ultivariate-and-tree-based-methods

April 19, 2024

1 6. Multivariate and Tree-based Methods

1.1 6.1 Multivariate EDA, and Principal Components Analysis

Principal components analysis (PCA) is a useful tool for exploring multivariate data. It aims to condense the original variables into a smaller set of "principal components" that capture most of the variation in the data. The first principal component explains the most variation, followed by subsequent components that capture the remaining variation unexplained by previous components.

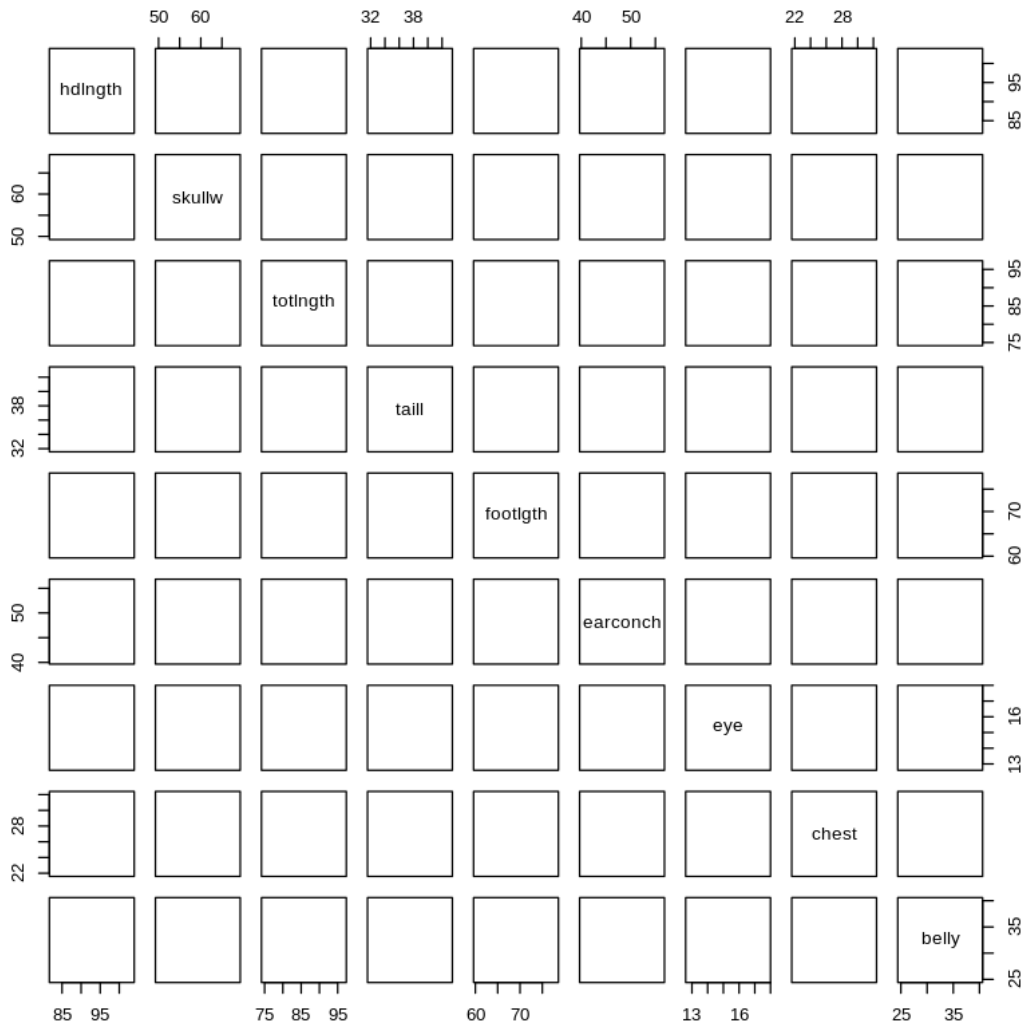
The **measure of variation** used in PCA is typically the sum of variances of the variables, which can be scaled to have variance one. Analyzing unscaled variables gives more weight to those with larger variances, while scaling to equal variances, equivalent to working with correlation matrices, provides equal importance to each variable. Logarithmic transformation is often applied to biological measurement data to standardize variability.

In the dataset "possum," consisting of nine morphometric measurements from 102 mountain brushtail possums across seven sites, examining scatterplot matrices is a good starting point. These plots can reveal any notable patterns or outliers in the data, such as differences between sites or sexes. Logarithmic transformation may not significantly alter the appearance of these plots if the range of values across variables is relatively small.

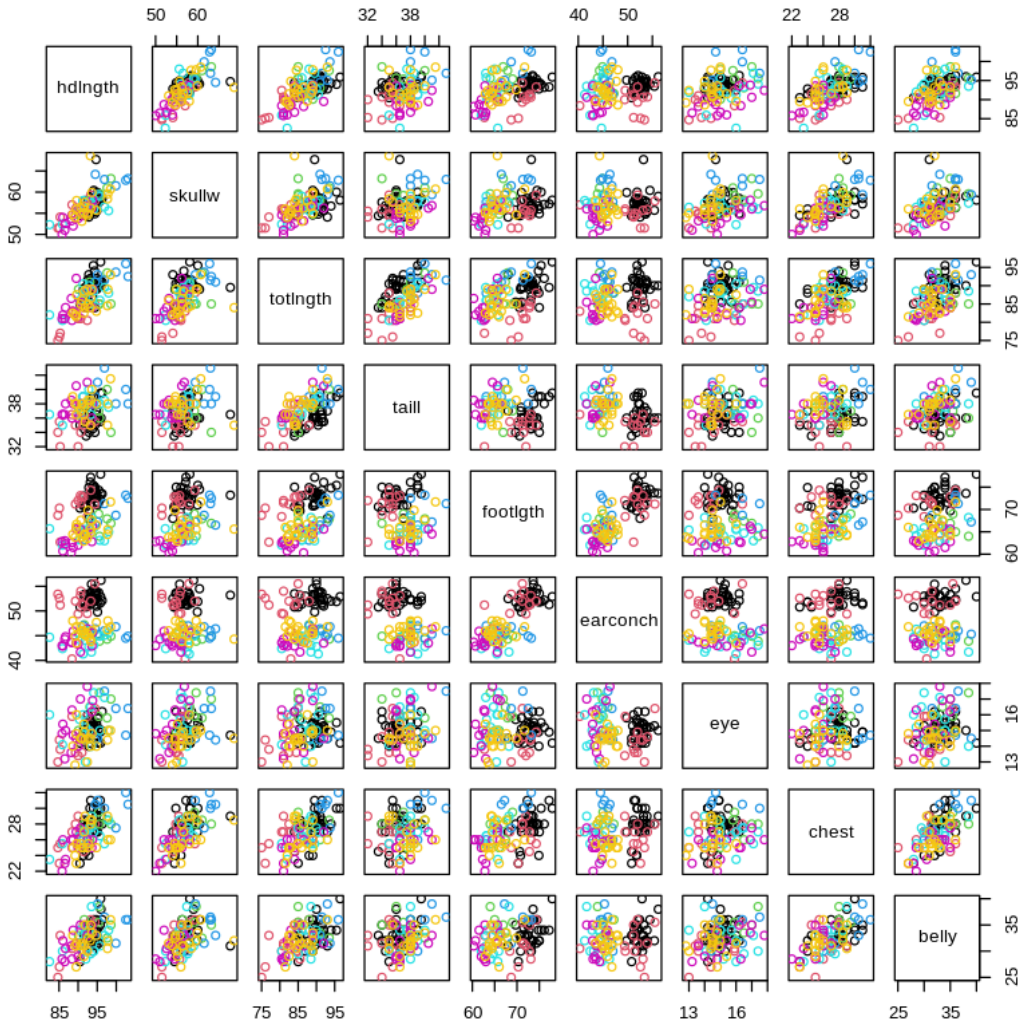
Overall, PCA helps in simplifying complex multivariate data while retaining most of the information, making it easier to identify patterns and relationships within the dataset.

[3]:
```
possum <- read.csv("/content/possum.csv")
pairs(possum[,6:14], col=palette()[as.integer(possum$sex)])
```

```
Warning message in pairs.default(possum[, 6:14], col =
palette()[as.integer(possum$sex)]):
"NAs introduced by coercion"
```



[5]:
```
pairs(possum[,6:14], col=palette()[as.integer(possum$site)])
```



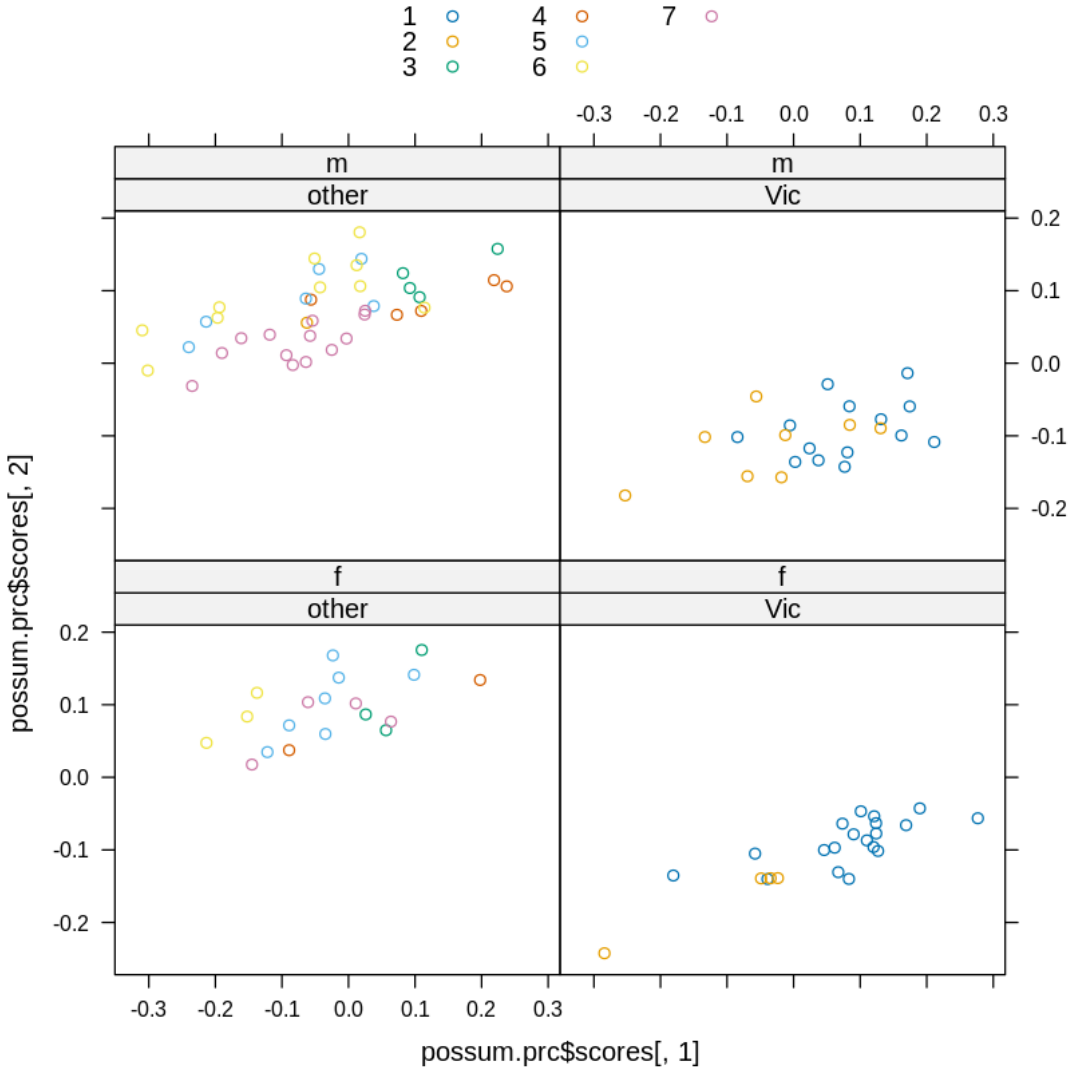
[6]:
```
here<-!is.na(possum$footlgth) # We need to exclude missing values
print(sum(!here)) # Check how many values are missing
```

```
[1] 1
```

[8]:
```
#We now look (Figure 21) at particular views of the data that we get from a␣
↪principal components analysis:

possum.prc <- princomp(log(possum[here,6:14])) # Principal components
```

[11]:
```
# Load the lattice package
library(lattice)
```

```
# Print scores on the second PC versus scores on the first PC,
# by populations and sex, identified by site
xyplot(possum.prc$scores[, 2] ~ possum.prc$scores[, 1] | possum$Pop[here] +␣
 ↪possum$sex[here],
       groups = possum$site,
       auto.key = list(columns = 3))
```



1.2 6.2 Cluster Analysis

Cluster analysis is a type of unsupervised classification where the clusters are not known beforehand. There are two main types of algorithms: hierarchical agglomeration and iterative relocation.

1. **Hierarchical Agglomeration**: Each observation starts as its own group. Similar groups

are then merged successively, creating a hierarchical clustering tree. Eventually, a judgement is made on when to stop merging further.

2. **Iterative Relocation**: This algorithm starts with an initial classification and tries to improve it. The initial classification is often obtained using hierarchical agglomeration.

In R, you can perform cluster analysis using the `mva` package. The `dist()` function calculates distances between observations, `hclust()` performs hierarchical agglomerative clustering, and `kmeans()` implements k-means clustering through iterative relocation.

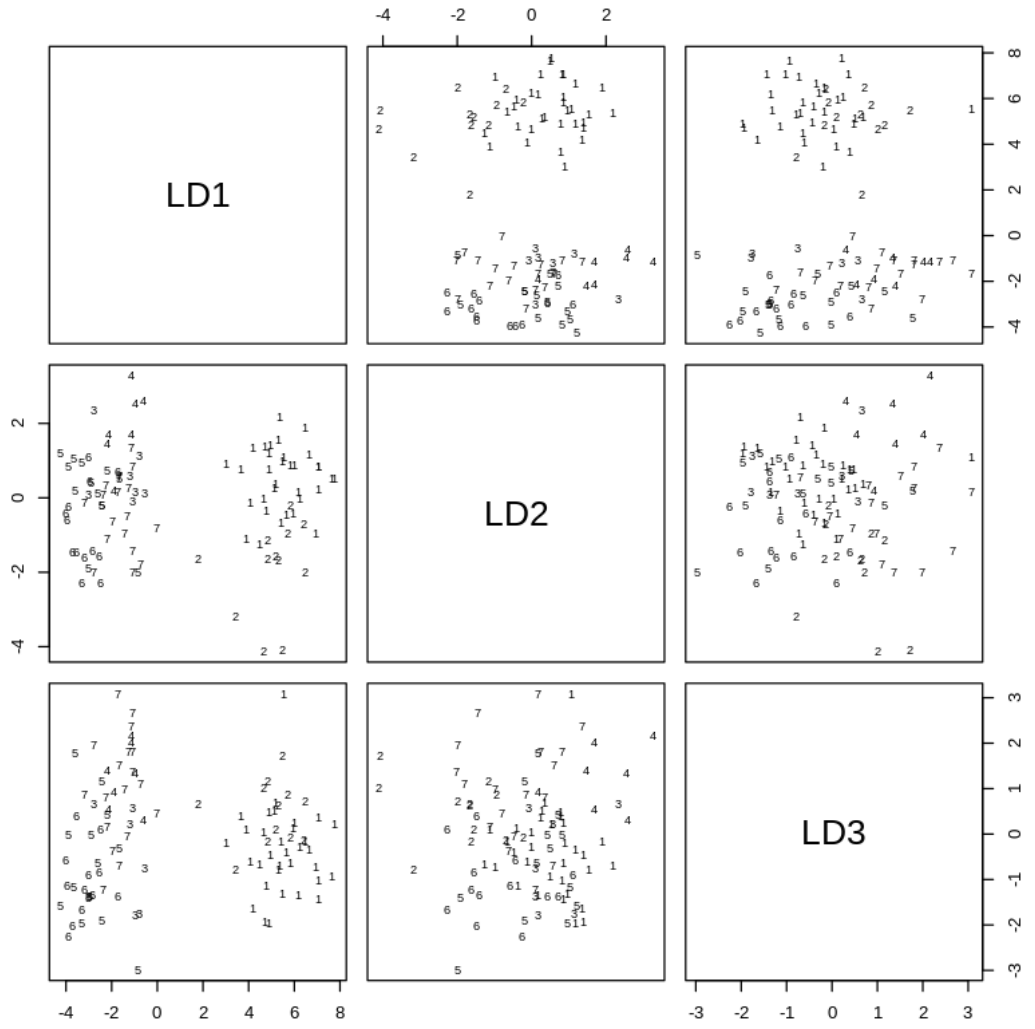
1.3 6.3 Discriminant Analysis

Supervised classification involves predicting the group to which new data will belong based on previous classifications. For example, we may want to predict whether future patients will remain free of disease symptoms for twelve months or more, based on measurements and outcomes of previous patients.

In the context of the possum dataset, we're using the `lda()` function from the MASS package to see if we can distinguish animals from different sites based on morphometric measurements. A simpler classification is between populations, such as sites in Victoria versus sites in other states like New South Wales or Queensland. Since we're mainly interested in comparing variable values, we haven't taken logarithms of the data. Further discussion on this is provided below.

[13]:
```
library(MASS) # Only if not already attached.
here <- !is.na(possum$footlgth)
possum.lda <- lda(site ~ hdlngth + skullw + totlngth +
                    taill + footlgth + earconch + eye + chest + belly,
                    data = possum, subset = here)
options(digits = 4)
possum.lda$svd # Examine the singular values
plot(possum.lda, dimen = 3)
# Scatterplot matrix for scores on 1st 3 canonical variates, as in Figure 22
```

1. 15.7577837861851 2. 3.93721357323141 3. 3.18597292414751 4. 1.50784610801749 5. 1.14201027231266 6. 0.777194725136929



The singular values represent the ratio of between-group to within-group sums of squares for each canonical variate. Larger singular values indicate greater discriminatory power in distinguishing between groups. Canonical variates beyond the third typically have minimal discriminatory power.

Using `predict.lda()`, we can obtain scores on the first few canonical variates.

Taking logarithms of biological measurement data can offer interpretative advantages. Allometric growth patterns, characterized by linear relationships between the logarithms of measurements, serve as a standard comparison. Differences between sites indicate variations in allometric growth patterns. Consider repeating the analysis using logarithms of measurements for further insights.

1.4 6.4 Decision Tree models (Tree-based models)

Tree-based classification is a method used for multivariate supervised classification or discrimination tasks. It's also applicable for regression problems with a tree-based regression approach. While

tree-based methods are well-suited for binary regression and classification, they may be less optimal for regression involving ordinal or continuous dependent variables.

Tree-based models, often referred to as "Classification and Regression Trees" (CART), are particularly effective when there's ample data available. One notable advantage of these methods is their automatic handling of non-linearities and interactions. The output typically includes a decision tree, which is directly applicable for making predictions.

[15]:
```
library(rpart)
# Use fgl: Forensic glass fragment data; from MASS package
glass.tree <- rpart(type ~ RI+Na+Mg+Al+Si+K+Ca+Ba+Fe, data=fgl)
plot(glass.tree); text(glass.tree)
summary(glass.tree)
```

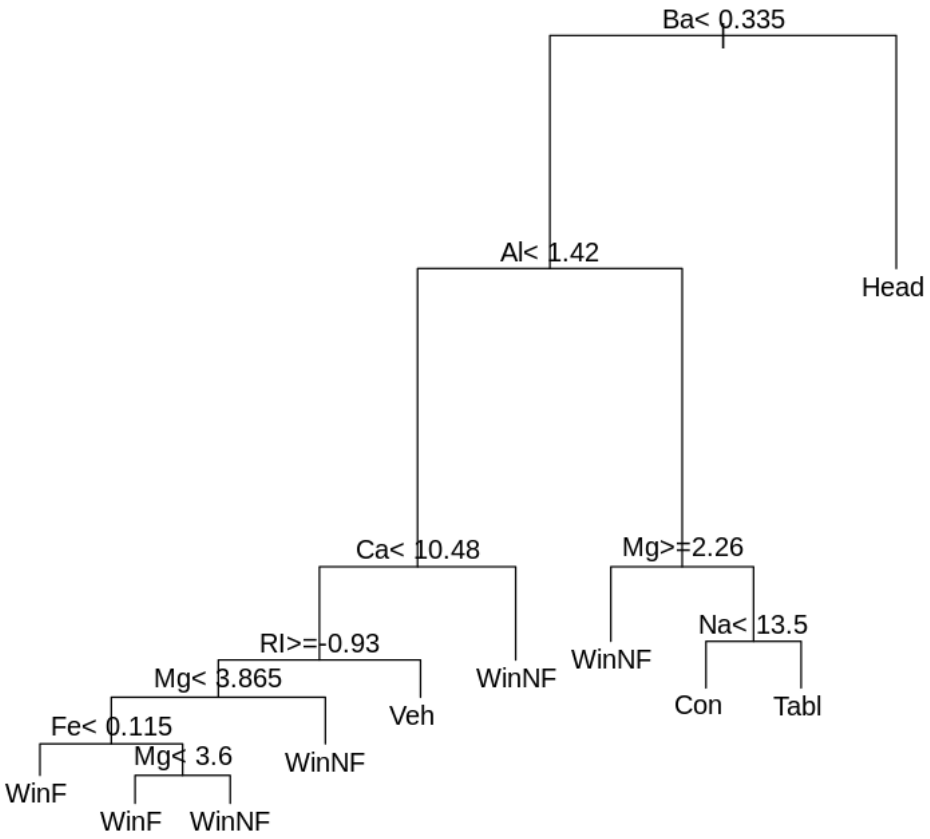
```
Call:
rpart(formula = type ~ RI + Na + Mg + Al + Si + K + Ca + Ba +
    Fe, data = fgl)
  n= 214

        CP nsplit rel error xerror    xstd
1 0.20652      0    1.0000 1.0507 0.04955
2 0.07246      2    0.5870 0.6014 0.05165
3 0.05797      3    0.5145 0.5652 0.05102
4 0.03623      4    0.4565 0.5290 0.05026
5 0.03261      5    0.4203 0.5072 0.04973
6 0.01087      7    0.3551 0.5290 0.05026
7 0.01000      9    0.3333 0.5145 0.04991

Variable importance
Mg Al Ca Ba RI Na  K Si Fe
18 16 15 14 13 10  8  5  2

Node number 1: 214 observations,    complexity param=0.2065
  predicted class=WinNF  expected loss=0.6449  P(node) =1
    class counts:    70    76    17    13     9    29
   probabilities: 0.327 0.355 0.079 0.061 0.042 0.136
  left son=2 (185 obs) right son=3 (29 obs)
  Primary splits:
      Ba < 0.335  to the left,  improve=26.04, (0 missing)
      Mg < 2.695  to the right, improve=21.53, (0 missing)
      Al < 1.775  to the left,  improve=20.04, (0 missing)
      Na < 14.06  to the left,  improve=17.50, (0 missing)
      K  < 0.055  to the right, improve=14.62, (0 missing)
  Surrogate splits:
      Al < 1.92   to the left,  agree=0.935, adj=0.517, (0 split)
      Na < 14.22  to the left,  agree=0.902, adj=0.276, (0 split)
      Mg < 0.165  to the right, agree=0.883, adj=0.138, (0 split)
      Ca < 6.56   to the right, agree=0.883, adj=0.138, (0 split)
```

```
      K  < 0.055  to the right, agree=0.879, adj=0.103, (0 split)

Node number 2: 185 observations,    complexity param=0.2065
  predicted class=WinNF  expected loss=0.5946  P(node) =0.8645
    class counts:    69    75    17    12     9     3
   probabilities: 0.373 0.405 0.092 0.065 0.049 0.016
  left son=4 (113 obs) right son=5 (72 obs)
  Primary splits:
      Al < 1.42   to the left,  improve=16.090, (0 missing)
      RI < -0.845 to the right, improve=12.240, (0 missing)
      Mg < 2.56   to the right, improve=11.820, (0 missing)
      Ca < 8.325  to the left,  improve=10.600, (0 missing)
      K  < 0.01   to the right, improve= 7.352, (0 missing)
  Surrogate splits:
      RI < -0.845 to the right, agree=0.757, adj=0.375, (0 split)
      Ca < 8.235  to the right, agree=0.746, adj=0.347, (0 split)
      K  < 0.625  to the left,  agree=0.730, adj=0.306, (0 split)
      Mg < 2.4    to the right, agree=0.665, adj=0.139, (0 split)
      Si < 73.1   to the left,  agree=0.627, adj=0.042, (0 split)

Node number 3: 29 observations
  predicted class=Head   expected loss=0.1034  P(node) =0.1355
    class counts:     1     1     0     1     0    26
   probabilities: 0.034 0.034 0.000 0.034 0.000 0.897

Node number 4: 113 observations,    complexity param=0.07246
  predicted class=WinF   expected loss=0.4425  P(node) =0.528
    class counts:    63    31    13     1     3     2
   probabilities: 0.558 0.274 0.115 0.009 0.027 0.018
  left son=8 (101 obs) right son=9 (12 obs)
  Primary splits:
      Ca < 10.48  to the left,  improve=8.668, (0 missing)
      Mg < 3.29   to the right, improve=7.343, (0 missing)
      RI < 5.65   to the left,  improve=4.840, (0 missing)
      K  < 0.01   to the right, improve=2.747, (0 missing)
      Si < 72.84  to the right, improve=2.194, (0 missing)
  Surrogate splits:
      Mg < 1.71   to the right, agree=0.965, adj=0.667, (0 split)
      RI < 5.65   to the left,  agree=0.947, adj=0.500, (0 split)
      Si < 71.29  to the right, agree=0.929, adj=0.333, (0 split)
      Na < 11.59  to the right, agree=0.912, adj=0.167, (0 split)

Node number 5: 72 observations,    complexity param=0.05797
  predicted class=WinNF  expected loss=0.3889  P(node) =0.3364
    class counts:     6    44     4    11     6     1
   probabilities: 0.083 0.611 0.056 0.153 0.083 0.014
  left son=10 (52 obs) right son=11 (20 obs)
  Primary splits:
```

```
      Mg < 2.26   to the right, improve=11.340, (0 missing)
      Ca < 9.245  to the left,  improve= 7.932, (0 missing)
      K  < 0.1    to the right, improve= 6.603, (0 missing)
      RI < -0.675 to the left,  improve= 6.246, (0 missing)
      Na < 13.99  to the left,  improve= 5.496, (0 missing)
  Surrogate splits:
      Ca < 9.245  to the left,  agree=0.931, adj=0.75, (0 split)
      RI < 0.8    to the left,  agree=0.889, adj=0.60, (0 split)
      Si < 73.38  to the left,  agree=0.792, adj=0.25, (0 split)
      K  < 0.745  to the left,  agree=0.792, adj=0.25, (0 split)
      Na < 13.88  to the left,  agree=0.778, adj=0.20, (0 split)

Node number 8: 101 observations,    complexity param=0.03261
  predicted class=WinF   expected loss=0.3762  P(node) =0.472
    class counts:    63    21    13     0     2     2
   probabilities: 0.624 0.208 0.129 0.000 0.020 0.020
  left son=16 (85 obs) right son=17 (16 obs)
  Primary splits:
      RI < -0.93  to the right, improve=5.534, (0 missing)
      Mg < 3.865  to the left,  improve=4.721, (0 missing)
      Fe < 0.115  to the right, improve=3.295, (0 missing)
      Ca < 8.525  to the right, improve=2.565, (0 missing)
      Si < 72.25  to the left,  improve=2.519, (0 missing)
  Surrogate splits:
      K  < 0.665  to the left,  agree=0.871, adj=0.188, (0 split)
      Na < 12.18  to the right, agree=0.861, adj=0.125, (0 split)
      Mg < 1.205  to the right, agree=0.861, adj=0.125, (0 split)
      Si < 74.5   to the left,  agree=0.861, adj=0.125, (0 split)
      Ca < 7.985  to the right, agree=0.851, adj=0.063, (0 split)

Node number 9: 12 observations
  predicted class=WinNF  expected loss=0.1667  P(node) =0.05607
    class counts:     0    10     0     1     1     0
   probabilities: 0.000 0.833 0.000 0.083 0.083 0.000

Node number 10: 52 observations
  predicted class=WinNF  expected loss=0.2115  P(node) =0.243
    class counts:     6    41     4     0     1     0
   probabilities: 0.115 0.788 0.077 0.000 0.019 0.000

Node number 11: 20 observations,    complexity param=0.03623
  predicted class=Con    expected loss=0.45  P(node) =0.09346
    class counts:     0     3     0    11     5     1
   probabilities: 0.000 0.150 0.000 0.550 0.250 0.050
  left son=22 (12 obs) right son=23 (8 obs)
  Primary splits:
      Na < 13.5   to the left,  improve=6.117, (0 missing)
      K  < 0.35   to the right, improve=3.343, (0 missing)
```

```
      RI < 1.425  to the right, improve=2.119, (0 missing)
      Si < 72.2   to the left,  improve=1.760, (0 missing)
      Ca < 11.08  to the right, improve=1.716, (0 missing)
  Surrogate splits:
      K  < 0.065  to the right, agree=0.85, adj=0.625, (0 split)
      Mg < 1.985  to the left,  agree=0.80, adj=0.500, (0 split)
      Ca < 10.02  to the right, agree=0.75, adj=0.375, (0 split)
      RI < 1.015  to the right, agree=0.65, adj=0.125, (0 split)
      Al < 1.75   to the right, agree=0.65, adj=0.125, (0 split)

Node number 16: 85 observations,    complexity param=0.03261
  predicted class=WinF   expected loss=0.2941  P(node) =0.3972
    class counts:    60    17     6     0     1     1
   probabilities: 0.706 0.200 0.071 0.000 0.012 0.012
  left son=32 (77 obs) right son=33 (8 obs)
  Primary splits:
      Mg < 3.865  to the left,  improve=5.680, (0 missing)
      Fe < 0.185  to the left,  improve=4.038, (0 missing)
      RI < -0.125 to the left,  improve=3.539, (0 missing)
      Ca < 8.525  to the left,  improve=2.874, (0 missing)
      K  < 0.29   to the right, improve=2.514, (0 missing)

Node number 17: 16 observations
  predicted class=Veh    expected loss=0.5625  P(node) =0.07477
    class counts:     3     4     7     0     1     1
   probabilities: 0.188 0.250 0.438 0.000 0.062 0.062

Node number 22: 12 observations
  predicted class=Con    expected loss=0.08333  P(node) =0.05607
    class counts:     0     1     0    11     0     0
   probabilities: 0.000 0.083 0.000 0.917 0.000 0.000

Node number 23: 8 observations
  predicted class=Tabl   expected loss=0.375  P(node) =0.03738
    class counts:     0     2     0     0     5     1
   probabilities: 0.000 0.250 0.000 0.000 0.625 0.125

Node number 32: 77 observations,    complexity param=0.01087
  predicted class=WinF   expected loss=0.2338  P(node) =0.3598
    class counts:    59    11     5     0     1     1
   probabilities: 0.766 0.143 0.065 0.000 0.013 0.013
  left son=64 (57 obs) right son=65 (20 obs)
  Primary splits:
      Fe < 0.115  to the left,  improve=3.744, (0 missing)
      RI < 0.015  to the left,  improve=2.270, (0 missing)
      Mg < 3.615  to the left,  improve=2.232, (0 missing)
      Al < 1.265  to the right, improve=1.950, (0 missing)
      Na < 13.62  to the left,  improve=1.719, (0 missing)
```

```
  Surrogate splits:
      Si < 73.47  to the left,  agree=0.779, adj=0.15, (0 split)
      Na < 12.51  to the right, agree=0.766, adj=0.10, (0 split)
      K  < 0.605  to the left,  agree=0.766, adj=0.10, (0 split)
      Ba < 0.045  to the left,  agree=0.766, adj=0.10, (0 split)
      Mg < 2.785  to the right, agree=0.753, adj=0.05, (0 split)

Node number 33: 8 observations
  predicted class=WinNF  expected loss=0.25  P(node) =0.03738
    class counts:     1     6     1     0     0     0
   probabilities: 0.125 0.750 0.125 0.000 0.000 0.000

Node number 64: 57 observations
  predicted class=WinF   expected loss=0.1404  P(node) =0.2664
    class counts:    49     3     3     0     1     1
   probabilities: 0.860 0.053 0.053 0.000 0.018 0.018

Node number 65: 20 observations,    complexity param=0.01087
  predicted class=WinF   expected loss=0.5  P(node) =0.09346
    class counts:    10     8     2     0     0     0
   probabilities: 0.500 0.400 0.100 0.000 0.000 0.000
  left son=130 (10 obs) right son=131 (10 obs)
  Primary splits:
      Mg < 3.6    to the left,  improve=1.6000, (0 missing)
      RI < 0.6    to the left,  improve=1.5170, (0 missing)
      K  < 0.55   to the right, improve=1.5170, (0 missing)
      Al < 1.17   to the right, improve=1.2670, (0 missing)
      Na < 13.04  to the left,  improve=0.8121, (0 missing)
  Surrogate splits:
      Si < 72.61  to the right, agree=0.85, adj=0.7, (0 split)
      RI < -0.04  to the left,  agree=0.80, adj=0.6, (0 split)
      Na < 12.9   to the left,  agree=0.80, adj=0.6, (0 split)
      Al < 1.135  to the right, agree=0.75, adj=0.5, (0 split)
      Ca < 8.93   to the left,  agree=0.65, adj=0.3, (0 split)

Node number 130: 10 observations
  predicted class=WinF   expected loss=0.3  P(node) =0.04673
    class counts:     7     2     1     0     0     0
   probabilities: 0.700 0.200 0.100 0.000 0.000 0.000

Node number 131: 10 observations
  predicted class=WinNF  expected loss=0.4  P(node) =0.04673
    class counts:     3     6     1     0     0     0
   probabilities: 0.300 0.600 0.100 0.000 0.000 0.000
```



To effectively use these models, understanding tree pruning and cross-validation is crucial. Methods for reducing tree complexity, based on significance tests at each node, often result in trees that over-predict.

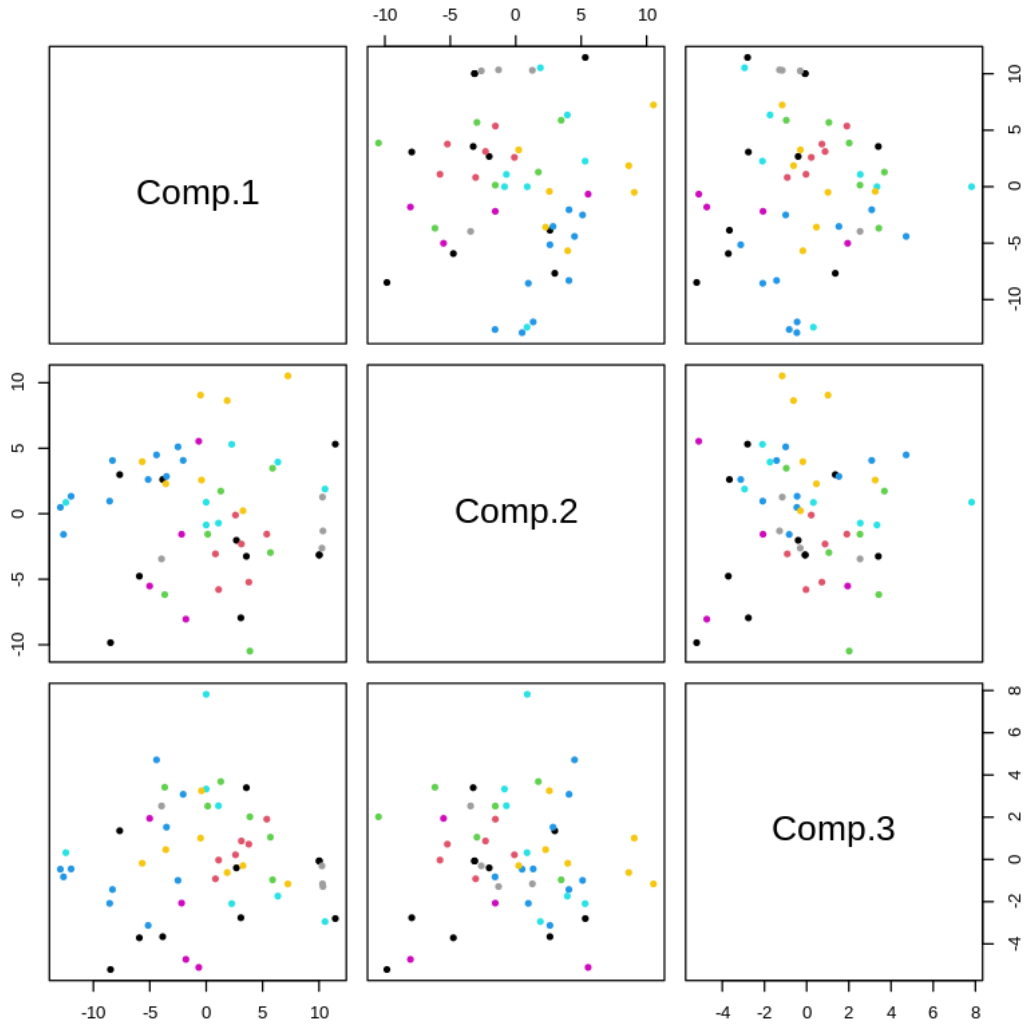
The rpart package by Atkinson and Therneau for recursive partitioning is more aligned with CART than the S-PLUS tree library. It incorporates cross-validation into the tree formation algorithm.

1.5 6.5 Exercises

1. Using the data set painters (MASS package), apply principal components analysis to the scores for Composition, Drawing, Colour, and Expression. Examine the loadings on the first three principal components. Plot a scatterplot matrix of the first three principal components, using different colours or symbols to identify the different schools

[16]:
```r
# Load required library
library(MASS)

# Load the painters dataset
data(painters, package = "MASS")

# Extract the scores for Composition, Drawing, Colour, and Expression
painters_scores <- painters[, c("Composition", "Drawing", "Colour",␣
 ↪"Expression")]

# Perform principal components analysis
pca_result <- princomp(painters_scores)

# Examine the loadings on the first three principal components
print(pca_result$loadings[, 1:3])

# Plot a scatterplot matrix of the first three principal components
pairs(pca_result$scores[, 1:3], col = painters$School, pch = 20)
```

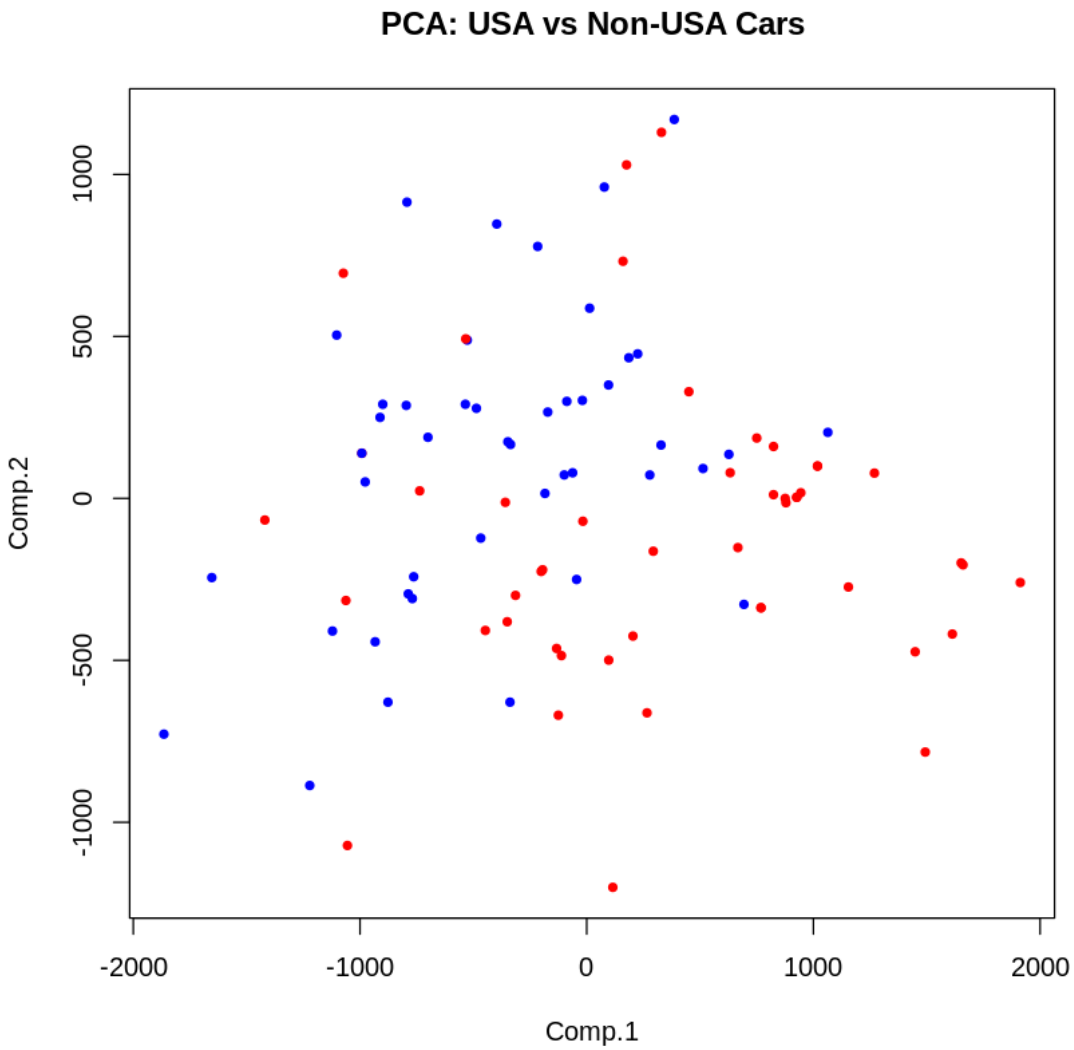
```
             Comp.1  Comp.2  Comp.3
Composition  0.4835  0.3764  0.7838
Drawing      0.4240 -0.1872 -0.2797
Colour      -0.3808  0.8452 -0.2108
Expression   0.6644  0.3299 -0.5128
```

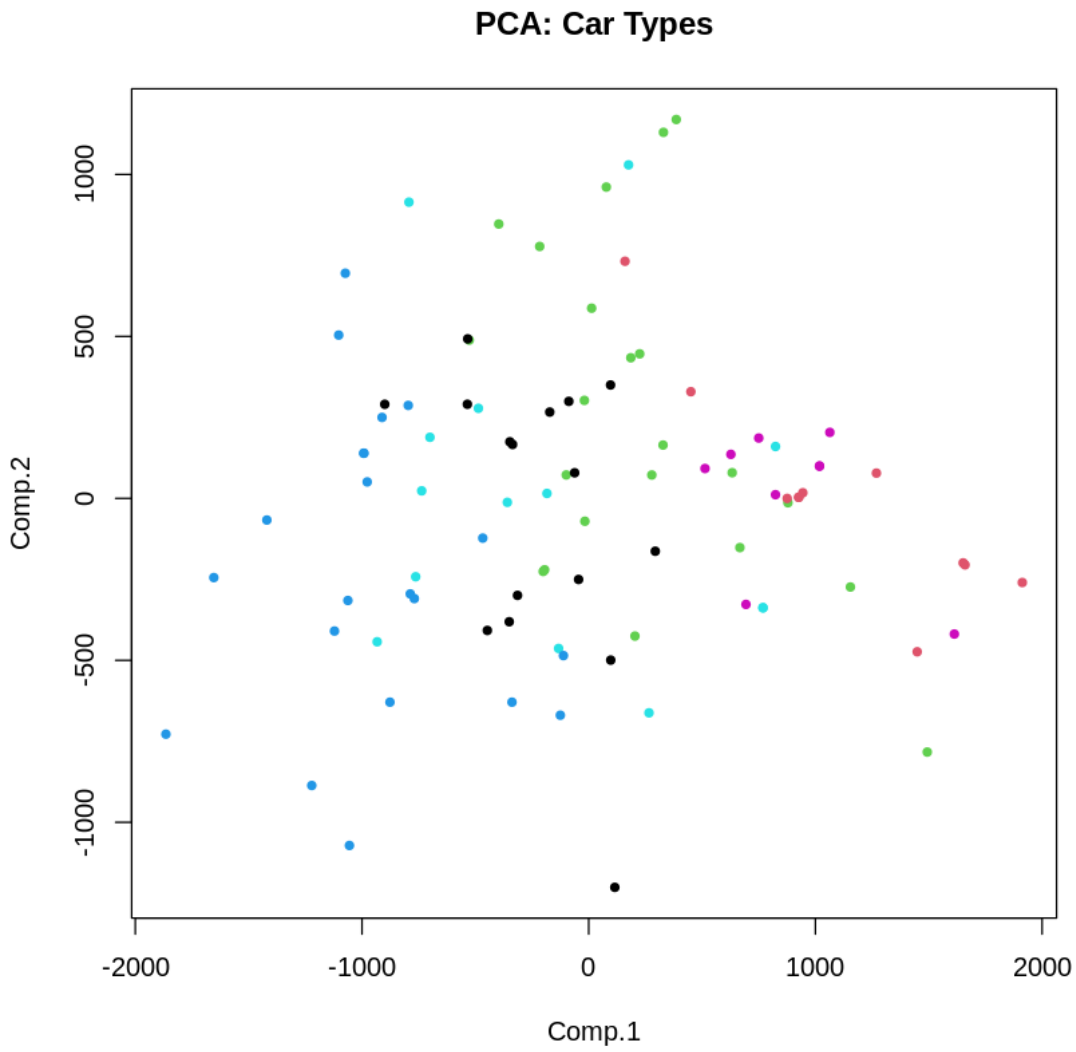


2. The data set Cars93 is in the MASS package. Using the columns of continuous or ordinal data, determine scores on the first and second principal components. Investigate the comparison between (i) USA and non-USA cars, and (ii) the six different types (Type) of car. Now create a new data set in which binary factors become columns of 0/1 data, and include these in the principal components analysis

[18]:
```
# Load required library
library(MASS)

# Load the Cars93 dataset
data(Cars93, package = "MASS")

# Select columns of continuous or ordinal data
```

```r
cars_data <- subset(Cars93, select = c("Price", "MPG.city", "MPG.highway",␣
 ↪"EngineSize", "Horsepower", "RPM", "Rev.per.mile", "Fuel.tank.capacity",␣
 ↪"Length", "Width", "Turn.circle", "Weight"))

# Perform principal components analysis
pca_result <- princomp(cars_data)

# Extract scores on the first and second principal components
scores <- predict(pca_result, newdata = cars_data)[, 1:2]

# Identify whether a car is from the USA or not
comparison_usa <- ifelse(Cars93$Origin == "USA", "red", "blue")  # Use colors␣
 ↪for visualization

# Investigate comparison between car types
car_types <- as.factor(Cars93$Type)

# Plot scores with different colors for USA and non-USA cars
plot(scores, col = comparison_usa, pch = 20, main = "PCA: USA vs Non-USA Cars")

# Plot scores with different colors for car types
plot(scores, col = car_types, pch = 20, main = "PCA: Car Types")
```


PCA: USA vs Non-USA Cars
Comp.2
1000
500
0
-500
-1000
-2000
-1000
0
1000
2000
Comp.1

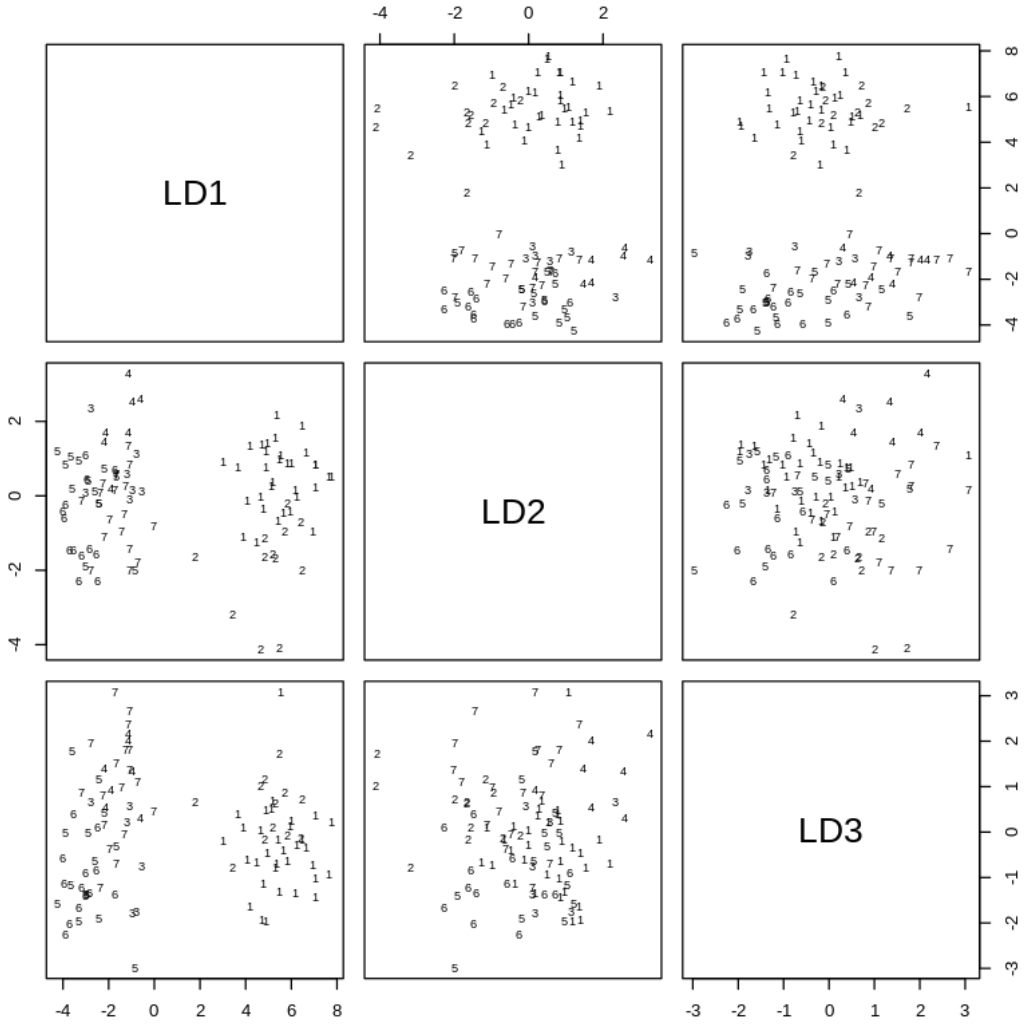


3. Repeat the calculations of exercises 1 and 2, but this time using the function lda() from the MASS package to derive canonical discriminant scores, as in section 6.3.

[20]:
```
# Load required library
library(MASS)

# Filter out NA values
here <- !is.na(possum$footlgth)

# Perform Linear Discriminant Analysis (LDA)
possum.lda <- lda(site ~ hdlngth + skullw + totlngth +
                    taill + footlgth + earconch + eye + chest + belly,
                  data = possum, subset = here)
```

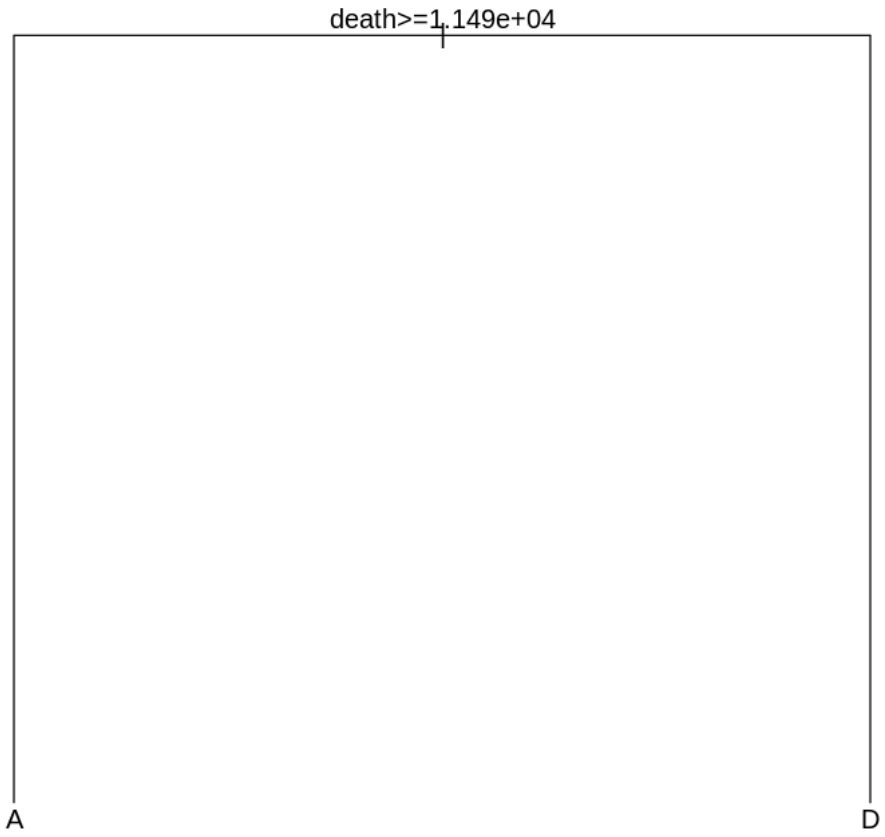
```
# Examine the singular values
options(digits = 4)
possum.lda$svd

# Plot a scatterplot matrix for scores on the first three canonical variates
plot(possum.lda, dimen = 3)
```

1. 15.7577837861851 2. 3.93721357323141 3. 3.18597292414751 4. 1.50784610801749 5. 1.14201027231266 6. 0.777194725136929



4. The MASS package has the Aids2 data set, containing de-identified data on the survival status of patients diagnosed with AIDS before July 1 1991. Use tree-based classification (rpart()) to identify major influences on survival.

[23]:
```r
# Load required library
library(MASS)
library(rpart)

# Load the Aids2 dataset
data(Aids2, package = "MASS")

# Perform tree-based classification using rpart()
tree_model <- rpart(status ~ ., data = Aids2)

# Visualize the resulting tree
plot(tree_model)
text(tree_model)
```



5. Investigate discrimination between plagiotropic and orthotropic species in the data set leafshape34.

[25]:
```r
# Load the leafshape dataset
leafshape <- read.csv("/content/Leafshape.csv")

# Perform discriminant analysis
discriminant_model <- lda(Species ~ ., data = leafshape)

# Plot discriminant scores
plot(discriminant_model)
```