Time Series Analysis and Its Applications: With R Examples

Third 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 downloaded tsa3.rda and loaded it into R [`load("tsa3.rda")`]. Occasionally, we will improve, correct or streamline the code below, so there may be a few cases where the code here differs slightly from what is in the text. Also, some differences exist simply because there is more room here than in the text. You can check the revision history & notes page for info about changes to tsa3.rda.

Click the [+] to expand or collapse a section. Click the [-] to collapse the entire page.

[+] Chapter 1

Example 1.1

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

Example 1.2

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

Example 1.3

```
plot(speech)
```

Example 1.4

```
plot(nyse, ylab="NYSE Returns")
```

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], lty=c(1,2,4,5), ylab="BOLD", xlab="", main="Cortex")
ts.plot(fmri1[,6:9], lty=c(1,2,4,5), 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(200,0,1)
x = cumsum(w)
wd = w +.2
xd = cumsum(wd)
plot.ts(xd, ylim=c(-5,55), main="random walk")
lines(x)
lines(.2*1:200, lty="dashed")
```

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

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

Example 1.25

```
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 1.26

```
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 1.27

```
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])
rs3 = rbind(rs2[41:2,], rs2)
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(gtemp~time(gtemp)))  # regress gtemp on time
plot(gtemp, type="o", ylab="Global Temperature Deviation")
abline(fit)  # add 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))
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
AIC(fit, k=log(num))/num - log(2*pi)  # BIC
(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(gtemp~time(gtemp), na.action=NULL) # regress gtemp on time
par(mfrow=c(2,1))
plot(resid(fit), type="o", main="detrended")
plot(diff(gtemp), type="o", main="first difference")
dev.new()
par(mfrow=c(3,1))    # plot ACFs
acf(gtemp, 48, main="gtemp")
acf(resid(fit), 48, main="detrended")
acf(diff(gtemp), 48, main="first difference")
```

Example 2.6

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

Example 2.7

```
lag.plot1(soi, 12)
dev.new()
lag.plot2(soi, rec, 8)
```

Example 2.8

```
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
plot.ts(x, lty="dashed")
lines(fitted(fit), lwd=2)
```

Example 2.9

```
I = abs(fft(x))^2/500  # the periodogram
P = (4/500)*I[1:250]   # the scaled periodogram
f = 0:249/500          # frequencies
plot(f, P, type="l", xlab="Frequency", ylab="Scaled Periodogram")
```

Example 2.10

```
ma5 = filter(cmort, sides=2, rep(1,5)/5)
ma53 = filter(cmort, sides=2, rep(1,53)/53)
plot(cmort, type="p", ylab="mortality")
lines(ma5)
lines(ma53)
```

Example 2.11

```
wk = time(cmort) - mean(time(cmort))  # wk is essentially t/52 centered at zero
wk2 = wk^2
wk3 = wk^3
cs = cos(2*pi*wk)
sn = sin(2*pi*wk)
reg1 = lm(cmort~wk + wk2 + wk3, na.action=NULL)
reg2 = lm(cmort~wk + wk2 + wk3 + cs + sn, na.action=NULL)
plot(cmort, type="p", ylab="mortality")
lines(fitted(reg1))
lines(fitted(reg2))
```

Example 2.12

```
plot(cmort, type="p", ylab="mortality")
lines(ksmooth(time(cmort), cmort, "normal", bandwidth=5/52))
lines(ksmooth(time(cmort), cmort, "normal", bandwidth=2))
```

Example 2.13

```
par(mfrow=c(2,1))
plot(cmort, type="p", ylab="mortality", main="nearest neighbor")
lines(supsmu(time(cmort), cmort, span=.5))
lines(supsmu(time(cmort), cmort, span=.01))
plot(cmort, type="p", ylab="mortality", main="lowess")
lines(lowess(cmort, f=.02))
lines(lowess(cmort, f=2/3))
```

Example 2.14

```
plot(cmort, type="p", ylab="mortality")
lines(smooth.spline(time(cmort), cmort))
lines(smooth.spline(time(cmort), cmort, spar=1))
```

Example 2.15

```
par(mfrow=c(2,1), mar=c(3,2,1,0)+.5, mgp=c(1.6,.6,0))
plot(tempr, cmort, main="lowess", xlab="Temperature", ylab="Mortality")
lines(lowess(tempr,cmort))
plot(tempr, cmort, main="smoothing splines", xlab="Temperature", ylab="Mortality")
lines(smooth.spline(tempr, cmort))
```

[-]

[+] Chapter 3

Example 3.1

```
par(mfrow=c(2,1))                          # in the expression 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 3.4

```
par(mfrow=c(2,1))
plot(arima.sim(list(order=c(0,0,1), ma=.5), n=100), ylab="x", main=(expression(MA(1)~~~theta==+.5)))
plot(arima.sim(list(order=c(0,0,1), ma=-.5), n=100), ylab="x", main=(expression(MA(1)~~~theta==-.5)))
```

Example 3.10

```
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(90210)
ar2 = arima.sim(list(order=c(2,0,0), ar=c(1.5,-.75)), n = 144)
plot(1:144/12, ar2, type="l", xlab="Time (one unit = 12 points)")
abline(v=0:12, lty="dotted", lwd=2)

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

Example 3.11

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

Example 3.15

```
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 3.17

```
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 3.24

```
regr = ar.ols(rec, order=2, demean=FALSE, intercept=TRUE)
fore = predict(regr, n.ahead=24)
ts.plot(rec, fore$pred, col=1:2, xlim=c(1980,1990))
lines(fore$pred, type="p", col=2)
lines(fore$pred + fore$se, lty="dashed", col=4)
lines(fore$pred - fore$se, lty="dashed", col=4)
```

Example 3.27

```
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 3.28

```
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 3.29

```
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 3.35

```
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="")
lines(density(phi.yw, bw=.015))

x = ar1boot
m = mean(x)   # estimate of mu
fit = ar.yw(x, order=1)
phi = fit$ar  # estimate of phi
nboot = 200   # number of bootstrap replicates
resids = fit$resid[-1]  # the first resid is NA
x.star = x    # initialize x.star
phi.star.yw = rep(NA, nboot)
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
 }
dev.new()
hist(phi.star.yw, 10, main="", prob=TRUE, ylim=c(0,14), xlim=c(.75,1.05))
lines(density(phi.star.yw, bw=.02))
u = seq(.75, 1.05, by=.001)
lines(u, dnorm(u,mean=.96,sd=.03), lty="dashed", lwd=2)
```

Example 3.37

```
set.seed(666)   # not that 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-&lambda here
plot(x.ima)
```

Example 3.38

```
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 3.40

```
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

```
phi = c(rep(0,11),.8)
ACF = ARMAacf(ar=phi, ma=-.5, 50)[-1]
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 3.46

```
acf2(prodn,48)
dev.new()
acf2(diff(prodn), 48)
dev.new()
acf2(diff(diff(prodn),12), 48)
dev.new()
sarima(prodn,2,1,1,0,1,3,12)        # fit model (ii)
dev.new()
sarima.for(prodn,12,2,1,1,0,1,3,12) # 12 month ahead 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 4.6

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

Example 4.8

```
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
abs(fft(x))^2/5  # the periodogram (as a check)
```

Example 4.10

```
par(mfrow=c(2,1))
soi.per = spec.pgram(soi, taper=0, log="no")
abline(v=1/4, lty="dotted")
rec.per = spec.pgram(rec, taper=0, 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 4.11

```
par(mfrow=c(2,1))
k = kernel("daniell",4)
soi.ave = spec.pgram(soi, k, taper=0, log="no")
abline(v=c(.25,1,2,3), lty=2)
soi.ave$bandwidth       # 0.0649519;  reported bandwidth
soi.ave$bandwidth*(1/12)*sqrt(12)  # 0.01875; our Bw
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]        # 0.04952026
soi.ave$spec[40]        # 0.1190800
# intervals
df*soi.ave$spec[10]/L   # 0.02787891
df*soi.ave$spec[10]/U   # 0.1113333
df*soi.ave$spec[40]/L   # 0.06703963
df*soi.ave$spec[40]/U   # 0.2677201
# Repeat above commands with soi replaced by rec
```

Example 4.12

```
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))
```

Example 4.13

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

par(mfrow=c(2,1))
k = kernel("modified.daniell", c(3,3))
soi.smo = spec.pgram(soi, k, taper=0, log="no")
  abline(v=1, lty="dotted")
  abline(v=1/4, lty="dotted")
rec.smo = spec.pgram(rec, k, taper=0, log="no")
  abline(v=1, lty="dotted")
  abline(v=1/4, lty="dotted")

df = soi.smo$df           # df = 17.42618
Lh = 1/sum(k[-k$m:k$m]^2) # Lh = 9.232413
Bw = Lh/480               # Bw = 0.01923419

# An easier way to obtain soi.smo (you can add log="no"):
soi.smo = spectrum(soi, spans=c(7,7), taper=0)
```

Example 4.14

```
s0 = spectrum(soi, spans=c(7,7), taper=0, plot=FALSE)
s50 = spectrum(soi, spans=c(7,7), taper=.5, plot=FALSE)
plot(s0$freq, s0$spec, log="y", type="l", lty=2, ylab="spectrum", xlab="frequency")  # dashed line
lines(s50$freq, s50$spec)   # solid line
title("Tapering")
legend("topright", c("with you", "without you"), lty=1:2)
```

Example 4.15

```
spaic = spec.ar(soi, log="no", ylim=c(0,.3))  # min AIC spec
text(frequency(soi)*1/52, .07, substitute(omega==1/52))  # El Nino Cycle
text(frequency(soi)*1/12, .29, substitute(omega==1/12))  # Yearly Cycle
sp16 = spec.ar(soi,order=16, log="no", plot=F)
lines(sp16$freq, sp16$spec, lty="dashed")  # ar16 spec
(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 ICs
n = length(soi)
AIC = rep(0,30) -> AICc -> BIC
for (k in 1:30){
  fit = ar(soi, order=k, aic=FALSE)
  sigma2 = var(fit$resid, na.rm=TRUE)
  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) }
IC = cbind(AIC, BIC+1)
dev.new()
ts.plot(IC, type="o", xlab="p", ylab="AIC / BIC")
text(15, -1.5, "AIC")
text(15, -1.38, "BIC")
grid()
```

Example 4.18

```
sr = spec.pgram(cbind(soi,rec), kernel("daniell",9), taper=0, plot=FALSE)
sr$df                     # df = 35.8625
f = qf(.999, 2, sr$df-2)  # f = 8.529792
C = f/(18+f)              # C = 0.3188779
plot(sr, plot.type = "coh", ci.lty = 2)
abline(h = C)
```

Example 4.19

```
par(mfrow=c(3,1))
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()                          # open new graphics device
spectrum(soif, spans=9, log="no")  # spectral analysis
abline(v=12/52, lty="dashed")
dev.new()
w = seq(0, .5, length=500)         # frequency response
FR = abs(1-exp(2i*pi*w))^2
plot(w, FR, type="l")
```

Example 4.21

```
nobs = length(EXP6)  # number of observations
wsize = 256          # window size
overlap = 128        # overlap
ovr = wsize-overlap
nseg = floor(nobs/ovr)-1;  # number of segments
krnl = kernel("daniell", c(1,1))     # kernel
ex.spec = matrix(0, wsize/2, nseg)
 for (k in 1:nseg) {
  a = ovr*(k-1)+1
  b = wsize+ovr*(k-1)
  ex.spec[,k] = spectrum(EXP6[a:b], krnl, taper=.5, plot=F)$spec
  }
x = seq(0, 10,  len = nrow(ex.spec)/2)
y = seq(0, ovr*nseg, len = ncol(ex.spec))
z = ex.spec[1:(nrow(ex.spec)/2),]
# below is text version
filled.contour(x,y,log(z),ylab="time",xlab="frequency (Hz)",nlevels=12,col=gray(11:0/11),main="Explos
dev.new()  # a nicer version with color
filled.contour(x, y, log(z), ylab="time", xlab="frequency (Hz)", main="Explosion")
dev.new()  # below not shown in text
persp(x,y,z,zlab="Power",xlab="frequency (Hz)",ylab="time",ticktype="detailed",theta=25,d=2,main="Exp
```

Example 4.22

```
library(wavethresh)
eq = scale(EQ5)  # standardize the series
ex = scale(EXP6)
eq.dwt = wd(eq, filter.number=8)
ex.dwt = wd(ex, filter.number=8)
# plot the wavelet transforms
par(mfrow = c(1,2))
plot(eq.dwt, main="Earthquake")
plot(ex.dwt, main="Explosion")
# total power
TPe = rep(NA,11)  # for the earthquake series
for (i in 0:10){TPe[i+1] = sum(accessD(eq.dwt, level=i)^2)}
TotEq = sum(TPe)  # check with sum(eq^2)
TPx = rep(NA,11)  # for the explosion series
for (i in 0:10){TPx[i+1] = sum(accessD(ex.dwt, level=i)^2)}
TotEx = sum(TPx)  # check with sum(ex^2)
# make a nice table
Power = round(cbind(11:1, 0:10, 100*TPe/TotEq, 100*TPx/TotEx), digits=3)
colnames(Power) = c("Level", "Resolution", "EQ(%)", "EXP(%)")
Power
```

Example 4.23

```
library(wavethresh)
eq = scale(EQ5)
par(mfrow=c(3,1))
eq.dwt = wd(eq, filter.number=8)
eq.smo = wr(threshold(eq.dwt, levels=5:10))
ts.plot(eq, main="Earthquake", ylab="Data")
ts.plot(eq.smo, ylab="Signal")
ts.plot(eq-eq.smo, ylab="Resid")
```

Example 4.24

```
LagReg(soi, rec, L=15, M=32, threshold=6)
LagReg(rec, soi, L=15, M=32, inverse=TRUE, threshold=.01)
```

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])
per3 = rbind(per2[32:2,], per2)
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

```
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 5.2

```
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
```

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
sarima(gnpgr, 1, 0, 0)  # fit an AR(1)
acf2(innov^2, 24)       # get (p)acf of the squared residuals

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

Example 5.5

```
library(fGarch)
summary(nyse.g <- garchFit(~garch(1,1), nyse))
u = nyse.g@sigma.t
plot(window(nyse, start=900, end=1000), ylim=c(-.22,.2), ylab="NYSE Returns")
lines(window(nyse-2*u, start=900, end=1000), lty=2, col=4)
lines(window(nyse+2*u, start=900, end=1000), lty=2, col=4)
```

Example 5.6

```
plot(flu, type="c")  # first 3 lines plot the data with months as symbols
Months = c("J","F","M","A","M","J","J","A","S","O","N","D")
points(flu, pch=Months, cex=.8, font=2)

dflu = diff(flu)
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])  # get predictions
b1 = fit1$coef
b2 = fit2$coef
p1 = D%*%b1
p2=D%*%b2
prd = ifelse(Z[,2] < thrsh, p1, p2)
plot(dflu, type="p", pch=2, ylim=c(-.5,.5))
lines(prd, lty=2, col=2)
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)
lines(prd + 2*prde, col=3)
lines(prd - 2*prde, col=3)
```

Example 5.7

```
trend = time(cmort)
temp = tempr - mean(tempr)
```

```
temp2 = temp^2
fit = lm(cmort~trend + temp + temp2 + part, na.action=NULL)
acf2(resid(fit), 52)   # implies AR2
(fit2 = arima(cmort, order=c(2,0,0), xreg=cbind(trend,temp,temp2,part)))
dev.new()
acf2(resid(fit2), 52)  # no obvious departures from whiteness
(Q = Box.test(resid(fit2), 12, type="Ljung")$statistic)  # X-squared = 6.91
pchisq(as.numeric(Q), df=(12-2), lower=FALSE)  # p-value = .73
```

Example 5.8

```
acf2(soi)
(fit = arima(soi, xreg=time(soi), order=c(1, 0, 0)))
ar1 = as.numeric(fit$coef[1])    # = 0.5875387
soi.pw = resid(fit)
rec.d = resid(lm(rec~time(rec), na.action=NULL))
rec.fil = filter(rec.d, filter=c(1, -ar1), method="conv", sides=1)
ccf(soi.pw, rec.fil, main="", ylab="CCF", na.action=na.omit)
```

Example 5.9

```
rec.d = resid(lm(rec~time(rec), na.action=NULL))
soi.d = resid(lm(soi~time(soi), na.action=NULL))
fish = ts.intersect(rec.d, rec.d1=lag(rec.d,-1), soi.d5=lag(soi,-5), dframe=TRUE)
summary(fish.fit <- lm(rec.d~0+rec.d1+soi.d5, data=fish))
om1 = as.numeric(fish.fit$coef[1])
eta.hat = filter(resid(fish.fit), filter=c(1,-om1), method="recur", sides=1)
acf2(eta.hat)
(eta.fit <- arima(eta.hat, order=c(3,0,0)))
```

Example 5.10

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

Example 5.11

```
# continued from 5.10
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
```

[-]

[+] Chapter 6

Example 6.1

```
blood = cbind(WBC, PLT, HCT)
blood = replace(blood, blood==0, NA)
plot(blood, type="o", pch=19, xlab="day", main="")
```

Example 6.2

```
ts.plot(HL, FL, lty=1:2, 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
Time = 1:num; par(mfrow=c(3,1))
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$P0n)   # initial value info
# In case you can't see the differences in the 3 figures for this example, try this...
# Predictor, Filter, Smoother on Same Plot (not shown)
dev.new()
plot(Time, mu[-1])
lines(ks$xp, col=4)
lines(ks$xf, col=3)
lines(ks$xs, col=2)
names = c("predictor","filter","smoother")
legend("bottomright", names, col=4:2, lty=1)
```

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(gtemp, gtemp2)
num = nrow(y)
input = rep(1,num)
A = array(rep(1,2), dim=c(2,1,num))
mu0 = -.26; Sigma0 = .01;  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
xsmooth = ts(as.vector(ks$xs), start=1880)
plot(xsmooth, lwd=2, ylim=c(-.5,.8), ylab="Temp Deviations")
lines(gtemp, col="blue", lty=2)     # color helps here
lines(gtemp2, col="red", lty=2)
```

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,0,0,cQ,cR,0,100,.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), mar=c(4,4,1,1)+.2)
plot(WBC, type="p", pch=19, ylim=c(1,5), xlab="day")
 lines(y1s); lines(y1s+p1, lty=2); lines(y1s-p1, lty=2)
plot(PLT, type="p",  ylim=c(3,6), pch=19, xlab="day")
 lines(y2s); lines(y2s+p2, lty=2); lines(y2s-p2, lty=2)
plot(HCT, type="p", pch=19, ylim=c(20,40), xlab="day")
 lines(y3s); lines(y3s+p3, lty=2); lines(y3s-p3, lty=2)
```

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(1,4); cQ[1,1]=cQ1; cQ[2,2]=cQ2
cR = est$par[4]
ks = Ksmooth0(num,jj,A,mu0,Sigma0,Phi,cQ,cR)
# Plot
Tsm = ts(ks$xs[1,,], start=1960, freq=4)
Ssm = ts(ks$xs[2,,], start=1960, freq=4)
p1 = 2*sqrt(ks$Ps[1,1,]); p2 = 2*sqrt(ks$Ps[2,2,])
par(mfrow=c(3,1))
plot(Tsm, main="Trend Component", ylab="Trend")
 lines(Tsm+p1, lty=2, col=4); lines(Tsm-p1,lty=2, col=4)
plot(Ssm, main="Seasonal Component", ylim=c(-5,4), ylab="Season")
 lines(Ssm+p2,lty=2, col=4); lines(Ssm-p2,lty=2, col=4)
plot(jj, type="p", main="Data (points) and Trend+Season (line)")
 lines(Tsm+Ssm)
# Forecast
n.ahead=12; y = ts(append(jj, rep(0,n.ahead)), start=1960, freq=4)
rmspe = rep(0,n.ahead); x00 = ks$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)  }
dev.new()
plot(y, type="o", main="", ylab="", 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)
lines(upp, lty=2);  lines(low, lty=2);  abline(v=1980.75, lty=3)
```

Example 6.12

```
dm = cmort-mean(cmort)   # center mortality
trend = time(cmort) - mean(time(cmort))
```

```
ded = ts.intersect(dm, u1=lag(tempr,-1), u2=part, u3=lag(part,-4), u4=trend, dframe=TRUE)
y = ded$dm; input = cbind(ded$u1, ded$u2, ded$u3, ded$u4)
num = length(y); A = array(c(1,0), dim = c(1,2,num))
# Function to Calculate Likelihood
Linn=function(para){
  phi1=para[1]; phi2=para[2]; cR=para[3]
  b1=para[4]; b2=para[5]; b3=para[6]; b4=para[7]
  mu0 = matrix(c(0,0),2,1); Sigma0=diag(100,2)
  Phi = matrix(c(phi1, phi2, 1, 0), 2)
  Theta = matrix(c(phi1, phi2), 2)
  Ups = matrix(c(b1,0,b2,0,b3,0,0,0),2,4)
  Gam = matrix(c(0,0,0,b4),1,4)
  cQ = cR;  S = cR^2
  kf = Kfilter2(num,y,A,mu0,Sigma0,Phi,Ups,Gam,Theta,cQ,cR,S,input)
  return(kf$like)
  }
# Estimation - if this takes a long time to run on your machine, use "factr" or "pgtol" in optim cont
phi1=.4;  phi2=.4;  cR=5; b1=-.1; b2=.1; b3=.1; b4=-1.5
init.par = c(phi1,phi2,cR,b1,b2,b3,b4)  # initial parameter values
est = optim(init.par,Linn,NULL,method="L-BFGS-B",hessian=TRUE, control=list(trace=1,REPORT=1))
SE = sqrt(diag(solve(est$hessian)))
# Results
u = cbind(estimate=est$par, SE)
rownames(u)=c("phi1","phi2","sigv","TL1","P","PL4","trnd"); u
```

Example 6.13

```
# NOTE: Change line below to tol=.01 or tol=.001 if this takes a long time to run - depends on your m
tol = sqrt(.Machine$double.eps)
#
nboot = 500  # number of bootstrap replicates
y = window(qinfl, c(1953,1), c(1965,2))  # inflation
z = window(qintr, c(1953,1), c(1965,2))  # interest
num = length(y)
A = array(z, dim=c(1,1,num))
input = matrix(1,num,1)
# Function to Calculate Likelihood
Linn=function(para){
  phi=para[1]; alpha=para[2]; b=para[3]; Ups=(1-phi)*b
  cQ=para[4]; cR=para[5]
  kf=Kfilter2(num,y,A,mu0,Sigma0,phi,Ups,alpha,1,cQ,cR,0,input)
  return(kf$like)
  }
# Parameter Estimation
mu0=1; Sigma0=.01; phi=.84; alpha=-.77; b=.85; cQ=.12; cR=1.1
init.par = c(phi,alpha,b,cQ,cR)    # initial parameters
est = optim(init.par, Linn, NULL, method="BFGS", hessian=TRUE, control=list(trace=1,REPORT=1,reltol=t
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]
cbind(estimate=est$par, SE)
# BEGIN BOOTSTRAP
# Likelihood for the bootstrapped data
Linn2=function(para){
  phi=para[1]; alpha=para[2]; b=para[3]; Ups=(1-phi)*b
  cQ=para[4]; cR=para[5]
  kf=Kfilter2(num,y.star,A,mu0,Sigma0,phi,Ups,alpha,1,cQ,cR,0,input)
  return(kf$like)  }
# 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; y.star=y; xp.star=xp; k=4:50
para.star = matrix(0, nboot, 5)  # to store estimates
init.par=c(.84,-.77,.85,.12,1.1)
for (i in 1:nboot){cat("iteration:", i, "\n")
 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, Linn2, NULL, 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") }
# phi and sigw
phi = para.star[,1]; sigw = abs(para.star[,4])
phi = ifelse(phi<0, NA, phi)  # any φ < 0 not plotted
panel.hist <- function(x, ...){
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(usr[1:2], 0, 1.5) )
  h <- hist(x, plot = FALSE)
  breaks <- h$breaks; nB <- length(breaks)
  y <- h$counts; y <- y/max(y)
  rect(breaks[-nB], 0, breaks[-1], y, ...)}
u = cbind(phi, sigw);  colnames(u) = c("f","s")
pairs(u, cex=1.5, pch=1, diag.panel=panel.hist, cex.labels=1.5, font.labels=5)
```

Example 6.17

```
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)      # prec 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);
    denom = pip1*den1 + pip2*den2;
    pif1 = pip1*den1/denom; pif2 = pip2*den2/denom;
    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]
par(mfrow=c(3,1), mar=c(2,3,1,1)+.1, cex=.9)
plot(Time, y[k], type="o", ylim=c(0,1),ylab="")
 lines(Time, predepi[k],  lty="dashed", lwd=1.2)
 text(1979,.95,"(a)")
plot(Time, xfilter[1,,k], type="l", ylim=c(-.1,.4), ylab="")
 lines(Time, xfilter[3,,k]); lines(Time, xfilter[4,,k])
 text(1979,.35,"(b)")
plot(Time, y[k], type="p", pch=1, ylim=c(.1,.9),ylab="")
 prde1 = 2*sqrt(innov.sig[1]); prde2 = 2*sqrt(innov.sig[2])
 prde = ifelse(predepi[k]<.5, prde1,prde2)
 lines(Time, yp[k]+prde, lty=2, lwd=1.5)
 lines(Time, yp[k]-prde, lty=2, lwd=1.5)
 text(1979,.85,"(c)")
```

Example 6.18

```
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(); par(mfrow=c(2,1)); Time=801:1000
plot(Time, y[Time], type="l", main="log(Squared NYSE Returns)")
plot(Time, sv$xp[Time],type="l", main="Predicted log-Volatility", ylim=c(-1.5,1.8), ylab="", xlab="")
 lines(Time, sv$xp[Time]+2*sqrt(sv$Pp[Time]), lty="dashed")
 lines(Time, sv$xp[Time]-2*sqrt(sv$Pp[Time]), lty="dashed")
```

Example 6.19

```
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="")
u = seq(.4, 1.2, by=.01)
lines(u,dnorm(u, mean=.8790267, sd=.1061884), lty="dashed", lwd=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

```
attach(climhyd)
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; M = 100;
alpha = .001;  fdr = .001
nq = 2         # number of inputs  (Temp and Precip)
```

```
# Spectral Matrix
Yspec = mvspec(Y, spans=L, kernel="daniell", detrend=TRUE, demean=FALSE, taper=.1)
n = Yspec$n.used          # effective sample size
Fr = Yspec$freq           # fundamental freqs
n.freq = length(Fr)       # number of frequencies
Yspec$bandwidth*sqrt(12)  # = 0.050 - the bandwidth
# Coherencies (see sec 4.7 also)
Fq = qf(1-alpha, 2, L-2); cn = Fq/(L-1+Fq)
plt.name = c("(a)","(b)","(c)","(d)","(e)","(f)")
dev.new()
par(mfrow=c(2,3), cex.lab=1.2)
# The coherencies are listed as 1,2,...,15=choose(6,2)
for (i in 11:15){
 plot(Fr,Yspec$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 is called eF; the use of F alone should be avoided)
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$Betahat[,1], type = "h", xlab = "", ylab =names(climhyd[1]), ylim = c(-.025, .055), lw
abline(h=0)
title(main = "Impulse Response Functions")
plot(S, coh.15$Betahat[,2], type = "h", xlab = "Index", ylab = names(climhyd[5]), ylim = c(-.015, .05
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]))
beam = rowMeans(Y)
par(mfrow=c(4,1), mar=c(0,5.1,0,5.1), oma=c(6,0,5,0))
plot.ts(sensor1, xaxt="no")
title(main="Infrasonic Signals and Beam", outer=TRUE)
plot.ts(sensor2, xaxt="no");  plot.ts(sensor3, xaxt="no")
plot.ts(beam);  title(xlab="Time", outer=TRUE)
```

Example 7.4

```
attach(beamd)
L = 9; fdr = .001; N = 3
Y = cbind(beamd, beam=rowMeans(beamd))
```

```
n = nextn(nrow(Y))
Y.fft = mvfft(as.ts(Y))/sqrt(n)
Df = Y.fft[,1:3]  # fft of the data
Bf = Y.fft[,4]  # beam fft
ssr = N*Re(Bf*Conj(Bf))  # raw signal spectrum
sse = Re(rowSums(Df*Conj(Df))) - ssr  # raw error spectrum
# Smooth
SSE = filter(sse, sides=2, filter=rep(1/L,L), circular=TRUE)
SSR = filter(ssr, sides=2, filter=rep(1/L,L), circular=TRUE)
SST = SSE + SSR
par(mfrow=c(2,1), mar=c(4,4,2,1)+.1)
Fr = 0:(n-1)/n;  nFr = 1:200   # number of freqs to plot
plot(Fr[nFr], SST[nFr], type="l", ylab="log Power", xlab="", main="Sum of Squares",log="y")
lines(Fr[nFr], SSE[nFr], type="l", lty=2)
eF = (N-1)*SSR/SSE; df1 = 2*L;  df2 = 2*L*(N-1)
pvals = pf(eF, df1, df2, lower=FALSE)  # p values for FDR
pID = FDR(pvals, fdr); Fq = qf(1-fdr, df1, df2)
plot(Fr[nFr], eF[nFr], type="l", ylab="F-statistic", xlab="Frequency",  main="F Statistic")
abline(h=c(Fq, eF[pID]), lty=1:2)
```

Example 7.6

```
attach(beamd)
L = 9; M = 100; M2 = M/2; N = 3
Y = cbind(beamd, beam <- rowMeans(beamd))
n = nextn(nrow(Y));  n.freq=n/2
Y[,1:3] = Y[,1:3]-Y[,4]
Y.fft = mvfft(as.ts(Y))/sqrt(n)
Ef = Y.fft[,1:3]  # fft of the error
Bf = Y.fft[,4]  # beam fft
ssr = N*Re(Bf*Conj(Bf))  # Raw Signal Spectrum
sse = Re(rowSums(Ef*Conj(Ef))) # Raw Error Spectrum
# Smooth
SSE=filter(sse, sides=2, filter=rep(1/L,L), circular=TRUE)
SSR=filter(ssr, sides=2, filter=rep(1/L,L), circular=TRUE)
# Estimate Signal and Noise Spectra
fv = SSE/(L*(N-1))  # Equation (7.77)
fb = (SSR-SSE/(N-1))/(L*N)  # Equation (7.78)
fb[fb<0] = 0
H0 = N*fb/(fv+N*fb)
H0[ceiling(.04*n):n] = 0  # zero out H0 beyond frequency .04
# Extend components to make it a valid transform
H0 = c(H0[1:n.freq], rev(H0[2:(n.freq+1)]))
h0 = Re(fft(H0, inverse = TRUE))  # Impulse Response
h0 = c(rev(h0[2:(M2+1)]), h0[1:(M2+1)])  # center it
h1 = spec.taper(h0, p = .5)  # taper it
k1 = h1/sum(h1)  # normalize it
f.beam = filter(Y$beam, filter=k1, sides=2) # filter it
# Graphics
nFr = 1:50  # number of freqs displayed
Fr = (nFr-1)/n  # frequencies
layout(matrix(c(1, 2, 4, 1, 3, 4), nc=2))
par(mar=c(4,4,2,1) + .1)
plot(10*Fr, fb[nFr], type="l", ylab="Power", xlab="Frequency (Hz)")
 lines(10*Fr, fv[nFr], lty=2)
 text(.24, 5, "(a)", cex=1.2)
plot(10*Fr,H0[nFr],type="l",ylab="Frequency Response",xlab="Frequency (Hz)")
 text(.23, .84, "(b)", cex=1.2)
plot(-M2:M2,k1,type="l",ylab="Impulse Response",xlab="Index", lwd=1.5)
 text(45, .022, "(c)", cex=1.2)
```

```
ts.plot(cbind(f.beam,beam), lty=1:2, ylab="beam")
 text(2040, 2, "(d)", cex=1.2)
```

Example 7.7

```
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)   # df 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 7.8

```
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.subject = sum(N)
n.para = 6    # number of parameters
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 7.9

```
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.subject = sum(N); L = 3
# 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 7.10

```
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 Matrice
abline(h = qchisq(.9999, p.dim^2))
```

Example 7.11

```
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 7.12

```
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)$fxx
 f.ex[i,,,] = mvspec(cbind(ex[P,i],ex[S,i]), spans=L, taper=.5)$fxx}
```

```
 u = mvspec(cbind(nz[P],nz[S]), spans=L, taper=.5)
 f.NZ = u$fxx
bndwidth = u$bandwidth*sqrt(12)*40  # ~.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)
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 7.13

```
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)$fxx
 f[i+8,,,] = mvspec(cbind(ex[P,i],ex[S,i]), spans=L, taper=.5)$fxx }
f[17,,,] = mvspec(cbind(nz[P],nz[S]), spans=L, taper=.5)$fxx
JD = matrix(0,17,17)
# calculate symmetric information criteria
for (i in 1:16){
 for (j in (i+1):17){
  for (k in 1:256) {    # only use freqs out to .25
    tr1 = Re(sum(diag(solve(f[i,,,k],f[j,,,k]))))
    tr2 = Re(sum(diag(solve(f[j,,,k], f[i,,,k]))))
    JD[i,j] = JD[i,j] + (tr1 + tr2 - 2*p.dim)}}}
 JD = (JD + t(JD))/n
colnames(JD) = c(colnames(eq), colnames(ex), "NZ")
rownames(JD) = colnames(JD)
cluster.2 = pam(JD, k = 2, diss = TRUE)
summary(cluster.2)  # print results
par(mgp = c(1.6,.6,0), cex=3/4, cex.lab=4/3, cex.main=4/3)
clusplot(JD, cluster.2$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 7.14

```
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)$fxx
l.val = rep(NA,64)
for (k in 1:64) {
 u = eigen(fxx[,,k], symmetric=TRUE, only.values=TRUE)
 l.val[k] = u$values[1]}
dev.new()
plot(l.val, type="l", xaxp=c(0,64,8), xlab="Cycles (Frequency x 128)", ylab="First Principal Componen
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 7.15

```
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 7.16

```
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, demean=FALSE, detrend=FALSE, taper=.25)
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 7.18

```
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))
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 7.19

```
u = arima(diff(log(gnp)), order=c(0,0,2))$resid  # residuals
x = cbind(u, abs(u), u^2)  # transformation set
Var = var(x)
xspec = mvspec(x, spans=c(5,3), taper=.1)
fxxr = Re(xspec$fxx);  ev = eigen(Var)
Q = ev$vectors%*%diag(1/sqrt(ev$values))%*%t(ev$vectors)
num = xspec$n.used;  nfreq = length(xspec$freq)
specenv = matrix(0, nfreq, 1); beta = matrix(0,nfreq,3)
for (k in 1:nfreq){
  ev = eigen(2*Q%*%fxxr[,,k]%*%Q/num)
  specenv[k] = ev$values[1]
  b = Q%*%ev$vectors[,1]
  beta[k,] = b/b[1]  }
# output and graphics
frequency = xspec$freq
plot(frequency, 100*specenv, type="l", ylab="Spectral Envelope (%)")
output = cbind(frequency, specenv, beta)
colnames(output) = c("freq","specenv","x", "|x|", "x^2")
round(output,4)  # results (not shown)
# plot transformation
b = output[1, 3:5]; g = function(x) b[1]*x+b[2]*abs(x)+b[3]*x^2
curve(g, -.04, .04)
```

Example 7.20

```
set.seed(90210)
u = exp(3*sin(2*pi*1:500*.1) + rnorm(500,0,4))   # the data
spec.pgram(u, spans=c(5,3), taper=.5, log="dB")
dev.new()
x = cbind(u, sqrt(u), u^(1/3))           # transformation set
Var = var(x)
xspec = mvspec(x, spans=c(5,3), taper=.5)
fxxr = Re(xspec$fxx)
ev = eigen(Var)
Q = ev$vectors%*%diag(1/sqrt(ev$values))%*%t(ev$vectors)
num = xspec$n.used
nfreq = length(xspec$freq)
specenv = matrix(0,nfreq,1)
beta=matrix(0,nfreq,3)
 for (k in 1:nfreq){
  ev = eigen(2*Q%*%fxxr[,,k]%*%Q/num)
  specenv[k] = ev$values[1]
  b = Q%*%ev$vectors[,1]
  beta[k,] = b/sign(b[1]) }
# output and graphics
frequency = xspec$freq
plot(frequency, 100*specenv, type="l", ylab="Spectral Envelope (%)")
m = xspec$kernel$m
etainv = sqrt(sum(xspec$kernel[-m:m]^2))
thresh = 100*(2/num)*exp(qnorm(.999)*etainv)*rep(1,nfreq)
```

```
lines(frequency,thresh, lty="dashed", col="blue")
output = cbind(frequency, specenv, beta)
colnames(output) = c("freq","specenv","x", "sqrt(x)", "x^(1/3)")
round(output,4)
# plot transform
dev.new()
b = output[50,3:5]
g = function(x) 4.5 + b[1]*x+b[2]*sqrt(x)+b[3]*x^(1/3)
curve(g, 1, 4000)
lines(log(1:4000), lty=2)
```

[-]