MATH 545: INTRODUCTION TO TIME SERIES ANALYSIS
STATE SPACE MODELS AND THE KALMAN FILTER

Let $\mathbf{y}_{1:t} = \{\mathbf{y}_1, \ldots, \mathbf{y}_t\}$ be the observed data, and $\mathbf{X}_t$ be the corresponding unobserved state vector at time t, for $t = 1, \ldots, n$. 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

$$\mathbf{Y}_t = \mathbf{G}_t \mathbf{X}_t + \mathbf{d}_t + \mathbf{H}_{1t} \mathbf{Z}_t, \quad t = 1, \ldots, n \tag{1a}$$

$$\mathbf{X}_{t+1} = \mathbf{F}_t \mathbf{X}_t + \mathbf{e}_t + \mathbf{H}_{2t} \mathbf{Z}_t. \tag{1b}$$

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

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

These matrices are referred to as the *system matrices* and are assumed to be non-stochastic. Process $\{\mathbf{Z}_t\}$ is an $((K + L) \times 1)$ vector of serially uncorrelated (independent), identically distributed disturbances with $\mathbb{E}[\mathbf{Z}_t] = \mathbf{0}$ and $\text{Var}[\mathbf{Z}_t] = \mathbf{I}_{K+L}$, the identity matrix. We may define $\mathbf{W}_t = \mathbf{H}_{1t}\mathbf{Z}_t$ and $\mathbf{V}_t = \mathbf{H}_{2t}\mathbf{Z}_t$ be the possibly dependent stochastic terms in the two equations; if $\mathbf{H}_{1t}\mathbf{H}_{2t}^\top = \mathbf{0}$ for all t, then the measurement and transition equations are uncorrelated. If the system matrices do not evolve with time, the state space model is called *time-invariant* or *time-homogeneous*. If the disturbances, $\mathbf{Z}_t$, and initial state vector, $\mathbf{X}_1$, are assumed to have a normal distribution, then the model is termed Gaussian.

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 simple recursions that compute the latter quantities. For simplicity, assume that

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

and

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

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 $t = 1, \ldots, n$, the recursions

$$\mathbf{r}_t = \mathbf{y}_t - \mathbf{G}_t\mathbf{a}_{t|t-1}, \tag{2a}$$

$$\mathbf{M}_t = \mathbf{G}_t\mathbf{P}_{t|t-1}\mathbf{G}_t^\top + \mathbf{\Sigma}_t \tag{2b}$$

$$\mathbf{a}_{t|t} = \mathbf{a}_{t|t-1} + \mathbf{P}_{t|t-1}\mathbf{G}_t^\top\mathbf{M}_t^{-1}\mathbf{r}_t, \tag{2c}$$

$$\mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \mathbf{P}_{t|t-1}\mathbf{G}_t^\top\mathbf{M}_t^{-1}\mathbf{G}_t\mathbf{P}_{t|t-1}, \tag{2d}$$

$$\mathbf{a}_{t+1|t} = \mathbf{F}_t\mathbf{a}_{t|t}, \tag{2e}$$

$$\mathbf{P}_{t+1|t} = \mathbf{F}_t\mathbf{P}_{t|t}\mathbf{F}_t^\top + \mathbf{\Omega}_t. \tag{2f}$$

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 tth 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

$$\mathbf{a}_{t+1|t} = \mathbf{F}_t\mathbf{a}_{t|t-1} + \mathbf{K}_t\mathbf{r}_t \tag{3a}$$

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

$$\mathbf{P}_{t+1|t} = \mathbf{F}_t\mathbf{P}_{t|t-1}\mathbf{L}_t^\top + \mathbf{\Omega}_t \tag{3c}$$

$$\mathbf{L}_t = \mathbf{F}_t - \mathbf{K}_t\mathbf{G}_t. \tag{3d}$$

Parameter Estimation: Another application of the Kalman filter is the estimation of any unknown parameters θ that appear in the system matrices. The likelihood for data $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ can be constructed as

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

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

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

Crucially, the one-step-ahead prediction density $p\left(\mathbf{y}_t | \mathbf{y}_{1:(t-1)}\right)$ is the density of a multivariate normal random variable with mean $\mathbf{G}_t \mathbf{a}_{t|t-1}$ 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

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

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.

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 $t = 1, \ldots, n-1$ 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).

In particular, 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 $t = n - 1, \ldots, 0$

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

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 $\widehat{\theta}$ 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

$$y_t = m_t + s_t + w_t$$

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 $k = 1, 2, \ldots$ such that $s_1 + s_2 + \cdots + s_{12} = 0$; note that the periodicity constraints imply that for $t = 12, 13, \ldots$.

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

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

  $$M_t = M_{t-1} + B_{t-1} + V_{1(t-1)}$$

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

  $$B_t = B_{t-1} + V_{2(t-1)}$$

  where $\{V_{1t}\}$ and $\{V_{2t}\}$ 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

  $$S_{t+1} = -S_t - S_{t-1} - \cdots - S_{t-10} + V_{3t}$$

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

Using the stochastic version of the model above

$$Y_t = M_t + S_t + W_t$$

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

$$\begin{aligned} Y_t &= \mathbf{G}\mathbf{X}_t + W_t \\ \mathbf{X}_{t+1} &= \mathbf{F}\mathbf{X}_t + \mathbf{V}_t \end{aligned}$$

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

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

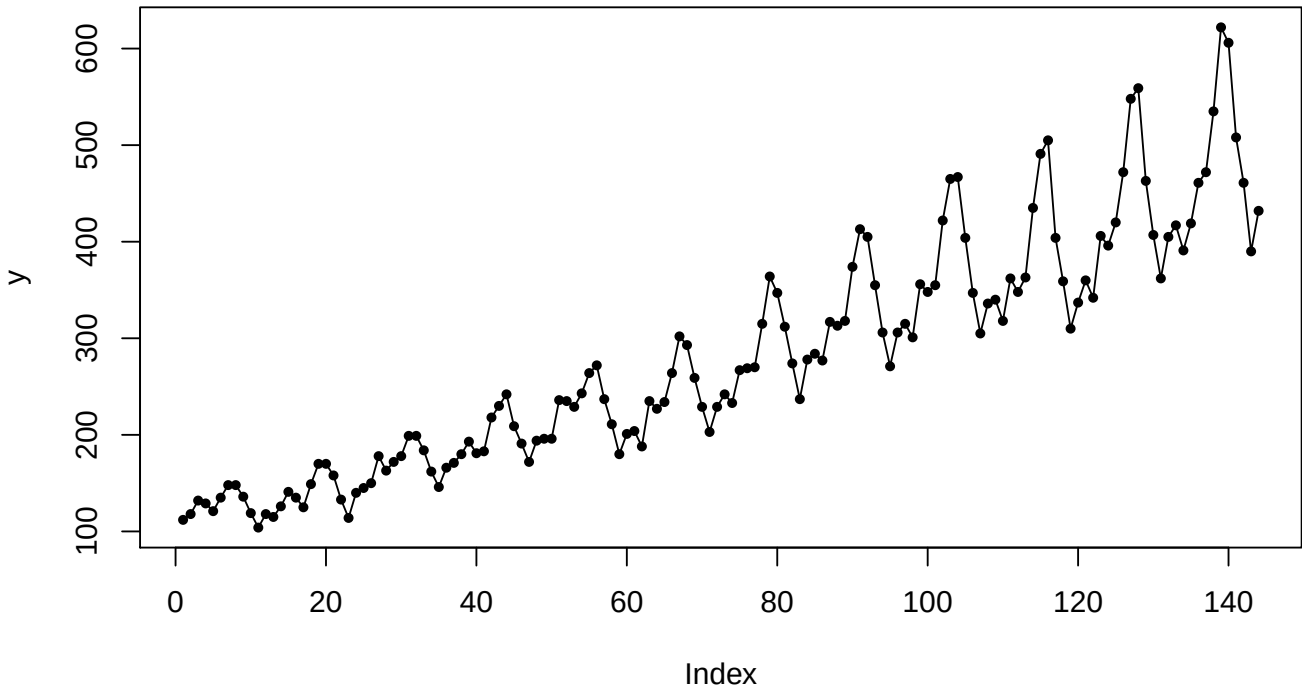
where $\{W_t\} \sim \text{WN}(0, \sigma_W^2)$ and $\{\mathbf{V}_t\} \sim \text{WN}(0, \Omega_t)$ with $\Omega_t = \text{diag}(\sigma_1^2, \sigma_2^2, \sigma_3^2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$.

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

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

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

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

```
##Compute Estimates

ml.filter.new<-function(x,Fm,Gm,nv,Lval,Kval,yv){

        Yt<-yv
        Sigmat<-exp(x[4])
```

```
        Omegat<-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,,]<-Omegat

        #Filter Forward

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

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

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

                att[i,]<-apt[i,] + Ppt[i,,] %*% t(Gmat) %*% Mtinv %*% rt[i,]
                Ptt[i,,]<-Ppt[i,,]-Ppt[i,,] %*% t(Gmat) %*% Mtinv %*% Gmat %*% Ppt[i,,]
                apt[i+1,]<-Fm %*% att[i,]
                Ppt[i+1,,]<-Fm %*% Ptt[i,,] %*% t(Fm) + Omegat

                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
```

```
AirPass.ml

+ $par
+  [1]   5.149729 -10.685894   2.389603  -5.733018 146.892581   2.180959
+  [7] -34.864625 -34.100620 -47.198791 -17.116667  22.988504  54.253096
+ [13]  58.635484  33.733251   2.077881  -4.235745  -6.926070
+
+ $value
+ [1] -475.4875
+
+ $counts
+ function gradient
+       79       66
+
+ $convergence
+ [1] 0
+
+ $message
+ NULL

X.t<-matrix(0,nrow=length(AirPassengers),ncol=length(AirPass.ml$par)-4)
X.t[1,]<-AirPass.ml$par[5:length(AirPass.ml$par)]
sig.1<-exp(AirPass.ml$par[1])
sig.2<-exp(AirPass.ml$par[2])
sig.3<-exp(AirPass.ml$par[3])
sig.w<-exp(AirPass.ml$par[4])
c(sig.w,sig.1,sig.2,sig.3)

+ [1]   0.00323729127 172.38468599564   0.00002286521  10.90915883806

Y.t<-array(0,c(n,K))
for(i in 1:n){
        G.t[i,,]<-G1
        F.t[i,,]<-F1
        Sigma.t[i,,]<-sig.w
        Omega.t[i,,]<-diag(c(sig.1,sig.2,sig.3,rep(0,10)))
        Y.t[i,]<-y[i]
}
```

```
#Produce filtering and smoothing estimates

a.tt<-array(0,c(n,L))
a.pt<-array(0,c(n+1,L))
new.a.pt<-array(0,c(n+1,L))
P.tt<-array(0,c(n,L,L))
P.pt<-array(0,c(n+1,L,L))
new.P.pt<-array(0,c(n+1,L,L))
r.t<-array(0,c(n,K))
new.r.t<-array(0,c(n,K))
M.t<-array(0,c(n,K,K))
F.t<-array(0,c(n,L,L))
L.t<-array(0,c(n,L,K))
K.t<-array(0,c(n,L,L))

a.pt[1,]<-X.t[1,]
```

```
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
for(i in 1:n){

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

        F.t[i,,]<-F1

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

        ###
        new.r.t[i,]<-Y.t[i,] - Gmat %*% new.a.pt[i,]
        L.t[i,,]<-F.t[i,,] %*% new.P.pt[i,,] %*% t(Gmat) %*% solve(M.t[i,,])
        new.a.pt[i+1,]<- F.t[i,,] %*% new.a.pt[i,] +
                         matrix(L.t[i,,],nrow=L,ncol=K) %*% new.r.t[i,]
        K.t[i,,]<-F.t[i,,] -  matrix(L.t[i,,],nrow=L,ncol=K) %*% Gmat
        new.P.pt[i+1,,]<-F.t[i,,] %*% new.P.pt[i,,] %*% t(K.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,]
}

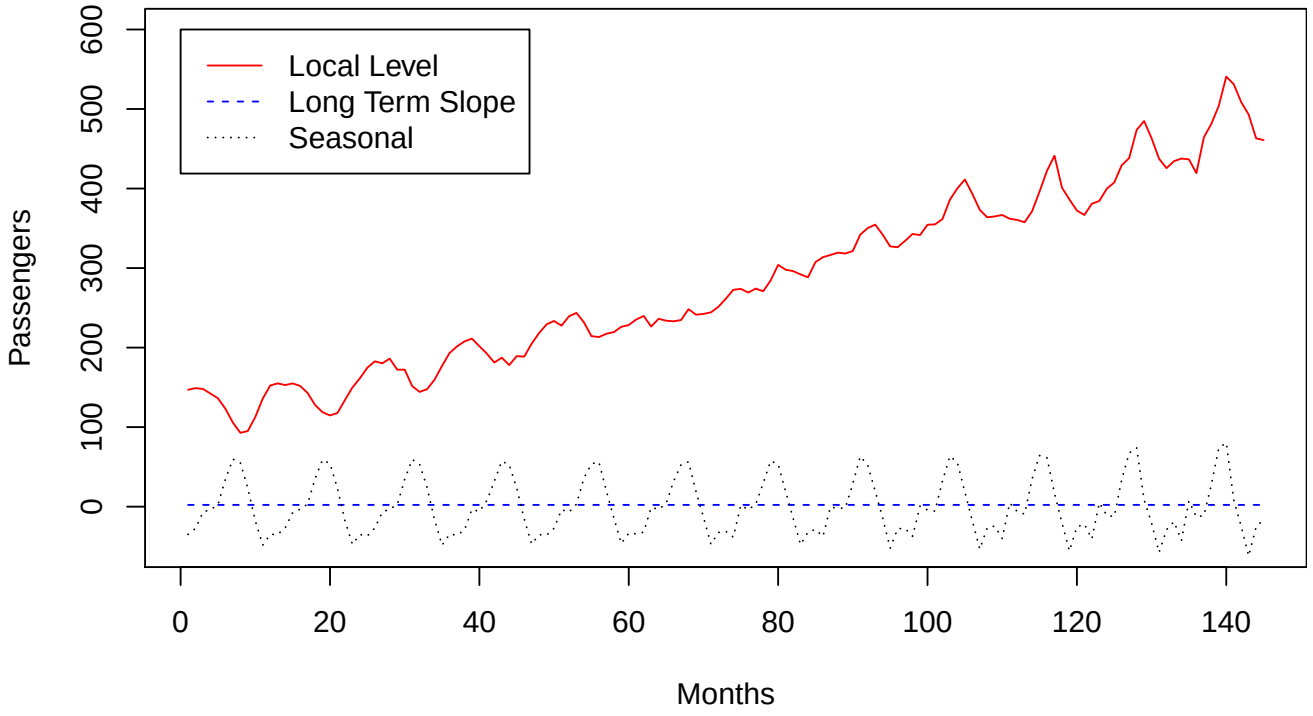
#Smooth Backwards

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,,]

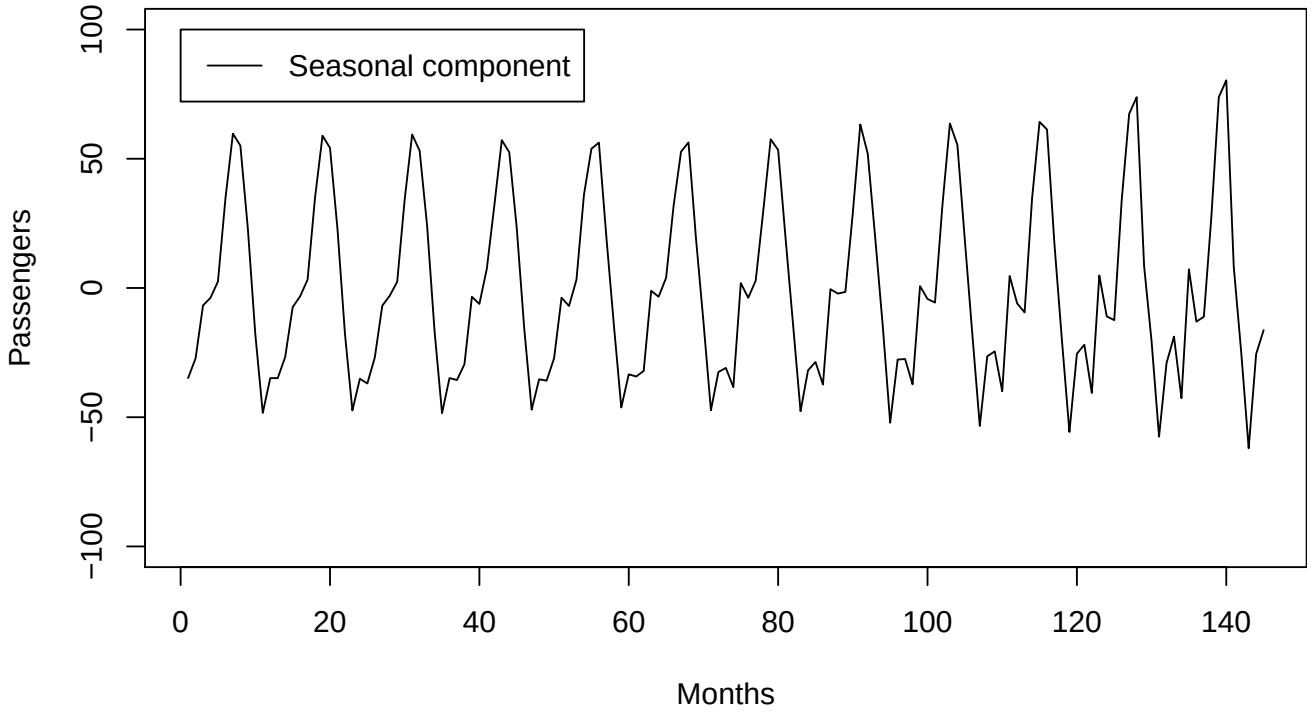
for(i in (n-1):(L-3)){
        Pstar.t[i,,]<-P.tt[i,,] %*% t(F.t[i,,]) %*% P.pt[i+1,,]
        a.tT[i,]<-a.tt[i,]+Pstar.t[i,,] %*% (a.tT[i+1,] - a.pt[i+1,])
        P.tT[i,,]<-P.tt[i,,] + Pstar.t[i,,] %*% (P.tT[i+1,,]-P.pt[i+1,,]) %*% Pstar.t[i,,]
}

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

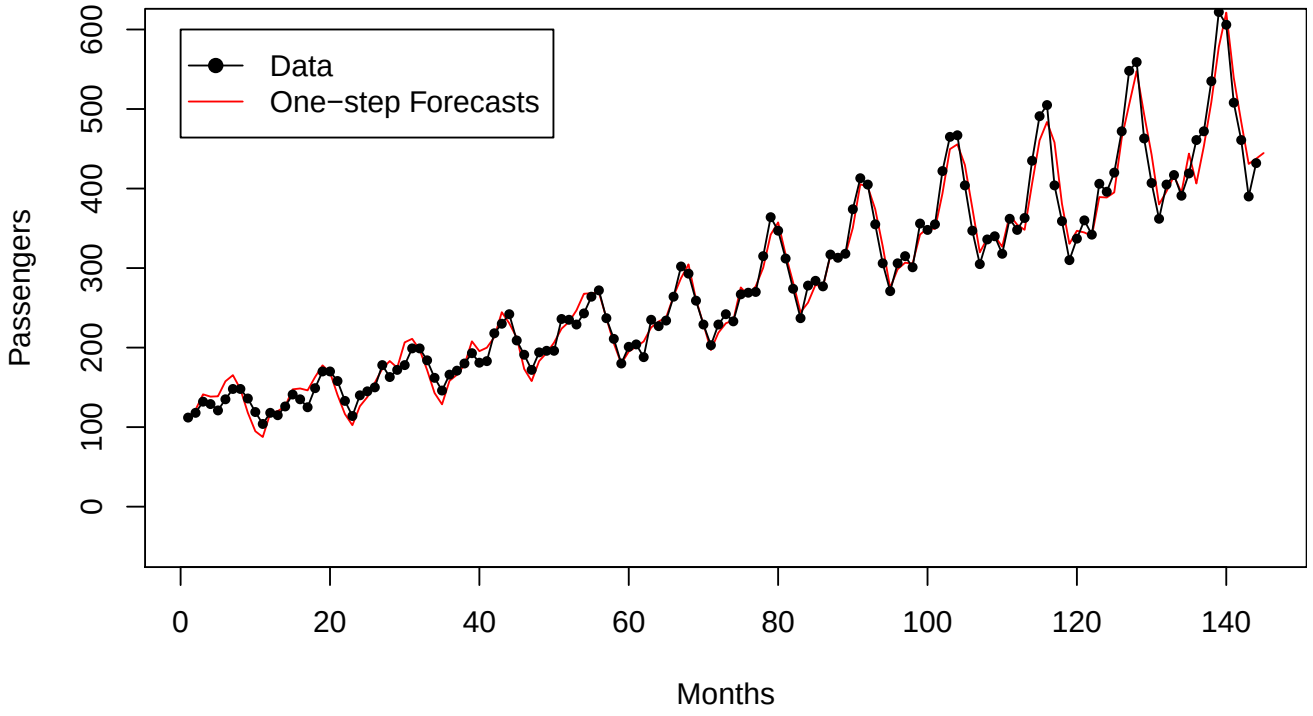
plot(a.pt[,1],type="l",ylim=range(-50,600),ylab="Passengers",xlab="Months",col="red",lwd=1)
lines(a.pt[,2],lty=2,col="blue",lwd=1)
lines(a.pt[,3],lty=3,col="black",lwd=1)
legend(0,600,c("Local Level","Long Term Slope","Seasonal"),
       lty=c(1,2,3),col=c("red","blue","black"),lwd=1)
```



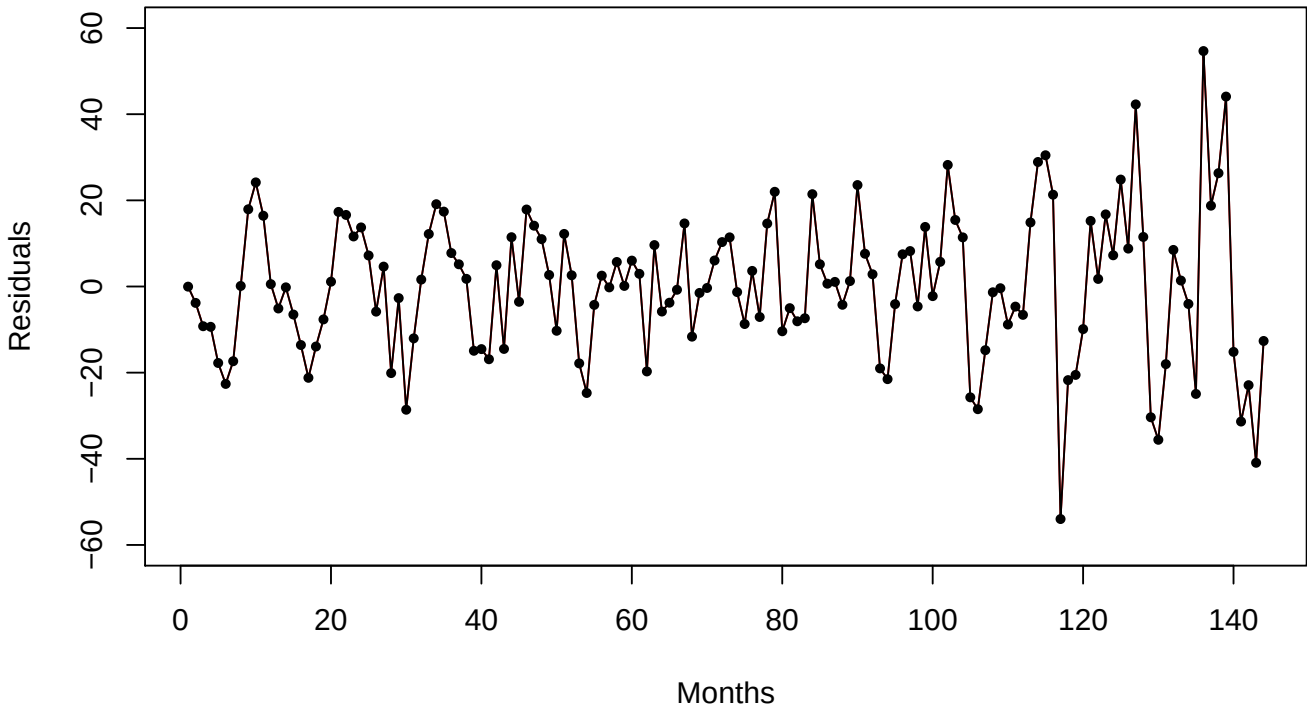

```
plot(a.pt[,3],type="l",ylim=range(-100,100),ylab="Passengers",xlab="Months",col="black",lwd=1)
legend(0,100,c("Seasonal component"),col=c("black"),lwd=1)
```



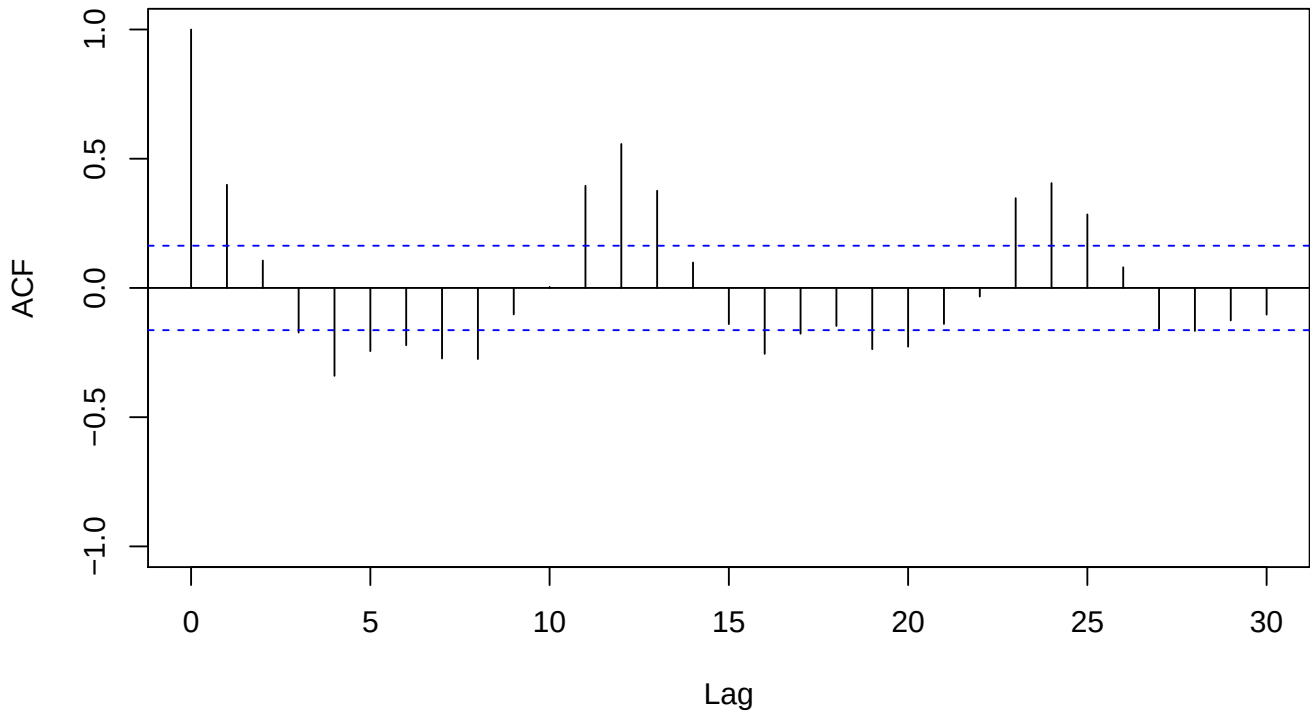

```
Yfit<-as.vector(matrix(G1,nrow=1) %*% t(a.pt))
plot(Yfit,type="l",ylim=range(-50,600),ylab="Passengers",xlab="Months",col="red",lwd=1)
lines(y);points(y,pch=19,cex=0.6)
legend(0,600,c("Data","One-step Forecasts"),lty=c(1,1),col=c("black","red"),lwd=1,pch=c(19,NA))
```



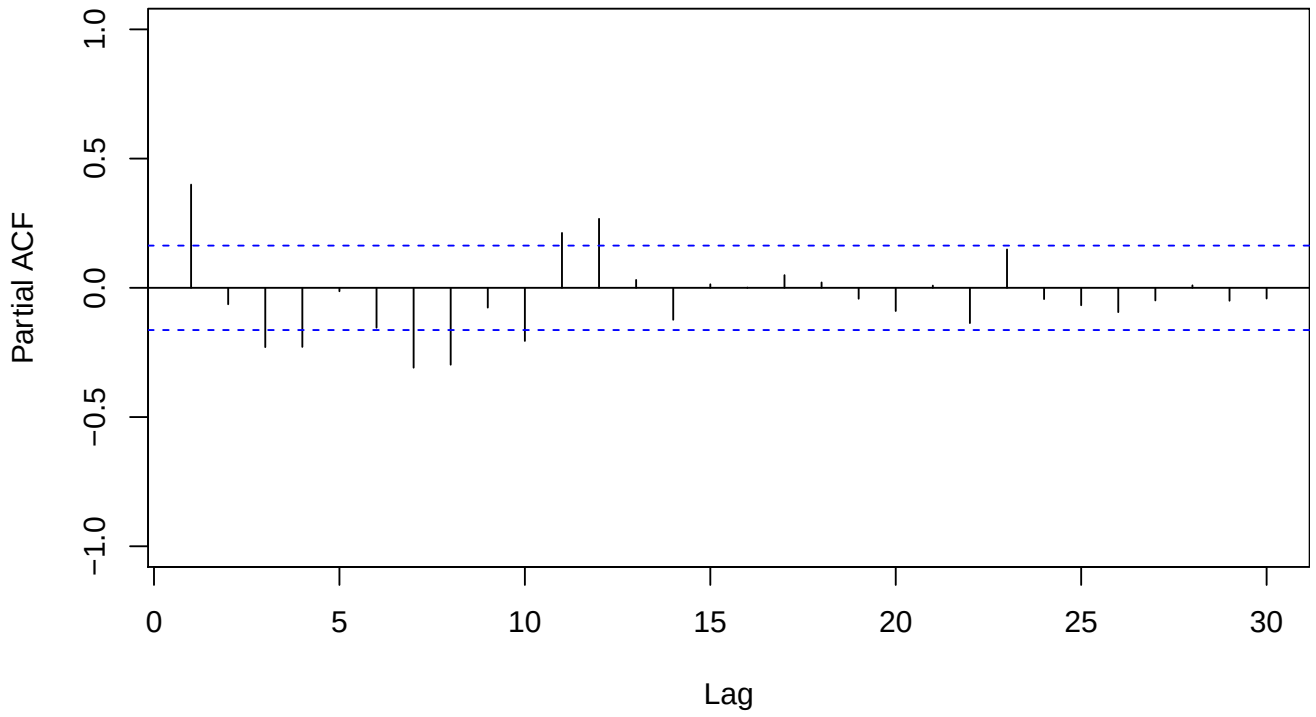

```
Yresids<-Y.t-Yfit[-n]
plot(Yresids,type="l",ylim=range(-60,60),ylab="Residuals",xlab="Months",col="red",lwd=1)
lines(Yresids);points(Yresids,pch=19,cex=0.6)
```




```
acf(Yresids,lag.max=30,ylim=range(-1,1),main='ACF of Residuals')
```




```
pacf(Yresids,lag.max=30,ylim=range(-1,1),main='PACF of Residuals')
```



For the Air Passenger data the estimates are

$$\sigma_1^2 = 172.384686 \qquad \sigma_2^2 = 0.0000229 \qquad \sigma_3^2 = 10.9091588 \qquad \sigma_W^2 = 0.0032373$$

The initial seasonal component values are

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