

MATH 545: INTRODUCTION TO TIME SERIES ANALYSIS

WINTER 2025

STATE SPACE MODELS AND THE KALMAN FILTER

Let $\mathbf{y}_{1:t} = \{\mathbf{y}_1, \dots, \mathbf{y}_t\}$ be the observed data, and $\mathbf{X}_t$ be the corresponding unobserved state vector at time t , for $t = 1, \dots, n$. Inference goals for state space models are to

- **filtering (or tracking)**: estimate $\mathbf{X}_t$ from all observations up to time t via the distribution

$$p(\mathbf{x}_t | \mathbf{y}_{1:t})$$

- **prediction (or forecasting)**: estimate $\mathbf{X}_t$ from observations up to a previous time point $t - s$ via the distribution

$$p(\mathbf{x}_t | \mathbf{y}_{1:(t-s)})$$

- **smoothing**: estimate $\mathbf{X}_t$ from all past and possibly some future observations via the distribution

$$p(\mathbf{x}_t | \mathbf{y}_{1:(t+s)})$$

as well as estimating system parameters. The quantities of interest are computed sequentially in the following recursive calculation:

$$\begin{aligned} \textbf{Prediction} \quad p(\mathbf{x}_t | \mathbf{y}_{1:(t-1)}) &= \int p(\mathbf{x}_t | \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} | \mathbf{y}_{1:(t-1)}) d\mathbf{x}_{t-1} \\ \textbf{Filtering} \quad p(\mathbf{x}_t | \mathbf{y}_{1:t}) &= \frac{p(\mathbf{y}_t | \mathbf{x}_t) p(\mathbf{x}_t | \mathbf{y}_{1:(t-1)})}{p(\mathbf{y}_t | \mathbf{y}_{1:(t-1)})} \end{aligned}$$

The above equations only provide an analytical solution if all densities in the state and observation equation are Gaussian, and both the state and the observation equation are linear. The general form of a linear state space model is given by the following two equations: for $t = 1, \dots, n$

$$\mathbf{Y}_t = \mathbf{F}_t \mathbf{X}_t + \mathbf{d}_t + \mathbf{H}_{1t} \mathbf{Z}_t, \quad (1a)$$

$$\mathbf{X}_{t+1} = \mathbf{G}_t \mathbf{X}_t + \mathbf{e}_t + \mathbf{H}_{2t} \mathbf{Z}_t. \quad (1b)$$

where $\mathbf{Y}_t$ is $(K \times 1)$, $\mathbf{X}_t$ is $(L \times 1)$. In (1)

- $\mathbf{F}_t$ is $(K \times L)$,
- $\mathbf{d}_t$ is $(K \times 1)$,
- $\mathbf{H}_{1t}$ is $(K \times (K + L))$,
- $\mathbf{G}_t$ is $(L \times L)$
- $\mathbf{e}_t$ is $(L \times 1)$
- $\mathbf{H}_{2t}$ is $(L \times (K + L))$.

These matrices are assumed to be non-stochastic. Process $\{\mathbf{Z}_t\}$ is an $((K + L) \times 1)$ process with

$$\mathbb{E}[\mathbf{Z}_t] = \mathbf{0} \quad \text{Var}[\mathbf{Z}_t] = \mathbf{I}_{K+L}.$$

We may write

$$\mathbf{V}_t = \mathbf{H}_{1t} \mathbf{Z}_t \quad \mathbf{W}_t = \mathbf{H}_{2t} \mathbf{Z}_t$$

as the possibly dependent stochastic error terms in the two equations: if $\mathbf{H}_{1t} \mathbf{H}_{2t}^\top = \mathbf{0}$ for all t , then the error terms equations are uncorrelated. If the system matrices do not evolve with time, the state space model is called *time-invariant* or *time-homogeneous*. Often, $\{\mathbf{Z}_t\}$, and initial state vector, $\mathbf{X}_1$, are assumed to be Gaussian random variables.

The formulation is completed by the assumption that the initial state vector $\mathbf{X}_1$ is independent of $\mathbf{Z}_t$ at all times t and has unconditional mean and variance $\mathbf{a}_{1|0}$ and $\mathbf{P}_{1|0}$ respectively.

The Kalman Filter: The Kalman filter is used for prediction, filtering and smoothing.

- *Filtering* is concerned with computing $\mathbb{E}[\mathbf{X}_t | \mathbf{y}_{1:t}]$,
- *Prediction* is concerned with computing $\mathbb{E}[\mathbf{X}_{t+1} | \mathbf{y}_{1:t}]$.
- *Smoothing* is concerned with estimating $\mathbb{E}[\mathbf{X}_t | \mathbf{y}_{1:n}]$, for all $t < n$.

The Kalman filter consists of a set of recursions that compute the latter quantities. For simplicity, assume that

$$\mathbf{H}_{1t} \mathbf{H}_{2t}^\top = \mathbf{0} \quad \mathbf{d}_t = \mathbf{e}_t = \mathbf{0}$$

and

$$\mathbf{H}_{1t} \mathbf{H}_{1t}^\top = \boldsymbol{\Sigma}_t \quad \mathbf{H}_{2t} \mathbf{H}_{2t}^\top = \boldsymbol{\Omega}_t.$$

The Kalman filter recursively computes the quantities

- $\mathbf{a}_{t|t} = \mathbb{E}[\mathbf{X}_t | \mathbf{y}_{1:t}]$,
- $\mathbf{a}_{t+1|t} = \mathbb{E}[\mathbf{X}_{t+1} | \mathbf{y}_{1:t}]$,
- $\mathbf{P}_{t|t} = \text{MSE}[\mathbf{X}_t | \mathbf{y}_{1:t}]$,
- $\mathbf{P}_{t+1|t} = \text{MSE}[\mathbf{X}_{t+1} | \mathbf{y}_{1:t}]$.

where MSE is the *mean-square error* or *one-step ahead prediction variance*. Then, starting with $\mathbf{a}_{1|0}$ and $\mathbf{P}_{1|0}$, $\mathbf{a}_{t|t}$, $\mathbf{a}_{t+1|t}$ and their MSEs are obtained by running for $t = 1, \dots, n$, the recursions

$$\mathbf{r}_t = \mathbf{y}_t - \mathbf{F}_t \mathbf{a}_{t|t-1}, \quad (2a)$$

$$\mathbf{M}_t = \mathbf{F}_t \mathbf{P}_{t|t-1} \mathbf{F}_t^\top + \boldsymbol{\Sigma}_t \quad (2b)$$

$$\mathbf{a}_{t|t} = \mathbf{a}_{t|t-1} + \mathbf{P}_{t|t-1} \mathbf{F}_t^\top \mathbf{M}_t^{-1} \mathbf{r}_t, \quad (2c)$$

$$\mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \mathbf{P}_{t|t-1} \mathbf{F}_t^\top \mathbf{M}_t^{-1} \mathbf{F}_t \mathbf{P}_{t|t-1}, \quad (2d)$$

$$\mathbf{a}_{t+1|t} = \mathbf{G}_t \mathbf{a}_{t|t}, \quad (2e)$$

$$\mathbf{P}_{t+1|t} = \mathbf{G}_t \mathbf{P}_{t|t} \mathbf{G}_t^\top + \boldsymbol{\Omega}_t. \quad (2f)$$

Notes:

- The quantities $\mathbf{r}_t$ and $\mathbf{M}_t$ in (2b) denote the one-step-ahead error in forecasting $\mathbf{y}_t$ conditional on the information set at time $t - 1$ and its MSE respectively. This facilitates computation of the likelihood function as in the innovations algorithm.
- Under Gaussianity, the quantities $\mathbf{a}_{t|t}$ and $\mathbf{a}_{t|t-1}$ are optimal (in MSE) estimators of $\mathbf{x}_t$ conditional on the available information at after and before the t th data point is observed; In general, $\mathbf{a}_{t|t}$ and $\mathbf{a}_{t|t-1}$ are optimal only within the class of **linear** estimators, that is, they are the *minimum mean square linear estimators* of $\mathbf{x}_t$ conditional on $\mathbf{y}_{1:t}$ and $\mathbf{y}_{1:(t-1)}$.
- Finally, equations (2c) and (2e) can be combined together to yield recursions that only compute the one-step-ahead prediction estimates of $\mathbf{x}_{t+1}$ given $\mathbf{y}_{1:t}$. Similarly, by taking together (2d) and (2f), a single set of recursions for the MSE is obtained, which goes directly from $\mathbf{P}_{t|t-1}$ to $\mathbf{P}_{t+1|t}$.

The resulting recursions, along with (2b), are

$$\mathbf{a}_{t+1|t} = \mathbf{G}_t \mathbf{a}_{t|t-1} + \mathbf{K}_t \mathbf{r}_t \quad (3a)$$

$$\mathbf{K}_t = \mathbf{G}_t \mathbf{P}_{t|t-1} \mathbf{F}_t^\top \mathbf{M}_t^{-1} \quad (3b)$$

$$\mathbf{P}_{t+1|t} = \mathbf{G}_t \mathbf{P}_{t|t-1} \mathbf{L}_t^\top + \boldsymbol{\Omega}_t \quad (3c)$$

$$\mathbf{L}_t = \mathbf{G}_t - \mathbf{K}_t \mathbf{F}_t. \quad (3d)$$

Parameter Estimation: The likelihood for data $\mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_n)$ can be constructed as

$$p(\mathbf{y}_1, \dots, \mathbf{y}_n) = p(\mathbf{y}_n | \mathbf{y}_{1:(n-1)}) \cdots p(\mathbf{y}_2 | \mathbf{y}_1) p(\mathbf{y}_1) = \prod_{t=1}^n p(\mathbf{y}_t | \mathbf{y}_{1:(t-1)}).$$

For a Gaussian model, by taking conditional expectations on both sides of the observation equation, with $\mathbf{d}_t \equiv 0$, then

$$\mathbb{E}[\mathbf{Y}_t | \mathbf{y}_{1:(t-1)}] = \mathbf{F}_t \mathbf{a}_{t|t-1} \quad \text{Var}[\mathbf{Y}_t | \mathbf{y}_{1:(t-1)}] = \mathbf{M}_t.$$

Crucially, the one-step-ahead prediction density $p(\mathbf{y}_t | \mathbf{y}_{1:(t-1)})$ is the density of a multivariate normal random variable with mean $\mathbf{F}_t \mathbf{a}_{t|t-1}$ and covariance matrix $\mathbf{M}_t$. Thus, the log-likelihood function is given by

$$\ell = -\frac{n}{2} \log(2\pi) - \frac{1}{2} \sum_{t=1}^n \log(\det \mathbf{M}_t) - \frac{1}{2} \sum_{t=1}^n \mathbf{r}_t^\top \mathbf{M}_t^{-1} \mathbf{r}_t.$$

where

$$\mathbf{r}_t = (\mathbf{y}_t - \mathbf{F}_t \mathbf{a}_{t|t-1})$$

Typically, numerical procedures are used in order to maximize the log-likelihood to obtain the ML estimates of the parameters θ which are consistent and asymptotically normal.

Smoothing: A *smoothing* algorithm can be applied to a state space model; given a fixed set of data, estimates of the state vector are computed at each time t in the sample period taking into account the full information set available.

- This algorithm computes $\mathbf{a}_{t|n} = \mathbb{E}[\mathbf{X}_t | y_{1:n}]$ along with its MSE, $\mathbf{P}_{t|n}$, for all $t = 1, \dots, n-1$ via a set of backward recursions and use the quantities $\mathbf{a}_{t+1|t}$, $\mathbf{a}_{t|t}$ and the MSE matrices $\mathbf{P}_{t|t}$, $\mathbf{P}_{t+1|t}$, which are obtained from (2).
- To obtain $\mathbf{a}_{t|n}$ and $\mathbf{P}_{t|n}$, we start with $\mathbf{a}_{n|n}$ and $\mathbf{P}_{n|n}$ and we run backwards for $t = n-1, \dots, 0$

$$\begin{aligned} \mathbf{a}_{t|n} &= \mathbf{a}_{t|t} + \mathbf{P}_t^* (\mathbf{a}_{t+1|n} - \mathbf{a}_{t+1|t}) \\ \mathbf{P}_{t|n} &= \mathbf{P}_{t|t} + \mathbf{P}_t^* (\mathbf{P}_{t+1|n} - \mathbf{P}_{t+1|t}) \mathbf{P}_t^* \\ \mathbf{P}_t^* &= \mathbf{P}_t \mathbf{G}_t^\top \mathbf{P}_{t+1|t}^{-1}. \end{aligned}$$

If the state space model is not Gaussian, the likelihood can still be constructed in the same way using the minimum mean square linear estimators of the state vector. However, the estimators $\hat{\theta}$ that maximize the likelihood are the *quasi-maximum likelihood* (QML) estimators of the parameters, which are also consistent and asymptotically normal.

EXAMPLE: AIR PASSENGERS DATA The data below represent monthly counts (in thousands) of international airline passengers, 1949 to 1960 ($n = 144$). There is evidence of a trend, and a twelve month periodicity. The usual model for these data $\{y_t\}$ is

$$y_t = m_t + s_t + v_t$$

where m_t is the trend, and s_t is the periodic function with period 12 (so that $s_{t+12k} = s_t$ for each t , and $k = 1, 2, \dots$ such that $s_1 + s_2 + \dots + s_{12} = 0$; note that the periodicity constraints imply that for $t = 12, 13, \dots$

$$s_{t+1} = -s_t - s_{t-1} - \dots - s_{t-10}$$

Both components are presumed deterministic. A generalization of the model allows for a *stochastic level*, *stochastic trend*, and *stochastic seasonal* component.

- A random level with random trend model has $\{M_t\}$ as a stochastic process

$$M_t = M_{t-1} + B_{t-1} + W_{1,t-1}$$

where $\{B_t\}$ is another stochastic process with

$$B_t = B_{t-1} + W_{2,t-1}$$

where $\{W_{1t}\}$ and $\{W_{2t}\}$ are independent white noise processes with variances σ_1^2 and σ_2^2 .

- For the stochastic periodic component consider the series $\{S_t\}$ given by

$$S_{t+1} = -S_t - S_{t-1} - \dots - S_{t-10} + W_{3t}$$

where $\{W_{3t}\}$ is another independent white noise process with variance σ_3^2 .

Using the stochastic version of the model above

$$Y_t = M_t + S_t + V_t$$

and defining the $L = 13$ -dimensional state vector $\mathbf{X}_t = [M_t \ B_t \ S_t \ S_{t-1} \ \dots \ S_{t-10}]^\top$, a proposed state space model for the air passenger data takes the form

$$Y_t = \mathbf{F}\mathbf{X}_t + V_t$$

$$\mathbf{X}_{t+1} = \mathbf{G}\mathbf{X}_t + \mathbf{W}_t$$

where Y_t is scalar ($K = 1$). Here $\mathbf{F} = [1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]^\top$ and

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

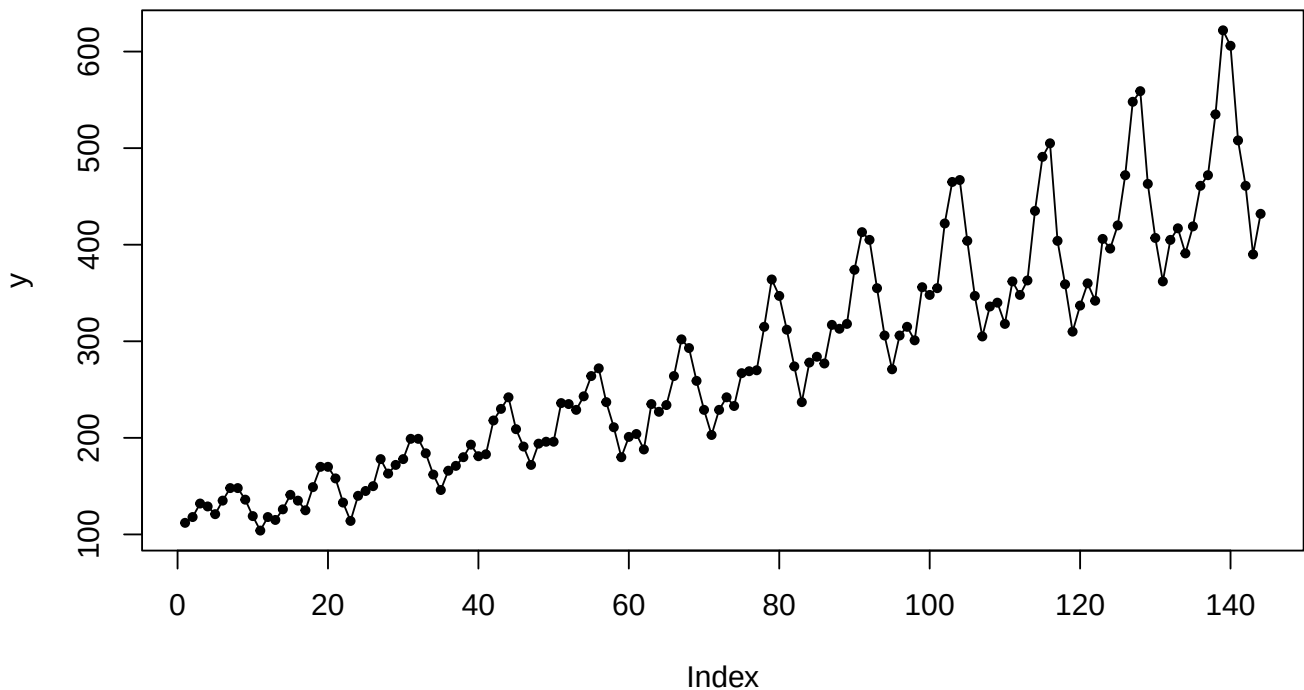
where $\{V_t\} \sim \text{WN}(0, \sigma_V^2)$ and $\{\mathbf{W}_t\} \sim \text{WN}(0, \Omega_t)$ with $\Omega_t = \text{diag}(\sigma_1^2, \sigma_2^2, \sigma_3^2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$.

Using the Kalman Filter, this model can be fitted using maximum (quasi) likelihood; the seventeen parameters of the model to be estimated are

$$(\sigma_1^2, \sigma_2^2, \sigma_3^2, \sigma_V^2, M_1, B_1, S_1, S_2, S_3, \dots, S_{11})$$

After maximizing the likelihood, the states can be estimated.

```
library(MASS)
par(mar=c(4,4,1,0))
y<-as.numeric(AirPassengers);n<-length(y)
plot(y,pch=19,cex=0.6)
lines(y)
```



```
#K is the dimension of Y
#L is the dimension of X

K<-1;L<-13
F.t<-array(0,c(n,K,L));G.t<-array(0,c(n,L,L))
Sigma.t<-array(0,c(n,K,K));Omega.t<-array(0,c(n,L,L))

F1<-c(1,0,1,rep(0,10))
G1<-matrix(0,L,L)
G1[1,1:2]<-1;G1[2,2]<-1
G1[3,3:13]<-1;for(i in 3:12){G1[i+1,i]<-1}
```

```
##Compute Estimates

ml.filter.new<-function(x,Fm,Gm,nv,Lval,Kval,yv){
  Yt<-yv
  Sigmat<-exp(x[4])
```

```

Omevat<-diag(c(exp(x[1:3]),rep(0,10)))

att<-array(0,c(nv,Lval))
apt<-array(0,c(nv+1,Lval))
Ptt<-array(0,c(nv,Lval,Lval))
Ppt<-array(0,c(nv+1,Lval,Lval))
rt<-array(0,c(nv,Kval))
Mt<-array(0,c(nv,Kval,Kval))

apt[1,]<-x[5:length(x)]
Ppt[1,,]<-Omevat

#Filter Forward

loglike<-0
for(i in 1:nv){

  Fmat<-matrix(Fm,nrow=Kval,ncol=Lval); tFmat<-t(Fmat)
  rt[i,]<-Yt[i] - Fmat %*% apt[i,]
  Mt[i,,]<-Fmat %*% Ppt[i,,] %*% t(Fmat) + Sigmat

  if(Sigmat == Inf || Mt[i,,] == 0 || is.na(Mt[i,,])){
    return(Inf)
  }
  Mtinv<-solve(Mt[i,,])

  att[i,]<-apt[i,] + Ppt[i,,] %*% tFmat %*% Mtinv %*% rt[i,]
  Ptt[i,,]<-Ppt[i,,]-Ppt[i,,] %*% tFmat %*% Mtinv %*% Fmat %*% Ppt[i,,]
  apt[i+1,]<-Gm %*% att[i,]
  Ppt[i+1,,]<-Gm %*% Ptt[i,,] %*% t(Gm) + Omevat

  loglike<-loglike-0.5*log(det(matrix(Mt[i,,],Kval,Kval)))-
    0.5*t(rt[i,]) %*% Mtinv %*% rt[i,]
}

return(as.numeric(loglike))
}

```

```

#Perform the optimization using optim
xv<-c(log(170),-10,log(11),log(0.01),rep(0,13))
AirPass.ml<-optim(xv,ml.filter.new,Fm=F1,Gm=G1,nv=n,Kval=K,Lval=L,yv=y,
  method="BFGS",control=list(trace=10,fnscale=-1,maxit=500,abstol=1e-6))

+ initial value 623.061553
+ iter 10 value 544.637705
+ iter 20 value 490.994533
+ iter 30 value 477.866712
+ iter 40 value 475.671693
+ iter 50 value 475.504672
+ iter 60 value 475.490133
+ final value 475.487542
+ converged

xv<-AirPass.ml$par

```



```

P.pt[1,,]<-diag(c(sig.1,sig.2,sig.3,rep(0,10)))
new.a.pt[1,<-X.t[1,]
new.P.pt[1,,]<-diag(c(sig.1,sig.2,sig.3,rep(0,10)))

#Filter Forwards
log.like<-0.0
for(i in 1:n){

  Fmat<-matrix(F.t[i,,],nrow=K,ncol=L); tFmat<-t(Fmat)
  r.t[i,<-Y.t[i,] - Fmat %*% a.pt[i,]
  M.t[i,,]<-Fmat %*% P.pt[i,,] %*% t(Fmat) + Sigma.t[i,,]
  Mt.inv<-solve(M.t[i,,])

  G.t[i,,]<-G1

  a.tt[i,<-a.pt[i,] + P.pt[i,,] %*% tFmat %*% Mt.inv %*% r.t[i,]
  P.tt[i,,]<-P.pt[i,,] - P.pt[i,,] %*% tFmat %*% Mt.inv %*% Fmat %*% P.pt[i,,]
  a.pt[i+1,<- G.t[i,,] %*% a.tt[i,]
  P.pt[i+1,,]<-G.t[i,,] %*% P.tt[i,,] %*% t(G.t[i,,]) + Omega.t[i,,]

  ### Second recursions
  new.r.t[i,<-Y.t[i,] - Fmat %*% new.a.pt[i,]
  K.t[i,,]<-G.t[i,,] %*% new.P.pt[i,,] %*% tFmat %*% solve(M.t[i,,])
  new.a.pt[i+1,<- G.t[i,,] %*% new.a.pt[i,] +
    matrix(K.t[i,,],nrow=L,ncol=K) %*% new.r.t[i,]
  L.t[i,,]<-G.t[i,,] - matrix(K.t[i,,],nrow=L,ncol=K) %*% Fmat
  new.P.pt[i+1,,]<-G.t[i,,] %*% new.P.pt[i,,] %*% t(L.t[i,,]) + Omega.t[i,,]
  log.like<-log.like-0.5*log(det(matrix(M.t[i,,],K,K)))-
    0.5*t(r.t[i,]) %*% Mt.inv %*% r.t[i,]
}

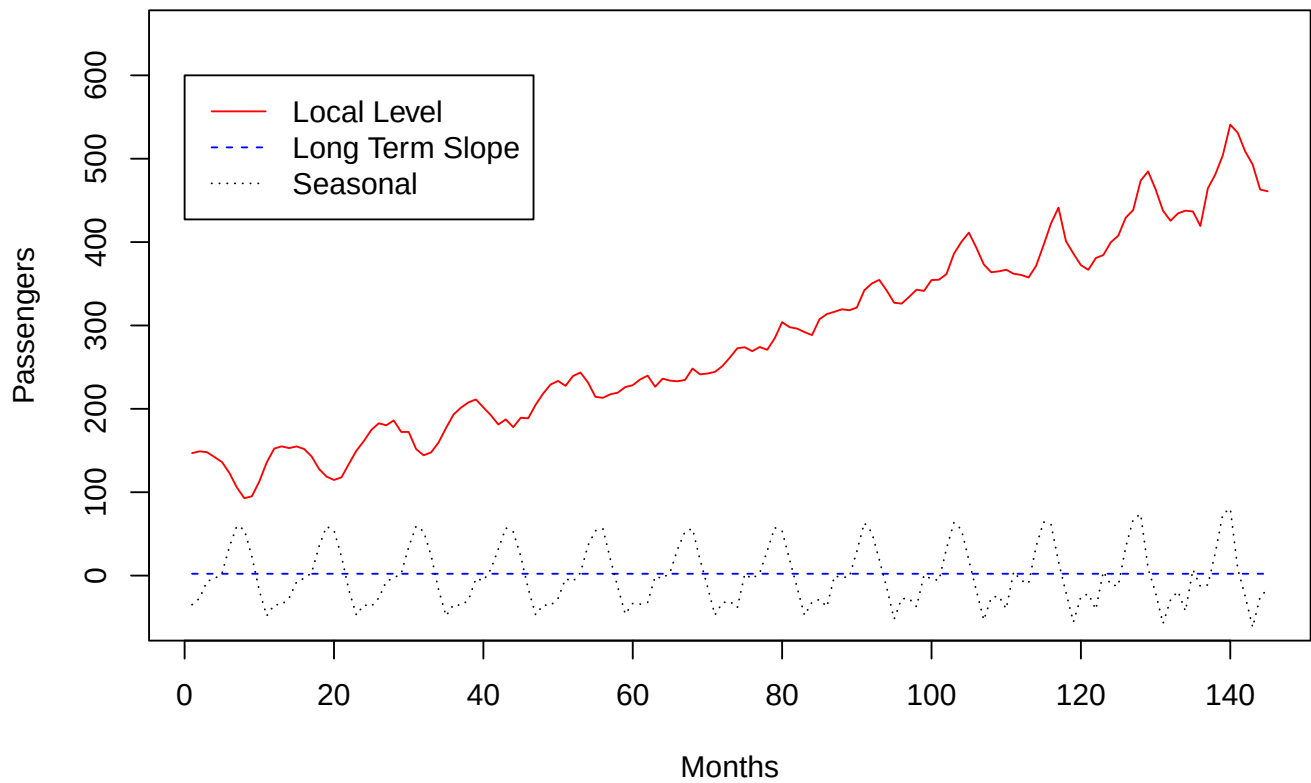
#Smooth Backwards
Pstar.t<-array(0,c(n,L,L))
a.tT<-array(0,c(n,L))
P.tT<-array(0,c(n,L,L))
Pstar.t[n,,]<-P.tt[n,,]
a.tT[n,<-a.tt[n,]
P.tT[n,,]<-P.tt[n,,]

for(i in (n-1):1){
  Pstar.t[i,,]<-P.tt[i,,] %*% t(G.t[i,,]) %*% solve(P.pt[i+1,,]+diag(1e-10,L))
  a.tT[i,<-a.tt[i,]+Pstar.t[i,,] %*% (a.tT[i+1,] - a.pt[i+1,])
  P.tT[i,,]<-P.tt[i,,]+Pstar.t[i,,]%*%(P.tT[i+1,,]-P.pt[i+1,,])%*%Pstar.t[i,,]
}
#a.tT

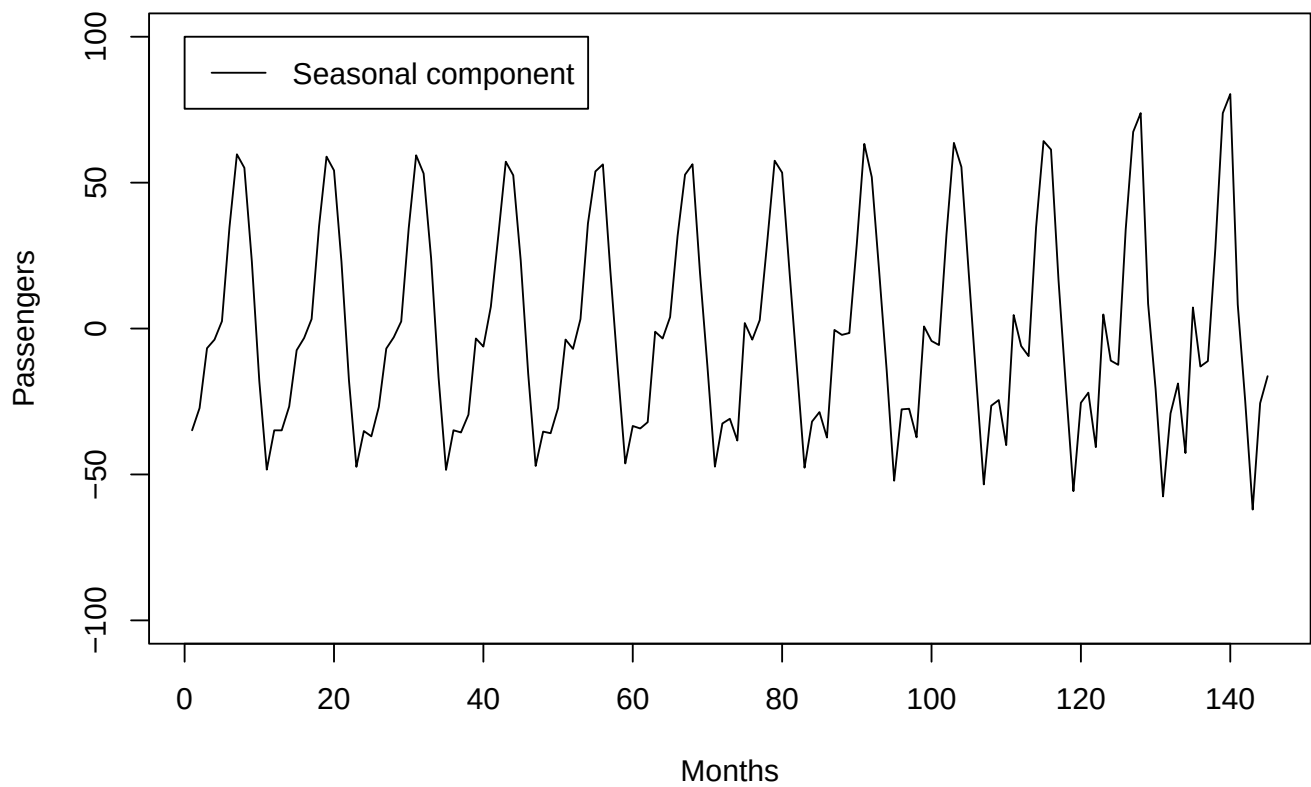
par(mar=c(4,4,1,0))

plot(a.pt[,1],type="l",ylim=range(-50,650),
      ylab="Passengers",xlab="Months",col="red",lwd=1)
lines(a.pt[,2],lty=2,col="blue",lwd=1)
lines(a.pt[,3],lty=3,col="black",lwd=1)
legend(0,600,c("Local Level","Long Term Slope","Seasonal"),
      lty=c(1,2,3),col=c("red","blue","black"),lwd=1)

```



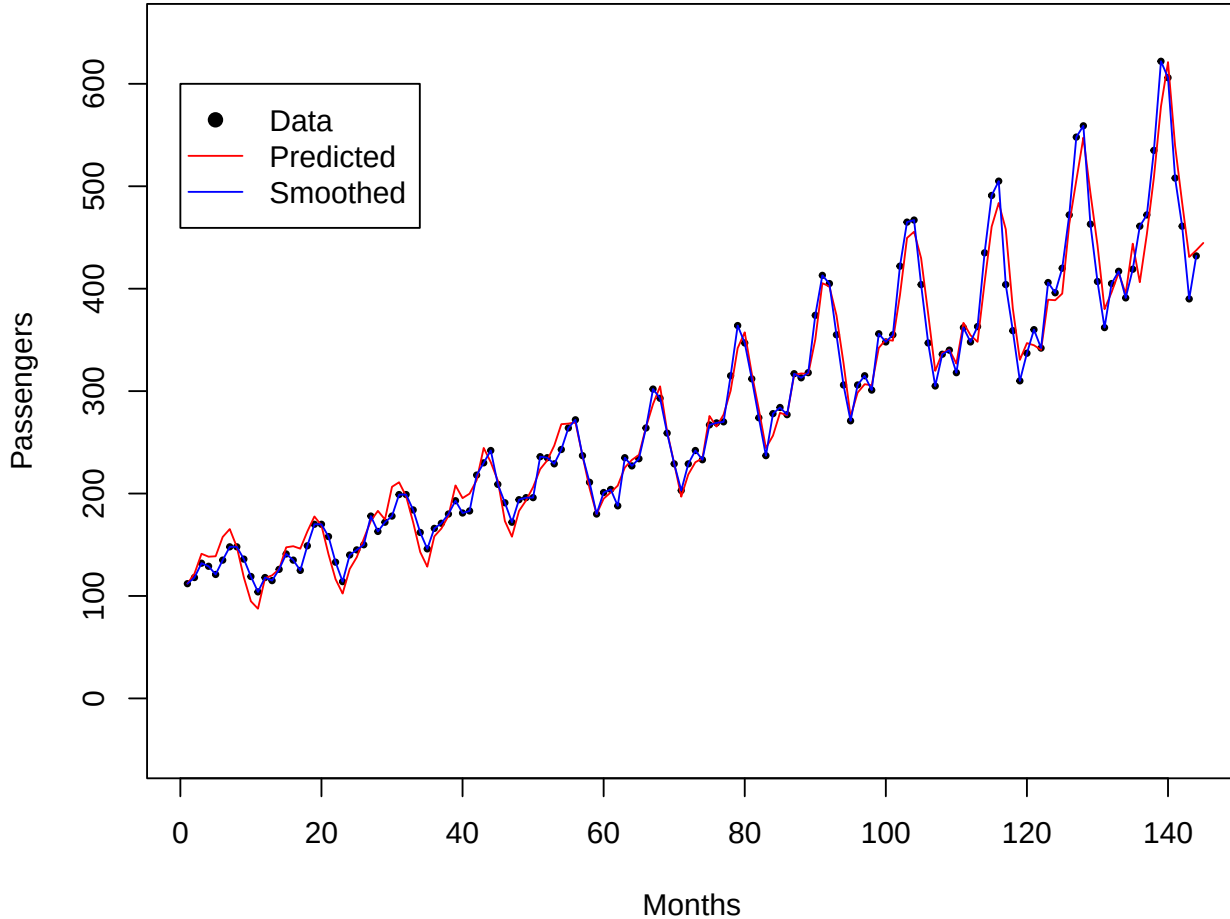
```
plot(a.pt[,3],type="l",ylim=range(-100,100),
     ylab="Passengers",xlab="Months",col="black",lwd=1)
legend(0,100,c("Seasonal component"),col=c("black"),lwd=1)
```



```

par(mar=c(4,4,2,2))
Ypred<-as.vector(matrix(F1,nrow=1) %*% t(a.pt))
Ysmooth<-as.vector(matrix(F1,nrow=1) %*% t(a.tT))
plot(y,type="p",ylim=range(-50,650),pch=19,cex=0.4,
      ylab="Passengers",xlab="Months",lwd=1)
lines(Ypred,col='red')
lines(Ysmooth,col='blue')
legend(0,600,c("Data","Predicted","Smoothed"),lty=c(NA,1,1),
      col=c("black","red","blue"),lwd=1,pch=c(19,NA,NA))

```



For the Air Passenger data the estimates are

$$\sigma_1^2 = 172.384686 \quad \sigma_2^2 = 0.0000229 \quad \sigma_3^2 = 10.9091588 \quad \sigma_V^2 = 0.0032373$$

The initial seasonal component values are

$$146.89, 2.18, -34.86, -34.1, -47.2, -17.12, 22.99, 54.25, 58.64, 33.73, 2.08, -4.24, -6.93$$

We note that this model does explain the observed variation in the data perfectly, as the variance of the prediction residuals does not seem to be constant. Also, the observation noise $\{V_t\}$ has very small variance, $\sigma_V^2 = 0.0032373$, implying that the latent structure is explaining the variance almost perfectly.

Residuals: There are different ways that we can form residuals, for example, we can inspect the prediction residuals

$$r_t = y_t - \mathbf{F}a_{t|t-1}$$

that measure discrepancies in one-step-ahead forecasts, and are used in the likelihood construction, or the filtering residuals

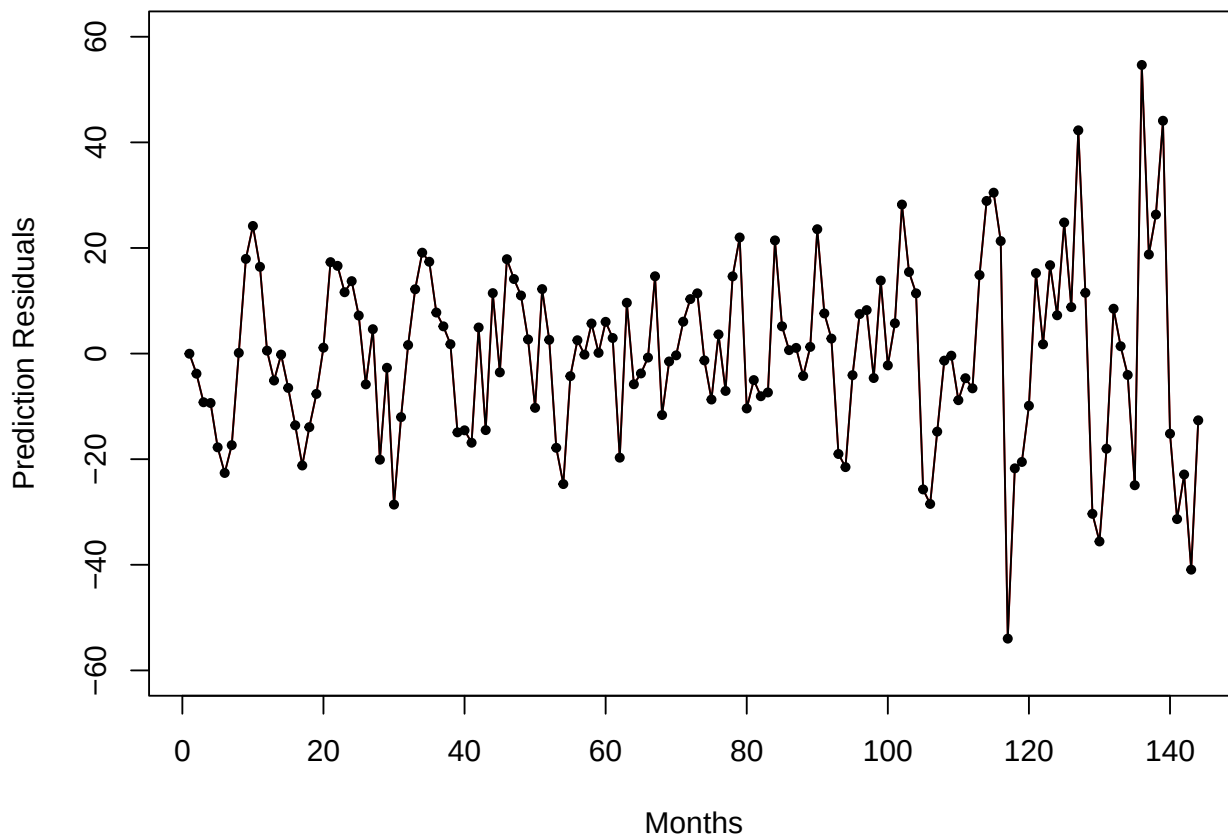
$$r_t = y_t - \mathbf{F}a_{t|t}$$

to assess the fit in-sample, or the smoothing residuals

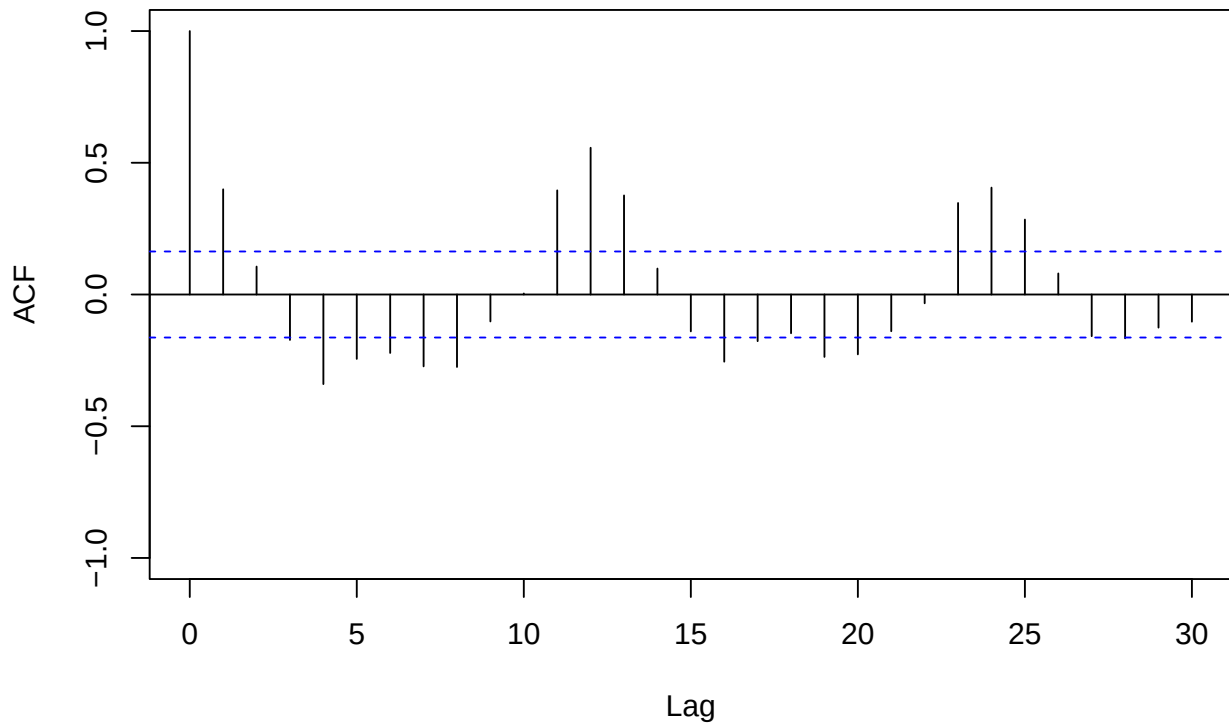
$$r_t = y_t - \mathbf{F}a_{t|n}$$

to assess the fit in light of the complete data set.

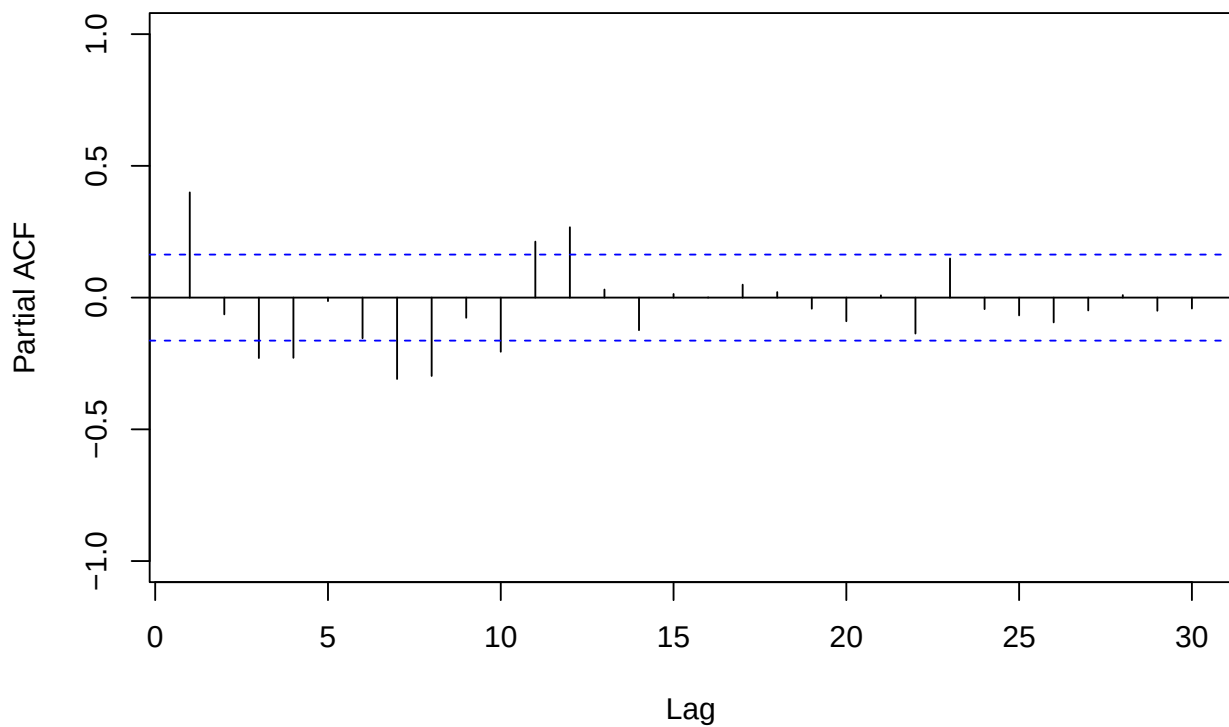
```
Yresids<-Y.t-Ypred[-n]
par(mar=c(4,4,2,2))
plot(Yresids,type="l",ylim=range(-60,60),
     ylab="Prediction Residuals",xlab="Months",col="red",lwd=1)
lines(Yresids);points(Yresids,pch=19,cex=0.6)
```



ACF of Residuals

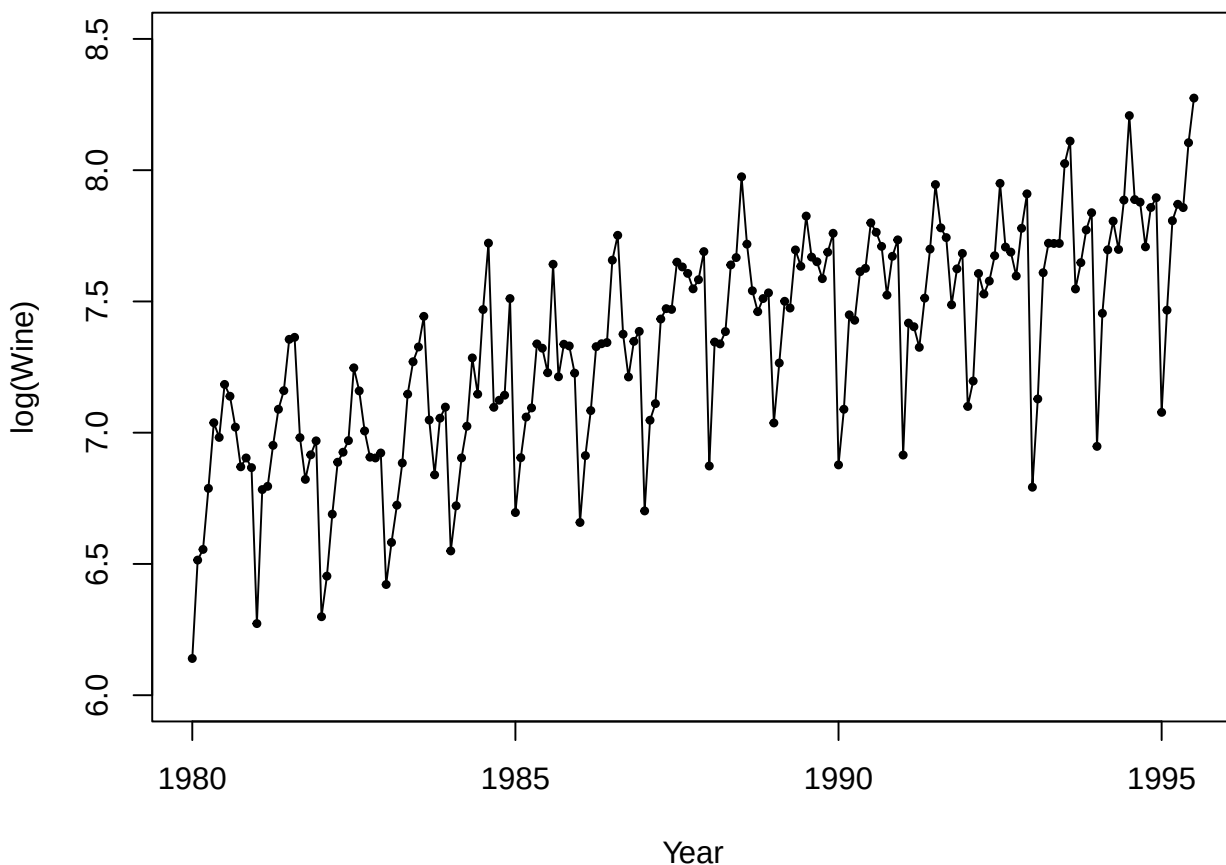


PACF of Residuals



Example: Wine data These data comprise sales of red wine in Australia, from January 1980 to July 1995, measured in thousands of litres. The data are plotted on the log scale below:

```
library(Rssa)
data(AustralianWine)
Wine<-as.numeric(AustralianWine[,4])
n<-length(Wine)
tvec<-c(1:n)
lWine<-log(Wine)
par(mar=c(4,4,2,2))
plot(ts(log(AustralianWine[,4]),start=c(1980,1),freq=12),
      xlab='Year',ylab='log(Wine)',ylim=range(6,8.5))
ind <- 1980 + c(0:186)/12
points(ind, lWine, pch = 19, cex=0.5)
```



```
y<-lWine
```

We fit the same model

$$Y_t = M_t + S_t + V_t$$

with

$$M_t = M_{t-1} + B_{t-1} + W_{1,t-1} \quad B_t = B_{t-1} + W_{2,t-1}$$

yielding state vector $\mathbf{X}_t = [M_t \ B_t \ S_t \ S_{t-1} \ \cdots \ S_{t-10}]^\top$ and

$$Y_t = \mathbf{F}\mathbf{X}_t + V_t$$

$$\mathbf{X}_{t+1} = \mathbf{G}\mathbf{X}_t + \mathbf{W}_t$$

```

#Perform the optimization using optim

xv<-c(0,0,0,0,6,0,rnorm(11,0,0.1))
LogWine.ml<-optim(xv,ml.filter.new,Fm=F1,Gm=G1,nv=n,Kval=K,Lval=L,yv=y,
                  method="BFGS",control=list(trace=10,fnscale=-500,maxit=500,abstol=1e-6))

+ initial   value 0.484079
+ final     value -0.188151
+ converged

xv<-LogWine.ml$par
LogWine.ml

+ $par
+ [1] -2.152643498 -4.053504768 -6.059391595 -3.348508817  6.018558221
+ [6]  0.027019686  0.069457423 -0.130070737  0.002400959  0.051714857
+ [11] -0.111010408  0.355034992  0.016684730 -0.047932588 -0.007425003
+ [16] -0.049708427  0.054312137
+
+ $value
+ [1] 94.07558
+
+ $counts
+ function gradient
+          7          3
+
+ $convergence
+ [1] 0
+
+ $message
+ NULL

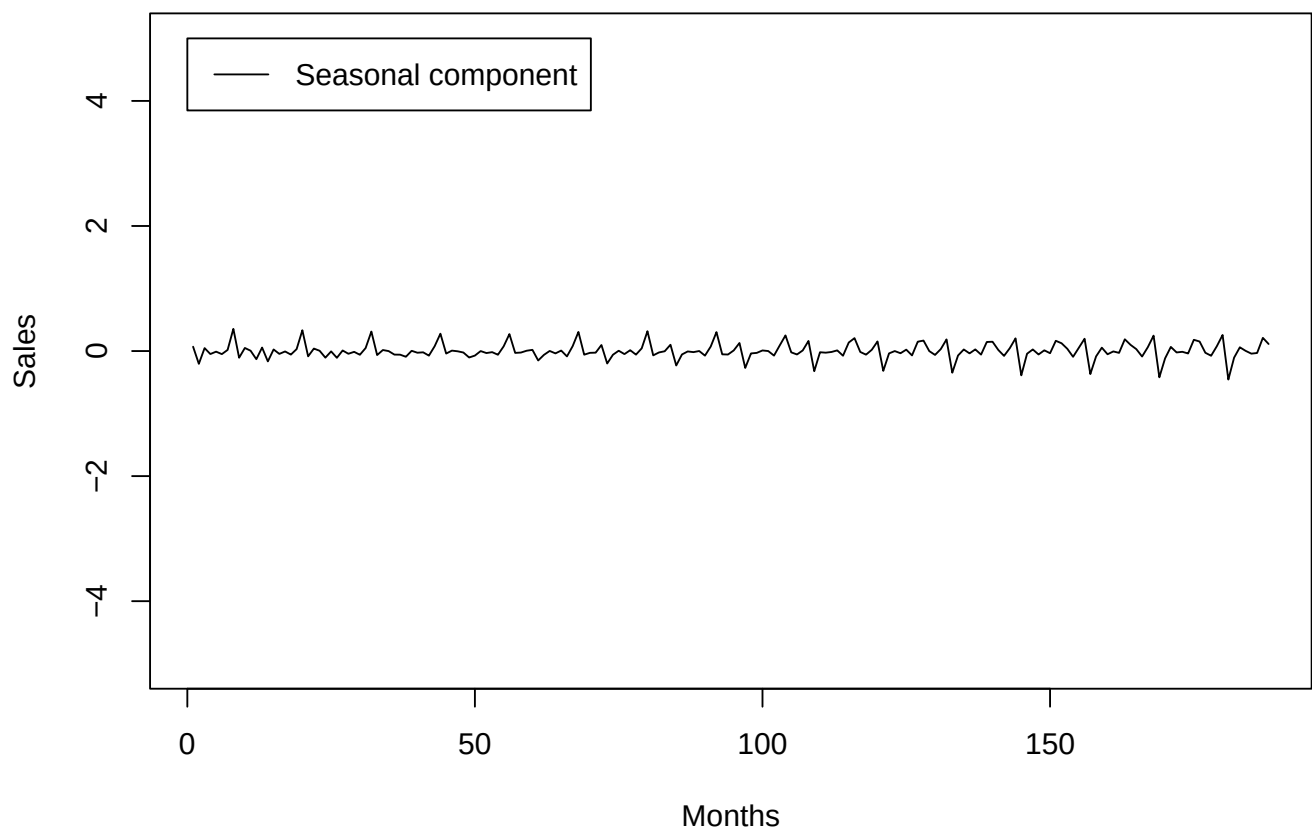
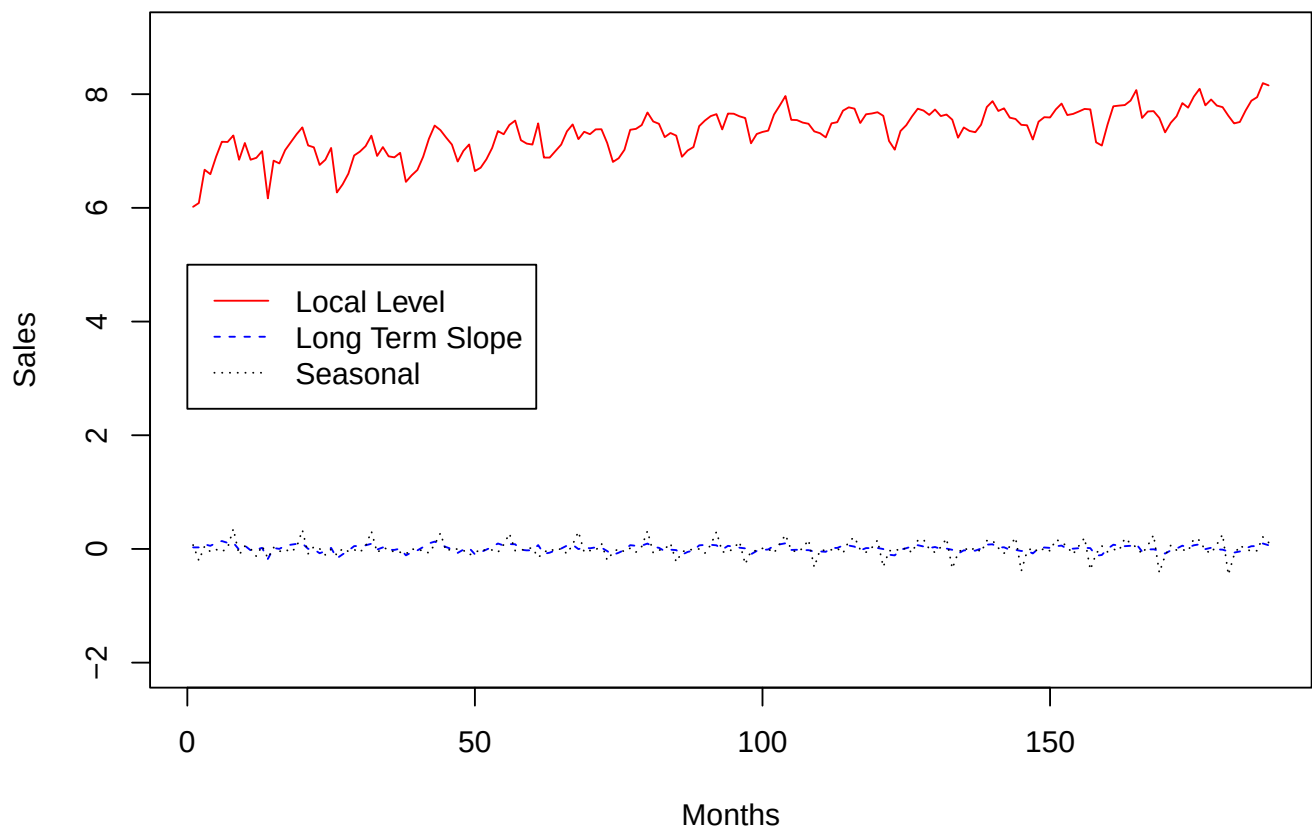
X.t<-matrix(0,nrow=length(y),ncol=length(LogWine.ml$par)-4)
X.t[1,]<-LogWine.ml$par[5:length(LogWine.ml$par)]
sig.1<-exp(LogWine.ml$par[1])
sig.2<-exp(LogWine.ml$par[2])
sig.3<-exp(LogWine.ml$par[3])
sig.v<-exp(LogWine.ml$par[4])

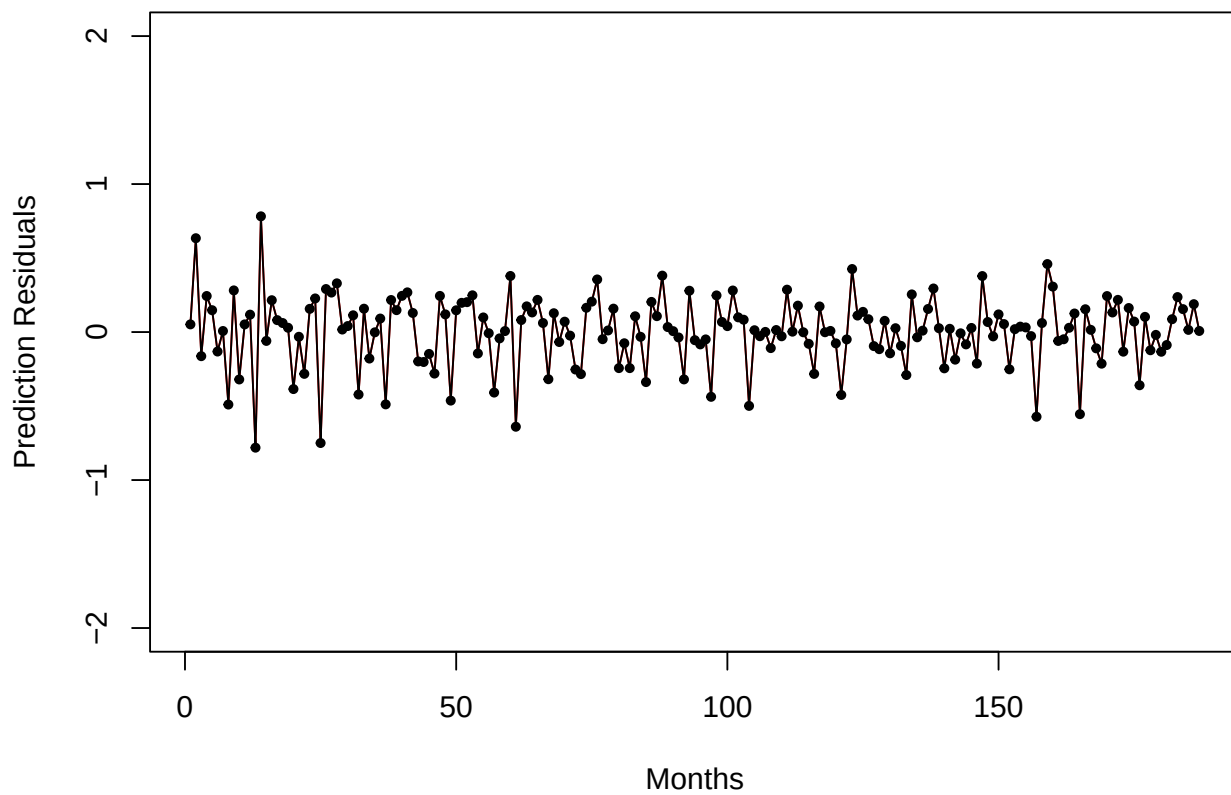
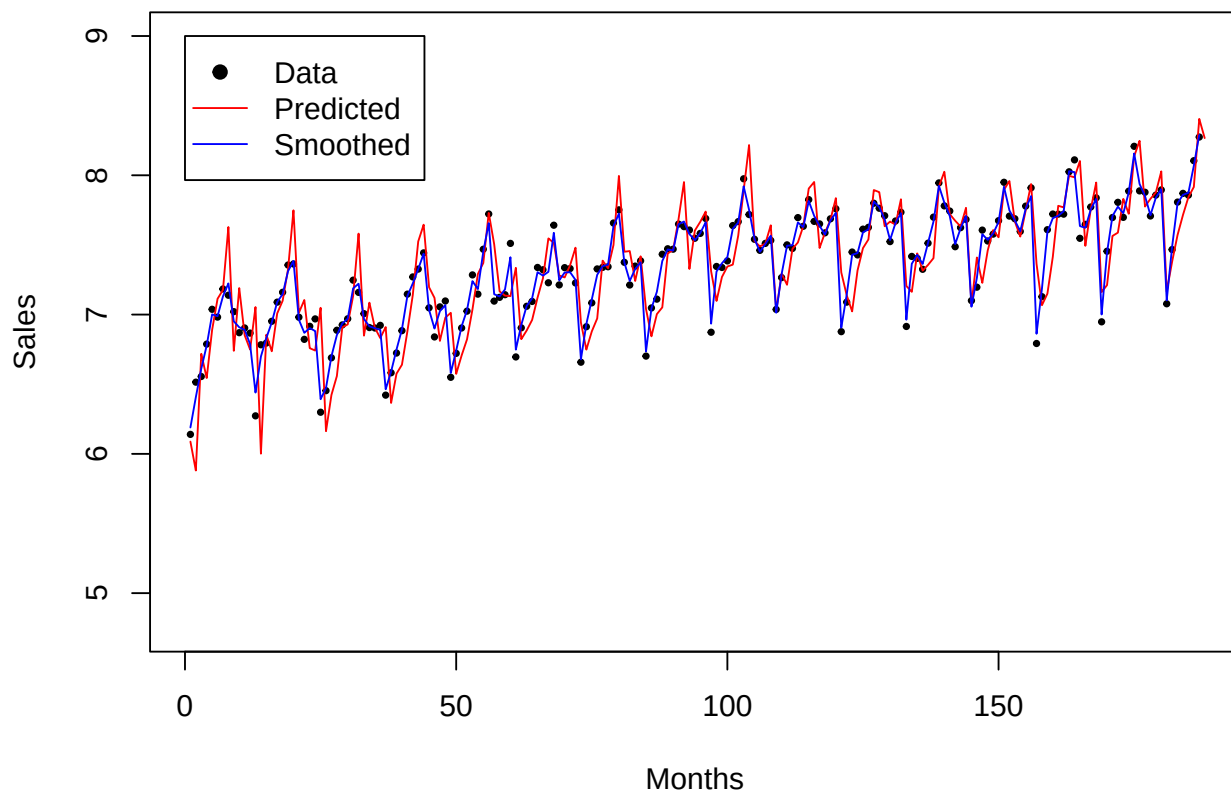
```

For the Wine data the estimates are

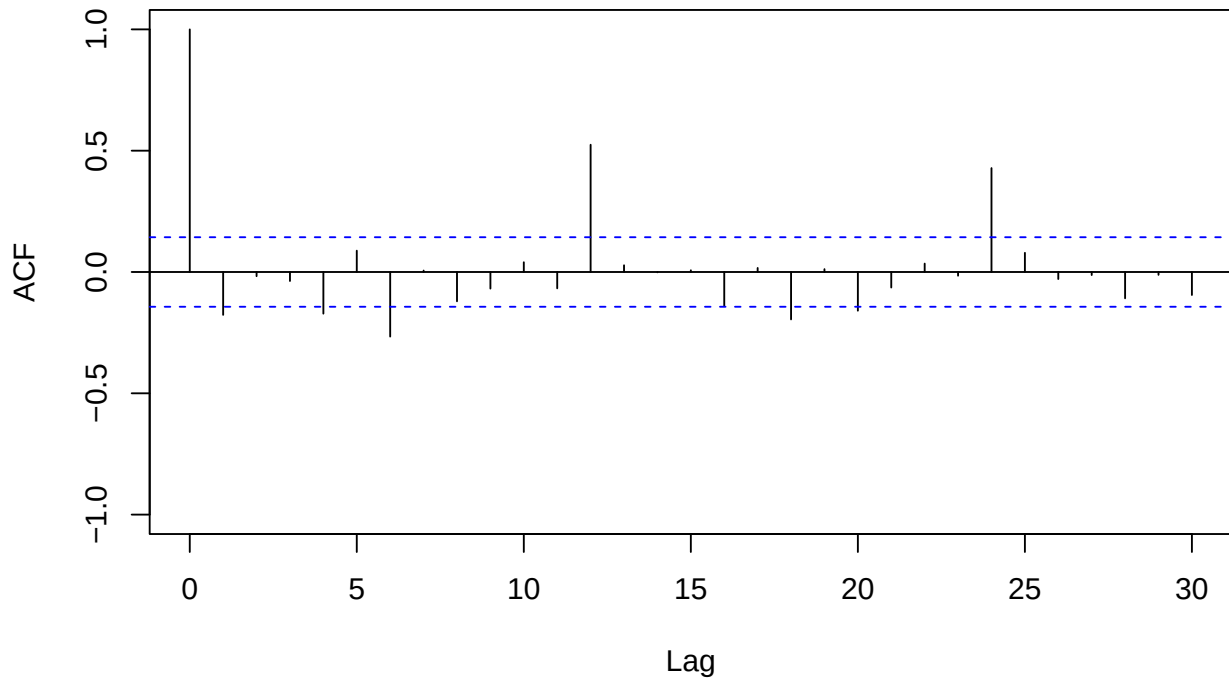
$$\sigma_1^2 = 0.1161766 \quad \sigma_2^2 = 0.0173614 \quad \sigma_3^2 = 0.0023358 \quad \sigma_V^2 = 0.0351367$$

with seasonal component estimates 6.02, 0.03, 0.07, -0.13, 0, 0.05, -0.11, 0.36, 0.02, -0.05, -0.01, -0.05, 0.05.

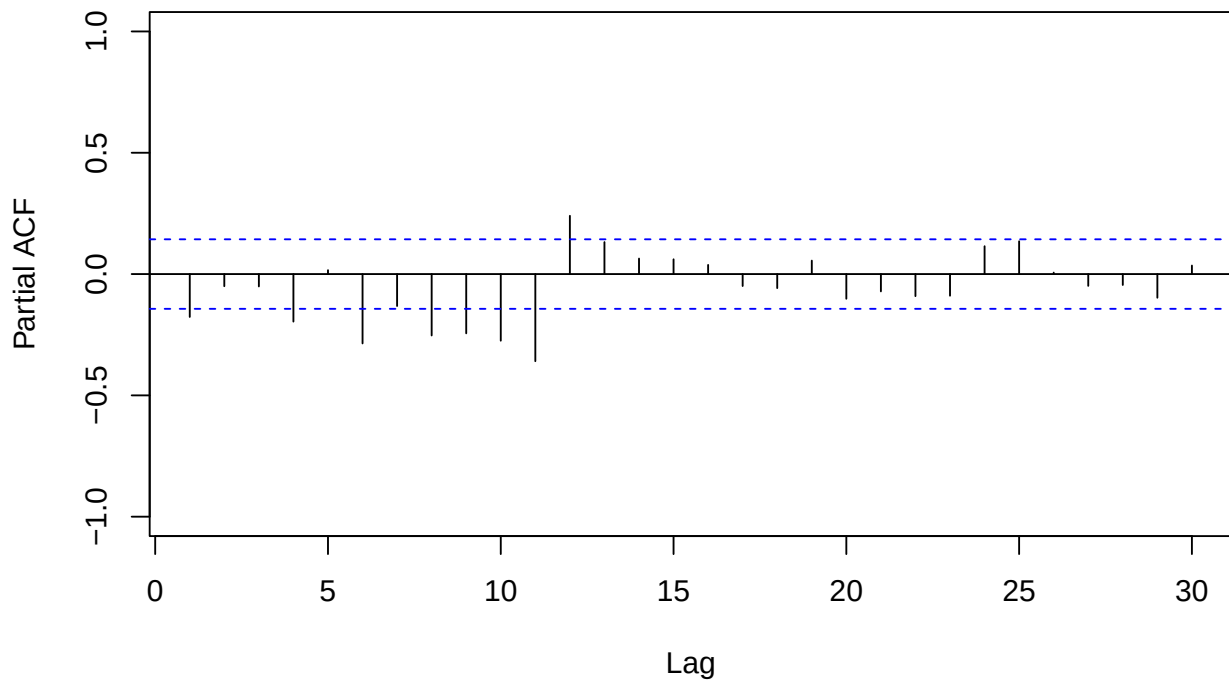




ACF of Residuals



PACF of Residuals



For comparison, we now fit the simpler local-level only model

$$Y_t = M_t + S_t + V_t$$

with

$$M_t = M_{t-1} + W_{1,t-1}$$

yielding state vector $\mathbf{X}_t = [M_t \ S_t \ S_{t-1} \ \cdots \ S_{t-10}]^\top$ and

$$Y_t = \mathbf{F}\mathbf{X}_t + V_t$$

$$\mathbf{X}_{t+1} = \mathbf{G}\mathbf{X}_t + \mathbf{W}_t$$

with $\mathbf{F} = [1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]^\top$ and

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

where $\{V_t\} \sim \text{WN}(0, \sigma_V^2)$ and $\{\mathbf{W}_t\} \sim \text{WN}(0, \Omega_t)$ with $\Omega_t = \text{diag}(\sigma_1^2, \sigma_2^2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$.

```
##Compute Estimates
d<-12
ncyc<-round(n/12)+1
wvec<-rep(0,12)
for(j in 1:12){
  ind<-c(0:ncyc)*12+j
  ind<-ind[ind<=n]
  wvec[j]<-mean(y[ind])
}
svec<-wvec-wvec[1]
xv<-c(0,0,0,6,rnorm(11,0,0.2))
LogWine.ml<-optim(xv,ml.filter.new,Fm=F1,Gm=G1,nv=n,Kval=K,Lval=L,yv=y,
  method="BFGS",control=list(trace=10,fnscale=-500,maxit=500,abstol=1e-6))

+ initial value 0.303577
+ final value -0.322187
+ converged

xv<-LogWine.ml$par
LogWine.ml

+ $par
+ [1] -6.24169457 -6.24955950 -6.45491608 6.05249101 0.10635616 0.10071118
+ [7] -0.28044533 0.05517088 -0.34029346 0.36649542 -0.18395989 -0.31301127
+ [13] -0.03262236 0.27540982 0.05948306
+
+ $value
```

```

+ [1] 161.0936
+
+ $counts
+ function gradient
+      26      2
+
+ $convergence
+ [1] 0
+
+ $message
+ NULL

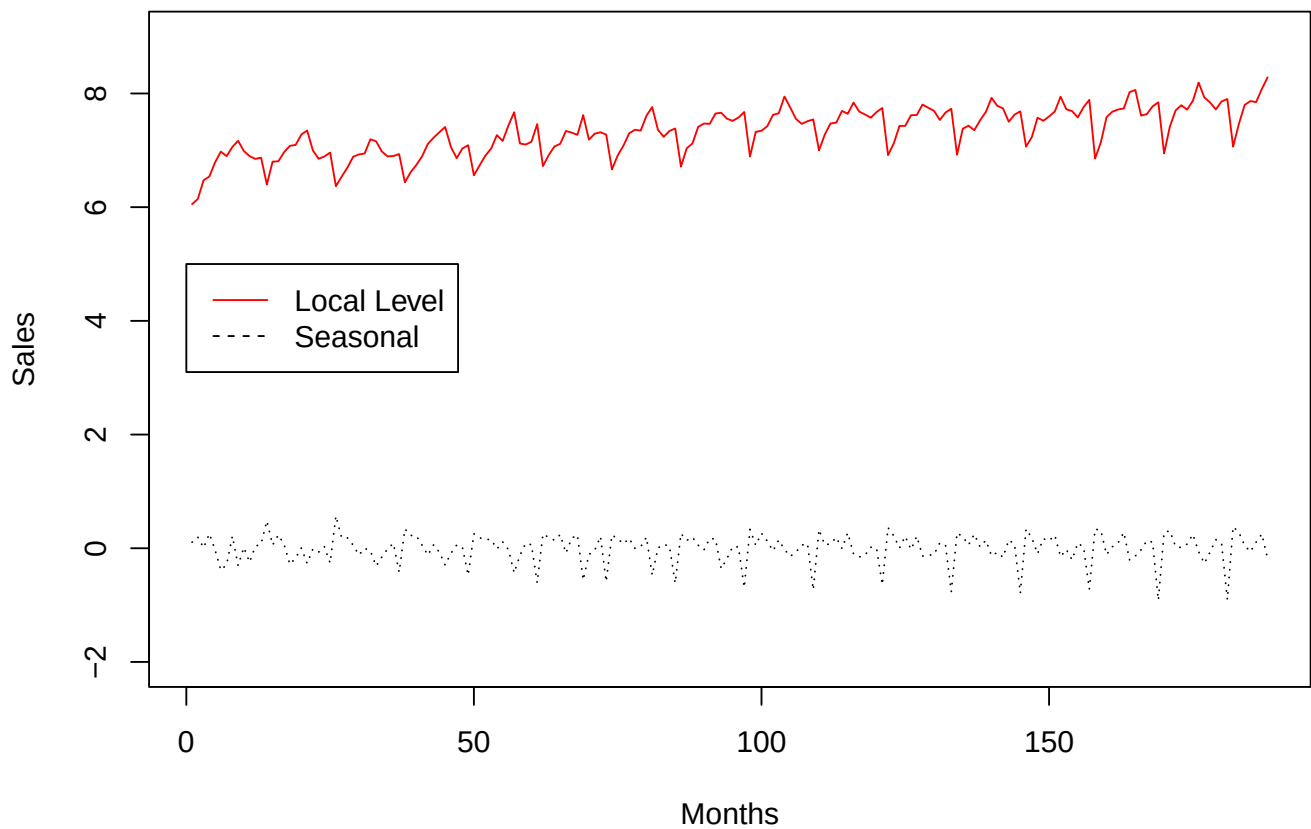
X.t<-matrix(0,nrow=length(y),ncol=length(LogWine.ml$par)-3)
X.t[1,<-LogWine.ml$par[4:length(LogWine.ml$par)]
sig.1<-exp(LogWine.ml$par[1])
sig.2<-exp(LogWine.ml$par[2])
sig.v<-exp(LogWine.ml$par[3])

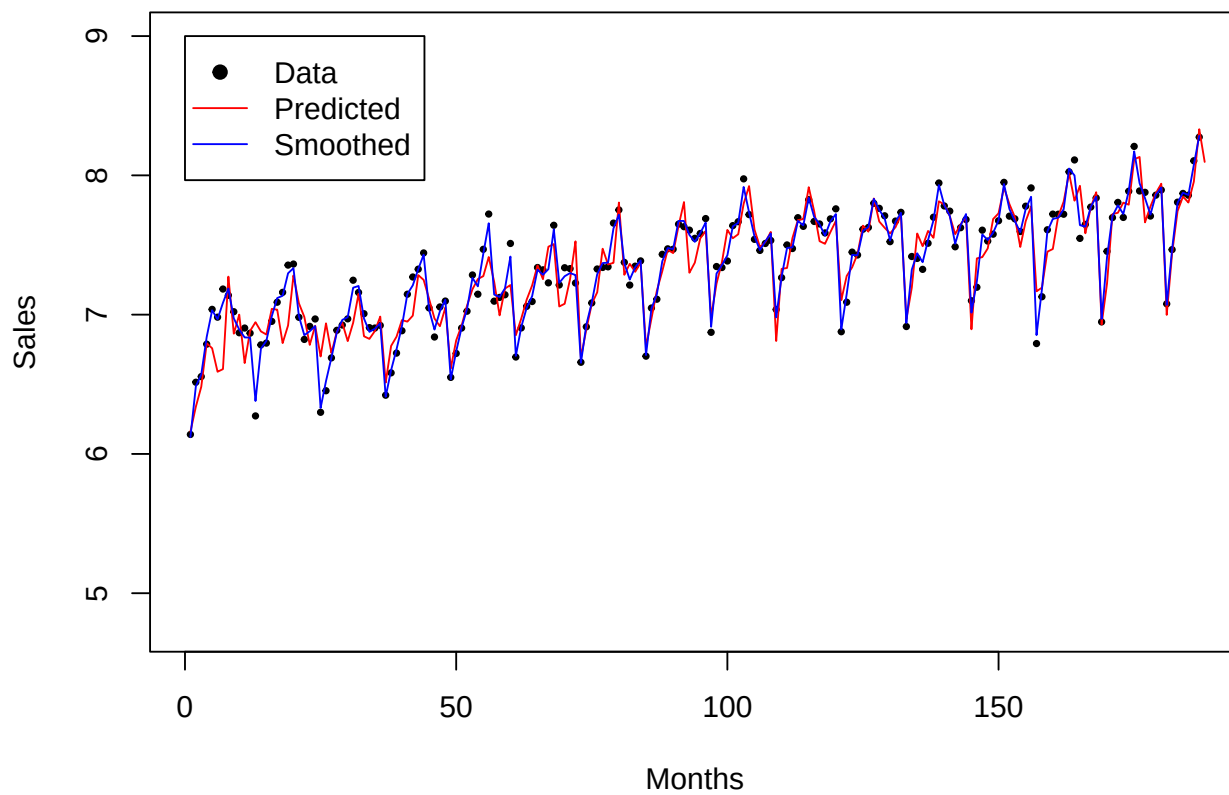
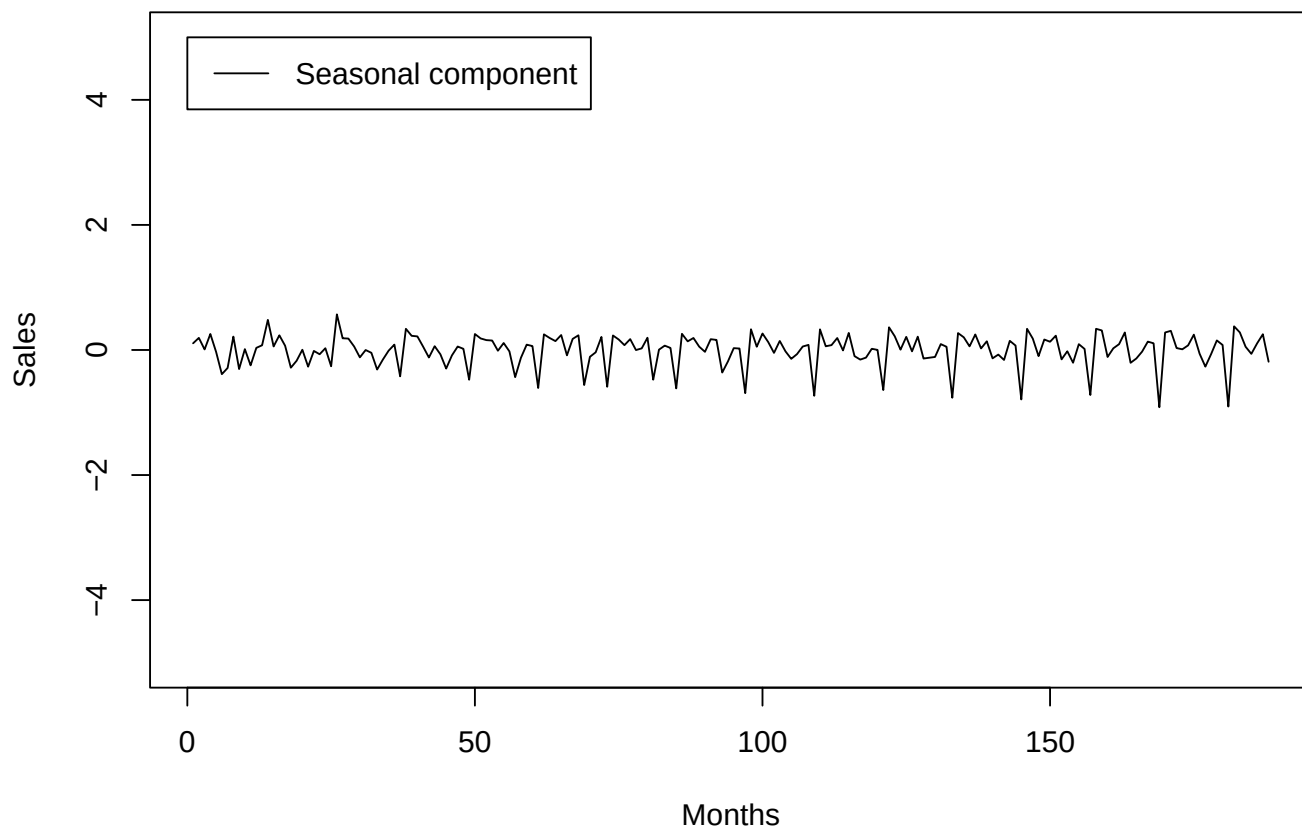
```

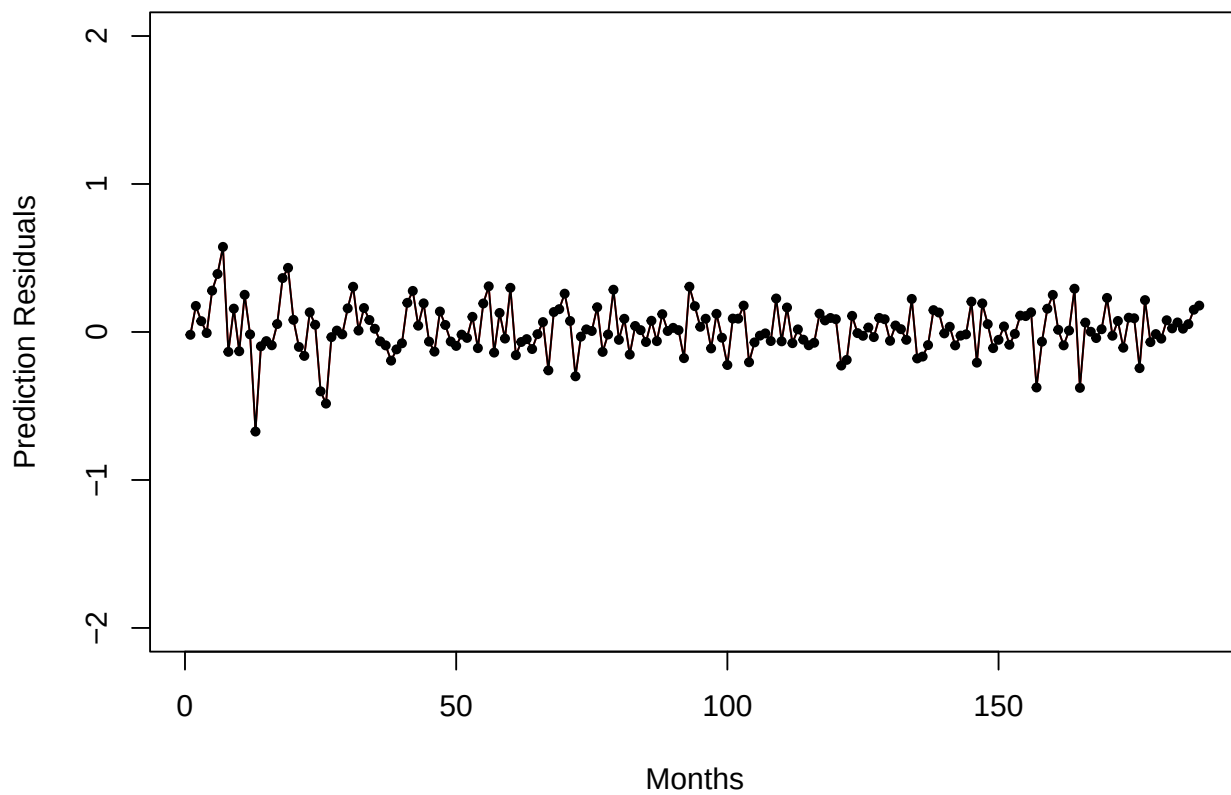
For the new model the estimates are

$$\sigma_1^2 = 0.0019466 \quad \sigma_2^2 = 0.0019313 \quad \sigma_V^2 = 0.0015728$$

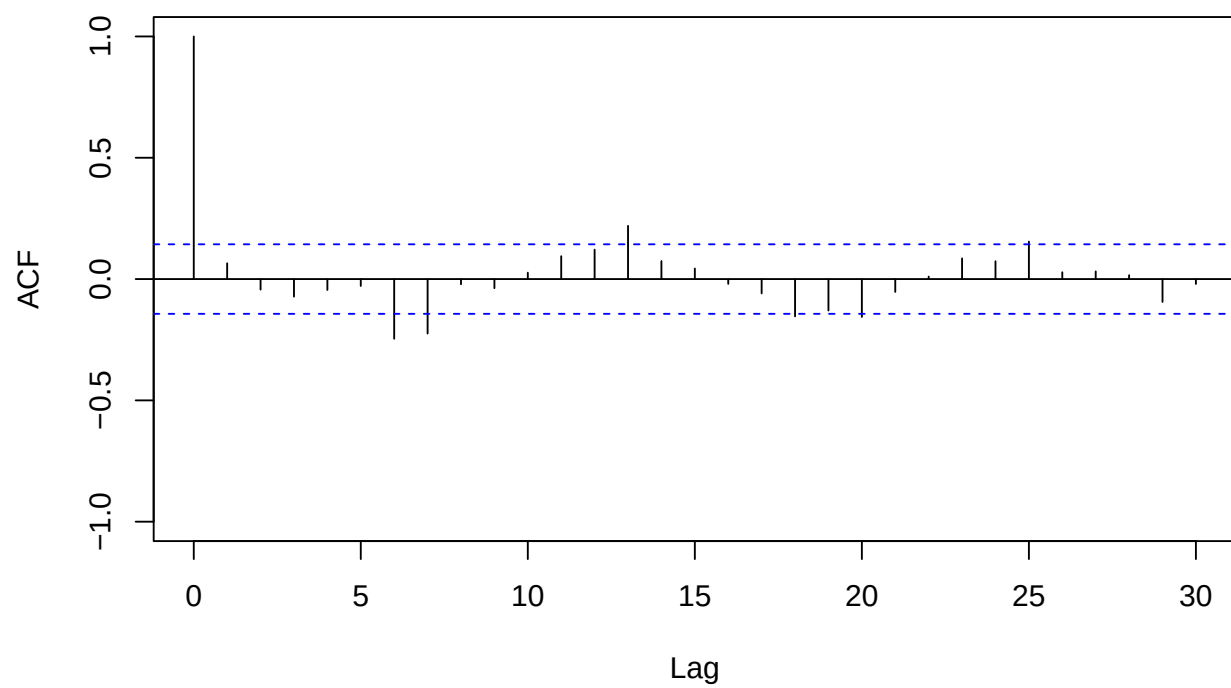
with seasonal component estimates 6.05, 0.11, 0.1, -0.28, 0.06, -0.34, 0.37, -0.18, -0.31, -0.03, 0.28, 0.06.







ACF of Residuals



PACF of Residuals

