Ronald Christensen  
Department of Mathematics and Statistics  
University of New Mexico  
© 2017

Preliminary Version of  
R Commands for –  
Advanced Linear Modeling III

Springer
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>1</td>
</tr>
<tr>
<td>0.1 Basic Information</td>
<td>1</td>
</tr>
<tr>
<td>0.2 Organization</td>
<td>2</td>
</tr>
<tr>
<td>1 Nonparametric Regression</td>
<td>3</td>
</tr>
<tr>
<td>1.1 Linear Approximations</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Simple Nonparametric Regression</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Estimation</td>
<td>3</td>
</tr>
<tr>
<td>1.3.1 Polynomials</td>
<td>3</td>
</tr>
<tr>
<td>1.3.2 Cosines</td>
<td>4</td>
</tr>
<tr>
<td>1.3.3 Haar wavelets</td>
<td>6</td>
</tr>
<tr>
<td>1.3.4 Cubic Splines</td>
<td>7</td>
</tr>
<tr>
<td>1.3.5 Orthogonal Series Estimation</td>
<td>9</td>
</tr>
<tr>
<td>1.4 Variable Selection</td>
<td>9</td>
</tr>
<tr>
<td>1.5 Heteroscedastic Simple Nonparametric Regression</td>
<td>9</td>
</tr>
<tr>
<td>1.6 Approximating-functions with Small Support</td>
<td>9</td>
</tr>
<tr>
<td>1.6.1 Polynomial Splines</td>
<td>9</td>
</tr>
<tr>
<td>1.6.2 Fitting Local Functions</td>
<td>12</td>
</tr>
<tr>
<td>1.6.3 Local regression</td>
<td>12</td>
</tr>
<tr>
<td>1.7 Nonparametric Multiple Regression</td>
<td>13</td>
</tr>
<tr>
<td>1.7.1 Redefining φ and the Curse of Dimensionality</td>
<td>13</td>
</tr>
<tr>
<td>1.7.2 Reproducing Kernel Hilbert Space Regression</td>
<td>13</td>
</tr>
<tr>
<td>1.8 Testing Lack of Fit in Linear Models</td>
<td>15</td>
</tr>
<tr>
<td>1.9 Regression Trees</td>
<td>15</td>
</tr>
<tr>
<td>1.9.1 Bagging and Boosting</td>
<td>17</td>
</tr>
<tr>
<td>1.10 Density Estimation</td>
<td>20</td>
</tr>
<tr>
<td>2 Penalized Regression</td>
<td>21</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>21</td>
</tr>
<tr>
<td>2.2 Ridge Regression</td>
<td>21</td>
</tr>
<tr>
<td>2.3 Lasso Regression</td>
<td>23</td>
</tr>
</tbody>
</table>

vii
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4</td>
<td>Bayesian Connections</td>
<td>25</td>
</tr>
<tr>
<td>2.5</td>
<td>Another Approach</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>Reproducing Kernel Hilbert Spaces</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>Covariance Parameter Estimation</td>
<td>29</td>
</tr>
<tr>
<td>5</td>
<td>Mixed Models</td>
<td>31</td>
</tr>
<tr>
<td>6</td>
<td>Frequency Domain</td>
<td>33</td>
</tr>
<tr>
<td>6.2</td>
<td>Basic Data Analysis</td>
<td>33</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Periodogram</td>
<td>33</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Figures</td>
<td>34</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Spectral Density</td>
<td>36</td>
</tr>
<tr>
<td>6.2.4</td>
<td>Detrended Spectral Density</td>
<td>37</td>
</tr>
<tr>
<td>6.3</td>
<td>The Random Effects Model</td>
<td>38</td>
</tr>
<tr>
<td>6.4</td>
<td>The Measurement Error Model</td>
<td>38</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Prediction</td>
<td>41</td>
</tr>
<tr>
<td>6.5</td>
<td>Linear filtering</td>
<td>43</td>
</tr>
<tr>
<td>6.5.1</td>
<td>Recursive filtering</td>
<td>43</td>
</tr>
<tr>
<td>6.6</td>
<td>The Coherence of Two Time Series</td>
<td>43</td>
</tr>
<tr>
<td>6.7</td>
<td>Fourier Analysis</td>
<td>43</td>
</tr>
<tr>
<td>6.8</td>
<td>Exercise Data Plots</td>
<td>44</td>
</tr>
<tr>
<td>7</td>
<td>Time Domain</td>
<td>45</td>
</tr>
<tr>
<td>7.1</td>
<td>Correlations</td>
<td>45</td>
</tr>
<tr>
<td>7.5</td>
<td>Estimation</td>
<td>46</td>
</tr>
<tr>
<td>7.5.1</td>
<td>Ljung-Box</td>
<td>47</td>
</tr>
<tr>
<td>7.5.2</td>
<td>General Commands</td>
<td>48</td>
</tr>
<tr>
<td>7.6</td>
<td>Model Selection</td>
<td>49</td>
</tr>
<tr>
<td>7.7</td>
<td>Seasonal Adjustment</td>
<td>50</td>
</tr>
<tr>
<td>7.8</td>
<td>The Multivariate State-Space Model and the Kalman Filter</td>
<td>51</td>
</tr>
<tr>
<td>8</td>
<td>Spatial Data</td>
<td>53</td>
</tr>
<tr>
<td>9</td>
<td>Multivariate Linear Models: General</td>
<td>55</td>
</tr>
<tr>
<td>9.1</td>
<td>Generating Multivariate Normals</td>
<td>55</td>
</tr>
<tr>
<td>9.2</td>
<td>Specifying Multivariate Data Matrices</td>
<td>55</td>
</tr>
<tr>
<td>10</td>
<td>Multivariate Linear Models: Applications</td>
<td>57</td>
</tr>
<tr>
<td>10.1</td>
<td>One-sample</td>
<td>57</td>
</tr>
<tr>
<td>10.2</td>
<td>Two-samples</td>
<td>57</td>
</tr>
<tr>
<td>10.3</td>
<td>One-Way MANOVA and Profile Analysis</td>
<td>58</td>
</tr>
<tr>
<td>10.3.1</td>
<td>Profile Analysis</td>
<td>59</td>
</tr>
<tr>
<td>10.3.2</td>
<td>Split Plot Analysis</td>
<td>61</td>
</tr>
</tbody>
</table>
## 11 Generalized Multivariate Linear Models

11.1 Generalized Multivariate Linear Models .......................... 63
11.2 Generalized least squares analysis .................................. 63
11.3 MACOVA analysis .................................................. 64
11.4 Raos 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

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 ................................. 72
12.4 Cross-validation .................................................. 73
12.5 Discussion ...................................................... 73
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

13.1 Binomial Regression ................................................ 77
13.1.1 Data Augmentation Ridge Regression ......................... 77
13.2 Binary Prediction .................................................. 77
13.3 Generalized Linear Models .......................................... 79
13.4 Best Prediction and Probability Estimation ....................... 80
13.5 Linear Prediction Rules .......................................... 80
13.6 Support Vector Machines .......................................... 83
13.7 Binary Discrimination ................................................ 83
13.7.1 Linearly Inseparable Groups: Logistic discrimination, LDA, and SVM .................................................. 83
13.7.2 Quadratically Separable Groups: Logistic discrimination, QDA, and SVM .................................................. 87
13.8 A simple example not in ALMIII ................................... 91

## 14 Principal Components, Classical Multidimensional Scaling, and Factor Analysis

14.1 Theory .......................................................... 97
14.1.1 Normal Density Plot for PA-V .................................. 98
14.2 Sample Principal Components ...................................... 99
14.2.1 Using Principal Components .................................. 100
14.3 Multidimensional Scaling .......................................... 101
14.3.1 Classical MDS .................................................. 101
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 and in particular with the material in Chapters 1 and 3 of my Preliminary Version of R Commands for Analysis of Variance, Design, and Regression: Linear Modeling of Unbalanced Data (R-code for ANREG2) which is elsewhere on my website. (Also the matrix commands at the beginning of Chapter 11.)

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.

car
DAAG
e1071
ellipse
FactoMineR
forecast
glmnet
GPArotation
lasso2
lme4
MASS
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: http://www.r-project.org/ Click on “CRAN”, click on a site, then under “software”, click on “packages”. The terms “package” and “library” are used 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

1.2 Simple Nonparametric Regression

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.

1.3.1 Polynomials

We fit the polynomial model necessary for obtaining Figure 1.1.

```r
rm(list = ls())
battery <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM1-1.dat", header=TRUE, sep="")
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 Figures 1.2 and 1.3 by picking “p” appropriately. Currently has p = 6.

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

```r
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)

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

```r
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 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
        b=(2*k-1)/2^f
        c=2*k/2^f
        kk=2^(f-1)-1+k
        Phi[,kk]=((as.numeric(x<=b)-as.numeric(x>b))
            *as.numeric(x<=c)*as.numeric(x>a))
    }
}
Phi

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

# 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>.5))
  *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=(2*k-2)/2^f
      b=(2*k-1)/2^f
      c=2*k/2^f
      kk=2^(f-1)+k
      Phinew[,kk]=((as.numeric(xx<=b)-as.numeric(xx>b))
        *as.numeric(xx<=c)*as.numeric(xx>a))
    }
}

Phinew

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

1.3.4 Cubic Splines

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)
Nonparametric Regression

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=seq(0,1,.01)
nff=length(xx)
Phinew=matrix(seq(1:(nff*(nk+4))),nff)
Phinew[,1]=1
Phinew[,2]=xx
Phinew[,3]=xx^2
Phinew[,4]=xx^3
for(k in 1:nk)
{k=k+4
Phinew[,k+4]=(as.numeric(xx>knot[k])*(xx-knot[k]))^3
}
Phinew
yy1=Phinew%*%Bhat
plot(x,y,type="p")
lines(xx,yy1,type="l")
1.6 Approximating-functions with Small Support

1.3.5 Orthogonal Series Estimation

This is basic stuff matrix manipulation stuff as discussed in the R code for ANREG2.

1.4 Variable Selection

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

1.5 Heteroscedastic Simple Nonparametric Regression

1.6 Approximating-functions with Small Support

1.6.1 Polynomial Splines

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

An example in which polynomial splines agree with \texttt{bsplines}.

1.6.1.1 Regular spline model

\begin{verbatim}
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
m=as.integer(d)
Phi=matrix(seq(1:(nf*(nk+d))),nf)
Phi
for(kk in 1:d)
{

}
\end{verbatim}
\[
\Phi[,kk] = x^{kk} 
\]

knot = seq(1:nk)/(nk+1)
knot
for(k in 1:nk)
{
\Phi[, (k+d)] = (as.numeric(x>knot[k]) \times (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[, kk] = xx^kk
}
for(k in 1:nk)
{k
kk = k+d+1
Phinew[, kk] = (as.numeric(xx>knot[k]) \times (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
  if (d == 2)
    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)
  else
    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
}

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
}
\[ \Psi_2 = (\text{as.numeric}(u \geq 0)) \times (\text{as.numeric}(u \leq 1)) \times \left( \frac{u^2}{2} \right) + (\text{as.numeric}(u > 1)) \times (\text{as.numeric}(u \leq 2)) \times \left( \frac{(u-1.5)^2 - .75}{2} \right) + (\text{as.numeric}(u > 2)) \times (\text{as.numeric}(u \leq 3)) \times \left( \frac{(3-u)^2}{2} \right) \]

\[ \Psi_3 = (\text{as.numeric}(u \geq 0)) \times (\text{as.numeric}(u \leq 1)) \times \left( \frac{u^3}{3} \right) + (\text{as.numeric}(u > 1)) \times (\text{as.numeric}(u \leq 2)) \times \left( -u^3 + 4u^2 - 4u + \frac{4}{3} \right) + (\text{as.numeric}(u > 2)) \times (\text{as.numeric}(u \leq 3)) \times \left( -\left( 4-u \right)^3 + 4 \times \left( 4-u \right)^2 - 4 \times (4-u) + \frac{4}{3} \right) + (\text{as.numeric}(u > 3)) \times (\text{as.numeric}(u \leq 4)) \times \left( \frac{(4-u)^3}{3} \right) \]

1.6.2 Fitting Local Functions

1.6.3 Local regression

The default \texttt{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",
```
1.7 Nonparametric Multiple Regression

The package \texttt{np} does kernel smoothing.

1.7 Nonparametric Multiple Regression

The package \texttt{gam} fits generalized additive models using smoothing splines or local regression.

1.7.1 Redefining $\phi$ and the Curse of Dimensionality

1.7.2 Reproducing Kernel Hilbert Space Regression

\textbf{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.

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])
  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.
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
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(8.525,8.525),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)
lines(c(6.235,6.235),c(5,8),type="o",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
c0 <- lm(y ~ x1+x2+x3+x4+x5)
cop=summary(c0)
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
1.9 Regression Trees

\[ Z \leftarrow \text{model.matrix(co)}[,-1] \]
\[ \text{co} \leftarrow \text{lm(y } \sim \text{ Z)} \]
\[ \text{anova(co)} \]
\[ \text{confint(co, level=0.95)} \]

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

\[ \text{infv = c(y, co$fit, hatvalues(co), rstandard(co),} \]
\[ \text{rstudent(co), cooks.distance(co)})} \]
\[ \text{inf=matrix(infv, I(cop$df[1]+cop$df[2])}, 6, \text{dimnames = list(NULL,} \]
\[ \text{c("y", "yhat", "lev", "r", "t", "C")})} \]
\[ \text{inf} \]
\[ \text{qqnorm(rstandard(co), ylab="Standardized residuals")} \]

# Wilk-Francia
\[ \text{rankit=qnorm(ppoints(rstandard(co), a=I(3/8))} \]
\[ \text{ys=sort(rstandard(co))} \]
\[ \text{Wprime=(cor(rankit, ys))^2} \]
\[ \text{Wprime} \]
\[ \text{plot(co$fit, rstandard(co), xlab="Fitted",} \]
\[ \text{ylab="Standardized residuals", main="Residual-Fitted plot"} \]

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.

\[ \text{rm(list = ls())} \]
\[ \text{nfull=200} \]
\[ \text{ntst=100} \]
\[ \text{n=nfull-ntst} \]
Nonparametric Regression

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

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

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

PredSS=seq(1:9)
PredSS[1]=sum((Tst-xbar)^2)/ntst
PredSS[2]=sum((Tst-xtld)^2)/ntst
PredSS[3]=sum((Tst-mid)^2)/ntst
PredSS[4]=sum((Tst-mean(Bxbar))^2)/ntst
PredSS[5]=sum((Tst-mean(Bxtld))^2)/ntst
PredSS[6]=sum((Tst-mean(Bmid))^2)/ntst
PredSS[7]=sum((Tst-0)^2)/ntst
PredSS[8]=8
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 our 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.

```r
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)
  xtld=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)
    Bxtld[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- xtld)^2)/ntst
}
```
\texttt{PredSS[3]=sum((Tst \text{- mid})^2)/ntst}
\texttt{PredSS[4]=sum((Tst \text{- mean(Bxbar)})^2)/ntst}
\texttt{PredSS[5]=sum((Tst \text{- mean(Bxtld)})^2)/ntst}
\texttt{PredSS[6]=sum((Tst \text{- mean(Bmid)})^2)/ntst}
\texttt{PredSS[7]=sum((Tst \text{- 0})^2)/ntst}
\texttt{APress=APress+PredSS}
\texttt{)}
\texttt{APress=APress/SS}
\texttt{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="")#,col.names=c("Case","y","t","x"))
attach(battery)
battery

nf=length(x)
p=10
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.
\texttt{Phi6=Phi[,c(2,3,4,5,6,7)]}

\texttt{cos6 = lm(y\sim Phi6)}
\texttt{coss6 = summary(cos6)}
\texttt{coss6}
\texttt{anova(cos6)}
\texttt{Bhat6=coefficients(cos6)}
\texttt{Bhat6=c(Bhat6,0,0,0,0)}
\texttt{Bhat6}

Create and fit the augmented Ridge model

\texttt{w=seq(1:10)}
\texttt{zip=w-w}
\texttt{w=w}
\texttt{w=w*w}
\texttt{Dw=.2*diag(w)}
\texttt{Dwx=cbind(zip,Dw)}
\texttt{Dwx}
\texttt{XR=rbind(Phi,Dwx)}
\texttt{XR}
\texttt{YR=c(y,zip)}
\texttt{cosr=lm(YR \sim XR-1)}
\texttt{BhatR=coefficients(cosr)}
\texttt{BhatR}

\texttt{(cor(y,Phi%*%Bhat6)^2)}
\texttt{(cor(y,Phi%*%BhatR)^2)}

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

\texttt{yy1=Phinew%*%Bhat6}
\texttt{yyr=Phinew%*%BhatR}

\texttt{plot(x,y,type="p")}
\texttt{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 ANREG2. This is for comparing the \( p = 30 \) cosine lasso fit with 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[,k]=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
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")))
A
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
y1=-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)
```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 `nlme` has an option `gls` for doing generalized least squares (GLS) and also fits a variety of covariance structures. The library `MASS` also has a program `ls.gls` for doing GLS. The library `lme4` (due to Bates [Donald, not Norman] and associates) does many things better than the older `nlme` but according to the 2016 online documentation, “`lme4` does not currently offer the same flexibility as `nlme` for composing complex variance-covariance structures” nor does it seem to have an option for GLS.
Chapter 5
Mixed Models

The library \texttt{lme4} has a command \texttt{lmer}. The package \texttt{mnle} has \texttt{lme}.
An overview of R packages for time series analysis is available at https://cran.r-project.org/web/views/TimeSeries.html.

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

```r
rm(list = ls())
c oal.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

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))


```r
par(mfrow=c(1,1))
tt = seq(100, (n*100))/100
c1t = cos(2*pi*1*tt/n)
c15t = cos(2*pi*15*tt/n)
c30t = cos(2*pi*30*tt/n)
s1t = sin(2*pi*1*tt/n)
s15t = sin(2*pi*15*tt/n)
s30t = sin(2*pi*30*tt/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, c1t)
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, s1t)
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))
```

**Figure 6.3 4**

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

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

6.2.3 Spectral Density

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)
floor((n-1)/2)
P.lm = seq(1, nf)
f = P.lm / n
par(mfrow=c(3,1))
f1 = spec.pgram(y, taper=0, detrend=FALSE, fast=FALSE)
6.2 Basic Data Analysis

\[
f_3 = \text{spec.pgram}(y, \text{taper}=0, \text{detrend}=\text{FALSE}, \text{fast}=\text{FALSE}, \text{kernel}=\text{"daniell", 1})
\]
\[
f_5 = \text{spec.pgram}(y, \text{taper}=0, \text{detrend}=\text{FALSE}, \text{fast}=\text{FALSE}, \text{kernel}=\text{"daniell", 2})
\]
\[
\text{FFTtable} = \text{cbind}(\text{freq}, f_1\text{spec}, f_3\text{spec}, f_5\text{spec})
\]
\[
\text{par(mfrow=c(1,1))}
\]
\[
\text{plot}(\text{freq}, \log(f_1\text{spec}), \text{type}=\text{"l"}, \text{lty}=2,
\quad \text{ylab}=\text{"Log Density"}, \text{xlab}=\text{"Frequency"}, \text{ylim}=c(4, 12))
\]
\[
\text{lines}(\text{freq}, \log(f_3\text{spec}), \text{type}=\text{"l"}, \text{lty}=5)
\]
\[
\text{lines}(\text{freq}, \log(f_5\text{spec}), \text{type}=\text{"l"}, \text{lty}=1)
\]
\[
\text{legend}(\text{"topright"}, c(\text{"Estimate"}, \text{"P"}, \text{"fhat_3"}, \text{"fhat_5"}),
\quad \text{lty}=c(\text{NA}, 2, 5, 1))
\]

\[
\text{spec.pgram(}\text{residuals(fit)}, \text{taper}=0, \text{detrend}=\text{FALSE}, \text{fast}=\text{FALSE})
\]

6.2.4 Detrended Spectral Density

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

\[
\text{rm(list = lapply())}
\]
\[
\text{coal.production} <- \text{read.table(}
\quad \text{"C:}\text{"E-drive}\text{"Books}\text{"LINMOD23}\text{"DATA}\text{"ALM6-1.dat"},
\quad \text{sep}=\text{"", col.names=c(\"Year", \"Wt\")}
\)
\]
\[
\text{attach(coal.production)}
\]
\[
\text{coal.production}
\]
\[
y=Wt
\]
\[
\text{par(mfrow=c(2,1))}
\]
\[
\# f1=f3=\text{spec.pgram}(y, \text{taper}=0, \text{fast}=\text{FALSE})
\]
\[
\# f3=\text{spec.pgram}(y, \text{taper}=0, \text{fast}=\text{FALSE}, \text{kernel}=\text{"daniell", 1})
\]
\[
f_5=\text{spec.pgram}(y, \text{taper}=0, \text{detrend}=\text{FALSE}, \text{fast}=\text{FALSE}, \text{kernel}=\text{"daniell", 2})
\]
\[
\text{par(mfrow=c(1,1))}
\]
\[
\text{plot}(f_5\text{freq}, \log(f_5\text{spec}), \text{type}=\text{"l"}, \text{lty}=1,
\quad \text{ylab}=\text{"Log Density"}, \text{xlab}=\text{"Frequency"}, \text{ylim}=c(4, 12))
\]
\[
\text{lines}(f_5\text{freq}, \log(f_5\text{spec}), \text{type}=\text{"l"}, \text{lty}=5)
\]
\[
\text{legend}(\text{"bottomleft"}, c(\text{"Estimate"}, \text{"Detrended"}, \text{"Regular"}),
\quad \text{lty}=c(\text{NA}, 1, 5))
\]
38 6 Frequency Domain

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

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))

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)
6.4 The Measurement Error Model

coal.production
y = W + e
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$spec)
FFTtable
par(mfrow = c(1, 1))
plot(freq, log(f1$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("topleft", c("Estimate", "P", "fhat_3", "fhat_5"),
    lty = c(NA, 2, 5, 1))

a = 0.05
r = 1
fhat = f1$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

a = 0.05
r = 3
fhat = f3$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

a = 0.05
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 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())
c Coastal.production <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM6-1.dat", sep="",col.names=c("Year","Wt"))
attach(coal.production)
c Coastal.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))^2 + (sum(S1*y))^2)/n
P2= ((sum(C2*y))^2 + (sum(S2*y))^2)/n
P4= ((sum(C4*y))^2 + (sum(S4*y))^2)/n
P5= ((sum(C5*y))^2 + (sum(S5*y))^2)/n
P6= ((sum(C6*y))^2 + (sum(S6*y))^2)/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)
```

C4t = cos(2*pi*4*ttt/n)
S4t = sin(2*pi*4*ttt/n)
C5t = cos(2*pi*5*ttt/n)
S5t = sin(2*pi*5*ttt/n)
C6t = cos(2*pi*6*ttt/n)
S6t = sin(2*pi*6*ttt/n)

Zt = data.frame(C1=C1t, S1=S1t, C2=C2t, S2=S2t, C4=C4t, S4=S4t, C5=C5t, S5=S5t, C6=C6t, S6=S6t)

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

# Mixed model prediction
sigaa = (P1+P2)/2
siga = (2/n)*(sigaa-MSE)
sigbb = (P4+P5+P6)/3
sigb = (2/n)*(sigbb-MSE)

MSE
sigaa
siga
sigbb
sigb

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

gammat = c((siga/sigaa)*(sum(C1*y)), (siga/sigaa)*(sum(S1*y)), (siga/sigaa)*(sum(C2*y)), (siga/sigaa)*(sum(S2*y)), (siga/sigaa)*(sum(C4*y)), (siga/sigaa)*(sum(S4*y)), (siga/sigbb)*(sum(C5*y)), (siga/sigbb)*(sum(S5*y)), (siga/sigbb)*(sum(C6*y)), (siga/sigbb)*(sum(S6*y)))

Ztt = as.matrix(Zt, c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))

MMpred = Ztt %*% gammat + mean(y)
MMfit = Z %*% gammat + mean(y)

plot(Year, Wt, type="l", xlim=c(20, 87), ylim=c(300, 925))
lines(Yearfuture, Wtfuture, type="l")
lines(Yearfuture, MMpred, type="l", lty=5)
6.5 Linear filtering

```r
rm(list = ls())
goal.production <- read.table("C:\E-drive\Books\LINMOD23\DATA\ALM6-1.dat", sep="",&nbcolnames=c("Year","Wt"))
attach(goal.production)
goal.production
y=Wt
# The first three filters are equivalent
RA5 = filter(y,filter=c(.2,.2,.2,.2,.2))
RA5a = filter(y,filter=c(.2,.2,.2,.2,.2),
method = "convolution",sides=2)
RA5b = filter(y,rep(.2,5))
RA6 = filter(y,filter=c(1/12,1/6,1/6,1/6,1/6,1/12))
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.
While the analysis of the coal data is consolidated in Subsection 7.6.3 of the book, we distribute the computations through the earlier sections.

7.1 Correlations

```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

# 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))
```

45
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 a package.

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

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

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

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

# Variance estimate
SSE = t(fit$residuals)%*%fit$residuals
# Eliminate "+1" if "constant=FALSE"
```
# Correlation Matrix
\[ \text{diag}(1/\text{SE}, \text{nrow} = \text{I}(\text{MO}[1]+\text{MO}[3]+1)) \ast \text{fit}\$\text{var.coef} \ast \text{diag}(1/\text{SE}, \text{nrow} = \text{I}(\text{MO}[1]+\text{MO}[3]+1)) \]

# Figs 7-5
\[
\text{par(mfrow = c(2, 1))}
\]
\[\text{Acf(residuals(fit), ylim = c(-1, 1))}\]
\[\text{Pacf(residuals(fit), ylim = c(-1, 1))}\]
\[\text{par(mfrow = c(1, 1))}\]

# Results not shown
# normal plot, periodogram
# fhat_5 spectral density
\[\text{qqnorm(residuals(fit), ylab = "Wt residuals")}\]
\[\text{spec.pgram(residuals(fit), taper = 0, detrend = FALSE, fast = FALSE)}\]
\[\text{spec.pgram(residuals(fit), taper = 0, detrend = FALSE, fast = FALSE, kernel("daniell", 2))}\]

# Fig 7-6
\[\text{plot(Year, residuals(fit), type = "b")}\]

# Fig 7-7
\[\text{fitpred = forecast(fit, 7)}\]
\[\text{fitpred}\]
\[\text{Yearfuture = c(81, 82, 83, 84, 85, 86, 87)}\]
\[\text{Wtfuture = c(818.4, 833.5, 777.9, 891.8, 879.0, 886.1, 912.7)}\]
\[\text{plot(Year, Wt, type = "l", xlim = c(20, 87), ylim = c(300, 925))}\]
\[\text{lines(Yearfuture, Wtfuture, type = "l")}\]
\[\text{lines(Yearfuture, fitpred$mean, type = "l", lty = 5)}\]
\[\text{lines(Year, fit$fit, type = "l", lty = 5)}\]
\[\text{legend("topleft", c("Actual", "ARIMA"), lty = c(1, 5))}\]

### 7.5.1 Ljung-Box

\[\text{MO = c(2, 1, 1)}\]
\[\text{fit = Arima(Wt, order = MO), include.constant = TRUE, method = "ML"}\]
\[\text{fit}\]
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 an 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

```r
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)
```

```r
lag(ts, k) lagged time series, shifted back k observations
diff(ts, differences=d) difference the time series d times
ndiffs(ts) differences required to stationarity (forecast)
```

```r
adf.test(ts) # Augmented 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.

```r
# fit an ARIMA model of order P, D, Q
fit <- arima(myts, order=c(p, d, q))
# predictive accuracy
library(forecast)
accuracy(fit)
```

```r
# predict next 5 observations
library(forecast)
forecast(fit, 5)
plot(forecast(fit, 5))
```
7.6 Model Selection

Remember that for a time series \( y_1, \ldots, y_n \), the time series being analyzed in an 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)
cbind(coal.production)

#install.packages("forecast")
library(forecast)
ndiffs(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(1,1,1)
fit = Arima(Wt,order=MD,include.constant=TRUE,method = "ML")
#,seasonal=list(order=c(0,0,0),period=7))
fit
fit$sigma2
fit$aic # redundant calculation below
fit$loglik
n=length(Wt)-MD[2]
AAIC=-2*(fit$loglik)+2*kk`
Time Domain

$$BBIC = -2(\text{fit}'\text{loglik}) + \log(n) \times (kk)$$
$$AAICc = AAIC + 2(kk) \times (kk+1)/(n-kk-1)$$
$$AAIC$$
$$BBIC$$
$$AAICc$$

```
#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

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

#install.packages("forecast")
library(forecast)
ndiffs(WIHosp)  # Automated determination of differencing
# Assumes stationarity, min diff that is consistent
# Uses KPSS test
# Automated determination of seasonal differencing
WI=ts(WIHosp,frequency=7)  # creates time series object
nsdiffs(WI)
```

# Figs 7-8
7.8 The Multivariate State-Space Model and the Kalman Filter

An overview of R procedures for time series analysis is available at https://cran.r-project.org/web/views/Spatial.html. One for spatiotemporal 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 `krige.conv` for conventional kriging and `krige.bayes` for Bayesian kriging.
- **gstat** package, see programs `krige gstat predict RandomFields`
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.
n=length(Y1)
Y1=matrix(c(Y1,Y2,Y3,Y4),n,4)
Y1

Finally, within various programs (like those for discrimination analysis, principal components, or factor analysis) we can specify data matrix in a manner similar to specifying a model (matrix)

˜Y1+Y2+Y3+Y4)

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, often copies incorrectly so that you need to delete the copied version 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
rmtree(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)
re
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.

\begin{verbatim}
\begin{verbatim}
par(mfrow=c(2,2))
plot(re$fit[,1],rstudent(re)[,1],xlab="Fitted y1", ylab="Standardized residuals y1",main="Residual-Fitted plot")
plot(re$fit[,2],rstudent(re)[,2],xlab="Fitted y2", ylab="Standardized residuals y2",main="Residual-Fitted plot")
plot(re$fit[,3],rstudent(re)[,3],xlab="Fitted y3", ylab="Standardized residuals y3",main="Residual-Fitted plot")
plot(re$fit[,4],rstudent(re)[,4],xlab="Fitted y4", ylab="Standardized residuals y4",main="Residual-Fitted plot")
\end{verbatim}
\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., 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)
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 One-Way MANOVA and Profile Analysis

Constructing

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
```

63
# Nicer output comes from the "car" package.
library(car)
re.manova=Manova(re)
summary(re.manova)
# To obtain E and H
Et1=re.manova$SSPE
St1=Et1/re$df.residual
St1
Ht1=re.manova$SSP

Yt10=Yt1[,1]
yt11=Yt1[,2]
Yt12=Yt1[,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
Yt1=Y%*%W
Yt11=Y%*%W1
Yt12=Y%*%W2
```
# Summary tables
D = factor(Drug)
re <- lm(Ytilde1 ~ D + Ytilde2 - 1)
rep = summary(re)
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$SSPE
Stilde = Etilde1 / re$df.residual
Etilde
Htilde = re.manova$SSP

# Summary tables
D = factor(Drug)
re0 <- lm(Ytilde1 ~ Ytilde2)
rep0 = summary(re0)
re0$coef

re0.manova = Manova(re0)
summary(re0.manova)
# To obtain E and H
Etilde0 = re0.manova$SSPE
Stilde = Etilde1 / re$df.residual
Etilde0
Htilde = Etilde0 - Etilde
Htilde

**Example 11.3.2**

df <- 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)
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)
re0$coef

re0.manova = Manova(re0)
summary(re0.manova)
# To obtain E and H
Etilde0 = re0.manova$SSPE
Stilde = Etilde0 / re$df.residual
Etilde0
Htilde=Etilde0-Etilde
Htilde
library(psych)
tr(Htilde%*%solve(Stilde))

11.4 Raos 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)
abt
#summary(cush)
TL = log(Tetra)
PL = log(Preg)
```

The plot from ANREG2

```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)
x1=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)")
```

69
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,.31),c(0,.0277))
lines( c(-.31,-.31),c(0,.114))
legend("topright",c("QDA","Mahalanobis"),pch=c(22,16))
cbind(x,y,y1)
par(mfrow=c(1,1))
```
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(-.31,-.31),c(0,.114))
legend("topright",c("QDA","Mahalanobis"),pch=c(22,16))
cbind(x,y,y1)
par(mfrow=c(1,1))
```

It is surprisingly futzy 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 Discrimination and Allocation

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
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
plot(y21,y22,xlab=expression(y[1]),
ylab=expression(y[2]),main="Discriminant Analysis")
lines(y11,y12,type="p",pch=4)

plot(yt11,yt22,xlim=c(1.5,2.5))
lines(yt21,yt22,type="p",pch=4)

In ALM-III the Examples are accumulated into Section 5 but here we give the linear the examples in sections that describe the theory. Don’t think this is still true.
```
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.

# K-fold cross-validation
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)
Spec=factor(s)
r12=y1/y2
y=cbind(y1,y2,y3,y4,y5,y6,r12)

# None of the following?
beat <- lm(y ~ Spec)
anova(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.
        "Placebo", "Placebo", "Placebo", "Placebo", "Placebo",
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)

rm(list = ls())
b1 = 0
b2 = 0
d = 1
b3 = -0.8320503 * d
b4 = .5547002 * d
A = 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)

plot(E, type = 'l', ylim = c(-1.5, 2), xlim = c(-2, 1.5), xlab = expression(y[1]),
     ylab = expression(y[2]), main = "Discriminant Analysis")
lines(E1, type = "l", lty = 1)
text((b1 + .1), (b2 - .15), expression(mu[1]), lwd = 1, cex = 1)
lines(b1, b2, type = "p", pch = 16)
text((b3 + .1), (b4 - .15), expression(mu[2]), lwd = 1, cex = 1)
lines(b3, b4, type = "p", pch = 19)

12.7.2 Discrimination Coordinates

Copied from ANREG commands

Get \( H \) and \( E \) from a one-way manova. Perhaps from \texttt{fitted.values}\ or \texttt{residuals}\ and \texttt{df.residuals}\ from \texttt{lm}\ or

\[
\begin{align*}
\text{rm}(\text{list = ls()}) \\
b1 & = 0 \\
b2 & = 0 \\
d & = 1 \\
b3 & = -0.8320503 * d \\
b4 & = .5547002 * d \\
A & = .5 * \text{matrix}(c(1, 1.2, 1.2, 2), 2, 2, \text{dimnames = list(NULL, c("b1", "b2")))} \\
E & \leftarrow \text{ellipse}(A, \text{centre = c(b1, b2), t = .95, npoints = 100}) \\
E1 & \leftarrow \text{ellipse}(A, \text{centre = c(b3, b4), t = .95, npoints = 100})
\end{align*}
\]

The following plots 20 random data points over the ellipse.

\[
\begin{align*}
\text{library}(&\texttt{mvtnorm}) \\
T & \leftarrow \text{rmvnorm}(20, c(b1, b2), A) \\
\text{plot}(T, \text{type = 'p'}, \text{ylim = c(-1.5, 2)}, \text{xlim = c(-2, 1.5)}, \text{xlab = expression(y[1]),} \\
\text{ylab = expression(y[2]), main = "Discriminant Analysis")} \\
T1 & \leftarrow \text{rmvnorm}(20, c(b3, b4), A) \\
\text{lines}(T1, \text{type = "p", pch = 4})
\end{align*}
\]

TT = \text{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

*glm* fits the generalized linear models but it does not incorporate penalties except through augmenting the data.

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

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)",
lim=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)
#PropProb=
# matrix(post,15,3,dimnames = list(NULL,c("Group","B","C")))
#PropProb

x1=seq(1,4.5,.01)
b=ac2$coef
y2=(b[1]+b[2]*x1)/-b[3]
lines(x1,y2,type="l")
legend("topright",c("Logistic","Probit","SVM"),lty=c(1,2,5))

# 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="l",lty=2)

# SVM
13.3 Generalized Linear Models

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

```r
#install.packages("e1071")
library(e1071)
T=factor(Type)
# Typically you would want to have
# \texttt{scale=T} in \texttt{svm}.
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$rho+b[1]*x1)/-b[2]  # = (fit$rho - b[1]*x1)/b[2]
lines(x1,ySVM,type="l",lty=5)

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))
```
80 13 Binary Discrimination and Regression

This does NOT give the other half SVM

#\textbf{y}_4 = -2 \cdot \log(\exp(-0.5 \cdot x + 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 Figures 13.3 and 13.5. They involves fitting logistic and probit regressions and an SVM classifier. The linear structures are quadratic in $T_L$ and $P_L$. 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 Linear Prediction Rules

13.5.0.2 Quadratic Logistic and Probit Regression

This is a continuation of the previous plotting program.

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

\[
\begin{align*}
ac2 &= \text{glm}(Tp2 \sim TL + PL + TL2 + PL2 + TPL, \\
& \quad \text{family=binomial})
\end{align*}
\]

#solve quadratic equation
\[
\begin{align*}
x1 &= \text{seq}(1,4.5,.1) \\
bb &= ac2$coef \\
prior &= .5 \\
a &= bb[5] \\
delta &= b^2 - 4 \times a \times c \\
yLR &= (-b + \sqrt{\delta}) / (2 \times a) \\
yLR2 &= (-b - \sqrt{\delta}) / (2 \times a) \\
\text{lines}(x1,yLR,\text{type}="1",\text{lty}=1) \\
\text{lines}(x1,yLR2,\text{type}="1",\text{lty}=1)
\end{align*}
\]

\[
\begin{align*}
ac3 &= \text{glm}(Tp2 \sim TL + PL + TL2 + PL2 + TPL, \\
& \quad \text{family=binomial(link="probit"))}
\end{align*}
\]

#solve quadratic equation
\[
\begin{align*}
x1 &= \text{seq}(1,4.5,.1) \\
bb &= ac3$coef \\
prior &= .5 \\
a &= bb[5] \\
delta &= b^2 - 4 \times a \times c \\
yPR &= (-b + \sqrt{\delta}) / (2 \times a) \\
yPR2 &= (-b - \sqrt{\delta}) / (2 \times a) \\
\text{lines}(x1,yPR,\text{type}="1",\text{lty}=1) \\
\text{lines}(x1,yPR2,\text{type}="1",\text{lty}=1)
\end{align*}
\]
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 the coefficients \( w \) (fit$coefs) that are the nonzero coefficients of \( \frac{1}{2} \sum (\lambda^T \lambda'_{11}, -\lambda'_{10}). \) 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

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

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 \( \hat{\beta}_s \) and \( \hat{\beta}_0 \) as discussed in the book. Now run `svm` with `scale=T` and compute \( \hat{\beta}_s \) and \( \hat{\beta}_0 \). How do you transform \( \hat{\beta}_s \) 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/

### 13.7 Binary Discrimination

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

First we produce 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 Carcin
```
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),xlim=c(1,4.5),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))
13.7 Binary Discrimination

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

\[ \text{library(MASS)} \]

\[ \text{fit} \leftarrow \text{lda}(T \sim TL + PL, \text{prior}=c(1/2,1/2)) \]

\[ \text{fit} \]

\[ \text{summary(fit)} \]

\[ \text{pfit=predict(fit)} \]

\[ \text{pfit} \]

\[ \text{ct} \leftarrow \text{table(pfit$\text{class},T)} \]

\[ \text{ct} \]

\[ \text{md=fit$\text{means}[1,]+fit$\text{means}[2,]} \]

\[ \text{bb=fit$\text{scaling}} \]

\[ \text{yLDA}=(\log(\text{prior}/(1-\text{prior}))-.5*\text{sum(md*bb)+bb[1]*x1})/-\text{bb[2]} \]

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

#install.packages("e1071")

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

\[ \text{fit} \leftarrow \text{svm}(T \sim TL + PL, \text{kernel}="\text{linear}", \text{scale}=\text{F}) \]

\[ \text{fit$SV} \]

\[ \text{fit$coefs} \]

\[ \text{fit$rho} \]

\[ \text{x1}=\text{seq}(1,4.5,.005) \]

\[ \text{b=t(fit$SV)%*%fit$coefs} \]

\[ \#\text{solve} <(x1,y\text{SVM}',b)=\text{fit$rho} \]

\[ \text{ySVM}=(-\text{fit$rho} + \log(\text{prior}/(1-\text{prior}))+\text{b[1]*x1})/-\text{b[2]} \]

\[ \# = (\text{fit$rho} - \text{b[1]*x1})/\text{b[2]} \]

\[ \text{lines(x1,ySVM,type="l",lty=5)} \]

13.7.1.1 **ANREG2, Subsection 21.9: Modified**

ANREG2, 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 **ANREG2** code to handle just 2 groups: Bilateral hyperplasia and Carcinoma.

\[ \text{rm(list = ls())} \]

\[ \text{cush} \leftarrow \text{read.table("C:\E-drive\Books\LINMOD23\DATA\ALM12-1c.DAT", sep="",col.names=c("Syn","Tetra","Preg"))} \]
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)
dat1 = matrix(datal, 30, 5, dimnames =
   list(NULL, c("y", "i", "j", "Tet", "Pre")))
dat1

# 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)
ddp = summary(ld)
ddp
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
## Table 21.13 Equal prior probabilities.

\[
p = p_1 + p_2 \\
p_{1} = p_1 / p \\
p_{2} = p_2 / p \\
\text{post} = c(Syn, p_{1}, p_{2})
\]

\[
\text{EqProb} = \text{matrix}(\text{post}, 15, 3, \text{dimnames} = \text{list(NULL, c("Group", "B", "C")))}
\]

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

We now produce 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 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 ANREG2 and added a legend.

#### 13.7.2.1 Enter and plot data

Entering the data:

\[
\begin{align*}
xx2 &= c(8.3, 3.8, 3.9, 7.8, 9.1, 15.4, 7.7, 6.5, 5.7, 13.6) \\
yy2 &= c(1.00, .20, .60, 1.20, .60, 3.60, 1.60, .40, .40, 1.60) \\
xx3 &= c(10.2, 9.2, 9.6, 53.8, 15.8) \\
yy3 &= c(6.40, 7.90, 3.10, 2.50, 7.60) \\
x2 &= \log(xx2) \\
x3 &= \log(xx3) \\
y2 &= \log(yy2) \\
y3 &= \log(yy3)
\end{align*}
\]

\[
\text{plot}(x2, y2, \text{pch}=16, \text{ylab}=\text{"log(Pregnanetriol)"}, \\
\text{ylim}=c(-2,3), \text{xlim}=c(1,4.5), \text{xlab}=\text{"log(Tetrahydrocortisone)"}) \\
\text{points}(x3, y3, \text{pch}=22) \\
\text{legend}(\text{"bottomright"}, c(\text{"Bilateral Hyperplasia","Carcinoma"}), \\
\text{pch}=c(16,22)) \\
\text{legend}(\text{"topleft"}, c(\text{"Logistic","QDA","SVM"}), \text{lty}=c(1,2,5))
\]

Reading in the data:

\[
\text{rm(list = ls())} \\
\text{cush <- read.table(} \\
\text{"C:\E-drive\Books\LINMOD23\DATA\ALM12-1c.DAT",} \\
\text{sep="", col.names=c("Type","Tetra","Preg"))}
\]
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
13.7 Binary Discrimination

# solve quadratic equation
x1=seq(1,4.5,.1)
bb=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 \cdot Q1[,2]) - \text{sum}(Q2m2 \cdot Q2[,2])) + \\
(\text{sum}(Q2[,1] \cdot Q2[,2]) - \text{sum}(Q1[,1] \cdot Q1[,2])) \cdot x1
\]
\[a = 0.5 \cdot (\text{sum}(Q2[,2] \cdot Q2[,2]) - \text{sum}(Q1[,2] \cdot Q1[,2]))
\]
\[\text{delta} = b^2 - 4 \cdot a \cdot c
\]
\[y_{QDA} = \frac{-b + \sqrt{\text{delta}}}{2 \cdot a}
\]
\[y_{QDA2} = \frac{-b - \sqrt{\text{delta}}}{2 \cdot a}
\]
\[\text{lines}(x1, y_{QDA}, \text{type} = "l", \text{lty} = 2)
\]
\[\text{lines}(x1, y_{QDA2}, \text{type} = "l", \text{lty} = 2)
\]

### 13.7.2.4 Quadratic SVM

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

```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 the groups that each case was allocated to # fitted establishes 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 + \text{sum}(w) + 2 \cdot \text{sum}(w \cdot \text{fit}$SV[,1]) \cdot x1 + \\
\text{sum}(w \cdot (\text{fit}$SV[,1])^2) \cdot x1^2
b=2 \cdot \text{sum}(w \cdot \text{fit}$SV[,2]) + 2 \cdot \text{sum}(w \cdot \text{fit}$SV[,1] \cdot \text{fit}$SV[,2]) \cdot x1
a=\text{sum}(w \cdot (\text{fit}$SV[,2])^2)
\]
\[\text{delta} = b^2 - 4 \cdot a \cdot c
\]
\[y_{SVM} = \frac{-b + \sqrt{\text{delta}}}{2 \cdot a}
\]
\[y_{SVM2} = \frac{-b - \sqrt{\text{delta}}}{2 \cdot a}
\]
13.8 A simple example not in *ALMIII*

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

```r
lines(x1, ySVM, type="l", lty=5)
lines(x1, ySVM2, type="l", lty=5)

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

**Fig. 13.1** Simple Binary Data.

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

Fig. 13.3 Logistic Discrimination and R default SVM.
Fig. 13.4 Logistic Discrimination, R default SVM, LR augmented data ridge.

Fig. 13.5 Logistic Discrimination and SVM with C=100.
#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
b=a$cR$coef
yLRR=(b[1]+log(n2/n1)+log(prior/(1-prior)))+b[2]*x1/-b[3]
lines(x1,yLRR,type="l",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
```
98  14 Principal Components, Classical Multidimensional Scaling, and Factor Analysis

\[
bb1 = 1 + (0.8618708 \times K)
\]

\[
bb2 = 2 - (0.507128 \times K)
\]

\[
AA = \text{matrix}(\text{c}(1, 0.9, 0.9, 2), 2, 2, \text{dimnames} = \text{list(NULL, c("bb1", "bb2")))}
\]

\[
AA
\]

\[
EE <- \text{ellipse}(AA, \text{centre} = \text{c}(bb1, bb2), t = 0.95, \text{npoints} = 100)
\]

\[
bb = \text{seq}(0, 1.52, 0.01)
\]

\[
x = 1 + 0.507128 \times bb
\]

\[
y = 2 + 0.8618708 \times bb
\]

\[
bb = \text{seq}(-0.65, 0, 0.01)
\]

\[
x1 = 1 + (0.8618708 \times bb)
\]

\[
y1 = 2 - (0.507128 \times bb)
\]

\[
\text{plot}(E, \text{type} = 'l', \text{ylim} = \text{c}(0.5, 3.5), \text{xlim} = \text{c}(-1.5, 2.5), \text{xlab} = \text{expression}(y[1]), \text{ylab} = \text{expression}(y[2]), \text{main} = \text{"Principal Components"})
\]

\[
\text{text}(bb1 + 0.1, bb2 - 0.1, \text{expression}(\mu), lwd = 1, cex = 1)
\]

\[
\text{lines}(EE, \text{type} = 'l', lty = 1)
\]

\[
\text{lines}(bb1, bb2, \text{type} = 'p', pch = 19)
\]

\[
\text{lines}(x, y, \text{type} = 'l', lty = 1)
\]

\[
\text{text}(bb1 + 0.5, bb2 + 0.53, \text{expression}(a[1]), lwd = 1, cex = 1)
\]

\[
\text{text}(bb1 - 0.3, bb2 + 0.12, \text{expression}(a[2]), lwd = 1, cex = 1)
\]

\[
\#	ext{ The following plots 20 random data points over the ellipse.}
\]

\[
\text{library(mvtnorm)}
\]

\[
T = \text{rmvnorm}(20, c(1, 2), A)
\]

\[
\text{lines}(T[, 1], T[, 2], \text{type} = 'p')
\]

14.1.1 Normal Density Plot for PA-V

\[
\text{#install.packages("ellipse")}
\]

\[
\text{#Do this only once on your computer}
\]

\[
\text{library(ellipse)}
\]

\[
b1 = 1
\]

\[
b2 = 2
\]

\[
A = \text{matrix}(\text{c}(1, 0.9, 0.9, 2), 2, 2, \text{dimnames} = \text{list(NULL, c("b1", "b2")))}
\]

\[
A
\]

\[
E <- \text{ellipse}(A, \text{centre} = \text{c}(b1, b2), t = 0.95, \text{npoints} = 100)
\]

\[
E1 <- \text{ellipse}(A, \text{centre} = \text{c}(b1, b2), t = 0.5, \text{npoints} = 100)
\]
14.2 Sample Principal Components

There is a section of commands in the R code for ANREG2 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)
# 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$values
```
e$vectors
PC <- X %*% e$vec  #Matrix of principal component scores

14.2.1 Using Principal Components

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="l",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(-.4,.46),ylim=c(-.46,.46),
     main="CMDS: Squared Correlations", type="n")
text(x2, y2, labels = ID, cex=.9)
```

14.3.1 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

\[ Y = \text{cbind}(Y_1, Y_2, Y_3, Y_4) \]

ID = seq(1, 30)

Yd = dist(Y)  # default is

Yfit <- cmdscale(Yd, eig=TRUE, k=2)  # k is the number of dim

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.

```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
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  
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",
     xlim=c(-.46,.35),ylim=c(-.26,.26),
     main="isoMDS: Correlations", type="n")
text(x, y, labels = ID, cex=.9)
```

14.4 Factor Analysis

An overview of R procedures for multivariate analysis is available at 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
R = matrix(c(1.000, 0.439, 0.410, 0.288, 0.329, 0.248,
             0.439, 1.000, 0.351, 0.320, 0.329, 0.248,
             0.410, 0.351, 1.000, 0.164, 0.190, 0.181,
             0.288, 0.320, 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)
```

correlation data

```r
test = factanal(covmat = R, n.obs = 220, factors = 2, rotation = "none")
test
test$unique
test$loadings[, 1:2]
```

Illustration of maximum likelihood estimation from raw data: male turtle shell data with one factor. **Not in book.**

```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)
test = factanal(X, factors = 1, rotation = "none")
```
14.4 Factor Analysis

\begin{verbatim}
test=factanal(~x1+x2+x3,factors=1,rotation="none")
\end{verbatim}

14.4.2 Maximum Likelihood Theory

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

\begin{verbatim}
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))

# 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
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)
\end{verbatim}
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 = 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)

#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$vepea)
plotnScree(ns)
# PCA Variable Factor Map

library(FactoMineR)

result <- PCA(mydata) # graphs generated automatically
Chapter 15
Other Topics

This chapter contains short explanations and commands for Multivariate Analysis procedures that do not appear in *ALM-III*.

15.1 Hierarchical Cluster Analysis

15.1.1 Background

Cluster analysis takes an unstructured data matrix $Y_{n \times q}$ and turns it into a one-way MANOVA data structure that places every observation into a group. The process is hierarchical in that it starts (stage $s = n$) with every row of $Y$ constituting a different group/cluster. At stage $s = n - 1$, it combines the two closest observations into one cluster, to give $n - 1$ clusters. At stage $s = n - 2$, it combines the two clusters that are closest together to give $n - 2$ clusters. When $s = 1$, everything is together in one cluster.

There are two things that need to be specified: (1) the distance between two observations and (2) the distance between two clusters. There are several ways to define the distance between two observations and there are several ways to use observation distances to define cluster differences.

Hierarchical clustering defines a sequence of one-way MANOVA data structures. For each $s = n, \ldots, 1$ define the data structure

$$y_{sij}, \quad i = 1, \ldots, s, \quad j = 1, \ldots, N_{sij}.$$  

At every stage $s$, $n = N_{s1} + \cdots + N_{ss}$.

Code is given in the final subsection.
15.1.1.1 Pointwise distance measures

The squared Euclidean distance between two vectors \( y = (y_1, \ldots, y_q)' \) and \( w = (y - w)'(y - w) \). In general, we can define the norm of a vector, say, \( \|y\| \) and define the distance between two vectors as \( \|y - w\| \). A frequently used class of norms are the \( L^p \) norms. For \( p \geq 1 \) the \( L^p \) norm is defined as

\[
\|y\|_p \equiv \left( \sum_{h=1}^{q} |y_h|^p \right)^{1/p}.
\]

Also, the \( L^\infty \) norm is defined as

\[
\|y\|_\infty \equiv \max\{ |y_1|, |y_2|, \ldots, |y_q| \}.
\]

The \( L^2 \) norm gives Euclidean distance. Probably the second most frequently used \( L^p \) norm is \( L^1 \). Finally, for a positive definite matrix \( W \) one can define a Mahalanobis norm via

\[
\|y\|_W^2 \equiv y'Wy.
\]

Typically, \( W \) is chosen as the inverse of some estimate for the covariance matrix of \( y \).

15.1.1.2 Clusterwise “distance” measures

Now that we can talk about the distance between two observations, \( \|y - w\| \), we can discuss alternative definitions of the distance between two clusters of points. We do this for a fixed number of clusters, so we suppress the subscript \( s \) in the data. None of these measures of cluster distance satisfy the mathematical definition of a distance measure. Measures are summarized in Figure 15.1

<table>
<thead>
<tr>
<th>Linkage Method</th>
<th>Distance Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple (Nearest Neighbor)</td>
<td>( D_S(C_i, C_k) = \min_{j,h} |y_{ij} - y_{kh}| )</td>
</tr>
<tr>
<td>Complete (Farthest Neighbor)</td>
<td>( D_C(C_i, C_k) = \max_{j,h} |y_{ij} - y_{kh}| )</td>
</tr>
<tr>
<td>Average</td>
<td>( D_A(C_i, C_k) = \frac{1}{N_i} \frac{1}{N_k} \sum_{i=1}^{N_i} \sum_{h=1}^{N_k} |y_{ij} - y_{kh}| )</td>
</tr>
<tr>
<td>Centroid</td>
<td>( D_C(C_i, C_k) = |\bar{y}_i - y_k| )</td>
</tr>
<tr>
<td>Ward</td>
<td>( D_W(C_i, C_k) ) equation (1)</td>
</tr>
</tbody>
</table>
15.1.3 Ward

Ward’s method is to join the pair of clusters that minimizes the increase in sum of cluster variances. At first step, all cluster variances are 0. Pick the two points closest together.

I think this is what Ward is doing. Find the smallest value of

\[
D_{Wd}(C_i, C_k) = \sum_{r \neq i,k} \text{tr}(S_r)^2 + \frac{N_i - 1}{N_i + N_k - 1} \text{tr}(S_i)^2 + \frac{N_k - 1}{N_i + N_k - 1} \text{tr}(S_k)^2 + \frac{N_i N_k}{(N_i + N_k)(N_i + N_k - 1)} (\bar{y}_i - \bar{y}_k) (\bar{y}_i - \bar{y}_k) \]

I illustrate clustering methods on one-way MANOVA data so we can look at how well the procedure reproduces the actual groups. I decided to illustrate the Cushing data because the small sample size makes it more amenable to comparing methods.

Fig. 15.1 Hierarchical clustering. Single and complete linkage. Cushing Syndrome Data. Five clusters identified.

15.1.2 Code

First you have to create an \( n \times n \) matrix of distances between every point. Use \texttt{dist} for that. The default pointwise distance measure in \texttt{dist method="euclidean"}. 
Fig. 15.2 Hierarchical clustering. Average and centroid linkage. Cushing Syndrome Data. Five clusters identified.

Fig. 15.3 Ward hierarchical clustering: Cushing Syndrome Data. Five clusters identified.
Other options are that $L^\infty$ is maximum, $L^1$ is manhattan, $L^p$ for, say, $p = 1.5$ is minkowski, $p=1.5$. If you wanted to use Mahalanobis, I think that you can, (1) find $W = QQ'$, (2) transform the data matrix $Y$ into $YQ$, and (3) do Euclidean on the transformed matrix **but I have not tried that**.

```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
CSd

The actual clustering is done in hclust. Specifying the linkage method is straightforward except that R seems to think that the old ward.D is wrong and the new version ward.D2 is correct. They give the same answers for the Cushing syndrome data.

```r
par(mfrow=c(2,1))
CS1 =hclust(CSd, method = "single")
plot(CS1,labels=ID,main=NULL)
groups <- cutree(CS1, k=5) # cut tree into 5 clusters
rect.hclust(CS1, k=5, border="red")

CS1 =hclust(CSd, method = "complete")
plot(CS1,labels=ID,main=NULL)
groups <- cutree(CS1, k=5) # cut tree into 5 clusters
rect.hclust(CS1, k=5, border="red")

CS1 =hclust(CSd, method = "average")
plot(CS1,labels=ID,main=NULL)
groups <- cutree(CS1, k=5) # cut tree into 5 clusters
rect.hclust(CS1, k=5, border="red")

CS1 =hclust(CSd, method = "centroid")
plot(CS1,labels=ID,main=NULL)
groups <- cutree(CS1, k=5) # cut tree into 5 clusters
```
rect.hclust(CS1, k=5, border="red")

par(mfrow=c(2,1))
CS1 =hclust(CSd, method = "ward.D")
plot(CS1,labels=ID)
groups <- cutree(CS1, k=5) # cut tree into 5 clusters
rect.hclust(CS1, k=5, border="red")

CS1 =hclust(CSd, method = "ward.D2")
plot(CS1,labels=ID)
groups <- cutree(CS1, k=5) # cut tree into 5 clusters
rect.hclust(CS1, k=5, border="red")
par(mfrow=c(1,1))

15.2 K-means Clustering

The data matrix $Y$ has rows $y_k$, $k = 1, \ldots, n$. Similar to hierarchical clustering, at each stage of the process, say $s$, we will create repeated versions of one-way MANOVA data, but unlike hierarchical clustering, the number of clusters always remains $K$, i.e.,

$$y_{ij}, \quad i = 1, \ldots, K, \quad j = 1, \ldots, N_s.$$ 

At every stage $s$, $n = N_{s1} + \cdots + N_{sK}$. We will use both notations simultaneously.

Pick $K$ points, perhaps randomly, as cluster centers. Call these, $\bar{y}_{0i}, i = 1, \ldots, K$. Apparently, the process is very sensitive to how these initial points are selected. Assign $y_k$ to the cluster that minimizes, $||y_k - \bar{y}_{0i}||$. This defines the first set of MANOVA data,

$$y_{1ij}, \quad i = 1, \ldots, K, \quad j = 1, \ldots, N_{1i}.$$ 

Compute the cluster means $\bar{y}_{1i} = (1/N_{1i}) \sum_{j=1}^{N_{1i}}$. Assign $y_k$ to the cluster that minimizes, $||y_k - \bar{y}_{1i}||$ and use this to define the next set of MANOVA data

$$y_{2ij}, \quad i = 1, \ldots, K, \quad j = 1, \ldots, N_{2i}.$$ 

Repeat this process until the clusters stop changing.

rmls <-
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)
fit <- kmeans(CS, 5)
# get cluster means
aggregate(CS, by = list(fit$cluster), FUN = mean)
# append cluster assignment
Clust = fit$cluster
data.frame(cbind(TL, PL, ID, Clust))
data.frame(cbind(ID[order(Clust)], Clust[order(Clust)]))

The default algorithm is algorithm = "Hartigan-Wong". Other choices are "MacQueen", "Lloyd", "Forgy" although the last two names define the same algorithm.
Index

, 56
ALM-III, 1
ANREG2, 1
Bates, 29
copying commands
problems, 56
Fox, 1
multivariate normal

sample, 55
rescaling, 23
Ripley, 1
scaling, 23
standardized variables, 23
tilde, 56
Venables, 1
Weisberg, 1