Time Series: A First Course with Bootstrap Starter

Contents

Lesson 6: Active Recall space

Lesson 6: Takes and tools

Takes

Questions

key = small TPs on my data

Links to JD+

Overview in book

Concepts

R skills

Lesson 6-1: Spectral Density

- We define the spectral density, which allows us to do time series analysis in the frequency (or Fourier) domain.

Definition 6.1.2.

- The *spectral density* of a stationary time series is

$$f(\lambda) = \sum_{k=-\infty}^{\infty} \gamma(k) e^{-i\lambda k},$$

for $\lambda \in [-\pi, \pi]$.
- Also the spectral density is the restriction of the AGF to the unit circle: $f(\lambda) = G(e^{-i\lambda})$.
- A sufficient condition for existence is absolute summability of the autocovariances. This also guarantees $f(\lambda)$ is continuous.

Remark 6.1.6. Spectral Representation of the Autocovariance.

- By Fourier inversion, we can recover the autocovariances from the spectral density:

$$\gamma(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(\lambda) e^{i\lambda k} \, d\lambda.$$

- So for $k = 0$, we see the process' variance is the average integral of f.

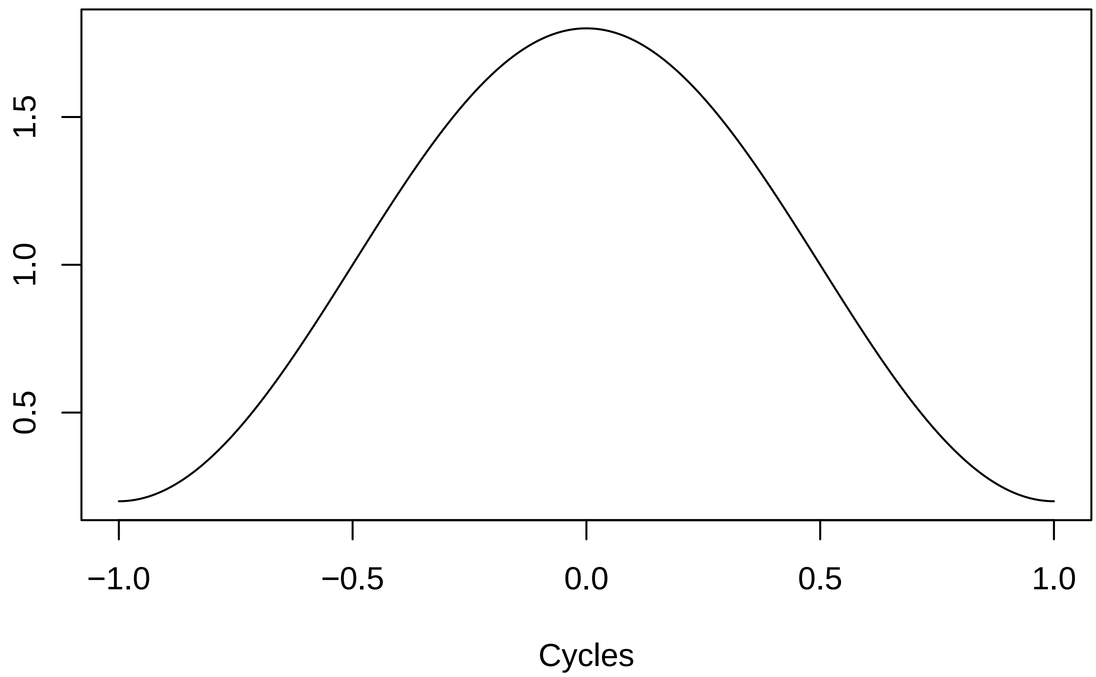
Exercise 6.4. MA(1) Spectral Density.

- For an MA(1), the spectral density is

$$f(\lambda) = \gamma(0) + 2\gamma(1)\cos(\lambda) = \gamma(0)\, \left(1 + 2\rho(1)\cos(\lambda)\right).$$

- We plot $f(\lambda)$ for $\rho(1) = .4$ and $\gamma(0) = 1$.
- The units are in π, called "Cycles".

```
mesh <- 1000
lambda <- pi*seq(-mesh,mesh)/mesh
rho.1 <- .4
spec <- 1 + 2*rho.1*cos(lambda)
plot(ts(spec,start=-1,frequency=mesh),xlab="Cycles",ylab="")
abline(h=0,col=2)
```



Fact 6.1.8. Further Properties of the Spectral Density

- Because $\gamma(k) = \gamma(-k)$, $f(\lambda)$ is real and even.
- Also $f(\lambda) \geq 0$ follows from positive definite property.

Exercise 6.7. MA(q) Spectral Density Computation.

- For an MA(q), the spectral density is

$$f(\lambda) = \gamma(0)\left(1 + 2\sum_{k=1}^{q} \rho(k)\cos(\lambda k)\right).$$

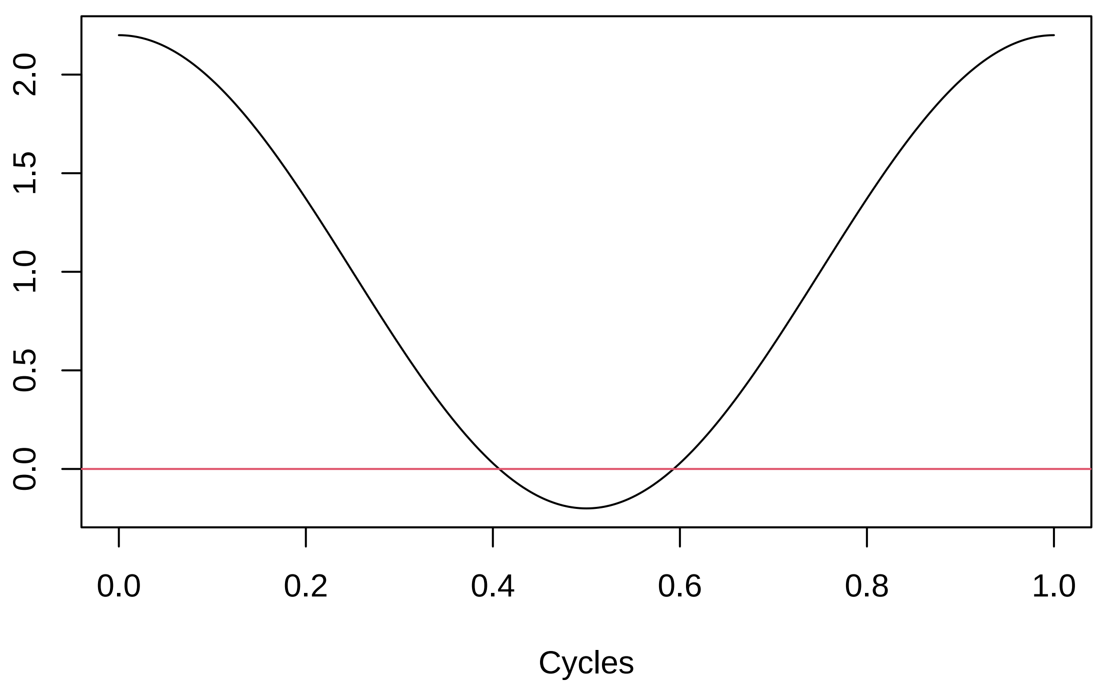
- Since it is even, we can just focus on $\lambda \in [0, \pi]$.

```
maq.spec <- function(ma.acf,mesh)
{
    q <- length(ma.acf)-1
    lambda <- pi*seq(0,mesh)/mesh
    spec <- ma.acf[1]*cos(0*lambda)
    if(q > 0)
    {
        for(k in 1:q)
        {
            spec <- spec + 2*ma.acf[k+1]*cos(k*lambda)
        }
    }
```

```r
    return(spec)
}
```

- We plot $f(\lambda)$ with $q = 2$, $\gamma(0) = 1$, $\rho(1) = 0$, and $\rho(2) = .6$.
- However, these values do not correspond to a positive definite autocovariance, and the resulting function takes negative values.
- Q: est ce grave ? see recording
- arguments fonction: detail ?
- variance et q autocorreclations (1 à q)
- ma.acf stocke ces q+1 valeurs, q est deduit

```r
spec <- maq.spec(c(1,0,.6),mesh)
plot(ts(spec,start=0,frequency=mesh),xlab="Cycles",ylab="")
abline(h=0,col=2)
```



Corollary 6.1.9.

- Suppose we filter stationary $\{X_t\}$ with some $\psi(B)$, yielding $Y_t = \psi(B)X_t$. Then the spectral densities of input and output are related by

$$f_y(\lambda) = |\psi(e^{-i\lambda})|^2 f_x(\lambda).$$

- This follows from Theorem 5.6.6.
- We call $\psi(e^{-i\lambda})$ the *frequency response function* of the filter $\psi(B)$.
- We call $|\psi(e^{-i\lambda})|^2$ the *squared gain function* of the filter $\psi(B)$.

Theorem 6.1.12.

Let $\{X_t\}$ be a stationary ARMA(p,q) process such that $\phi(B) X_t = \theta(B) Z_t$, for $Z_t \sim \mbox{WN}(0, \sigma^2)$. Suppose $\phi(z)$ has no roots on the unit circle. Then the spectral density exists:

$$f(\lambda) = \sigma^2 \, \frac{ {|\theta (e^{-i \lambda})|}^2 }{ {|\phi (e^{-i \lambda})|}^2 }$$

Exercise 6.12. ARMA(p,q) Spectral Density.

- We write code for the ARMA spectral density, based on the formula of Theorem 6.1.12, taking as input the $\theta(z)$ and $\phi(z)$ polynomials.

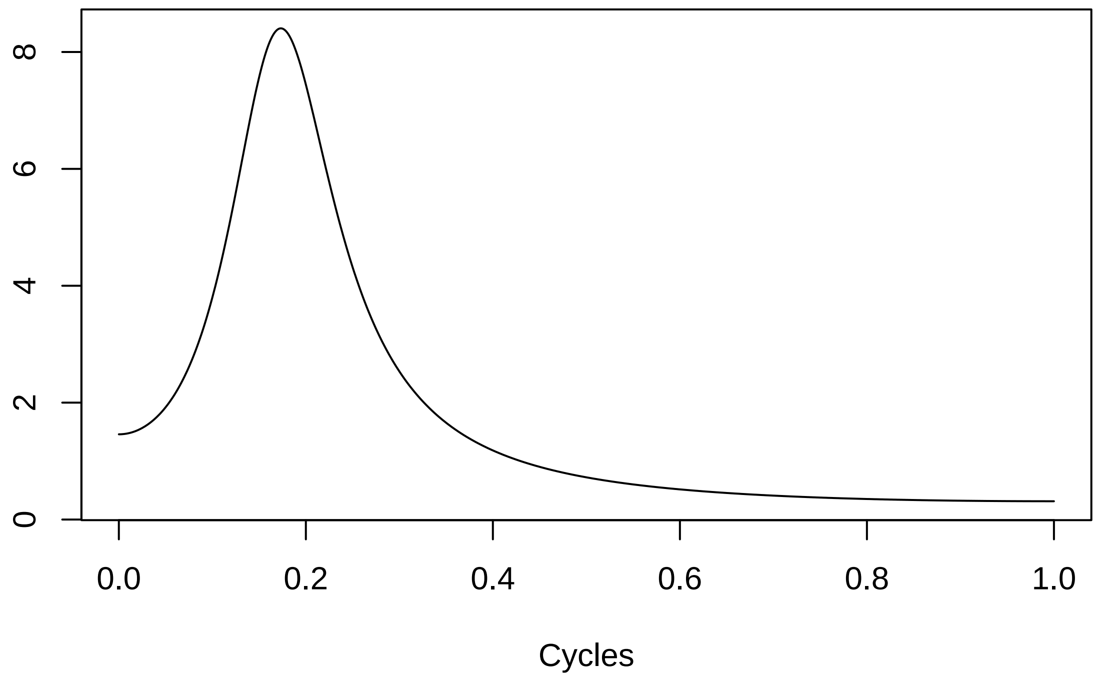
```
armapq.spec <- function(ar.coef,ma.coef,sigma,mesh)
{
    p <- length(ar.coef)
    q <- length(ma.coef)
    lambda <- pi*seq(0,mesh)/mesh
    spec.ar <- rep(1,mesh+1)
    if(p > 0)
    {
        for(k in 1:p)
        {
            spec.ar <- spec.ar - ar.coef[k]*exp(-1i*lambda*k)
        }
    }
    spec.ma <- rep(1,mesh+1)
    if(q > 0)
    {
        for(k in 1:q)
        {
            spec.ma <- spec.ma + ma.coef[k]*exp(-1i*lambda*k)
        }
    }
    spec <- sigma^2*Mod(spec.ma)^2/Mod(spec.ar)^2
    return(spec)
}
```

- We plot the spectral density of the cyclic ARMA(2,1) process of Example 5.7.2: for $\rho \in (0,1)$ and $\omega \in (0, \pi)$, let $\{ X_t \}$ satisfy

$$(1 - 2 \rho \cos (\omega) B + \rho^2 B^2) X_t = (1 - \rho \cos (\omega) B) Z_t.$$

- We set $\rho = .8$ and $\omega = \pi/6$.

```
spec <- NULL
mesh <- 1000
rho <- .8
omega <- pi/6
ar.coef <- c(2*rho*cos(omega),-1*rho^2)
ma.coef <- -1*rho*cos(omega)
spec <- armapq.spec(ar.coef,ma.coef,1,mesh)
plot(ts(spec,start=0,frequency=mesh),xlab="Cycles",ylab="",main="")
```



Corollary 6.1.14.

- Let $\{X_t\}$ be a weakly stationary, mean zero time series with strictly positive spectral density of form given in Theorem 6.1.12. Then there exists a white noise $\{Z_t\}$ such that $\phi(B)X_t = \theta(B)Z_t$.
- This is proved by defining $Z_t = \psi(B)X_t$ with $\psi(z) = \phi(z)/\theta(z)$, and checking that $\{Z_t\}$ is white noise.
- This $\psi(B)$ is a *whitening filter*. It transforms a time series to white noise!

Theorem 6.1.16. MA(∞) Representation.

Let $\{X_t\}$ be a weakly stationary, mean zero time series with autocovariance function $\gamma(k)$ that is absolutely summable, and positive spectral density. Then $\{X_t\}$ is an MA(∞) process with respect to some white noise $\{Z_t\}$:

$$X_t = \sum_{j\geq 0} \psi_j Z_{t-j},$$

and $\psi_0 = 1$.

Corollary 6.1.17. AR(∞) Representation.

Under the assumptions of Theorem 6.1.16, $\{X_t\}$ is an AR(∞) process with respect to the same white noise $\{Z_t\}$:

$$X_t = -\sum_{j\geq 1} \pi_j X_{t-j} + Z_t.$$

Lesson 6-2: Filtering in Frequency Domain

- Filters extract (or suppress) features of interest from a time series.

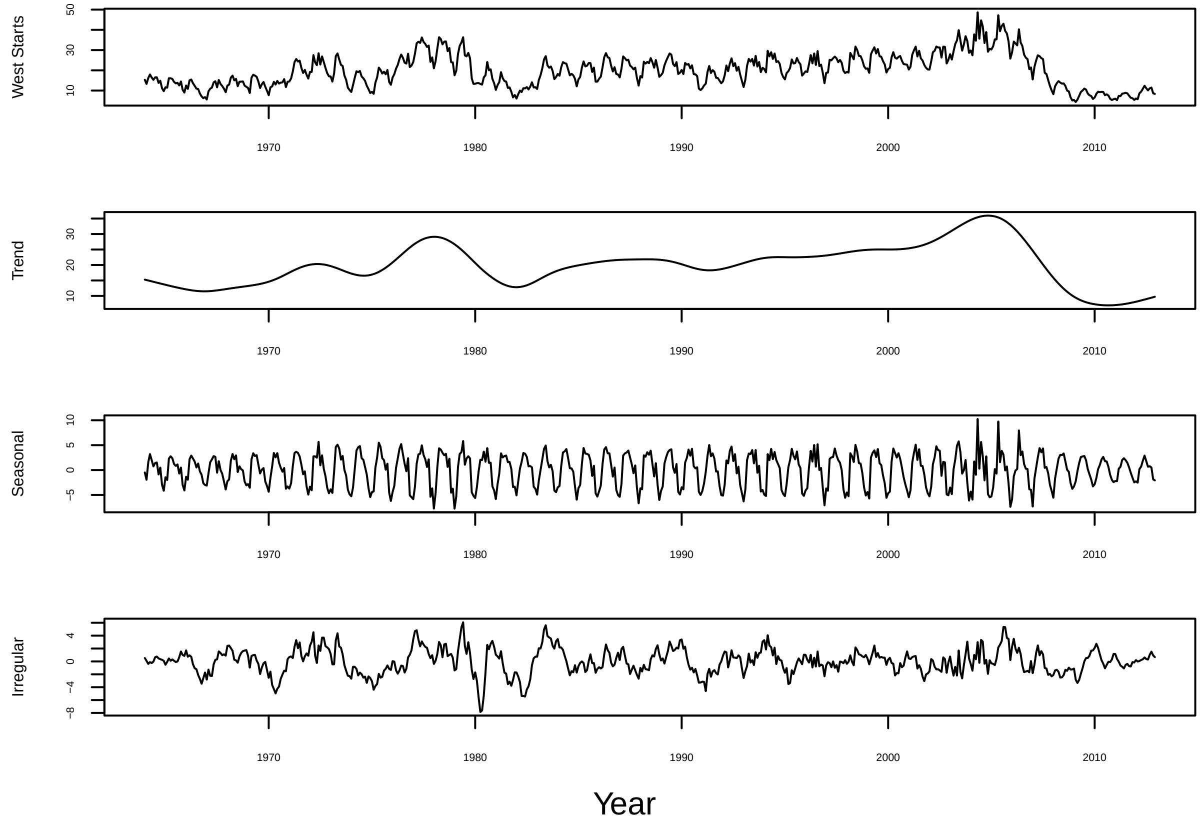
- Using Corollary 6.1.9, we can see in frequency domain how extraction and suppression occurs.

Example 6.2.1. Business Cycle in Housing Starts.

- Example 3.6.13 decomposed West Housing Starts into Trend, Seasonal, and Irregular.

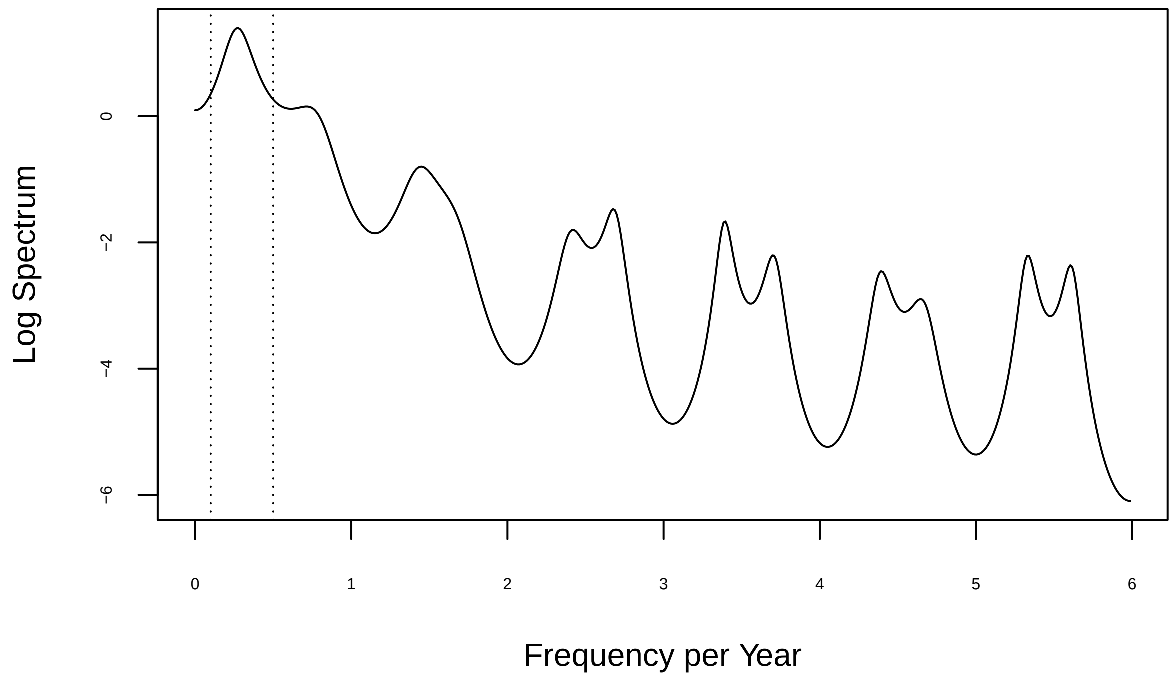
```
hpsa <- function(n,period,q,r)
{
    # hpsa
    #     gives an HP filter for seasonal data
    #   presumes trend+seas+irreg structure
    #       trend is integrated rw
    #       seas is seasonal rw
    #       irreg is wn
    #   q is snr for trend to irreg
    #   r is snr for seas to irreg

# define trend differencing matrix

delta.mat <- diag(n)
temp.mat <- 0*diag(n)
temp.mat[-1,-n] <- -2*diag(n-1)
delta.mat <- delta.mat + temp.mat
temp.mat <- 0*diag(n)
temp.mat[c(-1,-2),c(-n,-n+1)] <- 1*diag(n-2)
delta.mat <- delta.mat + temp.mat
diff.mat <- delta.mat[3:n,]

# define seasonal differencing matrix

delta.mat <- diag(n)
temp.mat <- 0*diag(n)
inds <- 0
for(t in 1:(period-1))
{
    temp.mat <- 0*diag(n)
    temp.mat[-(1+inds),-(n-inds)] <- 1*diag(n-t)
    delta.mat <- delta.mat + temp.mat
    inds <- c(inds,t)
}
sum.mat <- delta.mat[period:n,]

# define two-comp sig ex matrices

#trend.mat <- solve(diag(n) + t(diff.mat) %*% diff.mat/q)
#seas.mat <- solve(diag(n) + t(sum.mat) %*% sum.mat/r)
trend.mat <- diag(n) - t(diff.mat) %*% solve(q*diag(n-2) + diff.mat %*%
    t(diff.mat)) %*% diff.mat
seas.mat <- diag(n) - t(sum.mat) %*% solve(r*diag(n-period+1) + sum.mat %*%
    t(sum.mat)) %*% sum.mat

# define three-comp sig ex matrices

trend.filter <- solve(diag(n) - trend.mat %*% seas.mat) %*%
```

```
    trend.mat %*% (diag(n) - seas.mat)
seas.filter <- solve(diag(n) - seas.mat %*% trend.mat) %*%
    seas.mat %*% (diag(n) - trend.mat)
irreg.filter <- diag(n) - (trend.filter + seas.filter)

filters <- list(trend.filter,seas.filter,irreg.filter)
return(filters)
}

Wstarts <- read.table("Wstarts.b1",skip=2)[,2]
Wstarts <- ts(Wstarts,start = 1964,frequency=12)
n <- length(Wstarts)
q <- .0001
r <- 1
hp.filters <- hpsa(n,12,q,r)

wstarts.trend <- ts(hp.filters[[1]] %*% Wstarts,start=1964,frequency=12)
wstarts.seas <- ts(hp.filters[[2]] %*% Wstarts,start=1964,frequency=12)
wstarts.irreg <- ts(hp.filters[[3]] %*% Wstarts,start=1964,frequency=12)

par(oma=c(2,0,0,0),mar=c(2,4,2,2)+0.1,mfrow=c(4,1),cex.lab=.8)
plot(Wstarts, ylab="West Starts",xlab="",yaxt="n",xaxt="n")
axis(1,cex.axis=.5)
axis(2,cex.axis=.5)
plot(wstarts.trend,xlab="",ylab = "Trend",yaxt="n",xaxt="n")
axis(1,cex.axis=.5)
axis(2,cex.axis=.5)
plot(wstarts.seas,xlab="",ylab = "Seasonal",yaxt="n",xaxt="n")
axis(1,cex.axis=.5)
axis(2,cex.axis=.5)
plot(wstarts.irreg,xlab="",ylab = "Irregular",yaxt="n",xaxt="n")
axis(1,cex.axis=.5)
axis(2,cex.axis=.5)
mtext(text="Year",side=1,line=1,outer=TRUE)
```



- We fit (e.g. via ordinary least squares) an AR(26) to the Irregular component, and plot the log spectral density.
- The units are in terms of $2\pi/12$, so the x-axis numbers represent multiples of $\pi/6$.
- There are vertical bands for frequency between .5 and .1, corresponding to period between 2 and 10 years. This is the *business cycle* range.

```
ar.fit <- spec.ar(ts(wstarts.irreg,frequency=12),plot=FALSE)
plot(ts(log(ar.fit$spec),start=0,frequency=500/6),xlab="Frequency per Year",
    ylab="Log Spectrum",yaxt="n",xaxt="n")
axis(1,cex.axis=.5)
axis(2,cex.axis=.5)
abline(v=.5,lty=3)
abline(v=.1,lty=3)
```



Remark 6.2.2. Spectral Peaks and Oscillation Frequencies

- Higher values of the spectral density correspond to frequencies with more variability.
- A peak in the spectral density at a frequency λ corresponds to an oscillation, or cyclical effect in the process.

Fact 6.2.5. Suppression and Extraction

- Suppose $\{X_t\}$ is filtered with $\psi(B)$, so that $Y_t = \psi(B) X_t$.
- If $\psi(e^{-i\lambda}) = 0$, then $f_y(\lambda) = 0$, and λ is *suppressed*. The set of such frequencies is the *stop-band*.
- If $\psi(e^{-i\lambda}) = 1$, then $f_y(\lambda) = f_x(\lambda)$, and λ is *extracted*. The set of such frequencies is the *pass-band*.

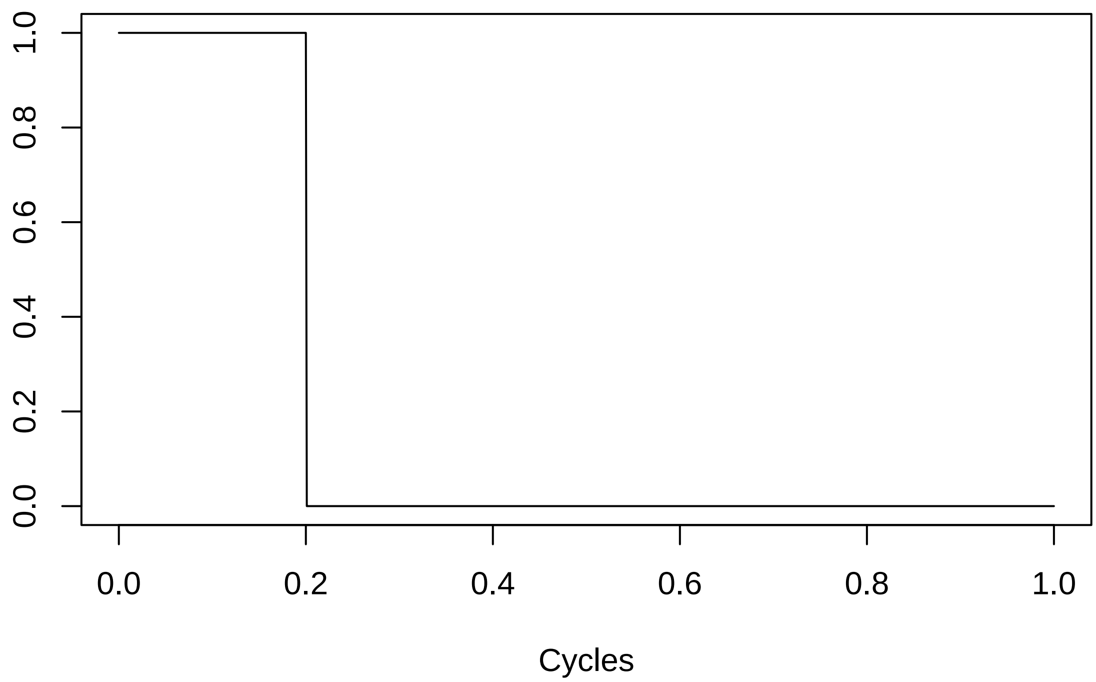
Exercise 6.22. The Ideal Low-Pass Filter

- The Ideal Low-pass is defined with frequency response function

$$\psi(e^{-i\lambda}) = \begin{cases} 1 & \text{if } |\lambda| \leq \mu \\ 0 & \text{else.} \end{cases}$$

- We plot with cut-off $\mu = \pi/5$.

```
mu <- pi/5
mesh <- 1000
lambda <- pi*seq(0,mesh)/mesh
psi.frf <- rep(0,mesh+1)
psi.frf[lambda <= mu] <- 1
plot(ts(psi.frf,start=0,frequency=mesh),ylab="",xlab="Cycles")
```



- It is hard to implement, since the filter coefficients decay slowly (and hence truncation is expensive).

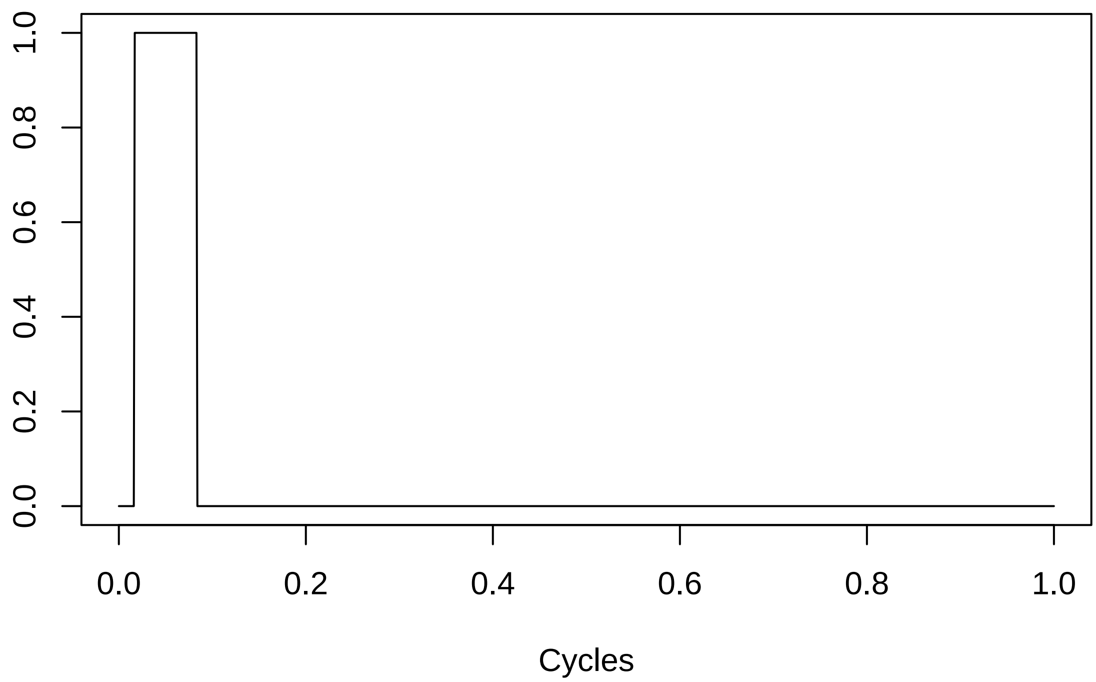
Exercise 6.23. The Ideal Band-Pass Filter

- The Ideal Band-pass is defined with frequency response function

$$\psi(e^{-i\lambda}) = \begin{cases} 1 & \text{if } \mu_1 < |\lambda| \leq \mu_2 \\ 0 & \text{else.} \end{cases}$$

- We plot with cut-offs $\mu_1 = \pi/60$ and $\mu_2 = \pi/12$.

```
mu1 <- pi/60
mu2 <- pi/12
mesh <- 1000
lambda <- pi*seq(0,mesh)/mesh
psi.frf <- rep(0,mesh+1)
psi.frf[lambda <= mu2] <- 1
psi.frf[lambda <= mu1] <- 0
plot(ts(psi.frf,start=0,frequency=mesh),ylab="",xlab="Cycles")
```



Example 6.2.7. The Hodrick-Prescott Filter

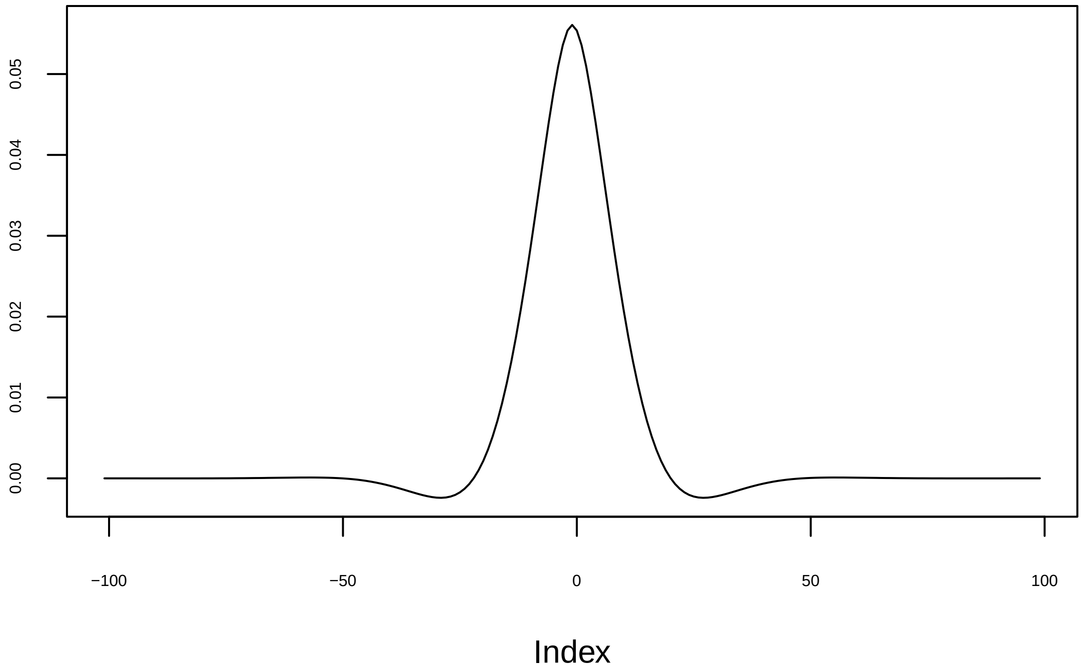
- Proposed by Whitaker in 1927, but known as Hodrick-Prescott, the filter does trend extraction.
- The frequency response function resembles an ideal low-pass.
- For the decomposition of West Housing Starts, we used a modified Hodrick-Prescott that is adapted for finite samples, and accounts for seasonality.
- The Hodrick-Prescott filter depends on a parameter $q > 0$:

$$\psi(e^{-i\lambda}) = \frac{q}{q + |1 - e^{-i\lambda}|^4}.$$

- There is a formula for the coefficients. We plot the filter coefficients with $q = 1/1600$.

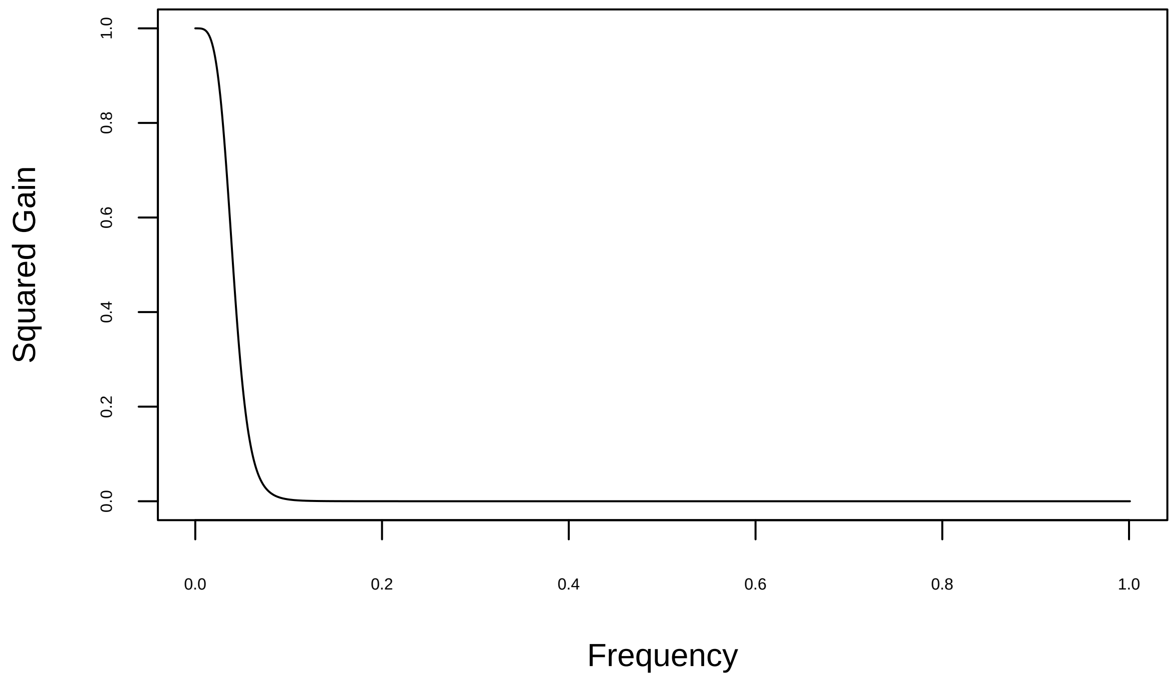
```
q <- 1/1600

s <- (2*q + 2*q^(1/2)*(q+16)^(1/2))^(1/2)
r <- (q^(1/2) + (q+16)^(1/2) + s)/4
c <- q/r^2
phi1 <- 2*(q^(1/2)-(q+16)^(1/2))/(4*r)
phi2 <- (q^(1/2)+(q+16)^(1/2) - s)/(4*r)
theta <- atan(s/4)

lags <- seq(0,100)
psi <- 2*c*r^(4-lags)*sin(theta)*(r^2*sin(theta*(1+lags)) - sin(theta*(lags-1)))
psi <- psi/((1-2*r^2*cos(2*theta)+r^4)*(r^2-1)*(1-cos(2*theta)))
psi <- c(rev(psi),psi[-1])
```

```
plot(ts(psi,start=-101),xlab="Index",ylab="",yaxt="n",xaxt="n")
axis(1,cex.axis=.5)
axis(2,cex.axis=.5)
```



- Next, we show the squared gain function.

```
grid <- 1000
lambda <- pi*seq(0,grid+1)/grid
gain <- q/(q + (2 -2*cos(lambda))^2)
sq.gain <- gain^2

plot(ts(sq.gain,start=0,frequency=grid),xlab="Frequency",ylab="Squared Gain",
    yaxt="n",xaxt="n")
axis(1,cex.axis=.5)
axis(2,cex.axis=.5)
```



Lesson 6-3: Inverse Autocovariance

- Inverse autocovariances are related to whitening filters, and can be used for model identification.

Paradigm 6.3.1. Whitening a Time Series

- A *whitening filter* reduces a time series to white noise.
- Suppose $\{X_t\}$ is stationary with positive spectral density $f(\lambda)$. Then a whitening filter $\psi(B)$ has squared gain function

$$\left|\psi(e^{-i\lambda})\right|^2 \propto 1/f(\lambda).$$

- So $\psi(B)$ depends on the spectral density of the time series we are whitening. We can find $\psi(B)$ causal, i.e., $\psi(z)$ is a power series.

Definition 6.3.2

- A weakly stationary process is *invertible* if its spectral density is positive.
- By Corollary 6.1.17, the process has an AR(∞) representation, so we can "invert" the time series into a white noise.
- For prediction problems, a process should be invertible.

Example 6.3.3. Prediction of an MA(1) from an Infinite Past

- Let $\{X_t\}$ be an *invertible* MA(1) process with MA polynomial $1 + \theta_1 z$.
- Suppose we want to forecast 1-step ahead: we seek $\widehat{X}_{t+1} = P_{\overline{\text{sp}}\{X_s, s \leq t\}}[X_{t+1}]$.
- This forecast is a causal filter: $\widehat{X}_{t+1} = \sum_{j \geq 0} \psi_j X_{t-j}$, with ψ_j to be determined from normal equations.

- The normal equations give us, for any $h \geq 0$:

$$\gamma(h+1) = \mathrm{Cov}[X_{t+1}, X_{t-h}] = \mathrm{Cov}[\widehat{X}_{t+1}, X_{t-h}] = \sum_{j\geq 0} \psi_j \mathrm{Cov}[X_{t-j}, X_{t-h}] = \sum_{j\geq 0} \psi_j \gamma(h-j).$$

- To solve this, rewrite the right hand side using Fourier inversion:

$$\sum_{j\geq 0} \psi_j \gamma(h-j) = \sum_{j\geq 0} \psi_j (2\pi)^{-1} \int_{-\pi}^{\pi} e^{i\lambda(h-j)} f(\lambda) d\lambda = (2\pi)^{-1} \int_{-\pi}^{\pi} e^{i\lambda h} \sum_{j\geq 0} \psi_j e^{-i\lambda j} f(\lambda) d\lambda = (2\pi)^{-1} \int_{-\pi}^{\pi} e^{i\lambda h} \psi(e^{-i\lambda}) f(\lambda) d\lambda$$

- Recall that $f(\lambda) = \sigma^2 |1 + \theta_1 e^{-i\lambda}|^2$. The invertibility assumption means that $1 + \theta_1 e^{-i\lambda}$ is non-zero for all λ.
- Claim: the prediction filter has frequency response function

$$\psi(e^{-i\lambda}) = \frac{\theta_1}{1 + \theta_1 e^{-i\lambda}},$$

  which is well-defined by the invertibility assumption.
- To prove this claim, we plug in and check! The right hand side becomes

$$\sum_{j\geq 0} \psi_j \gamma(h-j) = (2\pi)^{-1} \int_{-\pi}^{\pi} e^{i\lambda h} \frac{\theta_1}{1 + \theta_1 e^{-i\lambda}} f(\lambda) d\lambda = (2\pi)^{-1} \int_{-\pi}^{\pi} e^{i\lambda h} \theta_1 (1 + \theta_1 e^{i\lambda}) \sigma^2 d\lambda = 1_{\{h=0\}} \theta_1 \sigma^2.$$

  This is the same as $\gamma(h+1)$ for $h \geq 0$, so the claim is true!
- To get the coefficients:

$$\psi(z) = \theta_1 (1 + \theta_1 z)^{-1} = \theta_1 \sum_{j\geq 0} (-\theta_1)^j z^j.$$

  So $\psi_j = \theta_1^{j+1} (-1)^j$.

Definition 6.3.4

- Suppose $\{X_t\}$ is an invertible weakly stationary time series with autocovariance $\gamma(h)$. Then the *inverse autocovariance* is the sequence $\xi(k)$ such that

$$\sum_{k=-\infty}^{\infty} \gamma(k) \xi(j-k) = 1_{\{j=0\}}.$$

- The inverse autocorrelation is $\zeta(k) = \xi(k)/\xi(0)$.
- We can compute the inverse autocovariance from the spectral density:

$$\xi(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{i\lambda k} \frac{1}{f(\lambda)} d\lambda.$$

Example 6.3.6. The Inverse Autocovariance of an MA(1)

- Consider an MA(1) process with $\theta_1 \in (-1, 1)$, which implies it is invertible.
- So the inverse autocovariance is

$$\xi(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{i\lambda k} \sigma^{-2} |1 + \theta_1 e^{-i\lambda}|^{-2} d\lambda.$$

- This resembles the autocovariance of an AR(1), with parameter $\phi_1 = -\theta_1$, and input variance σ^{-2}.
- So by using the formula for AR(1) autocovariance, we find

$$\xi(k) = \sigma^{-2} \frac{(-\theta_1)^{|k|}}{1 - \theta_1^2}.$$

Exercise 6.34. Inverse Autocovariances of an AR(1)

- Consider the AR(1) process with $\phi(z) = 1 - \phi_1 z$. What are the inverse autocovariances?
- So the inverse autocovariance is

$$\xi(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{i\lambda k} \sigma^2 {|1 - \phi_1 e^{-i\lambda}|}^2 d\lambda.$$

- This resembles the autocovariance of an MA(1), with parameter $\theta_1 = -\phi_1$, and input variance σ^2.
- So by using the formula for MA(1) autocovariance, we find

$$\xi(k) = \sigma^2 \begin{cases} 1 + \phi_1^2 & \text{if } k = 0 \\ -\phi_1 & \text{if } k = \pm 1 \\ 0 & \text{if } |k| > 1. \end{cases}$$

Example 6.3.8. Inverse ACF and Optimal Interpolation

- Suppose $\{X_t\}$ is stationary, mean zero, and invertible.
- Suppose that X_0 is missing. What is the optimal estimator?
- We seek $\widehat{X_0} = P_{\overline{\text{sp}}\{X_j, j \neq 0\}}[X_0]$, which is a linear filter $\psi(B)$ of the data, such that $\psi_0 = 0$.
- Claim: $\psi_j = -\zeta(j)$ for $j \neq 0$, and $\psi_0 = 0$.
- Proof: check the normal equations. First $X_0 - \widehat{X}_0 = \sum_j \zeta(j) X_{-j}$, since $\zeta(0) = 1$. The covariance of this with any X_{-k} for $k \neq 0$ is

$$\text{Cov}[X_0 - \widehat{X}_0, X_{-k}] = \sum_j \zeta(j) \gamma(k - j),$$

which is zero (since $k \neq 0$) by definition of inverse autocovariance. This verifies the normal equations!

Lesson 6-4: Toeplitz Matrices

- We discuss a decomposition of Toeplitz matrices, with connections to the spectral density.
- This is useful for model fitting and prediction.

Fact 6.4.2. Spectral Representation of a Symmetric Matrix

- Let A^* denote the conjugate transpose of a matrix A.
- A matrix A is Hermitian if $A^* = A$.
- A matrix U is unitary if $U^{-1} = U^*$.
- For any Hermitian A, there exists unitary U such that $A = U D U^*$, where D is diagonal with real entries.
- The columns of U are the eigenvectors of A, and D has the eigenvalues of A.

Definition 6.4.3. Fourier Frequencies

- For any n, the *Fourier frequencies* are defined as $\lambda_\ell = 2\pi \ell / n$, for $[n/2] - n + 1 \leq \ell \leq [n/2]$.
- When n is even, this excludes the frequencies π and $-\pi$ both being in the set, since these are redundant.
- We define an $n \times n$-dimensional matrix Q, whose entries are complex exponentials evaluated at Fourier frequencies:

$$Q_{jk} = n^{-1/2} e^{ij\lambda_{[n/2]-n+k}}.$$

- The matrix Q is unitary, and can be used in an approximation result for Toeplitz matrices.

Theorem 6.4.5. Spectral Decomposition of Toeplitz Covariance Matrices

- Let Γ_n denote the autocovariance matrix of a sample of size n from a stationary time series. Suppose the autocovariances $\gamma(k)$ are absolutely summable. Then

$$\Gamma_n \approx Q \Lambda Q^*,$$

with Λ diagonal with entries $f(\lambda_{[n/2]-n+k})$. The approximation $\approx$ means that the difference entry-by-entry tends to zero as n tends to ∞.
- It can also be shown that $\Lambda \approx Q^* \Gamma_n Q$.
- If the process is invertible, then $\Gamma_n^{-1} \approx Q \Lambda^{-1} Q^*$ as well.

Remark 6.4.11. Positive Spectral Density

Since the eigenvalues of a symmetric non-negative definite matrix Γ_n are real and non-negative, we can show that the spectral density of a stationary process must be non-negative.

Exercise 6.43. Eigenvalues of an MA(1) Toeplitz Matrix.

- Consider an MA(1) with parameter $\theta = .8$.
- Compute the eigenvalues of Γ_n for various n, and compare to the spectral density evaluated at the Fourier frequencies.

```
theta <- .8

n <- 10
lambda <- 2*pi*seq(0,n-1)/n
Gamma <- toeplitz(c(1+theta^2, theta, rep(0,n-2)))
eigen(Gamma)$values
```

```
##  [1] 3.1751888 2.9860057 2.6877772 2.3046640 1.8677037 1.4122963 0.9753360
##  [8] 0.5922228 0.2939943 0.1048112
```

```
rev(sort(1+theta^2 + 2*theta*cos(lambda)))
```

```
##  [1] 3.2400000 2.9344272 2.9344272 2.1344272 2.1344272 1.1455728 1.1455728
##  [8] 0.3455728 0.3455728 0.0400000
```

```
n <- 20
lambda <- 2*pi*seq(0,n-1)/n
Gamma <- toeplitz(c(1+theta^2, theta, rep(0,n-2)))
eigen(Gamma)$values
```

```
##  [1] 3.22212932 3.16891649 3.08155019 2.96198204 2.81288299 2.63758368
##  [7] 2.44000000 2.22454564 1.99603349 1.75956815 1.52043185 1.28396651
## [13] 1.05545436 0.84000000 0.64241632 0.46711701 0.31801796 0.19844981
## [19] 0.11108351 0.05787068
```

```
rev(sort(1+theta^2 + 2*theta*cos(lambda)))
```

```
##  [1] 3.2400000 3.1616904 3.1616904 2.9344272 2.9344272 2.5804564 2.5804564
##  [8] 2.1344272 2.1344272 1.6400000 1.6400000 1.1455728 1.1455728 0.6995436
## [15] 0.6995436 0.3455728 0.3455728 0.1183096 0.1183096 0.0400000
```

```
n <- 30
lambda <- 2*pi*seq(0,n-1)/n
Gamma <- toeplitz(c(1+theta^2, theta, rep(0,n-2)))
eigen(Gamma)$values
```

```
##  [1] 3.23179092 3.20724791 3.16662281 3.11033250 3.03895459 2.95322151
##  [7] 2.85401300 2.74234707 2.61936957 2.48634242 2.34463064 2.19568840
## [13] 2.04104405 1.88228444 1.72103867 1.55896133 1.39771556 1.23895595
## [19] 1.08431160 0.93536936 0.79365758 0.66063043 0.53765293 0.42598700
## [25] 0.32677849 0.24104541 0.16966750 0.11337719 0.07275209 0.04820908
```

```r
rev(sort(1+theta^2 + 2*theta*cos(lambda)))
```

```
##  [1] 3.24000000 3.20503616 3.20503616 3.10167273 3.10167273 2.93442719
##  [7] 2.93442719 2.71060897 2.71060897 2.44000000 2.44000000 2.13442719
## [13] 2.13442719 1.80724554 1.80724554 1.47275446 1.47275446 1.14557281
## [19] 1.14557281 0.84000000 0.84000000 0.56939103 0.56939103 0.34557281
## [25] 0.34557281 0.17832727 0.17832727 0.07496384 0.07496384 0.04000000
```

Exercise 6.45. Eigenvalues of an MA(1) Inverse Toeplitz Matrix.

- Consider an MA(1) with parameter $\theta = .8$.
- Compute the eigenvalues of Γ_n^{-1} for various n, and compare to the reciprocal spectral density evaluated at the Fourier frequencies.

```r
theta <- .8

n <- 10
lambda <- 2*pi*seq(0,n-1)/n
Gamma <- toeplitz(c(1+theta^2, theta, rep(0,n-2)))
eigen(solve(Gamma))$values
```

```
##  [1] 9.5409612 3.4014259 1.6885536 1.0252877 0.7080667 0.5354168 0.4339027
##  [8] 0.3720547 0.3348955 0.3149419
```

```r
1/sort(1+theta^2 + 2*theta*cos(lambda))
```

```
##  [1] 25.0000000  2.8937462  2.8937462  0.8729257  0.8729257  0.4685098
##  [7]  0.4685098  0.3407820  0.3407820  0.3086420
```

```r
n <- 20
lambda <- 2*pi*seq(0,n-1)/n
Gamma <- toeplitz(c(1+theta^2, theta, rep(0,n-2)))
eigen(solve(Gamma))$values
```

```
##  [1] 17.2799081  9.0022362  5.0390574  3.1444765  2.1407913  1.5566230
##  [7]  1.1904762  0.9474593  0.7788365  0.6577079  0.5683213  0.5009936
## [13]  0.4495300  0.4098361  0.3791349  0.3555071  0.3376118  0.3245120
## [19]  0.3155653  0.3103538
```

```r
1/sort(1+theta^2 + 2*theta*cos(lambda))
```

```
##  [1] 25.0000000  8.4524013  8.4524013  2.8937462  2.8937462  1.4295035
##  [7]  1.4295035  0.8729257  0.8729257  0.6097561  0.6097561  0.4685098
## [13]  0.4685098  0.3875283  0.3875283  0.3407820  0.3407820  0.3162865
## [19]  0.3162865  0.3086420
```

```r
n <- 30
lambda <- 2*pi*seq(0,n-1)/n
Gamma <- toeplitz(c(1+theta^2, theta, rep(0,n-2)))
eigen(solve(Gamma))$values
```

```
##  [1] 20.7429793 13.7453088  8.8201163  5.8938806  4.1485958  3.0601769
##  [7]  2.3474895  1.8599359  1.5137056  1.2599892  1.0690964  0.9222441
## [13]  0.8071312  0.7154532  0.6414527  0.5810445  0.5312693  0.4899453
## [19]  0.4554380  0.4265064  0.4021972  0.3817713  0.3646511  0.3503838
## [25]  0.3386133  0.3290605  0.3215090  0.3157938  0.3117938  0.3094260
```

```r
1/sort(1+theta^2 + 2*theta*cos(lambda))
```

```
##  [1] 25.0000000 13.3397651 13.3397651  5.6076674  5.6076674  2.8937462
##  [7]  2.8937462  1.7562623  1.7562623  1.1904762  1.1904762  0.8729257
## [13]  0.8729257  0.6789998  0.6789998  0.5533282  0.5533282  0.4685098
## [19]  0.4685098  0.4098361  0.4098361  0.3689208  0.3689208  0.3407820
## [25]  0.3407820  0.3224067  0.3224067  0.3120090  0.3120090  0.3086420
```

Lesson 6-5: Partial Autocorrelation

- Recall from linear models that partial correlation allows us to explore the relationship between a dependent variable and a covariate, while accounting for other covariates.
- We apply this concept to stationary time series, where we look at the relationship between time present and time past, while accounting for the in-between times.

Definition 6.5.1.

- The partial correlation function (PACF) of stationary time series $\{X_t\}$ is a sequence $\kappa(h)$ defined by $\kappa(1) = \text{Corr}[X_1, X_0]$ and
$$\kappa(h) = \text{Corr}[X_h, X_0 | X_1, \ldots, X_{h-1}]$$
when $h \geq 2$. The conditioning stands for projection (of the demeaned time series) on the random variables.
- What does this mean? Linearly predict X_h from $X_1, \ldots, X_{h-1}$, and call that $\widehat{X}_h$. Also linearly predict X_0 from $X_1, \ldots, X_{h-1}$, and call that $\widehat{X}_0$. Then $\kappa(h)$ is the correlation of the prediction errors:
$$\kappa(h) = \text{Corr}[X_h - \widehat{X}_h, X_0 - \widehat{X}_0].$$
- Because of stationarity, we could also write
$$\kappa(h) = \text{Corr}[X_{t+h}, X_t | X_{t+1}, \ldots, X_{t+h-1}]$$
for any t.

Example 6.5.2. Partial Autocorrelation of an AR(p) Process

- Suppose $\{X_t\}$ is an AR(1), where $\phi(z) = 1 - \phi_1 z$. Then $\kappa(1) = \phi_1$.
- Also for $h \geq 2$, $\widehat{X}_h = \phi_1 X_{h-1}$ and $\widehat{X}_0 = \phi_1 X_1$ (follows from the normal equations).
- The prediction errors are then
$$\begin{aligned} X_h - \widehat{X}_h &= Z_h \\ X_0 - \widehat{X}_0 &= (1 - \phi_1^2) X_0 - \phi_1 Z_1. \end{aligned}$$
These are uncorrelated for $h \geq 2$. So $\kappa(h) = 0$.
- The argument can be generalized to the case of an AR(p), for which $\kappa(h) = 0$ when $h > p$.

Proposition 6.5.5.

If $\{X_t\}$ has mean zero, the PACF at lag h is given by solving the Yule-Walker equations of order h, and taking the last coefficient, i.e., letting $\underline{e}_h$ denote the length h unit vector with one in the last position,
$$\kappa(h) = \underline{e}_h' \, \Gamma_h^{-1} \, \underline{\gamma}_h.$$

Exercise 6.54. PACF of MA(q)

- We use the formula of Proposition 6.5.5 to compute the PACF for the MA(3) process with $\theta(z) = 1 + .4z + .2z^2 - .3z^3$.
- First we load the ARMAauto.r function from earlier notebooks.

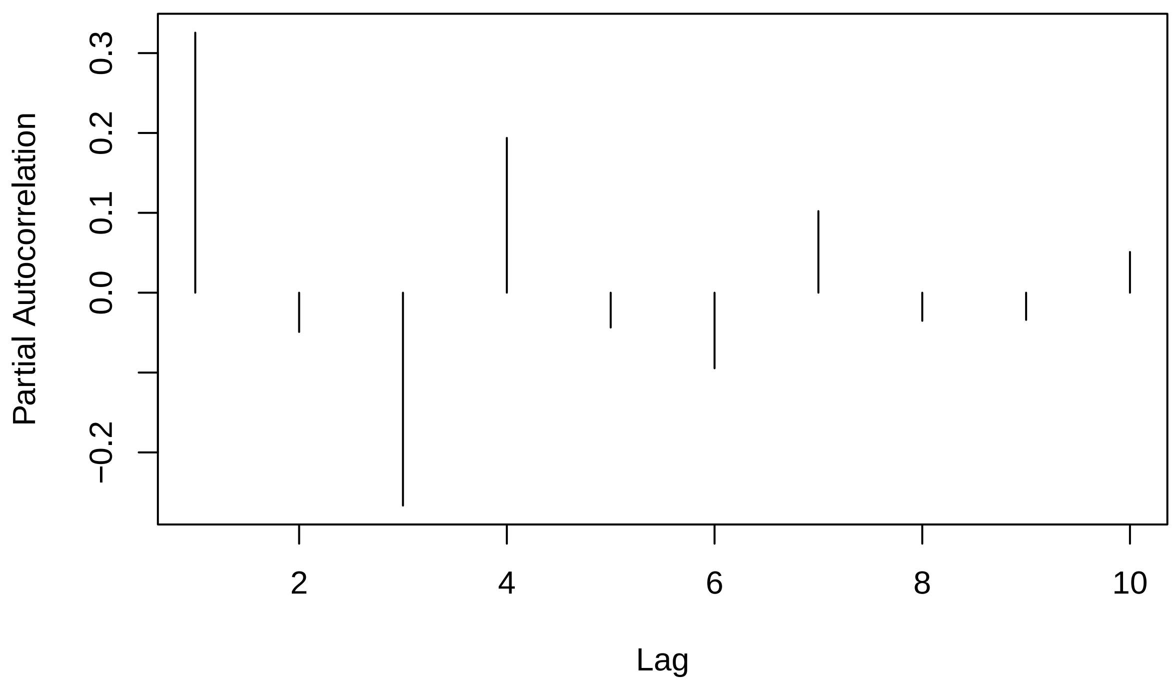
```r
polymult <- function(a,b) {
bb <- c(b,rep(0,length(a)-1))
B <- toeplitz(bb)
B[lower.tri(B)] <- 0
aa <- rev(c(a,rep(0,length(b)-1)))
prod <- B %*% matrix(aa,length(aa),1)
return(rev(prod[,1]))
}

ARMAauto <- function(phi,theta,maxlag)
{
    p <- length(phi)
    q <- length(theta)
    gamMA <- polymult(c(1,theta),rev(c(1,theta)))
    gamMA <- gamMA[(q+1):(2*q+1)]
    if (p > 0)
    {
        Amat <- matrix(0,nrow=(p+1),ncol=(2*p+1))
        for(i in 1:(p+1))
        {
            Amat[i,i:(i+p)] <- c(-1*rev(phi),1)
        }
        Amat <- cbind(Amat[,(p+1)],as.matrix(Amat[,(p+2):(2*p+1)]) +
            t(matrix(apply(t(matrix(Amat[,1:p],p+1,p)),2,rev),p,p+1)))
        Bmat <- matrix(0,nrow=(q+1),ncol=(p+q+1))
        for(i in 1:(q+1))
        {
            Bmat[i,i:(i+p)] <- c(-1*rev(phi),1)
        }
        Bmat <- t(matrix(apply(t(Bmat),2,rev),p+q+1,q+1))
        Bmat <- matrix(apply(Bmat,2,rev),q+1,p+q+1)
        Bmat <- Bmat[,1:(q+1)]
        Binv <- solve(Bmat)
        gamMix <- Binv %*% gamMA
        if (p <= q) { gamMix <- matrix(gamMix[1:(p+1),],p+1,1)
            } else gamMix <- matrix(c(gamMix,rep(0,(p-q))),p+1,1)
        gamARMA <- solve(Amat) %*% gamMix
    } else gamARMA <- gamMA[1]

    gamMA <- as.vector(gamMA)
    if (maxlag <= q) gamMA <- gamMA[1:(maxlag+1)] else gamMA <- c(gamMA,rep(0,(maxlag-q)))
    gamARMA <- as.vector(gamARMA)
    if (maxlag <= p) gamARMA <- gamARMA[1:(maxlag+1)] else {
    for(k in 1:(maxlag-p))
    {
        len <- length(gamARMA)
        acf <- gamMA[p+1+k]
        if (p > 0) acf <- acf + sum(phi*rev(gamARMA[(len-p+1):len]))
        gamARMA <- c(gamARMA,acf)
    } }
    return(gamARMA)
}
```

- Then we implement Proposition 6.5.5.

```r
armapq.pacf <- function(ar.coefs,ma.coefs,max.lag)
{
    gamma <- ARMAauto(ar.coefs,ma.coefs,max.lag)
    kappa <- NULL
    for(k in 1:max.lag)
    {
        new.kappa <- solve(toeplitz(gamma[1:k]),gamma[2:(k+1)])[k]
        kappa <- c(kappa,new.kappa)
    }
    return(kappa)
}
```

- Then we apply to the given MA(3) process.

```r
ma.coefs <- c(.4,.2,-.3)
kappa <- armapq.pacf(NULL,ma.coefs,10)
plot(ts(kappa,start=1),xlab="Lag",ylab="Partial Autocorrelation",
     ylim=c(min(kappa),max(kappa)),type="h")
```

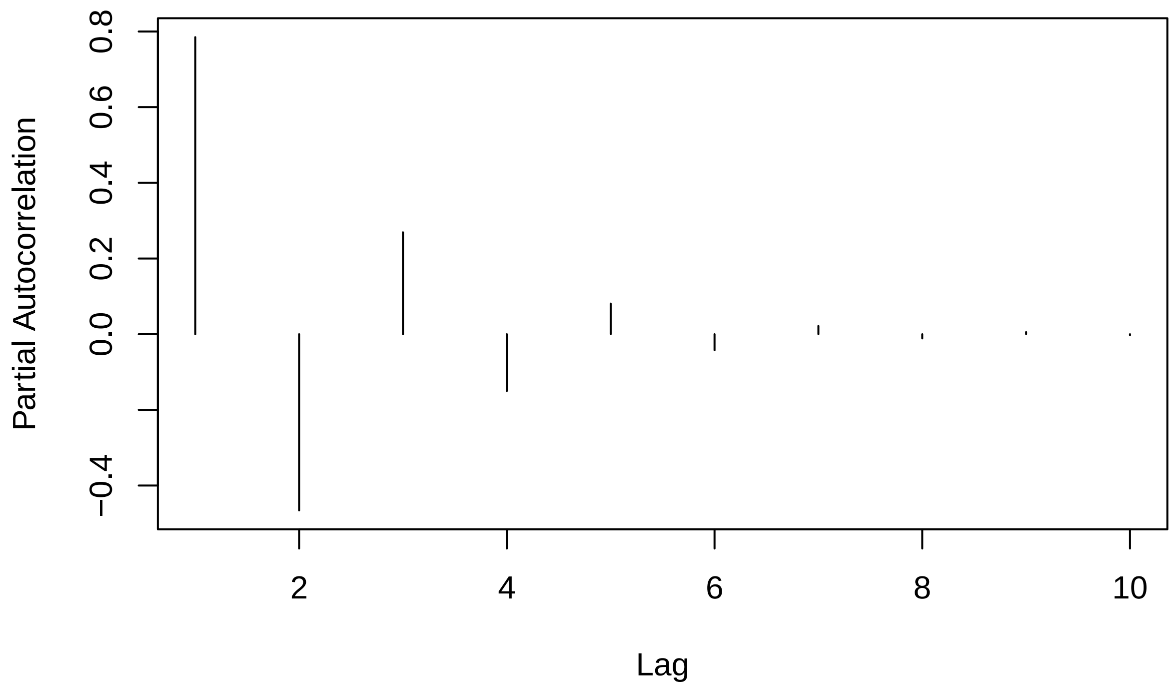


Exercise 6.55. PACF pf ARMA(p,q)

- Compute the PACF of Example 5.5.7, which is an ARMA(1,2) with $\phi(z) = 1 - .5z$ and $\theta(z) = 1 + (5/6)z + (1/6)z^2$.

```r
phi1 <- .5
theta1 <- 5/6
theta2 <- 1/6
```

```r
kappa <- armapq.pacf(phi1,c(theta1,theta2),10)
plot(ts(kappa,start=1),xlab="Lag",ylab="Partial Autocorrelation",
     ylim=c(min(kappa),max(kappa)),type="h")
```



Lesson 6-6: AR and MA Identification

- How do we determine a model to be fitted? AR, MA, ARMA, or something else?
- How do we determine the order of the model?

Paradigm 6.6.1. Characterizing AR and MA Processes

- The autocorrelation function (ACF), inverse autocorrelation (IACF), and partial autocorrelation function (PACF) have distinctive behavior for AR and MA processes.

ACF

- For an MA(q) process, the ACF truncates at lag q, i.e., $\gamma(h) = 0$ if $|h| > q$. However, it is possible for $\gamma(h) = 0$ for $0 < h < q$ as well.
- For an AR(p) process, or for an ARMA(p,q) process (with $p > 0$), the ACF decays at geometric rate. The correlations can oscillate, but they are bounded by some $Cr^{|h|}$ for $0 < r < 1$ and $C > 0$.

IACF

- Generalize Exercise 6.34 to get IACF behavior for AR(p) processes.
- For an AR(p) process, the IACF truncates at lag p, i.e., $\zeta(h) = 0$ if $|h| > p$.
- for an MA(q) process, or for an ARMA(p,q) process (with $q > 0$), the IACF decays at geometric rate.

PACF

- For an AR(p) process, the PACF truncates at lag p, i.e., $\kappa(h) = 0$ if $|h| > p$.
- for an MA(q) process, or for an ARMA(p,q) process (with $q > 0$), the PACF decays at geometric rate.

Finding Truncation

We can plot estimates of the ACF and PACF, and see if there is a lag cut-off where one or the other seems to negligible.

Example 6.6.2. MA(3) Identification

- Suppose we observe the ACF, IACF, and PACF of a process.

```
polymult <- function(a,b) {
bb <- c(b,rep(0,length(a)-1))
B <- toeplitz(bb)
B[lower.tri(B)] <- 0
aa <- rev(c(a,rep(0,length(b)-1)))
prod <- B %*% matrix(aa,length(aa),1)
return(rev(prod[,1]))
}

ARMAauto <- function(phi,theta,maxlag)
{
    p <- length(phi)
    q <- length(theta)
    gamMA <- polymult(c(1,theta),rev(c(1,theta)))
    gamMA <- gamMA[(q+1):(2*q+1)]
    if (p > 0)
    {
        Amat <- matrix(0,nrow=(p+1),ncol=(2*p+1))
        for(i in 1:(p+1))
        {
            Amat[i,i:(i+p)] <- c(-1*rev(phi),1)
        }
        Amat <- cbind(Amat[,(p+1)],as.matrix(Amat[,(p+2):(2*p+1)]) +
            t(matrix(apply(t(matrix(Amat[,1:p],p+1,p)),2,rev),p,p+1)))
        Bmat <- matrix(0,nrow=(q+1),ncol=(p+q+1))
        for(i in 1:(q+1))
        {
            Bmat[i,i:(i+p)] <- c(-1*rev(phi),1)
        }
        Bmat <- t(matrix(apply(t(Bmat),2,rev),p+q+1,q+1))
        Bmat <- matrix(apply(Bmat,2,rev),q+1,p+q+1)
        Bmat <- Bmat[,1:(q+1)]
        Binv <- solve(Bmat)
        gamMix <- Binv %*% gamMA
        if (p <= q) { gamMix <- matrix(gamMix[1:(p+1),],p+1,1)
            } else gamMix <- matrix(c(gamMix,rep(0,(p-q))),p+1,1)
        gamARMA <- solve(Amat) %*% gamMix
    } else gamARMA <- gamMA[1]

    gamMA <- as.vector(gamMA)
    if (maxlag <= q) gamMA <- gamMA[1:(maxlag+1)] else gamMA <- c(gamMA,rep(0,(maxlag-q)))
```

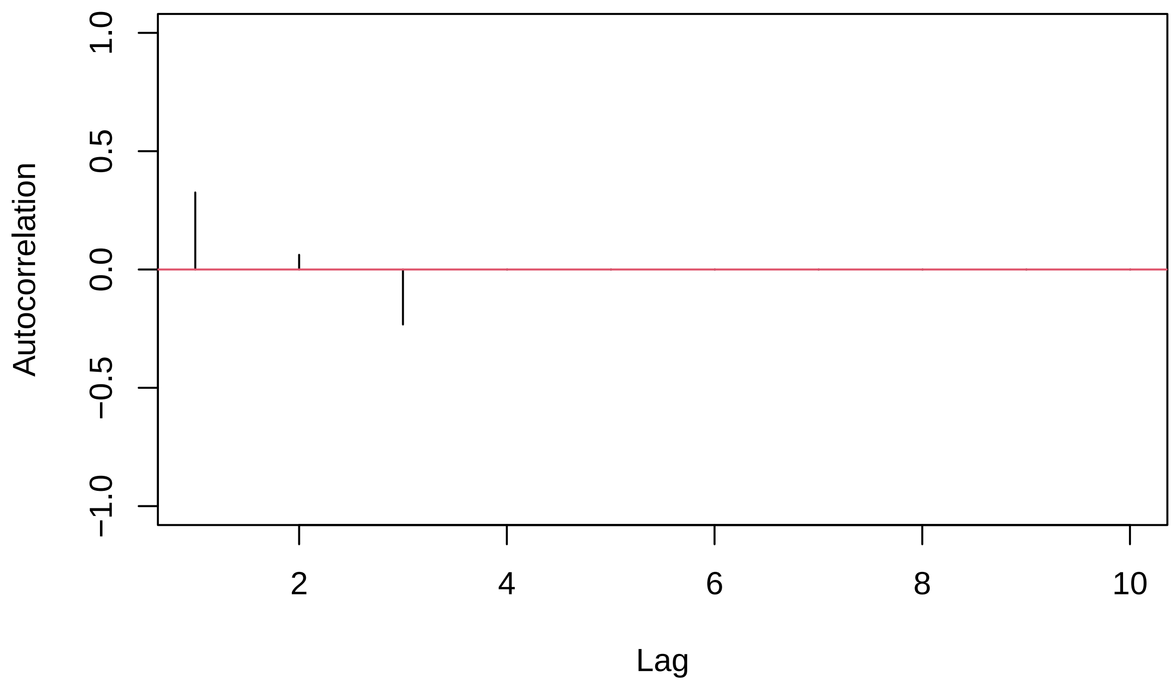
```
    gamARMA <- as.vector(gamARMA)
    if (maxlag <= p) gamARMA <- gamARMA[1:(maxlag+1)] else {
    for(k in 1:(maxlag-p))
    {
        len <- length(gamARMA)
        acf <- gamMA[p+1+k]
        if (p > 0) acf <- acf + sum(phi*rev(gamARMA[(len-p+1):len]))
        gamARMA <- c(gamARMA,acf)
    } }
    return(gamARMA)
}

armapq.pacf <- function(ar.coefs,ma.coefs,max.lag)
{
    gamma <- ARMAauto(ar.coefs,ma.coefs,max.lag)
    kappa <- NULL
    for(k in 1:max.lag)
    {
        new.kappa <- solve(toeplitz(gamma[1:k]),gamma[2:(k+1)])[k]
        kappa <- c(kappa,new.kappa)
    }
    return(kappa)
}
```

- We construct and plot these functions for an MA(3) process.

```
ma.coefs <- c(.4,.2,-.3)
gamma <- ARMAauto(NULL,ma.coefs,10)
rho <- gamma/gamma[1]
xi <- ARMAauto(-1*ma.coefs,NULL,10)
zeta <- xi/xi[1]
kappa <- armapq.pacf(NULL,ma.coefs,10)
```

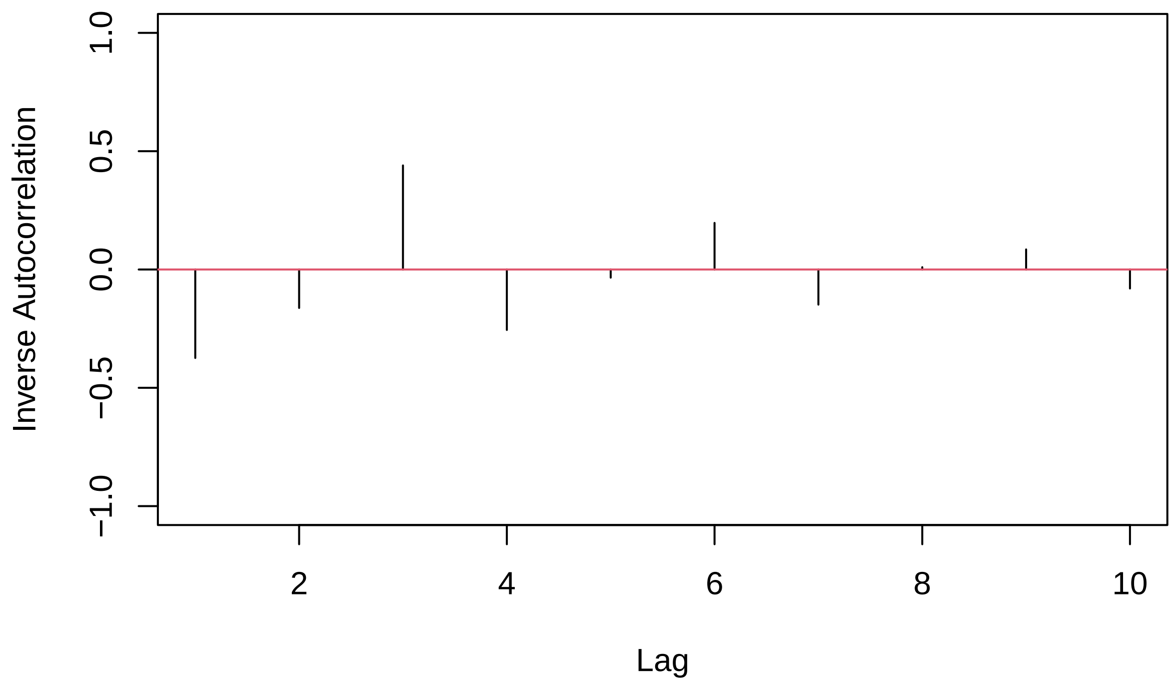
- The ACF plot. We start at lag 1, since the lag 0 value is always zero.

```
plot(ts(rho[-1],start=1),xlab="Lag",ylab="Autocorrelation",
     ylim=c(-1,1),type="h")
abline(h=0,col=2)
```



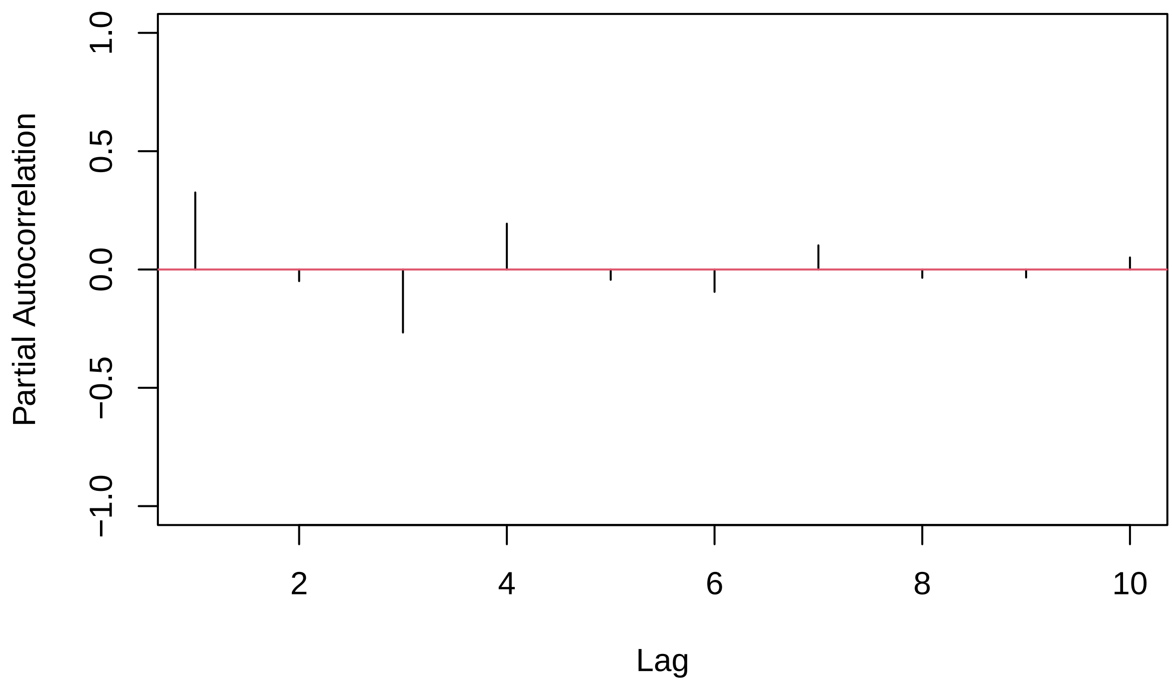
- The IACF plot. We start at lag 1, since the lag 0 value is always zero.

```
plot(ts(zeta[-1],start=1),xlab="Lag",ylab="Inverse Autocorrelation",
     ylim=c(-1,1),type="h")
abline(h=0,col=2)
```



- The PACF plot. We start at lag 1, since the lag 0 value is not defined.

```
plot(ts(kappa,start=1),xlab="Lag",ylab="Partial Autocorrelation",
     ylim=c(-1,1),type="h")
abline(h=0,col=2)
```



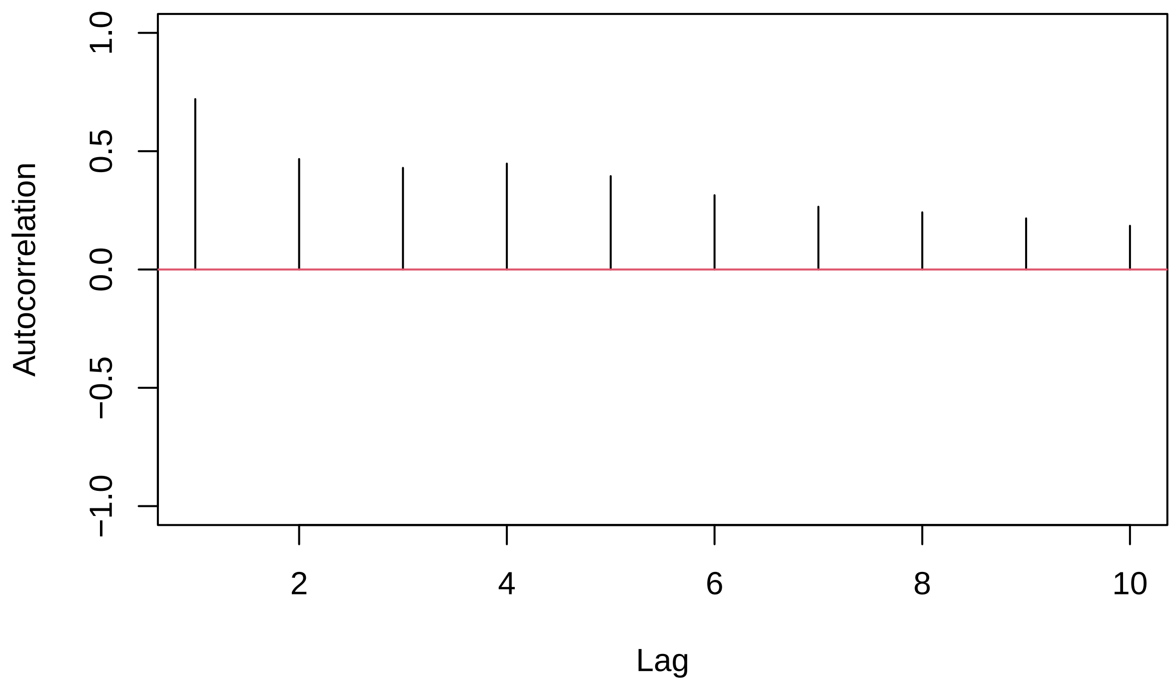
Example 6.6.3. AR(4) Identification

- Suppose we observe the ACF, IACF, and PACF of a process.
- We construct and plot these functions for an AR(4) process.

```
ar.coefs <- c(.8,-.3,.2,.1)
gamma <- ARMAauto(ar.coefs,NULL,10)
rho <- gamma/gamma[1]
xi <- ARMAauto(NULL,-1*ar.coefs,10)
zeta <- xi/xi[1]
kappa <- armapq.pacf(ar.coefs,NULL,10)
```

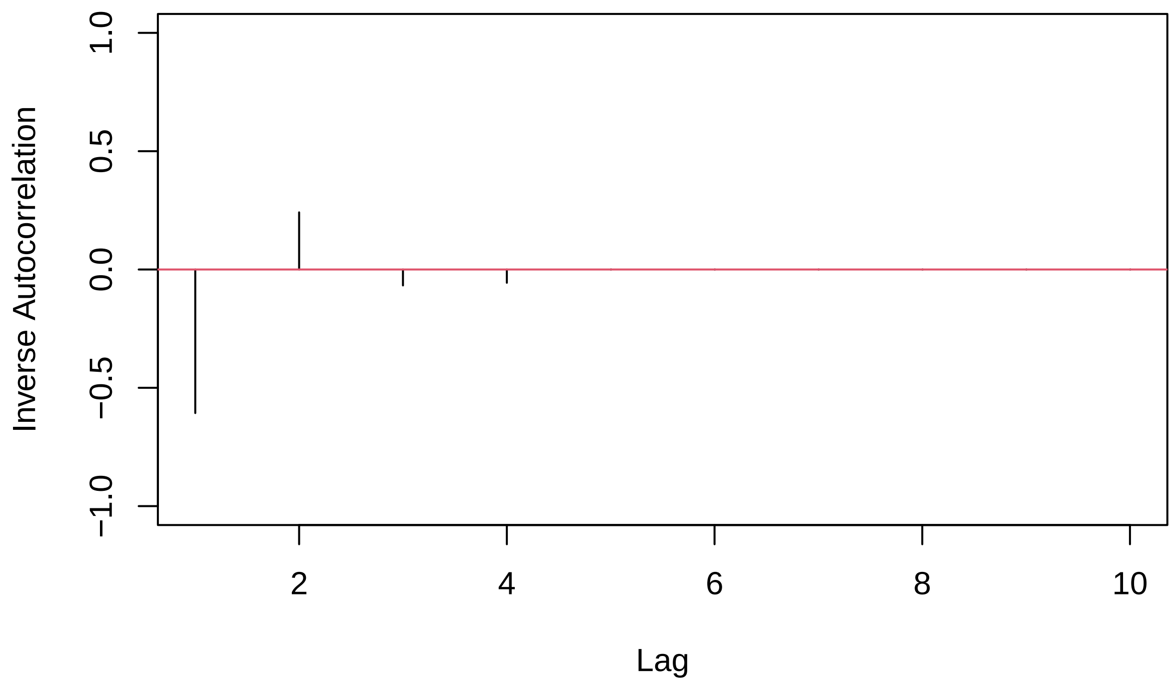
- The ACF plot. We start at lag 1, since the lag 0 value is always zero.

```
plot(ts(rho[-1],start=1),xlab="Lag",ylab="Autocorrelation",
     ylim=c(-1,1),type="h")
abline(h=0,col=2)
```



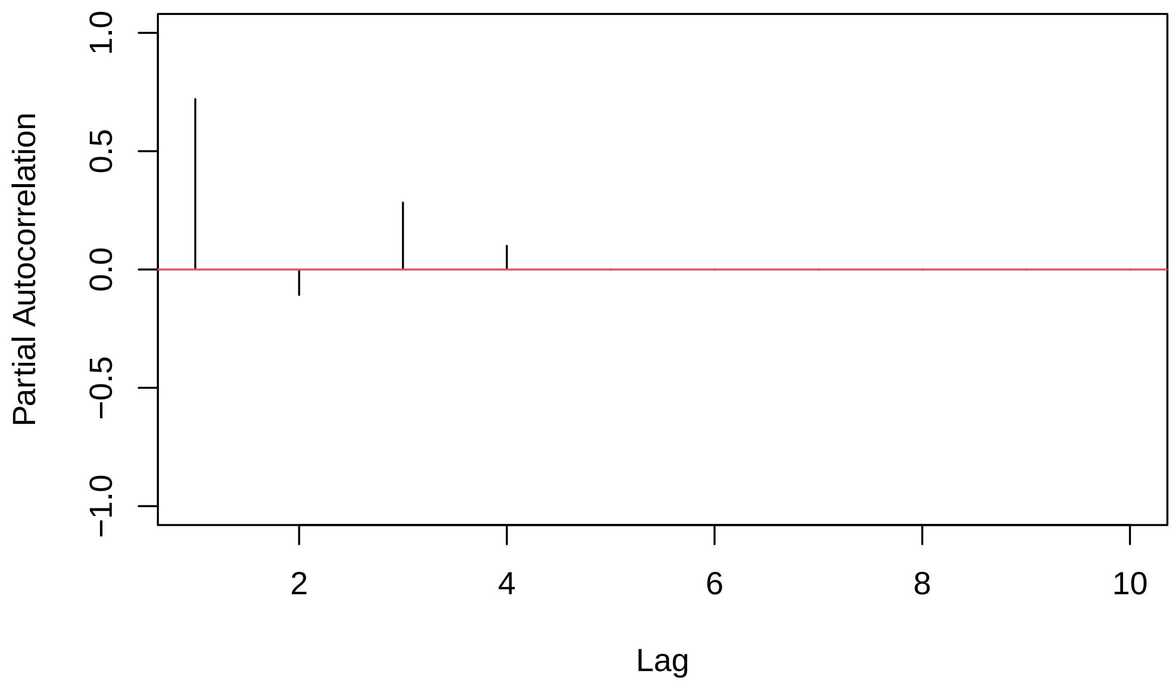
- The IACF plot. We start at lag 1, since the lag 0 value is always zero.

```
plot(ts(zeta[-1],start=1),xlab="Lag",ylab="Inverse Autocorrelation",
     ylim=c(-1,1),type="h")
abline(h=0,col=2)
```



- The PACF plot. We start at lag 1, since the lag 0 value is not defined.

```
plot(ts(kappa,start=1),xlab="Lag",ylab="Partial Autocorrelation",
     ylim=c(-1,1),type="h")
abline(h=0,col=2)
```



Paradigm 6.6.7. Identification by Whitening

- Suppose we apply some filter $\psi(B)$ to the data $\{X_t\}$, and the output appears to be white noise $\{Z_t\}$ (e.g., we ran some statistical tests of serial independence).
- $\psi(B)$ is called a *whitening filter*.
- We infer that $X_t = \psi(B)^{-1} Z_t$, which gives a model for $\{X_t\}$.
- So we can try out classes of filters $\psi(B)$, attempt to whiten the data, and deduce the original model.

Example 6.6.8. AR(p) Whitening Models

- Consider the class of filters $\psi(B) = 1 - \sum_{j=1}^{p} \psi_j B^j$, which are AR polynomial filters.
- We would apply these to the data, seeking p and coefficient values such that the data is whitened.
- We can estimate coefficients using ordinary least squares (or the Yule-Walker method, discussed later), for any p. These are fast to calculate, so we can just try over many choices of p.

Exercise 6.61. Whitening an AR(p) Process

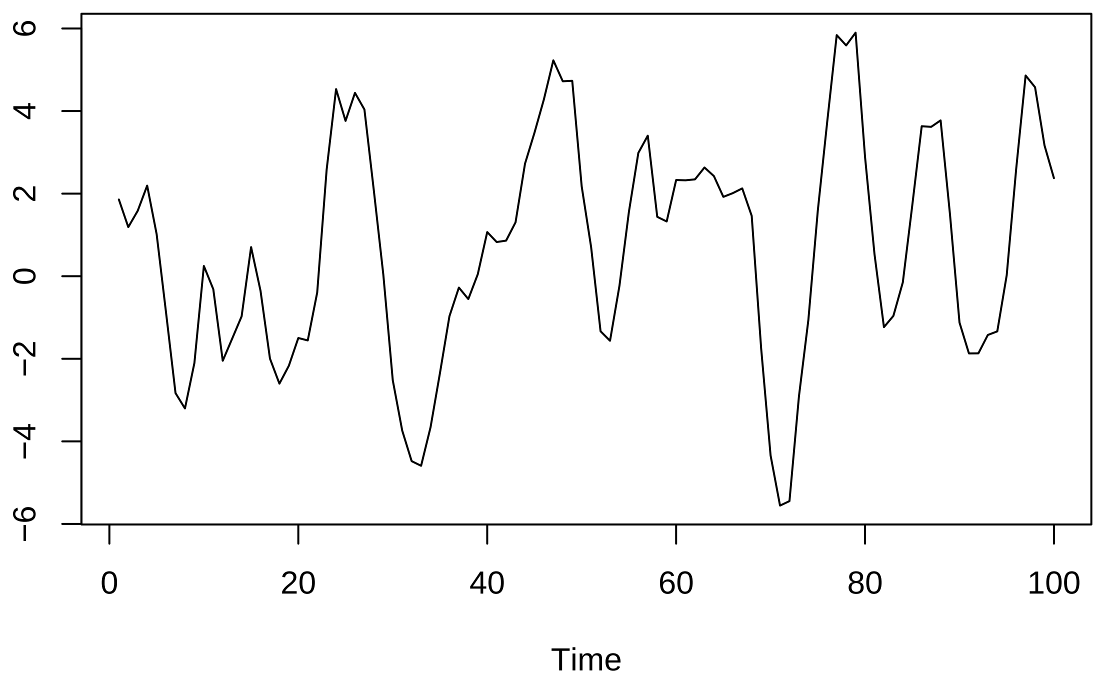
- We implement the method of Example 6.6.8, and apply to an AR(2) simulation.

```
arp.sim <- function(n,burn,ar.coefs,innovar)
{
    p <- length(ar.coefs)
    z <- rnorm(n+burn+p,sd=sqrt(innovar))
    x <- z[1:p]
    for(t in (p+1):(p+n+burn))
    {
```

```
        next.x <- sum(ar.coefs*x[(t-1):(t-p)]) + z[t]
        x <- c(x,next.x)
    }
    x <- x[(p+burn+1):(p+burn+n)]
    return(x)
}
```

- First we generate a simulation of a cyclic AR(2).

```
set.seed(777)
n <- 100
rho <- .8
omega <- pi/6
phi1 <- 2*rho*cos(omega)
phi2 <- -rho^2
ar.coef <- c(phi1,phi2)
x.sim <- arp.sim(n,500,ar.coef,1)
plot.ts(x.sim,ylab="")
```



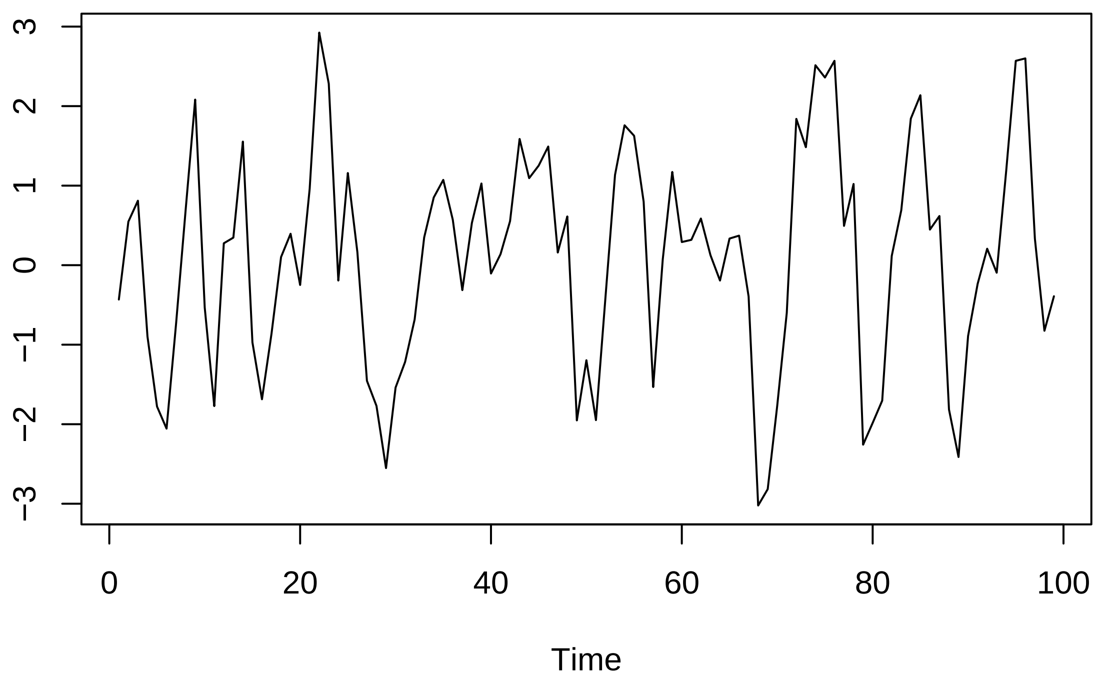
- Then we obtain fitted autoregressive filters, for various p up to 5.
- We use OLS to fit. Note that regression residuals are the filter output.

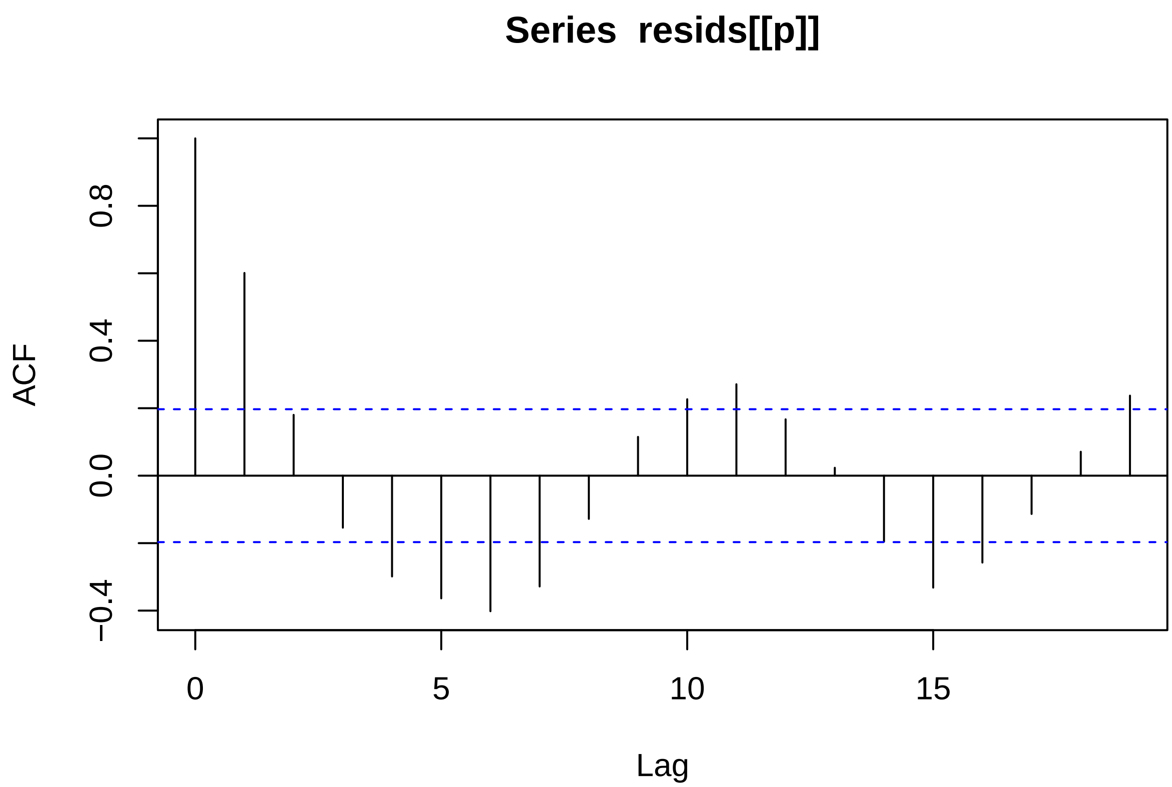
```
covars <- as.matrix(x.sim[-n])
coeffs <- list()
resids <- list()
for(p in 1:5)
{
```

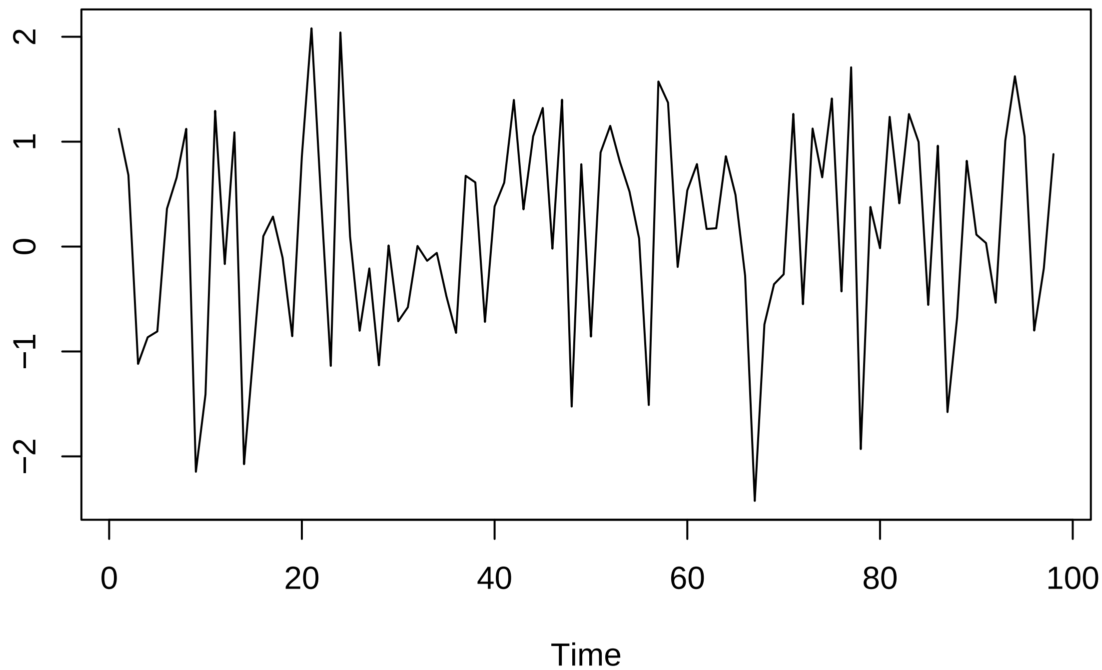
```
  ar.fit <- lm(x.sim[-seq(1,p)] ~ covars - 1)
  coeffs[[p]] <- ar.fit$coefficients
  resids[[p]] <- ar.fit$residuals
  covars <- cbind(covars[-1,],x.sim[-seq(n-p,n)])
}
```

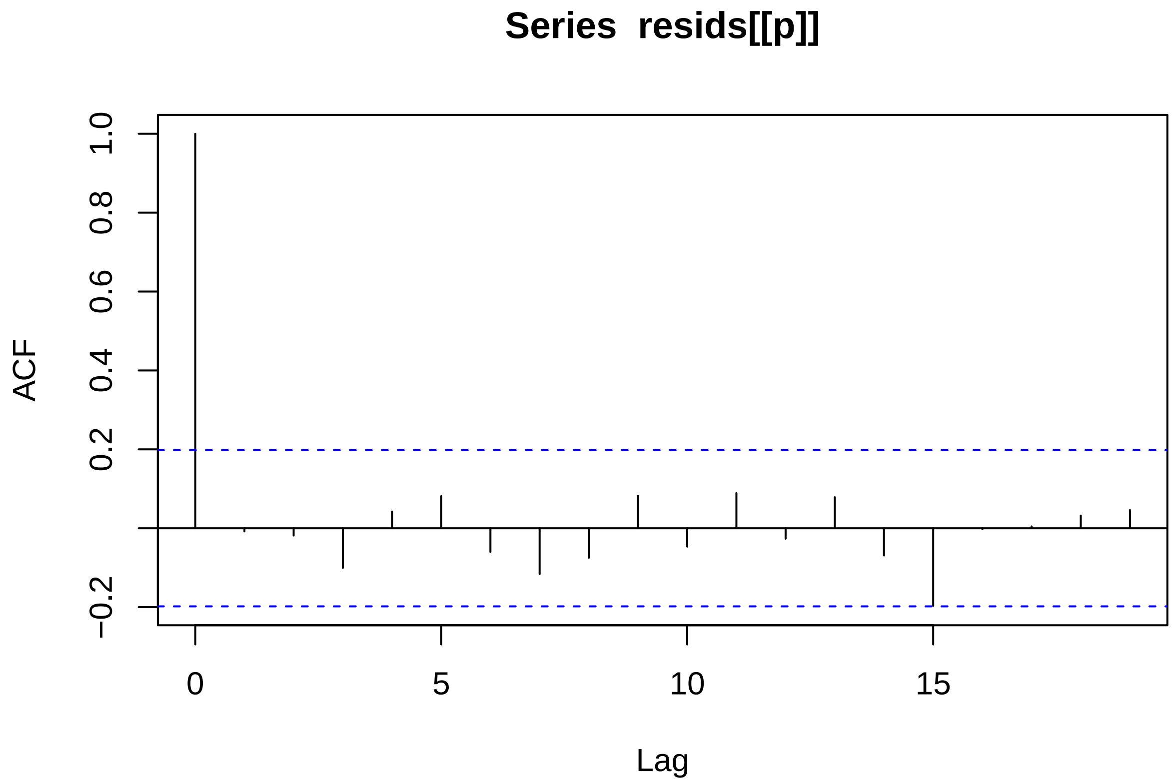
- We plot the filter outputs (the regression residuals) and estimates of the ACF.

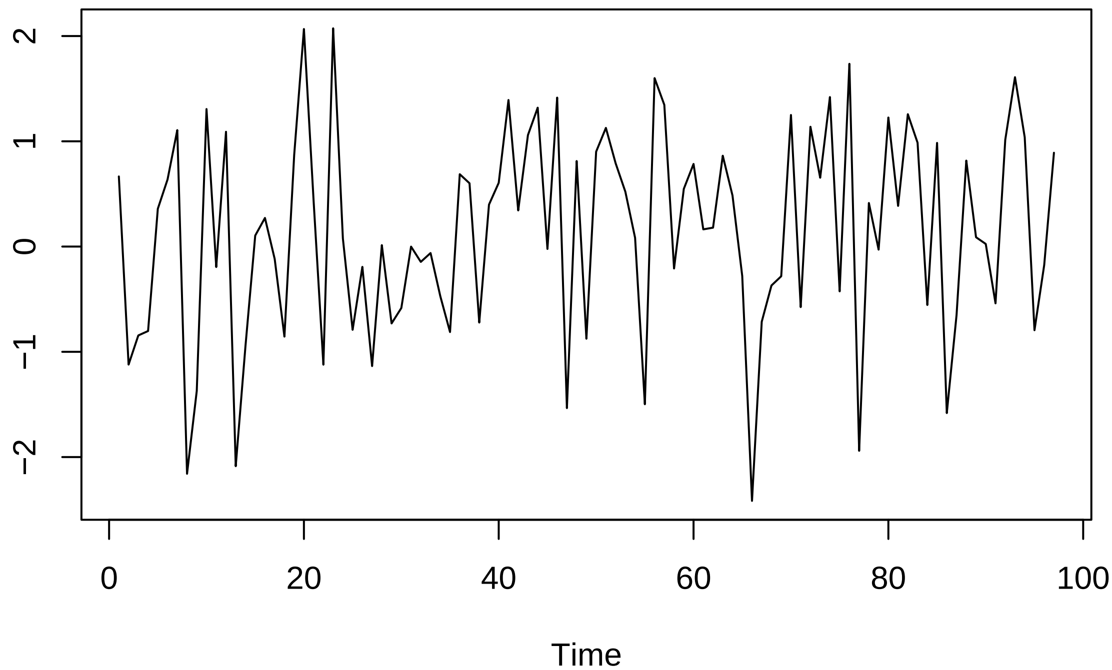
```
for(p in 1:5)
{
  plot.ts(resids[[p]],ylab="")
  acf(resids[[p]])
}
```



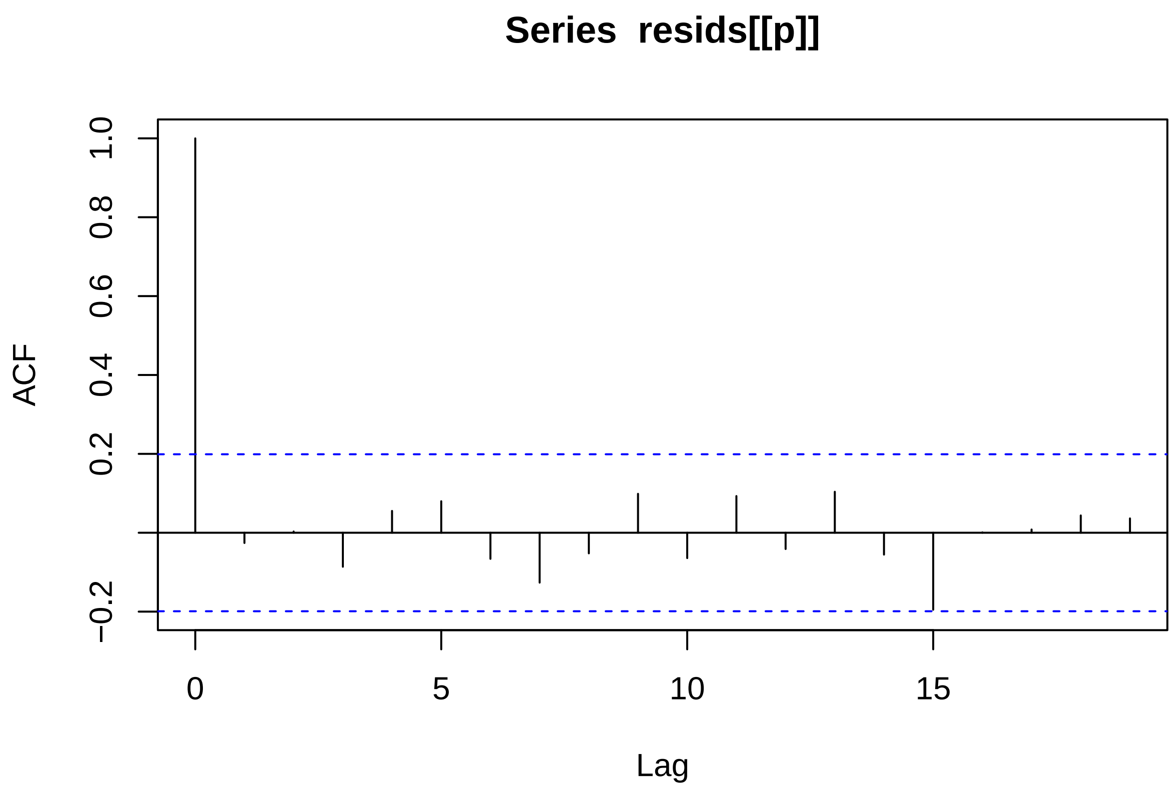

Series resids[[p]]
ACF
Lag
0.8
0.4
0.0
−0.4
0
5
10
15

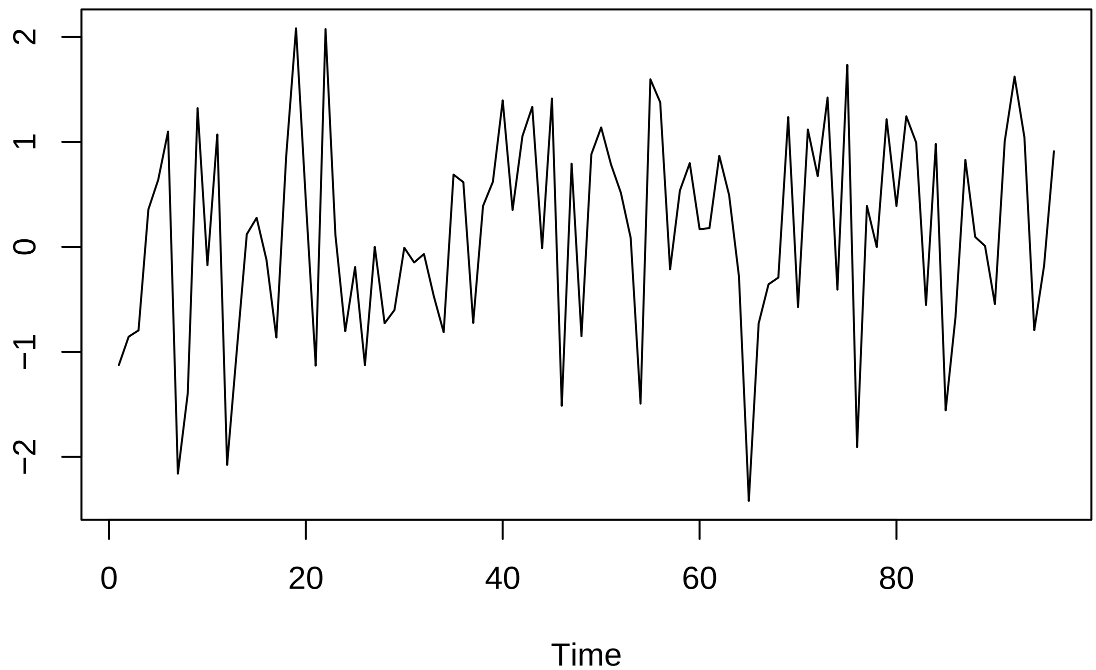

2
1
0
−1
−2
0
20
40
60
80
100
Time

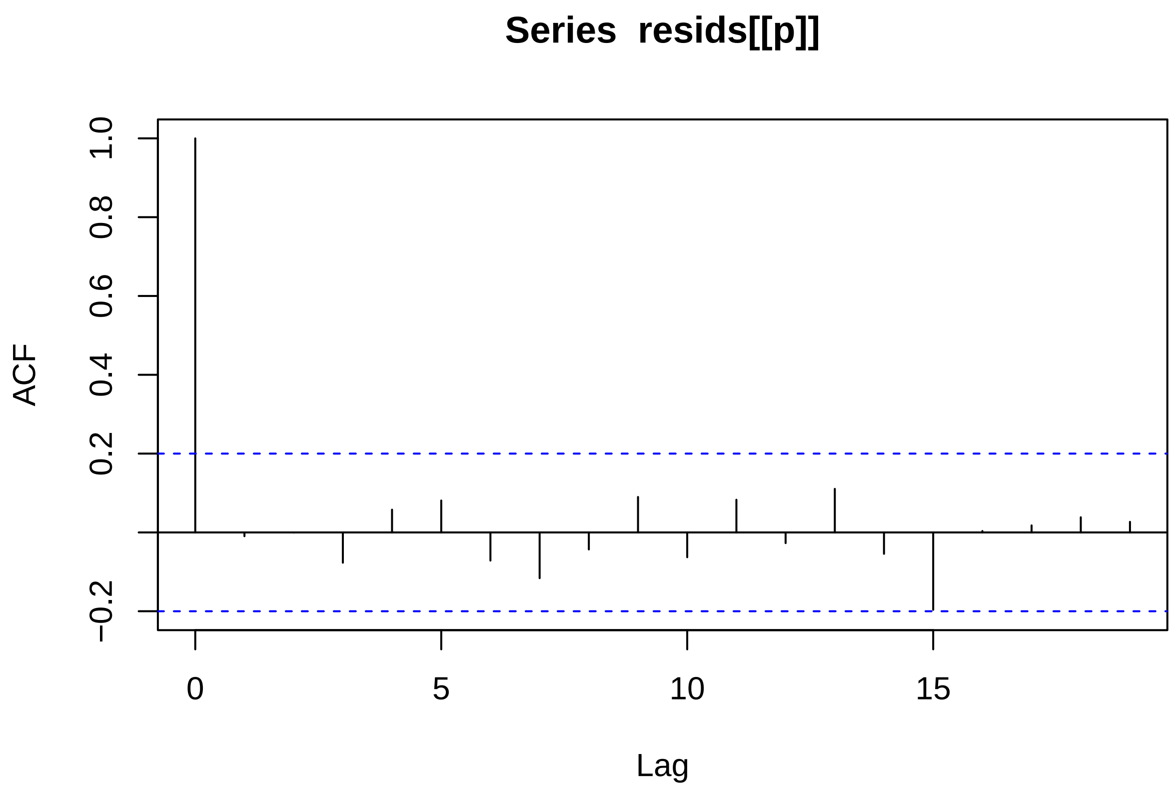

Series resids[[p]]
ACF
1.0
0.8
0.6
0.4
0.2
−0.2
0
5
10
15
Lag

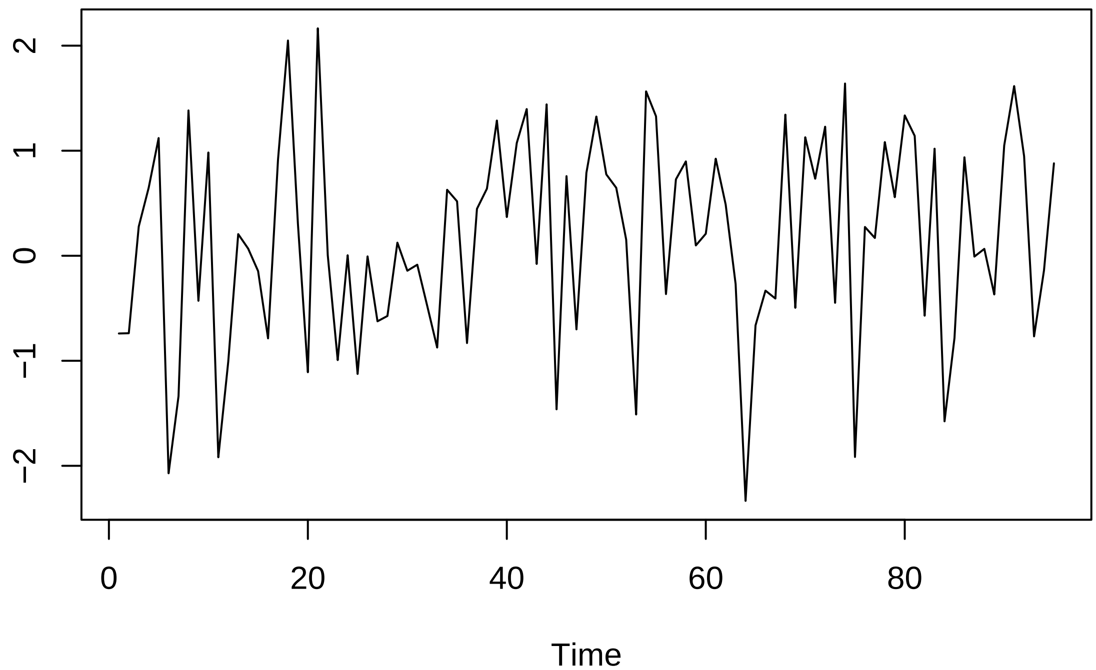


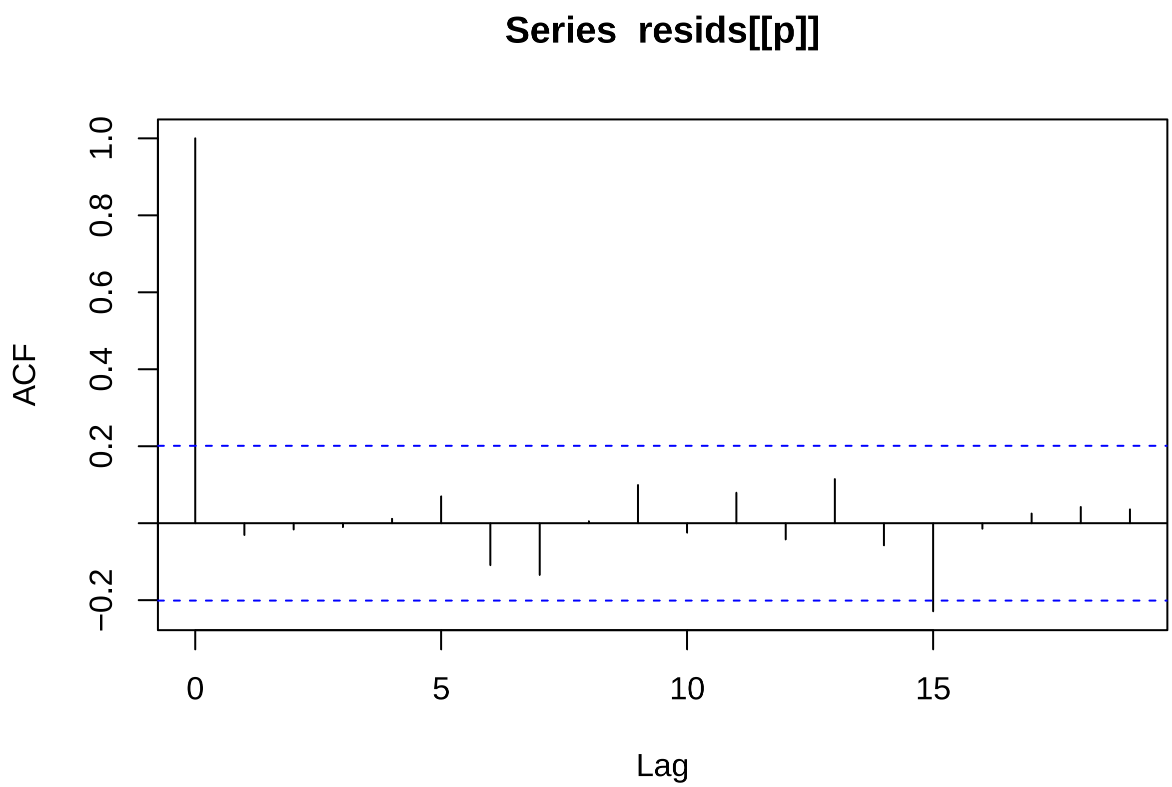
2
1
0
−1
−2
0
20
40
60
80
100
Time


Series resids[[p]]
ACF
1.0
0.8
0.6
0.4
0.2
−0.2
0
5
10
15
Lag


2
1
0
−1
−2
0
20
40
60
80
Time


Series resids[[p]]
ACF
1.0
0.8
0.6
0.4
0.2
−0.2
0
5
10
15
Lag


2
1
0
−1
−2
0
20
40
60
80
Time



- The correct filter is for $p = 2$. We print the coefficients and their estimates.

```
print(c(phi1,phi2))
```

```
## [1]  1.385641 -0.640000
```

```
print(coeffs[[2]])
```

```
##    covars1    covars2
##  1.4773729 -0.6966449
```

Lesson 7-2: Discrete Fourier Transform

We introduce the data analysis tool of Discrete Fourier Transform.

Definition 7.2.3.

Given a sample $X_1, \ldots, X_n$, the **Discrete Fourier Transform** (DFT) at the Fourier frequency $\lambda_l = 2\pi l/n$ (for $[n/2] - n + 1 \leq l \leq [n/2]$) is

$$\widetilde{X}(\lambda_l) = n^{-1/2} \sum_{t=1}^{n} X_t e^{-i\lambda_l t}.$$

Definition 7.2.4.

- The **periodogram** $I(\lambda)$ is a non-negative function of $\lambda \in [-\pi, \pi]$, constructed from the sample $X_1, \ldots, X_n$:

$$I(\lambda) = n^{-1} \left| \sum_{t=1}^{n} (X_t - \overline{X}) e^{-i\lambda t} \right|^2 .$$

- So $I(0) = 0$. Sometimes we consider an "uncentered" periodogram, where there is no centering by the sample mean. We denote this by $\widetilde{I}(\lambda)$, and

$$\widetilde{I}(\lambda_l) = |\widetilde{X}(\lambda_l)|^2 .$$

- The periodogram is an empirical version of the spectral density, and shares certain properties. Higher values correspond to cyclical effects in the data.

Proposition 7.2.7.

- Let the vector of DFTs be denoted $\underline{\widetilde{X}}$, which has components $\widetilde{X}(\lambda_l)$.
- This is a linear function of the sample vector $\underline{X}$:

$$\underline{\widetilde{X}} = Q^* \, \underline{X},$$

  where Q was defined in Definition 6.4.3.
- Since Q is unitary, $Q^{-1} = Q^*$, so we can recover the data from the DFT vector via

$$\underline{X} = Q \, \underline{\widetilde{X}}.$$

Exercise 7.14. DFT of an AR(1).

- We simulate an AR(1).

```
arp.sim <- function(n,burn,ar.coefs,innovar)
{
    p <- length(ar.coefs)
    z <- rnorm(n+burn+p,sd=sqrt(innovar))
    x <- z[1:p]
    for(t in (p+1):(p+n+burn))
    {
        next.x <- sum(ar.coefs*x[(t-1):(t-p)]) + z[t]
        x <- c(x,next.x)
    }
    x <- x[(p+burn+1):(p+burn+n)]
    return(x)
}

phi <- .8
innovar <- 1
n <- 200
x.sim <- arp.sim(n,500,phi,innovar)
```

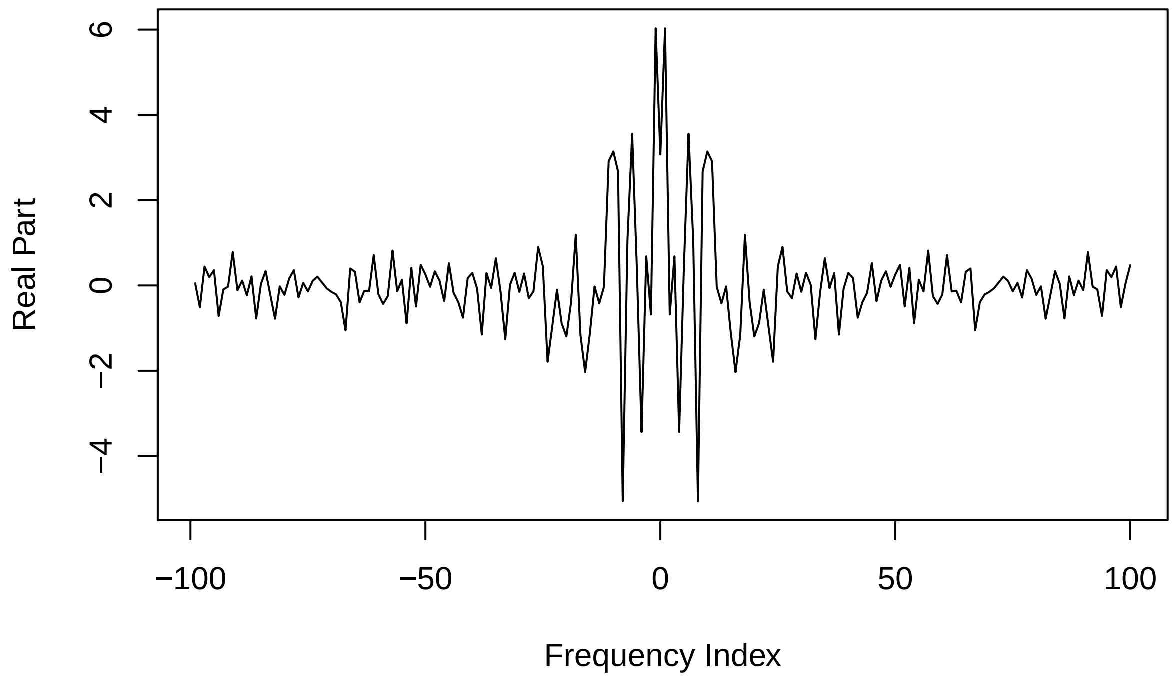
- We compute the DFT. We begin with code to compute Q.

```
get.qmat <- function(mesh)
{
    mesh2 <- floor(mesh/2)
    inds <- seq(mesh2-mesh+1,mesh2)
```

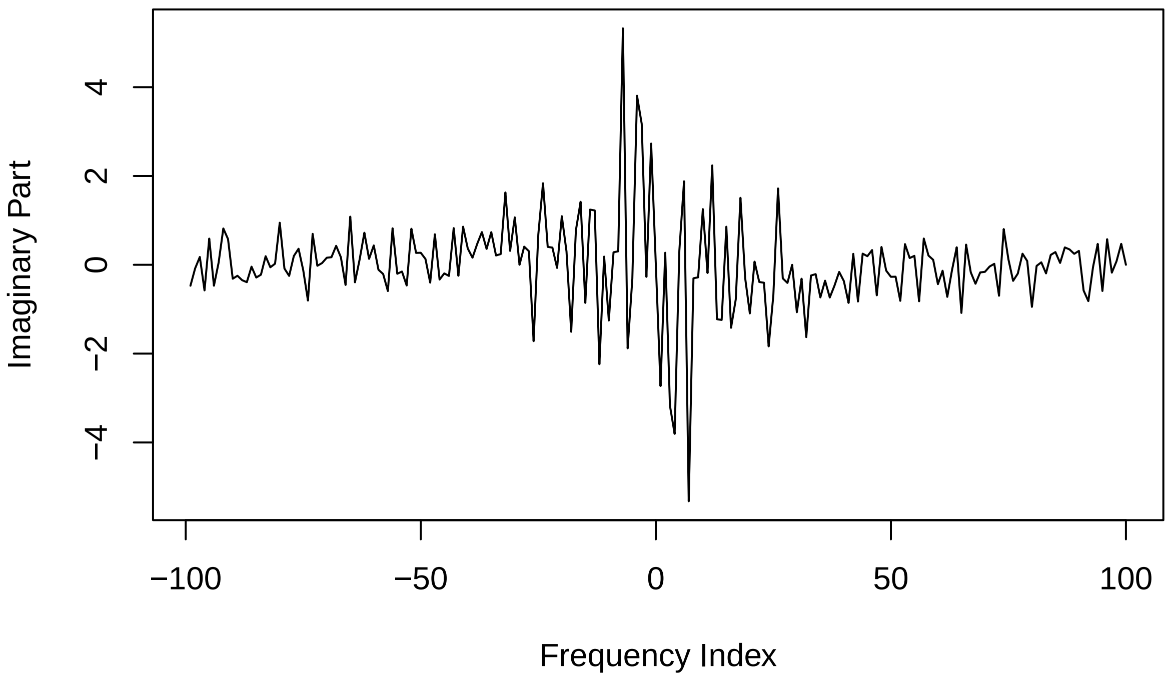
```
    Q.mat <- exp(1i*2*pi*mesh^{-1}*t(t(seq(1,mesh)) %x% inds))*mesh^{-1/2}
    return(Q.mat)
}
Q.mat <- get.qmat(n)
```

- Then we use Proposition 7.2.7 to get the DFT vector. We plot the real part, imaginary part, and the modulus.

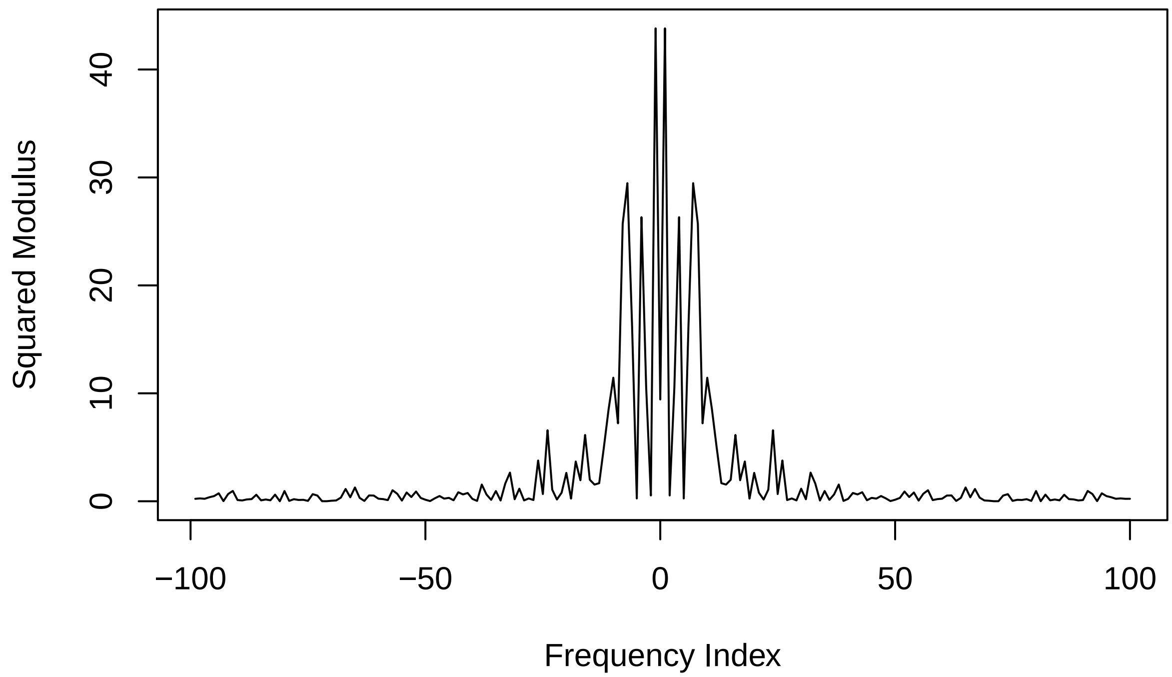
```
x.dft <- Conj(t(Q.mat)) %*% x.sim
plot(ts(Re(x.dft),start=-floor(n/2)+1,frequency=1),
    xlab="Frequency Index",ylab="Real Part")
```




```
plot(ts(Im(x.dft),start=-floor(n/2)+1,frequency=1),
    xlab="Frequency Index",ylab="Imaginary Part")
```




```
plot(ts(Mod(x.dft)^2,start=-floor(n/2)+1,frequency=1),
    xlab="Frequency Index",ylab="Squared Modulus")
```



Corollary 7.2.9. Decorrelation Property of the DFT.

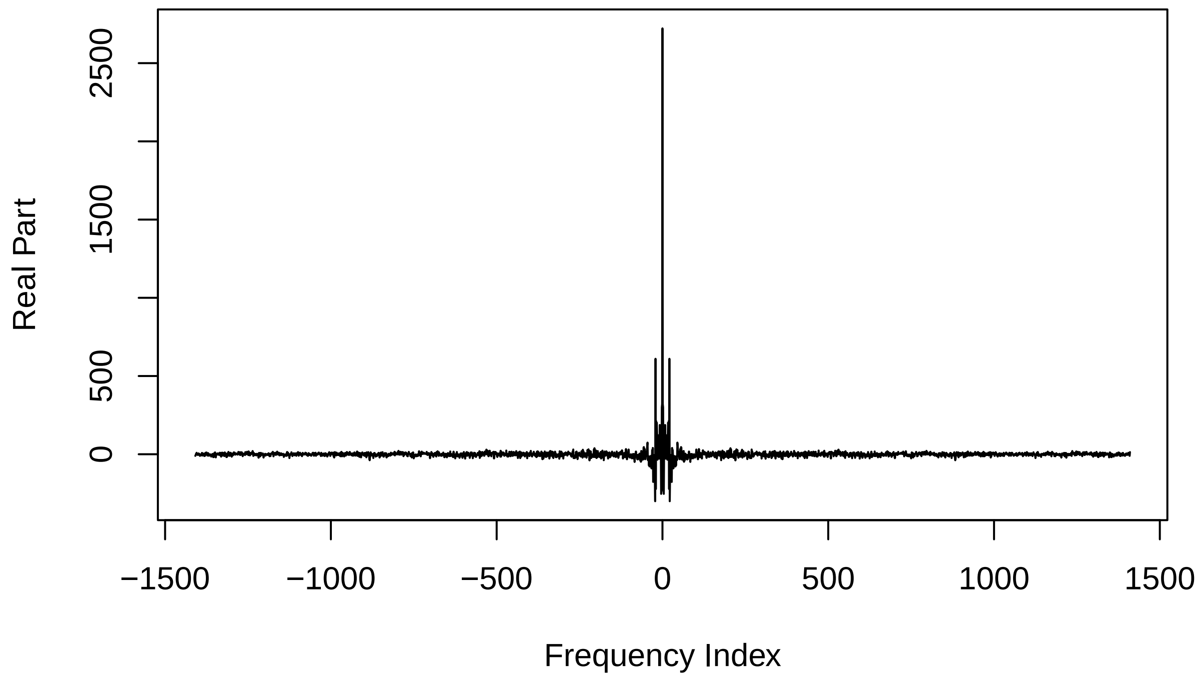
- Let $X_1, \ldots, X_n$ be a sample from a mean zero, covariance stationary time series with absolutely summable ACVF and spectral density f. Then $\underline{\widetilde{X}}$ has approximate covariance matrix $\mathrm{diag}\{f(\lambda_{[n/2]-n+k})\}$.
- This follows from Theorem 6.4.5:

$$\mathrm{Cov}[\underline{\widetilde{X}}] = Q^* \, \mathrm{Cov}[\underline{X}] \, Q \approx \Lambda.$$

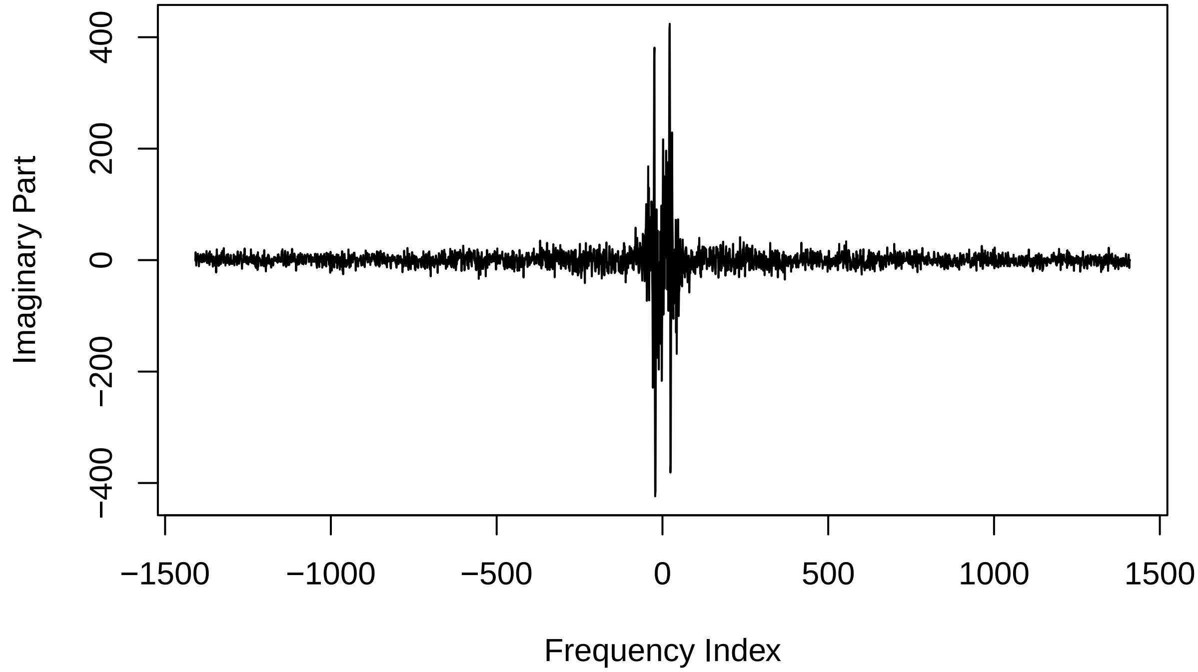
Exercise 7.18. DFT of Wolfer Sunspots.

- We compute the DFT of the Wolfer sunspot data, and plot.

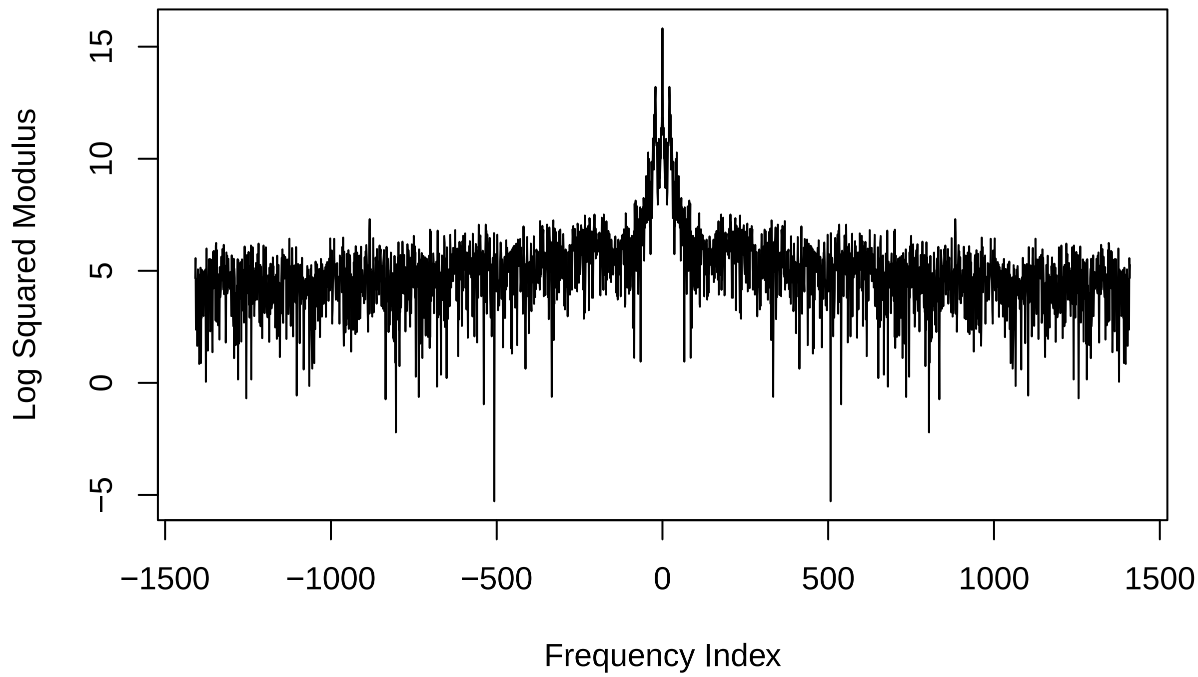
```
wolfer <- read.table("wolfer.dat")
wolfer <- ts(wolfer,start=1749,frequency=12)
n <- length(wolfer)
Q.mat <- get.qmat(n)
x.dft <- Conj(t(Q.mat)) %*% wolfer
plot(ts(Re(x.dft),start=-floor(n/2)+1,frequency=1),
    xlab="Frequency Index",ylab="Real Part")
```




```r
plot(ts(Im(x.dft),start=-floor(n/2)+1,frequency=1),
    xlab="Frequency Index",ylab="Imaginary Part")
```




```
plot(ts(log(Mod(x.dft)^2),start=-floor(n/2)+1,frequency=1),
    xlab="Frequency Index",ylab="Log Squared Modulus")
```

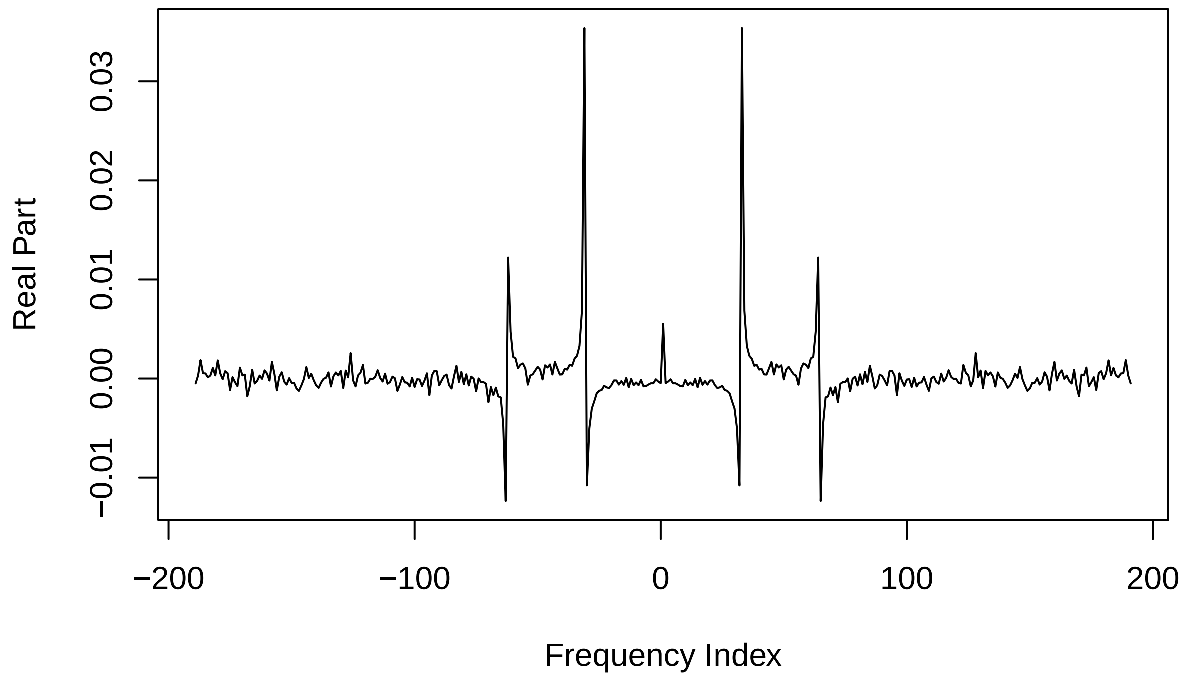


- We see higher values of the squared modulus (the uncentered periodogram) near frequency zero.

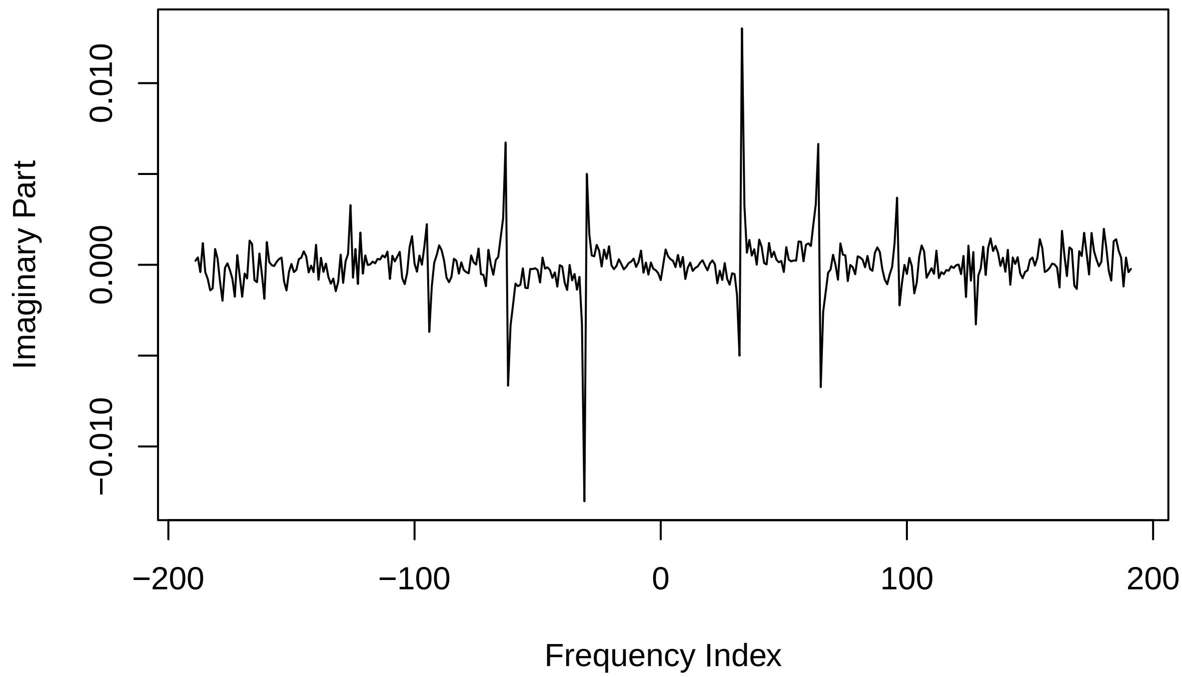
Exercise 7.20. DFT of Mauna Loa Growth Rate.

- We compute the DFT of the Mauna Loa growth rate, and plot.

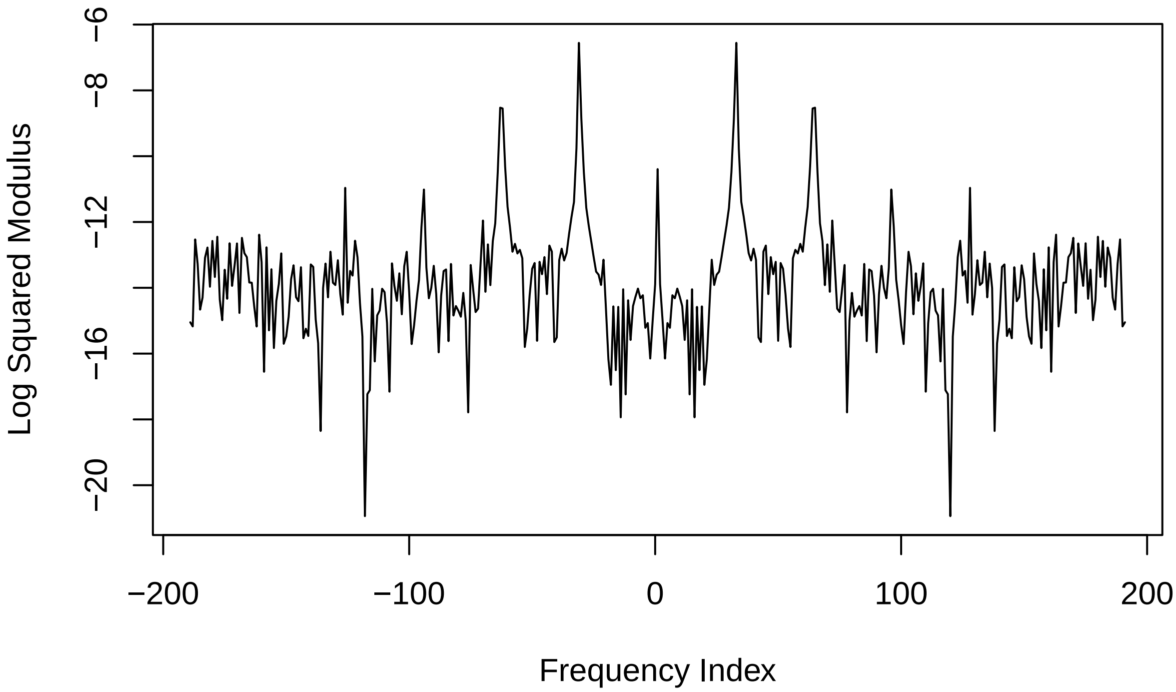
```
mau <- read.table("mauna.dat",header=TRUE,sep="")
mau <- ts(mau,start=1958,frequency=12)
mau.gr <- diff(log(mau))
n <- length(mau.gr)
Q.mat <- get.qmat(n)
x.dft <- Conj(t(Q.mat)) %*% mau.gr
plot(ts(Re(x.dft),start=-floor(n/2)+1,frequency=1),
    xlab="Frequency Index",ylab="Real Part")
```




```
plot(ts(Im(x.dft),start=-floor(n/2)+1,frequency=1),
    xlab="Frequency Index",ylab="Imaginary Part")
```




```r
plot(ts(log(Mod(x.dft)^2),start=-floor(n/2)+1,frequency=1),
    xlab="Frequency Index",ylab="Log Squared Modulus")
```



- We see higher values of the uncentered periodogram in the shape of peaks, at four non-zero frequencies. These correspond to cyclical seasonal effects.

Lesson 7-3: Spectral Representation

We discuss a representation of a stationary time series as a sum of stochastic cosines.

Definition 7.3.1.

- For a given spectral distribution F, a **spectral increment process** Z is a complex-valued continuous-time stochastic process defined on the interval $[-\pi, \pi]$, which has mean zero with *orthogonal increments*: for $\lambda_1 < \lambda_2 < \lambda_3 < \lambda_4$, the random variables $Z(\lambda_2) - Z(\lambda_1)$ and $Z(\lambda_4) - Z(\lambda_3)$ are orthogonal. Also,

$$\mathrm{Var}[Z(\lambda_2) - Z(\lambda_1)] = \frac{1}{2\pi} \left(F(\lambda_2) - F(\lambda_1) \right).$$

- The variance of a complex random variable is the expectation of its squared modulus.
- We abbreviate the above expression by

$$\mathrm{Var}[dZ(\lambda)] = \frac{1}{2\pi} dF(\lambda).$$

Paradigm 7.3.4. Time Series Defined as a Stochastic Integral

- We can define a time series via a stochastic integral as follows:

$$X_t = \int_{-\pi}^{\pi} e^{i\lambda t} dZ(\lambda).$$

- This resembles a sum of stochastic cosines, where $dZ(\lambda)$ is the amplitude for a sinusoid $e^{i\lambda t}$.
- Then $\{X_t\}$ has mean zero and autocovariance

$$\text{Cov}[X_{t+h}, X_t] = \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} e^{i\lambda(t+h)} e^{-i\omega t} \text{Cov}[dZ(\lambda), dZ(\omega)] = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{i\lambda h} dF(\lambda).$$

- So $\{X_t\}$ is weakly stationary with ACVF $\gamma(h)$ given by above formula.
- Conversely: any mean zero weakly stationary time series $\{X_t\}$ with spectral distribution function F can be represented by the above stochastic integral!

Corollary 7.3.8.

- Suppose $\{X_t\}$ is a mean zero weakly stationary time series with spectral representation, and let $Y_t = \psi(B) X_t$. Then

$$Y_t = \sum_j \psi_j X_{t-j} = \sum_j \psi_j \int_{-\pi}^{\pi} e^{i\lambda(t-j)} dZ(\lambda) = \int_{-\pi}^{\pi} \sum_j \psi_j e^{-i\lambda j} e^{i\lambda t} dZ(\lambda) = \int_{-\pi}^{\pi} \psi(e^{-i\lambda}) e^{i\lambda t} dZ(\lambda).$$

- So $\{Y_t\}$ also has a spectral representation, but its increment process is $\psi(e^{-i\lambda}) dZ(\lambda)$, where $\psi(e^{-i\lambda})$ is the filter frequency response function. We can use this result to understand the impact of a filter in the frequency domain.
- The ACVF is

$$\text{Cov}[Y_{t+h}, Y_t] = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{i\lambda h} \left|\psi(e^{-i\lambda})\right|^2 dF(\lambda).$$

Example 7.3.10. Time Shift

- Consider $\psi(B) = B^k$, a shift by k time units. Then $\psi(e^{-i\lambda}) = e^{-i\lambda k}$ and $Y_t = \int_{-\pi}^{\pi} e^{i\lambda(t-k)} dZ(\lambda)$.
- This is the same as $B^k X_t = X_{t-k}$.

Definition 7.3.11.

- We can decompose the frequency response function into the **gain** function and the **phase delay** function, each with an interpretation from the spectral representation.
- We use the polar decomposition of a complex number: for any λ,

$$\psi(e^{-i\lambda}) = |\psi(e^{-i\lambda})| \exp\{i \text{Arg} \psi(e^{-i\lambda})\}.$$

- The magnitude $|\psi(e^{-i\lambda})|$ is called the **gain** function of the filter. It is computed by taking the square root of sum of squares of real and imaginary parts.
- The angular portion $\text{Arg}\psi(e^{-i\lambda})$ is called the **phase** function of the filter. It is computed by taking the arc tangent of the ratio of imaginary to real parts.
- When the phase function is differentiable with respect to λ, we define the **phase delay** via

$$\Upsilon(\lambda) = \frac{-\text{Arg}\psi(e^{-i\lambda})}{\lambda}$$

  for $\lambda \neq 0$, and the limit of such for $\lambda = 0$.
- The phase delay may be discontinuous in λ.
- The gain is an even function; both phase and phase delay are odd. So we usually just plot them over $[0, \pi]$ instead of $[-\pi, \pi]$.

Fact 7.3.12. Action of Phase Delay.

- From Corollary 7.3.8,

$$Y_t = \int_{-\pi}^{\pi} e^{i(\lambda t + \text{Arg}\psi(e^{-i\lambda}))} |\psi(e^{-i\lambda})| dZ(\lambda) = \int_{-\pi}^{\pi} e^{i\lambda(t - \Upsilon(\lambda))} |\psi(e^{-i\lambda})| dZ(\lambda).$$

- So at frequency λ, time t is delayed by $\Upsilon(\lambda)$ time units.
- Also, the gain modifies the autocovariances.

Example 7.3.13. Simple Moving Average Filters Cause Delay

- Consider the simple moving average filter $\psi(B) = (1 + B + B^2)/3$, which gives the average of the past and present 3 observations.
- We directly compute the frequency response function:

$$\psi(e^{-i\lambda}) = \frac{1 + e^{-i\lambda} + e^{-i2\lambda}}{3} = e^{-i\lambda}\,\frac{e^{i\lambda} + 1 + e^{-i\lambda}}{3} = e^{-i\lambda}\,\frac{1 + 2\cos(\lambda)}{3}.$$

- The gain function is

$$\frac{1}{3}|1 + 2\cos(\lambda)|.$$

- Note that $1 + 2\cos(\lambda)$ is non-negative for $\lambda \in [0, 2\pi/3]$, so the phase function equals $-\lambda$ over that set. Otherwise, we need a -1 factor which leads to a phase function equal to $-\lambda - \pi$.
- The phase delay function is then

$$\Upsilon(\lambda) = \begin{cases} 1 & \text{if } \lambda \in [0, 2\pi/3] \\ 1 + \pi/\lambda & \text{else.} \end{cases}$$

- Since this function is positive, the filter always provides a delay.
- The gain function attenuates higher frequencies, due to the cosine shape, so the filter is a "low-pass."

Example 7.3.14. Differencing Causes an Advance

- Consider the differencing filter $\psi(B) = 1 - B$.
- The frequency response function is

$$\psi(e^{-i\lambda}) = 1 - e^{-i\lambda} = 1 - \cos(\lambda) - i\sin(\lambda).$$

- The squared gain function is $2 - 2\cos(\lambda)$.
- The phase function is $(\pi - \lambda)/2$.
- Away from $\lambda = 0$, the phase delay is $\Upsilon(\lambda) = .5(1 - \pi/\lambda)$. This is negative for $\lambda \in (0, \pi]$, so differencing causes an advance.
- The gain function attenuates lower frequencies, so the filter is a "high-pass."

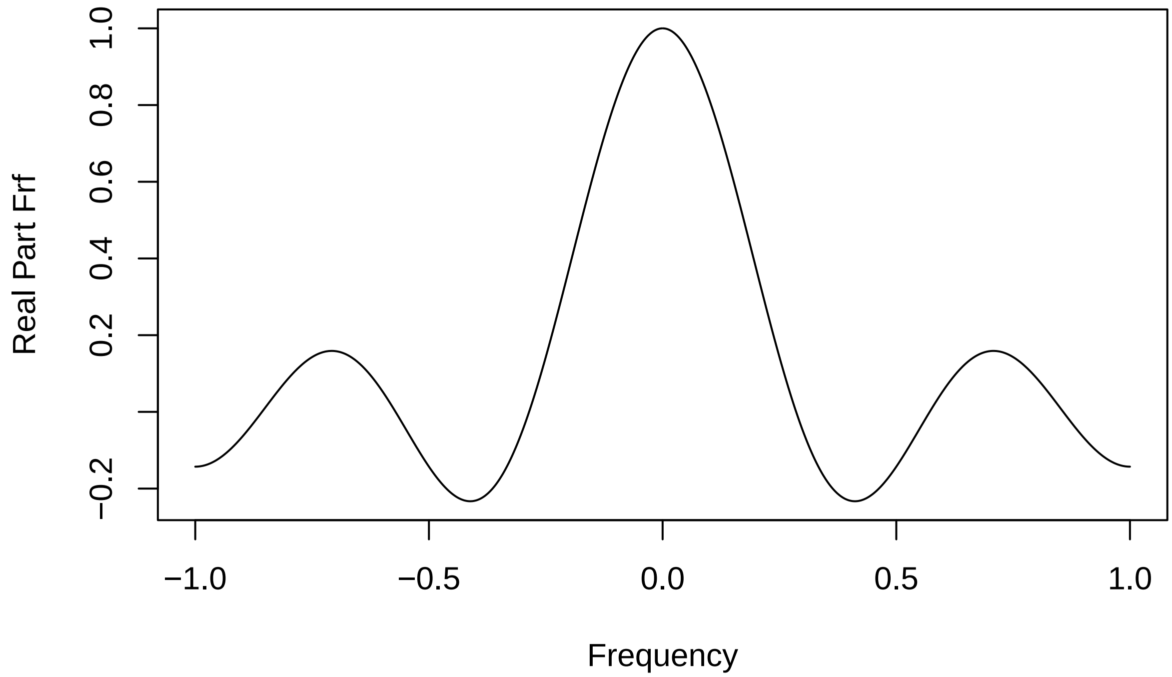
Exercise 7.29. Phase and Gain for Simple Moving Average.

- Consider the simple moving average of order p: $\psi(B) = (2p+1)^{-1} \sum_{j=-p}^{p} B^j$.
- It can be shown that

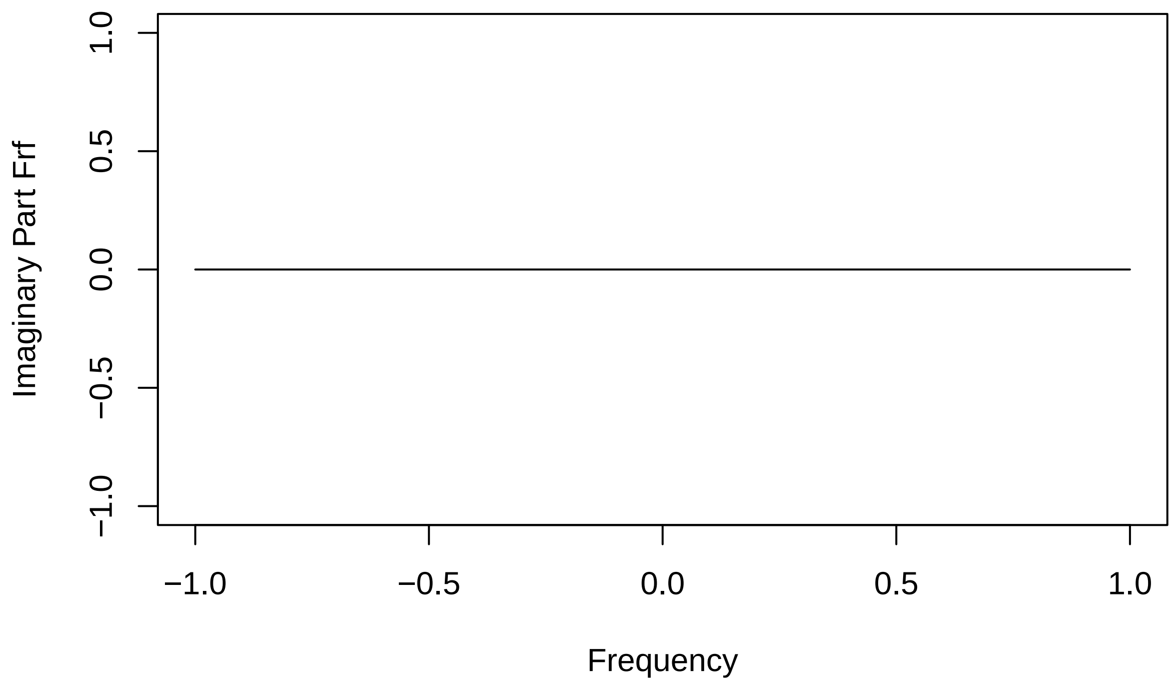
$$\psi(e^{-i\lambda}) = \frac{\sin(\lambda(p + 1/2))}{(2p+1)\sin(\lambda/2)}$$

- We use this formula to encode and display the gain and phase functions, as well as the real and imaginary parts of the frequency response function.

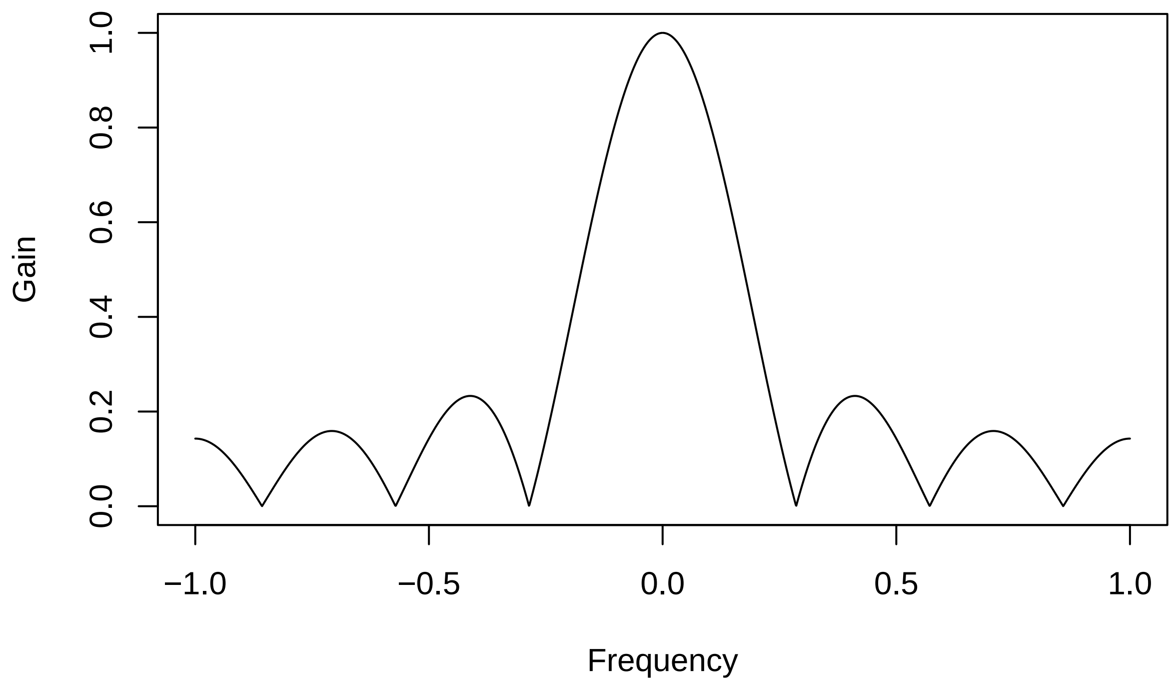
```
lambda <- pi*seq(-1000,1000)/1000
p <- 3
simplema.frf <- sin((p+1/2)*lambda)/((2*p+1)*sin(lambda/2))
simplema.gain <- Mod(simplema.frf)
simplema.phase <- atan(Im(simplema.frf)/Re(simplema.frf))
simplema.delay <- -1*simplema.phase/lambda
plot(ts(Re(simplema.frf),start=-1,frequency=1000),
    xlab="Frequency",ylab="Real Part Frf")
```



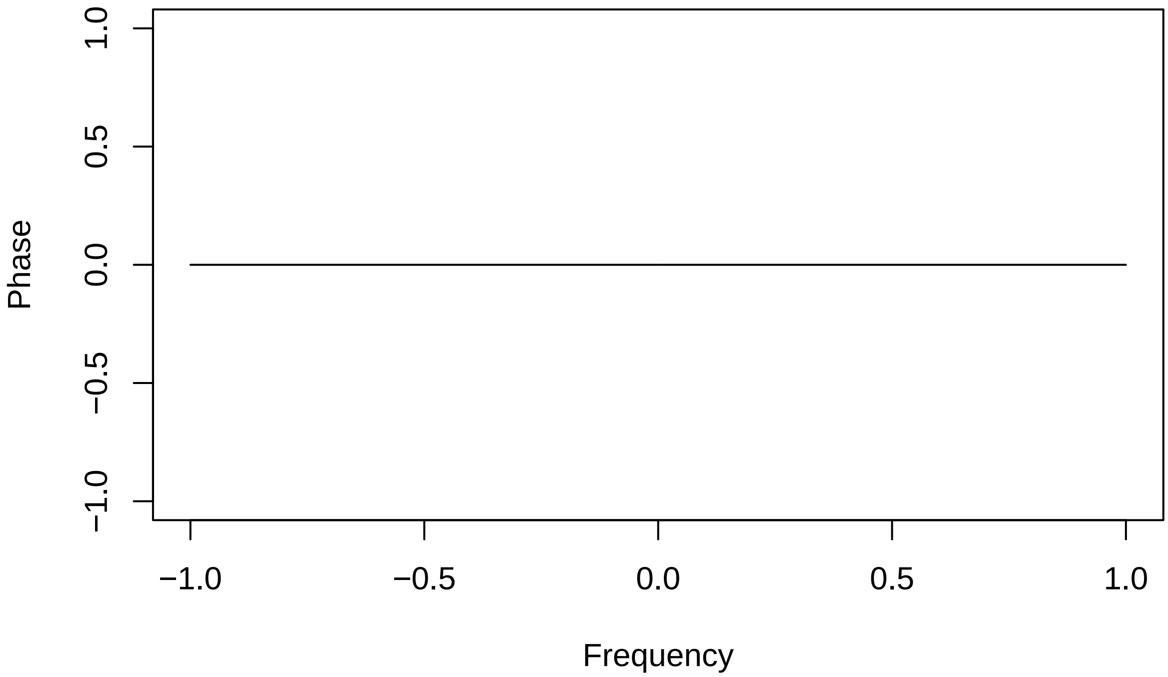

```
plot(ts(Im(simplema.frf),start=-1,frequency=1000),
    xlab="Frequency",ylab="Imaginary Part Frf")
```



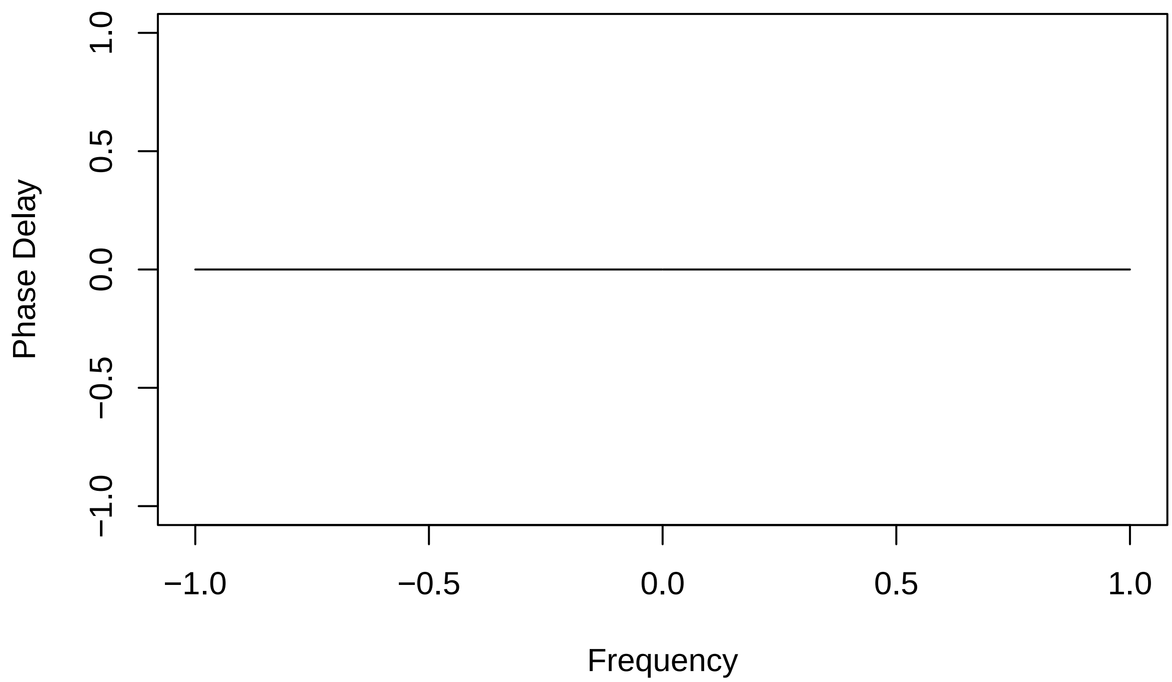

```
plot(ts(simplema.gain,start=-1,frequency=1000),
    xlab="Frequency",ylab="Gain")
```




```r
plot(ts(simplema.phase,start=-1,frequency=1000),
    xlab="Frequency",ylab="Phase")
```



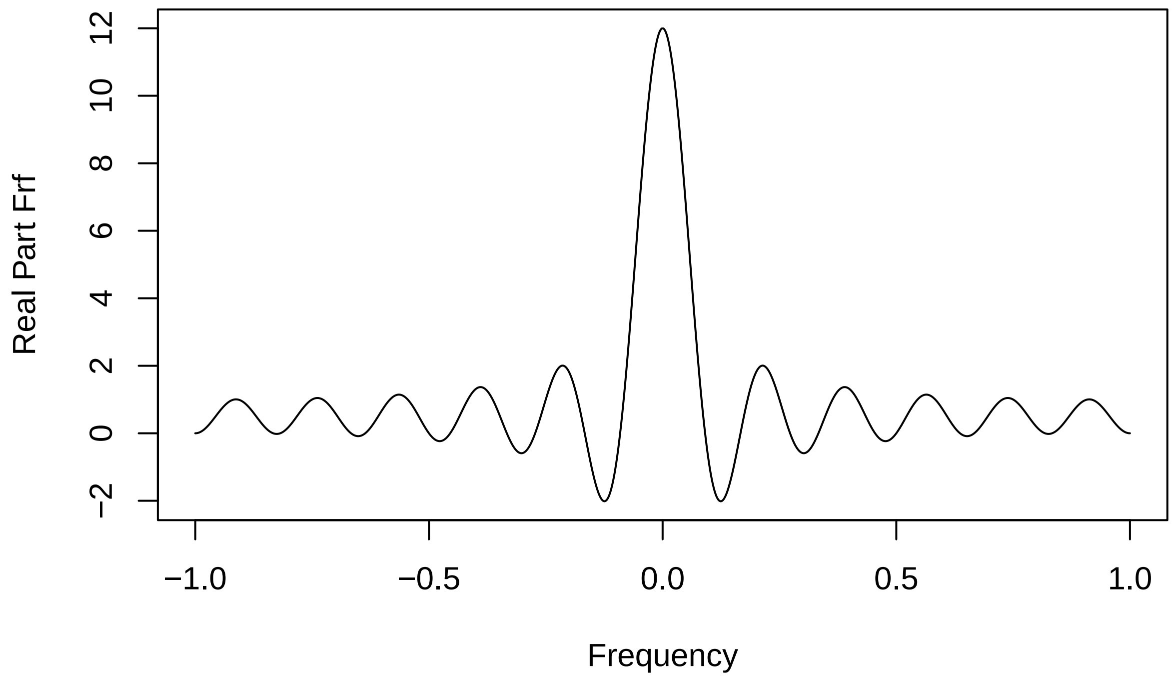

```
plot(ts(simplema.delay,start=-1,frequency=1000),
    xlab="Frequency",ylab="Phase Delay")
```



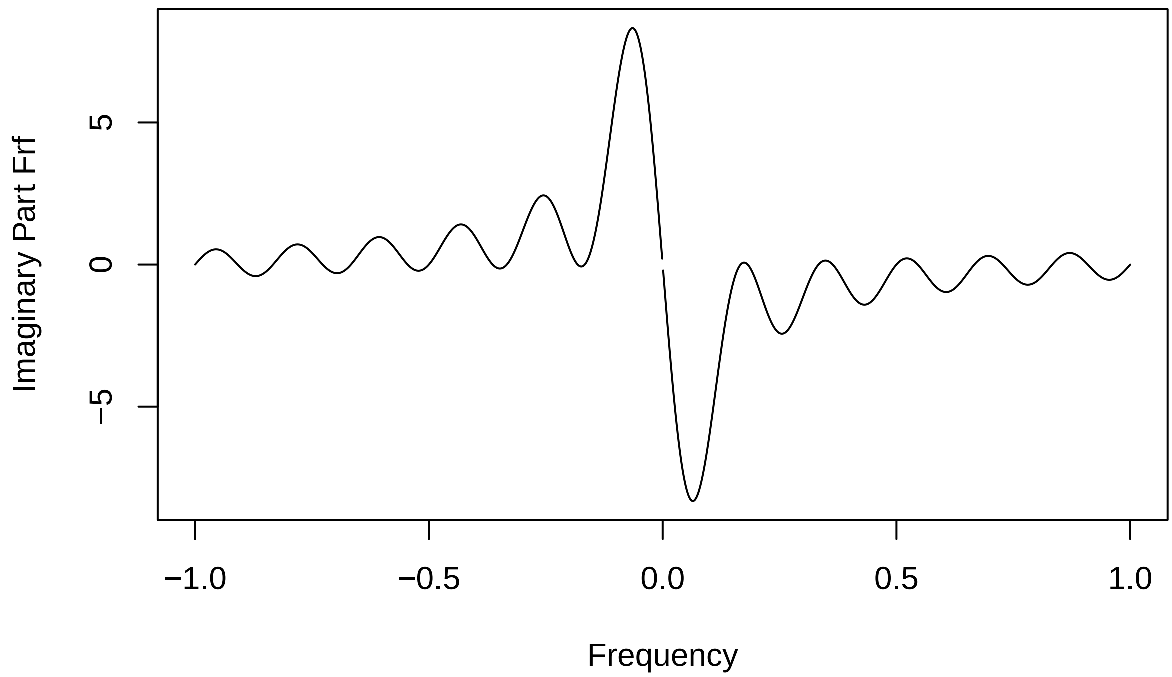
Exercise 7.30. Phase and Gain for Seasonal Aggregation Filter.

- Consider the seasonal aggregation filter: $\psi(B) = \sum_{j=0}^{s-1} B^j$.
- We encode and display the gain and phase functions, as well as the real and imaginary parts of the frequency response function.

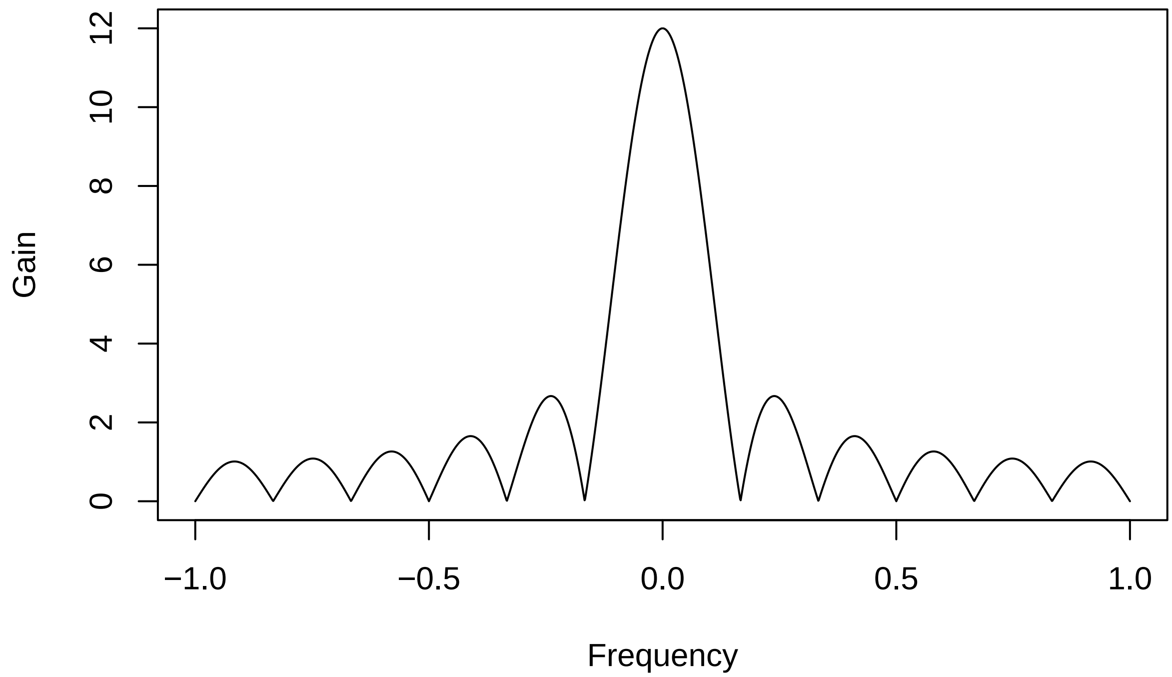
```
lambda <- pi*seq(-1000,1000)/1000
s <- 12
seasagg.frf <- exp(-1i*lambda*(s-1)/2)*sin((s/2)*lambda)/sin(lambda/2)
seasagg.gain <- Mod(seasagg.frf)
seasagg.phase <- atan(Im(seasagg.frf)/Re(seasagg.frf))
seasagg.delay <- -1*seasagg.phase/lambda
seasagg.delay[1001] <- NA
plot(ts(Re(seasagg.frf),start=-1,frequency=1000),
    xlab="Frequency",ylab="Real Part Frf")
```



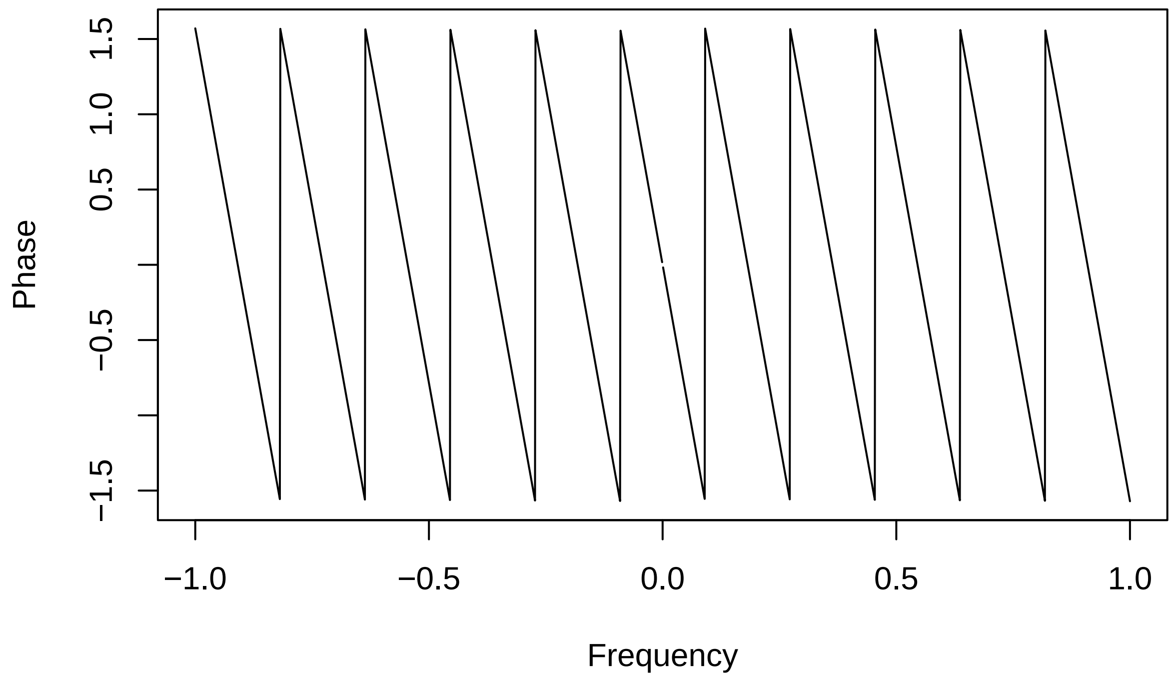

```
plot(ts(Im(seasagg.frf),start=-1,frequency=1000),
    xlab="Frequency",ylab="Imaginary Part Frf")
```



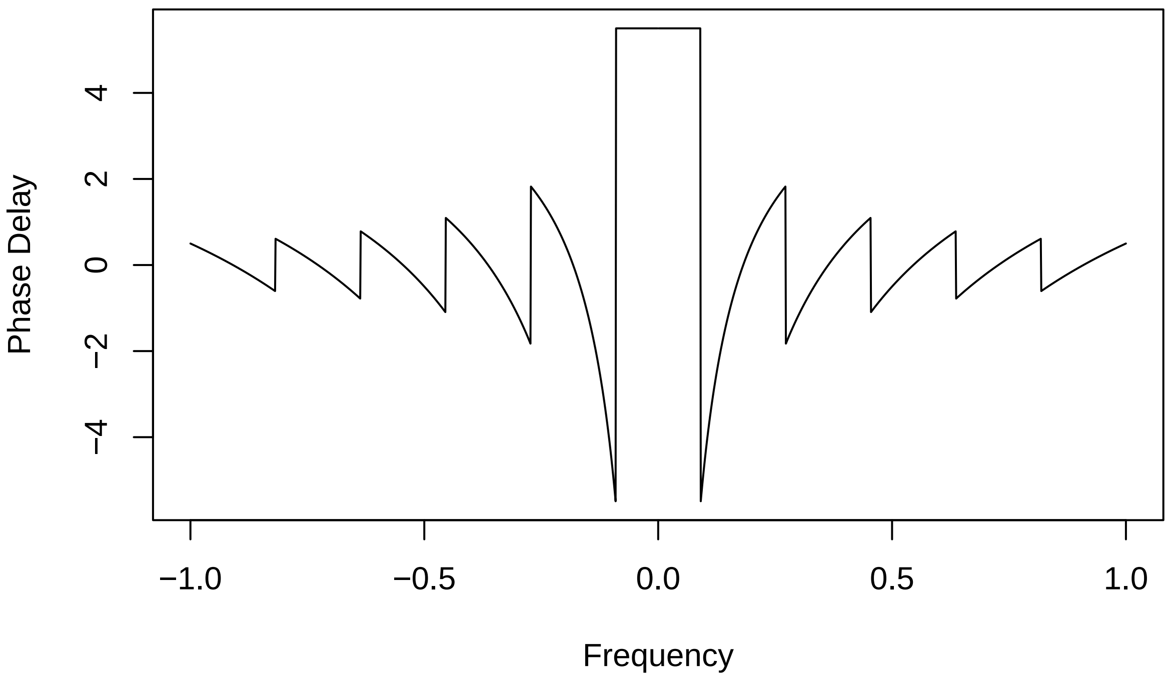

```r
plot(ts(seasagg.gain,start=-1,frequency=1000),
    xlab="Frequency",ylab="Gain")
```




```
plot(ts(seasagg.phase,start=-1,frequency=1000),
    xlab="Frequency",ylab="Phase")
```




```
plot(ts(seasagg.delay,start=-1,frequency=1000),
    xlab="Frequency",ylab="Phase Delay")
```


Phase Delay
4
2
0
−2
−4
−1.0
−0.5
0.0
0.5
1.0
Frequency