sigma_pls <- t(sigma(forcast_dcc_pls)[1,,])
mse_pls_temp <- (fitted_pls-V_norm[(nrow(V_norm)-N_test):nrow(V_norm),])^2
print('MSEs are')
```

```
## [1] "MSEs are"
```

```r
apply(mse_pls_temp,2,mean)
```

```
##      Comp 1      Comp 2      Comp 3
## 0.002004707 0.006474068 0.026270767
```

```r
print('Means of sd are')
```

```
## [1] "Means of sd are"
```

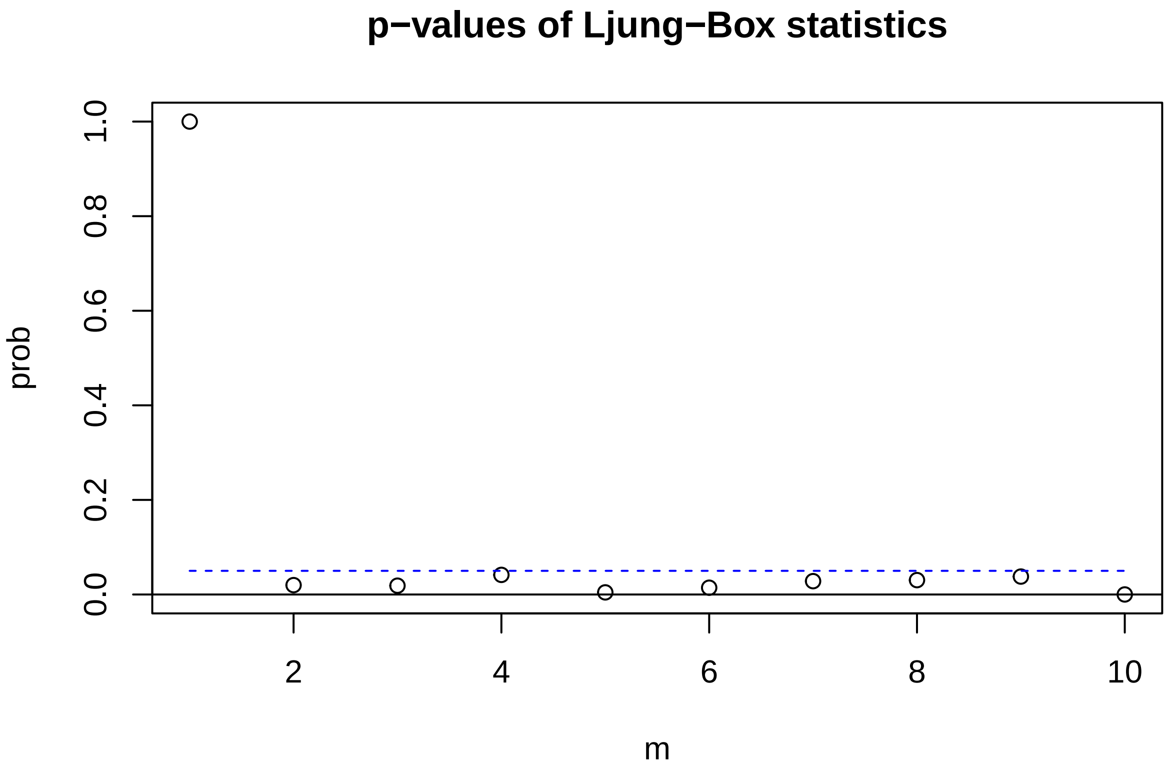
```r
apply(sigma_pls,2,mean)
```

```
##    Comp 1    Comp 2    Comp 3
## 0.1422534 0.2218959 0.4133676
```

```r
# Multivariate Ljung Box test for residuals
res_pls <- fit_pls@mfit$stdresid
mq(res_pls,lag=10,adj=2)
```

```
## Ljung-Box Statistics:
##         m       Q(m)     df    p-value
##  [1,]   1.0      11.1     7.0     1.00
##  [2,]   2.0      29.7    16.0     0.02
##  [3,]   3.0      41.9    25.0     0.02
##  [4,]   4.0      49.5    34.0     0.04
##  [5,]   5.0      71.2    43.0     0.00
##  [6,]   6.0      76.8    52.0     0.01
##  [7,]   7.0      83.8    61.0     0.03
##  [8,]   8.0      93.8    70.0     0.03
```

```
##  [9,]   9.0     102.7    79.0     0.04
## [10,]  10.0     151.8    88.0     0.00
```




```
# Multi-variate ARCH LM test for residuals
MarchTest(res_pls,lag=10)
```

```
## Q(m) of squared series(LM test):
## Test statistic:  16.91351  p-value:  0.07629986
## Rank-based Test:
## Test statistic:  8.621854  p-value:  0.5683269
## Q_k(m) of squared series:
## Test statistic:  470.6955  p-value:  0
## Robust Test(5%) :  109.1461  p-value:  0.08291687
```

```
# Model for R^2
xspec_pls_1 <- ugarchspec(mean.model = list(armaOrder = c(2, 0),include.mean=TRUE), variance.model = li
xspec_pls_2 <- ugarchspec(mean.model = list(armaOrder = c(2, 0),include.mean=TRUE), variance.model = li
xspec_pls_3 <- ugarchspec(mean.model = list(armaOrder = c(2, 0),include.mean=TRUE), variance.model = li
xspec_pls_4 <- ugarchspec(mean.model = list(armaOrder = c(2, 0),include.mean=TRUE), variance.model = li
xspec_pls_5 <- ugarchspec(mean.model = list(armaOrder = c(2, 0),include.mean=TRUE), variance.model = li
uspec_pls2 <- multispec(c(xspec_pls_1, xspec_pls_2, xspec_pls_3,xspec_pls_4,xspec_pls_5))
spec1_pls2 <- dccspec(uspec = uspec_pls2, VAR=TRUE, robust=TRUE,lag=2, dccOrder = c(1,1), model="DCC", 
fit_pls2 <- dccfit(spec1_pls2, data = V_norm2, fit.control = list(eval.se = TRUE), out.sample=N_test)
fit_pls2
```

```
##
## *---------------------------------*
## *          DCC GARCH Fit          *
## *---------------------------------*
##
## Distribution         :  mvt
```

```
## Model                 :  DCC(1,1)
## No. Parameters        :  93
## [VAR GARCH DCC UncQ] : [55+25+3+10]
## No. Series            :  5
## No. Obs.              :  1519
## Log-Likelihood        :  905.8643
## Av.Log-Likelihood     :  0.6
##
## Optimal Parameters
## -----------------------------------
##                   Estimate  Std. Error  t value Pr(>|t|)
## [Comp 1].omega  -0.230342    0.115944 -1.98667 0.046959
## [Comp 1].alpha1  0.138518    0.041005  3.37808 0.000730
## [Comp 1].beta1   0.941189    0.030285 31.07805 0.000000
## [Comp 1].gamma1  0.161771    0.054629  2.96127 0.003064
## [Comp 1].shape   5.689902    0.779345  7.30088 0.000000
## [Comp 2].omega  -0.288755    0.146335 -1.97325 0.048467
## [Comp 2].alpha1  0.084132    0.036887  2.28079 0.022561
## [Comp 2].beta1   0.891664    0.055086 16.18682 0.000000
## [Comp 2].gamma1  0.163037    0.053998  3.01932 0.002533
## [Comp 2].shape   6.291117    0.908894  6.92173 0.000000
## [Comp 3].omega  -0.142789    0.189519 -0.75343 0.451192
## [Comp 3].alpha1  0.046270    0.033083  1.39860 0.161935
## [Comp 3].beta1   0.906177    0.128496  7.05219 0.000000
## [Comp 3].gamma1  0.210651    0.184348  1.14268 0.253171
## [Comp 3].shape   5.850682    0.855729  6.83707 0.000000
## [Comp 4].omega  -0.070523    0.065770 -1.07227 0.283600
## [Comp 4].alpha1  0.031326    0.019614  1.59718 0.110226
## [Comp 4].beta1   0.964250    0.034225 28.17350 0.000000
## [Comp 4].gamma1  0.120868    0.068099  1.77489 0.075917
## [Comp 4].shape   6.171364    0.887355  6.95479 0.000000
## [Comp 5].omega  -0.065291    0.046533 -1.40312 0.160579
## [Comp 5].alpha1  0.050978    0.024321  2.09610 0.036074
## [Comp 5].beta1   0.973003    0.019121 50.88559 0.000000
## [Comp 5].gamma1  0.118812    0.030673  3.87351 0.000107
## [Comp 5].shape   6.844106    1.112652  6.15117 0.000000
## [Joint]dcca1     0.042925    0.007009  6.12427 0.000000
## [Joint]dccb1     0.930205    0.014559 63.89113 0.000000
## [Joint]mshape    6.460573    0.373088 17.31649 0.000000
##
## Information Criteria
## ---------------------
##
## Akaike       -1.07026
## Bayes        -0.74419
## Shibata      -1.07720
## Hannan-Quinn -0.94887
##
##
## Elapsed time : 6.238932
```

```
fit_pls2@model$varcoef
```

```
##          Comp.1.l1    Comp.2.l1    Comp.3.l1   Comp.4.l1    Comp.5.l1
## Comp.1  0.93498321 -0.002202762 -0.059358821 0.011511213  0.001582237
```

```
## Comp.2 -0.38289342  0.982758957 -0.102544003 0.020511078 -0.034033528
## Comp.3 -0.39903427 -0.199905409  0.672518398 0.104212166 -0.010409977
## Comp.4 -0.08870651 -0.028436183 -0.004519977 0.844413872 -0.031459042
## Comp.5  0.07611627 -0.208058370 -0.016331056 0.007287677  0.831099729
##          Comp.1.l2   Comp.2.l2   Comp.3.l2   Comp.4.l2     Comp.5.l2
## Comp.1  0.04051139 -0.01391677  0.04401740 -0.01658546 -0.0008733358
## Comp.2  0.33464987 -0.03622044  0.06829509 -0.03417504  0.0326750450
## Comp.3  0.32262025  0.12799459  0.22838405 -0.11693634  0.0006331959
## Comp.4  0.06787649 -0.01308616 -0.01266268  0.09948524  0.0379589228
## Comp.5 -0.09050138  0.20425074 -0.02191768 -0.02357943  0.1274962827
##              const
## Comp.1 -0.01587200
## Comp.2 -0.02658336
## Comp.3 -0.04070360
## Comp.4 -0.01888248
## Comp.5 -0.00264566
```

```r
# In-sample
print('In-sample MSEs are')
```

```
## [1] "In-sample MSEs are"
```

```r
apply(fit_pls2@model$residuals^2,2,mean)
```

```
## [1] 0.03417489 0.08060138 0.26338328 0.15720701 0.10484089
```

```r
# Out-of-sample
forcast_dcc_pls2 <- dccforecast(fit_pls2,n.ahead=1,n.roll=N_test)
fitted_pls2 <- t(fitted(forcast_dcc_pls2)[1,,])
sigma_pls2 <- t(sigma(forcast_dcc_pls2)[1,,])
mse_pls_temp2 <- (fitted_pls2-V_norm2[(nrow(V_norm2)-N_test):nrow(V_norm2),])^2
print('MSEs are')
```

```
## [1] "MSEs are"
```

```r
apply(mse_pls_temp2,2,mean)
```

```
##      Comp 1      Comp 2      Comp 3      Comp 4      Comp 5
## 0.001918897 0.013083255 0.068013260 0.019218988 0.010761554
```

```r
print('Means of sd are')
```

```
## [1] "Means of sd are"
```

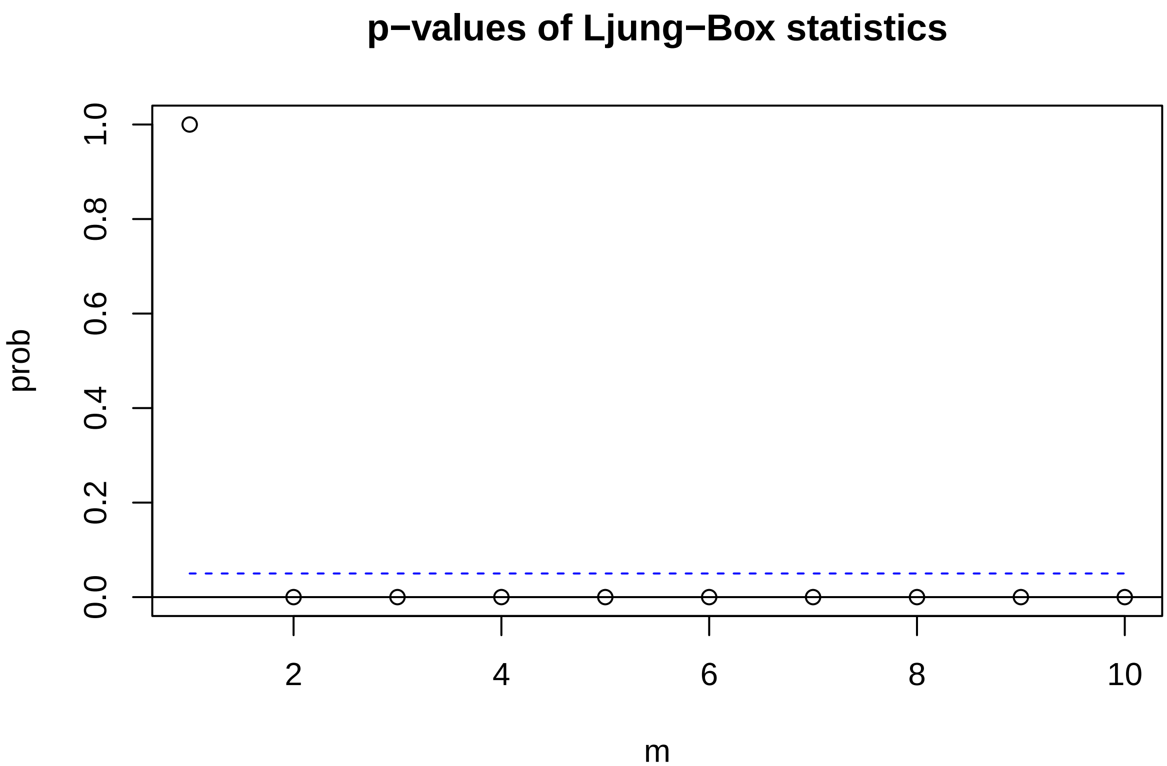
```r
apply(sigma_pls2,2,mean)
```

```
##    Comp 1    Comp 2    Comp 3    Comp 4    Comp 5
## 0.1450583 0.2816587 0.5035648 0.4061554 0.3462916
```

```r
# Multivariate Ljung Box test for residuals
res_pls2 <- fit_pls2@mfit$stdresid
mq(res_pls2,lag=10,adj=2)
```

```
## Ljung-Box Statistics:
##         m       Q(m)     df    p-value
##  [1,]   1.0      43.3    23.0        1
##  [2,]   2.0     116.7    48.0        0
##  [3,]   3.0     140.5    73.0        0
##  [4,]   4.0     163.6    98.0        0
```

```
## [5,]   5.0     219.3   123.0         0
## [6,]   6.0     260.4   148.0         0
## [7,]   7.0     288.1   173.0         0
## [8,]   8.0     320.7   198.0         0
## [9,]   9.0     337.1   223.0         0
## [10,] 10.0     421.7   248.0         0
```




```r
# Multi-variate ARCH LM test for residuals
MarchTest(res_pls2,lag=10)
```

```
## Q(m) of squared series(LM test):
## Test statistic:  13.46799  p-value:  0.1986698
## Rank-based Test:
## Test statistic:  16.22592  p-value:  0.09334515
## Q_k(m) of squared series:
## Test statistic:  843.1999  p-value:  0
## Robust Test(5%) :  290.6471  p-value:  0.0394493
```

Cohen's factors

```r
rm(list=ls())
library(rugarch)
library(car)
library(tseries)
library(Dowd)
library(seastests)
library(vars)

# import data
Data_co <- read.csv("/Users/benjye/Dropbox/Pricing/Data_R/Data_co.csv",header=TRUE)
```

```r
X_co <- Data_co[,2:3]
S <- Data_co[,4]
R <- diff(log(S))
S <- S[-length(S)]
X_co <- X_co[-nrow(X_co),]
```

The factors are both stationary,

```r
# ADF test for factors
adf.test(X_co[,1],k=10)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  X_co[, 1]
## Dickey-Fuller = -3.6102, Lag order = 10, p-value = 0.03164
## alternative hypothesis: stationary
```

```r
adf.test(X_co[,2],k=10)
```

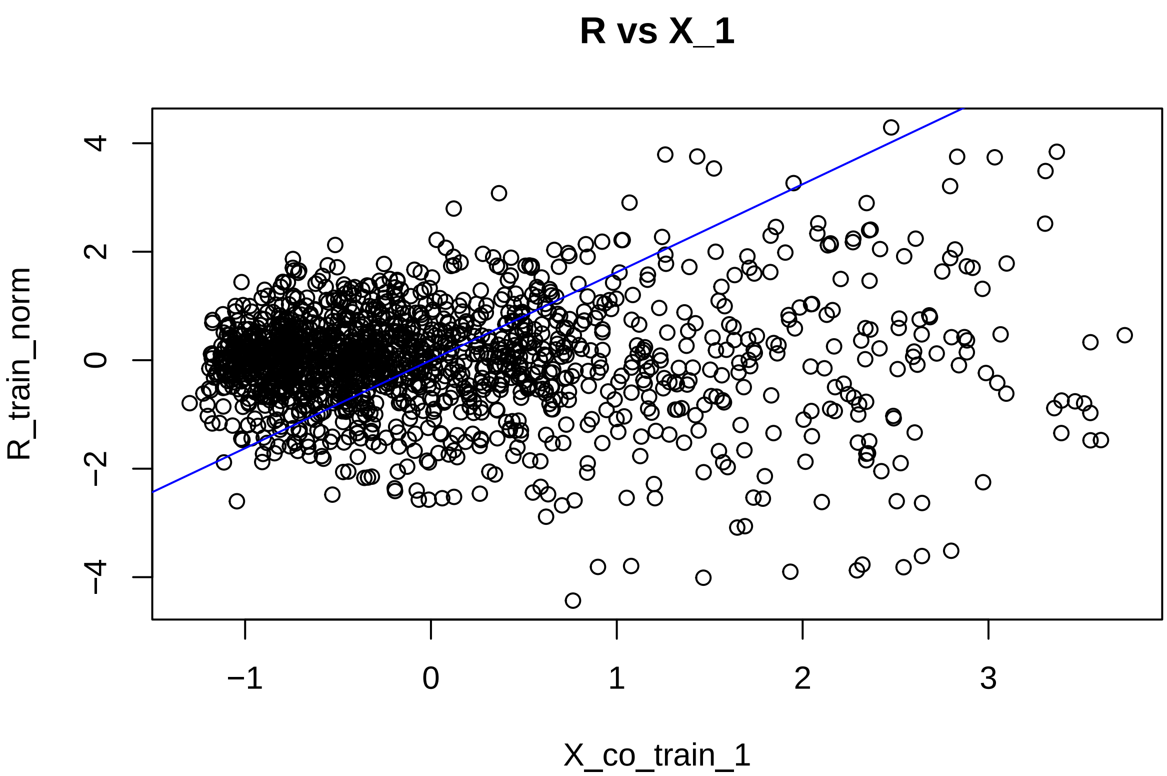
```
## Warning in adf.test(X_co[, 2], k = 10): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  X_co[, 2]
## Dickey-Fuller = -5.7203, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

```r
# Normalize data
N_test <- 1000
start <- 1767
S_train <- S[start:(length(S)-N_test)]
S_test <- S[(length(S)-N_test+1):length(S)]
R_train <- R[start:(length(R)-N_test)]
R_test <- R[(length(R)-N_test+1):length(R)]
X_co_train <- X_co[start:(nrow(X_co)-N_test),]
X_co_test <- X_co[(nrow(X_co)-N_test+1):nrow(X_co),]

S_train_norm <- (S_train-mean(S_train))/sd(S_train)
S_test_norm <- (S_test-mean(S_train))/sd(S_train)
R_train_norm <- (R_train-mean(R_train))/sd(R_train)
R_test_norm <- (R_test-mean(R_train))/sd(R_train)
X_co_train_1 <- (X_co_train[,1]-mean(X_co_train[,1]))/sd(X_co_train[,1])
X_co_test_1 <- (X_co_test[,1]-mean(X_co_train[,1]))/sd(X_co_train[,1])
X_co_train_2 <- (X_co_train[,2]-mean(X_co_train[,2]))/sd(X_co_train[,2])
X_co_test_2 <- (X_co_test[,2]-mean(X_co_train[,2]))/sd(X_co_train[,2])
```

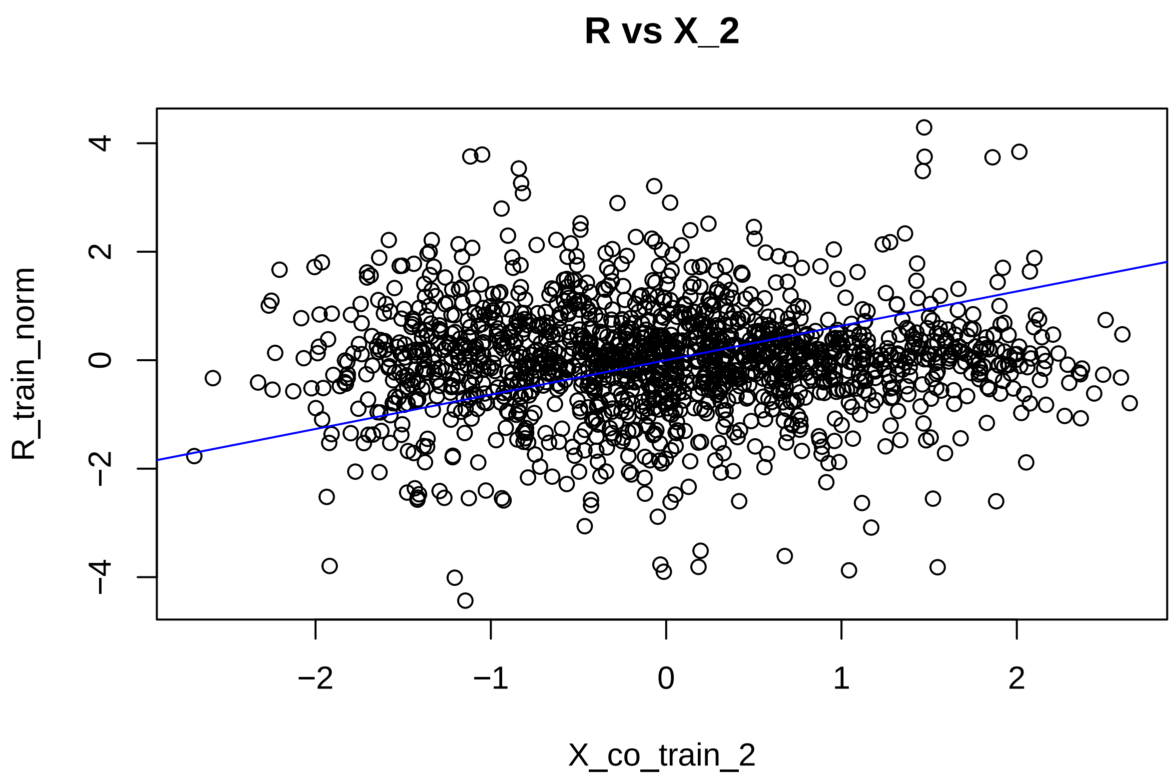
```r
# Plot R vs X_co
plot(X_co_train_1,R_train_norm,main='R vs X_1')
lm_2 <- lm(R_train_norm~poly(X_co_train_1,2))
abline(lm_2,col='blue')
```

```
## Warning in abline(lm_2, col = "blue"): only using the first two of 3 regression
## coefficients
```



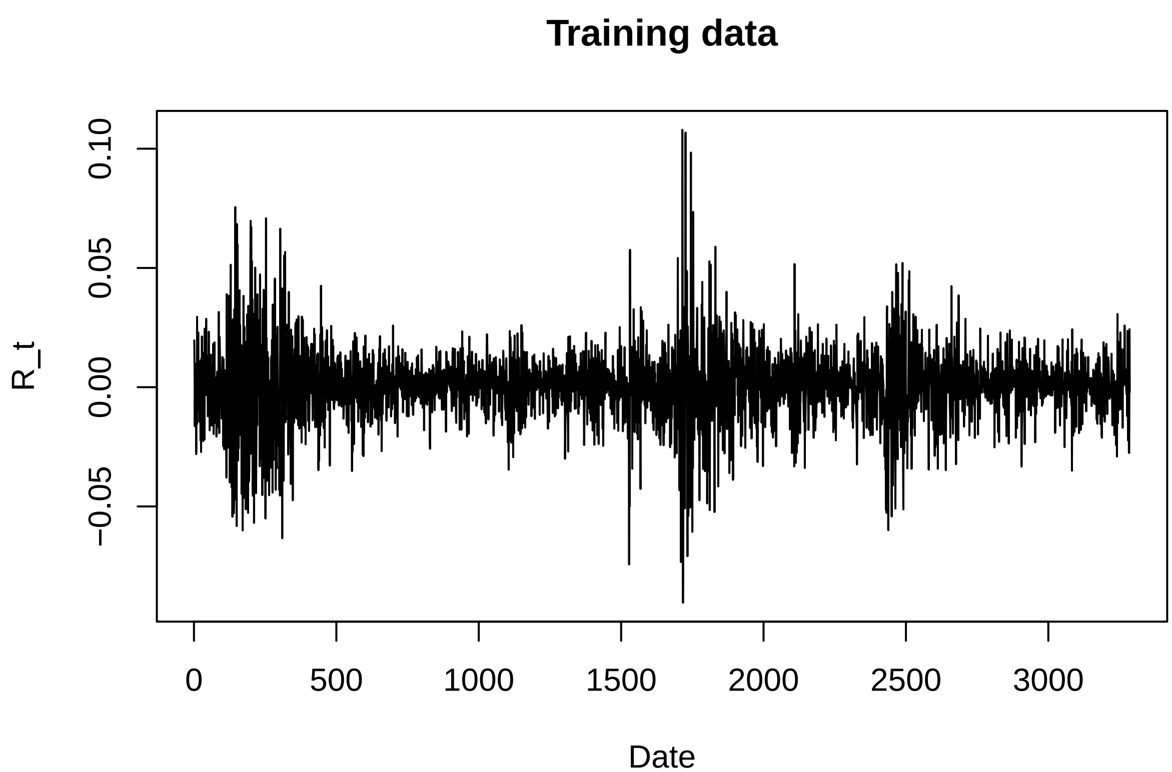

```
plot(X_co_train_2,R_train_norm,main='R vs X_2')
lm_3 <- lm(R_train_norm~poly(X_co_train_2,2))
abline(lm_3,col='blue')
```

```
## Warning in abline(lm_3, col = "blue"): only using the first two of 3 regression
## coefficients
```

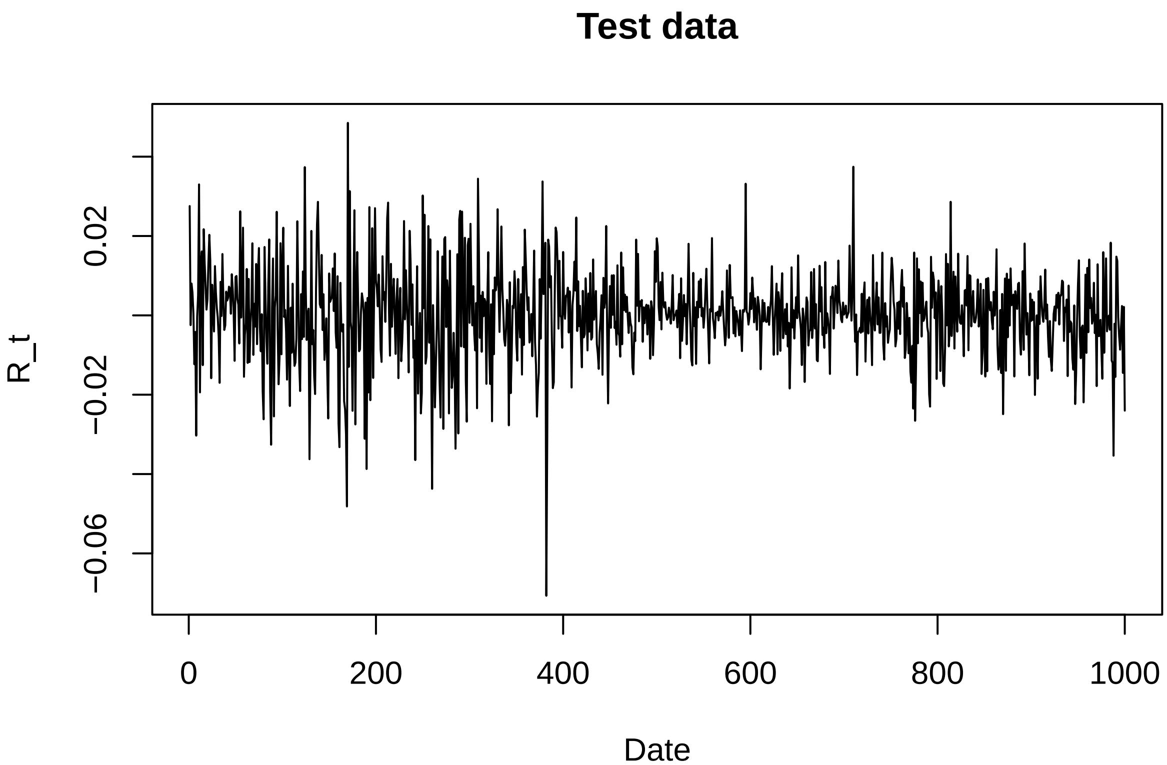


The estimation is better in the test data since it has less abnormal data points.

```
plot(R[1:(length(R)-N_test)],type='l',xlab='Date',ylab='R_t',main='Training data')
```



```r
plot(R[(length(R)-N_test+1):length(R)],type='l',xlab='Date',ylab='R_t',main='Test data')
```



We first try the standard GARCH model on the data. However, it does not eliminate the autocorrelation in the residuals and the residuals are not normal. However, it does not pass the coverage test.

```r
# sGARCH model
X_train <- cbind(X_co_train_1,X_co_train_2)
X_test <- cbind(X_co_test_1,X_co_test_2)
X_all <- rbind(X_train,X_test)
R_all <- c(R_train_norm,R_test_norm)
S_all <- c(S_train_norm,S_test_norm)

spec.GARCH_sim <- ugarchspec(variance.model=list(model="sGARCH",
                garchOrder=c(0,1)), mean.model=list(armaOrder = c(0,0),include.mean=FALSE),
                distribution.model="norm")
GARCH_sim <- ugarchfit(R_all, spec=spec.GARCH_sim,out.sample = N_test)
GARCH_sim
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(0,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
```

```
## ------------------------------------
##        Estimate  Std. Error   t value Pr(>|t|)
## omega  0.000676     2.0e-05    34.352        0
## beta1  0.999000     3.2e-05 30937.342        0
##
## Robust Standard Errors:
##        Estimate  Std. Error   t value Pr(>|t|)
## omega  0.000676     6.4e-05    10.587        0
## beta1  0.999000     4.9e-05 20264.953        0
##
## LogLikelihood : -2144.771
##
## Information Criteria
## ------------------------------------
##
## Akaike       2.8247
## Bayes        2.8317
## Shibata      2.8247
## Hannan-Quinn 2.8273
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      2.405  0.1210
## Lag[2*(p+q)+(p+q)-1][2]     2.834  0.1556
## Lag[4*(p+q)+(p+q)-1][5]     4.633  0.1850
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic   p-value
## Lag[1]                      23.12 1.520e-06
## Lag[2*(p+q)+(p+q)-1][2]     54.71 3.664e-15
## Lag[4*(p+q)+(p+q)-1][5]    157.96 0.000e+00
## d.o.f=1
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale   P-Value
## ARCH Lag[2]     63.01 0.500 2.000 2.109e-15
## ARCH Lag[4]    157.33 1.397 1.611 0.000e+00
## ARCH Lag[6]    212.27 2.222 1.500 0.000e+00
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  9.169
## Individual Statistics:
## omega 2.965
## beta1 3.129
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          0.61 0.749 1.07
## Individual Statistic:     0.35 0.47 0.75
```

```
##
## Sign Bias Test
## ------------------------------------
##                    t-value      prob sig
## Sign Bias           0.09207 9.267e-01
## Negative Sign Bias  5.18005 2.516e-07 ***
## Positive Sign Bias  1.75876 7.882e-02   *
## Joint Effect       36.03642 7.357e-08 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     108.6    1.424e-14
## 2    30     128.0    2.095e-14
## 3    40     135.7    1.268e-12
## 4    50     144.3    2.538e-11
##
##
## Elapsed time : 0.02182198
```

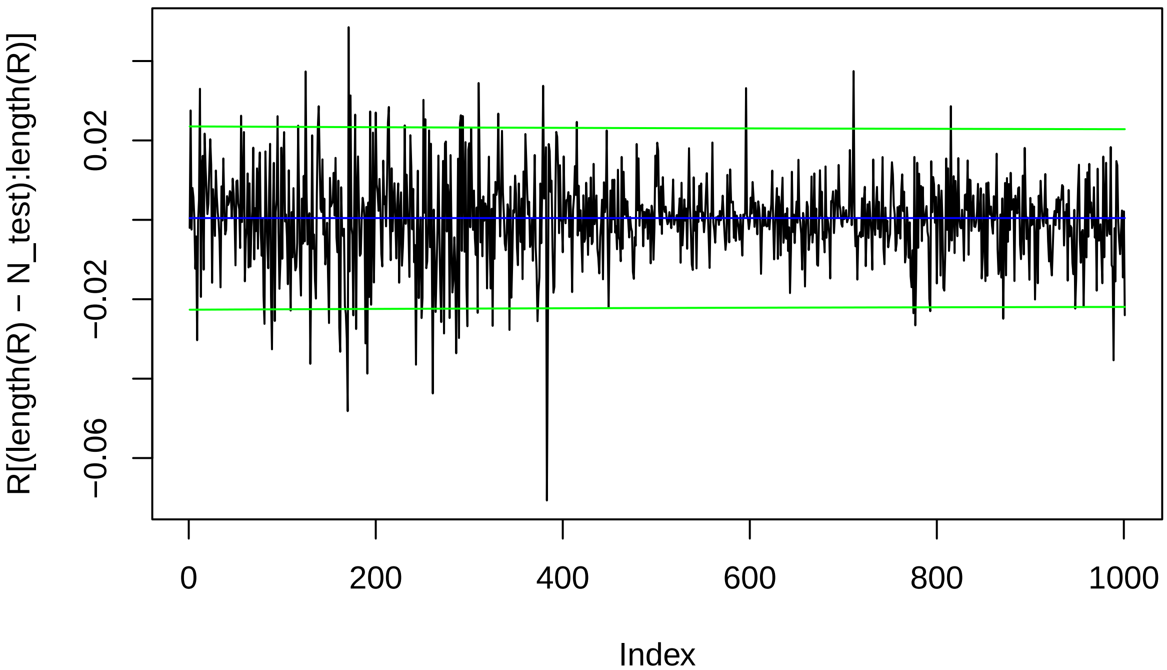
```
# In-sample MSE
sprintf('In-sample MSE is %g',mean((GARCH_sim@fit$fitted.values*sd(R_train)+mean(R_train)-R_train)^2))
```

```
## [1] "In-sample MSE is 0.00018555"
```

```
forecast_sim<-ugarchforecast(GARCH_sim, data = R, n.ahead = 1, n.roll = N_test,out.sample =N_test)
sigma_sim<-sigma(forecast_sim)
fitted_sim<-fitted(forecast_sim)
sprintf('Out-of-sample MSE is %g',mean((forecast_sim@forecast$seriesFor*sd(R_train)+mean(R_train)-R[(len
```

```
## [1] "Out-of-sample MSE is 0.00013675"
```

```
plot(R[(length(R)-N_test):length(R)],type='l')
lines(t(fitted_sim)*sd(R_train)+mean(R_train),col='blue')
lines(t(fitted_sim)*sd(R_train)+mean(R_train)+1.96*t(sigma_sim)*sd(R_train),col='green')
lines(t(fitted_sim)*sd(R_train)+mean(R_train)-1.96*t(sigma_sim)*sd(R_train),col='green')
```




```r
# Coverage test
roll_sim<-ugarchroll(spec=spec.GARCH_sim, data=R, n.ahead=1, forecast.length=N_test, refit.every=253, s
report(roll_sim, type="VaR", VaR.alpha = 0.05, conf.level = 0.95)
```

```
## VaR Backtest Report
## ===========================================
## Model:              sGARCH-norm
## Backtest Length: 1000
## Data:
##
## ==========================================
## alpha:              5%
## Expected Exceed: 50
## Actual VaR Exceed:  37
## Actual %:           3.7%
##
## Unconditional Coverage (Kupiec)
## Null-Hypothesis: Correct Exceedances
## LR.uc Statistic: 3.895
## LR.uc Critical:     3.841
## LR.uc p-value:      0.048
## Reject Null:     YES
##
## Conditional Coverage (Christoffersen)
## Null-Hypothesis: Correct Exceedances and
##                  Independence of Failures
## LR.cc Statistic: 7.774
## LR.cc Critical:     5.991
## LR.cc p-value:      0.021
## Reject Null:     YES
```

```r
# Residual check
Box.test(GARCH_sim@fit$residuals^2, lag = 10, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  GARCH_sim@fit$residuals^2
## X-squared = 507.89, df = 10, p-value < 2.2e-16
```

```r
jarque.bera.test(GARCH_sim@fit$residuals)
```

```
##
##  Jarque Bera Test
##
## data:  GARCH_sim@fit$residuals
## X-squared = 332.42, df = 2, p-value < 2.2e-16
```

Next, we add the latent factors to the model. In-sample MSE is 0.0001855, while out-of-sample MSE is 0.0001367. However, the residuals are not normal and are heteroscedastic. It passes the coverage test.

```r
lags <- 0
N_train <- nrow(X_all)
X_co_1_all <- X_all[(lags+1):N_train,1]
X_co_2_all <- X_all[(lags+1):N_train,2]
for(i in 1:lags){
  if(lags==0){
    break
  }else{
    temp_1 <- X_all[(lags+1-i):(N_train-i),1]
    temp_2 <- X_all[(lags+1-i):(N_train-i),2]

    X_co_1_all <- cbind(X_co_1_all,temp_1)
    X_co_2_all <- cbind(X_co_2_all,temp_2)
  }
}


R_co <- R_all[(lags+1):N_train]
X_train_new <- cbind(as.matrix(X_co_1_all),as.matrix(X_co_2_all))

# gjrGARCH
spec.GARCH_1 <- ugarchspec(variance.model=list(model="sGARCH",
                           garchOrder=c(1,1),external.regressors = as.matrix(X_train_new)),
                           mean.model=list(armaOrder = c(0,0),include.mean=TRUE,
                                           external.regressors = as.matrix(X_train_new)),
                           distribution.model="norm")
sGARCH <- ugarchfit(R_co, spec=spec.GARCH_1,out.sample=N_test)
sGARCH
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
```

```
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.000000    0.024148  0.000000  1.00000
## mxreg1  0.020773    0.035704  0.581818  0.56069
## mxreg2 -0.018831    0.024298 -0.775003  0.43834
## omega   0.004360    0.006380  0.683491  0.49430
## alpha1  0.069387    0.010984  6.317243  0.00000
## beta1   0.928774    0.011473 80.951615  0.00000
## vxreg1  0.000000    0.005907  0.000002  1.00000
## vxreg2  0.000000    0.002227  0.000004  1.00000
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.000000    0.026199  0.000000 1.000000
## mxreg1  0.020773    0.043052  0.482520 0.629436
## mxreg2 -0.018831    0.027388 -0.687557 0.491732
## omega   0.004360    0.006597  0.660997 0.508614
## alpha1  0.069387    0.017893  3.877906 0.000105
## beta1   0.928774    0.017238 53.880802 0.000000
## vxreg1  0.000000    0.006564  0.000002 0.999999
## vxreg2  0.000000    0.002566  0.000004 0.999997
##
## LogLikelihood : -1986.898
##
## Information Criteria
## ------------------------------------
##
## Akaike       2.6249
## Bayes        2.6529
## Shibata      2.6248
## Hannan-Quinn 2.6353
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.9281  0.3354
## Lag[2*(p+q)+(p+q)-1][2]    1.0727  0.4753
## Lag[4*(p+q)+(p+q)-1][5]    1.2845  0.7926
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.4895  0.4841
## Lag[2*(p+q)+(p+q)-1][5]    4.4542  0.2026
## Lag[4*(p+q)+(p+q)-1][9]    5.4664  0.3642
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
```

```
## ARCH Lag[3]     3.142 0.500 2.000 0.07629
## ARCH Lag[5]     3.257 1.440 1.667 0.25468
## ARCH Lag[7]     3.316 2.315 1.543 0.45588
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  5.3279
## Individual Statistics:
## mu     0.49397
## mxreg1 0.13942
## mxreg2 0.03438
## omega  0.22624
## alpha1 0.03384
## beta1  0.02004
## vxreg1 0.14197
## vxreg2 0.08867
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.89 2.11 2.59
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value      prob sig
## Sign Bias           1.3709 0.1706092
## Negative Sign Bias  0.1409 0.8879729
## Positive Sign Bias  2.6337 0.0085315 ***
## Joint Effect       20.9642 0.0001071 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     65.61    4.867e-07
## 2    30     75.84    4.623e-06
## 3    40     90.84    5.101e-06
## 4    50    107.89    2.573e-06
##
##
## Elapsed time : 0.09198093
```

```
# In-sample
sprintf('In-sample MSE is %g',mean((sGARCH@fit$residuals*sd(R_train))^2))
```

```
## [1] "In-sample MSE is 0.000185518"
```

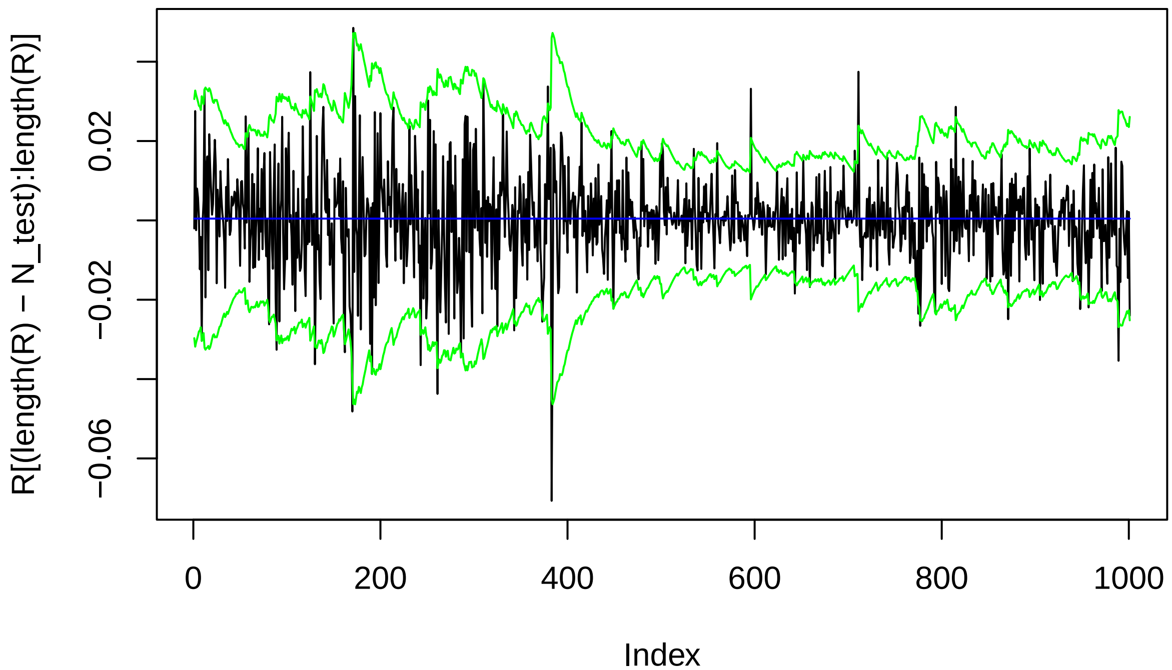
```
# Out-of-sample
forecast_GARCH1<-ugarchforecast(sGARCH, data = R_co, n.ahead = 1, n.roll = N_test,out.sample =N_test)
sigma_GARCH1<-sigma(forecast_GARCH1)
fitted_GARCH1<-fitted(forecast_GARCH1)

sprintf('Out-of-sample MSE is %g',mean((t(fitted_GARCH1)*sd(R_train)+mean(R_train)-R[(length(R)-N_test)
```

```
## [1] "Out-of-sample MSE is 0.00013675"
```

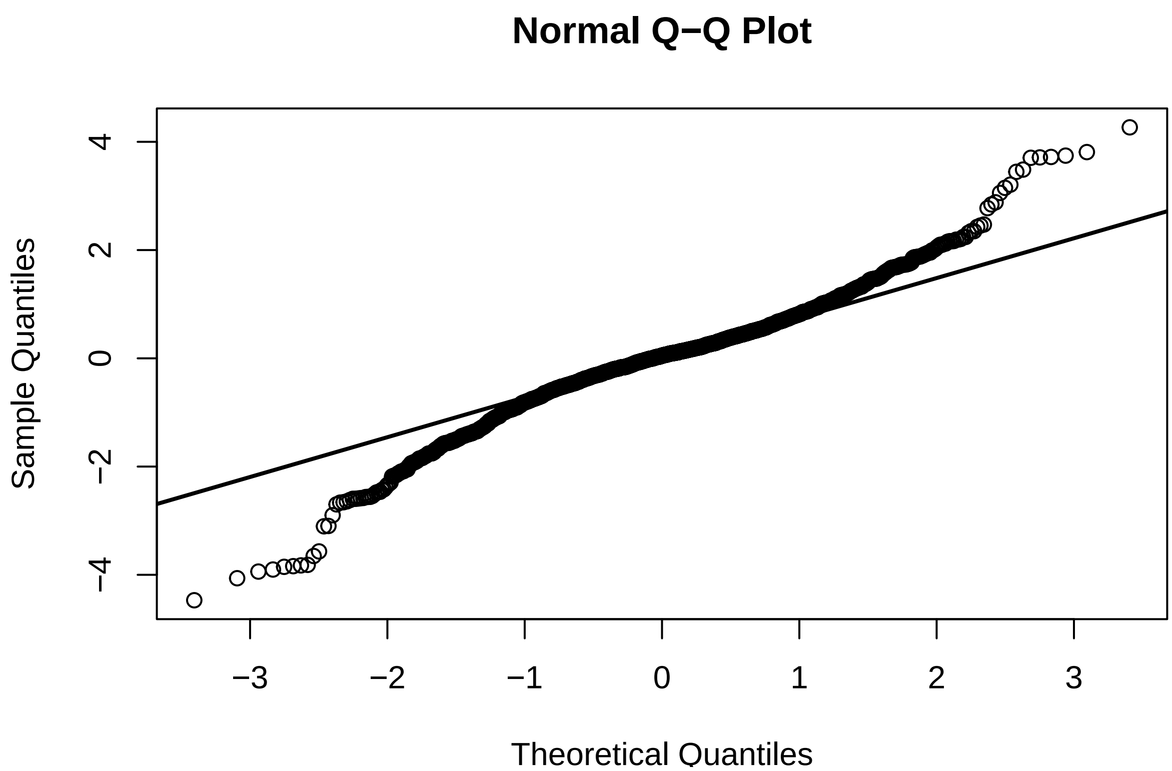
```
sprintf('Out-of-sample mean of sd is %g',mean(sigma_GARCH1))
```

```
## [1] "Out-of-sample mean of sd is 0.841102"
```

```
plot(R[(length(R)-N_test):length(R)],type='l')
lines(t(fitted_GARCH1)*sd(R_train)+mean(R_train),col='blue')
lines(t(fitted_GARCH1)*sd(R_train)+mean(R_train)+1.96*t(sigma_GARCH1)*sd(R_train),col='green')
lines(t(fitted_GARCH1)*sd(R_train)+mean(R_train)-1.96*t(sigma_GARCH1)*sd(R_train),col='green')
```




```
# Residual check
qqnorm(sGARCH@fit$residuals)
qqline(sGARCH@fit$residuals,lwd = 2)
```




```
jarque.bera.test(sGARCH@fit$residuals)
```

```
##
##  Jarque Bera Test
##
## data:  sGARCH@fit$residuals
## X-squared = 340.83, df = 2, p-value < 2.2e-16
```

```
Box.test(sGARCH@fit$residuals, lag = 10, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  sGARCH@fit$residuals
## X-squared = 8.1012, df = 10, p-value = 0.619
```

```
Box.test(sGARCH@fit$residuals^2, lag = 10, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  sGARCH@fit$residuals^2
## X-squared = 515.28, df = 10, p-value < 2.2e-16
```

```
# Coverage test
roll_GARCH<-ugarchroll(spec=spec.GARCH_1, data=R_co, n.ahead=1, forecast.length=N_test, refit.every=253
report(roll_GARCH, type="VaR", VaR.alpha = 0.05, conf.level = 0.95)
```

```
## VaR Backtest Report
## ===========================================
## Model:              sGARCH-norm
## Backtest Length: 1000
```

```
## Data:
##
## ==========================================
## alpha:              5%
## Expected Exceed: 50
## Actual VaR Exceed:   55
## Actual %:           5.5%
##
## Unconditional Coverage (Kupiec)
## Null-Hypothesis: Correct Exceedances
## LR.uc Statistic: 0.51
## LR.uc Critical:      3.841
## LR.uc p-value:       0.475
## Reject Null:     NO
##
## Conditional Coverage (Christoffersen)
## Null-Hypothesis: Correct Exceedances and
##                  Independence of Failures
## LR.cc Statistic: 5.11
## LR.cc Critical:      5.991
## LR.cc p-value:       0.078
## Reject Null:     NO
```

Finally, we add lags into the model. In-sample MSE is 0.0001842, while the out-of-sample MSE is 0.0001291. It passes the coverage test.

```
# Create external regressor with lags
lags <- 1
N_train <- nrow(X_all)
X_co_1_all <- X_all[(lags+1):N_train,1]
X_co_2_all <- X_all[(lags+1):N_train,2]
for(i in 1:lags){
  if(lags==0){
    break
  }else{
    temp_1 <- X_all[(lags+1-i):(N_train-i),1]
    temp_2 <- X_all[(lags+1-i):(N_train-i),2]

    X_co_1_all <- cbind(X_co_1_all,temp_1)
    X_co_2_all <- cbind(X_co_2_all,temp_2)
  }
}


R_co <- R_all[(lags+1):N_train]
X_train_new <- cbind(as.matrix(X_co_1_all),as.matrix(X_co_2_all))

# eGARCH
spec.gjrGARCH <- ugarchspec(variance.model=list(model="eGARCH",
                            garchOrder=c(2,2),external.regressors = as.matrix(X_train_new)),
                            mean.model=list(armaOrder = c(1,1),include.mean=FALSE,
                                            external.regressors = as.matrix(X_train_new)),
                            distribution.model="std")
gjrGARCH <- ugarchfit(R_co, spec=spec.gjrGARCH,out.sample=N_test)
gjrGARCH
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : eGARCH(2,2)
## Mean Model   : ARFIMA(1,0,1)
## Distribution : std
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## ar1    -0.514040    0.164460  -3.12562 0.001774
## ma1     0.542826    0.161366   3.36394 0.000768
## mxreg1 -0.135394    0.049477  -2.73652 0.006209
## mxreg2  0.163873    0.051832   3.16162 0.001569
## mxreg3 -0.115957    0.024389  -4.75451 0.000002
## mxreg4  0.131591    0.028652   4.59274 0.000004
## omega  -0.027069    0.008398  -3.22340 0.001267
## alpha1 -0.249180    0.056244  -4.43033 0.000009
## alpha2  0.087585    0.047066   1.86088 0.062761
## beta1   1.000000    0.021031  47.54864 0.000000
## beta2  -0.083007    0.008241 -10.07273 0.000000
## gamma1 -0.163259    0.061338  -2.66162 0.007776
## gamma2  0.238526    0.061855   3.85622 0.000115
## vxreg1  0.150096    0.201743   0.74399 0.456880
## vxreg2 -0.103082    0.204527  -0.50400 0.614262
## vxreg3 -0.230672    0.094441  -2.44250 0.014586
## vxreg4  0.228621    0.094794   2.41177 0.015875
## shape   9.110548    2.080151   4.37975 0.000012
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## ar1    -0.514040    0.051963  -9.89236 0.000000
## ma1     0.542826    0.051383  10.56429 0.000000
## mxreg1 -0.135394    0.014484  -9.34795 0.000000
## mxreg2  0.163873    0.022136   7.40286 0.000000
## mxreg3 -0.115957    0.013244  -8.75560 0.000000
## mxreg4  0.131591    0.020699   6.35724 0.000000
## omega  -0.027069    0.008246  -3.28288 0.001028
## alpha1 -0.249180    0.064182  -3.88240 0.000103
## alpha2  0.087585    0.050348   1.73959 0.081931
## beta1   1.000000    0.005100 196.08593 0.000000
## beta2  -0.083007    0.019377  -4.28383 0.000018
## gamma1 -0.163259    0.053915  -3.02807 0.002461
## gamma2  0.238526    0.050236   4.74812 0.000002
## vxreg1  0.150096    0.208819   0.71878 0.472274
## vxreg2 -0.103082    0.210827  -0.48894 0.624885
## vxreg3 -0.230672    0.102147  -2.25823 0.023931
## vxreg4  0.228621    0.102691   2.22630 0.025994
## shape   9.110548    1.958090   4.65277 0.000003
##
```

```
## LogLikelihood : -1895.433
##
## Information Criteria
## ------------------------------------
##
## Akaike       2.5193
## Bayes        2.5824
## Shibata      2.5191
## Hannan-Quinn 2.5428
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                   0.004186  0.9484
## Lag[2*(p+q)+(p+q)-1][5]  0.410322  1.0000
## Lag[4*(p+q)+(p+q)-1][9]  1.045950  0.9997
## d.o.f=2
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                     0.03335  0.8551
## Lag[2*(p+q)+(p+q)-1][11]   1.71419  0.9761
## Lag[4*(p+q)+(p+q)-1][19]   4.13957  0.9743
## d.o.f=4
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[5]    0.3231 0.500 2.000  0.5698
## ARCH Lag[7]    0.9606 1.473 1.746  0.7684
## ARCH Lag[9]    1.1870 2.402 1.619  0.9041
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  3.3173
## Individual Statistics:
## ar1    0.08881
## ma1    0.09134
## mxreg1 0.05700
## mxreg2 0.07403
## mxreg3 0.08005
## mxreg4 0.08286
## omega  0.15527
## alpha1 0.09233
## alpha2 0.05726
## beta1  0.06188
## beta2  0.06107
## gamma1 0.10385
## gamma2 0.03942
## vxreg1 0.08801
## vxreg2 0.08054
## vxreg3 0.07294
```

```
## vxreg4 0.08302
## shape  0.37019
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          3.83 4.14 4.73
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.9353 0.3498
## Negative Sign Bias  0.2923 0.7701
## Positive Sign Bias  0.8217 0.4113
## Joint Effect        1.6999 0.6370
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     27.29       0.0980
## 2    30     30.95       0.3679
## 3    40     39.53       0.4464
## 4    50     52.79       0.3298
##
##
## Elapsed time : 1.080042
```

```
# In-sample mse
sprintf('In-sample MSE is %g',mean((gjrGARCH@fit$residuals*sd(R_train))^2))
```

```
## [1] "In-sample MSE is 0.00018422"
```

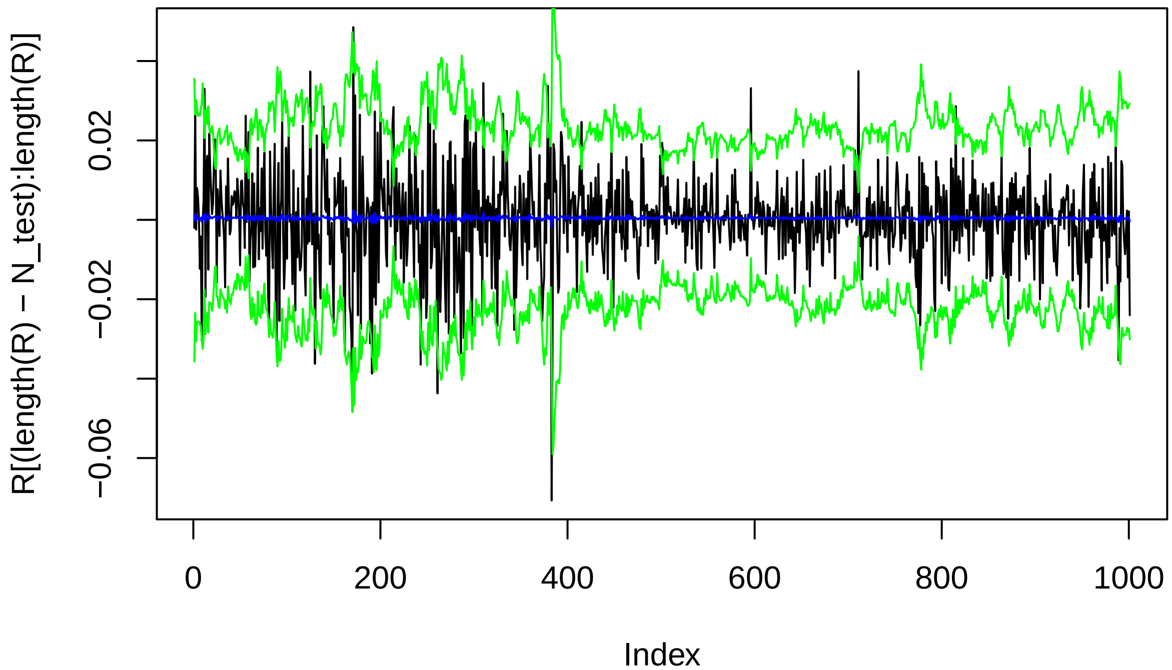
```
# Out-of-sample
forecast_gjrGARCH<-ugarchforecast(gjrGARCH, data = R_co, n.ahead = 1, n.roll = N_test,out.sample =N_tes
sigma_gjrGARCH<-sigma(forecast_gjrGARCH)
fitted_gjrGARCH<-fitted(forecast_gjrGARCH)
sprintf('Out-of-sample MSE is %g',mean(((t(fitted_gjrGARCH)-R_co[(length(R_co)-N_test):length(R_co)])*s
```

```
## [1] "Out-of-sample MSE is 0.000129122"
```

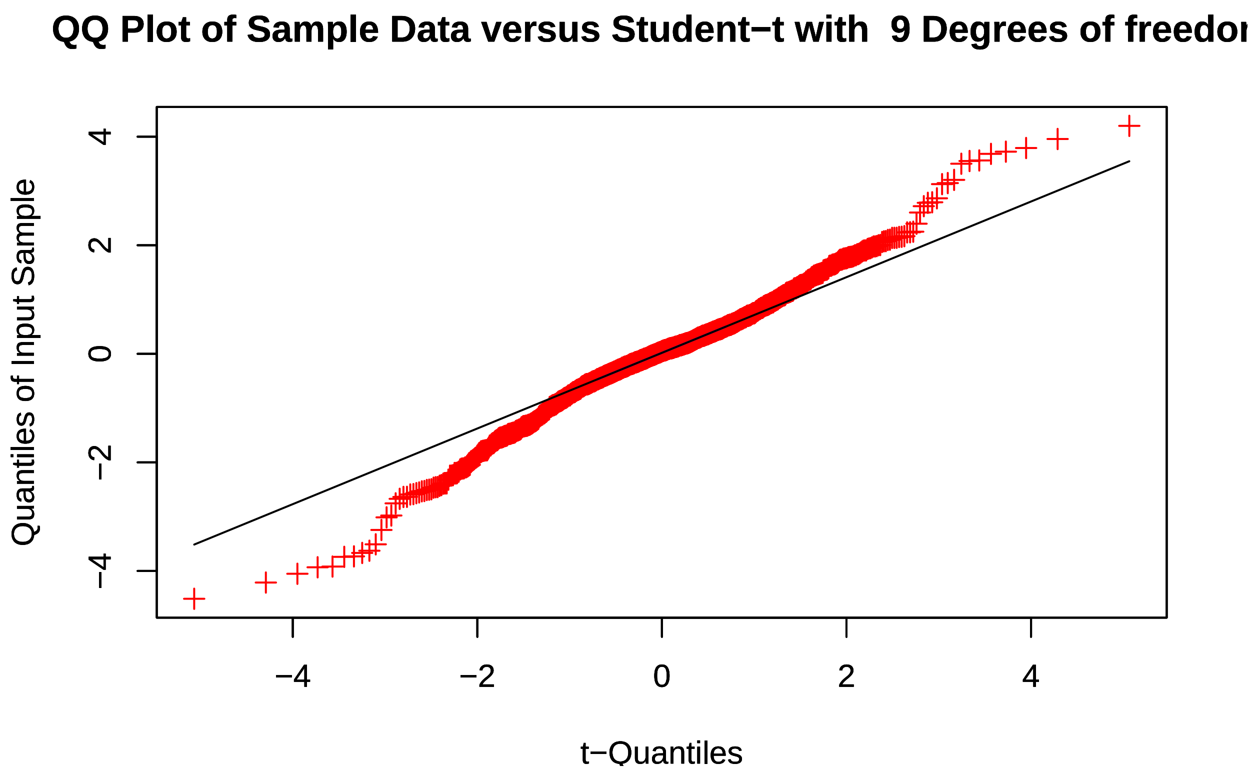
```
sprintf('Out-of-sample mean of sd is %g',mean(sigma_gjrGARCH))
```

```
## [1] "Out-of-sample mean of sd is 0.880826"
```

```
plot(R[(length(R)-N_test):length(R)],type='l')
lines(t(fitted_gjrGARCH)*sd(R_train)+mean(R_train),col='blue')
lines(t(fitted_gjrGARCH)*sd(R_train)+mean(R_train)+1.96*t(sigma_gjrGARCH)*sd(R_train),col='green')
lines(t(fitted_gjrGARCH)*sd(R_train)+mean(R_train)-1.96*t(sigma_gjrGARCH)*sd(R_train),col='green')
```




```r
# Residual check
TQQPlot(gjrGARCH@fit$residuals, 9)
```




```r
Box.test(gjrGARCH@fit$residuals, lag = 10, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  gjrGARCH@fit$residuals
```

```
## X-squared = 6.2979, df = 10, p-value = 0.7896
```

```
Box.test(gjrGARCH@fit$residuals^2, lag = 10, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  gjrGARCH@fit$residuals^2
## X-squared = 470.74, df = 10, p-value < 2.2e-16
```

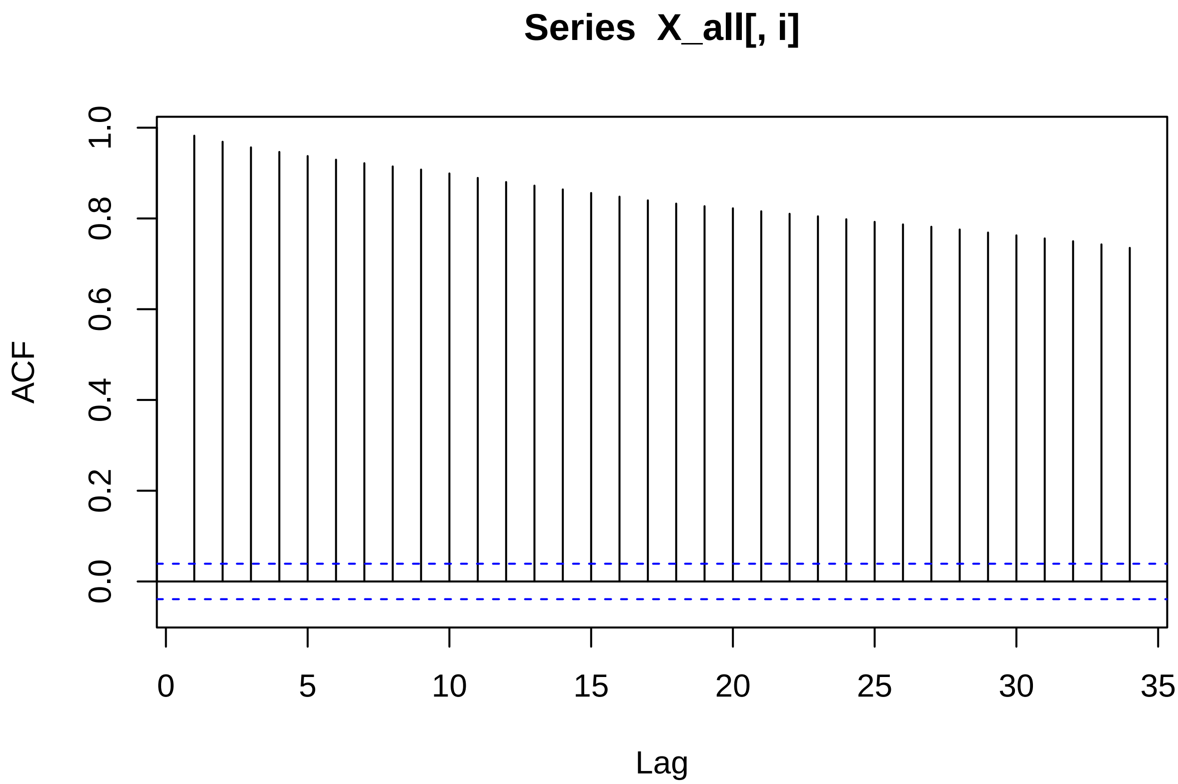
```
# Coverage test
roll_GARCH<-ugarchroll(spec=spec.gjrGARCH, data=R_co, n.ahead=1, forecast.length=N_test, refit.every=25
report(roll_GARCH, type="VaR", VaR.alpha = 0.05, conf.level = 0.95)
```

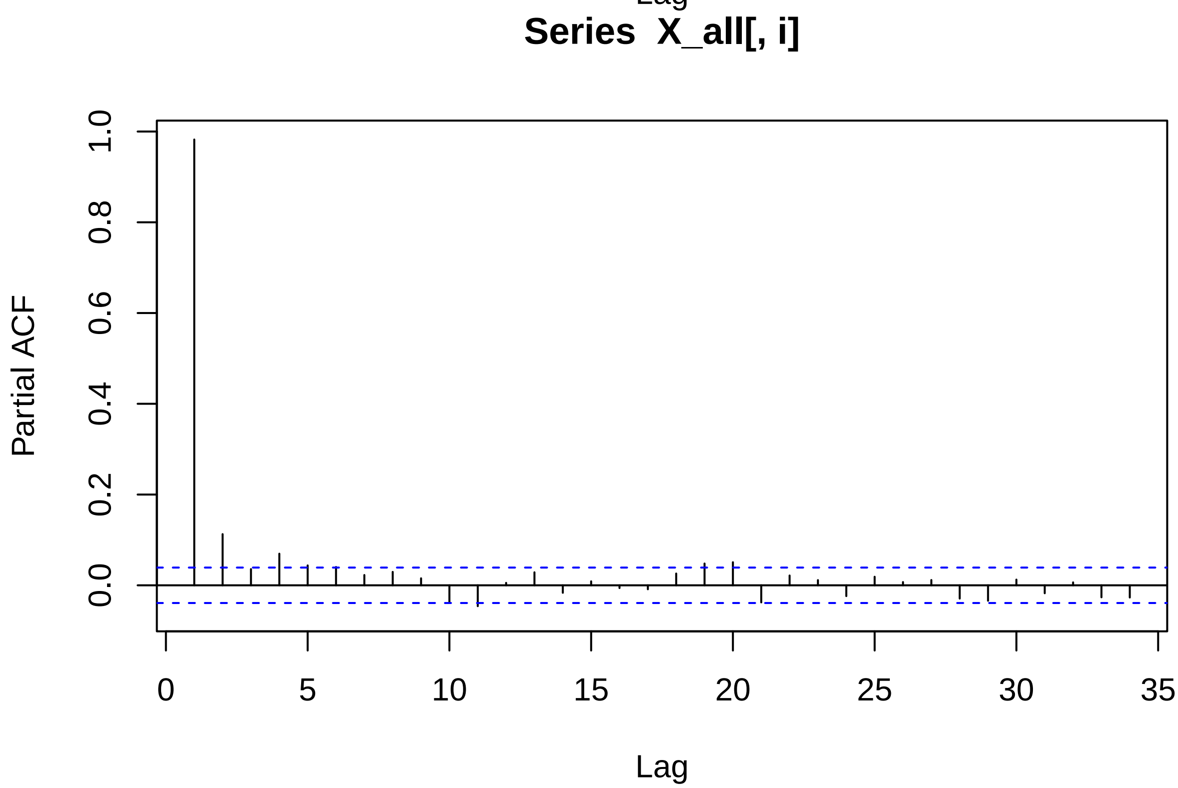
```
## VaR Backtest Report
## ===========================================
## Model:               eGARCH-std
## Backtest Length: 1000
## Data:
##
## ==========================================
## alpha:               5%
## Expected Exceed: 50
## Actual VaR Exceed:   57
## Actual %:            5.7%
##
## Unconditional Coverage (Kupiec)
## Null-Hypothesis: Correct Exceedances
## LR.uc Statistic: 0.989
## LR.uc Critical:      3.841
## LR.uc p-value:       0.32
## Reject Null:     NO
##
## Conditional Coverage (Christoffersen)
## Null-Hypothesis: Correct Exceedances and
##                  Independence of Failures
## LR.cc Statistic: 1.202
## LR.cc Critical:      5.991
## LR.cc p-value:       0.548
## Reject Null:     NO
```

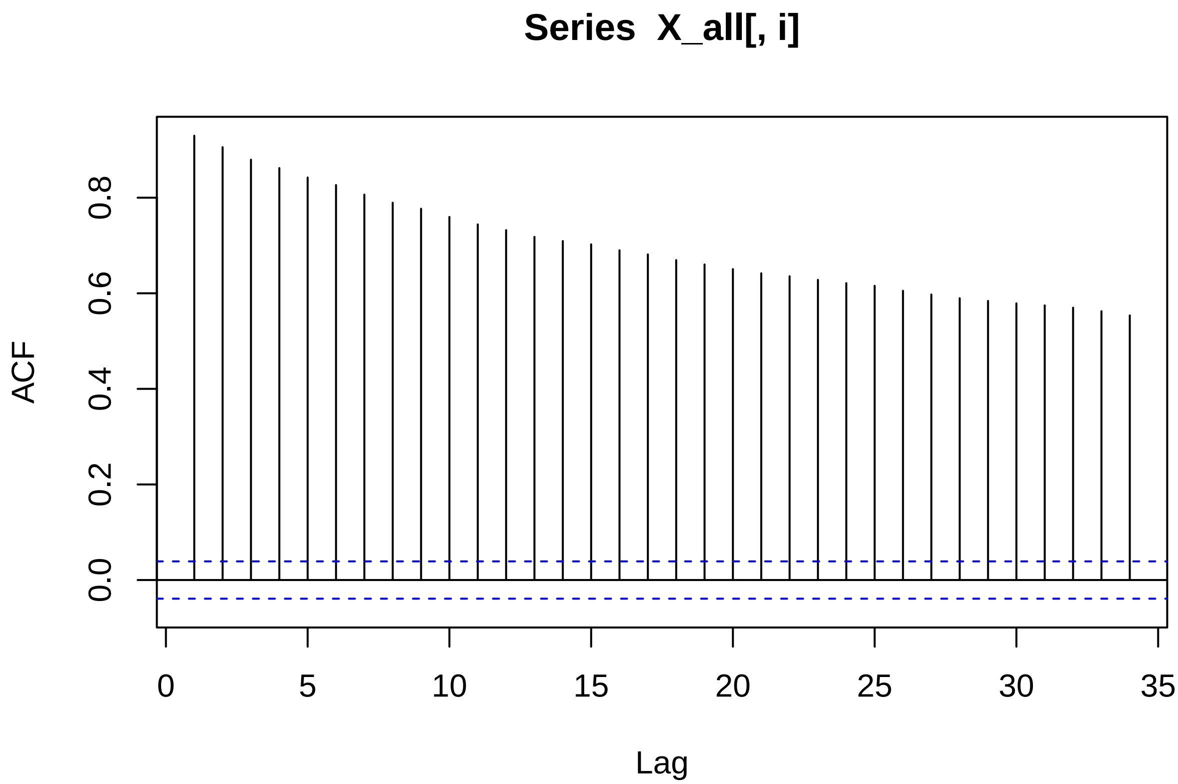
DCC VAR for factors

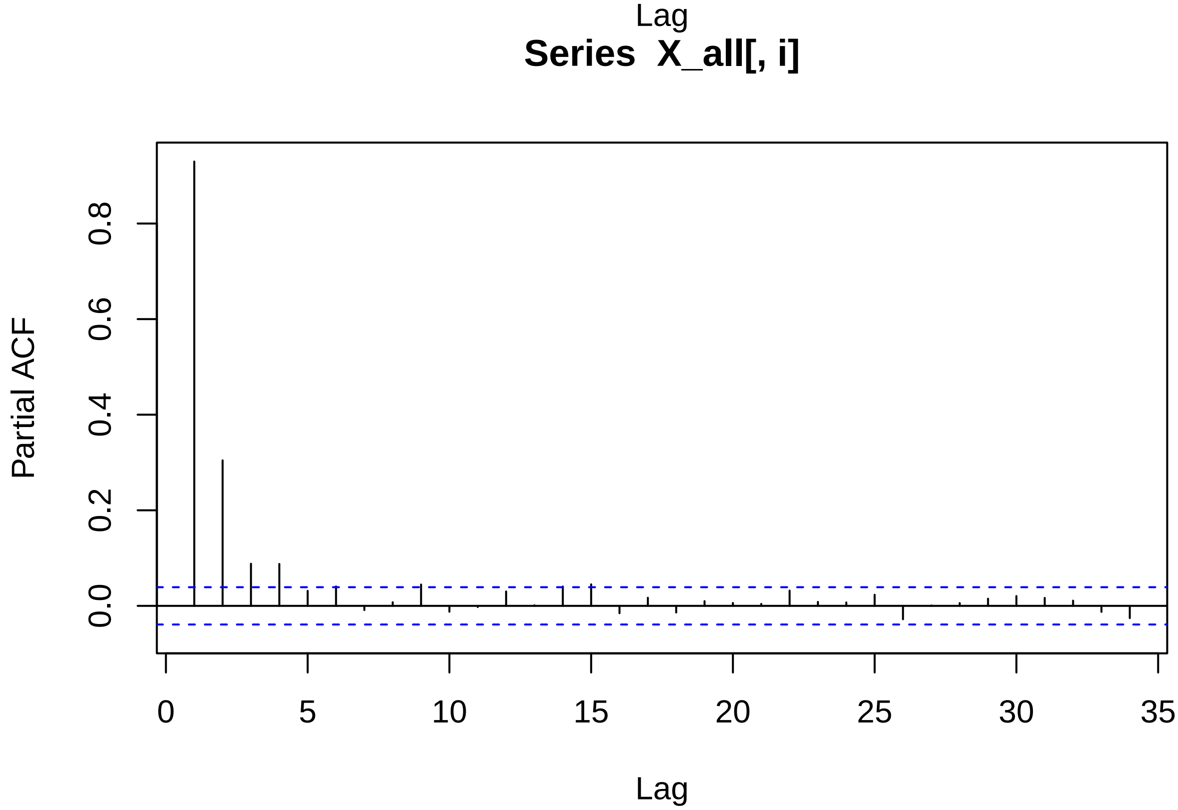
The optimal number of VAR components is 4 by VARselect.

```
library(rmgarch)
library(MTS)

# ACF and PACF for factors
for(i in 1:2){
  Acf(X_all[,i])
  Pacf(X_all[,i])
}
```


Series X_all[, i]
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
35
Lag


Series X_all[, i]
Partial ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
35
Lag






```
VARselect(X_all,lag.max=10,type="none")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      6      4      4      6
```

```
##
## $criteria
##                  1            2           3           4            5
## AIC(n) -5.655029143 -5.758441800 -5.76874676 -5.77950344 -5.782719933
## HQ(n)  -5.651657950 -5.751699413 -5.75863318 -5.76601867 -5.765863967
## SC(n)  -5.645741433 -5.739866380 -5.74088363 -5.74235260 -5.736281383
## FPE(n)  0.003499871  0.003156026  0.00312367  0.00309025  0.003080326
##                  6            7            8            9         10
## AIC(n) -5.784986137 -5.784361281 -5.784589564 -5.784003624 -5.7811070
## HQ(n)  -5.764758977 -5.760762928 -5.757620018 -5.753662884 -5.7473951
## SC(n)  -5.729259877 -5.719347311 -5.710287884 -5.700414234 -5.6882299
## FPE(n)  0.003073353  0.003075275  0.003074573  0.003076376  0.0030853
```

However, if we use VAR(4), its residuals does not pass the ARCH-LM test.

```
VAR_co <- vars::VAR(X_co,p=4,type="none")
summary(VAR_co)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: X_1, X_2
## Deterministic variables: none
## Sample size: 4282
## Log Likelihood: 37313.121
## Roots of the characteristic polynomial:
## 0.993 0.9765 0.528 0.528 0.4808 0.4673 0.4673 0.4387
## Call:
## vars::VAR(y = X_co, p = 4, type = "none")
##
##
## Estimation results for equation X_1:
## ====================================
## X_1 = X_1.l1 + X_2.l1 + X_1.l2 + X_2.l2 + X_1.l3 + X_2.l3 + X_1.l4 + X_2.l4
##
##         Estimate Std. Error t value Pr(>|t|)
## X_1.l1  0.822075   0.015344  53.575  < 2e-16 ***
## X_2.l1  0.115793   0.010790  10.732  < 2e-16 ***
## X_1.l2  0.030226   0.020157   1.499  0.13382
## X_2.l2 -0.076837   0.012529  -6.133  9.4e-10 ***
## X_1.l3  0.028362   0.020153   1.407  0.15941
## X_2.l3 -0.036106   0.012529  -2.882  0.00397 **
## X_1.l4  0.109520   0.015342   7.139  1.1e-12 ***
## X_2.l4  0.006309   0.010794   0.585  0.55891
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.002617 on 4274 degrees of freedom
## Multiple R-Squared: 0.9738,  Adjusted R-squared: 0.9738
## F-statistic: 1.988e+04 on 8 and 4274 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation X_2:
## ====================================
```

```
## X_2 = X_1.l1 + X_2.l1 + X_1.l2 + X_2.l2 + X_1.l3 + X_2.l3 + X_1.l4 + X_2.l4
##
##        Estimate Std. Error t value Pr(>|t|)
## X_1.l1  0.17630    0.02180   8.085 8.00e-16 ***
## X_2.l1  0.55052    0.01533  35.905  < 2e-16 ***
## X_1.l2 -0.11475    0.02864  -4.006 6.28e-05 ***
## X_2.l2  0.20524    0.01780  11.528  < 2e-16 ***
## X_1.l3 -0.10316    0.02864  -3.602 0.000319 ***
## X_2.l3  0.08595    0.01780   4.827 1.43e-06 ***
## X_1.l4  0.04261    0.02180   1.955 0.050692 .
## X_2.l4  0.11614    0.01534   7.572 4.47e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.003719 on 4274 degrees of freedom
## Multiple R-Squared: 0.8646,  Adjusted R-squared: 0.8643
## F-statistic:  3411 on 8 and 4274 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##            X_1        X_2
## X_1  6.847e-06 -1.351e-06
## X_2 -1.351e-06  1.383e-05
##
## Correlation matrix of residuals:
##         X_1     X_2
## X_1  1.0000 -0.1388
## X_2 -0.1388  1.0000
```

```
# heteroscedasticity test
arch.test(VAR_co)
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object VAR_co
## Chi-squared = 2393.6, df = 45, p-value < 2.2e-16
```

The best model is DCC(2,2)+VAR(4). The residuals pass the Ljung-Box and ARCH-LM test.

```
xspec_co_1 <- ugarchspec(mean.model = list(armaOrder = c(4, 0),include.mean=TRUE), variance.model = lis
xspec_co_2 <- ugarchspec(mean.model = list(armaOrder = c(4, 0),include.mean=TRUE), variance.model = lis
uspec_co <- multispec(c(xspec_co_1, xspec_co_2))
spec1_co <- dccspec(uspec = uspec_co, VAR=TRUE, robust=TRUE,lag=4, dccOrder = c(2,2), model="DCC", dist
fit_co <- dccfit(spec1_co, data = X_all, fit.control = list(eval.se = TRUE), out.sample=N_test)
fit_co
```

```
##
## *---------------------------------*
## *          DCC GARCH Fit          *
## *---------------------------------*
##
## Distribution         :  mvt
## Model                :  DCC(2,2)
```

```
## No. Parameters       :  34
## [VAR GARCH DCC UncQ] : [18+10+5+1]
## No. Series           :  2
## No. Obs.             :  1520
## Log-Likelihood       :  971.3774
## Av.Log-Likelihood    :  0.64
##
## Optimal Parameters
## -----------------------------------
##                        Estimate  Std. Error    t value Pr(>|t|)
## [X_co_train_1].omega  -0.232044    0.006970 -33.292389 0.000000
## [X_co_train_1].alpha1  0.213017    0.004983  42.751263 0.000000
## [X_co_train_1].beta1   0.947304    0.001032 918.116860 0.000000
## [X_co_train_1].gamma1  0.118610    0.038922   3.047328 0.002309
## [X_co_train_1].shape   4.629890    0.475079   9.745508 0.000000
## [X_co_train_2].omega  -0.089989    0.075270  -1.195539 0.231876
## [X_co_train_2].alpha1  0.008973    0.020740   0.432639 0.665277
## [X_co_train_2].beta1   0.958362    0.034976  27.400332 0.000000
## [X_co_train_2].gamma1  0.180159    0.074735   2.410637 0.015925
## [X_co_train_2].shape   8.720130    1.649227   5.287406 0.000000
## [Joint]dcca1           0.066903    0.019682   3.399236 0.000676
## [Joint]dcca2           0.112614    0.022083   5.099652 0.000000
## [Joint]dccb1           0.000003    0.049235   0.000052 0.999958
## [Joint]dccb2           0.797272    0.052520  15.180460 0.000000
## [Joint]mshape          5.178174    0.340078  15.226439 0.000000
##
## Information Criteria
## ---------------------
##
## Akaike       -1.2334
## Bayes        -1.1142
## Shibata      -1.2344
## Hannan-Quinn -1.1890
##
##
## Elapsed time : 3.210966
```

```r
fit_co@model$varcoef
```

```
##              X_co_train_1.l1 X_co_train_2.l1 X_co_train_1.l2 X_co_train_2.l2
## X_co_train_1      0.90966827      0.04795011     0.001218938     -0.02546627
## X_co_train_2      0.02187174      0.61232853    -0.074523982      0.20971246
##              X_co_train_1.l3 X_co_train_2.l3 X_co_train_1.l4 X_co_train_2.l4
## X_co_train_1      0.01304149    -0.007957851      0.03858480     -0.01262996
## X_co_train_2     -0.01115742     0.067426116      0.02659432      0.06149068
##                     const
## X_co_train_1 -0.025356105
## X_co_train_2 -0.006004219
```

```r
# In-sample
print('In-sample MSEs are')
```

```
## [1] "In-sample MSEs are"
```

```r
apply(fit_co@model$residuals^2,2,mean)
```

```
## [1] 0.04285797 0.13020144
```

```r
# out-of-sample mse and mean of sd
forcast_dcc_co <- dccforecast(fit_co,n.ahead=1,n.roll=N_test)
fitted_co <- t(fitted(forcast_dcc_co)[1,,])
sigma_co <- t(sigma(forcast_dcc_co)[1,,])
mse_co_temp <- (fitted_co-X_co[(nrow(X_co)-N_test):nrow(X_co),])^2
print('MSEs are')
```

```
## [1] "MSEs are"
```

```r
apply(mse_co_temp,2,mean)
```

```
##       X_1       X_2
## 0.5377432 1.2387076
```

```r
print('Means of sd are')
```

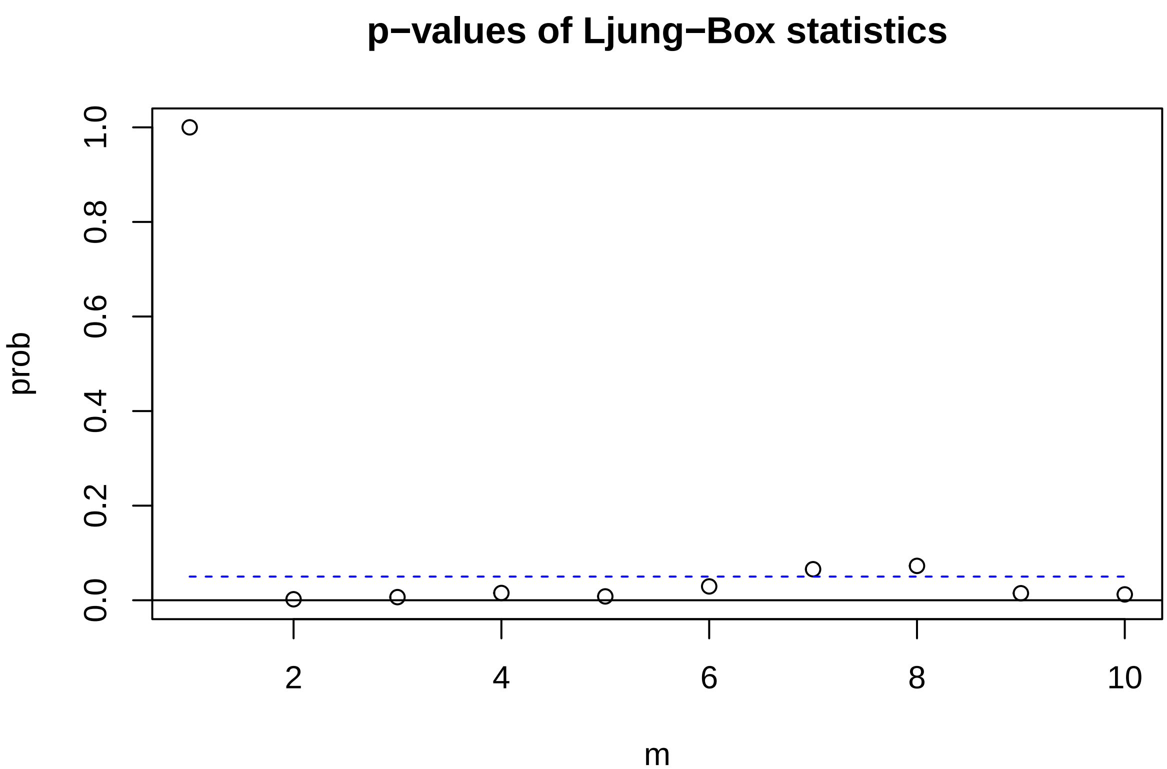
```
## [1] "Means of sd are"
```

```r
apply(sigma_co,2,mean)
```

```
## X_co_train_1 X_co_train_2
##    0.1225550    0.3688504
```

```r
# Ljung-Box test
res_co <- fit_co@mfit$stdresid
mq(res_co,lag=10,adj=4)
```

```
## Ljung-Box Statistics:
##          m       Q(m)     df    p-value
##  [1,]  1.00      6.08     0.00     1.00
##  [2,]  2.00     16.87     4.00     0.00
##  [3,]  3.00     21.23     8.00     0.01
##  [4,]  4.00     24.89    12.00     0.02
##  [5,]  5.00     32.69    16.00     0.01
##  [6,]  6.00     33.56    20.00     0.03
##  [7,]  7.00     35.20    24.00     0.07
##  [8,]  8.00     39.54    28.00     0.07
##  [9,]  9.00     51.82    32.00     0.01
## [10,] 10.00     57.68    36.00     0.01
```




```
# ARCH-LM test
MarchTest(res_co,lag=10)
```

```
## Q(m) of squared series(LM test):
## Test statistic:  14.82856  p-value:  0.1384385
## Rank-based Test:
## Test statistic:  16.70047  p-value:  0.08126024
## Q_k(m) of squared series:
## Test statistic:  110.5098  p-value:  1.58591e-08
## Robust Test(5%) :  40.13442  p-value:  0.4642968
```