"Practice with Data Mining Tools"

John Maindonald

June 13, 2009

Contents

Preliminaries

See the document "Basic Ideas and Tools for Data Miners". Points made in that document can be summarized under the headings below.

Software Issues: R versus Matlab: I am not in a position to supply Matlab code. Those who are thoroughly familiar with Matlab may be able to do equivalent computations using Matlab. Otherwise the choice is either to work through the computations using the R system (instructions are relatively detailed, or to be content with studying the computer output and associated commentary.

Issues of Purpose, Interpretation of Parameters, and Generalization

Key points from the document "Basic Ideas and Tools for Data Miners" are:

- Note the importance of settling on a clear purpose.
- There is a broad distinction between projects where it is predictive accuracy that matters, and projects where the aim is scientific or other insight. The aim of scientifc insight, commonly based on parameter estimates, offers a potential for misinterpretation that is largely absent when the aim is prediction.
- Key issues for any use of the data are:
  - How widely do results apply?
  - How widely applicable is whatever can be learned from the data? (Will it apply to a city that is different from those that generated the sample data? Will it apply to a different time – next year, or two years' time?)
- The source/target is important for assessing how widely results apply. When data are observational (rather than experimental), the target population to which it is hoped to apply results is typically distinct, usually in time and perhaps also in space or some other manner, from the source population from which the data were obtained.

Analysis Types and Tools

Points made under this heading are:

- The data mining literature has been distinguished by a preference for specific "data mining" methodologies.
- Model and variable selection can be huge issues, with wide scope for over-optimistic assessments.
- The common tools for assessing predictive accuracy have been empirical – training/test methodology, cross-validation, and bootstrap sampling.

The Laboratory Exercises

There will not be time to work through all these exercises. If the discussion in Laboratory Notes I makes sense without working through the calculations, and if practice with R is not an issue, move on immediately to laboratory exercises II.

The laboratory notes divide up thus:

Laboratory Exercises I may serve the purposes:

- For those who want to experiment with the use of R, they give an introduction to graphics and model fitting.
- They are designed to assist understandings or interpretations that may be available when transformation to a logarithmic scale leads to a relatively simple form of relationship, here a line.

Laboratory Exercises II: Traps for the interpretation of model parameters are an issue both for regression models with a continuous outcome and for classification models. Two data sets are used, one small dataset where the outcome is a continuous variable, and other largish ($\simeq$26,000 rows) data set with a binary (0/1) outcome.

Laboratory Exercises III: These give practice in using linear discriminant analysis and random forests, with the diabetes and forensic glass datasets.

Laboratory exercises IV: Here the focus is on the source/target distinction. What clues can be extracted from data (the "source"), believed to be relevant to one or other target population, that may help in assessing whether the belief is justified?

Laboratory exercises V: Where there is extensive variable selection, there is huge potential for finding spurious relationships. This is true both for data with a continuous outcome, and for data where the outcome is categorical. The cross-validation methodology is introduced as a way to get a measure of predictive accuracy that is not biased from the effects of model and/or variable selection.

Further exercises: A further exercise may be provided that gives practice with the use of the random forest methodology. This methodology is strongly in the style that is popular in the data mining literature. It can also be remarkably effective.

Part I
Time Versus Distance for Road and Track Data

Motivations: There are (at least) two possible motivations:

- Those who want to experiment with the use of the R system should find these exercises useful as an introduction to graphics and model fitting.
- These exercises, and the accompanying discussion, are desgned to assist understanding or interpretations that may be available when transformation to a logarithmic scale leads to a relatively simple form of relationship (here a line).

Use of R packages For plotting graphs, there is extensive use of the function `xyplot()`, from R's *lattice* package. The primary function for fitting linear models is `lm()`, where the `lm` stands for linear model. Here it will be used for a very simple form of model – the straight line model.

1 Time versus Distance – A Graph

The data that are used here are world track and road record times, as at 9th August 2006. Data are from the web page `http://www.gbrathletics.com/wrec.htm`. Data are in the dataset `worldRecords` in the `DAAG` package. To make the data available to your R session, type:

```
> library(lattice)
> library(DAAG)  # The worldRecords data are included with the DAAG package
> ## Now check what columns are included
> str(worldRecords)

'data.frame':        40 obs. of  5 variables:
 $ Distance   : num  0.1 0.15 0.2 0.3 0.4 0.5 0.6 0.8 1 1.5 ...
 $ roadORtrack: Factor w/ 2 levels "road","track": 2 2 2 2 2 2 2 2 2 2 ...
 $ Place      : chr  "Athens" "Cassino" "Atlanta" "Pretoria" ...
 $ Time       : num  0.163 0.247 0.322 0.514 0.72 ...
 $ Date       :Class 'Date'  num [1:40] 12948 4889 9709 11040 10829 ...
```

The following gives a plot of `Time` against `Distance`, using the data set `worldRecords`, as in Figure 1:

```
> pltA <- xyplot(Time ~ Distance,
+                groups=roadORtrack,
+                data=worldRecords,
+                auto.key=list(columns=2))
```

Actually, one does not yet have the plot. For that, type:

```
> print(pltA)
```

Or, if you are working from the command line, it is enough to do:

```
> xyplot(Time ~ Distance,
+        groups=roadORtrack,
+        data=worldRecords,
```

```
+         xlab="Distance (km)",
+         ylab="Time (min)",
+         auto.key=list(columns=2))
```

The output from `xyplot()` goes to the command line, which causes the object to be plotted.

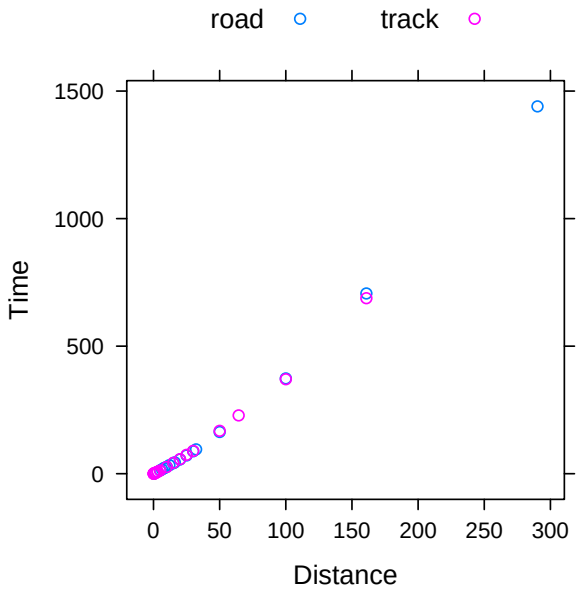

Figure 1: World record times versus distance, for field and road events. The left panel is for the raw data, while the right panel plots log(Time) against log(Distance).

Here again is the code:

```
> xyplot(Time ~ Distance,
+         groups=roadORtrack,
+         data=worldRecords,
+         xlab="Distance (km)",
+         ylab="Time (min)",
+         auto.key=list(columns=2))
```

Note: In printing the graph for inclusion in this document, there has been a further refinement, so that the text and tick marks are scaled down to a suitable size for the size of the graph on the printed page:

```
> ## Reduce text and point size by a factor of 0.8
> print(update(pltA, cex=0.8, scales=list(tck=0.8)))
```

Clearly, time is not proportional to distance.

Figure 2 examines the use of a power law relationship, albeit where the curvature is much stronger than in Figure 1. It is, actually equivalent to straight-line plot in which both scales are logarithmic.

The equation $t = As^b$ implies that

$$\log(t) = log(A) + b\log(s) = a + b\log(s), \text{ where} a = \log(a).$$

and hence that

$$\frac{dt}{t} = \frac{ds}{s}.$$

Figure 2 grossly exaggerates ($b = 2.25$ in place of $b = 1.125$), for purposes of demonstration, the effect of increasing relative distance.

2 A Plot, and Fitted Line, with Logarithmic Scales

The relationship between `Time` and `Distance` is best teased out, for the `worldRecords` data, by fitting a line to a plot in which both scales are logarithmic (natural logarithms).

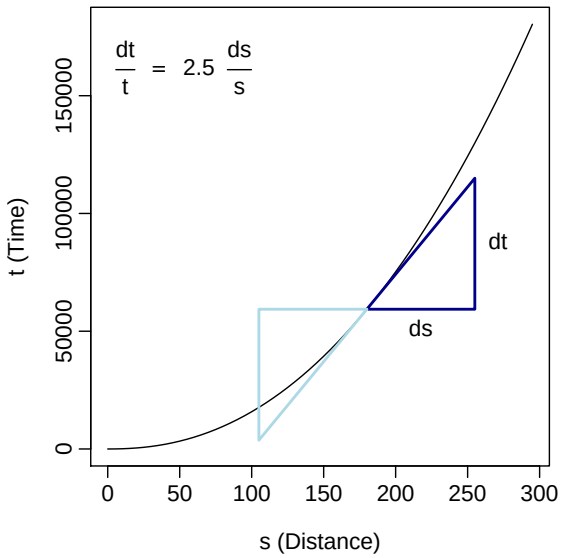

Figure 2: Plot of `Time` against `Distance`, under a scenario where `Time` is proportional to $\texttt{Distance}^{2.25}$. An exponent of 2.25 has been used in order to exaggerate the effect of increasing relative distance. The slope of the tangent line increases with s (`Distance`). For the scenario shown in the graph, $\frac{dt}{t} = 2.25\,\frac{ds}{s}$. For the `worldRecords` data, a model that has an exponent of 1.125 closely approximates the data.

For now, any difference between `road` and `track` will be ignored. Actually, it will turn out that there is no detectable difference.

```
> logplt <- xyplot(Time ~ Distance,
+                  type=c("p","r"),
+                  scales=list(log="e"),
+                  data=worldRecords)
```

Again note that, on the command line, it will be enough to type:

```
> xyplot(Time ~ Distance,
+        type=c("p","r"),
+        scales=list(log="e"),
+        data=worldRecords)
```

Now examine the line:

```
> worldRecs.lm <- lm(log(Time) ~ log(Distance), data=worldRecords)
> print(worldRecs.lm)

Call:
lm(formula = log(Time) ~ log(Distance), data = worldRecords)

Coefficients:
  (Intercept)  log(Distance)
       0.7316          1.1248
```

Check that the coefficient of $\log(Distance)$ is 1.125.

Checking Out the Line

The line is a very good fit. There are however small systematic departures from a line. The best way to see the pattern in these departures is to plot the residuals (really, the departures from a line) against either the distances or the fitted values, as in Figure 4. For this, it is highly desirable to distinguish between `road` and `track`:

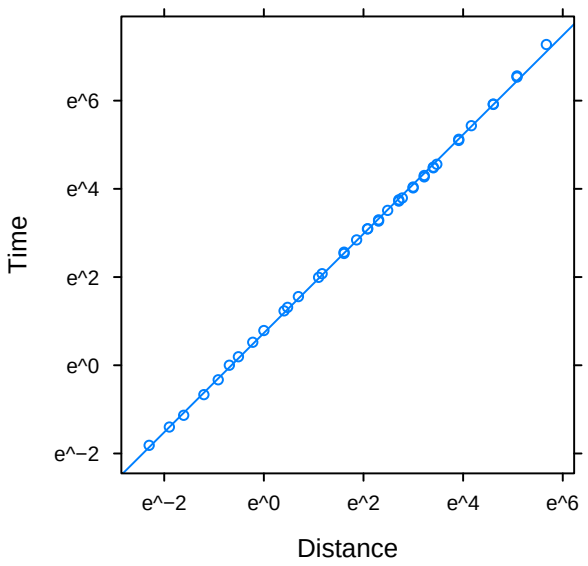

Figure 3: Regression line, fitted to the plot of log(Time) against log(Distance).

```
> xyplot(Time ~ Distance,
+        type=c("p","r"),
+        scales=list(log="e"),
+        data=worldRecords)
```

```
> yexpr <- expression("Residuals (log"[e]*"scale)")
> resplot <- xyplot(resid(worldRecs.lm) ~ log(Distance),
+                   groups=roadORtrack,
+                   ylab=yexpr,
+                   data=worldRecords)
```

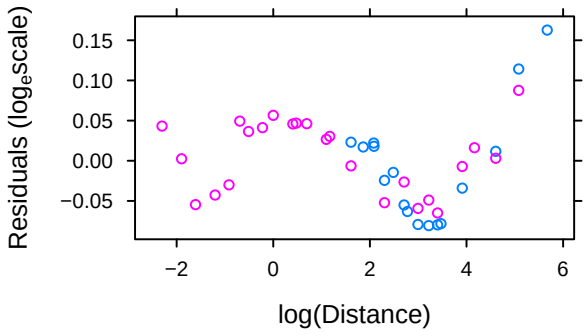

Figure 4: Residuals from regression line of log(time) against log(distance).

Some of the residuals are actually quite large. For small x,

$$\log(1+x) \simeq x.$$

(The error is, for $x << 1$, approximately $\frac{1}{2}\, x^2$. Thus, for $x = 0.14$, the error is close to 0.01, which for present purposes is of no consequence).
The largest of the residuals are thus rather more than 15%. Why did the residuals appear small in Figure 3? They are small relative to the range in the y-direction, where the range is from a time of 0.16 min to 1440 min (= 24h), i.e., variation by a factor of close to 9000.

Part II
Time vs Distance & Climb for Hill Race Data

Motivation: The analyses that will be described here demonstrate the careful thought that may be necessary, in order to arrive at meaningful interpretations of model parameters. Where scientific understanding is weak, definitive interpretation may be hazardous or impossible.

Problems of interpretation of model parameters become much more severe when there are many parameters. Analyses where there is extensive variable selection are particularly difficult, though not necessarily impossible. For example, one variable may have an effect that strongly dominates, irrespective of what over variables may be included in the model.

Two examples will be given, one where the outcome is a continuous variable, and one where it is categorical, though with just two categories.

Data: Both datasets are in the `DAAG` package.

The dataset `nihills` has record times for Northern Ireland mountain races. Data were taken from the web site `http://www.nimra.org.uk/calendar.asp`.

The dataset `nassCDS` is intended to be a random sample from all police-reported crashes in the USA, over 1997 – 2002, that met the inclusion criteria. They were accidents in which there was a harmful event (to people or property), and from which at least one vehicle is towed.

1 Data Exploration

First, get a few details of the data:

```
> str(nihills)
```

```
'data.frame':        23 obs. of  5 variables:
 $ dist    : num  7.5 4.2 5.9 6.8 5 4.8 4.3 3 2.5 12 ...
 $ climb   : int  1740 1110 1210 3300 1200 950 1600 1500 1500 5080 ...
 $ time    : num  0.858 0.467 0.703 1.039 0.541 ...
 $ timef   : num  1.064 0.623 0.887 1.214 0.637 ...
 $ gradient: num  232 264 205 485 240 ...
```

A scatterplot matrix, which plots every column against every other column and shows the result in the layout used for correlation matrices, is useful for an initial look at the data. The scatterplot matrix is a graphical counterpart of the correlation matrix.

For identifying the axes for each panel

- look along the row to the diagonal to identify the variable on the vertical axis.
- look up or down the column to the diagonal to identify the variable on the horizontal axis.

Note that the data are positively skewed, i.e., there is a long tail to the right, for all variables. For such data, a logarithmic transformation often gives more nearly linear relationships.

```
> ## Create a data frame that holds the logged data
> lognihills <- log(nihills)
> names(lognihills) <- c("ldist", "lclimb", "ltime", "ltimef")
```

```
> ## Create scatterplot matrix
> library(lattice)
> print(splom(nihills, par.settings=size10))
```

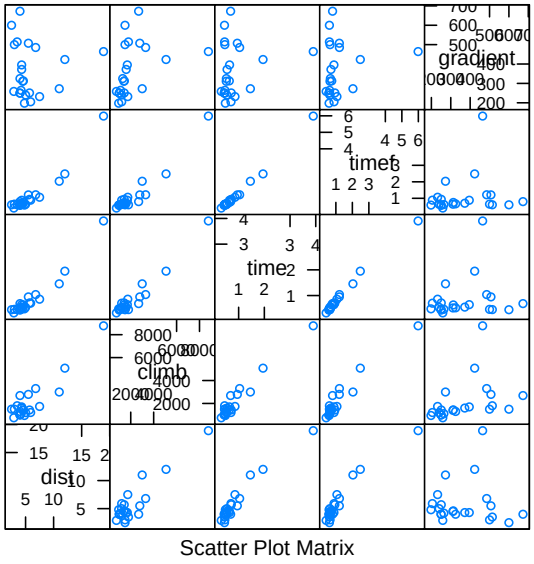

Figure 5: Scatterplot matrix for the Northern Ireland mountain racing data, with the correlation matrix given alongside. The function `cor()` takes a matrix or data frame as argument, by default giving the Pearson linear correlation.

The correlation matrix is:

```
> ## Correlation matrix
> round(cor(nihills), 2)
```

	dist	climb	time	timef	gradient
dist	1.00	0.91	0.97	0.95	0.03
climb	0.91	1.00	0.97	0.96	0.40
time	0.97	0.97	1.00	1.00	0.20
timef	0.95	0.96	1.00	1.00	0.19
gradient	0.03	0.40	0.20	0.19	1.00

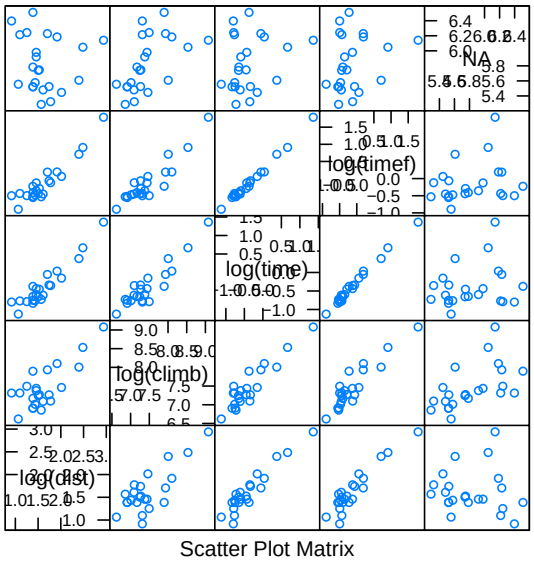

Figure 6: Scatterplot matrix for the Northern Ireland mountain racing data, with logarithmic scales.

```
> print(splom(lognihills,
+       varnames=c("log(dist)", "log(climb)",
+                  "log(time)", "log(timef)"),
+       par.settings=size10))
> round(cor(lognihills), 2)
```

	ldist	lclimb	ltime	ltimef	<NA>
ldist	1.00	0.78	0.95	0.93	-0.07
lclimb	0.78	1.00	0.92	0.92	0.57
ltime	0.95	0.92	1.00	0.99	0.24
ltimef	0.93	0.92	0.99	1.00	0.25
<NA>	-0.07	0.57	0.24	0.25	1.00

The relationships between explanatory variables, and between the dependent variable and explanatory variables, are closer to linear when logarithmic scales are used. The log transformed data are consistent with a form of parsimony that is advantageous if we hope to find a relatively simple form of model. We will see that this also leads to more readily interpretable results. Also the distributions for individual variables are more symmetric.

1.1 The regression fit

The following fits a regression plane, with logarithmic scales for all variables:

```
> lognihills <- log(nihills)
> names(lognihills) <- paste("l", names(nihills), sep="")
> lognihills.lm <- lm(ltime ~ ldist + lclimb, data=lognihills)
> round(coef(lognihills.lm),3)

(Intercept)       ldist      lclimb
     -4.961       0.681       0.466
```

This translates to:

$$\begin{aligned} \widehat{\text{time}} &= e^{3.205} \times \text{dist}^{0.686} \times \text{climb}^{0.502} \\ &= 24.7e^{3.205} \times \text{dist}^{0.686} \times \text{climb}^{0.502} \end{aligned}$$

Thus for constant `climb`, the prediction is that time per mile will decrease with increasing distance. Shorter races with the same climb will involve steeper ascents and descents; thus this seems reasonable.

A result that is easier to interpret can be obtained by regressing `log(time)` on `log(dist)` and `log(gradient)`, where `gradient` is `dist/climb`.

```
> nihills$gradient <- with(nihills, climb/dist)
> lognihills <- log(nihills)
> names(lognihills) <- paste("l", names(nihills), sep="")
> lognigrad.lm <- lm(ltime ~ ldist + lgradient, data=lognihills)
> round(coef(lognigrad.lm),3)

(Intercept)       ldist   lgradient
     -4.961       1.147       0.466
```

Thus, with `gradient` held constant, the prediction is that `time` will increase at the rate of $\text{dist}^{1.147}$. This makes good intuitive sense.

We pause to look more closely at the model that has been fitted. Does `log(time)` really depend linearly on the terms `ldist` and `log(lclimb)`? The function `termplot()` gives a good graphical indication (Figure 7).

The vertical scales show changes in `ltime`, about the mean of `ltime`. The lines show the effect of each explanatory variable when the other variable is held at its mean value. The lines, which are the contributions of the individual linear terms ("effects") in this model, are shown in gray so that they do not obtrude unduly. The dashed curves, which are smooth curves that are passed through the residuals, are the primary feature of interest in these plots. Notice that, in the plot for `ldist`, the smooth dashed line does not quite track the fitted line; there is a small but noticeable indication of curvature. Note also that until we have modeled effectively the clear trend that seems evident in this plot, there is not too much point in worrying about possible outliers. The trend can be very adequately modeled with a quadratic curve.

Now try the exercise that follows.

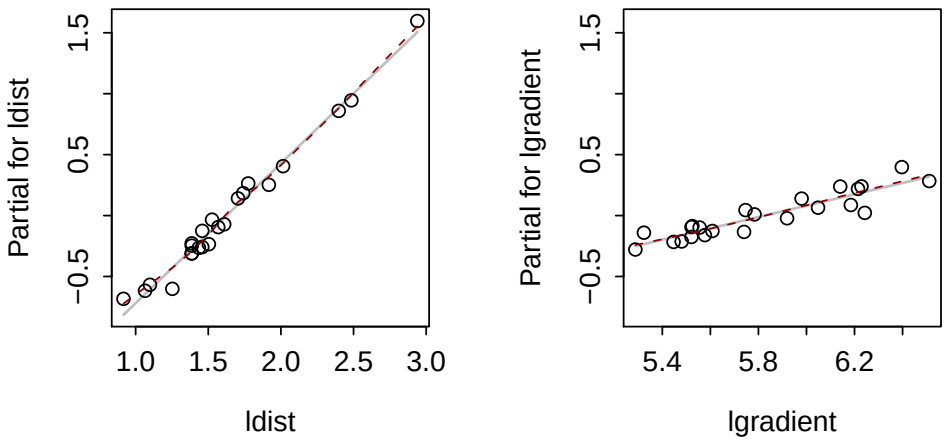

Figure 7: In these "term plots" the vertical scales in both panels show log(time), but centered to a mean of zero. The left panel shows partial residuals for `ldist`, while the right panel shows partial residuals for `lgradient`, i.e., `log(climb/dist)`. Smooth curves (dashes) have been passed through the points.

```
> par(mfrow=c(1,2))   # Ask for a 2 by 1 layout
> ## Plot the terms in the model
> termplot(lognigrad.lm, col.term="gray",
+          partial=TRUE, col.res="black",
+          smooth=panel.smooth)
```

Exercise 1
For the data frame `oddbooks` (*DAAG*),

(a) Add a further column that gives the density.

(b) Use the function `pairs()`, or the *lattice* function `splom()`, to display the scatterplot matrix. Which pairs of variables show evidence of a strong relationship?

(c) In each panel of the scatterplot matrix, record the correlation for that panel. (Use `cor()`) to calculate correlations).

(d) Fit the following regression relationships:

(i) `log(weight)` on `log(thick)`, `log(height)` and `log(breadth)`.

(ii) `log(weight)` on `log(thick)` and `0.5*(log(height) + log(breadth))`. What feature of the scatterplot matrix suggests that this might make sense to use this form of equation?

(e) Take whichever of the two forms of equation seems preferable and rewrite it in a form that as far as possible separates effects that arise from changes in the linear dimensions from effects that arise from changes in page density.

[NB: To regress `log(weight)` on `log(thick)` and `0.5*(log(height)+log(breadth))`, the model formula needed is `log(weight) ~ log(thick) + I(0.5*(log(height)+log(breadth)))` The reason for the use of the wrapper function `I()` is to prevent the parser from giving `*` the special meaning that it would otherwise have in a model formula.]

2 Which multi-way table? It can be important!

Each year the National Highway Traffic Safety Administration (NHTSA) in the USA collects, using a random sampling method, data from all police-reported crashes in which there is a harmful event (people or property), and from which at least one vehicle is towed. The data frame `nassCDS` (*DAAGxtras*) is derived from NHTSA data.[1]

The data are a sample, for the years 19997 – 2002. The use of a complex sampling scheme has the consequence that the sampling fraction differs between observations. Each point has to be multiplied by the relevant sampling fraction, in order to get a proper estimate of its contribution to the total number of accidents. The column `weight` (`national` = *national inflation factor* in the SAS dataset) gives the relevant multiplier.

Other variables than those included in `nassCDS` might be investigated – those extracted into `nassCDS` are enough for present purposes.

The following uses `xtabs()` to estimate numbers of front seat passengers alive and dead, classified by airbag use:

```
library(DAAGxtras)
> abtab <- xtabs(weight ~ dead + airbag, data=nassCDS)
> abtab
       airbag
dead          none     airbag
  alive 5445245.90 6622690.98
  dead    39676.02   25919.11
```

The function `prop.table()` can then be used to obtain the proportions in margin 1, i.e., the proportions dead, according to airbag use:

```
> round(prop.table(abtab, margin=2)["dead", ], 4)
  none airbag
0.0072 0.0039
## Alternatively, the following gives proportions alive & dead
## round(prop.table(abtab, margin=2), 4)
```

The above might suggest that the deployment of an airbag substantially reduces the risk of mortality. Consider however:

```
> abSBtab <- xtabs(weight ~ dead + seatbelt + airbag, data=nassCDS)
> ## Take proportions, retain margins 2 & 3, i.e. airbag & seatbelt
> round(prop.table(abSBtab, margin=2:3)["dead", , ], 4)
        seatbelt
airbag     none belted
  none   0.0176 0.0038
  airbag 0.0155 0.0021
```

The results are now much less favorable to airbags. The clue comes from examination of:

```
> margin.table(AStab, margin=2:3)  # Add over margin 1
        airbag
seatbelt      none    airbag
  none   1366088.6  885635.3
  belted 4118833.4 5762974.8
```

[1] They hold a subset of the columns from a corrected version of the data analyzed in the Meyer (2005) paper that is referenced on the help page for `nassCDS`. More complete data are available from one of the web pages
`http://www.stat.uga.edu/~mmeyer/airbags.htm` (SAS transport file)
or `http://www.maths.anu.edu.au/~johnm/datasets/airbags/` (R image file).

In the overall table, the results without airbags are mildly skewed (4.12:1.37) to the results for `belted`, while with airbags they are highly skewed (57.6:8.86) to the results for `belted`.

Exercise 2
Do an analysis that accounts, additionally, for estimated force of impact (`dvcat`):

```
ASdvtab <- xtabs(weight ~ dead + seatbelt + airbag + dvcat,
                      data=nassCDS)
round(prop.table(ASdvtab, margin=2:4)["dead", , , ], 6)
## Alternative: compact, flattened version of the table
round(ftable(prop.table(ASdvtab, margin=2:4)["dead", , , ]), 6)
```

It will be apparent that differences between `none` and `airbag` are now below any reasonable threshold of statistical detectability.

Exercise 3
The package *DAAGxtras* includes the function `excessRisk()`. Run it with the default arguments, i.e. type

```
> excessRisk()
```

Compare the output with that obtained in Exercise 2 when the classification was a/c seatbelt (and airbag), and check that the output agrees.
Now do the following calculations, in turn:

(a) Classify according to `dvcat` as well as `seatbelt`. All you need do is add `dvcat` to the first argument to `excessRisk()`. What is now the total number of excess deaths?
[The categories are 0-9 kph, 10-24 kph, 25-39 kph, 40-54 kph, and 55+ kph]

(b) Classify according to `dvcat`, `seatbelt` and `frontal`, and repeat the calculations. What is now the total number of excess deaths?

Explain the dependence of the estimates of numbers of excess deaths on the choice of factors for the classification.

Note: Farmer (2006) argues that these data, tabulated as above, have too many uncertainties and potential sources of bias to give reliable results. He presents a different analysis, based on the use of front seat passenger mortality as a standard against which to compare driver mortality, and limited to cars without passenger airbags. In the absence of any effect from airbags, the ratio of driver mortality to passenger mortality should be the same, irrespective of whether or not there was a driver airbag. In fact the ratio of driver fatalities to passenger fatalities was 11% lower in the cars with driver airbags.

Part III
Discriminant Methods & Associated Ordinations

Packages:`DAAGxtras`, `randomForest`

These exercises will introduce classification into three or more groups. They examine and compare two methods – linear discriminant analysis and random forests.

Linear discriminant analysis generates scores that can be plotted directly. For random forests the pairwise proximity of points, calculated as the proportion of trees for whiah the two observations end up in the same terminal node, is a natural measure of the relative nearness. These can be subtracted from 1.0 to yield relative distances, with non-metric scaling then be used to obtain a representation in a low-dimensional space. In favorable cases, metric scaling may yield a workable represention, without going the further step to non-metric scaling.

Again, it will be handy to have the function `confusion()` available.

```
> confusion <- function(actual, predicted, names=NULL,
+                       printit=TRUE, prior=NULL){
+   if(is.null(names))names <- levels(actual)
+   tab <- table(actual, predicted)
+   acctab <- t(apply(tab, 1, function(x)x/sum(x)))
+   dimnames(acctab) <- list(Actual=names,
+                       "Predicted (cv)"=names)
+   if(is.null(prior)){
+     relnum <- table(actual)
+     prior <- relnum/sum(relnum)
+     acc <- sum(tab[row(tab)==col(tab)])/sum(tab)
+   } else
+   {
+     acc <- sum(prior*diag(acctab))
+     names(prior) <- names
+ }
+   if(printit)print(round(c("Overall accuracy"=acc,
+                            "Prior frequency"=prior),4))
+   if(printit){
+     cat("\nConfusion matrix", "\n")
+     print(round(acctab,4))
+   }
+   invisible(acctab)
+ }
```

1 Discrimination with Multiple Groups

With three groups, there are three group centroids. These centroids determine a plane, onto which data points can be projected. Linear discrinimant analysis assumes, effectively, that discriminants can be represented without loss of information in this plane. It yields two sets of linear discriminant scores that can be plotted, giving an accurate graphical summary of the analysis.

More generally, with $g \geq 4$ groups, there are $max(g-1, p)$ dimensions, and $max(g-1, p)$ sets of linear discriminant scores. With three (or more) sets of scores, the function

plot3d() in the *rgl* package can be used to give a 3D scatterplot that rotates dynamically under user control.

The output from printing an lda object that is called with CV=FALSE (the default) includes "proportion of trace" information. The successive linear discriminants explain successively smaller proportions of the trace, i.e., of the ratio of the between to within group variance. It may turn out that the final discriminant or the final few discriminants explain a very small proportion of the trace, so that they can be dropped.

With methods other than lda(), representation in a low-dimensional space is still in principle possible. However any such representation arises less directly from the analysis, and there is nothing directly comparable to the "proportion of trace" information.

2 The diabetes dataset

The package *DAAGxtras* has the data set diabetes.

2.1 Linear discriminant analysis

First try linear discriminant analysis. We specify CV=TRUE, in order to obtain predictions of class membership that are derived from leave-one-out cross-validation. We then run the calculations a second time, now with CV=FALSE, in order to obtain an object from which we can obtain the discriminant scores

```
> library(MASS)
> library(lattice)
> diabetesCV.lda <- lda(clinclass ~ ., data=diabetes, CV=TRUE)
> confusion(diabetes$clinclass, diabetesCV.lda$class)
> diabetes.lda <- lda(clinclass ~ ., data=diabetes)
> diabetes.lda              # Linear discriminant 1 explains most of the variance
> hat.lda <- predict(diabetes.lda)$x
> xyplot(hat.lda[,2] ~ hat.lda[,1], groups=diabetes$clinclass,
+        auto.key=list(columns=3), par.settings=simpleTheme(pch=16))
```

2.2 Quadratic discriminant analysis

The plot of linear discriminant scores makes it clear that the variance-covariance structure is very different between the three classes. It is therefore worthwhile to try qda().

```
> diabetes.qda <- qda(clinclass ~ ., data=diabetes, CV=TRUE)
> confusion(diabetes$clinclass, diabetes.qda$class)
```

2.3 Use of randomForest()

First, fit a randomForest discriminant model, calculating at the same time the proximities. The proximity of any pair of points is the proportion of trees in which the two points appear in the same terminal node:

```
> ## Use randomForest(), obtain proximities
> library(randomForest)
> diabetes.rf <- randomForest(clinclass~., data=diabetes, proximity=TRUE)
> print(diabetes.rf)
```

Note the overall error rate.

2.4 Plot derived using `cmdscale()`

```
> distmat <- 1-diabetes.rf$proximity
> diabetes.cmd <- cmdscale(distmat)
> xyplot(diabetes.cmd[,2] ~ diabetes.cmd[,1],
+        groups=diabetes$clinclass, auto.key=list(columns=3))
```

Such plots are likely to exaggerate the separation between the groups. They represent visually the effectiveness of the algorithm in classifying the training data. To get a fair assessment, the plot should show points for a separate set of test data.

It would be more reasonable to regard the proximities as relative distances. It is easy enough to get such a plot. The above plot is adequate for present purposes.

3 The forensic glass dataset

The 'fgl' data frame, dating from 1987, was collected by B. German on fragments of glass encountered in forensic work. The variables are percentages of various elements in the glass (`Na`, `Mg`, ...), plus refractive index.

The data consist of 214 rows × 10 columns.

WinF = Window float	WinNF = Window non-float
Veh = Vehicle window	Con = Containers
Tabl = Tableware	Head = Headlamps

3.1 Linear Discriminant Analysis

The following code repeats the calculations for the `diabetes` dataset, now with the forensic glass data.

```
> library(MASS)
> library(lattice)
> fglCV.lda <- lda(type ~ ., data=fgl, CV=TRUE)
> confusion(fgl$type, fglCV.lda$class)
> fgl.lda <- lda(type ~ ., data=fgl)
> fgl.lda              # Linear discriminant 1 explains most of the variance
> hat.lda <- predict(fgl.lda)$x
> xyplot(hat.lda[,2] ~ hat.lda[,1], groups=fgl$type,
+        auto.key=list(columns=3), par.settings=simpleTheme(pch=16))
```

3.2 Random Forest Analysis

The following code repeats the calculations for the `diabetes` dataset, now with the forensic glass data.

```
> ## Use randomForest(), obtain proximities
> library(randomForest)
> fgl.rf <- randomForest(type~., data=fgl, proximity=TRUE)
> print(fgl.rf)
```

Note the overall error rate.

Now obtain a plot that is nased on the proximities:

```
> distmat <- 1-fgl.rf$proximity
> fgl.cmd <- cmdscale(distmat)
> xyplot(fgl.cmd[,2] ~ fgl.cmd[,1],
+        groups=fgl$type, auto.key=list(columns=3))
```

Part IV
Is the Match Between Source and Target Close Enough to be Useful?

Motivations: Where data are observational, there is rarely a really good match between source and target.

Data: Data are two sources:

- The main data are from an experimental study,conducted under the aegis of the the US National Supported Work (NSW) Demonstration program, of individuals who had a history of employment and related difficulties. Over 1975–1977, an experiment randomly assigned individuals who met the eligibility criteria either to a treatment group that participated in 6–18 months training program, or to a control group that did not participate.
- Also available are control (untreated) observations from a two control groups, from neighbouring regions that were though to be closely similar. The studies have the mnemonics CPS and PSID.

There has been an ongoing debate in the economometric literature, arguing whether the CPS and PSID studies provide data that might resonably be used to replace the experimental control data. Accepting that the answer is that the non-experimental data cannot be slotted in as a straight substitute, is it possible to adjust for the differences, and to use as a control one that adjustment has been made. I am convinced that the answer is, again, "No".

Methodology: Detailed comparisons will be made between the experimental control data, and the two non-experimental "controls".

1 The Labor Training Data

Data are from an experimental study,conducted under the aegis of the the US National Supported Work (NSW) Demonstration program, of individuals who had a history of employment and related difficulties. Over 1975–1977, an experiment randomly assigned individuals who met the eligibility criteria either to a treatment group that participated in 6–18 months training program, or to a control group that did not participate.

The results for males, because they highlight methodological problems more sharply, have been studied more extensively than the corresponding results for females. Participation in the training gave an increase in male 1978 earnings, relative to those in the control group, by an average of $886 [SE $472].

Can the same results be obtained by from data that matches the NSW training group with a non-experimental control group that received no such training? Lalonde (1986) and Dehejia and Wahba (1999) (see `help(nswdemo)` for the references) both investigated this question, using two different non-experimental control groups. These were

(a) The Panel Study of Income Dynamics study (PSID: 2490 males, data in `psid1`, filtered data in `psid2` and `psid3`),

(b) Westat's Matched Current Population Survey – Social Security Administration file (CPS: 16 289 males, data in `cps1`, filtered data in `cps2` and `cps3`).

Variables are

```
trt (0 = control 1=treatment)
age (years)
educ (years of education)
black (0=white 1=black)
hisp (0=non-hispanic 1=hispanic)
marr (0 = not married 2=married)
nodeg (0=completed high-school 1=dropout); i.e. educ <= 11
re74 (real earnings in 1974; available for a subset of the
     experimental data only)
re75 (real earnings in 1975)
re78 (real earnings in 1978)
```

Observe that `trt`, `black`, `hisp`, `marr` and `nodeg` are all binary variables. Here, they will be treated as dummy variables. Observations that have the value zero are the baseline, while the coefficient for observations that have the value 1 will give differences from this baseline. (For `marr`, where values are 0 or 2, the coefficient for observations that have the value 2 will be half the difference from the baseline.)

Note that `nodeg` is a categorical summary of the data in `educ`. It will not be used, additionally to `educ`, as an explanatory variable in the various analyses.

Summary information on the data

Table **??** has summary information on proportions on discrete categories that are of interest.

```
> showprop <-
+   function(dframe=psid1, facCols=4:7, zeroCols=9:10){
+     info <- numeric(length(facCols)+length(zeroCols))
+     info[1:length(facCols)] <- sapply(dframe[,facCols], function(x){
+       z <- table(x); z[2]/sum(z)})
+     info[-(1:length(facCols))] <- sapply(dframe[,zeroCols], function(x)
+       sum(x>0)/sum(!is.na(x)))
+     info
+   }
> ## Create matrix to hold resuilt
> propmat <- matrix(0, ncol=6, nrow=8)
> dimnames(propmat) <-
+   list(c("psid1", "psid2", "psid3", "cps1", "cps2", "cps3",
+          "nsw-ctl", "nsw-trt"), names(nswdemo)[c(4:7, 9:10)])
> ## Run function
> for(k in 1:8){
+   dframe <- switch(k, psid1, psid2, psid3, cps1, cps2, cps3,
+                    subset(nswdemo, trt==0), subset(nswdemo, trt==1))
+   propmat[k,] <- showprop(dframe)
+ }
```

Information on `re74` is complete for the non-experimental sets of control data, but incomplete for the experimental data. We will take us the issue of how to handle `re74` below.

Notice the big differences, for `black`, `marr` and `nodeg` (dropout), between the non-experimental controls (first six lines) and both sets of experimental data (final two lines). Even in the filtered data sets (`psid2`, `psid3`, `cps2` and `cps3`), the differences are substantial. The big changes that the filtering has made to the proportion with non-zero

	Proportion					
	Black	Hispanic	Married	Dropout	re75> 0	re78> 0
psid1	0.25	0.03	0.87	0.31	0.90	0.89
psid2	0.39	0.07	0.74	0.49	0.66	0.66
psid3	0.45	0.12	0.70	0.51	0.39	0.49
cps1	0.07	0.07	0.71	0.30	0.89	0.86
cps2	0.11	0.08	0.46	0.45	0.82	0.83
cps3	0.20	0.14	0.51	0.60	0.69	0.77
nsw-ctl	0.80	0.11	0.16	0.81	0.58	0.70
nsw-trt	0.80	0.09	0.17	0.73	0.63	0.77

Table 1: Proportion in the stated category, for each of the data sets indicated. Proportions for the experimental data are in the final two lines of the table.

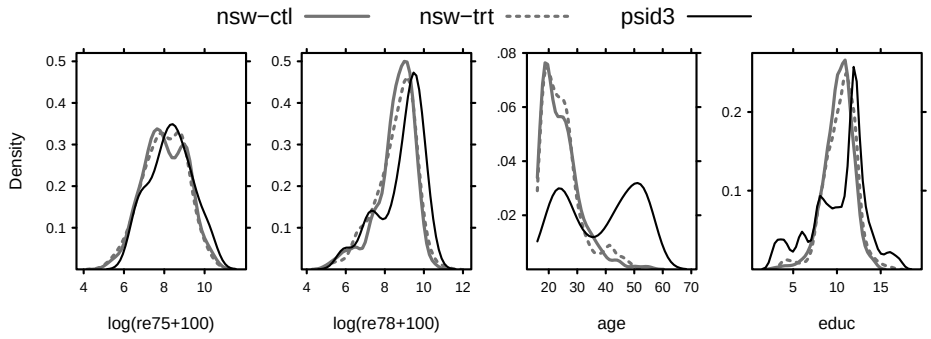

Figure 8: Overlaid density plots, comparing treatment groups with the experimental control data in `nswdemo` and with the non-experimental control data in `psid3`, for the variables `age` `educ`,`log(re75+30)`, and `log(re78+30)`.

earnings is worrying. Notice particularly the huge differences between `psid3` and `psid1`, both for `re75` and `re78`.

For those who did earn an income, how do the distributions compare? The very heavy tails in the distributions of `re75` and `re78` make use of a logarithmic transformation desirable. Figure **??** compares the distributions of values, in the control and treatment groups, for the explanatory variables `age`, `educ`, `log(re75+30)`, and `log(re78+30)`. The offset of 30 is around half the minimum non-zero value for each of these variables, in the combined data.

Examination of Figure **??** makes it clear that there are large differences between treatment and controls, whichever set of non-experimental controls is chosen.

The distributions of non-zero values of $\log(re78 + 30)$ are almost identical between experimental treated and control observations, just as similar as for $\log(re75 + 30)$. A more careful comparison will use qq-plots. The comparison can be repeated with several bootstrap samples, as a check that such small differences as are apparent are not maintained under bootstrap sampling. This is pursued in the exercises at the end of the chapter.

Follow-up on the comparison

A document has been posted on the web site, essentially Section 13.2 from the upcoming third edition of "Data Analysis and Grophics Using R" (Maindonald and Braun, Cambridge University Press), that examines whether a "propensity score" methodology that adjusts for differences between the experimental treatment and non-experimental controls

can be effective.

Part V
Variable Selection in Regression and Classification

Motivations: Where there is extensive variable selection, there is huge scope for getting spurious results.

Data: Data will be simulated that is pure noise, and the attempt made to extract a signal.

Methodology: The *DAAG* package has a function `bestsetNoise()` that will be used for the calculations.

1 Simulation – Regression with a Continuous Outcome

In the following, 100 sets of observations, on each of 10 variables, are generated. A further set of 100 values, again noise, are generated for an outcome variable. A search is then made for the one variable that best "explains" the values of the outcome variable. Here is an example:

```
> bestsetNoise(m=100, n=20, nvmax=1)

Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-2.45400 -0.59669  0.00369  0.57482  1.83155

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.0320     0.0893   -0.36    0.721
x            -0.2306     0.0946   -2.44    0.017

Residual standard error: 0.892 on 98 degrees of freedom
Multiple R-squared: 0.0572,Adjusted R-squared: 0.0476
F-statistic: 5.95 on 1 and 98 DF,  p-value: 0.0165
```

Broadly, $|t| > 2$ may, if there has been no variable selection, be taken to indicate that there is an effect that stands out above statistical noise.

Exercise 1
Repeat the above exercise a number of times, and check the frequency with which the $|t| >= 2$. You can do it thus:

```
> library(DAAG)
> tstat <- numeric(200)
> for (i in 1:200){
+     obj <- bestsetNoise(m=100, n=20, nvmax=1)
+     tstat[i] <- coef(summary(obj))["x", "t value"]   # get the t statistic
+ }
> sum(abs(tstat) > 2)/200
```

Run this calculation several times.

Exercise 2
The simulations generate random normal data, independent between variables as well as between observations. The probability that each of the 20 coefficients will have an absolute value of more than 2 is slightly less than 0.05, independently between coefficients. In repeated simulations, how frequently can it be expected that at least one of the 20 coefficients will have an absolute value of 2 or more?
[Hint: Calculate the probability that none of the coefficients will have an absolute value of 2 or more?]

Exercise 3
Repeat the above exercise a number of times, and check the frequency with which the $|t| >= 2.3$. (Use 2.3 rather than 2 as the cutoff, because we will select the best of the two coefficients that might meet the statistical detectability criterion.) You can do it thus:

```
> b <- numeric(200)
> for (i in 1:200){
+     obj <- bestsetNoise(m=100, n=20, nvmax=2, print.summary=FALSE)
+     b[i] <- max(abs(coef(summary(obj))[2:3, "t value"]))
+ }
> sum(abs(b) > 2.3)/200
```

Run this calculation several times.
[NB: It is now more challenging to derive a theoretical estimate of the relevant probability.

Exercise 4
The function `bsnCV()` also chooses random data, with no relationship between the outcome variable and the explanatory variables. However, it splits the chosen data into two subsets (this is the default; > 2 subsets are possible). It makes the first subset the training data, which it uses to select one or more explanatory variables that best account for the outcome values. Having thus selected one or more explanatory variables, a model is fitted, using those explanatory variables, to the remainder of the data (for the time being, the test data). Assuming two subsets, the roles of the two sets, as training and test, are then reversed, and the process repeated. Here is an example:

```
> bsnCV(m=100, n=20, nvmax=1, print.summary=FALSE)
```

Now repeat the calculations in Exercise 2 above:

```
> tstat1 <- tstat2 <- numeric(200)
> for (i in 1:200){
+     out <- bsnCV(m=100, n=20, nvmax=1, print.summary=FALSE)
+     tstat1[i] <- max(abs(coef(summary(out[[1]]))["x", "t value"]))
+     tstat2[i] <- max(abs(coef(summary(out[[2]]))["x", "t value"]))
+ }
> print(sum(abs(tstat1) > 2)/200)

[1] 0.06

> print(sum(abs(tstat2) > 2)/200)

[1] 0.085
```

Both the values that are printed should be within perhaps ± 0.025 or so of 0.05.

2 Variable Selection in Classification

Motivations: Where there is extensive variable selection, there is huge scope for getting spurious results.

Data: Data will be simulated that is pure noise, and the attempt made to extract a signal.

Methodology: The *hddplot* package for R has a function `simulateScores()` that will be used for the calculations.

Simulations

The following will mimic data such as comes from expression array experiments. Data may be available for some thousands of gene sequences, but typically for a rather small number of organisms, or (e.g.) cancer types. For the following, suppose that 80 individuals are randomly split between three cancer types. Assume that, for each of the cancer samples, expression indices are available for each of 7000 different gene sequences. The aim is to determine which of the 7000 indices differ between the cancer types, to an extent that can be distinguished from statistical variation.

A good way to check out a proposed methodology, and to foster intuition, is to try it out on random data. This is the rationale for the folowing exercise.

In the following:

(a) 80 sets of observations, on each of 7000 variables (features), are generated. The 80 observations are randomly divided between three groups.

(b) The 15 variables are then selected that, individually, best discriminate between the three groups. (What is best? The "best" discriminators are those that show the largest ratio of the between to the within groups sum of squares.)

(c) We apply linear discriminant analysis, using the `lda()` function from R's *MASS* package.

(d) From the linear discriminant analysis results, it is possible to determine sets of axes, with scores along each axis uncorrelated with scores along previous axes once account has been taken of the within groups correlation structure, and such that each successive set of scores accounts for as much as possible of the remaining variation. (The theory uses the singular value, or spectral, decomposition.)

(e) The scores on the two axes that give the best separation in two dimensions can then be plotted, the second set against the first.

Detailed code now follows. An alternative is to use functions in the *hddplot* package that shortcut the detail. Use of these will be explained following the detailed code.

```
> ## Step 1: Generate random data
> gps <- sample(c("A","B","C"), size=80, replace=TRUE)
> table(gps)           # Check that the groups are very roughly equal in size
> X <- matrix(rnorm(7000*80), ncol=80)
>   # NB: Rows of X will be features, and columns will be (eg) cancer samples
> ## Step 2: Select the best 15 features
> ord15 <- orderFeatures(X, gps)[1:15]
> ## Get the relevant subset of X; transpose to cancer samples by features
> X15 <- t(X[ord15, ])
> ## Step 3: Apply linear discriminant analysis
> library(MASS)
> X15.lda <- lda(X15, gps)
> scores <- predict(X15.lda, dimen=2)$x
> ## Plot the results
> library(lattice)
> theme <- simpleTheme(pch=16, cex=1.5)
>   # Filled circles, enlarge; makes it easier to see colours
> xyplot(scores[,2] ~ scores[,1], groups=gps, par.settings=theme,
+        auto.key=list(columns=3))
```

Exercise 5
Do the following

(a) Repeat the above several times.

(b) Repeat, but varying the number of features and the number of observations. With 500 features from which to choose, is there still a strong selection effect?

(c) Run the shortcut alternative:

```
> simscores <- simulateScores(nrow=7000, cl=gps)
> scoreplot(simscores)
```

Exercise 6
We can split the columns of X (cancer samples, or observations) into a training set and a test set. We use the training set to select the features, and then use those features to classify the observations in the test set. Here is how it can be done, starting with the matrix X that was obtained above:

```
> ## Divide columns of X randomly into two groups
> ## Split gps (identifying groups) in the same way.
> n <- length(gps)
> in2 <- sample(1:2, size=n, replace=TRUE)
> Xone <- X[ , in2==1]
> Xtwo <- X[ , in2==2]
> gps1 <- gps[in2==1]
> gps2 <- gps[in2==2]

> ## Select the best 15 features, using Xone
> ord15a <- orderFeatures(Xone, gps1)[1:15]
> ## Get the relevant subsets of Xone and Xtwo;
> ## transpose to observations (cancer samples) by features
> Xone15a <- t(Xone[ord15a, ])
> Xtwo15a <- t(Xtwo[ord15a, ])
```

Exercise 6, continued
Now apply linear discriminant analysis, and plot the graph that results from making predictions for Xtwo

```
> ## Apply linear discriminant analysis to Xone15a
> library(MASS)
> Xone15a.lda <- lda(Xone15a, gps1)
> ## Now apply the predictions to the data in Xtwo15a
> scoresA <- predict(Xone15a.lda, newdata=Xtwo15a, dimen=2)$x
> ## Plot the results
> xyplot(scoresA[,2] ~ scoresA[,1], groups=gps2, par.settings=theme,
+        auto.key=list(columns=3))
```

(a) Run the above code. Is any pattern discernable in the plot that results?

(b) Swap the roles of X1 and X2, i.e., use X2 to select features and fit the model, then apply the predictions to X1. If necessary, see Exercise 3 for hints on how to do this.

Note: The calculations just done illustrate the cross-validation approach, with the data split into two parts. Each part is left out in turn, and the model fitted to the reamining data. Predictions are then made for the part that is left out.

Exercise 7
Repeat exercise 2, but now keeping details of the predicted class. These can be compared with the correct assignments.

(a)
```
> predgp <- character(n)
> ## Fit model to data Xone15a
> Xone15a.lda <- lda(Xone15a, gps1)
> ## Now apply the predictions to the data in Xtwo15
> predgp[in2==2] <- predict(Xone15a.lda, newdata=Xtwo15a)$class
> ## Now make Xtwo the training data, with Xone for testing
> ord15b <- orderFeatures(Xtwo, gps2)[1:15]
> ## Get the relevant subsets of Xone and Xtwo;
> ## transpose to observations (cancer samples) by features
> Xone15b <- t(Xone[ord15b, ])
> Xtwo15b <- t(Xtwo[ord15b, ])
> ## Fit model to Xtwo15b
> Xtwo15b.lda <- lda(Xtwo15b, gps2)
> ## Apply predictons to Xone15b
> predgp[in2==1] <- predict(Xtwo15b.lda, newdata=Xone15b)$class
```

(b) Now compare predictions with actual group membership:

```
> table(Actual=gps, Predicted=predgp)
```

There should be no relationship between actual group assignments and predictions.

Exercise 7, continued

(c) Now run these calculations using the function `accTrainTest()` from *hddplot*:

```
> accTrainTest(x=X, cl=gps, traintest=in2, nfeatures=15)
```

Exercise 8
The calculations should now be run for a subset of the *Golub* data that are included in the *hddplot* package. In the interim, try:

```
> example(Golub)
```

Watch this page!