Stat Econ 2 Assignment 1

Victor Z. Nygaard

Last edited 28th of October 2021. Recompiled Feb 2024.

Stat Econ 2 Assignment 1

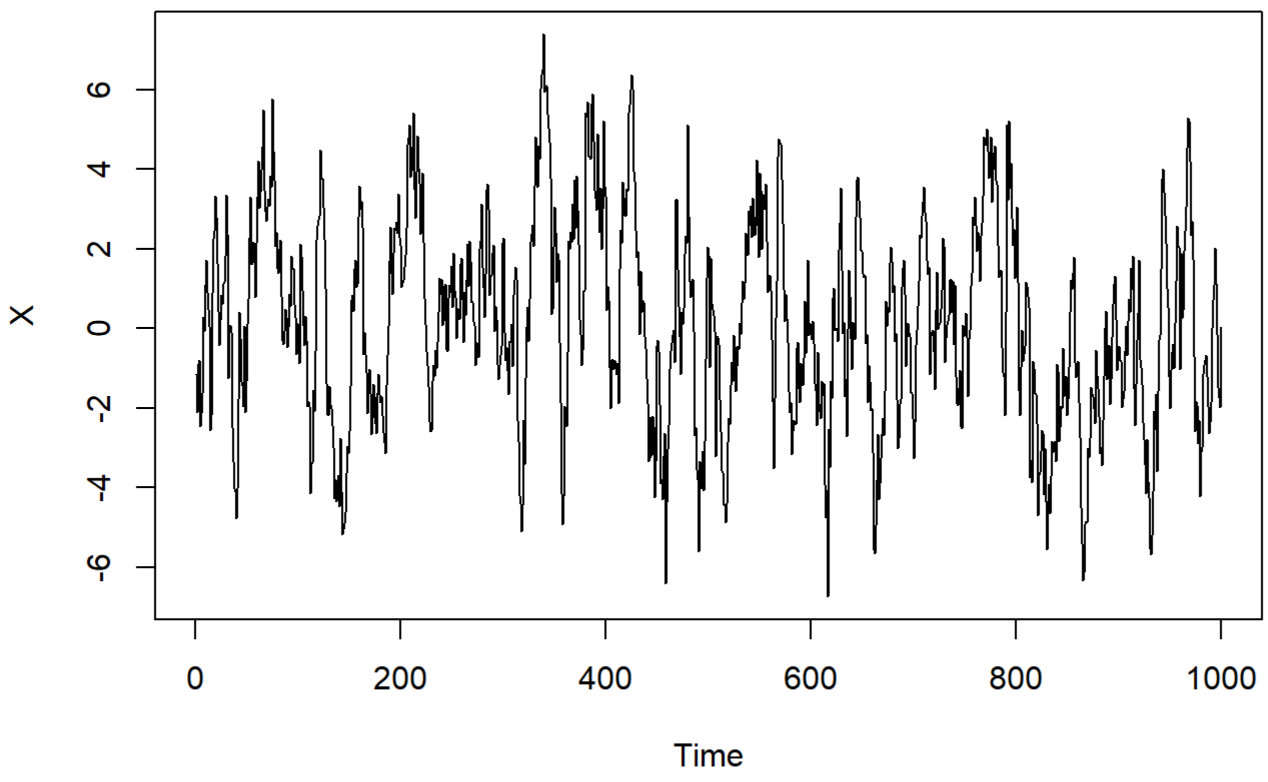
Exercise 1

We might note that we are dealing with an AR(1) process with slope 0.9 and, assuming a central requirement, central t -distributed noise with ten degrees of freedom. We simulate noise terms Z_t for $t = 1, \dots, 1000$

```
set.seed(314)
desiredlag <- 20
phi <- 0.9
n <- 10^3
Z <- rt(df = 10, n=n)
```

As such we may simulate the AR(1) process using the update scheme defining the ‘stopped’ AR(1) process

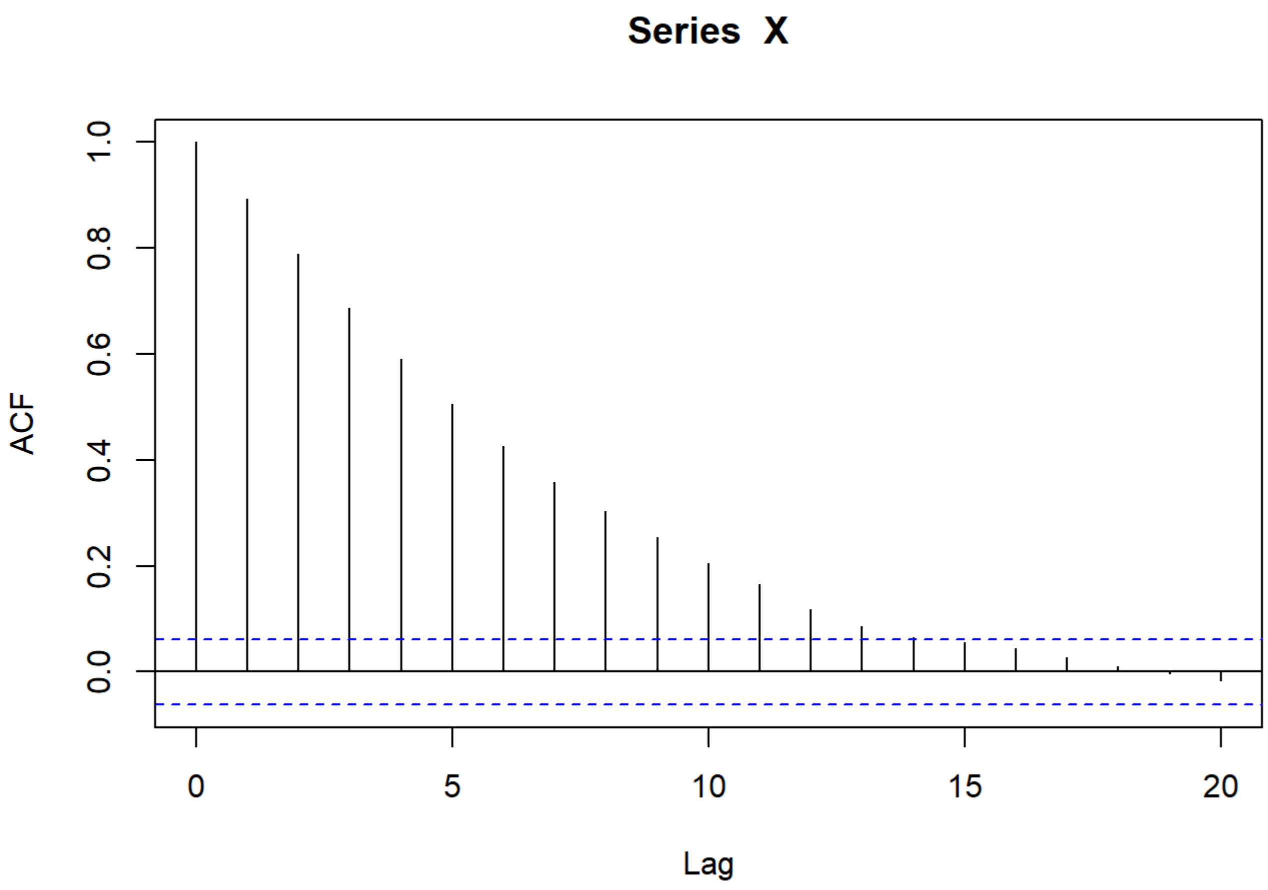
```
X <- rep(NA,n)
X[1] <- Z[1]
for (j in 2:n) {
   X[j] <- phi*X[j-1]+Z[j]
}
ts.plot(X)
```



a)

We plot the `acf` confidence bands:

```
acf(X, lag.max = desiredlag, plot = T)
```



```
#acf(X, lag.max = desiredlag, plot = F)
```

Assuming the requirement to sample from X, and not from Z which would be more in line with the theoretical foundation for permutations of data, which are in and of themselves understood here, to be a random reordering of all data points. We may create a matrix containing 10^3 rows, each a permutation of the X data as such:

```
m <- 10^3
M <- matrix(NA, nrow = m, ncol = n)
for (i in 1:m) {
   M[i,] <- sample(X)
}
```

For each of these 'permuted' data sets, we will use `acf` to calculate sample autocorrelations for lags once again up to `desiredlag` $= 20$, and then the 95% quantiles for each of the lags:

```
AutM <- apply(M,1,acf,lag.max=desiredlag, plot = F)
temp <- matrix(NA,nrow = m, ncol = desiredlag+1)
for (i in 1:1000) {
   temp[i,] <- AutM[[i]][[1]]
}
quanties <- apply(temp,2,quantile, prob=c(0.025,0.975))
```

We may then also plot the resulting quantiles:

```
#ggplot(quanties) + geom_line(aes())
#groupby?
```

b)

We note that our AR(1) model is causal as $|\phi| < 1.$ Following example 4.25, we will need to calculate the sample autocovariance function:

$$\gamma_{n,X}(h) := \frac{1}{n}\sum_{t=1}^{n-h}\left(X_t - \overline{X}_n\right)\left(X_{t+h} - \overline{X}_n\right)$$

and autocorrelation function:

$$\rho_{n,X}(h) := \frac{\gamma_{n,X}(h)}{\gamma_{n,X}(0)}$$

Note thus in particular that we may calculate $\gamma_{n,X}(1)$, $\gamma_{n,X}(0)$ in R with the following homemade function

```
gammaf <- function(X,h) {
   n <- length(X)
   gamt <- 0
   for (t in 1:(n-h)) {
      tempt <- (X[t]-mean(X))*(X[t+h]-mean(X))
      gamt <- gamt + tempt
   }
   1/n*gamt
}
```

Yielding

```
gammaf(X,1)
```

```
## [1] 5.640264
```

```
gammaf(X,0)
```

```
## [1] 6.317588
```

Such that for

$$\hat{\phi}_n = \frac{\gamma_{n,X}(1)}{\gamma_{n,X}(0)} \equiv \rho_{n,X}(1)$$

$$\hat{\sigma}_n^2 = \gamma_{n,X}(0)\left(1 - \rho_{n,X}^2(1)\right)$$

```
(rho1 <- gammaf(X,1)/gammaf(X,0))
```

```
## [1] 0.8927876
```

```
(phih <- rho1)
```

```
## [1] 0.8927876
```

```
(sigmah2 <- gammaf(X,0)*(1-rho1^2))
```

```
## [1] 1.28203
```

Let $\nu = 10$ be the degrees of freedom of our student-t distributed random noise Z_t. As is surmised on page 47-48 in the lecture notes, in dealing with a causal AR(1) process driven by iid noise Z_t with variance $\sigma^2 = \frac{\nu}{\nu-2} = \frac{10}{8} = \frac{5}{4}$, we have asymptotic normality of $\hat{\phi}$ with corresponding asymptotic mean ϕ and asymptotic variance $\frac{\sigma^2\Gamma_p^{-1}}{n}$ i.e.

$$\hat{\phi}_n \overset{as}{\sim} \mathcal{N}\left(\phi, \frac{\sigma^2\Gamma_{p=1}^{-1}}{n}\right)$$

or equivalently

$$\sqrt{n}\left(\hat{\phi} - \phi\right) \xrightarrow{d} \mathcal{N}\left(0, \sigma^2\Gamma_1^{-1}\right).$$

With this we may for our fixed $n = 1000$ determine that $\left(\phi - \frac{1.96}{\sqrt{n}}\sqrt{\sigma^2\Gamma_1^{-1}},\, \phi + \frac{1.96}{\sqrt{n}}\sqrt{\sigma^2\Gamma_1^{-1}}\right)$ will be an asymptotic 95% confidence interval for ϕ. Estimating Γ_1 via the sample autocovariance function, we find:

$$\tilde{\Gamma}_1 := \gamma_{n,X}(1-1) = \gamma_{n,X}(0) = 6.3175881$$

such that

$$\tilde{\Gamma}_1^{-1} = \frac{1}{\tilde{\Gamma}_1} = 0.1582883 \neq 0$$

such that we may rewrite the confidence bands as

$$\left(\phi - \frac{1.96}{\sqrt{n}}\sqrt{\sigma^2 0.1582883},\, \phi + \frac{1.96}{\sqrt{n}}\sqrt{\sigma^2 0.1582883}\right)$$

Inserting the other estimates and $n = 1000$ we get

$$\left(\hat{\phi} - \frac{1.96}{\sqrt{10^3}}\sqrt{\hat{\sigma}^2 0.1582883},\, \hat{\phi} + \frac{1.96}{\sqrt{10^3}}\sqrt{\hat{\sigma}^2 0.1582883}\right)$$
$$= \left(0.8927876 - \frac{1.96}{\sqrt{10^3}}\sqrt{0.2029302},\, 0.8927876 + \frac{1.96}{\sqrt{10^3}}\sqrt{0.2029302}\right)$$
$$= (0.8648667, 0.9207085)$$

c)

i+ii)

We calculate the residuals:

```
Zh <- rep(NA,n)
Zh[1] <- X[1]
for (j in 2:n) {
   Zh[j] <- X[j]+phih*X[j-1]
}
```

We do a reordering of these in the requested fashion using `sample`:

```
Zhs<-sample(Zh)
```

We define a new sample:

```r
Xhs <- rep(NA,n)
Xhs[1] <- Zhs[1]
for (j in 2:n) {
   Xhs[j] <- phih*Xhs[j-1]+Zhs[j]
}
```

iii)

part1)

As in b)

```r
(hsrho1 <- gammaf(Xhs,1)/gammaf(Xhs,0))
```

```
## [1] 0.8541978
```

```r
(hsphih <- hsrho1)
```

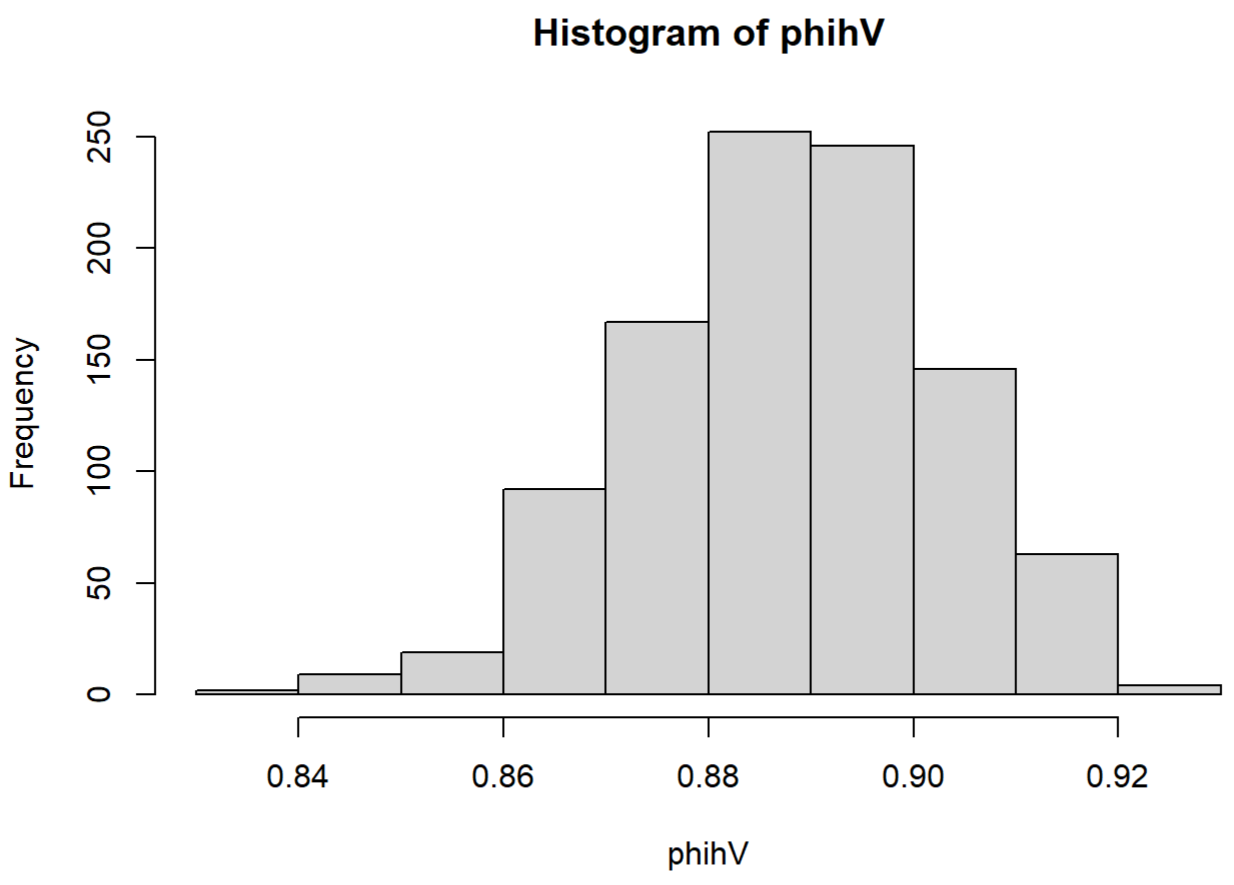
```
## [1] 0.8541978
```

part2)

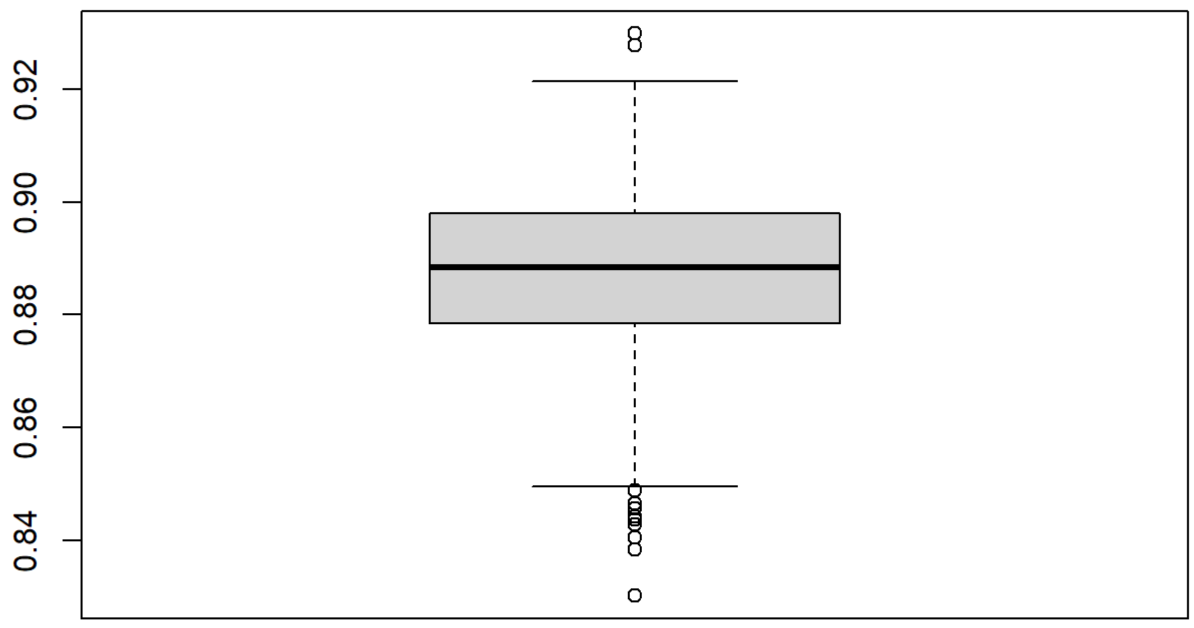
We repeat the tasks done in the previous exercises by writing a function to do so

```r
boots <- function(Zh,n) { #Simulating the Bootstrap data
   q <- length(Zh)
   XhsM <- matrix(NA,nrow=n,ncol=q)
   for (i in 1:n) {
      Zhsi <- sample(Zh)
      XhsM[i,1] <- Zhsi[1]
      for (j in 2:q) {
        XhsM[i,j] <- phih*XhsM[i,j-1]+Zhsi[j]
      }
   }
   XhsM
}

dat <- boots(Zh,1000)
YW <- function(Zh,n) { #Calculating the YW's
   q <- length(Zh)
   dat <- boots(Zh,n)
   phihV <- rep(NA,n)
   for (i in 1:n) {
      phihV[i] <- gammaf(dat[i,],1)/gammaf(dat[i,],0)
   }
   phihV
}
phihV <- YW(Zh,n)
hist(phihV)
```



```
boxplot(phihV)
```



```
quantile(phihV, c(0.025, 0.975))
```

```
##      2.5%     97.5% 
## 0.8587749 0.9149991
```

Comparing this confidence interval to the asymptotic one achieved in b):

```
rbind(quantile(phihV, c(0.025, 0.975)), c(phih - 1.96/(sqrt(n))*sqrt(sigmah2*1/gammaf(X,0)),phih + 1.9
6/(sqrt(n))*sqrt(sigmah2*1/gammaf(X,0))))
```

```
##           2.5%     97.5%
## [1,] 0.8587749 0.9149991
## [2,] 0.8648667 0.9207085
```

we notice a great similarity, though the asymptotic confidence interval seems shifted approximately $\cong 0.007$ in comparison to the bootstrap interval.

Exercise 2

We may import the data

```
Data <- read_csv("Data.csv",col_types =
                       cols(col_date(),
                            col_double(),
                            col_double(),
                            col_double(),
                            col_double(),
                            col_double(),
                            col_integer()))
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
head(Data, 10)
```

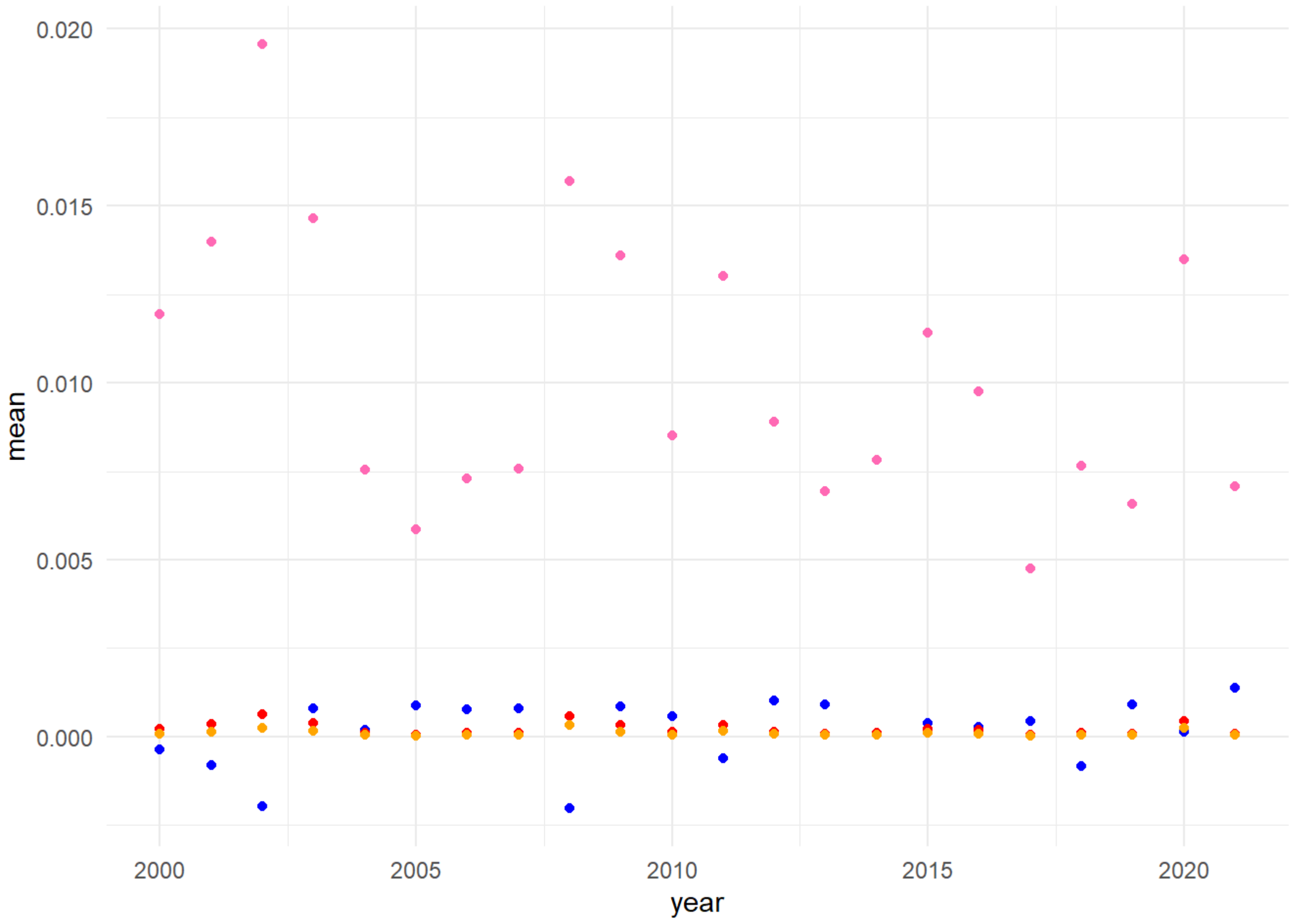
```
## # A tibble: 10 × 7
##    Date        Open  High   Low Close `Adj Close` Volume
##    <date>     <dbl> <dbl> <dbl> <dbl>       <dbl>  <int>
##  1 1990-03-01 1796. 1796. 1796. 1796.       1796.      0
##  2 1990-03-02 1805. 1805. 1805. 1805.       1805.      0
##  3 1990-03-05 1838. 1838. 1838. 1838.       1838.      0
##  4 1990-03-06 1820. 1820. 1820. 1820.       1820.      0
##  5 1990-03-07 1842. 1842. 1842. 1842.       1842.      0
##  6 1990-03-08 1862. 1862. 1862. 1862.       1862.      0
##  7 1990-03-09 1859. 1859. 1859. 1859.       1859.      0
##  8 1990-03-12 1844. 1844. 1844. 1844.       1844.      0
##  9 1990-03-13 1867. 1867. 1867. 1867.       1867.      0
## 10 1990-03-14 1877. 1877. 1877. 1877.       1877.      0
```

We may create a yearly data set, and filter for data after 2000 and remove `NA` 's

```
Data_new <- Data %>% filter(year(Date)>=2000) %>%  mutate(returns = (Close - lag(Close))/lag(Close),
                                                    logreturns = log(1+returns),
                                                    abslogreturn = abs(logreturns),
                                                    absdiff = abs(returns - logreturns))
Data_new_clean <- Data_new %>% na.omit()
Data_new_new <- Data_new %>% group_by(year(Date)) %>% rename('year' = 'year(Date)') %>% summarise(mean
= mean(logreturns, na.rm = T), var = var(logreturns, na.rm = T), absmean = mean(abs(logreturns), na.rm
= T), absvar = var(abs(logreturns), na.rm = T))
```

We plot these:

```
ggplot(Data_new_new) + geom_point(aes(x=year, y=mean), colour = 'blue') + geom_point(aes(x=year, y=va
r), colour = 'red') + geom_point(aes(x=year, y=absmean), colour = 'hotpink') + geom_point(aes(x=year, y
=absvar), colour = 'orange')
```



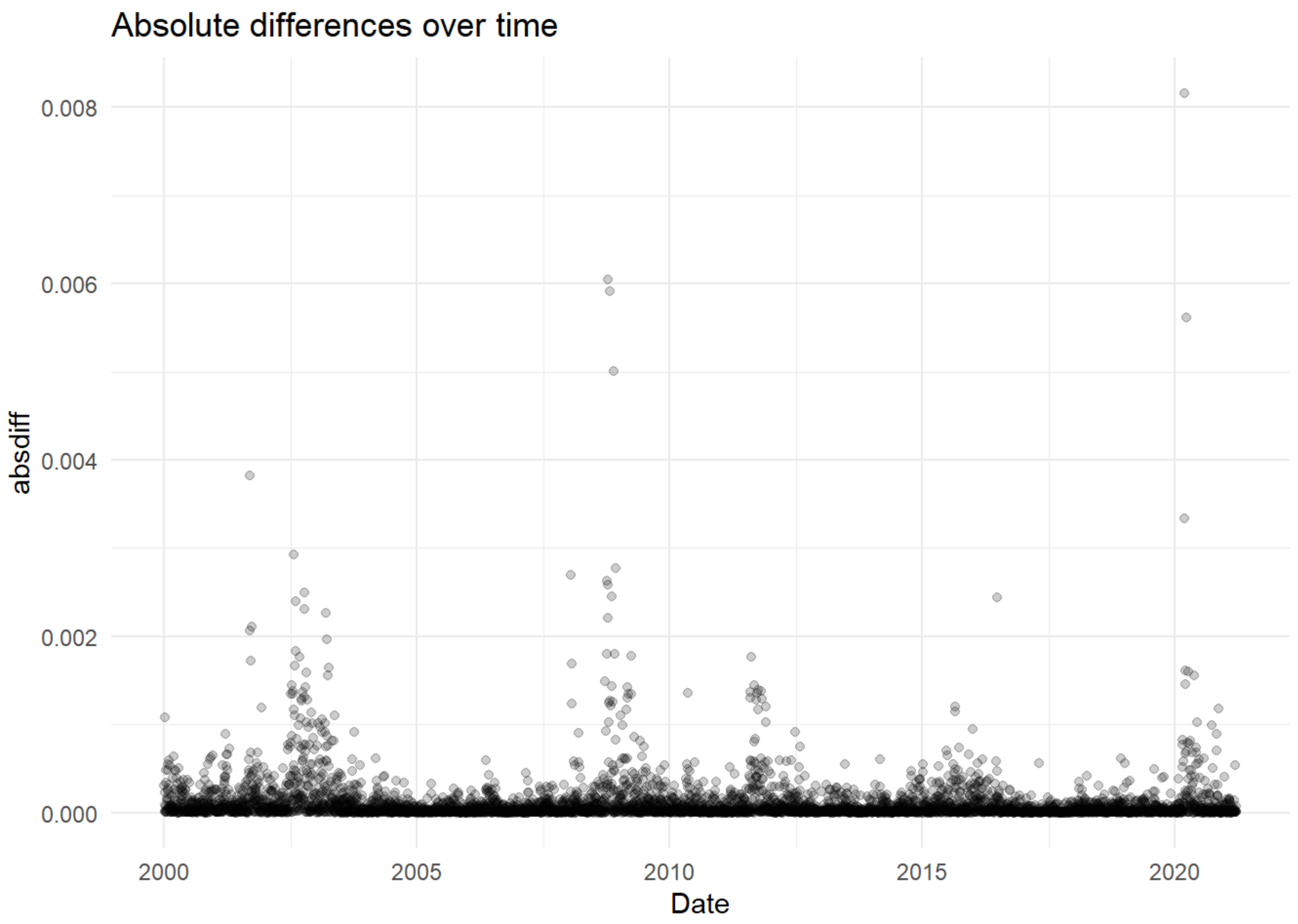
None of the requested quantiles vary a lot, so we cannot reject the possibility of underlying ergodicity.

Exercise 3

a)

We might plot the absolute differences:

```
ggplot(Data_new_clean, aes(x=Date, y=absdiff)) + geom_point(alpha = 0.2) + ggtitle("Absolute difference
s over time")
```



and calculate the maximum of the absolute difference between returns and logreturns:

```
max(Data_new_clean$absdiff)
```

```
## [1] 0.008162438
```

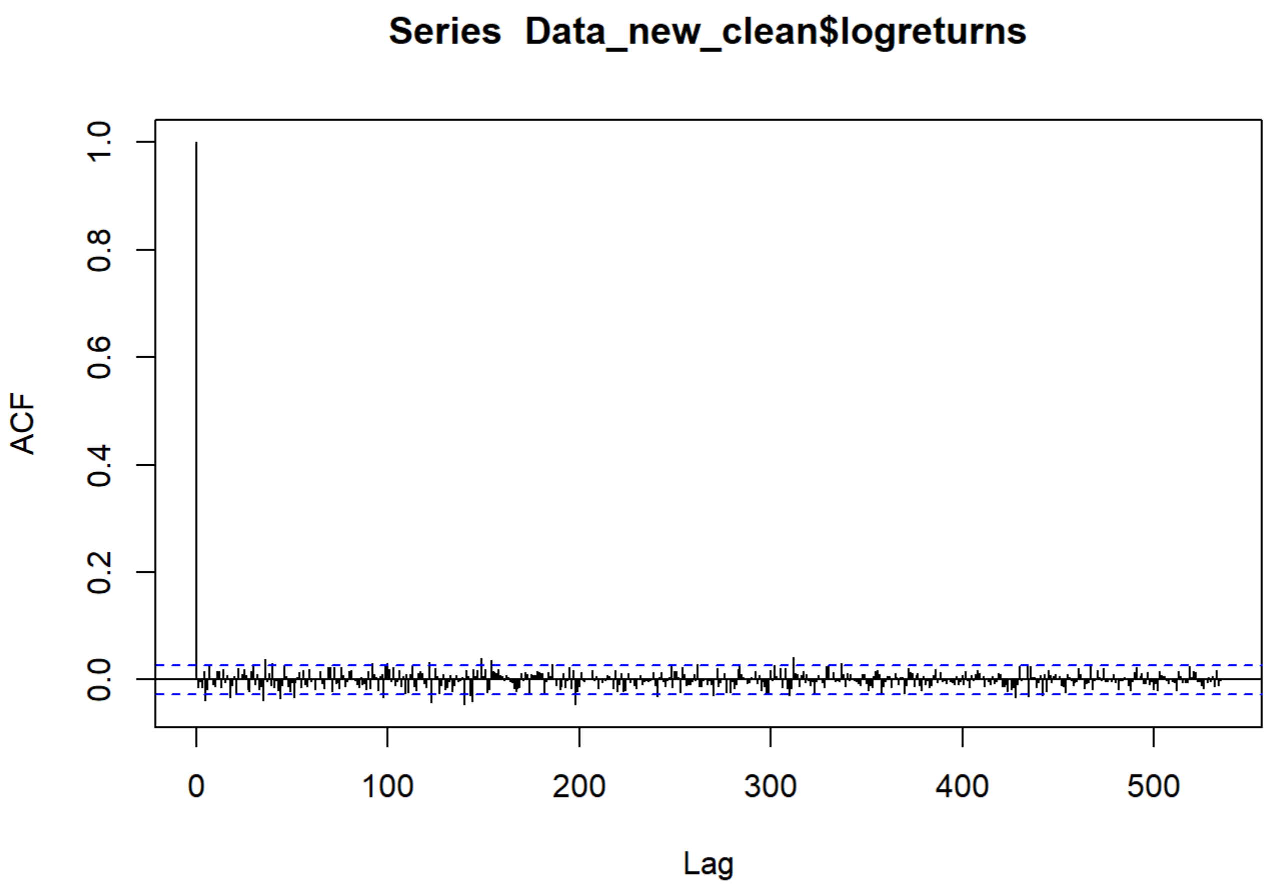
b)

We note that the dataset `Data_new_clean` has `nrow(Data_new_clean) =5353` data points, such that 10% of the sample size of the log return time series will be of the size `floor(nrow(Data_new_clean)/10) =535` . We may use `acf` to calculate this many lags for the sample autocorrelation function for the log-return time series, its absolute value, and its square:

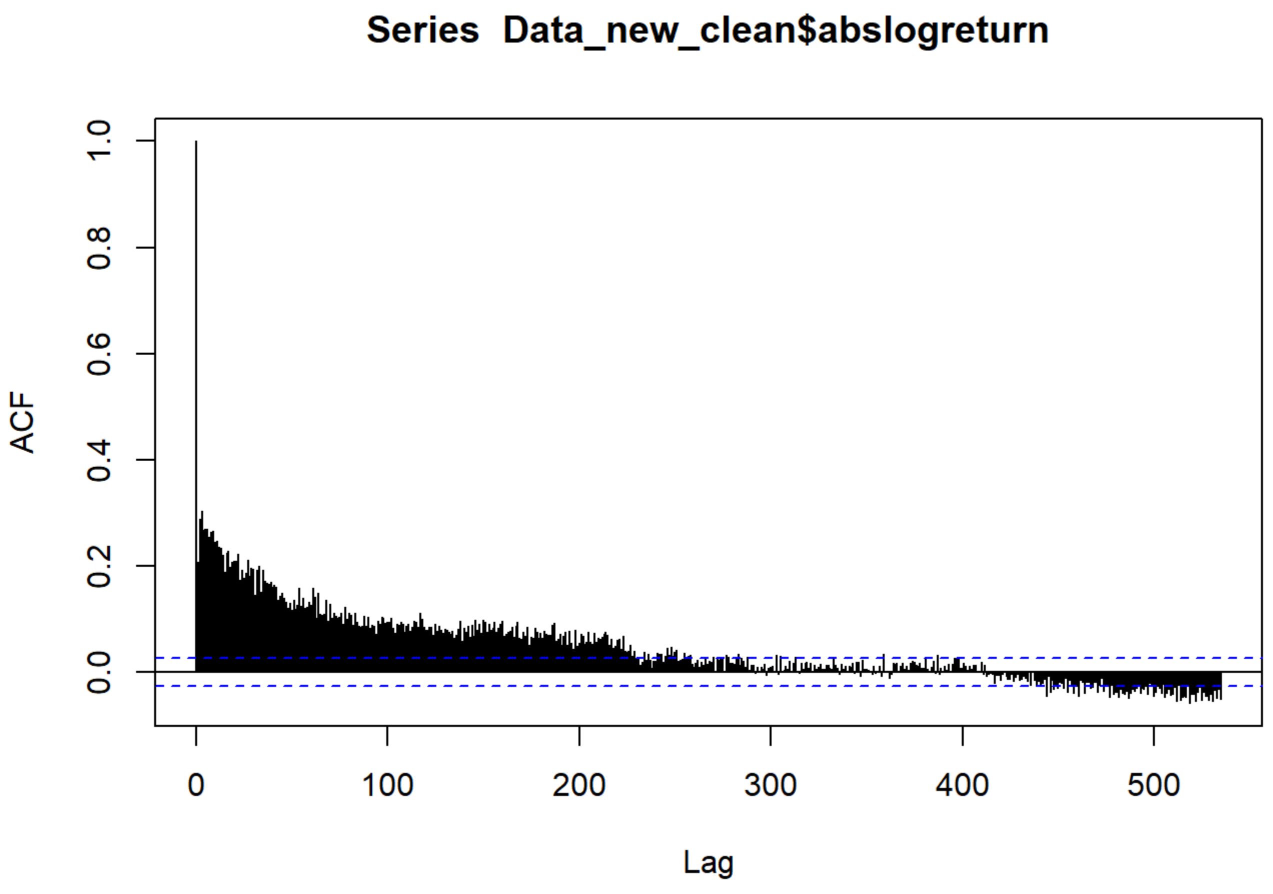
```
autocorLogRet <- acf(Data_new_clean$logreturns, lag.max = floor(nrow(Data_new_clean)/10), plot=F)
autocorLogRetAbs <- acf(Data_new_clean$abslogreturn, lag.max = floor(nrow(Data_new_clean)/10), plot=F)
autocorLogRetSquared <- acf((Data_new_clean$logreturns)^2, lag.max = floor(nrow(Data_new_clean)/10), pl
ot=F)
```

We may also choose to plot each of these:

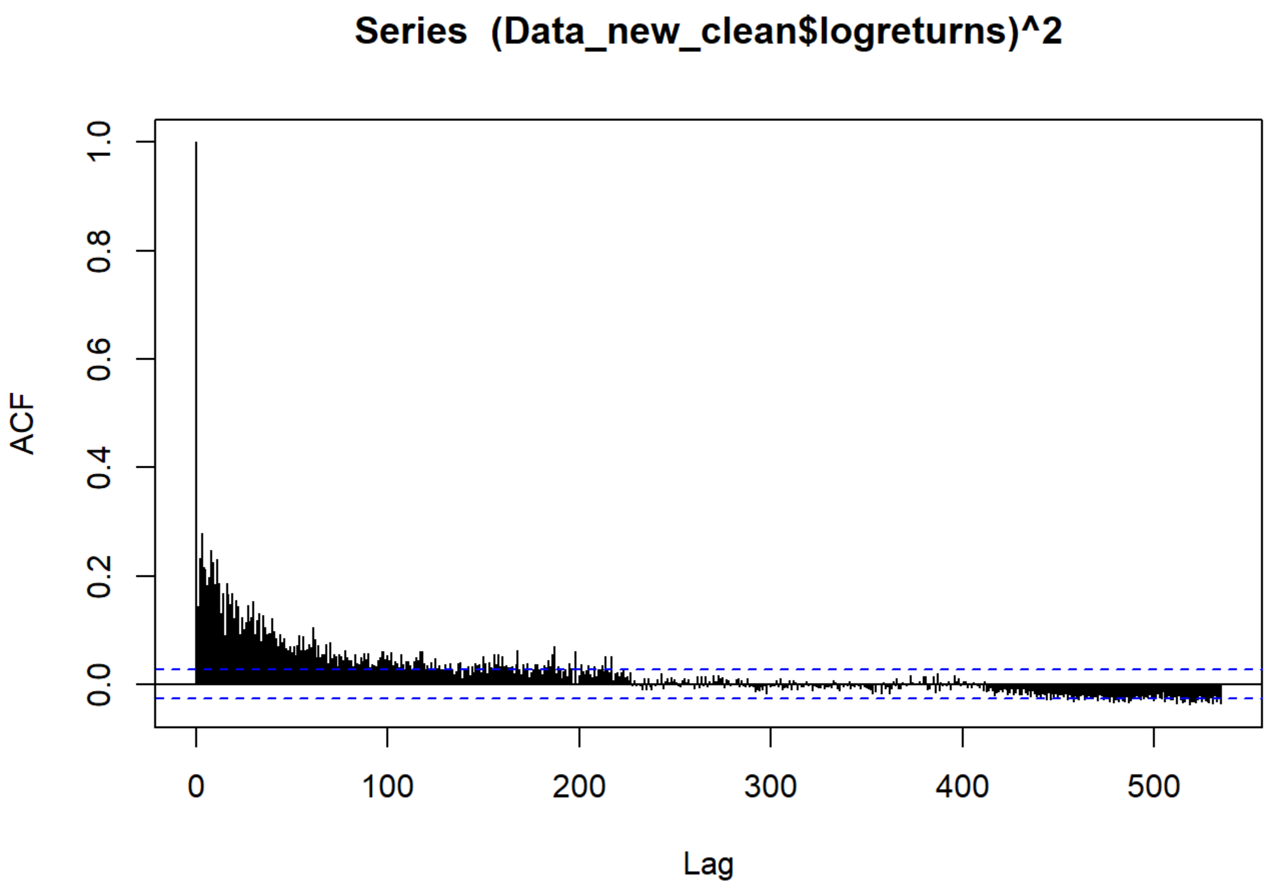
```
acf(Data_new_clean$logreturns, lag.max = floor(nrow(Data_new_clean)/10), plot=T)
```



```
acf(Data_new_clean$abslogreturn, lag.max = floor(nrow(Data_new_clean)/10), plot=T)
```



```
acf((Data_new_clean$logreturns)^2, lag.max = floor(nrow(Data_new_clean)/10), plot=T)
```



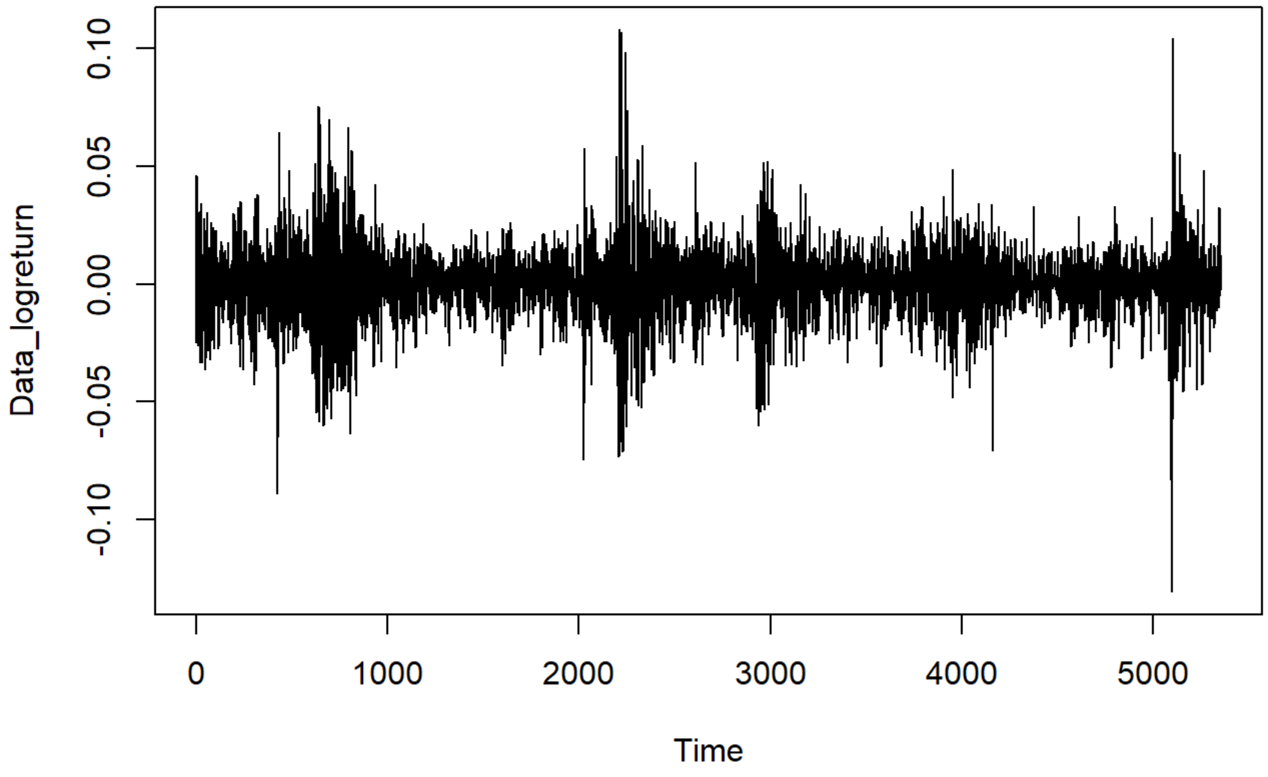
c)

We will fit the AR model using `ar.yw`

```
Data_logreturn <- Data_new_clean[,9][[1]]
head(Data_logreturn)
```

```
## [1] -0.024564608 -0.012969888 -0.004184320  0.046182439  0.021094463
## [6] -0.004960651
```

```
ts.plot(Data_logreturn)
```



```
modellr <- ar.yw(Data_logreturn, aic = T)
```

We may see a summary of the model:

```
print(modellr)
```

```
##
## Call:
## ar.yw.default(x = Data_logreturn, aic = T)
##
## Coefficients:
##       1        2        3        4        5        6        7
## -0.0132  -0.0030  -0.0145   0.0156  -0.0385  -0.0183   0.0289
##
## Order selected 7  sigma^2 estimated as  0.0002197
```

```
summary(modellr)
```

```
##              Length Class  Mode
## order           1   -none- numeric
## ar              7   -none- numeric
## var.pred        1   -none- numeric
## x.mean          1   -none- numeric
## aic            38   -none- numeric
## n.used          1   -none- numeric
## n.obs           1   -none- numeric
## order.max       1   -none- numeric
## partialacf     37   -none- numeric
## resid        5353   -none- numeric
## method          1   -none- character
## series          1   -none- character
## frequency       1   -none- numeric
## call            3   -none- call
## asy.var.coef   49   -none- numeric
```

d)

We may simulate based on the built up model:

```
modellr$ar
```

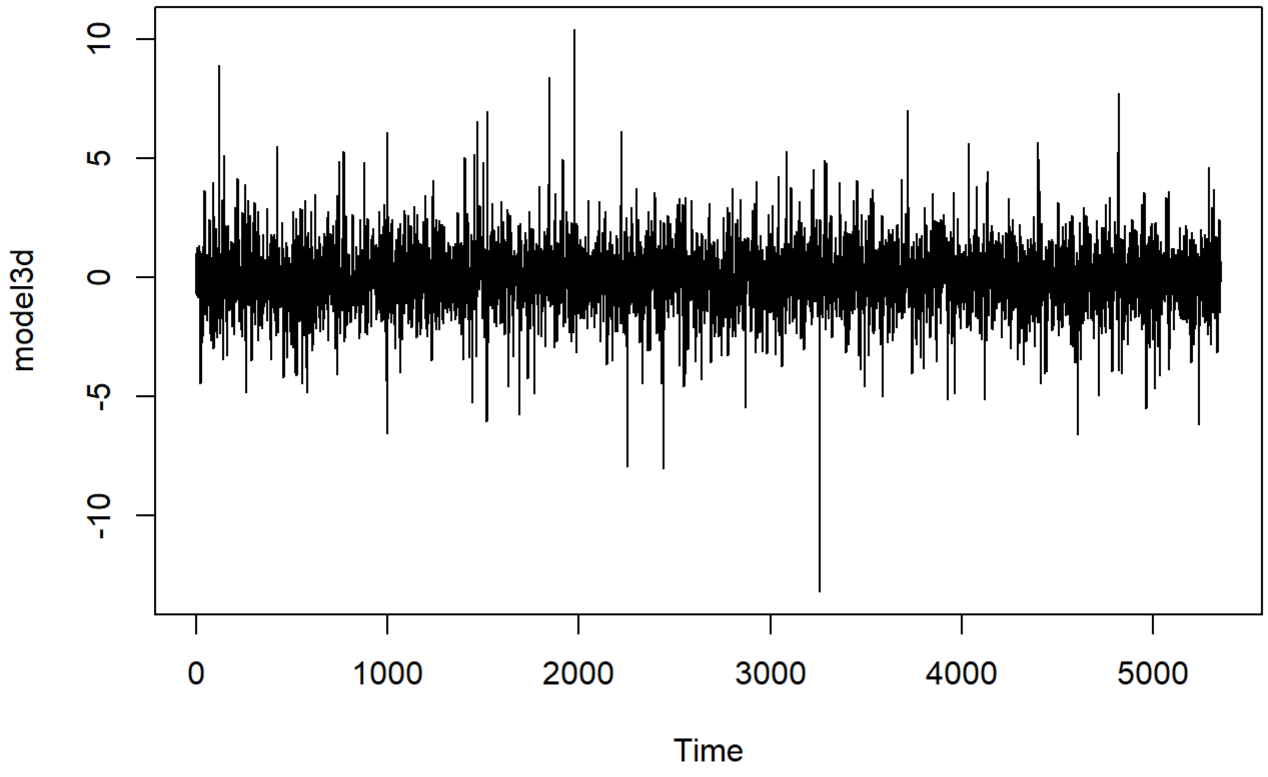
```
## [1] -0.013222942 -0.003024756 -0.014527039  0.015641328 -0.038478367
## [6] -0.018285413  0.028921894
```

```
model3d <- arima.sim(model = list(ar = modellr$ar), n = length(Data_logreturn), rand.gen = function
(n,...) rt(n,df=4))
summary(model3d)
```

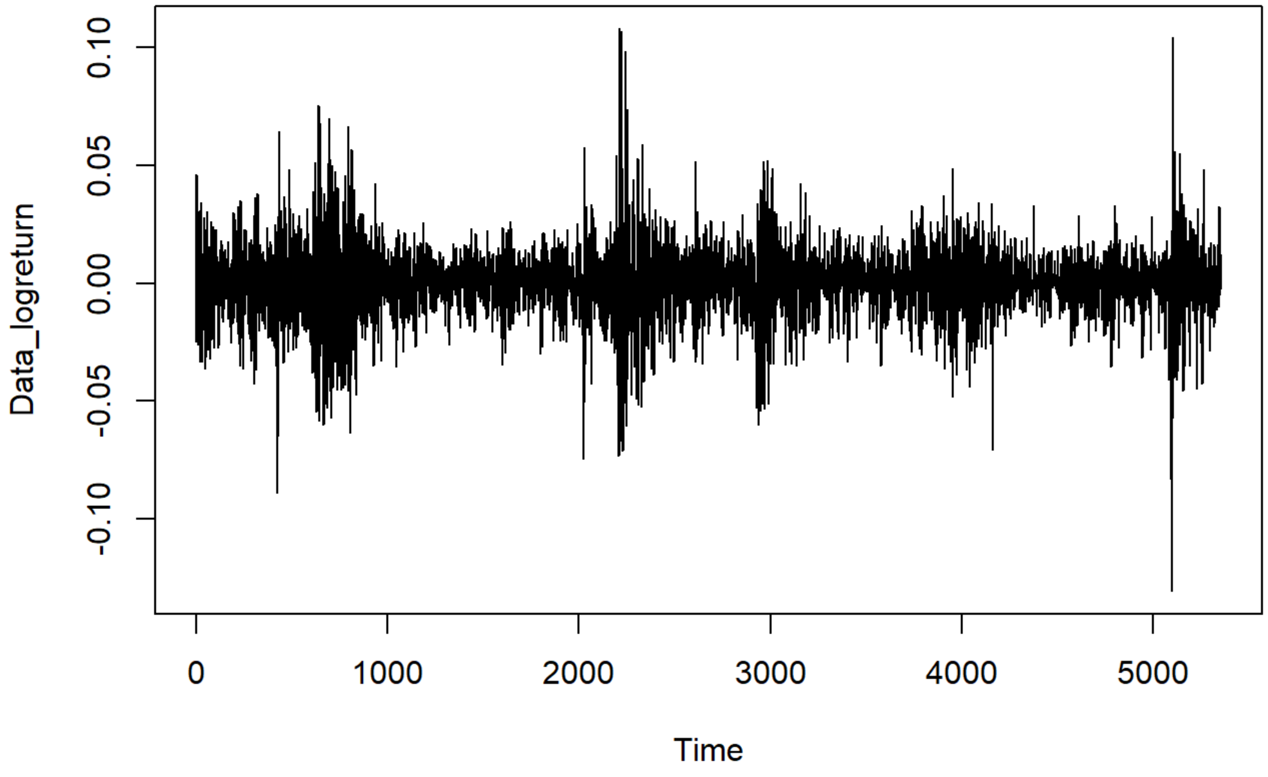
```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -13.198054  -0.729220  -0.010417  -0.007745   0.730539  10.396367
```

We may plot the series, together with the original log-returns:

```
plot(model3d)
```

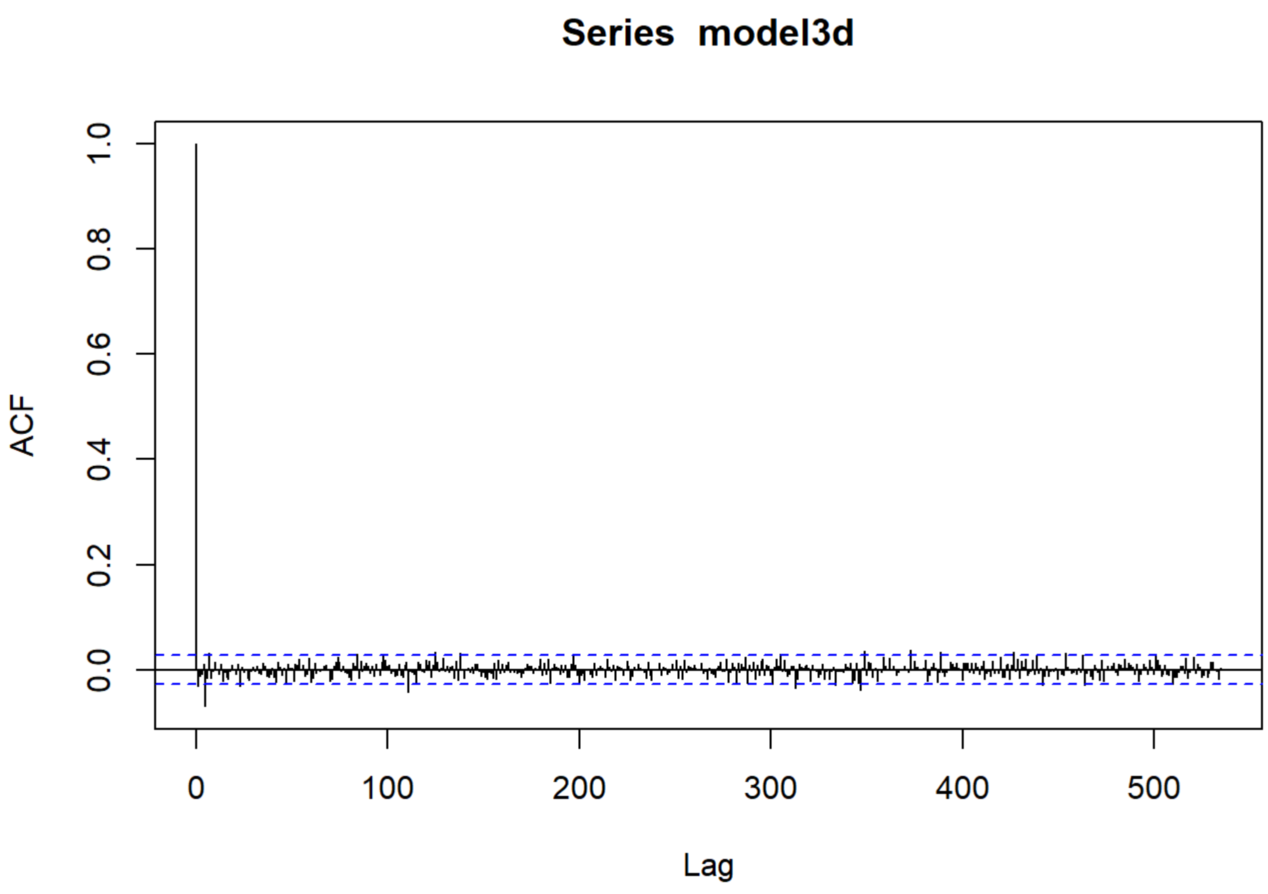


```
ts.plot(Data_logreturn)
```

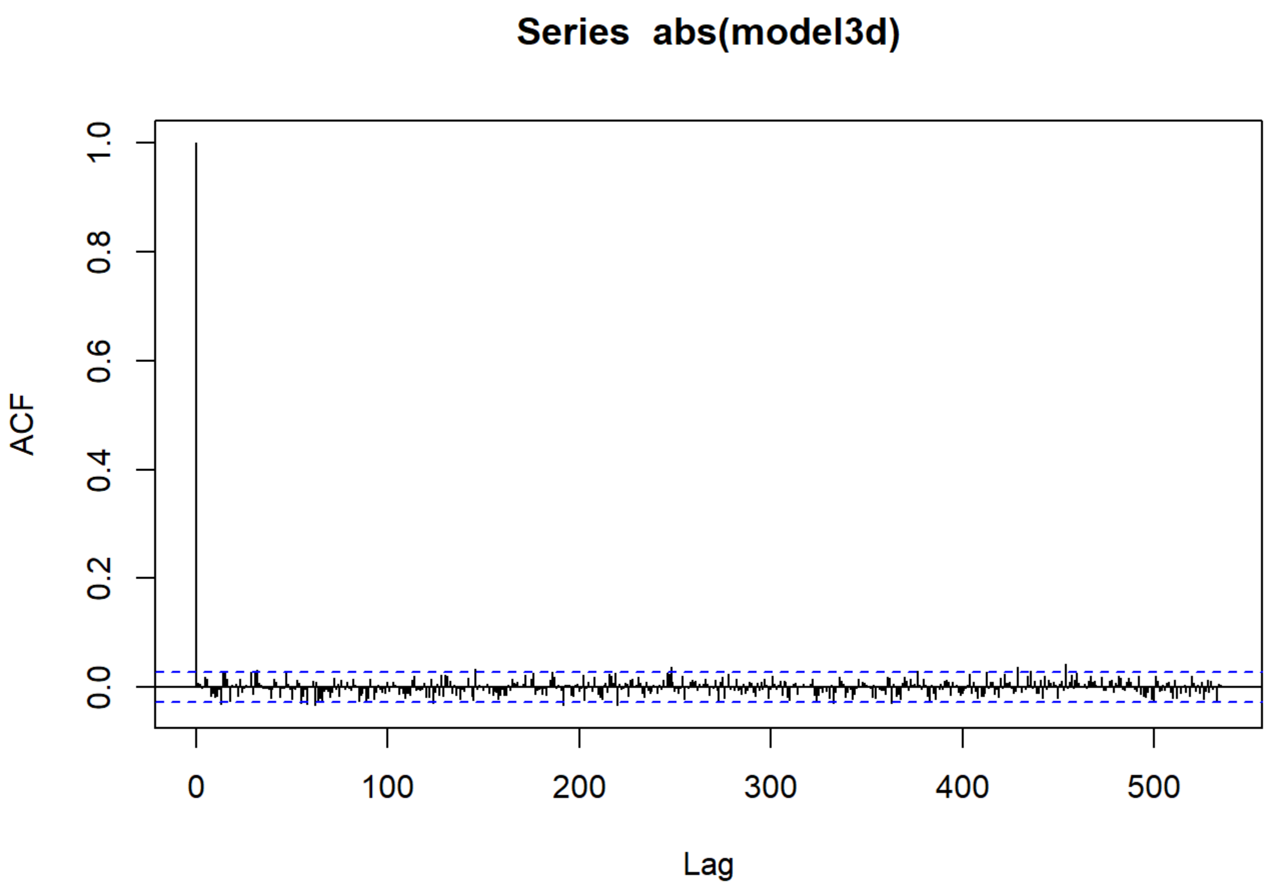


and the requested autocorrelations:

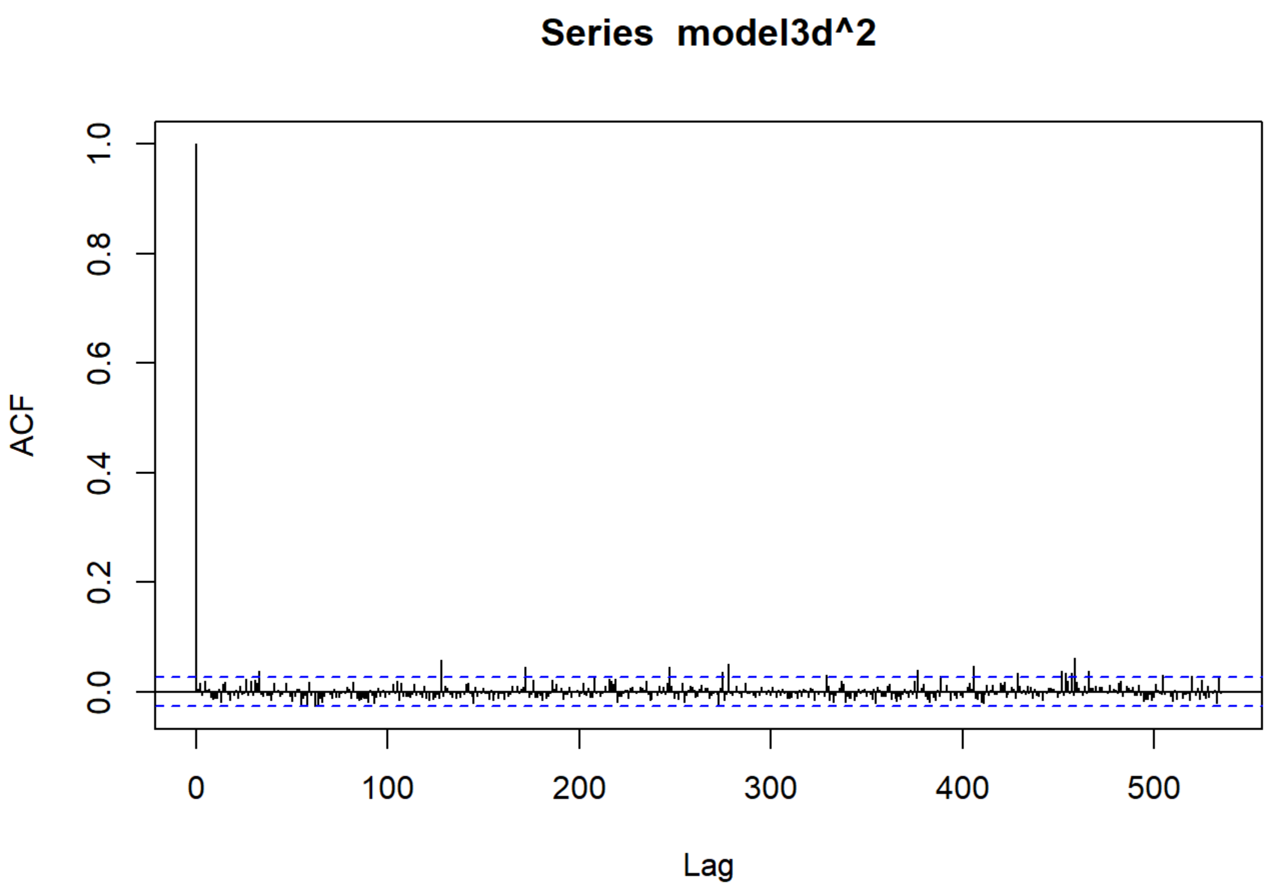
```
acf(model3d,lag.max = floor(nrow(Data_new_clean)/10), plot=T)
```



```
acf(abs(model3d), lag.max = floor(nrow(Data_new_clean)/10), plot=T)
```



```
acf(model3d^2, lag.max = floor(nrow(Data_new_clean)/10), plot=T)
```



We might note that comparing the plot of `model3d` to the log-return series, a striking difference appears, in that the model seems to attain values orders of magnitude higher, than there is in the original log-returns time series:

We may also note that while the autocorrelation function of the untransformed `model3d` looks rather similar to that of untransformed log-return series, the transformations appear dissimilar, possibly hinting at another problem. It seems that `model3d` as a model does not perform adequately in these circumstances, such that a different model, or a different choice of noise term might be needed.

Exercise 4

a)

Noting that $\rho_X(h) = \phi^{|h|}$ we may compute

$$\begin{aligned}
w_{hh} &= \sum_{k=1}^{\infty} \left(\rho_X(k+h) + \rho_X(k-h) - 2\rho_X(h)\rho_X(k)\right)^2 \\
&= \sum_{k=1}^{\infty} \left(\phi^{|k+h|} + \phi^{|k-h|} - \phi^{|k|}\phi^{|h|}\right)^2 \\
&= \sum_{k=1}^{\infty} \left(\phi^{|k-h|} - \phi^{|k+h|}\right)^2 \\
&= \sum_{k=1}^{h} \left(\phi^{h-k} - \phi^{k+h}\right)^2 + \sum_{k=h+1}^{\infty} \left(\phi^{k-h} - \phi^{k+h}\right)^2 \\
&= \sum_{k=1}^{h} \phi^{2h}\left(\phi^{-k} - \phi^{k}\right)^2 + \sum_{k=h+1}^{\infty} \phi^{2k}\left(\phi^{-h} - \phi^{h}\right)^2 \\
&= \phi^{2h} \sum_{k=1}^{h} \left(\phi^{-k} - \phi^{k}\right)^2 + \left(\phi^{-h} - \phi^{h}\right)^2 \sum_{k=h+1}^{\infty} \phi^{2k} \\
&= -\frac{\phi^{2h}\left(1 + 2h - \phi^2 - 2h\phi^2 - \phi^{-2h} + \phi^{2+2h}\right)}{1-\phi^2} + \frac{\phi^{2h+2}\left(\phi^{-h} - \phi^{h}\right)^2}{1-\phi^2}
\end{aligned}$$

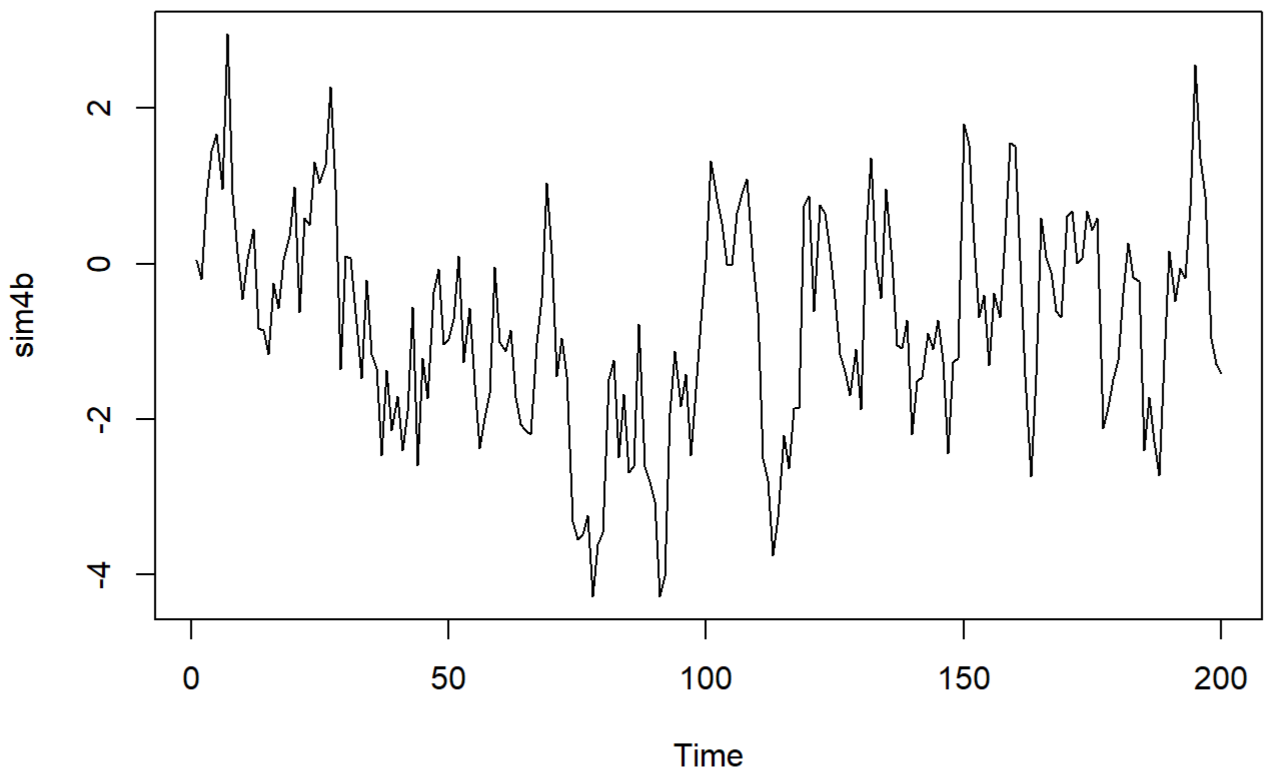
such that for $0 < \phi < 1$ we may conclude that for $h \to \infty$

$$w_{hh} \to \frac{1}{1-\phi^2} + \frac{\phi^2}{1-\phi^2}.$$

b)

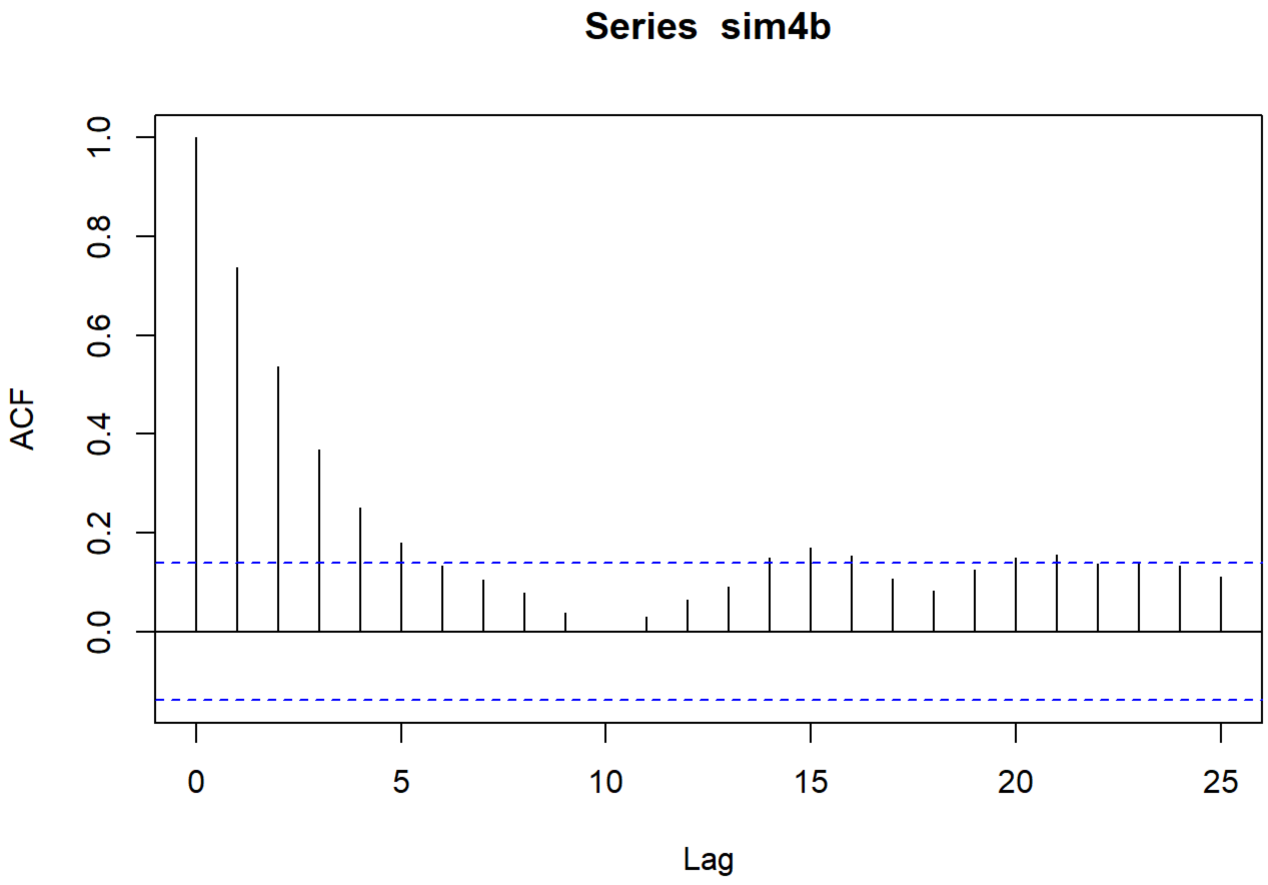
We will simulate a size 200 sample from the AR(1) process as follows:

```
sim4b <- arima.sim(model = list(ar = 0.8), n=200)
ts.plot(sim4b)
```



We may draw the sample auotcorrelations, along with the confidence band for the iid white noise for a 25 lag:

```
acf(sim4b, lag.max = 25)
```

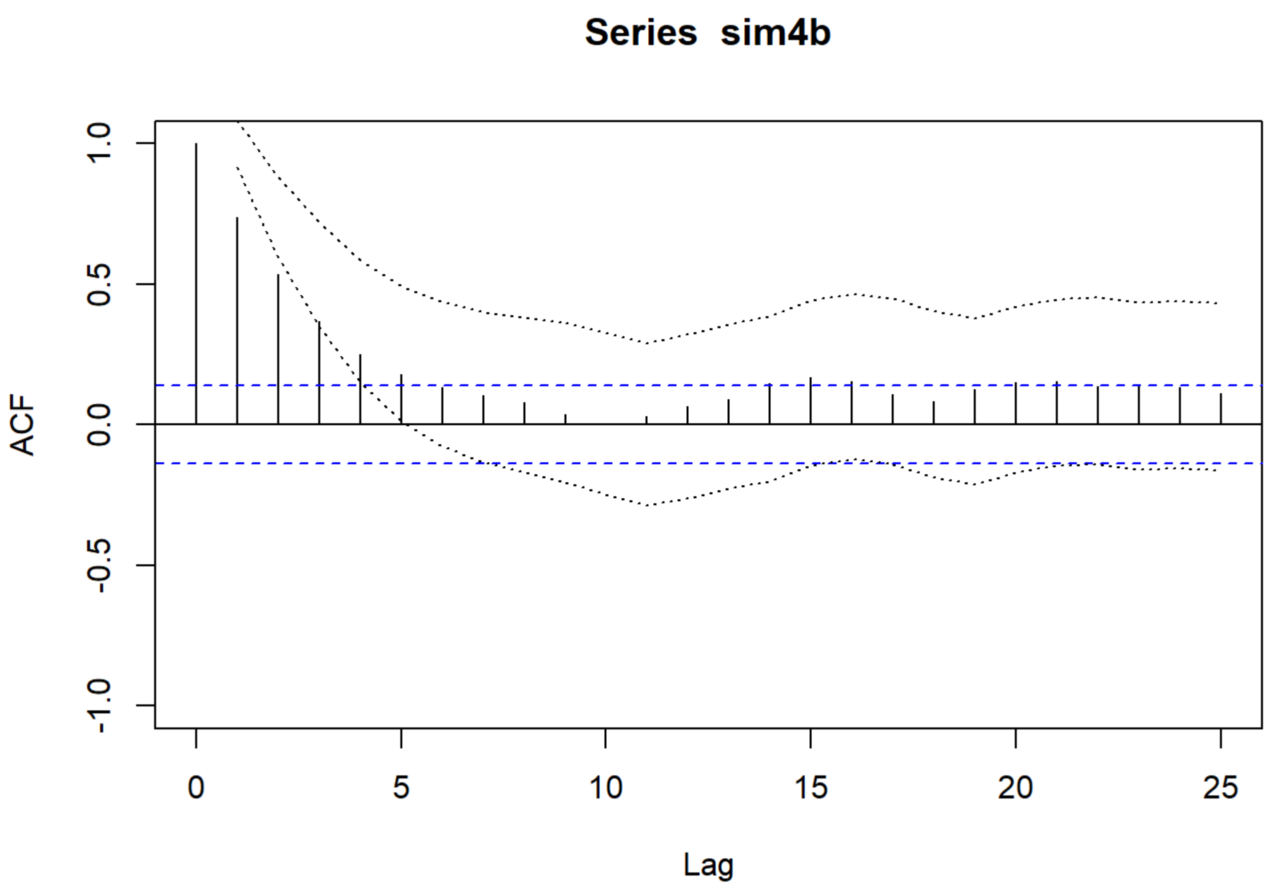


for each lag up to 25 we may do the calculation of the 95% asymptotic confidence bands from a)

```
n4b <- 200
phi4b <- 0.8
rhonX4b <- as.numeric(acf(sim4b, lag.max = 25, plot=F)[[1]])
w <- 1:25
autocorconfp <- 1:25
autocorconfm <- 1:25
for (h in w) {
   w[h] <- -((phi4b^(2*h))*(1+2*h-phi4b^2-2*h*phi4b^2-phi4b^(-2*h)+phi4b^(2+2*h)))/(1-phi4b^2)+(phi4b^
(2*h+2)*(phi4b^(-h)-phi4b^h)^2)/(1-phi4b^2)
   autocorconfp[h] <- rhonX4b[h]+qnorm(0.975)*sqrt(w[h]/n4b)
   autocorconfm[h] <- rhonX4b[h]-qnorm(0.975)*sqrt(w[h]/n4b)
}
autocorconf <- rbind(autocorconfp, autocorconfm)
```

We may plot the asymptotic correlation confidence bands together with the previous sample auto correlation function

```
acf(sim4b, lag.max = 25, ylim = c(-1,1));lines(1:25, autocorconfp, lty = 3);lines(1:25, autocorconfm, l
ty = 3)
```



Exercise 5

We may import the dataset, convert it to `xts` and calculate last day of each year

```
ss <- as.xts(sunspots)
ep <- ss %>% endpoints("years")
```

With this, we will aggregate the data averaging over each year:

```
ssy <- ss %>% period.apply(INDEX = ep, FUN = mean)
```

```
## NOTE: `period.apply(..., FUN = mean)` operates by column, unlike other math
##   functions (e.g. median, sum, var, sd). Please use `FUN = colMeans` instead,
##   and use `FUN = function(x) mean(x)` to take the mean of all columns. Set
##   `options(xts.message.period.apply.mean = FALSE)` to suppress this message.
```

```
head(ssy)
```

```
##              [,1]
## Dec 1749 80.92500
## Dec 1750 83.39167
## Dec 1751 47.65833
## Dec 1752 47.80000
## Dec 1753 30.69167
## Dec 1754 12.21667
```

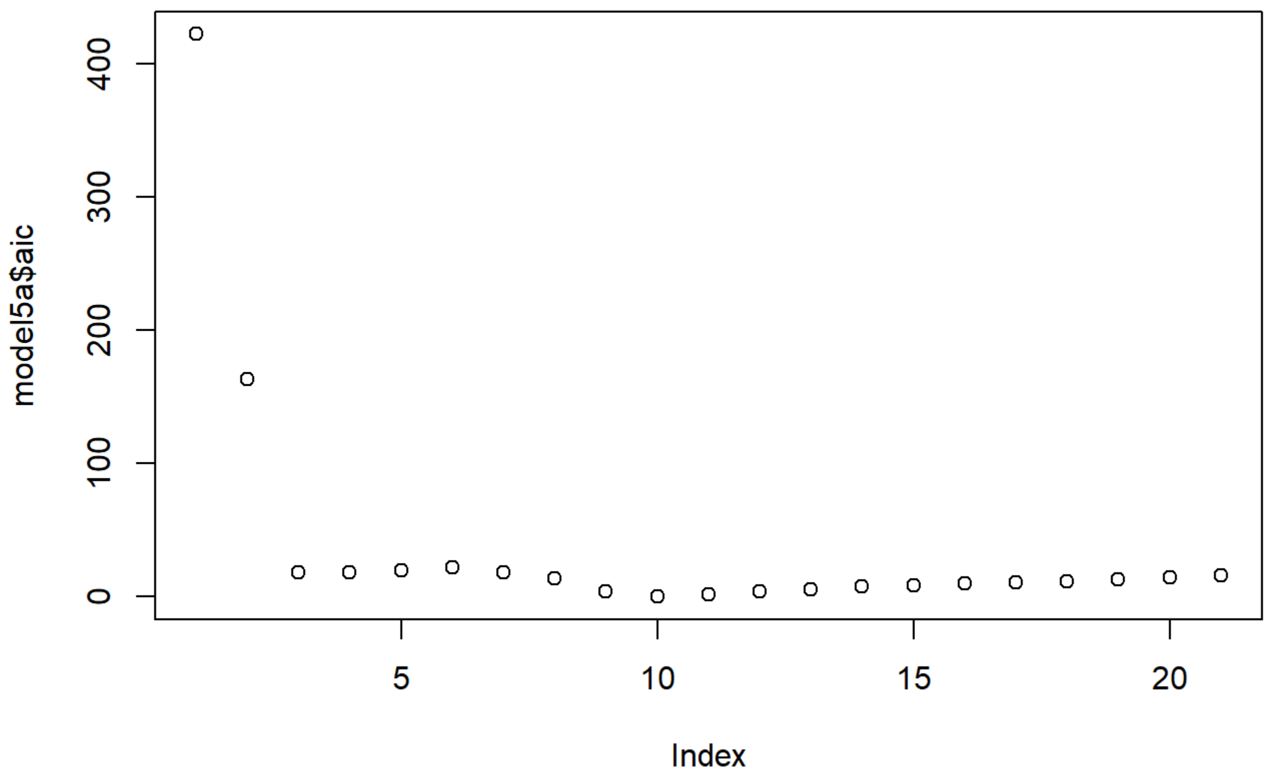
a)

We may calculate AIC using the `ar.yw` command:

```
model5a <- ar.yw(ssy, order.max= 20)
model5a$aic
```

```
##          0          1          2          3          4          5          6 
## 422.180447 162.851001  18.446755  18.064145  19.704643  21.684112  17.947992 
##          7          8          9         10         11         12         13 
##  13.778283   4.123101   0.000000   1.836597   3.759563   5.598154   7.522672 
##         14         15         16         17         18         19         20 
##   8.578587   9.886723  10.866008  11.055926  12.595664  14.138207  16.136743
```

```
plot(model5a$aic)
```



b)

We may let `ar.yw` choose the order of the AR model that minimizes the AIC over `ssy` :

```
model5b <- ar.yw(ssy, aic = T)
print(model5b)
```

```
##
## Call:
## ar.yw.default(x = ssy, aic = T)
##
## Coefficients:
##       1        2        3        4        5        6        7        8
##  1.2187  -0.4644  -0.1339   0.1351  -0.1133   0.0730  -0.0450   0.0189
##       9
##  0.1604
##
## Order selected 9  sigma^2 estimated as  264.2
```

```
summary(model5b)
```

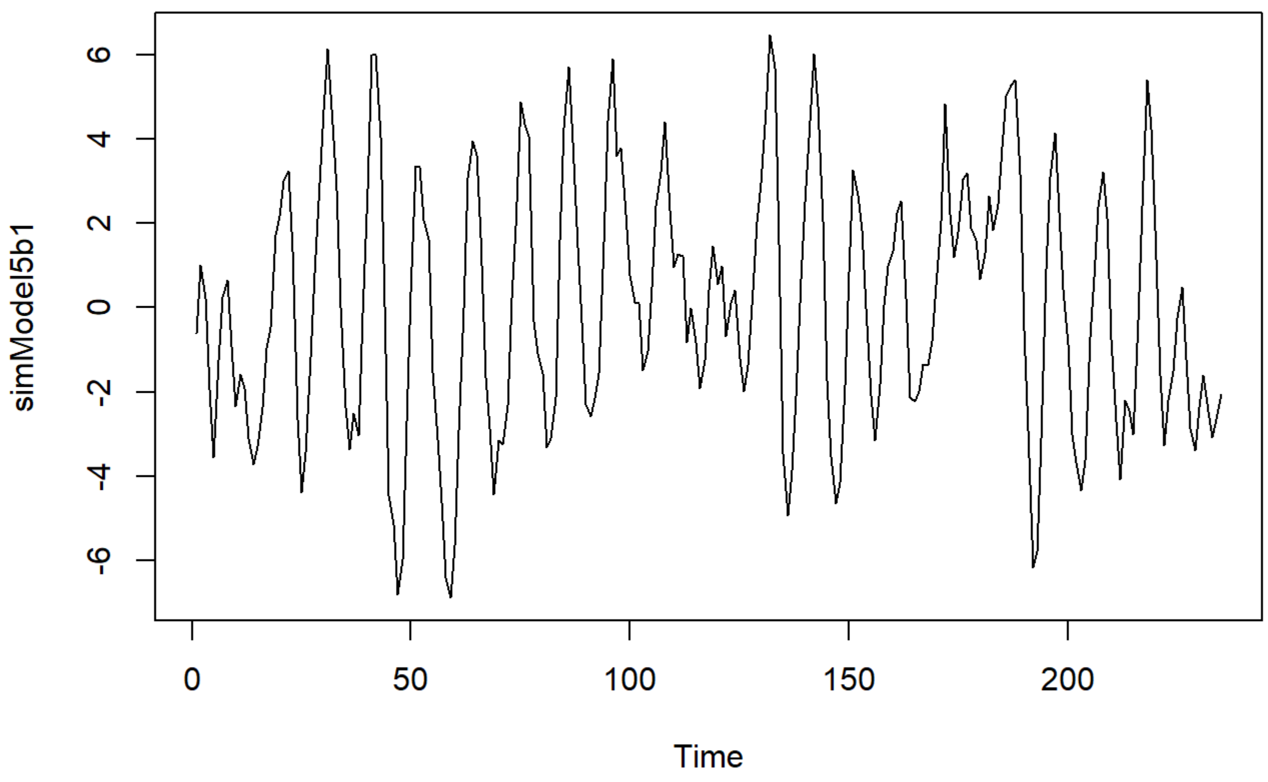
```
##               Length Class  Mode
## order           1    -none- numeric
## ar              9    -none- numeric
## var.pred        1    -none- numeric
## x.mean          1    -none- numeric
## aic            24    -none- numeric
## n.used          1    -none- numeric
## n.obs           1    -none- numeric
## order.max       1    -none- numeric
## partialacf     23    -none- numeric
## resid         235    -none- numeric
## method          1    -none- character
## series          1    -none- character
## frequency       1    -none- numeric
## call            3    -none- call
## asy.var.coef   81    -none- numeric
```

We may then simulate from the model

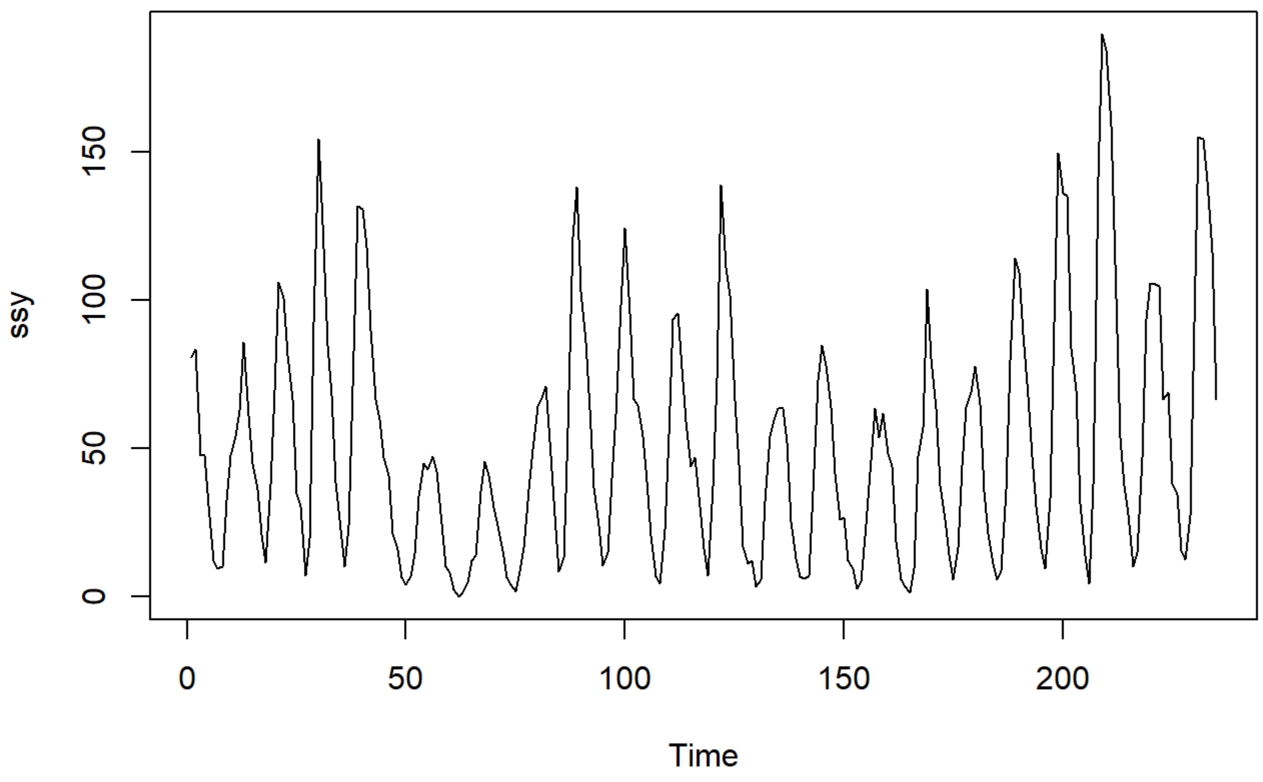
```
simModel5b1 <- arima.sim(model = list(ar = model5b$ar), n = length(ssy))
```

And provide the respective plots:

```
ts.plot(simModel5b1)
```



```
ts.plot(ssy)
```



It seems that the noise terms in the `simModel5b1` that by default are chosen by `arima.sim` to be iid standard normally distributed have one major problem: `simModel5b1` attains the wrong range of values, than required in order to model `ssy`

We will attempt to solve these problems by implementing the suggested solutions to these problems provided. Staying within iid normal noise, we may take a look at the residuals of `model5b` attain their mean and standard deviation, in order to attempt a non-standard normal noise term model

```
head(model5b$resid,15)
```

```
##  [1]         NA         NA         NA         NA         NA         NA
##  [7]         NA         NA         NA  -7.691004  -9.609204   7.470140
## [13]  21.343604 -22.998983   5.214451
```

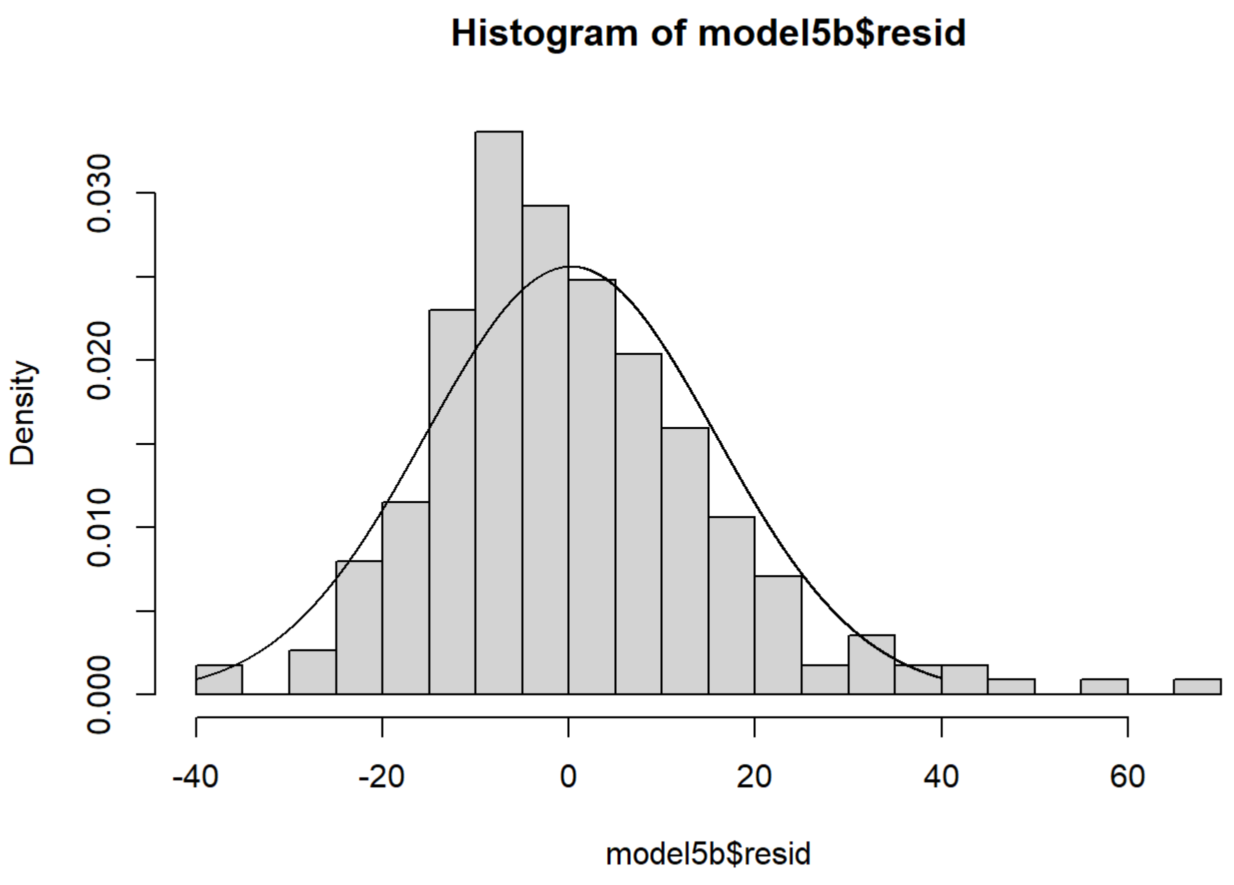
```
mean(model5b$resid, na.rm = T)
```

```
## [1] 0.2039417
```

```
sd(model5b$resid, na.rm = T)
```

```
## [1] 15.58728
```

```
hist(model5b$resid, prob = T, breaks = 20);lines(seq(-40,40,by=0.01), dnorm(seq(-40,40,by=0.01), mean =
mean(model5b$resid, na.rm = T), sd = sd(model5b$resid, na.rm = T)))
```

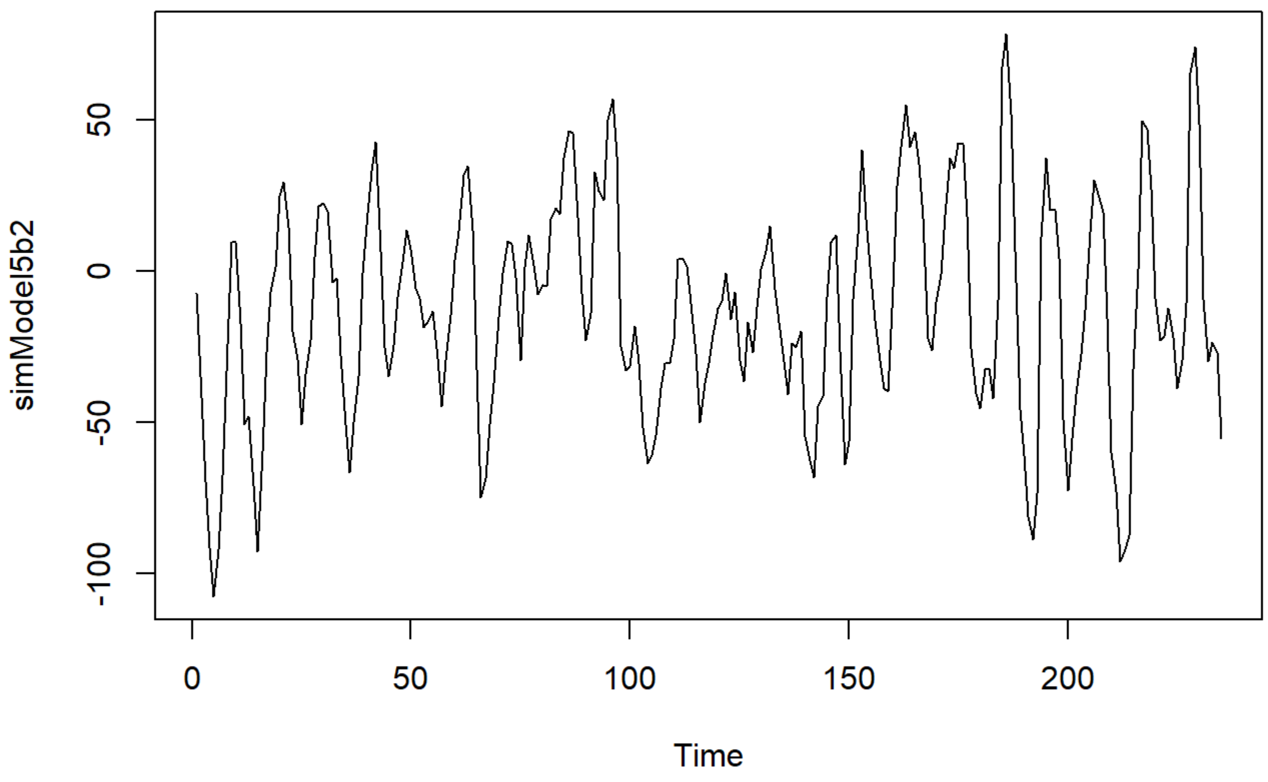


We note from the simple histogram plot above that the fit is adequate, but with resolution lost in the center, probably caused by a few larger values dragging out the summary statistics.

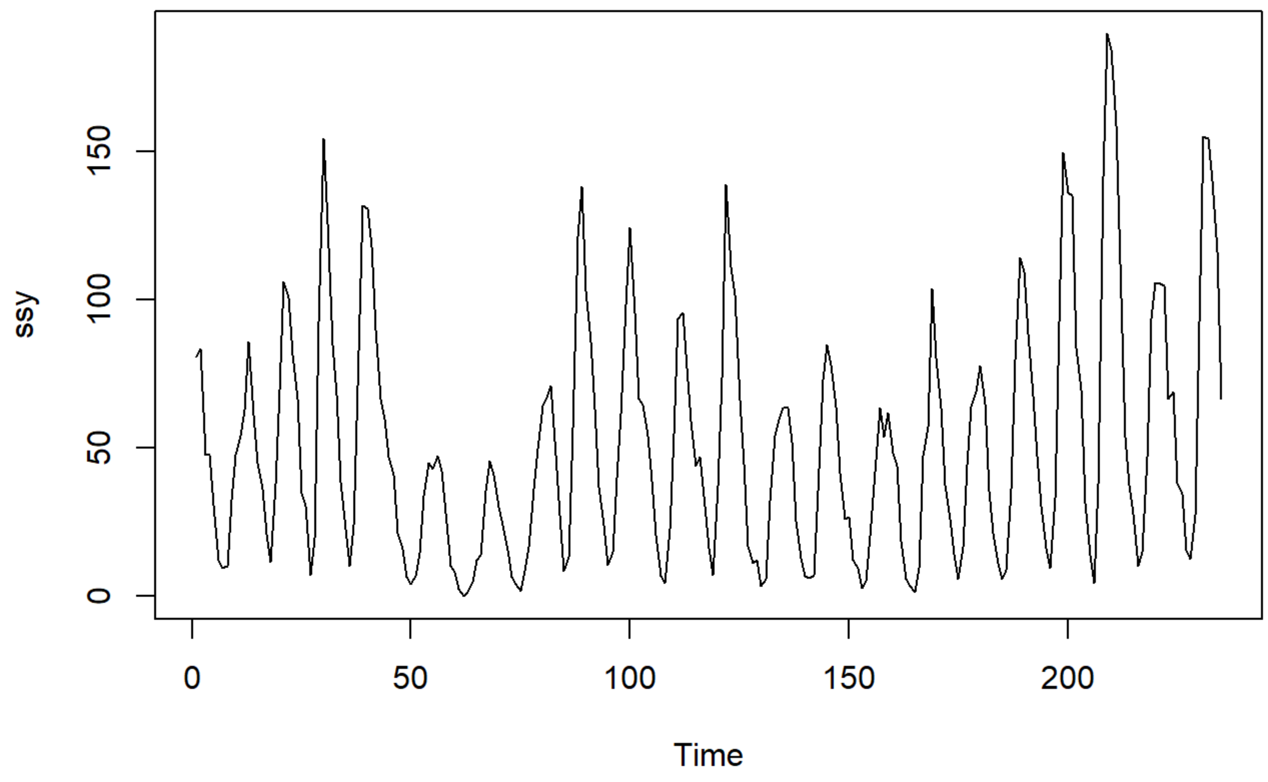
```
simModel5b2 <- arima.sim(model = list(ar = model5b$ar), n = length(ssy), rand.gen = function(n,...) rno
rm(n,mean(model5b$resid, na.rm = T),sd(model5b$resid, na.rm = T)))
```

And provide the respective plots:

```
ts.plot(simModel5b2)
```



```
ts.plot(ssy)
```

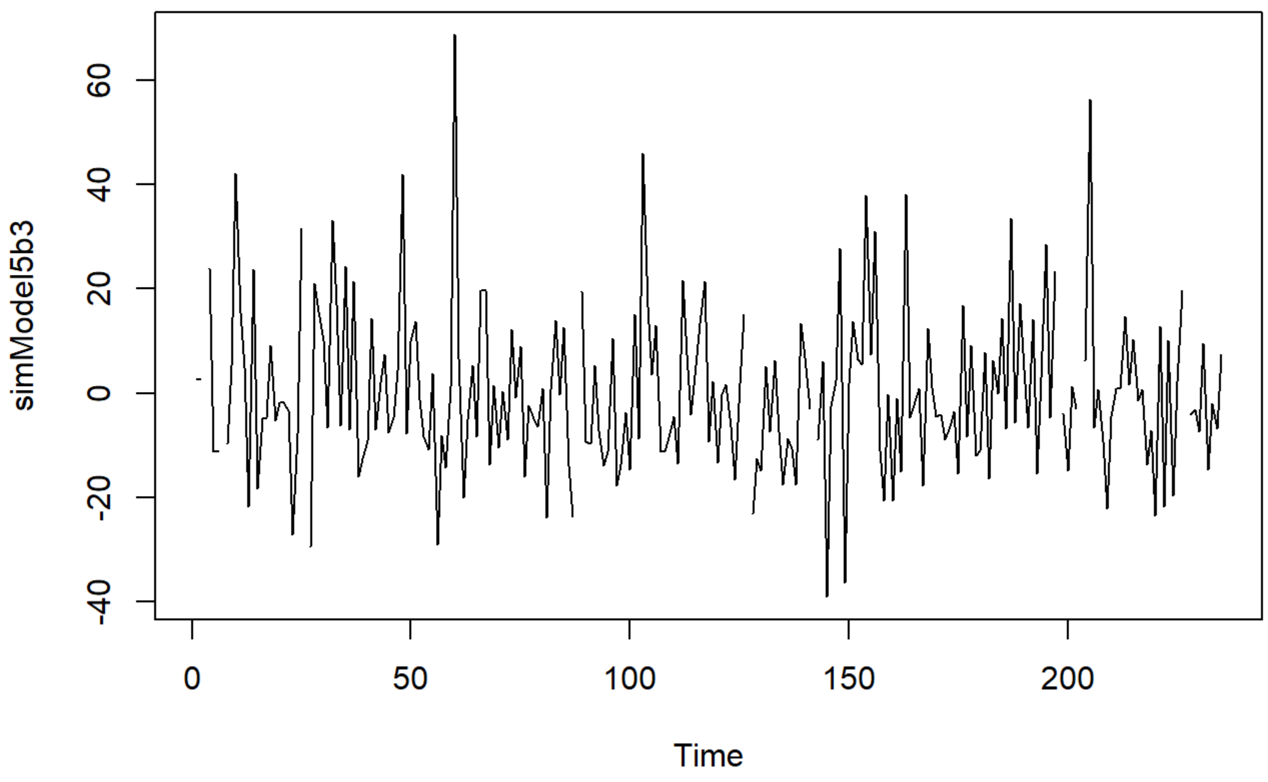


we see that the variability of `simModel5b2` is now appropriate, though still attaining negative values thus requiring further investigation.

```r
simModel5b2new <- arima.sim(model = list(ar = model5b$ar), n = length(ssy), rand.gen = function(n,...)
rcauchy(n,mean(model5b$resid, na.rm = T),sd(model5b$resid, na.rm = T)))
```

Permuting the residuals, we get the following model

```r
sRes <- sample(model5b$resid)
simModel5b3 <- arima.sim(model = list(ar = model5b$coefficients), n = length(ssy), rand.gen = function
(...) sRes)
ts.plot(simModel5b3)
```

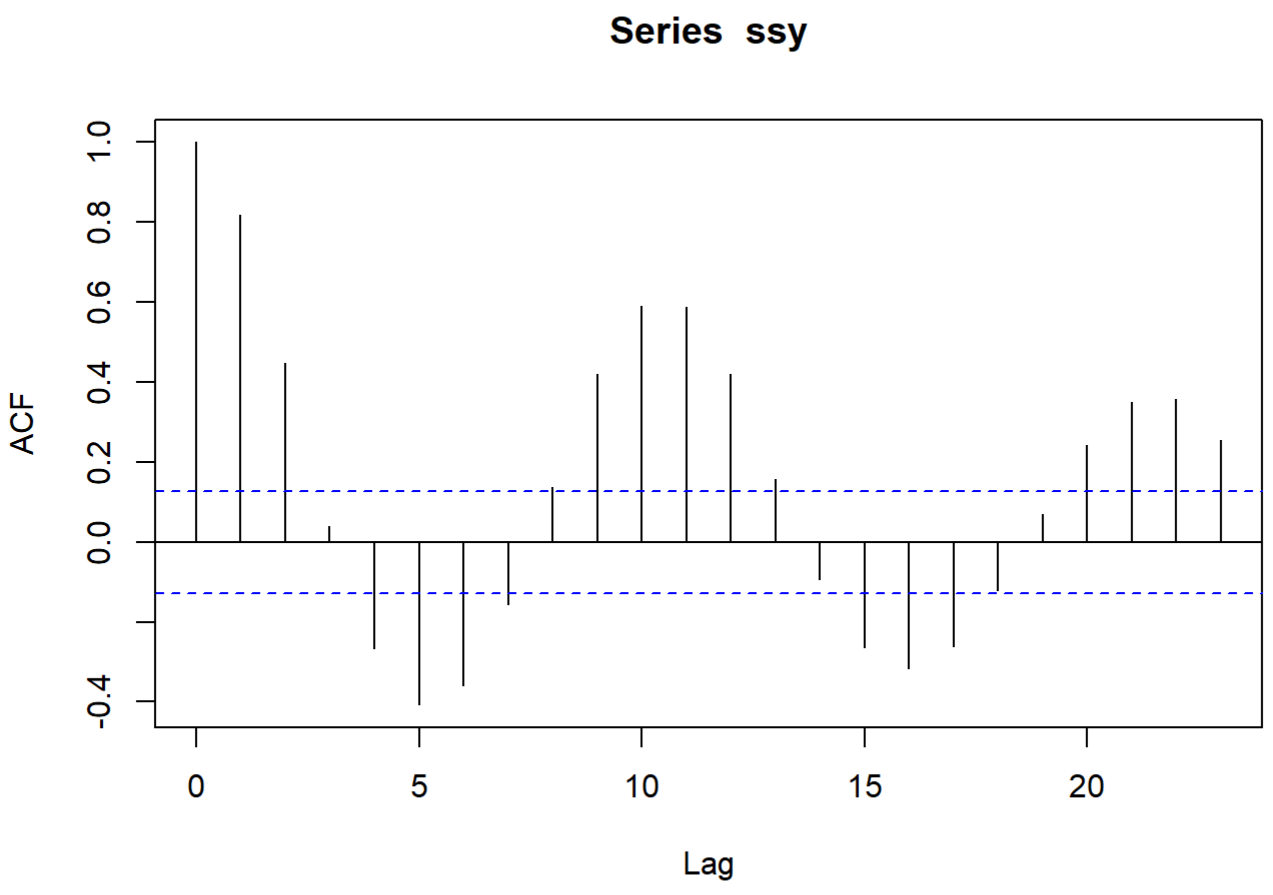


this however seems to attain too large a degree of variability, than what is ‘required’ for the data.

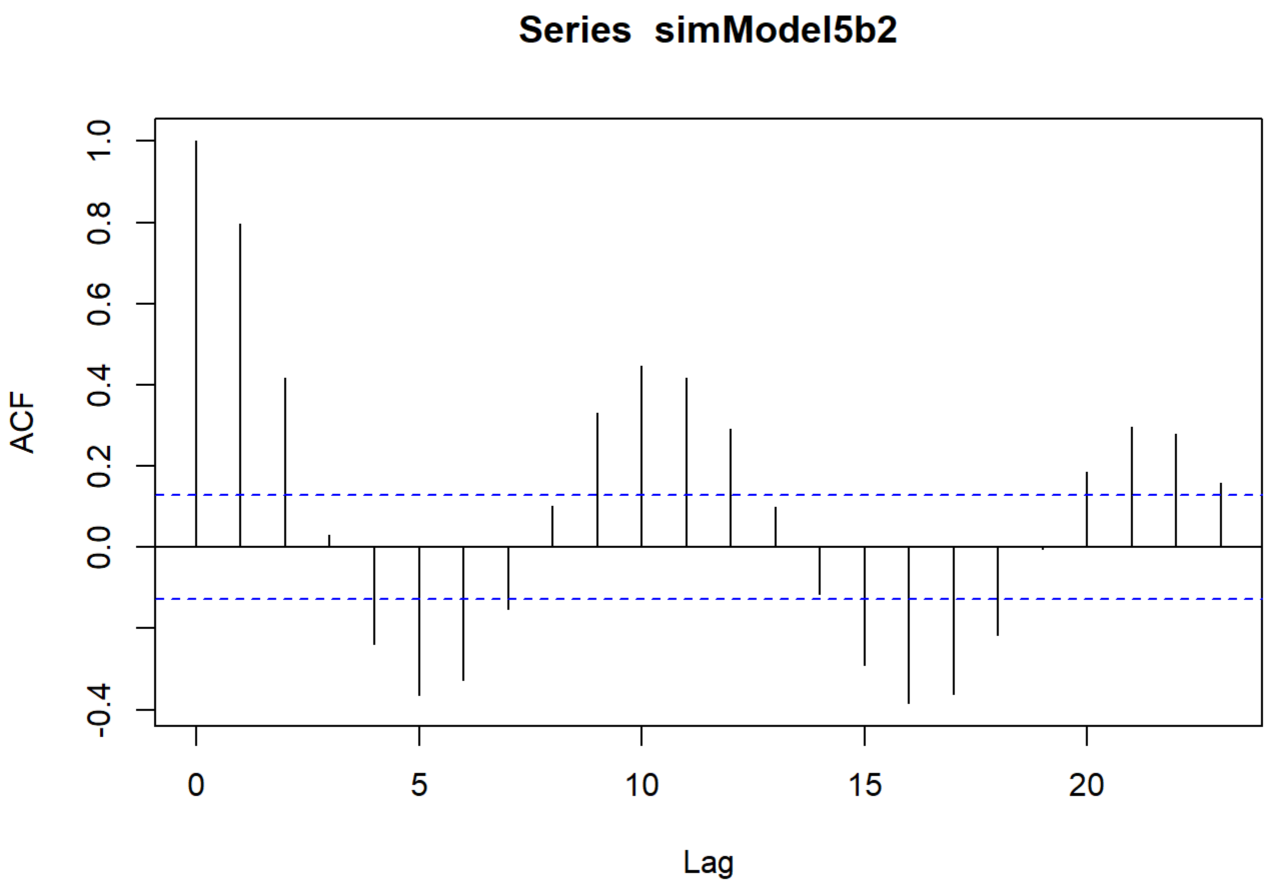
c)

We note that the simulated model attains the same kind of oscillatory autocorrelation behaviour as `ssy` , if not slightly less pronounced.

```r
acf(ssy)
```



```
acf(simModel5b2)
```


Series simModel5b2
ACF
Lag