R Commands for –
Advanced Linear Modeling III
This is a work in progress!
But it should be useful as is.
Contents

Preface ................................................................. 1
  0.1 Basic Information ........................................... 1
  0.2 Organization .................................................. 2

1 Nonparametric Regression ........................................ 3
  1.1 Linear Approximations ....................................... 3
  1.2 Simple Nonparametric Regression ............................ 3
  1.3 Estimation ....................................................... 3
    1.3.1 Polynomials ................................................ 3
    1.3.2 Cosines ..................................................... 5
    1.3.3 Haar wavelets .............................................. 6
    1.3.4 Cubic Splines .............................................. 8
    1.3.5 Orthogonal Series Estimation ......................... 9
  1.4 Variable Selection ............................................ 9
  1.5 Heteroscedastic Simple Nonparametric Regression ........ 9
  1.6 Approximating functions with Small Support .............. 9
    1.6.1 Polynomial Splines ...................................... 9
    1.6.2 Fitting Local Functions ................................ 13
    1.6.3 Local regression ......................................... 13
  1.7 Nonparametric Multiple Regression .......................... 13
    1.7.1 Redefining $\phi$ and the Curse of Dimensionality ... 14
    1.7.2 Reproducing Kernel Hilbert Space Regression .... 14
  1.8 Testing Lack of Fit in Linear Models ....................... 15
  1.9 Regression Trees .............................................. 16
    1.9.1 Bagging and Boosting ................................... 18
  1.10 Density Estimation ........................................... 20

2 Penalized Regression .............................................. 21
  2.1 Introduction .................................................. 21
  2.2 Ridge Regression .............................................. 21
  2.3 Lasso Regression .............................................. 23
11 Generalized Multivariate Linear Models .................. 63
  11.1 Generalized Multivariate Linear Models ............... 63
  11.2 Generalized least squares analysis ................... 63
  11.3 MACOVA analysis .................................. 64
  11.4 Rao’s Simple Covariance Structure ................... 67
  11.5 Longitudinal Data .................................. 67
  11.6 Generalized Split Plot Models ......................... 67
  11.7 Functional Data .................................... 67

12 Discrimination and Allocation ............................ 69
  12.1 The General Allocation Problem ....................... 70
    12.1.1 Figure 2 ..................................... 70
    12.1.2 One dimensional plots .......................... 71
  12.2 Estimated Allocation and Quadratic Discriminant Analysis .... 72
  12.3 Linear Discriminant Analysis: LDA .................... 73
  12.4 Cross-validation ................................... 73
  12.5 Discussion ........................................ 74
  12.6 Stepwise discriminant analysis ....................... 74
  12.7 Discrimination Coordinates ........................... 74
    12.7.1 Discrimination plot ............................ 75
    12.7.2 Discrimination Coordinates ..................... 76

13 Binary Discrimination and Regression .................... 79
  13.1 Binomial Regression ................................ 79
    13.1.1 Data Augmentation Ridge Regression ............. 79
  13.2 Binary Prediction .................................. 79
  13.3 Generalized Linear Models ........................... 81
  13.4 Best Prediction and Probability Estimation .......... 82
  13.5 Linear Prediction Rules ............................. 82
  13.6 Support Vector Machines ............................. 85
  13.7 Binary Discrimination ............................... 85
    13.7.1 Linearly Inseparable Groups: Logistic discrimination, LDA, and SVM .... 85
    13.7.2 Quadratically Separable Groups: Logistic discrimination, QDA, and SVM .. 89
  13.8 A simple example not in ALMIII ....................... 93

14 Principal Components, Classical Multidimensional Scaling, and Factor Analysis .................. 99
  14.1 Theory ............................................. 99
    14.1.1 Normal Density Plot for PA-V .................. 100
  14.2 Sample Principal Components ........................ 101
    14.2.1 Using Principal Components .................... 102
  14.3 Multidimensional Scaling ............................ 103
    14.3.1 Classical MDS ................................. 103
14.3.2 Nonmetric MDS ................................................. 105
14.3.3 Example 14.3.2 ............................................. 107
14.4 Factor Analysis .................................................. 110
  14.4.1 Terminology and Applications ......................... 110
  14.4.2 Maximum Likelihood Theory ......................... 111
  14.4.3 .............................................................. 112

15 Other Multivariate Analysis Topics .......................... 115
Preface

0.1 Basic Information

This online book is an R companion to *Advanced Linear Modeling*, Third Edition (*ALM-III*). It presupposes that the reader is already familiar with the basics of R. In particular it presupposes that the reader is already familiar with downloading R, plotting data, reading data files, transforming data, basic housekeeping, loading R packages, specifying basic linear models, and basic commands for performing matrix algebra. That is the material in Chapters 1, 3, and the beginning of 11 of my *Preliminary Version of R Commands for Analysis of Variance, Design, and Regression: Linear Modeling of Unbalanced Data* (R-code for *ANREG-II*), which is available at [http://www.stat.unm.edu/~fletcher/Rcode.pdf](http://www.stat.unm.edu/~fletcher/Rcode.pdf). Two other tools that I have found very useful are Tom Short’s R Reference card, [http://cran.r-project.org/doc/contrib/Short-refcard.pdf](http://cran.r-project.org/doc/contrib/Short-refcard.pdf) and Robert I. Kabacoff’s Quick-R website, [http://www.statmethods.net/index.html](http://www.statmethods.net/index.html).


Since the point is to do *Advanced Linear Modeling*, it is not surprising that the R commands needed are often not intrinsic to R, but are often relegated to R libraries. These libraries will be discussed as needed in the chapters but two packages that are of quite general use include *car* (regression algorithms due to Fox, Weisberg, and associates) and *MASS* (algorithms due to Venables, Ripley, and associates). By google-ing “r library(name)” you can access pdf files that both give information on the packages and give details on who was nice enough to provide these tools.

On November 8, 2018, to install *forecast* I had to right click the R icon and find a way to run R as an administrator. I have never had to do that before to install a package. There is more information on dealing with this in the other document. As of December 18, 2018 all of the packages discussed are listed below.

```r
  car
  DAAG
```
R has available several overviews to subject matters, a number of which are relevant to *ALM-III*. These are available at
https://cran.r-project.org/web/views/
and will be mentioned as appropriate.

Information about all available packages is available by going to the R site:
https://cran.r-project.org/web/packages/  
I use the terms “package” and “library” interchangeably (outside of programming R).

I seem to find myself spending a lot of time googling R commands and packages.

### 0.2 Organization

The chapter and section structure of this guide mimics the chapter and section structure of *ALM-III*. 
Chapter 1
Nonparametric Regression

1.1 Linear Approximations

No computing.

1.2 Simple Nonparametric Regression

No computing.

1.3 Estimation

The big new computing issue with linear approximation nonparametric regression is creating the model matrix. There is a good chance that you could find R packages to help with these tasks but doing it by brute force increases the chances that you know what you are doing.

1.3.1 Polynomials

We fit the polynomial model necessary for obtaining ALM-III Figure 1.1.

    rm(list = ls())
    battery <- read.table(
      "C:\E-drive\Books\LINMOD23\DATA\ALM1-1.dat",
      header=TRUE,
      sep="\n")#,col.names=c("Case","y","t","x")
    attach(battery)
battery

xb=mean(x)
xc=x-xb

poly = lm(y ~ xc+I(xc^2)+I(xc^3)+I(xc^4))
polys = summary(poly)
polys
anova(poly)

xx=seq(0,1,.01)
yy1= (14.5804 + 7.673 *(xx-.5) - 63.424 *(xx-.5)^2 -
 25.737 *(xx-.5)^3 + 166.418 *(xx-.5)^4)

plot(x,y,type="p")
lines(xx,yy1,type="l")

Fig. 1.1 Cosine model
1.3.2 Cosines

This produces *ALM-III* Figures 1.2 and 1.3 by picking “p” appropriately. Currently has $p = 6$.

```r
rm(list = ls())
battery <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM1-1.dat", header=TRUE, sep="\")#,col.names=c("Case","y","t","x"))
attach(battery)
battery

nf=length(x)
p=6
Phi=matrix(seq(1:(nf*p)),nf)
for(k in 1:p)
{
  Phi[,k]=cos(pi*k*x)
  #S=sin(pi*k*x)
}
#Phi

cos = lm(y ~ Phi)
coss = summary(cos)
coss
anova(cos)
Bhat=coefficients(cos)

xx=seq(0,1,.01)
nff=length(xx)
Phinew=matrix(seq(1:(nff*(p+1))),nff)
#Phinew
for(k in 1:(p+1))
{
  Phinew[,k]=cos(pi*(k-1)*xx)
  #S=sin(pi*k*xx)
}
Phinew

yy1=Phinew%*%Bhat
```
plot(x,y,type="p")
lines(xx,yy1,type="l")
par(mfrow=c(1,1))

# FOR SINES AND COSINES, CODE HAS NOT BEEN RUN
Phi=matrix(seq(1:nf*2*p),p*2)
for(k in 1:p)
  {r=k
   Phi[2r]=cos(pi*k*x)
   Phi[2r+1]=sin(pi*k*x)
  }

1.3.3 Haar wavelets

rm(list = ls())
battery <- read.table(  "C:\E-drive\Books\LINMOD23\DATA\ALM1-1.dat",
  header=TRUE,
  sep=""#,col.names=c("Case","y","t","x"))
attach(battery)
battery

nf=length(x)
# p determines the highest level of wavelets used, i.e., m_{pk}
p=5
pp=nf*(2^p-1)
Phi=matrix(seq(1:pp),nf)
#Phi is the Haar wavelet model matrix WITHOUT a column of 1s

Phi[,1]=((as.numeric(x<=.5)-as.numeric(x>.5))
 *as.numeric(x<=1)*as.numeric(x>0)) #m_{1}
Phi[,2]=((as.numeric(x<=.25)-as.numeric(x>.25))
 *as.numeric(x<=.5)*as.numeric(x>0)) #m_{21}
Phi[,3]=((as.numeric(x<=.75)-as.numeric(x>.75))
 *as.numeric(x<=1)*as.numeric(x>.5)) #m_{22}

# this defines the rest
for(f in 3:p){
  for(k in 1:(2^(f-1)))
  {
    a=(2*k-2)/2^f
  }
1.3 Estimation

\[ b = \frac{(2k-1)}{2^f} \]
\[ c = \frac{2k}{2^f} \]
\[ kk = 2^{(f-1)} + k \]
\[ \Phi[kk] = \left( (\text{as.numeric}(x \leq b) - \text{as.numeric}(x > b)) \right. \\
\left. \times \text{as.numeric}(x \leq c) \times \text{as.numeric}(x > a) \right) \]

\[ \Phi \]

Haar = lm(y ~ Phi)
Haars = summary(Haar)
Haars
anova(Haar)
Bhat = coef(Haar)

# Plotting the fitted values for Haar wavelets.
xx = seq(0, 1, .01)
nff = length(xx)
ppf = nff * (2^p)
Phinew = matrix(seq(1:ppf), nff)

Phinew[, 1] = 1
Phinew[, 2] = (as.numeric(xx <= .5) - as.numeric(xx > .50)) \\
* as.numeric(xx <= 1) * as.numeric(xx > 0)
Phinew[, 3] = (as.numeric(xx <= .25) - as.numeric(xx > .25)) \\
* as.numeric(xx <= .5) * as.numeric(xx > 0)
Phinew[, 4] = (as.numeric(xx <= .75) - as.numeric(xx > .75)) \\
* as.numeric(xx <= 1) * as.numeric(xx > .5)

for (f in 3:p) {
    for (k in 1:2^{(f-1)}) {
        a = \frac{(2k-2)}{2^f} \\
        b = \frac{(2k-1)}{2^f} \\
        c = \frac{2k}{2^f} \\
        kk = 2^{(f-1)} + k \\
        Phinew[, kk] = \left( (\text{as.numeric}(xx \leq b) - \text{as.numeric}(xx > b)) \right. \\
        \left. \times \text{as.numeric}(xx \leq c) \times \text{as.numeric}(xx > a) \right) \\
    }
    Phinew
}

yy1 = Phinew %*% Bhat
1.3.4 Cubic Splines

For constructing splines from scratch, the R-code for *ANREG-II* has a section on splines. This is more systematic. There is another section later that discusses another option.

```r
rm(list = ls())
battery <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM1-1.dat", header=TRUE, sep=""#,col.names=c("Case","y","t","x"))
attach(battery)
battery

nf=length(x)
#nk is the number of interior knots
nk=30  # or 4
# Create a matrix the correct size for the predictor variables.
Phi=matrix(seq(1:(nf*(nk+3))),nf)
Phi
# Create matrix of predictor variables.
Phi[,1]=x
Phi[,2]=x^2
Phi[,3]=x^3

knot=seq(1:nk)/(nk+1)
knot
for(k in 1:nk)
{
  Phi[, (k+3)]=(as.numeric(x>knot[k])*(x-knot[k]))^3
}
Phi  #Phi differs from book. It has no column of 1s.

spln = lm(y~Phi)
splns = summary(spln)
splns
anova(spln)
Bhat=coefficients(spln)
```
\[ xx = \text{seq}(0, 1, .01) \]
\[ nff = \text{length}(xx) \]
\[ \text{Phinew} = \text{matrix}(\text{seq}(1: (nff \times (nk + 4))), nff) \]
\[ \text{Phinew}[1] = 1 \]
\[ \text{Phinew}[2] = xx \]
\[ \text{Phinew}[3] = xx^2 \]
\[ \text{Phinew}[4] = xx^3 \]
\[ \text{for}(k \text{ in } 1: nk) \]
\[ \{ kk = k + 4 \]
\[ \text{Phinew}[kk] = (\text{as.numeric}(xx > \text{knot}[k]) \times (xx - \text{knot}[k])^3 \]
\[ \} \]
\[ \text{Phinew} \]
\[ yy1 = \text{Phinew} \times \text{Bhat} \]
\[ \text{plot}(x, y, \text{type} = \text{"p")} \]
\[ \text{lines}(xx, yy1, \text{type} = \text{"l")} \]

### 1.3.5 Orthogonal Series Estimation

This is basic matrix manipulation stuff as discussed in the R code for ANREG-II.

### 1.4 Variable Selection

Just get the sequential sums of squares printed from \text{lm}

### 1.5 Heteroscedastic Simple Nonparametric Regression

### 1.6 Approximating-functions with Small Support

#### 1.6.1 Polynomial Splines

For constructing them from scratch, the R-code for ANREG-II has a section on splines.

A basis matrix for cubic splines can be obtained using \text{ns}(x, df = \text{NULL}, knots = \text{NULL}, intercept = \text{FALSE}, Boundary.knots = \text{range}(x))

An example in which polynomial splines agree with bsplines.
### 1.6.1.1 Regular spline model

```r
rm(list = ls())
battery <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM1-1.dat", header=TRUE, sep=""#, col.names=c("Case","y","t","x"))
attach(battery)
battery

nf=length(x)
# nk = number of interior knots
nk=40 # or 4
# d=dimension of poly
d=3
d=as.integer(d)
Phi=matrix(seq(1:(nf*(nk+d))),nf)
Phi
for(kk in 1:d)
{
  Phi[,kk]=x^kk
}

knot=seq(1:nk)/(nk+1)
knot
for(k in 1:nk)
{
  Phi[, (k+d)]=(as.numeric(x>knot[k])*(x-knot[k]))^d
}
Phi

spln = lm(y~Phi)
splns = summary(spln)
splns
anova(spln)
Bhat=coefficients(spln)

xx=seq(0,1,.01)
nff=length(xx)
Phinew=matrix(seq(1:(nff*(nk+d+1))),nff)
for(k in 1:(d+1))
{k=kk=k-1
Phinew[,k]=xx^kk

knot=seq(1:nk)/(nk+1)
knot
for(k in 1:nk)
{
  Phi[, (k+d)]=(as.numeric(x>knot[k])*(x-knot[k]))^d
}
Phi

spln = lm(y~Phi)
splns = summary(spln)
splns
anova(spln)
Bhat=coefficients(spln)

xx=seq(0,1,.01)
nff=length(xx)
Phinew=matrix(seq(1:(nff*(nk+d+1))),nff)
for(k in 1:(d+1))
{k=kk=k-1
Phinew[,k]=xx^kk
```


1.6 Approximating-functions with Small Support

for(k in 1:nk)
{kk=k+d+1
Phinew[,kk]=(as.numeric(xx>knot[k])*(xx-knot[k]))^d
}
Phinew
yy1=Phinew%*%Bhat
plot(x,y,type="p")
lines(xx,yy1,type="l")

1.6.1.2 B-splines

B-splines for \( d = 2, 3. \)

```r
rm(list = ls())
battery <- read.table(
  "C:\E-drive\Books\LINMOD23\DATA\ALM1-1.dat",
  header=TRUE,
  sep="",#col.names=c("Case","y","t","x"))
attach(battery)
battery

nf=length(x)
#nk = m = number of interior knots +1
nk=31 # or 4
# d=dimension of poly
d=3
d=as.integer(d)
Phi=matrix(seq(1:(nf*(nk+d))),nf)
for(k in 1:(nk+d))
{
  j=k-1
  u=nk*x-j+d
  # d=2 mother
  Psi2 = (as.numeric(u >= 0))*(as.numeric(u <= 1))*(u^2/2) -
         (as.numeric(u > 1))*(as.numeric(u <= 2))*((u-1.5)^2-.75) +
         (as.numeric(u>2))*(as.numeric(u<=3))*((3-u)^2/2)
  # d=3 mother
  Psi3 = (as.numeric(u >= 0))*(as.numeric(u <= 1))*(u^3/3) +
         (as.numeric(u > 1))*(as.numeric(u <= 2))*(-u^3+4*u^2-4*u+4/3) -
         (as.numeric(u>2))*(as.numeric(u<=3)) *(-(-4-u)^3+4*(4-u)^2-4*(4-u)+4/3)+
```

(as.numeric(u>3))*(as.numeric(u<=4))*((4-u)^3/3)
Phi[,k]=Psi3 # currently using d=3

bspln = lm(y~Phi-1)
bsplns = summary(bspln)
bsplns
anova(bspln)
Bhat=coefficients(bspln)

# setup for plotting fitted curve
xx=seq(0,1,.01) # .01 determines grid
nff=length(xx)
# define matrix dimensions
Phinew=matrix(seq(1:(nff*(nk+d))),nff)
for(k in 1:(nk+d))
{
  j=k-1
  u=nk*xx-j+d
  Psi2 = (as.numeric(u >= 0))*(as.numeric(u <= 1))*(u^2/2) +
  (as.numeric(u > 1))*(as.numeric(u <= 2))*((u-1.5)^2-.75) +
  (as.numeric(u>2))*(as.numeric(u<=3))*((3-u)^2/2)
  Psi3 = (as.numeric(u >= 0))*(as.numeric(u <= 1))*(u^3/3) +
  (as.numeric(u > 1))*(as.numeric(u <= 2))*(-u^3+4*u^2-4*u+4/3) +
  ((as.numeric(u>2))*(as.numeric(u<=3)) * (-4*u^3+4*(4-u)^2-4*(4-u)+4/3) +
  (as.numeric(u>3))*(as.numeric(u<=4)) * ((4-u)^3/3)
  Phinew[,k]=Psi3
}
yy1=Phinew%*%Bhat
plot(x,y,type="p")
lines(xx,yy1,type="l")

The package splines2 supposedly constructs b-spline model matrices. I have
not checked.

Plot of \( \Psi_2 \)

xx=seq(0,1,.01) # .01 determines grid
u=xx*.5
Psi2 = (as.numeric(u >= 0))*(as.numeric(u <= 1))*(u^2/2) -
  (as.numeric(u > 1))*(as.numeric(u <= 2))*((u-1.5)^2-.75) +
  (as.numeric(u>2))*(as.numeric(u<=3))*((3-u)^2/2)
plot(u,Psi2,type="l")

Plot of \( \Psi_3 \)
1.7 Nonparametric Multiple Regression

```r
xx = seq(0, 1, .01)  # .01 determines grid
u = xx * 5 - .5
Psi3 = (as.numeric(u >= 0)) * (as.numeric(u <= 1)) * (u^3 / 3) + 
      (as.numeric(u > 1)) * (as.numeric(u <= 2)) * (-u^3 + 4*u^2 - 4*u + 4/3) + 
      ((as.numeric(u > 2)) * (as.numeric(u <= 3)) * 
       (- (4-u)^3 + 4*(4-u)^2 - 4*(4-u) + 4/3)) + 
      (as.numeric(u > 3)) * (as.numeric(u <= 4)) * ((4-u)^3 / 3)
plot(u, Psi3, type = "l")
```

1.6.2 Fitting Local Functions

1.6.3 Local regression

The default `loess` fit seems to oversmooth the data. It gives $R^2 = 0.962$.

```r
rm(list = ls())
battery <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM1-1.dat", header = TRUE, sep = "", col.names = c("Case", "y", "t", "x"))
attach(battery)
battery
lr = loess(y ~ x)
summary(lr)
xx = seq(0, 1, .01)  # .01 determines grid
lr2 = predict(lr, data.frame(x = xx))
plot(x, y)
lines(xx, lr2, type = "l")
(cor(y, lr$fit))^2
```

The package `np` does kernel smoothing.

1.7 Nonparametric Multiple Regression

The package `gam` fits generalized additive models using smoothing splines or local regression.
1.7.1 Redefining φ and the Curse of Dimensionality

1.7.2 Reproducing Kernel Hilbert Space Regression

Example 1.7.1. I fitted the battery data with the R language’s \texttt{lm} command using the three functions $R(u,v) = (u'v)^4$, $R(u,v) = (1 + u'v)^4$, and $R(u,v) = 5(7 + u'v)^4$. I got identical fitted values $\hat{y}_i$ to those from fitting a fourth degree polynomial.

```r
rm(list = ls())
battery <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM1-1.dat", header=TRUE, sep="\")#,col.names=c("Case","y","t","x"))
attach(battery)
battery

# obtain the X data matrix from the untransformed data
lin=lm(y ~ x)
X <- model.matrix(lin)

# fit the 4th degree polynomial regression 4 equivalent ways
xb=mean(x)
xc=x-xb
poly = lm(y ~ xc+I(xc^2)+I(xc^3)+I(xc^4))
polys = summary(poly)
polys
anova(poly)
Phi <- model.matrix(poly)
Z=Phi[, -1]
poly2 = lm(y ~ Z)
poly3 = lm(y ~ Phi - 1)
poly4 = lm(y ~ (Phi%*%t(Phi))-1)

# create polynomial r.k. matrix Rtilde
d=4
nf=length(x)
Rtilde=matrix(seq(1:(nf*nf)),nf)

for(k in 1:nf)
{
  u=X[k,]
}
for (kk in 1:nf)
{
  v=X[kk,]
  # Rtilde[k,kk]=(1+t(u)%*%v)^d
  # Rtilde[k,kk]=exp(-1000*t(u-v)%*%(u-v))
  Rtilde[k,kk]=tanh(1* (t(u)%*%v) +0)
}

# Fit the RKHS regression and compare fitted values
poly5 = lm(y ~ Rtilde-1)
# or lm(y ~ Rtilde)
poly5$fit
poly$fit
poly2$fit
poly3$fit
poly4$fit

# Check if Rtilde is nonsingular
eigen(Rtilde)$values
Rtilde%*%solve(Rtilde)

tanhh = lm(y ~ Rtilde-1)
summary(tanhh)

xx=seq(0,1,.01)
yy1=tanhh$coef[1] * tanh(1*(1+xx*x[1])+0) + tanhh$coef[2] * tanh(1*(1+xx*x[2])+0) + tanhh$coef[3] * tanh(1*(1+xx*x[3])+0) + tanhh$coef[4] * tanh(1*(1+xx*x[4])+0) + tanhh$coef[11]*tanh(1*(1+xx*x[11])+0) + tanhh$coef[16]*tanh(1*(1+xx*x[16])+0) + tanhh$coef[29]*tanh(1*(1+xx*x[29])+0) + tanhh$coef[41] * tanh(1*(1+xx*x[41])+0)
plot(x,y,type="p")
lines(xx,yy1,type="l")

See Chapter 3.

1.8 Testing Lack of Fit in Linear Models

Not much new computing here.
1.9 Regression Trees

Also known as *recursive partitioning*

Package and program *rpart*.

```r
coleman <- read.table(
  "C:\E-drive\Books\ANREG2\newdata\tab6-4.dat",
  sep="",col.names=c("School","x1","x2","x3","x4","x5","y"))
attach(coleman)
coleman

#install.packages("rpart")
library(rpart)
fit=rpart(y˜x3+x5,method="anova",control=rpart.control(minsplit=7))
# minsplit=7 means that if a partition set contains less than 7 observations
# a split will not be attempted
# I set it at 7 so as to get a split based on x5
# The default minimum number of observations in a partition set
# is minsplit/3 rounded off.
# The default minsplit is 20, so not very interesting for data with n=20.

printcp(fit)
plotcp(fit)
rsq.rpart(fit)
print(fit)
summary(fit)
plot(fit)
text(fit)
post(fit, file="C:\\E-drive\\Books\\ANREG2\\newdata\\colemantree.pdf")
cr
```

```r
par(mfrow=c(3,2))
plot(x3,x5,main="Recursive Partitioning")
lines(c(-11.35,-11.35),c(5,8),type="o",lty=1,lwd=1)
plot(x3,x5,main="Recursive Partitioning")
lines(c(-11.35,-11.35),c(5,8),type="o",lty=1,lwd=1)
lines(c(8.525,8.525),c(5,8),type="o",lty=1,lwd=1)
plot(x3,x5,main="Recursive Partitioning")
lines(c(-11.35,-11.35),c(5,8),type="o",lty=1,lwd=1)
lines(c(6.235,6.235),c(5,8),type="o",lty=1,lwd=1)
plot(x3,x5,main="Recursive Partitioning")
lines(c(-11.35,-11.35),c(5,8),type="o",lty=1,lwd=1)
lines(c(8.525,8.525),c(5,8),type="o",lty=1,lwd=1)
lines(c(12.51,12.51),c(5,8),type="o",lty=1,lwd=1)
```
1.9 Regression Trees

```r
lines(c(6.235, 6.235), c(5, 8), type="o", lty=1, lwd=1)
plot(x3, x5, main="Recursive Partitioning")
lines(c(-11.35, -11.35), c(5, 8), type="o", lty=1, lwd=1)
lines(c(8.525, 8.525), c(5, 8), type="o", lty=1, lwd=1)
lines(c(12.51, 12.51), c(5, 8), type="o", lty=1, lwd=1)
lines(c(6.235, 6.235), c(5, 8), type="o", lty=1, lwd=1)
lines(c(-11.35, 6.235), c(5.675, 5.675), type="l", lty=1, lwd=1)
plot(x3, x5, main="Alternative Second Partition")
lines(c(-11.35, -11.35), c(5, 8), type="o", lty=1, lwd=1)
lines(c(-11.35, 20), c(6.46, 6.46), type="l", lty=1, lwd=1)
par(mfrow=c(1,1))

# Summary tables
co <- lm(y ~ x1+x2+x3+x4+x5)
cop = summary(co)
cop
anova(co)
# The default for anova gives sequential sums of squares.
# The following device nearly prints out the three line ANOVA table.
# See Chapter 11 for details
Z <- model.matrix(co)[-1]
co <- lm(y ~ Z)
anova(co)

confint(co, level=0.95)

# Predictions
new = data.frame(x1=2.07, x2=9.99, x3=-16.04, x4= 21.6, x5=5.17)
predict(lm(y~x1+x2+x3+x4+x5), new, se.fit=T, interval="confidence")
predict(lm(y~x1+x2+x3+x4+x5), new, interval="prediction")

infv = c(y, co$fit, hatvalues(co), rstandard(co),
         rstudent(co), cooks.distance(co))
inftime = matrix(infv, ncol(cop$df[1]+cop$df[2]), 6, dimnames = list(NULL, c("y", "yhat", "lev", "r", "t", "C")))
inf

qqnorm(rstandard(co), ylab="Standardized residuals")
# Wilk-Francia
rankit= qnorm(ppoints(rstandard(co), a=1/3))
```
ys = sort(rstandard(co))
Wprime = (cor(rankit, ys))^2
Wprime

plot(co$fit, rstandard(co), xlab = "Fitted",
     ylab = "Standardized residuals", main = "Residual-Fitted plot")

Regression trees don’t seem to be very good without “bagging” them into random forests.
http://cran.r-project.org/web/packages/randomForest/index.html.
Useful commands? natural splines ns, modcv()

1.9.1 Bagging and Boosting

This section is not in ALM-III but discussed in PA-V, Section 14.4.

rm(list = ls())
nfull = 200
ntst = 100
n = nfull - ntst

Tst = rnorm(ntst, 0, 1)
Trn = rnorm(n, 0, 1)
xbar = mean(Trn)
xtd = median(Trn)
mid = (max(Trn) + min(Trn))/2

B = 1000
Bxbar = seq(1, B)
Bxtd = Bxbar
Bmid = Bxbar

for (k in 1:B)
{
    Temp = sample(Trn, n, replace = T)
    Bxbar[k] = mean(Temp)
    Bxtd[k] = median(Temp)
    Bmid[k] = (max(Temp) + min(Temp))/2
}

PredSS = seq(1:9)
PredSS[1] = sum((Tst - xbar)^2)/ntst
Regression Trees

$$\text{PredSS}[2] = \frac{\sum (Tst - xtld)^2}{ntst}$$

$$\text{PredSS}[3] = \frac{\sum (Tst - \text{mid})^2}{ntst}$$

$$\text{PredSS}[4] = \frac{\sum (Tst - \text{mean}(Bxbar))^2}{ntst}$$

$$\text{PredSS}[5] = \frac{\sum (Tst - \text{mean}(Bxtld))^2}{ntst}$$

$$\text{PredSS}[6] = \frac{\sum (Tst - \text{mean}(Bmid))^2}{ntst}$$

$$\text{PredSS}[7] = \frac{\sum (Tst - 0)^2}{ntst}$$

$$\text{PredSS}[8] = 8$$

$$\text{PredSS}[9] = 9$$

# PredSS = matrix(PredSS,3,3)

Simulation Program: For the Laplace distribution (and many other slightly unusual distributions) the package of repeated measures utilities `rmutil` is handy.

We generate \(n\) training observations and \(ntst\) test observations. Want to estimate the expected value of the distribution. Compute the sample mean, sample median, and sample midrange and compute the average of the bootstrap means, medians, and midranges as out estimates. Mean is optimal for normal and nonparametrically.

For normal data, sample mean is optimal and bootstrap cannot improve. For uniform data, midrange is optimal and bootstrap cannot improve. For laplace data, median is optimal and bootstrap cannot improve. (Nearly true for \(t(3)\).) For nonparametric data, sample mean is optimal. But bootstrap can improve the suboptimal estimates by averaging them.

The sample mean and the bootstrapped sample mean should be almost identical. See how close bootstrap is to optimal and how much better bootstrap is than suboptimal estimates.

```
rm(list = ls())
# set size of full data, test data, and implicitly training data.
nfull=200
ntst=100
n=nfull-ntst
# Define simulation size and bootstrap sample size.
SS=3000  # No. of data sets generated.
B=1000   # No. of boot samples from data

# Define sizes for vectors.
PredSS=seq(1:9)
APress=c(0,0,0,0,0,0,0,0,0)
Bxbar=seq(1,B)
Bxtld=Bxbar
Bmid=Bxbar

# Simulation
for(kk in 1:SS) {
```
# Generate data
Tst = rt(ntst, 3)
Trn = rt(n, 3)
#install.packages("rmutil")
#library(rmutil)
#Tst=rlaplace(ntst,0,1)
#Trn=rlaplace(n,0,1)
#Compute estimates.
xbar = mean(Trn)
xtd = median(Trn)
mid = (max(Trn)+min(Trn))/2

# Obtain estimates from bootstrapping
for (k in 1:B)
{
    Temp = sample(Trn, n, replace=T)
    Bxbar[k] = mean(Temp)
    Bxtd[k] = median(Temp)
    Bmid[k] = (max(Temp)+min(Temp))/2
}
# Prediction error variance for each estimate.
PredSS[1] = sum((Tst - xbar)^2)/ntst
PredSS[2] = sum((Tst - xtd)^2)/ntst
PredSS[3] = sum((Tst - mid)^2)/ntst
PredSS[4] = sum(((Tst - mean(Bxbar))^2)/ntst
PredSS[5] = sum(((Tst - mean(Bxtd))^2)/ntst
PredSS[6] = sum(((Tst - mean(Bmid))^2)/ntst
PredSS[7] = sum(((Tst - 0)^2)/ntst
APress = APress + PredSS
}
APress = APress/SS
matrix(APress, 3, 3)

1.10 Density Estimation
Chapter 2
Penalized Regression

2.1 Introduction

libraries
  lasso2 program l1ce
  glmnet library and program

2.2 Ridge Regression

Below are programs for fitting the augmented linear model. One might also look at lm.ridge in Venable and Ripley’s MASS package.

    rm(list = ls())
    battery <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM1-1.dat", header=TRUE, sep="")
    attach(battery)
    battery
    nf=length(x)
    p=10
    Phi=matrix(seq(1:(nf*(p+1))),nf)

    for(k in 1:(p+1))
      {Phi[,k]=cos(pi*(k-1)*x)
      #S=sin(pi*k*x)
      }
This section fits the p-1=6 model.

```r
Phi6 = Phi[, c(2, 3, 4, 5, 6, 7)]

cos6 = lm(y ~ Phi6)
coss6 = summary(cos6)
coss6
anova(cos6)
Bhat6 = coefficients(cos6)
Bhat6 = c(Bhat6, 0, 0, 0, 0)
Bhat6
```

Create and fit the augmented Ridge model

```r
w = seq(1:10)
zip = w - w
w = w
w = w * w
Dw = .2 * diag(w)
Dwx = cbind(zip, Dw)
Dwx
XR = rbind(Phi, Dwx)
XR
YR = c(y, zip)
cosr = lm(YR ~ XR - 1)
BhatR = coefficients(cosr)
BhatR

(cor(y, Phi %*% Bhat6)^2)
(cor(y, Phi %*% BhatR)^2)

xx = seq(0, 1, .01)
nff = length(xx)
Phinew = matrix(seq(1: (nff * (p + 1))), nff)
for (k in 1:11)
{
Phinew[, k] = cos(pi * (k - 1) * xx)
#S = sin(pi * k * xx)
}
Phinew

yy1 = Phinew %*% Bhat6
yyr = Phinew %*% BhatR

plot(x, y, type = "p")
lines(xx, yy1, type = "l", lty = 5)
It is most natural to apply ridge regression (and the lasso) to predictor variables that have been standardized by subtracting their mean and dividing by their standard deviation. If we create a matrix of the predictor variables, R has a command `scale` that does that.

```r
rm(list = ls())
coleman <- read.table("C:\E-drive\Books\ANREG2\newdata\tab6-4.dat", sep="",col.names=c("School","x1","x2","x3","x4","x5","y"))
attach(coleman)
coleman
X = coleman[,2:6]
Xs = scale(X)
```

Or you could do the same thing by brute force using the matrix commands of Chapter 11.

```r
X=matrix(c(x1-mean(x1),x2-mean(x2),x3-mean(x3),x4-mean(x4),x5-mean(x5),ncol=5)
Xs = X %*% diag(c(sd(x1),sd(x2),sd(x3),sd(x4),sd(x5))^{-1})
Xs
```

If you want, you could only center the variables or rescale them without centering them.

```r
scale(coleman[,2:6], center = TRUE, scale = TRUE)
```

### 2.3 Lasso Regression

You can get a lot information on lasso from Rob Tibshirani's website: [http://statweb.stanford.edu/~tibs/lasso.html](http://statweb.stanford.edu/~tibs/lasso.html). There is another example in the R code for Section 10.2 of ANREG-II. This is for comparing the $p = 30$ cosine lasso fit with the $p = 6$ cosine least squares fit.

```r
rm(list = ls())
battery <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM1-1.dat", header=TRUE, sep="",col.names=c("Case","y","t","x"))
attach(battery)
battery

nf=length(x)
p=30
```
Phi=matrix(seq(1:(nf*(p+1))),nf)
for(k in 1:(p+1))
{
Phi[,]=cos(pi*(k-1)*x)
#S=sin(pi*k*x)
}

#This section fits the p-1=6 model.
Phi6=Phi[,c(2,3,4,5,6,7)]
cos6 = lm(y˜Phi6)
coss6 = summary(cos6)
coss6
anova(cos6)
Bhat6=coefficients(cos6)
Bhat6=c(Bhat6,rep(0,24))
Bhat6

cos = lm(y˜Phi[-1])
coss = summary(cos)
coss
anova(cos)
Bhat=coefficients(cos)

# install.packages("lasso2")
library(lasso2)
tib <- l1ce(y ˜ Phi[-,1],data=battery,standardize=FALSE,bound=.5)
# This is the default boundary.
# Generalized lasso requires weights=
BhatL=coef(tib)
#tib <- l1ce(y ˜ Phi[-,1],bound = 0.5+(seq(1:10)/100))
tib
BhatL

(cor(y,Phi%*%Bhat6)^2)
(cor(y,Phi%*%Bhat)^2)
(cor(y,Phi%*%BhatL)^2)

xx=seq(0,1,.01)
nff=length(xx)
Phinew=matrix(seq(1:(nff*(p+1))),nff)
#Phinew
2.5 Another Approach

```r
for(k in 1:(p+1))
{
Phinew[,k]=cos(pi*(k-1)*xx)
#S=sin(pi*k*xx)
}
#Phinew
yy6=Phinew%*%Bhat6
yyL=Phinew%*%BhatL
yy30=Phinew%*%Bhat

plot(x,y,type="p")
lines(xx,yy6,type="l",lty=5)
lines(xx,yy30,type="l",lty=4)
lines(xx,yyL,type="l")
par(mfrow=c(1,1))

2.4 Bayesian Connections

2.5 Another Approach

#install.packages("ellipse") #Do this only once on your computer
library(ellipse)
#SHRINKAGE, NO ZEROS

b1=1
b2=2
A = matrix(c(1,.9,.9,2),2,2, dimnames = list(NULL, c("b1", "b2")))
E <- ellipse(A, centre = c(b1, b2), t = .907, npoints = 100)
E1 <- ellipse(A, centre = c(b1, b2), t = .5, npoints = 100)
x=seq(0,1,.05)
y=1-x
ty1=-1+x
x1=-x
y2=1+x1
y3=-1-x1
plot(E,type = ’l’,ylim=c(-1,3),xlim=c(-1,3),
     xlab=expression(˚beta[1]),
     ylab=expression(˚beta[2]),main=expression(˚delta==1))
text((b1+.1),(b2-.15),expression(hat(˚beta)),lwd=1,cex=1)
#plot(E,type = ’l’,ylim=c(-1,3),xlim=c(-1,3), #xlab=expression(beta[1]),ylab=expression(beta[2]))
lines(E1,type="l",lty=1)
```
26  2  Penalized Regression

```r
lines(x,y,type="l",lty=1)
lines(x,y1,type="l",lty=1)
lines(x1,y2,type="l",lty=1)
lines(x1,y3,type="l",lty=1)
lines(0,0,type="p",lty=3)
lines(b1,b2,type="p",lty=3)
#text((b1+.1),(b2-.15),(b1,b2),lwd=1,cex=1)
```
Chapter 3
Reproducing Kernel Hilbert Spaces

(I think this was all written by Al, but I’m not positive.)
Chapter 4
Covariance Parameter Estimation

Most of the software available is for more specific models than considered in this chapter.

It is clear from ALM-III Chapter 4 that you cannot perform covariance estimation without being able to do generalized least squares estimation. The R package \texttt{nlme} has an option \texttt{gls} for doing generalized least squares (GLS) and also fits a variety of covariance structures. The library \texttt{MASS} also has a program \texttt{ls.gls} for doing GLS. The library \texttt{lme4} (due to Bates \cite{bates2011nlme}, not Norman) and associates) does many things better than the older \texttt{nlme} but according to the 2016 online documentation, \texttt{lme4} does not currently offer the same flexibility as \texttt{nlme} for composing complex variance-covariance structures" nor does it seem to have an option for GLS.

4.1 Generalized Least Squares

Consider a random walk. Start with independent variables $w_1, \ldots, w_{10}$ with $E(w_j) = \mu$ and $\text{Var}(w_j) = \sigma^2$. From these define $y_1 = w_1$ and $y_i = w_1 + \cdots + w_i$. Now suppose you want to estimate $\mu$. If you can see the $w_i$, you just average them. If you see the $y_i$s, you can reconstruct the $w_i$s as $w_1 = y_1$, $w_2 = y_2 - y_1$, $w_3 = y_3 - y_2$, etc. It is easy to see that $\bar{w} = y_{10}/10$. Linear model theory will give the same estimate.

$$E(y_i) = i\mu; \quad \text{Var}(y_i) = i\sigma^2; \quad \text{Cov}(y_i, y_{i'}) = \min(i, i')\sigma^2,$$

so with $Y = (y_1, \ldots, y_{10})'$

$$E(Y) = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ 10 \end{bmatrix} \mu; \quad \text{Cov}(Y) = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 2 & \cdots & 2 \\ 1 & 2 & 3 & \cdots & 3 \\ 1 & 2 & 3 & 4 & \cdots & 4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 2 & 3 & 4 & \cdots & 10 \end{bmatrix}.$$
Do an example to show that this generalized least squares estimate give the sample mean \( \bar{w} = \frac{y_{10}}{10} \).

```r
# At some point you need to have run install.packages("MASS")
library(MASS)

lm.gls(formula, data, W, subset, na.action, inverse = FALSE,
method = "qr", model = FALSE, x = FALSE, y = FALSE,
contrasts = NULL, ...)

W=weight matrix, inverse=false
Chapter 5
Mixed Models

The library \texttt{lme4} has a command \texttt{lmer}. The package \texttt{mnle} has \texttt{lme}. Also see http://cran.r-project.org/doc/Rnews/Rnews_2005-1.pdf and https://cran.r-project.org/web/packages/lmm/index.html.
Chapter 6
Frequency Domain

An overview of R packages for time series analysis is available at https://cran.r-project.org/web/views/TimeSeries.html.

```
rm(list = ls())
coal.production <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM6-1.dat", sep="",col.names=c("Year","Wt"))
attach(coal.production)
coal.production

#par(mfrow=c(2,1))
plot(Year,Wt,type="l")
# plot.ts(Wt) works almost the same
```

6.2 Basic Data Analysis

6.2.1 Periodogram

Periodogram from the definition using orthogonality.

```
rm(list = ls())
coal.production <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM6-1.dat", sep="",col.names=c("Year","Wt"))
attach(coal.production)

y=Wt
n=length(y)
tt=seq(1,n)
```
nf=floor((n-1)/2)
P.lm=seq(1,nf)
freq=P.lm/n

for(k in 1:nf)
{
C=cos(2*pi*k*tt/n)
S=sin(2*pi*k*tt/n)
P.lm[k]= ((sum(C*y))^2 + (sum(S*y))^2)/n
}
Ptable=cbind(freq,P.lm)
Ptable

An alternative way to find the periodogram is to use the fast Fourier transform (which involves complex numbers). The following code, performed after the previous code, shows that the results are identical for the coal production data.

pp = abs(fft(y)/sqrt(n))^2
P.fft = pp[2:(nf+1)]
q=spec.pgram(y,taper=0,detrend=FALSE,fast=FALSE)
P.spec=q$spec[1:nf]
f.spec=q$freq[1:nf]
FFTtable=cbind(freq,P.lm,P.fft,f.spec,P.spec)
FFTtable

### 6.2.2 Figures

**ALM-III** Figure 6.2

rm(list = ls())
coal.production <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM6-1.dat", sep="",col.names=c("Year","Wt"))
attach(coal.production)
coal.production

y=Wt
#PGM(y) = function(y)
n=length(y)
tt=seq(1,n)
 nf=floor((n-1)/2)
P.lm=seq(1,nf)
freq=P.lm/n
par(mfrow=c(3,2))
par(mfrow=c(1,1))
ttt=seq(100,(n*100))/100
c1t=cos(2*pi*1*ttt/n)
c15t=cos(2*pi*15*ttt/n)
c30t=cos(2*pi*30*ttt/n)
s1t=sin(2*pi*1*ttt/n)
s15t=sin(2*pi*15*ttt/n)
s30t=sin(2*pi*30*ttt/n)
c1 = cos(2*pi*1*tt/n)
s1 = sin(2*pi*1*tt/n)
c30 = cos(2*pi*30*tt/n)
s30 = sin(2*pi*30*tt/n)
c15 = cos(2*pi*15*tt/n)
s15 = sin(2*pi*15*tt/n)

par(mfrow=c(3,2))
plot(Year,c1,type="p",pch=16,ylab="")
mtext("cos(2*pi*t*1/n)", side=2, line=2.2, cex=1.1)
lines(ttt+19,clt)
plot(Year,s1,type="p",pch=16,ylab="")
mtext("sin(2*pi*t*1/n)", side=2, line=2.2, cex=1.1)
lines(ttt+19,slt)
plot(Year,c15,type="p",pch=16,ylab="")
mtext("cos(2*pi*t*15/n)", side=2, line=2.2, cex=1.1)
lines(ttt+19,c15t)
plot(Year,s15,type="p",pch=16,ylab="")
mtext("sin(2*pi*t*15/n)", side=2, line=2.2, cex=1.1)
lines(ttt+19,s15t)
plot(Year,c30,type="p",pch=16,ylab="")
mtext("cos(2*pi*t*30/n)", side=2, line=2.2, cex=1.1)
lines(ttt+19,c30t)
plot(Year,s30,type="p",pch=16,ylab="")
mtext("sin(2*pi*t*30/n)", side=2, line=2.2, cex=1.1)
lines(ttt+19,s30t)
par(mfrow=c(1,1))

**ALM-III** Figure 6.3.4

rm(list = ls())
periodograms <- read.table(
"C:\E-drive\Books\LINMOD23\DATA\ALM9-2.dat",
sep="",col.names=c("F1","P1","P2","F2","P3","F3"))
attach(periodograms)
```r
q1 = filter(P1, filter = c(.2, .2, .2, .2, .2))
q2 = filter(P2, filter = c(.2, .2, .2, .2, .2))
q3 = filter(P3, filter = c(.2, .2, .2, .2, .2))
P1 = log(P1)
P2 = log(P2)
P3 = log(P3)

par(mfrow = c(1, 1))
plot(F1, P1, type = "l", ylab = "Log Periodogram", xlab = "Frequency"
    , ylim = c(6, 12))
lines(F2, P2, type = "l", lty = 2)
lines(F3, P3, type = "l", lty = 5)
legend("topright", c("Computation", "Definition", "BMDP",
    "R default"), lty = c(NA, 1, 2, 5))

q1 = log(q1)
q2 = log(q2)
q3 = log(q3)
plot(F1, q1, type = "l", ylab = "Log Spectral Estimate", xlab = "Frequency",
    ylim = c(6, 12))
lines(F2, q2, type = "l", lty = 2)
lines(F3, q3, type = "l", lty = 5)
legend("topright", c("Computation", "Definition", "BMDP", "R default"),
    lty = c(NA, 1, 2, 5))
par(mfrow = c(1, 1))
```

### 6.2.3 Spectral Density

```r
rm(list = ls())
coal.production <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM6-1.dat",
    sep = "", col.names = c("Year", "Wt"))
attach(coal.production)
y = Wt
n = length(y)
tt = seq(1, n)
f = floor((n - 1) / 2)
P.1m = seq(1, f)
freq = P.1m / n
par(mfrow = c(3, 1))
f1 = spec.pgram(y, taper = 0, detrend = FALSE, fast = FALSE)
```
6.2 Basic Data Analysis

```r
f3 = spec.pgram(y, taper=0, detrend=FALSE, fast=FALSE, 
    kernel("daniell", 1))
f5 = spec.pgram(y, taper=0, detrend=FALSE, fast=FALSE, 
    kernel("daniell", 2))
FFTtable = cbind(freq, f1$spec, f3$spec, f5$spec)
FFTtable
```

```r
par(mfrow=c(1,1))
plot(freq, log(f1$spec), type="l", lty=2, 
    ylab="Log Density", xlab="Frequency", ylim=c(4,12))
lines(freq, log(f3$spec), type="l", lty=5)
lines(freq, log(f5$spec), type="l", lty=1)
legend("topright", c("Estimate", "P", "fhat_3", "fhat_5"), 
    lty=c(NA,2,5,1))
```

```r
spec.pgram(residuals(fit), taper=0, detrend=FALSE, fast=FALSE)
spec.pgram(residuals(fit), taper=0, detrend=FALSE, fast=FALSE, 
    kernel("daniell", 2))
```

### 6.2.4 Detrended Spectral Density

The two scripts give the same results. The first one relies on `spec.pgram` to detrend. The second one detrends by using `lm` to find the residuals.

```r
rm(list = ls())
coal.production <- read.table( 
    "C:\E-drive\Books\LINMOD23\DATA\ALM6-1.dat", 
    sep="", col.names=c("Year","Wt"))
attach(coal.production)
coal.production
y=Wt
par(mfrow=c(2,1))
#f1 = f3 = spec.pgram(y, taper=0, fast=FALSE)
#f3 = spec.pgram(y, taper=0, fast=FALSE, kernel("daniell", 1))
f5 = spec.pgram(y, taper=0, fast=FALSE, kernel("daniell", 2))
f5 = spec.pgram(y, taper=0, detrend=FALSE, 
    fast=FALSE, kernel("daniell", 2))
par(mfrow=c(1,1))
plot(fd5$freq, log(fd5$spec), type="l", lty=1, 
    ylab="Log Density", xlab="Frequency", ylim=c(4,12))
lines(fd5$freq, log(fd5$spec), type="l", lty=5)
legend("bottomleft", c("Estimate", "Detrended", "Regular"), 
    lty=c(NA,1,5))
```
fit = lm(Wt ~ Year)
y = residuals(fit)
# fr1 = spec.pgram(y, taper=0, detrend=FALSE, fast=FALSE)
# fr3 = spec.pgram(y, taper=0, detrend=FALSE, fast=FALSE, 
# kernel("daniell", 1))
fr5 = spec.pgram(y, taper=0, detrend=FALSE, fast=FALSE, 
# kernel("daniell", 2))
fd5$spec - fr5$spec

6.3 The Random Effects Model
6.4 The Measurement Error Model

EXAMPLE 6.5.1.

rm(list = ls())
coal.production <- read.table( 
"C:\\E-drive\\Books\\LINMOD23\\DATA\\ALM6-1.dat",
  sep = "", col.names = c("Year", "Wt"))
attach(coal.production)
coal.production
y = Wt
par(mfrow = c(2, 1))
# f1 = f3 = spec.pgram(y, taper = 0, fast = FALSE)
# f3 = spec.pgram(y, taper = 0, fast = FALSE, kernel("daniell", 1))
f5 = spec.pgram(y, taper = 0, fast = FALSE, kernel("daniell", 2))
f5 = spec.pgram(y, taper = 0, fast = FALSE, trend = FALSE, 
  kernel("daniell", 2))
coal.production
y=Wt
n=length(y)
tt=seq(1,n)
nf=floor((n-1)/2)
P.lm=seq(1,nf)
freq=P.lm/n
par(mfrow=c(3,1))
f1=spec.pgram(y,taper=0, detrend=FALSE, fast=FALSE)
f3=spec.pgram(y,taper=0, detrend=FALSE, fast=FALSE,
  kernel="daniell", 1))
f5=spec.pgram(y, taper=0, detrend=FALSE, fast=FALSE,
  kernel="daniell", 2))
FFTtable=cbind(freq,f1$spec,f3$spec,f5$spec)

par(mfrow=c(1,1))
plot(freq,log(f1$spec),type="l",lty=2,
  ylab="Log Density",xlab="Frequency",ylim=c(4,12))
lines(freq,log(f3$spec),type="l",lty=5)
lines(freq,log(f5$spec),type="l",lty=1)
legend("topright",c("Estimate","P","fhat_3","fhat_5"),
  lty=c(NA,2,5,1))

a=.05
r=1
fhat=f1$spec
q=length(freq)
cl=c(freq,fhat,2*r*fhat/ qchisq(1-a/2,2*r),
  2*r*fhat/ qchisq(a/2,2*r))
CIa = matrix(cl,q,4,dimnames = list(NULL,
  c("Freq", "fhat", "Lower Limit","Upper Limit")))
CIa

a=.05
r=3
fhat=f3$spec
q=length(freq)
cl=c(freq,fhat,2*r*fhat/ qchisq(1-a/2,2*r),
  2*r*fhat/ qchisq(a/2,2*r))
CIa = matrix(cl,q,4,dimnames = list(NULL,
  c("Freq", "fhat", "Lower Limit","Upper Limit")))
CIa

a=.05
r=5
fhat=f5$spec
q=length(freq)

ci=c(freq,fhat,2+r*fhat/ qchisq(1-a/2,2*r),
2+r*fhat/ qchisq(a/2,2*r))
CIa = matrix(ci,q,4,dimnames = list(NULL,
c("Freq", "fhat", "Lower Limit","Upper Limit")))
CIa

Using the previous results, we create ALM-III Figures 6.7 and 6.8.

q=length(freq)
r=1
f1sort = sort(f1$spec)
scores = qchisq(seq(1,q)/(q+1),2*r)
plot(scores,f1sort,xlab="Chi-square(2) Scores",
ylab="Order Statistics")

qq=q-2
f1sort=f1sort[1:qq]
scores = qchisq(seq(1,qq)/(qq+1),2*r)
plot(scores,f1sort,xlab="Chi-square(2) Scores",
ylab="Order Statistics",
main="One frequency deleted")

q=length(freq)
r=5
f5sort = sort(f5$spec)
scores = qchisq(seq(1,q)/(q+1),2*r)
plot(scores,f5sort,xlab="Chi-square(10) Scores",
ylab="Order Statistics",main="")

qq=q-4
f5sort=f5sort[1:qq]
scores = qchisq(seq(1,qq)/(qq+1),2*r)
plot(scores,f5sort,xlab="Chi-square(10) Scores",
ylab="Order Statistics")
6.4.1 Prediction

Work out the predictions. I’m going to compare the least squares predictions to the mixed model predictions.

```r
rm(list = ls())
coal.production <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM6-1.dat", sep="", col.names=c("Year","Wt"))
attach(coal.production)
coal.production

y=Wt
n=length(y)
tt=seq(1,n)
nf=floor((n-1)/2)
#Define predictors for measurement error model.
C1=cos(2*pi*1*tt/n)
S1=sin(2*pi*1*tt/n)
C2=cos(2*pi*2*tt/n)
S2=sin(2*pi*2*tt/n)
C4=cos(2*pi*4*tt/n)
S4=sin(2*pi*4*tt/n)
C5=cos(2*pi*5*tt/n)
S5=sin(2*pi*5*tt/n)
C6=cos(2*pi*6*tt/n)
S6=sin(2*pi*6*tt/n)
#Define their periodograms
P1= ((sum(C1*y))ˆ2 + (sum(S1*y))ˆ2)/n
P2= ((sum(C2*y))ˆ2 + (sum(S2*y))ˆ2)/n
P4= ((sum(C4*y))ˆ2 + (sum(S4*y))ˆ2)/n
P5= ((sum(C5*y))ˆ2 + (sum(S5*y))ˆ2)/n
P6= ((sum(C6*y))ˆ2 + (sum(S6*y))ˆ2)/n

Yearfuture=c(81,82,83,84,85,86,87)
Wtfuture=c(818.4,833.5,777.9,891.8,879.0,886.1,912.7)
ttt=Yearfuture-19

yt=Wtfuture
nt=length(yt)
C1t=cos(2*pi*1*ttt/n)
S1t=sin(2*pi*1*ttt/n)
C2t=cos(2*pi*2*ttt/n)
S2t=sin(2*pi*2*ttt/n)
```

```
\[ C_{4t} = \cos(2\pi \cdot 4 \cdot \frac{ttt}{n}) \]
\[ S_{4t} = \sin(2\pi \cdot 4 \cdot \frac{ttt}{n}) \]
\[ C_{5t} = \cos(2\pi \cdot 5 \cdot \frac{ttt}{n}) \]
\[ S_{5t} = \sin(2\pi \cdot 5 \cdot \frac{ttt}{n}) \]
\[ C_{6t} = \cos(2\pi \cdot 6 \cdot \frac{ttt}{n}) \]
\[ S_{6t} = \sin(2\pi \cdot 6 \cdot \frac{ttt}{n}) \]

\[ Z_{t} = \text{data.frame}(C1=C_{1t}, S1=S_{1t}, C2=C_{2t}, S2=S_{2t}, C4=C_{4t}, S4=S_{4t}, C5=C_{5t}, S5=S_{5t}, C6=C_{6t}, S6=S_{6t}) \]

# Least squares prediction
\[ \text{predLS} = \text{lm}(y \sim C1+S1+C2+S2+C4+S4+C5+S5+C6+S6) \]
\[ \text{anova(predLS)} \]
\[ \text{predL=summary(predLS)} \]
\[ \text{MSE=(predL}$\text{sigma}^2) \]
\[ \text{Wtfit=predict(predLS,interval="none")} \]
\[ \text{Wtpred=predict(predLS,Zt,interval="none",level=0.95)} \]

# Mixed model prediction
\[ \text{sigaa=}(P1+P2)/2 \]
\[ \text{siga=}(2/n)\ast\text{(sigaa-MSE)} \]
\[ \text{sigbb=}(P4+P5+P6)/3 \]
\[ \text{sigb=}(2/n)\ast\text{(sigbb-MSE)} \]

\[ \text{MSE} \]
\[ \text{sigaa} \]
\[ \text{siga} \]
\[ \text{sigbb} \]
\[ \text{sigb} \]

\[ Z = \text{model.matrix(predLS)}[,\text{-1}] \]

\[ \text{gammat=}(\text{siga/sigaa}\ast\text{sum(C1}\ast y)), (\text{siga/sigaa}\ast\text{sum(S1}\ast y)), (\text{siga/sigaa}\ast\text{sum(C2}\ast y)), (\text{siga/sigaa}\ast\text{sum(S2}\ast y)), (\text{sigb/sigbb}\ast\text{sum(C4}\ast y)), (\text{sigb/sigbb}\ast\text{sum(S4}\ast y)), (\text{sigb/sigbb}\ast\text{sum(C5}\ast y)), (\text{sigb/sigbb}\ast\text{sum(S5}\ast y)), (\text{sigb/sigbb}\ast\text{sum(C6}\ast y)), (\text{sigb/sigbb}\ast\text{sum(S6}\ast y)) \]

\[ Z_{tt} = \text{as.matrix}(Z_{t},c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)) \]
\[ \text{MMpred=}(Z_{tt} \ast\%\ast\%\text{gammat})+\text{mean(y)} \]
\[ \text{MMfit=(Z} \ast\%\ast\%\text{gammat})+\text{mean(y)} \]

\[ \text{plot(Year,Wt,type="l",xlim=c(20,87),ylim=c(300,925))} \]
\[ \text{lines(Yearfuture,Wtfuture,type="l")} \]
\[ \text{lines(Yearfuture,MMpred,type="l",lty=5)} \]
6.7 Fourier Analysis

```r
lines(Yearfuture, Wtpred, type="l", lty=4)
lines(Year, MMfit, type="l", lty=5)
lines(Year, Wtfit, type="l", lty=4)
legend("topleft", c("Actual", "Mixed Model", "Least Squares"), lty=c(1,5,4))
```

6.5 Linear filtering

```r
rm(list = ls())
coal.production <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM6-1.dat", sep="", col.names=c("Year", "Wt"))
attach(coal.production)
coal.production
y=Wt
# The first three filters are equivalent
RA5 = filter(y, filter=c(0.2,0.2,0.2,0.2,0.2))
RA5a = filter(y, filter=c(0.2,0.2,0.2,0.2,0.2), method = "convolution", sides=2)
RA5b = filter(y, rep(0.2, 5))
RA6 = filter(y, filter=c(1/12,1/6,1/6,1/6,1/6,1/6,1/12))
par(mfrow=c(2,1))
plot(Year, RA5, type="l", main="Low Pass Filter")
WtDiff = diff(Wt, 1)
plot(WtDiff, type="l", main="High Pass Filter")
par(mfrow=c(1,1))
```

6.5.1 Recursive filtering

```r
filter(x, filter=, method = "recursive"), init=)
```

6.6 The Coherence of Two Time Series

6.7 Fourier Analysis

We have already been using the FFT as a data analytic tool.
6.8 Exercise Data Plots

```r
rm(list = ls())
y <- as.vector(t(as.matrix(read.table("C:\E-drive\Books\LINMOD23\DATA\ALM9-6.dat", fill=TRUE))))
y = y[!is.na(y)]
C:\E-drive\Books\LINMOD23\DATA\ALM9-7.dat
```
Chapter 7
Time Domain

An overview of R packages for time series analysis is available at https://cran.r-project.org/web/views/TimeSeries.html.

The package written go go along with Shumway and Stoffer (2011), https://cran.r-project.org/web/packages/astsa/astsa.pdf, has a variety of capabilities and data including Kalman filter capabilities. (I was not aware of it when I was doing the non-Kalman filter parts of this chapter.)

While the analysis of the coal data is consolidated in Subsection 7.6.3 of the book, we distribute the computations though the earlier sections.

7.1 Correlations

rm(list = ls())
coil.production <- read.table("C:\E\drives\Books\LINMOD23\DATA\ALM6-1.dat",
sep="",col.names=c("Year","Wt"))
attach(coil.production)
coil.production

# Figs 7-1
par(mfrow=c(2,1))
Wacf=acf(Wt,ylim=c(-1,1))
Wacf
Wpacf=pacf(Wt,ylim=c(-1,1))
Wpacf
par(mfrow=c(1,1))

# Fig 7-3
WTDiff = diff(Wt,1)
plot(WTDiff,type="l")
# Fig 7-4
par(mfrow=c(2,1))
WtDiffacf = acf(WtDiff,ylim=c(-1,1))
WtDiffacf
WtDiffpacf = pacf(WtDiff,ylim=c(-1,1))
WtDiffpacf
par(mfrow=c(2,1))

7.5 Estimation

The R programs arima and Arima use an ARIMA parameterization where 
the moving average coefficients are the negatives of the ones used in ALM-III! 
We will use the R library forecast by Rob Hyndman and coauthors. The use 
of Arima, Acf, and Pacf; as opposed to arima, acf, and pacf; requires the 
library forecast.

On November 8, 2018, to install forecast I had to right click the R icon and 
find a way to run R as an administrator. I have never had to do that before to install 
an package.

We illustrate fitting an ARIMA(1,1,2) model with a constant using maximum 
likelihood.

rm(list = ls())
coal.production <- read.table(
"C:\E-drive\Books\LINMOD23\DATA\ALM6-1.dat",
    sep="",col.names=c("Year","Wt"))
attach(coal.production)
coal.production

#install.packages("forecast")
library(forecast)

MO=c(2,1,1)
fit = Arima(Wt,order=MO,include.constant=TRUE,method = "ML")
fit

Est=coef(fit)
SE=sqrt(diag(fit$var.coef))
Tratio=Est/SE
Tabcd = c(Est,SE,Tratio)
TabCoef
7.5 Estimation

# Variance estimate
SSE = t(fit$residuals) %*% fit$residuals
# Eliminate "+1" if "constant=FALSE"

# Correlation Matrix

# Figs 7-5
par(mfrow=c(2,1))
Acf(residuals(fit), ylim=c(-1,1))
Pacf(residuals(fit), ylim=c(-1,1))
par(mfrow=c(1,1))
# Results not shown
# normal plot, periodogram
# fhat_5 spectral density
qqnorm(residuals(fit), ylab="Wt residuals")
spec.pgram(residuals(fit), taper=0, detrend=FALSE, fast=FALSE)
spec.pgram(residuals(fit), taper=0, detrend=FALSE, fast=FALSE, kernel="daniell", 2))

# Fig 7-6
plot(Year, residuals(fit), type="b")

# Fig 7-7

fitpred = forecast(fit, 7)
fitpred

Yearfuture = c(81, 82, 83, 84, 85, 86, 87)
Wtfuture = c(818.4, 833.5, 777.9, 891.8, 879.0, 886.1, 912.7)
plot(Year, Wt, type="l", xlim=c(20, 87), ylim=c(300, 925))
lines(Yearfuture, Wtfuture, type="l")
lines(Yearfuture, fitpred$mean, type="l", lty=5)
lines(Year, fit$fit, type="l", lty=5)
legend("topleft", c("Actual", "ARIMA"), lty=c(1, 5))
7.5.1 Ljung-Box

MO=c(2,1,1)
fit = Arima(Wt,order=MO),include.constant=TRUE,method = "ML")
fit
Box.test(residuals(fit),lag=5, type = c("Ljung-Box"), fitdf = MO[1]+MO[3]+1)
Box.test(residuals(fit),lag=10, type = c("Ljung-Box"), fitdf = MO[1]+MO[3]+1)
Box.test(residuals(fit),lag=15, type = c("Ljung-Box"), fitdf = MO[1]+MO[3]+1)
Box.test(residuals(fit),lag=20, type = c("Ljung-Box"), fitdf = MO[1]+MO[3]+1)

For testing the residuals of a ARIMA(p,d,q) set MO=c(p,d,q) and perform the test with Box.test(x, type = c("Ljung-Box"), fitdf = MO[1]+MO[3])

7.5.2 General Commands

fit <- arima(myts, order=c(p, d, q))
arima(x, order = c(0L, 0L, 0L),
    seasonal = list(order = c(0L, 0L, 0L), period = NA),
    xreg = NULL, include.mean = TRUE,
    transform.pars = TRUE,
    fixed = NULL, init = NULL,
    method = c("CSS-ML", "ML", "CSS"), n.cond,
    SSinit = c("Gardner1980", "Rossignol2011"),
    optim.method = "BFGS",
    optim.control = list(), kappa = 1e6)
lag(ts, k) lagged time series, shifted back k observations
diff(ts, differences=ddifference the time series d times
ndiffs(ts) differences required to stationarity (forecast)

adf.test(ts) #Augemented Dickey-Fuller test.
#Rejecting the null suggests time series is stationary (tseries)
Box.test(x, type="Ljung-Box")
    test that observations in vector or time series x are independent

Note that the forecast package has somewhat nicer versions of
acf() and pacf() called Acf() and Pacf() respectively.
# fit an ARIMA model of order P, D, Q
fit <- arima(myts, order=c(p, d, q))
# predictive accuracy
7.6 Model Selection

Remember that for a time series $y_1, \ldots, y_n$, the time series being analyzed in an \textit{ARIMA}$(p,d,q)$ model has only $n - d$ observations.

```r
rm(list = ls())
coal.production <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM6-1.dat", sep="", col.names=c("Year","Wt"))
attach(coal.production)
coal.production

#install.packages("forecast")
library(forecast)
diffs(Wt)  # Automated determination of differencing
# Assumes stationarity, min diff that is consistent
# Uses KPSS test

# Fitting the ARIMA models
# to get AIC, AICc, and BIC
MD=c(0,1,0)
#MD=c(1,1,1)
#MD=c(2,1,2)
#MD=c(3,1,3)
#MD=c(2,1,1)
#MD=c(1,1,2)
#MD=c(1,1,2)
fitted <- Arima(Wt, order=MD, include.constant=TRUE, method = "ML")
#seasonal=list(order=c(0,0,0),period=7)
```

```r
library(forecast)
accuracy(fit)

# predict next 5 observations
library(forecast)
forecast(fit, 5)
plot(forecast(fit, 5))

library(forecast)
# Automated forecasting using an exponential model
fit <- ets(myts)
```
7 Time Domain

```r
fit
fit$sigma2
fit$aic # redundant calculation below
fit$loglik
n=length(Wt)-MD[2]
AAIC=-2*(fit$loglik)+2*kk
BBIC=-2*(fit$loglik)+log(n)*(kk)
AAICc=AAIC+2*(kk)*((kk+1)/(n-kk-1))
AAIC
BBIC
AAICc

#MD=c(1,1,2)
#MD=c(0,1,2)
fit = Arima(Wt,order=MD,include.constant=FALSE,method ="ML")
fit
n=length(Wt)-MD[2]
AAIC=-2*(fit$loglik)+2*kk
BBIC=-2*(fit$loglik)+log(n)*(kk)
AAICc=AAIC+2*(kk)*((kk+1)/(n-kk-1))
AAIC
BBIC
AAICc

This process can be automated by fit=auto.arima(Wt,d=1,D=0,max.p=3,max.q=3,max.order=6,stepwise=FALSE,ic="aic")

7.7 Seasonal Adjustment

```r
rm(list = ls())
WIHospData <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM7-3.dat",sep="",col.names=c("WIHosp"))
attach(WIHospData)

#install.packages("forecast")
library(forecast)
nfdiffs(WIHosp) # Automated determination of differencing
# Assumes stationarity, min diff that is consistent
# 7.8 The Multivariate State-Space Model and the Kalman Filter

The package written go go along with Shumway and Stoffer (2011), [https://cran.r-project.org/web/packages/astsa/astsa.pdf](https://cran.r-project.org/web/packages/astsa/astsa.pdf), has Kalman filter capabilities.

An overview of R procedures for spatial analysis is available at https://cran.r-project.org/web/views/Spatial.html. One for spatio-temporal analysis is at https://cran.r-project.org/web/views/SpatioTemporal.html.

- spatial package has surf.gls which fits polynomials using GLS and the Gaussian, exponential, or spherical covariance functions. Predictions with predict.
- geoR package has krig.conv for conventional kriging and krig.bayes for Bayesian kriging.
- gstat package, see programs krig gstat predict
- RandomFields
Chapter 9
Multivariate Linear Models: General

There is no data analysis in this chapter, but there are some computation issues worth addressing.

9.1 Generating Multivariate Normals

Randomly generate observations from a $N(\mu, \Sigma)$.

```r
#install.packages("mvtnorm")
library(mvtnorm)
rmvnorm(n,mu,Sigma)
```

9.2 Specifying Multivariate Data Matrices

We illustrate different ways of creating $Y = [Y_1, \ldots, Y_q]$ after reading in the $Y$s.

cbind combines variables of the same length into a matrix.

```r
rm(list = ls())
resp <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM10-2.dat", sep="", col.names=c("Y1","Y2","Y3","Y4","Drug"))
attach(resp)
resp
Y=cbind(Y1,Y2,Y3,Y4)
Y
```

matrix forms a matrix out of a string of numbers. To use it, we need to create a string out of our variables and then reform them into a matrix.
Finally, within various programs (like those for discriminant analysis, principal components, or factor analysis) we can specify a data matrix in a manner similar to specifying a model (matrix)

\[ \hat{Y} = \text{matrix}(c(Y_1, Y_2, Y_3, Y_4), n, 4) \]

I don’t think this will work in either `lm` or `glm` to specify the dependent variable matrix. And if you are copying R code from a pdf file into R, “tilde”

\[ \hat{Y} = \text{matrix}(c(Y_1, Y_2, Y_3, Y_4), n, 4) \]

\[ \hat{Y} = \text{matrix}(c(Y_1, Y_2, Y_3, Y_4), n, 4) \]

\[ \hat{Y} = \text{matrix}(c(Y_1, Y_2, Y_3, Y_4), n, 4) \]

will often copy incorrectly so that you need to delete the copied version of tilde and retype it.
Chapter 10
Multivariate Linear Models: Applications

An overview of R procedures for multivariate analysis is available at https://cran.r-project.org/web/views/Multivariate.html.

10.1 One-sample

The only test, and the test we want, is the test of the (Intercept) term.

```r
rm(list = ls())
resp <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM10-1.dat", sep="",col.names=c("x2","x5"))
attach(resp)
resp
# transform x2 and subtract hypothesized means
r2=sqrt(x2)-sqrt(50)
r5=x5-6
rr=cbind(r2,r5)
full <- lm(rr ~ 1)
anova(full,test = c("Wilks"))
# all 4 tests give same result
#anova(full,test = c("Roy"))
#anova(full,test = c("Hotelling-Lawley"))
#anova(full) # Default is Pillai
```

10.2 Two-samples

This works just like the next section. I cannot find the data for this example.
10.3 One-Way MANOVA and Profile Analysis

Prior to the subsection on profile analysis, we will discuss how to construct MANOVA variables from a data file constructed for doing a split plot analysis. The data file used here was constructed for doing MANOVA

```r
rm(list = ls())
resp <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM10-2.dat", sep="",col.names=c("yy1","yy2","yy3","yy4","Drug"))
attach(resp)
resp
y=cbind(yy1,yy2,yy3,yy4)
y #y is a matrix with individual dependent variables as columns.
#Summary tables
D=factor(Drug)
re <- lm(y ~ D)
coef=summary(re)
coef
anova(re,test = c("Wilks"))
anova(re,test = c("Roy"))
anova(re,test = c("Hotelling-Lawley"))
anova(re) # Default is Pillai
anova(re,test = c("Spherical")) #Similar to split plot analysis

# Nicer output comes from the "car" package.
library(car)
re.manova=Manova(re)
summary(re.manova)
# To obtain E and H
E=re.manova$SSPE
H=re.manova$SSP

par(mfrow=c(2,2))
qqnorm(rstudent(re)[,1],ylab="Standardized residuals y1")
qqnorm(rstudent(re)[,2],ylab="Standardized residuals y2")
qqnorm(rstudent(re)[,3],ylab="Standardized residuals y3")
qqnorm(rstudent(re)[,4],ylab="Standardized residuals y4")
ys = yy1+yy2+yy3+yy4
yss=lm(ys ~ D-1)
par(mfrow=c(1,1))
qqnorm(rstudent(yss),ylab="Standardized residuals of sum")
```
To turn split plot data into MANOVA data identify the factor variable that determines the multiple observations. Here the time variable \( t \) determines the multiple observations for MANOVA

```r
\begin{verbatim}
rm(list = ls())
abraid <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM10-2s.dat", sep="",col.names=c("yy","d","t"))
attach(abraid)
abraid

yy1=yy[t==1]
yy2=yy[t==2]
yy3=yy[t==3]
yy4=yy[t==4]
Drug=d[t==1]
ym=cbind(yy1,yy2,yy3,yy4)
\end{verbatim}
```

There can be no missing values in the subplots or there will be missing values in the MANOVA. When specifying the whole plot effects, the drugs here, it does not matter which dependent variable you specify, e.g., \( \text{Drug}=d[t==2] \) works just as well.

### 10.3.1 Profile Analysis

Plot the profiles. This presumes the commands previously discussed.

```r
#Profiles
rep <- lm(y ~ D-1)
rep$coef #These are just all the mean values
par(mfrow=c(1,1))
time=c(1,2,3,4)
plot(time,rep$coef[1,],xlab="Times",ylab="Drug-Time Means",type="o",ylim=c(70,85),lty=1)
```
In profile analysis, we first want to test if all the curves are parallel, i.e., have no interaction. **If they are parallel** it makes sense to test whether the curves are all horizontal, i.e., have no Time effects and if the parallel curves sit on top of each other, i.e., have no Drug effects. Similar things can be done with a split-plot analysis, but a split-plot makes stronger assumptions.

```r
rm(list = ls())
resp <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM10-2.dat",
sep="",col.names=c("yy1","yy2","yy3","yy4","Drug"))
attach(resp)
resp
D=factor(Drug)
sum=yy1+yy2+yy3+yy4
d2=yy2-yy1
d3=yy3-yy1
d4=yy4-yy1
dd=cbind(d2,d3,d4)

# interaction/parallelism multivariate tests
inter = lm(dd~D)
anova(inter,test = c("Wilks"))
anova(inter,test = c("Roy"))
anova(inter,test = c("Hotelling-Lawley"))
anova(inter)  # Default is Pillai
  # The tests for D are the tests for parallelism
  # The tests for (Intercept) are the tests for Time effects, i.e., horizontal profiles given parallel.

# Alternative method of getting test for Time main effect
full <- lm(dd ~ 1)
red <- lm(dd ~ -1)
anova(inter,full,red,test=c("Wilks"))

# Drug main effect test, i.e.,
# test for all only one curve, given parallel
whole = lm(sum ~ D)
anova(whole)
```
### 10.3.2 Split Plot Analysis

This gives the split-plot ANOVA table. It allows us to test Drug by Time interaction, i.e., parallelism. If the curves are parallel, it allows us to test for no Time main effects, i.e., the curves are all horizontal and for no Drug effects, i.e., the parallel curves sit on top of each other.

```r
rm(list = ls())
abraid <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM10-2s.dat", sep="",col.names=c("y","d","t"))
attach(abraid)
abraid
summary(abraid)
rp=rep(seq(1:10),12)
rp
Drug=factor(d)
Time=factor(t)
RP=factor(rp)

bsfp <- aov(y ~ Drug + Error(RP:Drug) + Time + Drug:Time)
summary(bsfp)
```

I doubt that this would work for unbalanced MANOVA data.

To turn MANOVA data into split plot data haven’t tried this Assume a data matrix y that is \( n \times q \) and a factor \( d \).

```r
q=4
n=30
yy=matrix(y,nrows=q*n,ncols=1)
dd=rep(d,q)
```
Chapter 11
Generalized Multivariate Linear Models

An overview of R procedures for multivariate analysis is available at https://cran.r-project.org/web/views/Multivariate.html.

11.1 Generalized Multivariate Linear Models

11.2 Generalized least squares analysis

```r
rm(list = ls())
resp <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM10-2.dat", sep="",col.names=c("yy1","yy2","yy3","yy4","Drug"))
attach(resp)
resp
Y=cbind(yy1,yy2,yy3,yy4)
Y #y is a matrix with individual dependent variables as columns.

zz=c(1,1,1,1,2,7,12,17,4,49,144,289)
Z = matrix(zz,4,3)
Z
W1=Z%*%solve(t(Z)%*%Z)
W1
Ytilde1=Y%*%W1

#Summary tables
D=factor(Drug)
re <- lm(Ytilde1 ~ D-1)
rep=summary(re)
rep
```
# Nicer output comes from the "car" package.
library(car)
re.manova=Manova(re)
summary(re.manova)
# To obtain E and H
Etilde1=re.manova$SSPE
Stilde1=Etilde1/re$df.residual
Stilde1
Htilde1=re.manova$SSP

Ytilde10=Ytilde1[,1]
ytilde11=Ytilde1[,2]
Ytilde12=Ytilde1[,3]

The things that have been left out are univariate analyses.

### 11.3 MACOVA analysis

**Example 11.3.1**

```r
rm(list = ls())
resp <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM10-2.dat", sep="",col.names=c("yy1","yy2","yy3","yy4","Drug"))
attach(resp)
resp
Y=cbind(yy1,yy2,yy3,yy4)
Y #y is a matrix with individual dependent variables as columns.

zz=c(1,1,1,1,2,7,12,17,4,49,144,289)
Z = matrix(zz,4,3)
Z
W1=Z%*%solve(t(Z)%*%Z)
W1
W2=c(-1,3,-3,1)
W=cbind(W1,W2)
W
Ytilde=Y%*%W
Ytilde1=Y%*%W1
Ytilde2=Y%*%W2
```
**#Summary tables**

```r
D = factor(Drug)
re <- lm(Y_{tilde}1 ~ D + Y_{tilde}2 - 1)
rep = summary(re)
rep
re$coef
```

# Nicer output comes from the "car" package.

```r
library(car)
re.manova = Manova(re)
summary(re.manova)
```

# To obtain E and H

```r
E_{tilde} = re.manova$SSPE
S_{tilde} = E_{tilde}/re$df.residual
E_{tilde}
H_{tilde} = re.manova$SSP
```

**#Summary tables**

```r
D = factor(Drug)
re0 <- lm(Y_{tilde}1 ~ Y_{tilde}2)
rep0 = summary(re0)
rep0
re0$coef
```

```r
re0.manova = Manova(re0)
summary(re0.manova)
```

# To obtain E and H

```r
E_{tilde0} = re0.manova$SSPE
S_{tilde} = E_{tilde}/re$df.residual
E_{tilde0}
H_{tilde} = E_{tilde0} - E_{tilde}
```

**Example 11.3.2**

```r
rm(list = ls())
resp <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM10-2.dat", sep="", col.names=c("yy1","yy2","yy3","yy4","Drug"))
attach(resp)
resp
Y = cbind(yy1, yy2, yy3, yy4)
Y # y is a matrix with individual dependent variables as columns.
```
zz=c(1,1,1,1,2,7,12,17,4,49,144,289)
Z = matrix(zz,4,3)
Z
W1=Z%*%solve(t(Z)%*%Z)
W1
W2=c(-1,3,-3,1)
W=cbind(W1,W2)
W
Ytilde=Y%*%W
Ytilde1=Y%*%W1
Ytilde2=Y%*%W2

#Summary tables
D=factor(Drug)
re <- lm(Ytilde1 ~ D + Ytilde2 - 1)
rep=summary(re)
rep
re$coef

# Nicer output comes from the "car" package.
library(car)
re.manova=Manova(re)
summary(re.manova)
# To obtain E and H
Etilde=re.manova$SSPE
Stilde=Etilde/re$df.residual
Etilde
Htilde=re.manova$SSP

#Summary tables
D=factor(Drug)
re0 <- lm(Ytilde1 ~ D - 1)
rep0=summary(re0)
rep0
re0$coef

re0.manova=Manova(re0)
summary(re0.manova)
# To obtain E and H
Etilde0=re0.manova$SSPE
Stilde=Etilde1/re$df.residual
11.7 Functional Data

\[ \tilde{E} \]
\[ \tilde{H} = \tilde{E}^2 - \tilde{E} \]
\[ \tilde{H} \]
\[ \text{library(psych)} \]
\[ \text{tr} (\tilde{H}) \]

11.4 Rao’s Simple Covariance Structure

11.5 Longitudinal Data

11.6 Generalized Split Plot Models

11.7 Functional Data
Chapter 12
Discrimination and Allocation

An overview of R procedures for multivariate analysis is available at https://cran.r-project.org/web/views/Multivariate.html.

**EXAMPLE 12.0.1.** Read in the data and plot the data as follows.

```r
rm(list = ls())
cush <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM12-1.DAT",
sep="",col.names=c("Type","Tetra","Preg"))
attach(cush)
cush
#summary(cush)
TL = log(Tetra)
PL = log(Preg)
```

The plot from ANREG-II

```r
xx1=c( 3.1, 3.0, 1.9, 3.8, 4.1, 1.9)
yy1=c(11.70, 1.30, 0.10, 0.04, 1.10, 0.40)
xx2=c( 8.3, 3.8, 3.9, 7.8, 9.1, 15.4, 7.7, 6.5, 5.7, 13.6)
yy2=c(1.00, 0.20, 0.60, 1.20, 0.60, 3.60, 1.60, 0.40, 0.40, 1.60)
xx3=c(10.2, 9.2, 9.6, 53.8, 15.8)
yy3=c(6.40, 7.90, 3.10, 2.50, 7.60)
xl=log(xx1)
x2=log(xx2)
x3=log(xx3)
y1=log(yy1)
y2=log(yy2)
y3=log(yy3)
plot(x1,y1, pch=3, ylab="log(Pregnanetriol)",
```
12 Discrimination and Allocation

12.1 The General Allocation Problem

12.1.1 Figure 2

```r
rm(list = ls())
par(mfrow=c(2,1))
x=seq(-1,8,.1)
y=dnorm(x,2,1)
y1=dnorm(x,5,1)
plot(x,y,xlab="y",ylab="f(y)",type="l",lty=1,xlim=c(-1,8))
lines(x,y1,lty=1)
points( 3.5, 0, pch=16)

x=seq(-1,11,.01)
y=dnorm(x,2,1)
y1=dnorm(x,5,3)
plot(x,y,,ylab="f(y)",xlab="y",type="l",lty=1,xlim=c(-1,11))
lines(x,y1,lty=1)
c= 11
b= -26
a=8
delta=b^2-4*a*c
M1 = (-b+sqrt(delta))/(2*a)
M2 = (-b-sqrt(delta))/(2*a)
points( c(M1,M2),c(0,0), pch=16)
points( c(-.31,3.595),c(0,.0277))
lines( c(-.31,-.31),c(0,.114))
legend("topright",c("QDA","Mahalanobis"),pch=c(22,16))
```

12.1 The General Allocation Problem

12.1.2 One dimensional plots

```r
rm(list = ls())
par(mfrow=c(2,1))
x=seq(-1,8,.1)
y=dnorm(x,2,1)
y1=dnorm(x,5,1)
plot(x,y,type="l",lty=1,xlim=c(-1,8))
lines(x,y1,lty=1)
points(3.5,0,pch=16)

x=seq(-1,11,.01)
y=dnorm(x,2,1)
y1=dnorm(x,5,3)
plot(x,y,type="l",lty=1,xlim=c(-1,11))
lines(x,y1,lty=1)
c=11
b=-26
a=8
delta=b^2-4*a*c
M1 = (-b+sqrt(delta))/(2*a)
M2 = (-b-sqrt(delta))/(2*a)
points(c(M1,M2),c(0,0),pch=16)
points(c(-.31,3.595),c(0,.0277))
lines(c(3.595,3.595),c(0,.114))
legend("topright",c("QDA","Mahalanobis"),pch=c(22,16))
cbind(x,y,y1)
par(mfrow=c(1,1))
```

It is surprisingly fuzzy to transform from Cartesian to polar coordinates. I actually drew the plots by generating the data in polar coordinates and making the simpler transformation to Cartesian. The library `useful` has programs for doing both.

```r
rm(list = ls())
n1=200
n2=200
yt11=runif(n1,1.5,1.75)
yt12=runif(n1,0,2*pi)
yt21=runif(n2,2.25,2.45)
yt22=runif(n2,0,2*pi)
y11=yt11*sin(yt12)
y12=yt11*cos(yt12)
y21=yt21*sin(yt22)
y22=yt21*cos(yt22)
```
12.2 Estimated Allocation and Quadratic Discriminant Analysis

The following commands reproduce Table 12.3 associated with Example 12.5.1.

```r
rm(list = ls())
cush <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM12-1.DAT", sep="",col.names=c("Type","Tetra","Preg"))
attach(cush)
cush
#summary(cush)
TL = log(Tetra)
PL = log(Preg)
T=factor(Type)
library(MASS)
fit <- qda(T ~ TL + PL,prior=c(1/3,1/3,1/3))
fit
pfit=predict(fit)
pfit
t <- table(pfit$class,T)
t
fit2 <- qda(T ~ TL + PL,prior=c(1/3,1/3,1/3),CV=TRUE)
fit2
pfit2=predict(fit2)
pfit2
t2 <- table(fit2$class,T)
t2
```
12.3 Linear Discriminant Analysis: LDA

The following commands reproduce Table 12.2 associated with Example 12.5.1.

```r
rm(list = ls())
cush <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM12-1.DAT", sep="", col.names=c("Type","Tetra","Preg"))
attach(cush)
cush
#summary(cush)
TL = log(Tetra)
PL = log(Preg)
T=factor(Type)

library(MASS)
fit <- lda(T ~ TL + PL,prior=c(1/3,1/3,1/3))
fit
pfit=predict(fit)
pfit
t <- table(pfit$class,T)
t

fit2 <- lda(T ~ TL + PL,prior=c(1/3,1/3,1/3),CV=TRUE)
fit2
pfit2=predict(fit2)
pfit2
t2 <- table(fit2$class,T)
t2
```

12.4 Cross-validation

Commands for cross-validation were contained in the two previous sections. For $K$-fold cross validation look up the following which relies on the association between discrimination and one-way MANOVA as discussed in the next section.

```r
# K-fold cross-validation
library(DAAG)
cv.lm(df=mydata, fit, m=3) # 3 fold cross-validation
```
12.5 Discussion

No computing

12.6 Stepwise discriminant analysis

The results for Example 12.5.1 were given in Section 1 and 2. Here we only discuss the results in Example 12.2.2, which illustrates stepwise discriminant analysis.

Does the step command apply?

```r
rm(list = ls())
beetles <- read.table("C:\E-drive\Books\LINMOD23\DATA\Examp12-2-2.dat", sep="", col.names=c("y1", "y2", "y3", "y4", "y5", "y6", "s"))
attach(beetles)
beetles
Spec=factor(s)
r12=y1/y2
y=cbind(y1,y2,y3,y4,y5,y6,r12)

# None of the following?
beat <- lm(y ~ Spec)
anoa(full,test = c("Wilks"))
# all 4 tests give same result
#anova(beat,test = c("Roy"))
#anova(beat,test = c("Hotelling-Lawley"))
#anova(beat) # Default is Pillai
```

12.7 Discrimination Coordinates

```r
rm(list = ls())
resp <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM10-2.dat", sep="", col.names=c("yy1", "yy2", "yy3", "yy4", "Drug"))
attach(resp)
resp
D=factor(Drug)
library(MASS)
re <- lda(D ~ yy1+yy2+yy3+yy4)
re
```
rep = predict(re)
rep

# Eigenvectors are are not unique.
# The following commands transform the MASS
# output to agree with the book.
# It is not important to do this!
YA1 = (-1.9245)*rep$x[,1]
YA2 = (-1.9245)*rep$x[,2]

# Relabel X axis. Makes the plots look nicer.
Drr=c("Placebo","Placebo","Placebo","Placebo","Placebo",
   "Placebo","Placebo","Placebo","Placebo",
   "A","A","A","A","A","A","A","A","A",
   "B","B","B","B","B","B","B","B","B")
Dr=factor(Drr)

# Create plots
#dotchart(YA1,groups=Dr)
par(mfrow=c(1,2))
plot(Drug,YA1,xlab="Drug Index")
plot(Dr,YA1,ylab="YA1")
plot(Drug,YA2,xlab="Drug Index")
plot(Dr,YA2,ylab="YA1")
par(mfrow=c(1,1))
#dotchart(YA2,groups=Dr)
plot(YA2[D==1],YA1[D==1],pch=3,ylim=c(-9,8),xlim=c(-5,8),
   xlab="YA2",ylab="YA1")
lines(YA2[D==2],YA1[D==2],pch=16,type="p")
lines(YA2[D==3],YA1[D==3],pch=22,type="p")
legend("bottomright",c("Placebo","A","B"),pch=c(3,16,22))

An alternate way to do the linear discrimination.

y=cbind(yy1,yy2,yy3,yy4)
# a matrix with individual dependent variables as columns.
y
re <- lda(y,D)

### 12.7.1 Discrimination plot

library(ellipse)
12.7.2 Discrimination Coordinates

Copied from ANREG commands

Get $H$ and $E$ from a one-way manova. Perhaps from `fitted.values` or `residuals` from `lm` or

```
rm(list = ls())
b1=0
b2=0
d=1
b3=-.8320503*d
b4=.5547002*d
A = .5* matrix(c(1,1.2,1.2,2),2,2, dimnames = list(NULL, c("b1", "b2")))
E <- ellipse(A, centre = c(b1, b2), t = .95, npoints = 100)
E1 <- ellipse(A, centre = c(b3, b4), t = .95, npoints = 100)
```

The following plots 20 random data points over the ellipse.

```
library(mvtnorm)
T= rmvnorm(20,c(b1,b2),A)
plot(T,type = 'p',ylim=c(-1.5,2),xlim=c(-2,1.5),xlab=expression(y[1]),
ylab=expression(y[2]),main="Discriminant Analysis")
lines(T1,type="p",pch=4)
```
TT=rbind(T,T1)
Chapter 13
Binary Discrimination and Regression

In order to construct the figures, I had to fit the models. So the information on fitting
the models is embedded in the code for the figures.

13.1 Binomial Regression

\texttt{glm} fits the generalized linear models but it does not incorporate penalties except
through augmenting the data.

Library \texttt{LiblineaR} has program \texttt{LiblineaR} that performs logistic regression
with both $L^1$ (lasso) and $L^2$ (ridge) regularization (penalties).

13.1.1 Data Augmentation Ridge Regression

13.2 Binary Prediction

Library \texttt{LiblineaR} has program \texttt{LiblineaR} that performs Logistic and SVM
with both $L^1$ (lasso) and $L^2$ (ridge) regularization (penalties).

\textit{ALM-III} Figure 13.1 involves fitting logistic and probit regressions and an SVM
classifier. The linear structures are $\beta_0 + \beta_1 TL + \beta_2 PL$, i.e., linear in $TL$ and $PL$.

```R
rm(list = ls())
cush <- read.table(
"C:\E-drive\Books\LINMOD23\DATA\ALM12-1c.DAT",
    sep="",col.names=c("Type","Tetra","Preg"))
attach(cush)
cush
```
TL = log(Tetra)
PL = log(Preg)
Tp=Type-2
Tp2=3-Type
# Tp2 is 1 for Bilat hyp and 0 for Carcin

# Plot of points and discrimination curves
xx2=c( 8.3, 3.8, 3.9, 7.8, 9.1, 15.4, 7.7, 6.5, 5.7, 13.6)
yy2=c(1.00, .20, .60, 1.20, .60, 3.60, 1.60, .40, .40, 1.60)
xx3=c(10.2, 9.2, 9.6, 53.8, 15.8)
yy3=c(6.40, 7.90, 3.10, 2.50, 7.60)
x2=log(xx2)
x3=log(xx3)
y2=log(yy2)
y3=log(yy3)
plot(x2,y2, pch=16, ylab="log(Pregnanetriol)",
ylim=c(-2,3),xlim=c(1,4.5),xlab="log(Tetrahydrocortisone)")
points(x3, y3, pch=22)
legend("bottomright",c("Bilateral Hyperplasia","Carcinoma"),
pch=c(16,22))

# LOGISTIC REGRESSION
ac2 = glm(Tp2 ˜ TL + PL,family=binomial)
#summary(ac2)
#anova(ac2)
#post=c(Type,ac2$fit,1-ac2$fit)

# LOGISTIC REGRESSION
ac2 = glm(Tp2 ˜ TL + PL,family=binomial)
#summary(ac2)
#anova(ac2)

# Logistic Regression
ac2 = glm(Tp2 ˜ TL + PL,family=binomial)
#summary(ac2)
#anova(ac2)

# PROBIT REGRESSION
ac3=glm(Tp2 ˜ TL + PL,family=binomial(link="probit"))
b=ac3$coef
y3=(b[1]+b[2]*x1)/-b[3]
lines(x1,y3,type="1")
legend("topright",c("Logistic","Probit","SVM"),lty=c(1,2,5))

# SVM


13.3 Generalized Linear Models

This produces ALM-III Figures 13.2 and 13.4 that compares the logistic and probit loss functions and the logistic and SVM loss functions, respectively.

```r
rm(list = ls())
par(mfrow=c(1,2))
x=seq(-3.5,3.5,.05)
# Logit
y=2*log((1+exp(-x)))
# SVM
##y2=(1-x)*as.numeric((1-x)>0)
# Probit
y2=-2*log(pnorm(x))
# This gives HALF of a SVM
#y2=-2*log(1-pexp(-.5*x+.5))
plot(x,y,type="l",xlab=expression(x^T*beta),ylab="Loss")
lines(x,y2,,lty=5)
#legend("topright",c("Logistic","SVM"),lty=c(1,5))
legend("topright",c("Logistic","Probit"),lty=c(1,5))
legend("bottomleft",c("y=1"))
# Logit
y=-2*log(1/(1+exp(x)))
# SVM
##y4=(x+1)*as.numeric((x+1)>0)
# Probit
y4=-2*log(1-pnorm(x))
```
# This does NOT give the other half SVM
# \( y_4 = -2 \log(\text{pexp}(-0.5x+0.5)) \)
plot(x, y, type="l", xlab=expression(x^T*beta), ylab="Loss")
lines(x, y4,, lty=5)
#legend("topleft", c("Logistic","SVM"), lty=c(1,5))
legend("topleft", c("Logistic","Probit"), lty=c(1,5))
legend("bottomright", c("y=0"))
par(mfrow=c(1,1))

13.4 Best Prediction and Probability Estimation

13.5 Linear Prediction Rules

This 3-part program produces ALM-III Figures 13.3 and 13.5. They involves fitting logistic and probit regressions and an SVM classifier. The linear structures are quadratic in \( TL \) and \( PL \). Part 1 plots the data. Part 2 plots the two regressions. Part 3 fits the default SVM classifier and has commented out the SVM classifier with the tuning parameter reduced by a factor of 100 (cost increased to 100).

13.5.0.1 Plotting the data

rm(list = ls())
cush <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM12-1c.DAT", sep="", col.names=c("Type","Tetra","Preg"))
attach(cush)
cush
T = factor(Type)
TL = log(Tetra)
PL = log(Preg)
plot(TL[T==2], PL[T==2], pch=16, ylab="log(Pregnanetriol)", ylim=c(-2,3), xlim=c(1,4.5), xlab="log(Tetrahydrocortisone)"
points(TL[T==3], PL[T==3], pch=22)
legend("bottomright", c("Bilateral Hyperplasia","Carcinoma"), pch=c(16,22))
legend("topleft", c("Logistic","Probit","SVM"), lty=c(1,2,5))
13.5.0.2 Quadratic Logistic and Probit Regression

This is a continuation of the previous plotting program.

\[
\begin{align*}
&TL2 = TL \times TL \\
&PL2 = PL \times PL \\
&TPL = PL \times TL \\
&Tp = \text{Type-2} \\
&Tp2 = \text{3-Type} \\
# Tp2 is 1 for Bilat hyp and 0 for Carcin
\end{align*}
\]

```r
ac2 = glm(Tp2 ~ TL + PL + TL2 + PL2 + TPL,
          family=binomial)
```

```r
# solve quadratic equation
x1=seq(1,4.5,.1)
b=ac2$coef
prior=.5
c=bb[1]+bb[2]*x1+bb[4]*x1^2
b= bb[3] + bb[6]*x1
a=bb[5]
delta=b^2-4*a*c
yLR = (-b+sqrt(delta))/(2*a)
yLR2 = (-b-sqrt(delta))/(2*a)
lines(x1,yLR,type="l",lty=1)
lines(x1,yLR2,type="l",lty=1)
```

```r
ac3 = glm(Tp2 ~ TL + PL + TL2 + PL2 + TPL,
          family=binomial(link="probit"))
```

```r
# solve quadratic equation
x1=seq(1,4.5,.1)
b=ac3$coef
prior=.5
c=bb[1]+bb[2]*x1+bb[4]*x1^2
b= bb[3] + bb[6]*x1
a=bb[5]
delta=b^2-4*a*c
yPR = (-b+sqrt(delta))/(2*a)
yPR2 = (-b-sqrt(delta))/(2*a)
lines(x1,yPR,type="l",lty=1)
lines(x1,yPR2,type="l",lty=1)
```
13.5.0.3 Quadratic SVM

This is a continuation of the previous program. Typically you would want to have `scale=T` in `svm`, which is the default.

```r
#install.packages("e1071")
library(e1071)
T=factor(Type)
# define SVM and display outputs
fit <- svm(T ˜ TL + PL,kernel="polynomial",degree=2,
    gamma=1,coef0=1, scale=F)
# Next line reduces tuning parameter, i.e., increases cost.
#fit <- svm(T ˜ TL + PL,kernel="polynomial",degree=2,
    gamma=1,coef0=1, scale=F, cost=100)
fit$SV
fit$coefs
fit$rho
fit$fitted # shows groups that each case was allocated to
# fitted shows that my parabolic solution is reasonable.
# predict(fit)

# define and solve quadratic equation.
# add curves to previous plot
# must solve matrix equation that involves inner products
x1=seq(1,4.5,.01)
w=fit$coefs
C=-fit$rho + sum(w)+2*sum(w*fit$SV[,1])*x1
    +sum(w*(fit$SV[,1])^2)*x1^2
b=2*sum(w*fit$SV[,2]) + 2*sum(w*fit$SV[,1]*fit$SV[,2])*x1
a=sum(w*(fit$SV[,2])^2)

delta=b^2-4*a*C
ySVM = (-b+sqrt(delta))/(2*a)
ySVM2 = (-b-sqrt(delta))/(2*a)
lines(x1,ySVM,type="l",lty=5)
lines(x1,ySVM2,type="l",lty=5)

# A plot that svm provides
plot(fit,cush)
```

We now illustrate how to get the SVM prediction results from output (in R, `fit = svm(...)` that provides the matrix of \( d - 1 \) dimensional support vectors \( X_s \) (fit$SV) which is \( s \times (d-1) \) and extracted from \( X \), the negative intercept \( K \) (fit$rho), and the coefficients \( w \) (fit$coef) that are the nonzero coefficients of \( \frac{1}{2\epsilon}(\lambda'_1, -\lambda'_0) \). The elements of the vector \( w \) are positive for one group and negative
for the other. The two groups are identified as positive and negative. As in Chapter 3 of ALM-III, the kernel function (specified in the program) defines the inner product between two vectors. To predict the group of a new vector \( x \), evaluate the matrix expression

\[
\begin{bmatrix}
<x,x_{S1}> \\
\vdots \\
<x,x_{Ss}>
\end{bmatrix}w - K.
\]

The sign of result determines the group allocation. Note that this simplifies greatly when using the standard Euclidean inner product.

**Exercise:** Manually scale the variables, run `svm` with `scale=F` and compute \( \hat{\beta}_c \) and \( \hat{\beta}_0 \) as discussed in the book. Now run `svm` with `scale=T` and compute \( \hat{\beta}_c \) and \( \hat{\beta}_0 \). How do you transform \( \hat{\beta}_c \) and \( \hat{\beta}_0 \) computed with `scale=T` into an appropriate vector \( \hat{\beta} \) on the original scale of the data?

### 13.6 Support Vector Machines

The computations were made in the previous section. [http://svms.org/tutorials/](http://svms.org/tutorials/)

### 13.7 Binary Discrimination

#### 13.7.1 Linearly Inseparable Groups: Logistic discrimination, LDA, and SVM

First we produce ALM-III Figure 13.6. In the book I treated Carcinoma as the “1” group and bilateral hyperplasia as the “0” group. In the code, that is reversed.

```r
rm(list = ls())
cush <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM12-1c.DAT", sep="", col.names=c("Type","Tetra","Preg"))
attach(cush)
cush
TL = log(Tetra)
PL = log(Preg)
Tp=Type-2
Tp2=3-Type
# Tp2 is 1 for Bilat hyp and 0 for Carcinoma
```
ac2 = glm(Tp2 ~ TL + PL, family=binomial)
summary(ac2)
anova(ac2)
post=c(Type, ac2$fit, 1-ac2$fit)
PropProb=matrix(post, 15, 3, dimnames =
  list(NULL, c("Group", "B", "C"))
PropProb

# Without adjustment,
# logistic regression uses prior probabilities
# proportional to sample sizes.

# Estimated posterior probabilities
prior=.5 # prior probability for first group (coded as 1)
n=length(Tp)
n1=10 # number of observations in first group
n2=n-n1
PostOdds=(ac2$fit/(1-ac2$fit))*(n2/n1)*(prior/(1-prior))
PostProb=PostOdds/(1+PostOdds)
posttab=c(Type, PostProb, 1-PostProb)
PosteriorTable=matrix(posttab, 15, 3, dimnames =
  list(NULL, c("Group", "B", "C"))
PosteriorTable

# Plot of points and discrimination curves
xx2=c(8.3, 3.8, 3.9, 7.8, 9.1, 15.4, 7.7, 6.5, 5.7, 13.6)
yy2=c(1.00, .20, .60, 1.20, .60, 3.60, 1.60, .40, .40, 1.60)
xx3=c(10.2, 9.2, 9.6, 53.8, 15.8)
yy3=c(6.40, 7.90, 3.10, 2.50, 7.60)
x2=log(xx2)
x3=log(xx3)
y2=log(yy2)
y3=log(yy3)
plot(x2, y2, pch=16, ylab="log(Pregnanetriol)",
  ylim=c(-2,3), xlab="log(Tetrahydrocortisone)")
points(x3, y3, pch=22)
legend("bottomright", c("Bilateral Hyperplasia", "Carcinoma"),
  pch=c(16, 22))
x1=seq(1, 4.5, .01)
b=ac2$coef
y2=(b[1]+log(n2/n1)+log(prior/(1-prior))+b[2]*x1)/-b[3]
lines(x1, y2, type="l")
legend("topright", c("Logistic", "LDA", "SVM"), lty=c(1, 2, 5))
T = factor(Type)

library(MASS)
fit <- lda(T ~ TL + PL, prior = c(1/2, 1/2))
fit
summary(fit)
pfit = predict(fit)
pfit
c <- table(pfit$class, T)

md = fit$means[1,] + fit$means[2,]
bb = fit$scaling
yLDA = (log(prior/(1-prior)) - .5*sum(md*bb) + bb[1]*x1) / -bb[2]
lines(x1, yLDA, type = "l", lty = 2)

# install.packages("e1071")
library(e1071)
fit <- svm(T ~ TL + PL, kernel = "linear", scale = F)

fit$SV
fit$coefs
fit$rho
x1 = seq(1, 4.5, .005)
b = t(fit$SV) %*% fit$coefs
# solve <(x1, ySVM)'b >= fit$rho
ySVM = (-fit$rho + log(prior/(1-prior)) + b[1]*x1) / -b[2]
# = (fit$rho - b[1]*x1)/b[2]
lines(x1, ySVM, type = "l", lty = 5)

13.7.1.1 ANREG-II, Subsection 21.9: Modified

ANREG-II, Subsection 21.9 uses a loglinear model to perform logistic discrimination for all three groups in the Cushing Syndrome data. To double check my earlier logistic two-group regression work, I modified the ANREG-II code to handle just 2 groups: Bilateral hyperplasia and Carcinoma.

rm(list = ls())
cush <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM12-1c.DAT", sep = ",", col.names = c("Syn", "Tetra", "Preg"))
attach(cush)
cush

# Create a 2 x 15 table of 0-1 entries, each row has 1's for a different type of syndrome
j=rep(seq(1,15),2)
i=c(rep(1,15),rep(2,15))
Tet=c(Tetra,Tetra)
Pre=c(Preg,Preg)
y=c(Syn,Syn)
y[1:10]=1
y[11:15]=0
y[16:25]=0
y[26:30]=1
datal=c(y,i,j,Tet,Pre)
datl=matrix(datal,30,5,dimnames =
   list(NULL,c("y", "i", "j","Tet","Pre")))
datl

# Fit the log-linear model for logistic discrimination.
i=factor(i)
j=factor(j)
lp=log(Pre)
lt=log(Tet)
ld <- glm(y ˜ i + j + i:lt +i:lp ,family = poisson)
ldp=summary(ld)
ldp
anova(ld)

# Table 21.12
q=ld$fit
# Divide by sample sizes
p1=ld$fit[1:15]/10
p2=ld$fit[15:30]/5
# Produce table
estprob = c(Syn,p1,p2)
EstProb=matrix(estprob,15,3,dimnames =
   list(NULL,c("Group","B","C")))
EstProb

# Table 21.13 Proportional prior probabilities.
post = c(Syn,ld$fit)
PropProb=matrix(post,15,3,dimnames =
   list(NULL,c("Group","B","C")))
PropProb
13.7 Binary Discrimination

# Table 21.13 Equal prior probabilities.
\[ p = p_1 + p_2 \]
\[ pp_1 = \frac{p_1}{p} \]
\[ pp_2 = \frac{p_2}{p} \]
\[ \text{post} = c(Syn, pp_1, pp_2) \]
\[ \text{EqProb} = \text{matrix(post, 15, 3, dimnames = list(NULL, c("Group", "B", "C")))} \]
\[ \text{EqProb} \]

### 13.7.2 Quadratically Separable Groups: Logistic discrimination, QDA, and SVM

We now produce ALM-III Figures 13.7 and 13.8. The only difference is in the choice of the tuning parameter for SVM and the SVM fits are unchanged from ALM-III Figures 13.3 and This begins a four part program. It creates the data plot and draws the QDA discrimination lines. First I borrowed the plot of the data points from ANREG-II and added a legend

#### 13.7.2.1 Enter and plot data

Entering the data:

\[ xx_2 = c(8.3, 3.8, 3.9, 7.8, 9.1, 15.4, 7.7, 6.5, 5.7, 13.6) \]
\[ yy_2 = c(1.00, .20, .60, 1.20, .60, 3.60, 1.60, .40, .40, 1.60) \]
\[ xx_3 = c(10.2, 9.2, 9.6, 53.8, 15.8) \]
\[ yy_3 = c(6.40, 7.90, 3.10, 2.50, 7.60) \]
\[ x_2 = \log(xx_2) \]
\[ x_3 = \log(xx_3) \]
\[ y_2 = \log(yy_2) \]
\[ y_3 = \log(yy_3) \]
plot(x_2, y_2, pch=16, ylab="log(Pregnanetriol)",
\[ y\text{lim}=c(-2,3), x\text{lim}=c(1,4.5), x\text{lab}="log(Tetrahydrocortisone)"\]
points(x_3, y_3, pch=22)
legend("bottomright", c("Bilateral Hyperplasia", "Carcinoma"),
\[ \text{pch}=c(16,22)) \]
legend("topleft", c("Logistic", "QDA", "SVM"), lty=c(1,2,5))

Reading in the data:

\[ \text{rm(list = ls())} \]
\[ \text{cush} <- \text{read.table("C:\E-drive\Books\LINMOD23\DATA\ALM12-1c.DAT",}\]
\[ \text{sep="", col.names=c("Type", "Tetra", "Preg"))} \]
attach(cush)
cush
T = factor(Type)
TL = log(Tetra)
PL = log(Preg)
plot(TL[T==2], PL[T==2], pch=16, ylab="log(Pregnantriol)",
     ylim=c(-2,3), xlim=c(1,4.5), xlab="log(Tetrahydrocortisone)")
points(TL[T==3], PL[T==3], pch=22)
legend("bottomright", c("Bilateral Hyperplasia", "Carcinoma"),
       pch=c(16,22))
legend("topleft", c("Logistic", "QDA", "SVM"), lty=c(1,2,5))

13.7.2.2 Quadratic Logistic Discrimination

TL2=TL*TL
PL2=PL*PL
TPL=PL*TL
Tp=Type-2
Tp2=3-Type
# Tp2 is 1 for Bilat hyp and 0 for Carcin

ac2 = glm(Tp2 ~ TL + PL + TL2 + PL2 + TPL, family=binomial)
summary(ac2)
anova(ac2)
post=c(Type,ac2$fit,1-ac2$fit)
PropProb=matrix(post,15,3,dimnames =
                   list(NULL,c("Group","B","C")))
PropProb
#Without adjustment, logistic regression uses prior probabilities
#proportional to sample sizes.

# Estimated posterior probabilities
prior=.5 #prior probability for first group (coded as 1)
n=length(Tp)
n1=10 #number of observations in first group
n2=n-n1
# Correct for sample sizes
PostOdds=(ac2$fit/(1-ac2$fit))*(n2/n1)*(prior/(1-prior))
PostProb=PostOdds/(1+PostOdds)
posttab=c(Type,PostProb,1-PostProb)
PosteriorTable=matrix(posttab,15,3,dimnames =
                      list(NULL,c("Group","B","C")))
PosteriorTable
#solve quadratic equation
x1=seq(1,4.5,.1)
b=ac2$coef
prior=.5
c=log(n2/n1)+log(prior/(1-prior))+bb[1]+bb[2]*x1+bb[4]*x1^2
b= bb[3] + bb[6]*x1
a=bb[5]
delta=b^2-4*a*c
yLR = (-b+sqrt(delta))/(2*a)
yLR2 = (-b-sqrt(delta))/(2*a)
lines(x1,yLR,type="1",lty=1)
lines(x1,yLR2,type="1",lty=1)

## 13.7.2.3 Quadratic Discriminant Analysis

This is a continuation of the previous program.

T=factor(Type)
library(MASS)
fit <- qda(T ˜ TL + PL,prior=c(.5,.5))
fit
summary(fit)
pfit=predict(fit)
pfit

# simplify notation.
# Qi is a nonsymmetric square root of
# the sample covariance matrix in group i
mu1=fit$means[,1]
mu2=fit$means[,2]
Q1=fit$scaling[,1]
Q2=fit$scaling[,2]
Q1=t(Q1)
Q2=t(Q2)
Q1m1=Q1%*%mu1
Q2m2=Q2%*%mu2

# define and solve quadratic equation.
# add curves to previous plot
prior=.5
c=log(prior/(1-prior)) -(fit$ldet[2] - fit$ldet[1]) +
.5*(sum(Q2m2*Q2m2)-sum(Q1m1*Q1m1)) +
(sum(Q1m1*Q1[,1]) - sum(Q2m2*Q2[,1]))*x1 +
.5*(sum(Q2[,1]*Q2[,1])- sum(Q1[,1]*Q1[,1]))*x1^2
\[ b = (\text{sum}(Q1m1 \times Q1[,2]) - \text{sum}(Q2m2 \times Q2[,2])) + \\
(\text{sum}(Q2[,1] \times Q2[,2]) - \text{sum}(Q1[,1] \times Q1[,2])) \times x1 \]

\[ a = 0.5 \times (\text{sum}(Q2[,2] \times Q2[,2]) - \text{sum}(Q1[,2] \times Q1[,2])) \]

\[ \delta = b^2 - 4 \times a \times c \]

\[ y_{QDA} = \frac{-b + \sqrt{\delta}}{2 \times a} \]

\[ y_{QDA2} = \frac{-b - \sqrt{\delta}}{2 \times a} \]

\[ \text{lines}(x1, y_{QDA}, \text{type}="l", \text{lty}=2) \]

\[ \text{lines}(x1, y_{QDA2}, \text{type}="l", \text{lty}=2) \]

### 13.7.2.4 Quadratic SVM

This is a continuation of the previous program. Typically you would want to have scale=T in \texttt{svm}, which is the default. \textit{This is exactly the same SVM program as given earlier}.

\[
\texttt{#install.packages("e1071")}
\]

\[
\texttt{library(e1071)}
\]

\[
\texttt{T=factor(Type)}
\]

\[
\texttt{# define SVM and display outputs}
\]

\[
\texttt{fit <- svm(T ˜ TL + PL, kernel="polynomial", degree=2, gamma=1, coef0=1, scale=F)}
\]

\[
\texttt{# Next line reduces tuning parameter, i.e., increases cost.}
\]

\[
\texttt{#fit <- svm(T ˜ TL + PL, kernel="polynomial", degree=2, gamma=1, coef0=1, scale=F, cost=100)}
\]

\[
\texttt{fit$SV}
\]

\[
\texttt{fit$coefs}
\]

\[
\texttt{fit$rho}
\]

\[
\texttt{fit$fitted} \quad \text{shows the groups that each case was allocated to}
\]

\[
\texttt{# fitted establishes that my parabolic solution is reasonable.}
\]

\[
\texttt{# predict(fit)}
\]

\[
\texttt{# define and solve quadratic equation.}
\]

\[
\texttt{# add curves to previous plot}
\]

\[
\texttt{# must solve matrix equation that involves inner products}
\]

\[
\texttt{x1=seq(1,4.5,.01)}
\]

\[
\texttt{w=fit$coefs}
\]

\[
\texttt{c=-fit$rho + \text{sum}(w) + 2 \times \text{sum}(w \times \text{fit$SV}[1]) \times x1} + \\
\text{sum}(w \times (\text{fit$SV}[1])^2) \times x1^2
\]

\[
\texttt{b=2 \times \text{sum}(w \times \text{fit$SV}[2]) + 2 \times \text{sum}(w \times \text{fit$SV}[1] \times \text{fit$SV}[2]) \times x1}
\]

\[
\texttt{a=\text{sum}(w \times (\text{fit$SV}[2])^2)}
\]

\[
\texttt{delta=b^2 - 4 \times a \times c}
\]

\[
\texttt{y_{SVM} = (-b + \text{sqrt}(\delta))/(2 \times a)}
\]

\[
\texttt{y_{SVM2} = (-b - \text{sqrt}(\delta))/(2 \times a)}
\]
A simple example not in *ALMIII*

This example is not in the book. It was used for classroom discussion.

---

### Fig. 13.1 Simple Binary Data.

```r
rm(list = ls())
T = c(1, 1, 1, 1, 0, 0, 0, 0, 0)
TT = factor(T)
Y1 = c(1, 1, 1, 1, -1, -1, -1, -1, -1)
Y2 = c(.9, .8, .7, .6, -1.1, -.9, -.8, -.7, -.6, -.5)
Y11 = Y1[TT == 1]
Y10 = Y1[TT == 0]
Y21 = Y2[TT == 1]
Y20 = Y2[TT == 0]
plot(Y11, Y21, pch = 16, ylab = "y2", xlab = "y1")
```
Fig. 13.2 Logistic Regression.

Fig. 13.3 Logistic Discrimination and R default SVM.
13.8 A simple example not in ALIII

Fig. 13.4 Logistic Discrimination, R default SVM, LR augmented data ridge.

Fig. 13.5 Logistic Discrimination and SVM with C=100.
## Binary Discrimination and Regression

```r
#ylim=c(-2,3),xlim=c(1,4.5),
points(Y10, Y20, pch=22)
x1=seq(-1.1,1.1,.05)
#lines(x1,x1)
ac2 = glm(T ˜ Y1 + Y2,family=binomial)
prior=.5 #prior probability for first group (coded as 1)
n=length(T)
n1=5 #number of observations in first group
n2=n-n1
b=ac2$coef
yLR=(b[1]+log(n2/n1)+log(prior/(1-prior))+b[2]*x1)/-b[3]
lines(x1,yLR,type="l")
legend("topleft",c("LogReg","Log Ridge","SVM"),lty=c(1,2,5))

#install.packages("e1071")
library(e1071)
fit <- svm(TT ˜ Y1 + Y2,kernel="linear",scale=F)
# add cost=100 and get LR. Default is cost=1
fit$SV
fit$coefs
fit$rho
b=t(fit$SV)%%fit$coefs
ySVM=(-fit$rho+b[1]*x1)/-b[2]
# = (fit$rho - b[1]*x1)/b[2]
lines(x1,ySVM,type="l",lty=5)

# No new software logistic ridge
k=1
TR=c(T,.5,.5)
Y1R =c(Y1,1,0)
Y2R =c(Y2,0,1)
W=c(Y1/Y1,k,k)
J=c(Y1/Y1,0,0)
acR = glm(TR ˜ J + Y1R + Y2R -1,family=binomial,weights=W)
prior=.5 #prior probability for first group (coded as 1)
n=length(T)
n1=5 #number of observations in first group
n2=n-n1
```
13.8 A simple example not in *ALMIII*

\[ b = \text{acoef} \]
\[ yLRR = (b[1] + \log(n2/n1) + \log(\text{prior}/(1-\text{prior}))) + b[2] \times x1) / -b[3] \]
\[ \text{lines(x1,yLRR,type="1",lty=2)} \]
Chapter 14
Principal Components, Classical
Multidimensional Scaling, and Factor Analysis

https://cran.r-project.org/web/views/Multivariate.html contains an overview of R procedures for multivariate analysis.

14.1 Theory

This code is for showing an ellipse along with its major and minor axes. While the code is correct, you have to futz with the axes lengths in R or latex to get the orthogonal axes to actually look perpendicular. The main thing is that that in R, since the plot appears as a square, the lengths of the plotted x and y axes have to be the same to get the vectors to look approximately orthogonal. However, latex also allows you to stretch the axes, so you again need the figure’s height and width to be the same. But even then, if you want the axes to really look orthogonal, you need to futz a bit.

```r
library(ellipse)
rm(list = ls())
b1=1
b2=2
A = matrix(c(1,.9,.9,2),2,2, dimnames=list(NULL, c("b1","b2")))
A
E <- ellipse(A,centre = c(b1, b2),t=.95, npoints=100)

b=seq(0,1.52,.01)
x=1+.507128*b
y=2+.8618708*b
bq=seq(-.65,0,.01)
x1=1+(.8618708*bq)
y1=2-(.507128*bq)

K=5
```
100 14 Principal Components, Classical Multidimensional Scaling, and Factor Analysis

\[ bb_1 = 1 + (0.8618708 \cdot K) \]
\[ bb_2 = 2 - (0.507128 \cdot K) \]
\[ AA = \text{matrix} \left( c(1, .9, .9, 2), 2, 2, \text{dimnames} = \text{list}(\text{NULL}, c("bb1", "bb2")) \right) \]
\[ EE \leftarrow \text{ellipse} \left( AA, \text{centre} = c(bb_1, bb_2), t = .95, \text{npoints} = 100 \right) \]

\[ bb = \text{seq}(0, 1.52, .01) \]
\[ x = 1 + 0.507128 \cdot bb \]
\[ y = 2 + 0.8618708 \cdot bb \]
\[ bb = \text{seq}(-0.65, 0, .01) \]
\[ x_1 = 1 + (0.8618708 \cdot bb) \]
\[ y_1 = 2 - (0.507128 \cdot bb) \]

\[
\text{plot}(E, \text{type} = 'l', \text{ylim} = c(.5, 3.5), \text{xlim} = c(-.5, 2.5), \\
\text{xlab} = \text{expression}(y[1]), \\
\text{ylab} = \text{expression}(y[2]), \text{main} = \text{"Principal Components"}) \\
\text{text}(b1+.01, (b2-.1), \text{expression}(\mu), \text{lwd} = 1, \text{cex} = 1) \\
\text{lines}(EE, \text{type} = 'l', \text{lty} = 1) \\
\text{lines}(b1, b2, \text{type} = 'p', \text{pch} = 19) \\
\text{lines}(x, y, \text{type} = 'l', \text{lty} = 1) \\
\text{lines}(x_1, y_1, \text{type} = 'l', \text{lty} = 1) \\
\text{text}((b1+.5), (b2+.53), \text{expression}(a[1]), \text{lwd} = 1, \text{cex} = 1) \\
\text{text}((b1-.3), (b2+.12), \text{expression}(a[2]), \text{lwd} = 1, \text{cex} = 1) \\
\]

# The following plots 20 random data points over the ellipse.
\[
\text{library(mvtnorm)} \\
\text{T} = \text{rmvnorm}(20, c(1, 2), A) \\
\text{lines}(T[,1], T[,2], \text{type} = 'p')
\]

14.1.1 Normal Density Plot for PA-V

\[
\text{#install.packages("ellipse")} \\
\text{#Do this only once on your computer} \\
\text{library(ellipse)} \\
\]
\[ b1 = 1 \]
\[ b2 = 2 \]
\[ A = \text{matrix} \left( c(1, .9, .9, 2), 2, 2, \text{dimnames} = \text{list}(\text{NULL}, c("b1", "b2")) \right) \]
\[ A \]
\[ E \leftarrow \text{ellipse}(A, \text{centre} = c(b1, b2), t = .95, \text{npoints} = 100) \]
\[ E1 \leftarrow \text{ellipse}(A, \text{centre} = c(b1, b2), t = .5, \text{npoints} = 100) \]
14.2 Sample Principal Components

There is a section of commands in the R code for ANREG-II for doing principal component regression.

There are three data files available for the turtle shell data: combined, females, and males. This example uses the males.

```r
rm(list = ls())
turtm <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM10-3m.dat", sep="", col.names=c("L","W","H"))
attach(turtm)
turtm
x3=10^(3/2)*log(L)
x2=10^(3/2)*log(W)
x1=10^(3/2)*log(H)
n=length(L)
X = matrix(c(x1,x2,x3),n,3)
S=var(X)
S
R=cor(X)
R
# two equivalent ways of specifying the data
fit <- prcomp(~ x1+x2+x3, scale=F)
summary(fit)
fit$rotation
fit <- prcomp(X, scale=F)
summary(fit)
fit$rotation

e <- eigen(S,symmetric=TRUE)
e$values
```
14.2.1 Using Principal Components

```R
rm(list = ls())
b1=1
b2=2
A = matrix(c(1,.9,.9,2),2,2, dimnames=list(NULL, c("b1","b2")))
A
E <- ellipse(A,centre = c(b1, b2),t=.95, npoints=100)

b=seq(0,1.52,.01)
x=1+.507128*b
y=2+.8618708*b
bq=seq(-.65,0,.01)
x1=1+.8618708*bq
y1=2-.507128*bq

K=-.8
bb1=1+(.507128*K)
bb2=2-(.8618708*K)
AA = matrix(c(1,.9,.9,2),2,2, dimnames=list(NULL, c("bb1","bb2")))
AA
EE <- ellipse(AA,centre = c(bb1, bb2),t=.95, npoints=100)

bb=seq(0,1.52,.01)
x=1+.507128*bb
y=2+.8618708*bb
bb=seq(-.65,0,.01)
x1=1+.8618708*bb
y1=2-.507128*bb

plot(E,type = 'l',ylim=c(0,5),xlim=c(-1,2.5),
    xlab=expression(y[1]),
    ylab=expression(y[2]),main="Principal Components")
lines(b1,b2,type="p",pch=19)
text((b1+.02),(b2-.15),expression(mu[1]),lwd=1,cex=1)
lines(EE,type="1",lty=1)
```
14.3 Multidimensional Scaling

You have some measure of the distances between items and you want to find where these points are located relative to one another. We are going to reconsider the school test data on which we did factor analysis. We don’t have the data on the 225 students but we do have the correlation matrix which is something like the opposite of a distance matrix. Correlations are large when things are close and small when they are far away.

14.3.1 Classical MDS

First we do the examples in the book. Then do an extra example. Then do other forms of multidimensional scaling.

Example 14.3.1

```r
rm(list = ls())
cush <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM12-1.DAT", sep="", col.names=c("Syn","Tetra","Preg"))
attach(cush)
cush
ID = list("a1","a2","a3","a4","a5","a6","b1","b2","b3","b4","b5","b6","b7","b8","b9","b10","c1","c2","c3","c4","c5")
TL=log(Tetra)
PL=log(Preg)
CS=cbind(TL,PL)
CSD=dist(CS)  # default is
CSfit <- cmdscale(CSD,eig=TRUE, k=2)
CSfit

x <- CSfit$points[,1]
y <- CSfit$points[,2]
plot(y, x, xlab="Coordinate 2", ylab="Coordinate 1", main="Classical Multidimensional Scaling", type="n")
text(y, x, labels = ID, cex=.9)
```
Example 14.3.2

```r
rm(list = ls())

ID = list("Gaelic","English","History",
    "Arithmetic","Algebra","Geometry")
R = matrix(c(1.000,0.439,0.410,0.288,0.329,0.248,
    0.439,1.000,0.351,0.354,0.320,0.329,
    0.410,0.351,1.000,0.164,0.190,0.181,
    0.288,0.354,0.164,1.000,0.595,0.470,
    0.329,0.320,0.190,0.595,1.000,0.464,
    0.248,0.329,0.181,0.470,0.464,1.000),6,6)
#R=-log(R)
R2=R*R
D=1-R
D2=1-R2
par(mfrow=c(2,1))
Dfit <- cmdscale(D,eig=TRUE, k=2)
Dfit
x <- Dfit$points[,1]
y <- Dfit$points[,2]
plot(x, y, xlab="Coordinate 1", ylab="Coordinate 2",
    xlim=c(-.46,.35),ylim=c(-.26,.26),
    main="CMDS: Correlations", type="n")
text(x, y, labels = ID, cex=.9)

D2fit <- cmdscale(D2,eig=TRUE, k=2)
D2fit
x2 <- D2fit$points[,1]
y2 <- D2fit$points[,2]
plot(x2, y2, xlab="Coordinate 1", ylab="Coordinate 2",
    xlim=c(-.46,.46),ylim=c(-.46,.46),
    main="CMDS: Squared Correlations", type="n")
text(x2, y2, labels = ID, cex=.9)
```

14.3.1.1 Application to Heart Rate Data

Find distances from MANOVA heart rate data and plot CMDS coordinates.

```r
rm(list = ls())
resp <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM10-2.dat", sep="", col.names=c("Y1","Y2","Y3","Y4","Drug"))
attach(resp)
resp
```
14.3 Multidimensional Scaling

Y=cbind(Y1,Y2,Y3,Y4)

ID=seq(1,30)
Yd=dist(Y)  # default is

Yfit <- cmdscale(Yd,eig=TRUE, k=2)  # k is the number of dim
Yfit
x <- Yfit$points[,1]
y <- Yfit$points[,2]
plot(x, y, xlab="Coordinate 1", ylab="Coordinate 2",
    main="Metric MDS", type="n")
text(x, y, labels = ID, cex=.9)

Fig. 14.1 Multidimensional Scaling: Heart Data.

14.3.2 Nonmetric MDS

This is not discussed in the text!

For cases in which “distance” is nothing more than an ordering. Generally, a building 10 miles away is twice as far away as a building that is 5 miles away. A
true distance measure has that property, but often we want to apply multidimensional scaling to “discrepancies” rather than distances. The program below uses the Classical solution as a starting point.

```
rm(list = ls())
cush <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM12-1.DAT", sep="", col.names=c("Syn","Tetra","Preg"))
attach(cush)
cush
ID = list("a1","a2","a3","a4","a5","a6", "b1","b2","b3","b4","b5","b6","b7","b8","b9","b10", "c1","c2","c3","c4","c5")
TL=log(Tetra)
PL=log(Preg)
CS=cbind(TL,PL)
CSD=dist(CS) # default is
library(MASS)
CSfit <- isoMDS(CSD,k=2) # k is the number of dim
CSfit
```

Fig. 14.2 Nonmetric Multidimensional Scaling: Cushing Syndrome Data.
14.3 Multidimensional Scaling

x <- CSfit$points[,1]
y <- CSfit$points[,2]
plot(y, -x, xlab="Coordinate 2", ylab="-Coordinate 1",
    main="Nonmetric MDS", type="n")
text(y, -x, labels = ID, cex=.9)
rm(list = ls())
resp <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM10-2.dat",
    sep="",col.names=c("Y1","Y2","Y3","Y4","Drug"))
attach(resp)
resp
Y=cbind(Y1,Y2,Y3,Y4)

ID=seq(1,30)
Yd=dist(Y)  # default is
class library(MASS)
Yfit <- isoMDS(Yd,k=2)  # k is the number of dim
Yfit
x <- Yfit$points[,1]
y <- Yfit$points[,2]
plot(x, y, xlab="Coordinate 1", ylab="Coordinate 2",
    main="Nonmetric MDS", type="n")
text(x, y, labels = ID, cex=.9)

14.3.3 Example 14.3.2

This is our factor analysis example on correlations between 6 tests. All we have is a correlation matrix.

\[
R = \begin{bmatrix}
1.000 & 0.439 & 0.410 & 0.288 & 0.329 & 0.248 \\
0.439 & 1.000 & 0.351 & 0.354 & 0.320 & 0.329 \\
0.410 & 0.351 & 1.000 & 0.164 & 0.190 & 0.181 \\
0.288 & 0.354 & 0.164 & 1.000 & 0.595 & 0.470 \\
0.329 & 0.320 & 0.190 & 0.595 & 1.000 & 0.464 \\
0.248 & 0.329 & 0.181 & 0.470 & 0.464 & 1.000 \\
\end{bmatrix}
\]

Not looking at how far apart the 225 people are, looking at how far apart the 6 tests are. \(Y\) would be \(6 \times 225\) with 225 objects measured on every item of interest.
Fig. 14.3 Nonmetric Multidimensional Scaling: Heart Data.

Fig. 14.4 Multidimensional Scaling: Test Data, $D = 1 - R$. 
14.3 Multidimensional Scaling

14.3.3.1 Code

CMDS

```r
rm(list = ls())

ID = list("Gaelic","English","History",
            "Arithmetic","Algebra","Geometry")
R = matrix(c(1.000,0.439,0.410,0.288,0.329,0.248,
            0.439,1.000,0.351,0.354,0.320,0.329,
            0.410,0.351,1.000,0.164,0.190,0.181,
            0.288,0.354,0.164,1.000,0.595,0.470,
            0.329,0.320,0.190,0.595,1.000,0.464,
            0.248,0.329,0.181,0.470,0.464,1.000),6,6)
#R=-log(R)
R2=R*R
D=1-R
D2=1-R2
par(mfrow=c(2,1))
Dfit <- cmdscale(D,eig=TRUE, k=2)
Dfit
x <- Dfit$points[,1]
y <- Dfit$points[,2]
plot(x, y, xlab="Coordinate 1", ylab="Coordinate 2",
     xlim=c(-.46,.35),ylim=c(-.26,.26),
     main="CMDS: Correlations", type="n")
text(x, y, labels = ID, cex=.9)

D2fit <- cmdscale(D2,eig=TRUE, k=2)
D2fit
x2 <- D2fit$points[,1]
y2 <- D2fit$points[,2]
plot(x2, y2, xlab="Coordinate 1", ylab="Coordinate 2",
     xlim=c(-.4,.46),ylim=c(-.46,.46),
     main="CMDS: Squared Correlations", type="n")
text(x2, y2, labels = ID, cex=.9)

Other approaches.

library(MASS)
Rfit <- isoMDS(R,k=2) # k is the number of dim
Rfit

x <- Rfit$points[,1]
y <- Rfit$points[,2]
plot(x, y, xlab="Coordinate 1", ylab="Coordinate 2",
```
14.4 Factor Analysis

An overview of R procedures for multivariate analysis is available at [https://cran.r-project.org/web/views/Multivariate.html](https://cran.r-project.org/web/views/Multivariate.html).

14.4.1 Terminology and Applications

This section uses only the base function `factanal`. A subsection of the next section uses the facilities from the library `psych`.

The example presents maximum likelihood estimation from correlation matrix (rather than raw data). Unrotated factor loadings.

```
R = matrix(c(1.000, 0.439, 0.410, 0.288, 0.329, 0.248,
             0.439, 1.000, 0.351, 0.354, 0.320, 0.329,
             0.410, 0.351, 1.000, 0.164, 0.190, 0.181,
             0.288, 0.354, 0.164, 1.000, 0.595, 0.470,
             0.329, 0.320, 0.190, 0.595, 1.000, 0.464,
             0.248, 0.329, 0.181, 0.470, 0.464, 1.000), 6, 6)
```

test=factanal(covmat=R,n.obs=220,factors=2,rotation="none")

test

test$unique
test$loadings[,1:2]
```

```
rm(list = ls())
turtm <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM10-3m.dat", sep=" ", col.names=c("L","W","H"))
attach(turtm)
turtm
x3=10^(3/2)*log(L)
x2=10^(3/2)*log(W)
x1=10^(3/2)*log(H)
n=length(L)
X = matrix(c(x1,x2,x3),n,3)
test=factanal(X,factors=1,rotation="none")
```
14.4 Factor Analysis

\[ \text{test} = \text{factanal}(\sim x_1 + x_2 + x_3, \text{factors} = 1, \text{rotation} = \text{"none"}) \]

### 14.4.2 Maximum Likelihood Theory

This illustrates the varimax rotation and the three plots. `factanal` does not do quartimax, but the next subsubsection uses a different program that allows quartimax rotation.

```r
plot(test$loadings[,1], test$loadings[,2],
     xlim=c(-1,1), ylim=c(-1,1),
     type="n", xlab="Factor 1", ylab="Factor 2")
x = seq(-1,1,.01)
lines(x, 0*x)
lines(0*x, x)
text(test$loadings[,1], test$loadings[,2],
     labels=c(1,2,3,4,5,6))
```

```r
test = factanal(covmat = R, n.obs = 220, factors = 2,
                 rotation = "varimax")
# varimax is the default rotation
```

```r
test$unique
```

```r
test$loadings[,1:2]
```

```r
plot(test$loadings[,1], test$loadings[,2],
     xlim=c(-1,1), ylim=c(-1,1),
     type="n", xlab="Factor 1", ylab="Factor 2")
lines(x, 0*x)
lines(0*x, x)
text(test$loadings[,1], test$loadings[,2],
     labels=c(1,2,3,4,5,6))
```

```r
# I read the quartimax loadings from the book
# factanal does not do quartimax
f1 = c(0.260, 0.344, 0.111, 0.777, 0.731, 0.580)
f2 = c(0.650, 0.536, 0.587, 0.139, 0.184, 0.188)
test$loadings = matrix(c(f1,f2), 6, 2)
test$loadings
```

```r
plot(test$loadings[,1], test$loadings[,2],
     xlim=c(-1,1), ylim=c(-1,1),
     type="n", xlab="Factor 1", ylab="Factor 2")
lines(x, 0*x)
lines(0*x, x)
```
14.4.2.1 Psych-ed out

Everything done thus far and more can be done using the library `psych`. To get the quartimax rotation, another library is also needed.

\[
R = \begin{pmatrix}
1.000 & 0.439 & 0.410 & 0.288 & 0.329 & 0.248 \\
0.439 & 1.000 & 0.351 & 0.320 & 0.329 & \\
0.410 & 0.351 & 1.000 & 0.164 & 0.190 & 0.181 \\
0.288 & 0.354 & 0.164 & 1.000 & 0.595 & 0.470 \\
0.329 & 0.320 & 0.190 & 0.595 & 1.000 & 0.464 \\
0.248 & 0.329 & 0.181 & 0.470 & 0.464 & 1.000 \\
\end{pmatrix}, 6, 6
\]

#install.packages("psych")
library(psych)
#install.packages("GPArotation")
library(GPArotation)

fit <- fa(R, fm="ml", nfactors=2, n.obs=220, rotate="none")
fit
fit <- fa(R, fm="ml", nfactors=2, n.obs=220, rotate="varimax")
fit
fit <- fa(R, fm="ml", nfactors=2, n.obs=220, rotate="quartimax")
fit
fit <- fa(R, fm="pa", nfactors=2, rotate="none")

14.4.3

Still need to check these out.

# Determine Number of Factors to Extract
library(nFactors)
ev <- eigen(cor(mydata)) # get eigenvalues
ap <- parallel(subject=nrow(mydata), var=ncol(mydata), rep=100, cent=.05)
nS <- nScree(x=ev$values, aparallel=ap$eigen$qevpea)
plotnScree(nS)
# PCA Variable Factor Map
library(FactoMineR)
result <- PCA(mydata) # graphs generated automatically
Chapter 15
Other Multivariate Analysis Topics

My non-book *Statistical Learning: A Second Course in Regression* contains discussion of some additional multivariate techniques such as clustering. Code is at [www.stat.unm.edu/~fletcher/R-SL.pdf](http://www.stat.unm.edu/~fletcher/R-SL.pdf).