*The following supplement accompanies the article*

# Disentangling and quantifying sources of otolith shape variation across multiple scales using a new hierarchical partitioning approach

**M. Vignon***

*Corresponding author: matthias.vignon@univ-pau.fr

*Marine Ecology Progress Series 534: 163–177 (2015)*

---

## Overview

This is a brief tutorial to accompany a set of functions that I have written to facilitate disentangling sources of otolith shape variation. The novelty of the proposed hierarchical disparity method (i.e. partitioning of morphological variation into contributions from two or more subgroups) relies on the conjugation of the *multivariate regression tree* (MRT) embedded in a *geometric morphometric context*. The tutorial provides ready-to-use functions and makes their use easy to apply to typical ichthyological data.

The tutorial is aimed at helping you to use the functions and to develop a MRT model in R. It does not explain what a MRT model is. For detailed explanations and example applications, see the following references at the end of the tutorial (De'ath & Fabricius 2000, De'ath 2002, 2007, Elith et al. 2008). Please, note that I developed these codes for my own research, and it simply builds on the base functions provided within the `mvpart` package by Therneau, Atkinson, Ripley, Oksanen, and De'ath (Therneau et. al. 2004, *http://cran.r-project.org/web/packages/mvpart/index.html*). Read this documentation to find out more about that. Whilst I am willing to answer simple questions and to fix code that doesn't work where possible, I apologize in advance that I will not be able to spend much time in this capacity. If you use the code in published research, I would appreciate your acknowledgement by referring to me as its author.

***Contact details:***

Dr Matthias Vignon

UPPA, UFR Sciences & Techniques de la Côte Basque
1 Allée du parc Montaury
64600 Anglet, France.
Tel.: +33 5 59 57 44 48
Email: matthias.vignon@univ-pau.fr
mvignon@st-pee.inra.fr

# Content

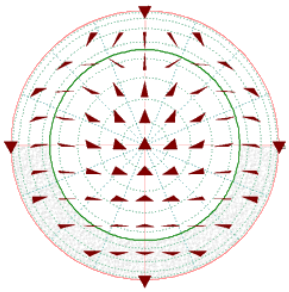
## I. General software considerations

To run the examples you need to download and install R. R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows, and MacOS. R is `GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modeling, statistical tests, time series analysis, classification, clustering, *etc.* CRAN (*cran.r-project.org/*) is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R.

While this tutorial is addressed to beginners and has been written as a stand-alone manuscript that does not request particular knowledge, it is recommended to understand how to use R prior in using my functions. This is not covered here but could help appropriating and developing the codes for your own analysis. For this purpose, you can read any of a number of books and tutorials to develop your knowledge - see the list on the R homepage (*http://www.r-project.org/*) under "Documentation".

For peoples interested in regression tree that would like to improve one's knowledge, there are a couple of excellent manuscripts on the web describing the subject. In particular, you can look at *An Introduction to Recursive Partitioning Using the RPART Routines* (2015) by Therneau and Atkinson:

*http://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf*



While data analysis is performed using R, data acquisition [including among others, position of (semi-)landmarks, sliding of semi-landmarks and generalized orthogonal least square procrustes superpositions] and shape visualisation [e.g. thin-plate spline deformation grids] are not covered here and were computed with other appropriate software. An (almost exhaustive and up-to-date) list of the available software in the field is available on the Stony Brook University (*http://life.bio.sunysb.edu/morph/*). The high contrast images were thresholded and binarized for contour extraction by ImageJ software (freely available at http://rsb.info.nih.gov/ij/). The coordinates of the landmark and semi-landmarks were digitized using the software tpsDig version 2.10 and all subsequent morphological analysis were performed using the tps package (from the TPS package, Rohlf 2006).

**In the following sections all R code is in blue, consolas font, with grey background.**

## II. Installing *mvpart*

**Mvpart** (mvpart.zip) must be saved on the local computer before being installed. The package can then be manually installed using the related Windows binaries. Please, note that the *mvpart* package is no longer maintained on CRAN (archived on 2014-12-13) and the available archive (mvpart_1.6-2.tar.gz) may not compile neither install properly on your computer. The author have set up an R-forge repository for mvpart (*http://r-forge.r-project.org/projects/mvpart/*) that will host the future development of the R-package mvpart and provide access for users of mvpart to an up-to-date version of the package. Unfortunately, the repository is currently (May 2015) empty and the user need to use an old Windows binaries (mvpart.zip) from the provided supplementary files. This Windows compiled version of the package works well but can only be used on Windows machines. Then, use Packages -> Install package(s) from local zip files ... from menu bar (Fig.1). Alternatively, one may try to automatically install the package using the "*URLinstall.R*" script. After installation, package can be loaded manually through the *Load package*... button from the same *Packages* menu (Fig.1).

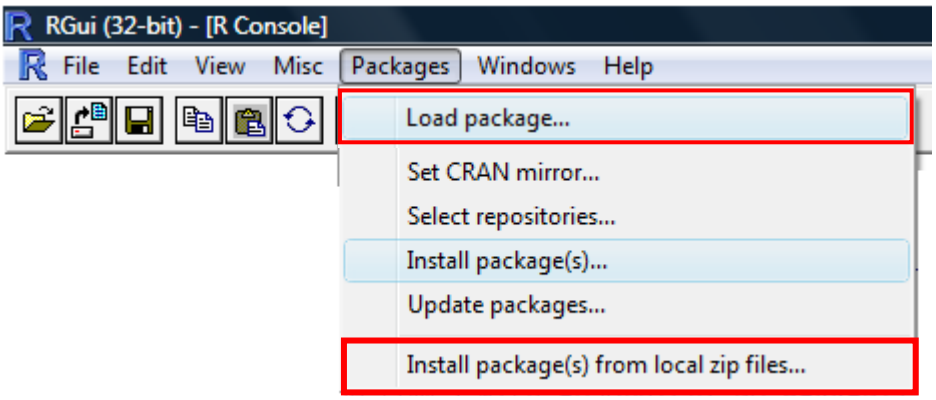



**Fig.1** The *Packages* menu allows to install and load the mvpart package

After installation, users can alternatively load the `mvpart` package using the following command*:

```
> library(mvpart)
```

*It may be necessary to install other packages (dependencies) depending on your R version.

**Note**: from here on if you prefer to run code from within R, you can use the file: `Tutorial_cutdown.R` Open it within R (File/Open script..) and when you come on a line you want to run, just highlight the line and send it to R (through the following button : ). See R documentation on running scripts if you can't work this out. Alternatively, you can copy and paste directly from this tutorial (omitting the ">" symbol).

Let's start

## III. Importing and checking simple example data

Raw data can be organized in any spreadsheet software, such as Microsoft Excel or Open Office. The specific format that best suits for morphometric consists in either a distance matrix between all pairs of otoliths or raw coordinates (Fig.2a). For the matrix, distances MUST be provided as Procrustes distances in the tangent space. For 2D raw data, X and Y coordinates must be in separated columns. For each otolith, all successive (semi-)landmarks must be provided in consecutive columns. Each column must contains homologous coordinates among otoliths (this is automatically achieved during the generalized orthogonal least square procrustes superpositions by sliding semi-landmarks). In addition the user needs a (possibly separate) file containing potential explicative variables (Fig.2b) that can be either categorical, continuous or both.

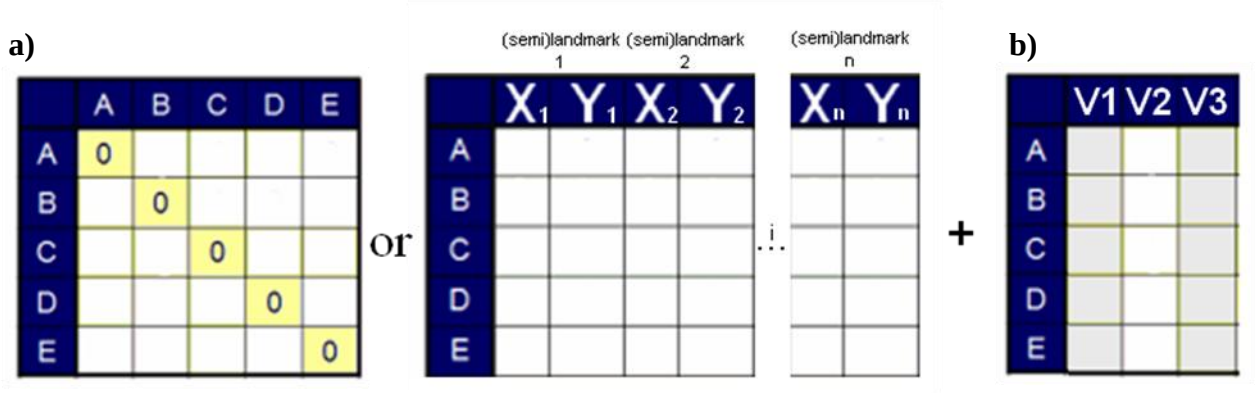



**Fig.2** Example of raw morphological data from five otoliths (from A to E) (Fig.2a) and three potential explicative variables (from V1 to V3) (Fig.2b).

Please, before importing data, make sure that file(s) is(are) located in the working directory. To be sure about the current directory or change it, type the following commands:

```
> getwd()
> setwd("C:/Users/…")
> source("MRTG.R")          # source new functions from the current directory
```

Users can alternatively browse the working directory manually through the *File* menu (Fig.3).

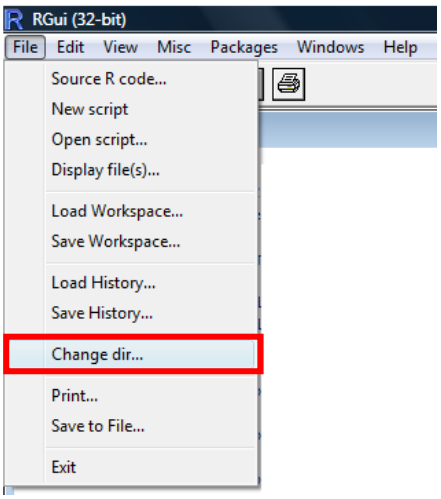
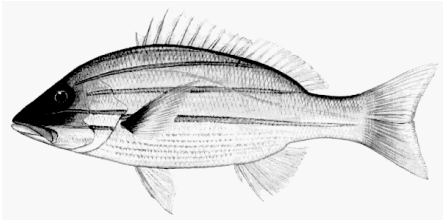



**Fig.3** How to change manually the current directory.

Before analyzing more complex dataset, we will primarily focus on a really simple dataset containing only 16 otoliths from *Lutjanus kasmira* (only one otolith per individual). Data is stored in the provided "Coord1.rda" file that can be loaded using the following command:



```
> load("Coord1.Rda")          # load raw "data"
> dim(data)                   # number of lines and columns
> head(data)                  # first lines
```

Once loaded, a "data" object is accessible in R. It contains 1600 lines (16 otoliths, 100 (semi-)landmarks per otolith) and two columns ("X" and "Y" coordinates, respectively) exported from the tps software. You can look the coordinates (Fig.4a) using:

```
> plot(data,col="black",pch=21,bg="grey",asp=1)
```

Individual outlines can also be visualized. This will results in the following figure 4b.

```
> plot(data,col="white",xaxt="n",yaxt="n",asp=1,bg="red")
> for (j in 0:15) {
    for (i in 1:99) {
      segments(data[100*j+i,1],data[100*j+i,2],
      data[100*j+i+1,1],data[100*j+i+1,2],col="black")
    }
    segments(data[100*j+1,1],data[100*j+1,2],
    data[100*j+100,1],data[100*j+100,2],col="black")
  }
```

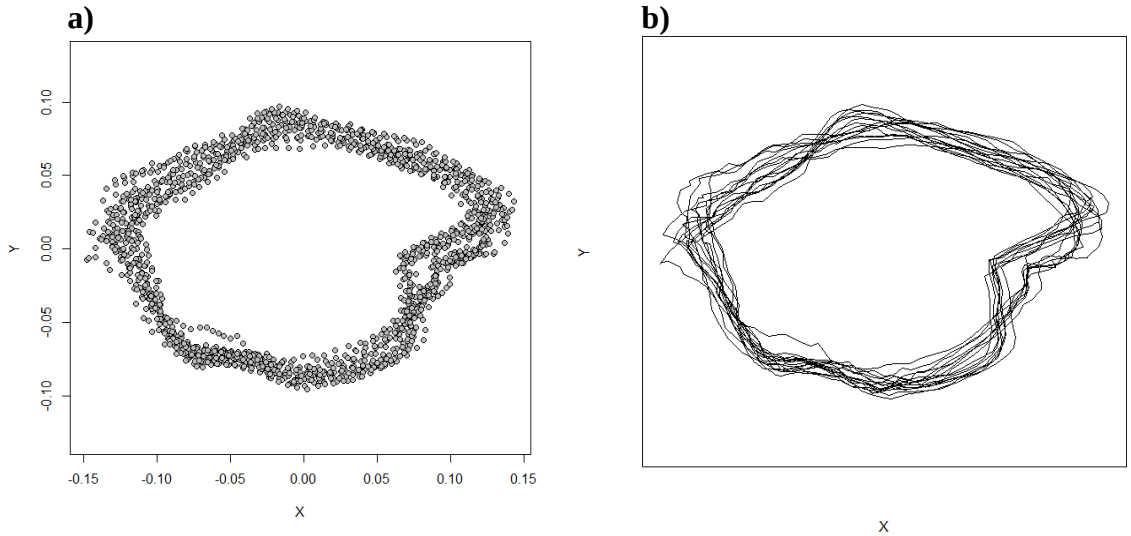



**Fig.4** Original Procruste superimposition of 16 otolith coordinates (a) and outlines (b).

Potential explicative variables are stored in the provided "Var1.rda" file that contains an individual identifier, two factorial variables and one numeric (Table1). The file can be loaded using the following command:

```
> load("Var1.Rda")            # load "Var"
```

**Table1.** Potential explicative variables. ID is not a predictive variable, just otolith identifier.

| **Variables** | | ID | Var1 | Var2 | Var3 |
|---|---|---|---|---|---|
| **type** | | Factor | Factor | Factor | Numeric |
| **values** | | Lk01-16 | "A" "B" | "a" "b" "c" "d" | [9-34] |

Before being analyzed, "data" needs to be formatted into the appropriate format (see figure 2) and combined with "Var" in a single object called "Newdata" using the following command:

```
> Coord<-matrix(nrow=16,ncol=200)
> Coord[1,]<-c(as.matrix(data[1:100,]))
  for (i in 1:15) {
     Coord[i+1,]<-c(as.matrix(data[((100*i)+1):((100*i)+100),]))
  }
> Newdata<-data.frame(cbind(Coord,Var))
> dim(Newdata)                          # number of lines and columns
```

"Newdata" now contains all the elements to perform the multivariate regression tree. The first 200 columns contain 2D coordinates of the 100 (semi-)landmarks. The last three columns (respectively named "Var1", "Var2" and "Var3") contain the potential explicative variables. "ID" will not be used in the current analysis as it correspond to individual identifiers. Please, remind that no statistical assumptions exist with this data mining method. There is therefore no preliminary test to compute and we can proceed immediately with the data (Fig.5).

```
> library(mvpart)                       # load the mvpart package
> mrt1<-mvpart(data.matrix(Newdata[,1:200])~Var1+Var2+Var3,
  Newdata,size=4)
```

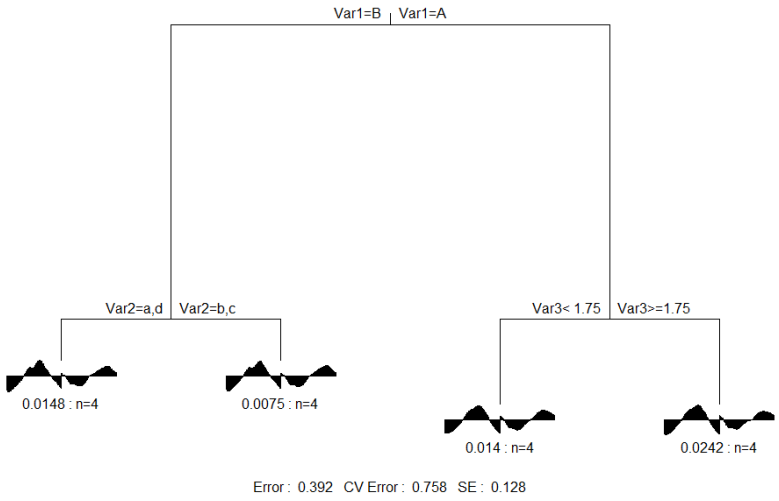



**Fig.5** Typical presentation of a regression tree created using the *mvpart* package. Here, the first tree splits (i.e. resulting in four groups, each of them containing 4 otoliths "n=4") are shown. The overall tree explains roughly 60% of total morphological variance using the three explicative variables available. Branch lengths are proportional to percentage of explained variance and splits are numbered in descending order of importance.

Figure 5 presents a typical regression tree. In this example, we asked the *mvpart* function to create four groups (using 3 internal splits) using the `Size=4` argument. MRT is a tree-building recursive method that constructs a hierarchical tree by repeatedly splitting the set of observations into two mutually exclusive subgroups, each of which is as homogeneous as possible. MRT can handle numerical as well as categorical variables. For categorical explanatory variables with more than two levels, any combinations of levels can be used to form a split and for $k$ levels there are $2^{k-1}$-1 possible splits. For numerical explanatory variables, a split is defined by values less than, and greater than, some chosen value. Thus, only the rank order of numerical variables determines a split, and for $u$ unique values there are $u$-1 possible splits. At each step, from all possible splits of all explanatory variables, the method selects the one that maximize the between group variation. In this example, Var1 appears as the most important variable when discriminating otoliths with Var1="B" from whose with Var1="A". The second most important split concerns the Var1="B" group that can be splited into two groups on the basis of the Var2 variable (one group with Var2="a" or "d", and another group with either Var2="b" or "c"). Ultimately, the Var1="A" group can be splited using the numeric variable Var3. Otoliths for which Var3<1.75 are separated from whose with Var3 ≥1.75.

The relative Error ("Error", see bottom of Figure 5) is described as the fit of the tree. Therefore, the variance explained by the tree is the inverse of the Error. However, the relative error gives an over-optimistic view of how the tree will predict new data. This is better described by the "CV-Error" (i.e. based on cross-validation). The CV-error varies from 0 for a perfect predictor to 1 for a poor predictor (De'ath 2002). For the specific contribution of each split, remember that branch lengths are proportional to percentage of explained variance. This clearly signify that in this Example, the contribution of Var1>Var3>Var2. Unfortunately, visual interpretation can be tedious for complex tree. Numerical investigation (% of explained variance) for each split can be investigated using the following command:

```
> mrt1$cptable
           CP nsplit rel error    xerror      xstd
0.42638574      0 1.0000000 1.1313224 0.1510864
0.12254140      1 0.5736143 0.7314498 0.1210605
0.05921997      2 0.4510729 0.7739648 0.1406124
0.01000000      3 0.3918529 0.7582066 0.1283965
> abs(diff(mrt1$cptable[,3])) # variance explained by each split
```

When "nsplit" is 0 the relative error is 1, so the variance explained (1-relative error) is 0. When "nsplit" is 1 the relative error is 0.57, so the variance explained by the first split is 0.43 When "nsplit" is 2 the relative error is 0.45, so the variance explained by both splits is 0.55 (0.12 for the second split. Splits are numbered in descending order of importance and specific contribution is simply obtained by subtracting consecutive relative error). Ultimately, when "nsplit" is 3 the relative error is 0.39, so the variance explained by overall tree is 0.61 with 6% of morphological variance explained by the third split alone.

The fit of the model using different sizes is depicted in the figure 6. The following command produces two additional plots. The first plots the apparent and relative (from the cross-validation) R-square values for various tree size (the number of splits appear in abscissa, from 0 to Size-1). The second plots the Relative Error (from the cross-validation) ± 1-SE from cross-validation versus the number of splits. Note that the two plots are closely related due to the fact that R-square = 1- Relative Error.

```
> par(mfrow=c(1,2));rsqrpart(mrt1)
```

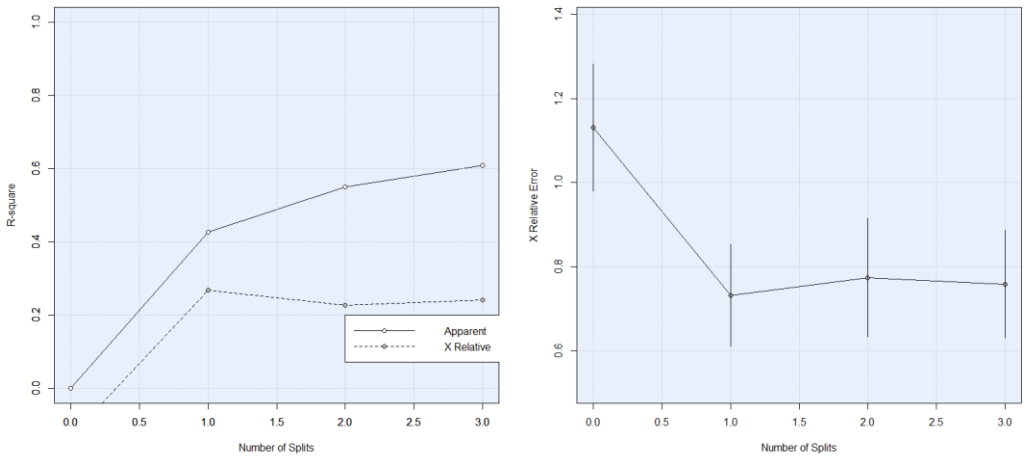



**Fig.6** $R^2$ (left) and cross-validation relative error (right) for the first three splits.

At this point, it is possible to combine information from figure 5 and cptable to visualize the percentage of morphological variance explained at each split. For this purpose, the excellent 'MVPARTwrap' package (Ouellette et al. 2012) is appropriate. Unfortunately, due to its dependency to mvpart, the package was archived and hardly usable. Here [from source("MRTG.R") ] I embedded main function that does the job properly:

```
> m<-MRT(mrt1)
> plot(m)
```

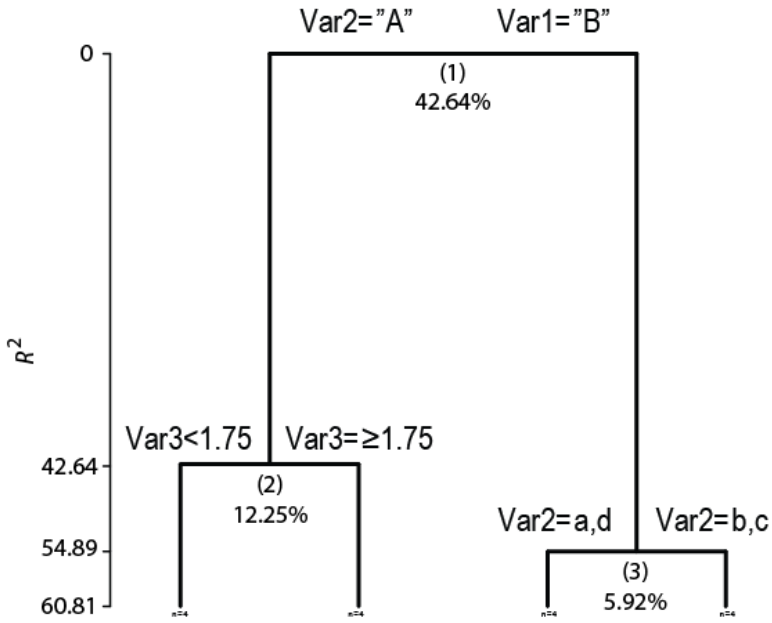



**Fig.7** Regression tree with percentage of explained variance at each split and (cumulative) $R^2$ in relation to tree size.

The contributions of each variable is simply determined by summing the percentage contributions of all the splits associated with this variable. In the present study, each variable appear only once in the tree. The relative contribution of Var1 is therefore 70.12% (100/60.81*42.64), Var3: 20.14% and Var2 9.73% (see table2).

**Table2.** Relative and absolute contribution of each variable.

| **Variables** | | % explained variance | Total variance explained | Relative contribution (∑=100) |
|---|---|---|---|---|
| Var1 | | 42.64 | 60.81 | 70.12 |
| Var2 | | 5.92 | | 9.73 |
| Var3 | | 12.25 | | 20.14 |

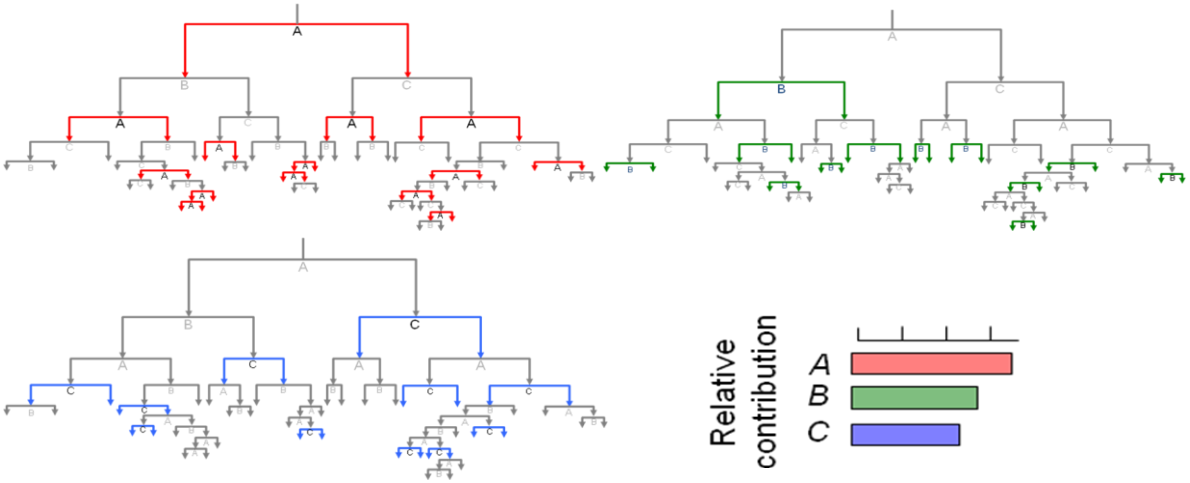



**Fig.8** Hypothetical regression tree depicting the variables (either A, B or C) associated with each split using colours. Each variable is represented by many splits and the contribution of each variable is obtained by summing the contributions of all associated splits.

Unfortunately, this can be complicated for large and complex trees, as for example in figure 8. Here I provide a new function that extract the relative importance of each variable for any tree obtained with the *mvpart* function:

```
> importance<-VarImp(mrt1,pretty=F)     # pretty=T for coloured plot
> importance$Relative                   # to access numerical values
```

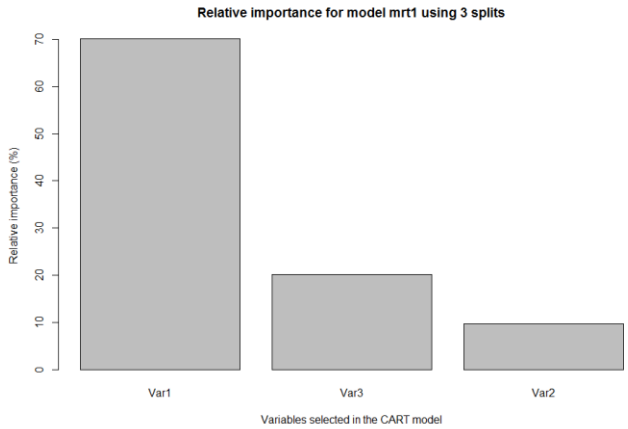



**Fig.9** Relative contribution of all variables from an *mvpart* object using the VarImp function

In a morphological context, graphs at the bottom of the figure 5 are not directly interpretable. To visualize otolith morphology within each group, we need first to extract group belonging:

```
> gr<-mrt1$where                    # extract group belonging
> a<-1;gr2<-rep(1,length(gr))       # renumber clusters sequentially
   for(i in 2:length(gr)) {
     if (gr[i]!=gr[i-1]) a <-a+1
     gr2[i] <- a
  }
> gr2
      1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4
```

This indicates that the first four individuals from the 'data' object belong to the same group, the next four to group 2 and so on. We can now plot otoliths from the different groups using different colours :

```
> plot(data,col="white",xaxt="n",yaxt="n",asp=1,bg="red")
> for (i in 1:nlevels(as.factor(gr2))) {
     c<-Coord[which(gr2==i),]
     cc<-matrix(c,ncol=2)
     points(cc[,1],cc[,2],asp=1,col="black",pch=21,bg=i)
  }
```

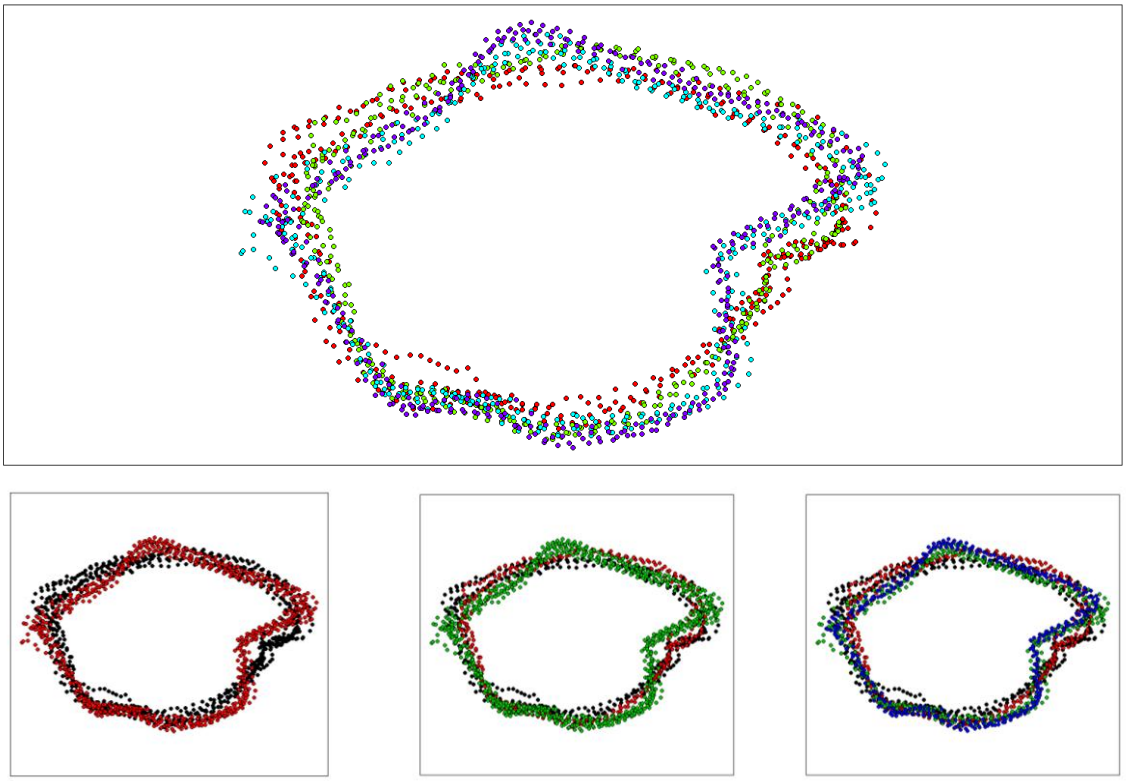


**Fig.10** Procruste superimposition of 16 otolith coordinates according to their group (upper panel). Associated plots for 1,2 and 3 splits respectively (lower panel).

By combining MRT computed in R and other morphometric software, we can easily obtain the following plot that explains the decomposition of morphological variation, as well as shape differences between the four obtained groups:

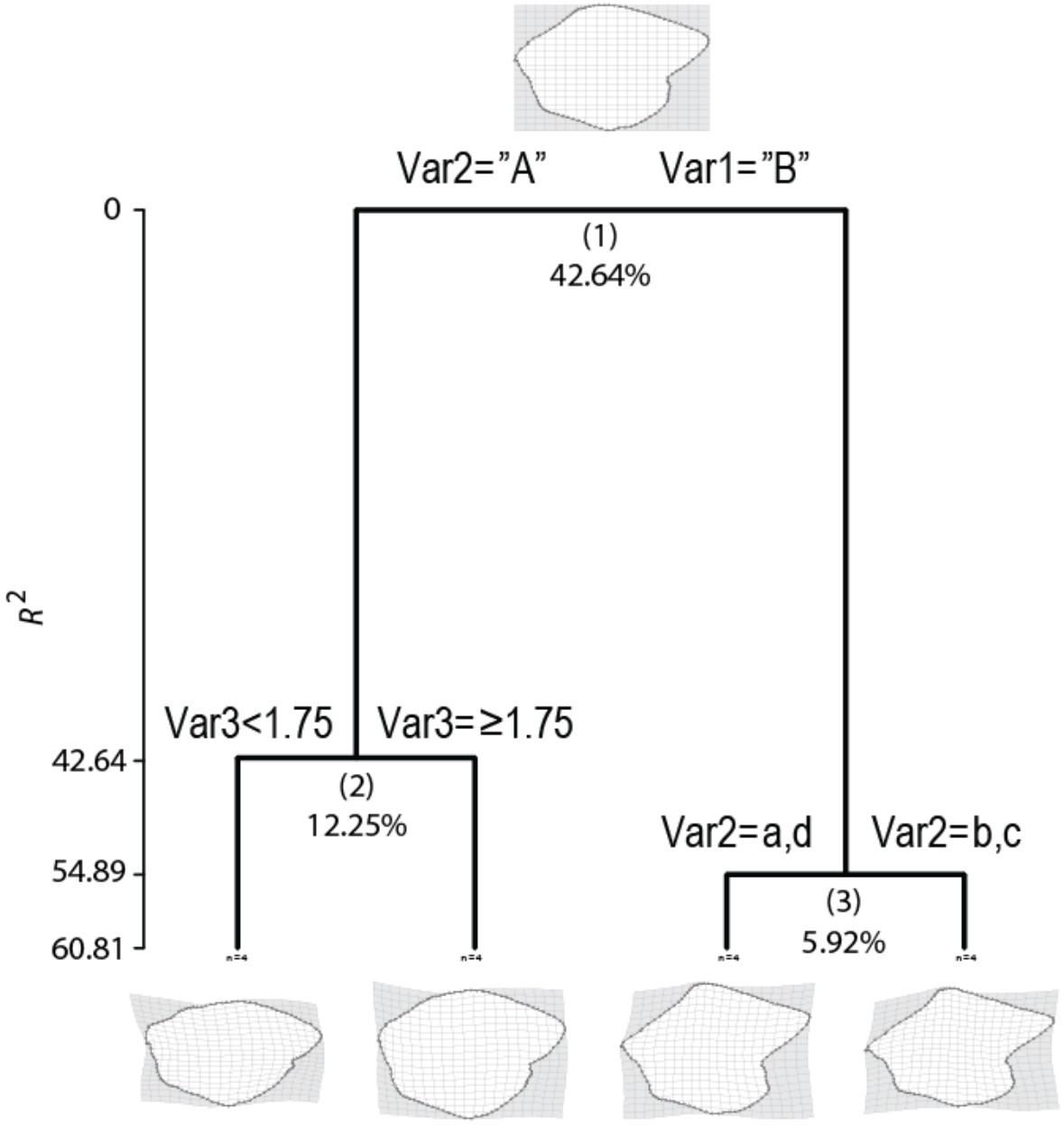



**Fig.11** Regression tree with percentage of explained variance at each split and total $R^2$ of the tree. Thin-plate spline deformation grids allow visualizing mean shape differences between the four groups, compared to the overall mean shape.

The MRT procedure ensure to repeatedly split observations into mutually exclusive subgroups, each of which being as homogeneous as possible. This tends to reduce the morphospace occupied by each subgroup in a significant manner. Equivalently, the procedure maximizes the morphological differences between the mean shapes of the two groups (i.e. maximize the between subgroups contribution). The morphospace partition can be visualized by using the following command. Please, note that PCA computed below is non-related to relative warps analysis (as it can be done using tpsRelw) and is therefore not a correct representation of morphospace occupancy. The plot is however sufficient to visualize the creation of homogeneous groups.

```
> PCA<-prcomp(Coord)
> PC12<-data.frame(X=PCA$x[,1],Y=PCA$x[,2])
> plot(PC12,xlab="PC1",ylab="PC2")
> PP(mrt1,PC12)
```

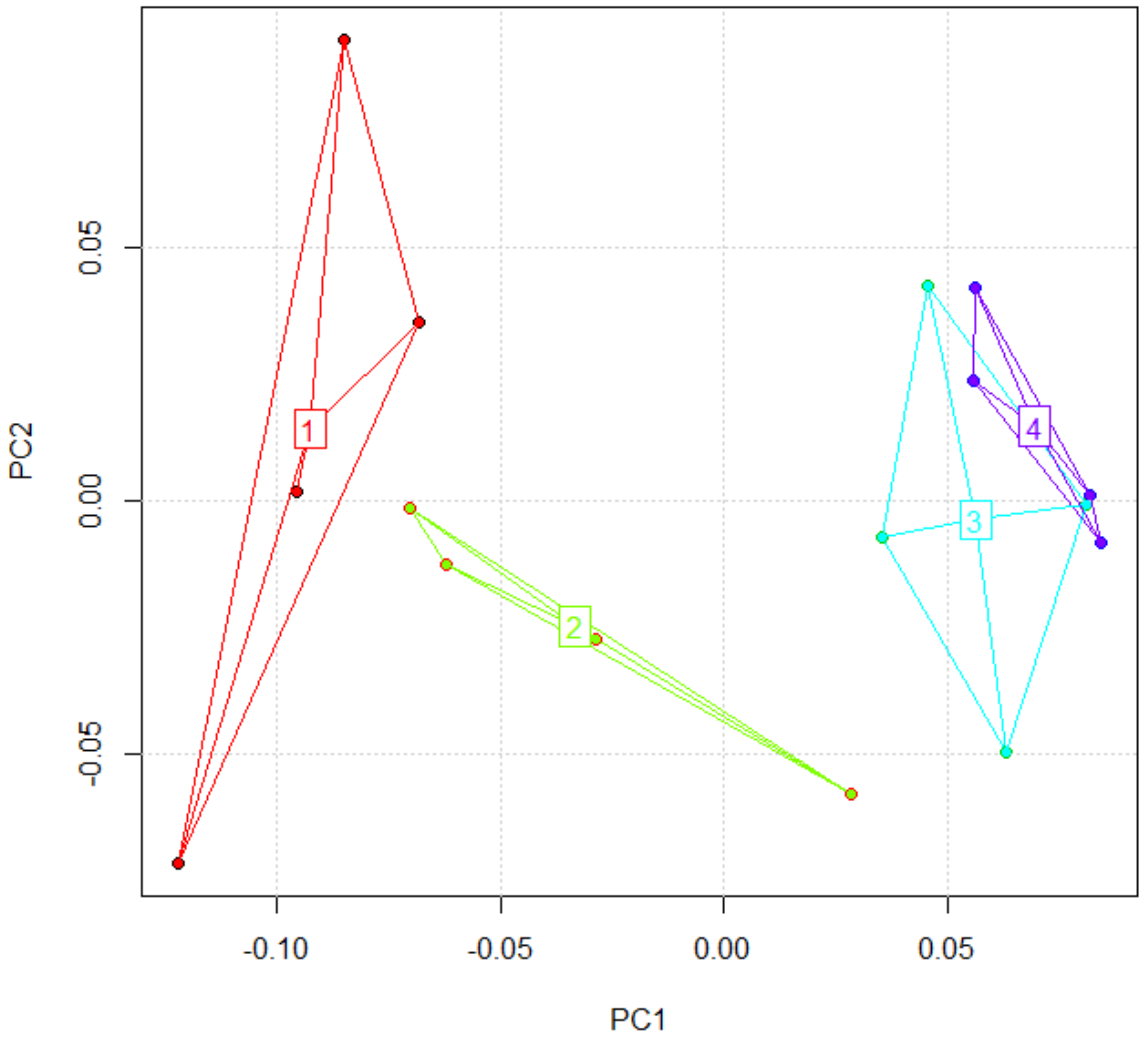



**Fig.12** First two PCs for the shape coordinates and centroid of each group from mrt1.

The first analysis used partitioned the data using an arbitrary tree size (i.e. number of terminal nodes (groups) to be obtained). In this context, the model is forced to obtain the requested size. Here, we can change the *size* parameter within the *mvpart* function to obtain only two groups:

```
> mrt2<-mvpart(data.matrix(Newdata[,1:200])~Var1+Var2+Var3,
  Newdata,size=2)
> PP(mrt2,PC12)
```

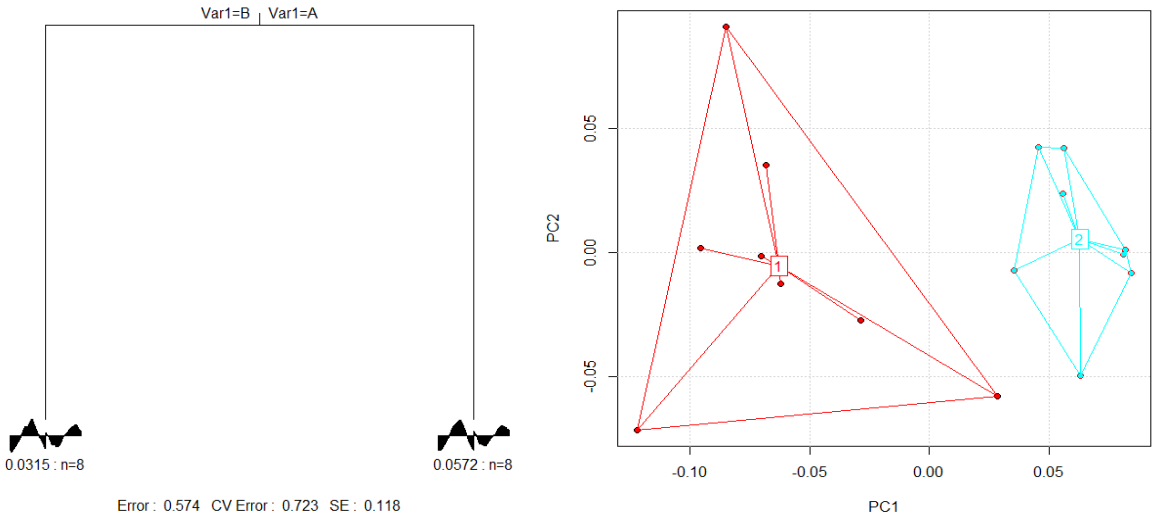



**Fig.13** Same data with other arbitrary size (one split, two groups).

However, there is commonly no *a priori* reasons to specify a given tree size and a major task in MRT is selecting the optimal tree size as the partition must be optimal in terms of model fitting (predictive error) and model simplicity (tree size). If a large tree size is used, the tree has very small groups corresponding to many homogeneous terminal nodes.. Such a tree would sample noise signal (over-fitting). On the other hand, using a small tree generates few groups exhibiting large morphospace occupacy. Such a tree would typically result in a poor fit. To deal with this optimization issue and preclude overfitting, tree algorithms typically calculate the fully grown tree (i.e. numerous terminal groups with very few measurements) and then prune the lower branches on the basis of 10-fold cross-validation error (the selected tree size minimizes the cross-validated error based on the '1se' rule*, Breiman et al. 1984, De'ath 2002). Please, note that the results of cross-validations arises from randomizations and are therefore not perfectly stable. The default is a '10-fold' cross-validation, meaning that only 10% of the data is left out of each training run (i.e. train on 90% datasets and test on 10%). This can be easily performed by omitting the `size` argument within the `mrt` function and by specifying the cross-validation method:

```
> mrt3<-mvpart(data.matrix(Newdata[,1:200])~Var1+Var2+Var3,
  Newdata, xv = "1se")
```

In this example, the first split (resulting of two groups) is selected alone because other consecutive splits will decrease error in a limited manner (remember Figure 6).

Mvpart can also deal with distance matrix, instead of row coordinates. The two approach give the same results as far as distances are provided as Procrustes distances in the tangent space. In this case, the square distance matrix must contain equal number of rows/columns that the number of rows in the variable object. Here is an example using the distance matrix associated with the same 16 otoliths:

```
> load("Dist1.Rda")                    # load "Distance" square matrix
> mrt4<-mvpart(Distance~Var1+Var2+Var3,Var,method="dist",size=4)
```

This results in the same tree that the one in Figure 5. However, please, note that due to different computations, the total explained variance may slightly differ between distance-based tree and coordinates-based tree (mrt4 explains 36% of total variance, while mrt1 explains 39% of total variance).

To conclude on this simple dataset, the following figure summaries the main splitting decomposition realized by recursive partitioning of the data.

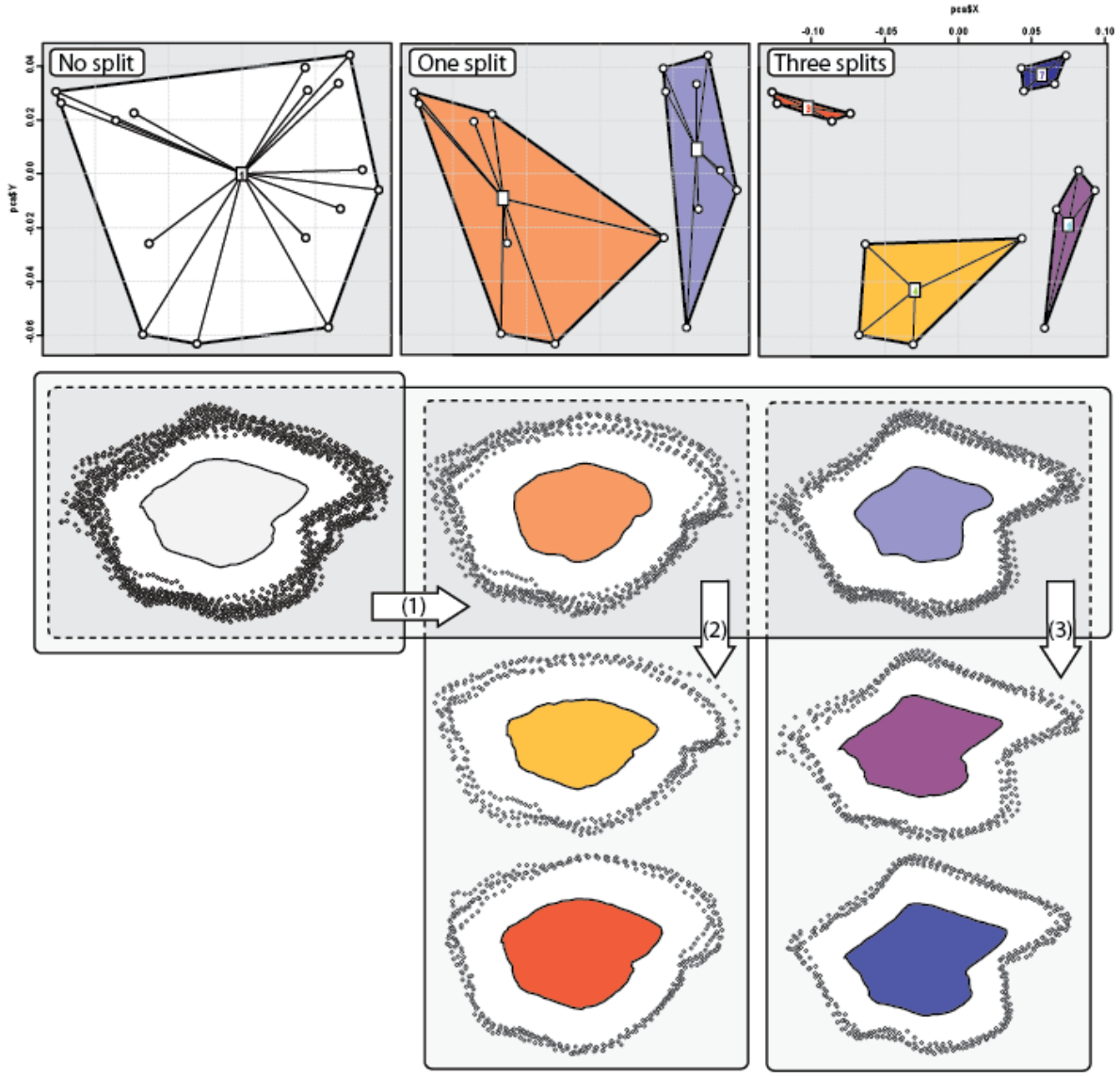
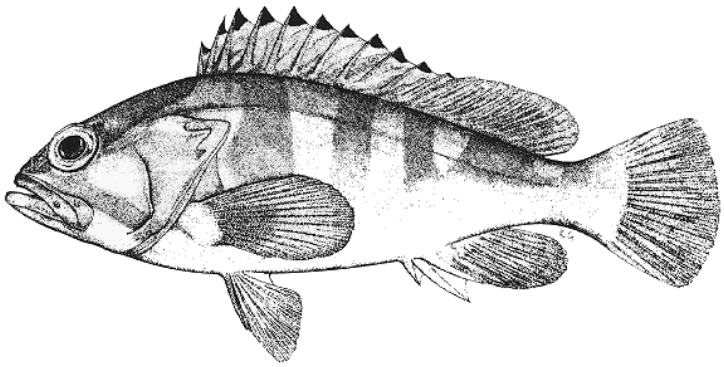



**Fig.14** Reduction of morphospace (partial warps scores) occupied by otoliths with 1,2 and 4 groups, (from left to right, top panels). Associated shapes depicted using coordinates superimposition in grey et group's mean shape in colour (bottom panels).

## IV. More complex and realistic data



The next example is a subset of the original data used in the publication. The dataset contains Procrustes distances between 380 otolithes from 190 *Cephalopholis fasciatus* (Serranidae) collected in Moorea Island (French Polynesia. The associated potential explicative variables are also provided in separate files.

We can first load the data using the following commands:

```
> load("Dist2.Rda")          # load "Distance" square matrix
> load("Var2.Rda")           # load "Var" dataframe
> str(Var)                   # summary variables
```

"Distance" is a 380*380 square matrix containing all pairwise distances. "Var" is a dataframe containing the standardized [0-1] fish length (*Size*), coastal distance (*CD*), depth (*Depth*), habitat (*Hab*), year (Year), residuals of fulton's index (*Fulton*), fish identifier (*ID*), island (*Island*), genus (*Genus*) and species (*Species*). Please, be sure that *Hab*, *Year* and *ID* are factors. The three last variables are non-informative in the present context and won't be used. All the remaining variables are potentially important to various extent and we can proceed immediately:

```
> mrt5<-mvpart(Distance ~ Size+CD+Depth+Hab+Year+Fulton, Var,
  xv="none",method="dist",cp=0.009)
```

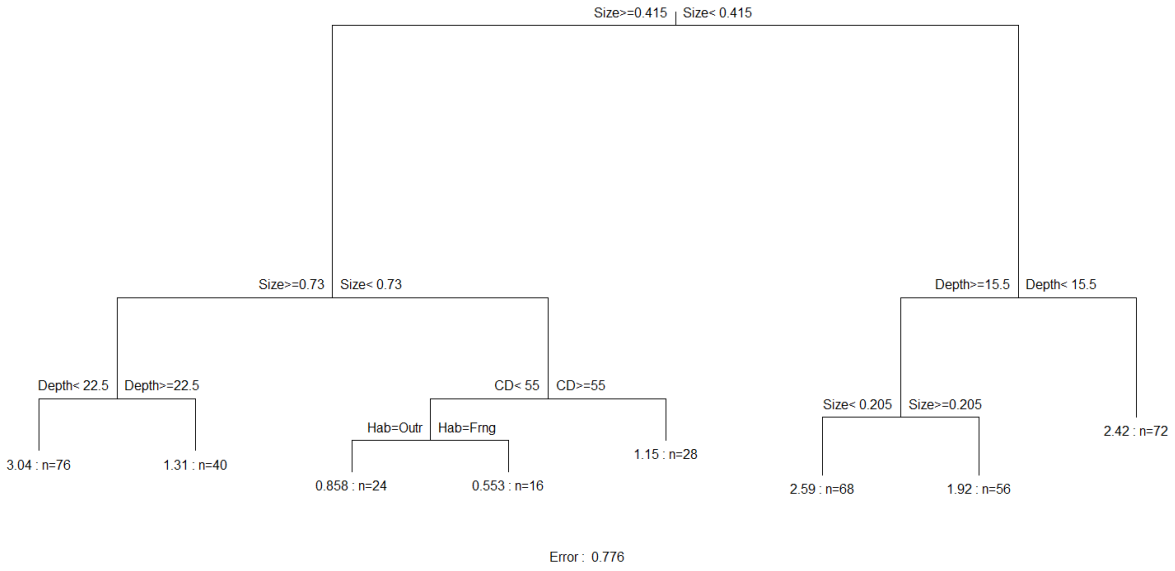



**Fig.15** Regression tree obtained for 380 otoliths. The tree contains 7 splits, involves 4 variables and explains 23% of total morphological variance.

The relative importance of each variable can then be obtained:

```
> importance<-VarImp(mrt5,pretty=F)
> importance$Relative
```

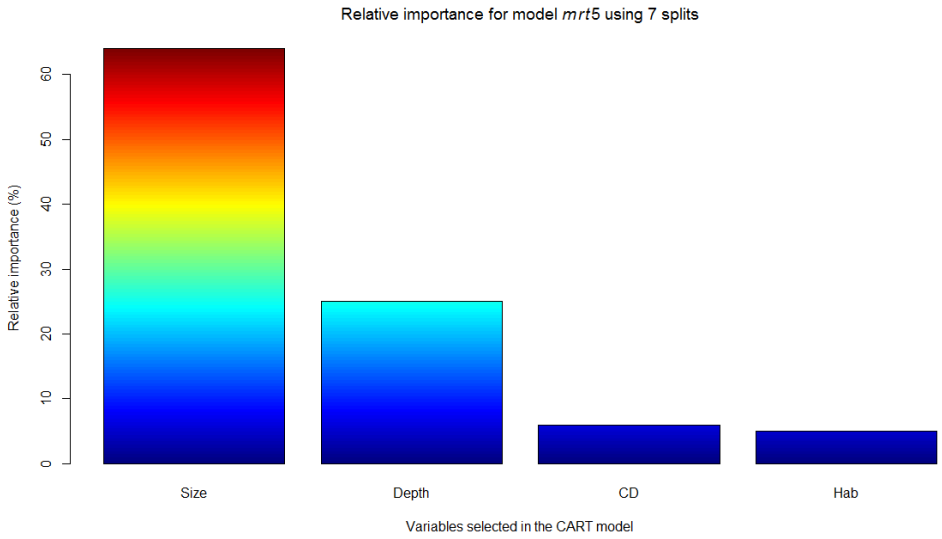



**Fig.16** Relative contribution of all variables from *mvpart5*.

The morphospace reduction can be visualized:

```
> fit <- cmdscale(Distance,eig=TRUE, k=2)
> XY<-as.data.frame(cbind(fit$points[,1],fit$points[,2]))
> colnames(XY)<-c("X","Y")
> PP(mrt5,XY)
```

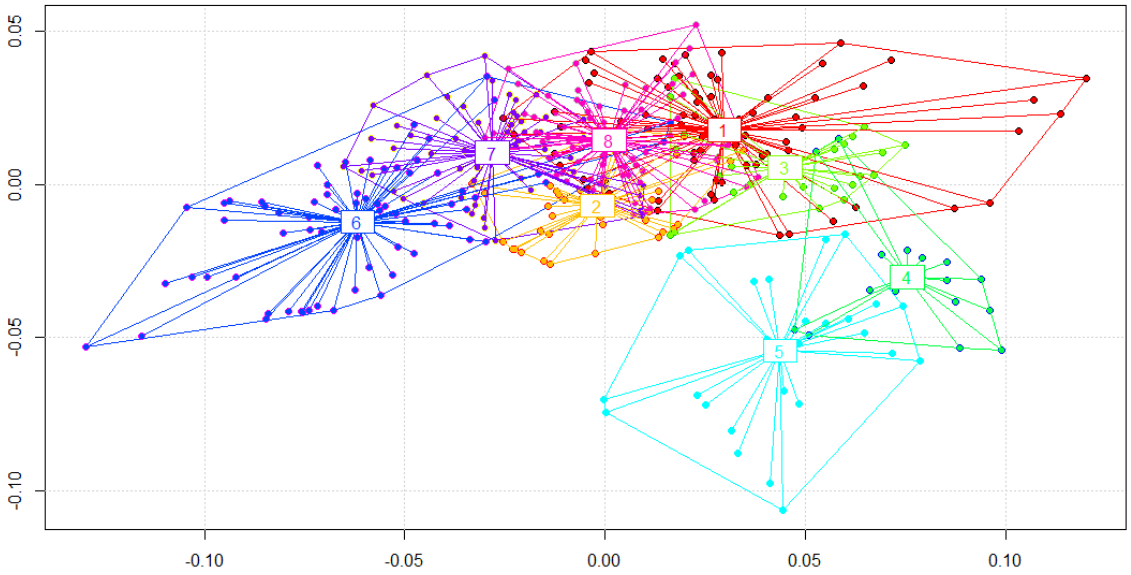



**Fig.17** MDS, centroid, convex hull and spider for each group from mrt5.

So far, the absolute and relative contribution of the main factor was estimated, except for the *Side* and *Individual* effects. The individual (*ID*) effect was not used directly in the partitioning process and its influence must therefore be interpreted as the percentage of unexplained inter-individual variance (residuals) when all other component have been quantified. Before quantifying the residual inter-individual effect, we need to estimate the side effect (between right and left otoliths). To do so, we will use the new *Asym* function that quantify the absolute degree of variance due to within-individual effect (i.e. asymmetry). The function works (internally using the vegan package) as far as Var$ID contains two rows for each individuals.

```
> Asy<-Asym(Distance,Var$ID)
```

Here, asymmetry accounts for 13.5% of total variance. The influence of all variables (including individuals) can now be calculated (all of them sums to 1):

```
> importance$Absolute                  # main influences
> Asy$Side                             # Side effect
> 1- sum(importance$Absolute[,2], Asy$Side)  # Individual effect
```

The variable selection at each node is intrinsically case-sensitive (instability to small perturbations of the data). To examine properly the main determinants of otolith shape and ensure that the models produced robust splits, the data were further resampled for model building using a given ("in-bag") fraction of the original data. Characteristics of a large ensemble of MRT models provide a more accurate characterization of morphological disparities (Prasad et al. 2006, De'ath 2007). Please, note that there is no consensual rule for choosing the bagging fraction, nor the number of bootstraps and the confidence interval should always be provided with variable importance. Just remember that using a high bagging fraction would generate neglecting instability in the data and that using a low bagging fraction would generate important instability that do not reflect the original data. The 0.5 parameters (Bagging=0.5) should therefore perform quite well in most cases (trade-off between instability and representativeness). In the publication, I used a bagging fraction of 0.5 (half of the original data are resampled, without replacement) and bootstrap procedure was applied 500 times. There is various ways to perform bootstrap on the original data. Here is a code example:

```
> Bagging<-0.5
> sampID<-sample(1:nlevels(Var$ID),size=Bagging*nlevels(Var$ID))
> samprow<-numeric()
> for (i in 1:length(sampID)){
     samprow<-c(samprow,which(as.numeric(Var$ID)==sampID[i]))}
> DistBoot<-Distance[samprow,samprow]
> VarBoot<-Var[samprow,]
> mrtBoot<-mvpart(DistBoot~Size+CD+Depth+Hab+Year+Fulton, VarBoot
  ,xv="none",method="dist",cp=0.009)
```

This results in a new tree whose explicative variables can be compared to full-dataset-based tree:

```
> par(mfrow=c(1,2))
> ImpBoot<-importance$Relative
> barplot(ImpBoot [,2],names.arg= ImpBoot [,1])
> VarImp(mrtBoot,pretty=F)
```

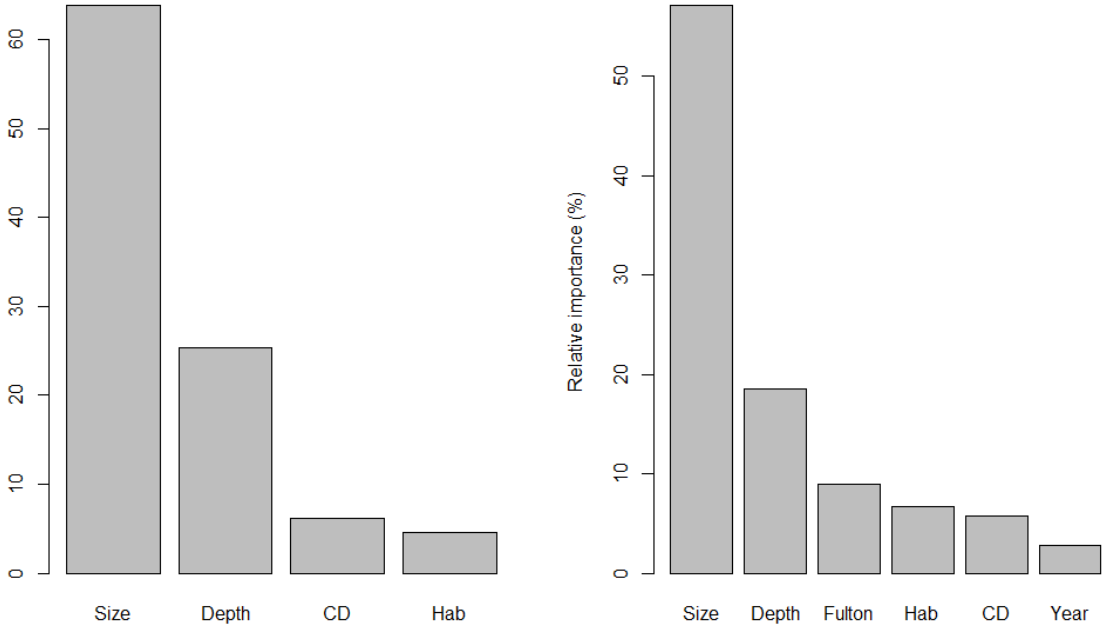



**Fig.18** Relative contribution of all variables from *mvpart5* and *mrtBoot*. Tree structure (and associated variable importance) is subject to high variability depending on data (i.e. model instability due to small changes in the data). Bagging (bootstrap aggregating) is a nonparametric machine-learning ensemble designed to improve the stability and accuracy of machine learning algorithms such as MRT. It does so by building a large number of regression trees based on a subset of randomly sampled (boostrap) otoliths. This typically renders the model output stochastic and variance is decreased through model averaging.

## Bibliography


Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and Regression Trees, Vol. Chapman & Hall, Belmont

De'ath G (2002) Multivariate regression trees: A new technique for modeling species-environment relationships. Ecology 83:1105-1117

De'ath G (2007) Boosted trees for ecological modeling and prediction. Ecology 88:243-251

De'ath G, Fabricius KE (2000) Classification and regression trees: A powerful yet simple technique for ecological data analysis. Ecology 81:3178–3192

Elith J, Leathwick JR, Hastie T (2008) A working guide to boosted regression trees. Journal of Animal Ecology 77:802-813

Ouellette M-H, Legendre P, Borcard D (2012) Cascade multivariate regression tree: a novel approach for modelling nested explanatory sets. Methods in Ecology and Evolution 3:234-244

Prasad AM, Iverson LR, Liaw LA (2006) Newer classification and regression tree techniques: Bagging and random forests for ecological prediction. Ecosystems 9:181-199

Rohlf FJ (2006) tpsDig, digitize landmarks and outlines. Department of Ecology and Evolution, State University of New York at Stony Brook


I hope that this tutorial has been informative. Please email any feedback to me (matthias.vignon@univ-pau.fr) or (mvignon@st-pee.inra.fr)