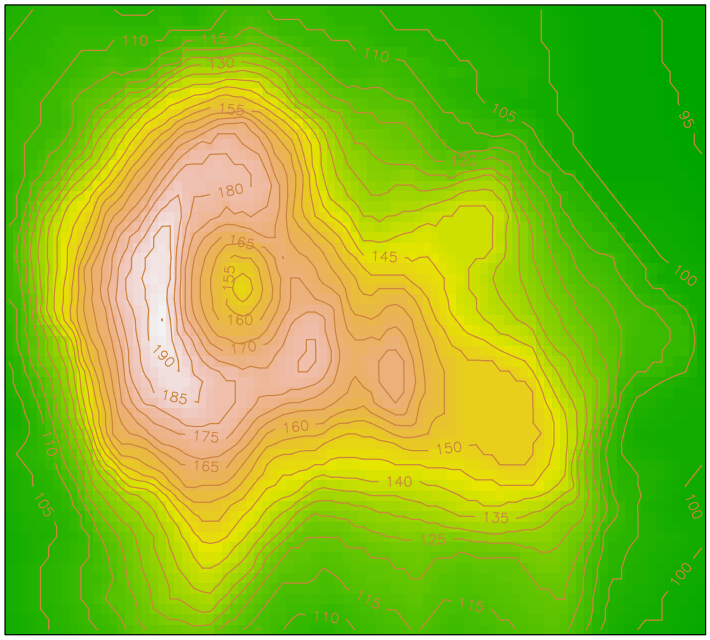
CSIRO Mathematical and Information Sciences

An Introduction to R:
Software for Statistical Modelling & Computing

Institute for Mathematical Sciences
National University of Singapore
28 February - 3 March 2005



Course Exercises

Petra Kuhnert and Bill Venables

CSIRO Mathematical and Information Sciences
Cleveland, Australia

Contents

Workshop Outline

Overview

This three day course will provide an overview of the R, software for statistical modelling and computing. The course will provide an elementary introduction to the software, introduce participants to the statistical modelling and graphical capabilities of R as well as provide an overview of three advanced topics: programming, neural networks and classification and regression trees.

Day 1 **28 February**	**AN ELEMENTARY INTRODUCTION TO R**
0830-0900	Registration
0900-1000	**01** Whirlwind Tour of R
1000-1015	Break
1015-1115	**02** R and the Tinn-R Editor (includes demonstration)
1115-1145	Tutorial and Morning Tea
1145-1230	**03** R Objects
1230-1300	Tutorial
1300-1400	Lunch/Break
1400-1530	**04** Graphics: An Introduction (includes tutorial)
1530-1600	Afternoon Tea
1600-1700	**05** Manipulating Data
1700-1730	Tutorial

Day 2 **1 March**	**STATISTICAL MODELLING AND GRAPHICS**
0830-0930	**06** Classical Linear Models (includes tutorial)
0930-0945	Break
0945-1030	**07** Non-Linear Regression (includes tutorial)
1030-1100	Morning Tea
1100-1200	**08** Generalized Linear Modelling and Extensions (includes tutorial)
1200-1215	Break
1215-1300	**09** Generalized Additive Models: An Introduction (includes tutorial)
1300-1345	Lunch/Break
1345-1445	**10** Advanced Graphics (includes tutorial)
1445-1500	Break
1500-1545	**11** Importing and Exporting
1545-1615	Afternoon Tea
1615-1730	**12** Mixed Effects Models: An Introduction (includes tutorial)

Day 3 **2 March**	**ADVANCED TOPICS**
0900-1030	**13** Programming (includes tutorial)
1030-1100	Morning Tea
1100-1200	**14** Neural Networks: An Introduction (includes tutorial)
1200-1300	Lunch/Break
1300-1400	**15** Tree-based Models I: Classification Trees
1430-1500	Tutorial
1500-1530	Afternoon Tea
1530-1630	**16** Tree-based Models II: Regression Trees and Advanced Topics
1630-1700	Tutorial
1700-1730	Review and Feedback

Laboratory Exercises

These exercises are an adaptation of Bill Venable's Data Analysis and Graphics Course for S-PLUS.

Lab 1: R - An Introductory Session

The following session is intended to introduce you to some features of R, some of which were highlighted in the first session of the workshop.

Starting the R Session

`Create Directory`	Create a new directory on the desktop. Call it Session1.
`Right Click R Icon`	Create a shortcut by right clicking on the R icon. Rename the R session to `Session1`. Right click the R Icon and select Properties. Alter the Start in directory to reflect the new directory that you set up on the desktop. Click on OK once it has been entered.
`Double Click R Icon`	Start R by double clicking on the R icon that you have just created.
`Edit GUI Preferences`	Go to the Edit Menu and select GUI Preferences. Select SDI for Single window display. Click on OK.

Setting up the Graphics Device

`> require(lattice)`	Attach the trellis library
`> ?  trellis.par.set(col.whitebg())`	Get help on trellis parameter settings
`> show.settings()`	- Show current trellis settings
`> trellis.par.set(col.whitebg())`	Change the color of the background
`> windows()`	Standard graphics device

Simple Model

We will start off with a little artificial simple linear regression example, just to get you used to assignments, calculations and plotting.

The data we generate will be of the form $y = 1 + 2x + \text{errors}$ and then try to estimate the intercept and slope, which we know are 1 and 2 respectively and see how well we do.

The errors will be made to *fan out* as you go along the line. Such *heteroscedasticity* is a common feature of many real data sets.

```
> x <- 1:50
```
Make $x = (1, 2, \ldots, 49, 50)$.

```
> w <- 1 + x/2
```
We will use w to make the errors *fan out* along the line.

```
> rdata <- data.frame(x=x,
 y=1+2*x+rnorm(x)*w)
> rdata
```
Make a *data frame* of two columns, x and y, and look at it.

```
> fm <- lm(y~x,data=rdata)
> summary(fm)
```
Fit a simple linear regression of y on x and look at the analysis.

```
> attach(rdata)
```
Make the columns in the data frame visible as variables.

```
> search()
> objects(2)
```
Look at the *search list*. This is the database from which R finds its objects. Look at those in position 2.

```
> plot(x,y)
```
Standard point plot.

```
> abline(1,2,lty=3)
```
The true regression line: (intercept: $a = 1$, slope: $b = 2$, line type: 3)

```
> abline(coef(fm),col="blue")
```
Mark in the estimated regression line

```
> segments(x,fitted(fm),x,y,
 lty=4,col="red")
```
Join the points vertically to the fitted line, thus showing the errors or *residuals*. We will look more closely at the residuals below. At any time you can make a hard copy of the graphics by clicking on the `File` menu and selecting `Print`.

```
> plot(fitted(fm),resid(fm),
 xlab="Fitted values",
 ylab="Residuals",main=
 "Residuals vs Fitted")
> abline(h=0,lty=4)
```
A standard regression diagnostic plot to check for unequal variance. Can you see it?

```
> qqnorm(resid(fm),main=
 "Normal Scores Plot")
> qqline(resid(fm))
```
A Normal scores plot to check for skewness, kurtosis and outliers in the residuals. (Not very useful here since these properties are bound up with unequal variance.

```
> detach("rdata")
```
Remove data frame from the search list.

```
> rm(fm,x,rdata)
```
Clean up.

Cars93 Dataset

```
> require(MASS)
```
Now we look briefly at some actual data.

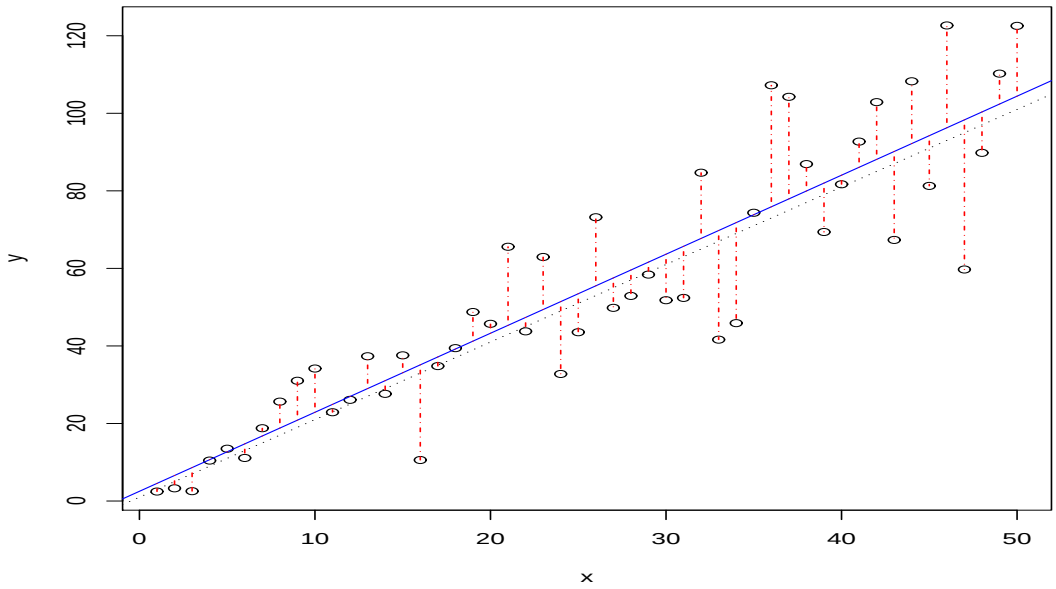


Figure 1: A simulated heteroscedastic regression.

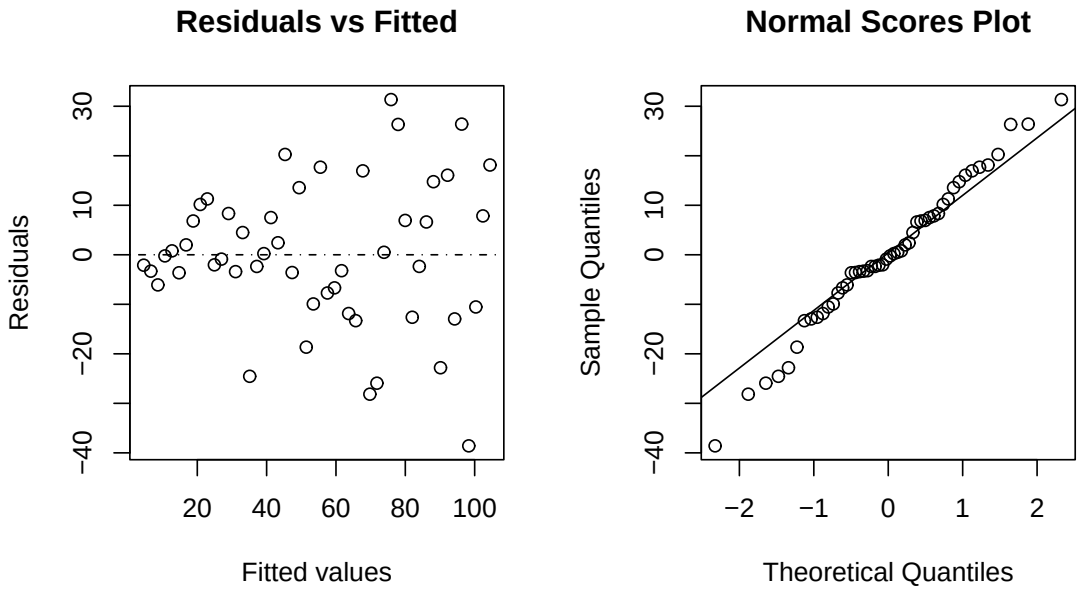


Figure 2: Heteroscedastic regression: diagnostic plots

```
> ?Cars93
```

The `Cars93` data frame contains information about 93 makes of car (or van) on sale in the USA in 1993. For each make there are 26 variables recorded ranging from `Price` (of the basic model) through to `Weight`. The help document gives more information.

```
> with(Cars93,plot(Type,
 Price,
ylab="Price (in $1,000)")
```

`Type` is a factor, not a quantitative variable. In this case a plot gives a boxplot by default.

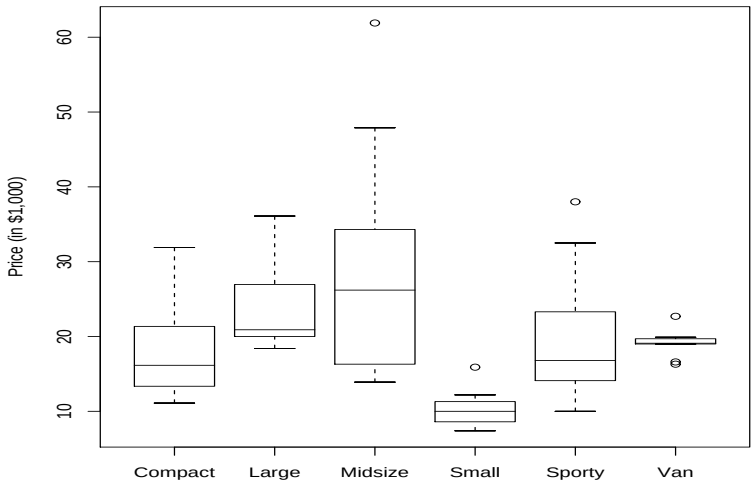


Figure 3: Boxplots of price on type of car

```
> attach(Cars93)
> Tf <- table(Type)
> Tf
> Cf <- table(Cylinders)
> Cf
> TC <- table(Type,Cylinders)
> TC
> Make[Cylinders=="5"]
> rbind(cbind(TC,Tf),
 c(Cf,sum(Cf)))
> plot(Weight,MPG.city)
> Cars93T <- transform(Cars93,
```

How many cars are there of each type?
With each number of cylinders?

A two-way table. What types of car were the two with five cylinders?
Which makes are they? (Note *two* equal signs.
Put the marginal totals on the table.

City MPG drops off with increasing weight, but not in a straight line, which would be simpler.
Try *gallons per mile* instead.

```
 GPM=1/MPG.city)
> with(Cars93T,
 plot(Weight,GPM,pch=3))
> fm <- lm(GPM Weight,Cars93T)
> abline(fm,lty=4,col="blue")
```

The regression looks much more linear. Note the use of the `transform` and `with` functions. Fit a straight line model and add it to the curve. There is a lot of deviation, but the model seems appropriate.

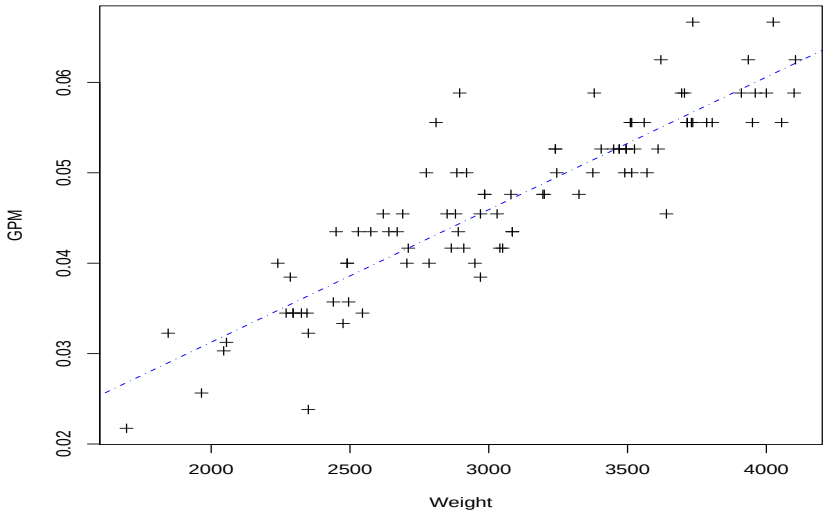


Figure 4: Gallons per mile vs car weight

```
> plot(fitted(fm),
 resid(fm))
> abline(h=0,lty=2)
```

Large positive residuals indicate good fuel economy (after allowing for weight), whereas large negative residuals indicate poor fuel economy.

```
> identify(fitted(fm),
 resid(fm),Make)
```

With the mouse, position the cursor in the plot window. It will change shape. Place it next to one of the more extreme residuals and click with the *left* mouse button. Repeat for as many as you like. When you have identified enough, stop the process by clicking on the `stop` button in the top left hand corner of the plotting region.

Graphics

We now look at some more graphical facilities: contour and 3-dimensional perspective plots.

```
> x <- seq(-pi,pi,len=50)
> y <- x
```
x is a vector of 50 equally spaced values in $-\pi \le x \le \pi$. y is the same.

```
> f <- outer(x,y,
 function(x,y)
 cos(y)/(1+x^2))
```
After this command, f is a square matrix with rows and columns indexed by x and y respectively, of values of the function $\cos(y)/(1+x^2)$.

```
> par(pty="s")
```
region to *square*.

```
> contour(x,y,f)
> contour(x,y,f,
 nint=15,add=T)
```
Make a contour map of f and add in more lines for more detail.

```
> fa <- (f-t(f))/2
```
`fa` is the *asymmetric* part of f. `t()` is transpose).

```
> contour(x,y,fa,
 nint=15)
```
Make a contour and

```
> persp(x,y,f)
> persp(x,y,fa)
> image(x,y,f)
> image(x,y,fa)
```
Make some pretty perspective and high density image plots, of which, you can get hard copies if you wish.

```
> objects()
> rm(x,y,f,fa)
```
and clean up before moving on.

Complex Arithmetic

```
> th <- seq(-pi,pi,len=200)
> z <- exp(1i*th)
```
R can do complex arithmetic also. `1i` is used for the complex number i.

```
> par(pty="s")
> plot(z,type="l")
```
Plotting complex arguments means plot imaginary versus real parts. This should be a circle.

```
> w <- rnorm(1000)+
 rnorm(1000)*1i
> w <- ifelse(abs(w)>1,
 1/w,w)
```
Suppose we want to sample points within the unit disc. One method would be to take complex numbers with random standard normal real and imaginary parts and map any points outside the disc onto their reciprocal.

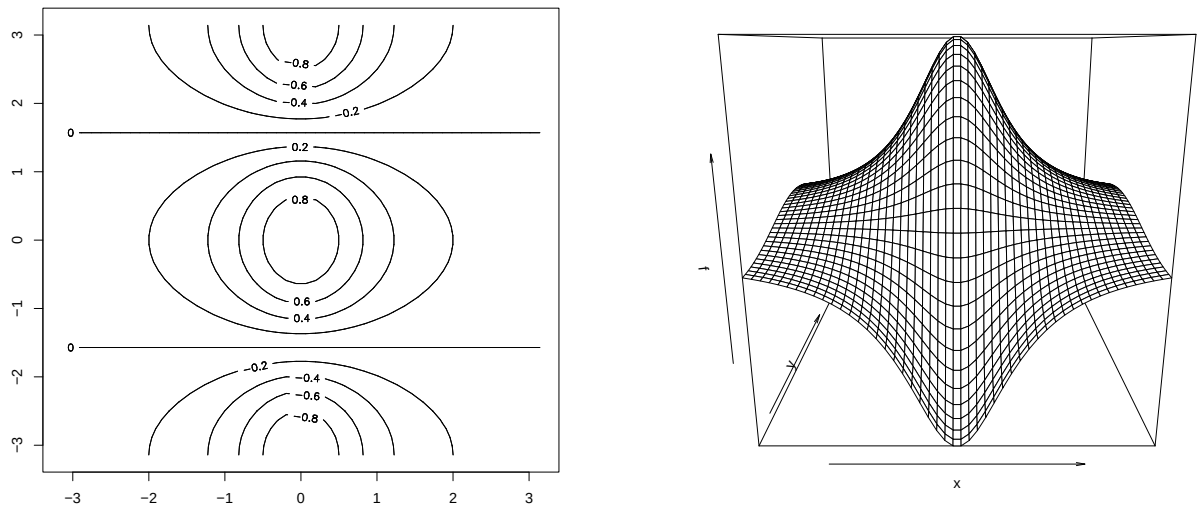


Figure 5: Contour and perspective mesh plot of f

```
> plot(w,xlim=c(-1,1),
 ylim=c(-1,1),pch=4,
 xlab="",ylab="",
 axes=F)
> lines(z)
```

All points are inside the unit disc but the distribution is not uniform.

```
> w <- sqrt(runif(1000))*
 exp(2*pi*runif(1000)*li)
```

The second method uses the uniform distribution. The points should now look more evenly spaced over the disc.

```
> plot(w,xlim=c(-1,1),
 ylim=c(-1,1),pch=1,
 xlab="",ylab="",
 axes=F)
> lines(z)
> rm(th,w,z)
> q()
```

You are not expected to know why this works by the way,but working out why it does so is a little problem for the mathematically adept.

Clean up again
Quit the R session

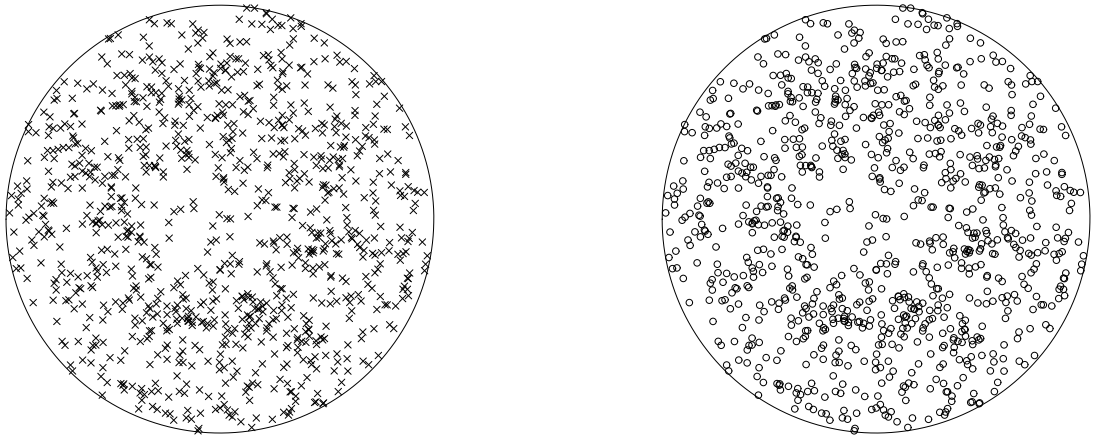


Figure 6: Two samplings of the unit disc

Lab 2: Understanding R Objects

In this exercise you will be expected to do some of the calculations for yourself. Use the help facilities as much as you need. After the pace of the introductory session this one may seem a little slow. All of the datasets are located in the MASS library so do not forget to attach it. Incidentally, if you want to locate a dataset use the `find` command.

Animal Brain and Body Sizes

Read the help file for the `Animals` data frame, which gives average brain and body weights for certain species of land animals. Attach the data frame at level 2 of the search list and work through the following.

Try using the **Tinn-R** editor to write and run scripts produced in this session.

1. Make a *dotchart* of `log(body)` sizes.
   `> dotchart(log(body),row.names(Animals),xlab="log(body)")`
   The second argument to `dotchart()` gives the names to appear on the left. With data frames, the *row names* become the *names* of each variable.

   Notice that the information from the chart is somewhat unclear with the animals in no special order. One way to improve this is to arrange them in sorted order.

   ```
   > s <- sort.list(body)
   > dotchart(log(body[s]),row.names(Animals[s,]),xlab="log(body)")
   ```

   The dotchart should now be much more comprehensible.

2. Produce a similar dotchart of log(brain) sizes arranging the animals in sorted order by *body* size. What interesting features do you notice, if any?

Detach the data frame and cleanup temporaries before proceeding to the next question.

H O Holck's Cat Data

Examine the data frame `cats` through its help file.

1. Attach the data frame at position 2 and look at the levels of the factor `Sex`.

   ```
   > sex <- levels(Sex)
   > sex
   ```

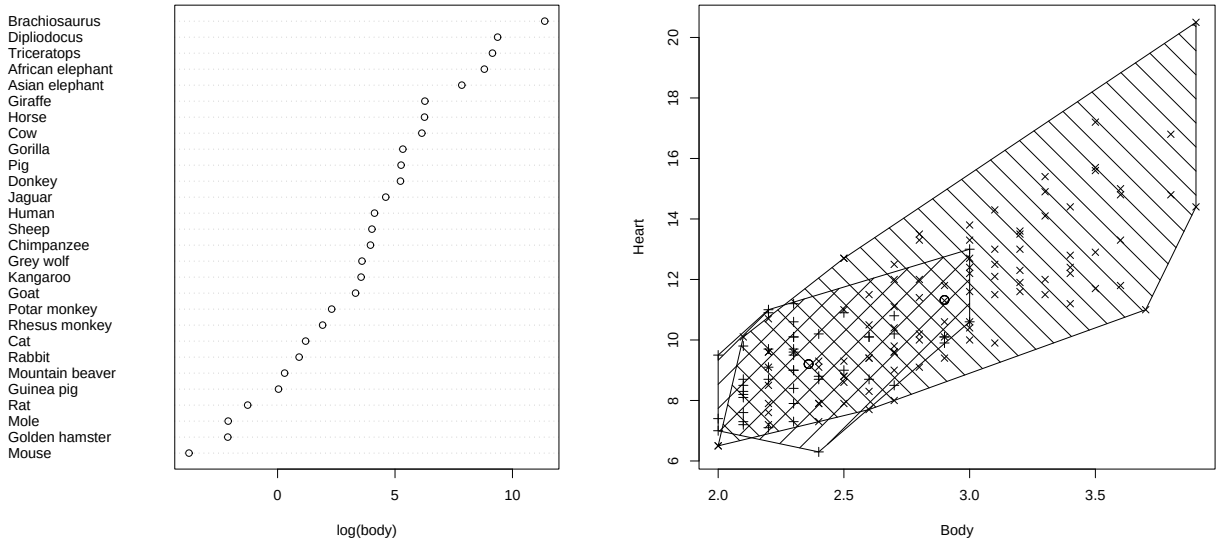


Figure 7: Two anatomical data displays

Note: lower case here

2. Plot heart weight against body weight, marking in the two samples with a different character. One way to do this is using a simple loop:

```
> plot(Bwt,Hst,type="n",xlab="Body",ylab="Heart") # axes only
> for(i in 1:2)
      points(Bwt[Sex==sex[i]],Hwt[Sex==sex[i]],pch=2+i)
```

(Loops will be discussed later in the course.)

Another way to do something similar would be to use `text()` in place of the last two commands:

```
> text(Bwt,Hwt,c("*","+")[Sex])
```

but this way does not allow the more precise plotting symbols to be used. Try both.

3. Find the means of each sample and print them. A direct way to do this is as follows:

```
> BwtF <- mean(Bwt[Sex=="F"]); ...
```

With only two groups this is probably the simplest. With more than two groups though, it becomes exceedingly tedious. A better way is to use the `tapply()` function, (whose semantics you should look up).

```
> Bmeans <- tapply(Bwt,Sex,mean)
> Hmeans <- tapply(Hwt,Sex,mean)
```

4. Look up the `symbols()` command and use it to add $\frac{1}{10}$inch circles to your plot at the two sample mean points.

```
> symbols(Bmeans,Hmeans,circles=c(1,1),inches=0.05,add=T)
```

5. **[Optional]** Try to put polygons on the diagram showing the convex hull of each sex.

   Here is a fairly slick way to do it, but it could be done in a loop as well:

```
> hF <- chull(xF <- as.matrix(cats[Sex=="F",-1]))
> hM <- chull(xM <- as.matrix(cats[Sex=="M",-1]))
> polygon(xF[hF,],dens=5,angle=45)
> polygon(xM[hM,],dens=5,angle=135)
```

Combining Two Data Frames with Some Common Rows

The data frame `mammals` contains brain and body weight data for 62 species of land mammals, including some of the entries also contained in the `Animals` data frame. Our problem here is to put the two data frames together, removing duplicate entries.

Since the variable names of the two data frames are the same, and the row names are identical where they refer to the same animal, we can discover where the duplicates are by looking at the two `row.names` vectors together. There is a useful function for checking whether an entry in a vector has already occurred earlier:

```
> nam <- c(row.names(Animals),row.names(mammals))
> dup <- duplicated(nam)
```

We now put the data frames together and remove duplicate entries. The function `rbind()` may be used to *bind* data frames together *by rows*, that is, stack them on top of each other, aligning variables with the same name.

```
> AnimalsALL <- rbind(Animals,mammals)[!dup,]
> rm(dup,nam)
```

Note the empty second subscript position. This subsets elements by row.

The Tuggeranong House Data

1. Attach the Tuggeranong house price data frame and look at the variables

   N.B. The Tuggeranong house data is not part of the MASS dataset. The data is provided as a comma delimited text file and can be read in using the `read.csv` function which will be discussed in a later session.

   ```
   > house <- read.csv("houses.csv")
   > attach(house)
   > house
   ```

2. Make a factor from the `Rooms` variable directly (3 classes) and from the `Age` variable by cutting the range into three equal parts. Rename the `Cent.Heat` factor for convenience.

   ```
   > rms <- factor(Rooms)
   > age <- factor(cut(Age,3))
   > cht <- CentralHeating
   ```

   Note, instead of explicitly creating these variables and storing them temporarily in the workspace, we could have used the `transform` function to create a new data frame with extra columns corresponding to these objects.

   ```
   > houseT <- transform(house,rms=factor(Rooms),
     age=factor(cut(Age,3)),cht=CentralHeating)
   ```

3. Find two-way frequency tables for each pair of factors. Although the table is of only a small number of frequencies, does their appear to be any two way association?

4. Use Pearson χ^2 statistic to test for association in any of the three tables you choose.

   [The statistic is defined as follows:

   $$\chi^2 = \sum_i \sum_j \frac{(F_{ij} - E_{ij})^2}{E_{ij}}$$

   If the two classifying factors are A and B, then F_{ij} is the frequency of class (A_i,B_j) and E_{ij} is the estimated expected frequency. The F_{ij}'s and E_{ij}'s may be calculated simply as follows:

   ```
   > Fij <- table(A,B)
   > Eij <- outer(table(A),table(B))/sum(Fij)
   ```

Look up the `outer` function in the help documents.]

5. Using the `help` facility look up the functions `chisq.test` and `fisher.test`.

   Use the first to check your answer from the previous question and the latter to check how well the χ^2 test approximates the Fisher exact test in this instance.

6. Detach the house data and clean up.

The Anorexia Data

The `anorexia` data frame gives information on three samples of young female anorexia patients undergoing treatment. There are three groups, namely *Control*, *Cognitive Behavioural treatment* and *Family treatment*, as given by the factor `Treat`. For each patient the pre-study weight and post-study weight, in pounds, are given as variables `Prewt` and `Postwt` respectively.

1. Select the control patients and test the hypothesis of no change in weight over the study period. Compare the results of a paired t-test and a two-sample t-test:

```
> attach(anorexia[anorexia$Treat=="Cont",])
> t.test(Prewt,Postwt,paired=T)
> t.test(Prewt,Postwt)
> detach()
```

2. Use a coplot to compare pre-weight with post-weight for the three treatment groups. Inside the panels, plot the points as well as the least squares line.

```
> attach(anorexia)
> panel.fn <- function(x,y,...){
      points(x,y,pch=3)
      abline(lm(y~x),col="blue")
  }
> coplot(Postwt ~ Prewt | Treat,panel=panel.fn)
```

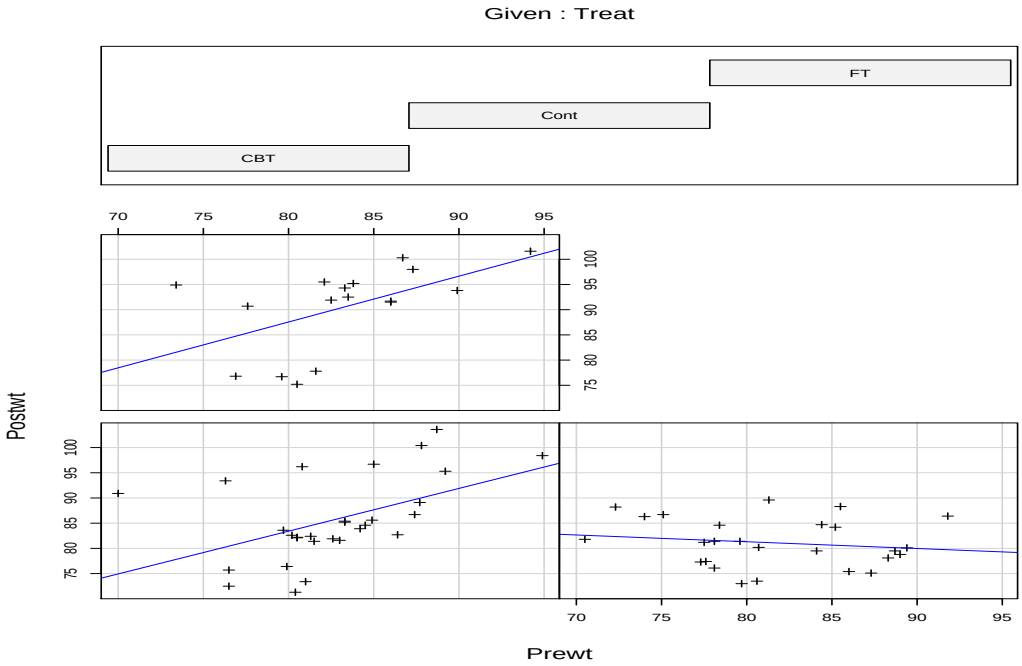


Figure 8: Scatterplot of anorexia data conditioned by treatment. Each panel is overlayed with a least squares line.

Lab 3: Elementary Graphics

Scatterplots and Related Issues

1. Attach the `AnimalsALL` data frame that you made at the end of the last laboratory session. (The `Animals` data frame will do if you do not get that far). Also open a small graphics window.

```
> attach(AnimalsALL}
> windows()
```

2. Plot `brain` against `body`, firstly using ordinary axis scales and then using logarithmic scales in both directions, and compare the results. Put in explicit axis labels:

```
> plot(body,brain, xlab="Average body weight (kg)",
       ylab="Average brain weight (gm)")
> plot(body,brain, xlab="Average body weight (kg)",
       ylab="Average brain weight (gm)",log="xy")
```

3. Using `identify` discover which animals lie above or below the notional regression line of the log-transformed values.

```
> identify(body,brain,row.names(AnimalsALL))
```

4. Now plot both variables log-transformed and include on the plot both the ordinary least squares and the robust *least trimmed squares* regression lines, using a different line type for the second.

```
> plot(log(body),log(brain),pch=4)
> abline(lsfit(log(body),log(brain)),lty=1)
> abline(ltsreg(log(body),log(brain)),lty=2)
```

   Why are the two so different do you think?

5. Next we will insert a legend to identify ordinary least squares (OLS) and least trimmed squares (LTS) regression lines by line type. The legend box should be placed in an otherwise empty part of the plot and a convenient way of doing this interactively is to use the `locator` function in conjunction with the mouse.

```
> legend(locator(1),legend=c("OLS line","LTS line"),lty=1:2)
```

At this point you should place the mouse in the plot window, locate the point where you want the top left hand corner of the legend box to appear and click with the left button.

Student Survey Data

The data frame `survey` contains the results of a survey of 237 first-year Statistics students at Adelaide University.

1. Attach the data frame for later use, and have a look at the help file for this data frame. For a graphical summary of all the variables, type

   ```
   > plot(survey}
   ```

   Note that this produces a dotchart for factor variables and a normal scores plot for the numeric variables.

2. One component of this data frame, `Exer` is a factor object containing the responses to a question asking how often the students exercised. Produce a barplot of these responses using `plot()`.

   The `table()` function when applied to a factor object returns the frequencies of each level of the factor. Use this to create a pie chart of the responses with the `pie()` function. Do you like this better than the bar plot? Which is more informative? Which gives a better picture of exercise habits of students? The `pie()` function takes an argument `names=` which can be used to put labels on each pie slice. Redraw the pie chart with labels.

   (**Hint:** use the `levels()` function to generate the labels.)

   You could also add a legend to identify the slices using the `locator()` function to position it. The function call to do this will be something like

   ```
   > legend(locator(1), legend = ..., fill=1:3)
   ```

3. You might like to try the same things with the `Smoke` variable, which records responses to the question *How often do you smoke?* Note that `table()` and `levels()` ignore missing values. If you wish to include non-respondents in your chart use `summary()` to generate the values and `names()` on the summary object to generate the labels.

4. Is there a relationship between pulse rate and exercising? Create boxplots of `Pulse` for each level of `Exer` with the command

   ```
   > plot(Exer,Pulse}
   ```

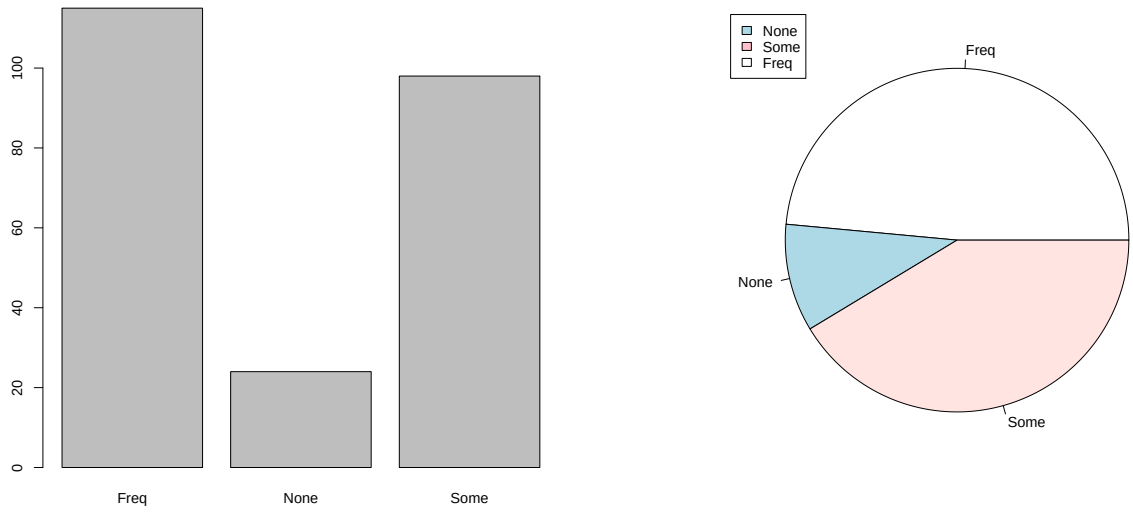


Figure 9: Two representations of the `Exer` frequencies

Does there appear to be a relationship? You may wish to test this formally with the `aov` command. What other relationships can you find in these data?

The Swiss Banknote Data

In an effort to detect counterfeits, the dimensions of 100 known fake Swiss banknotes and 100 supposedly legal notes were measured. The length of the note and its diagonal were measured along with the length of each side.

This dataset is located in the `alr3` library. To access a copy, you must first attach the library and then load the dataset.

```
> require(alr3)
> data(banknote)
```

1. Attach the data frame and plot the variables against each other in a scatterplot matrix:

   ```
   > attach(banknote)
   > pairs(banknote)
   ```

   This is not very informative since legitimate and fake notes are not separated. To separate them we need to do something more specific within each panel with a

panel function.

```
> pairs(banknote[,-1],panel=
    function(x,y,fake){
      xy <- cbind(x,y)
      points(xy[fake==0,],pch=15)
      points(xy[fake==1,],pch=0)
    }, fake=Y)
```

Note that if the panel function has more arguments than x and y, these may be supplied as named arguments to `pairs`, usually at the end.

Which two variables seem best to discriminate the counterfeit and legal notes?

2. Generate a co-plot of these two variables, given `Y`.

```
> coplot(<var1> ~ <var2> | Y,data=banknote)
```

Does your co-plot suggest that some of the *legal* notes may in fact be undetected.

Lab 4: Manipulating Data

Birth Dates

Repeat the birth date example discussed in the lecture for your own birth date.

The Cloud Data

In this exercise we do some polynomial regression calculations from scratch using raw matrix manipulation and verify the results using R tools. (A side effect is to reinforce the value of learning to use those tools effectively!)

1. The cloud data is not part of the MASS library but is available as text file and can be read in using `read.table`. After reading in the data, look at the data.

```
> cloud <- read.table("cloud.txt",header=T)
> cloud
```

2. Attach the data and for ease of typing make two new variables, x and y equal to the I_s% and cloud point vectors respectively. Let n be the number of observations, for future reference.

```
> attach(cloud)
> x <- Ispc
> y <- Cloudpt
> n <- length(x)
```

3. Mean correct the x vector and construct a model matrix for a cubic regression of y on x, (where the powers are of the mean corrected x-vector). Include the constant term, of course.

```
> xmc <- x-mean(x)
> X <- cbind(1,xmc,xmc^2,xmc^3)
> X
```

4. The regression coefficients are $b = (X'X)^{-1}X'y$. First calculate the matrices. Here are two alternative ways of calculating $X'X$.

```
> XX <- t(X) %*% X    # the literalist way
```

```
> XX <- crossprod(X)  # more efficient alternative
```

The $X'y$ matrix is, similarly,

```
> Xy <- crossprod(X,y)  # 2 argument form
```

Now calculate the regression coefficients in `b` using `solve()`.

5. Calculate
   - the fitted values, $f = Xb$,
   - the residuals, $r = y - f$,
   - the residual mean square $s^2 = (\sum_i r_i^2)/(n-4)$ and
   - the variance matrix of the regression coefficients, $s^2(X'X)^{-1}$, which you call, say `Vb`.
6. The square roots of the diagonal entries of `Vb` give the standard errors, and the ratio of the regression coefficients to these give the t-statistics.

```
> se <- sqrt(diag(Vb))
> ts <- b/se
```

7. Finally, arrange the results in a *comprehensible* form and print them out:

```
> R <- cbind(b,se,ts,1-pt(ts,n-3))
> rnames <- c("Constant","Linear","Quadratic","Cubic")
> cnames <- c("Coef","Std. Err.","t-stat","Tail prob.")
> dimnames(R) <- list(rnames,cnames)
> R
```

8. Check the calculations by the standard tools, for example

```
> fm <- lm(y~1+xmc+I(xmc^2)+I(xmc^3))
> summary(fm)  # check the table
> b-coef(fm)   # check for numerical discrepancies
```

9. Detach data frame and clean up

The Longley Data

This is a famous data set used in a benchmark study of least squares software accuracy. It comes as a system dataset and `?longley` will give more information. Our purpose is to study this data, possibly seeing why it is so commonly used to check out least squares calculations. The data set is too small to study effectively by graphical methods, so we will stick to calculations.

1. Find the means and variances of each variable. Do this any way you like, but compare your answer with the results of

   ```
   > mns <- apply(longley,2,mean)
   > mns
   > vrs <- apply(longley,2,var)
   > vrs
   ```

2. Compare the results of the two commands

   ```
   > apply(longley,2,var)
   > var(longley)
   ```

   What is going on?

3. Find the correlation matrix between all variables. (`cor()`)

4. Find the singular value decomposition of the mean corrected X-matrix and print out the singular values. (`svd()`)

   The simplest way to calculate and subtract the sample mean from each column of a matrix is to use the scale function:

   ```
   > Xc <- scale(longley[,-7],scale=F)
   ```

   What happens if we leave off the `scale=F` argument? Investigate. Note that `scale` works either with matrices or data frames.

   Another way is to use the model fitting functions:

   ```
   > Xc <- resid(lm(longley.x~1))
   ```

   A different way, that keeps the mean vector as well, is as follows

   ```
   > xb <- apply(longley.x,2,mean) # calculate the 6 means
   > dm <- dim(longley.x)
   > Xc <- longley.x - matrix(xb,dm[1],dm[2],byrow=T) # correct
   ```

Convince yourself why all three should work and verify that they lead to the same result.

The ratio of the largest to the smallest singular value is called the *condition number* of the matrix. Calculate it and print it out.

Lab 5: Classical Linear Models

H O Holck's cats data, revisited

As noted earlier, the data frame `cats` gives the sex heart and body weights of 144 hapless domestic cats used in animal experiments. We will explore in a simple way the relationship between heart and body weight and check for differences between sexes.

1. Use a coplot to show the relationship of heart weight to body weight separately for the two sexes. Include the least squares line as a guide.
2. Fit separate regressions of heart weight on body weight for each sex. Also fit parallel regression lines and a single regression line for the two sexes. Test the models and comment.

```
> cats.m0 <- aov(Hwt~Sex/Bwt,data=cats) # separate
> cats.m1 <- aov(Hwt~Sex+Bwt,data=cats) # parallel
> cats.m2 <- aov(Hwt~Bwt,data=cats)     # identical
> anova(cats.m2,cats.m1,cats.m0)
```

3. Consider an alternative model that postulates that heart weight is proportional to body weight. This leads fairly naturally to a model of the form

$$log(\mathrm{Hwt}) = \beta_0 + \beta_1 \log \mathrm{Bwt} + \epsilon$$

with a natural null hypothesis of $\beta_1 = 1$. Explore various models with the log transformed data and comment.

Cars Dataset

Read the details of the data frame `Cars93`.

Build a regression equation for predicting the miles per gallon in city travel using the other variables (except `MPG.highway`, of course).

Check your final suggested equation with suitable diagnostic plots. If there appears to be variance heterogeneity (which is suspected) repeat the construction using either a log-transformed or reciprocal response variable.

Notice that the reciprocal transformation leads to an easily appreciated quantity, the *gallons per mile* used in city travel.

The Painters Data

Read the description of the `painters` data either from the notes or using `?painters` within R.

1. To see how the schools differ on each of the characteristics, plot the data frame using the method appropriate to a design:

   ```
   > plot.design(painters
   ```

   This will give a series of four plots that should be understandable.

2. Perform a single classification analysis of variance on each variable on school.

3. **Optional** Find the matrix of residuals for the four variables and hence the *within school* variance and correlation matrices.

Lab 6: Non-Linear Regression

The Stormer Viscometer Data

1. Work through the material covered in the lecture notes for fitting a non-linear regression model to the stormer dataset.
2. As a graphical challenge, display the fitted regression surface and the fitted points on a perspective diagram. Try also a contour diagram and an image plot.

The Steam Data

The steam data (data frame `steam`, see the online help documents) has (at least) two sensible nonlinear regression models, depending on whether the error is taken as additive of multiplicative on the original scale. For an additive error model it becomes

$$P = \alpha \exp\{\frac{\beta t}{\gamma + t}\} + E$$

and for a multiplicative error model:

$$\log P = \log \alpha + \{\frac{\beta t}{\gamma + t}\} + E^*$$

1. Devise a suitably ingenious method for finding initial values for the parameters. For example, plot `log(Press)` against `Temp/(g+Temp)` for some value of `g`, say 1. Then adjust `g` by either doubling or halving until the plot seems to be an approximately straight line.

   (This leads to an initial value somewhere near $\hat{\gamma}_0 = 200$.)
2. Fit both models and compare
   (a) The parameter estimates, and
   (b) The fitted values and simple pointwise confidence regions for them

   on the original scale. Comment

Lab 7& 8: Generalized Linear Models and GAMs

Snail Mortality Data

The snail mortality dataset consists of and experiment conducted on a group of 20 snails, which were held for periods of 1, 2, 3 or 4 weeks in carefully controlled conditions of temperature and relative humidity. Two species of snail were examined: A and B.

1. Read the help file on the snails data and attach the data frame.
2. Coerce the species variable into a factor and create a suitable response variable and call it `Y`.

   ```
   > snails$Species <- as.factor(Species)
   > snails$Y <- cbind(Deaths,20-Deaths)
   ```

   Optional: Try using the `transform` function to create new variables for `Species` and `Y`.
3. Fit separate linear models for the two species.

   ```
   > fm <- glm(Y~Species/(Temp+Rel.Hum+Exposure),
      family=binomial, data=snails, trace=T)
   ```
4. Now determine whether the response surfaces for the two species should be considered parallel.

   ```
   > fm0 <- update(fm,
      ~ Species+Temp+Rel.Hum+Exposure)
   > anova(fm0,fm,test="Chisq")
   ```
5. Produce some diagnostic plots and assess the fit of the model.

The Janka Data

1. Fit a locally weighted regression to the Janka data and compare it with the fitted curve from the model fitted when last you used the Janka data. (Remember, the janka data is located in a `.csv` file.)

   ```
   > fma <- gam(Hardness ~ lo(Density),data=janka)
   ```

2. Experiment with the `span=` and `degree=` parameters of the `lo()` function to see how (if anything much) it affects the characteristics of the fitted curve in this case.
3. Compare this fitted curve with a *local regression model* obtained by

```
> fmb <- loess(Hardness ~ Density,data=janka)
```

The Birth Weight Data

Look at the `birthwt` data frame. Build up a model (as done in the lectures) for estimating the probability of low birth weight in terms of the other predictions (excluding actual birth weight, of course).

See how the picture changes if *low* birth weight is defined differently, that is at some other truncation level. At the moment, `low` is an indicator of `birthwt` for values less than 2.5kg.

Lab 9: Advanced Graphics

Graphics Examples

Work through the material covered in the lecture notes. Make sure you use the Tinn-R editor for submitting scripts. Define your own colour palettes and view the results.

The Akima Data

1. The `akima` dataset is located in the `akima` library. To access the data

   ```
   > require(akima)
   > data(akima)
   ```

   Read about the akima data which we will be using for 3D graphs.
2. Interpolate the data using the `interp` function. Get help on the function so you understand its usage.
3. Produce a contour, image and perspective mesh plot of the data. Investigate setting up your own colour scheme.

Heights of New York Choral Society Singers

1. The `singer` dataset is located in the `lattice` package. Load this package and read the help supplied for this dataset.
2. Look up the `histogram` function in the help section and see what parameters can be specified. Produce a histogram of heights broken down by voice type using the `histogram` lattice function.
3. Add to the previous plot a density
4. Investigate the *Normality* of the data for each voice type by producing a Normal scores plot. Use the `qqmath` function to create this plot.

Lab 10: Mixed Effects Models

The Rail Dataset

The `Rail` dataset is part of the `nlme` library. It contains information on travel time for a certain type of wave resulting from longitudinal stress of rails on tracks. The data consists of a factor, `Rail` that gives the number of the rail that the measurement was taken and the travel time, `travel` for the head-waves in the rail.

1. Attach the `nlme` library and read the help provided for the Rail dataset.
2. Fit a linear model to the `Rail` data of the form

$$y_{ij} = \beta + \epsilon_{ij}$$

   and look at the results. This is a simple mean model.
3. Produce some diagnostic plots. A useful plot to produce is a boxplot of residuals by rail number and comment on the plot. Is there any variability between rails?
4. Now fit a fixed effects model of the form

$$y_{ij} = \beta_i + \epsilon_{ij}$$

   This model fits a separate fixed effects term to each rail. (Do not fit the intercept). Examine the results and pay particular attention to the residual standard error.
5. Produce a residual plot for this new model and compare the results.
6. Although this model accounts for the effects due to rails, it models the specific sample of rails and not the population of rails. This is where a random effects model may prove useful.

   A random effects model of the form

$$y_{ij} = \bar{\beta} + (\beta_i - \bar{\beta}) + \epsilon_{ij}$$

   treats the rail effects as random variations around the population mean. Try fitting this model using `lme` and examine the results:

```
> fitRail.lme <- lme(travel~1,data=Rail,random=~1|Rail)
> summary(fitRail.lme)
```

What does the fitted model tell us? Produce plots of the residuals from the fitted model using the `plot` function. Is this model any better? More appropriate?

The Pixel Dataset

The `Pixel` dataset is another dataset in the `nlme` library that contains information on the pixel intensities of CT scans of dogs over time. Read the help section for more information about this dataset.

1. Using the functions in the lattice (trellis) library, produce an exploratory plot that shows the panel intensity versus the time (in days) of the `Pixel` data, for each dog and for each side.
2. It seems plausible to fit some random terms in the model to capture the variability between dogs and sides within dogs (there is some variation apparent between sides but not much).

   Fit a mixed effects model using `lme` with a quadratic term for `Day` in the fixed effects part of the model and random terms for `Dog` and `Side` within `Dog`.

   ```
   > pixel.lme <- lme(pixel~day + I(day^2),data=Pixel,
      random=list(Dog=~day,Side=~1))
   > intervals(pixel.lme)
   ```

   Comment on the results.
3. Produce a plot of the predicted values for each dog/side combination.

   ```
   > plot(augPred(pixel.lme))
   ```

   Comment on the plot.

Lab 11: Programming

Elementary Programming Examples

The trick with programming is to start with the simplest prototype version of the function, test is as you go, and gradually incorporate the more general and flexible features you need. A second tip is to write simple functions to do single, special tasks reliably, and gradually build the final function from these building blocks.

These two principles are sometimes called *bottom up design* and *modular design* respectively.

1. Your task is to write a function `T.test` that will perform a two sample t-test, similar to the job done by the system function `t.test`.

   (a) The formula for the statistic is

   $$t = \frac{\bar{y}_1 - \bar{y}_2}{s\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

   where n_1 and n_2 are the two samples sizes, $\bar{y}_1$ and $\bar{y}_2$ are the two sample means, and s is the square root of the pooled estimate of variance, s^2, given by

   $$s^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{(n_1 - 1) + (n_2 - 1)}$$

   where s_1^2 and s_2^2 are the two sample variances. Suppose the function initially has the header

   ```
   T.test <- function(y1,y2){ ...
   }
   ```

   i. First find n_1 and n_2.

   ii. Next calculate the numerator of t

   iii. Then calculate the pooled variance and hence s

   iv. Finally, calculate the value of t and return that as the value of the function

   (b) Suppose the test is two-tailed. The significance probability is then defined as $1 - 2\Pr(|T| > t)$ where $T \sim t_{n_1+n_2-2}$. In R this is calculated as

   ```
   2*(1-pt(abs(t),n1+n2-2))
   ```

Amend your function so that it calculates this quantity and returns as its value a list with two components: the t-statistic and the significance probability.

(c) Include a preliminary test of equality of variances. The significance probability for this test may be found either by the calculation (which is easy enough) or as `var.test(y1,y2)$p.value`. Perform the test early in the problem and issue a warning if the significance probability is less than 0.05.

(d) Sometimes it may be useful to have a graphical representation of the samples as well as the numerical result. Allow the user optionally to request boxplots with an additional logical parameter `boxplots`:

```
T.test <- function(y1,y2,boxplots=F){
   ...
   if(boxplots) {
      f <- factor(rep(paste(''Sample'',1:2),c(n1,n2(((
      y <- c(y1,y2)
      plot(f,y)
   }
   ...
}
```

(e) Finally, it might be useful to allow the user to specify the two samples as a two-level factor and vector rather than as two separate sample vectors. Modify the function to allow this alternative call sequence:

```
T.test <- function(y1,y2,boxplots=F){
   if(is.factor(y1)) { # y1=factor, y2=all y's
      l <- levels(y1)
      tm <- y2[y1==l[1]]
      y2 <- y2[y1==l[1]]
      y1 <- tm
    }
    ... # proceed as before
}
```

(f) Check for all likely error conditions and issue either warnings or stops to aid the user.

(g) As a last check, test the program on the `Cars93` data, comparing the Price levels for the two car origins (omitting large cars).

2. The Box and Cox power family of transformations was covered in one of the lectures. It is defined as

$$f(y,\lambda) = \begin{cases} (y^\lambda - 1)/\lambda & \text{if } \lambda \neq 0 \\ \log y & \text{if } \lambda = 0 \end{cases}$$

Write a function to calculate such a transformation. The function should be of two arguments, `y`: a vector of data values and `lambda`: a scalar specifying the exponent. Stop with an error message if any `y` value is negative or zero.

(a) For the Janka hardness data, consider a second degree polynomial regression of `y=Hardness` on `x=Density`. Notice how with untransformed y the residuals show a distinct pattern of increasing variance with the mean. Consider alternative regressions with a Box and Cox power transformed y using powers $\lambda = 1, \frac{1}{2}, \frac{1}{4}$ and 0. Which model would you prefer if it were important to operate in a y scale where the variance were uniform?

(b) With the `Cars93` data we noticed that the reciprocal of `MPG.city` was in some sense simpler to use as a response than the variable in the original scale. This is effectively a Box and Cox power transform with $\lambda = -1$. Consider other values for λ, say $\lambda = 0, -\frac{1}{2}, -1$ and $-\frac{3}{2}$ and recommend a value accordingly. Use the regression variable `Weight` and the factors `Type` and `Cylinders` as the predictors, if necessary.

Round Robin Tournaments

Write a function of one integer variable to generate a round robin tournament draw. One possible algorithm is as follows:

1. Begin with an arbitrary pairing of teams. If there is an odd number of teams increase them by a *team 0* representing a bye.
2. Write the original pairings as two columns and cycle the teams in the way shown in the diagram: If there are n teams the tournament draw is complete after $n-1$ cycles.
3. Return the result as an $n/2 \times 2 \times (n-1$ array of integers.
4. Give the array a class, say `robin` and write a print method that will print the result in a convenient way.
5. The functions `round.robin()` and `print.round.robin` provide a solution

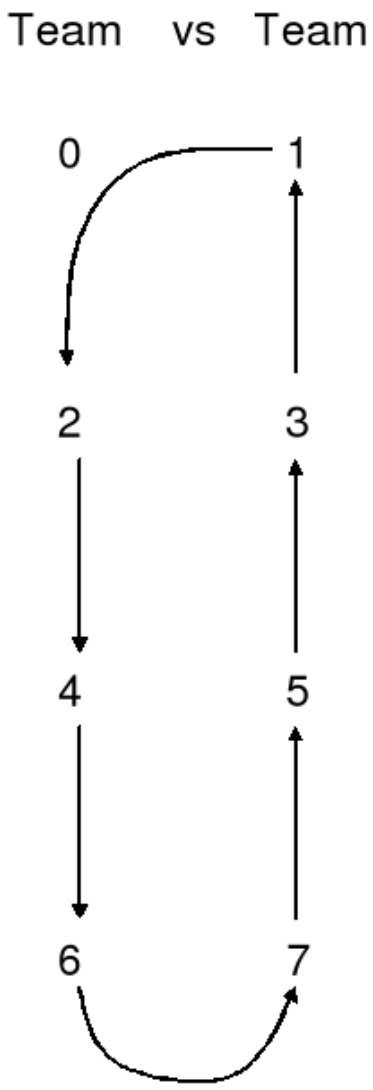


Figure 10: Cycle method for round robin tournament draws

Lab 12: Neural Networks

The Rock Data

1. The rock data is part of the `datasets` package. Call up help on this dataset.
2. Attach the rock dataset to the current session and divide the area and perimeter variables by 10000. Form a new dataset that has the transformed area, perimeter and the shape variable included. Call this dataset `rock.x`.
3. Fit a neural network using the `nnet` function. Remember to attach the neural network library.

```
> rock.nn <- nnet(rock.x,log(perm), size=3, decay=1e-3,
   linout=T,skip=T,maxit=100)
> summary(rock.nn)
```

4. Investigate the fit of the model

The Crab Data

We now consider fitting a neural network to a classification problem: classifying the sex of crabs. Get up the help on the `crabs` dataset and read about it.

1. Let's start by fitting a generalized linear model to the data. We first need to transform the explanatory variables `FL`, `RW`, `CL`, `CW`, and `BD` on to the log scale.

```
> dcrabs <- log(crabs[,4:7])
> dcrabs <- cbind(dcrabs,sex=crabs$sex)
```

2. Now fit a logistic regression model to the data and look at the results.
3. Produce some predictions using the `predict` function with `type="response"`.
4. **Optional**: Try to perform a cross-validation study to assess the performance of the model. Hint: Try using the `subset` argument in the glm function.
5. Now fit a neural network to the crabs dataset and compare the results.

```
> cr.nn <- nnet(dcrabs,crabs$sex=="M",size=2,decay=1e-3,
    skip=T,entropy=T,maxit=500)
```

6. **Optional**: Repeat the cross-validation study but base it around the neural network. How do the two models compare?

Lab 13& 14: Classification and Regression Trees

The Crab Data Revisited

In the previous laboratory session, the crab data was used to fit a logistic regression using the `glm` function and a neural network.

Try fitting a classification tree to the crab data and compare with previous models.

The Cuckoo Data

The `cuckoo` data shows the lengths and breadths of various cuckoo eggs of the European cuckoo, *Cuculus canoris* classified by the species of bird in the nest of which the egg was found.

1. The data is located in an Excel spreadsheet. Attach the `RODBC` library and read in the dataset.
2. Reduce the data frame to a smaller one got by excluding those eggs for which the host species was unknown or for which the numbers of eggs found were less than 5.
3. Working with the reduced data frame, develop a classification tree using the `rpart` package for predicting the host species from the egg dimensions, to the extent to which this seems to be possible.
4. Try and amend the code provided in the lecture for partitioning the tree for this data.
5. Use the tree model to predict the host species for the unnamed host eggs of the full data frame.

The Student Survey Data

The survey data comes from a student survey done to produce *live* data for the first year statistics class in the University of Adelaide, 1992. The variables recorded included `Sex`, `Pulse`, `Exer` (an exercise level rating), `Smoke`, `Height` and `Age`.

There are missing values in the data frame, so be sure to use appropriate actions to accommodate them.

1. Fit a naive linear model for `Pulse` in terms of the other variables listed above.

2. Construct a regression tree for `Pulse` in terms of the other variables listed. Compare the results with that of the linear model. Comment.
3. Prune the tree to what you consider to be a sensible level and compare the residuals with those of the linear model, for example, using histograms.
4. Cross-validation with this data often suggests using a tree with just one node! That is, accurate prediction of pulse rates from the other variables is most reliably done using the simple average. However, cross validation is itself a stochastic technique. Try it and see what is suggested.
5. **Optional**: Try bagging out on this tree and see what improvements are made with regards to prediction.