Dynamic linear models (aka state-space models)[1]

Advanced Econometris: Time Series
Hedibert Freitas Lopes
INSPER

[1] Part of this lecture is based on Gamerman and Lopes (2006) *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. Chapman & Hall/CRC.

Dynamic linear models[2]

Dynamic linear models are defined by a pair of equations, called the observation equation and the evolution or system equation, respectively given by

$$\begin{aligned} \text{Observation equation}: \quad y_t &= F_t'\theta_t + v_t, \qquad v_t \sim N(0, V_t) \\ \text{Evolution equation}: \quad \theta_t &= G_t\theta_{t-1} + w_t, \quad w_t \sim N(0, W_t) \end{aligned}$$

with $\theta_0 \sim N(m_0, C_0)$ and $\{y_t\}$ is a sequence of observations through time, conditionally independent given θ_t and V_t.

- F_t: vector of explanatory variables,
- θ_t: regression coefficients or state variables at time t,
- G_t: evolution matrix,
- The errors v_t and w_t are mutually independent.

[2] Harrison and Stevens (1976), West and Harrison (1997)

Dynamic regression

Dynamic regression models are defined by $G_t = I_d, \forall t$.

$$\begin{aligned} \text{Observation equation}: \quad y_t &= F_t'\theta_t + v_t, \qquad v_t \sim N(0, V_t) \\ \text{Evolution equation}: \quad \theta_t &= \theta_{t-1} + w_t, \quad w_t \sim N(0, W_t) \end{aligned}$$

These are also known, for instance in the macro econometrics literature, as time-varying parameter (TVP) models.

If, in addition, $W_t = 0$, the standard (heteroskedastic) normal linear regression model is obtained:

$$y_t = F_t'\theta_t + v_t, \qquad v_t \sim N(0, V_t).$$

First order DLM

The simplest time series model is the first order model, also known as local level model. It is given by equations

$$\begin{aligned} y_t &= \theta_t + v_t, \quad v_t \sim N(0, V_t) \\ \theta_t &= \theta_{t-1} + w_t, \quad w_t \sim N(0, W_t) \end{aligned}$$

and θ_t is scalar.

The model can be thought of as a first order Taylor series approximation of a smooth function representing the time trend of the series.

This model is useful for stock control, production planning and financial data analysis. Observational and system variances may evolve in time, offering great scope for modeling the variability of the system.

Second order DLM

The linear growth model is slightly more elaborate by incorporation of an extra time-varying parameter θ_2 representing the growth of the level of the series:

$$\begin{aligned} y_t &= \theta_{1,t} + v_t \\ \theta_{1,t} &= \theta_{1,t-1} + \theta_{2,t-1} + w_{1,t} \\ \theta_{2,t} &= \theta_{2,t-1} + w_{2,t}, \end{aligned}$$

where $v_t \sim N(0, V_t)$ and $w_t = (w_{1,t}, w_{2,t})' \sim N(0, W_t)$.

This model can be written in the general form with

$$F_t' = (1, 0) \quad \text{and} \quad G_t = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

The choice of F_t and G_t depends on the desired model and the nature of the series one wishes to describe.

Complete specification of the model requires full description of the variances V_t and W_t.

In general they are assumed to be constant in time with V_t typically larger than the entries of W_t in applications.

Sequential inference

One of the main aspects of a dynamic model is that at any time t, inference can be based on the updated distribution of $\theta_t|y^t$.

Sequential inference then carries this through time.

There are three basic operations involved here: evolution, prediction and updating. These operations are presented here in this order.

Evolution from $p(\theta_{t-1}|y^{t-1})$ to $p(\theta_t|y^{t-1})$

Consider that at time $t-1$, the updated distribution is

$$\theta_{t-1}|y^{t-1} \sim N(m_{t-1}, C_{t-1}).$$

The system equation can be written as $\theta_t|\theta_{t-1} \sim N(G_t\theta_{t-1},\ W_t)$.

These specifications can be combined and lead to:

$$\theta_t|y^{t-1} \sim N(a_t, R_t)$$

with $a_t = G_t m_{t-1}$ and $R_t = G_t C_{t-1} G_t' + W_t$;

One-step-ahead prediction $p(y_t|y^{t-1})$

One-step-ahead prediction can be made by noting that

$$p(y_t, \theta_t|y^{t-1}) = f(y_t|\theta_t)p(\theta_t|y^{t-1}).$$

Again, the joint distribution of $y_t, \theta_t|y^{t-1}$ can be reconstructed and lead to the marginal distribution

$$y_t|y^{t-1} \sim N(f_t, Q_t)$$

with $f_t = F_t' a_t$ and $Q_t = F_t' R_t F_t + V_t$ is obtained.

Updated $p(\theta_t|y^t)$

Finally, updating is achieved by the standard Bayes' theorem operation of including the observed y_t into the set of available information. The updated posterior distribution is obtained by

$$p(\theta_t|y^t) = p(\theta_t|y_t, y^{t-1}) \propto f(y_t|\theta_t)\, p(\theta_t|y^{t-1}) \ .$$

The resulting posterior distribution is

$$\theta_t|y^t \sim N(m_t, C_t)$$

with $m_t = a_t + A_t e_t$ and $C_t = R_t - A_t A_t' Q_t$, where $A_t = R_t F_t / Q_t$ and $e_t = y_t - f_t$.

This result follows the identity $C_t^{-1} = R_t^{-1} + F_t' F_t / V_t$.

It is sometimes referred to as the **Kalman filter**.

Smoothing

After sequentially obtaining the updated distributions of $\theta_t|y^t$ for $t = 1, \ldots, n$ (Kalman filter), time orientation is reversed from the distribution of $\theta_n|y^n$ so as to successively obtain the distributions of $\theta_t|y^n$ for $t = n-1, \ldots, 1$.

It can be shown that

$$\theta_t|y^n \sim N(m_t^n, C_t^n)$$

where

$$\begin{aligned} m_t^n &= m_t + C_t G_{t+1}' R_{t+1}^{-1}(m_{t+1}^n - a_{t+1}) \\ C_t^n &= C_t - C_t G_{t+1}' R_{t+1}^{-1}(R_{t+1} - C_{t+1}^n) R_{t+1}^{-1} G_{t+1} C_t. \end{aligned}$$

Stationary AR(2) model

The stationary AR(2) model ca be written as

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \epsilon_t \qquad \epsilon_t \sim NID(0, \tau^2).$$

Define the state vector $\theta_t = (y_t, y_{t-1})'$ so the transition equation is

$$\begin{pmatrix} y_t \\ y_{t-1} \end{pmatrix} = \begin{pmatrix} \phi_1 & \phi_2 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} y_{t-1} \\ y_{t-2} \end{pmatrix} + \begin{pmatrix} \epsilon_t \\ 0 \end{pmatrix}$$

and measurement equation

$$y_t = (1, 0)\theta_t + v_t.$$

It is easy to see that

- $F_t' = (1, 0)$ and $G_t = \begin{pmatrix} \phi_1 & \phi_2 \\ 1 & 0 \end{pmatrix}$
- $v_t = 0$ and $V_t = 0$
- $w_t = (\epsilon_t, 0)'$ and $W_t = \begin{pmatrix} \tau^2 & 0 \\ 0 & 0 \end{pmatrix}$

DLMs in R

Kalman filte/smoother for ML estimation using DLMs can easily be implemented in R or Matlab.

We will discuss the use of the dlm package for R.

Model Parameter	List Name
F	FF
V	V
G	GG
W	W
C0	C0
m0	m0

R code implementing several time series models using the dlm package will be available on the course website.

Specifying a DLM with the `dlm` package[3]

Function	Model
`dlm`	generic DLM
`dlmModARMA`	ARMA process
`dlmModPoly`	*n*th order polynomial DLM
`dlmModReg`	Linear regression
`dlmModSeas`	Periodic, seasonal factors
`dlmModTrig`	Periodic, trigonometric form

Function	Task
dlmFilter	Kalman filtering
dlmSmooth	Kalman smoothing
dlmForecast	Forecasting
dlmLL	Likelihood
dlmMLE	ML estimation

[3] This and the next several slides are taken from Sebastian Fossati's notes.

Simulating mean-zero AR(1) data

```
> # simulate AR(1) process
> set.seed(4321)
> yt = arima.sim(n=250,list(ar=0.75,ma=0),sd=0.5)
>
> model = Arima(yt,order=c(1,0,0),include.mean=FALSE)
>
> model
Series: yt
ARIMA(1,0,0) with zero mean

Coefficients:
         ar1
      0.7101
s.e.  0.0441

sigma^2 estimated as 0.2312:  log likelihood=-172.04
AIC=348.09   AICc=348.14   BIC=355.13
```

Setting up the DLM

The mean-zero AR(1) process

$$y_t = \phi y_{t-1} + \epsilon_t \qquad \epsilon_t \sim NID(0, \sigma^2)$$

can be written in state-space form as

$$\begin{aligned} y_t &= \theta_t \\ \theta_t &= \phi\theta_{t-1} + w_t \end{aligned}$$

with

$$F = 1, V = 0, G = \phi \text{ and } W = \sigma^2$$

```
# set up DLM
dlm0 = function(parm){
return(dlm(FF=1,V=0,GG=parm[1],W=parm[2]^2,
           m0=0,C0=solve(1-parm[1]^2)*parm[2]^2))
}
```

MLE and standard errors

The function `dlmMLE` may be used to compute the MLEs of the unknown parameters in the dynamic linear model.

```
> #  estimate DLM
> fit = dlmMLE(y=yt,parm=c(0.5,1.0),build=dlm0,hessian=T)
>
>
> # get estimates
> coef = fit$par
> var = solve(fit$hessian)
>
> # print results
> coef; sqrt(diag(var))
[1] 0.7100796 0.4808688
[1] 0.04409398 0.02150515
>
> # get estimated variance
> coef[2]^2
[1] 0.2312348
```

G, W and C_0

```
# print DLM model
> dlm0(fit$par)
$GG
           [,1]
[1,] 0.7100796

$W
           [,1]
[1,] 0.2312348

$m0
[1] 0

$C0
           [,1]
[1,] 0.4663996
```

Forecasting with dlmForecast

The function dlmForecast may be used to compute h-step ahead predictions from the state space model.

```
> # Forecast next 5 observations using dlmForecast
>
> mod  = dlm0(fit$par)
> modf = dlmFilter(yt,mod)
> fore = dlmForecast(modf,nAhead=5,method="plain")
>
> fore$f
Time Series:
Start = 251
End = 255
Frequency = 1
         Series 1
[1,] -0.07911367
[2,] -0.05617700
[3,] -0.03989014
[4,] -0.02832518
[5,] -0.02011313
```

AR(1) with intercept

The stationary AR(1) model with intercept can be written as

$$y_t - \mu = \phi(y_{t-1} - \mu) + \epsilon_t \qquad \epsilon_t \sim NID(0, \sigma^2)$$

Define the state vector $\theta_t = (\mu, y_t - \mu)'$, so the DLM is

$$\begin{aligned} y_t &= (1,1)\theta_t \\ \theta_t &= \begin{pmatrix} \mu \\ y_t - \mu \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & \phi \end{pmatrix} \begin{pmatrix} \mu \\ y_{t-1} - \mu \end{pmatrix} + \begin{pmatrix} 0 \\ \epsilon_t \end{pmatrix} \end{aligned}$$

with

$$F' = (1,1), V = 0, G = \begin{pmatrix} 1 & 0 \\ 0 & \phi \end{pmatrix}, W = \begin{pmatrix} 0 & 0 \\ 0 & \sigma^2 \end{pmatrix}$$

Adding determinist terms with `dlmModPoly`

The function `dlmModPoly` may be used to incorporate deterministic terms (mean, time trend, etc.).

For example, `dlmModPoly(1,dV,dW)` produces

$$\begin{aligned} y_t &= \theta_t + v_t \\ \theta_t &= \theta_{t-1} + w_t \end{aligned}$$

with $F = 1$, $V =$`dV`, $G = 1$ e $W =$`dW`.

Then, to allow for a non-zero mean set `dW`=0 such that

$$\begin{aligned} y_t &= \theta_t + v_t \\ \theta_t &= \theta_{t-1} (= \beta) \end{aligned}$$

Combining dlmModPoly and dlmModARMA

```
n.obs = 250
yt = 2 + arima.sim(n=n.obs,list(ar=.75,ma=0),sd=.5)

# set parameter restrictions (only variances here)
parm_rest = function(parm){
  return( c(parm[1],exp(parm[2])) )
}

# set up DLM
dlm1 = function(parm){
  parm = parm_rest(parm)
  dlm = dlmModPoly(1,dV=1e-7,dW=c(0)) +
  dlmModARMA(ar=parm[1], ma=NULL, sigma2=parm[2])
  # set initial state distribution
  dlm$C0[2,2] <- solve(1-parm[1]^2)*parm[2]
  return(dlm)
}
```

Get MLEs and Standard Errors

```
> # estimate parameters
> fit1 = dlmMLE(y=yt,parm=c(0,0),build=dlm1,hessian=T)
>
> # get parameter estimates of AR(1) part
> coef = parm_rest(fit1$par)
>
> # get standard errors using delta method
> jac = jacobian(func=parm_rest,x=fit1$par)
> var = jac%*%solve(fit1$hessian)%*%t(jac)
>
> # print results
> coef; sqrt(diag(var))
[1] 0.8217966 0.2516931
[1] 0.03704438 0.02255759
```

Get MLEs and Standard Errors

The MLE of μ is the first element of m_T (i.e., the full sample filtered value).

```
> # get parameter estimates (intercept)
> # these are the last filtered values
> mod1 = dlm1(fit1$par)
> mod1filt = dlmFilter(yt,mod1)
>
> # get parameters
> coef = mod1filt$m[n.obs+1]
> covar = dlmSvd2var(mod1filt$U.C[[n.obs+1]],mod1filt$D.C[n.obs+1,])
> coef.se = sqrt(covar[1,1])
> coef; coef.se
[1] 2.061879
[1] 0.1748569
```

Displaying DLM

```
> dlm1(fit1$par)
$GG
     [,1]      [,2]
[1,]    1 0.0000000
[2,]    0 0.8217966

$W
     [,1]      [,2]
[1,]    0 0.0000000
[2,]    0 0.2516931
```

TVP-CAPM

Consider the time-varying parameter CAPM regression

$$r_t = \alpha_t + \beta_t r_{M,t} + \epsilon_t \qquad \epsilon_t \sim N(0, \sigma^2)$$

in the normal dynamic linear model (NDLM) form:

$$\begin{aligned} r_t &= F_t'\theta_t + v_t \qquad w_t \sim N(0, V) \\ \theta_t &= \theta_{t-1} + w_t \qquad w_t \sim N(0, W) \end{aligned}$$

with $F_t' = (1, r_{M,t})$, $\theta_t = (\alpha_t, \beta_t)'$, $V = \sigma^2$, and $W = \text{diag}(\sigma_\alpha^2, \sigma_\beta^2)$.

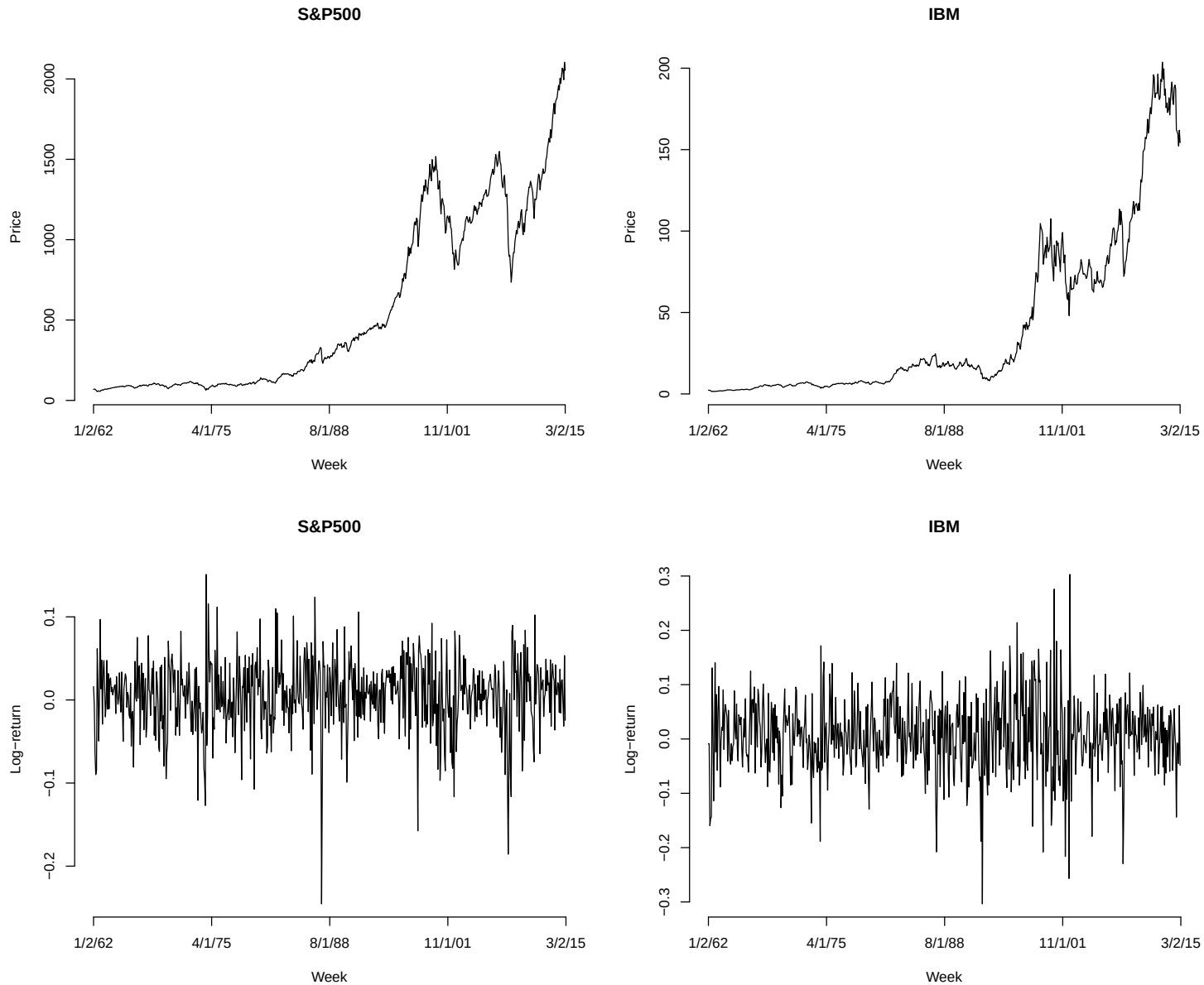
Additionally,

$$\theta_0 \sim N(0, \kappa I_2),$$

for large κ.

S&P500 and IBM

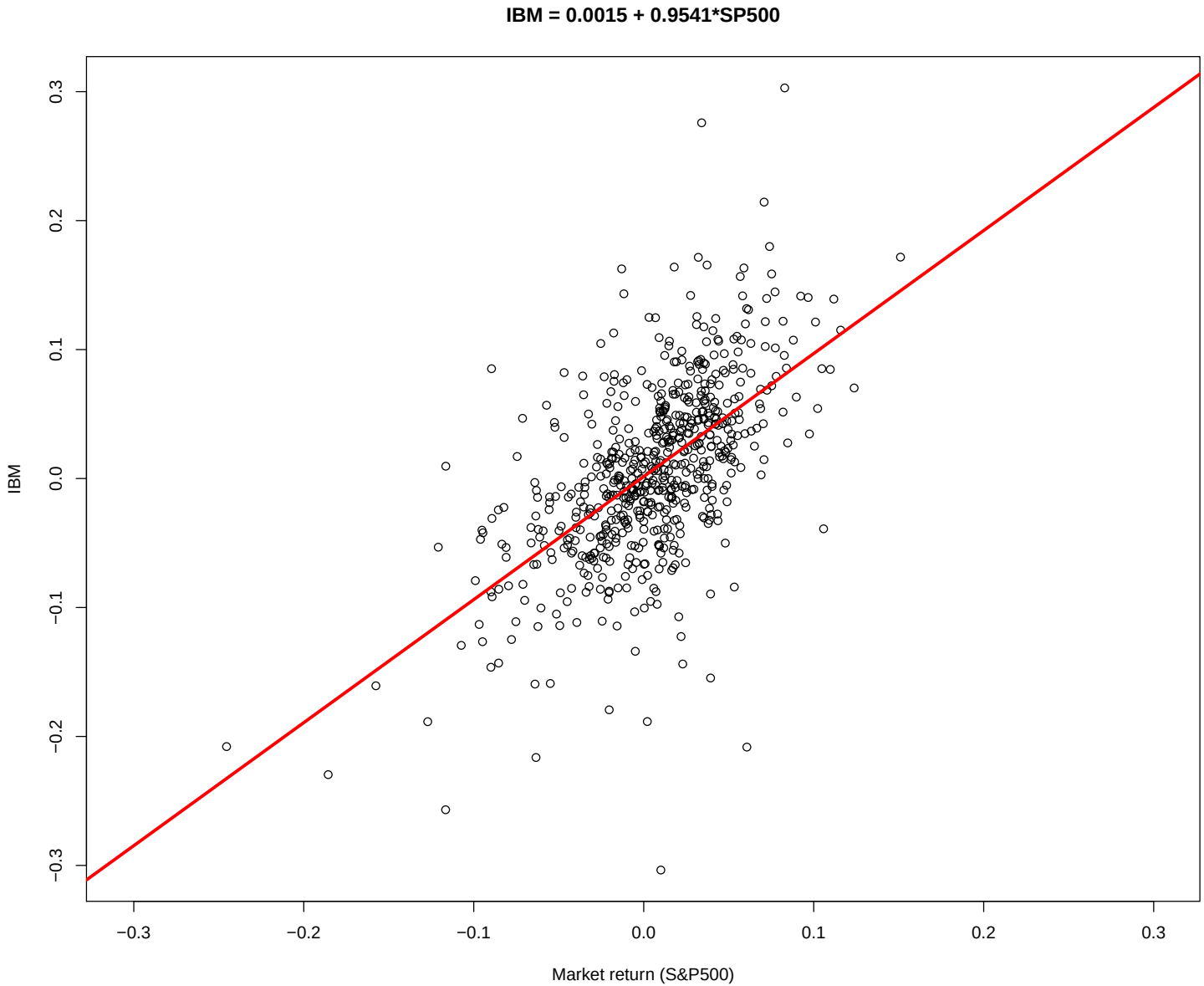
Monthly returns - Jan/62 to Feb/15 - $n = 638$ obs.



Time-invariant CAPM

```
> capm = lm(r~rM)
> summary(capm)

Call:
lm(formula = r ~ rM)

Residuals:
      Min        1Q    Median        3Q       Max
-0.314704 -0.029848 -0.001228  0.030513  0.241841

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.001528   0.002214    0.69     0.49
rM          0.954062   0.050769   18.79   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.05551 on 636 degrees of freedom
Multiple R-squared:  0.357, Adjusted R-squared:  0.356
F-statistic: 353.1 on 1 and 636 DF,  p-value: < 2.2e-16
```

Time-invariant CAPM



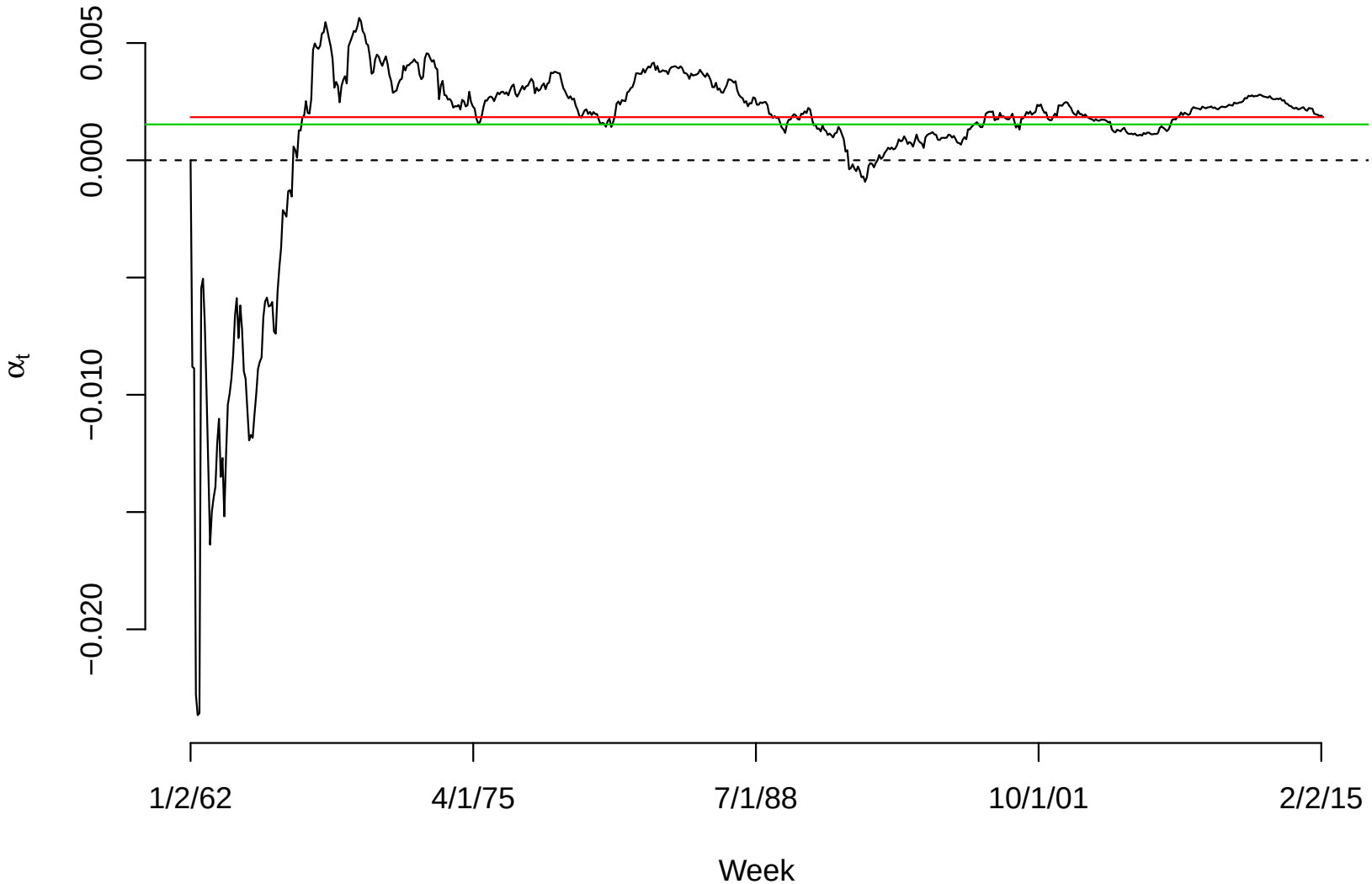
Set up DLM

```
> # set up DLM
> dlm2 = function(parm,x.mat){
+   parm = exp(parm)
+   return(dlmModReg(X=x.mat,dV=parm[1],dW=c(parm[2],parm[3])))
+ }
>
> # estimate parameters
> fit2 = dlmMLE(y=r,parm=c(0.1,0.1,0.1),x.mat=rM,build=dlm2,hessian=T)
>
> # get estimates
> se =  sqrt(exp(fit2$par))
> se
[1] 5.403817e-02 3.528289e-06 4.628380e-02
```

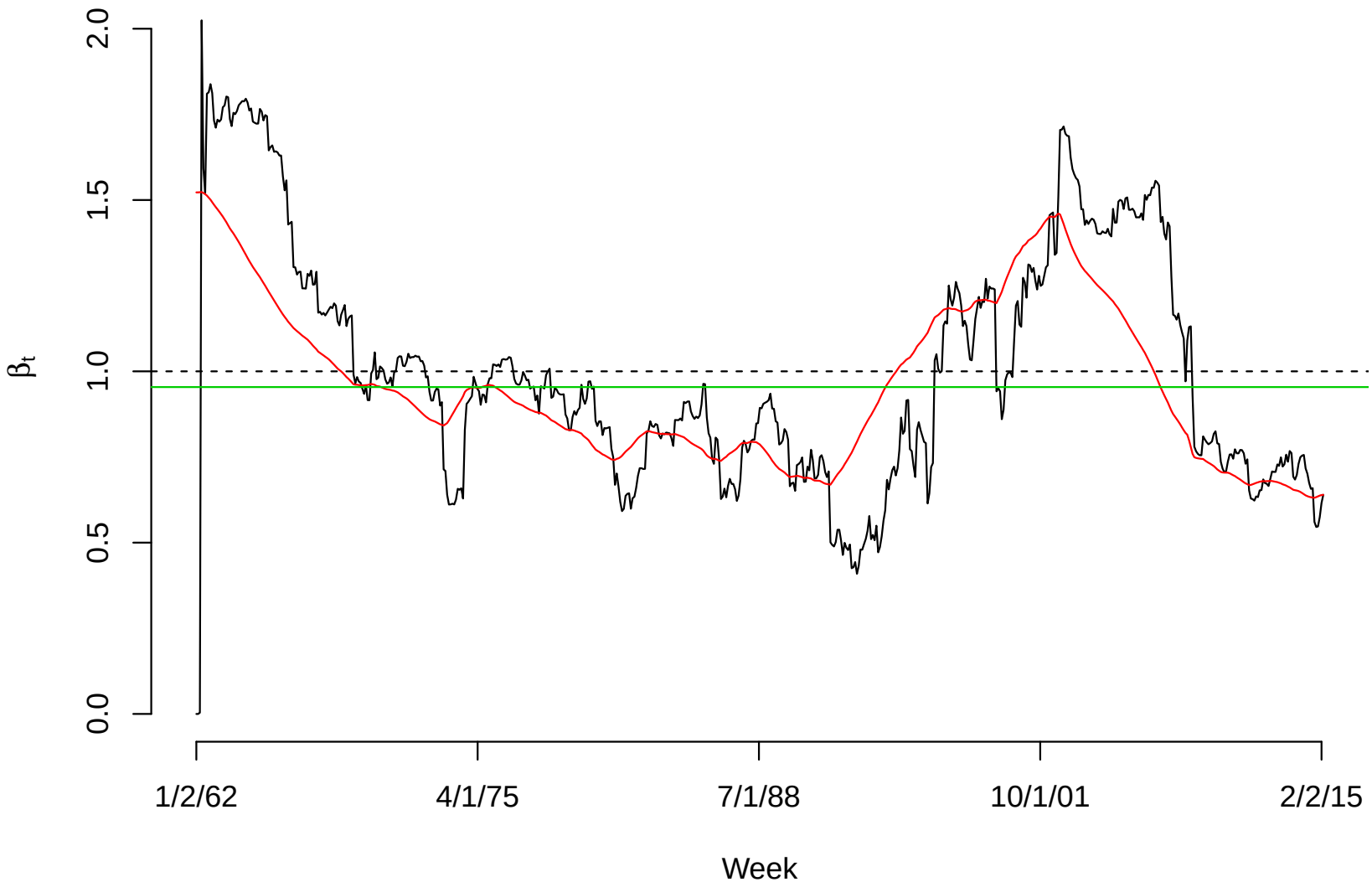
Smoothed states

```
# get parameter estimates over time
# these are the smoothed state values
mod2  = dlm2(fit2$par,rM)
mod2f = dlmFilter(r,mod2)
mod2s = dlmSmooth(mod2f)

# plot filtered and smoothed states
ind = seq(1,length(r),length=5)
date = data[ind,1]
pdf(file="capm-filtering-and-smoothing-intercept.pdf",width=8,height=6)
plot(mod2f$m[,1],axes=FALSE,xlab="Week",ylab=expression(alpha[t]),type="l",main="")
axis(2);axis(1,at=ind,lab=date)
lines(mod2s$s[,1],col=2)
abline(h=capm$coef[1],col=3)
dev.off()

pdf(file="capm-filtering-and-smoothing-slope.pdf",width=8,height=6)
plot(mod2f$m[,2],axes=FALSE,xlab="Week",ylab=expression(beta[t]),type="l",main="")
axis(2);axis(1,at=ind,lab=date)
lines(mod2s$s[,2],col=2)
abline(h=capm$coef[2],col=3)
dev.off()
```

Filtered and smoothed intercept



Filtered and smoothed slope



Free form seasonal factor - `dlmModSeas`

For a seasonal model with period s, one can consider an $(s-1)$-dimensional state space, with

$$F' = (1, 0, \cdots, 0)$$

and

$$G = \left(\begin{array}{c|ccc|c} -1 & -1 & \cdots & -1 & -1 \\ \hline 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{array}\right)$$

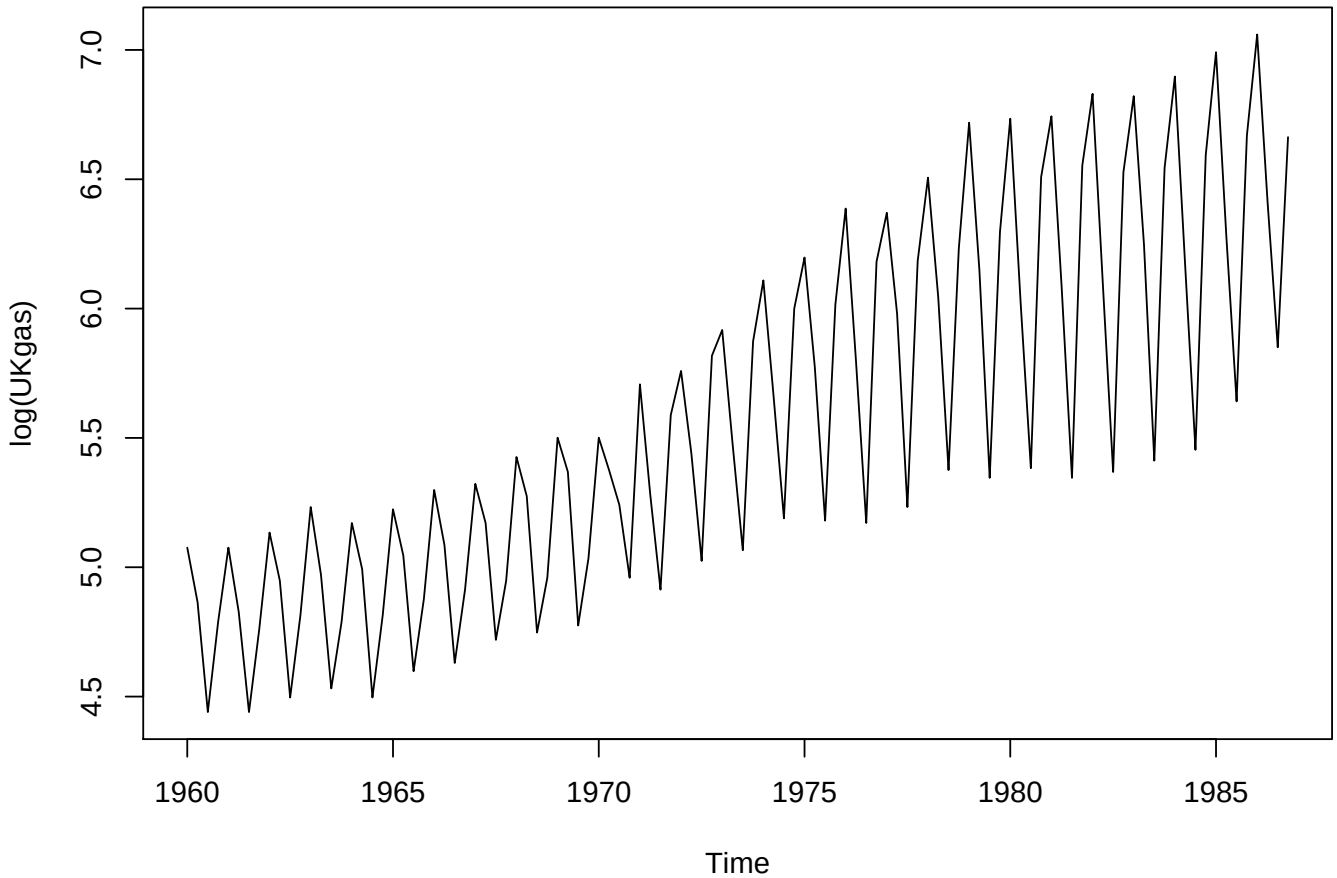
Seasonal dynamic variation: $\text{diag}(W, 0, \ldots, 0)$.

Example: Quarterly data with $W = 4.2$ and $V = 3.5$

```
mod = dlmModSeas(frequency=4,dV=3.5,dW=c(4.2,0,0))
```

```
> mod
$FF
     [,1] [,2] [,3]
[1,]    1    0    0
$V
     [,1]
[1,]  3.5
$GG
     [,1] [,2] [,3]
[1,]   -1   -1   -1
[2,]    1    0    0
[3,]    0    1    0
$W
     [,1] [,2] [,3]
[1,]  4.2    0    0
[2,]  0.0    0    0
[3,]  0.0    0    0
```

Quarterly UK gas consumption from 1960 to 1986



Suppose that we want to describe the series, on a logarithmic scale, by a DLM containing a local linear trend, T_t, and a quarterly seasonal component, S_t.

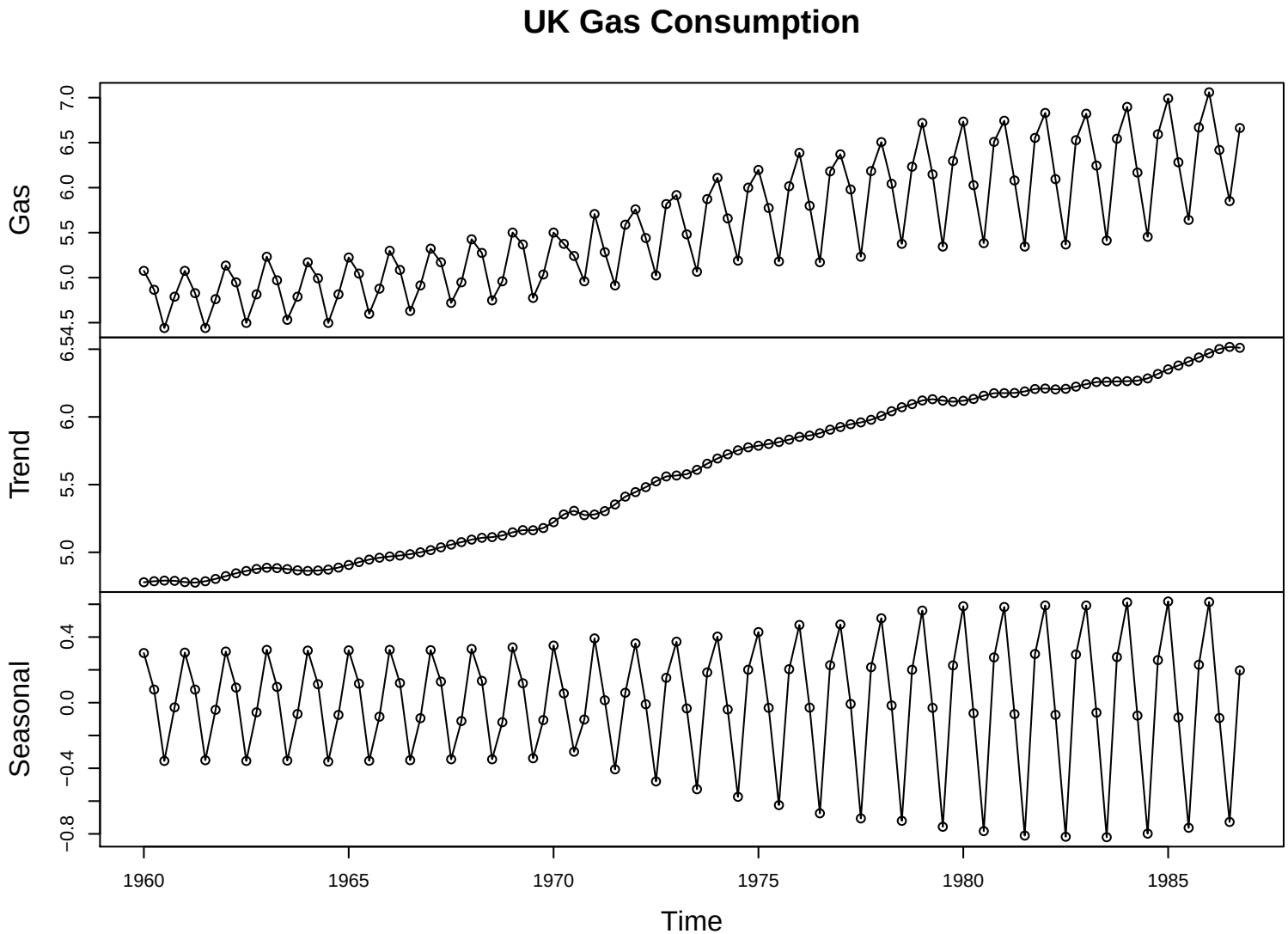
Free-form seasonality

```
y = log(UKgas)

dlm3 = dlmModPoly() + dlmModSeas(4)

buildFun = function(x) {
  diag(W(dlm3))[2:3] = exp(x[1:2])
  V(dlm3) = exp(x[3])
  return(dlm3)
}

fit3 = dlmMLE(y,parm=c(0.1,0.1,0.1),build=buildFun)

dlm3 = buildFun(fit3$par)

ySmooth = dlmSmooth(y, mod = dlm3)

x = cbind(y, dropFirst(ySmooth$s[, c(1, 3)]))

colnames(x) = c("Gas", "Trend", "Seasonal")

plot(x, type = "o", main = "UK Gas Consumption")
```

Fourier-form seasonality

```
dlm4 = dlmModPoly() + dlmModTrig(4)

buildFun = function(x) {
  diag(W(dlm4))[2:3] = exp(x[1:2])
  V(dlm4) = exp(x[3])
  return(dlm4)
}

fit4 = dlmMLE(y,parm=c(0.1,0.1,0.1),build=buildFun)

dlm4 = buildFun(fit4$par)

ySmooth = dlmSmooth(y, mod = dlm4)

x = cbind(y, dropFirst(ySmooth$s[, c(1, 3)]))

colnames(x) = c("Gas", "Trend", "Seasonal")

plot(x, type = "o", main = "UK Gas Consumption")
```

Free-form seasonality



Fourier-form seasonality