DATA MINING AND BUSINESS ANALYTICS WITH R

JOHANNES LEDOLTER

UNIVERSITY OF IOWA

WILEY 2013

Data Sets

Data sets used in this book can be downloaded from the author's website. The data are arranged in comma-separated values (CSV) Excel files, in plain text form with a header line. You should download the data from **http://www.biz.uiowa.edu/faculty/jledolter/DataMining** to your own computer. A few data sets are already part of various R packages, and those data sets can be accessed directly from R. The data sets are listed in the order they appear in the book.

Data sets in the text

```
births2006.smpl          (in R package nutshell)
contribution.csv
oj.csv
FuelEfficiency.csv
ToyotaCorolla.csv
OldFaithful.csv
ethanol.csv
prostate.csv
DeathPenalty.csv
DeathPenaltyOther.csv
FlightDelays.csv
UniversalBank.csv
germancredit.csv
germancreditDescription (a word file describing the variables)
fgl                      (in R package MASS)
iris3                    (in R package MASS)
admission.csv
mcycle                   (in R package MASS)
protein.csv
unempstates.csv
unemp.csv
adj3unempstates.csv
```

```
lastfm.csv
AdultUCI                (in R package arules)
w8there                 (in R package textir)
congress109             (in R package textir)
firenze.csv
faux.mesa.high          (in R package statnet)
```

Data sets for the exercises

```
HousePrices.csv
DirectMarketing.csv
GenderDiscrimination.csv
LoanData.csv data
FinancialIndicators.csv
weather.csv
weatherAUS.csv
audit.csv
cup98LRN_csv.zip
cup98VAL_csv.zip
cup98LRN.csv
cup98VAL.csv
cup98VALtargt.csv
byssinosisWeights.csv
toxaemiaWeights.csv
soybean15.csv
ContactLens.csv
credit.csv
hepatitis.csv
labor.csv
PimaIndians.csv
cpu.csv
wine.csv
```

A note about reading data into R programs

You can use the **read.csv** command to read the comma-separated values (CSV) Excel files into R. Use the following command if you have stored the data files on your computer in directory C:/DataMining/Data:

```
FuelEff <- read.csv("C:/DataMining/Data/FuelEfficiency.csv")
```

R packages used

In this text we use several R packages. These packages must be installed and loaded before they can be used.

ares	arules	car	class	cluster	ellipse	igraph	lars
lattice	leaps	locfit	MASS	mixOmics	mixtools	nutshell	
ROCR	statnet	textir	tree	VGAM			

Reference Materials for R

There are many helpful books on how to use R. References that I have found useful are listed below. You can also use the help function in R to learn about packages and commands.

Adler, J.: R In a Nutshell: A Desktop Quick Reference. O'Reilly Media, 2010.

Albert, J. and Rizzo, M.: R by Example (Use R!). New York: Springer, 2012.

Crawley, M.J.: The R Book. New York: Wiley, 2007.

Kabacoff, R.I.: R In Action: Data Analysis and Graphics with R. Greenwich, CT: Manning Publications, 2011.

Maindonald, J.H.: Using R for Data Analysis and Graphics: Introduction, Code and Commentary, 2008. http://cran.r-project.org/doc/contrib/usingR.pdf (free resource)

Matloff, N.: The Art of R Programming: A Tour of Statistical Software Design. No Starch Press, 2011.

Murrell, P.: R Graphics. Chapman & Hall, 2005. http://www.stat.auckland.ac.nz/~paul/RGraphics/rgraphics.html (free resource)

Spector, P.: Data Manipulation with R (Use R!). New York: Springer, 2008.

Teetor, P.: R Cookbook. O'Reilly Media, 2011.

Torgo, L.: Data Mining with R: Learning with Case Studies. Chapman & Hall, 2010.

Venables, W.N., Smith, D.M., and the R Core Team: An Introduction to R, 2012.
http://cran.r-project.org/doc/manuals/R-intro.pdf (free resource)

R PROGRAMS

The source code can be found in the text file **LedolterDataMiningWileyRCodeApril2013**

CHAPTER 2: PROCESSING THE INFORMATION AND GETTING TO KNOW YOUR DATA

Example 1: 2006 Birth Data

```
## Install packages from CRAN; use any USA mirror
library(lattice)
library(nutshell)
data(births2006.smpl)
births2006.smpl[1:5,]
dim(births2006.smpl)

births.dow=table(births2006.smpl$DOB_WK)
births.dow
barchart(births.dow,ylab="Day of Week",col="black")
## for color, use col="red" or omit the color argument
dob.dm.tbl=table(WK=births2006.smpl$DOB_WK,MM=births2006.smpl$DMETH_REC)
dob.dm.tbl
dob.dm.tbl=dob.dm.tbl[,-2]
dob.dm.tbl
trellis.device()
barchart(dob.dm.tbl,ylab="Day of Week")
barchart(dob.dm.tbl,horizontal=FALSE,groups=FALSE,xlab="Day of
Week",col="black")
## for color, omit the color argument

histogram(~DBWT|DPLURAL,data=births2006.smpl,layout=c(1,5),col="black")
histogram(~DBWT|DMETH_REC,data=births2006.smpl,layout=c(1,3),col="black")
densityplot(~DBWT|
DPLURAL,data=births2006.smpl,layout=c(1,5),plot.points=FALSE,col="black")
densityplot(~DBWT,groups=DPLURAL,data=births2006.smpl,plot.points=FALSE)
dotplot(~DBWT|
DPLURAL,data=births2006.smpl,layout=c(1,5),plot.points=FALSE,col="black")

xyplot(DBWT~DOB_WK,data=births2006.smpl,col="black")
xyplot(DBWT~DOB_WK|DPLURAL,data=births2006.smpl,layout=c(1,5),col="black")
xyplot(DBWT~WTGAIN,data=births2006.smpl,col="black")
xyplot(DBWT~WTGAIN|DPLURAL,data=births2006.smpl,layout=c(1,5),col="black")
smoothScatter(births2006.smpl$WTGAIN,births2006.smpl$DBWT)

## boxplot is the command for a box plot in the standard graphics
## package
boxplot(DBWT~APGAR5,data=births2006.smpl,ylab="DBWT",xlab="AGPAR5")
boxplot(DBWT~DOB_WK,data=births2006.smpl,ylab="DBWT",xlab="Day of Week")
## bwplot is the command for a box plot in the lattice graphics
## package. There you need to declare the conditioning variables as
```

```
## factors
bwplot(DBWT~factor(APGAR5)|factor(SEX),data=births2006.smpl,xlab="AGPAR5")
bwplot(DBWT~factor(DOB_WK),data=births2006.smpl,xlab="Day of Week")

fac=factor(births2006.smpl$DPLURAL)
res=births2006.smpl$DBWT
t4=tapply(res,fac,mean,na.rm=TRUE)
t4
t5=tapply(births2006.smpl$DBWT,INDEX=list(births2006.smpl$DPLURAL,births2006.s
mpl$SEX),FUN=mean,na.rm=TRUE)
t5
barplot(t4,ylab="DBWT")
barplot(t5,beside=TRUE,ylab="DBWT")

t5=table(births2006.smpl$ESTGEST)
t5
new=births2006.smpl[births2006.smpl$ESTGEST != 99,]
t51=table(new$ESTGEST)
t51
t6=tapply(new$DBWT,INDEX=list(cut(new$WTGAIN,breaks=10),cut(new$ESTGEST,breaks
=10)),FUN=mean,na.rm=TRUE)
t6
levelplot(t6,scales = list(x = list(rot = 90)))
contourplot(t6,scales = list(x = list(rot = 90)))
```

Example 2: Alumni Donations

```
## Install packages from CRAN; use any USA mirror
library(lattice)
don <- read.csv("C:/DataMining/Data/contribution.csv")
don[1:5,]
table(don$Class.Year)
barchart(table(don$Class.Year),horizontal=FALSE,xlab="Class Year",col="black")

don$TGiving=don$FY00Giving+don$FY01Giving+don$FY02Giving+don$FY03Giving+don$FY
04Giving
mean(don$TGiving)
sd(don$TGiving)
quantile(don$TGiving,probs=seq(0,1,0.05))
quantile(don$TGiving,probs=seq(0.95,1,0.01))
hist(don$TGiving)
hist(don$TGiving[don$TGiving!=0][don$TGiving[don$TGiving!=0]<=1000])

## or, if you want to achieve the above histogram slower in two steps
## ff1=don$TGiving[don$TGiving!=0]
## ff1
## ff2=ff1[ff1<=1000]
## ff2
## hist(ff2,main=paste("Histogram of TGivingTrunc"),xlab="TGivingTrunc")

boxplot(don$TGiving,horizontal=TRUE,xlab="Total Contribution")
boxplot(don$TGiving,outline=FALSE,horizontal=TRUE,xlab="Total Contribution")

ddd=don[don$TGiving>=30000,]
ddd
ddd1=ddd[,c(1:5,12)]
ddd1
ddd1[order(ddd1$TGiving,decreasing=TRUE),]

boxplot(TGiving~Class.Year,data=don,outline=FALSE)
boxplot(TGiving~Gender,data=don,outline=FALSE)
boxplot(TGiving~Marital.Status,data=don,outline=FALSE)
boxplot(TGiving~AttendenceEvent,data=don,outline=FALSE)

t4=tapply(don$TGiving,don$Major,mean,na.rm=TRUE)
t4
t5=table(don$Major)
t5
t6=cbind(t4,t5)
t7=t6[t6[,2]>10,]
t7[order(t7[,1],decreasing=TRUE),]
barchart(t7[,1],col="black")

t4=tapply(don$TGiving,don$Next.Degree,mean,na.rm=TRUE)
t4
t5=table(don$Next.Degree)
t5
t6=cbind(t4,t5)
t7=t6[t6[,2]>10,]
t7[order(t7[,1],decreasing=TRUE),]
barchart(t7[,1],col="black")
```

```
densityplot(~TGiving|factor(Class.Year),
data=don[don$TGiving<=1000,][don[don$TGiving<=1000,]$TGiving>0,],
plot.points=FALSE,col="black")

t11=tapply(don$TGiving,don$Class.Year,FUN=sum,na.rm=TRUE)
t11
barplot(t11,ylab="Average Donation")

barchart(tapply(don$FY04Giving,don$Class.Year,FUN=sum,na.rm=TRUE),horizontal=F
ALSE,ylim=c(0,225000),col="black")
barchart(tapply(don$FY03Giving,don$Class.Year,FUN=sum,na.rm=TRUE),horizontal=
FALSE,ylim=c(0,225000),col="black")
barchart(tapply(don$FY02Giving,don$Class.Year,FUN=sum,na.rm=TRUE),horizontal=
FALSE,ylim=c(0,225000),col="black")
barchart(tapply(don$FY01Giving,don$Class.Year,FUN=sum,na.rm=TRUE),horizontal=
FALSE,ylim=c(0,225000),col="black")
barchart(tapply(don$FY00Giving,don$Class.Year,FUN=sum,na.rm=TRUE),horizontal=
FALSE,ylim=c(0,225000),col="black")

don$TGivingIND=cut(don$TGiving,c(-1,0.5,10000000),labels=FALSE)-1
mean(don$TGivingIND)
t5=table(don$TGivingIND,don$Class.Year)
t5
barplot(t5,beside=TRUE)
mosaicplot(factor(don$Class.Year)~factor(don$TGivingIND))
t50=tapply(don$TGivingIND,don$Class.Year,FUN=mean,na.rm=TRUE)
t50
barchart(t50,horizontal=FALSE,col="black")

don$FY04GivingIND=cut(don$FY04Giving,c(-1,0.5,10000000),labels=FALSE)-1
t51=tapply(don$FY04GivingIND,don$Class.Year,FUN=mean,na.rm=TRUE)
t51
barchart(t51,horizontal=FALSE,col="black")

Data=data.frame(don$FY04Giving,don$FY03Giving,don$FY02Giving,don$FY01Giving,do
n$FY00Giving)
correlation=cor(Data)
correlation
plot(Data)
library(ellipse)
plotcorr(correlation)

mosaicplot(factor(don$Gender)~factor(don$TGivingIND))
mosaicplot(factor(don$Marital.Status)~factor(don$TGivingIND))
t2=table(factor(don$Marital.Status),factor(don$TGivingIND))
mosaicplot(t2)
mosaicplot(factor(don$AttendenceEvent)~factor(don$TGivingIND))
t2=table(factor(don$Marital.Status),factor(don$TGivingIND),factor(don$Attenden
ceEvent))
t2
mosaicplot(t2[,,1])
mosaicplot(t2[,,2])
```

Example 3: Orange Juice

```
## Install packages from CRAN; use any USA mirror
library(lattice)
oj <- read.csv("C:/DataMining/Data/oj.csv")
oj$store <- factor(oj$store)
oj[1:2,]
t1=tapply(oj$logmove,oj$brand,FUN=mean,na.rm=TRUE)
t1
t2=tapply(oj$logmove,INDEX=list(oj$brand,oj$week),FUN=mean,na.rm=TRUE)
t2
plot(t2[1,],type= "l",xlab="week",ylab="dominicks",ylim=c(7,12))
plot(t2[2,],type= "l",xlab="week",ylab="minute.maid",ylim=c(7,12))
plot(t2[3,],type= "l",xlab="week",ylab="tropicana",ylim=c(7,12))
logmove=c(t2[1,],t2[2,],t2[3,])
week1=c(40:160)
week=c(week1,week1,week1)
brand1=rep(1,121)
brand2=rep(2,121)
brand3=rep(3,121)
brand=c(brand1,brand2,brand3)
xyplot(logmove~week|factor(brand),type= "l",layout=c(1,3),col="black")

boxplot(logmove~brand,data=oj)
histogram(~logmove|brand,data=oj,layout=c(1,3))
densityplot(~logmove|brand,data=oj,layout=c(1,3),plot.points=FALSE)
densityplot(~logmove,groups=brand,data=oj,plot.points=FALSE)

xyplot(logmove~week,data=oj,col="black")
xyplot(logmove~week|brand,data=oj,layout=c(1,3),col="black")
xyplot(logmove~price,data=oj,col="black")
xyplot(logmove~price|brand,data=oj,layout=c(1,3),col="black")
smoothScatter(oj$price,oj$logmove)

densityplot(~logmove,groups=feat, data=oj, plot.points=FALSE)
xyplot(logmove~price,groups=feat, data=oj)

oj1=oj[oj$store == 5,]
xyplot(logmove~week|brand,data=oj1,type="l",layout=c(1,3),col="black")
xyplot(logmove~price,data=oj1,col="black")
xyplot(logmove~price|brand,data=oj1,layout=c(1,3),col="black")
densityplot(~logmove|brand,groups=feat,data=oj1,plot.points=FALSE)
xyplot(logmove~price|brand,groups=feat,data=oj1)

t21=tapply(oj$INCOME,oj$store,FUN=mean,na.rm=TRUE)
t21
t21[t21==max(t21)]
t21[t21==min(t21)]

oj1=oj[oj$store == 62,]
oj2=oj[oj$store == 75,]
oj3=rbind(oj1,oj2)
xyplot(logmove~price|store,data=oj3)
xyplot(logmove~price|store,groups=feat,data=oj3)
## store in the wealthiest neighborhood
mhigh=lm(logmove~price,data=oj1)
```

```
summary(mhigh)
plot(logmove~price,data=oj1,xlim=c(0,4),ylim=c(0,13))
abline(mhigh)
## store in the poorest neighborhood
mlow=lm(logmove~price,data=oj2)
summary(mlow)
plot(logmove~price,data=oj2,xlim=c(0,4),ylim=c(0,13))
abline(mlow)
```

CHAPTER 3: STANDARD LINEAR REGRESSION

Example 1: Fuel Efficiency of Automobiles

```
## first we read in the data
FuelEff <- read.csv("C:/DataMining/Data/FuelEfficiency.csv")
FuelEff
plot(GPM~MPG,data=FuelEff)
plot(GPM~WT,data=FuelEff)
plot(GPM~DIS,data=FuelEff)
plot(GPM~NC,data=FuelEff)
plot(GPM~HP,data=FuelEff)
plot(GPM~ACC,data=FuelEff)
plot(GPM~ET,data=FuelEff)
FuelEff=FuelEff[-1]
FuelEff

## regression on all data
m1=lm(GPM~.,data=FuelEff)
summary(m1)

cor(FuelEff)

## best subset regression in R
library(leaps)
X=FuelEff[,2:7]
y=FuelEff[,1]
out=summary(regsubsets(X,y,nbest=2,nvmax=ncol(X)))
tab=cbind(out$which,out$rsq,out$adjr2,out$cp)
tab

m2=lm(GPM~WT,data=FuelEff)
summary(m2)

## cross-validation (leave one out) for the model on all six regressors
n=length(FuelEff$GPM)
diff=dim(n)
percdiff=dim(n)
for (k in 1:n) {
train1=c(1:n)
train=train1[train1!=k]
## the R expression "train1[train1!=k]" picks from train1 those
## elements that are different from k and stores those elements in the
## object train.
## For k=1, train consists of elements that are different from 1; that
## is 2, 3, …, n.
m1=lm(GPM~.,data=FuelEff[train,])
pred=predict(m1,newdat=FuelEff[-train,])
obs=FuelEff$GPM[-train]
diff[k]=obs-pred
percdiff[k]=abs(diff[k])/obs
}
me=mean(diff)
rmse=sqrt(mean(diff**2))
mape=100*(mean(percdiff))
```

```
me   # mean error
rmse # root mean square error
mape # mean absolute percent error

## cross-validation (leave one out) for the model on weight only
n=length(FuelEff$GPM)
diff=dim(n)
percdiff=dim(n)
for (k in 1:n) {
train1=c(1:n)
train=train1[train1!=k]
m2=lm(GPM~WT,data=FuelEff[train,])
pred=predict(m2,newdat=FuelEff[-train,])
obs=FuelEff$GPM[-train]
diff[k]=obs-pred
percdiff[k]=abs(diff[k])/obs
}
me=mean(diff)
rmse=sqrt(mean(diff**2))
mape=100*(mean(percdiff))
me   # mean error
rmse # root mean square error
mape # mean absolute percent error
```

Example 2: Toyota Used Car Prices

```
toyota <- read.csv("C:/DataMining/Data/ToyotaCorolla.csv")
toyota[1:3,]
summary(toyota)
hist(toyota$Price)
## next we create indicator variables for the categorical variable
## FuelType with its three nominal outcomes: CNG, Diesel, and Petrol
v1=rep(1,length(toyota$FuelType))
v2=rep(0,length(toyota$FuelType))
toyota$FuelType1=ifelse(toyota$FuelType=="CNG",v1,v2)
toyota$FuelType2=ifelse(toyota$FuelType=="Diesel",v1,v2)
auto=toyota[-4]
auto[1:3,]
plot(Price~Age,data=auto)
plot(Price~KM,data=auto)
plot(Price~HP,data=auto)
plot(Price~MetColor,data=auto)
plot(Price~Automatic,data=auto)
plot(Price~CC,data=auto)
plot(Price~Doors,data=auto)
plot(Price~Weight,data=auto)

## regression on all data
m1=lm(Price~.,data=auto)
summary(m1)

set.seed(1)
## fixing the seed value for the random selection guarantees the
## same results in repeated runs
n=length(auto$Price)
n1=1000
n2=n-n1
train=sample(1:n,n1)

## regression on training set
m1=lm(Price~.,data=auto[train,])
summary(m1)
pred=predict(m1,newdat=auto[-train,])
obs=auto$Price[-train]
diff=obs-pred
percdiff=abs(diff)/obs
me=mean(diff)
rmse=sqrt(sum(diff**2)/n2)
mape=100*(mean(percdiff))
me   # mean error
rmse # root mean square error
mape # mean absolute percent error

## cross-validation (leave one out)
n=length(auto$Price)
diff=dim(n)
percdiff=dim(n)
for (k in 1:n) {
train1=c(1:n)
train=train1[train1!=k]
```

```
m1=lm(Price~.,data=auto[train,])
pred=predict(m1,newdat=auto[-train,])
obs=auto$Price[-train]
diff[k]=obs-pred
percdiff[k]=abs(diff[k])/obs
}
me=mean(diff)
rmse=sqrt(mean(diff**2))
mape=100*(mean(percdiff))
me   # mean error
rmse # root mean square error
mape # mean absolute percent error

## cross-validation (leave one out): Model with just Age
n=length(auto$Price)
diff=dim(n)
percdiff=dim(n)
for (k in 1:n) {
train1=c(1:n)
train=train1[train1!=k]
m1=lm(Price~Age,data=auto[train,])
pred=predict(m1,newdat=auto[-train,])
obs=auto$Price[-train]
diff[k]=obs-pred
percdiff[k]=abs(diff[k])/obs
}
me=mean(diff)
rmse=sqrt(mean(diff**2))
mape=100*(mean(percdiff))
me   # mean error
rmse # root mean square error
mape # mean absolute percent error

## Adding the squares of Age and KM to the model
auto$Age2=auto$Age^2
auto$KM2=auto$KM^2
m11=lm(Price~Age+KM,data=auto)
summary(m11)
m12=lm(Price~Age+Age2+KM+KM2,data=auto)
summary(m12)
m13=lm(Price~Age+Age2+KM,data=auto)
summary(m13)
plot(m11$res~m11$fitted)
hist(m11$res)
plot(m12$res~m12$fitted)
```

CHAPTER 4: LOCAL POLYNOMIAL REGRESSION: A NONPARAMETRIC REGRESSION APPROACH

Example 1: Old Faithful

```
library(locfit)
## first we read in the data
OldFaithful <- read.csv("C:/DataMining/Data/OldFaithful.csv")
OldFaithful[1:3,]

## density histograms and smoothed density histograms
## time of eruption
hist(OldFaithful$TimeEruption,freq=FALSE)
fit1 <- locfit(~lp(TimeEruption),data=OldFaithful)
plot(fit1)

## waiting time to next eruption
hist(OldFaithful$TimeWaiting,freq=FALSE)
fit2 <- locfit(~lp(TimeWaiting),data=OldFaithful)
plot(fit2)

## experiment with different smoothing constants
fit2 <- locfit(~lp(TimeWaiting,nn=0.9,deg=2),data=OldFaithful)
plot(fit2)
fit2 <- locfit(~lp(TimeWaiting,nn=0.3,deg=2),data=OldFaithful)
plot(fit2)

## cross-validation of smoothing constant
## for waiting time to next eruption
alpha<-seq(0.20,1,by=0.01)
n1=length(alpha)
g=matrix(nrow=n1,ncol=4)
for (k in 1:length(alpha)) {
g[k,]<-gcv(~lp(TimeWaiting,nn=alpha[k]),data=OldFaithful)
}
g
plot(g[,4]~g[,3],ylab="GCV",xlab="degrees of freedom")
## minimum at nn = 0.66

fit2 <- locfit(~lp(TimeWaiting,nn=0.66,deg=2),data=OldFaithful)
plot(fit2)

## local polynomial regression of TimeEruption on TimeWaiting
plot(TimeWaiting~TimeEruption,data=OldFaithful)
# standard regression fit
fitreg=lm(TimeWaiting~TimeEruption,data=OldFaithful)
plot(TimeWaiting~TimeEruption,data=OldFaithful)
abline(fitreg)
# fit with nearest neighbor bandwidth
fit3 <- locfit(TimeWaiting~lp(TimeEruption),data=OldFaithful)
plot(fit3)
fit3 <- locfit(TimeWaiting~lp(TimeEruption,deg=1),data=OldFaithful)
plot(fit3)
```

```
fit3 <- locfit(TimeWaiting~lp(TimeEruption,deg=0),data=OldFaithful)
plot(fit3)
```

Example 2: NOx Exhaust Emissions

```
library(locfit)

## first we read in the data
ethanol <- read.csv("C:/DataMining/Data/ethanol.csv")
ethanol[1:3,]

## density histogram
hist(ethanol$NOx,freq=FALSE)
## smoothed density histogram
fit <- locfit(~lp(NOx),data=ethanol)
plot(fit)

## experiment with the parameters of locfit
fit <- locfit(~lp(NOx,deg=1),data=ethanol)
plot(fit)
fit <- locfit(~lp(NOx,nn=0.7,deg=1),data=ethanol)
plot(fit)
fit <- locfit(~lp(NOx,nn=0.5,deg=1),data=ethanol)
plot(fit)

fit <- locfit(~lp(NOx,deg=2),data=ethanol)
plot(fit)
fit <- locfit(~lp(NOx,nn=0.7,deg=2),data=ethanol)
plot(fit)
fit <- locfit(~lp(NOx,nn=0.5,deg=2),data=ethanol)
plot(fit)

fit <- locfit(~lp(NOx,deg=3),data=ethanol)
plot(fit)
fit <- locfit(~lp(NOx,nn=0.7,deg=3),data=ethanol)
plot(fit)
fit <- locfit(~lp(NOx,nn=0.5,deg=3),data=ethanol)
plot(fit)

## standard regression of NOx on the equivalence ratio
plot(NOx~EquivRatio,data=ethanol)
fitreg=lm(NOx~EquivRatio,data=ethanol)
plot(NOx~EquivRatio,data=ethanol)
abline(fitreg)

## local polynomial regression of NOx on the equivalence ratio
## fit with a 50% nearest neighbor bandwidth.
fit <- locfit(NOx~lp(EquivRatio,nn=0.5),data=ethanol)
plot(fit)

## experiment with the parameters of locfit
fit <- locfit(NOx~lp(EquivRatio,nn=0.2),data=ethanol)
plot(fit)
fit <- locfit(NOx~lp(EquivRatio,nn=0.8),data=ethanol)
plot(fit)
fit <- locfit(NOx~lp(EquivRatio,deg=1),data=ethanol)
plot(fit)
```

```
fit <- locfit(NOx~lp(EquivRatio,deg=2),data=ethanol)
plot(fit)
## cross-validation
alpha<-seq(0.20,1,by=0.01)
n1=length(alpha)
g=matrix(nrow=n1,ncol=4)
for (k in 1:length(alpha)) {
g[k,]<-gcv(NOx~lp(EquivRatio,nn=alpha[k]),data=ethanol)
}
g
plot(g[,4]~g[,3],ylab="GCV",xlab="degrees of freedom")
f1=locfit(NOx~lp(EquivRatio,nn=0.30),data=ethanol)
f1
plot(f1)

## local polynomial regression on both E and C
plot(NOx~EquivRatio,data=ethanol)
plot(NOx~CompRatio,data=ethanol)
fit <- locfit(NOx~lp(EquivRatio,CompRatio,scale=TRUE),data=ethanol)
plot(fit)

## experiment with the parameters of locfit
fit <- locfit(NOx~lp(EquivRatio,CompRatio,nn=0.5,scale=TRUE),data=ethanol)
plot(fit)
fit <- locfit(NOx~lp(EquivRatio,CompRatio,deg=0,scale=TRUE),data=ethanol)
plot(fit)
```

CHAPTER 5: IMPORTANCE OF PARSIMONY IN STATISTICAL MODELING

Example 1

```
## Example 1
## We specify a seed to make the results reproducible. Omitting the
## set.seed statement would lead to a different set of random numbers
## and the results would vary somewhat
set.seed(10)
alpha=0.10
m=100
p=dim(m)
index=dim(m)
for (i in 1:5) {
x=rnorm(25,1,1)
t=-abs(mean(x)/(sd(x)/sqrt(25)))
p[i]=2*pt(t,24)
index[i]=i
}
for (i in 6:m) {
x=rnorm(25)
t=-abs(mean(x)/(sd(x)/sqrt(25)))
p[i]=2*pt(t,24)
index[i]=i
}
count=p<=0.05
table(count)
ps=sort(p)
logps=log(ps)
logindex=log(index)
y=log(index*alpha/m)
plot(logps~logindex,xlab="log(j)",ylab="log(ProbValue(j))",main="False
Discovery Rate")
points(y~logindex,type="l")
ps
ps[6]
```

Example 2

```
## Example 2
set.seed(10)
alpha=0.20
m=500
p=dim(m)
index=dim(m)
for (i in 1:5) {
x=rnorm(25,1,1)
t=-abs(mean(x)/(sd(x)/sqrt(25)))
p[i]=2*pt(t,24)
index[i]=i
}
for (i in 6:m) {
x=rnorm(25)
t=-abs(mean(x)/(sd(x)/sqrt(25)))
p[i]=2*pt(t,24)
index[i]=i
}
count=p<=0.05
table(count)
ps=sort(p)
logps=log(ps)
logindex=log(index)
y=log(index*alpha/m)
plot(logps~logindex,xlab="log(j)",ylab="log(ProbValue(j))",main="False
Discovery Rate")
points(y~logindex,type="l")
ps
ps[7]
```

CHAPTER 6: PENALTY-BASED VARIABLE SELECTION IN REGRESSION MODELS WITH MANY PARAMETERS (LASSO)

Example 1: Prostate Cancer

```
prostate <- read.csv("C:/DataMining/Data/prostate.csv")
prostate[1:3,]
m1=lm(lcavol~.,data=prostate)
summary(m1)
## the model.matrix statement defines the model to be fitted
x <- model.matrix(lcavol~age+lbph+lcp+gleason+lpsa,data=prostate)
x=x[,-1]
## stripping off the column of 1s as LASSO includes the intercept
## automatically
library(lars)
## lasso on all data
lasso <- lars(x=x,y=prostate$lcavol,trace=TRUE)
## trace of lasso (standardized) coefficients for varying penalty
plot(lasso)
lasso
coef(lasso,s=c(.25,.50,0.75,1.0),mode="fraction")
## cross-validation using 10 folds
cv.lars(x=x,y=prostate$lcavol,K=10)
## another way to evaluate lasso's out-of-sample prediction performance
MSElasso25=dim(10)
MSElasso50=dim(10)
MSElasso75=dim(10)
MSElasso100=dim(10)
set.seed(1)
for(i in 1:10){
      train <- sample(1:nrow(prostate),80)
      lasso <- lars(x=x[train,],y=prostate$lcavol[train])
      MSElasso25[i]=
            mean((predict(lasso,x[-train,],s=.25,mode="fraction")$fit-
            prostate$lcavol[-train])^2)
      MSElasso50[i]=
            mean((predict(lasso,x[-train,],s=.50,mode="fraction")$fit-
            prostate$lcavol[-train])^2)
      MSElasso75[i]=
            mean((predict(lasso,x[-train,],s=.75,mode="fraction")$fit-
            prostate$lcavol[-train])^2)
      MSElasso100[i]=
            mean((predict(lasso,x[-train,],s=1.00,mode="fraction")$fit-
            prostate$lcavol[-train])^2)
      }
mean(MSElasso25)
mean(MSElasso50)
mean(MSElasso75)
mean(MSElasso100)
boxplot(MSElasso25,MSElasso50,MSElasso75,MSElasso100,ylab="MSE", sub="LASSO
model",xlab="s=0.25           s=0.50           s=0.75          s=1.0(LS)")
```

Example 2: Orange Juice

```
oj <- read.csv("C:/DataMining/Data/oj.csv")
oj$store <- factor(oj$store)
oj[1:2,]
x <- model.matrix(logmove ~ log(price)*(feat + brand
   + AGE60 + EDUC + ETHNIC + INCOME + HHLARGE + WORKWOM
   + HVAL150 + SSTRDIST + SSTRVOL + CPDIST5 + CPWVOL5)^2, data=oj)
dim(x)

## First column of x consists of ones (the intercept)
## We strip the column of ones as intercept is included automatically
x=x[,-1]
## We normalize the covariates as they are of very different magnitudes
## Each normalized covariate has mean 0 and standard deviation 1
for (j in 1:209) {
x[,j]=(x[,j]-mean(x[,j]))/sd(x[,j])
}

## One could consider the standard regression model
reg <- lm(oj$logmove~x)
summary(reg)
p0=predict(reg)

## Or, one could consider LASSO
library(lars)
lasso <- lars(x=x, y=oj$logmove, trace=TRUE)
coef(lasso, s=c(.25,.50,0.75,1.00), mode="fraction")
## creates LASSO estimates as function of lambda
## gives you the estimates for four shrinkage coef

## Check that predictions in regression and lars (s=1) are the same
p1=predict(lasso,x,s=1,mode="fraction")
p1$fit
pdiff=p1$fit-p0
pdiff  ## zero differences

## out of sample prediction; estimate model on 20,000 rows
MSElasso10=dim(10)
MSElasso50=dim(10)
MSElasso90=dim(10)
MSElasso100=dim(10)
set.seed(1) ## fixes seed to make random draws reproducible
for(i in 1:10){
      train <- sample(1:nrow(oj), 20000)
      lasso <- lars(x=x[train,], y=oj$logmove[train])
      MSElasso10[i]=
            mean((predict(lasso,x[-train,], s=.10, mode="fraction")
            $fit - oj$logmove[-train])^2)
      MSElasso50[i]=
            mean((predict(lasso,x[-train,], s=.50, mode="fraction")
            $fit - oj$logmove[-train])^2)
      MSElasso90[i]=
            mean((predict(lasso,x[-train,], s=.90, mode="fraction")
            $fit - oj$logmove[-train])^2)
      MSElasso100[i]=
```

```
            mean((predict(lasso,x[-train,], s=1.0, mode="fraction")
            $fit - oj$logmove[-train])^2)
            }
mean(MSElasso10)
mean(MSElasso50)
mean(MSElasso90)
mean(MSElasso100)
boxplot(MSElasso10,MSElasso50,MSElasso90,MSElasso100,ylab="MSE", sub="LASSO
model",xlab="s=0.10           s=0.50            s=0.9          s=1.0(LS)")

## out of sample prediction; estimate model on 1,000 rows
set.seed(1) ## fixes seed to make random draws reproducible
for(i in 1:10){
      train <- sample(1:nrow(oj), 1000)
      lasso <- lars(x=x[train,], y=oj$logmove[train])
      MSElasso10[i]=
            mean((predict(lasso,x[-train,], s=.10, mode="fraction")
            $fit - oj$logmove[-train])^2)
      MSElasso50[i]=
            mean((predict(lasso,x[-train,], s=.50, mode="fraction")
            $fit - oj$logmove[-train])^2)
      MSElasso90[i]=
            mean((predict(lasso,x[-train,], s=.90, mode="fraction")
            $fit - oj$logmove[-train])^2)
      MSElasso100[i]=
            mean((predict(lasso,x[-train,], s=1.0, mode="fraction")
            $fit - oj$logmove[-train])^2)
            }
mean(MSElasso10)
mean(MSElasso50)
mean(MSElasso90)
mean(MSElasso100)
boxplot(MSElasso10,MSElasso50,MSElasso90,MSElasso100,ylab="MSE", sub="LASSO
model",xlab="s=0.10           s=0.50            s=0.9          s=1.0(LS)")
```

CHAPTER 7: LOGISTIC REGRESSION

Example 1: Death Penalty Data

```
## analyzing individual observations
dpen <- read.csv("C:/DataMining/Data/DeathPenalty.csv")
dpen[1:4,]
dpen[359:362,]
m1=glm(Death~VRace+Agg,family=binomial,data=dpen)
m1
summary(m1)
## calculating logits
exp(m1$coef[2])
exp(m1$coef[3])
## plotting probability of getting death penalty as a function of aggravation
## separately for black (in black) and white (in red) victim
fitBlack=dim(501)
fitWhite=dim(501)
ag=dim(501)
for (i in 1:501) {
ag[i]=(99+i)/100
fitBlack[i]=exp(m1$coef[1]+ag[i]*m1$coef[3])/
(1+exp(m1$coef[1]+ag[i]*m1$coef[3]))
fitWhite[i]=exp(m1$coef[1]+m1$coef[2]+ag[i]*m1$coef[3])/
(1+exp(m1$coef[1]+m1$coef[2]+ag[i]*m1$coef[3]))
}
plot(fitBlack~ag,type="l",col="black",ylab="Prob[Death]",xlab="Aggravation",yl
im=c(0,1),main="red line for white victim; black line for black victim")
points(fitWhite~ag,type="l",col="red")

## analyzing summarized data
dpenother <- read.csv("C:/DataMining/Data/DeathPenaltyOther.csv")
dpenother
m1=glm(Death~VRace+Agg,family=binomial,weights=Freq,data=dpenother)
m1
summary(m1)
exp(m1$coef[2])
exp(m1$coef[3])
```

Example 2: Delayed Airplanes

```
library(car) ## needed to recode variables
set.seed(1)

## read and print the data
del <- read.csv("C:/DataMining/Data/FlightDelays.csv")
del[1:3,]

## define hours of departure
del$sched=factor(floor(del$schedtime/100))
table(del$sched)
table(del$carrier)
table(del$dest)
table(del$origin)
table(del$weather)
table(del$dayweek)
table(del$daymonth)
table(del$delay)
del$delay=recode(del$delay,"'delayed'=1;else=0")
del$delay=as.numeric(levels(del$delay)[del$delay])
table(del$delay)
## Delay: 1=Monday; 2=Tuesday; 3=Wednesday; 4=Thursday;
## 5=Friday; 6=Saturday; 7=Sunday
## 7=Sunday and 1=Monday coded as 1
del$dayweek=recode(del$dayweek,"c(1,7)=1;else=0")
table(del$dayweek)
## omit unused variables
del=del[,c(-1,-3,-5,-6,-7,-11,-12)]
del[1:3,]
n=length(del$delay)
n
n1=floor(n*(0.6))
n1
n2=n-n1
n2
train=sample(1:n,n1)

## estimation of the logistic regression model
## explanatory variables: carrier, destination, origin, weather, day of week
## (weekday/weekend), scheduled hour of departure
## create design matrix; indicators for categorical variables (factors)
Xdel <- model.matrix(delay~.,data=del)[,-1]
Xdel[1:3,]
xtrain <- Xdel[train,]
xnew <- Xdel[-train,]
ytrain <- del$delay[train]
ynew <- del$delay[-train]
m1=glm(delay~.,family=binomial,data=data.frame(delay=ytrain,xtrain))
summary(m1)

## prediction: predicted default probabilities for cases in test set
ptest <- predict(m1,newdata=data.frame(xnew),type="response")
data.frame(ynew,ptest)[1:10,]
## first column in list represents the case number of the test element
plot(ynew~ptest)
```

```
## coding as 1 if probability 0.5 or larger
gg1=floor(ptest+0.5)    ## floor function; see help command
ttt=table(ynew,gg1)
ttt
error=(ttt[1,2]+ttt[2,1])/n2
error

## coding as 1 if probability 0.3 or larger
gg2=floor(ptest+0.7)
ttt=table(ynew,gg2)
ttt
error=(ttt[1,2]+ttt[2,1])/n2
error

bb=cbind(ptest,ynew)
bb
bb1=bb[order(ptest,decreasing=TRUE),]
bb1
## order cases in test set according to their success prob
## actual outcome shown next to it

## overall success (delay) prob in the evaluation data set
xbar=mean(ynew)
xbar

## calculating the lift
## cumulative 1's sorted by predicted values
## cumulative 1's using the average success prob from evaluation set
axis=dim(n2)
ax=dim(n2)
ay=dim(n2)
axis[1]=1
ax[1]=xbar
ay[1]=bb1[1,2]
for (i in 2:n2) {
axis[i]=i
ax[i]=xbar*i
ay[i]=ay[i-1]+bb1[i,2]
}

aaa=cbind(bb1[,1],bb1[,2],ay,ax)
aaa[1:100,]

plot(axis,ay,xlab="number of cases",ylab="number of successes",main="Lift: Cum
successes sorted by pred val/success prob")
points(axis,ax,type="l")
```

Example 3: Loan Acceptance

```
library(car) ## needed to recode variables
set.seed(1)

loan <- read.csv("C:/DataMining/Data/UniversalBank.csv")
loan[1:3,]

## familiarize yourself with the data
hist(loan$Age)
hist(loan$Experience)
hist(loan$Income)
hist(loan$Family) ## below we treat loan$Family as categorical
hist(loan$CCAvg)
hist(loan$Mortgage)
hist(loan$SecuritiesAccount)
hist(loan$CDAccount)
hist(loan$Online)
hist(loan$CreditCard)
hist(loan$Education) ## below we treat loan$Education as categorical
response=loan$PersonalLoan
hist(response)
MeanRes=mean(response)
MeanRes

## creating indicator variables for loan$Family and loan$Education
v1=rep(1,dim(loan)[1])
v2=rep(0,dim(loan)[1])
## creating indicator variables for family size (4 groups: 1, 2, 3, 4)
loan$FamSize2=ifelse(loan$Family==2,v1,v2)
loan$FamSize3=ifelse(loan$Family==3,v1,v2)
loan$FamSize4=ifelse(loan$Family==4,v1,v2)
## creating indicator variables for education level (3 groups: 1, 2, 3)
loan$Educ2=ifelse(loan$Education==2,v1,v2)
loan$Educ3=ifelse(loan$Education==3,v1,v2)

xx=cbind(response,Age=loan$Age,Exp=loan$Experience,Inc=loan$Income,Fam2=loan$F
amSize2,Fam3=loan$FamSize3,Fam4=loan$FamSize4,CCAve=loan$CCAvg,Mort=loan$Mortg
age,SecAcc=loan$SecuritiesAccount,CD=loan$CDAccount,Online=loan$Online,CreditC
ard=loan$CreditCard,Educ2=loan$Educ2,Educ3=loan$Educ3)
xx[1:3,]

## split the data set into training and test (evaluation) set
n=dim(loan)[1]
n
n1=floor(n*(0.6))
n1
n2=n-n1
n2
train=sample(1:n,n1)

## model fitted on all data
m1=glm(response~.,family=binomial,data=data.frame(xx))
summary(m1)

xx=xx[,-1]
```

```
xtrain <- xx[train,]
xnew <- xx[-train,]
ytrain <- response[train]
ynew <- response[-train]

## model fitted on the training data set
m2=glm(response~.,family=binomial,data=data.frame(response=ytrain,xtrain))
summary(m2)

## create predictions for the test (evaluation) data set
ptest=predict(m2,newdata=data.frame(xnew),type="response")
## predicted probabilities
hist(ptest)
plot(ynew~ptest)

## coding as 1 if probability 0.5 or larger
gg1=floor(ptest+0.5)
ttt=table(ynew,gg1)
ttt
error=(ttt[1,2]+ttt[2,1])/n2
error

## coding as 1 if probability 0.3 or larger
gg2=floor(ptest+0.7)
ttt=table(ynew,gg2)
ttt
error=(ttt[1,2]+ttt[2,1])/n2
error

bb=cbind(ptest,ynew)
bb
bb1=bb[order(ptest,decreasing=TRUE),]
bb1
## order cases in test set according to their success prob
## actual outcome shown next to it

## overall success probability in evaluation (test) data set
xbar=mean(ynew)
xbar

## calculating the lift
## cumulative 1's sorted by predicted values
## cumulative 1's using the average success prob from evaluation set
axis=dim(n2)
ax=dim(n2)
ay=dim(n2)
axis[1]=1
ax[1]=xbar
ay[1]=bb1[1,2]
for (i in 2:n2) {
axis[i]=i
ax[i]=xbar*i
ay[i]=ay[i-1]+bb1[i,2]
}

aaa=cbind(bb1[,1],bb1[,2],ay,ax)
aaa[1:20,]
```

```
plot(axis,ay,xlab="number of cases",ylab="number of successes",main="Lift: Cum
successes sorted by pred val/success prob")
points(axis,ax,type="l")
```

Example 4: German Credit Data

```
#### ******* German Credit Data ******* ####
#### ******* data on 1000 loans ******* ####
## read data and create relevant variables
credit <- read.csv("C:/DataMining/Data/germancredit.csv")
credit
credit$Default <- factor(credit$Default)

## re-level the credit history and a few other variables
credit$history = factor(credit$history,
levels=c("A30","A31","A32","A33","A34"))
levels(credit$history) = c("good","good","poor","poor","terrible")
credit$foreign <- factor(credit$foreign, levels=c("A201","A202"),
labels=c("foreign","german"))
credit$rent <- factor(credit$housing=="A151")
credit$purpose <- factor(credit$purpose,
levels=c("A40","A41","A42","A43","A44","A45","A46","A47","A48","A49","A410"))
levels(credit$purpose) <-
c("newcar","usedcar",rep("goods/repair",4),"edu",NA,"edu","biz","biz")

## for demonstration, cut the dataset to these variables
credit <- credit[,c("Default","duration","amount","installment","age",
"history", "purpose","foreign","rent")]
credit[1:3,]
summary(credit) # check out the data

## create a design matrix
## factor variables are turned into indicator variables
## the first column of ones is omitted
Xcred <- model.matrix(Default~.,data=credit)[,-1]
Xcred[1:3,]

## creating training and prediction datasets
## select 900 rows for estimation and 100 for testing
set.seed(1)
train <- sample(1:1000,900)
xtrain <- Xcred[train,]
xnew <- Xcred[-train,]
ytrain <- credit$Default[train]
ynew <- credit$Default[-train]
credglm=glm(Default~.,family=binomial,data=data.frame(Default=ytrain,xtrain))
summary(credglm)

## prediction: predicted default probabilities for cases in test set
ptest <- predict(credglm,newdata=data.frame(xnew),type="response")
data.frame(ynew,ptest)

## What are our misclassification rates on that training set?
## We use probability cutoff 1/6
## coding as 1 (predicting default) if probability 1/6 or larger
gg1=floor(ptest+(5/6))
ttt=table(ynew,gg1)
ttt
error=(ttt[1,2]+ttt[2,1])/100
error
```

CHAPTER 8: BINARY CLASSIFICATION, PROBABILITIES AND EVALUATING CLASSIFICATION PERFORMANCE

Example: German Credit Data

```
#### ******* German Credit Data ******* ####
#### ******* data on 1000 loans ******* ####

## read data and create some `interesting' variables
credit <- read.csv("C:/DataMining/Data/germancredit.csv")
credit
credit$Default <- factor(credit$Default)

## re-level the credit history and a few other variables
credit$history = factor(credit$history,
levels=c("A30","A31","A32","A33","A34"))
levels(credit$history) = c("good","good","poor","poor","terrible")
credit$foreign <- factor(credit$foreign, levels=c("A201","A202"),
labels=c("foreign","german"))
credit$rent <- factor(credit$housing=="A151")
credit$purpose <- factor(credit$purpose,
levels=c("A40","A41","A42","A43","A44","A45","A46","A47","A48","A49","A410"))
levels(credit$purpose) <-
c("newcar","usedcar",rep("goods/repair",4),"edu",NA,"edu","biz","biz")

## for demonstration, cut the dataset to these variables
credit <- credit[,c("Default","duration","amount","installment","age",
"history", "purpose","foreign","rent")]
credit
summary(credit) # check out the data

## create a design matrix
## factor variables are turned into indicator variables
## the first column of ones is omitted
Xcred <- model.matrix(Default~.,data=credit)[,-1]
Xcred[1:3,]

## creating training and prediction datasets
## select 900 rows for estimation and 100 for testing
set.seed(1)
train <- sample(1:1000,900)
xtrain <- Xcred[train,]
xnew <- Xcred[-train,]
ytrain <- credit$Default[train]
ynew <- credit$Default[-train]
credglm=glm(Default~.,family=binomial,data=data.frame(Default=ytrain,xtrain))
summary(credglm)

## Now to prediction: what are the underlying default probabilities
## for cases in the test set
ptest <- predict(credglm, newdata=data.frame(xnew),type="response")
data.frame(ynew,ptest)

## What are our misclassification rates on that training set?
```

```
## We use probability cutoff 1/6
## coding as 1 (predicting default) if probability 1/6 or larger
cut=1/6
gg1=floor(ptest+(1-cut))
ttt=table(ynew,gg1)
ttt
truepos <- ynew==1 & ptest>=cut
trueneg <- ynew==0 & ptest<cut
# Sensitivity (predict default when it does happen)
sum(truepos)/sum(ynew==1)
# Specificity (predict no default when it does not happen)
sum(trueneg)/sum(ynew==0)

## Next, we use probability cutoff 1/2
## coding as 1 if probability 1/2 or larger
cut=1/2
gg1=floor(ptest+(1-cut))
ttt=table(ynew,gg1)
ttt
truepos <- ynew==1 & ptest>=cut
trueneg <- ynew==0 & ptest<cut
# Sensitivity (predict default when it does happen)
sum(truepos)/sum(ynew==1)
# Specificity (predict no default when it does not happen)
sum(trueneg)/sum(ynew==0)

## R macro for plotting the ROC curve
## plot the ROC curve for classification of y with p
roc <- function(p,y){
  y <- factor(y)
  n <- length(p)
  p <- as.vector(p)
  Q <- p > matrix(rep(seq(0,1,length=500),n),ncol=500,byrow=TRUE)
  fp <- colSums((y==levels(y)[1])*Q)/sum(y==levels(y)[1])
  tp <- colSums((y==levels(y)[2])*Q)/sum(y==levels(y)[2])
  plot(fp, tp, xlab="1-Specificity", ylab="Sensitivity")
  abline(a=0,b=1,lty=2,col=8)
}

## ROC for hold-out period
roc(p=ptest,y=ynew)

## ROC for all cases (in-sample)
credglmall <- glm(credit$Default ~ Xcred,family=binomial)
roc(p=credglmall$fitted, y=credglmall$y)

## using the ROCR package to graph the ROC curves
library(ROCR)
## input is a data frame consisting of two columns
## predictions in first column and actual outcomes in the second

## ROC for hold-out period
predictions=ptest
labels=ynew
data=data.frame(predictions,labels)
data
```

```
## pred: function to create prediction objects
pred <- prediction(data$predictions,data$labels)
pred
## perf: creates the input to be plotted
## sensitivity and one minus specificity (the false positive rate)
perf <- performance(pred, "sens", "fpr")
perf
plot(perf)

## ROC for all cases (in-sample)
credglmall <- glm(credit$Default ~ Xcred,family=binomial)
predictions=credglmall$fitted
labels=credglmall$y
data=data.frame(predictions,labels)
pred <- prediction(data$predictions,data$labels)
perf <- performance(pred, "sens", "fpr")
plot(perf)
```

CHAPTER 9: CLASSIFICATION USING A NEAREST NEIGHBOR ANALYSIS

Example 1: Forensic Glass

```
#### ******* Forensic Glass ****** ####

library(textir) ## needed to standardize the data
library(MASS)   ## a library of example datasets

data(fgl)         ## loads the data into R; see help(fgl)
fgl
## data consists of 214 cases
## here are illustrative box plots of the features stratified by
## glass type
par(mfrow=c(3,3), mai=c(.3,.6,.1,.1))
plot(RI ~ type, data=fgl, col=c(grey(.2),2:6))
plot(Al ~ type, data=fgl, col=c(grey(.2),2:6))
plot(Na ~ type, data=fgl, col=c(grey(.2),2:6))
plot(Mg ~ type, data=fgl, col=c(grey(.2),2:6))
plot(Ba ~ type, data=fgl, col=c(grey(.2),2:6))
plot(Si ~ type, data=fgl, col=c(grey(.2),2:6))
plot(K ~ type, data=fgl, col=c(grey(.2),2:6))
plot(Ca ~ type, data=fgl, col=c(grey(.2),2:6))
plot(Fe ~ type, data=fgl, col=c(grey(.2),2:6))

## for illustration, consider the RIxAl plane
## use nt=200 training cases to find the nearest neighbors for
## the remaining 14 cases. These 14 cases become the evaluation
## (test, hold-out) cases

n=length(fgl$type)
nt=200
set.seed(1) ## to make the calculations reproducible in repeated runs
train <- sample(1:n,nt)

## Standardization of the data is preferable, especially if
## units of the features are quite different
## could do this from scratch by calculating the mean and
## standard deviation of each feature, and use those to
## standardize.
## Even simpler, use the normalize function in the R-package textir;
## it converts data frame columns to mean-zero sd-one

x <- normalize(fgl[,c(4,1)])
x[1:3,]

library(class)
nearest1 <- knn(train=x[train,],test=x[-train,],cl=fgl$type[train],k=1)
nearest5 <- knn(train=x[train,],test=x[-train,],cl=fgl$type[train],k=5)
data.frame(fgl$type[-train],nearest1,nearest5)

## plot them to see how it worked
par(mfrow=c(1,2))
## plot for k=1 (single) nearest neighbor
plot(x[train,],col=fgl$type[train],cex=.8,main="1-nearest neighbor")
points(x[-train,],bg=nearest1,pch=21,col=grey(.9),cex=1.25)
```

```
## plot for k=5 nearest neighbors
plot(x[train,],col=fgl$type[train],cex=.8,main="5-nearest neighbors")
points(x[-train,],bg=nearest5,pch=21,col=grey(.9),cex=1.25)
legend("topright",legend=levels(fgl$type),fill=1:6,bty="n",cex=.75)

## calculate the proportion of correct classifications on this one
## training set

pcorrn1=100*sum(fgl$type[-train]==nearest1)/(n-nt)
pcorrn5=100*sum(fgl$type[-train]==nearest5)/(n-nt)
pcorrn1
pcorrn5

## cross-validation (leave one out)
pcorr=dim(10)
for (k in 1:10) {
pred=knn.cv(x,fgl$type,k)
pcorr[k]=100*sum(fgl$type==pred)/n
}
pcorr
## Note: Different runs may give you slightly different results as ties
## are broken at random

## using all nine dimensions (RI plus 8 chemical concentrations)

x <- normalize(fgl[,c(1:9)])
nearest1 <- knn(train=x[train,],test=x[-train,],cl=fgl$type[train],k=1)
nearest5 <- knn(train=x[train,],test=x[-train,],cl=fgl$type[train],k=5)
data.frame(fgl$type[-train],nearest1,nearest5)

## calculate the proportion of correct classifications

pcorrn1=100*sum(fgl$type[-train]==nearest1)/(n-nt)
pcorrn5=100*sum(fgl$type[-train]==nearest5)/(n-nt)
pcorrn1
pcorrn5

## cross-validation (leave one out)

pcorr=dim(10)
for (k in 1:10) {
pred=knn.cv(x,fgl$type,k)
pcorr[k]=100*sum(fgl$type==pred)/n
}
pcorr
```

Example 2: German Credit Data

```
#### ******* German Credit Data ******* ####
#### ******* data on 1000 loans ******* ####

library(textir)   ## needed to standardize the data
library(class)    ## needed for knn

## read data and create some `interesting' variables
credit <- read.csv("C:/DataMining/Data/germancredit.csv")
credit

credit$Default <- factor(credit$Default)

## re-level the credit history and a few other variables
credit$history = factor(credit$history,
levels=c("A30","A31","A32","A33","A34"))
levels(credit$history) = c("good","good","poor","poor","terrible")
credit$foreign <- factor(credit$foreign, levels=c("A201","A202"),
labels=c("foreign","german"))
credit$rent <- factor(credit$housing=="A151")
credit$purpose <- factor(credit$purpose,
levels=c("A40","A41","A42","A43","A44","A45","A46","A47","A48","A49","A410"))
levels(credit$purpose) <-
c("newcar","usedcar",rep("goods/repair",4),"edu",NA,"edu","biz","biz")

## for demonstration, cut the dataset to these variables
credit <- credit[,c("Default","duration","amount","installment","age",
"history", "purpose","foreign","rent")]
credit[1:3,]
summary(credit) # check out the data

## for illustration we consider just 3 loan characteristics:
## amount,duration,installment
## Standardization of the data is preferable, especially if
## units of the features are quite different
## We use the normalize function in the R-package textir;
## it converts data frame columns to mean-zero sd-one

x <- normalize(credit[,c(2,3,4)])
x[1:3,]

## training and prediction datasets
## training set of 900 borrowers; want to classify 100 new ones
set.seed(1)
train <- sample(1:1000,900) ## this is training set of 900 borrowers
xtrain <- x[train,]
xnew <- x[-train,]
ytrain <- credit$Default[train]
ynew <- credit$Default[-train]

## k-nearest neighbor method
library(class)
nearest1 <- knn(train=xtrain, test=xnew, cl=ytrain, k=1)
nearest3 <- knn(train=xtrain, test=xnew, cl=ytrain, k=3)
data.frame(ynew,nearest1,nearest3)[1:10,]
```

```
## calculate the proportion of correct classifications
pcorrn1=100*sum(ynew==nearest1)/100
pcorrn3=100*sum(ynew==nearest3)/100
pcorrn1
pcorrn3

## plot for 3nn
plot(xtrain[,c("amount","duration")],col=c(4,3,6,2)
[credit[train,"installment"]],pch=c(1,2)[as.numeric(ytrain)],main="Predicted
default, by 3 nearest neighbors",cex.main=.95)
points(xnew[,c("amount","duration")],bg=c(4,3,6,2)
[credit[train,"installment"]],pch=c(21,24)
[as.numeric(nearest3)],cex=1.2,col=grey(.7))
legend("bottomright",pch=c(1,16,2,17),bg=c(1,1,1,1),legend=c("data 0","pred
0","data 1","pred 1"),title="default",bty="n",cex=.8)
legend("topleft",fill=c(4,3,6,2),legend=c(1,2,3,4),title="installment
%",horiz=TRUE,bty="n",col=grey(.7),cex=.8)

## above was for just one training set
## cross-validation (leave one out)
pcorr=dim(10)
for (k in 1:10) {
      pred=knn.cv(x,cl=credit$Default,k)
      pcorr[k]=100*sum(credit$Default==pred)/1000
                }
pcorr
```

CHAPTER 10: THE NAÏVE BAYESIAN ANALYSIS: A MODEL FOR PREDICTING A CATEGORICAL RESPONSE FROM MOSTLY CATEGORICAL PREDICTOR VARIABLES

Example: Delayed Airplanes

```
set.seed(1)
library(car)      #used to recode a variable

## reading the data
delay <- read.csv("C:/DataMining/Data/FlightDelays.csv")
delay
del=data.frame(delay)
del$schedf=factor(floor(del$schedtime/100))
del$delay=recode(del$delay,"'delayed'=1;else=0")
response=as.numeric(levels(del$delay)[del$delay])
hist(response)
mm=mean(response)
mm

## determining test and evaluation data sets
n=length(del$dayweek)
n
n1=floor(n*(0.6))
n1
n2=n-n1
n2
train=sample(1:n,n1)

## determining marginal probabilities

tttt=cbind(del$schedf[train],del$carrier[train],del$dest[train],del$origin[tra
in],del$weather[train],del$dayweek[train],response[train])
tttrain0=tttt[tttt[,7]<0.5,]
tttrain1=tttt[tttt[,7]>0.5,]

## prior probabilities

tdel=table(response[train])
tdel=tdel/sum(tdel)
tdel

## scheduled time

ts0=table(tttrain0[,1])
ts0=ts0/sum(ts0)
ts0
ts1=table(tttrain1[,1])
ts1=ts1/sum(ts1)
ts1

## scheduled carrier
```

```
tc0=table(tttrain0[,2])
tc0=tc0/sum(tc0)
tc0
tc1=table(tttrain1[,2])
tc1=tc1/sum(tc1)
tc1

## scheduled destination

td0=table(tttrain0[,3])
td0=td0/sum(td0)
td0
td1=table(tttrain1[,3])
td1=td1/sum(td1)
td1

## scheduled origin

to0=table(tttrain0[,4])
to0=to0/sum(to0)
to0
to1=table(tttrain1[,4])
to1=to1/sum(to1)
to1

## weather

tw0=table(tttrain0[,5])
tw0=tw0/sum(tw0)
tw0
tw1=table(tttrain1[,5])
tw1=tw1/sum(tw1)
tw1
## bandaid as no observation in a cell
tw0=tw1
tw0[1]=1
tw0[2]=0

## scheduled day of week

tdw0=table(tttrain0[,6])
tdw0=tdw0/sum(tdw0)
tdw0
tdw1=table(tttrain1[,6])
tdw1=tdw1/sum(tdw1)
tdw1

## creating test data set
tt=cbind(del$schedf[-train],del$carrier[-train],del$dest[-train],del$origin[-
train],del$weather[-train],del$dayweek[-train],response[-train])

## creating predictions, stored in gg

p0=ts0[tt[,1]]*tc0[tt[,2]]*td0[tt[,3]]*to0[tt[,4]]*tw0[tt[,5]+1]*tdw0[tt[,6]]
p1=ts1[tt[,1]]*tc1[tt[,2]]*td1[tt[,3]]*to1[tt[,4]]*tw1[tt[,5]+1]*tdw1[tt[,6]]
gg=(p1*tdel[2])/(p1*tdel[2]+p0*tdel[1])
hist(gg)
```

```
plot(response[-train]~gg)

## coding as 1 if probability 0.5 or larger
gg1=floor(gg+0.5)
ttt=table(response[-train],gg1)
ttt
error=(ttt[1,2]+ttt[2,1])/n2
error

## coding as 1 if probability 0.3 or larger
gg2=floor(gg+0.7)
ttt=table(response[-train],gg2)
ttt
error=(ttt[1,2]+ttt[2,1])/n2
error

## Here we calculate the lift (see Chapter 4)
## The output is not shown in the text
bb=cbind(gg,response[-train])
bb
bb1=bb[order(gg,decreasing=TRUE),]
bb1
## order cases in test set naccording to their success prob
## actual outcome shown next to it

## overall success (delay) prob in evaluation set
xbar=mean(response[-train])
xbar

## calculating the lift
## cumulative 1's sorted by predicted values
## cumulative 1's using the average success prob from training set
axis=dim(n2)
ax=dim(n2)
ay=dim(n2)
axis[1]=1
ax[1]=xbar
ay[1]=bb1[1,2]
for (i in 2:n2) {
axis[i]=i
ax[i]=xbar*i
ay[i]=ay[i-1]+bb1[i,2]
}

aaa=cbind(bb1[,1],bb1[,2],ay,ax)
aaa[1:100,]

plot(axis,ay,xlab="number of cases",ylab="number of successes",main="Lift: Cum
successes sorted by pred val/success prob")
points(axis,ax,type="l")
```

CHAPTER 11: MULTINOMIAL LOGISTIC REGRESSION

Example 1: Forensic Glass

```
## Program 1: Estimation on all 214 shards
## Forensic Glass
library(VGAM)     ## VGAM to estimate multinomial logistic regression
library(textir)   ## to standardize the features
library(MASS)     ## a library of example datasets
data(fgl)         ## loads the data into R; see help(fgl)
fgl
## standardization, using the normalize function in the library textir
covars <- normalize(fgl[,1:9],s=sdev(fgl[,1:9]))
sd(covars) ## convince yourself that features are standardized
dd=data.frame(cbind(type=fgl$type,covars))
gg <- vglm(type~ Na+Mg+Al,multinomial,data=dd)
summary(gg)
predict(gg) ## obtain log-odds relative to last group
round(fitted(gg),2)  ## probabilities
cbind(round(fitted(gg),2),fgl$type)
## boxplots of estimated probabilities against true group
dWinF=fgl$type=="WinF"
dWinNF=fgl$type=="WinNF"
dVeh=fgl$type=="Veh"
dCon=fgl$type=="Con"
dTable=fgl$type=="Tabl"
dHead=fgl$type=="Head"
yy1=c(fitted(gg)[dWinF,1],fitted(gg)[dWinNF,2],fitted(gg)[dVeh,3], fitted(gg)
[dCon,4],fitted(gg)[dTable,5],fitted(gg)[dHead,6])
xx1=c(fgl$type[dWinF],fgl$type[dWinNF],fgl$type[dVeh],fgl$type[dCon],fgl$type[
dTable],fgl$type[dHead])
boxplot(yy1~xx1,ylim=c(0,1),xlab="1=WinF,2=WinNF,3=Veh,4=Con,5=Table,6=Head")
```

```
## Program 2: Estimation on all 194 shards and predicting 20 new cases
## performance in predicting a single set of 20 new cases
library(VGAM)
library(textir)
library(MASS) ## a library of example datasets
data(fgl)     ## loads the data into R; see help(fgl)
fgl
covars <- normalize(fgl[,1:9],s=sdev(fgl[,1:9]))
dd=data.frame(cbind(type=fgl$type,covars))
n=length(fgl$type)
nt=n-20
set.seed(1)
train <- sample(1:n,nt)
## predict
gg <- vglm(type ~ Na+Mg+Al,multinomial,data=dd[train,])
p1=predict(gg,newdata=dd[-train,])
p1=exp(p1)
## we calculate the probabilities from the predicted logits
sum=(1+p1[,1]+p1[,2]+p1[,3]+p1[,4]+p1[,5])
probWinF=round(p1[,1]/sum,2)        ## WinF
```

```
probWinNF=round(p1[,2]/sum,2)       ## WinNF
probVeh=round(p1[,3]/sum,2)         ## Veh
probCon=round(p1[,4]/sum,2)         ## Con
probTable=round(p1[,5]/sum,2)       ## Table
probHead=round(1/sum,2)             ## Head
ppp=data.frame(probWinF,probWinNF,probVeh,probCon,probTable,probHead,fgl$type[
-train])
ppp


## Program 3: Estimation on all 194 shards and predicting 20 new cases; 100 reps
## performance from 100 replications predicting 20 new cases
library(VGAM)
library(textir)
library(MASS) ## a library of example datasets
data(fgl)     ## loads the data into R; see help(fgl)
fgl
covars <- normalize(fgl[,1:9],s=sdev(fgl[,1:9]))
dd=data.frame(cbind(type=fgl$type,covars))
## out-of-sample prediction
set.seed(1)
out=dim(20)
proportion=dim(100)
prob=matrix(nrow=20,ncol=6)
n=length(fgl$type)
nt=n-20
for (kkk in 1:100) {
train <- sample(1:n,nt)
## predict
gg <- vglm(type ~ Na+Mg+Al,multinomial,data=dd[train,])
p1=predict(gg,newdata=dd[-train,])
p1=exp(p1)
## we calculate the probabilities from the predicted logits
sum=(1+p1[,1]+p1[,2]+p1[,3]+p1[,4]+p1[,5])
prob[,1]=p1[,1]/sum     ## WinF
prob[,2]=p1[,2]/sum     ## WinNF
prob[,3]=p1[,3]/sum     ## Veh
prob[,4]=p1[,4]/sum     ## Con
prob[,5]=p1[,5]/sum     ## Table
prob[,6]=1/sum          ## Head
for (k in 1:20) {
pp=prob[k,]
out[k]=max(pp)==pp[fgl$type[-train]][k]
}
proportion[kkk]=sum(out)/20
}
## proportion of correct classification
proportion
mean(proportion)
boxplot(ylim=c(0,1),ylab="percent correct classification",proportion)
```

Example 2: Forensic Glass Revisited

```
## Program 1: Cross-validation to determine the penalty in mnlm
library(textir)
set.seed(1)
library(MASS) ## a library of example datasets
data(fgl)     ## loads the data into R; see help(fgl)
covars <- normalize(fgl[,1:9],s=sdev(fgl[,1:9]))
n=length(fgl$type)
prop=dim(30)
pen=dim(30)
out=dim(n)

for (j in 1:30) {
pen[j]=0.1*j
for (k in 1:n) {
train1=c(1:n)
train=train1[train1!=k]
glasslm <- mnlm(counts=fgl$type[train],penalty=pen[j],covars=covars[train,])
prob=predict(glasslm,covars[-train,])
prob=round(prob,3)
out[k]=max(prob)==prob[fgl$type[-train]]
}
prop[j]=sum(out)/n
}
## proportion of correct classifications using Laplace scale penalty
output=cbind(pen,prop)
round(output,3)


## Program 2: Detailed mnlm output for penalty = 1
library(textir)
library(MASS) ## a library of example datasets
data(fgl)     ## loads the data into R; see help(fgl)
fgl$type
covars <- normalize(fgl[,1:9],s=sdev(fgl[,1:9]))
glasslm <- mnlm(counts=fgl$type,penalty=1.0,covars=covars)
glasslm$intercept
glasslm$loadings
round(as.matrix(glasslm$loadings)[,],2)
fitted(glasslm)
as.matrix(fitted(glasslm)[1,])
round(predict(glasslm,covars),2)
plot(glasslm)


## Program 3: Estimation on all 194 shards and predicting 20 new cases
## multinomial logistic regression with linear logits
library(textir)
library(MASS) ## a library of example datasets
data(fgl)     ## loads the data into R; see help(fgl)
covars <- normalize(fgl[,1:9],s=sdev(fgl[,1:9]))
sd(covars)
set.seed(1)
pp=dim(6)
out=dim(20)
```

```
proportion=dim(100)
n=length(fgl$type)
nt=n-20
for (kkk in 1:100) {
train <- sample(1:n,nt)
glasslm=mnlm(counts=fgl$type[train],penalty=1,covars=covars[train,])
prob=predict(glasslm,covars[-train,])
for (k in 1:20) {
pp=prob[k,]
out[k]=max(pp)==pp[fgl$type[-train]][k]
}
proportion[kkk]=sum(out)/20
}
proportion
mean(proportion)
boxplot(proportion)


## Program 4: Estimation on all 194 shards and predicting 20 new cases
## multinomial logistic regression with linear and cross-products
library(textir)
library(MASS) ## a library of example datasets
data(fgl)     ## loads the data into R; see help(fgl)
X <- model.matrix(~.+.^2, data=fgl[,1:9])[,-1]
X[1:3,]     ## to see the contents
## -1 removes the intercept
dim(X)      ## X has 45 columns
covars <- normalize(X,s=sdev(X))
sd(covars)
set.seed(1)
pp=dim(6)
out=dim(20)
proportion=dim(100)
n=length(fgl$type)
nt=n-20
for (kkk in 1:100) {
train <- sample(1:n,nt)
glasslm=mnlm(counts=fgl$type[train],penalty=1,covars=covars[train,])
prob=predict(glasslm,covars[-train,])
for (k in 1:20) {
pp=prob[k,]
out[k]=max(pp)==pp[fgl$type[-train]][k]
}
proportion[kkk]=sum(out)/20
}
proportion
mean(proportion)
boxplot(proportion)
```

Appendix: Specification of a Simple Triplet Matrix

```
## working with simple triplet matrices

i=c(1,2,3,4,5,6)
j=c(1,1,1,2,2,2)
v=c(5,5,5,6,6,6)
b=simple_triplet_matrix(i,j,v)
b
as.matrix(b)[,]

v=c(11,12,22,33,44,55)
b=simple_triplet_matrix(i,j,v)
as.matrix(b)[,]

i=c(1,2,3,4,5,6)
j=c(1,2,3,4,5,6)
v=c(5,5,5,6,6,6)
b=simple_triplet_matrix(i,j,v)
b
as.matrix(b)[,]

i=c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16)
j=c(1,1,2,3,3,4,4,4,4,5,5,6,6,6,6,6)
v=c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)
b=simple_triplet_matrix(i,j,v)
b
as.matrix(b)[,]
```

CHAPTER 12: MORE ON CLASSIFICATION AND A DISCUSSION OF DISCRIMINANT ANALYSIS

Example 1: German Credit Data

```
#### ******* German Credit Data ******* ####
#### ******* data on 1000 loans ******* ####

library(MASS)     ## includes lda and qda for discriminant analysis
set.seed(1)

## read data and create some 'interesting' variables
credit <- read.csv("C:/DataMining/Data/germancredit.csv")
credit

credit$Default <- factor(credit$Default)

## re-level the credit history and a few other variables
credit$history = factor(credit$history,
levels=c("A30","A31","A32","A33","A34"))
levels(credit$history) = c("good","good","poor","poor","terrible")
credit$foreign <- factor(credit$foreign, levels=c("A201","A202"),
labels=c("foreign","german"))
credit$rent <- factor(credit$housing=="A151")
credit$purpose <- factor(credit$purpose,
levels=c("A40","A41","A42","A43","A44","A45","A46","A47","A48","A49","A410"))
levels(credit$purpose) <-
c("newcar","usedcar",rep("goods/repair",4),"edu",NA,"edu","biz","biz")

## take the continuous variables duration, amount, installment, age
## with indicators the assumptions of a normal distribution would be
## tenuous at best; hence these variables are not considered here

cred1=credit[,c("Default","duration","amount","installment","age")]
cred1
summary(cred1)

hist(cred1$duration)
hist(cred1$amount)
hist(cred1$installment)
hist(cred1$age)
cred1$Default
cred1=data.frame(cred1)

## linear discriminant analysis
## class proportions of the training set used as prior probabilities
zlin=lda(Default~.,cred1)
predict(zlin,newdata=data.frame(duration=6,amount=1100,installment=4,age=67))
predict(zlin,newdata=data.frame(duration=6,amount=1100,installment=4,age=67))
$class
zqua=qda(Default~.,cred1)
predict(zqua,newdata=data.frame(duration=6,amount=1100,installment=4,age=67))
predict(zqua,newdata=data.frame(duration=6,amount=1100,installment=4,age=67))
$class
```

```
n=1000
neval=1
errlin=dim(n)
errqua=dim(n)

## leave one out evaluation
for (k in 1:n) {
train1=c(1:n)
train=train1[train1!=k]
## linear discriminant analysis
zlin=lda(Default~.,cred1[train,])
predict(zlin,cred1[-train,])$class
tablin=table(cred1$Default[-train],predict(zlin,cred1[-train,])$class)
errlin[k]=(neval-sum(diag(tablin)))/neval
## quadratic discriminant analysis
zqua=qda(Default~.,cred1[train,])
predict(zqua,cred1[-train,])$class
tablin=table(cred1$Default[-train],predict(zqua,cred1[-train,])$class)
errqua[k]=(neval-sum(diag(tablin)))/neval
}
merrlin=mean(errlin)
merrlin
merrqua=mean(errqua)
merrqua
```

Example 2: Fisher Iris Data

```
library(MASS)     ## includes lda and qda for discriminant analysis
set.seed(1)
iris3
Iris=data.frame(rbind(iris3[,,1],iris3[,,2],iris3[,,3]),Sp=rep(c("s","c","v"),
rep(50,3)))
Iris
## linear discriminant analysis
## equal prior probabilities as same number from each species
zlin=lda(Sp~.,Iris,prior=c(1,1,1)/3)
predict(zlin,newdata=data.frame(Sepal.L.=5.1,Sepal.W.=3.5,Petal.L.=1.4,
Petal.W.=0.2))
predict(zlin,newdata=data.frame(Sepal.L.=5.1,Sepal.W.=3.5,Petal.L.=1.4,
Petal.W.=0.2))$class
## quadratic discriminant analysis
zqua=qda(Sp~.,Iris,prior=c(1,1,1)/3)
predict(zqua,newdata=data.frame(Sepal.L.=5.1,Sepal.W.=3.5,Petal.L.=1.4,
Petal.W.=0.2))
predict(zqua,newdata=data.frame(Sepal.L.=5.1,Sepal.W.=3.5,Petal.L.=1.4,
Petal.W.=0.2))$class

n=150
nt=100
neval=n-nt
rep=1000
errlin=dim(rep)
errqua=dim(rep)
for (k in 1:rep) {
train=sample(1:n,nt)
## linear discriminant analysis
m1=lda(Sp~.,Iris[train,],prior=c(1,1,1)/3)
predict(m1,Iris[-train,])$class
tablin=table(Iris$Sp[-train],predict(m1,Iris[-train,])$class)
errlin[k]=(neval-sum(diag(tablin)))/neval
## quadratic discriminant analysis
m2=qda(Sp~.,Iris[train,],prior=c(1,1,1)/3)
predict(m2,Iris[-train,])$class
tablin=table(Iris$Sp[-train],predict(m2,Iris[-train,])$class)
errqua[k]=(neval-sum(diag(tablin)))/neval
}
merrlin=mean(errlin)
merrlin
merrqua=mean(errqua)
merrqua
```

Example 3: Forensic Glass Data

```
library(MASS)     ## includes lda and qda for discriminant analysis
set.seed(1)
data(fgl)
glass=data.frame(fgl)
glass

## linear discriminant analysis
m1=lda(type~.,glass)
m1
predict(m1,newdata=data.frame(RI=3.0,Na=13,Mg=4,Al=1,Si=70,K=0.06,Ca=9,Ba=0,Fe
=0))
predict(m1,newdata=data.frame(RI=3.0,Na=13,Mg=4,Al=1,Si=70,K=0.06,Ca=9,Ba=0,Fe
=0))$class

## quadratic discriminant analysis: Not enough data as each 9x9
## covariance matrix includes (9)(8)/2 = 45 unknown coefficients

n=length(fgl$type)
nt=200
neval=n-nt

rep=100
errlin=dim(rep)

for (k in 1:rep) {
train=sample(1:n,nt)
glass[train,]
## linear discriminant analysis
m1=lda(type~.,glass[train,])
predict(m1,glass[-train,])$class
tablin=table(glass$type[-train],predict(m1,glass[-train,])$class)
errlin[k]=(neval-sum(diag(tablin)))/neval
}
merrlin=mean(errlin)
merrlin

n=214
neval=1
errlin=dim(n)
errqua=dim(n)

for (k in 1:n) {
train1=c(1:n)
train=train1[train1!=k]
## linear discriminant analysis
m1=lda(type~.,glass[train,])
predict(m1,glass[-train,])$class
tablin=table(glass$type[-train],predict(m1,glass[-train,])$class)
errlin[k]=(neval-sum(diag(tablin)))/neval
}
merrlin=mean(errlin)
merrlin
```

Example 4: MBA Admission Data

```
library(MASS)
set.seed(1)
## reading the data
admit <- read.csv("C:/DataMining/Data/admission.csv")
adm=data.frame(admit)
adm
plot(adm$GPA,adm$GMAT,col=adm$De)

## linear discriminant analysis
m1=lda(De~.,adm)
m1
predict(m1,newdata=data.frame(GPA=3.21,GMAT=497))

## quadratic discriminant analysis
m2=qda(De~.,adm)
m2
predict(m2,newdata=data.frame(GPA=3.21,GMAT=497))

n=85
nt=60
neval=n-nt

rep=100
errlin=dim(rep)

for (k in 1:rep) {
train=sample(1:n,nt)
## linear discriminant analysis
m1=lda(De~.,adm[train,])
predict(m1,adm[-train,])$class

tablin=table(adm$De[-train],predict(m1,adm[-train,])$class)
errlin[k]=(neval-sum(diag(tablin)))/neval
}

merrlin=mean(errlin)
merrlin
```

CHAPTER 13: DECISION TREES

Example 1: Prostate Cancer

```
prostate <- read.csv("C:/DataMining/Data/prostate.csv")
prostate

library(tree)
## Construct the tree
pstree <- tree(lcavol ~., data=prostate, mindev=0.1, mincut=1)
pstree <- tree(lcavol ~., data=prostate, mincut=1)
pstree
plot(pstree, col=8)
text(pstree, digits=2)

pstcut <- prune.tree(pstree,k=1.7)
plot(pstcut)
pstcut

pstcut <- prune.tree(pstree,k=2.05)
plot(pstcut)
pstcut

pstcut <- prune.tree(pstree,k=3)
plot(pstcut)
pstcut

pstcut <- prune.tree(pstree)
pstcut
plot(pstcut)

pstcut <- prune.tree(pstree,best=3)
pstcut
plot(pstcut)

## Use cross-validation to prune the tree
set.seed(2)
cvpst <- cv.tree(pstree, K=10)
cvpst$size
cvpst$dev
plot(cvpst, pch=21, bg=8, type="p", cex=1.5, ylim=c(65,100))

pstcut <- prune.tree(pstree, best=3)
pstcut
plot(pstcut, col=8)
text(pstcut)

## Plot what we end up with
plot(prostate[,c("lcp","lpsa")],cex=0.2*exp(prostate$lcavol))
abline(v=.261624, col=4, lwd=2)
lines(x=c(-2,.261624), y=c(2.30257,2.30257), col=4, lwd=2)
```

Example 2: Motorcycle Acceleration

```
library(MASS)
library(tree)
data(mcycle)
mcycle
plot(accel~times,data=mcycle)
mct <- tree(accel ~ times, data=mcycle)
mct
plot(mct, col=8)
text(mct, cex=.75) ## we use different font size to avoid print overlap

## scatter plot of data with overlay of fitted function
x=c(1:6000)
x=x/100
y1=seq(-4.357,-4.357,length.out=1510)
y2=seq(-39.120,-39.120,length.out=140)
y3=seq(-86.31,-86.31,length.out=300)
y4=seq(-114.7,-114.7,length.out=490)
y5=seq(-42.49,-42.49,length.out=300)
y6=seq(10.25,10.25,length.out=240)
y7=seq(40.72,40.72,length.out=520)
y8=seq(3.291,3.291,length.out=2500)
y=c(y1,y2,y3,y4,y5,y6,y7,y8)
plot(accel~times,data=mcycle)
lines(y~x)
```

Example 3: Fisher Iris Data Revisited

```
library(MASS)
library(tree)
## read in the iris data
iris
iristree <- tree(Species~.,data=iris)
iristree
plot(iristree)
plot(iristree,col=8)
text(iristree,digits=2)
summary(iristree)
irissnip=snip.tree(iristree,nodes=c(7,12))
irissnip
plot(irissnip)
text(irissnip)
```

CHAPTER 14: FURTHER DISCUSSION ON REGRESSION AND CLASSIFICATION TREES, COMPUTER SOFTWARE, AND OTHER USEFUL CLASSIFICATION METHODS

no programs

CHAPTER 15: CLUSTERING

Example 1: European Protein Consumption

```
### *** European Protein Consumption, in grams/person-day *** ###

## read in the data
food <- read.csv("C:/DataMining/Data/protein.csv")
food[1:3,]
## first, clustering on just Red and White meat (p=2) and k=3 clusters
set.seed(1) ## to fix the random starting clusters
grpMeat <- kmeans(food[,c("WhiteMeat","RedMeat")], centers=3, nstart=10)
grpMeat
## list of cluster assignments
o=order(grpMeat$cluster)
data.frame(food$Country[o],grpMeat$cluster[o])
## plotting cluster assignments on Red and White meat scatter plot
plot(food$Red, food$White, type="n", xlim=c(3,19), xlab="Red Meat",
ylab="White Meat")
text(x=food$Red, y=food$White, labels=food$Country, col=grpMeat$cluster+1)

## same analysis, but now with clustering on all protein groups
## change the number of clusters to 7
set.seed(1)
grpProtein <- kmeans(food[,-1], centers=7, nstart=10)
o=order(grpProtein$cluster)
data.frame(food$Country[o],grpProtein$cluster[o])
plot(food$Red, food$White, type="n", xlim=c(3,19), xlab="Red Meat",
ylab="White Meat")
text(x=food$Red, y=food$White, labels=food$Country, col=rainbow(7)
[grpProtein$cluster])
```

Example 2: Monthly US Unemployment Rates

```
## read the data; series are stored column-wise with labels in first row
raw <- read.csv("C:/DataMining/Data/unempstates.csv")
raw[1:3,]

## time sequence plots of three series
plot(raw[,5],type="l",ylim=c(0,12),xlab="month",ylab="unemployment rate") ##
CA
points(raw[,32],type="l", cex = .5, col = "dark red")    ## New York
points(raw[,15],type="l", cex = .5, col = "dark green")  ## Iowa

## transpose the data
## then we have 50 rows (states) and 416 columns (time periods)
rawt=matrix(nrow=50,ncol=416)
rawt=t(raw)
rawt[1:3,]

## k-means clustering in 416 dimensions
set.seed(1)
grpunemp2 <- kmeans(rawt, centers=2, nstart=10)
sort(grpunemp2$cluster)
grpunemp3 <- kmeans(rawt, centers=3, nstart=10)
sort(grpunemp3$cluster)
grpunemp4 <- kmeans(rawt, centers=4, nstart=10)
sort(grpunemp4$cluster)
grpunemp5 <- kmeans(rawt, centers=5, nstart=10)
sort(grpunemp5$cluster)

## another analysis
## data set unemp.csv with means and standard deviations for each state
## k-means clustering on 2 dimensions (mean, stddev)
unemp <- read.csv("C:/DataMining/Data/unemp.csv")
unemp[1:3,]

set.seed(1)
grpunemp <- kmeans(unemp[,c("mean","stddev")], centers=3, nstart=10)
## list of cluster assignments
o=order(grpunemp$cluster)
data.frame(unemp$state[o],grpunemp$cluster[o])
plot(unemp$mean,unemp$stddev,type="n",xlab="mean", ylab="stddev")
text(x=unemp$mean,y=unemp$stddev,labels=unemp$state, col=grpunemp$cluster+1)
```

Example 3: European Protein Consumption Revisited (Mixture Model)

```
library(mixtools)

## for a brief description of mvnormalmixEM
## mvnormalmixEM(x, lambda = NULL, mu = NULL, sigma = NULL, k = 2,
##               arbmean = TRUE, arbvar = TRUE, epsilon = 1e-08,
##               maxit = 10000, verb = FALSE)
## arbvar=FALSE               same cov matrices
## arbvar=TRUE (default)      different cov matrices
## arbmean=TRUE (default)     different means
## k  number of groups

food <- read.csv("C:/DataMining/Data/protein.csv")
## Consider just Red and White meat clusters
food[1:3,]
X=cbind(food[,2],food[,3])
X[1:3,]

set.seed(1)
## here we use an iterative procedure and the results in repeated runs may
## not be exactly the same
## set.seed(1) is used to obtain reproducible results

## mixtures of two normal distributions on the first 2 features
## we consider different variances
out2<-mvnormalmixEM(X,arbvar=TRUE,k=2,epsilon=1e-02)
out2
prob1=round(out2$posterior[,1],digits=3)
prob2=round(out2$posterior[,2],digits=3)
prob=round(out2$posterior[,1])
o=order(prob)
data.frame(food$Country[o],prob1[o],prob2[o],prob[o])
plot(food$Red, food$White, type="n",xlab="Red Meat", ylab="White Meat")
text(x=food$Red,y=food$White,labels=food$Country,col=prob+1)

## mixtures of two normal distributions on all 9 features
## we consider equal variances
X1=cbind(food[,2],food[,3],food[,4],food[,5],food[,6],food[,7],
food[,8],food[,9],food[,10])
X1[1:3,]
set.seed(1)
out2all<-mvnormalmixEM(X1,arbvar=FALSE,k=2,epsilon=1e-02)
out2all
prob1=round(out2all$posterior[,1],digits=3)
prob2=round(out2all$posterior[,2],digits=3)
prob=round(out2all$posterior[,1])
data.frame(food$Country,prob1,prob2,prob)
o=order(prob)
data.frame(food$Country[o],prob[o])
```

R program to create Figure 15.1

```
library(cluster)
dis=matrix(nrow=5,ncol=5)
dis[1,1]=0
dis[2,2]=0
dis[3,3]=0
dis[4,4]=0
dis[5,5]=0
dis[2,1]=9
dis[3,1]=3
dis[4,1]=6
dis[5,1]=11
dis[3,2]=7
dis[4,2]=5
dis[5,2]=10
dis[4,3]=9
dis[5,3]=2
dis[5,4]=8
dis[1,2]=dis[2,1]
dis[1,3]=dis[3,1]
dis[1,4]=dis[4,1]
dis[1,5]=dis[5,1]
dis[2,3]=dis[3,2]
dis[2,4]=dis[4,2]
dis[2,5]=dis[5,2]
dis[3,4]=dis[4,3]
dis[3,5]=dis[5,3]
dis[4,5]=dis[5,4]
plot(agnes(x=dis,diss=TRUE,metric="eucledian",method="single"))
plot(agnes(x=dis,diss=TRUE,metric="eucledian",method="complete"))


## correction with dis[5,3]=9
dis=matrix(nrow=5,ncol=5)
dis[1,1]=0
dis[2,2]=0
dis[3,3]=0
dis[4,4]=0
dis[5,5]=0
dis[2,1]=9
dis[3,1]=3
dis[4,1]=6
dis[5,1]=11
dis[3,2]=7
dis[4,2]=5
dis[5,2]=10
dis[4,3]=9
dis[5,3]=9  ## corrected
dis[5,4]=8
dis[1,2]=dis[2,1]
dis[1,3]=dis[3,1]
dis[1,4]=dis[4,1]
dis[1,5]=dis[5,1]
dis[2,3]=dis[3,2]
```

```
dis[2,4]=dis[4,2]
dis[2,5]=dis[5,2]
dis[3,4]=dis[4,3]
dis[3,5]=dis[5,3]
dis[4,5]=dis[5,4]
plot(agnes(x=dis,diss=TRUE,metric="eucledian",method="single"))
plot(agnes(x=dis,diss=TRUE,metric="eucledian",method="complete"))
```

Example 4: European Protein Consumption Revisited (Agglomerative Clustering)

```
library(cluster)
## Protein Data
food <- read.csv("C:/DataMining/Data/protein.csv")
food[1:3,]
## we use the program agnes in the package cluster
## argument diss=FALSE indicates that we use the dissimilarity
## matrix that is being calculated from raw data.
## argument metric="euclidian" indicates that we use Euclidian distance
## no standardization is used as the default
## the default is "average" linkage

## first we consider just Red and White meat clusters
food2=food[,c("WhiteMeat","RedMeat")]
food2agg=agnes(food2,diss=FALSE,metric="euclidian")
food2agg
plot(food2agg)    ## dendrogram
food2agg$merge    ## describes the sequential merge steps
## identical result obtained by first computing the distance matrix
food2aggv=agnes(daisy(food2),metric="euclidian")
plot(food2aggv)

## Using data on all nine variables (features)
## Euclidean distance and average linkage
foodagg=agnes(food[,-1],diss=FALSE,metric="euclidian")
plot(foodagg)     ## dendrogram
foodagg$merge     ## describes the sequential merge steps
## Using data on all nine variables (features)
## Euclidean distance and single linkage
foodaggsin=agnes(food[,-1],diss=FALSE,metric="euclidian",method="single")
plot(foodaggsin)  ## dendrogram
foodaggsin$merge  ## describes the sequential merge steps
## Euclidean distance and complete linkage
foodaggcomp=agnes(food[,-1],diss=FALSE,metric="euclidian",method="single")
plot(foodaggcomp) ## dendrogram
foodaggcomp$merge ## describes the sequential merge steps
```

Example 4: Monthly US Unemployment Rates (Agglomerative Clustering)

```
library(cluster)
## US unemployment data
library(cluster)
raw <- read.csv("C:/DataMining/Data/unempstates.csv")
raw[1:3,]
rawt=matrix(nrow=50,ncol=416)
rawt=t(raw)
rawt[1:3,]
## transpose so that we have 50 rows (states) and 416 columns
## (time periods)
## Agglomerative clustering unemployment 50 states ###
## dissimilarity matrix calculated from the raw data.
```

```
## Euclidian distance and default "average" linkage
outagg=agnes(rawt,diss=FALSE,metric="euclidian")
plot(outagg)      ## dendrogram
outagg$merge      ## describes the sequential merge steps
## we see about three clusters
## Cluster 1: AL, IL, OH, TN, KY, OR, WA, PA, IN, MO, WI, NC, NV, SC,
##            AR, NM, ID, MT, TX, AZ, FL, GA, ME, NJ, NY, RI, CA
## Cluster 2: AK, LA, MS, WV, MI
## Cluster 3: CO, IA, MN, UT, KS, OK, WY, NE, SD, ND, CT, MA, DE, MD,
##            VT, VA, NH, HI
```

Example 5: Monthly US Unemployment Rates Revisited

```
## agglomerative clustering on the correlation between the series
## 2 versions: levels and differences

library(cluster)
raw <- read.csv("C:/DataMining/Data/unempstates.csv")
raw[1:3,]

## Correlation on levels
corlevel=cor(data.frame(raw))
disslevel=1-corlevel
outcorlevel=agnes(disslevel,diss=TRUE,metric="euclidian",method="single")
plot(outcorlevel) ## dendrogram; single linkage
outcorlevel=agnes(disslevel,diss=TRUE,metric="euclidian",method="complete")
plot(outcorlevel) ## dendrogram; complete linkage
outcorlevel=agnes(disslevel,diss=TRUE,metric="euclidian")
plot(outcorlevel) ## dendrogram; average linkage

## Correlation on differences
X=matrix(nrow=415,ncol=50)
for (j in 1:50) {
for (i in 1:415) {
X[i,j]=raw[i+1,j]-raw[i,j]
}
}
colnames(X)=colnames(raw)
cordiff=cor(data.frame(X))
dissdiff=1-cordiff
outcordiff=agnes(dissdiff,diss=TRUE,metric="euclidian",method="single")
plot(outcordiff)  ## dendrogram; single linkage
outcordiff=agnes(dissdiff,diss=TRUE,metric="euclidian",method="complete")
plot(outcordiff)  ## dendrograml; complete linkage
outcordiff=agnes(dissdiff,diss=TRUE,metric="euclidian")
plot(outcordiff)  ## dendrogram; average linkage
```

CHAPTER 16: MARKET BASKET ANALYSIS: ASSOCIATION RULES AND LIFT

Example 1: Online Radio

```
### *** Play counts *** ###

lastfm <- read.csv("C:/DataMining/Data/lastfm.csv")
lastfm[1:19,]
length(lastfm$user)   ## 289,955 records in the file
lastfm$user <- factor(lastfm$user)
levels(lastfm$user)   ## 15,000 users
levels(lastfm$artist) ##  1,004 artists

library(arules) ## a-rules package for association rules
## Computational environment for mining association rules and
## frequent item sets

## we need to manipulate the data a bit for arules
playlist <- split(x=lastfm[,"artist"],f=lastfm$user) ## split into a list of
users
playlist <- lapply(playlist,unique)     ## remove artist duplicates
playlist[1:2]
## the first two listeners (1 and 3) listen to the following bands

playlist <- as(playlist,"transactions")
## view this as a list of "transactions"
## transactions is a data class defined in arules

itemFrequency(playlist)
## lists the support of the 1,004 bands
## number of times band is listed to on the shopping trips of 15,000 users
## computes the rel freq each artist mentioned by the 15,000 users

itemFrequencyPlot(playlist,support=.08,cex.names=1.5)
## plots the item frequencies (only bands with > % support)

## Finally, we build the association rules
## only rules with support > 0.01 and confidence > .50
## so it can't be a super rare band

musicrules <- apriori(playlist,parameter=list(support=.01,confidence=.5))

inspect(musicrules)

## let's filter by lift > 5.
## Among those associations with support > 0.01 and confidence > .50,
## only show those with lift > 5

inspect(subset(musicrules, subset=lift > 5))

## lastly, order by confidence to make it easier to understand

inspect(sort(subset(musicrules, subset=lift > 5), by="confidence"))
```

Example 2: Predicting Income

```
library(arules)
data(AdultUCI)
dim(AdultUCI)
AdultUCI[1:3,]

AdultUCI[["fnlwgt"]] <- NULL
AdultUCI[["education-num"]] <- NULL
AdultUCI[["age"]] <- ordered(cut(AdultUCI[["age"]], c(15, 25, 45, 65, 100)),
labels = c("Young", "Middle-aged", "Senior", "Old"))
AdultUCI[["hours-per-week"]] <- ordered(cut(AdultUCI[["hours-per-week"]], c(0,
25, 40, 60, 168)), labels = c("Part-time", "Full-time", "Over-time",
"Workaholic"))
AdultUCI[["capital-gain"]] <- ordered(cut(AdultUCI[["capital-gain"]], c(-Inf,
0, median(AdultUCI[["capital-gain"]][AdultUCI[["capital-gain"]] > 0]), Inf)),
labels = c("None", "Low", "High"))
AdultUCI[["capital-loss"]] <- ordered(cut(AdultUCI[["capital-loss"]], c(-Inf,
0, median(AdultUCI[["capital-loss"]][AdultUCI[["capital-loss"]] > 0]), Inf)),
labels = c("none", "low", "high"))

Adult <- as(AdultUCI, "transactions")
Adult
summary(Adult)

aa=as(Adult,"matrix") # transforms transaction matrix into incidence matrix
aa[1:2,]   # print the first two rows of the incidence matrix
itemFrequencyPlot(Adult[, itemFrequency(Adult) > 0.2], cex.names = 1)

rules <- apriori(Adult, parameter = list(support = 0.01, confidence = 0.6))
rules
summary(rules)
rulesIncomeSmall <- subset(rules, subset = rhs %in% "income=small" & lift >
1.2)
inspect(sort(rulesIncomeSmall, by = "confidence")[1:3])
rulesIncomeLarge <- subset(rules, subset = rhs %in% "income=large" & lift >
1.2)
inspect(sort(rulesIncomeLarge, by = "confidence")[1:3])
```

CHAPTER 17: DIMENSION-REDUCTION: FACTOR MODELS AND PRINCIPAL COMPONENTS

Example 1: European Protein Consumption

```
food <- read.csv("C:/DataMining/Data/protein.csv")
food
## correlation matrix
cor(food[,-1])
pcafood <- prcomp(food[,-1], scale=TRUE)
## we strip the first column (country labels) from the data set
## scale = TRUE: variables are first standardized. Default is FALSE
pcafood
foodpc <- predict(pcafood)
foodpc
## how many principal components do we need?
plot(pcafood, main="")
mtext(side=1, "European Protein Principal Components",  line=1, font=2)
## how do the PCs look?
par(mfrow=c(1,2))
plot(foodpc[,1:2], type="n", xlim=c(-4,5))
text(x=foodpc[,1], y=foodpc[,2], labels=food$Country)
plot(foodpc[,3:4], type="n", xlim=c(-3,3))
text(x=foodpc[,3], y=foodpc[,4], labels=food$Country)
pcafood$rotation[,2]
```

Example 2: Monthly US Unemployment Rates

```
library(cluster)  ## needed for cluster analysis

states=c("AL","AK","AZ","AR","CA","CO","CT","DE","FL","GA","HI","ID","IL","IN"
,"IA","KS","KY","LA","ME","MD","MA","MI","MN","MS","MO","MT","NE","NV","NH","N
J",
"NM","NY","NC","ND","OH","OK","OR","PA","RI","SC","SD","TN","TX","UT","VT",
"VA","WA","WV","WI","WY")
states

raw <- read.csv("C:/DataMining/Data/unempstates.csv")
raw[1:3,]

## transpose so that we have 50 rows (states) and 416 columns
rawt=matrix(nrow=50,ncol=416)
rawt=t(raw)
rawt[1:3,]

pcaunemp <- prcomp(rawt,scale=FALSE)
pcaunemp

plot(pcaunemp, main="")
mtext(side=1,"Unemployment: 50 states",line=1,font=2)

pcaunemp$rotation[,1]
pcaunemp$rotation[1:10,1]   ## just the first 10 values

ave=dim(416)
for (j in 1:416) {
ave[j]=mean(rawt[,j])
}

par(mfrow=c(1,2))
plot(-pcaunemp$rotation[,1])  ## plot negative loadings for first principal
comp
## plot monthly averages of unemployment rates
plot(ave,type="l",ylim=c(3,10),xlab="month",ylab="ave unemployment rate")
abs(cor(ave,pcaunemp$rotation[,1]))

pcaunemp$rotation[,2]
pcaunemp$rotation[,3]

## below we obtain the scores of the principal components
## the first 2-3 principal components do a good job
unemppc <- predict(pcaunemp)
unemppc

## below we construct a scatter plot of the first two princ components
## we assess whether an informal clustering on the first two principal
components
## would have lead to a similar clustering than the clustering results of the
## k-means clustering approach applied on all 416 components
## the graph indicates that it does
```

```
set.seed(1)
grpunemp3 <- kmeans(rawt,centers=3,nstart=10)
par(mfrow=c(1,1))
plot(unemppc[,1:2],type="n")
text(x=unemppc[,1],y=unemppc[,2],labels=states,col=rainbow(7)
[grpunemp3$cluster])
```

CHAPTER 18: REDUCING THE DIMENSION IN REGRESSIONS WITH MULTICOLLINEAR INPUTS: PRINCIPAL COMPONENTS REGRESSION AND PARTIAL LEAST SQUARES

Example 1: Generated Data

```
## PLS algorithm, following algorithm 3.3 in Hastie et al
## standardize X's. PLS depends on scale
## we can't have too many partial least squares directions (nc)
## otherwise problems
## here we simulate observations

set.seed(1)
nrow=400    ## row dimension of X
ncol=100    ## column dimension of X
nc=2        ## number of PLS directions
nc1=nc+1

Y1=dim(nrow)
X=matrix(nrow=nrow,ncol=ncol)
X1=matrix(nrow=nrow,ncol=ncol)
Y=matrix(nrow=nrow,ncol=nc1)
Z=matrix(nrow=nrow,ncol=nc)
F=matrix(nrow=nrow,ncol=ncol)
FN=matrix(nrow=nrow,ncol=ncol)
me=dim(ncol)
s=dim(ncol)

## enter data into matrix X1 and column Y1
## data simulation
for (jj in 1:ncol) {
X1[,jj]=rnorm(nrow)
}
Y1=rnorm(nrow)
Y1=2*Y1+6

## standardization
for (j in 1:ncol) {
me[j]=mean(X1[,j])
s[j]=sd(X1[,j])
}
for (j in 1:ncol) {
for (i in 1:nrow) {
X[i,j]=(X1[i,j]-me[j])/s[j]
}
}

## Algorithm 3.3 starts
y=Y1
F=X
Y[,1]=mean(y)*y/y

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

```
phi=t(F)%*%y
Z[,k]=F%*%phi
fra=(t(Z[,k])%*%y)/(t(Z[,k])%*%Z[,k])
Y[,k+1]=Y[,k]+fra*Z[,k]
for (j in 1:ncol) {
fru=(t(Z[,k])%*%F[,j])/(t(Z[,k])%*%Z[,k])
FN[,j]=F[,j]-fru*Z[,k]
}
F=FN
}
fp=Y[,nc+1]
## Algorithm 3.3 ends

cor(y,fp)**2

ZZ=data.frame(Z[,1:nc])
m1=lm(y~.,data=ZZ)
cor(y,m1$fitted)**2

XX=data.frame(X)
mall=lm(y~.,data=XX)
cor(y,mall$fitted)**2

## even with few PLS directions, R**2 of largest model is
## approached very quickly

## comparison with library(mixOmics)
library(mixOmics)
mpls=pls(X1,Y1,ncomp=2,mode="classic")
x1=mpls$variates$X[,1]
x2=mpls$variates$X[,2]
m3=lm(y~x1+x2)
cor(y,m3$fitted)**2

fmpls=m3$fitted
## fmpls and fp (and m1$fitted are all the same)
```

Example 2: Predicting Next Month's Unemployment Rate of a Certain State from Past Unemployment Rates of all 50 States

```
library(mixOmics)

nrow=412    ## row dimension of X
ncol=200    ## column dimension of X
nstates=50  ## number of states

X=matrix(nrow=nrow,ncol=ncol)
Y=matrix(nrow=nrow,ncol=nstates)

raw <- read.csv("C:/DataMining/Data/unempstates.csv")
raw[1:3,]

X=matrix(nrow=412,ncol=200)
Y=matrix(nrow=412,ncol=50)

for (j in 1:50) {
for (i in 1:412) {
Y[i,j]=raw[i+4,j]
}
}

for (j in 1:50) {
for (i in 1:412) {
X[i,j]=raw[i+3,j]
X[i,j+50]=raw[i+2,j]
X[i,j+100]=raw[i+1,j]
X[i,j+150]=raw[i,j]
}
}

nc=1        ## number of PLS directions
## pls on nc components
mpls=pls(X,Y[,1],ncomp=nc,mode="classic")
m1=lm(Y[,1]~.,data.frame(mpls$variates$X))
summary(m1)
cor(Y[,1],m1$fitted)**2

nc=2        ## number of PLS directions
## pls on nc components
mpls=pls(X,Y[,1],ncomp=nc,mode="classic")
m2=lm(Y[,1]~.,data.frame(mpls$variates$X))
summary(m2)
cor(Y[,1],m2$fitted)**2

nc=3        ## number of PLS directions
## pls on nc components
mpls=pls(X,Y[,1],ncomp=nc,mode="classic")
m3=lm(Y[,1]~.,data.frame(mpls$variates$X))
summary(m3)
cor(Y[,1],m3$fitted)**2

## regression on all columns of X
```

```
mreg=lm(Y[,1]~.,data.frame(X))
mreg
cor(Y[,1],mreg$fitted)**2
```

Example 3: Predicting Next Month's Unemployment Rate. Comparing Several Methods in Terms of their Out-Of-Sample Prediction Performance

R Code for predicting levels

```
## the program will run for some time
library(mixOmics)
library(lars)
set.seed(1)

nrow=412    ## row dimension of X
ncol=200    ## column dimension of X
nstates=50  ## number of states
nc=10       ## number of PLS directions

X=matrix(nrow=nrow,ncol=ncol)
Y=matrix(nrow=nrow,ncol=nstates)

raw <- read.csv("C:/DataMining/Data/unempstates.csv")
raw[1:3,]

X=matrix(nrow=412,ncol=200)
Y=matrix(nrow=412,ncol=50)

for (j in 1:50) {
for (i in 1:412) {
Y[i,j]=raw[i+4,j]
}
}

for (j in 1:50) {
for (i in 1:412) {
X[i,j]=raw[i+3,j]
X[i,j+50]=raw[i+2,j]
X[i,j+100]=raw[i+1,j]
X[i,j+150]=raw[i,j]
}
}

KK=50
sSAR=dim(KK)      ## univariate (single) AR(4)
sVARL25=dim(KK)   ## VAR(4) under various Lasso constraints
sVARL50=dim(KK)
sVARL75=dim(KK)
sVARL100=dim(KK)
sPC10=dim(KK)     ## regression on 10 principal components
sPC200=dim(KK)    ## regression on 200 principal components
sPLS10=dim(KK)    ## partial least squares with 10 PLS directions
sPLS100=dim(KK)   ## partial least squares with 100 PLS directions
predmpc=dim(25)

## We select 25 periods as the evaluation (holdout) sample
## We repeat this KK = 50 times
## We calculate the root mean square forecast error
```

```
## from the 25 periods and the 50 states

for (jj in 1:KK) {
eval=sample(1:412,25)

## Forecasting from individual (univariate) AR(4) models:
s=0
for (i in 1:50) {
y=Y[-eval,i]
p1=X[-eval,i]
p2=X[-eval,i+50]
p3=X[-eval,i+100]
p4=X[-eval,i+150]
mSAR=lm(y~p1+p2+p3+p4)
pr1=X[eval,i]
pr2=X[eval,i+50]
pr3=X[eval,i+100]
pr4=X[eval,i+150]
new=data.frame(p1=pr1,p2=pr2,p3=pr3,p4=pr4)
predSAR=predict(mSAR,new)
s=s+sum((Y[eval,i]-predSAR)**2)
}
sSAR[jj]=sqrt(s/(50*25))

## Forecasting from VAR(4) models and various LASSO constraints
s1=0
s2=0
s3=0
s4=0
for (i in 1:50) {
lasso=lars(X[-eval,],Y[-eval,i])
predLASSO25=predict(lasso,X[eval,],s=.25,mode="fraction")
s1=s1+sum((Y[eval,i]-predLASSO25$fit)**2)
predLASSO50=predict(lasso,X[eval,],s=0.50,mode="fraction")
s2=s2+sum((Y[eval,i]-predLASSO50$fit)**2)
predLASSO75=predict(lasso,X[eval,],s=0.75,mode="fraction")
s3=s3+sum((Y[eval,i]-predLASSO75$fit)**2)
predLASSO100=predict(lasso,X[eval,],s=1.00,mode="fraction")
s4=s4+sum((Y[eval,i]-predLASSO100$fit)**2)
}
sVARL25[jj]=sqrt(s1/(50*25))
sVARL50[jj]=sqrt(s2/(50*25))
sVARL75[jj]=sqrt(s3/(50*25))
sVARL100[jj]=sqrt(s4/(50*25))

## Forecasting from regressions on first 10 principal components:
pcaX <- prcomp(X[-eval,])
pred=predict(pcaX,data.frame(X[-eval,]))
pred=data.frame(pred)
names(pred)=paste("p", 1:200, sep="")
pred1=predict(pcaX,data.frame(X[eval,]))
s=0
for (i in 1:50) {
mpc=lm(Y[-eval,i]~p1+p2+p3+p4+p5+p6+p7+p8+p9+p10,data.frame(pred))
g=mpc$coef
for (j in 1:25) {
h=pred1[j,1:10]
```

```
h1=c(1,h)
predmpc[j]=sum(h1*g)
}
s=s+sum((Y[eval,i]-predmpc)**2)
}
sPC10[jj]=sqrt(s/(50*25))

## Forecasting from regressions on all 200 principal components
s=0
for (i in 1:50) {
mpc=lm(Y[-eval,i]~.,data.frame(pred))
g=mpc$coef
for (j in 1:25) {
h=pred1[j,]
h1=c(1,h)
predmpc[j]=sum(h1*g)
}
s=s+sum((Y[eval,i]-predmpc)**2)
}
sPC200[jj]=sqrt(s/(50*25))

## Forecasting from regressions on first 10 PLS components:
s=0
for (i in 1:50) {
m1=pls(X[-eval,],Y[-eval,i],ncomp=10,mode="classic")
p1=m1$variates$X[,1]
p2=m1$variates$X[,2]
p3=m1$variates$X[,3]
p4=m1$variates$X[,4]
p5=m1$variates$X[,5]
p6=m1$variates$X[,6]
p7=m1$variates$X[,7]
p8=m1$variates$X[,8]
p9=m1$variates$X[,9]
p10=m1$variates$X[,10]
mpc=lm(Y[-eval,i]~p1+p2+p3+p4+p5+p6+p7+p8+p9+p10)
g=mpc$coef
pre1=predict(m1,X[eval,])
for (j in 1:25) {
h=pre1$variates[j,1:10]
h1=c(1,h)
predmpc[j]=sum(h1*g)
}
s=s+sum((Y[eval,i]-predmpc)**2)
}
sPLS10[jj]=sqrt(s/(50*25))

## Forecasting from regressions on first 100 PLS components:
s=0
for (i in 1:50) {
m1=pls(X[-eval,],Y[-eval,i],ncomp=100,mode="classic")
ppp=data.frame(m1$variates$X)
mpc=lm(Y[-eval,i]~.,data=ppp)
g=mpc$coef
pre1=predict(m1,X[eval,])
for (j in 1:25) {
h=pre1$variates[j,1:100]
```

```
h1=c(1,h)
predmpc[j]=sum(h1*g)
}
s=s+sum((Y[eval,i]-predmpc)**2)
}
sPLS100[jj]=sqrt(s/(50*25))

}
## Output
sSAR
sVARL25
sVARL50
sVARL75
sVARL100
sPC10
sPC200
sPLS10
sPLS100

mean(sSAR)
mean(sVARL25)
mean(sVARL50)
mean(sVARL75)
mean(sVARL100)
mean(sPC10)
mean(sPC200)
mean(sPLS10)
mean(sPLS100)

boxplot(sSAR,sVARL25,sVARL50,sVARL75,sVARL100,sPC10,sPC200,sPLS10,sPLS100,ylab
="RMSE",xlab="  AR(4)  VAR(4)  VAR(4)  VAR(4)  VAR(4)  PCA10 PCA200 PLS10
PLS100",sub="   AR(4)  s=0.25   s=0.50   s=0.75   s=1.00   PCA10 PCA200 PLS10
PLS100")
```

R Code for predicting changes

```
## the program will run for some time
library(mixOmics)
library(lars)
set.seed(1)

nrow=412    ## row dimension of X
ncol=200    ## column dimension of X
nstates=50  ## number of states
nc=10       ## number of PLS directions

X=matrix(nrow=nrow,ncol=ncol)
Y=matrix(nrow=nrow,ncol=nstates)

raw <- read.csv("C:/DataMining/Data/unempstates.csv")
raw[1:3,]

X1=matrix(nrow=412,ncol=200)
Y1=matrix(nrow=412,ncol=50)
X=matrix(nrow=411,ncol=200)
```

```
Y=matrix(nrow=411,ncol=50)

## defining the data matrices
for (j in 1:50) {
for (i in 1:412) {
Y1[i,j]=raw[i+4,j]
}
}

for (j in 1:50) {
for (i in 1:412) {
X1[i,j]=raw[i+3,j]
X1[i,j+50]=raw[i+2,j]
X1[i,j+100]=raw[i+1,j]
X1[i,j+150]=raw[i,j]
}
}

## calculating differences
for (j in 1:200) {
for (i in 1:411) {
X[i,j]=X1[i+1,j]-X1[i,j]
}
}
for (j in 1:50) {
for (i in 1:411) {
Y[i,j]=Y1[i+1,j]-Y1[i,j]
}
}

KK=50
sSAR=dim(KK)      ## univariate (single) AR(4)
sVARL25=dim(KK)   ## VAR(4) under various Lasso constraints
sVARL50=dim(KK)
sVARL75=dim(KK)
sVARL100=dim(KK)
sPC10=dim(KK)     ## regression on 10 principal components
sPC200=dim(KK)    ## regression on 200 principal components
sPLS10=dim(KK)    ## partial least squares with 10 PLS directions
sPLS100=dim(KK)   ## partial least squares with 100 PLS directions
predmpc=dim(25)

## We select 25 periods as the evaluation (holdout) sample
## We repeat this KK = 50 times
## We calculate the root mean square forecast error
## from the 25 periods and the 50 states

for (jj in 1:KK) {
eval=sample(1:411,25)

## Forecasting from individual (univariate) AR(4) models:
s=0
for (i in 1:50) {
y=Y[-eval,i]
p1=X[-eval,i]
p2=X[-eval,i+50]
p3=X[-eval,i+100]
```

```
p4=X[-eval,i+150]
mSAR=lm(y~p1+p2+p3+p4)
pr1=X[eval,i]
pr2=X[eval,i+50]
pr3=X[eval,i+100]
pr4=X[eval,i+150]
new=data.frame(p1=pr1,p2=pr2,p3=pr3,p4=pr4)
predSAR=predict(mSAR,new)
s=s+sum((Y[eval,i]-predSAR)**2)
}
sSAR[jj]=sqrt(s/(50*25))

## Forecasting from VAR(4) models and various LASSO constraints
s1=0
s2=0
s3=0
s4=0
for (i in 1:50) {
lasso=lars(X[-eval,],Y[-eval,i])
predLASSO25=predict(lasso,X[eval,],s=.25,mode="fraction")
s1=s1+sum((Y[eval,i]-predLASSO25$fit)**2)
predLASSO50=predict(lasso,X[eval,],s=0.50,mode="fraction")
s2=s2+sum((Y[eval,i]-predLASSO50$fit)**2)
predLASSO75=predict(lasso,X[eval,],s=0.75,mode="fraction")
s3=s3+sum((Y[eval,i]-predLASSO75$fit)**2)
predLASSO100=predict(lasso,X[eval,],s=1.00,mode="fraction")
s4=s4+sum((Y[eval,i]-predLASSO100$fit)**2)
}
sVARL25[jj]=sqrt(s1/(50*25))
sVARL50[jj]=sqrt(s2/(50*25))
sVARL75[jj]=sqrt(s3/(50*25))
sVARL100[jj]=sqrt(s4/(50*25))

## Forecasting from regressions on first 10 principal components:
pcaX <- prcomp(X[-eval,])
pred=predict(pcaX,data.frame(X[-eval,]))
pred=data.frame(pred)
names(pred)=paste("p", 1:200, sep="")
pred1=predict(pcaX,data.frame(X[eval,]))
s=0
for (i in 1:50) {
mpc=lm(Y[-eval,i]~p1+p2+p3+p4+p5+p6+p7+p8+p9+p10,data.frame(pred))
g=mpc$coef
for (j in 1:25) {
h=pred1[j,1:10]
h1=c(1,h)
predmpc[j]=sum(h1*g)
}
s=s+sum((Y[eval,i]-predmpc)**2)
}
sPC10[jj]=sqrt(s/(50*25))

## Forecasting from regressions on all 200 principal components
s=0
for (i in 1:50) {
mpc=lm(Y[-eval,i]~.,data.frame(pred))
g=mpc$coef
```

```
for (j in 1:25) {
h=pred1[j,]
h1=c(1,h)
predmpc[j]=sum(h1*g)
}
s=s+sum((Y[eval,i]-predmpc)**2)
}
sPC200[jj]=sqrt(s/(50*25))

## Forecasting from regressions on first 10 PLS components:
s=0
for (i in 1:50) {
m1=pls(X[-eval,],Y[-eval,i],ncomp=10,mode="classic")
p1=m1$variates$X[,1]
p2=m1$variates$X[,2]
p3=m1$variates$X[,3]
p4=m1$variates$X[,4]
p5=m1$variates$X[,5]
p6=m1$variates$X[,6]
p7=m1$variates$X[,7]
p8=m1$variates$X[,8]
p9=m1$variates$X[,9]
p10=m1$variates$X[,10]
mpc=lm(Y[-eval,i]~p1+p2+p3+p4+p5+p6+p7+p8+p9+p10)
g=mpc$coef
pre1=predict(m1,X[eval,])
for (j in 1:25) {
h=pre1$variates[j,1:10]
h1=c(1,h)
predmpc[j]=sum(h1*g)
}
s=s+sum((Y[eval,i]-predmpc)**2)
}
sPLS10[jj]=sqrt(s/(50*25))

## Forecasting from regressions on first 100 PLS components:
s=0
for (i in 1:50) {
m1=pls(X[-eval,],Y[-eval,i],ncomp=100,mode="classic")
ppp=data.frame(m1$variates$X)
mpc=lm(Y[-eval,i]~.,data=ppp)
g=mpc$coef
pre1=predict(m1,X[eval,])
for (j in 1:25) {
h=pre1$variates[j,1:100]
h1=c(1,h)
predmpc[j]=sum(h1*g)
}
s=s+sum((Y[eval,i]-predmpc)**2)
}
sPLS100[jj]=sqrt(s/(50*25))

}
## Output
sSAR
sVARL25
sVARL50
```

```
sVARL75
sVARL100
sPC10
sPC200
sPLS10
sPLS100

mean(sSAR)
mean(sVARL25)
mean(sVARL50)
mean(sVARL75)
mean(sVARL100)
mean(sPC10)
mean(sPC200)
mean(sPLS10)
mean(sPLS100)

boxplot(sSAR,sVARL25,sVARL50,sVARL75,sVARL100,sPC10,sPC200,sPLS10,sPLS100,ylab
="RMSE",xlab="  AR(4)  VAR(4)  VAR(4)  VAR(4)  VAR(4)  PCA10 PCA200 PLS10
PLS100",sub="   AR(4)  s=0.25   s=0.50   s=0.75   s=1.00   PCA10 PCA200 PLS10
PLS100")
```

CHAPTER 19: TEXT AS DATA: TEXT MINING AND SENTIMENT ANALYSIS

Example 1: Restaurant Reviews

```
library(textir)

data(we8there)          ## 6166 reviews and 2640 bigrams
dim(we8thereCounts)
dimnames(we8thereCounts)
dim(we8thereRatings)
we8thereRatings[1:3,]
## ratings (restaurants ordered on overall rating from 5 to 1)
as.matrix(we8thereCounts)
as.matrix(we8thereCounts)[12,400]   ## count for bigram 400 in review 12

## get to know what's in the matrix
g1=min(as.matrix(we8thereCounts)[,]) ## min count over reviews/bigrams
g2=max(as.matrix(we8thereCounts)[,]) ## max count over reviews/bigrams
g1
g2
## a certain bigram was mentioned in a certain review 13 times
hh=as.matrix(we8thereCounts)[,1000]
hh
## here we look at the frequencies of the bigram in column 1000
## the data are extremely sparce

overall=as.matrix(we8thereRatings[,5])
## overall rating

## we determine frequencies of the 2640 different bigrams
## this will take some time
nn=2640
cowords=dim(nn)
for (i in 1:nn) {
cowords[i]=sum(as.matrix(we8thereCounts)[,i])
}
cowords
cowords[7]
plot(sort(cowords,decreasing=TRUE))

## analysis per review
## we determine the frequencies of bigrams per review
## this will take some time
nn=6166
coreview=dim(nn)
for (i in 1:nn) {
coreview[i]=sum(as.matrix(we8thereCounts)[i,])
}
plot(sort(coreview,decreasing=TRUE))
```

```
## Multinomial logistic regression and fitted reduction
we8mnlm=mnlm(we8thereCounts,overall,bins=5)
## bins: for faster inference if covariates are factors
## covariate is a factor with 5 levels
we8mnlm
we8mnlm$intercept       ## estimates of alphas
we8mnlm$loadings        ## estimates of betas
fitted(we8mnlm)
as.matrix(fitted(we8mnlm))[1,]      ## fitted counts for first review

## following provides fitted multinomial probabilities
pred=predict(we8mnlm,overall,type="response")
pred[1,]    ## predicted multinomial probs for review 1
sum(pred[1,])     ## must add to one

## following predicts inverse prediction (fitted reduction)
predinv=predict(we8mnlm,we8thereCounts,type="reduction")
predinv[1:10]     ## prints predicted ratings for first 10 reviews
plot(predinv)
plot(predinv~overall)
corr(predinv,overall)
boxplot(predinv~overall)
## procedure works. Predicted ratings increase with actual ratings
## question of cutoff. Which cutoff to use for excellent review?

## ROC curve for classification of y with p
roc <- function(p,y){
  y <- factor(y)
  n <- length(p)
  p <- as.vector(p)
  Q <- p > matrix(rep(seq(0,1,length=500),n),ncol=500,byrow=TRUE)
  fp <- colSums((y==levels(y)[1])*Q)/sum(y==levels(y)[1])
  tp <- colSums((y==levels(y)[2])*Q)/sum(y==levels(y)[2])
  plot(fp, tp, xlab="1-Specificity", ylab="Sensitivity")
  abline(a=0,b=1,lty=2,col=8)
}

c2=overall==4
c3=overall==5
c=c2+c3
min=min(predinv)
max=max(predinv)
pp=(predinv-min)/(max-min)

## plot of ROC curve
roc(p=pp, y=c)

cut <- 0
truepos <- c==1 & predinv>=cut
trueneg <- c==0 & predinv<cut
# hit-rate / sensitivity (predict good review if review is good)
sum(truepos)/sum(c==1)

sum(trueneg)/sum(c==0)
## Zero may be a good cutoff.
```

```
## Sensitivity (true positive rate) of 0.89
## False positive rate of 1 – 0.81 = 0.19
## If inverse prediction > 0, conclude overall quality rating 4 or 5.
```

Example 2: Political Sentiment

```
library(textir)
data(congress109)                ## 529 speakers 1000 trigrams
dimnames(congress109Counts)
as.matrix(congress109Counts)[1,] ## Chris Cannon's counts
as.matrix(congress109Counts)[,1] ## "gifted.talented.student" counts
congress109Ideology
as.matrix(congress109Ideology)[,1]
repshare=as.matrix(congress109Ideology[,5])
repshare    ## Republican vote share

## get to know what is in the matrix

g1=min(as.matrix(congress109Counts)[,])
g2=max(as.matrix(congress109Counts)[,])
g1
g2
## a certain trigram was mentioned by a certain speaker 631 times
hh=as.matrix(congress109Counts)[,1000]
hh
## here we look at the frequencies of bigram in column 1000

## Multinomial logistic regression and fitted reduction
congmnlm=mnlm(congress109Counts,repshare)
## this may take some time
congmnlm
congmnlm$intercept      ## estimates of alphas
congmnlm$loadings       ## estimates of betas
fitted(congmnlm)
as.matrix(fitted(congmnlm))[1,]     ## fitted counts for first rep
maxf=max(as.matrix(fitted(congmnlm))[1,])
maxf
maxc=max(as.matrix(congress109Counts)[1,])
maxc

## following provides fitted multinomial probabilities
pred=predict(congmnlm,repshare,type="response")
pred[1,]    ## predicted multinomial probs for first rep

## following predicts inverse prediction (fitted reduction)
predinv=predict(congmnlm,congress109Counts,type="reduction")
predinv[1:10]     ## prints predicted ratings for first 10 reps

plot(predinv~repshare)
plot(repshare~predinv)

corr(predinv,repshare)

model1=lm(repshare~predinv)
model1

plot(repshare~predinv)
abline(model1)
```

Appendix: Relationship between the Gentzkow/Shapiro Estimate of "Slant" and Partial Least Squares

```
library(textir)
data(congress109) ## data form Gentzkow/Shapiro

## Gentzkow/Shapiro slant (unstandardized relative frequencies)
a=dim(529)
b=dim(529)
d=dim(1000)
hh=as.matrix(freq(congress109Counts))
x=congress109Ideology$repshare
for (j in 1:1000) {
m1=lm(hh[,j]~x)
a[j]=m1$coef[1]
b[j]=m1$coef[2]
}
for (i in 1:529) {
d[i]=sum((hh[i,]-a)*b)
}
cor(d,x)**2

## Gentzkow/Shapiro slant (standardized relative frequencies)
hh=as.matrix(freq(congress109Counts))
for (j in 1:1000) {
hh[,j]=(hh[,j]-mean(hh[,j]))/sd(hh[,j])
}
x=congress109Ideology$repshare
for (j in 1:1000) {
m1=lm(hh[,j]~x)
a[j]=m1$coef[1]
b[j]=m1$coef[2]
}
for (i in 1:529) {
d[i]=sum((hh[i,]-a)*b)
}
cor(d,x)**2

## Using PLS (textir) on first partial least squares direction
## scaling FALSE means unstandardized relative frequencies are used
library(textir)
fit=pls(freq(congress109Counts),congress109Ideology$repshare,scale=FALSE,K=1)
cor(congress109Ideology$repshare,fit$fitted)**2

## Using PLS (textir) on first partial least squares direction
## scaling TRUE means standardized relative frequencies
## mean zero and variance 1
library(textir)
fit=pls(freq(congress109Counts),congress109Ideology$repshare,scale=TRUE,K=1)
cor(congress109Ideology$repshare,fit$fitted)**2

## Using PLS (mixOmics) on first partial least squares direction
## standardized relative frequencies (mean zero and variance 1)
library(mixOmics)
```

```
mpls=pls(freq(congress109Counts),congress109Ideology$repshare,ncomp=1,mode="cl
assic",freqCut=0.000001,uniqueCut=0.000001)
x1=mpls$variates$X[,1]
m1=lm(congress109Ideology$repshare~x1)
fmpls=m1$fitted
cor(x,m1$fitted)**2
```

CHAPTER 20: NETWORK DATA

```
library(igraph)
m=matrix(nrow=3,ncol=3)
m[1,1]=0
m[1,2]=1
m[1,3]=1
m[2,1]=1
m[2,2]=0
m[2,3]=0
m[3,1]=0
m[3,2]=1
m[3,3]=0
m
lab=c(1,2,3)
object <- graph.adjacency(m,mode="directed")
set.seed(1)
plot(object,vertex.label=lab)
```

Example 1: Marriage and Power in 15th Century Florence

```
library(igraph) ## load the package
## read the data
florence <- as.matrix(read.csv("C:/DataMining/Data/firenze.csv"))
florence

marriage <- graph.adjacency(florence,mode="undirected", diag=FALSE)
## use the help function to understand the options for the graph
set.seed(1)
plot(marriage,layout=layout.fruchterman.reingold,vertex.label=V(marriage)
$name,vertex.color="red",vertex.label.color="black",
vertex.frame.color=0,vertex.label.cex=1.5)

data.frame(V(marriage)$name,degree(marriage))

## calculate and plot the shortest paths
V(marriage)$color <- 8
E(marriage)$color <- 8
PtoA <- get.shortest.paths(marriage, from="Peruzzi", to="Acciaiuoli")
E(marriage, path=PtoA[[1]])$color <- "magenta"
V(marriage)[PtoA[[1]] ]$color <- "magenta"
GtoS <- get.shortest.paths(marriage, from="Ginori", to="Strozzi")
E(marriage, path=GtoS[[1]])$color <- "green"
V(marriage)[ GtoS[[1]] ]$color <- "green"
V(marriage)[ "Medici" ]$color <- "cyan"

set.seed(1)
plot(marriage,  layout=layout.fruchterman.reingold, vertex.label=V(marriage)
$name,vertex.label.color="black", vertex.frame.color=0, vertex.label.cex=1.5)

data.frame(V(marriage)$name, betweenness(marriage))
```

Example 2: Connections in a Friendship Network

```
library(statnet)
data(faux.mesa.high)          ## load the network object
summary(faux.mesa.high)       ## summarize the data set
lab=network.vertex.names(faux.mesa.high)=c(1:205)
## assigns numbers to nodes
grd=faux.mesa.high%v%"Grade"
sx=faux.mesa.high%v%"Sex"
race=faux.mesa.high%v%"Race"  ## we don't look at race in this example
vs=c(4,12)[match(sx,c("M","F"))]
## used for graph later on; boys by square (4 sides); girls by 12-sided
col=c(6,5,3,7,4,2)            ## used for graph later on
as.sociomatrix(faux.mesa.high)## gives adjacency matrix
faux.mesa.high[1,]
faux.mesa.high[5,]
faux.mesa.high[,3]
m=faux.mesa.high[,]           ## adjacency matrix
network.density(faux.mesa.high)
## density of network = NuEdges/[nodes*(nodes-1)/2]

deg=degree(faux.mesa.high)/2
## degree of network nodes (number of connections)
## Statnet double-counts the connections in an undirected network
## Edge between nodes i and j in an undirected network is counted twice
## We divide by 2 in order to make the results consistent with our
## discussion in the text and the output from igraph (in Example 1)
deg

betw=betweenness(faux.mesa.high)/2
## betweenness of network
## Statnet double-counts the betweenness in an undirected network
## We divide by 2 in order to make the results consistent with our
## discussion in the text and the output from igraph
betw

plot(deg)
plot(betw)
hist(deg,breaks=
c(-0.5,0.5,1.5,2.5,3.5,4.5,5.5,6.5,7.5,8.5,9.5,10.5,11.5,12.5,13.5))

plot(deg,betw)

boxplot(deg~grd)
boxplot(deg~sx)

## faux.mesa.high is already a network object
## below we illustrate how to create an undirected network
## from the edge list
## first we obtain the edge list of a network object
attributes(faux.mesa.high)
vv=faux.mesa.high$mel
edge=matrix(nrow=203,ncol=2)
for (i in 1:203) {
vvv=vv[[203+i]]
edge[i,1]=vvv$inl
```

```
edge[i,2]=vvv$outl
}
edge
## edge contains the edge list
## in an undirected network, edge information is stored in the
## second half of faux.mesa.high$mel
faux1=network(edge,directed=FALSE,matrix.type="edgelist")
faux1
faux1[,]
deg=degree(faux1)/2
betw=betweenness(faux1)/2
plot(deg)
plot(betw)
plot(deg,betw)

## faux.mesa.high is already a network object
## below we illustrate how to create an undirected network
## from the adjacency matrix
## the adjacency matrix had been stored previously in m
faux2=network(m,directed=FALSE,matrix.type="adjacency")
faux2
faux2[,]
deg=degree(faux2)/2
betw=betweenness(faux2)/2
plot(deg)
plot(betw)
plot(deg,betw)

## visual display of the network
set.seed(654)          ## to get reproducible graphs
plot(faux.mesa.high)   ## generic graph without labels/covariates

set.seed(654)          ## to get reproducible graphs
plot(faux.mesa.high,label=lab) ## generic graph with labels

set.seed(654)          ## to get reproducible graphs
plot(faux.mesa.high,vertex.sides=vs,vertex.rot=45,vertex.cex=2,
vertex.col=col[grd-6],edge.lwd=2,cex.main=3,displayisolates=FALSE)
legend("bottomright",legend=7:12,fill=col,cex=0.75)
## 45 rotates square
## isolates are not displayed

## density of interaction among students from the
## same grade (ignoring gender)

m1=m[grd==7,grd==7]
sum(m1)/(nrow(m1)*(ncol(m1)-1))

m1=m[grd==8,grd==8]
sum(m1)/(nrow(m1)*(ncol(m1)-1))

m1=m[grd==9,grd==9]
sum(m1)/(nrow(m1)*(ncol(m1)-1))

m1=m[grd==10,grd==10]
sum(m1)/(nrow(m1)*(ncol(m1)-1))
```

```
m1=m[grd==11,grd==11]
sum(m1)/(nrow(m1)*(ncol(m1)-1))

m1=m[grd==12,grd==12]
sum(m1)/(nrow(m1)*(ncol(m1)-1))

## density of interaction among students from a given grade
## with students from all grades (ignoring gender)
## matrix m1 shown below is not square; it has r rows and c columns
## the c columns include the r nodes that determine the rows of m1
## the number of possible edges in m1 are r(r-1) + r(c-r) = r(c-1)

m1=m[grd==7,]
sum(m1)/(nrow(m1)*(ncol(m1)-1))

m1=m[grd==8,]
sum(m1)/(nrow(m1)*(ncol(m1)-1))

m1=m[grd==9,]
sum(m1)/(nrow(m1)*(ncol(m1)-1))

m1=m[grd==10,]
sum(m1)/(nrow(m1)*(ncol(m1)-1))

m1=m[grd==11,]
sum(m1)/(nrow(m1)*(ncol(m1)-1))

m1=m[grd==12,]
sum(m1)/(nrow(m1)*(ncol(m1)-1))


## density of interaction among students from the
## same gender group (ignoring grade)

m1=m[sx=="F",sx=="F"]
sum(m1)/(nrow(m1)*(ncol(m1)-1))

m1=m[sx=="M",sx=="M"]
sum(m1)/(nrow(m1)*(ncol(m1)-1))


## density of interaction among students from a given gender
## group with students of either gender (ignoring grade)

m1=m[sx=="F",]
sum(m1)/(nrow(m1)*(ncol(m1)-1))

m1=m[sx=="M",]
sum(m1)/(nrow(m1)*(ncol(m1)-1))


## density of interaction among students from the
## same grade, for given gender

## female seventh graders
```

```
m1=m[sx=="F",sx=="F"]
grd1=grd[sx=="F"]
m2=m1[grd1==7,grd1==7]
sum(m2)/(nrow(m2)*(ncol(m2)-1))

## male seventh graders
m1=m[sx=="M",sx=="M"]
grd1=grd[sx=="M"]
m2=m1[grd1==7,grd1==7]
sum(m2)/(nrow(m2)*(ncol(m2)-1))

## female twelfth graders
m1=m[sx=="F",sx=="F"]
grd1=grd[sx=="F"]
m2=m1[grd1==12,grd1==12]
sum(m2)/(nrow(m2)*(ncol(m2)-1))

## male twelfth graders
m1=m[sx=="M",sx=="M"]
grd1=grd[sx=="M"]
m2=m1[grd1==12,grd1==12]
sum(m2)/(nrow(m2)*(ncol(m2)-1))

## density of interaction among students from a given grade
## with students from all grades, for given gender

## female seventh graders
m1=m[sx=="F",sx=="F"]
grd1=grd[sx=="F"]
m2=m1[grd1==7,]
sum(m2)/(nrow(m2)*(ncol(m2)-1))

## male seventh graders
m1=m[sx=="M",sx=="M"]
grd1=grd[sx=="M"]
m2=m1[grd1==7,]
sum(m2)/(nrow(m2)*(ncol(m2)-1))

## female twelfth graders
m1=m[sx=="F",sx=="F"]
grd1=grd[sx=="F"]
m2=m1[grd1==12,]
sum(m2)/(nrow(m2)*(ncol(m2)-1))

## male twelfth graders
m1=m[sx=="M",sx=="M"]
grd1=grd[sx=="M"]
m2=m1[grd1==12,]
sum(m2)/(nrow(m2)*(ncol(m2)-1))

## Plotting options. Not that easy. Will make pictures look differently
## Principles of Fruchterman/Reingold:
##    Distribute vertices evenly in the frame
##    Minimize the number of edge crossings
##    Make edge lengths uniform
##    Reflect inherent symmetry
##    Conform to the frame
```

```
set.seed(654)                 ## to get reproducible graphs
plot(faux.mesa.high,mode="fruchtermanreingold",label=lab,vertex.sides=vs,verte
x.rot=45,vertex.cex=2.5,vertex.col=col[grd-
6],edge.lwd=2,cex.main=3,displayisolates=FALSE)
legend("bottomright",legend=7:12,fill=col,cex=0.75)

set.seed(654)                 ## to get reproducible graphs
plot(faux.mesa.high,mode="kamadakawai",label=lab,vertex.sides=vs,vertex.rot=45
,vertex.cex=2.5,vertex.col=col[grd-
6],edge.lwd=2,cex.main=3,displayisolates=FALSE)
legend("bottomright",legend=7:12,fill=col,cex=0.75)

set.seed(654)                 ## to get reproducible graphs
plot(faux.mesa.high,mode="circle",label=lab,vertex.sides=vs,vertex.rot=45,vert
ex.cex=2.5,vertex.col=col[grd-6],edge.lwd=2,cex.main=3,displayisolates=FALSE)
legend("bottomright",legend=7:12,fill=col,cex=0.75)
```

EXERCISES

Exercises 2 (Wolfgang Jank: Business Analystics for Managers. Springer, 2011)

```
hp <- read.csv("C:/DataMining/Data/HousePrices.csv")
hp[1:3,]
dm <- read.csv("C:/DataMining/Data/DirectMarketing.csv")
dm[1:3,]
gd <- read.csv("C:/DataMining/Data/GenderDiscrimination.csv")
gd[1:3,]
ld <- read.csv("C:/DataMining/Data/LoanData.csv")
ld[1:3,]
fi <- read.csv("C:/DataMining/Data/FinancialIndicators.csv")
fi[1:3,]
```

Exercises 3 (Graham Williams: Data Mining with Rattle and R. Springer, 2011)

```
weather <- read.csv("C:/DataMining/Data/weather.csv")
weather[1:3,]
weatherAUS <- read.csv("C:/DataMining/Data/weatherAUS.csv")
weatherAUS[1:3,]
audit <- read.csv("C:/DataMining/Data/audit.csv")
audit[1:3,]
```

Exercises 4 (1989/99 KDD Cup)

```
## read the data
cup98LRN <- read.csv("C:/DataMining/Data/cup98LRN.csv")
cup98LRN[1:3,]

## read the data
cup98VAL <- read.csv("C:/DataMining/Data/cup98VAL.csv")
cup98VAL[1:3,]

## read the data
cup98VALtargt <- read.csv("C:/DataMining/Data/cup98VALtargt.csv")
cup98VALtargt[1:3,]
```

Exercise 5: Byssinosis

```
## read the data
bys <- read.csv("C:/DataMining/Data/byssinosisWeights.csv")
bys
```

Exercise 6: Toxaemia

```
## read the data
tox <- read.csv("C:/DataMining/Data/toxaemiaWeights.csv")
tox
```

Exercises 7 (8 examples)

Example 1: Classification Tree for Identifying Soybean Disease

```
library(ares)
## needed to determine the proportion of missing observations
library(tree)     ## classification trees

## reading the data
soybean15 <- read.csv("C:/DataMining/Data/soybean15.csv")
soybean15[1:3,]

## converting the attributes into factors (nominal scale)
## calculating the proportion of missing observations
miss=dim(36)
for (j in 1:36) {
soybean15[,j]=factor(soybean15[,j])
miss[j]=count.na(soybean15[,j])$na/length(soybean15[,j])
}
miss
## fifth attribute (presence/absence of hail) has 8.27% missing observations

## constructing the classification tree
soytree <- tree(disease ~., data = soybean15, mincut=1)
soytree
summary(soytree)
plot(soytree, col=8)
text(soytree, digits=2)

## cross-validation to prune the tree
set.seed(2)
cvsoy <- cv.tree(soytree, K=10)
cvsoy$size
cvsoy$dev
plot(cvsoy, pch=21, bg=8, type="p", cex=1.5, ylim=c(0,1400))
## shows that the tree has many terminal nodes

soycut <- prune.tree(soytree, best=17)
soycut
summary(soycut)
plot(soycut, col=8)
text(soycut)

plot(soycut, col=8)
## below we have omitted the text as it is difficult to read
## terminal node under 31 is the one on the far right of the graph
## first split: C15ac (to left) and C15b (to the right)
## second split: C1abcd (to left) and C1efg (to right)
## third split: C19a (to left) and C19b (to right)
## fourth split: C28a (to left) and C28bcd (to right)
```

Example 2: Classification Tree for Fitting Contact Lenses

```
library(tree)

## read the data
ContactLens <- read.csv("C:/DataMining/Data/ContactLens.csv")
levels(ContactLens[,1])  ## age
levels(ContactLens[,2])  ## spectacle presription
levels(ContactLens[,3])  ## astigmatism
levels(ContactLens[,4])  ## tear production rate
levels(ContactLens[,5])  ## contact lens
ContactLens

## constructing the classification tree that fits the data perfectly
cltree <- tree(ContactLens ~., data = ContactLens, mindev=0, minsize=1)
cltree
summary(cltree)
plot(cltree, col=8)
text(cltree, digits=2)
## pruning the tree to get a simpler tree
clcut <- prune.tree(cltree, best=3)
clcut
summary(clcut)
plot(clcut, col=8)
text(clcut)
```

Example 3: Determining the Credit Risk Using a Classification Tree

```
library(tree)

## first we read in the data
credit <- read.csv("C:/DataMining/Data/credit.csv")
credit
credit[,1]
credit[,2]
credit[,3]
credit[,4]
credit[,5]

## constructing the classification tree that fits the data perfectly
credittree <- tree(Risk ~., data = credit, mindev=0, minsize=1)
credittree
summary(credittree)
plot(credittree, col=8)
text(credittree, digits=2)
```

Example 4: Determining the Progression of Liver Disease Using a Classification Tree

```
library(tree)

## data set from Witten
## missing data
hepatitis <- read.csv("C:/DataMining/Data/hepatitis.csv")
hepatitis
## calculating YWD = (Age – YWOD)
hepatitis[,20]=hepatitis[,18]-hepatitis[,17]
colnames(hepatitis)[20]= "YWD"
hepatitis[1:3,]
## cleaning up the data set
hh=hepatitis[,c(-2:-4,-17)]
hh[1:3,]
## create factors for the categorical variables
for (j in 1:13) {
hh[,j]=factor(hh[,j])
}
hh[1:3,]
levels(hh[,6])
levels(hh[,8])
levels(hh[,13])

## constructing the classification tree
heptree <- tree(Bx ~., data = hh)
heptree
summary(heptree)
plot(heptree, col=8)
text(heptree, digits=2)

## cross-validation to prune the tree
set.seed(2)
cvhep <- cv.tree(heptree, K=10)
cvhep$size
cvhep$dev
plot(cvhep, pch=21, bg=8, type="p", cex=1.5, ylim=c(400,750))

hepcut <- prune.tree(heptree, best=6)
hepcut
summary(hepcut)
plot(hepcut, col=8)
text(hepcut)
```

Example 5: Predicting the Outcome of Labor Negotiations Using a Classification Tree

```
library(tree)

## read the data
labor <- read.csv("C:/DataMining/Data/labor.csv")
labor[1:3,]
## omit variables with lots of missing values
ll=labor[,c(-3:-5,-7:-11,-13:-16)]
ll[1:3,]
levels(ll[,4])  ## vacation benefits
levels(ll[,5])  ## response: overall contract quality

## constructing the classification tree
labortree <- tree(Class ~., data = ll)
labortree
summary(labortree)
plot(labortree, col=8)
text(labortree, digits=2)
p1=snip.tree(labortree,nodes=2)
p1
summary(p1)
plot(p1)
text(p1)
```

Example 6: Diabetes among Pima Indians

```
## read the data and create plots
PimaIndians <- read.csv("C:/DataMining/Data/PimaIndians.csv")
PimaIndians
plot(PimaIndians)
PI=data.frame(PimaIndians)

## logistic regression model
## mm1: model fitted to all data
mm1=glm(Class~.,family=binomial,data=PI)
mm1
summary(mm1)

## simplifying the model through backward elimination
RPI=PI[,-4] ## dropping triceps skin fold thickness
mm1=glm(Class~.,family=binomial,data=RPI)
mm1
summary(mm1)
RPI=RPI[,-7]  ## dropping age
mm1=glm(Class~.,family=binomial,data=RPI)
mm1
summary(mm1)
RPI=RPI[,-4]  ## dropping serum insulin
RPI[1:3,]
mm1=glm(Class~.,family=binomial,data=RPI)
mm1
summary(mm1)

## evaluation of the full model
## split the data set into a training (50%) and a test (evaluation) set (50%)
set.seed(1)
n=length(PI$Class)
n
n1=floor(n*(0.5))
n1
n2=n-n1
n2
train=sample(1:n,n1)

PI1=data.frame(PI[train,])
PI2=data.frame(PI[-train,])

## mm2: model fitted on the training data set
mm2=glm(Class~.,family=binomial,data=PI1)
mm2
summary(mm2)

## create predictions for the test (evaluation) data set
gg=predict(mm2,newdata=PI2,type= "response")
gg
hist(gg)
plot(PI$Class[-train]~gg)

## coding as 1 if probability 0.5 or larger
gg1=floor(gg+0.5)
```

```
ttt=table(PI$Class[-train],gg1)
ttt
error=(ttt[1,2]+ttt[2,1])/n2
error

## evaluation of the simplified model
## mm2: model fitted on the training data set
mm2=glm(Class~NuPregnancy+Glucose+DiastolicBP+BodyMassIndex+DiabetesPedigree,f
amily=binomial,data=PI1)
mm2
summary(mm2)

## create predictions for the test (evaluation) data set
gg=predict(mm2,newdata=PI2,type= "response")
gg
hist(gg)
plot(PI$Class[-train]~gg)

## coding as 1 if probability 0.5 or larger
gg1=floor(gg+0.5)
ttt=table(PI$Class[-train],gg1)
ttt
error=(ttt[1,2]+ttt[2,1])/n2
error

## read the data
PimaIndians <- read.csv("C:/DataMining/Data/PimaIndians.csv")
PimaIndians

## CART analysis
library(tree)
PimaIndians$Class=factor(PimaIndians$Class)
## constructing the classification tree
PItree <- tree(Class ~., data = PimaIndians,mindev=0.01)
PItree
summary(PItree)
plot(PItree, col=8)
text(PItree, digits=2)

## cross-validation to prune the tree
set.seed(2)
cvPI <- cv.tree(PItree, K=10)
cvPI$size
cvPI$dev
plot(cvPI, pch=21, bg=8, type="p", cex=1.5, ylim=c(700,1000))

PIcut <- prune.tree(PItree, best=7)
PIcut
summary(PIcut)
plot(PIcut, col=8)
text(PIcut)

P1=snip.tree(PIcut,nodes=c(2,7))
P1
summary(P1)
plot(P1)
```

```
text(P1)
```

Example 7: Predicting the CPU Performance with Regression and Regression Trees

```
## read the data and create a matrix plot
cpu <- read.csv("C:/DataMining/Data/cpu.csv")
cpu
xx=cpu[,c(-1,-9)]
xx[1:3,]
plot(xx)

## regression model
regfit=lm(PRP~.,data=xx)
regfit
summary(regfit)

## cross-validation (leave one out): regression model on all six regressors
n=length(cpu$PRP)
diff=dim(n)
percdiff=dim(n)
for (k in 1:n) {
train1=c(1:n)
train=train1[train1!=k]
m1=lm(PRP~.,data=xx[train,])
pred=predict(m1,newdat=xx[-train,])
obs=xx[-train,7]
diff[k]=obs-pred
percdiff[k]=abs(diff[k])/obs
}
me=mean(diff)
rmse=sqrt(mean(diff**2))
mape=100*(mean(percdiff))
me   # mean error
rmse # root mean square error
mape # mean absolute percent error


library(tree)
## Construct the regression tree
cputree <- tree(PRP ~., data=xx, mindev=0.1, mincut=1)
cputree <- tree(PRP ~., data= xx, mincut=1)
cputree
summary(cputree)
plot(cputree, col=8)
text(cputree, digits=2)

## Use cross-validation to prune the regression tree
set.seed(2)
cvcpu <- cv.tree(cputree, K=10)
cvcpu$size
cvcpu$dev
plot(cvcpu, pch=21, bg=8, type="p", cex=1.5, ylim=c(0,6000000))

cpucut <- prune.tree(cputree, best=7)
cpucut
```

```
summary(cpucut)
plot(cpucut, col=8)
text(cpucut)
```

Example 8: Inferring the Cultivar of Wine Using Classification Trees, Discriminant Analysis and Multinomial Logistic Regression

```
## read the data and plots
wine <- read.csv("C:/DataMining/Data/wine.csv")
wine[1:3,]
plot(wine)

## CART
library(tree)
wine$Class=factor(wine$Class)
## constructing the classification tree
Winetree <- tree(Class ~., data = wine)
Winetree
summary(Winetree)
plot(Winetree, col=8)
text(Winetree, digits=2)
## cross-validation to prune the tree
set.seed(1)
cvWine <- cv.tree(Winetree, K=10)
cvWine$size
cvWine$dev
plot(cvWine, pch=21, bg=8, type="p", cex=1.5, ylim=c(100,400))
Winecut <- prune.tree(Winetree, best=4)
Winecut
summary(Winecut)
plot(Winecut, col=8)
text(Winecut)

## Clustering
## standardizing the attributes as units considerably different
wines=matrix(nrow=length(wine[,1]),ncol=length(wine[1,]))
for (j in 2:14) {
wines[,j]=(wine[,j]-mean(wine[,j]))/sd(wine[,j])
}
wines[,1]=wine[,1]
winesr=wines[,-1]
winesr[1:3,]
## kmeans clustering with 13 standardized attributes
grpwines <- kmeans(winesr, centers=3, nstart=20)
grpwines
grpwines$cluster  ## displaying clustering results
wine$Class        ## actual classes
## 6 mistakes made among 178 wines


## Discriminant analysis (linear/quadratic)
library(MASS)
## linear discriminant analysis using the standardized attributes
wines[1:3,]
ws=data.frame(wines)
ws[1:3,]
zlin=lda(X1~.,ws,prior=c(1,1,1)/3)
zlin
## quadratic discriminant analysis
```

```
zqua=qda(X1~.,ws,prior=c(1,1,1)/3)
zqua
n=dim(ws)[1]
errorlin=1-(sum(ws$X1==predict(zlin,ws)$class)/n)
errorlin
errorqua=1-(sum(ws$X1==predict(zqua,ws)$class)/n)
errorqua
neval=1
corlin=dim(n)
corqua=dim(n)
## leave one out evaluation
for (k in 1:n) {
train1=c(1:n)
train=train1[train1!=k]
## linear discriminant analysis
zlin=lda(X1~.,ws[train,],prior=c(1,1,1)/3)
corlin[k]=ws$X1[-train]==predict(zlin,ws[-train,])$class
## quadratic discriminant analysis
zqua=qda(X1~.,ws[train,],prior=c(1,1,1)/3)
corqua[k]=ws$X1[-train]==predict(zqua,ws[-train,])$class
}
merrlin=1-mean(corlin)
merrlin
merrqua=1-mean(corqua)
merrqua

## Multinomial logistic regression
## using VGAM
library(VGAM)
ws=data.frame(wines)
gg <- vglm(X1 ~ .,multinomial,data=ws)
summary(gg)
predict(gg) ## log-odds relative to last group
round(fitted(gg),2)  ## probabilities
cbind(round(fitted(gg),2),ws$X1)
## perfect classification
```