Modern Applied Statistics with S-Plus

Benny Yakir

Abstract

These notes are based on the book *Modern Applied Statistics with S-Plus* by W.N. Venables and B.D. Ripley. No originality is claimed.

********** **********

Contents

1 Starting R

1.1 Installing R

Basic information on R can be found at the URL

http://stat.auckland.ac.nz/r/r.html

Information on how to download and install R can be found in the file:

installing-R.doc

1.2 Basic R Commands

`fix(f.name)`	edits the function *f.name* in an emacs window.
`ls()`	list the files in .Data.
`rm(filename)`	delete the file *filename*.
`q()`	quit R.

When the program is ready for your commands you will get the ">" prompt. When you push Enter, but the command is not completed yet, you will get the "+" prompt.

You can change the working directory using File → Change dir. The current environment (image) using File → Save Image. You can then restore the image with File → Load Image.

To get help either use Help → R language (standard), and then type the name of the comand/function you like to get more information on. Alternatively, you can use Help → R language (html), which uses an internet-style browser,

1.3 An Example of an R Session

```
> bpiq <- read.table("bpiq.dat", col.names=c("dep","iq","bp"))
> hist(bpiq[,"iq"],xlab="IQ",main="IQ: all children")
> iq.d <- bpiq[bpiq[,"dep"]=="d","iq"]
> iq.nd <- bpiq[bpiq[,"dep"]=="nd","iq"]
> hist(iq.d,xlab="IQ", main="IQ: d children")
```

```
> hist(iq.nd,xlab="IQ", main="IQ: nd children")
> t.test(iq.d,iq.nd)

         Welch Two Sample t-test

data:  iq.d and iq.nd t = -1.6388, df = 15.491, p-value = 0.1214
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -26.922823   3.481156
sample estimates: mean of x mean of y
 101.0667  112.7875

> t.test(iq.d,iq.nd, var.equal=T)

         Two Sample t-test

data:  iq.d and iq.nd t = -2.4801, df = 93, p-value = 0.01494
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -21.105819  -2.335847
sample estimates: mean of x mean of y
 101.0667  112.7875

> q()
```

2 Objects in R

2.1 Vectors and matrices

Objects are identified in S-Plus by their attributes.

```
> mydata <- c(2.9, 3.4, 3.4, 3.7)
> colour <- c("red", "green", "blue")
> x1 <- 25:30
> length(mydata)
[1] 4
> mode(mydata)
[1] "numeric"
> mode(x1)
[1] "numeric"
> colour[2:3]
[1] "green" "blue"
> x1[-1]
[1] 26 27 28 29 30
> colour != "green"
[1]  TRUE FALSE  TRUE
> colour[colour != "green"]
[1] "red"  "blue"
> names(mydata) <- c('a', 'b', 'c', 'd')
> mydata
   a   b   c   d
 2.9 3.4 3.4 3.7
> letters
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o"
"p" "q" "r" [19] "s" "t" "u" "v" "w" "x" "y" "z"
> mydata[letters[1:2]]
   a   b
 2.9 3.4
> matrix(x1, 2, 3)
     [,1] [,2] [,3]
[1,]   25   27   29
[2,]   26   28   30
> matrix(x1, 2, 3, byrow=T)
     [,1] [,2] [,3]
```

```
[1,]   25   26   27
[2,]   28   29   30
> x2 <- matrix(x1, 2, 3)
> x2[1,1]
[1] 25
> x1 * 3
[1] 75 78 81 84 87 90
> x1 + x1
[1] 50 52 54 56 58 60
> x1 + x2
     [,1] [,2] [,3]
[1,]   50   54   58
[2,]   52   56   60
> rbind(x2, x2)
     [,1] [,2] [,3]
[1,]   25   27   29
[2,]   26   28   30
[3,]   25   27   29
[4,]   26   28   30
```

2.2 Lists

Collections of other S-Plus objects.

```
> alist <- list(c(0,1,2), 1:10, "Name")
> alist
[[1]]:
[1] 0 1 2

[[2]]:
 [1]  1  2  3  4  5  6  7  8  9 10

[[3]]:
[1] "Name"

> blist <- list(x=matrix(1:10,ncol=2), y=c("A", "B"), z=alist)
> blist$x
     [,1] [,2]
[1,]    1    6
```

```
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10
> blist$z[[3]]
[1] "Name"
```

2.3 Factors

Special type of vectors. Hold categorical variables.

```
> height <- factor(c("H","L","M","L","H"), levels=c("L","M","H","VH"))
> height
[1] H L M L H
Levels:  L M H VH
> height <- ordered(c("H","L","M","L","H"), levels=c("L","M","H","VH"))
> height
[1] H L M L H
Levels:  L < M < H < VH
```

2.4 Data frames

Used to store Data matrix. List of variables of the same length.

```
> bpiq[1:5,c(2,3)]
   iq bp
1 103  4
2 124 12
3 124  9
4 104  3
5  96  3
```

2.5 Homework

1. Turn the file *bpiq.dat* into an R data frame with the function *read.table*.
2. Read the help files on the functions hist and t.test.
3. Produce an histogram with a different number of bars.

4. What would happen if we would choose the argument probability=T in the function hist?
5. Perform a one-side t.test, for $\alpha = 0.1, 0.05, 0.01$.
6. Test ($\alpha = 0.05$) that the mean of iq.d is 100. Repeat this analysis for iq.nd.

3 Functions

There are many built-in functions in S-Plus. One can write his/her own functions easily. The structure of a function is:

function(arguments)
expressions

3.1 Built-in functions

```
> data.ex <- matrix(1:50, ncol=5)
> sum(data.ex)
[1] 1275
> mean(data.ex)
[1] 25.5
> var(data.ex)
         [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 9.166667 9.166667 9.166667 9.166667 9.166667
[2,] 9.166667 9.166667 9.166667 9.166667 9.166667
[3,] 9.166667 9.166667 9.166667 9.166667 9.166667
[4,] 9.166667 9.166667 9.166667 9.166667 9.166667
[5,] 9.166667 9.166667 9.166667 9.166667 9.166667
> cor(data.ex)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    1    1    1    1
[2,]    1    1    1    1    1
[3,]    1    1    1    1    1
[4,]    1    1    1    1    1
[5,]    1    1    1    1    1
> apply(data.ex,2,mean)
[1]  5.5 15.5 25.5 35.5 45.5
> apply(data.ex,1,sum)
 [1] 105 110 115 120 125 130 135 140 145 150
> max(data.ex)
[1] 50
> min(data.ex)
[1] 1
> range(data.ex)
```

```
[1]  1 50
```

3.2 Writing functions

```
> sumsq <- function(){}
> fix(sumsq)
(In the editor window write:)
function(x) {
        ssq <- sum(x * x)
        scb <- sum(x * x * x)
        result <- list(ssq = ssq, scb = scb)
        result
}
(Exit and save.)
> sumsq(1:10)
$ssq:
[1] 385

$scb:
[1] 3025

> grid.cal <- function(){}
> fix(grid.cal)

function(x, y) {
        grid <- matrix(0, length(x), length(y))
        for(i in 1:length(x)) {
                for(j in 1:length(y))
                        grid[i, j] <- sqrt(x[i]^2 + y[j]^2)
        }
        grid
}

> grid.cal(1:3,1:4)
         [,1]     [,2]     [,3]     [,4]
[1,] 1.414214 2.236068 3.162278 4.123106
[2,] 2.236068 2.828427 3.605551 4.472136
[3,] 3.162278 3.605551 4.242641 5.000000
> outer(1:3,1:4,"+")
```

```
     [,1] [,2] [,3] [,4]
[1,]    2    3    4    5
[2,]    3    4    5    6
[3,]    4    5    6    7
> h.dist <- function(x,y) sqrt(x^2 + y^2)
> outer(1:3,1:4,h.dist)
         [,1]     [,2]     [,3]     [,4]
[1,] 1.414214 2.236068 3.162278 4.123106
[2,] 2.236068 2.828427 3.605551 4.472136
[3,] 3.162278 3.605551 4.242641 5.000000
> system.time(grid.cal(1:100,1:100))
[1]   NA   NA 2.46   NA   NA
> system.time(grid.cal(1:500,1:500))
[1]    NA    NA 61.41    NA    NA
>  system.time(outer(1:100,1:100,h.dist))
[1]   NA   NA 0.04   NA   NA
>  system.time(outer(1:500,1:500,h.dist))
Error: heap memory (6144 Kb) exhausted [needed 1953 Kb more]
       See "help(Memory)" on how to increase the heap size.
Timing stopped at: NA NA 0.53 NA NA

> sumpow <- function(){}
> fix(sumpow)

function(x, pow = 2:3)
{
        l <- length(pow)
        result <- vector("list", length = l)
        for(i in 1:l)
                result[[i]] <- sum(x^pow[i])
        result
}

> sumpow(1:10)
[[1]]:
[1] 385

[[2]]:
[1] 3025
```

```
> sumpow(1:3,c(3,5,6))
[[1]]:
[1] 36

[[2]]:
[1] 276

[[3]]:
[1] 794
```

3.3 Plotting functions

The R environment provides comprehensive graphical facilities: Many higher-level plotting functions which are controlled by a verity of parameters and lower-level plotting functions which can be used in order to add to existing plots.

```
> x <- rnorm(10)
> y <- rnorm(10)
> l <- paste("(x",1:10,",y",1:10,")", sep="")
> plot(x,y)
> text(x,y,l)
> title("Ten points")
```

3.4 Home Work

1. Write a function that calculates central moments of a vector.
2. Read the help file on the function tapply.
3. Use the function tapply to calculate the first 4 central moments of the nd and d children iq and bp.
4. Write a function that returns the indices of elements of a vector which are more then 2 std from the vector mean.
5. Identify the children which are more than 2 std from the mean
   (a) in the full list.
   (b) within each subgroup.
6. Write a function that produces a *bubble plot*, i.e. given 3 vectors x, y and r it plots circles of radius r, centered at (x, y). (Use the function *symbols*.) Generate 3 random vectors of length 20 and try the function.

4 Linear models

4.1 A simple regression example

Linear models are the core of classical statistics: Regression, ANOVA, Analysis of covariance and much more. R'S basic fitting function is *lm* (and *aov* for ANOVA). Here we consider a simple case of linear regression.

```
> trees.dat <- read.table("trees.dat", col.names=c("Diameter",
+ "Height", "Volume"))
> attach(trees.dat)
> trees.fit <- lm(Volume ~ Diameter + Height)
> summary(trees.fit)

Call: lm(formula = Volume ~ Diameter + Height)

Residuals:
    Min      1Q  Median      3Q     Max
-6.4065 -2.6493 -0.2876  2.2003  8.4847

Coefficients:
            Estimate Std. Error t value Pr(>t)
(Intercept) -57.9877     8.6382  -6.713 2.75e-07 *** Diameter
4.7082     0.2643  17.816  < 2e-16 *** Height        0.3393 0.1302
2.607   0.0145 *
---
Signif. codes:  0  '***'  0.001  '**'  0.01  '*'  0.05  '.'
0.1' '1

Residual standard error: 3.882 on 28 degrees of freedom Multiple
R-Squared: 0.948,      Adjusted R-squared: 0.9442 F-statistic: 255
on 2 and 28 degrees of freedom,      p-value:     0
```

4.2 Model formulae

The basic formula for a linear model is:

$$Y = \beta' X + \epsilon,$$

where Y is a random vector (or matrix), X is a design matrix of known constants, β is an unknown vector of parameters and ϵ is a random vector.

For example, for multiple regression with three explanatory variables the model

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon$$

is specified by the formula

$$y \sim x1 + x2 + x3$$

Note that

- The left hand side is a vector or a matrix.
- The intercept is implicit.
- In the right hand side vectors or matrices, separated by $+$.
- To remove intercept use $y \sim -1 + x1 + x2 + x3$.

Factors generate columns in X to allow separate parameters for each level. Hence, if $a1$ and $a2$ are two factors then

$$y \sim a1 + x1$$

is a model for parallel regression (analysis of covariance) and

$$y \sim a1 + a2$$

is a two-way ANOVA, with no interaction. To get the model with interaction use

$$y \sim a1 + a2 + a1 : a2,$$

or

$$y \sim a1 * a2.$$

The nested model

$$y = \beta_0 + \alpha_{a_1} + \beta_{a_1} x_1 + \epsilon$$

is formulated by $y \sim a1 + a1 : x1$, whereas the formula $y \sim a1 * x1$ corresponds to

$$y = \beta_0 + \alpha_{a_1} + \beta_1 x_1 + \beta_{a_1} x_1 + \epsilon$$

Terms like $a1 * a2 * a3$ would generate the model

$$a1 + a2 + a3 + a1 : a2 + a1 : a3 + a2 : a3 + a1 : a2 : a3,$$

and terms like $(a1 + a2 + a3)\hat{}2$ correspond to

$$a1 + a2 + a3 + a1 : a2 + a1 : a3 + a2 : a3.$$

To allow operators to be used with their arithmetic meaning use the $I(\cdot)$ function. For example

```
profit ^ I(dollar.inc + 1.55*pound.inc)
```

To fit polynomials use the function *poly*, hence

```
y ~ poly(x1, x2, 3)
```

corresponds to orthogonal polynomials of degree 3 in $x1$ and $x2$.

4.3 Diagnostics

The basic tool for examining the fit is the residuals. They are not independent.

$$\text{var}(\mathbf{e}) = \sigma^2[I - H],$$

where $H = X(X'X)^{-1}X'$ is the *hat matrix* and $\mathbf{e}$ is the vector of residuals. If the leverage h_{ii} of the observation is large (more than 3 or 2 times p/n, where p is the number of explanatory variables and n the number of observations) than the observation is influential.

The standardized residuals

$$e' = \frac{e}{s\sqrt{1-h}}$$

or the jackknifed residuals

$$e^* = \frac{y - \hat{y}_{(\cdot)}}{\text{var}(y - \hat{y}_{(\cdot)})} = e' \Big/ \left[\frac{n - p - e'^2}{n - p - 1}\right]^{1/2}$$

are used to examine normality and identify outlayers with the normal or half-normal probability plot. The residual plot is used to identify non-random patterns.

```
> trees.res <- residuals(trees.fit)
> trees.prd <- predict(trees.fit)
> s <- summary(trees.fit)$sigma
> h <- lm.influence(trees.fit)$hat
> trees.res <- trees.res/(s*sqrt(1-h))
```

```
> par(mfrow=c(2,1))
> plot(trees.dat[,"Diameter"],trees.res, xlab="Diameter",
+ ylab="Std. Residuals")
> abline(h=0, lty=2)
> title("Std. Residuals vs. Diameter")
> plot(trees.dat[,"Height"],trees.res, xlab="Height",
+ ylab="Std. Residuals")
> abline(h=0, lty=2)
> title("Std. Residuals vs. Height")
> par(mfrow=c(1,1))
> plot(trees.prd, trees.res, xlab="Predicted Volume",
+ ylab="Std. Residuals")
> abline(h=0, lty=2)
> title("Std. Residuals vs. Fitted Values")
> par(pty="s")
> qqnorm(trees.res, ylab="Std. Residuals")
> title("Normal plot of Std. Residuals")
> plot(1:31, h, type="n", xlab="index",
+ ylab="Diagonal of hat matrix")
> abline(h=mean(h))
> segments(1:31,h,1:31,mean(h))
> title("Leverage measure")
```

4.4 Model selection

Statistical hypothesis testing is all about comparing between models.

```
> trees1.fit <- lm(Volume ~ Diameter + I(Diameter^2) + Height)
> anova(trees1.fit,trees.fit)
Analysis of Variance Table

Model 1: Volume ~ Diameter + I(Diameter^2) + Height Model 2:
Volume ~ Diameter + Height
  Res.Df Res.Sum Sq Df  Sum Sq F value   Pr(>F)
1     27     186.01 2     28     421.92 -1 -235.91  34.243
3.13e-06 ***
---
Signif. codes:  0  ‘***’  0.001  ‘**’  0.01  ‘*’  0.05  ‘.’  0.1 ‘
’  1
```

A disadvantage of this approach that it does not take into account the complexity of the models. In principle, one prefers simpler models — i.e. models with less parameters. Hence, one may want to consider introducing a penalty which grows with the number of parameters which are used for the fit. This is known as *Information Criteria*. Examples include:

ACI: $2[\max_{\theta\in\mathcal{K}} \ell(\theta) - \max_{\theta\in\mathcal{H}} \ell(\theta)] + 2[\dim(\mathcal{K}) - \dim(\mathcal{H})]$.

BCI: $2[\max_{\theta\in\mathcal{K}} \ell(\theta) - \max_{\theta\in\mathcal{H}} \ell(\theta)] + \log n[\dim(\mathcal{K}) - \dim(\mathcal{H})]$.

C_p**:** $\hat{\sigma}^{-2}$[Sum of Squares Diff.] $+ 2[\dim(\mathcal{K}) - \dim(\mathcal{H})]$.

Given a nested sequence $\mathcal{H} \subset \mathcal{K}_1 \subset \mathcal{K}_2 \subset \cdots \subset \mathcal{K}_J$ the model with the smallest information criteria should be preferred

4.5 Homework

1. The data frame *cars.dat* gives the speed of cars and the distances taken to stop. Note that the data were recorded in the 1920s. It contains 50 observations on 2 variable:

   `speed:` (numeric) Speed (mph).

   `dist:` (numeric) Stopping distance (ft).

   Analyze the relation between these two variables.

2. The data frame *cereales.dat* gives the calories, fat and fiber content of different types of cereals. It contains 77 observations on 3 variable:

   `cal:` (numeric) Calories.

   `fat:` (numeric) Total fat.

   `fiber:` (factor) Fibers.

   Analyze the calories as a function of total fat and the fiber content.

5 Generalized Linear Models (GLM)

5.1 The basic model

GLM extends linear models to accommodate both non-normal response and transformations to linearity. Assumptions:

- The distribution of Y_i is of the form:

$$f_{\theta_i}(y_i) = \exp[A_i\{y_i\theta_i - \gamma(\theta_i)\}/\psi + \tau(y_i, \psi/A_i)],$$

  where ψ is a (nuisance) scale parameter, A_i known weights. The distribution is controlled by the θ_is

- The mean of Y, μ, is a function of a linear combination of the predictors:

$$\mu = m(\beta' x) = l^{-1}(\beta' x).$$

  l is called the *link* function. If $l = (\gamma')^{-1}$ then $\theta = \beta' x$ and l is called the *canonical link function.*

  Examples:

Normal: $\log f(y) = \{y\mu - \mu^2/2\}/\sigma^2 - \frac{1}{2}\{y^2/\sigma^2 + \log[2\pi\sigma^2]\}$.

$$\begin{aligned} \theta &= \mu \\ \gamma(\theta) &= \theta^2/2 \\ (\mu) &= \mu \\ l\psi &= \sigma^2. \end{aligned}$$

Poisson: $\log f(y) = y \log \mu - \mu - \log(y!)$.

$$\begin{aligned} \theta &= \log \mu \\ \gamma(\theta) &= e^\theta \\ \mu &= e^\theta \\ l(\mu) &= \log(\mu) \\ \psi &= 1. \end{aligned}$$

Binomial: $Y = k/n$. $\log f(y) = n\left[y \log \frac{p}{1-p} + \log(1-p)\right] + \log \binom{n}{ny}$.

$$\begin{aligned} \gamma(\theta) &= \log\left(1+e^{\theta}\right) \\ \mu &= e^{\theta}/\left(1+e^{\theta}\right) \\ l(\mu) &= \log(\mu/(1-\mu)) \\ \psi &= 1 \\ A &= n. \end{aligned}$$

The parameters are estimated using MLE. The maximum is calculated using iterative regression.

5.2 Analysis of deviance

The parameters in the saturated model S are not constrained. The mean of the ith observation is estimated by the observation itself. The deviance of the model M, $M \subset S$, is:

$$D_M = 2\sum_{i=1}^{n} A_i \left[\left\{y_i\hat{\theta}_S - \gamma(\hat{\theta}_S)\right\} - \left\{y_i\hat{\theta}_M - \gamma(\hat{\theta}_M)\right\}\right].$$

This is the unscaled generalized log-likelihood-ratio statistic.

In the Gaussian family

$$D_M/\psi \sim \chi^2_{n-p},$$

hence $\hat{\psi} = D_M/(n-p)$ is unbiased. In some other cases these relations may be approximately true or not true at all.

When ψ is known, testing the fit of the model M relative to the null model M_0, where $M_0 \subset M$, is performed with chi-square test of $(D_{M_0} - D_M)/\psi$. When ψ is unknown, testing is performed with a F test of $(D_{M_0} - D_M)/(\hat{\psi}(p-q))$.

An alternative approach is to apply the AIC criteria, where

$$AIC = D_M + 2p\hat{\psi}$$

($\hat{\psi}$ should be the same across all models.)

5.3 Fitting the model

The basic fitting function is *glm*, for which the basic arguments are

```
glm(formula, family, data, weights, control)
```

formula: of the linear predictor.

family: gives the family name with additional information. For example:

```
function = binomial(link=probit)
```

fits a binomial response with the probit link.

control: of the iterative process. For example *maxit* determines the maximal number of iterations.

5.4 Generic functions and method functions

R functions are designed to be as general as possible. The function *summary*, for example, can be used on many objects. This is achieved by a construction which includes a generic function which is associated with method functions. A method function performs the appropriate operation on an object of a specific class. For example, *summary.glm* produces a summary of objects of class *glm*. Each object has among its attributes a vector *class*, which is used by the generic function in order to call the appropriate method function.

Generic functions with methods for *glm* include *coef, resid, print, summary, anova, predict, deviance*.

5.5 A small Binomial example

```
> options(contrast=c("contr.treatment", "contr.poly"))
> ldose <- rep(0:5, 2)
> numdead <- c(1,4,9,13,18,20,0,2,6,10,12,16)
> sex <- factor(rep(c("M","F"), c(6,6)))
> SF <- cbind(numdead, numalive=20-numdead)
> budworm.fit <- glm(SF ~ sex*ldose, family=binomial)
> summary(budworm.fit)
```

```
Call: glm(formula = SF ~ sex * ldose, family = binomial)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.39849  -0.32094  -0.07592   0.38220   1.10375

Coefficients:
            Estimate Std. Error z value Pr(>z)
(Intercept)  -2.9935     0.5525  -5.418 6.02e-08 *** sex
0.1750     0.7781   0.225    0.822 ldose         0.9060     0.1671
5.424 5.84e-08 *** sex.ldose     0.3529     0.2699   1.307
0.191
---
Signif. codes: 0 ‘***’ 0.001  ‘**’ 0.01  ‘*’  0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 124.8756  on 11  degrees of freedom
Residual deviance:   4.9937  on  8  degrees of freedom AIC: 43.104

Number of Fisher Scoring iterations: 3

> plot(c(1,32), c(0,1), type="n", xlab="dose", ylab="prob", log="x")
> text(2^ldose, numdead/20, as.character(sex))
> ld <-seq(0, 5, 0.1)
> lines(2^ld, predict(budworm.fit, data.frame(ldose=ld,
+ sex=factor(rep("M", length(ld)), levels=levels(sex))),
+ type="response"))
> lines(2^ld, predict(budworm.fit, data.frame(ldose=ld,
+ sex=factor(rep("F", length(ld)), levels=levels(sex))),
+ type="response"))
> anova(update(budworm.fit, . ~ sex + ldose + factor(ldose)), test="Chisq")
Analysis of Deviance Table

Model: binomial, link: logit

Response: SF

Terms added sequentially (first to last)
```

```
              Df Deviance Resid. Df Resid. Dev P(>Chi)
NULL                             11    124.876
sex            1    6.077        10    118.799    0.014
ldose          1  112.042         9      6.757    0.000
factor(ldose)  4    1.744         5      5.013    0.783
```

5.6 Fitting other families

The families of distribution available for *glm* are the *gaussian, binomial, poisson, inverse.gaussian* and *gamma*. Other families can be defined.

The function `make.family` can be used with arguments:

name: The name of the family.

link: A list with the link function, its inverse, and its derivative.

variance: A list with the variance and deviance functions.

5.7 Frames

An *evaluation frame* is a list that associates names with values. The frame of the session is frame 0. Any expression evaluated in the interactive level is evaluated in frame 1. Frames are added as evaluations become more complex. A name is searched for in the local frame. If it is not there the search moves to frame 1, then to frame 0 and then through the search path.

5.8 A Poisson example

```
> quine.dat <- read.table("quine.dat",header=T)
> attach(quine.dat)
> glm(Days ~.^4, family=poisson, data = quine.dat)

Call:  glm(formula = Days ~ .^4, family = poisson, data = quine.dat)

Coefficients:
      (Intercept)               Eth               Sex              AgeF1
           3.0564           -0.1386           -0.4914           -0.6227
```

```
      AgeF2       AgeF3        Lrn    Eth.Sex
    -2.3632     -0.3784    -1.9577    -0.7524
Eth.AgeF1   Eth.AgeF2   Eth.AgeF3    Eth.Lrn
     0.1029     -0.5546     0.0633     2.2588
Sex.AgeF1   Sex.AgeF2   Sex.AgeF3    Sex.Lrn
     0.4092      3.1098     1.1145     1.5900
AgeF1.Lrn   AgeF2.Lrn   AgeF3.Lrn Eth.Sex.AgeF1
     2.6421      4.8585         NA    -0.3105
Eth.Sex.AgeF2 Eth.Sex.AgeF3  Eth.Sex.Lrn Eth.AgeF1.Lrn
       0.3469        0.8329     -0.1639     -3.5493
Eth.AgeF2.Lrn Eth.AgeF3.Lrn Sex.AgeF1.Lrn Sex.AgeF2.Lrn
      -3.3315            NA     -2.4285     -4.1914
Sex.AgeF3.Lrn Eth.Sex.AgeF1.Lrn Eth.Sex.AgeF2.Lrn Eth.Sex.AgeF3.Lrn
           NA            2.1711           2.1029                NA

Degrees of Freedom: 145 Total (i.e. Null);  118 Residual
Null Deviance:      2074
Residual Deviance: 1174         AIC: 1818
```

```
> days.mean <- tapply(Days, list(Eth, Sex, Age,Lrn), mean)
> days.var <- tapply(Days, list(Eth, Sex, Age,Lrn), var)
> days.std <- sqrt(days.var)
> plot(mays.mean, days.std)
```

5.9 Home Work

1. Use the **rep** and **seq** functions to produce the vectors:
   1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
   4 4 4 4 3 3 3 3 2 2 2 2 1 1 1 1
   1 2 2 3 3 3 4 4 4 4 5 5 5 5 5

2. Re-analyze the*budworm* data. Replace ldose by ldose-3 and see that sex is significant. What is the predicted death rate at dose=7 (i) if the sex is unknown? (ii) if sex="M"?

3. Write the density of negative-binomial family in the exponential-family form. What are θ, $\gamma(\theta)$, ψ, A, and the canonical link function?

4. The data set `quine.dat` contains the outcome of an observational study on number of days absent from school during one year in a town in Australia. The kids are classified by four factors:

`Eth:` Ethnic group. `A` = aboriginal, `N` = non-aboriginal.

`Sex:` `M`, `F`.

`Age:` `F0` = Primary, `F1` = first form, `F3` = second form, `F4` = third form.

`Lrn:` `AL` = average learner, `SL` = slow learner.

Investigate this data set using `glm`. Identify important factors and interactions.

6 Robust methods

6.1 Introduction

Robust methods are methods which will work under a wide spectrum of circumstances. Two of the main aspects of robustness are:

1. Resistance to outliers. Methods with high *breakdown point*. The estimates are not effected much even if the value of a large portion of the observation is given an arbitrary value.

2. Robustness to distributional assumptions. The method is still efficient even if the distributional assumptions are violated.

The usual least squares methods are neither resistant nor robust.

6.2 The `lqs` function for resistant regression

```
Usage
lqs(x, ...)
lqs.formula(formula, data = NULL, ...,
            method = c("lts", "lqs", "lms", "S", "model.frame"),
            subset, na.action = na.fail, model = TRUE,
            x = FALSE, y = FALSE, contrasts = NULL)
lqs.default(x, y, intercept, method = c("lts", "lqs", "lms", "S"),
            quantile, control = lqs.control(...), k0 = 1.548, seed, ...)
lmsreg(...)
ltsreg(...)
print.lqs(x, digits, ...)
residuals.lqs(x)

Arguments
formula  a formula of the form y ~ x1 + x2 + ...{}{}.
data  data frame from which variables specified in formula are
      preferentially to be taken.
subset  An index vector specifying the cases to be used in fitting.
        (NOTE: If given, this argument must be named exactly.)
na.action  A function to specify the action to be taken if NAs are
           found. The default action is for the procedure to fail.
```

```
           An alternative is na.omit, which leads to omission of cases
           with missing values on any required variable. (NOTE: If
           given, this argument must be named exactly.)
x  a matrix or data frame containing the explanatory variables.
y  the response: a vector of length the number of rows of x.
intercept  should the model include an intercept?
method  the method to be used. model.frame returns the model frame:
        for the others see the Details section. Using lmsreg or ltsreg
        forces "lms" and "lts" respectively.
quantile  the quantile to be used: see Details. This is over-ridden if
          method = "lms".
control  additional control items: see Details.
seed  the seed to be used for random sampling: see .Random.seed. The
      current value of .Random.seed will be preserved if it is set..
...  arguments to be passed to lqs.default or lqs.control.

Description
Fit a regression to the good points in the dataset, thereby achieving
a regression estimator with a high breakdown point. lmsreg and ltsreg
are compatibility wrappers.

Details
Suppose there are n data points and p regressors, including any intercept.
The first three methods minimize some function of the sorted squared
residuals. For methods "lqs" and "lms" is the quantile squared residual,
and for "lts" it is the sum of the quantile smallest squared residuals.
"lqs" and "lms" differ in the defaults for quantile, which are
floor((n+p+1)/2) and floor((n+1)/2) respectively. For "lts" the default
is 'floor(n/2) + floor((p+1)/2)'.

The "S" estimation method solves for the scale s such that the average of
a function chi of the residuals divided by s is equal to a given constant.

The control argument is a list with components item{psamp}{ the size of
each sample. Defaults to p. } item{nsamp}{ the number of samples or "best"
or "exact" or "sample". If "sample" the number chosen is min(5*p, 3000),
taken from Rousseeuw and Hubert (1997). If "best" exhaustive enumeration
is done up to 5000 samples: if "exact" exhaustive enumeration will be
attempted however many samples are needed. } item{adjust}{ should the
intercept be optimized for each sample? }
```

```
Value
An object of class "lqs".

NOTE
There seems no reason other than historical to use the lms and lqs options.
LMS estimation is of low efficiency (converging at rate n^{-1/3}) whereas
LTS has the same asymptotic efficiency as an M estimator with trimming at
the quartiles (Marazzi, 1993, p.201). LQS and LTS have the same maximal
breakdown value of (floor((n-p)/2) + 1)/n attained if
floor((n+p)/2) <= quantile <= floor((n+p+1)/2). The only drawback mentioned
of LTS is greater computation, as a sort was thought to be required
(Marazzi, 1993, p.201) but this is not true as a partial sort can be used
(and is used in this implementation).
Adjusting the intercept for each trial fit does need the residuals to be
sorted, and may be significant extra computation if n is large and p small.

Opinions differ over the choice of psamp. Rousseeuw and Hubert (1997) only
consider p; Marazzi (1993) recommends p+1 and suggests that more samples
are better than adjustment for a given computational limit.

The computations are exact for a model with just an intercept and
adjustment, and for LQS for a model with an intercept plus one regressor
and exhaustive search with adjustment. For all other cases the minimization
is only known to be approximate.
```

6.3 Some examples

Let us compare the two of the robust methods, `lms` and `lts` with the usual `lm`.

```
> .Random.seed <- 1:4
> library(lqs)
> x30 <- runif(30, 0.5, 4.5)
> e30 <- rnorm(30, 0, 0.2)
> y30 <- 2+ x30+ e30
> x20 <- runif(20, 5, 7)
> y20 <- rnorm(20, 2, 0.5)
> y <- c(y30, y20)
```

```
> x <- c(x30, x20)
> plot(x,y)
> abline(lm(y~x), lty=1)
> abline(lmsreg(x,y), lty=2)
> abline(ltsreg(x,y), lty=3)
> legend(4.5,6, legend=c("LS","LMS","LTS"), lty=1:3)
```

Figure 1: Plot of lm, lms and lts

Consider next the internal dataset *stackloss* obtained from 21 days of operation of a plant for the oxidation of ammonia (NH3) to nitric acid (HNO3). The nitric oxides produced are absorbed in a countercurrent absorption tower.

`x1:` represents the rate of operation of the plant.

`x2:` is the temperature of cooling water circulated through coils in the absorption tower.

`x3:` is the concentration of the acid circulating, minus 50, times 10: that is, 89 corresponds to 58.9 per cent acid.

`y:` (the dependent variable) is 10 times the percentage of the ingoing ammonia to the plant that escapes from the absorption column unabsorbed; that is, an (inverse) measure of the over-all efficiency of the plant.

```
> data(stackloss)
> stack.lm <- lm(stack.loss ~ stack.x)
> summary(stack.lm)

Call:
lm(formula = stack.loss ~ stack.x)

Residuals:
    Min      1Q  Median      3Q     Max
-7.2377 -1.7117 -0.4551  2.3614  5.6978

Coefficients:
                  Estimate Std. Error t value Pr(>t)
(Intercept)       -39.9197    11.8960  -3.356  0.00375 **
stack.xAir.Flow     0.7156     0.1349   5.307  5.8e-05 ***
stack.xWater.Temp   1.2953     0.3680   3.520  0.00263 **
stack.xAcid.Conc.  -0.1521     0.1563  -0.973  0.34405
---
Signif. codes:  0  ‘***’  0.001  ‘**’  0.01  ‘*’  0.05  ‘.’  0.1  ‘ ’  1

Residual standard error: 3.243 on 17 degrees of freedom
Multiple R-Squared: 0.9136,     Adjusted R-squared: 0.8983
F-statistic:  59.9 on 3 and 17 degrees of freedom,      p-value: 3.016e-009
```

```
> stack.lts <- ltsreg(stack.x, stack.loss)
> stack.lms <- lmsreg(stack.x, stack.loss)
> stack.lm$coef
  (Intercept)   stack.xAir.Flow stack.xWater.Temp stack.xAcid.Conc.
  -39.9196744         0.7156402         1.2952861        -0.1521225
> stack.lts$coef
  (Intercept)      Air.Flow    Water.Temp    Acid.Conc.
  -3.630556e+01  7.291667e-01  4.166667e-01  6.336445e-16
> stack.lms$coef
  (Intercept)      Air.Flow    Water.Temp    Acid.Conc.
  -3.425000e+01  7.142857e-01  3.571429e-01 -2.791275e-16
> par(mfrow=c(2,2))
> plot(fitted(stack.lm),resid(stack.lm))
> abline(h=0)
> title("LS regression")
> plot(fitted(stack.lts),resid(stack.lts))
> abline(h=0)
> title("LTS regression")
> plot(fitted(stack.lms),resid(stack.lms))
> abline(h=0)
> title("LMS regression")
```

6.4 Home work

Investigate the data set *hills.dat*. The 3 vriables in the data set are:

dist: The overall race distance.

climb: The total height climbed.

time: The record time.

Identify outliers using the robust methods presented in class. Try the model where observations are weighted by $1/\texttt{dist}^2$.

Figure 2: Plot of stackloss

7 Non-linear Regression

7.1 The basic model

The Basic model is:

$$Y = f(X, \beta) + \epsilon,$$

where Y is the response, f is a function of a known form, X are the covariates, β is a vector of unknown parameters and ϵ is the random noise. The structure of f is determined on scientific ground — modeling of the physical phenomenon.

The parameters are estimated by minimizing an objective function such as the least-squares:

$$\sum_{i=1}^{n} (Y_i - f(X_i, \beta))^2.$$

(When the ϵ_i are iid normal this is the MLE.) The model is fitted with the function `nls`. The arguments are:

`formula` The right hand side is a regular expression.

`data` An optional data frame.

`start` A list or vector specifying the starting values for parameters. The names of the components specify which of the components of the expression in `formula` are parameters.

`control` over the iterative procedure.

`algorithm` Which optimization algorithm is used.

`trace` Tracing information from the iterative algorithm.

7.2 A small example

```
> hormone.dat <- read.table("hormone.txt",col.names=c("f","b"))
> hormone.dat
       f    b
 1   84.6 12.1
 2   83.9 12.5
```

```
 3  148.2 17.2
 4  147.8 16.7
 5  463.9 28.3
 6  463.8 26.9
 7  964.1 37.6
 8  967.6 35.8
 9 1926.0 38.5
10 1900.0 39.9
> attach(hormone.dat)
> plot(f,b)
> title("Observed Values")
> hormone.fit <- nls(b ~ Bmax*f/(KD + f), start=list(Bmax=40,KD=250))
> summary(hormone.fit)

Formula: b ~ Bmax * f/(KD + f)

Parameters:
     Estimate Std. Error t value Pr(>t)
Bmax   44.378      1.129   39.31 1.93e-10 ***
KD    241.688     20.946   11.54 2.89e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.288 on 8 degrees of freedom

Correlation of Parameter Estimates:
     Bmax
KD 0.8281

> plot(f,b)
> Bmax.hat <- coef(hormone.fit)[1]
> KD.hat <- coef(hormone.fit)[2]
> f.seq <- seq(range(f)[1],range(f)[2],length=100)
> b.seq <- Bmax.hat*f.seq/(KD.hat + f.seq)
> lines(f.seq,b.seq)
> title("Observed Values and Fitted Curve")
> Bmax.val <- seq(20,60,length=30)
> KD.val <- seq(50,1000,length=30)
> S <- matrix(0,30,30)
> for(i in 1:30){
```

```
+ for(j in 1:30) S[i,j] <- sum((b-Bmax.val[i]*f/
+ (KD.val[j] + f))^2)}
> persp(Bmax.val,KD.val, S/100, xlab="Bmax",
+ ylab="KD", zlab="Objective Function/100")
```

A. B. C.

Figure 3: A=hormone data, B=predicted, C=objective function

7.3 Attributes

All objects in R have a `mode` and a `length` attributes. Some have other attributes like `names` or `dim`. Attributes are handled with the functions:

`attributes(x)` Return or change all attributes of the object `x`.

`attr(x, which)` Returns or changes the attribute `which` of the object `x`.

`dim, names ...` Returns or change specific attributes.

7.4 Prediction

The data set `wtloss.dat` contains information on a group of male patients that were involved in a weight rehabilitation programme. The variables are

`Days` the time since the start of the programme.

`Weight` in kilograms.

The model to fit is:

$$W = \beta_0 + \beta_1 2^{-t/\theta} + \epsilon.$$

where

β_0 is the stable lean weight.

β_1 is the total weight to be lost.

θ 'half life'; that time it takes to lose half of the weight.

Let $\eta(\beta_0, \beta_1, \theta) = \beta_0 + \beta_1 2^{-t/\theta}$. Then the gradient is

$$\frac{\partial \eta}{\partial \beta_1} = 1, \qquad \frac{\partial \eta}{\partial \beta_2} = 2^{-t/\theta}, \qquad \frac{\partial \eta}{\partial \theta} = \frac{\log(2)\beta_1 t 2^{-t/\theta}}{\theta^2}$$

A crucial question is how long should one expect to be on the programme in order to achieve a particular goal weight w_0. From the equation it follows that

$$t_0 = \tau(\beta_0, \beta_1, \theta) = -\theta \log_2\{(w_0 - \beta_0)/\beta_1\}.$$

Hence,

$$\hat{t}_0 = -\hat{\theta} \log_2\{(w_0 - \hat{\beta}_0)/\hat{\beta}_1\}.$$

Large sample (δ-method) approximation of the variance of $\hat{t}_0$ is given by

$$\mathrm{var}(\hat{t}_0) \approx (\dot{\tau})' \Sigma \dot{\tau},$$

where $\dot{\tau}$ is the gradient of τ and Σ is the variance-covariance matrix of the parameters estimates.

```
> wtloss.nls <- nls(Weight ~ b0 + b1*2^(-Days/th),
+ data = wtloss.dat, start=list(b0=90,b1=95,th=120),trace = T)
67.54349 :   90  95 120
40.18081 :   82.72629 101.30457 138.71374
39.24489 :   81.39868 102.65836 141.85860
39.2447 :   81.37375 102.68417 141.91052
> wtloss.eprn <- function(){}
> fix(wtloss.expn)

function(b0, b1, th, t) {
   temp <- 2^(-t/th)
```

```
   model.func <- b0 + b1 * temp
   Z <- cbind(1, temp, (b1 * t * temp * log(2))/th^2)
   dimnames(Z) <- list(NULL, c("b0", "b1", "th"))
   attr(model.func, "gradient") <- Z
   model.func
}

> wtloss1.nls <- nls(Weight ~ wtloss.expn(b0, b1, th, Days),
+ data = wtloss.dat, start=list(b0=90,b1=95,th=120),trace = T)
67.54349 :   90  95 120
40.18081 :   82.72629 101.30457 138.71374
39.24489 :   81.39868 102.65836 141.85859
39.2447 :   81.37375 102.68417 141.91053
> vcov <- function(fm){
+ sm <- summary(fm)
+ sm$cov.unscaled * sm$sigma^2}
> time.pds <- deriv(~ -th * log((w0 - b0)/b1)/log(2),
+ c("b0", "b1", "th"), function(w0, b0, b1, th) NULL)
> est.time <- function(){}
> fix(est.time)

function(w0, obj)
{
   b <- coef(obj)
   tmp <- time.pds(w0, b["b0"], b["b1"], b["th"])
   t0 <- as.vector(tmp)
   dot <- attr(tmp, "gradient")
   v <- (dot %*% vcov(obj) * dot) %*% matrix(1,3,1)
   a <- cbind( t0, sqrt(v))
   dimnames(a) <- list(paste(w0, "kg: "), c("t0", "SE"))
   a
}

> est.time(c(100, 90, 85), wtloss.nls)
               t0        SE
100 kg:  349.4979  8.175396
 90 kg:  507.0941 31.217871
 85 kg:  684.5185 98.900230
```

7.5 Profiles

Figure 4: Profile of wtloss

The large sample derivation is based on the assumption that the log-likelihood is (approximately) quadratic in the vicinity of the parameters true values.

This can be evaluated with the profile function. The profile of a parameter is the likelihood function of the given parameter, maximized with respect to all other parameters. R provides the function `profile` which compute the functions

$$\pi(\beta_j) = \frac{\text{sign}(\beta_j - \hat{\beta}_j)}{s_j}\sqrt{\text{PSS}(\beta_j) - \text{PSS}(\hat{\beta}_j)}.$$

These functions should look linear if the large sample assumptions are met.

```
> par(mfrow=c(2,2), pty="s")
> plot(profile(wtloss.nls), ask=F)
```

7.6 Home work

1. Fit the model to the data set `wtloss.dat` using the option `plinear`.
2. Fit to the data both quadratic and cubic polynomials. Plot the expected weight in the range of 0 to 700 days in the program as predicted by `nls`, the quadratic, and the cubic fit. Add to the plot the data points and a legend.

8 Non-parametric and semi-parametric Regression

8.1 Non-parametric smoothing

When the data is definitely non-linear on the one hand, and, on the other hand, there is no scientific or empirical data to suggest another parametric structure, non-parametric regression approaches are useful. The are useful in data smoothing in exploratory statistics or as the non-parametric part in a semi parametric model. In the following example we examine three such approaches:

Kernel smoother: Is implemented by the function `ksmooth`. The `kernel` is either uniform or normal. The smoothing parameter is `bandwidth`.

Splines: Is implemented by the function `smooth.spline`. Smoothness is determined by the parameter `spar`, which is the coefficient of the integral of the squared second derivative in the penalized log likelihood criterion. Alternatively one can use `df` or cross validation.

Local polynomials: Is implemented by the function `loess`. Fitting is done locally. That is, for the fit at point x, the fit is made using points in a neighborhood of x, weighted by their distance from x The size of the neighborhood is controlled by `span`.

```
> library(modreg)
> data(cars)
> attach(cars)
> plot(speed, dist)
> lines(ksmooth(speed, dist, "normal", bandwidth=2), col=2)
> lines(ksmooth(speed, dist, "normal", bandwidth=5), col=3)
> lines(ksmooth(speed, dist, "normal", bandwidth=10), col=4)
> plot(speed, dist, main = "data(cars)  &  smoothing splines")
> cars.spl <- smooth.spline(speed, dist)
> cars.spl
Call:
smooth.spline(x = speed, y = dist)

Smoothing Parameter (Spar): 0.7986363
Equivalent Degrees of Freedom (Df): 2.520569
```

```
Penalized Criterion: 4380.504
GCV: 244.1659
> lines(cars.spl, col = "blue")
> lines(smooth.spline(speed, dist, df=10), lty=2, col = "red")
> legend(5,120,c(paste("default [C.V.] => df =",round(cars.spl$df,1)),
+ "s( * , df = 10)"), col = c("blue","red"), lty = 1:2, bg='bisque')
> plot(speed, dist, main = "data(cars)  &  loess")
> lines(speed,predict(loess(dist ~ speed)),col="blue")
> lines(speed,predict(loess(dist ~ speed,span=0.5)),col="red", lty=2)
> legend(5,120, legend = c("default", "span=0.3"), col = c("blue","red"),
+ lty = 1:2,bg='bisque')
```

8.2 The projection-pursuit model

Assume that the explanatory vector $\mathbf{X} = (X_1, \ldots, X_p)'$ is of high dimension. In order to avoid the dimensionality problem one can fit an additive model of the form:

$$Y = \alpha_0 + \sum_{j=1}^{p} f_j(X_j) + \epsilon.$$

Drawbacks of the additive model are:

1. If the dimension p is too high the additive model may be an overfit of the data.

2. The model does not include interactions — the dependence structure in one variable depends on the level of the other variables.

An alternative is the projection-pursuit (PP) model. This model is of the form:

$$Y = \alpha_0 + \sum_{j=1}^{M} f_j(\alpha_j'\mathbf{X}) + \epsilon,$$

where $M \ll p$, f_j is smooth and α_j a direction vector.

The projection-pursuit model applies an additive model to projected variables. Rather than fitting p functions in p directions the dimension can be reduced by using only the M most significant directions. This is in the spirit of principal components. The terms are called *ridge functions* since they are constant in all but one direction.

The model is fit with the function `ppr` (`ppreg` in S-plus). Consider the following example:

```
> data(rock)
> attach(rock)
> area1 <- area/10000; peri1 <- peri/10000
> rock.ppr <- ppr(log(perm) ~ area1 + peri1 + shape,
+ data = rock, nterms = 2, max.terms = 5)
> summary(rock.ppr)
Call:
ppr.formula(formula = log(perm) ~ area1 + peri1 + shape, data = rock,
    nterms = 2, max.terms = 5)

Goodness of fit:
 2 terms  3 terms  4 terms  5 terms
8.737806 5.289517 4.745799 4.490378

Projection direction vectors:
      term 1      term 2
area1  0.34357179  0.37071027
peri1 -0.93781471 -0.61923542
shape  0.04961846  0.69218595

Coefficients of ridge terms:
   term 1    term 2
1.6079271 0.5460971
> par(mfrow=c(1,2), pty="s")
> plot(rock.ppr, main="default")
> plot(update(rock.ppr, bass=5), main = "bass=5")
> plot(update(rock.ppr, sm.method="gcv", gcvpen=2), main = "gcv")
```

8.3 A simulated example of PP-regression

```
> set.seed(14)
> x1 <- runif(400, -1, 1)
> x2 <- runif(400, -1, 1)
> eps <- rnorm(400, 0, 0.2)
> y  <- x1*x2 + eps
> sim.pp <- ppr(y ~ x1 + x2, nterms=1, max.terms=5)
> summary(sim.pp)
Call:
ppr.formula(formula = y ~ x1 + x2, nterms = 1, max.terms = 5)
```

```
Goodness of fit:
 1 terms  2 terms  3 terms  4 terms  5 terms
23.95768 13.51628 12.65749 11.83397 11.02952

Projection direction vectors:
       x1        x2
0.6411524 0.7674135

Coefficients of ridge terms:
[1] 0.2651939
> par(mfrow=c(3,2))
> plot(x1, y, sub="Y vs X1")
> plot(x2, y, sub="Y vs X2")
> sim.pp <- ppr(y ~ x1 + x2, nterms=1, max.terms=5)
> summary(sim.pp)
Call:
ppr.formula(formula = y ~ x1 + x2, nterms = 1, max.terms = 5)

Goodness of fit:
 1 terms  2 terms  3 terms  4 terms  5 terms
23.95768 13.51628 12.65749 11.83397 11.02952

Projection direction vectors:
       x1        x2
0.6411524 0.7674135

Coefficients of ridge terms:
[1] 0.2651939
> sim.pp2 <- ppr(y ~ x1 + x2, nterms=2)
> summary(sim.pp2)
Call:
ppr.formula(formula = y ~ x1 + x2, nterms = 2)

Goodness of fit:
 2 terms
13.32762

Projection direction vectors:
   term 1     term 2
```

```
x1  0.6474445  0.5988468
x2  0.7621126 -0.8008636

Coefficients of ridge terms:
   term 1    term 2
0.2051119 0.1749922
> plot(sim.pp$fitted,sim.pp$residuals)
> plot(sim.pp2$fitted,sim.pp2$residuals)
> plot(sim.pp2$fitted,sim.pp$fitted)
```

8.4 Home work

1. Create and investigate, using `ppreg`, a simulated example. Try different number of variables and functions thereof.

9 Tree-based models

A (binomial) tree is a structure which includes a root, nodes, and edges. Each node is connected by an edge to a father node and may have up to two son nodes. The final nodes are called leaves. Each node with two sons corresponds to a split in the data. The paths, from the root to the leaves, correspond to subsets of the data.

Tree-based modeling is a technique for:

1. Devising prediction rules that are interpretable.
2. Screening variables.
3. Assessing linear models.
4. Summarize large data sets.

We will consider tree-based classification and tree-based regression. The difference between the two is in the type of the response. In the later the response is quantitative whereas in the former it is a factor.

Regression trees can be compared to analysis of variance and/or to `lm` using `cut`. Unlike that approach, however, the division of the range of the explanatory variables is adaptive and it includes interactions automatically. A path is a predictor. The predicted value is attached to the leave. Submodels correspond to sub-trees. The deviance of a model is defined as

$$D = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2,$$

where $\hat{Y}_i$ is the predicted value at the leave to which case i belongs.

With each leave of a classification tree, a level of the response is associated. Each case which is associated with the path that leads to that leave is classified by that level. Deviances are defined by the multinomial likelihood of different levels. Criteria like the entropy or Gini index are used in growing the tree.

Trees are grown until the number of cases reaching each leave is small (by default < 10) or the leave is homogeneous (by default less than 1% of the total deviance.) This usually leads to trees that overfit the data and pruning is needed.

Functions for fitting trees can be found in the library `tree`. This library is not part of the basic R package. It can be downloaded from

`http://cran.r-project.org/src/contrib/PACKAGES.html`. Installing is done with the program `rwinst.exe` (which was used to install the basic package). This program is available at
`http://cran.r-project.org/bin/windows/windows-9x/base/`.

9.1 An example of a regression tree

The data set *cpus.dat* contains information on relative performance measure and characteristics of 209 CPUs. The components are:

`name` Manufacturer and model.

`syct` cycle time in nanoseconds.

`mmin` minimum main memory in kilobytes.

`mmax` maximum main memory in kilobytes.

`cach` cache size in kilobytes.

`chmin` minimum number of channels.

`chmax` maximum number of channels.

`perf` published performance on a benchmark mix relative to an IBM 370/158-3.

`estperf` estimated performance (by Ein-Dor & Feldmesser).

The data comes from P. Ein-Dor & J. Feldmesser (1987) Attributes of the performance of central processing units: a relative performance prediction model. *Comm. ACM.* **30**, 308-317.

```
> cpus.dat <- read.table("cpus.dat", header=T,sep=",")
> apply(cpus.dat[,-1], 2, mean)
       syct         mmin         mmax         cach        chmin
 203.822967  2867.980861 11796.153110    25.205742     4.698565
      chmax         perf      estperf
  18.267943   105.617225    99.330144
> sqrt(apply(cpus.dat[,-1], 2, var))
       syct         mmin         mmax         cach        chmin
 260.262926  3878.742758 11726.564377    40.628722     6.816274
      chmax         perf      estperf
```

```
   25.997318   160.830587   154.757102
> library(tree)
> cpus.tr <- tree(perf ~ syct+mmin+mmax+cach+chmin+chmax, cpus.dat)
> summary(cpus.tr)

Regression tree:
tree(formula = perf ~ syct + mmin + mmax + cach + chmin + chmax,
    data = cpus.dat)
Variables actually used in tree construction:
[1] "mmax"  "cach"  "chmax" "mmin"
Number of terminal nodes:  7
Residual mean deviance:  4838 = 977300 / 202
Distribution of residuals:
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-4.756e+02 -2.144e+01 -5.638e+00 -1.707e-14  1.836e+01  3.674e+02
> plot(cpus.tr)
> text(cpus.tr)
> title("Basic regression tree for cpus.dat")
> cpus.ltr <- tree(log10(perf) ~ syct + mmin + mmax + cach + chmin + chmax,
+ data=cpus.dat)
> summary(cpus.ltr)

Regression tree:
tree(formula = log10(perf) ~ syct + mmin + mmax + cach + chmin +
    chmax, data = cpus.dat)
Variables actually used in tree construction:
[1] "cach"  "mmax"  "syct"  "chmin"
Number of terminal nodes:  10
Residual mean deviance:  0.03187 = 6.342 / 199
Distribution of residuals:
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-4.945e-01 -1.191e-01  3.571e-04  3.240e-16  1.141e-01  4.680e-01
> plot(cpus.ltr, type="u")
> text(cpus.ltr)
> title("Regression tree for log10(perf)")
> plot(prune.tree(cpus.ltr))
> title("Deviance by size of cpus.ltr")
> cpus.ltr1 <- prune.tree(cpus.ltr, best=8)
> plot(cpus.ltr1)
> text(cpus.ltr1)
```

```
> title("Pruned down regression tree for log10(perf)")
> cpus.cv <- cv.tree(cpus.ltr,FUN=prune.tree)
> plot(cpus.cv)
> title("Deviance by size of cpus.cv")
> cpus.ltr2 <- prune.tree(cpus.ltr, best=4)
```

A. B. C.

Figure 5: (A)=default tree (B)=log(perf) (C)=size vs. deviance

A. B. C.

Figure 6: (A)=pruned tree (B)=size vs. dev, CV (C)=pruned tree, CV

```
> plot(cpus.ltr2)
> text(cpus.ltr2)
> title("Pruned down by CV regression tree for log10(perf)")
```

9.2 An example of a classification tree

Description The `birthwt` data frame contains information on risk factors associated with low infant birth weight. It has 189 observations on 10 variables. The data were collected at Baystate Medical Center, Springfield, Mass during 1986.

`low` indicator of birth weight less than 2.5kg.

`age` mother's age in years.

`lwt` mother's weight in pounds at last menstrual period.

`race` mother's race (1 = white, 2 = black, 3 = other).

`smoke` smoking status during pregnancy.

`ptl` number of previous premature labours.

`ht` history of hypertension.

`ui` presence of uterine irritability.

`ftv` number of physician visits during the first trimester.

`bwt` birth weight in grams.

```
> library(MASS)
> data(birthwt)
> summary(birthwt)
      low              age              lwt             race      
 Min.   :0.0000   Min.   :14.00   Min.   : 80.0   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:19.00   1st Qu.:110.0   1st Qu.:1.000  
 Median :0.0000   Median :23.00   Median :121.0   Median :1.000  
 Mean   :0.3122   Mean   :23.24   Mean   :129.8   Mean   :1.847  
 3rd Qu.:1.0000   3rd Qu.:26.00   3rd Qu.:140.0   3rd Qu.:3.000  
 Max.   :1.0000   Max.   :45.00   Max.   :250.0   Max.   :3.000  
     smoke             ptl               ht                ui        
```

```
 Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000
 Median :0.0000   Median :0.0000   Median :0.00000   Median :0.0000
 Mean   :0.3915   Mean   :0.1958   Mean   :0.06349   Mean   :0.1481
 3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.0000
 Max.   :1.0000   Max.   :3.0000   Max.   :1.00000   Max.   :1.0000
      ftv              bwt
 Min.   :0.0000   Min.   : 709
 1st Qu.:0.0000   1st Qu.:2414
 Median :0.0000   Median :2977
 Mean   :0.7937   Mean   :2945
 3rd Qu.:1.0000   3rd Qu.:3487
 Max.   :6.0000   Max.   :4990
> attach(birthwt)
> race <- factor(race, labels=c("white","black","other"))
> table(ptl)
ptl
  0   1   2   3
159  24   5   1
> ptd <- factor(ptl >0)
> table(ftv)
ftv
  0   1   2   3   4   6
100  47  30   7   4   1
> ftv <- factor(ftv)
> levels(ftv)[-(1:2)] <-"2+"
> table(ftv)
ftv
  0   1  2+
100  47  42
> bwt <- data.frame(low=factor(low),age,lwt,race, smoke = (smoke>0),
+ ptd, ht = (ht>0), ui = (ui>0), ftv)
> detach(birthwt)
> bwt.tr <- tree(low ~ ., bwt)
> summary(bwt.tr)

Classification tree:
tree(formula = low ~ ., data = bwt)
Variables actually used in tree construction:
[1] "ptd" "lwt" "age"
```

```
Number of terminal nodes:  8
Residual mean deviance:  1.042 = 188.6 / 181
Misclassification error rate: 0.2328 = 44 / 189
> plot(bwt.tr)
> text(bwt.tr)
> title("Low birth weight classification tree")
> bwt.tr1 <- prune.tree(bwt.tr, k=2)
> summary(bwt.tr1)

Classification tree:
tree(formula = low ~ ., data = bwt)
Variables actually used in tree construction:
[1] "ptd" "lwt" "age"
Number of terminal nodes:  8
Residual mean deviance:  1.042 = 188.6 / 181
Misclassification error rate: 0.2328 = 44 / 189
> plot(prune.tree(bwt.tr))
> title("Deviance by size of bwt.tr")
> plot(bwt.tr1)
> text(bwt.tr1)
> title("Low birth weight: bwt.tr1")
```

A. B. C.

Figure 7: (A)=bwt.tr (B)=size vs. deviance (C)=bwt.tr1

9.3 Homework

Investigate the data set `bwt` using both `glm` and `tree`. Compare the results.

10 Multivariate analysis

10.1 Multivariate data

Multivariate analysis is concerned with datasets which have more than one response variable for each observation. The dataset can be put in a matrix X with n rows and p columns.

One distinguishes between methods with a given division of the cases into groups (*supervised methods*), and methods which seek to discover a structure from the data matrix alone (*unsupervised methods*)

The dataset we consider is the system dataset *iris*. This data set contains information on the width and length of the petals and sepals of three different species of Iris: *Iris setosa*, *Iris virginica*, and *Iris versicolor*.

```
> data(iris)
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
 Median :5.800   Median :3.000   Median :4.350   Median :1.300
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
       Species
 setosa    :50
 versicolor:50
 virginica :50
> apply(iris[,-5],2, tapply, iris[,5],mean,trim=0.1)
           Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa           5.0025      3.4150       1.4600      0.2375
versicolor       5.9375      2.7800       4.2925      1.3250
virginica        6.5725      2.9625       5.5100      2.0325
> apply(iris[,-5],2, tapply, iris[,5],var)
           Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa        0.1242490  0.14368980   0.03015918  0.01110612
versicolor    0.2664327  0.09846939   0.22081633  0.03910612
virginica     0.4043429  0.10400408   0.30458776  0.07543265
```

10.2 Graphical methods

The function `pairs` plots an array of scatter plots. One can add labels or colors to different observations.

```
> ir <- iris[,-5]
> pairs(ir)
> ir.species <- c(rep("s",50),rep("c",50), rep("v",50))
> attributes(ir)$row.names <- ir.species
> library(mva)
> biplot(princomp(ir))
> attributes(ir)$row.names <- 1:150
```

10.3 Principal components analysis

Principal components analysis seeks linear combinations of the variables with maximal (or minimal) variance. Combinations are constraint to have unit length.

The covariance matrix is defined by

$$(n-p)\Sigma = (X - n^{-1}1'X1)'(X - n^{-1}1'X1) = (X'X - n\bar{x}\bar{x}')$$

The correlation matrix can be defined in a similar way. The covariance (or correlation) matrix is non-negative definite hence has an eigen decomposition

$$\Sigma = C'\Lambda C,$$

where Λ is diagonal and $C'C = I$. The principle components correspond to eigenvectors of maximal eigenvalues. A measure of how good are the first k principle components is

$$\sum_{i=1}^{k} \lambda_i / \sum_{i=1}^{p} \lambda_i.$$

The S function `prcomp` works directly with X and uses the singular value decomposition

$$X = U\Lambda V'.$$

```
> ir.pr <- prcomp(scale(log(ir)))
> plot(ir.pr$x[,1:2], type="n")
> text(ir.pr$x[,1:2], ir.species)
```

```
> ir.pr$sdev
[1] 1.7124583 0.9523797 0.3647029 0.1656840
> ir.pr$rot
                    PC1         PC2        PC3         PC4
Sepal.Length  0.5038236 -0.45499872 -0.7088547  0.19147575
Sepal.Width  -0.3023682 -0.88914419  0.3311628 -0.09125405
Petal.Length  0.5767881 -0.03378802  0.2192793 -0.78618732
Petal.Width   0.5674952 -0.03545628  0.5829003  0.58044745
```

10.4 cluster analysis

Cluster analysis is concerned with discovering group structure amongst the cases. A *dissimilarity coefficient* measures the distances between two cases. Several dissimilarity can be calculated with the function `dist`.

```
> hc <- hclust(dist(ir), "ave")
> ir.species <- c(rep(".",50),rep("'",50), rep("-",50))
> plot(hc, hang=-1, labels=ir.species)
> title("hclust(iris), method=average")
> hc <- hclust(dist(ir), "complete")
> plot(hc, hang=-1, labels=ir.species)
> title("hclust(iris), method=complete")
> hc <- hclust(dist(ir), "single")
> plot(hc, hang=-1, labels=ir.species)
> title("hclust(iris), method=single")

> hc <- hclust(dist(ir), "average")
> A <- list(factor(rep(cutree(hc,3), 4)),
+ factor(rep(1:4,rep(dim(ir)[1],4))))
> tapply(unlist(ir), A, mean)
         1        2        3        4
1 5.006000 3.428000 1.462000 0.246000
2 5.929688 2.757813 4.410938 1.439062
3 6.852778 3.075000 5.786111 2.097222
> initial <- tapply(unlist(ir), A, mean)
> km <- kmeans(ir, initial)
> km$centers
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.006000    3.428000     1.462000    0.246000
```

```
2     5.901613    2.748387     4.393548    1.433871
3     6.850000    3.073684     5.742105    2.071053
> km$size
[1] 50 62 38
> x <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),
+ matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2))
> cl <- kmeans(x, 2, 20)
> plot(x, pch = cl$cluster)
> points(cl$centers, col = 1:2, pch = 8)
> title("A simulated example of K-means")

> library(tree)
> ir.species <- factor(c(rep("s",50),rep("c",50), rep("v",50)))
> ird <- data.frame(ir)
> ir.tr <- tree(ir.species ~.,ird)
> summary(ir.tr)
Classification tree:
tree(formula = ir.species ~ ., data = ird)
Variables actually used in tree construction:
[1] "Petal.Length" "Petal.Width"  "Sepal.Length"
Number of terminal nodes:  6
Residual mean deviance:  0.1253 = 18.05 / 144
Misclassification error rate: 0.02667 = 4 / 150

> plot(ir.tr)
> text(ir.tr)
> ir.tr1 <- snip.tree(ir.tr)
node number:  12
   tree deviance =  18.05
   subtree deviance =  22.77
node number:  7
   tree deviance =  18.05
   subtree deviance =  22.28
> par(pty="s")
> plot(ird[,3],ird[,4], type ="n", xlab="petal length",
+ ylab="petal width")
> text(ird[,3],ird[,4], as.character(ir.species))
> par(cex=2)
> partition.tree(ir.tr1, add=T)
```

10.5 Home work

1. The system data `swiss` gives five measures of socio-economic data on Swiss provinces about 1888. Can you identify clusters? Use the function `kmeans`.

2. Plot the data points on the principal components axis. Use the cluster number as a mark, and add to the plot the clusters centers.

10.6 Classification

Modern usage of the term *classification* refers to allocation of future cases into one of g classes. This is similar to *discriminant analysis*. This is a part of pattern recognition, with cluster analysis is unsupervised and classification, or diagnosis is supervised.

The given cases with their classifications are a training set, and further cases for a test set. The primary measure of success is the misclassification rate, or error rate. A confusion matrix gives the number of cases with true class i classified as of class j. In some problems some errors are considered worse than others. We will get biased estimate of success by re-classifying the training set.

The function `lda` in the `MASS` library creates classification rules base on dividing the space with linear hyper-planes. Let W be the within-class covariance matrix, and let B be the between-class covariance matrix. Let M be the $g \times d$ matrix of class means, and G the $n \times g$ matrix of class indicators. Then the predictions are GM. Let $\bar{b}x$ be the overall mean. Thus,

$$W = \frac{(X - GM)'(X - GM)}{n - g}, \quad B = \frac{(GM - \mathbf{1}\bar{\mathbf{x}})'(GM - \mathbf{1}\bar{\mathbf{x}})}{g - 1}.$$

Fisher's linear discriminant rule, which is based on a linear functional, characterized by a vector $\mathbf{a}$. The vector $\mathbf{a}$ maximizes the ratio $\mathbf{a}'B\mathbf{a}/\mathbf{a}'W\mathbf{a}$. The vector $\mathbf{a}$ can be associated with largest eigenvalue of BW^{-1}. Classification is based on the level of $\mathbf{x}'a$ The procedure `lda` extends this approach by considering also the other eigenvalues of BW^{-1}.

```
> library(MASS)
> data(iris)
> ir <- iris[,1:4]
> ir.species <- rep(c("s","c","v"),c(50,50,50))
> a<- lda(log(ir), ir.species)
> a$svd^2/sum(a$svd^2)
[1] 0.996498601 0.003501399
> a.x <- predict(a, log(ir), dimen = 2)$x
> eqscplot(a.x, type="n", xlab="first linear discriminant",
+ ylab="second linear discriminat")
> text(a.x, as.character(ir.species))
> train <- sample(1:150, 75)
> table(ir.species[train])
```

```
 c  s  v
29 21 25
> z <- lda(ir.species ~ ., data.frame(ir,ir.species),
+ prior = c(1,1,1)/3, subset = train)
```

Figure 8: Linear discrimination

```
> predict(z, ir[-train, ])$class
 [1] s s s s s s s s s s s s s s s s s s s s s s s s s s s s s c c c c c c c
[37] c c c c c v c c c c c c c c c v v v v v v v v v v v v v v v v v v v v v
[73] v v v
Levels:  c s v
```

The function `qda` in `MASS` does quadratic discriminant analysis. This corresponds to the Bayes rule which minimizes

$$L(c) = (\mathbf{x} - \bar{\mathbf{x}}_c)' W_c^{-1} (\mathbf{x} - \bar{\mathbf{x}}_c) + \log(\det(W_c)) - 2 \log \pi_c,$$

where $\bar{\mathbf{x}}_c$ is the mean of class c, W_c is the estimate of the covariance matrix of the class and π_c is the prior probability of belonging to the class.

```
> tr <- sample(1:50,25); te <- (1:50)[-tr]
> train <- rbind(ir[tr,],ir[tr+50,], ir[tr+100,])
> test <- rbind(ir[te,],ir[te+50,],ir[te+100,])
> cl <- factor(c(rep("s",25),rep("c",25), rep("v",25)))
> z <- qda(train, cl)
> predict(z,test)$class
 [1] s s s s s s s s s s s s s s s s s s s s s s s s s c c c c c c c c c c c
[37] c c c c c c c c c c c c c c c v v v v v v v v v v v v v v v v v v v v v
[73] v v v
Levels:  c s v
```

Let us compare the two procedures in the following example. The dataset `Cushings` from the `MASS` package refers to some diagnostic tests on patients with Cushing's syndrome. Cushing's syndrome is a hypertensive disorder associated with over-secretion of cortisol by the adrenal gland. The observations are urinary excretion rates of two steroid metabolites. There are 27 observations. The variables are:

Tetrahydrocortisone: urinary excretion rate (mg/24hr) of Tetrahydrocortisone.

Pregnanetriol: urinary excretion rate (mg/24hr) of Pregnanetriol.

Type: underlying type of syndrome, coded "a" (adenoma) , "b" (bilateral hyperplasia), "c" (carcinoma) or "u" for unknown.

(Source: J. Aitchison and I. R. Dunsmore (1975) Statistical Prediction Analysis. Cambridge University Press, Tables 11.1-3.)

```
> predplot <- function(){}
> fix(predplot)
```

In the editor window write:

```
function(object, main="", len=100, ...)
{
   plot(Cushings[,1], Cushings[,2], type="n", log="xy",
    xlab="Tetrahydrocortisone", ylab="Pregnanetriol", main=main)
   text(Cushings[1:21,1], Cushings[1:21,2], as.character(tp))
   text(Cushings[22:27,1], Cushings[22:27,2], "u")
   xp <- seq(0.6, 4, length = len)
   yp <- seq(-3.25, 2.45, length = len)
   cushT <- expand.grid(Tetrahydrocortisone=xp,
             Pregnanetriol=yp)
   Z <- predict(object, cushT, ...)
   z <- unclass(Z$class)
   zp <- Z$post[,3] - pmax(Z$post[,2], Z$post[,1])
   contour(exp(xp), exp(yp), matrix(zp, len), add=T,
    levels=0, labex=0)
   zp <- Z$post[,1] - pmax(Z$post[,2], Z$post[,3])
   contour(exp(xp), exp(yp), matrix(zp, len), add=T,
    levels=0, labex=0)
   invisible()
}
```

After saving and exiting:

```
> data(Cushings)
> cush <- log(as.matrix(Cushings[,-3]))
> tp <- factor(Cushings$Type[1:21])
> cush.lda <- lda(cush[1:21,], tp); predplot(cush.lda, "LDA")
> cush.qda <- qda(cush[1:21,], tp); predplot(cush.qda, "QDA")
> predplot(cush.qda, "QDA, predictive", method="predictive")
> predplot(cush.qda, "QDA, debiased", method="debiased")
```

(**Remark:** Note that you should install the latest version of R (rw1001) and the latest version of MASS (VR.zip) on your computer. On my computer the above examples did not work with version rw1000.)

A. B.

Figure 9: A= Linear, B=Quadratic

A. B.

Figure 10: A= Predictive, B=Debiased

The function `knn` in the library `class` (part of the base R package) performs a k-nearest neighbor classification for test set from training set. For each row of the test set, the k nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the kth nearest vector, all candidates are included in the vote. The function `knn.cv` in the library `class` performs a k-nearest neighbor cross-validatory classification

```
> library(class)
> knn(train, test,cl, k=3, prob=TRUE)
 [1] s s s s s s s s s s s s s s s s s s s s s s s s s s c c c c c c c c c c c c
[37] c v c c c v c c c c c c c c v v v v c v v v v v c v v v v v v v v v v v v
[73] v v v
Levels:  c s v
> attributes(.Last.value)
$levels
[1] "c" "s" "v"
$class
[1] "factor"
$prob
 [1] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
 [8] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[15] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[22] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[29] 1.0000000 0.6666667 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[36] 1.0000000 1.0000000 0.6666667 1.0000000 1.0000000 1.0000000 0.6666667
[43] 1.0000000 1.0000000 1.0000000 0.6666667 1.0000000 1.0000000 1.0000000
[50] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[57] 1.0000000 1.0000000 1.0000000 1.0000000 0.6666667 1.0000000 1.0000000
[64] 1.0000000 1.0000000 0.6666667 0.6666667 1.0000000 1.0000000 0.6666667
[71] 1.0000000 1.0000000 1.0000000 0.6666667 1.0000000
> knn.cv(train, cl, k=3, prob=TRUE)
 [1] s s s s s s s s s s s s s s s s s s s s s s s s s s c v c c c c c c c c c
[37] c c c c v c c c c c c c c c v c v v v v v v v v v v v v v v v v v v v v c
[73] v v v
Levels:  c s v
> attributes(.Last.value)
$levels
[1] "c" "s" "v"
$class
```

```
[1] "factor"
$prob
 [1] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
 [8] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[15] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[22] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[29] 1.0000000 0.7500000 1.0000000 0.6666667 1.0000000 1.0000000 0.6666667
[36] 1.0000000 1.0000000 0.6666667 1.0000000 1.0000000 0.6666667 1.0000000
[43] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
[50] 1.0000000 1.0000000 0.5000000 1.0000000 1.0000000 1.0000000 1.0000000
[57] 0.6666667 1.0000000 0.6666667 1.0000000 1.0000000 1.0000000 1.0000000
[64] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 0.6666667 1.0000000
[71] 1.0000000 0.5000000 1.0000000 0.6666667 1.0000000
```

10.7 Multivariate linear models

A multivariate linear model is a model of the form $Y = XB + E$, where Y is the matrix of responses, X is the design matrix, B is the matrix of coefficients and E a matrix of normally distributed errors. Parameter restrictions can be included by the Z matrix: $ZB = 0$ (which has applications in MANOVA).

Multivariate linear models can be fit with the function `multilm` from the contributed library `multilm`. Additionally, this function calculates the Hotelling T^2-Test for the given test problem: $H0 : KB = 0$. An approximation by Laeuter is used for the distribution of the T^2-statistic (and therefore for the p-value).

```
> library(multilm)
> Y <- as.matrix(iris[,1:4])
> x <- c(rep(1,50), rep(0,150), rep(1, 50), rep(0, 150), rep(1,50))
> X <- matrix(x, ncol=3)
> Z <- c(0,1,1,1)
> K <- cbind(0,diag(2),-1)
> K
     [,1] [,2] [,3] [,4]
[1,]    0    1    0   -1
[2,]    0    0    1   -1
> mod <- multilm(Y ~ X, K,Z)
> summary(mod)
test procedure:  Hotelling
```

```
test statistic:  584.5918
degrees of freedom DF1:  8  DF2:  144
p-value:  0
```

10.8 Canonical correlations

The canonical correlation analysis seeks linear combinations of the 'y' variables which are well explained by linear combinations of the 'x' variables. The relationship is symmetric as 'well explained' is measured by correlations. Formally, we look for vectors $\mathbf{a}$ and such that the correlation

$$\text{corr}(\mathbf{x}'\mathbf{a}, \mathbf{y}'\mathbf{b}) = \frac{\mathbf{a}'\Sigma_{xy}\mathbf{b}}{(\mathbf{a}'\Sigma_{xx}\mathbf{a})^{1/2}(\mathbf{b}'\Sigma_{yy}\mathbf{b})^{1/2}}$$

is high.

The function `cancor` from the package `mva` (part of the basic installation compute the canonical correlations between two data matrices.

Here we apply it on the dataset `LifeCycleSavings`. Under the life-cycle savings hypothesis as developed by Franco Modigliani, the savings ratio (aggregate personal saving divided by disposable income) is explained by per-capita disposable income, the percentage rate of change in per-capita disposable income, and two demographic variables: the percentage of population less than 15 years old and the percentage of the population over 75 years old. The data are averaged over the decade 1960-1970 to remove the business cycle or other short-term fluctuations.

The data frame contains 50 observations on 5 variables:

sr: aggregate personal savings.

pop15: % of population under 15.

pop75: % of population over 75.

dpi: real per-capita disposable income.

ddpi: % growth rate of dpi.

```
> library(mva)
> data(LifeCycleSavings)
> pop <- LifeCycleSavings[, 2:3]
> oec <- LifeCycleSavings[, -(2:3)]
```

```
> cancor(pop, oec)
$cor
[1] 0.8247966 0.3652762
$xcoef
             [,1]        [,2]
[1,] -0.009110856 -0.03622206
[2,]  0.048647514 -0.26031158
$ycoef
             [,1]          [,2]          [,3]
[1,] 0.0084710221  3.337936e-02 -5.157130e-03
[2,] 0.0001307398 -7.588232e-05  4.543705e-06
[3,] 0.0041706000 -1.226790e-02  5.188324e-02
$xcenter
  pop15   pop75
35.0896  2.2930
$ycenter
       sr       dpi      ddpi
   9.6710 1106.7584    3.7576

> x <- matrix(rnorm(150), 50, 3)
> y <- matrix(rnorm(250), 50, 5)
> str(cxy <- cancor(x, y))
List of 5
 $ cor    : num [1:3] 0.3784 0.1365 0.0933
 $ xcoef  : num [1:3, 1:3]  0.10879 -0.09125  0.06089 -0.07601  0.00771 ...
 $ ycoef  : num [1:5, 1:5] -0.033498  0.167432 -0.022940 -0.000426 -0.077757 ...
 $ xcenter: num [1:3]  0.0440 -0.0723 -0.2179
 $ ycenter: num [1:5] -0.0121  0.1620 -0.2470 -0.0883 -0.0249
> all(abs(cor(x %*% cxy$xcoef,
+ y %*% cxy$ycoef)[,1:3] - diag(cxy $ cor)) < 1e-15)
[1] TRUE
> all(abs(cor(x %*% cxy$xcoef) - diag(3)) < 1e-15)
[1] TRUE
> all(abs(cor(y %*% cxy$ycoef) - diag(5)) < 1e-15)
[1] TRUE
```

10.9 Project 3

The data frame `crabs` in the package `MASS` contains morphological measurements on Leptograpsus crabs. This data frame has 200 rows and 8 columns, describing 5 morphological measurements on 50 crabs each of two colour forms and both sexes, of the species Leptograpsus variegatus collected at Fremantle, W. Australia. The variables are:

sp: species — "B" or "O" for blue or orange.

sex: as it says.

index: index 1:50 within each of the four groups.

FL: frontal lobe size (mm).

RW: rear width (mm).

CL: carapace length (mm).

CW: carapace width (mm).

BD: body depth (mm).

Do classification analysis for this data. Create a training set and a test set. Try different approaches and compare their performance.