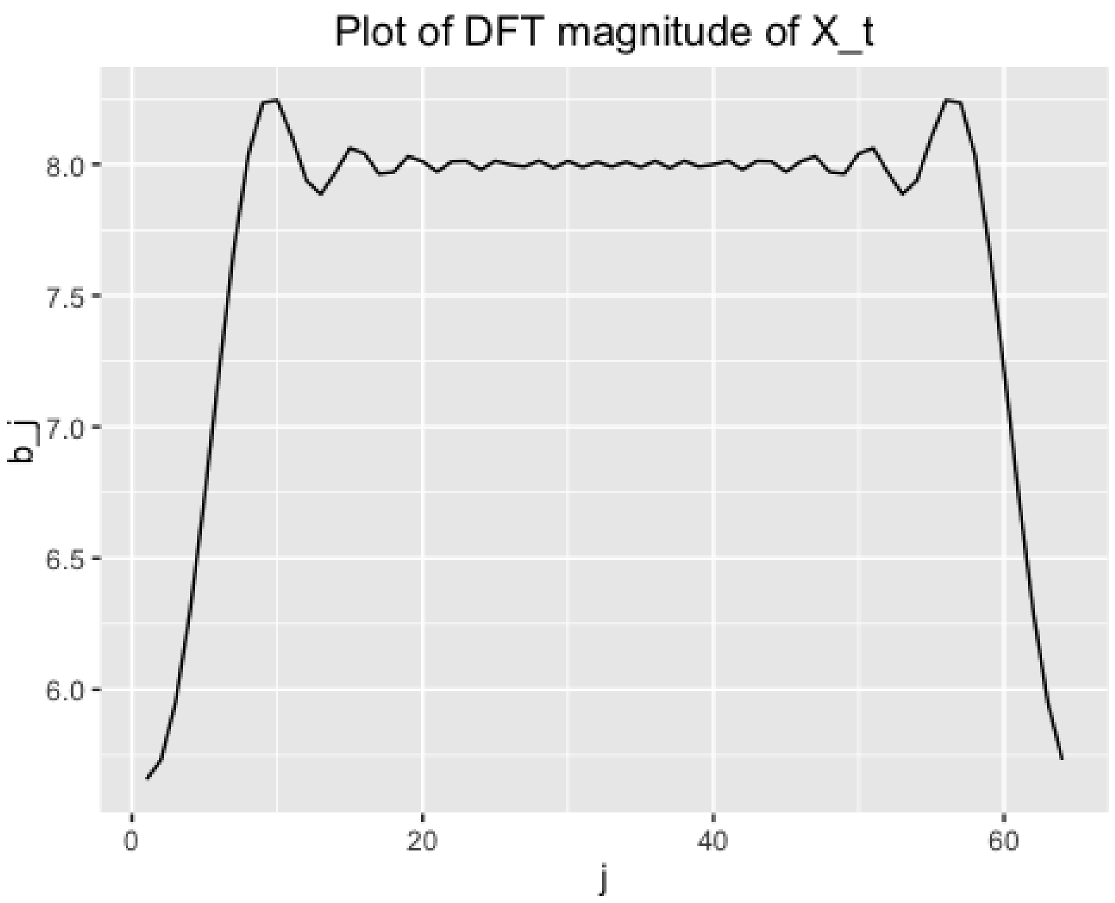
Hw6

2 (a) Consider the data xt = sin(πt2/256) for t = 0, 1, . . . , 127. Plot the data. Also plot the magnitude (absolute) of the DFT coefficients b1, . . . , b64. Comment on the two plots.

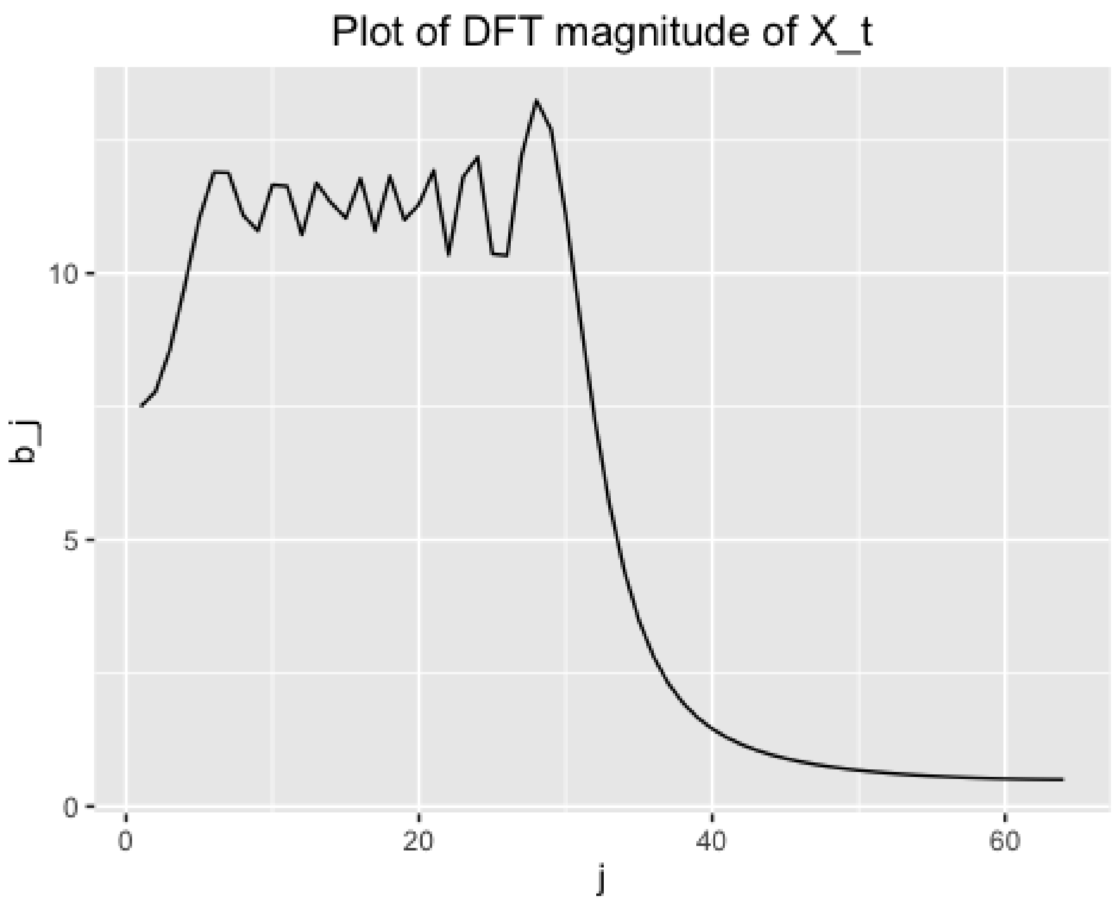
```
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.2.4

library(TSA)

## Loading required package: leaps

## Loading required package: locfit

## locfit 1.5-9.1    2013-03-22

## Loading required package: mgcv

## Loading required package: nlme

## This is mgcv 1.8-7. For overview type 'help("mgcv-package")'.

## Loading required package: tseries

## Warning: package 'tseries' was built under R version 3.2.5

##
## Attaching package: 'TSA'

## The following objects are masked from 'package:stats':
##
##     acf, arima

## The following object is masked from 'package:utils':
##
##     tar

library(astsa)

## Warning: package 'astsa' was built under R version 3.2.5

xs <- seq(0,127)
wave.1 <- sin((pi*xs^2)/256)
par(mfrow = c(1, 2))
plot(xs,wave.1,type="l",ylim=c(-1,1), xlab="time",ylab="X_t", main = "Plot of
X_t")
abline(h=0,lty=3)

data = data.frame(j = 1:64, b_j = abs(fft(wave.1))[1:64])
ggplot(data,aes(x=j,y=b_j)) + geom_line() + ggtitle('Plot of DFT magnitude of
X_t')
```



Comment: As time increases, the wave become more "compressed" since there are more spikes in the DFT magnitude. This is because the function cannot be expressed as a linear combination of sinusoids in Fourier frequency. The larger spikes corresponds to the lower-frequency stronger component; the smaller spikes corresponds to the higher-frequency weaker component.

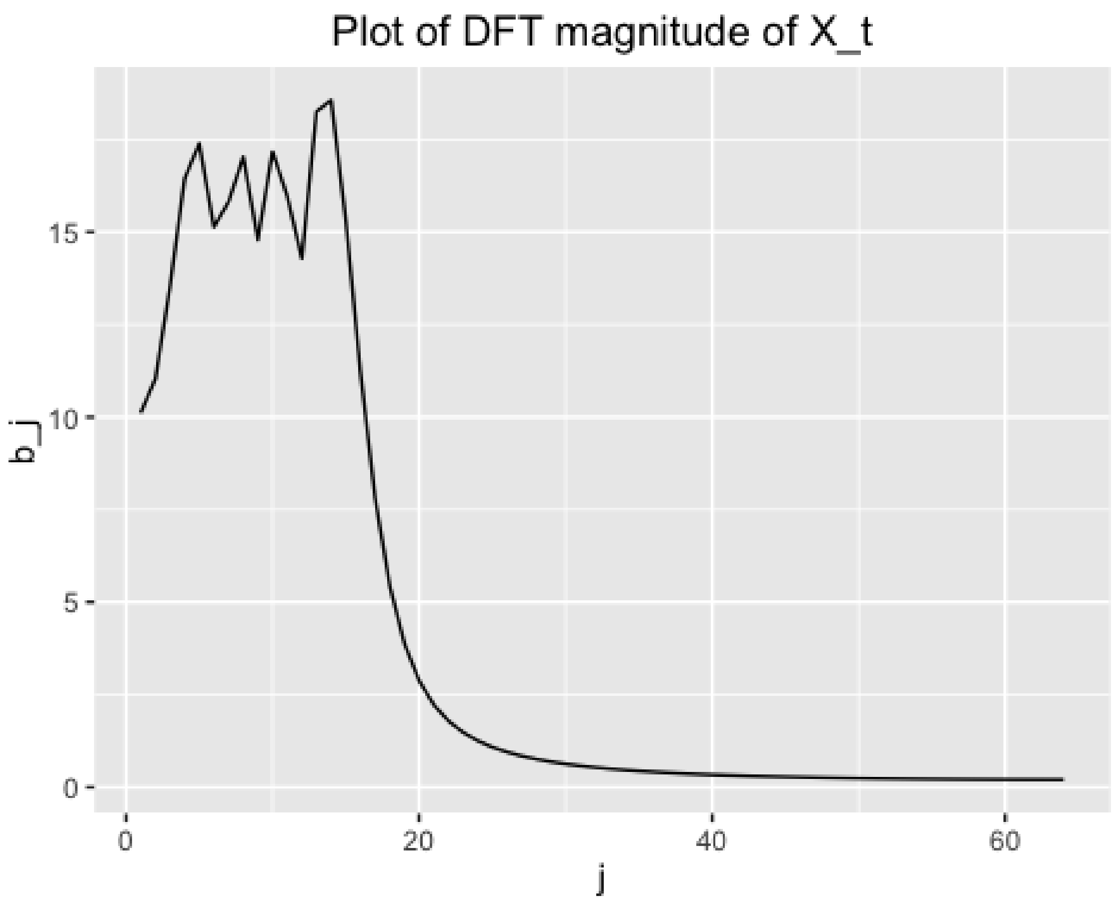
Case of 512

```
xs <- seq(0,127)
wave.2 <- sin((pi*xs^2)/512)
par(mfrow = c(1, 2))
plot(xs,wave.1,type="l",ylim=c(-1,1), xlab="time",ylab="X_t",main = "Plot of
X_t")
abline(h=0,lty=3)

data = data.frame(j = 1:64, b_j = abs(fft(wave.2))[1:64])
ggplot(data,aes(x=j,y=b_j)) + geom_line() + ggtitle('Plot of DFT magnitude of
X_t')
```



Case 1024

```
xs <- seq(0,127)
wave.3 <- sin((pi*xs^2)/1024)
par(mfrow = c(2,1))
plot(xs,wave.1,type="l",ylim=c(-1,1), xlab="time",ylab="X_t",main = "Plot of
X_t")
abline(h=0,lty=3)

data = data.frame(j = 1:64, b_j = abs(fft(wave.3))[1:64])
ggplot(data,aes(x=j,y=b_j)) + geom_line() + ggtitle('Plot of DFT magnitude of
X_t')
```

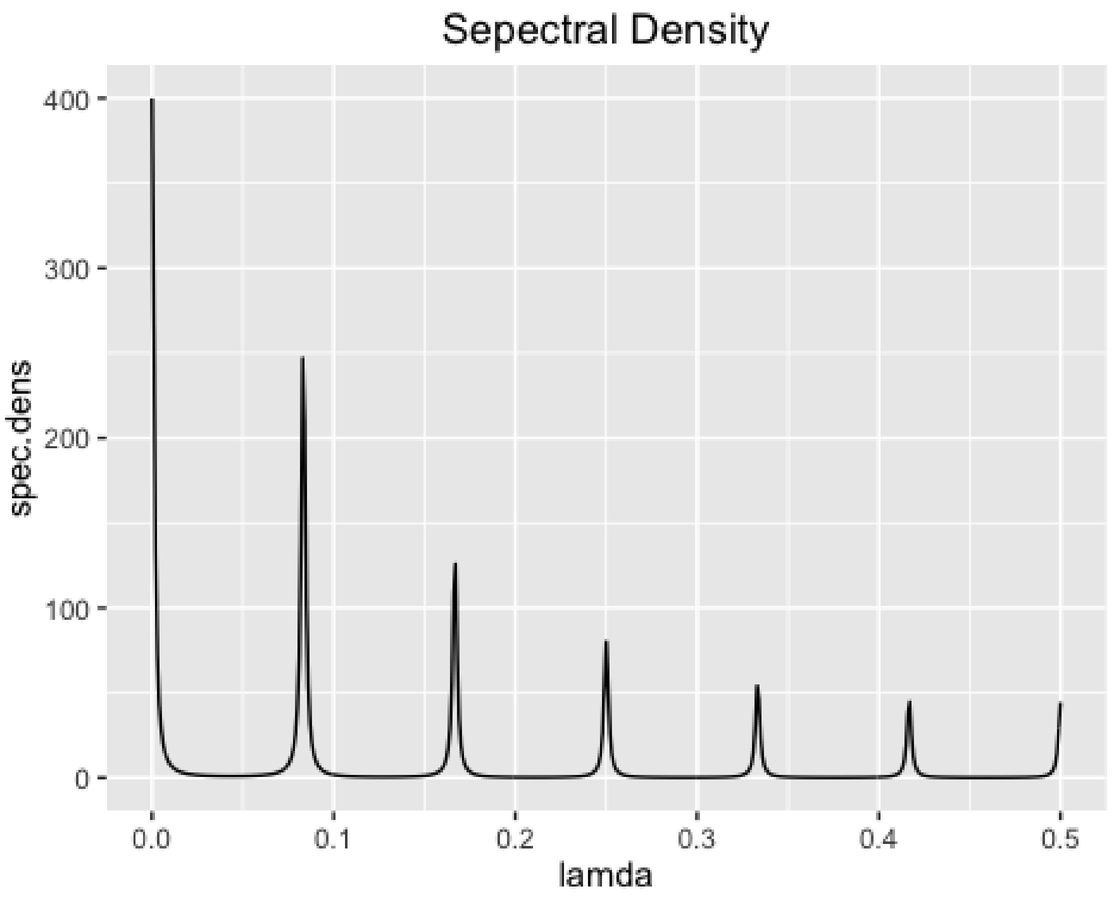


As the denominator increases, more weights are in lower frequency sinusoids. Higher frequencies are becoming less useful. The time series wavies are more compressed when t increases with smaller frequncy sinusods in this model.

Question 4 Consider the following seasonal AR model: (1 − φB)(1 − ΦBs)Xt = Zt, where {Zt} is white noise and |φ| < 1, |Φ| < 1. (b) Plot the spectral density for φ = 0.5, Φ = 0.9, σz^2 = 1 and s = 12

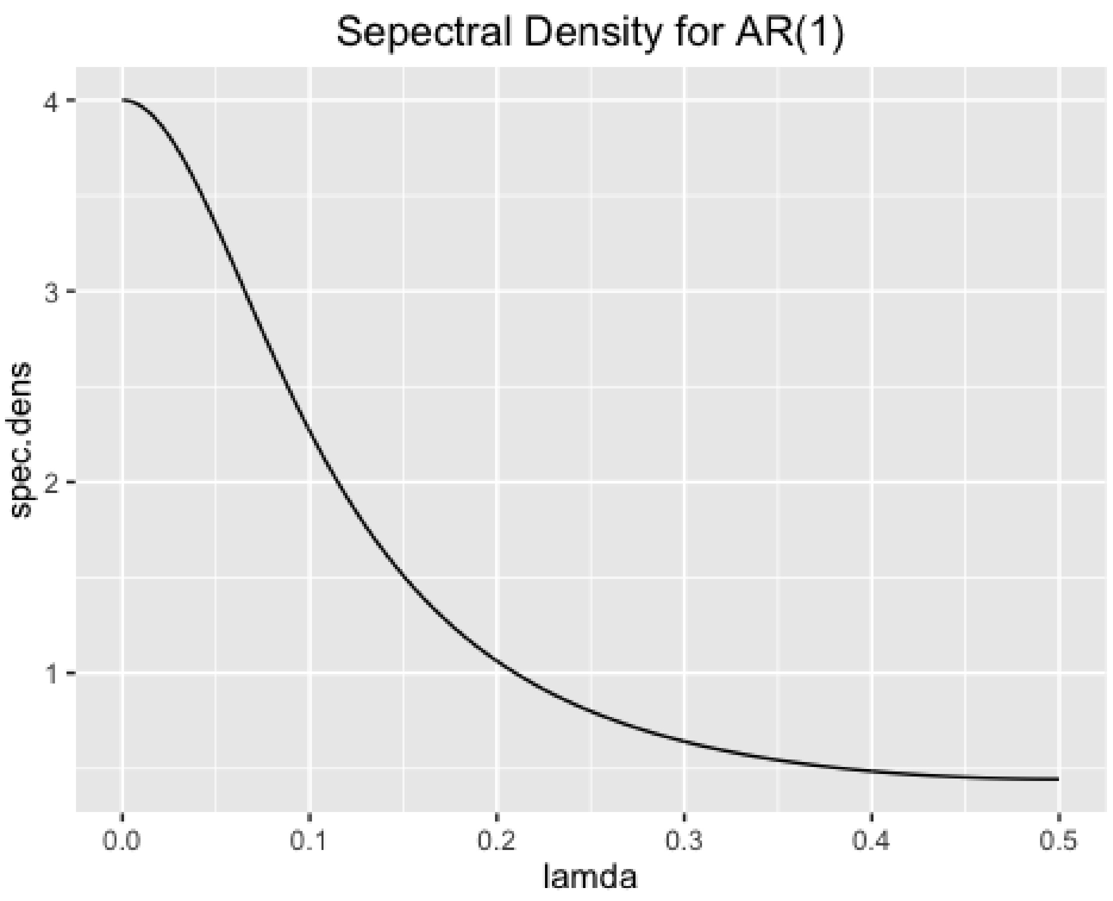
```
# Set seed for reproducibility
set.seed(5532)

# let sigma2 = 1
phi = 0.5
PHI = 0.9
sigma2 = 1
s = 12
scale = seq(0,500) / 1000

spec = function(lamda){
  spectral = ((sigma2^2) / (((1 + phi ^ 2 - 2 *phi*cos(2*pi*lamda)) *
(1+PHI^2-2*PHI*cos(2*pi*lamda*s)))))
}
```

```
data = data.frame(lamda = scale, spec.dens = spec(scale))
ggplot(data,aes(x=lamda,y=spec.dens)) + geom_line() + ggtitle('Sepectral
Density')
```



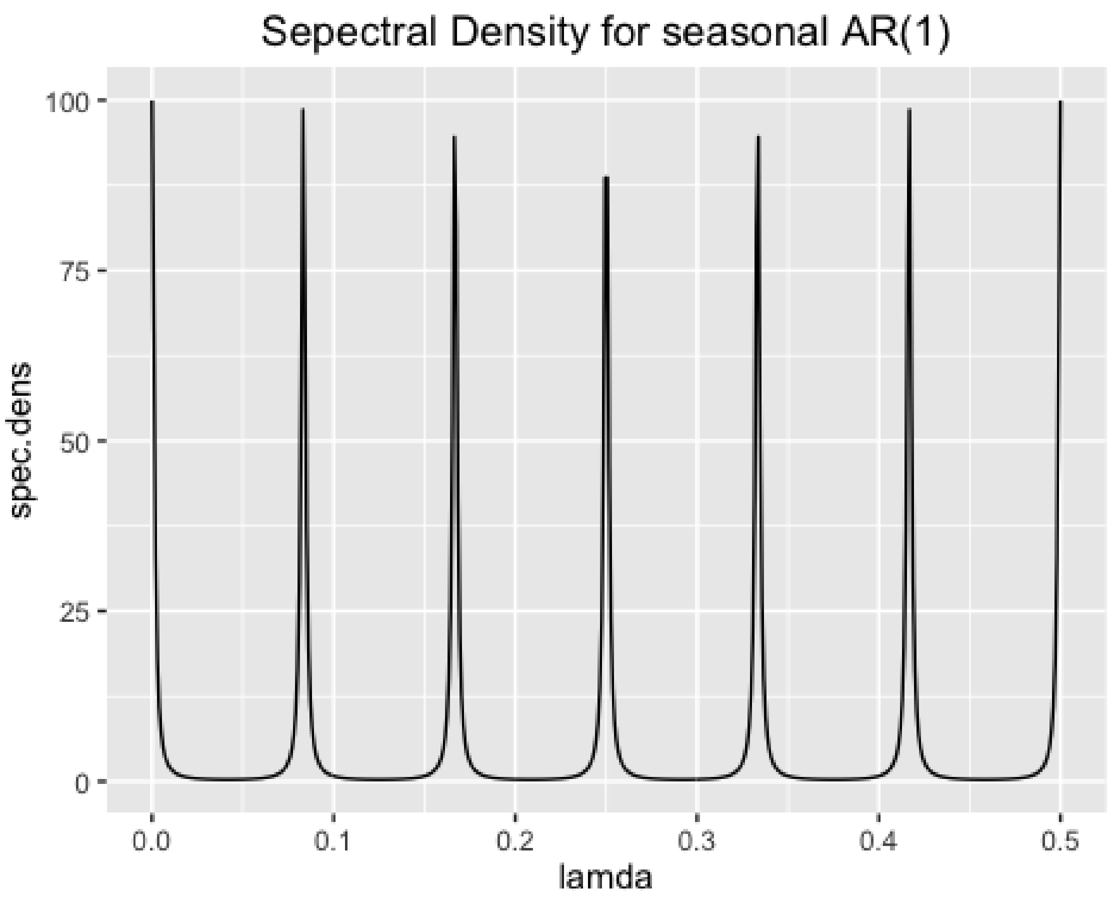
(c) Also plot the spectral density for the AR(1) process (1 − 0.5B)Xt = Zt and the seasonal AR(1) process (1 − 0.9B12)Xt = Zt.

```
# Set seed for reproducibility
set.seed(1234)

scale = seq(from = 0,to = 1/2, length = 500)

spec = function(lamda){
  spectral = (sigma2^2) / (1 + phi ^ 2 - 2 * phi * cos(2*pi*lamda))
}

data = data.frame(lamda = scale, spec.dens = spec(scale))
ggplot(data,aes(x=lamda,y=spec.dens)) + geom_line() + ggtitle('Sepectral
Density for AR(1)')
```



Seasonal AR(1) spectral density

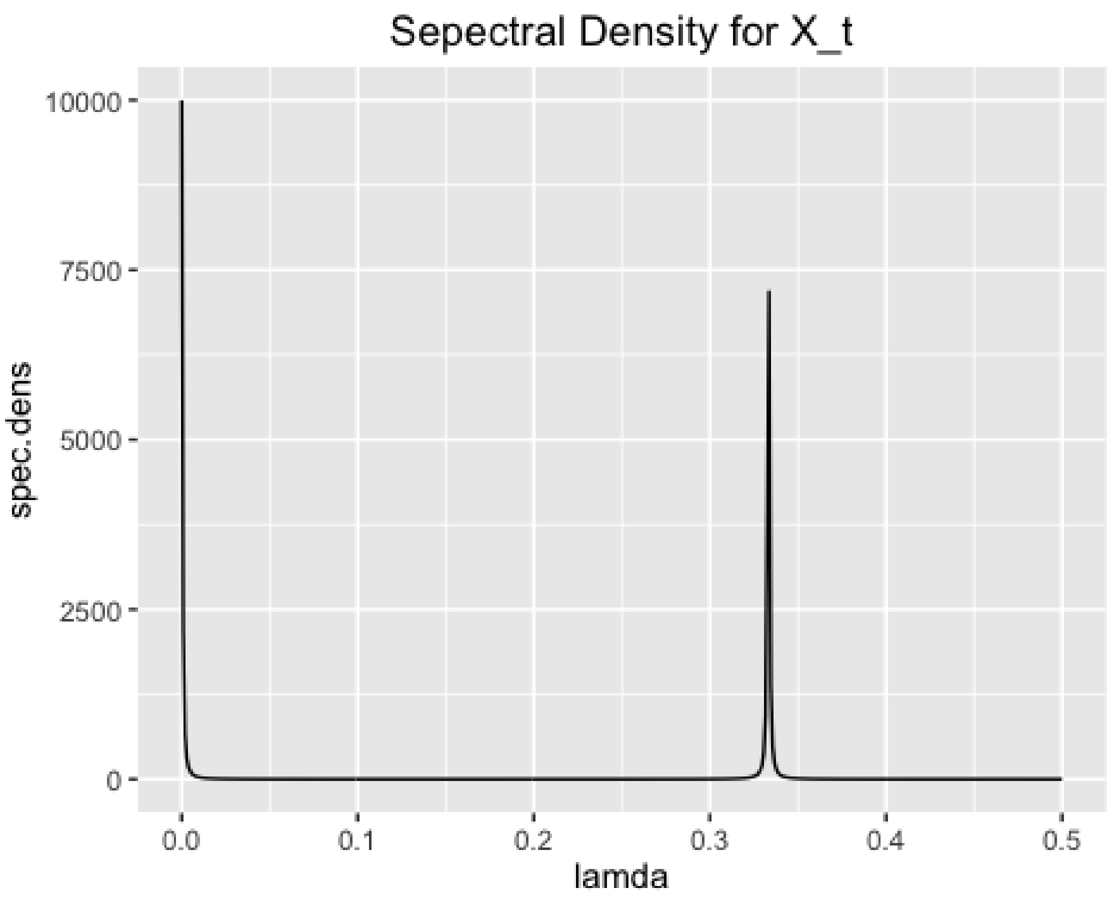
```
set.seed(1234)

spec = function(lamda){
  return ((sigma2^2) / (1 + PHI ^ 2 - 2 * PHI *cos(2*pi*lamda*s)))
}

data = data.frame(lamda = scale, spec.dens = spec(scale))
ggplot(data,aes(x=lamda,y=spec.dens)) + geom_line() + ggtitle('Sepectral
Density for seasonal AR(1)')
```



(d) Compare and comment on the different plots. AR(1) plot decreases smoothly from toward higher frequencies. Note that the spectral density of (a) is a combination of these two plots. We can see some periodic peaks in seasonal AR(1) as well as the spectral density of (a) but notice that the seasonal AR(1) has a higher periodic peaks.

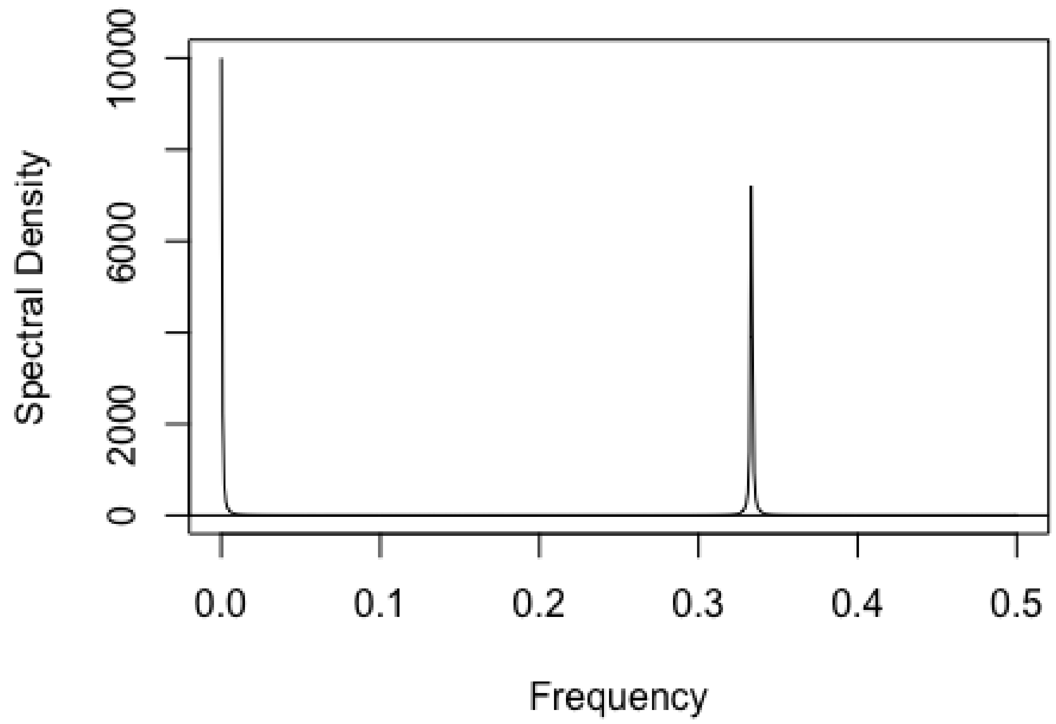
Question 6 Consider the stationary Autoregressive process: Xt − 0.99Xt−3 = Zt where {Zt} is white noise. (a) Compute and plot the spectral density of {Xt}.

```
# Let sigma2 = 1
# we found fx = 1/9 * (1+2cos(2*pi*lamda))^2
# let sigma2 = 1
sigma2 = 1

spec_fun = function(lamda){
  spectral = (sigma2)^2 / (1.9801 - 1.98 * cos(6*pi*lamda))
}

data = data.frame(lamda = scale, spec.dens = spec_fun(scale))
ggplot(data,aes(x=lamda,y=spec.dens)) + geom_line() + ggtitle('Sepectral
Density for X_t')
```



b) Does the spectral density suggest that the sample paths of {Xt} will exhibit approximately oscillatory behaviour? If yes, then with what period?

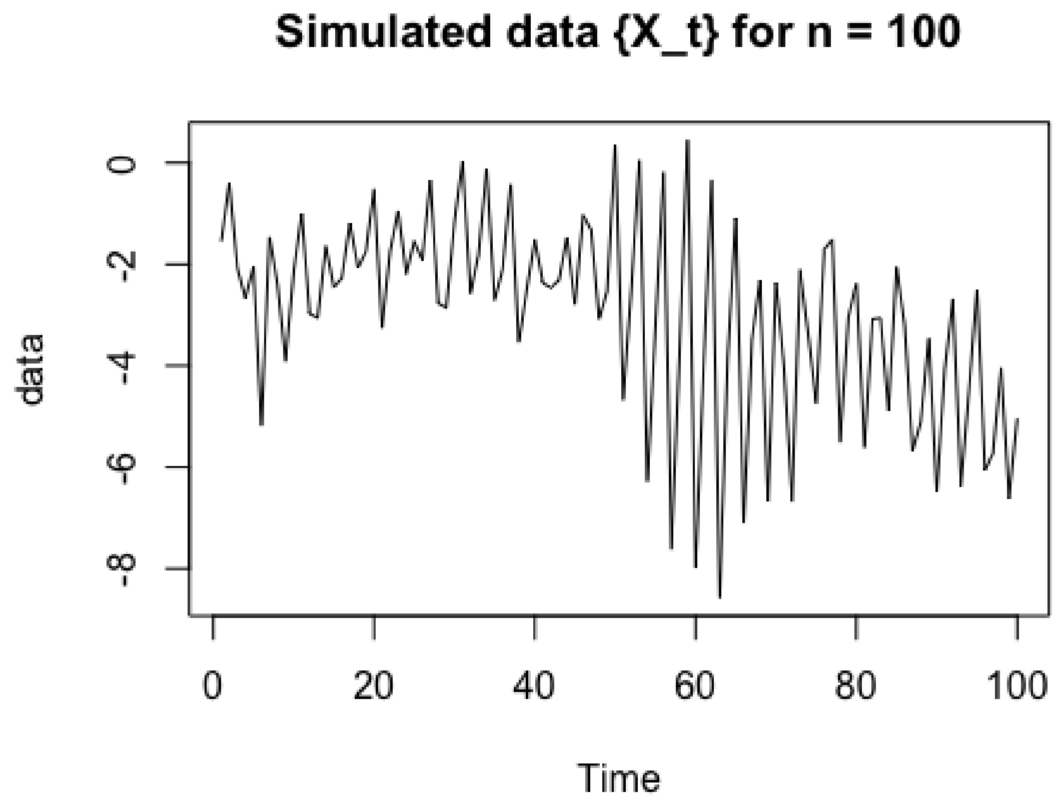
```
spec = ARMAspec(model = list(ar = c(0,0,0.99)))
```



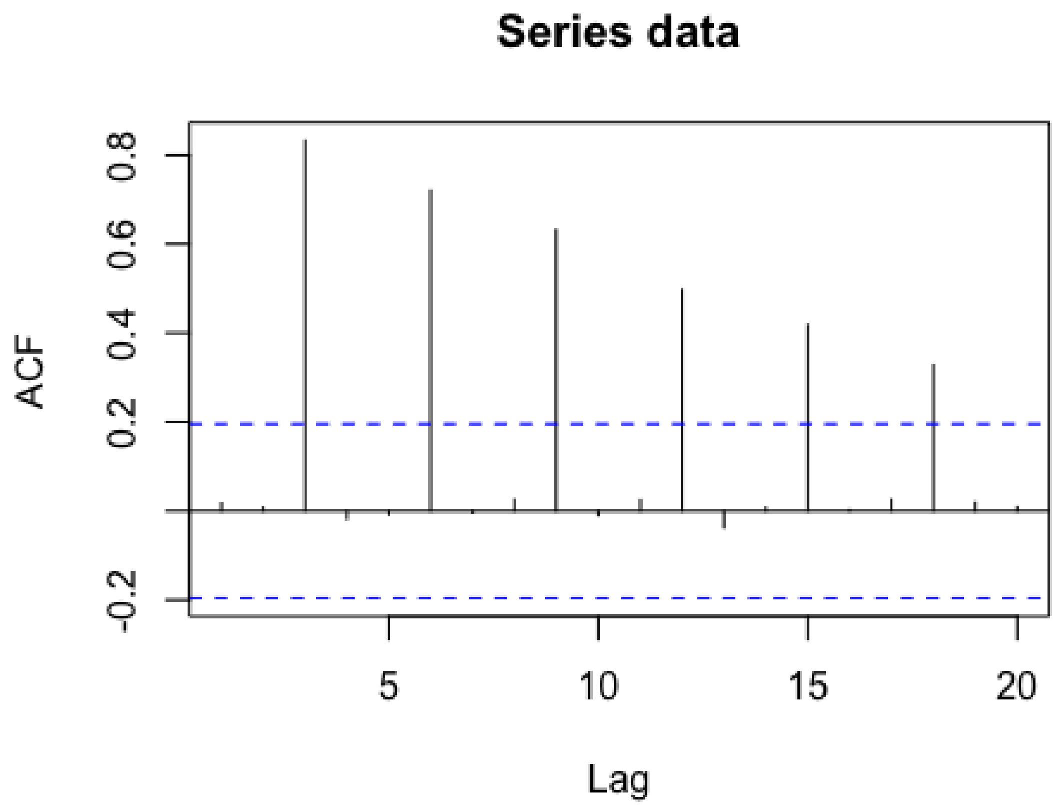
Yes, the spectral density suggest that the sample paths of {Xt} will exhibit approximately oscillatory behaviour. It oscillates with period 3.

(c) Simulate a sample of size 100 from this model. Plot the simulated data. Does this plot support the conclusion of part (b)?

```
# Recall AR(3) model
set.seed(12345) # for reproducibility
data = arima.sim(list(ar = c(0,0,0.99)), n = 100)
plot(data, ylab = "data", main = "Simulated data {X_t} for n = 100")
```

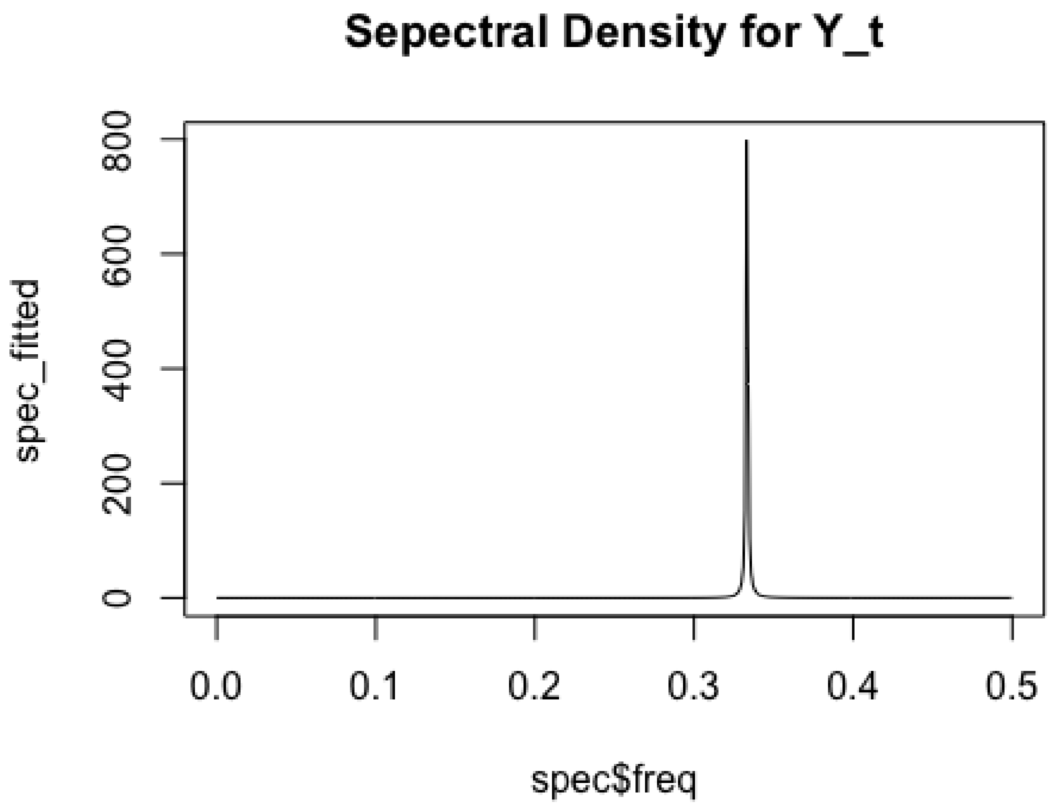


```r
acf(data)
```



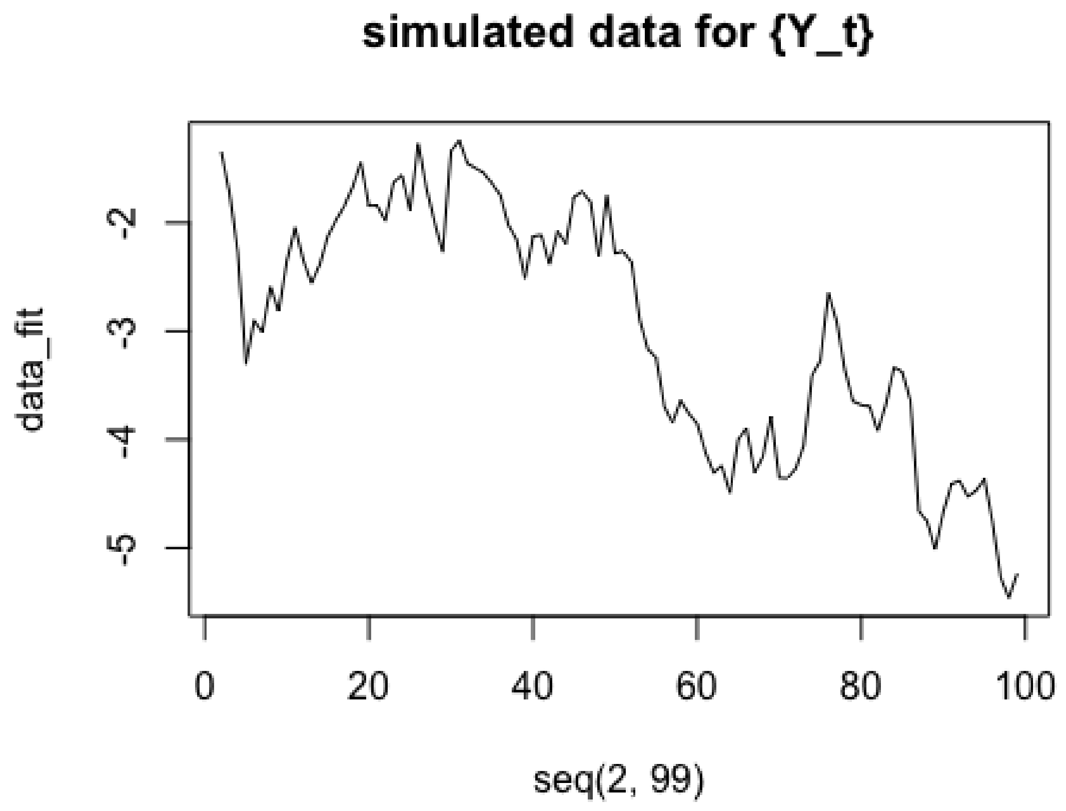
(d) Compute the spectral density of the filtered process: Yt = (Xt−1 + Xt + Xt+1)/3 (1) How does the spectral density of {Yt} compare to that of {Xt}?

```
# Compute Explicitly
spec_fitted_fun = function(lamda){
  #spectral = ((2 * sin(3*pi*lamda)/sin(pi*lamda)) * cos(pi*lamda) - 1)/ 3
  spectral = (1 + 2 * cos(2*pi*lamda))^2 / 9
}

spec_fitted = spec$spec * (abs(spec$freq)^2)
plot(x=spec$freq,y=spec_fitted, type = "l", main = 'Sepectral Density for
Y_t')
```

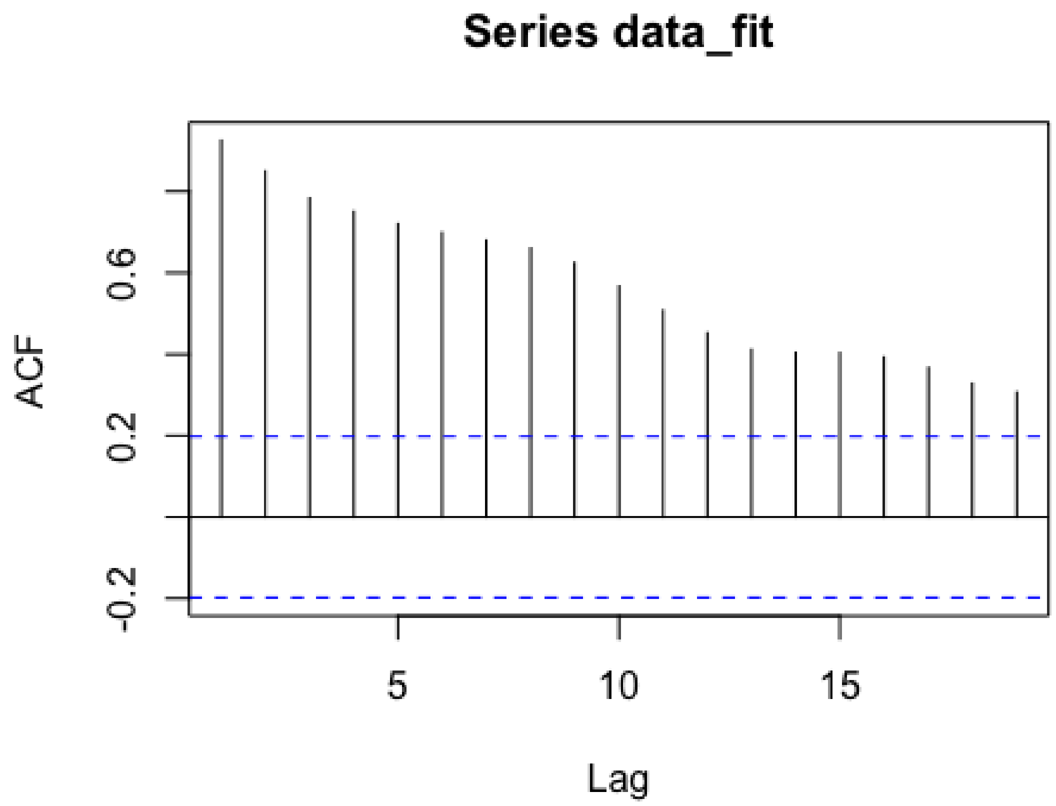


(e) From the simulated sample from {Xt} in part (c), perform the averaging as in (1) to obtain a simulated sample from {Yt}. Plot this sample. Does this plot support the spectral density plot in part (d)?

```
data_fit = numeric(98)
for(i in seq(2,99)){
    data_fit[i-1] = (data[i-1] + data[i] + data[i+1])/3
}
plot(x = seq(2,99), y = data_fit, type = "l", main = "simulated data for
{Y_t}")
```

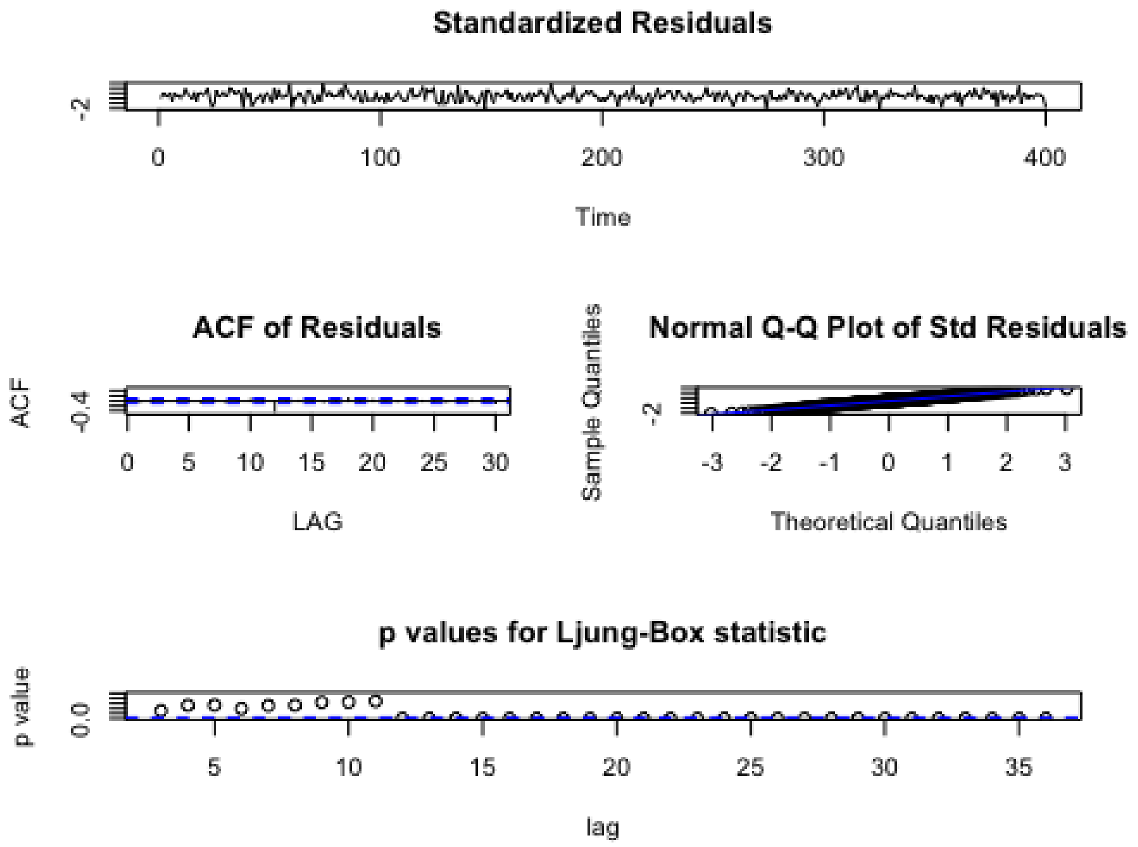


```r
acf(data_fit)
```



7. Without using the arima.sim() function in R, simulate n = 400 observations from the multiplicative seasonal ARMA model given by the difference equation: (1 − 0.5B)(1 − 0.7B 12)Xt = Zt where {Zt} is white noise. Plot the sample autocorrelation function of the simulated observations and compare it with the true acf of the process.

```
set.seed(12345)
v = rnorm(400,1,1)  # v contains 100 iid N(1,1) variates
x = cumsum(v)       # x is a random walk with drift = 1
sarima(x,1,0,0,1,0,0,12,  details=FALSE)

## Warning in sqrt(diag(fitit$var.coef)): NaNs produced

## Warning in sqrt(diag(fitit$var.coef)): NaNs produced
```

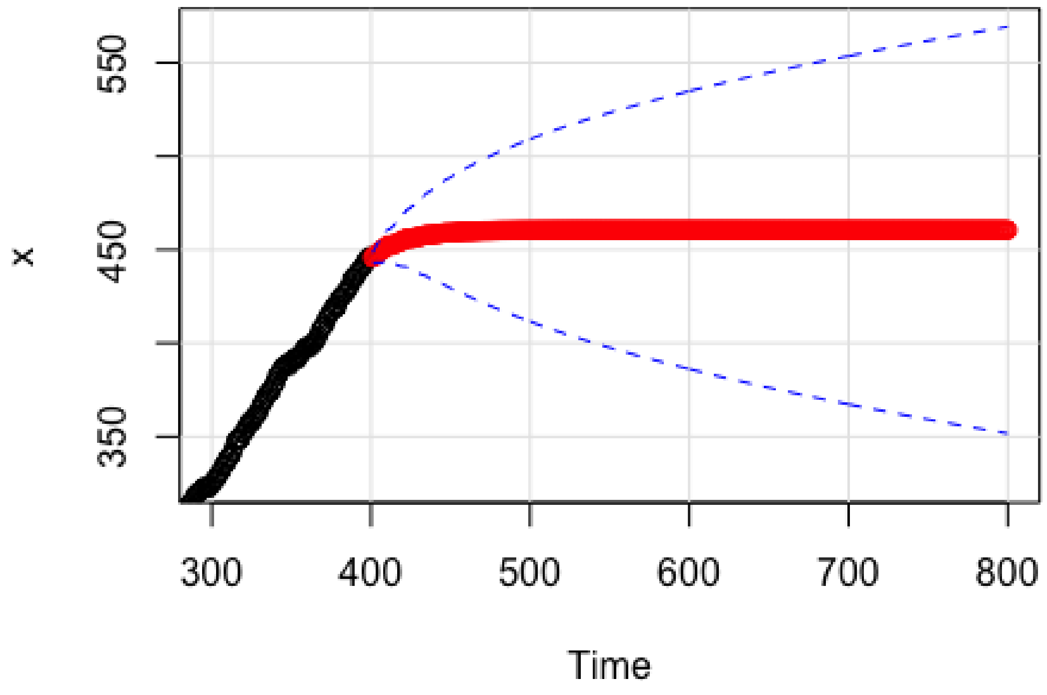


```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P,
D,
##     Q), period = S), xreg = xmean, include.mean = FALSE, optim.control =
list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
```

```
## Warning in sqrt(diag(x$var.coef)): NaNs produced
```

```
##       ar1    sar1     xmean
##         1  0.5573  226.1474
## s.e.    0  0.0410       NaN
##
## sigma^2 estimated as 1.546:  log likelihood = -657.4,  aic = 1322.8
##
## $degrees_of_freedom
## [1] 397
##
## $ttable
##        Estimate    SE      t.value p.value
## ar1      1.0000 0.000 5843769.697       0
## sar1     0.5573 0.041      13.577       0
## xmean 226.1474   NaN         NaN     NaN
##
## $AIC
## [1] 1.450428
##
## $AICc
## [1] 1.455681
##
## $BIC
## [1] 0.4803637
```

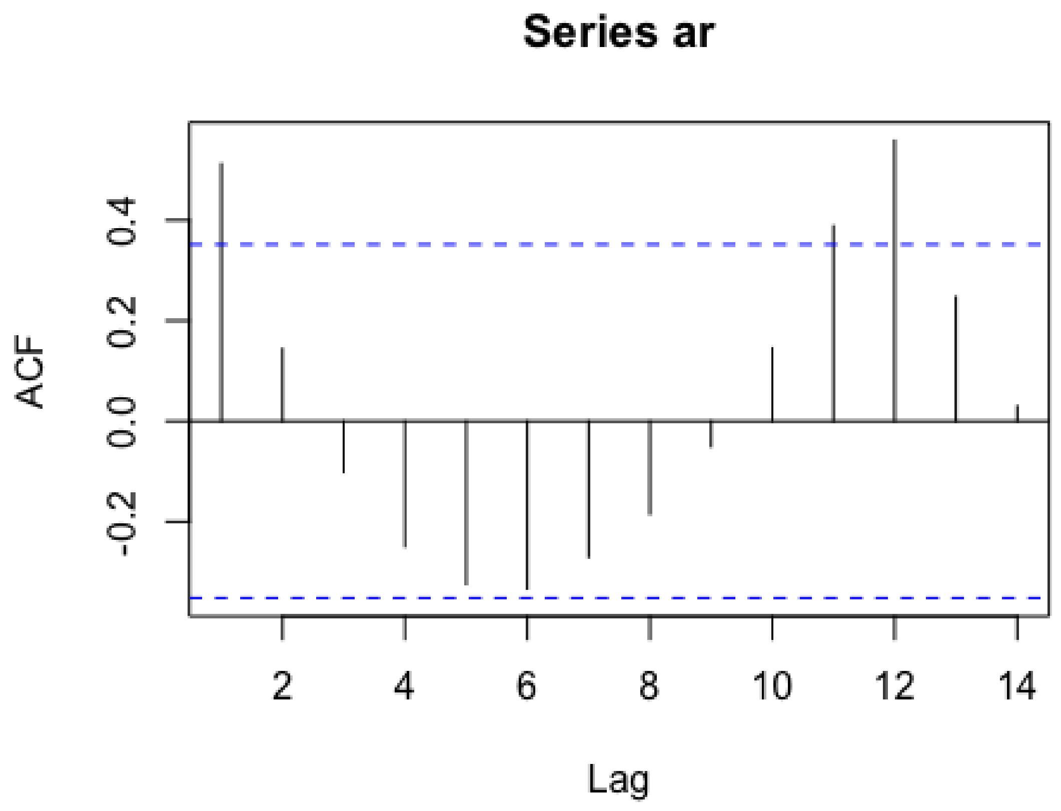
```
# Now, forecast
sarima.for(x,400,1,0,0,1,0,0,12) #<-- it's an ARIMA(1,0,0)x(1,0,0)12
```



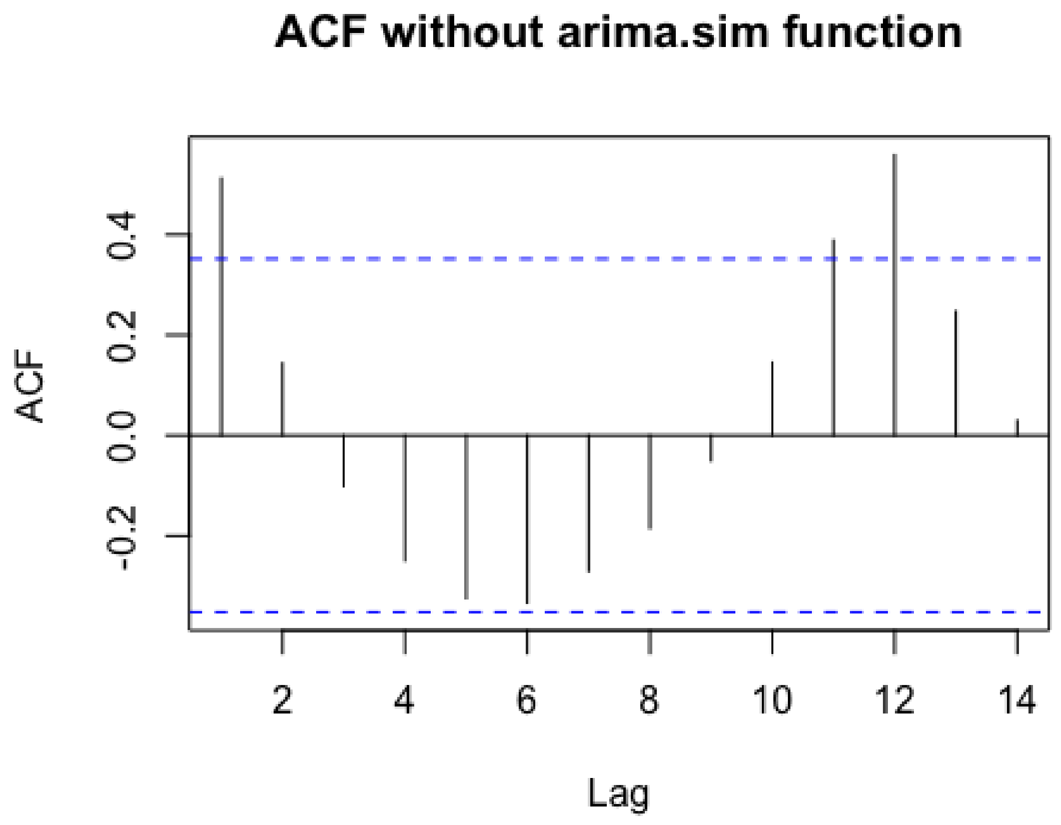
```
## $pred
## Time Series:
## Start = 401
## End = 800
## Frequency = 1
##   [1] 445.8407 446.1648 447.1669 448.0018 448.6257 448.9783 449.6818
##   [8] 450.3854 451.2998 452.0282 452.5670 452.3582 452.2976 452.4781
##  [15] 453.0366 453.5019 453.8495 454.0461 454.4381 454.8302 455.3397
##  [22] 455.7457 456.0459 455.9295 455.8958 455.9964 456.3076 456.5669
##  [29] 456.7606 456.8702 457.0886 457.3071 457.5911 457.8173 457.9846
##  [36] 457.9198 457.9010 457.9570 458.1305 458.2749 458.3829 458.4439
##  [43] 458.5657 458.6875 458.8457 458.9718 459.0650 459.0289 459.0184
##  [50] 459.0496 459.1463 459.2268 459.2870 459.3210 459.3888 459.4567
##  [57] 459.5448 459.6151 459.6671 459.6469 459.6411 459.6585 459.7123
##  [64] 459.7572 459.7907 459.8097 459.8475 459.8853 459.9344 459.9736
##  [71] 460.0025 459.9913 459.9881 459.9978 460.0278 460.0528 460.0715
##  [78] 460.0820 460.1031 460.1242 460.1515 460.1734 460.1895 460.1832
##  [85] 460.1814 460.1868 460.2035 460.2175 460.2279 460.2338 460.2455
##  [92] 460.2573 460.2725 460.2847 460.2937 460.2902 460.2892 460.2922
##  [99] 460.3015 460.3093 460.3151 460.3183 460.3249 460.3314 460.3399
## [106] 460.3467 460.3517 460.3498 460.3492 460.3509 460.3561 460.3604
## [113] 460.3636 460.3654 460.3691 460.3727 460.3775 460.3812 460.3840
## [120] 460.3829 460.3826 460.3836 460.3864 460.3889 460.3907 460.3917
## [127] 460.3937 460.3957 460.3984 460.4005 460.4020 460.4014 460.4012
## [134] 460.4018 460.4034 460.4047 460.4057 460.4063 460.4074 460.4085
## [141] 460.4100 460.4112 460.4120 460.4117 460.4116 460.4119 460.4128
## [148] 460.4135 460.4141 460.4144 460.4150 460.4157 460.4165 460.4171
## [155] 460.4176 460.4174 460.4174 460.4175 460.4180 460.4184 460.4187
## [162] 460.4189 460.4193 460.4196 460.4201 460.4204 460.4207 460.4206
## [169] 460.4206 460.4206 460.4209 460.4212 460.4213 460.4214 460.4216
## [176] 460.4218 460.4221 460.4223 460.4224 460.4223 460.4223 460.4224
## [183] 460.4225 460.4227 460.4227 460.4228 460.4229 460.4230 460.4232
## [190] 460.4233 460.4233 460.4233 460.4233 460.4233 460.4234 460.4235
## [197] 460.4235 460.4236 460.4236 460.4237 460.4238 460.4238 460.4239
## [204] 460.4238 460.4238 460.4238 460.4239 460.4239 460.4239 460.4240
## [211] 460.4240 460.4240 460.4241 460.4241 460.4241 460.4241 460.4241
## [218] 460.4241 460.4241 460.4242 460.4242 460.4242 460.4242 460.4242
## [225] 460.4242 460.4242 460.4243 460.4243 460.4242 460.4242 460.4243
## [232] 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243
## [239] 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243
## [246] 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243
## [253] 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243
## [260] 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243
## [267] 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243
## [274] 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243
## [281] 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243
## [288] 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243
## [295] 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243
## [302] 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243 460.4243
## [309] 460.4243 460.4242 460.4242 460.4242 460.4242 460.4242 460.4242
## [316] 460.4242 460.4242 460.4242 460.4242 460.4242 460.4242 460.4242
## [323] 460.4242 460.4242 460.4242 460.4242 460.4242 460.4242 460.4242
## [330] 460.4242 460.4242 460.4242 460.4242 460.4242 460.4242 460.4242
## [337] 460.4242 460.4242 460.4242 460.4242 460.4242 460.4242 460.4242
## [344] 460.4242 460.4242 460.4242 460.4242 460.4242 460.4241 460.4241
## [351] 460.4241 460.4241 460.4241 460.4241 460.4241 460.4241 460.4241
## [358] 460.4241 460.4241 460.4241 460.4241 460.4241 460.4241 460.4241
## [365] 460.4241 460.4241 460.4241 460.4241 460.4241 460.4241 460.4241
## [372] 460.4241 460.4241 460.4241 460.4241 460.4241 460.4241 460.4241
## [379] 460.4241 460.4241 460.4241 460.4241 460.4241 460.4241 460.4241
## [386] 460.4241 460.4240 460.4240 460.4240 460.4240 460.4240 460.4240
## [393] 460.4240 460.4240 460.4240 460.4240 460.4240 460.4240 460.4240
## [400] 460.4240
##
## $se
## Time Series:
## Start = 401
## End = 800
## Frequency = 1
##   [1]  1.243231  1.758195  2.153340  2.486463  2.779950  3.045282
3.289281
##   [8]  3.516389  3.729694  3.931442  4.123331  4.306679  4.721841
5.103340
##  [15]  5.458239  5.791430  6.106468  6.406032  6.692200  6.966623
7.230638
##  [22]  7.485347  7.731670  7.970383  8.301768  8.620423  8.927712
9.224770
##  [29]  9.512556  9.791887 10.063468 10.327910 10.585749 10.837454
11.083445
##  [36] 11.324093 11.604869 11.879009 12.146965 12.409135 12.665880
12.917523
##  [43] 13.164357 13.406647 13.644635 13.878543 14.108573 14.334913
14.579110
##  [50] 14.819283 15.055626 15.288315 15.517516 15.743380 15.966049
16.185655
##  [57] 16.402322 16.616163 16.827287 17.035794 17.252205 17.465934
17.677079
##  [64] 17.885732 18.091978 18.295900 18.497574 18.697072 18.894464
19.089815
##  [71] 19.283188 19.474640 19.669421 19.862292 20.053308 20.242522
20.429983
##  [78] 20.615740 20.799838 20.982321 21.163230 21.342606 21.520487
21.696910
##  [85] 21.874538 22.050737 22.225538 22.398975 22.571079 22.741881
22.911410
##  [92] 23.079694 23.246759 23.412633 23.577339 23.740903 23.904696
24.067374
##  [99] 24.228960 24.389476 24.548942 24.707379 24.864806 25.021243
25.176708
## [106] 25.331219 25.484793 25.637447 25.789897 25.941450 26.092123
26.241931
## [113] 26.390889 26.539011 26.686310 26.832801 26.978497 27.123410
27.267552
## [120] 27.410937 27.553941 27.696206 27.837745 27.978567 28.118684
28.258107
## [127] 28.396844 28.534908 28.672306 28.809049 28.945147 29.080607
```

```
29.215631
## [134] 29.350034 29.483825 29.617011 29.749601 29.881602 30.013023
30.143871
## [141] 30.274154 30.403878 30.533051 30.661680 30.789873 30.917535
31.044671
## [148] 31.171289 31.297395 31.422995 31.548095 31.672700 31.796818
31.920452
## [155] 32.043610 32.166296 32.288570 32.410383 32.531739 32.652645
32.773104
## [162] 32.893123 33.012705 33.131855 33.250579 33.368880 33.486763
33.604233
## [169] 33.721322 33.838006 33.954289 34.070175 34.185668 34.300773
34.415492
## [176] 34.529831 34.643792 34.757379 34.870597 34.983447 35.095951
35.208095
## [183] 35.319883 35.431318 35.542404 35.653144 35.763541 35.873598
35.983318
## [190] 36.092705 36.201762 36.310491 36.418903 36.526994 36.634766
36.742222
## [197] 36.849364 36.956196 37.062720 37.168938 37.274854 37.380470
37.485788
## [204] 37.590811 37.695546 37.799991 37.904148 38.008019 38.111608
38.214916
## [211] 38.317945 38.420698 38.523176 38.625383 38.727320 38.828990
38.930396
## [218] 39.031539 39.132421 39.233043 39.333408 39.433517 39.533373
39.632977
## [225] 39.732331 39.831438 39.930299 40.028915 40.127291 40.225425
40.323321
## [232] 40.420980 40.518404 40.615594 40.712552 40.809279 40.905778
41.002050
## [239] 41.098096 41.193918 41.289518 41.384898 41.480058 41.575001
41.669727
## [246] 41.764239 41.858537 41.952623 42.046498 42.140164 42.233623
42.326875
## [253] 42.419923 42.512767 42.605408 42.697849 42.790090 42.882133
42.973978
## [260] 43.065628 43.157082 43.248344 43.339413 43.430292 43.520980
43.611481
## [267] 43.701793 43.791920 43.881861 43.971619 44.061193 44.150586
44.239798
## [274] 44.328831 44.417685 44.506362 44.594863 44.683188 44.771339
44.859317
## [281] 44.947123 45.034757 45.122221 45.209516 45.296643 45.383603
45.470396
## [288] 45.557024 45.643487 45.729787 45.815925 45.901901 45.987716
46.073371
## [295] 46.158867 46.244206 46.329387 46.414411 46.499281 46.583995
46.668556
## [302] 46.752964 46.837220 46.921324 47.005278 47.089083 47.172738
```

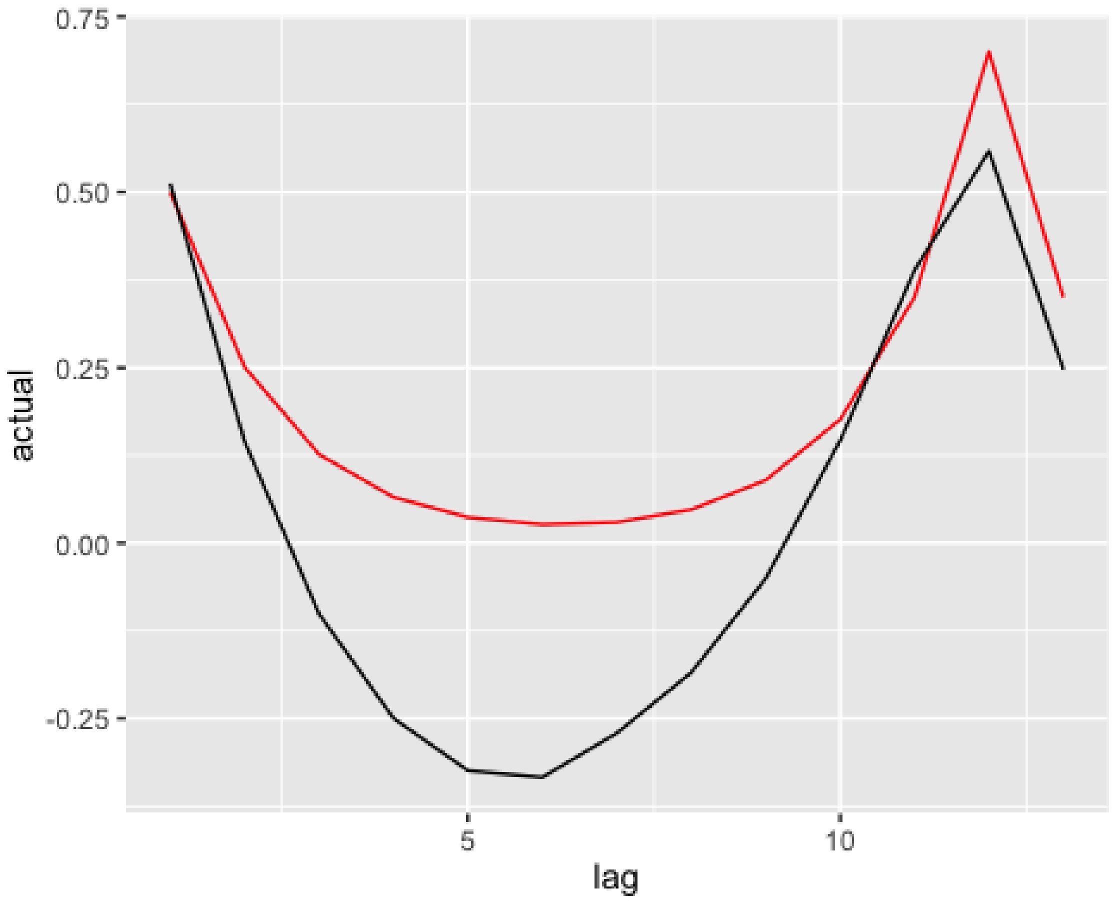
```
47.256246
## [309] 47.339606 47.422819 47.505887 47.588810 47.671588 47.754223
47.836716
## [316] 47.919066 48.001275 48.083343 48.165272 48.247062 48.328713
48.410226
## [323] 48.491602 48.572842 48.653947 48.734916 48.815751 48.896453
48.977021
## [330] 49.057457 49.137762 49.217935 49.297978 49.377892 49.457676
49.537331
## [337] 49.616859 49.696260 49.775534 49.854681 49.933704 50.012601
50.091374
## [344] 50.170024 50.248550 50.326954 50.405236 50.483397 50.561437
50.639356
## [351] 50.717156 50.794836 50.872398 50.949842 51.027169 51.104378
51.181471
## [358] 51.258448 51.335309 51.412056 51.488688 51.565207 51.641612
51.717904
## [365] 51.794083 51.870151 51.946107 52.021953 52.097688 52.173313
52.248829
## [372] 52.324236 52.399534 52.474724 52.549806 52.624782 52.699651
52.774413
## [379] 52.849070 52.923621 52.998068 53.072410 53.146648 53.220783
53.294814
## [386] 53.368743 53.442570 53.516294 53.589917 53.663440 53.736861
53.810183
## [393] 53.883404 53.956527 54.029550 54.102475 54.175302 54.248030
54.320662
## [400] 54.393196

# A seasonal AR(1) model has significant lags at 1, 12 and 13, with an
increasing trend before lag 12, sharply cut off at lag 13.
ar<-ARMAacf(ar = c(.5,0,0,0,0,0,0,0,0,0,0,.7,-.35),lag.max=30)
acf_data = acf(ar)
```



```r
plot(acf_data, main = "ACF without arima.sim function")
```



```
data3 = data.frame(lag = 1:13, actual = ar[2:14], simulated =
acf_data$acf[1:13])
ggplot(data3, aes(x = lag)) + geom_line(aes(y = actual), col = "red") +
geom_line(aes(y = simulated))
```



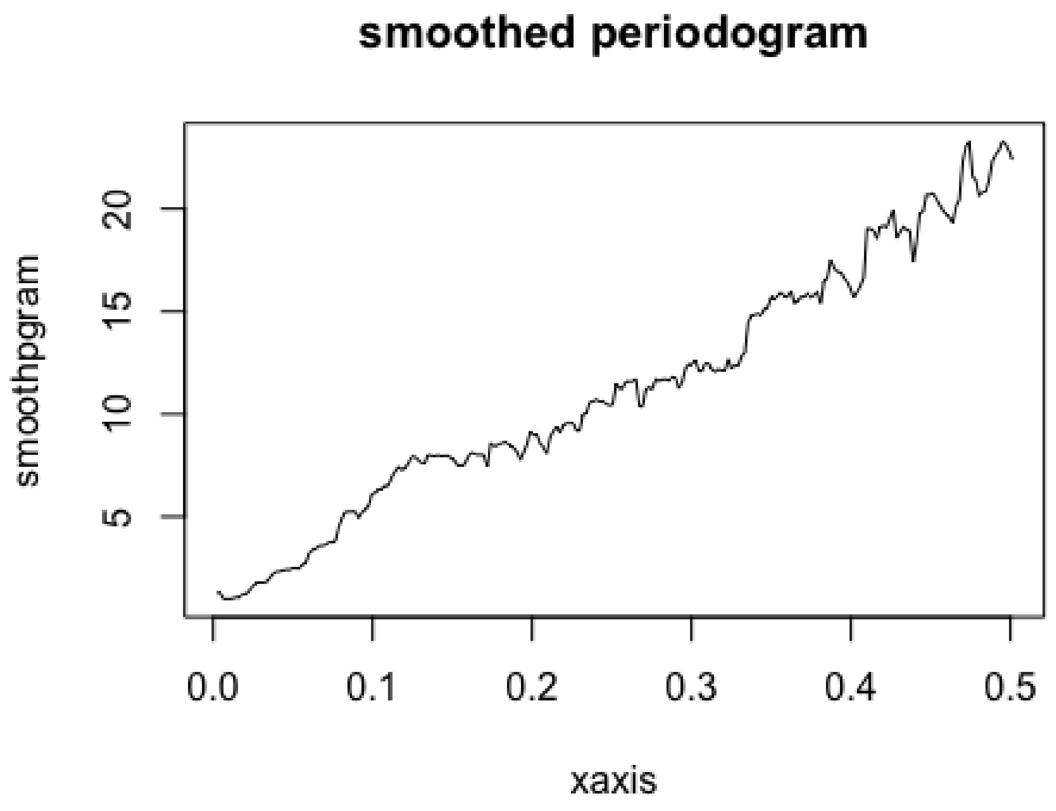
Comment: The simulated ACF is roughly the same shape as the acutal ACF plot. The simulation is pretty accurate.

8. Consider the first dataset, q1train.csv, which is uploaded to piazza. Remove the trend and seasonality by differencing first with order 52 and then a usual differencing. Call the resulting dataset xt, t = 1, . . . , n to which a stationary model can be fit.

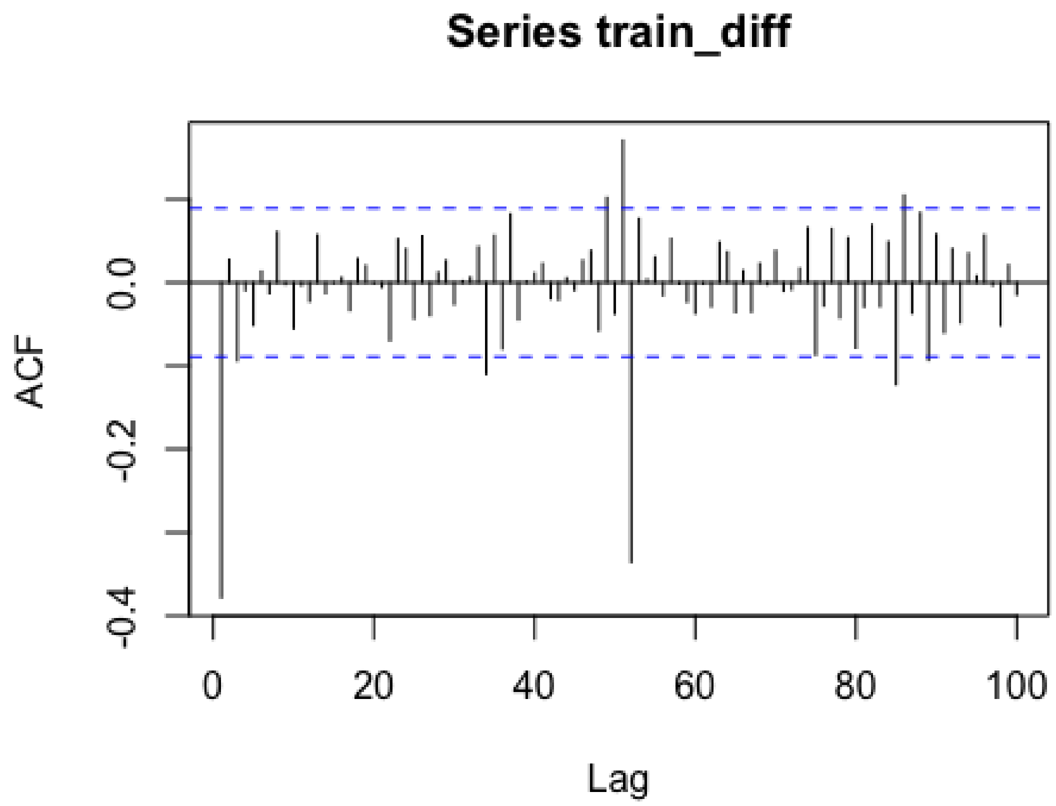
(a) Estimate the spectral density of {Xt} nonparametrically from the data {xt}

```
library(TSA)
q1train <- read.csv("~/Downloads/q1train.csv")
train_diff = diff(diff(q1train[,2],52))

n = length(train_diff)
xaxis = seq(1, round(n/2))/n
periodogram = abs(fft(train_diff)[2:(round(n/2) + 1)])^2 / n


pgram = c(rev(periodogram), periodogram, rev(periodogram))
smoothpar = 45
smoothpgram = filter(pgram,rep(1,smoothpar)/smoothpar)
smoothpgram = smoothpgram[(round(n/2) + 1 ): (2*round(n/2))]
```

```r
# perform nonparametric spectral estimate
plot(xaxis, smoothpgram, type = "l", main = "smoothed periodogram")
```

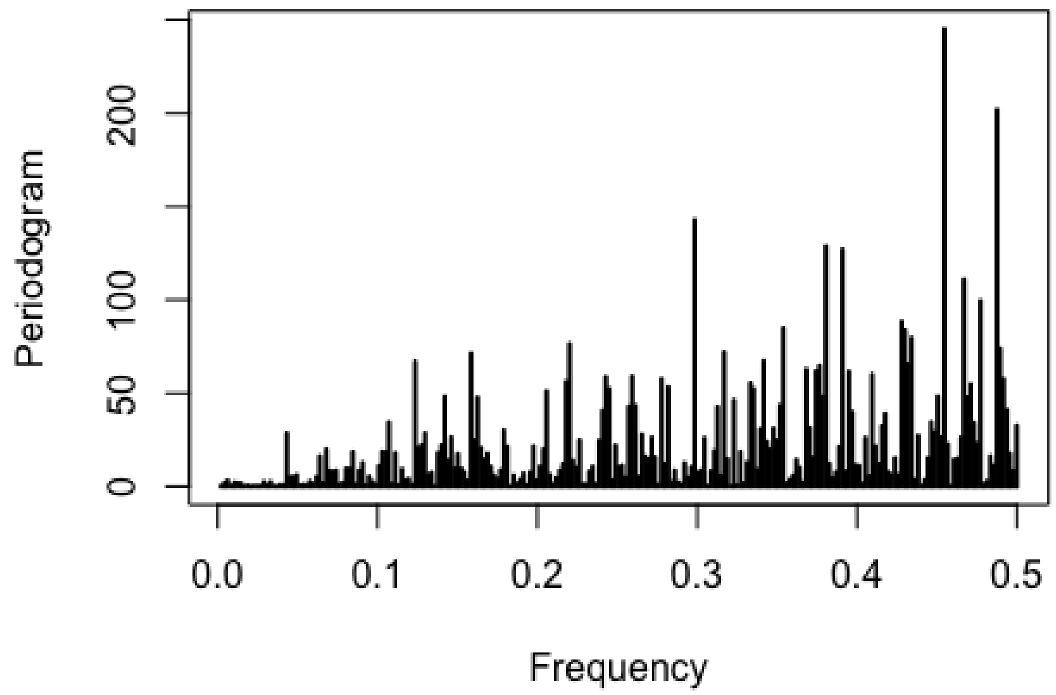


```r
acf(train_diff,lag.max = 100)
```

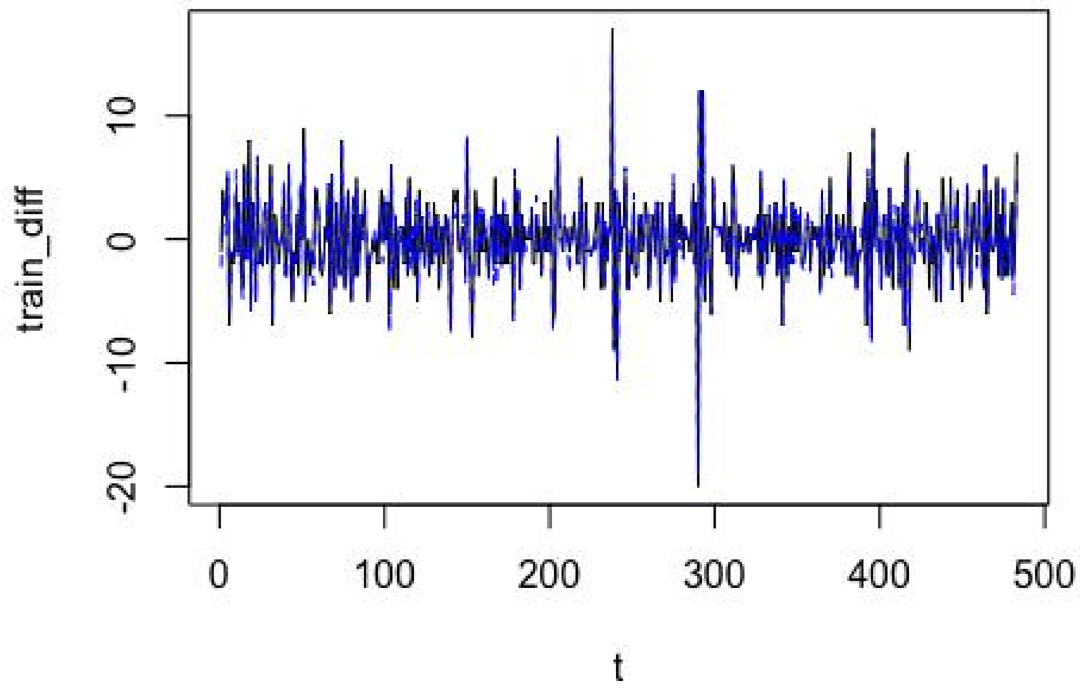


(b) Fit a reasonable stationary model to {xt} and estimate the spectral density of {Xt} by the spectral density of the fitted model. Consider the following model ∇Yt=∑j=1m[Ajcos(2πfjt)+Bjsin(2πfjt)]+et where m is the number of key frequencies, Aj and Bj are unknown constants, fj's are the key frequencies, {et} is normal white noise.

```
spec_diff = periodogram(train_diff)
```

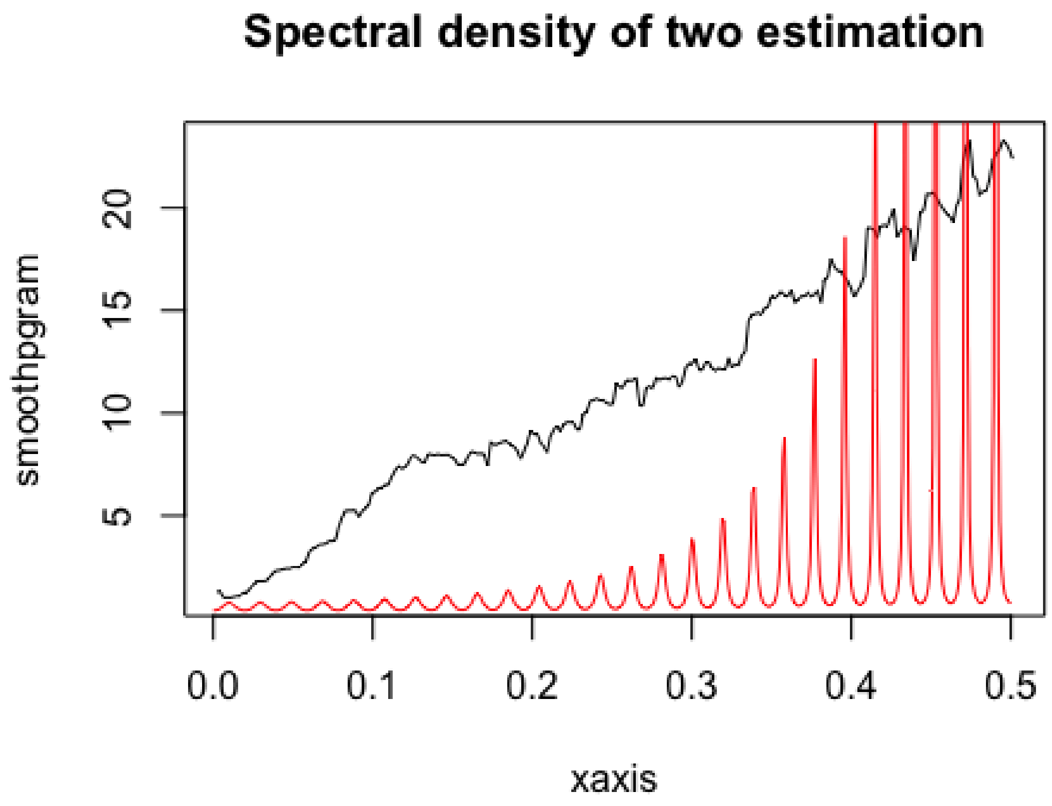


```r
# estimated spectral density greater than 9
key_freq <- spec_diff$freq[which(spec_diff$spec > 3^2)]
t <- 1:length(train_diff)
harmonics <- do.call(cbind, lapply(key_freq, function(freq){
  cbind(cos(2 * pi * freq * t), sin(2 * pi * freq * t))
}))
reslm <- lm(train_diff ~ harmonics)

plot(t, train_diff, type="l")
lines(fitted(reslm)~t, col=4, lty=2)
```



** Plot two estimates on the same plot **

```
model = arima(train_diff, order = c(1,0,0), seasonal =
list(order=c(1,0,0),period = 52))
len = seq(0,0.5,0.001)
ar1 = model$coef['ar1']
sar1 = model$coef['sar1']
ar = c(ar1, rep(0,50), sar1, ar1 * sar1)
spec.dens = ARMAspec(list(ar = ar), plot = FALSE)$spec
plot(xaxis,smoothpgram,type = "l", main = "Spectral density of two
estimation" )
abline(h = 0)
points(len, spec.dens, type = "l", col = "red")
```



Comment: The shape of the two estimates of the spectral density is increasing with a time period of 52. The non-paramete estimate seeems to be a better fit because it is more "smoothed" than the orginal periodogram. The periodgram of the orginal data shows an increasing trend but it is not very accurate since this method is a bit more wiggely and is difficult to find a good m. The nonparametric estiamtion can fit to the data better but it might be very far away from the periodogram.