Biplots in Practice

MICHAEL GREENACRE

Professor of Statistics at the Pompeu Fabra University

Appendix A Offprint

Computation of Biplots

First published: September 2010
ISBN: 978-84-923846-8-6

Supporting websites:
http://www.fbbva.es
http://www.multivariatestatistics.org

Computation of Biplots

In this appendix the computation of biplots is illustrated using the object-orientated programming language R, which can be freely downloaded from the R-project website:

```
http://www.r-project.org
```

It is assumed that the reader has some basic knowledge of R—if not, consult some of the web resources and tutorials given on this website and in the bibliography. An R script file is given on the website:

```
http://www.multivariatestatistics.org
```

as well as the data sets, so that readers can reproduce the biplots described in this book. The idea in this Appendix is to explain some of these commands, and so serves as an R tutorial in the context of the material presented in this book.

R commands will be indicated in slanted typewriter script in *brown*, while the results are given in non-slanted green. A + at the start of a line indicates the continuation of the command (in the script file the command is given as a single line). Notice that the idea in this appendix is to educate the user in the use of R by showing alternative ways of arriving at biplot solutions, including different ways of plotting the final results. In some cases the R code might not be the most efficient way of arriving at the final goal, but will illustrate different functions in the R toolbox that can be learnt by example.

Reading data into R

There are two recommended ways of reading data into R. Suppose that you want to read in the data in Exhibit 1.1. These data are either in a text file or an Excel file, for example. Suppose that the file `EU2008.txt` is in your R working directory and contains the following:

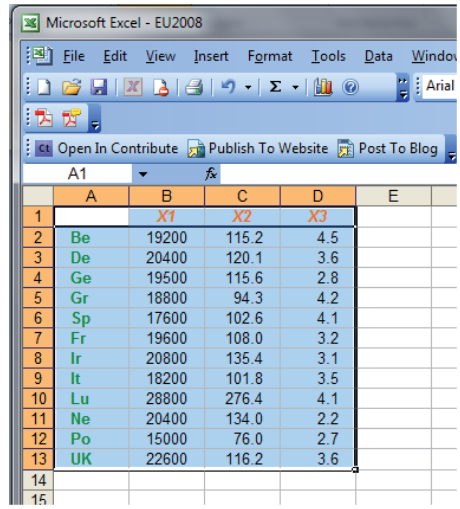
```
      X1     X2    X3
Be 19200  115.2  4.5
De 20400  120.1  3.6
Ge 19500  115.6  2.8
Gr 18800   94.3  4.2
Sp 17600  102.6  4.1
```

```
Fr   19600   108.0   3.2
Ir   20800   135.4   3.1
It   18200   101.8   3.5
Lu   28800   276.4   4.1
Ne   20400   134.0   2.2
Po   15000    76.0   2.7
UK   22600   116.2   3.6
```

Then the following command will read the data file into the data frame `EU2008`:

```
EU2008 <- read.table("EU2008.txt")
```

The alternative way (for Windows users) is to simply copy the file and then read it from the file called "clipboard". The copying can be done in the text file or in an Excel file (by painting out the data file and then using either the pull-down Edit menu, or Ctrl-C, or right-clicking on the mouse and selecting Copy), for example:



and then read the file from the clipboard using

```
EU2008 <- read.table("clipboard")
```

Notice that the function `read.table` successfully reads the table because of the blank cell in the upper left corner of the spreadsheet, which effectively signals to

the function that the first row contains the columns labels and the first column the row labels. Once the data file has been read, computations and graphical displays can commence.

The following commands reproduce Exhibit 1.2:

Chapter 1: Biplots—the Basic Idea

```
windows(width=11, height=6)
par(mfrow=c(1,2), cex.axis=0.7)
plot(EU2008[,2:1], type="n", xlab="GDP/capita",
+                            ylab="Purchasing power/capita")
text(EU2008[,2:1], labels=rownames(EU2008), col="green", font=2)
plot(EU2008[,2:3], type="n", xlab="GDP/capita",
+                            ylab="Inflation rate")
text(EU2008[,2:3], labels=rownames(EU2008), col="green", font=2)
```

The first command above sets the window size in inches—by default it would be 7 inches square—and the second command sets the plot layout with two plots side by side, and axis scale labelling in a font size 0.7 times the default. These settings remain in this window until it is closed.

Three-dimensional plotting is possible using the R package **rgl**, which should be downloaded separately—for example, using the pull-down menu in R, select Packages and Install packages, then choose a mirror site and finally choose "rgl" from the long alphabetical list of available packages. The three-dimensional display on which Exhibit 1.3 is based can then be obtained as follows:

```
library(rgl)
plot3d(EU2008[,c(2,1,3)], xlab="GDP", ylab="Purchasing power",
+                         zlab="Inflation", font=2, col="brown",
+                                           type="n")
text3d(EU2008[,c(2,1,3)], text=rownames(EU2008), font=2,
+                                                col="green")
```

The data set "bioenv" is assumed to have been read into the data frame `bioenv`, with 8 columns: the species *a* to *e*, and the three continuous variables *pollution*, *depth* and *temperature*. To calculate the linear regression of species *d* on *pollution* and *depth*:

Chapter 2: Regression Biplots

```
d <- bioenv[,4]
y <- bioenv[,6]
x <- bioenv[,7]
summary(lm(d~y+x))
```

```
(...)
Coefficients:
             Estimate  Std. Error   t value   Pr(>|t|)
(Intercept)   6.13518     6.25721     0.980    0.33554
y            -1.38766     0.48745    -2.847    0.00834 **
x             0.14822     0.06684     2.217    0.03520 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.162 on 27 degrees of freedom
Multiple R-squared: 0.4416,      Adjusted R-squared: 0.4003
F-statistic: 10.68 on 2 and 27 DF, p-value: 0.0003831
```

There are two ways to calculate the standardized regression coefficients: first by standardizing all the variables and repeating the regression:

```
ds <- (d-mean(d))/sd(d)
ys <- (y-mean(y))/sd(y)
xs <- (x-mean(x))/sd(x)
summary(lm(ds~ys+xs))

(...)
Coefficients:
               Estimate Std. Error   t value  Pr(>|t|)
(Intercept)   2.487e-17  1.414e-01 1.76e-16   1.00000
ys           -4.457e-01  1.566e-01   -2.847   0.00834 **
xs            3.472e-01  1.566e-01    2.217   0.03520 *
---
Residual standard error: 0.7744 on 27 degrees of freedom
Multiple R-squared: 0.4416,      Adjusted R-squared: 0.4003
F-statistic: 10.68 on 2 and 27 DF, p-value: 0.0003831
```

or by direct calculation using the unstandardized coefficients and the standard deviations of the variables:

```
lm(d~y+x)$coefficients[2]*sd(y)/sd(d)
          y
-0.4457286

lm(d~y+x)$coefficients[3]*sd(x)/sd(d)
        x
0.3471993
```

The standardized regression coefficients for all five variables can be calculated in a loop and stored in a matrix **B**, as in (2.2):

```
B <- lm(bioenv[,1]~y+x)$coefficients[2:3]*c(sd(y),
+                                    sd(x))/sd(bioenv[,1])
for(j in 2:5) B <- cbind(B,lm(bioenv[,j]~ y+x)$coefficients[2:3]
+                   *c(sd(y),sd(x))/sd(bioenv[j]))

B
```

```
            y           x
B -0.7171713  0.02465266
  -0.4986038  0.22885450
   0.4910580  0.07424574
  -0.4457286  0.34719935
  -0.4750841 -0.39952072
```

A regression biplot similar to the one in Exhibit 2.5 can be drawn as follows:[8]

```
plot(xs, ys, xlab="x*(depth)", ylab="y*(pollution)", type="n",
+  asp=1, cex.axis=0.7)
text(xs, ys, labels=rownames(bioenv))
text(B[,2:1], labels=colnames(bioenv[,1:5]), col="red", font=4)
arrows(0,0,0.95*B[,2],0.95*B[,1], col="red", angle=15,
+  length=0.1)
```

So far, in the plotting instructions, several graphical parameters have appeared to enhance the final figure, for example:

- `col`: sets the colour of a label or a line different from the default black, e.g. `col="red"`.
- `cex`: changes the font size of the label, e.g. `cex=0.8` scales the label to 80% of its default size.
- `cex.axis`: changes the font size of the scale on the axes.
- `font`: changes the font style, e.g. `font=4` is bold italic.

These options, and many more, are listed and explained as part of the `par` function in R—for help on this function, enter the command:

```
?par
```

To avoid repetition and commands that are full of these aesthetic enhancements of the plots, they will generally be omitted in this computational appendix from

8. Notice that any slight formatting change or improvement in Exhibit 2.5 compared to the R output, for example, the font sizes or positions of axis labels, has been done external to R to produce the final figure.

now on; but they nevertheless appear in the online script file. In addition, axis labelling will be generally omitted as well—this can be suppressed by including in the plot function the options `xlab="", ylab=""`, otherwise the default is to label the axes with the names of the variables being plotted.

Chapter 3: Generalized Linear Model Biplots

The species *d* is nonlinearly transformed by the fourth-root, and then regressed on standardized pollution and depth:

```
d0 <- d^0.25
summary(lm(d0~ys+xs))

(...)
Coefficients:
             Estimate  Std. Error  t value   Pr(>|t|)
(Intercept)   1.63908     0.09686   16.923   6.71e-16 ***
ys           -0.28810     0.10726   -2.686     0.0122 *
xs            0.05959     0.10726    0.556     0.5831
---
Residual standard error: 0.5305 on 27 degrees of freedom
Multiple R-squared: 0.2765,      Adjusted R-squared: 0.2229
F-statistic: 5.159 on 2 and 27 DF, p-value: 0.01266
```

Additional scripts are given in the online R script file for saving the coefficients for all the regressions (Exhibits 3.1, 3.4, 3.5). We give further examples just for species *d*:

Fitting a Poisson regression model for species *d*:

```
summary(glm(d~ys+xs, family=poisson))

(...)
Coefficients:
             Estimate   Std. Error  z value    Pr(>|z|)
(Intercept)   2.29617      0.06068   37.838     < 2e-16 ***
ys           -0.33682      0.07357   -4.578    4.69e-06 ***
xs            0.19963      0.06278    3.180     0.00147 **
---
(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 144.450 on 29 degrees of freedom
Residual deviance:  88.671 on 27 degrees of freedom
AIC: 208.55
```

To get the "error" deviance for this Poisson regression:

```
poisson.glm <- glm(d~ys+xs, family=poisson)
poisson.glm$deviance/poisson.glm$null.deviance
[1] 0.6138564
```

Fitting a logistic regression model, for example for species *d*, after converting its values to presence/absence (1/0):

```
d01 <- d>0
summary(glm(d01~ys+xs, family=binomial))

(...)
Coefficients:
             Estimate   Std. Error  z value    Pr(>|z|)
(Intercept)    2.7124       0.8533    3.179    0.00148 **
ys            -1.1773       0.6522   -1.805    0.07105 *
xs            -0.1369       0.7097   -0.193    0.84708
---
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 19.505 on 29 degrees of freedom
Residual deviance: 15.563 on 27 degrees of freedom
AIC: 21.563
```

To get the "error" deviance for this logistic regression:

```
logistic.glm <- glm(d01~ys+xs, family=binomial)
logistic.glm$deviance/logistic.glm$null.deviance
[1] 0.7979165
```


The data set "countries" (Exhibit 4.1) is assumed to have been read into the data frame `MT_matrix`—this is the 13 × 13 dissimilarity matrix between 13 countries given by student "MT". The R function `cmdscale` performs classical multidimensional scaling. Exhibit 4.2 is obtained as follows (notice the option `asp=1` which sets the aspect ratio equal to 1 so that the scales have identical unit intervals horizontally and vertically):

```
plot(cmdscale(MT_matrix), type="n", asp=1)
text(cmdscale(MT_matrix), labels=colnames(MT_matrix))
```

The data set "attributes" (first six columns of Exhibit 4.3) is assumed to have been read into the data frame `MT_ratings`, with 13 rows and 6 columns. To add the regression coefficients of each attribute to Exhibit 4.2 to eventually obtain Exhibit 4.5, first store the coordinates of the countries (rightmost pair of columns in Exhibit 4.3) in `MT_dims`:

```
MT_dims <- cmdscale(MT_matrix, eig=T, k=2)$points
colnames(MT_dims) <- c("dim1","dim2")
```

then calculate the regression coefficients and strore them in `MT_coefs` (Exhibit 4.4)

```
MT_coefs <- lm(MT_ratings[,1]~MT_dims[,1]+MT_dims[,2])
+  $coefficients
for(j in 2:ncol(MT_ratings)) MT_coefs<-rbind(MT_coefs,
+  lm(MT_ratings[,j]~MT_dims[,1]+MT_dims[,2])$coefficients)
```

Finally, plot the regression coefficients on the MDS plot (Exhibit 4.5)

```
plot(cmdscale(MT_matrix), type="n", asp=1)
text(cmdscale(MT_matrix), labels=colnames(MT_matrix))
arrows(0,0,MT_coefs[,2], MT_coefs[,3], length=0.1, angle=10)
text(1.2*MT_coefs[,2:3], labels=colnames(MT_ratings))
```

As an example of the definition of a function, the following is a function called `chidist` to compute the chi-square distances between the rows or columns of a supplied rectangular matrix.

```
chidist <- function(mat,rowcol=1) {
  if(rowcol= =1) {
    prof <- mat/apply(mat,1,sum)
    rootaveprof <- sqrt(apply(mat,2,sum)/sum(mat))
    }
  if(rowcol= =2) {
    prof <- t(mat)/apply(mat,2,sum)
    rootaveprof <- sqrt(apply(mat,1,sum)/sum(mat))
    }
    dist(scale(prof,FALSE,rootaveprof))
}
```

So `chidist(N,1)` calculates chi-square distances between row profiles (this is the default, so for row profiles, `chidist(N)` is sufficient), `chidist(N,2)` calculates chi-square distances between column profiles. The following code performs and saves the MDS (there are four dimensions in this problem—this is explained in Chapter 8 on correspondence analysis), and prints the percentages of variance explained on each dimension:

```
abcde <- bioenv[,1:5]
abcde_mds <- cmdscale(chidist(abcde), eig=T, k=4)
100*abcde_mds$eig/sum(abcde_mds$eig)
```

Then the site points are plotted, as in Exhibit 4.6—notice the extra parameters in the plot function for setting limits on the plot, anticipating additional points to be added to the plot, and also notice that if a matrix argument is given to the functions `plot` and `text`, then by default the first two columns are used:

```
plot(abcde.mds$points, type="n", asp=1, xlim=c(-1.2,1.6),
+   ylim=c(-1.1,1.8))
text(abcde.mds$points)
```

In this case, to add the species points, the species data are first converted to profiles, standardized by dividing them by the square roots of their marginal ("expected") values, as is the case when calculating chi-square distances:

```
abcde_prof <- abcde/apply(abcde,1,sum)
abcde_prof_stand <- t(t(abcde_prof)/sqrt(apply(abcde,2,sum)/
+   sum(abcde)))
```

The regressions are then performed on the dimensions, the coefficients saved and then added as arrows to the map plotted above:

```
mds_coefs <- lm(abcde_prof_stand[,1]~
+   abcde.mds$points[,1]+abcde.mds$points[,2])$coefficients
for(j in 2:5) mds_coefs<-rbind(mds_coefs,
+   lm(abcde_prof_stand[,j]~
+   abcde.mds$points[,1]+abcde.mds$points[,2])$coefficients)
arrows(0,0,mds_coefs[,2],mds_coefs[,3], length=0.1, angle=10)
text(1.1*mds_coefs[,2:3], labels=colnames(abcde))
```

Assuming the sediment variable has been read (in character form) into the vector `sediment`, convert it to a factor.

```
sediment <- as.factor(sediment)
```

Plot positions of sediment categories in two different ways. The first way is to average the positions of the site points for each category, to show average clay, gravel and sand site:

```
sediment.means <- cbind(tapply(abcde.mds$points[,1],
+   sediment, mean),tapply(abcde.mds$points[,2], sediment, mean))
text(sediment.means, labels=c("C","G","S"))
```

The second way is to think of them as dummy variables to be predicted by the biplot dimensions, for example by logistic regression as in Chapter 3. They are first

converted to zero/one dummies and then their logistic coefficients are used to plot them as biplot axes:

```
clay01   <- sediment=="C"
gravel01 <- sediment=="G"
sand01   <- sediment=="S"

sediment_coefs <-
+  glm(as.numeric(clay01)~abcde.mds$points[,1]+
+  abcde.mds$points[,2],family="binomial")$coefficients
sediment_coefs <- rbind(sediment_coefs,
+  glm(as.numeric(gravel01)~abcde.mds$points[,1]+
+  abcde.mds$points[,2], family="binomial")$coefficients)
sediment_coefs <- rbind(sediment_coefs,
+  glm(as.numeric(sand01)~abcde.mds$points[,1]+
+  abcde.mds$points[,2], family="binomial")$coefficients)
segments(0, 0, sediment_coefs[,2], sediment_coefs[,3])
text(sediment_coefs[,2:3], labels=c("C","G","S"))
```

Chapter 5: Reduced-dimension Biplots

This is the code to produce the biplot of the 5 × 4 matrix of rank 2 which was used as an introductory example in Chapter 1 and which is plotted here using the SVD:

```
Y <- matrix(c(8,5,-2,2,4,2,0,-3,3,6,2,3,3,-3,-6,-6,-4,1,-1,-2),
+   nrow=5)
colnames(Y) <- c("A","B","C","D")
rowcoord <- svd(Y)$u %*% diag(sqrt(svd(Y)$d))
colcoord <- svd(Y)$v %*% diag(sqrt(svd(Y)$d))
plot(rbind(rowcoord,colcoord), type="n", asp=1)
abline(h=0, v=0, lty="dotted")
text(rowcoord, labels=1:5)
text(colcoord, labels=colnames(Y))
```

Chapter 6: Principal Component Analysis Biplots

The "attributes" data set in the data frame `MT_ratings` is centred—notice the `sweep` function to subtract the column means from each column:

```
MT_means <- apply(MT_ratings,2,mean)
MT_Y <- sweep(MT_ratings, 2, MT_means)
```

The equal row and column weights are applied and the singular value decomposition (SVD) calculated:

```
MT_Y <- MT_Y/sqrt(nrow(MT_Y)*ncol(MT_Y))
MT_SVD <- svd(MT_Y)
```

The form biplot (Exhibit 6.1), showing the rows in principal coordinates and the columns in standard coordinates, is computed and plotted as follows:

```
MT_F <- sqrt(nrow(MT_Y))*MT_SVD$u%*%diag(MT_SVD$d)
MT_G <- sqrt(ncol(MT_Y))*MT_SVD$v
plot(rbind(MT_F,MT_G), type="n", asp=1, xlim=c(-3.6,2.3))
text(MT_F, labels=rownames(MT_ratings))
arrows(0, 0, MT_G[,1], MT_G[,2], length=0.1, angle=10)
text(c(1.07,1.3,1.07,1.35,1.2,1.4)*MT_G[,1],
+   c(1.07,1.07,1.05,1,1.16,1.1)*MT_G[,2],
+   labels=colnames(MT_ratings))
```

Notice two aspects of the above code: first, the plot command again contains an explicit `xlim` option to extend the horizontal axis limits slightly, to accommodate the labels of the extreme points Germany and Morocco; and second, in the text command there are explicit scaling factors—obtained by trial and error—to position the attribute labels in the plot so that they do not overlap (this is generally done externally to R, by hand, to clean up the final version of the figure).

The covariance biplot (Exhibit 6.2), showing the rows in standard coordinates and the columns in principal coordinates, is similarly computed and plotted as follows:

```
MT_F <- sqrt(nrow(MT_Y))*MT_SVD$u
MT_G <- sqrt(ncol(MT_Y))*MT_SVD$v%*%diag(MT_SVD$d)
plot(rbind(MT_F,MT_G), type="n", asp=1, xlim=c(-3.6, 2.3))
text(MT_F, labels=rownames(MT_ratings))
arrows(0, 0, MT_G[,1], MT_G[,2], length=0.1, angle=10)
text(c(1.07,1.20,1.07,1.25,1.07,1.3)*MT_G[,1],
+    c(1.07,1.07,1.04,1.02,1.16,1.07)*MT_G[,2],
+    labels=colnames(MT_ratings))
```

The basic graphical part of the scree plot of Exhibit 6.3 is drawn as follows:

```
MT_percents<-100*MT_SVD$d^2/sum(MT_SVD$d^2)
MT_percents<-MT_percents[seq(6,1)]
barplot(MT_percents, horiz=T, cex.axis=0.7)
```

The data set "`USArrests`" is in the R base package so is obtained simply with the `data` command:


```
data(USArrests)
```

Columns 1, 2 and 4 of the data frame will be used. The weighted log-ratio biplot, (7.1) to (7.4) is performed as follows, where `rm` and `cm` are the row and column margins **r** and **c**, and `mr` and `mc` are weighted means used in the double-centring:

```
N    <- USArrests[,c(1,2,4)]
P    <- N/sum(N)
rm   <- apply(P, 1, sum)
cm   <- apply(P, 2, sum)
Y    <- as.matrix(log(P))
mc   <- t(Y) %*% as.vector(rm)
Y    <- Y - rep(1,nrow(P)) %*% t(mc)
mr   <- Y %*% as.vector(cm)
Y    <- Y - mr %*% t(rep(1,ncol(P)))
Z    <- diag(sqrt(rm)) %*% Y %*% diag(sqrt(cm))
svdZ <- svd(Z)
```

The biplot of the row principal and column standard coordinates (Exhibit 7.1) is obtained as follows, where the column coordinates are scaled down by 20 to make the plot more legible. As a consequence there are two scales on the plot, indicated on the left and at the bottom for the row points, and at the top and on the right for the column points—in this case we show how the two sets of scales can be colour coded to agree with their respective points:

```
# compute form biplot coordinates from results of SVD
USA_F <- diag(1/sqrt(rm)) %*% svdZ$u[,1:2] %*% diag(svdZ$d[1:2])
USA_G <- diag(1/sqrt(cm)) %*% svdZ$v[,1:2]
# biplot – axes with different scales plotted individually
plot(rbind(USA_F, USA_G/20), xlim=c(-0.35,0.45),
+  ylim=c(-0.18,0.23), asp=1, type = "n", xaxt="n", yaxt="n")
axis(1, col.axis="green", col.ticks="green")
axis(2, col.axis="green", col.ticks="green", at=seq(-0.2,0.2,0.2))
axis(3, col.axis="brown", col.ticks="brown", at=seq(-0.4,0.4,0.2),
+  labels=seq(-8,8,4))
axis(4, col.axis="brown", col.ticks="brown", at=seq(-0.2,0.2,0.2),
+  labels=seq(-4,4,4))
text(USA_F, labels = rownames(N), col = "green")
text(USA_G/20, labels = colnames(N), col = "brown")
```

The total variance of the data can be calculated either as the sum of squares of the elements of the decomposed matrix (`Z` in the above code) or as the sum of its squared singular values:

```
sum(Z*Z)
[1] 0.01790182
```

```
sum(svdZ$d^2)
[1] 0.01790182
```

The fish morphology example goes through in a similar way, assuming that the data frame `fish` contains the data, with the first two columns being the sex and habitat (see the description later of this analysis in the computations for Chapter 11). The remaining columns are the morphometric data, stored in `fish.morph`.

```
fish.morph <- fish[,3:ncol(fish)]
```

The only difference in the plotting in Exhibit 7.3 compared to the previous example is that the column standard coordinates are divided by 50, not 20, since these data have even less variance—the sum of squares of the corresponding `Z` matrix is:

```
sum(Z*Z)
[1] 0.001960883
```

Then, instead of fish identity codes, their sex × habitat are used as labels, stored in `fish.labels`—the first statement below computes numerical codes for the four sex × habitat groups:

```
fish.sexhab <- 2*(fish[,2]-1)+fish[,1]
fish.labels <- rep("fL", nrow(fish))
fish.labels[fish.sexhab=="2"] <- "mL"
fish.labels[fish.sexhab=="3"] <- "fP"
fish.labels[fish.sexhab=="4"] <- "mP"
```

The plot of the two log-ratios in Exhibit 7.4 is obtained as follows (notice how variables can be picked out of the data.frame `fish.morph` by name):

```
logFdlFal <- log(fish.morph[,"Fdl"] / fish.morph[,"Fal"])
logFdwFal <- log(fish.morph[,"Fdw"] / fish.morph[,"Fal"])
plot(logFdlFal,logFdwFal, asp=1, pch=24, xlab="log(Fdl/Fal)",
+    ylab="log(Fdw/Fal)")
abline(a=0.0107, b=0.707, lty=2)
```

The predicted values of variable *Fdw* (dorsal fin width) are computed and then compared graphically to their actual values as follows:

```
Fdw_pred <-
+   1.0108 * fish.morph[,"Fdl"]^0.707 * fish.morph[,"Fal"]^0.293
plot(Fdw_pred, fish.morph[,"Fdw"], xlim=c(18,30), ylim=c(18,30),
+   pch=24, xlab="predicted Fdw", ylab="actual Fdw")
```

```
abline(a=0, b=1, lty=2, col="brown")
# correlation between predicted and observed
cor(Fdw_pred, fish.morph[,"Fdw"])
[1] 0.7496034
```


For the calculations of CA, and later MCA, we shall tend to use the package **ca** in R. This has to be installed from the CRAN package library first, and then loaded into an R session:

```
library(ca)
```

The "smoking" data set is included in the ca package:

```
data(smoke)
```

The commands for performing CA from first principles, as described in (8.1) and (8.2), are:

```
N     <- smoke
P     <- N/sum(N)
rm    <- apply(P, 1, sum)
cm    <- apply(P, 2, sum)
Dr    <- diag(rm)
Dc    <- diag(cm)
Z     <- diag(sqrt(1/rm))%*%(as.matrix(P)-rm%*%t(cm))
+        %*%diag(sqrt(1/cm))
svdZ <- svd(Z)
```

For the asymmetric map of Exhibit 8.1 the row principal and column standard coordinates are:

```
smoke_F <- diag(1/sqrt(rm))%*%svdZ$u %*%diag(svdZ$d)
smoke_G <- diag(1/sqrt(cm))%*%svdZ$v
```

and can be plotted in the usual way.

However, using the **ca** package Exhibit 8.2 can be obtained in just one instruction:

```
plot(ca(smoke),map="rowprincipal", col=c("green","brown"))
```

The `plot` function here is actually the `plot.ca` function, automatically recognizing the `ca` object, and the `col` option now defines the colours of the row and column symbols.

The numerical results, including the contributions to inertia, are listed using the `summary` function:[9]

```
summary(ca(smoke))

Principal inertias (eigenvalues):

dim    value       %      cum%    scree plot
1      0.074759    87.8    87.8   ************************
2      0.010017    11.8    99.5   ***
3      0.000414     0.5   100.0
       --------   -----
Total: 0.085190   100.0

Rows:
     name   mass  qlt  inr    k=1 cor ctr    k=2 cor ctr
1 |    SM |   57  893   31 |  -66  92   3 | -194 800 214 |
2 |    JM |   93  991  139 |  259 526  84 | -243 465 551 |
3 |    SE |  264 1000  450 | -381 999 512 |  -11   1   3 |
4 |    JE |  456 1000  308 |  233 942 331 |   58  58 152 |
5 |    SC |  130  999   71 | -201 865  70 |   79 133  81 |

Columns:
     name   mass  qlt  inr    k=1 cor ctr    k=2 cor ctr
1 |   non |  316 1000  577 | -393 994 654 |  -30   6  29 |
2 |   lgh |  233  984   83 |   99 327  31 |  141 657 463 |
3 |   mdm |  321  983  148 |  196 982 166 |    7   1   2 |
4 |   hvy |  130  995  192 |  294 684 150 | -198 310 506 |
```

Suppose that the "benthos" data set has been read into the data frame `benthos`. First perform the CA and calculate the row contributions to the two-dimensional biplot (note that the standard coordinates are stored in the `ca` object).

```
benthos.ca <- ca(benthos)
benthos.F <- benthos.ca$rowcoord %*% diag(benthos.ca$sv)
benthos.rowcon <- benthos.ca$rowmass * (benthos.F[,1]^2 +
+              benthos.F[,2]^2) / sum(benthos.ca$sv[1:2]^2)
```

Then set up a vector of species labels where those with contributions less than 1% are labelled "·".

9. See the paper online about the `ca` function, given in the bibliography, which describes the `ca` output.

```
benthos.names <- rownames(benthos)
benthos.names[benthos.rowcon<0.01] <- "·"
```

A nonlinear transformation is performed on the contributions above 1%, to be used for the character size of the labels.

```
benthos.rowsize <- log(1+exp(1)*benthos.rowcon^0.3)
benthos.rowsize[benthos.rowcon<0.01] <- 1
```

Exhibit 8.3 is plotted, with rows (species) in standard coordinates and columns (sites) in principal coordinates, with varying label sizes for the species.

```
FF <- benthos.ca$rowcoord
GG <- benthos.ca$colcoord %*% diag(benthos.ca$sv)
plot(rbind(FF,GG), type = "n", xlab "", ylab = "", asp=1)
text(FF[,1:2], labels = benthos.names, cex=benthos.rowsize)
text(GG[,1:2], labels = colnames(benthos))
```

The biplot showing point contributions is available in the **ca** package, using map option `"rowgreen"` or `"colgreen"` depending on which set is required in principal coordinates (in our case it would be the sites, or columns). The species labels are first substituted with those where the low contributing ones are replaced by "·".

```
benthos.ca$rownames <- benthos.names
```

Because rows are species (variables) and columns are sites (samples) the symbols need to be reversed—see `help(plot.ca)` in R for information about the plot option `pch` and execute the command `pchlist()` to get a list of plotting symbols (an alternative would be to transpose the data matrix from the start). For this biplot we also use the plot option `mass` to get symbols with sizes related to the species masses. The contribution biplot of Exhibit 8.4 is obtained using plot option `map="colgreen"`, which plots the columns in principal coordinates, as before, but the species (rows) in their contribution positions:

```
plot(benthos.ca, map="colgreen", mass=c(1,0), pch=c(17,24,16,1))
```

Lines are added connecting the origin with the species points, showing how their positions are computed as standard coordinates multiplied by the square roots of the species masses:

```
for(j in 1:nrow(benthos)) lines(
+ c(0,benthos.ca$rowcoord[j,1]*sqrt(benthos.ca$rowmass[j])),
+ c(0,benthos.ca$rowcoord[j,2]*sqrt(benthos.ca$rowmass[j])))
```

Alternatively, the `segments` function can be used, which automatically recycles the coordinates 0 and 0 in the following command:

```
segments(0, 0, benthos.ca$rowcoord[,1]*sqrt(benthos.ca$rowmass),
+              benthos.ca$rowcoord[,2]*sqrt(benthos.ca$rowmass))
```

Chapter 9: Multiple Correspondence Analysis Biplots I

The original data set "women" consists of the 2471 Spanish respondents in the 2002 ISSP survey on Family and Changing Gender Roles III, including their responses to the 8 substantive and 4 demographic variables listed in Chapter 9. In the supporting website it is shown how the concatenated matrix can be extracted from the original data. The simplest way is using a concept from Chapter 10 called the *Burt matrix*, and the most complicated is by converting all question responses first into zero—one dummy variables—both of these are explained in the next section on Chapter 10 . For the moment we assume that this matrix (part of which is shown in Exhibit 9.1), called `women.concat`, has already been computed, or input directly from an external source (the concatenated matrix itself is also provided on the website). The two categories *H4* and *H5* have also been combined into a category, labelled *H4,5*, so the matrix has 23 rows and 39 columns.

The symmetric CA of the concatenated matrix can be obtained using the default plot option in the **ca** package:

```
plot(ca(women.concat))
```

Depending on the version of the **ca** package (or any other software for correspondence analysis), an inversion of the axes might be obtained. The following code shows how the sign of the second axis is reversed, once the correspondence analysis object is saved:

```
women.ca <- ca(women.concat)
women.ca$rowcoord[,2] <- -women.ca$rowcoord[,2]
women.ca$colcoord[,2] <- -women.ca$colcoord[,2]
plot(women.ca)
```

Exhibit 9.2 was obtained by plotting the symbols first, and then adding the two sets of labels in different styles, also adding the inertias and their percentages on the axes:

```
plot(women.ca, labels=0)
women.F <- women.ca$rowcoord %*% diag(women.ca$sv)
women.G <- women.ca$colcoord %*% diag(women.ca$sv)
text(women.F, labels=women.ca$rownames, pos=4, offset=0.3)
text(women.G, labels=women.ca$colnames, pos=4, offset=0.3)
```

```
text(max(women.G[,1]), 0, "0.0571 (82.1%)", adj=c(0.6,-0.6))
text(0, max(women.G[,2]), "0.0030 (4.4%)", adj=c(-0.1,-3))
```

The map is plotted in a square window, which should then be pulled into the flatter shape of Exhibit 9.2 (the aspect ratio is not affected by this action). Having pulled the window into the desired shape, repeat the plotting so that the labels are properly positioned. Then the only difference between this result and Exhibit 9.2 is an adjustment of some of the overlapping labels, performed externally to R.

Exhibit 9.3 is plotted in a similar way, but using the plot option `map="rowprincipal"`

```
plot(women.ca, map="rowprincipal", labels=c(0,2))
```

Similarly, the contribution biplot in Exhibit 9.6, where the the standard coordinates are shrunk by the square roots of the category masses, is obtained with the plot option `map="rowgreen"` (here the `mass` option is also illustrated, to make the size of the column category symbols be related to their masses:

```
plot(women.ca, map="rowgreen", mass=c(F,T))
```

To add the supplementary points for sex (`m` = male, `f` = female) and age (`a1` to `a6`):

```
women.sex <- c(rep("m",6),rep("f",6))
women.age <- rep(c("a1","a2","a3","a4","a5","a6"),2)
women.sex.F <- cbind(tapply(women.F[12:23,1],women.sex,mean),
+                    tapply(women.F[12:23,2],women.sex,mean))
women.age.F <- cbind(tapply(women.F[12:23,1],women.age,mean),
+                    tapply(women.F[12:23,2],women.age,mean))
points(rbind(women.sex.F, women.age.F), pch=21)
text(rbind(women.sex.F, women.age.F),
+  labels=c("f","m","a1","a2","a3","a4","a5","a6"),
+  pos=4, offset=0.3)
```

Chapter 10: Multiple Correspondence Analysis Biplots II

The *Burt matrix* is a by-product of the `mjca` function in the **ca** package. If `women` contains the original response data, with the first 8 columns corresponding to the eight substantive questions *A* to *H*, then the Burt matrix is obtained as follows:

```
women.Burt <- mjca(women[,1:8])$Burt
```

If the categories *H4* and *H5* have not already been combined, then this can be done by combining the corresponding rows and columns of the Burt matrix:

```
women.Burt[,39] <- women.Burt[,39]+women.Burt[,40]
women.Burt[39,] <- women.Burt[39,]+women.Burt[40,]
women.Burt <- women.Burt[-40,-40]
rownames(women.Burt)[39] <- colnames(women.Burt)[39] <- "H4,5"
```

An alternative way to compute the Burt matrix, assuming that the "women" data set has been read as dummy variables into the data frame `women.Z` containing the indicator matrix of dummy variables for the 8 questions *A* to *H* (40 dummies if *H5* included, otherwise 39 if *H4* and *H5* have been combined). Then, as given in (10.2), the Burt matrix can be computed by premultiplying the indicator matrix by its transpose (notice the `as.matrix` commands if `women.Z` is a data frame, necessary for the multiplication):

```
women.Burt<- t(as.matrix(women.Z))%*%as.matrix(women.Z)
```

In the same way, the concatenated matrix `women.concat` can be obtained from the Burt matrix of all the variables, including the demographics, or via the indicator matrices. Suppose that `womenS.Z` contains the 2107×31 indicator matrix of dummy variables for the demographic variables (2 for sex, 5 for marital status, 6 for education, 6 for age, 12 sex-age combinations), then `women.concat` can be computed by premultiplying `women.Z` by the indicator matrix of dummy variables corresponding to sex, marital status, education and the sex-age combinations:

```
women.concat<- t(as.matrix(womenS.Z[,c(3:13,20:31)])) %*%
+                as.matrix(women.Z)
```

Alternatively, select just that part of the Burt matrix of all the variables, including the demographics, corresponding to the concatenated matrix:

```
women.concat <- mjca(women)$Burt[c(3:13,20:31),]
```

To obtain the total inertia of the Burt matrix, sum the squares of its singular values in the CA:

```
sum(ca(women.Burt)$sv^2)
[1] 0.677625
```

Then use (10.1) to calculate the adjusted total inertia:

```
(8/7)*(sum(ca(women.Burt)$sv^2)-(31/64))
[1] 0.2208571
```

Total inertia of the indicator matrix, calculated from the CA (from theory we know it will be equal to (39 – 8)/8 = 3.875:

```
sum(ca(women.Z)$sv^2)
[1] 3.875
```

Exhibit 10.3 can be obtained as follows (notice the change in the point symbols using the `pch` option, to get a smaller dot symbol for the cases, and the mass option to get triangle symbols related to the relative frequency of the category):

```
plot(ca(women.Z), map="rowprincipal", labels=c(0,2),
+    pch=c(149,1,17,24), mass=c(FALSE,TRUE))
```

To see how many of the singular values (i.e., square roots of principal inertias) in the analysis of the Burt matrix are larger than 1/8:

```
which(ca(women.Burt)$sv>(1/8))
[1] 1 2 3 4 5 6 7 8 9
```

So we apply the adjustment of (10.4) to the first 9 singular values:

```
(8/7)*(ca(women.Burt)$sv[1:9]-(1/8))
[1] 0.34219 0.23260 0.12415 0.11500 0.03451 0.02575 0.01489
+    0.00897 0.00681
```

To get parts of inertia explained on the axes, square these adjusted singular values and then express relative to the adjusted total inertia calculated previously:

```
(64/49)*(ca(women.Burt)$sv[1:9]-(1/8))^2/0.2208571
[1] 0.53017 0.24496 0.06979 0.05987 0.00539 0.00300 0.00100
+    0.00036 0.00021
```

Notice that the above parts do not add up to 1, since these 9 MCA axes cannot perfectly explain the total inertia in the off-diagonal tables: we would need to use joint correspondence analysis (JCA) to achieve this. Exhibit 10.4 is obtained by substituting the square roots of the adjusted inertias for the original ones in the CA of the Burt matrix:

```
women.Burt.ca     <- ca(women.Burt)
women.Burt.ca$sv  <- diag((8/7)*(ca(women.Burt)$sv[1:9]-(1/8)))
```

The website `http://www.multivariatestatistics.org` gives the full sets of instructions for plotting Exhibits 10.4 and 10.5. Exhibit 10.6 illustrates the computation of the contribution coordinates of the categories and including supplementary points (again, notice that axes may be inverted compared to Exhibit 10.6):

```
women.BurtS.ca    <- ca(rbind(women.Burt, women.concat),
+                    suprow=40:62)
women.BurtS.Gctr <- sqrt(women.BurtS.ca$colmass) *
+                    women.BurtS.ca$colcoord
women.BurtS.ca$colcoord <- women.BurtS.Gctr
women.BurtS.ca$sv[1:9] <- (8/7)*(women.BurtS.ca$sv[1:9]-(1/8))
plot(women.BurtS.ca, map="rowprincipal", what=c("none","all"),
+    labels=0, pch=c(20,1,24,24))
text(women.BurtS.Gctr, labels=women.Burt.ca$colnames, pos=2)
women.BurtS.Fsup <- women.BurtS.ca$rowcoord[40:62,] %*%
+    diag(women.BurtS.ca$sv)
points(women.BurtS.Fsup, pch=21)
text(women.BurtS.Fsup, labels=women.BurtS.ca$rownames[40:62],
+    pos=2)
```

Chapter 11: Discriminant Analysis Biplots

From the log-ratio biplot of the morphometric data of the "morphology" data set in chapter 7 we know that the total variance is equal to 0.001961. We now want to aggregate the data into the four sex × habitat groups and measure how much variance is lost. The original data should not be aggregated since we are working on a logarithmic scale. It is equivalent, however, to aggregate the log-transformed data, or aggregate the rows of the double-centred matrix of log-transformed values. We choose the second way as an illustration, first repeating the initial steps of the log-ratio analysis algorithm (see computations for Chapter 7) on the matrix `fish.morph`:

```
N  <- fish.morph
P  <- N/sum(N)
rm <- apply(P, 1, sum)
cm <- apply(P, 2, sum)
Y  <- as.matrix(log(P))
mc <- t(Y) %*% as.vector(rm)
Y  <- Y - rep(1,nrow(P)) %*% t(mc)
mr <- Y %*% as.vector(cm)
Y  <- Y - mr %*% t(rep(1,ncol(P)))
```

The group masses are calculated:

```
fish.centroids.rm <- tapply(rm, fish[,3], sum)
```

and the four centroids, by weighted averaging of the corresponding rows of **Y**:

```
fish.centroids <- tapply(rm * Y[,1], fish[,3], sum)
for(j in 2:ncol(fish.morph)) fish.centroids
+      <- cbind(fish.centroids, tapply(rm * Y[,j], fish[,3], sum))
fish.centroids <- fish.centroids / fish.centroids.rm
```

Then the LRA algorithm continues for the four centroids, using the weighted SVD:

```
Z <- diag(sqrt(fish.centroids.rm)) %*% fish.centroids %*%
+  diag(sqrt(cm))
svdZ <- svd(Z)
# principal coordinates of centroids, standard coordinates of
+  variables
FF <- diag(1/sqrt(fish.centroids.rm)) %*% svdZ$u %*% diag(svdZ$d)
GG <- diag(1/sqrt(cm)) %*% svdZ$v
```

The inertia of the centroids:

```
inertia.centroids <- sum(Z*Z)
inertia.centroids
[1] 0.000128325
```

which is 6.5% of the total variance 0.001961 of the individual fish, computed in Chapter 7.

The biplot of the centroids and variables Exhibit 11.2 has a scaling factor difference of 50 between the two sets of points, as in Exhibit 7.3.

CA-DA, which is a CA of a concatenated table where a set of variables is cross-tabulated against a single grouping variable, is illustrated by Exhibit 11.3 for the "women" data set, using the marital status categories in the first five lines of the matrix `women.concat` (see Chapter 9). In this case we chose the contribution biplot:

```
women.ca_da <- ca(women.concat[1;5,])
women.ca_da$rownames <-  c("married", "widowed", "divorced",
+                          "separated", "single")
plot(women.ca_da, map="rowgreen")
```

(again, as explained in the computations of Chapter 9, if the axes are reversed compared to Exhibit 11.3, their coordinates can be multiplied by –1).

To see some basic results of the CA object: principal inertias, their percentages, the row and column masses, chi-square distances to the centroid, inertias, standard coordinates, etc, just type the object name:

```
women.ca_da
Principal inertias (eigenvalues):
            1         2         3         4
Value       0.029316  0.002915  0.002321  0.000993
Percentage  82.48%    8.2%      6.53%     2.79%

 Rows:
           married    widowed    divorced   separated      single
Mass      0.554817   0.081633   0.021357    0.032748    0.309445
ChiDist   0.080668   0.413108   0.320868    0.226113    0.213684
Inertia   0.003610   0.013931   0.002199    0.001674    0.014129
Dim. 1    0.403377   2.291677  -0.619362   -0.617888   -1.219648
Dim. 2   -0.656420   2.210670  -2.610218   -0.462015    0.822790

 Columns:
```

etc. … More detailed results can be obtained using the `summary` function, as explained before.

The "iris" data set is available in R:

```
data(iris)
```

The first four columns contain the variables and the fifth column contains the classification into the three groups: `"setosa"`, `"versicolor"` and `"virginica"` (there are 50 in each group). Read the data into `X` and calculate the means in `G`:

```
X <- iris[,1:4]
n <-nrow(X)
p <-ncol(X)
G <- apply(X[iris[,5]=="setosa",],2,mean)
G <- rbind(G,apply(X[iris[,5]=="versicolor",],2,mean))
G <- rbind(G,apply(X[iris[,5]=="virginica",],2,mean))
g <- nrow(G)
rownames(G) <- c("setosa", "versicolor", "virginica")
colnames(G) <- c("SepL","SepW","PetL","PetW")
colnames(X) <- c("SepL","SepW","PetL","PetW")
```

Calculate the three within-group covariance matrices (notice that we prefer the definition where the sum of squares is divided by n and not $n-1$, hence the slight adjustment by $(n-1)/n = 49/50$.

```
C1 <- (49/50)*cov(X[iris[,5]=="setosa",])
C2 <- (49/50)*cov(X[iris[,5]=="versicolor",])
C3 <- (49/50)*cov(X[iris[,5]=="virginica",])
```

The average within-grouped covariance matrix **C** is just the arithmetic average since the groups are the same size (otherwise it should be the weighted average —see (11.3)):

```
C <- (C1+C2+C3)/3
```

To calculate the inverse square root of **C**, calculate its SVD (or eigenvalue-eigenvector decomposition) and then use the inverse square roots of the singular values:

```
C.svd <- svd(C)
Cminushalf <- C.svd$u %*% diag(1/sqrt(C.svd$d)) %*% t(C.svd$v)
```

Calculate the matrix **S** of (11.4), its SVD and coordinates for the contribution biplot:

```
oneg <- rep(1,g)
Ig   <- diag(oneg)
S    <- diag(rep(sqrt(1/g),g)) %*% (Ig - (1/g)* oneg %*% t(oneg))
+           %*% G %*% Sminushalf * sqrt(1/ncol(G))
S.svd <- svd(S)
S.rpc <- sqrt(g) * S.svd$u %*% diag(S.svd$d)
S.cbp <- S.svd$v
```

Calculate the coordinates of the individual $n = 150$ irises as supplementary points according to (11.5):

```
onen <- rep(1,n)
In <- diag(onen)
S.rsup <- (In - (1/n)* onen %*% t(onen)) %*% as.matrix(X)
+         %*% Cminushalf %*% S.svd$v * sqrt(1/p)
```

Plot the groups and individual points in three colours:

```
plot(S.rsup, type = "n", asp=1)
text(S.rsup, labels = "·", col = c(rep("green",50),,
+  rep("violet",50) rep("brown",50)), cex=2, font = 2)
text(S.rpc, labels = rownames(G),
+  col = c("green","violet","brown"), font = 2, adj=c(0.5,0.5))
text(S.cbp, labels = colnames(G), col = "brown", cex=0.8, font = 2)
segments(0,0,S.cbp[,1],S.cbp[,2],col="brown")
```

Variance of the group means (i.e., between-group variance) is the sum of squares of the elements of the **S** matrix:

```
sum(S*S)
[1] 8.11933
```

The total variance of the points is obtained by calculating the equivalent matrix for the individuals in the Mahalanobis metric):

```
S <- sqrt(1/n) * (I - (1/n)* onen %*% t(onen)) %*% as.matrix(X)
+    %*% Cminushalf * sqrt(1/p)
sum(S*S)
[1] 9.11933
```

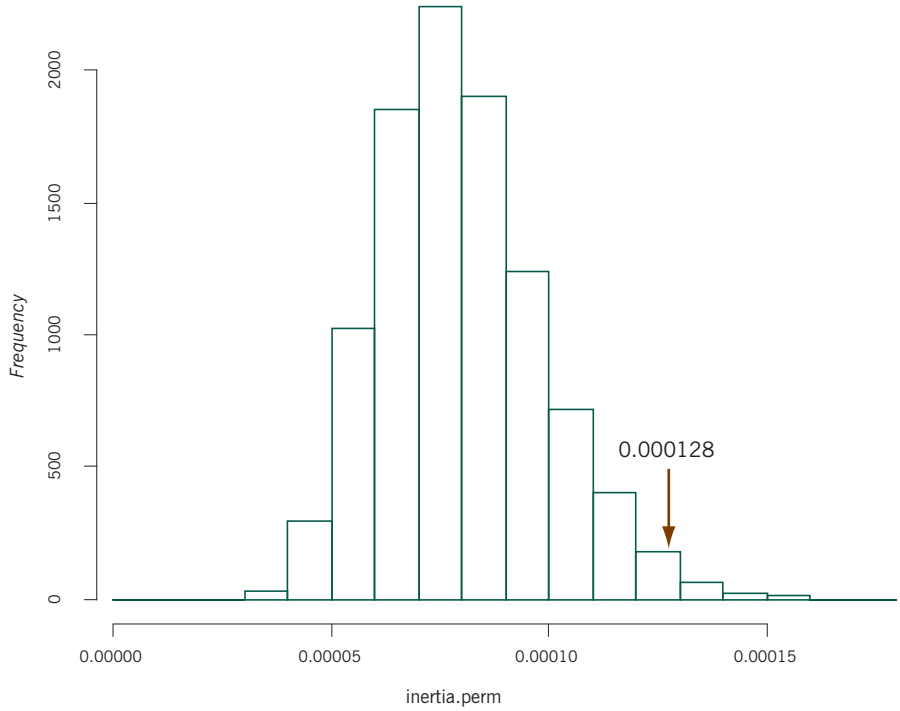
The difference between these two variance measures is exactly 1, which is the value of the within-group variance, by construction.

A Permutation Test

To test whether the between-group variance of 0.000128 in the above example of the four fish groups is significant, a permutation test consists in randomly shuffling the set of group labels assigned to the fish, recomputing the between-group variance each time (we reallocate the labels randomly 9,999 times) and seeing where the actual figure (which serves as the 10,000th permutation) lies in the permutation distribution. The following R code does the job, assuming that the same initial steps of the LRA algorithm are performed (see the 9 commands at the start of the computations for Chapter 11), ending with `Y` being the double-centred matrix of log-transformed data. Notice that the commands in the loop are just a repeat of the code previously used, but starting each iteration with a random sampling of the group labels, using the `sample` function. The initial `set.seed(317)` operation can be changed (or omitted) if you want a different set of random permutations.

```
set.seed(317)
inertia.perm <- rep(0,10000)
inertia.perm[1] <- inertia.centroids
for(iperm in 2:10000) {
  fish.perm<-sample(fish[,3])
  fish.centroids.rm <- tapply(rm, fish.perm, sum)
  fish.centroids <- tapply(rm * Y[,1], fish.perm, sum)
  for(j in 2:ncol(fish.morph)) fish.centroids
+      <- cbind(fish.centroids, tapply(rm * Y[,j], fish.perm, sum))
  fish.centroids <- fish.centroids / as.numeric(fish.centroids.rm)
  Z <- diag(sqrt(fish.centroids.rm)) %*% fish.centroids
+      %*% diag(sqrt(cm))
  inertia.perm[iperm] <- sum(Z*Z)
  }
```

Exhibit A.1:
Histogram of permutation distribution showing observed test statistic. The p*-value is the relative area of the distribution from the test statistic to the right*



To see where the value of 0.000128 lies in the permutation distribution:

```
which(sort(inertia.perm)==inertia.perm[1])
[1] 9847
```

so the number of permutations in the tail, including our observed value, is 154, which shows that it lies in the far upper tail of the distribution, with a *p*-value of 154/10,000 = 0.0154. A histogram of the permutation distribution, indicating the position of the observed value is shown in Exhibit A.1.

Chapter 12: Constrained Biplots

For the analysis of the fish morphometric data, we first add the body weight of the fish as a supplementary variable to the unconstrained log-ratio analysis. Again the nine commands at the start of the computations in Chapter 11 are repeated, up to the computation of the double-centred `Y`. Here we show the "column-principal" or "covariance" biplot, computing the SVD the usual way and then row standard and column principal coordinates:

```
Z    <- diag(sqrt(rm)) %*% Y %*% diag(sqrt(cm))
svdZ <- svd(Z)
```

```
FF    <- diag(1/sqrt(rm)) %*% svdZ$u
GG    <- diag(1/sqrt(cm)) %*% svdZ$v %*% diag(svdZ$d)
```

The body weight variable is standardized and regressed on the coordinates of the fish on the two dimensions of the morphometric log-ratio analysis. Since the fish are weighted according to their marginal totals, a weighted regression is performed, using the row masses in `rm` . Suppose that the body weight variable has been read into the vector `fish.weight`, then the R function `cov.wt` computes weighted means and variances:

```
fish.weight.mean  <- cov.wt(as.matrix(fish.weight),wt=rm)$center
fish.weight.var   <- cov.wt(as.matrix(fish.weight),wt=rm)$cov
fish.weight.stand <- (fish.weight-fish.weight.mean)/
+                     sqrt(fish.weight.var)
lm(fish.weight.stand~FF[,1]+FF[,2], weights=rm)$coefficients
  (Intercept)        FF[, 1]        FF[, 2]
-5.764505e-15   1.162973e-01   2.025847e-01
```

The coefficients 0.116 and 0.203 would define an arrow in the LRA biplot but only 5.5% of the variance of the body weight variable is explained, as can be seen by executing the `summary()` of the regression model above:

```
summary(lm(fish.weight.stand~FF[,1]+FF[,2], weights=rm))

(...)
Coefficients:
               Estimate   Std. Error     t value   Pr(>|t|)
(Intercept)  -5.765e-15    1.138e-01   -5.07e-14      1.0000
FF[, 1]       1.163e-01    1.138e-01       1.022      0.3101
FF[, 2]       2.026e-01    1.138e-01       1.781      0.0792
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1138 on 72 degrees of freedom
Multiple R-squared: 0.05531,    Adjusted R-squared: 0.02907
F-statistic: 2.108 on 2 and 72 DF,  p-value: 0.1290
```

To constrain the first axis to be perfectly correlated with body weight the definition of the projection matrix **Q** in (12.2) is particularly simple, because there is only a scalar to invert, not a matrix:

```
Q      <- diag(sqrt(rm)) %*% fish.weight.stand
+         %*% (1/((t(fish.weight.stand)%*%diag(rm)
```

```
+         %*%fish.weight.stand))) %*% t(fish.weight.stand)
+         %*% diag(sqrt(rm))
QZ     <- Q %*% Z
svdQZ  <- svd(QZ)
```

The orthogonal projection and corresponding SVD is:

```
QpZ <- Z - QZ
svdQpZ <- svd(QpZ)
```

The coordinates of the points are obtained from the first axis of the constrained analysis, and the first axis of the unconstrained one, and the body weight vector is obtained by weighted linear regression as before:

```
FF[,1] <- diag(1/sqrt(rm)) %*% svdQZ$u[,1]
GG[,1] <- diag(1/sqrt(cm)) %*% svdQZ$v[,1] * svdQZ$d[1]
FF[,2] <- diag(1/sqrt(rm)) %*% svdQpZ$u[,1]
GG[,2] <- diag(1/sqrt(cm)) %*% svdQpZ$v[,1] * svdQpZ$d[1]
fish.weight.coefs <- lm(fish.weight.stand~FF[,1]+FF[,2],
+                        weights=rm)$coefficients
```

After plotting the row and column points as before (again with two scales), body weight can be indicated by an arrow, as shown in Exhibit 12.1, as follows:

```
arrows(0, 0, 0.9*fish.weight.coefs[1], 0.9*fish.weight.coefs[2],
+      lwd=1.5, length=0.10, angle=15)
text(fish.weight.coefs[1], fish.weight.coefs[2], "weight")
```

The decomposition of the total variance into constrained and unconstrained parts:

```
sum(Z*Z)
[1] 0.001960883
sum(QZ*QZ)
[1] 7.860202e-05
sum(QpZ*QpZ)
[1] 0.001882281
100*sum(QZ*QZ)/sum(Z*Z)
[1] 4.008501
```

To perform a permutation test on the percentage of variance of the morphometric data explained by body weight, simply loop over the calculation of this percentage for random permutations of the body weight vector. The position of the

observed percentage of 4.01% in the sorted list of 10,000 permutations (9,999 plus the observed one) estimates the *p*-value:

```
set.seed(157)
bodyperm<-rep(0,10000)
total <- sum(Z*Z)
Q <- diag(sqrt(rm)) %*% fish.weight.stand %*%
+    (1/((t(fish.weight.stand) %*% diag(rm) %*% fish.weight.stand)))
+    %*% t(fish.weight.stand) %*% diag(sqrt(rm))
QZ <- Q %*% Z
bodyperm[1]<-100*sum(QZ*QZ)/total
# start permutations
for(iperm in 2:10000){
  fish.weight.stand.perm<-sample(fish.weight.stand)
  Q <- diag(sqrt(rm)) %*% fish.weight.stand.perm
  +  %*% (1/((t(fish.weight.stand.perm) %*% diag(rm)
  +  %*% fish.weight.stand.perm))) %*% t(fish.weight.stand.perm)
  +  %*% diag(sqrt(rm))
  QZ <- Q %*% Z
  bodyperm[iperm]<-100*sum(QZ*QZ)/total
  }
# find where the observed percentage is in the sorted list
which(sort(bodyperm)==bodyperm[1])
[1] 9991
```

The observed value is 10th from the top of the 10,000 values and the estimated *p*-value is thus 10/10,000 = 0.001.

For the final CCA of the data set "benthos", we illustrate the use of the function `cca` in the **vegan** package in R (this package needs to be installed separately). First read in the six environmental variables into the data frame `benthos_env`. Notice that this data set has the sites in the rows whereas benthos has the sites in the columns. For the `cca` function the sites need to be in the rows in both matrices, hence the use of `t(benthos)` in the code below. After log-transforming the environmental variables, the CCA is performed and the points and environmental variable arrows are plotted:

```
benthos_env <- log(benthos_env)
library(vegan)
benthos.cca <- cca(t(benthos), benthos_env)
plot(benthos.cca, display=c("lc","bp","sp"), type="n")
text(benthos.cca, display="bp", labels=colnames(benthos_env))
text(benthos.cca, display="sp", labels=rownames(benthos))
text(benthos.cca, display="lc", labels=colnames(benthos))
```

The plotting options chosen give the asymmetric biplot with sites in standard coordinates and species in principal coordinates (i.e., at weighted averages of the site points), and the environmental variables as biplot vectors using their regression coordinates on the CCA axes—these biplot coordinates are identical to the (weighted) correlation coefficients with the axes, since the site coordinates are standardized.

This plot is highly cluttered by all the species labels and so we can prune them down to the set of species that is most contributing to the display, as we have shown before in Chapter 8 for the same data set "benthos". The following code assigns "." labels to all species with less than a 1% contribution to the triplot:

```
benthos.cca.sp <- benthos.cca$CCA$v.eig
benthos.cca.spcon <- benthos.cca$colsum * (benthos.cca.sp[,1]^2 +
+  benthos.cca.sp[,2]^2) / sum(benthos.cca$CCA$eig[1:2])
benthos.names <- rownames(benthos)
benthos.names[benthos.cca.spcon<0.01] <- "·"
```

Plotting is then repeated as before, but substituting `benthos.names` for the original `rownames(benthos)` when species labels are plotted:

```
text(benthos.cca, display="sp", labels=benthos.names)
```

Further R scripts for the three case studies in Chapters 13 to 15 are given in the supporting website.

Biplot Software in R

The objective of this computational appendix is to educate readers in the use of R to construct biplots. Seeing and understanding the commands associated with specific figures in this book will assist users to perform their own analyses and biplots in the same way, as well as make them more proficient in R. Apart from these scripts, there are additional functions and one software package available for biplots.

The R functions `princomp` and `prcomp` for principal component analysis (PCA) both have a plotting function `biplot`. For example, `biplot(princomp(X))` draws the biplot of the PCA of the data matrix `X`. There are always two scales on the axes, one for the rows and one for the columns (similar to Exhibits 11.2, 12.1 and 12.5, for example). The `biplot` function (which is actually `biplot.princomp` or `biplot.prcomp` depending on which PCA function is used) has some scaling options for the axes:

`scaling=1`	form biplot (rows principal, columns standard)
`scaling=0`	covariance biplot (columns principal, rows standard)

(in fact, `scaling=alpha` produces a biplot where rows are scaled by the singular values to power `alpha` and the columns to power `1–alpha`, so `scaling=0.5` gives the symmetric biplot).

In addition there is a general biplot function `biplot` for plotting two given sets of points simultaneously.

The **ca** package described in Chapters 8 to 10 has several biplot options in the `plot.ca` function, although they are referred to as "maps". These are summarized below:

`map="rowprincipal"`	plots rows in principal, columns in standard coordinates
`map="colprincipal"`	plots columns in principal, rows in standard coordinates
`map="symbiplot"`	symmetric biplot, with row and column coordinates scaled by the square roots of the singular values on respective axes
`map="rowgreen"`	plots rows in principal, columns in contribution coordinates
`map="colgreen"`	plots columns in principal, rows in contribution coordinates

In addition, there are two options `"rowgab"` and `"colgab"`, due to Ruben Gabriel, who proposed multiplying the standard coordinates by the respective masses, whereas in the contribution biplots `"rowgreen"` and `"colgreen"` the square roots of the masses are used, which gives the specific interpretation in terms of contributions.

An R package **caGUI** is available as an interactive front end to the **ca** package. Finally, there is an interactive package **BiplotGUI** for R, mostly aimed at calibrated biplots, which were illustrated in Chapters 2 and 3 when introducing the biplot idea through regression and generalized linear models (see comments about calibrated biplots in the Epilogue).