4th Edition

- |
- Home
- |
- Text R Code
- |
- Graphics Fix
- |
- R Fix
- |
- R Issues
- |
- Kalman Filter
- |

astsa

The R package for the text is called astsa. The package can be obtained from CRAN and its mirrors in the usual way. See the package notes page for further information.

4th edition

The 4th edition will be out early in 2017 and available for the fall 2017 semester. Stay tuned. Production progress can be monitored at Springer.com

EZ time series

A gentle introduction to time series analysis is available (for free): tsaEZ. The preface has more details. A short course, ARIMA Modeling With R, which is based on this text, will be available at Data Camp in 2017.

getting R

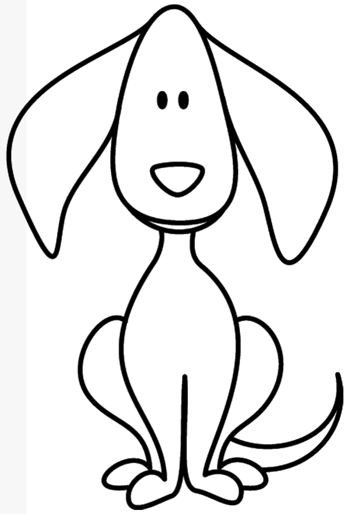
To download R, go to CRAN and download the appropriate installation.

The R issues for time series analysis page is still alive. There's a tutorial on Time Series Graphics and the R time series tutorial has been updated. Go here to get your Quick Fix.

text on nonlinear time series

A text on Nonlinear Time Series Analysis was published by Chapman-Hall in January 2014. The text can be used as a reference for a second course in time series analysis.

Dog Show



- Package Notes for astsa
- Code Used in the 4th Edition
- R Tutorial
- Time Series Graphics
- Errata
- Website for 2nd Edition
- Website for 3rd Edition
-
- EZ Time Series Short Course @ DataCamp Learn Data Science By Doing

4th Edition

- |
- Home
- |

Code Used in the Text Examples

Below is the code used for each numerical example in the text. This stuff won't work unless you have loaded astsa first. If this is your first time here, you might want to read the astsa package notes page for further information.

The code for plots in the text were often shortened for the display (to save space). The actual code for all the graphs in the text can be found at GitHub.

Click [+] to expand or collapse section. Click [-] to collapse entire page. Or [+ Expand Entire Page +]

[+] Chapter 1

Example 1.1

```
plot(jj, type="o", ylab="Quarterly Earnings per Share")
```

Example 1.2

```
plot(globtemp, type="o", ylab="Global Temperature Deviations")
```

Example 1.3

```
plot(speech)
```

Example 1.4

```
## djia is in astsa version 1.5+ - this is where it came from
# library(TTR)                              # install it first
# djia  = getYahooData("^DJI", start=20060420, end=20160420, freq="daily")
library(xts)                                # if don't use TTR
djiar = diff(log(djia$Close))[-1]          # approximate returns
plot(djiar, main="DJIA Returns", type="n")
lines(djiar)
```

Example 1.5

```
par(mfrow = c(2,1))  # set up the graphics
plot(soi, ylab="", xlab="", main="Southern Oscillation Index")
plot(rec, ylab="", xlab="", main="Recruitment")
```

Example 1.6

```
par(mfrow=c(2,1), mar=c(3,2,1,0)+.5, mgp=c(1.6,.6,0))
ts.plot(fmri1[,2:5], col=1:4, ylab="BOLD", xlab="", main="Cortex")
ts.plot(fmri1[,6:9], col=1:4, ylab="BOLD", xlab="", main="Thalamus & Cerebellum")
mtext("Time (1 pt = 2 sec)", side=1, line=2)
```

Example 1.7

```
par(mfrow=c(2,1))
plot(EQ5, main="Earthquake")
```

```
plot(EXP6, main="Explosion")
```

Example 1.9

```
w = rnorm(500,0,1)  # 500 N(0,1) variates
v = filter(w, sides=2, rep(1/3,3))  # moving average
par(mfrow=c(2,1))
plot.ts(w, main="white noise")
plot.ts(v, ylim=c(-3,3), main="moving average")

# now try this (not in text):
dev.new()  # open a new graphic device
ts.plot(w, v, lty=2:1, col=1:2, lwd=1:2)
```

Example 1.10

```
w = rnorm(550,0,1)  # 50 extra to avoid startup problems
x = filter(w, filter=c(1,-.9), method="recursive")[-(1:50)]
plot.ts(x, main="autoregression")
```

Example 1.11

```
set.seed(154) # so you can reproduce the results
w = rnorm(200); x = cumsum(w) # two commands in one line
wd = w +.2;    xd = cumsum(wd)
plot.ts(xd, ylim=c(-5,55), main="random walk", ylab='')
lines(x, col=4)
abline(h=0, col=4, lty=2)
abline(a=0, b=.2, lty=2)
```

Example 1.12

```
cs = 2*cos(2*pi*(1:500)/50 + .6*pi)
```

```
w = rnorm(500,0,1)
par(mfrow=c(3,1), mar=c(3,2,2,1), cex.main=1.5)   # help(par) for info
plot.ts(cs, main = expression(x[t]==2*cos(2*pi*t/50+.6*pi)))
plot.ts(cs + w, main = expression(x[t]==2*cos(2*pi*t/50+.6*pi)+N(0,1)))
plot.ts(cs + 5*w, main = expression(x[t]==2*cos(2*pi*t/50+.6*pi)+N(0,25)))
```

Example 1.24

```
set.seed(2)
x = rnorm(100)
y = lag(x, -5) + rnorm(100)
ccf(y, x, ylab='CCovF', type='covariance')
abline(v=0, lty=2)
text(11, .9, 'x leads')
text(-9, .9, 'y leads')
```

Example 1.25

```
(r = round(acf(soi, 6, plot=FALSE)$acf[-1], 3)) # first 6 sample acf values
par(mfrow=c(1,2))
plot(lag(soi,-1), soi); legend('topleft', legend=r[1])
plot(lag(soi,-6), soi); legend('topleft', legend=r[6])
```

Example 1.26

```
set.seed(101010)
x1 = 2*rbinom(11, 1, .5) - 1 # simulated sequence of coin tosses
x2 = 2*rbinom(101, 1, .5) - 1
y1 = 5 + filter(x1, sides=1, filter=c(1,-.7))[-1]
y2 = 5 + filter(x2, sides=1, filter=c(1,-.7))[-1]
plot.ts(y1, type='s')  # plot series
plot.ts(y2, type='s')
```

```
c(mean(y1), mean(y2))  # the sample means

acf(y1, lag.max=4, plot=FALSE)

acf(y2, lag.max=4, plot=FALSE)
```

Example 1.27

```
acf(speech, 250)
```

Example 1.28

```
par(mfrow=c(3,1))

acf(soi, 48, main="Southern Oscillation Index")

acf(rec, 48, main="Recruitment")

ccf(soi, rec, 48, main="SOI vs Recruitment", ylab="CCF")
```

Example 1.29

```
set.seed(1492)

num=120; t=1:num

X = ts(2*cos(2*pi*t/12) + rnorm(num), freq=12)

Y = ts(2*cos(2*pi*(t+5)/12) + rnorm(num), freq=12)

Yw = resid( lm(Y~ cos(2*pi*t/12) + sin(2*pi*t/12), na.action=NULL) )

par(mfrow=c(3,2), mgp=c(1.6,.6,0), mar=c(3,3,1,1) )

plot(X)

plot(Y)

acf(X,48, ylab='ACF(X)')

acf(Y,48, ylab='ACF(Y)')

ccf(X,Y,24, ylab='CCF(X,Y)')

ccf(X,Yw,24, ylab='CCF(X,Yw)', ylim=c(-.6,.6))
```

Example 1.30

```
persp(1:64, 1:36, soiltemp, phi=30, theta=30, scale=FALSE, expand=4,
```

```
ticktype="detailed",
          xlab="rows", ylab="cols", zlab="temperature")
dev.new()
plot.ts(rowMeans(soiltemp), xlab="row", ylab="Average Temperature")
```

Example 1.31

```
fs = abs(fft(soiltemp-mean(soiltemp)))^2/(64*36) # see Ch 4 for info on FFT
cs = Re(fft(fs, inverse=TRUE)/sqrt(64*36))  # ACovF
rs = cs/cs[1,1]                             # ACF

rs2 = cbind(rs[1:41,21:2], rs[1:41,1:21])   #  these lines are just to center
rs3 = rbind(rs2[41:2,], rs2)                #  the 0 lag

par(mar = c(1,2.5,0,0)+.1)
persp(-40:40, -20:20, rs3, phi=30, theta=30, expand=30, scale="FALSE",
ticktype="detailed",
       xlab="row lags", ylab="column lags", zlab="ACF")
```

[-]

[+] Chapter 2

Example 2.1

```
summary(fit <- lm(chicken~time(chicken))) # regress price on time
plot(chicken, ylab="cents per pound")
abline(fit)           # add the fitted regression line to the plot
```

Example 2.2

```
par(mfrow=c(3,1))
plot(cmort, main="Cardiovascular Mortality", xlab="", ylab="")
```

```
plot(tempr, main="Temperature", xlab="", ylab="")
plot(part, main="Particulates", xlab="", ylab="")

dev.new()
pairs(cbind(Mortality=cmort, Temperature=tempr, Particulates=part))

#  Regression
temp  = tempr-mean(tempr)  # center temperature
temp2 = temp^2             # square it
trend = time(cmort)        # time

fit = lm(cmort~ trend + temp + temp2 + part, na.action=NULL)

summary(fit)       # regression results
summary(aov(fit))  # ANOVA table   (compare to next line)
summary(aov(lm(cmort~cbind(trend, temp, temp2, part)))) # Table 2.1

num = length(cmort)                                     # sample size
AIC(fit)/num - log(2*pi)                                # AIC
BIC(fit)/num - log(2*pi)                                # BIC
# AIC(fit, k=log(num))/num - log(2*pi)                  # BIC (alt method)
(AICc = log(sum(resid(fit)^2)/num) + (num+5)/(num-5-2)) # AICc
```

Examples 2.3

```
fish = ts.intersect(rec, soiL6=lag(soi,-6), dframe=TRUE)
summary(fit <- lm(rec~soiL6, data=fish, na.action=NULL))
plot(fish$rec)  # plot the data and the fitted values (not shown in text)
lines(fitted(fit), col=2)
```

Examples 2.4 and 2.5

```
fit = lm(chicken~time(chicken), na.action=NULL) # regress chicken on time
par(mfrow=c(2,1))
plot(resid(fit), type="o", main="detrended")
plot(diff(chicken), type="o", main="first difference")


dev.new()
par(mfrow=c(3,1))     # plot ACFs
acf(chicken, 48, main="chicken")
acf(resid(fit), 48, main="detrended")
acf(diff(chicken), 48, main="first difference")
```

Example 2.6

```
par(mfrow=c(2,1))
plot(diff(globtemp), type="o")
 mean(diff(globtemp))     # drift estimate = .008
acf(diff(gtemp), 48)
```

Example 2.7

```
par(mfrow=c(2,1))
plot(varve, main="varve", ylab="")
plot(log(varve), main="log(varve)", ylab="" )
```

Example 2.8

```
lag1.plot(soi, 12)
dev.new()
lag2.plot(soi, rec, 8)
```

Example 2.9

```
dummy = ifelse(soi<0, 0, 1)

fish  = ts.intersect(rec, soiL6=lag(soi,-6), dL6=lag(dummy,-6), dframe=TRUE)

summary(fit <- lm(rec~ soiL6*dL6, data=fish, na.action=NULL))

attach(fish)

plot(soiL6, rec)

lines(lowess(soiL6, rec), col=4, lwd=2)

points(soiL6, fitted(fit), pch='+', col=2)

plot(resid(fit)) # not shown ...

acf(resid(fit))  # ... but obviously not noise
```

Example 2.10

```
set.seed(1000)  # so you can reproduce these results

x = 2*cos(2*pi*1:500/50 + .6*pi) + rnorm(500,0,5)

z1 = cos(2*pi*1:500/50)

z2 = sin(2*pi*1:500/50)

summary(fit <- lm(x~0+z1+z2))  # zero to exclude the intercept


par(mfrow=c(2,1))

plot.ts(x)

plot.ts(x, col=8, ylab=expression(hat(x)))

lines(fitted(fit), col=2)
```

Example 2.11

```
wgts = c(.5, rep(1,11), .5)/12

soif = filter(soi, sides=2, filter=wgts)

plot(soi)

lines(soif, lwd=2, col=4)

par(fig = c(.65, 1, .65, 1), new = TRUE) # the insert

nwgts = c(rep(0,20), wgts, rep(0,20))

plot(nwgts, type="l", ylim = c(-.02,.1), xaxt='n', yaxt='n', ann=FALSE)
```

Example 2.12

```
plot(soi)
lines(ksmooth(time(soi), soi, "normal", bandwidth=1), lwd=2, col=4)
par(fig = c(.65, 1, .65, 1), new = TRUE) # the insert
gauss = function(x) { 1/sqrt(2*pi) * exp(-(x^2)/2) }
x = seq(from = -3, to = 3, by = 0.001)
plot(x, gauss(x), type ="l", ylim=c(-.02,.45), xaxt='n', yaxt='n', ann=FALSE)
```

Example 2.13

```
plot(soi)
lines(lowess(soi, f=.05), lwd=2, col=4) # El Nino cycle
lines(lowess(soi), lty=2, lwd=2, col=2) # trend (with default span)
```

Example 2.14

```
plot(soi)
lines(smooth.spline(time(soi), soi, spar=.5), lwd=2, col=4)
lines(smooth.spline(time(soi), soi, spar= 1), lty=2, lwd=2, col=2)
```

Example 2.15

```
plot(tempr, cmort, xlab="Temperature", ylab="Mortality")
lines(lowess(tempr, cmort))
```

[-]

[+] Chapter 3

Example 3.2

```
par(mfrow=c(2,1))
```

```
# in the expressions below, ~ is a space and == is equal
plot(arima.sim(list(order=c(1,0,0), ar=.9), n=100), ylab="x", main=(expression(AR(1)
~~~phi==+.9)))
plot(arima.sim(list(order=c(1,0,0), ar=-.9), n=100), ylab="x", main=(expression(AR(1)
~~~phi==-.9)))
```

Example 3.5

```
par(mfrow=c(2,1))
plot(arima.sim(list(order=c(0,0,1), ma=.9), n=100), ylab="x", main=(expression(MA(1)
~~~theta==+.9)))
plot(arima.sim(list(order=c(0,0,1), ma=-.9), n=100), ylab="x", main=(expression(MA(1)
~~~theta==-.9)))
```

Example 3.7

```
set.seed(8675309)         # Jenny, I got your number
x = rnorm(150, mean=5)    # Jenerate iid N(5,1)s
arima(x, order=c(1,0,1))  # Jenstimation
```

Example 3.8

```
ARMAtoMA(ar = .9,  ma = .5,  10)   # first 10 psi-weights
ARMAtoMA(ar = -.5, ma = -.9, 10)   # first 10 pi-weights
```

Example 3.11

```
z = c(1,-1.5,.75)    # coefficients of the polynomial
(a = polyroot(z)[1]) # = 1+0.57735i,  print one root which is 1 + i 1/sqrt(3)
arg = Arg(a)/(2*pi)  # arg in cycles/pt
1/arg                # = 12,  the period

set.seed(8675309)    # Jenny, it's me again
```

```
ar2 = arima.sim(list(order=c(2,0,0), ar=c(1.5,-.75)), n = 144)
plot(ar2, axes=FALSE, xlab="Time")
axis(2); axis(1, at=seq(0,144,by=12)); box()  # work the plot machine
abline(v=seq(0,144,by=12), lty=2)

ACF = ARMAacf(ar=c(1.5,-.75), ma=0, 50)
plot(ACF, type="h", xlab="lag")
abline(h=0)
```

Example 3.12

```
ARMAtoMA(ar=.9, ma=.5, 50)       #  for a list
plot(ARMAtoMA(ar=.9, ma=.5, 50)) #  for a graph
```

Example 3.16

```
ar2.acf = ARMAacf(ar=c(1.5,-.75), ma=0, 24)[-1]
ar2.pacf = ARMAacf(ar=c(1.5,-.75), ma=0, 24, pacf=TRUE)
par(mfrow=c(1,2))
plot(ar2.acf, type="h", xlab="lag")
abline(h=0)
plot(ar2.pacf, type="h", xlab="lag")
abline(h=0)
```

Example 3.18

```
acf2(rec, 48)     # will produce values and a graphic
(regr = ar.ols(rec, order=2, demean=F, intercept=TRUE))  # regression
regr$asy.se.coef  # standard errors
```

Example 3.25

```
regr = ar.ols(rec, order=2, demean=FALSE, intercept=TRUE)
```

```
fore = predict(regr, n.ahead=24)
ts.plot(rec, fore$pred, col=1:2, xlim=c(1980,1990), ylab="Recruitment")
lines(fore$pred, type="p", col=2)
lines(fore$pred+fore$se, lty="dashed", col=4)
lines(fore$pred-fore$se, lty="dashed", col=4)
```

Example 3.26

```
set.seed(90210)
x = arima.sim(list(order = c(1,0,1), ar =.9, ma=.5), n = 100)
xr = rev(x) # xr is the reversed data
pxr = predict(arima(xr, order=c(1,0,1)), 10) # predict the reversed data
pxrp = rev(pxr$pred) # reorder the predictors (for plotting)
pxrse = rev(pxr$se) # reorder the SEs
nx = ts(c(pxrp, x), start=-9) # attach the backcasts to the data
plot(nx, ylab=expression(X[~t]), main='Backcasting')
U = nx[1:10] + pxrse; L = nx[1:10] - pxrse
xx = c(-9:0, 0:-9); yy = c(L, rev(U))
polygon(xx, yy, border = 8, col = gray(0.6, alpha = 0.2))
lines(-9:0, nx[1:10], col=2, type='o')
```

Example 3.28

```
rec.yw = ar.yw(rec, order=2)
rec.yw$x.mean  # = 62.26278 (mean estimate)
rec.yw$ar      # = 1.3315874, -.4445447  (parameter estimates)
sqrt(diag(rec.yw$asy.var.coef))  # = .04222637, .04222637  (standard errors)
rec.yw$var.pred  # = 94.79912 (error variance estimate)


rec.pr = predict(rec.yw, n.ahead=24)
U = rec.pr$pred + rec.pr$se
```

```
L = rec.pr$pred - rec.pr$se
minx = min(rec,L); maxx = max(rec,U)
ts.plot(rec, rec.pr$pred, xlim=c(1980,1990), ylim=c(minx,maxx))
lines(rec.pr$pred, col="red", type="o")
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
```

Example 3.29

```
set.seed(2)
ma1 = arima.sim(list(order = c(0,0,1), ma = 0.9), n = 50)
acf(ma1, plot=FALSE)[1]  # = .507 (lag 1 sample ACF)
```

Example 3.31

```
# Note: I'm not so sure this is the MLE...
#  ... but eventually 'sarima()' will be used
rec.mle = ar.mle(rec, order=2)
rec.mle$x.mean
rec.mle$ar
sqrt(diag(rec.mle$asy.var.coef))
rec.mle$var.pred
```

Example 3.33

```
x = diff(log(varve))
# Evaluate Sc on a Grid
c(0) -> w -> z
c() -> Sc -> Sz -> Szw
num = length(x)
th = seq(-.3,-.94,-.01)
for (p in 1:length(th)){
```

```
for (i in 2:num){ w[i] = x[i]-th[p]*w[i-1] }
Sc[p] = sum(w^2) }
plot(th, Sc, type="l", ylab=expression(S[c](theta)), xlab=expression(theta),
lwd=2)
# Gauss-Newton Estimation
r = acf(x, lag=1, plot=FALSE)$acf[-1]
rstart = (1-sqrt(1-4*(r^2)))/(2*r) # from (3.105)
c(0) -> w -> z
c() -> Sc -> Sz -> Szw -> para
niter = 12
para[1] = rstart
for (p in 1:niter){
for (i in 2:num){ w[i] = x[i]-para[p]*w[i-1]
z[i] = w[i-1]-para[p]*z[i-1] }
Sc[p] = sum(w^2)
Sz[p] = sum(z^2)
Szw[p] = sum(z*w)
para[p+1] = para[p] + Szw[p]/Sz[p] }

round(cbind(iteration=0:(niter-1), thetahat=para[1:niter] , Sc , Sz ), 3)
abline(v = para[1:12], lty=2)
points(para[1:12], Sc[1:12], pch=16)
```

Example 3.36

```
# generate data
set.seed(101010)
e   = rexp(150, rate=.5)
u   = runif(150,-1,1)
de  = e*sign(u)
dex = 50 + arima.sim(n=100, list(ar=.95), innov=de, n.start=50)
```

```
plot.ts(dex, type='o', ylab=expression(X[~t]))


# small sample and asymptotic distn
set.seed(111)
phi.yw = rep(NA, 1000)
for (i in 1:1000){
 e = rexp(150, rate=.5); u = runif(150,-1,1); de = e*sign(u)
 x = 50 + arima.sim(n=100,list(ar=.95), innov=de, n.start=50)
 phi.yw[i] = ar.yw(x, order=1)$ar
}
hist(phi.yw, prob=TRUE, main="", ylim=c(0,14), xlim=c(.70,1.05))
lines(density(phi.yw, bw=.015))
u = seq(.75, 1.1, by=.001)
lines(u, dnorm(u, mean=.96, sd=.03), lty=2, lwd=2)


# Bootstrap
set.seed(666)                 # not that 666
fit     = ar.yw(dex, order=1) # assumes the data were retained
m       = fit$x.mean          # estimate of mean
phi     = fit$ar              # estimate of phi
nboot   = 250                 # number of bootstrap replicates
resids  = fit$resid[-1]       # the 99 innovations
x.star  = dex                 # initialize x*
phi.star.yw = rep(NA, nboot)
#- start it up
for (i in 1:nboot) {
  resid.star = sample(resids, replace=TRUE)
  for (t in 1:99){ x.star[t+1] = m + phi*(x.star[t]-m) + resid.star[t]
  }
 phi.star.yw[i] = ar.yw(x.star, order=1)$ar
}
```

```
# Picture
culer = rgb(.5,.7,1,.5)
hist(phi.star.yw, 15, main="", prob=TRUE, xlim=c(.65,1.05), ylim=c(0,14),
col=culer, xlab=expression(hat(phi)))
lines(density(phi.yw, bw=.02), lwd=2) # from previous simulation
u = seq(.75, 1.1, by=.001) # normal approximation
lines(u, dnorm(u, mean=.96, sd=.03), lty=2, lwd=2)
legend(.65, 14, legend=c('true distribution', 'bootstrap distribution',
'normal approximation'), bty='n', lty=c(1,0,2), lwd=c(2,0,2),
col=1, pch=c(NA,22,NA), pt.bg=c(NA,culer,NA), pt.cex=2.5)
```

Example 3.38

```
set.seed(666)
x = arima.sim(list(order = c(0,1,1), ma = -0.8), n = 100)
(x.ima = HoltWinters(x, beta=FALSE, gamma=FALSE))  # α is 1-λ here
plot(x.ima)
```

Example 3.39, 3.40, and 3.43

```
plot(gnp)
acf2(gnp, 50)
gnpgr = diff(log(gnp))      # growth rate
plot(gnpgr)
acf2(gnpgr, 24)
sarima(gnpgr, 1, 0, 0)      # AR(1)
sarima(gnpgr, 0, 0, 2)      # MA(2)
ARMAtoMA(ar=.35, ma=0, 10)  # prints psi-weights
```

Example 3.41

```
sarima(log(varve), 0, 1, 1, no.constant=TRUE)   # ARIMA(0,1,1)
```

```
dev.new()
sarima(log(varve), 1, 1, 1, no.constant=TRUE)   # ARIMA(1,1,1)
```

Example 3.44

```
trend  = time(cmort)
temp   = tempr - mean(tempr)
temp2  = temp^2
summary(fit <- lm(cmort~trend + temp + temp2 + part, na.action=NULL))
acf2(resid(fit), 52) # implies AR2
sarima(cmort, 2,0,0, xreg=cbind(trend,temp,temp2,part) )
```

Example 3.45

```
# Note: this could benefit from a seasonal model fit, but it hasn't
#  been talked about yet - you could come back to this after the next section
dummy = ifelse(soi<0, 0, 1)
fish = ts.intersect(rec, soiL6=lag(soi,-6), dL6=lag(dummy,-6), dframe=TRUE)
summary(fit <- lm(rec ~soiL6*dL6, data=fish, na.action=NULL))
attach(fish)
plot(resid(fit))
acf2(resid(fit))     # indicates AR(2)
intract = soiL6*dL6  # interaction term
sarima(rec,2,0,0, xreg = cbind(soiL6, dL6, intract))
# not in text, but this works better
# sarima(rec,2,0,0,0,1,1,12, xreg = cbind(soiL6, dL6, intract))
```

Example 3.46

```
set.seed(666)
phi  = c(rep(0,11),.9)
sAR  = arima.sim(list(order=c(12,0,0), ar=phi), n=37)
sAR  = ts(sAR, freq=12)
```

```
layout(matrix(c(1,1,2, 1,1,3), nc=2))

par(mar=c(3,3,2,1), mgp=c(1.6,.6,0))

plot(sAR, axes=FALSE, main='seasonal AR(1)', xlab="year", type='c')

Months = c("J","F","M","A","M","J","J","A","S","O","N","D")

points(sAR, pch=Months, cex=1.25, font=4, col=1:4)

axis(1, 1:4)

abline(v=1:4, lty=2, col=gray(.7))

axis(2)

box()

ACF  = ARMAacf(ar=phi, ma=0, 100)

PACF = ARMAacf(ar=phi, ma=0, 100, pacf=TRUE)

plot(ACF,type="h", xlab="LAG", ylim=c(-.1,1))

abline(h=0)

plot(PACF, type="h", xlab="LAG", ylim=c(-.1,1))

abline(h=0)
```

Example 3.47

```
phi  = c(rep(0,11),.8)

ACF  = ARMAacf(ar=phi, ma=-.5, 50)[-1] # [-1] removes 0 lag

PACF = ARMAacf(ar=phi, ma=-.5, 50, pacf=TRUE)

par(mfrow=c(1,2))

plot(ACF, type="h", xlab="LAG", ylim=c(-.4,.8)); abline(h=0)

plot(PACF, type="h", xlab="LAG", ylim=c(-.4,.8)); abline(h=0)
```

Example 3.49

```
x     = AirPassengers

lx    = log(x)

dlx   = diff(lx)

ddlx  = diff(dlx, 12)

plot.ts(cbind(x, lx, dlx, ddlx), main="")
```

```
# below of interest for showing seasonal RW (not shown here):
par(mfrow=c(2,1))
monthplot(dlx)
monthplot(ddlx)

sarima(lx, 1,1,1, 0,1,1, 12)   # model 1
sarima(lx, 0,1,1, 0,1,1, 12)   # model 2 (the winner)
sarima(lx, 1,1,0, 0,1,1, 12)   # model 3

sarima.for(lx, 12, 0,1,1, 0,1,1,12)  # forecasts
```

[-]

[+] Chapter 4

Example 4.1

```
x1 = 2*cos(2*pi*1:100*6/100)  + 3*sin(2*pi*1:100*6/100)
x2 = 4*cos(2*pi*1:100*10/100) + 5*sin(2*pi*1:100*10/100)
x3 = 6*cos(2*pi*1:100*40/100) + 7*sin(2*pi*1:100*40/100)
x = x1 + x2 + x3

par(mfrow=c(2,2))
plot.ts(x1, ylim=c(-10,10), main = expression(omega==6/100~~~A^2==13))
plot.ts(x2, ylim=c(-10,10), main = expression(omega==10/100~~~A^2==41))
plot.ts(x3, ylim=c(-10,10), main = expression(omega==40/100~~~A^2==85))
plot.ts(x, ylim=c(-16,16), main="sum")
```

Example 4.2

```
P = abs(2*fft(x)/100)^2
Fr = 0:99/100
```

```
plot(Fr, P, type="o", xlab="frequency", ylab="periodogram")
```

Example 4.3

```
# modulation
t = 1:200
plot.ts(x <- 2*cos(2*pi*.2*t)*cos(2*pi*.01*t))    # not shown
lines(cos(2*pi*.19*t)+cos(2*pi*.21*t), col=2)     # the same
Px = Mod(fft(x))^2; plot(0:199/200, Px, type='o') # the periodogram


# star mag analysis
n    = length(star)
par(mfrow=c(2,1), mar=c(3,3,1,1), mgp=c(1.6,.6,0))
plot(star, ylab="star magnitude", xlab="day")
Per   = Mod(fft(star-mean(star)))^2/n
Freq  = (1:n -1)/n
plot(Freq[1:50], Per[1:50], type='h', lwd=3, ylab="Periodogram", xlab="Frequency")
u     = which.max(Per[1:50])     # 22 freq=21/600=.035 cycles/day
uu    = which.max(Per[1:50][-u]) # 25 freq=25/600=.041 cycles/day
1/Freq[22]; 1/Freq[26]           # period = days/cycle
text(.05, 7000, "24 day cycle")
text(.027, 9000, "29 day cycle")
#- another way to find the two peaks is to order on Per
y = cbind(1:50, Freq[1:50], Per[1:50]); y[order(y[,3]),]
```

Examples 4.5, 4.6, 4.7

```
par(mfrow=c(3,1))
arma.spec(log="no", main="White Noise")
arma.spec(ma=.5, log="no", main="Moving Average")
arma.spec(ar=c(1,-.9), log="no", main="Autoregression")
```

Example 4.10

```
x      = c(1,2,3,2,1)
c1     = cos(2*pi*1:5*1/5)
s1     = sin(2*pi*1:5*1/5)
c2     = cos(2*pi*1:5*2/5)
s2     = sin(2*pi*1:5*2/5)
omega1 = cbind(c1, s1)
omega2 = cbind(c2, s2)
anova(lm(x~ omega1+omega2) )  # ANOVA Table
Mod(fft(x))^2/5               # the periodogram (as a check)
```

Example 4.13

```
par(mfrow=c(2,1))
soi.per = mvspec(soi, log="no")
abline(v=1/4, lty="dotted")
rec.per = mvspec(rec, log="no")
abline(v=1/4, lty="dotted")

soi.per$spec[40]  # 0.97223;  soi pgram at freq 1/12 = 40/480
soi.per$spec[10]  # 0.05372;  soi pgram at freq 1/48 = 10/480

# conf intervals -  returned value:
U = qchisq(.025,2)    # 0.05063
L = qchisq(.975,2)    # 7.37775
2*soi.per$spec[10]/L  # 0.01456
2*soi.per$spec[10]/U  # 2.12220
2*soi.per$spec[40]/L  # 0.26355
2*soi.per$spec[40]/U  # 38.40108
```

```
# Repeat lines above using rec in place of soi
```

Example 4.14

```
soi.ave = mvspec(soi, kernel('daniell',4), log='no')
abline(v = c(.25,1,2,3), lty=2)
soi.ave$bandwidth      # = 0.225
df  = soi.ave$df       # df = 16.9875
U   = qchisq(.025, df) # U = 7.555916
L   = qchisq(.975, df) # L = 30.17425
soi.ave$spec[10]       # 0.0495202
soi.ave$spec[40]       # 0.1190800
# intervals
df*soi.ave$spec[10]/L  # 0.0278789
df*soi.ave$spec[10]/U  # 0.1113333
df*soi.ave$spec[40]/L  # 0.0670396
df*soi.ave$spec[40]/U  # 0.2677201

# Repeat above commands with soi replaced by rec, for example:
rec.ave = mvspec(rec, k, log="no")
abline(v=c(.25,1,2,3), lty=2)
# and so on.
```

Example 4.15

```
t = seq(0, 1, by=1/200)  # WARNING: using t is bad pRactice because it's reserved-
but let's be bad
amps = c(1, .5, .4, .3, .2, .1)
x = matrix(0, 201, 6)
for (j in 1:6) x[,j] = amps[j]*sin(2*pi*t*2*j)
x = ts(cbind(x, rowSums(x)), start=0, deltat=1/200)
ts.plot(x, lty=c(1:6, 1), lwd=c(rep(1,6), 2), ylab="Sinusoids")
```

```
names = c("Fundamental","2nd Harmonic","3rd Harmonic","4th Harmonic","5th Harmonic",
          "6th Harmonic","Formed Signal")
legend("topright", names, lty=c(1:6, 1), lwd=c(rep(1,6), 2))
rm(t)                    # Redemption
```

Example 4.16

```
kernel("modified.daniell", c(3,3))          # for a list
plot(kernel("modified.daniell", c(3,3)))    # for a graph

k        = kernel("modified.daniell", c(3,3))
soi.smo  = mvspec(soi, kernel=k, taper=.1, log="no")
abline(v = c(.25,1), lty=2)
## Repeat above lines with rec replacing soi
df       = soi.smo$df    # df = 17.42618
soi.smo$bandwidth        # B  = 0.2308103

# An easier way to obtain soi.smo:
soi.smo = mvspec(soi, spans=c(7,7), taper=.1, log="no")
```

Example 4.17

```
s0  = mvspec(soi, spans=c(7,7), plot=FALSE)            # no taper
s50 = mvspec(soi, spans=c(7,7), taper=.5, plot=FALSE)  # full taper
plot(s50$freq, s50$spec, log="y", type="l", ylab="spectrum", xlab="frequency")
lines(s0$freq, s0$spec, lty=2)
```

Example 4.18

```
spaic = spec.ar(soi, log="no", ylim=c(0,.3))             # min AIC spec, order = 15
text(frequency(soi)*1/52, .07, substitute(omega==1/52))  # El Nino Cycle
text(frequency(soi)*1/12, .27, substitute(omega==1/12))  # Yearly Cycle
```

```
(soi.ar = ar(soi, order.max=30))                        # estimates and AICs
dev.new()
plot(1:30, soi.ar$aic[-1], type="o")                    # plot AICs



# Better comparison of pseudo-ICs
n = length(soi)
c() -> AIC -> AICc -> BIC
for (k in 1:30){
  fit = ar(soi, order=k, aic=FALSE)
  sigma2  = fit$var.pred
  BIC[k]  = log(sigma2) + (k*log(n)/n)
  AICc[k] = log(sigma2) + ((n+k)/(n-k-2))
  AIC[k]  = log(sigma2) + ((n+2*k)/n)
}

dev.new()
IC = cbind(AIC, BIC+1)
ts.plot(IC, type="o", xlab="p", ylab="AIC / BIC")
grid()
text(15.2, -1.48, "AIC")
text(15,   -1.35, "BIC")
```

Example 4.21

```
sr = mvspec(cbind(soi,rec), kernel("daniell",9), plot.type="coh", plot=FALSE)
sr$df                       # df = 35.8625
f = qf(.999, 2, sr$df-2)  # f = 8.529792
C = f/(18+f)                # C = 0.3188779
abline(h = C)
```

Example 4.22

```
par(mfrow=c(3,1), mar=c(3,3,1,1), mgp=c(1.6,.6,0))
plot(soi)                          # plot data
plot(diff(soi))                    # plot first difference
k = kernel("modified.daniell", 6) # filter weights
plot(soif <- kernapply(soi, k))   # plot 12 month filter
dev.new()
spectrum(soif, spans=9, log="no") # spectral analysis (not shown)
abline(v=12/52, lty="dashed")
dev.new()
##-- frequency responses --##
par(mfrow=c(2,1), mar=c(3,3,1,1), mgp=c(1.6,.6,0))
w = seq(0, .5, by=.01)
FRdiff = abs(1-exp(2i*pi*w))^2
plot(w, FRdiff, type='l', xlab='frequency')
u = cos(2*pi*w)+cos(4*pi*w)+cos(6*pi*w)+cos(8*pi*w)+cos(10*pi*w)
FRma = ((1 + cos(12*pi*w) + 2*u)/12)^2
plot(w, FRma, type='l', xlab='frequency')
```

Example 4.24

```
LagReg(soi, rec, L=15, M=32, threshold=6)
LagReg(rec, soi, L=15, M=32, inverse=TRUE, threshold=.01)
# armax model
fish = ts.intersect(R=rec, RL1=lag(rec,-1), SL5=lag(soi,-5))
(u = lm(fish[,1]~fish[,2:3], na.action=NULL))
acf2(resid(u))       # suggests ar1
sarima(fish[,1], 1, 0, 0, xreg=fish[,2:3])
```

Example 4.25

```
SigExtract(soi, L=9, M=64, max.freq=.05)
```

Example 4.26

```
per = abs(fft(soiltemp-mean(soiltemp))/sqrt(64*36))^2
per2 = cbind(per[1:32,18:2], per[1:32,1:18])   # this and line below is just
rearranging
per3 = rbind(per2[32:2,], per2)                # results to get 0 frequency in the
middle

par(mar=c(1,2.5,0,0)+.1)
persp(-31:31/64, -17:17/36, per3, phi=30, theta=30, expand=.6, ticktype="detailed",
xlab="cycles/row",
       ylab="cycles/column", zlab="Periodogram Ordinate")
```

[-]

[+] Chapter 5

Example 5.1

```
# NOTE: I think 'fracdiff' is a dinosaur and I should have changed
# this in the new edition... so just below, I'll do it using 'arfima',
# which seems to work well
library(fracdiff)
lvarve = log(varve) - mean(log(varve))
varve.fd = fracdiff(lvarve, nar=0, nma=0, M=30)
varve.fd$d  # = 0.3841688
varve.fd$stderror.dpq  # = 4.589514e-06 (If you believe this, I have a bridge for
sale.)

p = rep(1,31)
```

```
for (k in 1:30){ p[k+1] = (k-varve.fd$d)*p[k]/(k+1) }
plot(1:30, p[-1], ylab=expression(pi(d)), xlab="Index", type="h", lwd=2)
res.fd = diffseries(log(varve), varve.fd$d)       # frac diff resids
res.arima = resid(arima(log(varve), order=c(1,1,1))) # arima resids

dev.new()
par(mfrow=c(2,1))
acf(res.arima, 100, xlim=c(4,97), ylim=c(-.2,.2), main="arima resids")
acf(res.fd, 100, xlim=c(4,97), ylim=c(-.2,.2), main="frac diff resids")
```

Example 5.1 redux

```
library(arfima)
summary(varve.fd <- arfima(log(varve)))  # d.hat = 0.3728, se(d,hat) = 0.0273
# residual stuff
innov = resid(varve.fd)
plot.ts(innov[[1]])
acf(innov[[1]])

## ... much better ...  sorry I didn't ...
## ... get it in for the newest edition ..
## ... once in awhile, they slip on by ...
```

Example 5.2

```
series = log(varve)  # specify series to be analyzed
d0 = .1              # initial value of d
n.per = nextn(length(series))
m = (n.per)/2  - 1
per = abs(fft(series-mean(series))[-1])^2  # remove 0 freq
per = per/n.per      # R doesn't scale fft by sqrt(n)
g = 4*(sin(pi*((1:m)/n.per))^2)
```

```
# Function to calculate -log.likelihood
whit.like = function(d){
 g.d=g^d
 sig2 = (sum(g.d*per[1:m])/m)
 log.like = m*log(sig2) - d*sum(log(g)) + m
 return(log.like)
}

# Estimation (?optim for details - output not shown)
(est = optim(d0, whit.like, gr=NULL, method="L-BFGS-B", hessian=TRUE, lower=-.5,
upper=.5,
             control=list(trace=1,REPORT=1)))

# Results  [d.hat = .380, se(dhat) = .028]
cat("d.hat =", est$par, "se(dhat) = ",1/sqrt(est$hessian),"\n")
g.dhat = g^est$par
sig2 = sum(g.dhat*per[1:m])/m
cat("sig2hat =",sig2,"\n")  # sig2hat = .229

u = spec.ar(log(varve), plot=FALSE)  # produces AR(8)
g = 4*(sin(pi*((1:500)/2000))^2)
fhat = sig2*g^{-est$par}             # long memory spectral estimate
plot(1:500/2000, log(fhat), type="l", ylab="log(spectrum)", xlab="frequency")
lines(u$freq[1:250], log(u$spec[1:250]), lty="dashed")
ar.mle(log(varve))                   # to get AR(8) estimates

# GPH estimate with big bandwidth or else estimate sucks
fdGPH(log(varve), bandw=.9)   # m = n^bandw
```

Example 5.3

```
library(tseries)
adf.test(log(varve), k=0)  # DF test
adf.test(log(varve))       # ADF test
pp.test(log(varve))        # PP test
```

Example 5.4

```
gnpgr = diff(log(gnp))          # get the returns
u     = sarima(gnpgr, 1, 0, 0)  # fit an AR(1)
acf2(resid(u$fit), 20)          # get (p)acf of the squared residuals

library(fGarch)
summary(garchFit(~arma(1,0)+garch(1,0), gnpgr))
```

Example 5.5 and 5.6

```
library(xts)   # needed to handle djia
djiar = diff(log(djia$Close))[-1]
acf2(djiar)    # exhibits some autocorrelation (not shown)
acf2(djiar^2)  # oozes autocorrelation (not shown)
library(fGarch)
# GARCH fit
summary(djia.g <- garchFit(~arma(1,0)+garch(1,1), data=djiar, cond.dist='std'))
plot(djia.g)    # to see all plot options
# APARCH fit
summary(djia.ap <- garchFit(~arma(1,0)+aparch(1,1), data=djiar, cond.dist='std'))
plot(djia.ap)
```

Example 5.7

```
plot(flu, type="c")
```

```
Months = c("J","F","M","A","M","J","J","A","S","O","N","D")
points(flu, pch=Months, cex=.8, font=2)
# Start analysis
dflu = diff(flu)
lag1.plot(dflu, corr=FALSE) # scatterplot with lowess fit
thrsh = .05 # threshold
Z = ts.intersect(dflu, lag(dflu,-1), lag(dflu,-2), lag(dflu,-3),
lag(dflu,-4) )
ind1 = ifelse(Z[,2] < thrsh, 1, NA) # indicator < thrsh
ind2 = ifelse(Z[,2] < thrsh, NA, 1) # indicator >= thrsh
X1 = Z[,1]*ind1
X2 = Z[,1]*ind2
summary(fit1 <- lm(X1~ Z[,2:5]) ) # case 1
summary(fit2 <- lm(X2~ Z[,2:5]) ) # case 2
D = cbind(rep(1, nrow(Z)), Z[,2:5]) # design matrix
p1 = D %*% coef(fit1) # get predictions
p2 = D %*% coef(fit2)
prd = ifelse(Z[,2] < thrsh, p1, p2)
plot(dflu, ylim=c(-.5,.5), type='p', pch=3)
lines(prd)
prde1 = sqrt(sum(resid(fit1)^2)/df.residual(fit1) )
prde2 = sqrt(sum(resid(fit2)^2)/df.residual(fit2) )
prde = ifelse(Z[,2] < thrsh, prde1, prde2)
tx = time(dflu)[-(1:4)]
xx = c(tx, rev(tx))
yy = c(prd-2*prde, rev(prd+2*prde))
polygon(xx, yy, border=8, col=gray(.6, alpha=.25) )
abline(h=.05, col=4, lty=6)

# Using tsDyn (not in text)
library(tsDyn)
```

```
# vignette("tsDyn")   # for package details (it's quirky, so you'll need this)
dflu = diff(flu)
(u = setar(dflu, m=4, thDelay=0))  # fit model and view results (thDelay=0 is lag 1
delay)
BIC(u); AIC(u)                      # if you want to try other models ... m=3 works
well too
plot(u)                             # graphics -  ?plot.setar for information
```

Example 5.8 and 5.9

```
soi.d   = resid(lm(soi~time(soi), na.action=NULL)) # detrended SOI
acf2(soi.d)
fit     = arima(soi.d, order=c(1,0,0))
ar1     = as.numeric(coef(fit)[1]) # = 0.5875
soi.pw  = resid(fit)
rec.fil = filter(rec, filter=c(1, -ar1), sides=1)
ccf(soi.pw, rec.fil, ylab="CCF", na.action=na.omit, panel.first=grid())

fish  = ts.intersect(rec, RL1=lag(rec,-1), SL5=lag(soi.d,-5))
(u    = lm(fish[,1]~fish[,2:3], na.action=NULL))
acf2(resid(u)) # suggests ar1
(arx  = sarima(fish[,1], 1, 0, 0, xreg=fish[,2:3])) # final model
pred  = rec + resid(arx$fit) # 1-step-ahead predictions
ts.plot(pred, rec, col=c('gray90',1), lwd=c(7,1))
```

Example 5.10 and 5.11

```
library(vars)
x = cbind(cmort, tempr, part)
summary(VAR(x, p=1, type="both"))  # "both" fits constant + trend


VARselect(x, lag.max=10, type="both")
```

```
summary(fit <- VAR(x, p=2, type="both"))

acf(resid(fit), 52)

serial.test(fit, lags.pt=12, type="PT.adjusted")


(fit.pr = predict(fit, n.ahead = 24, ci = 0.95))  # 4 weeks ahead

dev.new()

fanchart(fit.pr)  # plot prediction + error
```

Example 5.12

```
library(marima)

model   = define.model(kvar=3, ar=c(1,2), ma=c(1))

arp     = model$ar.pattern

map     = model$ma.pattern

cmort.d = resid(detr <- lm(cmort~ time(cmort), na.action=NULL))

xdata   = matrix(cbind(cmort.d, tempr, part), ncol=3)  # strip ts attributes

fit     = marima(xdata, ar.pattern=arp, ma.pattern=map, means=c(0,1,1), penalty=1)

# resid analysis (not displayed)

innov   = t(resid(fit))

plot.ts(innov)

acf(innov)

# fitted values for cmort

pred    = ts(t(fitted(fit))[,1], start=start(cmort), freq=frequency(cmort)) +

detr$coef[1] + detr$coef[2]*time(cmort)

plot(pred, ylab="Cardiovascular Mortality", lwd=2, col=4)

points(cmort)

# print estimates and corresponding t^2-statistic

short.form(fit$ar.estimates, leading=FALSE)

short.form(fit$ar.fvalues,   leading=FALSE)

short.form(fit$ma.estimates, leading=FALSE)

short.form(fit$ma.fvalues,   leading=FALSE)
```

```
fit$resid.cov # estimate of noise cov matrix
```

[-]

[+] Chapter 6

Example 6.1

```
plot(blood, type="o", pch=19, xlab="day", main="")
```

Example 6.2

```
ts.plot(globtemp, globtempl, col=c(6,4), ylab="Temperature Deviations")
```

Example 6.5

```
# generate data
set.seed(1)
num = 50
w = rnorm(num+1,0,1)
v = rnorm(num,0,1)

mu = cumsum(w)  # states:  mu[0], mu[1], . . ., mu[50]
y = mu[-1] + v  # obs:  y[1], . . ., y[50]

# filter and smooth (Ksmooth0 does both)
mu0 = 0; sigma0 = 1;  phi = 1; cQ = 1; cR = 1
ks = Ksmooth0(num, y, 1, mu0, sigma0, phi, cQ, cR)

# pictures
par(mfrow=c(3,1))
Time = 1:num
```

```
plot(Time, mu[-1], main="Prediction", ylim=c(-5,10))
  lines(ks$xp)
  lines(ks$xp+2*sqrt(ks$Pp), lty="dashed", col="blue")
  lines(ks$xp-2*sqrt(ks$Pp), lty="dashed", col="blue")

plot(Time, mu[-1], main="Filter", ylim=c(-5,10))
  lines(ks$xf)
  lines(ks$xf+2*sqrt(ks$Pf), lty="dashed", col="blue")
  lines(ks$xf-2*sqrt(ks$Pf), lty="dashed", col="blue")

plot(Time, mu[-1],  main="Smoother", ylim=c(-5,10))
  lines(ks$xs)
  lines(ks$xs+2*sqrt(ks$Ps), lty="dashed", col="blue")
  lines(ks$xs-2*sqrt(ks$Ps), lty="dashed", col="blue")

mu[1]; ks$x0n; sqrt(ks$P0n)   # initial value info

# In case you can't see the differences in the figures...
# ... either get new glasses or ...
# ... plot them on the same graph (not shown)
dev.new()
plot(Time, mu[-1], type='n')
abline(v=Time, lty=3, col=8)
abline(h=-1:5, lty=3, col=8)
lines(ks$xp, col=4, lwd=5)
lines(ks$xf, col=3, lwd=5)
lines(ks$xs, col=2, lwd=5)
points(Time, mu[-1], pch=19, cex=1.5)
names = c("predictor","filter","smoother")
legend("bottomright", names, col=4:2, lwd=5, lty=1, bg="white")
```

Example 6.6

```
# Generate Data
set.seed(999)
num = 100
N = num+1
x = arima.sim(n=N, list(ar = .8, sd=1))
y = ts(x[-1] + rnorm(num,0,1))

# Initial Estimates
u = ts.intersect(y, lag(y,-1), lag(y,-2))
varu = var(u)
coru = cor(u)
phi = coru[1,3]/coru[1,2]
q = (1-phi^2)*varu[1,2]/phi
r = varu[1,1] - q/(1-phi^2)
(init.par = c(phi, sqrt(q), sqrt(r)))

# Function to evaluate the likelihood
Linn=function(para){
  phi = para[1]; sigw = para[2]; sigv = para[3]
  Sigma0 = (sigw^2)/(1-phi^2); Sigma0[Sigma0<0]=0
  kf = Kfilter0(num,y,1,mu0=0,Sigma0,phi,sigw,sigv)
  return(kf$like)
  }

# Estimation
(est = optim(init.par, Linn, gr=NULL, method="BFGS", hessian=TRUE, control=list
(trace=1,REPORT=1)))
SE = sqrt(diag(solve(est$hessian)))
cbind(estimate=c(phi=est$par[1],sigw=est$par[2],sigv=est$par[3]), SE)
```

Example 6.7

```
# Setup
y = cbind(globtemp, globtempl)
num = nrow(y)
input = rep(1,num)
A = array(rep(1,2), dim=c(2,1,num))
mu0 = -.35; Sigma0 = 1;  Phi = 1

# Function to Calculate Likelihood
Linn=function(para){
 cQ = para[1]      # sigma_w
  cR1 = para[2]    # 11 element of chol(R)
  cR2 = para[3]    # 22 element of chol(R)
  cR12 = para[4]   # 12 element of chol(R)
 cR = matrix(c(cR1,0,cR12,cR2),2)  # put the matrix together
 drift = para[5]
 kf = Kfilter1(num,y,A,mu0,Sigma0,Phi,drift,0,cQ,cR,input)
 return(kf$like)
 }

# Estimation
init.par = c(.1,.1,.1,0,.05)  # initial values of parameters
(est = optim(init.par, Linn, NULL, method="BFGS", hessian=TRUE, control=list(trace=1,
REPORT=1)))
SE = sqrt(diag(solve(est$hessian)))

# Summary of estimation
estimate = est$par; u = cbind(estimate, SE)
rownames(u)=c("sigw","cR11", "cR22", "cR12", "drift"); u
```

```
# Smooth (first set parameters to their final estimates)
cQ    = est$par[1]
 cR1  = est$par[2]
 cR2  = est$par[3]
 cR12 = est$par[4]
cR    = matrix(c(cR1,0,cR12,cR2), 2)
(R    = t(cR)%*%cR)    #  to view the estimated R matrix
drift = est$par[5]
ks    = Ksmooth1(num,y,A,mu0,Sigma0,Phi,drift,0,cQ,cR,input)

# Plot
xsm  = ts(as.vector(ks$xs), start=1880)
rmse = ts(sqrt(as.vector(ks$Ps)), start=1880)
plot(xsm, ylim=c(-.6, 1), ylab='Temperature Deviations')
  xx = c(time(xsm), rev(time(xsm)))
  yy = c(xsm-2*rmse, rev(xsm+2*rmse))
polygon(xx, yy, border=NA, col=gray(.6, alpha=.25))
lines(globtemp, type='o', pch=2, col=4, lty=6)
lines(globtempl, type='o', pch=3, col=3, lty=6)
```

Example 6.8

```
library(nlme)   # loads package nlme

# Generate data (same as Example 6.6)
set.seed(999); num = 100; N = num+1
x = arima.sim(n=N, list(ar = .8, sd=1))
y = ts(x[-1] + rnorm(num,0,1))

# Initial Estimates
```

```
u = ts.intersect(y,lag(y,-1),lag(y,-2))
varu = var(u); coru = cor(u)
phi = coru[1,3]/coru[1,2]
q = (1-phi^2)*varu[1,2]/phi
r = varu[1,1] - q/(1-phi^2)
cr = sqrt(r); cq = sqrt(q); mu0 = 0; Sigma0 = 2.8
(em = EM0(num, y, 1, mu0, Sigma0, phi, cq, cr, 75, .00001))

# Standard Errors  (this uses nlme)
phi = em$Phi; cq = chol(em$Q); cr = chol(em$R)
mu0 = em$mu0; Sigma0 = em$Sigma0
para = c(phi, cq, cr)

# Evaluate likelihood at estimates
Linn=function(para){
  kf = Kfilter0(num, y, 1, mu0, Sigma0, para[1], para[2], para[3])
  return(kf$like)
  }
emhess = fdHess(para, function(para) Linn(para))
SE = sqrt(diag(solve(emhess$Hessian)))

# Display summary of estimation
estimate = c(para, em$mu0, em$Sigma0); SE = c(SE,NA,NA)
u = cbind(estimate, SE)
rownames(u) = c("phi","sigw","sigv","mu0","Sigma0")
u
```

Example 6.9

```
y    = cbind(WBC, PLT, HCT)
num  = nrow(y)
```

```
A    = array(0, dim=c(3,3,num))  # creates num 3x3 zero matrices
for(k in 1:num) if (y[k,1] > 0) A[,,k]= diag(1,3)


# Initial values
mu0    = matrix(0,3,1)
Sigma0 = diag(c(.1,.1,1) ,3)
Phi    = diag(1,3)
cQ     = diag(c(.1,.1,1), 3)
cR     = diag(c(.1,.1,1), 3)
(em = EM1(num, y, A, mu0, Sigma0, Phi, cQ, cR, 100, .001))


# Graph smoother
ks  = Ksmooth1(num, y, A, em$mu0, em$Sigma0, em$Phi, 0, 0, chol(em$Q), chol(em$R), 0)
y1s = ks$xs[1,,]
y2s = ks$xs[2,,]
y3s = ks$xs[3,,]
p1  = 2*sqrt(ks$Ps[1,1,])
p2  = 2*sqrt(ks$Ps[2,2,])
p3  = 2*sqrt(ks$Ps[3,3,])
par(mfrow=c(3,1))
plot(WBC, type='p', pch=19, ylim=c(1,5), xlab='day')
 lines(y1s)
 lines(y1s+p1, lty=2, col=4)
 lines(y1s-p1, lty=2, col=4)
plot(PLT, type='p', ylim=c(3,6), pch=19, xlab='day')
 lines(y2s)
 lines(y2s+p2, lty=2, col=4)
 lines(y2s-p2, lty=2, col=4)
plot(HCT, type='p', pch=19, ylim=c(20,40), xlab='day')
 lines(y3s)
 lines(y3s+p3, lty=2, col=4)
```

```
lines(y3s-p3, lty=2, col=4)
```

Example 6.10

```
num = length(jj)
A = cbind(1,1,0,0)

# Function to Calculate Likelihood
Linn=function(para){
 Phi = diag(0,4)
 Phi[1,1] = para[1]
 Phi[2,]=c(0,-1,-1,-1); Phi[3,]=c(0,1,0,0); Phi[4,]=c(0,0,1,0)
 cQ1 = para[2]; cQ2 = para[3]     # sqrt q11 and sqrt q22
 cQ=diag(0,4); cQ[1,1]=cQ1; cQ[2,2]=cQ2
 cR = para[4]                     # sqrt r11
 kf = Kfilter0(num,jj,A,mu0,Sigma0,Phi,cQ,cR)
 return(kf$like)
 }

# Initial Parameters
mu0      = c(.7,0,0,0)
Sigma0   = diag(.04,4)
init.par = c(1.03, .1, .1, .5)  # Phi[1,1], the 2 Qs and R

# Estimation
est = optim(init.par, Linn, NULL, method="BFGS", hessian=TRUE, control=list(trace=1,
REPORT=1))
SE  = sqrt(diag(solve(est$hessian)))
u   = cbind(estimate=est$par,SE)
rownames(u)=c("Phi11","sigw1","sigw2","sigv"); u
```

```
# Smooth
Phi      = diag(0,4)
Phi[1,1] = est$par[1]
Phi[2,]  = c(0,-1,-1,-1)
Phi[3,]  = c(0,1,0,0)
Phi[4,]  = c(0,0,1,0)
cQ1      = est$par[2]
cQ2      = est$par[3]
cQ       = diag(0,4)
cQ[1,1]  = cQ1
cQ[2,2]  = cQ2
cR       = est$par[4]
ks       = Ksmooth0(num, jj, A, mu0, Sigma0, Phi, cQ, cR)


# Plots
Tsm   = ts(ks$xs[1,,], start=1960, freq=4)
Ssm   = ts(ks$xs[2,,], start=1960, freq=4)
p1    = 3*sqrt(ks$Ps[1,1,]); p2 = 3*sqrt(ks$Ps[2,2,])
par(mfrow=c(2,1))
plot(Tsm, main='Trend Component', ylab='Trend')
  xx  = c(time(jj), rev(time(jj)))
  yy  = c(Tsm-p1, rev(Tsm+p1))
polygon(xx, yy, border=NA, col=gray(.5, alpha = .3))
plot(jj, main='Data & Trend+Season', ylab='J&J QE/Share', ylim=c(-.5,17))
  xx  = c(time(jj), rev(time(jj)) )
  yy  = c((Tsm+Ssm)-(p1+p2), rev((Tsm+Ssm)+(p1+p2)) )
polygon(xx, yy, border=NA, col=gray(.5, alpha = .3))


# Forecast
dev.new()
n.ahead = 12
```

```
y       = ts(append(jj, rep(0,n.ahead)), start=1960, freq=4)
rmspe   = rep(0,n.ahead)
x00     = ks$xf[,,num]
P00     = ks$Pf[,,num]
Q       = t(cQ)%*%cQ
R       = t(cR)%*%(cR)
for (m in 1:n.ahead){
       xp = Phi%*%x00
       Pp = Phi%*%P00%*%t(Phi)+Q
      sig = A%*%Pp%*%t(A)+R
        K = Pp%*%t(A)%*%(1/sig)
      x00 = xp
      P00 = Pp-K%*%A%*%Pp
 y[num+m] = A%*%xp
 rmspe[m] = sqrt(sig)
}
plot(y, type='o', main='', ylab='J&J QE/Share', ylim=c(5,30),
xlim = c(1975,1984))
upp  = ts(y[(num+1):(num+n.ahead)]+2*rmspe, start=1981, freq=4)
low  = ts(y[(num+1):(num+n.ahead)]-2*rmspe, start=1981, freq=4)
 xx  = c(time(low), rev(time(upp)))
 yy  = c(low, rev(upp))
polygon(xx, yy, border=8, col=gray(.5, alpha = .3))
abline(v=1981, lty=3)
```

Example 6.12

```
# Preliminary analysis
fit1   = sarima(cmort, 2,0,0, xreg=time(cmort))
acf(cbind(dmort <- resid(fit1$fit), tempr, part))
lag2.plot(tempr, dmort, 8)
```

```
lag2.plot(part, dmort, 8)


# quick and dirty fit (detrend then fit ARMAX)
trend   = time(cmort) - mean(time(cmort))
dcmort  = resid(fit2 <- lm(cmort~trend, na.action=NULL))  # detrended mort
u       = ts.intersect(dM=dcmort, dM1=lag(dcmort,-1), dM2=lag(dcmort,-2), T1=lag
(tempr,-1),
             P=part, P4=lag(part,-4))
sarima(u[,1], 0,0,0, xreg=u[,2:6])  # ARMAX fit with residual analysis


# all estimates at once
trend   = time(cmort) - mean(time(cmort)) # center time
const   = time(cmort)/time(cmort)         # appropriate time series of 1s
ded     = ts.intersect(M=cmort, T1=lag(tempr,-1), P=part, P4=lag(part,-4), trend,
const)
y       = ded[,1]
input   = ded[,2:6]
num     = length(y)
A       = array(c(1,0), dim = c(1,2,num))
# Function to Calculate Likelihood
Linn=function(para){
 phi1   = para[1]; phi2 = para[2]; cR = para[3];  b1 = para[4]
 b2     = para[5];   b3 = para[6]; b4 = para[7]; alf = para[8]
 mu0    = matrix(c(0,0), 2, 1)
 Sigma0 = diag(100, 2)
 Phi    = matrix(c(phi1, phi2, 1, 0), 2)
 Theta  = matrix(c(phi1, phi2), 2)
 Ups    = matrix(c(b1, 0, b2, 0, b3, 0, 0, 0, 0, 0), 2, 5)
 Gam    = matrix(c(0, 0, 0, b4, alf), 1, 5); cQ = cR; S = cR^2
 kf     = Kfilter2(num, y, A, mu0, Sigma0, Phi, Ups, Gam, Theta, cQ, cR, S, input)
return(kf$like)
```

```
}
# Estimation - prelim analysis gives good starting values
init.par = c(phi1=.3, phi2=.3, cR=5, b1=-.2, b2=.1, b3=.05, b4=-1.6, alf=mean(cmort))
L = c( 0,  0,  1, -1,  0,  0, -2, 70)   # lower bound on parameters
U = c(.5, .5, 10,  0, .5, .5,  0, 90)   # upper bound - used in optim
est      = optim(init.par, Linn, NULL, method='L-BFGS-B', lower=L, upper=U,
             hessian=TRUE, control=list(trace=1, REPORT=1, factr=10^8))
SE       = sqrt(diag(solve(est$hessian)))
round(cbind(estimate=est$par, SE), 3) # results

# Residual Analysis (not shown)
phi1   = est$par[1]; phi2 = est$par[2]
cR     = est$par[3]; b1   = est$par[4]
b2     = est$par[5]; b3   = est$par[6]
b4     = est$par[7]; alf  = est$par[8]
mu0    = matrix(c(0,0), 2, 1)
Sigma0 = diag(100, 2)
Phi    = matrix(c(phi1, phi2, 1, 0), 2)
Theta  = matrix(c(phi1, phi2), 2)
Ups    = matrix(c(b1, 0, b2, 0, b3, 0, 0, 0, 0, 0), 2, 5)
Gam    = matrix(c(0, 0, 0, b4, alf), 1, 5)
cQ     = cR
S      = cR^2
kf     = Kfilter2(num, y, A, mu0, Sigma0, Phi, Ups, Gam, Theta, cQ, cR, S, input)
res    = ts(as.vector(kf$innov), start=start(cmort), freq=frequency(cmort))
sarima(res, 0,0,0, no.constant=TRUE)  # gives a full residual analysis

# Similar fit with but with trend in the X of ARMAX
trend  = time(cmort) - mean(time(cmort))
u      = ts.intersect(M=cmort, M1=lag(cmort,-1), M2=lag(cmort,-2), T1=lag(tempr,-1),
```

```
            P=part, P4=lag(part -4), trend)
sarima(u[,1], 0,0,0, xreg=u[,2:7])
```

Example 6.13

```
library(psych)                      # load psych package for scatter.hist
library(plyr)                       # load plyr to track iterations

################
# NOTE: Change lines below to tol=.01 or tol=.001 and nboot=100 or nboot=200
#            if this takes a long time to run - depends on your machine

tol = sqrt(.Machine$double.eps)  # determines convergence of optimizer
nboot = 500                      # number of bootstrap replicates
#################

y     = window(qinfl, c(1953,1), c(1965,2))  # inflation
z     = window(qintr, c(1953,1), c(1965,2))  # interest
num   = length(y)
A     = array(z, dim=c(1,1,num))
input = matrix(1,num,1)
# Function to Calculate Likelihood
Linn  = function(para, y.data){  # pass data also
   phi = para[1]; alpha = para[2]
   b   = para[3]; Ups   = (1-phi)*b
   cQ  = para[4]; cR    = para[5]
   kf  = Kfilter2(num,y.data,A,mu0,Sigma0,phi,Ups,alpha,1,cQ,cR,0,input)
   return(kf$like)
}
# Parameter Estimation
mu0 = 1; Sigma0 = .01
init.par = c(phi=.84, alpha=-.77, b=.85, cQ=.12, cR=1.1) # initial values
```

```
est = optim(init.par,  Linn, NULL, y.data=y, method="BFGS", hessian=TRUE,
             control=list(trace=1, REPORT=1, reltol=tol))
SE  = sqrt(diag(solve(est$hessian)))
phi = est$par[1]; alpha = est$par[2]
b   = est$par[3]; Ups   = (1-phi)*b
cQ  = est$par[4]; cR    = est$par[5]
round(cbind(estimate=est$par, SE), 3)
# BEGIN BOOTSTRAP
# Run the filter at the estimates
kf = Kfilter2(num,y,A,mu0,Sigma0,phi,Ups,alpha,1,cQ,cR,0,input)
# Pull out necessary values from the filter and initialize
xp       = kf$xp
innov    = kf$innov
sig      = kf$sig
K        = kf$K
e        = innov/sqrt(sig)
e.star   = e                      # initialize values
y.star   = y
xp.star = xp
k        = 4:50                   # hold first 3 observations fixed
para.star = matrix(0, nboot, 5)  # to store estimates
init.par  =  c(.84, -.77, .85, .12, 1.1)
pr <- progress_text()            # displays progress
pr$init(nboot)
for (i in 1:nboot){
 pr$step()
 e.star[k] = sample(e[k], replace=TRUE)
 for (j in k){ xp.star[j] = phi*xp.star[j-1] + Ups+K[j]*sqrt(sig[j])*e.star[j] }
 y.star[k] = z[k]*xp.star[k] + alpha + sqrt(sig[k])*e.star[k]
 est.star  = optim(init.par, Linn, NULL, y.data=y.star, method="BFGS", control=list
(reltol=tol))
```

```
 para.star[i,] = cbind(est.star$par[1], est.star$par[2], est.star$par[3],
                       abs(est.star$par[4]), abs(est.star$par[5]))
}
# Some summary statistics
rmse = rep(NA,5)                 # SEs from the bootstrap
for(i in 1:5){rmse[i]=sqrt(sum((para.star[,i]-est$par[i])^2)/nboot)
              cat(i, rmse[i],"\n")
             }
# Plot phi and sigw
phi  = para.star[,1]
sigw = abs(para.star[,4])
phi  = ifelse(phi<0, NA, phi)    # any phi < 0 not plotted
scatter.hist(sigw, phi, ylab=expression(phi), xlab=expression(sigma[~w]),
smooth=FALSE, correl=FALSE,
              density=FALSE, ellipse=FALSE, title='', pch=19, col=gray(.1,alpha=.33),
              panel.first=grid(lty=2), cex.lab=1.2)
```

Example 6.14

```
set.seed(123)
num   = 50
w     = rnorm(num,0,.1)
x     = cumsum(cumsum(w))
y     = x + rnorm(num,0,1)
plot.ts(x, ylab="", lwd=2, ylim=c(-1,8))
lines(y, type='o', col=8)
## State Space ##
Phi   = matrix(c(2,1,-1,0),2)
A     = matrix(c(1,0),1)
mu0   = matrix(0,2); Sigma0 = diag(1,2)
Linn  = function(para){
```

```
  sigw = para[1]
  sigv = para[2]
  cQ   = diag(c(sigw,0))
  kf   = Kfilter0(num, y, A, mu0, Sigma0, Phi, cQ, sigv)
return(kf$like)
}
## Estimation ##
init.par = c(.1, 1)
(est  = optim(init.par, Linn, NULL, method="BFGS", hessian=TRUE, control=list(trace=1,
REPORT=1)))
SE    = sqrt(diag(solve(est$hessian)))
# Summary of estimation
estimate = est$par; u = cbind(estimate, SE)
rownames(u) = c("sigw","sigv"); u
# Smooth
sigw  = est$par[1]
cQ    = diag(c(sigw,0))
sigv  = est$par[2]
ks    = Ksmooth0(num, y, A, mu0, Sigma0, Phi, cQ, sigv)
xsmoo = ts(ks$xs[1,1,]); psmoo = ts(ks$Ps[1,1,])
upp   = xsmoo+2*sqrt(psmoo)
low   = xsmoo-2*sqrt(psmoo)
lines(xsmoo, col=4, lty=2, lwd=3)
lines(upp, col=4, lty=2); lines(low, col=4, lty=2)
lines(smooth.spline(y), lty=1, col=2)
legend("topleft", c("Observations","State"), pch=c(1,-1), lty=1, lwd=c(1,2), col=c
(8,1))
legend("bottomright", c("Smoother", "GCV Spline"), lty=c(2,1), lwd=c(3,1), col=c(4,2))
```

Example 6.16

```
library(depmixS4)
model <- depmix(EQcount ~1, nstates=2, data=data.frame(EQcount), family=poisson())
set.seed(90210)
summary(fm <- fit(model))   # estimation results
##-- Get Parameters --##
u = as.vector(getpars(fm))  # ensure state 1 has smaller lambda
 if (u[7] <= u[8]) {  para.mle = c(u[3:6], exp(u[7]), exp(u[8]))
  }  else  {  para.mle = c(u[6:3], exp(u[8]), exp(u[7])) }
mtrans = matrix(para.mle[1:4], byrow=TRUE, nrow=2)
lams   = para.mle[5:6]
pi1    = mtrans[2,1]/(2 - mtrans[1,1] - mtrans[2,2]);  pi2 = 1-pi1
##-- Graphics --##
layout(matrix(c(1,2,1,3), 2))
par(mar = c(3,3,1,1), mgp = c(1.6,.6,0))
# data and states
plot(EQcount, main="", ylab='EQcount', type='h', col=gray(.7))
text(EQcount, col=6*posterior(fm)[,1]-2, labels=posterior(fm)[,1], cex=.9)
# prob of state 2
plot(ts(posterior(fm)[,3], start=1900), ylab = expression(hat(pi)[~2]*'(t|n)'));
abline(h=.5, lty=2)
# histogram
hist(EQcount, breaks=30, prob=TRUE, main="")
xvals = seq(1,45)
u1 = pi1*dpois(xvals, lams[1])
u2 = pi2*dpois(xvals, lams[2])
lines(xvals, u1, col=4);   lines(xvals, u2, col=2)
##--  Bootstap --##
# function to generate data
pois.HMM.generate_sample = function(n,m,lambda,Mtrans,StatDist=NULL){
 # n = data length, m = number of states,
 # Mtrans = transition matrix, StatDist = stationary distn
```

```
 if(is.null(StatDist)) StatDist = solve(t(diag(m)-Mtrans +1),rep(1,m))
  mvect = 1:m
  state = numeric(n)
  state[1] = sample(mvect ,1, prob=StatDist)
  for (i in 2:n)
       state[i] = sample(mvect ,1,prob=Mtrans[state[i-1] ,])
  y = rpois(n,lambda=lambda[state ])
  list(y= y, state= state)
}
# start it up
set.seed(10101101)
nboot     = 100
nobs      = length(EQcount)
para.star = matrix(NA, nrow=nboot, ncol = 6)
for (j in 1:nboot){
 x.star = pois.HMM.generate_sample(n=nobs, m=2, lambda=lams, Mtrans=mtrans)$y
 model <- depmix(x.star ~1, nstates=2, data=data.frame(x.star), family=poisson())
 u = as.vector(getpars(fit(model, verbose=0)))
 # make sure state 1 is the one with the smaller intensity parameter
 if (u[7] <= u[8]) { para.star[j,] = c(u[3:6], exp(u[7]), exp(u[8])) }
     else  { para.star[j,] = c(u[6:3], exp(u[8]), exp(u[7])) }         }
# bootstrapped std errors
SE = sqrt(apply(para.star,2,var) + (apply(para.star,2,mean)-para.mle)^2)[c(1,4:6)]
names(SE)=c('seM11/M12', 'seM21/M22', 'seLam1', 'seLam2'); SE
```

Example 6.17

```
library(depmixS4)
y = ts(sp500w, start=2003, freq=52)  # make data depmix friendly
mod3 <- depmix(y~1, nstates=3, data=data.frame(y))
set.seed(2)
```

```
summary(fm3 <- fit(mod3))

##-- Graphics --##

layout(matrix(c(1,2, 1,3), 2), heights=c(1,.75))

par(mar=c(2.5,2.5,.5,.5), mgp=c(1.6,.6,0))

plot(y, main="", ylab='S&P500 Weekly Returns', col=gray(.7), ylim=c(-.11,.11))

 culer = 4-posterior(fm3)[,1];  culer[culer==3]=4  # switch labels 1 and 3

 text(y, col=culer, labels=4-posterior(fm3)[,1])

##-- MLEs --##

 para.mle    = as.vector(getpars(fm3)[-(1:3)])

 permu       = matrix(c(0,0,1,0,1,0,1,0,0), 3,3)   # for the label switch

 (mtrans.mle = permu%*%round(t(matrix(para.mle[1:9],3,3)),3)%*%permu)

 (norms.mle  = round(matrix(para.mle[10:15],2,3),3)%*%permu)

acf(y^2, xlim=c(.02,.5), ylim=c(-.09,.5), panel.first=grid(lty=2) )

hist(y, 25, prob=TRUE, main='')

 culer=c(1,2,4); pi.hat = colSums(posterior(fm3)[-1,2:4])/length(y)

 for (i in 1:3) { mu=norms.mle[1,i]; sig = norms.mle[2,i]

 x = seq(-.15,.12, by=.001)

lines(x, pi.hat[4-i]*dnorm(x, mean=mu, sd=sig), col=culer[i])   }

##-- Bootstrap --##

set.seed(666);  n.obs = length(y);  n.boot = 100

para.star = matrix(NA, nrow=n.boot, ncol = 15)

respst <- para.mle[10:15];  trst <- para.mle[1:9]

for ( nb in 1:n.boot ){

  mod <- simulate(mod3)

  y.star = as.vector(mod@response[[1]][[1]]@y)

  dfy = data.frame(y.star)

  mod.star <- depmix(y.star~1, data=dfy, respst=respst, trst=trst, nst=3)

  fm.star = fit(mod.star, emcontrol=em.control(tol = 1e-5), verbose=FALSE)

  para.star[nb,] = as.vector(getpars(fm.star)[-(1:3)]) }

# bootstrap stnd errors

SE = sqrt(apply(para.star,2,var) + (apply(para.star,2,mean)-para.mle)^2)
```

```
(SE.mtrans.mle = permu%*%round(t(matrix(SE[1:9],3,3)),3)%*%permu)

(SE.norms.mle = round(matrix(SE[10:15], 2,3),3)%*%permu)
```

Example 6.18

```
library(MSwM)

set.seed(90210)

dflu  = diff(flu)

model = lm(dflu~ 1)

mod   = msmFit(model, k=2, p=2, sw=rep(TRUE,4)) # 2 regimes, AR(2)s

summary(mod)

plotProb(mod, which=3)
```

Example 6.22

```
y = as.matrix(flu); num = length(y); nstate = 4;

M1 = as.matrix(cbind(1,0,0,1))  # obs matrix normal

M2 = as.matrix(cbind(1,0,1,1))  # obs matrix flu epi

prob = matrix(0,num,1); yp = y  # to store pi2(t|t-1) & y(t|t-1)

xfilter = array(0, dim=c(nstate,1,num)) # to store x(t|t)

# Function to Calculate Likelihood

Linn = function(para){

  alpha1 = para[1]; alpha2 = para[2]; beta0 = para[3]

  sQ1 = para[4];  sQ2 = para[5];  like=0

  xf  = matrix(0, nstate, 1)  # x filter

  xp  = matrix(0, nstate, 1)  # x pred

  Pf  = diag(.1, nstate)      # filter cov

  Pp  = diag(.1, nstate)      # pred cov

  pi11 <- .75 -> pi22;  pi12 <- .25 -> pi21; pif1 <- .5 -> pif2

  phi = matrix(0,nstate,nstate)

  phi[1,1] = alpha1; phi[1,2] = alpha2; phi[2,1]=1; phi[4,4]=1

  Ups = as.matrix(rbind(0,0,beta0,0))
```

```
  Q   = matrix(0,nstate,nstate)
  Q[1,1] = sQ1^2; Q[3,3] = sQ2^2; R=0  # R=0 in final model
  # begin filtering #
    for(i in 1:num){
    xp   = phi%*%xf + Ups; Pp = phi%*%Pf%*%t(phi) + Q
    sig1 = as.numeric(M1%*%Pp%*%t(M1) + R)
    sig2 = as.numeric(M2%*%Pp%*%t(M2) + R)
    k1   = Pp%*%t(M1)/sig1; k2 = Pp%*%t(M2)/sig2
    e1   = y[i]-M1%*%xp; e2 = y[i]-M2%*%xp
    pip1 = pif1*pi11 + pif2*pi21; pip2 = pif1*pi12 + pif2*pi22
    den1 = (1/sqrt(sig1))*exp(-.5*e1^2/sig1)
    den2 = (1/sqrt(sig2))*exp(-.5*e2^2/sig2)
    denm = pip1*den1 + pip2*den2
    pif1 = pip1*den1/denm; pif2 = pip2*den2/denm
    pif1 = as.numeric(pif1); pif2 = as.numeric(pif2)
    e1   = as.numeric(e1); e2=as.numeric(e2)
    xf   = xp + pif1*k1*e1 + pif2*k2*e2
    eye  = diag(1, nstate)
    Pf   = pif1*(eye-k1%*%M1)%*%Pp + pif2*(eye-k2%*%M2)%*%Pp
    like = like - log(pip1*den1 + pip2*den2)
    prob[i]<<-pip2; xfilter[,,i]<<-xf; innov.sig<<-c(sig1,sig2)
    yp[i]<<-ifelse(pip1 > pip2, M1%*%xp, M2%*%xp)
    }
return(like)
}
# Estimation
alpha1 = 1.4; alpha2 = -.5; beta0 = .3; sQ1 = .1; sQ2 = .1
init.par = c(alpha1, alpha2, beta0, sQ1, sQ2)
(est = optim(init.par, Linn, NULL, method='BFGS', hessian=TRUE, control=list(trace=1,
REPORT=1)))
```

```
SE   = sqrt(diag(solve(est$hessian)))
u    = cbind(estimate=est$par, SE)
rownames(u)=c('alpha1','alpha2','beta0','sQ1','sQ2'); u
# Graphics
predepi = ifelse(prob<.5,0,1); k = 6:length(y)
Time    = time(flu)[k]
regime  = predepi[k]+1
par(mfrow=c(3,1), mar=c(2,3,1,1)+.1)
plot(Time, y[k], type="n", ylab="")
 grid(lty=2); lines(Time, y[k],  col=gray(.7))
 text(Time, y[k], col=regime, labels=regime, cex=1.1)
 text(1979,.95,"(a)")
plot(Time, xfilter[1,,k], type="n", ylim=c(-.1,.4), ylab="")
 grid(lty=2); lines(Time, xfilter[1,,k])
 lines(Time, xfilter[3,,k]); lines(Time, xfilter[4,,k])
 text(1979,.35,"(b)")
plot(Time, y[k], type="n",   ylim=c(.1,.9),ylab="")
 grid(lty=2); points(Time, y[k], pch=19)
 prde1 = 2*sqrt(innov.sig[1]); prde2 = 2*sqrt(innov.sig[2])
 prde = ifelse(predepi[k]<.5, prde1,prde2)
   xx = c(Time, rev(Time))
   yy = c(yp[k]-prde, rev(yp[k]+prde))
 polygon(xx, yy, border=8, col=gray(.6, alpha=.3))
 text(1979,.85,"(c)")
```

Example 6.23

```
y   = log(nyse^2)
num =length(y)


# Initial Parameters
```

```
phi0=0; phi1=.95; sQ=.2; alpha=mean(y); sR0=1; mu1=-3; sR1=2
init.par = c(phi0,phi1,sQ,alpha,sR0,mu1,sR1)

# Innovations Likelihood
Linn = function(para){
  phi0=para[1]; phi1=para[2]; sQ=para[3]; alpha=para[4]
  sR0=para[5]; mu1=para[6]; sR1=para[7]
  sv = SVfilter(num,y,phi0,phi1,sQ,alpha,sR0,mu1,sR1)
  return(sv$like)
}

# Estimation
(est = optim(init.par, Linn, NULL, method="BFGS", hessian=TRUE, control=list(trace=1,
REPORT=1)))
SE = sqrt(diag(solve(est$hessian)))
u = cbind(estimates=est$par, SE)
rownames(u)=c("phi0","phi1","sQ","alpha","sigv0","mu1","sigv1"); u

# Graphics   (need filters at the estimated parameters)
phi0=est$par[1]; phi1=est$par[2]; sQ=est$par[3]; alpha=est$par[4]
sR0=est$par[5]; mu1=est$par[6]; sR1=est$par[7]
sv = SVfilter(num,y,phi0,phi1,sQ,alpha,sR0,mu1,sR1)

# densities plot (f is chi-sq, fm is fitted mixture)
x = seq(-15,6,by=.01)
f = exp(-.5*(exp(x)-x))/(sqrt(2*pi))
f0 = exp(-.5*(x^2)/sR0^2)/(sR0*sqrt(2*pi))
f1 = exp(-.5*(x-mu1)^2/sR1^2)/(sR1*sqrt(2*pi))
fm = (f0+f1)/2
plot(x, f, type="l"); lines(x, fm, lty=2,lwd=2)
```

```
dev.new()
Time = 701:1100
plot (Time, nyse[Time], type='l', col=4, lwd=2, ylab='', xlab='', ylim=c(-.18,.12))
lines(Time, sv$xp[Time]/10, lwd=2, col=6)
```

Example 6.24

```
n.boot = 500        # number of bootstrap replicates
tol = sqrt(.Machine$double.eps)  # convergence limit

gnpgr = diff(log(gnp))
fit = arima(gnpgr, order=c(1,0,0))
y = as.matrix(log(resid(fit)^2))
num = length(y)
plot.ts(y, ylab="")

# Initial Parameters
phi1 = .9; sQ = .5; alpha = mean(y); sR0 = 1; mu1 = -3; sR1 = 2.5
init.par = c(phi1, sQ, alpha, sR0, mu1, sR1)

# Innovations Likelihood
Linn=function(para){
  phi1 = para[1]; sQ = para[2]; alpha = para[3]
  sR0 = para[4]; mu1 = para[5]; sR1 = para[6]
  sv = SVfilter(num, y, 0, phi1, sQ, alpha, sR0, mu1, sR1)
  return(sv$like)
  }

# Estimation
(est = optim(init.par, Linn, NULL, method="BFGS", hessian=TRUE, control=list(trace=1,
REPORT=1)))
```

```
SE = sqrt(diag(solve(est$hessian)))
u = cbind(estimates=est$par, SE)
rownames(u)=c("phi1","sQ","alpha","sig0","mu1","sig1"); u

# Bootstrap
para.star = matrix(0, n.boot, 6)  # to store parameter estimates
Linn2 = function(para){
  phi1 = para[1]; sQ = para[2]; alpha = para[3]
  sR0 = para[4]; mu1 = para[5]; sR1 = para[6]
  sv = SVfilter(num, y.star, 0, phi1, sQ, alpha, sR0, mu1, sR1)
  return(sv$like)
  }

for (jb in 1:n.boot){  cat("iteration:", jb, "\n")
 phi1 = est$par[1]; sQ = est$par[2]; alpha = est$par[3]
 sR0 = est$par[4]; mu1 = est$par[5]; sR1 = est$par[6]
 Q = sQ^2; R0 = sR0^2; R1 = sR1^2
 sv = SVfilter(num, y, 0, phi1, sQ, alpha, sR0, mu1, sR1)

 sig0 = sv$Pp+R0
 sig1 = sv$Pp+R1
 K0   = sv$Pp/sig0
 K1   = sv$Pp/sig1
 inn0 = y-sv$xp-alpha; inn1 = y-sv$xp-mu1-alpha
 den1 = (1/sqrt(sig1))*exp(-.5*inn1^2/sig1)
 den0 = (1/sqrt(sig0))*exp(-.5*inn0^2/sig0)
 fpi1 = den1/(den0+den1)

 # (start resampling at t=4)
 e0   = inn0/sqrt(sig0)
 e1   = inn1/sqrt(sig1)
```

```
 indx = sample(4:num, replace=TRUE)
 sinn = cbind(c(e0[1:3], e0[indx]), c(e1[1:3], e1[indx]))
 eF   = matrix(c(phi1, 1, 0, 0), 2, 2)
 xi   = cbind(sv$xp,y)    # initialize

   for (i in 4:num){    # generate boot sample
   G   = matrix(c(0, alpha+fpi1[i]*mu1), 2, 1)
   h21 = (1-fpi1[i])*sqrt(sig0[i]); h11 = h21*K0[i]
   h22 = fpi1[i]*sqrt(sig1[i]); h12 = h22*K1[i]
   H   = matrix(c(h11,h21,h12,h22),2,2)
   xi[i,] = t(eF%*%as.matrix(xi[i-1,],2) + G + H%*%as.matrix(sinn[i,],2))
   }

 # Estimates from boot data
 y.star = xi[,2]
 phi1 = .9; sQ = .5; alpha = mean(y.star); sR0 = 1; mu1 = -3; sR1 = 2.5
 init.par = c(phi1, sQ, alpha, sR0, mu1, sR1)   #  same as for data
 est.star = optim(init.par, Linn2, NULL, method="BFGS", control=list(reltol=tol))
 para.star[jb,] = cbind(est.star$par[1], abs(est.star$par[2]), est.star$par[3], abs
(est.star$par[4]),
                        est.star$par[5], abs(est.star$par[6]))
}

# Some summary statistics and graphics
rmse = rep(NA, 6)  # SEs from the bootstrap
for(i in 1:6){rmse[i] = sqrt(sum((para.star[,i]-est$par[i])^2)/n.boot)
                cat(i, rmse[i],"\n")
             }
dev.new()
phi = para.star[,1]
```

```
hist(phi, 15, prob=TRUE, main="", xlim=c(.4,1.2), xlab="")
 xx = seq(.4, 1.2, by=.01)
lines(xx, dnorm(xx, mean=u[1,1], sd=u[2,1]), lty=2, lwd=2)
```

Example 6.26

```
###########################################################################################
# Adapted from code by: Hedibert Freitas Lopes
###########################################################################################
##-- Notation --##
#           y(t) = x(t) + v(t);    v(t) ~ iid N(0,V)
#           x(t) = x(t-1) + w(t);  w(t) ~ iid N(0,W)
#  priors:  x(0) ~ N(m0,C0);  V ~ IG(a,b);  W ~ IG(c,d)
#    FFBS:  x(t|t) ~ N(m,C);  x(t|n) ~ N(mm,CC);  x(t|t+1) ~ N(a,R)
##--
ffbs = function(y,V,W,m0,C0){
  n  = length(y);  a  = rep(0,n);  R  = rep(0,n)
  m  = rep(0,n);   C  = rep(0,n);  B  = rep(0,n-1)
  H  = rep(0,n-1); mm = rep(0,n);  CC = rep(0,n)
  x  = rep(0,n); llike = 0.0
  for (t in 1:n){
    if(t==1){a[1] = m0; R[1] = C0 + W
      }else{ a[t] = m[t-1]; R[t] = C[t-1] + W }
    f      = a[t]
    Q      = R[t] + V
    A      = R[t]/Q
    m[t]   = a[t]+A*(y[t]-f)
    C[t]   = R[t]-Q*A**2
    B[t-1] = C[t-1]/R[t]
    H[t-1] = C[t-1]-R[t]*B[t-1]**2
    llike  = llike + dnorm(y[t],f,sqrt(Q),log=TRUE) }
  mm[n] = m[n]; CC[n] = C[n]
```

```
  x[n]  = rnorm(1,m[n],sqrt(C[n]))
  for (t in (n-1):1){
    mm[t] = m[t] + C[t]/R[t+1]*(mm[t+1]-a[t+1])
    CC[t] = C[t] - (C[t]^2)/(R[t+1]^2)*(R[t+1]-CC[t+1])
    x[t]  = rnorm(1,m[t]+B[t]*(x[t+1]-a[t+1]),sqrt(H[t]))  }
return(list(x=x,m=m,C=C,mm=mm,CC=CC,llike=llike))   }
# Simulate states and data
set.seed(1); W = 0.5; V = 1.0
n  = 100; m0 = 0.0; C0 = 10.0; x0 = 0
w  = rnorm(n,0,sqrt(W))
v  = rnorm(n,0,sqrt(V))
x  = y = rep(0,n)
x[1] = x0   + w[1]
y[1] = x[1] + v[1]
for (t in 2:n){
  x[t] = x[t-1] + w[t]
  y[t] = x[t] + v[t]   }
# actual smoother (for plotting)
ks = Ksmooth0(num=n, y, A=1, m0, C0, Phi=1, cQ=sqrt(W), cR=sqrt(V))
xsmooth = as.vector(ks$xs)
#
run = ffbs(y,V,W,m0,C0)
m   = run$m; C = run$C; mm = run$mm
CC  = run$CC; L1 = m-2*C; U1  = m+2*C
L2  = mm-2*CC; U2 = mm+2*CC
N   = 50
Vs  = seq(0.1,2,length=N)
Ws  = seq(0.1,2,length=N)
likes = matrix(0,N,N)
for (i in 1:N){
 for (j in 1:N){
```

```
   V   = Vs[i]
   W   = Ws[j]
   run = ffbs(y,V,W,m0,C0)
  likes[i,j] = run$llike  }  }
# Hyperparameters
a = 0.01; b = 0.01; c = 0.01; d = 0.01
# MCMC step
set.seed(90210)
burn  = 10;  M = 1000
niter = burn + M
V1    = V;  W1 = W
draws = NULL
all_draws = NULL
for (iter in 1:niter){
  run   = ffbs(y,V1,W1,m0,C0)
  x     = run$x
  V1    = 1/rgamma(1,a+n/2,b+sum((y-x)^2)/2)
  W1    = 1/rgamma(1,c+(n-1)/2,d+sum(diff(x)^2)/2)
  draws = rbind(draws,c(V1,W1,x))    }
all_draws = draws[,1:2]
q025  = function(x){quantile(x,0.025)}
q975  = function(x){quantile(x,0.975)}
draws = draws[(burn+1):(niter),]
xs    = draws[,3:(n+2)]
lx    = apply(xs,2,q025)
mx    = apply(xs,2,mean)
ux    = apply(xs,2,q975)
##  plot of the data
par(mfrow=c(2,2), mgp=c(1.6,.6,0), mar=c(3,3.2,1,1))
ts.plot(ts(x), ts(y), ylab='', col=c(1,8), lwd=2)
```

```
points(y)
legend(0, 11, legend=c("x(t)","y(t)"), lty=1, col=c(1,8), lwd=2, bty="n", pch=c(-1,1))
contour(Vs, Ws, exp(likes), xlab=expression(sigma[v]^2), ylab=expression(sigma[w]^2),
         drawlabels=FALSE, ylim=c(0,1.2))
points(draws[,1:2], pch=16, col=rgb(.9,0,0,0.3), cex=.7)
hist(draws[,1], ylab="Density",main="", xlab=expression(sigma[v]^2))
abline(v=mean(draws[,1]), col=3, lwd=3)
hist(draws[,2],main="", ylab="Density", xlab=expression(sigma[w]^2))
abline(v=mean(draws[,2]), col=3, lwd=3)
## plot states
par(mgp=c(1.6,.6,0), mar=c(2,1,.5,0)+.5)
plot(ts(mx), ylab='', type='n', ylim=c(min(y),max(y)))
grid(lty=2); points(y)
lines(xsmooth, lwd=4, col=rgb(1,0,1,alpha=.4))
lines(mx, col= 4)
 xx=c(1:100, 100:1)
 yy=c(lx, rev(ux))
polygon(xx, yy, border=NA, col= gray(.6,alpha=.2))
lines(y, col=gray(.4))
legend('topleft', c('true smoother', 'data', 'posterior mean', '95% of draws'),
lty=1,
         lwd=c(3,1,1,10), pch=c(-1,1,-1,-1), col=c(6, gray(.4), 4, gray(.6,
alpha=.5)),
         bg='white' )
```

Example 6.27

```
library(plyr)   # used to view progress (install it if you don't have it)
y = jj
### setup - model and initial parameters
set.seed(90210)
```

```
n = length(y)
F = c(1,1,0,0)      # this is A
G = diag(0,4)       # G is Phi
  G[1,1] = 1.03
  G[2,]  = c(0,-1,-1,-1); G[3,]=c(0,1,0,0); G[4,]=c(0,0,1,0)
a1 = rbind(.7,0,0,0)  # this is mu0
R1 = diag(.04,4)      # this is Sigma0
V = .1
W11 = .1
W22 = .1
##-- FFBS --##
ffbs = function(y,F,G,V,W11,W22,a1,R1){
  n  = length(y)
  Ws = diag(c(W11,W22,1,1))  # this is Q with 1s as a device only
  iW = diag(1/diag(Ws),4)
  a  = matrix(0,n,4)         # this is m_t
  R  = array(0,c(n,4,4))     # this is V_t
  m  = matrix(0,n,4)
  C  = array(0,c(n,4,4))
  a[1,]  = a1[,1]
  R[1,,] = R1
  f      = t(F)%*%a[1,]
  Q      = t(F)%*%R[1,,]%*%F + V
  A      = R[1,,]%*%F/Q[1,1]
  m[1,]  = a[1,]+A%*%(y[1]-f)
  C[1,,] = R[1,,]-A%*%t(A)*Q[1,1]
  for (t in 2:n){
    a[t,]  = G%*%m[t-1,]
    R[t,,] = G%*%C[t-1,,]%*%t(G) + Ws
    f      = t(F)%*%a[t,]
    Q      = t(F)%*%R[t,,]%*%F + V
```

```
    A      = R[t,,]%*%F/Q[1,1]
    m[t,]  = a[t,] + A%*%(y[t]-f)
    C[t,,] = R[t,,] - A%*%t(A)*Q[1,1]      }
  xb       = matrix(0,n,4)
  xb[n,]  = m[n,] + t(chol(C[n,,]))%*%rnorm(4)
  for (t in (n-1):1){
    iC  = solve(C[t,,])
    CCC = solve(t(G)%*%iW%*%G + iC)
    mmm = CCC%*%(t(G)%*%iW%*%xb[t+1,] + iC%*%m[t,])
    xb[t,] = mmm + t(chol(CCC))%*%rnorm(4)  }
  return(xb)                                }
##-- Prior hyperparameters --##
# b0 = 0    # mean for beta = phi -1
# B0 = Inf  # var for  beta  (non-informative => use OLS for sampling beta)
n0 = 10  # use same for all- the prior is 1/Gamma(n0/2, n0*s20_/2)
s20v = .001  # for V
s20w =.05    # for Ws
##-- MCMC scheme --##
set.seed(90210)
burnin  = 100
step    = 10
M       = 1000
niter   = burnin+step*M
pars    = matrix(0,niter,4)
xbs     = array(0,c(niter,n,4))
pr <- progress_text()           # displays progress
pr$init(niter)
for (iter in 1:niter){
    xb = ffbs(y,F,G,V,W11,W22,a1,R1)
     u = xb[,1]
```

```
    yu = diff(u); xu = u[-n]    # for phihat and se(phihat)
  regu = lm(yu~0+xu)                 # est of beta = phi-1
 phies = as.vector(coef(summary(regu)))[1:2] + c(1,0) # phi estimate and SE
   dft = df.residual(regu)
G[1,1] = phies[1] + rt(1,dft)*phies[2]  # use a t
    V  = 1/rgamma(1, (n0+n)/2, (n0*s20v/2) + sum((y-xb[,1]-xb[,2])^2)/2)
   W11 = 1/rgamma(1, (n0+n-1)/2, (n0*s20w/2) + sum((xb[-1,1]-phies[1]*xb[-n,1])^2)/2)
   W22 = 1/rgamma(1, (n0+ n-3)/2, (n0*s20w/2) + sum((xb[4:n,2] + xb[3:(n-1),2] +
                  xb[2:(n-2),2] +xb[1:(n-3),2])^2)/2)
   xbs[iter,,] = xb
   pars[iter,] = c(G[1,1], sqrt(V), sqrt(W11), sqrt(W22))
   pr$step()             }
# Plot results
ind = seq(burnin+1,niter,by=step)
names= c(expression(phi), expression(sigma[v]), expression(sigma[w~11]), expression
(sigma[w~22]))
dev.new(height=5)
par(mfcol=c(3,4), mar=c(2,2,.25,0)+.75, mgp=c(1.6,.6,0), oma=c(0,0,1,0))
for (i in 1:4){
 plot.ts(pars[ind,i],xlab="iterations", ylab="trace", main="")
 mtext(names[i], side=3, line=.5, cex=1)
 acf(pars[ind,i],main="", lag.max=25, xlim=c(1,25), ylim=c(-.4,.4))
 hist(pars[ind,i],main="",xlab="")
 abline(v=mean(pars[ind,i]), lwd=2, col=3)  }
 par(mfrow=c(2,1), mar=c(2,2,0,0)+.7, mgp=c(1.6,.6,0))
  mxb = cbind(apply(xbs[ind,,1],2,mean), apply(xbs[,,2],2,mean))
  lxb = cbind(apply(xbs[ind,,1],2,quantile,0.005), apply(xbs[ind,,2],2,
quantile,0.005))
  uxb = cbind(apply(xbs[ind,,1],2,quantile,0.995), apply(xbs[ind,,2],2,
quantile,0.995))
  mxb = ts(cbind(mxb,rowSums(mxb)), start = tsp(jj)[1], freq=4)
```

```
  lxb = ts(cbind(lxb,rowSums(lxb)), start = tsp(jj)[1], freq=4)
  uxb = ts(cbind(uxb,rowSums(uxb)), start = tsp(jj)[1], freq=4)
  names=c('Trend', 'Season', 'Trend + Season')
  L = min(lxb[,1])-.01; U = max(uxb[,1]) +.01
plot(mxb[,1],  ylab=names[1], ylim=c(L,U), type='n')
  grid(lty=2); lines(mxb[,1])
  xx=c(time(jj), rev(time(jj)))
  yy=c(lxb[,1], rev(uxb[,1]))
  polygon(xx, yy, border=NA, col=gray(.4, alpha = .2))
  L = min(lxb[,3])-.01; U = max(uxb[,3]) +.01
plot(mxb[,3],  ylab=names[3], ylim=c(L,U), type='n')
  grid(lty=2); lines(mxb[,3])
  xx=c(time(jj), rev(time(jj)))
  yy=c(lxb[,3], rev(uxb[,3]))
  polygon(xx, yy, border=NA, col=gray(.4, alpha = .2))
```

[-]

[+] Chapter 7

Code in Introduction

```
x = matrix(0, 128, 6)
for (i in 1:6) x[,i] = rowMeans(fmri[[i]])
colnames(x)=c("Brush", "Heat", "Shock", "Brush", "Heat", "Shock")
plot.ts(x, main="")
mtext("Awake", side=3, line=1.2, adj=.05,cex=1.2)
mtext("Sedated", side=3, line=1.2, adj=.85, cex=1.2)


attach(eqexp)
P = 1:1024; S = P+1024
x = cbind(EQ5[P], EQ6[P], EX5[P], EX6[P], NZ[P], EQ5[S], EQ6[S], EX5[S], EX6[S], NZ
```

```
[S])
x.name = c("EQ5","EQ6","EX5","EX6","NZ")
colnames(x) = c(x.name, x.name)
plot.ts(x, main="")
mtext("P waves", side=3, line=1.2, adj=.05, cex=1.2)
mtext("S waves", side=3, line=1.2, adj=.85, cex=1.2)
```

Example 7.1

```
plot.ts(climhyd)    # figure 7.3
Y = climhyd         # Y holds the transformed series
Y[,6] = log(Y[,6])  # log inflow
Y[,5] = sqrt(Y[,5]) # sqrt precipitation

L = 25              # setup
M = 100
alpha = .001
fdr = .001
nq = 2              # number of inputs  (Temp and Precip)

# Spectral Matrix
Yspec = mvspec(Y, spans=L, kernel="daniell", taper=.1, plot=FALSE)
n = Yspec$n.used          # effective sample size
Fr = Yspec$freq           # fundamental freqs
n.freq = length(Fr)       # number of frequencies
Yspec$bandwidth           # = 0.05

# Coherencies (see section 4.7 also)
Fq = qf(1-alpha, 2, L-2); cn = Fq/(L-1+Fq)
plt.name = c("(a)","(b)","(c)","(d)","(e)","(f)")
dev.new()
```

```
par(mfrow=c(2,3), cex.lab=1.2)

# The coherencies are listed as 1,2,...,15=choose(6,2)
for (i in 11:15){
 plot(Fr,Yspec$coh[,i], type="l", ylab="Sq Coherence", xlab="Frequency", ylim=c(0,1),
      main=c("Inflow with", names(climhyd[i-10])))
 abline(h = cn); text(.45,.98, plt.name[i-10], cex=1.2)
}

 # Multiple Coherency
coh.15 = stoch.reg(Y, cols.full = c(1,5), cols.red = NULL, alpha, L, M, plot.which =
"coh")
text(.45,.98, plt.name[6], cex=1.2)
title(main = c("Inflow with", "Temp and Precip"))

# Partial F (note F-stat is called eF in the code)
numer.df = 2*nq
denom.df = Yspec$df-2*nq

dev.new()
par(mfrow=c(3,1), mar=c(3,3,2,1)+.5, mgp = c(1.5,0.4,0), cex.lab=1.2)
out.15 = stoch.reg(Y, cols.full = c(1,5), cols.red = 5, alpha, L, M, plot.which = "F.
stat")
 eF = out.15$eF
 pvals = pf(eF, numer.df, denom.df, lower.tail = FALSE)
 pID = FDR(pvals, fdr)
abline(h=c(eF[pID]), lty=2)
title(main = "Partial F Statistic")

# Regression Coefficients
S = seq(from = -M/2+1, to = M/2 - 1, length = M-1)
```

```
plot(S, coh.15$Betahat[,1], type = "h", xlab = "", ylab =names(climhyd[1]),
        ylim = c(-.025, .055), lwd=2)
abline(h=0)
title(main = "Impulse Response Functions")

plot(S, coh.15$Betahat[,2], type = "h", xlab = "Index", ylab = names(climhyd[5]),
        ylim = c(-.015, .055), lwd=2)
abline(h=0)
```

Example 7.2

```
attach(beamd)
tau    = rep(0,3)
     u = ccf(sensor1, sensor2, plot=FALSE)
tau[1] = u$lag[which.max(u$acf)]    #  17
     u = ccf(sensor3, sensor2, plot=FALSE)
tau[3] = u$lag[which.max(u$acf)]    # -22

Y     = ts.union(lag(sensor1,tau[1]), lag(sensor2, tau[2]), lag(sensor3, tau[3]))
Y     = ts.union(Y, rowMeans(Y))
Time  = time(Y)
par(mfrow=c(4,1), mar=c(0, 3.1, 0, 1.1), oma=c(2.75, 0, 2.5, 0), mgp=c(1.6,.6,0))
plot(Time, Y[,1], ylab='sensor1', xaxt="no", type='n')
grid(); lines(Y[,1])
title(main="Infrasonic Signals and Beam", outer=TRUE)
plot(Time, Y[,2], ylab='sensor2', xaxt="no",  type='n')
grid(); lines(Y[,2])
plot(Time,Y[,3], ylab='sensor3', xaxt="no", type='n')
grid(); lines(Y[,3])
plot(Time, beam, type='n')
```

```
grid(); lines(Y[,4])
title(xlab="Time", outer=TRUE)
```

Example 7.5

```
n           = 128               # length of series
n.freq      = 1 + n/2           # number of frequencies
Fr          = (0:(n.freq-1))/n  # the frequencies
N           = c(5,4,5,3,5,4)    # number of series for each cell
n.subject   = sum(N)            # number of subjects (26)
n.trt       = 6                 # number of treatments
L           = 3                 # for smoothing
num.df      = 2*L*(n.trt-1)     # dfs for F test
den.df      = 2*L*(n.subject-n.trt)


# Design Matrix (Z):
Z1 = outer(rep(1,N[1]), c(1,1,0,0,0,0))
Z2 = outer(rep(1,N[2]), c(1,0,1,0,0,0))
Z3 = outer(rep(1,N[3]), c(1,0,0,1,0,0))
Z4 = outer(rep(1,N[4]), c(1,0,0,0,1,0))
Z5 = outer(rep(1,N[5]), c(1,0,0,0,0,1))
Z6 = outer(rep(1,N[6]), c(1,-1,-1,-1,-1,-1))

Z  = rbind(Z1, Z2, Z3, Z4, Z5, Z6)
ZZ = t(Z)%*%Z

SSEF <- rep(NA, n) -> SSER

HatF = Z%*%solve(ZZ, t(Z))
HatR = Z[,1]%*%t(Z[,1])/ZZ[1,1]
```

```
par(mfrow=c(3,3), mar=c(3.5,4,0,0), oma=c(0,0,2,2), mgp = c(1.6,.6,0))
loc.name = c("Cortex 1","Cortex 2","Cortex 3","Cortex 4","Caudate","Thalamus 
1","Thalamus 2",
             "Cerebellum 1","Cerebellum 2")

for(Loc in 1:9) {
 i = n.trt*(Loc-1)
 Y = cbind(fmri[[i+1]], fmri[[i+2]], fmri[[i+3]], fmri[[i+4]], fmri[[i+5]], fmri[[i
+6]])
 Y = mvfft(spec.taper(Y, p=.5))/sqrt(n)
 Y = t(Y)      # Y is now 26 x 128 FFTs

 # Calculation of Error Spectra
 for (k in 1:n) {
  SSY = Re(Conj(t(Y[,k]))%*%Y[,k])
  SSReg = Re(Conj(t(Y[,k]))%*%HatF%*%Y[,k])
 SSEF[k] = SSY - SSReg
  SSReg = Re(Conj(t(Y[,k]))%*%HatR%*%Y[,k])
 SSER[k] = SSY - SSReg
 }

# Smooth
sSSEF = filter(SSEF, rep(1/L, L), circular = TRUE)
sSSER = filter(SSER, rep(1/L, L), circular = TRUE)

eF =(den.df/num.df)*(sSSER-sSSEF)/sSSEF

plot(Fr, eF[1:n.freq], type="l", xlab="Frequency", ylab="F Statistic", ylim=c(0,7))
abline(h=qf(.999, num.df, den.df),lty=2)
text(.25, 6.5, loc.name[Loc], cex=1.2)
```

```
}
```

Example 7.6

```
n      = 128
n.freq = 1 + n/2
Fr     = (0:(n.freq-1))/n
nFr    = 1:(n.freq/2)
N      = c(5,4,5,3,5,4)
n.para = 6          # number of parameters
n.subject = sum(N)  # total number of subjects

L = 3
df.stm = 2*L*(3-1)              # stimulus (3 levels: Brush,Heat,Shock)
df.con = 2*L*(2-1)              # conscious (2 levels: Awake,Sedated)
df.int = 2*L*(3-1)*(2-1)        # interaction
den.df = 2*L*(n.subject-n.para) # df for full model

# Design Matrix:          mu  a1  a2   b  g1  g2
 Z1 = outer(rep(1,N[1]), c(1,  1,  0,  1,  1,  0))
 Z2 = outer(rep(1,N[2]), c(1,  0,  1,  1,  0,  1))
 Z3 = outer(rep(1,N[3]), c(1, -1, -1,  1, -1, -1))
 Z4 = outer(rep(1,N[4]), c(1,  1,  0, -1, -1,  0))
 Z5 = outer(rep(1,N[5]), c(1,  0,  1, -1,  0, -1))
 Z6 = outer(rep(1,N[6]), c(1, -1, -1, -1,  1,  1))

 Z = rbind(Z1, Z2, Z3, Z4, Z5, Z6)
 ZZ = t(Z)%*%Z

rep(NA, n)-> SSEF -> SSE.stm -> SSE.con -> SSE.int
HatF    = Z%*%solve(ZZ,t(Z))
```

```
Hat.stm = Z[,-(2:3)]%*%solve(ZZ[-(2:3),-(2:3)], t(Z[,-(2:3)]))
Hat.con = Z[,-4]%*%solve(ZZ[-4,-4], t(Z[,-4]))
Hat.int = Z[,-(5:6)]%*%solve(ZZ[-(5:6),-(5:6)], t(Z[,-
(5:6)]))

par(mfrow=c(5,3),mar=c(3.5,4,0,0),oma=c(0,0,2,2),mgp = c(1.6,.6,0))
loc.name = c("Cortex 1","Cortex 2","Cortex 3","Cortex 4","Caudate", "Thalamus
1","Thalamus 2",
             "Cerebellum 1","Cerebellum 2")
for(Loc in c(1:4,9)) {   # only Loc 1 to 4 and 9 used
  i = 6*(Loc-1)
  Y = cbind(fmri[[i+1]], fmri[[i+2]], fmri[[i+3]], fmri[[i+4]], fmri[[i+5]], fmri[[i
+6]])
  Y = mvfft(spec.taper(Y, p=.5))/sqrt(n)
  Y = t(Y)
  for (k in 1:n) {
    SSY=Re(Conj(t(Y[,k]))%*%Y[,k])
    SSReg= Re(Conj(t(Y[,k]))%*%HatF%*%Y[,k])
  SSEF[k]=SSY-SSReg
    SSReg=Re(Conj(t(Y[,k]))%*%Hat.stm%*%Y[,k])
  SSE.stm[k] = SSY-SSReg
    SSReg=Re(Conj(t(Y[,k]))%*%Hat.con%*%Y[,k])
  SSE.con[k]=SSY-SSReg
    SSReg=Re(Conj(t(Y[,k]))%*%Hat.int%*%Y[,k])
  SSE.int[k]=SSY-SSReg
  }
 # Smooth
  sSSEF    = filter(SSEF, rep(1/L, L), circular = TRUE)
  sSSE.stm = filter(SSE.stm, rep(1/L, L), circular = TRUE)
  sSSE.con = filter(SSE.con, rep(1/L, L), circular = TRUE)
  sSSE.int = filter(SSE.int, rep(1/L, L), circular = TRUE)
```

```
  eF.stm   = (den.df/df.stm)*(sSSE.stm-sSSEF)/sSSEF
  eF.con   = (den.df/df.con)*(sSSE.con-sSSEF)/sSSEF
  eF.int   = (den.df/df.int)*(sSSE.int-sSSEF)/sSSEF

 plot(Fr[nFr],eF.stm[nFr], type="l", xlab="Frequency", ylab="F Statistic", ylim=c
(0,12))
   abline(h=qf(.999, df.stm, den.df),lty=2)
  if(Loc==1) mtext("Stimulus", side=3, line=.3, cex=1)
  mtext(loc.name[Loc], side=2, line=3, cex=.9)
 plot(Fr[nFr],eF.con[nFr], type="l", xlab="Frequency", ylab="F Statistic", ylim=c
(0,12))
  abline(h=qf(.999, df.con, den.df),lty=2)
  if(Loc==1)  mtext("Consciousness", side=3, line=.3, cex=1)
 plot(Fr[nFr],eF.int[nFr], type="l", xlab="Frequency",ylab="F Statistic", ylim=c
(0,12))
  abline(h=qf(.999, df.int, den.df),lty=2)
  if(Loc==1) mtext("Interaction", side=3, line= .3, cex=1)
}
```

Example 7.7

```
n      = 128
n.freq = 1 + n/2
Fr     = (0:(n.freq-1))/n
nFr    = 1:(n.freq/2)
N      = c(5,4,5,3,5,4)
L      = 3
n.subject = sum(N)

# Design Matrix
Z1 = outer(rep(1,N[1]), c(1,0,0,0,0,0))
```

```
Z2 = outer(rep(1,N[2]), c(0,1,0,0,0,0))

Z3 = outer(rep(1,N[3]), c(0,0,1,0,0,0))

Z4 = outer(rep(1,N[4]), c(0,0,0,1,0,0))

Z5 = outer(rep(1,N[5]), c(0,0,0,0,1,0))

Z6 = outer(rep(1,N[6]), c(0,0,0,0,0,1))

Z  = rbind(Z1, Z2, Z3, Z4, Z5, Z6)

ZZ = t(Z)%*%Z


A      = rbind(diag(1,3), diag(1,3))   # Contrasts:  6 x 3

nq     = nrow(A)

num.df = 2*L*nq

den.df = 2*L*(n.subject-nq)

HatF   = Z%*%solve(ZZ, t(Z))           # full model hat matrix


rep(NA, n)-> SSEF -> SSER

eF = matrix(0,n,3)


par(mfrow=c(5,3), mar=c(3.5,4,0,0), oma=c(0,0,2,2), mgp = c(1.6,.6,0))


loc.name = c("Cortex 1","Cortex 2","Cortex 3","Cortex 4","Caudate","Thalamus
1","Thalamus 2",
             "Cerebellum 1","Cerebellum 2")

cond.name = c("Brush", "Heat", "Shock")


for(Loc in c(1:4,9)) {

 i = 6*(Loc-1)

 Y = cbind(fmri[[i+1]],fmri[[i+2]],fmri[[i+3]],fmri[[i+4]], fmri[[i+5]],fmri[[i
+6]])

 Y = mvfft(spec.taper(Y, p=.5))/sqrt(n); Y = t(Y)

 for (cond in 1:3){

  Q = t(A[,cond])%*%solve(ZZ, A[,cond])
```

```
  HR = A[,cond]%*%solve(ZZ, t(Z))
  for (k in 1:n){
    SSY = Re(Conj(t(Y[,k]))%*%Y[,k])
    SSReg= Re(Conj(t(Y[,k]))%*%HatF%*%Y[,k])
   SSEF[k]= (SSY-SSReg)*Q
    SSReg= HR%*%Y[,k]
   SSER[k] = Re(SSReg*Conj(SSReg))
  }

# Smooth
 sSSEF = filter(SSEF, rep(1/L, L), circular = TRUE)
 sSSER = filter(SSER, rep(1/L, L), circular = TRUE)
 eF[,cond]= (den.df/num.df)*(sSSER/sSSEF)   }
 plot(Fr[nFr], eF[nFr,1], type="l", xlab="Frequency", ylab="F Statistic", ylim=c(0,5))
  abline(h=qf(.999, num.df, den.df),lty=2)
  if(Loc==1) mtext("Brush", side=3, line=.3, cex=1)
  mtext(loc.name[Loc], side=2, line=3, cex=.9)
 plot(Fr[nFr], eF[nFr,2], type="l", xlab="Frequency", ylab="F Statistic", ylim=c(0,5))
  abline(h=qf(.999, num.df, den.df),lty=2)
  if(Loc==1)  mtext("Heat", side=3, line=.3, cex=1)
 plot(Fr[nFr], eF[nFr,3], type="l", xlab="Frequency", ylab="F Statistic", ylim=c(0,5))
  abline(h = qf(.999, num.df, den.df) ,lty=2)
  if(Loc==1) mtext("Shock", side=3, line=.3, cex=1)
}
```

Example 7.8

```
P = 1:1024
S = P+1024
N = 8
n = 1024
```

```
p.dim = 2
m = 10
L = 2*m+1

eq.P   = as.ts(eqexp[P,1:8])
eq.S   = as.ts(eqexp[S,1:8])
eq.m   = cbind(rowMeans(eq.P), rowMeans(eq.S))
ex.P   = as.ts(eqexp[P,9:16])
ex.S   = as.ts(eqexp[S,9:16])
ex.m   = cbind(rowMeans(ex.P), rowMeans(ex.S))
m.diff = mvfft(eq.m - ex.m)/sqrt(n)

eq.Pf = mvfft(eq.P-eq.m[,1])/sqrt(n)
eq.Sf = mvfft(eq.S-eq.m[,2])/sqrt(n)
ex.Pf = mvfft(ex.P-ex.m[,1])/sqrt(n)
ex.Sf = mvfft(ex.S-ex.m[,2])/sqrt(n)

fv11 = rowSums(eq.Pf*Conj(eq.Pf)) + rowSums(ex.Pf*Conj(ex.Pf))/(2*(N-1))
fv12 = rowSums(eq.Pf*Conj(eq.Sf)) + rowSums(ex.Pf*Conj(ex.Sf))/(2*(N-1))
fv22 = rowSums(eq.Sf*Conj(eq.Sf)) + rowSums(ex.Sf*Conj(ex.Sf))/(2*(N-1))
fv21 = Conj(fv12)

# Equal Means
T2 = rep(NA, 512)
for (k  in 1:512){
 fvk = matrix(c(fv11[k], fv21[k], fv12[k], fv22[k]), 2, 2)
 dk = as.matrix(m.diff[k,])
 T2[k] = Re((N/2)*Conj(t(dk))%*%solve(fvk,dk))  }
eF = T2*(2*p.dim*(N-1))/(2*N-p.dim-1)
par(mfrow=c(2,2), mar=c(3,3,2,1), mgp = c(1.6,.6,0), cex.main=1.1)
freq = 40*(0:511)/n  # in Hz (cycles per second)
```

```
plot(freq, eF, type="l", xlab="Frequency (Hz)", ylab="F Statistic", main="Equal 
Means")
abline(h=qf(.999, 2*p.dim, 2*(2*N-p.dim-1)))

# Equal P
kd    = kernel("daniell",m);
u     = Re(rowSums(eq.Pf*Conj(eq.Pf))/(N-1))
feq.P = kernapply(u, kd, circular=TRUE)
u     = Re(rowSums(ex.Pf*Conj(ex.Pf))/(N-1))
fex.P = kernapply(u, kd, circular=TRUE)

plot(freq, feq.P[1:512]/fex.P[1:512], type="l", xlab="Frequency (Hz)", ylab="F 
Statistic",
      main="Equal P-Spectra")
abline(h = qf(.999, 2*L*(N-1),  2*L*(N-1)))

# Equal S
u     = Re(rowSums(eq.Sf*Conj(eq.Sf))/(N-1))
feq.S = kernapply(u, kd, circular=TRUE)
u     = Re(rowSums(ex.Sf*Conj(ex.Sf))/(N-1))
fex.S = kernapply(u, kd, circular=TRUE)

plot(freq, feq.S[1:512]/fex.S[1:512], type="l", xlab="Frequency (Hz)", ylab="F 
Statistic",
      main="Equal S-Spectra")
abline(h=qf(.999, 2*L*(N-1),  2*L*(N-1)))

# Equal Spectra
u      = rowSums(eq.Pf*Conj(eq.Sf))/(N-1)
feq.PS = kernapply(u, kd, circular=TRUE)
```

```
u      = rowSums(ex.Pf*Conj(ex.Sf)/(N-1))
fex.PS = kernapply(u, kd, circular=TRUE)
fv11   = kernapply(fv11, kd, circular=TRUE)
fv22   = kernapply(fv22, kd, circular=TRUE)
fv12  = kernapply(fv12, kd, circular=TRUE)

Mi = L*(N-1)
M  = 2*Mi
TS = rep(NA,512)

for (k  in 1:512){
 det.feq.k = Re(feq.P[k]*feq.S[k] - feq.PS[k]*Conj(feq.PS[k]))
 det.fex.k = Re(fex.P[k]*fex.S[k] - fex.PS[k]*Conj(fex.PS[k]))
 det.fv.k  = Re(fv11[k]*fv22[k] - fv12[k]*Conj(fv12[k]))

 log.n1 = log(M)*(M*p.dim)
 log.d1 = log(Mi)*(2*Mi*p.dim)
 log.n2 = log(Mi)*2 +log(det.feq.k)*Mi + log(det.fex.k)*Mi
 log.d2 = (log(M)+log(det.fv.k))*M
 r = 1 - ((p.dim+1)*(p.dim-1)/6*p.dim*(2-1))*(2/Mi - 1/M)
 TS[k] = -2*r*(log.n1+log.n2-log.d1-log.d2)
}

plot(freq, TS, type="l", xlab="Frequency (Hz)", ylab="Chi-Sq Statistic", main="Equal
Spectral Matrices")
abline(h = qchisq(.9999, p.dim^2))
```

Example 7.9

```
P     = 1:1024
S     = P+1024
mag.P = log10(apply(eqexp[P,],2,max) - apply(eqexp[P,],2,min))
```

```
mag.S = log10(apply(eqexp[S,],2,max) - apply(eqexp[S,],2,min))
eq.P  = mag.P[1:8]
eq.S  = mag.S[1:8]
ex.P  = mag.P[9:16]
ex.S =  mag.S[9:16]
NZ.P  = mag.P[17]
NZ.S  = mag.S[17]

# Compute linear discriminant function
cov.eq     = var(cbind(eq.P, eq.S))
cov.ex     = var(cbind(ex.P, ex.S))
cov.pooled = (cov.ex + cov.eq)/2

means.eq   =  colMeans(cbind(eq.P,eq.S));
means.ex   =  colMeans(cbind(ex.P,ex.S))
slopes.eq  = solve(cov.pooled, means.eq)
inter.eq   = -sum(slopes.eq*means.eq)/2
slopes.ex  = solve(cov.pooled, means.ex)
inter.ex   = -sum(slopes.ex*means.ex)/2
d.slopes   = slopes.eq - slopes.ex
d.inter    = inter.eq - inter.ex

# Classify new observation
new.data = cbind(NZ.P, NZ.S)

d = sum(d.slopes*new.data) + d.inter
post.eq = exp(d)/(1+exp(d))

# Print (disc function, posteriors) and plot results
cat(d.slopes[1], "mag.P +" , d.slopes[2], "mag.S +" , d.inter,"\n")
cat("P(EQ|data) =", post.eq,  "  P(EX|data) =", 1-post.eq, "\n" )
```

```
plot(eq.P, eq.S, xlim=c(0,1.5), ylim=c(.75,1.25), xlab="log mag(P)", ylab ="log mag
(S)", pch = 8,
      cex=1.1, lwd=2, main="Classification Based on Magnitude Features")
 points(ex.P, ex.S, pch = 6, cex=1.1, lwd=2)
 points(new.data, pch = 3, cex=1.1, lwd=2)
 abline(a = -d.inter/d.slopes[2], b = -d.slopes[1]/d.slopes[2])
 text(eq.P-.07,eq.S+.005, label=names(eqexp[1:8]), cex=.8)
 text(ex.P+.07,ex.S+.003, label=names(eqexp[9:16]), cex=.8)
 text(NZ.P+.05,NZ.S+.003, label=names(eqexp[17]), cex=.8)
 legend("topright",c("EQ","EX","NZ"),pch=c(8,6,3),pt.lwd=2,cex=1.1)

# Cross-validation
all.data = rbind(cbind(eq.P,eq.S), cbind(ex.P,ex.S))
post.eq <- rep(NA, 8) -> post.ex

for(j in 1:16) {
 if (j <= 8) {samp.eq = all.data[-c(j, 9:16),];  samp.ex = all.data[9:16,]}
 if (j > 8)  {samp.eq = all.data[1:8,];    samp.ex = all.data[-c(j, 1:8),]}

 df.eq      = nrow(samp.eq)-1;  df.ex = nrow(samp.ex)-1
 mean.eq    = colMeans(samp.eq);  mean.ex = colMeans(samp.ex)
 cov.eq     = var(samp.eq);  cov.ex = var(samp.ex)
 cov.pooled = (df.eq*cov.eq + df.ex*cov.ex)/(df.eq + df.ex)
 slopes.eq  = solve(cov.pooled, mean.eq)
 inter.eq   = -sum(slopes.eq*mean.eq)/2
 slopes.ex  = solve(cov.pooled, mean.ex)
 inter.ex   = -sum(slopes.ex*mean.ex)/2
 d.slopes   = slopes.eq - slopes.ex
 d.inter    = inter.eq - inter.ex
```

```
 d = sum(d.slopes*all.data[j,]) + d.inter
 if (j <= 8) post.eq[j] = exp(d)/(1+exp(d))
 if (j > 8) post.ex[j-8] = 1/(1+exp(d))
}

Posterior = cbind(1:8, post.eq, 1:8, post.ex)
colnames(Posterior) = c("EQ","P(EQ|data)","EX","P(EX|data)")
# results from cross-validation
round(Posterior, 3)
```

Example 7.10

```
P = 1:1024
S = P+1024
p.dim = 2
n =1024

eq = as.ts(eqexp[,1:8])
ex = as.ts(eqexp[,9:16])
nz = as.ts(eqexp[,17])
f.eq <- array(dim=c(8,2,2,512)) -> f.ex
f.NZ = array(dim=c(2,2,512))

# below calculates determinant for 2x2 Hermitian matrix
det.c = function(mat){return(Re(mat[1,1]*mat[2,2]-mat[1,2]*mat[2,1]))}
L = c(15,13,5)  # for smoothing
for (i in 1:8){     # compute spectral matrices
 f.eq[i,,,] = mvspec(cbind(eq[P,i],eq[S,i]), spans=L, taper=.5, plot=FALSE)$fxx
 f.ex[i,,,] = mvspec(cbind(ex[P,i],ex[S,i]), spans=L, taper=.5, plot=FALSE)$fxx
}
```

```
u = mvspec(cbind(nz[P],nz[S]), spans=L, taper=.5)
f.NZ = u$fxx
bndwidth = u$bandwidth*40  # about .75 Hz
fhat.eq = apply(f.eq, 2:4, mean)  # average spectra
fhat.ex = apply(f.ex, 2:4, mean)

# plot the average spectra
par(mfrow=c(2,2), mar=c(3,3,2,1),  mgp = c(1.6,.6,0))
Fr = 40*(1:512)/n
plot(Fr, Re(fhat.eq[1,1,]), type="l", xlab="Frequency (Hz)", ylab="")
plot(Fr, Re(fhat.eq[2,2,]), type="l", xlab="Frequency (Hz)", ylab="")
plot(Fr, Re(fhat.ex[1,1,]), type="l", xlab="Frequency (Hz)", ylab="")
plot(Fr, Re(fhat.ex[2,2,]), type="l", xlab="Frequency (Hz)", ylab="")
mtext("Average P-spectra", side=3, line=-1.5, adj=.2, outer=TRUE)
mtext("Earthquakes", side=2, line=-1, adj=.8,  outer=TRUE)
mtext("Average S-spectra", side=3, line=-1.5, adj=.82, outer=TRUE)
mtext("Explosions", side=2, line=-1, adj=.2, outer=TRUE)

dev.new()
par(fig = c(.75, 1, .75, 1), new = TRUE)
ker = kernel("modified.daniell", L)$coef; ker = c(rev(ker),ker[-1])
plot((-33:33)/40,ker,type="l",ylab="",xlab="",cex.axis=.7,yaxp=c(0,.04,2))

# choose alpha
Balpha = rep(0,19)
for (i in 1:19){  alf=i/20
for (k in 1:256) {
Balpha[i]= Balpha[i] + Re(log(det.c(alf*fhat.ex[,,k] + (1-alf)*fhat.eq[,,k])/ det.c
(fhat.eq[,,k]))-
                           alf*log(det.c(fhat.ex[,,k])/det.c(fhat.eq[,,k])))} }
alf = which.max(Balpha)/20   # = .4
```

```
# calculate information criteria
rep(0,17) -> KLDiff -> BDiff -> KLeq -> KLex -> Beq -> Bex
for (i in 1:17){
 if (i <= 8) f0 = f.eq[i,,,]
 if (i > 8 & i <= 16) f0 = f.ex[i-8,,,]
 if (i == 17) f0 = f.NZ
 for (k in 1:256) {    # only use freqs out to .25
  tr = Re(sum(diag(solve(fhat.eq[,,k],f0[,,k]))))
  KLeq[i] = KLeq[i] + tr + log(det.c(fhat.eq[,,k])) - log(det.c(f0[,,k]))
  Beq[i] =  Beq[i] + Re(log(det.c(alf*f0[,,k]+(1-alf)*fhat.eq[,,k])/det.c(fhat.eq[,,
k])) -
                         alf*log(det.c(f0[,,k])/det.c(fhat.eq[,,k])))
  tr = Re(sum(diag(solve(fhat.ex[,,k],f0[,,k]))))
  KLex[i] = KLex[i] + tr +  log(det.c(fhat.ex[,,k])) - log(det.c(f0[,,k]))
  Bex[i] = Bex[i] + Re(log(det.c(alf*f0[,,k]+(1-alf)*fhat.ex[,,k])/det.c(fhat.ex[,,
k])) -
                       alf*log(det.c(f0[,,k])/det.c(fhat.ex[,,k])))
  }
KLDiff[i] = (KLeq[i] - KLex[i])/n
BDiff[i] =  (Beq[i] - Bex[i])/(2*n)
}

x.b = max(KLDiff)+.1; x.a = min(KLDiff)-.1
y.b = max(BDiff)+.01; y.a = min(BDiff)-.01

dev.new()
plot(KLDiff[9:16], BDiff[9:16], type="p", xlim=c(x.a,x.b), ylim=c(y.a,y.b), cex=1.1,
lwd=2,
      xlab="Kullback-Leibler Difference",ylab="Chernoff Difference",
```

```
main="Classification
      Based on Chernoff and K-L Distances", pch=6)
points(KLDiff[1:8], BDiff[1:8], pch=8, cex=1.1, lwd=2)
points(KLDiff[17], BDiff[17],  pch=3, cex=1.1, lwd=2)
legend("topleft", legend=c("EQ", "EX", "NZ"), pch=c(8,6,3), pt.lwd=2)
abline(h=0, v=0, lty=2, col="gray")
text(KLDiff[-c(1,2,3,7,14)]-.075, BDiff[-c(1,2,3,7,14)], label=names(eqexp[-c
(1,2,3,7,14)]), cex=.7)
text(KLDiff[c(1,2,3,7,14)]+.075, BDiff[c(1,2,3,7,14)], label=names(eqexp[c
(1,2,3,7,14)]), cex=.7)
```

Example 7.11

```
library(cluster)
P = 1:1024
S = P+1024
p.dim = 2
n =1024

eq = as.ts(eqexp[,1:8])
ex = as.ts(eqexp[,9:16])
nz = as.ts(eqexp[,17])

f = array(dim=c(17,2,2,512))
L = c(15,15)   # for smoothing
for (i in 1:8){     # compute spectral matrices
 f[i,,,] = mvspec(cbind(eq[P,i],eq[S,i]), spans=L, taper=.5, plot=FALSE)$fxx
 f[i+8,,,] = mvspec(cbind(ex[P,i],ex[S,i]), spans=L, taper=.5, plot=FALSE)$fxx
}
f[17,,,] = mvspec(cbind(nz[P],nz[S]), spans=L, taper=.5, plot=FALSE)$fxx

# calculate symmetric information criteria
```

```
JD = matrix(0,17,17)
for (i in 1:16){
 for (j in (i+1):17){
  for (k in 1:256) {    # only use freqs out to .25
    tr1 = Re(sum(diag(solve(f[i,,,k],f[j,,,k]))))
    tr2 = Re(sum(diag(solve(f[j,,,k], f[i,,,k]))))
    JD[i,j] = JD[i,j] + (tr1 + tr2 - 2*p.dim)
  }
 }
}

JD = (JD + t(JD))/n
colnames(JD) = c(colnames(eq), colnames(ex), "NZ")
rownames(JD) = colnames(JD)
cluster.2 = pam(JD, k = 2, diss = TRUE)

summary(cluster.2)  # print results
par(mgp = c(1.6,.6,0), cex=3/4, cex.lab=4/3, cex.main=4/3)
clusplot(JD, cluster.2$cluster, col.clus=1, labels=3, lines=0, col.p=1,
               main="Clustering Results for Explosions and Earthquakes")
text(-7,-.5, "Group I", cex=1.1, font=2)
text(1, 5, "Group II", cex=1.1, font=2)
```

Example 7.12

```
n = 128
Per = abs(mvfft(fmri1[,-1]))^2/n

par(mfrow=c(2,4), mar=c(3,2,2,1),  mgp = c(1.6,.6,0), oma=c(0,1,0,0))
for (i in 1:8) plot(0:20, Per[1:21,i], type="l", ylim=c(0,8), main=colnames(fmri1)[i
+1],
```

```
       xlab="Cycles", ylab="", xaxp=c(0,20,5))
mtext("Periodogram", side=2, line=-.3, outer=TRUE, adj=c(.2,.8))


fxx = mvspec(fmri1[,-1], kernel("daniell", c(1,1)), taper=.5, plot=FALSE)$fxx
l.val = rep(NA,64)
for (k in 1:64) {
 u = eigen(fxx[,,k], symmetric=TRUE, only.values=TRUE)
 l.val[k] = u$values[1]
}


dev.new()
plot(l.val, type="l", xaxp=c(0,64,8), xlab="Cycles (Frequency x 128)", ylab="First
Principal Component")
axis(1, seq(4,60,by=8), labels=FALSE)


# at freq k=4
u = eigen(fxx[,,4], symmetric=TRUE)
lam = u$values
evec = u$vectors
lam[1]/sum(lam) # % of variance explained
sig.e1 = matrix(0,8,8)
for (l in 2:5){  # last 3 evs are 0
 sig.e1 = sig.e1 + lam[l]*evec[,l]%*%Conj(t(evec[,l]))/(lam[1]-lam[l])^2
}
sig.e1 = Re(sig.e1)*lam[1]*sum(kernel("daniell", c(1,1))$coef^2)
p.val = round(pchisq(2*abs(evec[,1])^2/diag(sig.e1), 2, lower.tail=FALSE), 3)
cbind(colnames(fmri1)[-1], abs(evec[,1]), p.val) # print table values
```

Example 7.13

```
bhat = sqrt(lam[1])*evec[,1]
```

```
Dhat = Re(diag(fxx[,,4] - bhat%*%Conj(t(bhat))))
res = Mod(fxx[,,4] - Dhat - bhat%*%Conj(t(bhat)))
```

Example 7.14

```
gr = diff(log(ts(econ5, start=1948, frequency=4))) # growth rate
plot(100*gr, main="Growth Rates (%)")

# scale each series to have variance 1
gr = ts(apply(gr,2,scale), freq=4)   # scaling strips ts attributes
L = c(7,7)   # degree of smoothing
gr.spec = mvspec(gr, spans=L, detrend=FALSE, taper=.25, plot=FALSE)

dev.new()
plot(kernel("modified.daniell", L))  # view the kernel - not shown

dev.new()
plot(gr.spec, log="no",col=1, main="Individual Spectra", lty=1:5, lwd=2)
legend("topright", colnames(econ5), lty=1:5, lwd=2)

dev.new()
plot.spec.coherency(gr.spec, ci=NA,  main="Squared Coherencies")

# PCs
n.freq = length(gr.spec$freq)
lam = matrix(0,n.freq,5)
for (k in 1:n.freq) lam[k,] = eigen(gr.spec$fxx[,,k], symmetric=TRUE, only.
values=TRUE)$values

dev.new()
par(mfrow=c(2,1), mar=c(4,2,2,1), mgp=c(1.6,.6,0))
```

```
plot(gr.spec$freq, lam[,1], type="l", ylab="", xlab="Frequency", main="First
Eigenvalue")
abline(v=.25, lty=2)
plot(gr.spec$freq, lam[,2], type="l", ylab="", xlab="Frequency", main="Second
Eigenvalue")
abline(v=.125, lty=2)
e.vec1 = eigen(gr.spec$fxx[,,10], symmetric=TRUE)$vectors[,1]
e.vec2 =  eigen(gr.spec$fxx[,,5], symmetric=TRUE)$vectors[,2]
round(Mod(e.vec1), 2);  round(Mod(e.vec2), 3)
```

Example 7.16

```
u = factor(bnrf1ebv)  # first, input the data as factors and then
x = model.matrix(~u-1)[,1:3]  # make an indicator matrix
# x = x[1:1000,]  # select subsequence if desired

Var = var(x)  # var-cov matrix
xspec = mvspec(x, spans=c(7,7), detrend=FALSE, plot=FALSE)
fxxr = Re(xspec$fxx)  # fxxr is real(fxx)

# compute Q = Var^-1/2
ev = eigen(Var)
Q = ev$vectors%*%diag(1/sqrt(ev$values))%*%t(ev$vectors)

# compute spec env and scale vectors
num     = xspec$n.used  # sample size used for FFT
nfreq   = length(xspec$freq)  # number of freqs used
specenv = matrix(0,nfreq,1)   # initialize the spec envelope
beta    = matrix(0,nfreq,3)   # initialize the scale vectors

for (k in 1:nfreq){
```

```
  ev = eigen(2*Q%*%fxxr[,,k]%*%Q/num, symmetric=TRUE)
  specenv[k] = ev$values[1]   # spec env at freq k/n is max evalue
  b = Q%*%ev$vectors[,1]      # beta at freq k/n
  beta[k,] = b/sqrt(sum(b^2)) # helps to normalize beta
}

# output and graphics
frequency = xspec$freq
plot(frequency, 100*specenv, type="l", ylab="Spectral Envelope (%)")

# add significance threshold to plot
m = xspec$kernel$m
etainv = sqrt(sum(xspec$kernel[-m:m]^2))
thresh = 100*(2/num)*exp(qnorm(.9999)*etainv)*rep(1,nfreq)
lines(frequency, thresh, lty="dashed", col="blue")

# details
output = cbind(frequency, specenv, beta)
colnames(output) = c("freq","specenv", "A", "C", "G")
round(output,3)
```

Example 7.17

```
u   = astsa::nyse
x   = cbind(u, abs(u), u^2)   # possible transforms (identity, absolute value, square)
#
Var = var(x)                  # var-cov matrix
xspec = mvspec(x, spans=c(5,3), taper=.5, plot=FALSE)  # spectral matrices are called
fxx
fxxr = Re(xspec$fxx)          # fxxr is real(fxx)
#- compute Q = Var^-1/2
```

```
ev = eigen(Var)

Q = ev$vectors%*%diag(1/sqrt(ev$values))%*%t(ev$vectors)

#- compute spec env and scale vectors

num     = xspec$n.used             # sample size used for FFT

nfreq   = length(xspec$freq)       # number of freqs used

specenv = matrix(0,nfreq,1)        # initialize the spec envelope

beta    = matrix(0,nfreq,3)        # initialize the scale vectors

 for (k in 1:nfreq){

  ev = eigen(2*Q%*%fxxr[,,k]%*%Q/num)  # get evalues of normalized spectral matrix at 
freq k/n

  specenv[k] = ev$values[1]            # spec env at freq k/n is max evalue

  b = Q%*%ev$vectors[,1]               # beta at freq k/n

  beta[k,] = b/b[1]                    # first coef is always 1

  }

#--- output and graphics ---#

par(mar=c(2.5,2.75,.5,.5), mgp=c(1.5,.6,0))

frequency = xspec$freq

plot(frequency, 100*specenv, type="l", ylab="Spectral Envelope (%)", panel.first=grid
(lty=2))

 ## add significance threshold to plot ##

 m=xspec$kernel$m

 etainv=sqrt(sum(xspec$kernel[-m:m]^2))

thresh=100*(2/num)*exp(qnorm(.9999)*etainv)*matrix(1,nfreq,1)

lines(frequency,thresh, lty="dashed", col="blue")

#--  details  --#

output = cbind(frequency, specenv, beta)

colnames(output)=c("freq","specenv","x", "|x|", "x^2")

round(output,4)

b = sign(b[2])*output[2,3:5]


dev.new()
```

```
par(mar=c(2.5,2.5,.5,.5), mgp=c(1.5,.6,0))

# plot transform

g = function(x) {b[1]*x+b[2]*abs(x)+b[3]*x^2}

curve(g, -.2, .2, panel.first=grid())

g2 = function(x) {b[2]*abs(x)}

curve(g2, -.2,.2, add=TRUE, lty=6, col=4)
```

[-]