Time Series Analysis Coursework

JIAQI LI

17/12/2021

Before I start I will mention that my time series number is 106.

Question1

1(a)

```r
library("numbers")
library(ggplot2)
```

```
## Registered S3 methods overwritten by 'tibble':
##   method     from
##   format.tbl pillar
##   print.tbl  pillar
```

```r
library(stats)
```

By lecture notes, the spectral density function of ARMA is $S_X(f) = S_\epsilon(f)\frac{G_{|\theta(f)|^2}}{G_{|\phi(f)|^2}}$, then assume p and q are the lengths of phis and thetas, then we can divide into the following cases to calculate: $p = 0, q = 0$, $p = 0, q > 0$, $p > 0, q = 0$, $p > 0, q > 0$.

```r
S_ARMA <- function(f,phis,thetas,sigma2){
  p <- length(phis) # Represent the length of phis
  q <- length(thetas) # Represent the length of thetas
  theta <- 1
  phi <- 1
  for (j in 1:q){theta = theta - thetas[j]*(cos(-2*pi*f*j)+sin(-2*pi*f*j)*1i)}
  for (j in 1:p){phi = phi - phis[j]*(cos(-2*pi*f*j)+sin(-2*pi*f*j)*1i)}
  if (p == 0){
    if(q == 0){t = 0} else{t = sigma2*(abs(theta)^2)}}
    else
    {if(q == 0){t = sigma2/((abs(phi)^2))}
        else{t = sigma2*(abs(theta)^2)/(abs(phi)^2)}}
return(t)}
sigma2 <- 1
```

1(b)

To Simulate the Gaussian ARMA(2,2) with length N, so we can generate ϵ by Normal(0) and $sd = \sigma$.

```
ARMA22_sim <- function(phis,thetas,sigma2,N){
  X <- rep(0, 100+N)
  r <- rnorm((100+N),0,sqrt(sigma2))
  for (i in 3:(100+N)){  # X1 = X2 = 0 and start loop in X3
    X[i] <- phis[1]*X[i-1]+phis[2]*X[i-2]+r[i]-thetas[1]*r[i-1]-thetas[2]*r[i-2]}
  return(X[101:(100+N)])}
```

1(c)

By lecture notes, we have learned that $\hat{S}^{(p)}(f) = \frac{1}{N}|\sum_{t=1}^{N} X_t e^{-i2\pi ft}|^2$ with the fourier frequency $f_k = \frac{k}{N}$, $k = 0, 1, ..., N-1$.
periodogram(X):

```
periodogram <- function(x){
  period <- 1/(length(x))*(abs(fft(x)))^2
  return(period)}
```

Direct:
The p $\times$ 100% cosine taper is defined by

$$h(t) = \begin{cases} \frac{C}{2}[1 - cos(\frac{2\pi t}{\lfloor pN \rfloor + 1})] & 1 < t \leq \frac{\lfloor pN \rfloor}{2} \\ C & \frac{\lfloor pN \rfloor}{2} < t < N + 1 - \frac{\lfloor pN \rfloor}{2} \\ \frac{C}{2}[1 - cos(\frac{2\pi(N+1-t)}{\lfloor pN \rfloor + 1})] & N + 1 - \frac{\lfloor pN \rfloor}{2} \leq t \leq N \end{cases}$$

So I can use the cosine taper to find out the direct function:

```
direct <- function(X,p){
  N <- length(X)
  m <- floor(p*N)/2
  htnormalized <- function(t){
    ifelse(t<=m, (1-cos(2*pi*t/(floor(p*N)+1)))/2,ifelse(t < N + 1 - m, 1,
    (1-cos(2*pi*(N+1-t)/(floor(p*N)+1)))/2))} #Assume constant in h(t) is 1
  constant <- sqrt(1/sum(htnormalized(1:N)^2)) # Constant in h(t) function
  ht <- htnormalized(1:N)*constant #Comupte the exact h(t)
  f <- abs(fft(ht*X))^2
  return (f)}
```

1(d)_A

By notes we know that for a complex conjugate pair, we can calculate ϕ by f'. Now $1 - \phi_{1,2}z - \phi_{2,2}z^2 = (1 - az)(1 - bz) = 1 - (a + b)z + abz^2$, so the roots are $z_1 = \frac{1}{a}$ and $z_2 = \frac{1}{b}$ and $\phi_{1,2} = (a + b)$, $\phi_{2,2} = -ab$. Then $a = re^{i2\pi f'}$ and $b = re^{-i2\pi f'}$ and $\phi_{1,2} = 2rcos(2\pi f')$ and $\phi_{2,2} = -r^2$.

```
N <- 128
freq <- function(r){
p <-c(0.05, 0.1, 0.25, 0.5)
thetas <- c(-0.5, -0.2)
m <- matrix(nrow = 10000, ncol = 15) # create 10000*15 matrix m to store values
f <- c(12/128, 32/128, 60/128) #Store f to do ARMA22_sim later
phis <- c(2*r*cos(2*pi*12/128), -r^2) #Store phis to do ARMA22_sim later
for (k in 1:10000){for (i in 1:3){
```

```
  fi <- f[i]
    for (j in 1:4){
      pv <- p[j]
      X <- ARMA22_sim(phis, thetas,sigma2,N)
      #Periodogram and direct are random, so store them first
      period <- periodogram(X)
      directv <- direct(X,pv)
      m[k,(5*(i-1)+1)] <- period[fi*128+1]
      #column 1,6,9 are periodogram with frequency=12/128,32/128,60/128
      m[k,(5*(i-1)+1+j)] <- directv[fi*128+1]}}}
      #column 2-4,7-10,12-15 are direct with frequency = 12/128,32/128,60/128
return (m)}
```

1(d)_B

By lecture notes $bias(\hat{\theta}) = E(\hat{\theta}) - \theta$, so I just need to calculate the mean of each column here.

```
r <- 0.8
m <- freq(r)
bias <- function(r,matrix){
c <- rep(0,15)
d <- rep(0,15)
phis <- c(2*r*cos(2*pi*12/128), -r^2) #Store phis to do S_ARMA later
thetas <- c(-0.5, -0.2) #Store thetas to do S_ARMA later
sigma2 <- 1 #Store sigma2 to do S_ARMA later
for (i in 1:15){c[i] <- mean(m[,i])} # calculate each column mean
f <- c(12/128, 32/128, 60/128) #Store f to do S_ARMA later
for (i in 1:3){
  fv <- f[i] # Calculate the true spectral density
  d[5*(i-1)+1] <- S_ARMA(fv,phis,thetas,sigma2)
  d[5*(i-1)+2] <- S_ARMA(fv,phis,thetas,sigma2)
  d[5*(i-1)+3] <- S_ARMA(fv,phis,thetas,sigma2)
  d[5*(i-1)+4] <- S_ARMA(fv,phis,thetas,sigma2)
  d[5*(i-1)+5] <- S_ARMA(fv,phis,thetas,sigma2)}
bias <- c -d #Calculate the difference which is value of bias
return(bias)}
r <- 0.8
biasv <- bias(r,m)
namesv <- c('bias when period f=12/128','bias when direct p=0.05 f=12/128',
            'bias when direct p=0.1 f=12/128','bias when direct p=0.25 f=12/128',
            'bias when direct p=0.5 f=12/128','bias when period f=32/128',
            'bias when direct p=0.05 f=32/128','bias when direct p=0.1 f=32/128',
            'bias when direct p=0.25 f=32/128','bias when direct p=0.5 f=32/128',
            'bias when period f=60/128','bias when direct p=0.05 f=60/128',
            'bias when direct p=0.1 f=60/128','bias when direct p=0.25 f=60/128',
            'bias when direct0.5 60/128')
print(data.frame(names=namesv, bias = biasv))
```

```
##                                names          bias
## 1         bias when period f=12/128 -1.0302935989
## 2  bias when direct p=0.05 f=12/128 -1.0633642253
## 3   bias when direct p=0.1 f=12/128 -1.4240736096
## 4  bias when direct p=0.25 f=12/128 -1.3950462301
```

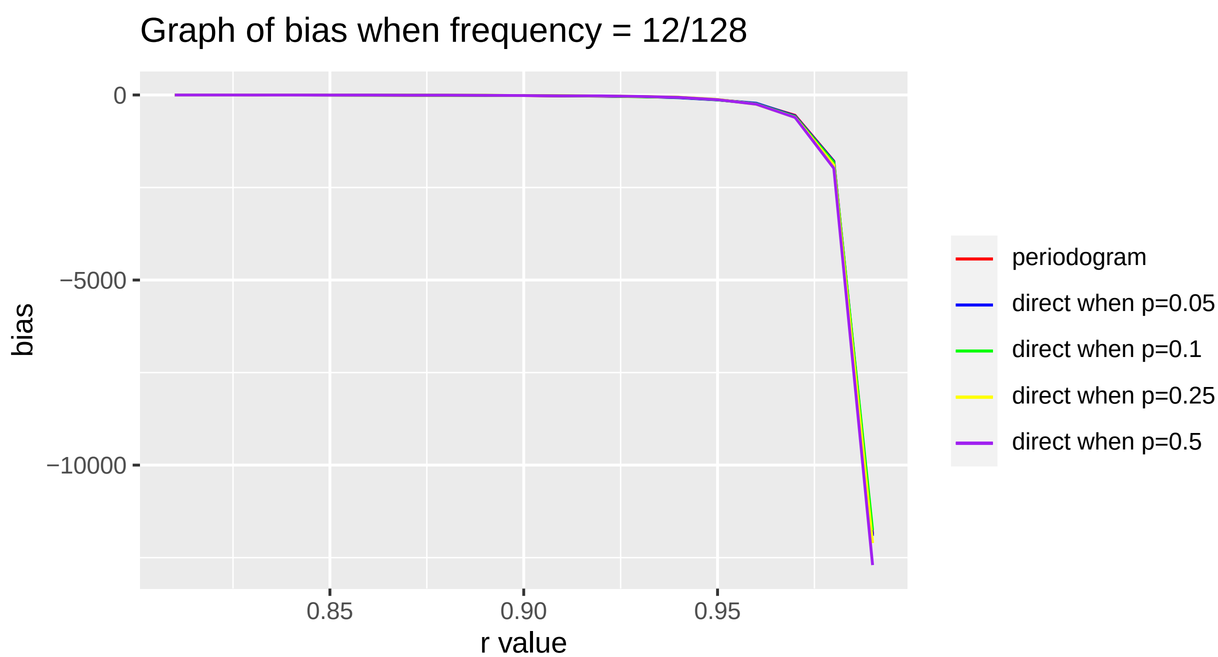
```
## 5   bias when direct p=0.5 f=12/128 -0.1900474906
## 6         bias when period f=32/128  0.1456857121
## 7  bias when direct p=0.05 f=32/128  0.0572582685
## 8   bias when direct p=0.1 f=32/128  0.0248172485
## 9  bias when direct p=0.25 f=32/128  0.0077977408
## 10  bias when direct p=0.5 f=32/128  0.0033826184
## 11        bias when period f=60/128  0.0516695047
## 12 bias when direct p=0.05 f=60/128  0.0010083907
## 13  bias when direct p=0.1 f=60/128 -0.0012964988
## 14 bias when direct p=0.25 f=60/128 -0.0012367625
## 15       bias when direct0.5 60/128 -0.0003540793
```

1(d)_C

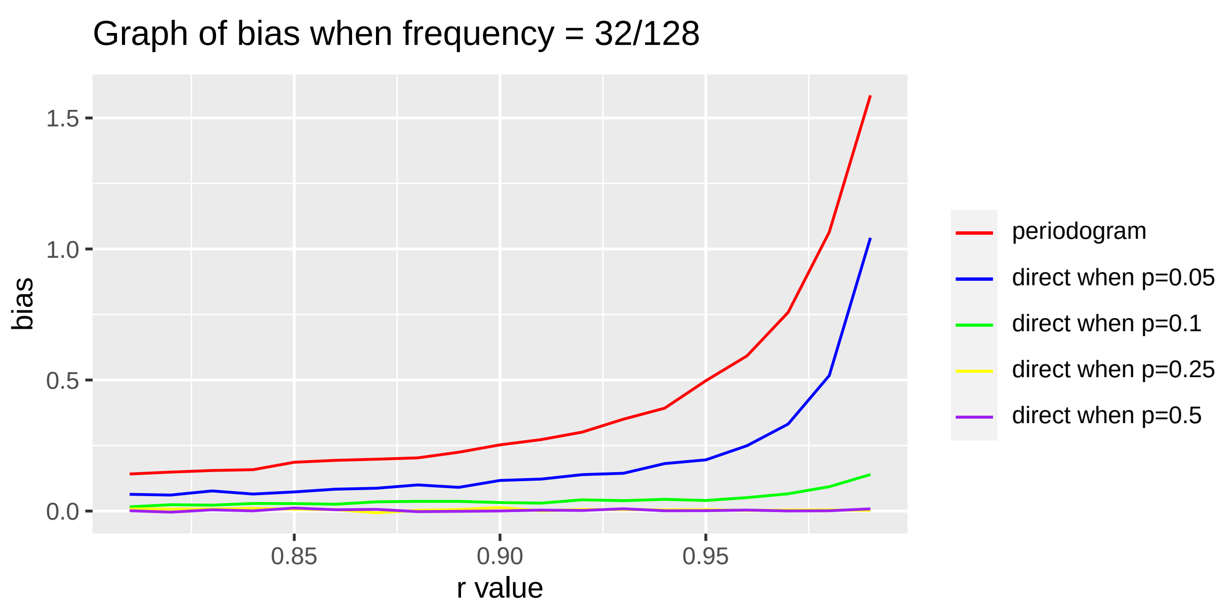
```
r <- seq(0.81, 0.99,0.01) #Set a sequence to store r value
# Set a 19*15 matrix m2 to store bias with different r
m2 <- matrix(nrow = 19, ncol = 15)
for (i in 1:19){
  rv <- r[i] #Take r = 0.81, 0.81,..., 0.99 step by step in for loop
  m <- freq(rv)
  # The function in 1(a) which store the values
  #for each at frequency=12/128,32/128,60/128
  biasv <- bias(rv,m)
  m2[i,] <- biasv}
```

1(d)_D Draw graph

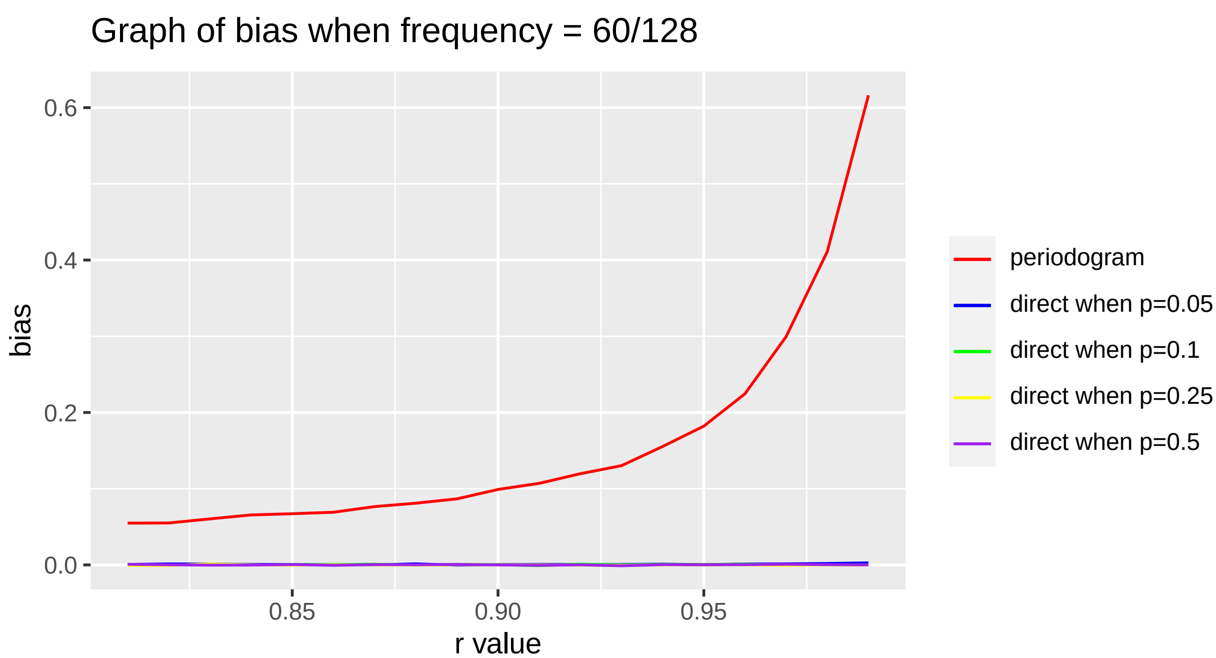
```
r <- seq(0.81, 0.99,0.01) #Set a sequence to store r value
df0 <- as.data.frame(m2) # m2 is the matrix store the bias
df <- cbind(r, df0) #combine together and create as the data frame
mylabs=list(expression("periodogram"),expression("direct when p=0.05"),
            expression("direct when p=0.1"),expression("direct when p=0.25"),
            expression("direct when p=0.5")) # Do the label
#column 1 of df is x_axis(frequencies) column 2-6 is bias under f=12/128
#column 2-6 are periodogram, direct when p=0.05, 0.1, 0.25 and 0.5 separately
p <- ggplot(df) #draw by ggplot method
p + geom_line(aes(df[,1],df[,2],colour="f1"))+
geom_line(aes(df[,1],df[,3],colour="f2"))+ geom_line(aes(df[,1],df[,4],colour="gx"))+
geom_line(aes(df[,1],df[,5],colour="gy"))+ geom_line(aes(df[,1],df[,6],colour="mx"))+
labs(x="r value",y="bias", title=expression("Graph of bias when frequency = 12/128"))+
scale_colour_manual("", values=c("f1" = "red","f2" ="blue","gx"="green","gy"="yellow",
"mx"="purple"), labels=mylabs)
```




```r
#column 1 of df is x_axis(frequencies), column 7-11 is bias under f=32/128
#similar with last graph
p <- ggplot(df)
p + geom_line(aes(df[,1],df[,7],colour="fx"))+
  geom_line(aes(df[,1],df[,8],colour="fy"))+
  geom_line(aes(df[,1],df[,9],colour="gx"))+
  geom_line(aes(df[,1],df[,10],colour="gy"))+
  geom_line(aes(df[,1],df[,11],colour="mx"))+
  labs(x="r value",y="bias",
  title=expression("Graph of bias when frequency = 32/128"))+
  scale_colour_manual("",values=c("fx"="red","fy"="blue","gx"="green",
  "gy"="yellow","mx"="purple"), labels=mylabs)
```



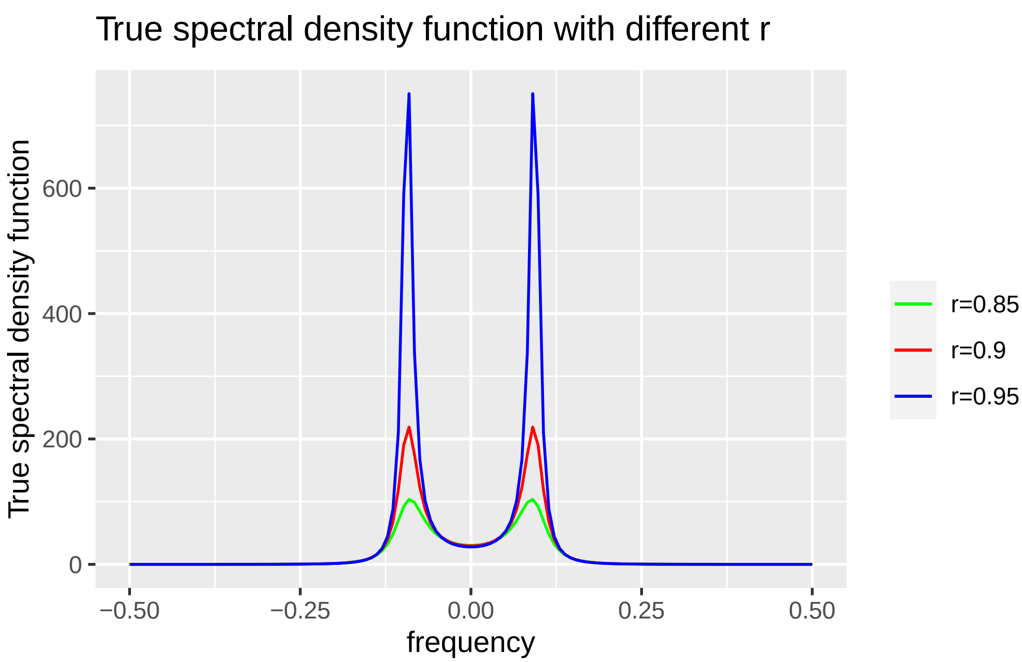
```r
#column 1 of df is x_axis(frequencies), column 7-11 is bias under f=60/128 similar with last graph
p <- ggplot(df)
p + geom_line(aes(df[,1],df[,12],colour="fx"))+
  geom_line(aes(df[,1],df[,13],colour="fy"))+
  geom_line(aes(df[,1],df[,14],colour="gx"))+
  geom_line(aes(df[,1],df[,15],colour="gy"))+
  geom_line(aes(df[,1],df[,16],colour="mx"))+
  labs(x="r value",y="bias",
  title=expression("Graph of bias when frequency = 60/128"))+
  scale_colour_manual("",values=c("fx"="red","fy"="blue","gx"="green",
  "gy"="yellow","mx"="purple"), labels=mylabs)
```



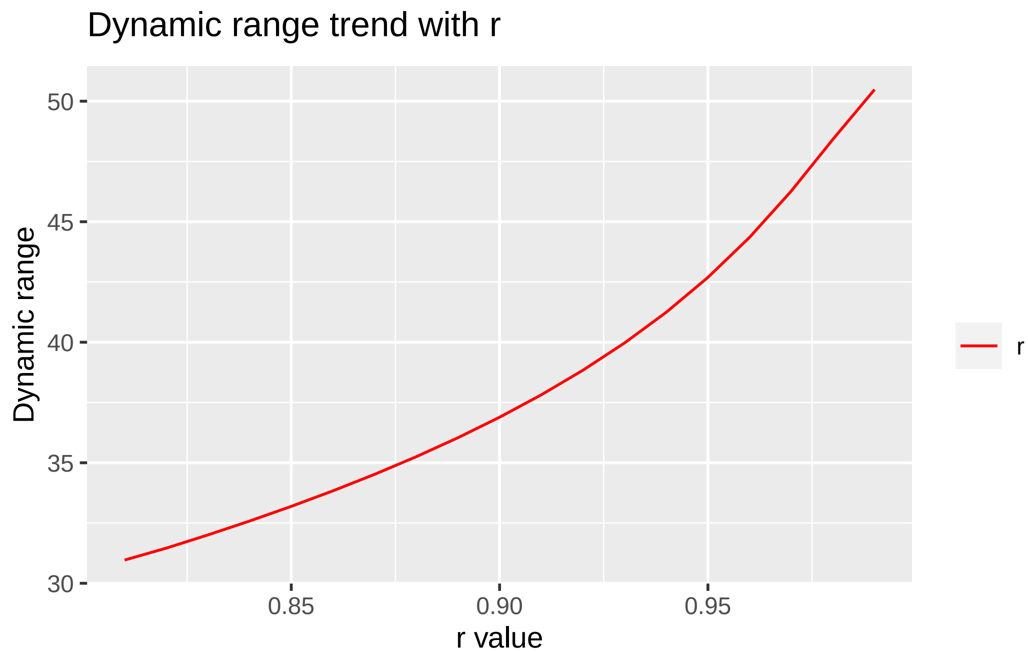
1(e)

```r
thetas <- c(-0.5, -0.2)
r <- 0.85
f <- seq(-0.5, 0.5,len=128)
phis <- c(2*r*cos(2*pi*12/128), -r^2) #Compute phis when r = 0.85
ts1 <- S_ARMA(f,phis,thetas,sigma2)
#set ts1 as the (theoretical) spectral density function for an ARMA(p, q) process
#when r = 0.85
r <- 0.9
phis <- c(2*r*cos(2*pi*12/128), -r^2) #Compute phis when r = 0.9
ts2 <- S_ARMA(f,phis,thetas,sigma2) #set ts2 similar with ts1 when r = 0.9
r <- 0.95
phis <- c(2*r*cos(2*pi*12/128), -r^2)  #Compute phis when r = 0.95
ts3 <- S_ARMA(f,phis,thetas,sigma2) #set ts3 similar with ts1 when r = 0.95
#Set ts1, ts2, ts3 as data frame in order to use the ggplot
tsd1 <- as.data.frame(ts1)
tsd2 <- as.data.frame(ts2)
tsd3 <- as.data.frame(ts3)
#Put ts1, ts2, ts3 and f together in order to use the ggplot
```

```r
tsd <- cbind(f,tsd1,tsd2,tsd3)
mylabs=list(expression("r=0.85"),expression("r=0.9"),expression("r=0.95"))
p <- ggplot(tsd)
p + geom_line(aes(tsd[,1],tsd[,2],colour="r=0.85"))+
  geom_line(aes(tsd[,1],tsd[,3],colour="r=0.9"))+
  geom_line(aes(tsd[,1],tsd[,4],colour="r=0.95"))+
  labs(x="frequency",y = "True spectral density function",
       title=expression("True spectral density function with different r"))+
  scale_colour_manual("",
        values=c("r=0.85"="green","r=0.9"="red","r=0.95"="blue"), labels=mylabs)
```




```r
r <- seq(0.81, 0.99,0.01) #Set a sequence to store r value
# Set a 19*1 matrix to store dynamic range with different r
drdf <- matrix(nrow = 19, ncol = 1)
for (i in 1:19){
  rv <- r[i]  #Take r = 0.81, 0.81,..., 0.99 step by step in for loop
  #Compute different phis with different r
  phis <- c(2*rv*cos(2*pi*12/128), -rv^2)
  X <- S_ARMA(f,phis,thetas,sigma2)
  #Store the value of dynamic range in matrix drdf
  drdf[i,] <- 10*log10(max(X)/min(X))}
drdf <- as.data.frame(drdf)
#Store as data frame and first column is r value, second is Dynamic range
rfdata <- cbind(r,drdf)
p <- ggplot(rfdata)
p + geom_line(aes(rfdata[,1],rfdata[,2],colour="r"))+
labs(x="r value",y="Dynamic range",title=expression("Dynamic range trend with r"))+
  scale_colour_manual("", values=c("r" = "red"))
```



At each frequency periodogram has larger bias than direct with all tapering, so tapering can reduce to sidelobes associated with F'{e}jer's kernel. And by th graph above we can find that the sidelobes becomes lower so the bias becomes lower. By lecture notes, we know that for a process with large dynamic range, defined as $10log_{10}(\frac{max_f S(f)}{min_f S(f)})$ since the expected value of the periodogram is convolution of F'{e}jer's kernel and the true spectrum, power from parts of the spectrum where S(f) is large can "leak" via the sidelobes to other frequencies where S(f) is small.So according to notes, we can find that a larger dynamic range will lead to a sidelobe leak and then increase bias. Here, different R will lead to changes in dynamic range, so the bias will also change accordingly. And by graph we can find that the bias is larger when the spectral density is large, and I can also find the bias increase as r increase, here the absolute value of bias increases as r increases, and increases sharply from r in range (0.9,0.95).

Question2

2(a)

```
install.packages("waved", repos = "http://cran.us.r-project.org")
```

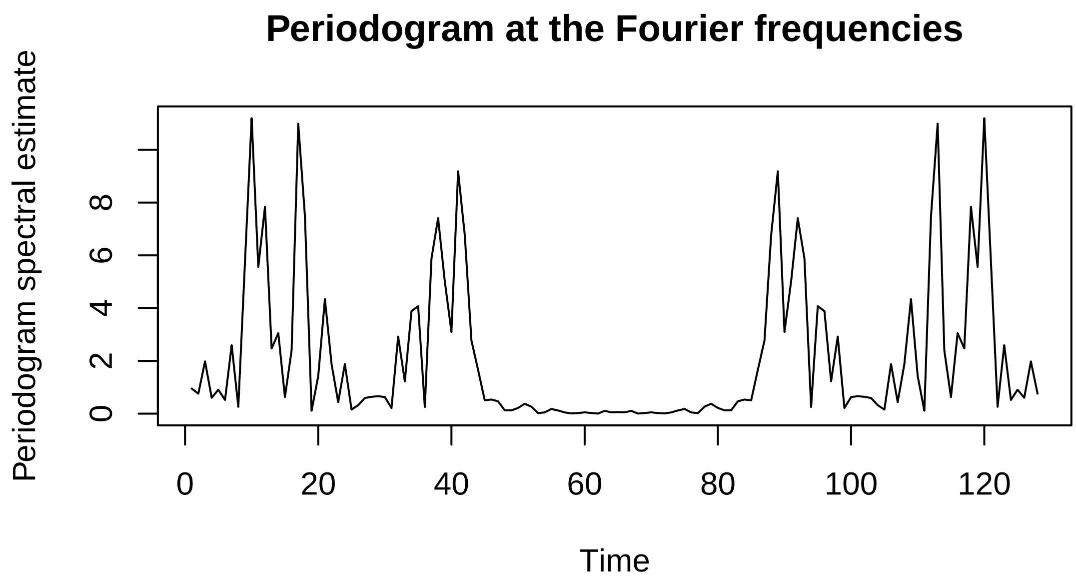
```
##
## The downloaded binary packages are in
##  /var/folders/41/06f286rn5v593zrddrl2vqd40000gn/T//RtmpU5Fe3I/downloaded_packages
```

```
library("waved")
```

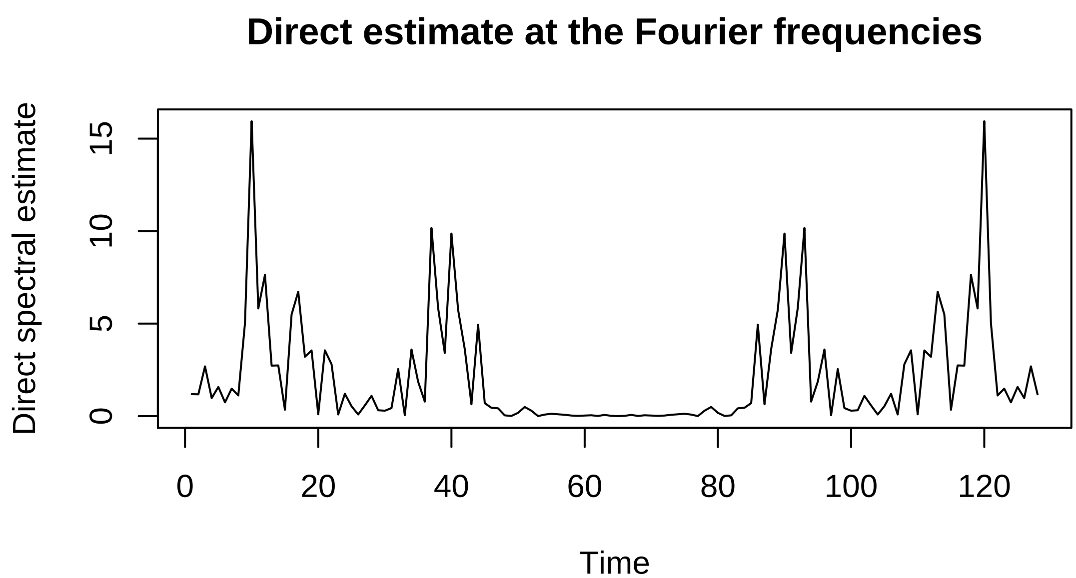
```
##
## Attaching package: 'waved'

## The following object is masked from 'package:base':
##
##     scale
```

```r
Xd <- as.numeric(read.csv("106.csv",header = FALSE)) #read dataset as number type
ts.plot(fftshift(periodogram(Xd)),ylab ="Periodogram spectral estimate",
        main ="Periodogram at the Fourier frequencies")
```




```r
ts.plot(fftshift(direct(Xd,0.5)),ylab ="Direct spectral estimate",
        main ="Direct estimate at the Fourier frequencies")
```



2(b)(1) Yule-Walker (untapered)

```r
YW <- function(X,p){
  n <- length(X)
  x <- sapply(0:p,function(t) X[1:(n-t)]%*%X[(1+t):n])
  x <- x/n
  #done toeplitz algorithm by function to compute toeplitz matrix
  matrix <- toeplitz(x[1:p])
  matrix_inv <- solve(matrix) #Use solve function to write the matrix_inv
```

```r
  phis <- matrix_inv%*%x[2:(p+1)] #Computing phis
  sigma2 <- x[1] - sum(x[2:(p+1)]*phis) #computing sigma2
  return(c(phis,sigma2))}
```

2(b)(2) Yule-Walker (50% cosine tapered)

$$h(t) = \begin{cases} \frac{C}{2}[1 - cos(\frac{2\pi t}{\lfloor pN \rfloor+1})] & 1 < t \leq \frac{\lfloor pN \rfloor}{2} \\ C & \frac{\lfloor pN \rfloor}{2} < t < N+1-\frac{\lfloor pN \rfloor}{2} \\ \frac{C}{2}[1 - cos(\frac{2\pi(N+1-t)}{\lfloor pN \rfloor+1})] & N+1-\frac{\lfloor pN \rfloor}{2} \leq t \leq N \end{cases}$$

```r
#The only difference with untapered one is X_new need equal to h(t)*X
YWtaper <- function(X,p){
  n <- length(X)
  m <- floor(0.5*n)/2
  q <- n + 1 - m
  htnormalized <- function(t){  #the function when constant is 1
    ifelse(t<=m,(1-cos(2*pi*t/(floor(0.5*n)+1)))/2,
           ifelse(t<q,1,(1-cos(2*pi*(n+1-t)/(floor(0.5*n)+1)))/2))}
  constant <- sqrt(1/sum(htnormalized(1:n)^2)) #computing the constant C
  ht <- htnormalized(1:n)*constant # the exactly ht
  X <- ht*X
  x <- sapply(0:p,function(t) X[1:(n-t)]%*%X[(1+t):n])
  matrix <- toeplitz(x[1:p]) # To compute toeplitz matrix
  matrix_inv <- solve(matrix)
  phis <- matrix_inv%*%x[2:(p+1)]
  sigma2 <- x[1] - sum(x[2:(p+1)]*phis)
  return(c(phis,sigma2))}
```

2(b)(3) Approximate Maximum Likelihood

```r
ML <- function(X,p){
  n <- length(X)
  m <- c()
  for(i in 1:(n-p)){
    m <- rbind(m,X[(p+i-1):i])} # (n-p)*n matrix}
    X <- X[(p+1):n] #store as new X
    phis <- solve(t(m)%*%m)%*%t(m)%*%X # phis is a vector of length p
    sigma2 <- t(X-m%*%phis)%*%(X-m%*%phis)/(n-p) #computing sigma2
    return (c(phis,sigma2))}
```

2(c)

```r
AIC <- function(X,p){
  n <- length(X)
  #The p+1 value in output of (b) is sigma2
  sigma2 <- c(YW(X,p)[p+1],YWtaper(X,p)[p+1],ML(X,p)[p+1])
  AIC <- 2*p+n*log(sigma2) #function of AIC
  return(AIC)} #Write AIC function
AICv <- c()
for (p in 1:20){AICv <- rbind(AICv, AIC(Xd,p))}
AICv <- data.frame(YuleWalker=AICv[,1],taperedYuleWalker=AICv[,2],
                   Maximumlikelihood=AICv[,3])
print(AICv)
```

```
##    YuleWalker taperedYuleWalker Maximumlikelihood
## 1  84.5190886       83.39170222        85.3294190
## 2  74.4019940       75.10063417        75.1659047
## 3  52.0044012       50.94379011        51.7372391
## 4  -1.6113512       -5.90308669        -8.7985266
## 5  -0.7703429       -4.68253167        -8.1861870
## 6   0.6628292       -4.10058122        -6.6603840
## 7   2.5493373       -2.85438943        -4.0147423
## 8   3.8385379       -3.29557361        -2.8806945
## 9   4.2734130       -2.70655416        -1.6109740
## 10  4.5825497       -1.75124422        -2.4592737
## 11  5.6230712       -1.77249701        -2.5030315
## 12  7.2314722        0.01722625         0.0342247
## 13  9.2234214        1.43899074         2.4142550
## 14 10.4207277        2.23427216         3.8191149
## 15 12.3788947        4.23363799         6.6701929
## 16 12.5243149        4.62337129         6.5099216
## 17 14.4174454        6.31173319         9.3653688
## 18 16.2675574        7.67924899         8.2580935
## 19 17.1849231        9.66936800        10.9382322
## 20 18.6440776       10.95721780        11.3332029
```

2(d)

For the untapered Yule-Walker by table in part(C), I can find that the minimum of AIC is occurred when order $p = 4$, and the $p + 1$ estimated parameter values for this method are shown below.

```r
YW(Xd,4)
```

```
## [1] -0.7926425 -0.7578738 -0.7403209 -0.5936416  0.9276612
```

Untapered Yule-Walker: $\phi = [-0.7926425 - 0.7578738 - 0.7403209 - 0.5936416]$ and $\hat{\sigma_\epsilon}^2 = 0.9276612$.
For the Yule-Walker with 50% cosine tapered, by table in part(C), I can find that the minimum of AIC is occurred when order $p = 4$, and the $p + 1$ estimated parameter values for this method are shown below.

```r
YWtaper(Xd,4)
```

```
## [1] -0.8172381 -0.7484325 -0.7675994 -0.6070856  0.8970732
```

Yule-Walker with 50% cosine tapered: $\phi = [-0.8172381 - 0.7484325 - 0.7675994 - 0.6070856]$ and $\hat{\sigma_\epsilon}^2 = 0.8970732$.
For the Approximate Maximum Likelihood, by table in part(C), I can find that the minimum of AIC is occurred when order $p = 4$, and the $p + 1$ estimated parameter values for this method are shown below.

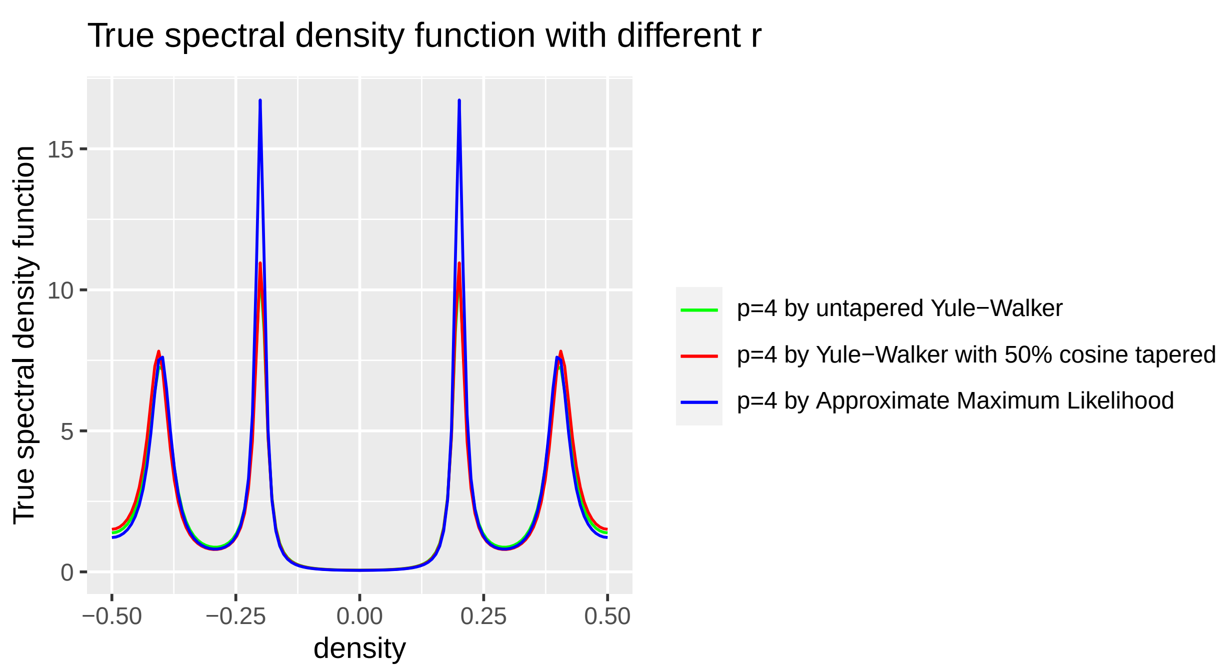
```r
ML(Xd,4)
```

```
## [1] -0.8023264 -0.8010116 -0.7881203 -0.6377428  0.8770086
```

Approximate Maximum Likelihood: $\phi = [-0.8023264 - 0.8010116 - 0.7881203 - 0.6377428]$ and $\hat{\sigma_\epsilon}^2 = 0.8770086$.

2(e)

```r
p <- 4
f <- seq(-0.5, 0.5,len=128)
#To simulate AR model we can use S_ARMA function and seem thatas as a empty set
thetas <- c()
phis <- YW(Xd,p)[1:4]
#set sdf1 as the theoretical spectral density function for AR(p)
sigma2 <- YW(Xd,p)[5] #when p=4 by untapered Yule-Walker
sdf1 <- S_ARMA(f,phis,thetas,sigma2)
phis <- YWtaper(Xd,p)[1:4]
sigma2 <- YWtaper(Xd,p)[5]
sdf2 <- S_ARMA(f,phis,thetas,sigma2)  #set sdf2 with 50% cosine tapered
phis <- ML(Xd,p)[1:4]
sigma2 <- ML(Xd,p)[5]
sdf3 <- S_ARMA(f,phis,thetas,sigma2)#set sdf3 by Approximate Maximum Likelihood
tsd1 <- as.data.frame(sdf1)
tsd2 <- as.data.frame(sdf2)
tsd3 <- as.data.frame(sdf3) #Set sdf1, sdf2, sdf3 as data frame to use the ggplot
tsd <- cbind(f,tsd1,tsd2,tsd3) #Put sdf1, sdf2, sdf3 and f together to use ggplot
mylabs=list(expression("p=4 by untapered Yule-Walker"),
            expression("p=4 by Yule-Walker with 50% cosine tapered"),
            expression("p=4 by Approximate Maximum Likelihood"))
p <- ggplot(tsd)
p + geom_line(aes(tsd[,1],tsd[,2],colour="p1"))+
  geom_line(aes(tsd[,1],tsd[,3],colour="p2"))+ geom_line(aes(tsd[,1],
      tsd[,4],colour="p3"))+ labs(x="density",y = "True spectral density function",
  title=expression("True spectral density function with different r"))+
  scale_colour_manual("", values=c("p1"="green","p2"="red","p3"="blue"),
                      labels=mylabs)
```



Question3

3(a)

By lecture notes we can do the forecast by following steps: first find out the best order p which is in the lowest AIC(here is p = 4), then setting the innovation terms by:

$$X_t(1) = \phi_{1,1}X_t + \phi_{1,2}X_t(-1) + \phi_{1,3}X_t(-2) + \phi_{1,4}X_t(-3) + 0$$

$$X_t(2) = \phi_{1,1}X_t(1) + \phi_{1,2}X_t + \phi_{1,3}X_t(-1) + \phi_{1,4}X_t(-2) + 0$$

$$\vdots$$

$$X_t(10) = \phi_{1,1}X_t(9) + \phi_{1,2}X_t(8) + \phi_{1,3}X_t(7) + \phi_{1,4}X_t(6) + 0$$

```
Xd3 <- Xd[1:118] #Pick first 118 values in my data set
AICv2 <- c()
for (p in 1:20){
  AICv2 <- rbind(AICv2, AIC(Xd3,p)[3])}
#find out the best order p to do the forecast (where AIC is lowest)
p <- which.min(AICv2)
phis <- ML(Xd3,p)[1:p] #The first 1-p terms are phis by question 2(b)
for (s in 119:128){
  Xdx <- sum(phis * Xd3[(s-1):(s-p)])
  Xd3 <- c(Xd3,Xdx)} # Create the table to show the forecasting
ftable <- data.frame(actual = Xd[119:128], forecast = Xd3[119:128],
  difference = abs(Xd[119:128] - Xd3[119:128]))
row.names(ftable) <- c("X119","X120","X121","X122","X123","X124","X125","X126",
                       "X127","X128")
print(ftable)
```

```
##        actual    forecast difference
## X119  0.17854  1.36541653 1.18687653
## X120  2.57330 -0.22772079 2.80102079
## X121 -1.98970 -1.37557155 0.61412845
## X122 -2.56610  0.45901922 3.02511922
## X123  1.70880 -0.01190541 1.72070541
## X124  2.09970  0.88048086 1.21921914
## X125 -0.27362 -0.21676019 0.05685981
## X126 -0.60064 -0.76721639 0.16657639
## X127 -3.07930  0.11708821 3.19638821
## X128  1.43540  0.10397118 1.33142882
```

3(b)

```
p <- which.min(AICv2)
sigma <- sqrt(ML(Xd3,p)[p+1])
dp <- c()
for (i in 1:999){ # loop for 999 times
  Xd3d <- Xd[1:118]
  innovation <- rnorm(10, mean=0, sd=sigma) #define the innovations
  for (s in 119:128){
    Xdx <- sum(phis * Xd3d[(s-1):(s-p)])+innovation[s-118]
    Xd3d <- c(Xd3d,Xdx)}
  dp <- cbind(dp, Xd3d[119:128])}
# By sort the values to find out the max and min
min <- sapply(1:10, function(x) sort(dp[x,])[50])
max <- sapply(1:10, function(x) sort(dp[x,])[950]) # which means the 50th and 950th
```

```r
t <- seq(100,128) # from t = 100 to t = 128
actual <- Xd[100:128] # from t = 100 to t = 128
prediction <- c(rep(NA,19),Xd3[119:128])
mind <- as.data.frame(c(rep(NA,19),min))
maxd <- as.data.frame(c(rep(NA,19),max))
tsd <- cbind(t,actual,prediction,mind,maxd) # combine and make the dataframe
mylabs1=list(expression("True trajectory"),expression("Predicted trajectory"))
mylabs2=list(expression("90% prediction intervals")) #To do the label
p <- ggplot(tsd)
p + geom_line(aes(tsd[,1],tsd[,2],colour="f1"),na.rm = T)+
  geom_line(aes(tsd[,1],tsd[,3],colour="f2"),na.rm = T)+
  geom_ribbon(aes(x=t,ymin=tsd[,4],ymax=tsd[,5],fill="f3",alpha=0.1))+
  labs(x="time",y = "Time Series", title=expression("Monte Carlo Simulation"))+
  scale_colour_manual("", values=c("f1"="green","f2"="red"),labels=mylabs1)+
  scale_fill_manual("", values=c("f3"="blue"), labels=mylabs2)
```