GINKGO user's manual

Version 1.4

Miquel De Cáceres Ainsa

Dept. Biologia Vegetal
Unitat de Botànica
Universitat de Barcelona

INDEX of CONTENTS

Chapter 1: Introduction

1.1 Program overview

1.1.1 What is GINKGO

GINKGO is a computer program that tries to make multivariate analysis methods easier to run for non-experts in statistics. It has been developed in the context of numerical ecology, but it can be used in other scientific disciplines (e.g. sociology, psychology). While it runs independently, it is the statistical module of VEGANA package, a working environment that provides several tools for editing and analysing flora and vegetation (De Cáceres et al. 2003).

The program has an easy-to-use graphical interface containing three main windows: a Data Editor, an Analysis Manager, and a Graphic Editor. The whole interface provides an intuitive integrated framework that allows users to explore, step by step, their multivariate data. First, scalar or vector transformations can be applied. Then, resemblance between sites or species is established by choosing between different similarity and dissimilarity coefficients. After that, data structure can be studied using ordination and classification methods, whose results are stored in the Analysis Manager. Clustering methods detect different cluster structures, so they may be used to query data structure from different points of view. The results obtained from those analyses can be evaluated and compared in the same Analysis Manager window. Finally, it is possible to re-copy result matrices back onto the data editor for subsequent analyses.

As a whole, the program makes it possible to do multivariate analyses of moderate complexity. This sort of ecological data mining can be done in several sessions because complete working projects are saved in compressed (.zip) files.

1.1.2 Technical information

The GINKGO program is written in Java programming language. Therefore, compiled code can be run under different operating systems (Windows, Linux, Mac, etc.). In fact, any platform supporting Java Runtime Environment version 1.4.0 or higher (http://www.java.com/) is suited to run the program. Like any of the VEGANA software modules, GINKGO is freely distributed and continuously improved. Installation instructions, program downloads and sample data sets are available at the VEGANA web site: http://biodiver.bio.ub.es/vegana.

Brief instructions on how to install and update the program would be as follows: First, the user should check that his computer has the Java Runtime Environment in a suitable version installed. If so, then the user should go to the VEGANA web page and click on the GINKGO icon. This will launch Java Web Start (JWS), download the program and run it. If desired, an icon of GINKGO can be placed on the desktop. Once the program is installed, subsequent updates are automatically done via JWS technology. Each time the user starts to run the program, if the computer is online, JWS checks whether there is a newer version and downloads it if there is.

The minimum estimated hardware requirements are a Pentium III processor and 256 MB of memory. Languages currently supported are English, Spanish and Catalan.

1.2 General features

1.2.1 Data matrix types

GINGKO can handle two types of data matrices: object-descriptor matrices and symmetric matrices.

Object Descriptor Data

Numerical data is often in the form of a set of n vectors in the feature space $\Re^p$ of p descriptors or variables. That is, the matrix to analyze is usually a rectangular (n x p) multivariate data matrix $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, where $\mathbf{x}_i \in \Re^p$ is a descriptor vector representing object ω_i (i = 1, ..., n), and $x_{ij} \in \Re$ (j = 1, ..., p) is the jth measured descriptor or variable of object ω_i. Object descriptor matrices are the most common form of original user data.

Symmetric Data

Square symmetric data matrices usually represent the association or resemblance, normally a similarity or dissimilarity, among objects. Three symmetric subcategories are available, used to label and distinguish symmetric matrices:

- UNKNOWN SYMMETRIC
- SIMILARITY MATRIX
- DISSIMILARITY MATRIX

Let $\mathbf{R} = [r_{hl}]$ denote a symmetric n x n relational or resemblance matrix, where r_{hl} measures the strength of the relationship between elements ω_h and ω_l. **R** may be either a similarity (**S**) or a dissimilarity (**D**) matrix (or a symmetric relation with unknown properties). A dissimilarity (or distance) n x n matrix $\mathbf{D} = [d_{hl}]$ satisfies the following three conditions:

$$d_{hh} = 0 \text{ for any } h = 1, ..., n; \quad (1.1a)$$

$$d_{hl} \geq 0 \text{ for any } h = 1, ..., n \text{ and } l = 1, ..., n; \text{ and} \quad (1.1b)$$

$$d_{hl} = d_{lh} \text{ for any } h = 1, ..., n \text{ and } l = 1, ..., n; \quad (1.1c)$$

Conversely, a similarity matrix $\mathbf{S} = [s_{hl}]$ fulfils:

$$s_{hl} \geq 0 \text{ for any } h = 1, ..., n \text{ and } l = 1, ..., n; \quad (1.2a)$$

$$s_{hl} = s_{lh} \text{ for any } h = 1, ..., n \text{ and } l = 1, ..., n; \text{ and} \quad (1.2b)$$

$$s_{hl} \leq s_{hh} \text{ for any } h = 1, ..., n \text{ and } l = 1, ..., n; \quad (1.2c)$$

1.2.2 Multivariate analysis types

GINKGO divides the provided multivariate methods into categories or analysis types. Each multivariate analysis type has a corresponding menu element in the GINKGO menu bar. Under each menu, those analyses related to the corresponding type are available as menu items:

Ordinations: These analyses pursue the representation of multivariate objects in a reduced dimension scatter diagram. Ordinations may be classed as canonical or non-canonical, depending on the number of input matrices. See chapter 3 for more information on ordination methods.

Classifications: These multivariate methods pursue the grouping of observations into classes. Classification can be done by *clustering* (using data solely) or *discriminant analysis* (using an initial classification for learning purposes) methods. See chapter 4 for more information on classification methods.

Comparisons: Allow the comparison of multivariate structures (either classifications or ordinations). See chapter 5 for more information on those methods.

Other analyses: This category includes results provided by GINKGO methods that cannot be included into the previous categories.

1.2.3 GINKGO main windows

GINKGO has three main windows. They can be opened and sent to the front by selecting the corresponding menu items in the **Window** menu of the GINKGO main menu bar or using keyboard shortcuts. Each window has its own menu bar, which allows actions related to it. The three windows are:

Data Matrix Editor (Alt-F1): Handles data matrices and enables editing them. Data matrices are arranged by type in a hierarchical tree. When you select one tree node, the associated data item is displayed on the right part of the window. The available menu options change depending on the tree node selected at the time. Data transformations, resemblance matrix building and other simple operations are done inside this window. Most multivariate analyses can only be launched when a suitable data matrix is selected in the Data Matrix Editor. Other analyses are enabled when the Data Matrix Editor includes enough matrices of a given type.

Analysis Manager (Alt-F2): This window stores the multivariate analysis running or already done. Analyses are arranged by analysis type in a hierarchical tree. When one tree node is selected, the associated analysis is displayed on the right part of the window. The available menu options change depending on the tree node selected at the time. Analyses which are still running are indicated in red. The user can stop their execution before finishing.

Graphic Editor (Alt-F3): This window is used as a sink for graphics obtained either from plotting original data or from plotting multivariate analysis' results. The Graphic Editor organizes its graphics in a tabbed pane. It makes it possible to change some plot features and to export plots as image files. The menu options available can change depending on the tab selected.

1.2.4 Working projects

GINKGO always runs using a working project. A working project includes both the source data matrices and the analysis results so far obtained from them. Projects can be saved as compressed zip files[1]. As a result, the analysis project can be continued in later sessions[2]. Graphics generated in GINKGO are not stored in the project but can be exported as image files.

All project actions (such as creating new projects or opening and saving projects) are handled using the **Project** menu in the GINKGO main menu bar. Project options can be changed by clicking on **Edit project options** menu item. The available project options are:

Project Home Directory: Directory where files are saved by default.

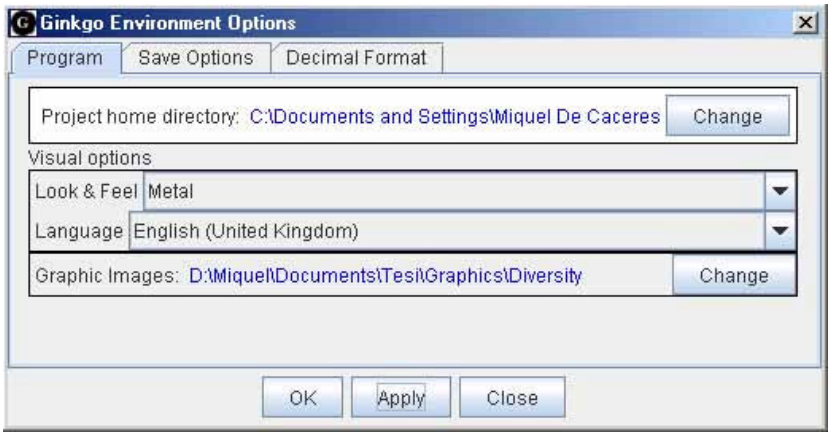
Language: Language used in menus.

Look And Feel: Platform look of the program (Windows, UNIX, Metal...)

Graphic Images Directory: Directory where graphic image files are exported by default.

Data Save Options: Here the user can select which data types and analysis results will be saved. Each data or analysis type has a different check-box.

Decimal Format Options: Here users can specify which decimal format has to be used, by default, to display values in each matrix type.



[1] XML format is also available. However, **ZIP** file format is highly recommended due to its data compression.
[2] The only analyses which are not stored in working project files are those pending in the 'Other analyses' node in Analysis Manager.

Chapter 2: Data management and operations

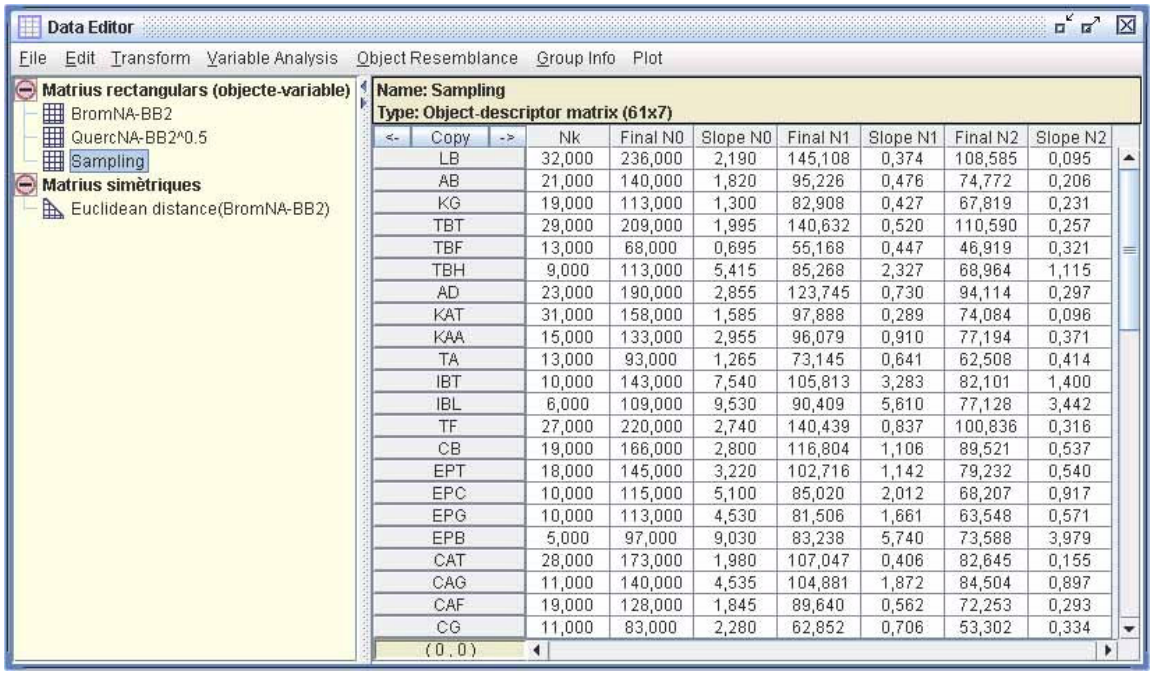
2.1 Data Editor Window

2.1.1 Introduction

Data Editor is the GINKGO window containing the data to be edited and analyzed. Nearly all analyses start from a data matrix in the Data Editor. All data matrices are listed in the Editor in the form of a tree on the left part of the module. Two main nodes are present on the tree, each one corresponding to one data type. Project data items are organized as tree nodes hanging from one of these. When a data item is selected in the tree, the corresponding data matrix is displayed on the right as an editable table. The Data Editor can be shown when not visible by pressing **Alt-F1**.

2.1.2 Data Matrix name and type

The Data Editor enables editing numerical data matrices, either object descriptor or symmetric ones. Each data item has a **name** and a **type**. The item name identifies the data matrix and the type restricts the operations that can be performed on the data. The two data types are distinguished in the Data Editor by grouping data items into two corresponding tree nodes. The name and type of a data item are displayed on a panel on top of the data table, along with the matrix's dimensions ("number of rows" x "number of columns").



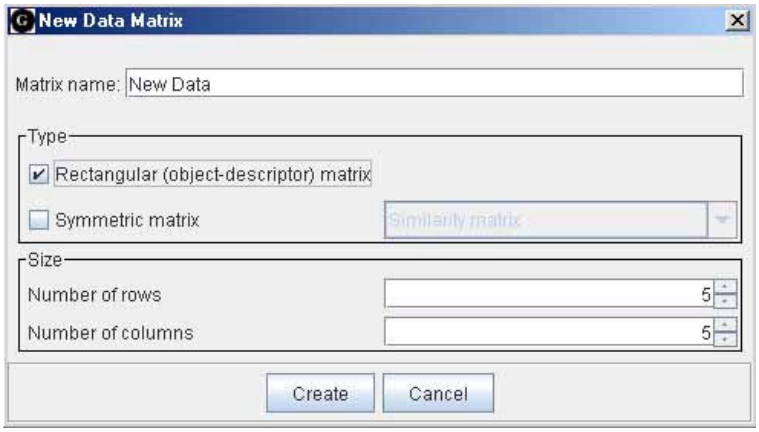
<- Copy ->	Nk	Final N0	Slope N0	Final N1	Slope N1	Final N2	Slope N2
LB	32,000	236,000	2,190	145,108	0,374	108,585	0,095
AB	21,000	140,000	1,820	95,226	0,476	74,772	0,206
KG	19,000	113,000	1,300	82,908	0,427	67,819	0,231
TBT	29,000	209,000	1,995	140,632	0,520	110,590	0,257
TBF	13,000	68,000	0,695	55,168	0,447	46,919	0,321
TBH	9,000	113,000	5,415	85,268	2,327	68,964	1,115
AD	23,000	190,000	2,855	123,745	0,730	94,114	0,297
KAT	31,000	158,000	1,585	97,888	0,289	74,084	0,096
KAA	15,000	133,000	2,955	96,079	0,910	77,194	0,371
TA	13,000	93,000	1,265	73,145	0,641	62,508	0,414
IBT	10,000	143,000	7,540	105,813	3,283	82,101	1,400
IBL	6,000	109,000	9,530	90,409	5,610	77,128	3,442
TF	27,000	220,000	2,740	140,439	0,837	100,836	0,316
CB	19,000	166,000	2,800	116,804	1,106	89,521	0,537
EPT	18,000	145,000	3,220	102,716	1,142	79,232	0,540
EPC	10,000	115,000	5,100	85,020	2,012	68,207	0,917
EPG	10,000	113,000	4,530	81,506	1,661	63,548	0,571
EPB	5,000	97,000	9,030	83,238	5,740	73,588	3,979
CAT	28,000	173,000	1,980	107,047	0,406	82,645	0,155
CAG	11,000	140,000	4,535	104,881	1,872	84,504	0,897
CAF	19,000	128,000	1,845	89,640	0,562	72,253	0,293
CG	11,000	83,000	2,280	62,852	0,706	53,302	0,334
(0,0)							

The data item's name and type can be changed by using the following options in the file menu. Alternatively, the matrix name can be edited by double-clicking on the corresponding tree node.

Command	Menu Item (File Menu)	Keystroke
Changes data type (only for symmetric matrices)	Change matrix type	**Ctrl-T**
Changes data item name	Change matrix name	**Ctrl-N**

2.1.3 Creating new data matrices

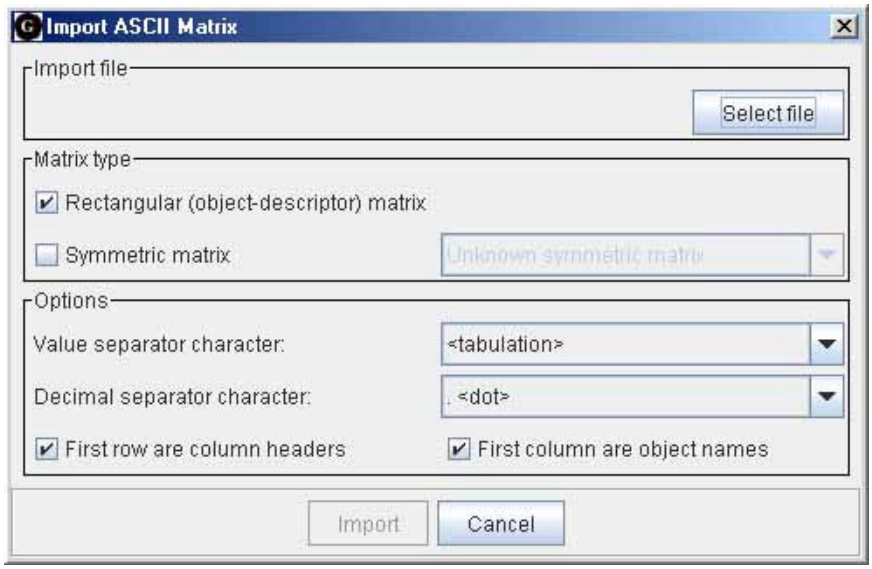
The simplest way to create a new empty data item in the editor is by selecting **Create new matrix** in the **File** menu. The dialog that appears enables the user to select the matrix type, name and matrix dimensions of the new data item. Once created, the new data item will be shown in the Editor's tree as a new tree node as a child of the corresponding data type node.



2.1.4 Importing/exporting data matrices

Importing

Data items are typically imported to GINKGO from an ASCII file. User can choose among several importing options: the value separation character, the decimal separation character, and whether column and/or row names are included in the ASCII input file.



ASCII object-descriptor matrix reading is as follows: The number of objects (n) is equal to the number of non-empty lines (excluding the first line if it is composed of column headers). For each line, a number of elements are identified using the indicated separator character. Then, the number of descriptors (p) is set to the maximum of row (line) elements found (minus one if the first column consists of object names). If column headers are included in the first line, the last p elements are used as descriptor names. ASCII symmetric matrices are read similarly.

Additional importing facilities are provided for users of CANOCO (ter Braak 1988) or other programs dealing with its file format, since it is possible to import CANOCO species data or environmental data files into the GINKGO Data Editor.

Exporting

The same ASCII and CANOCO formats are available for exporting data from GINKGO to other software tools. An alternative way to export tables from Ginkgo consists of simply selecting the desired table rows and pressing **Ctrl-C** to copy them to the system clipboard. The whole matrix can be copied by pressing the '**Copy**' button on the item panel. Other software, such as Microsoft Excel, can then read the table copied onto the clipboard.

Importing phytosociological relevé tables

QUERCUS is another computer program belonging to the VEGANA package. Relevé tables in XML QUERCUS format can be imported to GINKGO. The procedure is the same as the one that enables exporting from QUERCUS to GINKGO. The reader is referred to the corresponding section in the QUERCUS user's manual. Import/Export commands are summarized in the following table:

Command	Menu Item (File Menu)
Imports a text file	Import data... ASCII Text
Imports a CANOCO species data file	Import data... CANOCO/DECORANA (condensed) species data
Imports a CANOCO environmental data file	Import data... CANOCO environmental data
Exports a spaced text file	Export data... ASCII Text
Exports a CANOCO species data file	Export data... CANOCO/DECORANA (condensed) species data
Exports a CANOCO environmental data file	Export data... CANOCO environmental data
Imports a QUERCUS Relevé Table	Import XML relevé table

2.1.5 Deleting data items

Data items can be deleted from the project by simply pressing **Ctrl-Delete** when the data item in question is selected on the Editor's tree. The corresponding menu option in the **File** menu is called **Remove data matrix from project**.

2.1.6 Moving data items between GINKGO windows

Sometimes the user may be interested in moving classification matrices from the Data Editor to the Analysis Manager, or alternatively, in putting result matrices back into the Data Editor in order to perform subsequent analysis. The following table indicates where the menu items that launch these actions are:

Action	Menu Item	Menu	Window	Keystroke
Copies a data matrix onto the clipboard, in GINKGO format (not used for export).	Copy all matrix to clipboard	File	Data Editor	**Ctrl-A**
Pastes a data matrix from the clipboard into Data Editor (not used for import).	Paste matrix from clipboard to editor	File	Data Editor	**Ctrl-N**
Places a copy of the selected matrix in the Analysis Manager in the Data Editor	Copy selected matrix to Data Editor	Edit	Analysis Manager	**Ctrl-Shift-C**
Pastes matrix from clipboard in GINKGO format into the Analysis Manager	Paste clipboard matrix as new partition	Edit	Analysis Manager	**Ctrl-V**

2.1.7 Editing matrices

Value edition and visualization

Data matrix cell values can be edited manually as in a spreadsheet. When editing a cell (i,j) in a symmetric matrix the complementary cell (j,i) will be also be altered. Column width and decimal format is controlled using the following menu items of the menu **Edit**.

Action	Menu Item (Edit Menu)
Decimal precision is initially set using project defaults. It can be changed using this menu item. This menu item is also available when right-clicking on the table.	Change value format
Column width is 60 pixels by default. It can be changed using this menu item. In addition, the user can set this property by right-clicking on the table when one column is selected or by placing mouse arrow on the column header border and dragging it.	Change column width
Width of object label column can be changed using this menu item. It is also available when right-clicking on the object label column.	Change label width

Row and column manipulation

The following table lists all menu items of the menu **Edit** related to row and column manipulation:

Action	Menu Item (Edit Menu)
Adds new rows to a rectangular matrix.	Add New... rows
Adds new columns to a rectangular matrix.	Add New... columns
Adds new elements to a symmetric matrix.	Add New... elements
Filters empty rows in a rectangular matrix.	Filter... null rows
Filters empty columns in a rectangular matrix.	Filter... null columns
Filters species with low frequency in the matrix	Filter... low constancy species
Deletes rows in a rectangular matrix.	Remove... rows
Deletes columns in a rectangular matrix.	Remove... columns
Deletes elements in a symmetric matrix.	Remove... elements

All these actions can also be done by selecting a row or column in the table and right-clicking. Row/columns can be moved by selecting them and right-clicking on the new position desired (this is similar to a cut-paste action). Row/column labels (names) are changed through double-clicking on the desired row or column. Table rows may be reordered following the selected column values (increasing) or following row label alphabetical ordering.

All-table actions

Other **Edit** menu actions affect the data matrix as a whole:

Action	Menu Item
Clones a data matrix and creates a new item. A dialog is prompted to set the name of the new item.	Clone Matrix
Transposes a rectangular matrix. That is, changing rows by columns and vice versa. This means treating objects as descriptors and descriptors as objects.	Transpose Matrix
Crops a rectangular matrix. A dialog is prompted to select which elements are to be excluded.	Crop Matrix

Edit actions concerning groups

It is possible to generate a partition (a classification of objects to be put into the Analysis Manager) from a specific column of an object-descriptor matrix. This is done by selecting the **Create partition from column** menu item. The user has to choose which variable (column) will be interpreted as a group labeling column (by default the last column of the table). Each different value will be translated as a distinct group in the generated partition. As GINKGO 1.4 only accepts numerical values, the user should normally employ different integer values to identify groups in a column.

On the other hand, if object partitions are available on the Analysis Manager it is possible to split a given matrix into group submatrices (using **Groups**... **Extract data groups following a partition**) or to remove objects (rows) belonging to a certain group (using **Groups**... **Remove data groups following a partition**).

2.2 Operations on object-descriptor data

2.2.1 Transforming object-descriptor matrices

Data transformations are executed from the **Transform** menu in the Data Editor. In most of these transformations a dialog is prompted to enable the user to select which variables (descriptors) will be affected by the transformation and which will not. Also a check box is available in the dialog to enable replacing the current data matrix by the variable transformed matrix. Otherwise a new data matrix will be posted on the project. The operations included in the Transform menu are described below. Note that all transformations, excepting '**Transform values**', can only be applied to object descriptor matrices.

Action	Menu Item (Transform Menu)	Keystroke
Applies a simple numerical transformation (square root, logarithm, exponential...) to cell data values.	Transform values	Ctrl-F1
Standardizes variables in a rectangular matrix. The actual operation can be modified by selecting the value to subtract and the value to divide. A constant number can also be added.	Standardize variables	Ctrl-F2
Normalizes variables in a rectangular matrix. However, it skips zeros for both mean and variance computations and for the transformation.	Non-zero normalization	
For every row, subtracts the group average to each value weighted by the object membership to the group.	Center data by groups	
Modifies values in a rectangular data matrix so as to emulate a given distance.	Distance emulating transforms	
Replaces non-zero values in a rectangular data matrix by 1.0 so as the matrix becomes a binary (Presence/Absence) matrix.	Convert to P/A matrix	

Distance emulating transforms (Legendre & Gallagher 2001) create a new matrix of **Y** values from a rectangular matrix **X**, so that when computing the Euclidean distance on the transformed data (**Y**), the distance matrix obtained will be equal to the matrix obtained by computing the emulated distance on the original data (**X**). The available distance emulating transforms are:

Distance emulated	Transform formula	Description
Chord distance	$y_{ij} = x_{ij} \Big/ \sqrt{\sum_{l=1}^{p} x_{il}^2}$	Division by vector length. Rows normalized to unit length
Species profiles distance	$y_{ij} = x_{ij} \Big/ \sum_{l=1}^{p} x_{il}$	Division by vector sum. Species are row proportions.
Hellinger distance	$y_{ij} = \sqrt{x_{ij} \Big/ \sum_{l=1}^{p} x_{il}}$	Square root of the species proportion.
χ^2 distance	$y_{ij} = \sqrt{x_{++}} \cdot \frac{x_{ij}}{x_{i+} \cdot \sqrt{x_{+j}}}$	Double transform by row and column (see equations 2.1f-h for notation).

2.2.2 Analysis of descriptors

Some initial description of the variable structure is executed from the **Variable Analysis** menu in the Data Editor menu bar. The available menu items are described in the following:

Descriptive statistics **(Ctrl+F3)**

Calculates the commonest descriptive statistics for the selected variables (descriptors) of the current rectangular data matrix: *Average*, *Median*, *Variance*, *Standard Deviation*, *Variation Coefficient*, *Minimum*, *Maximum*, *Range*, *Skew* and *Kurtosis*. The results are placed at the '**Other analysis**' tree node of the Analysis Manager tree.

Covariation matrices **(Ctrl+F4)**

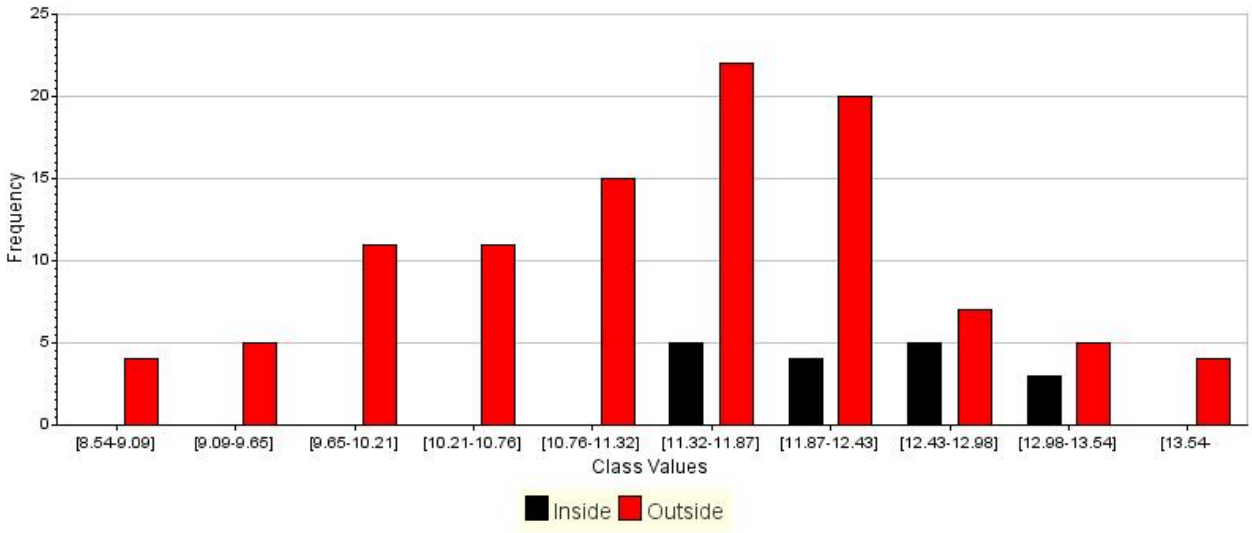
Calculates covariation matrices for the selected variables (descriptors) of the current rectangular data matrix. The following matrices can be obtained: *Sum of squares matrix*, *Covariance matrix*, *Pearson correlation matrix*, *Spearman correlation matrix*. If checkbox *correlation p-values* is selected, additional matrices of type-I error probabilities (p-values) of correlation signification will be produced.

Frequency histogram

Creates a bar plot of value frequencies (histogram) from a selected variable of the current object-descriptor matrix. User must indicate the variable to plot and the number of value classes to be made. The resulting bar plot is placed on the Data Editor.

Frequency histogram in cluster

As in the previous item, creates a bar plot of value frequencies from a selected variable of the current object-descriptor matrix. However, values of objects belonging to a given cluster are distinguished from values of objects not belonging to it. The user must indicate the variable to plot, the cluster to be used as reference and the number of value classes to be made. The resulting bar plot combines two histograms, one for the objects inside the group and the other for the external objects.



2.3 Symmetric matrices

2.3.1 Building Similarity/Dissimilarity matrices.

Symmetric matrices are common in most multivariate analyses, since they express relations between objects or descriptors. Some multivariate analyses (classifications or ordinations) can or must be run with a symmetric data matrix as input matrix. For instance, Metric Multidimensional Scaling (MDS) needs a dissimilarity (generally a distance) matrix to be executed. On the other hand Agglomerative Hierarchical Clustering algorithms can be applied to both similarity and dissimilarity matrices.

Remember that symmetric matrices can be imported as ASCII files. Alternatively, to construct such symmetric matrices or to transform similarities onto dissimilarities the **Resemblance** menu of the Data Editor must be used. The most important menu items are:

1. **Build similarity matrix (*Ctrl-F5*)**: Builds a similarity matrix and places it as a new data item in the Data Editor tree.
2. **Build dissimilarity matrix (*Ctrl-F6*)**: Builds a dissimilarity (distance) matrix and places it as a new data item in the Data Editor tree.

Many dissimilarity and similarity measures for ecological data are already available in GINKGO. Their formulae and short descriptions are given in the tables on the next two pages, for dissimilarity and similarity measures respectively. The following notation has to be taken into account:

$$a = I(x_{1j} > 0 \,\&\, x_{2j} > 0) \tag{2.1a}$$

$$b = I(x_{1j} > 0 \,\&\, x_{2j} = 0) \tag{2.1b}$$

$$c = I(x_{1j} = 0 \,\&\, x_{2j} > 0) \tag{2.1c}$$

$$d = I(x_{1j} = 0 \,\&\, x_{2j} = 0) \tag{2.1d}$$

$$x_{1+} = \sum_{j=1}^{p} x_{1j} \tag{2.1e}$$

$$x_{2+} = \sum_{j=1}^{p} x_{2j} \tag{2.1f}$$

$$x_{+j} = \sum_{i=1}^{n} x_{ij} \tag{2.1g}$$

$$x_{++} = \sum_{i=1}^{n} \sum_{j=1}^{p} x_{ij} \tag{2.1h}$$

$$A = \sum_{j=1}^{p} x_{1j} \cdot I(x_{2j} = 0) \tag{2.1i}$$

$$B = \sum_{j=1}^{p} x_{2j} \cdot I(x_{1j} = 0) \tag{2.1j}$$

$$C = \sum_{j=1}^{p} (x_{1j} + x_{2j}) \cdot I(x_{1j} > 0 \,\&\, x_{2j} > 0) \tag{2.1k}$$

$$W = \sum_{j=1}^{p} \min(x_{1j}, x_{2j}) \tag{2.1l}$$

Apart from those of the tables, two additional indices are provided by GINKGO (ver. 1.4): Gower and Goodall probabilistic similarity coefficients. See Legendre & Legendre (1998) for a detailed explanation of those and other resemblance measures for ecological data.

Distance	Formula	Short description
Euclidean Distance	$d_{ED}(\mathbf{x}_1,\mathbf{x}_2)=\sqrt{\sum_{j=1}^{p}(x_{1j}-x_{2j})^2}$	Pythagorean Euclidean Distance for quantitative data.
Squared Euclidean Distance	$d_{SQED}(\mathbf{x}_1,\mathbf{x}_2)=\sum_{j=1}^{p}(x_{1j}-x_{2j})^2$	Squared Pythagorean Euclidean Distance for quantitative data.
Binary Euclidean Distance	$d_{BINED}(\mathbf{x}_1,\mathbf{x}_2)=\sqrt{\sum_{j=1}^{p}(I(x_{1j}>0)-I(x_{2j}>0))^2}$	Pythagorean Euclidean Distance for binary data. It behaves like a distance complement for Simple Matching Coefficient.
Binary Squared Euclidean Distance	$d_{BINSQED}(\mathbf{x}_1,\mathbf{x}_2)=\sum_{j=1}^{p}(I(x_{1j}>0)-I(x_{2j}>0))^2$	Squared Pythagorean Euclidean Distance for binary data.
Manhattan Metric	$d_{Man}(\mathbf{x}_1,\mathbf{x}_2)=\sum_{j=1}^{p}\lvert x_{1j}-x_{2j}\rvert$	Manhattan or city block metric
Absolute Value Distance	$d_{ABS}(\mathbf{x}_1,\mathbf{x}_2)=\sqrt{\sum_{j=1}^{p}\lvert x_{1j}-x_{2j}\rvert}$	Square root of Manhattan metric
Bray-Curtis Distance	$d_{BC}(\mathbf{x}_1,\mathbf{x}_2)=\frac{\sum_{j=1}^{p}\lvert x_{1j}-x_{2j}\rvert}{\sum_{j=1}^{p}(x_{1j}+x_{2j})}=\frac{\sum_{j=1}^{p}\lvert x_{1j}-x_{2j}\rvert}{x_{1+}+x_{2+}}$	Bray Curtis ecological distance. It behaves as a distance complement of Motyka similarity for quantitative data and Sörensen index for binary data.
χ^2 Distance	$d_{\chi^2}(\mathbf{x}_1,\mathbf{x}_2)=\sqrt{\sum_{j=1}^{p}\frac{1}{x_{+j}/x_{++}}\left(\frac{x_{1j}}{x_{1+}}-\frac{x_{2j}}{x_{2+}}\right)^2}$	Chi-square distance as the one preserved in Correspondence Analysis. Uses table margins information. It has a linear relation to the Chi-square metric.
χ^2 Metric	$d_{m\chi^2}(\mathbf{x}_1,\mathbf{x}_2)=\sqrt{\sum_{j=1}^{p}\frac{1}{x_{+j}}\left(\frac{x_{1j}}{x_{1+}}-\frac{x_{2j}}{x_{2+}}\right)^2}$	Chi-square metric. Uses table margins information. It has a linear relation to the Chi-square distance.
Mahalanobis Distance	$d_{Mah}(\mathbf{x}_1,\mathbf{x}_2)=\mathbf{d_{12}}\mathbf{S}^{-1}\mathbf{d_{12}}^{'}$ $\mathbf{d_{12}}=(x_{11}-x_{21},\ldots,x_{1P}-x_{2P})$, i $\mathbf{S}^{-1}$ matriu var-cov.	Mahalanobis (one group) distance. Like Euclidean distance, but it takes into account variable correlations and "extracts" them.
Canberra Metric	$d_{Canb}(\mathbf{x}_1,\mathbf{x}_2)=\sum_{j=1}^{p}\frac{\lvert x_{1j}-x_{2j}\rvert}{(x_{1j}+x_{2j})}$	Canberra metric
Canberra Metric (Adkins form)	$d_{Canb-Ad}(\mathbf{x}_1,\mathbf{x}_2)=\frac{1}{(a+b+c)}\cdot\sum_{j=1}^{p}\frac{\lvert x_{1j}-x_{2j}\rvert}{(x_{1j}+x_{2j})}$	Canberra metric normalized by the number of species.
Chord Distance	$d_{Chord}(\mathbf{x}_1,\mathbf{x}_2)=\sqrt{\sum_{j=1}^{p}\left(x_{1j}\Big/\sqrt{\sum_{l=1}^{p}x_{1l}^2}-x_{2j}\Big/\sqrt{\sum_{l=1}^{p}x_{2l}^2}\right)^2}$	Chord distance proposed by Orloci (1967). This is the Euclidean distance computed after dividing the value by the norm or length of the vector.
Hellinger Distance	$d_{Hell}(\mathbf{x}_1,\mathbf{x}_2)=\sqrt{\sum_{j=1}^{p}\left[\sqrt{\frac{x_{1j}}{x_{1+}}}-\sqrt{\frac{x_{2j}}{x_{2+}}}\right]^2}$	The Hellinger distance, described by Rao (1995)
Species Profiles Distance	$d_{Sp\,\mathrm{Pr}of}(\mathbf{x}_1,\mathbf{x}_2)=\sqrt{\sum_{j=1}^{p}\left[\frac{x_{1j}}{x_{1+}}-\frac{x_{2j}}{x_{2+}}\right]^2}$	Pythagorean Euclidean distance computed after transforming data dividing by row sum.
Pearson correlation complement	$d_{CompPears}(\mathbf{x}_1,\mathbf{x}_2)=1-r_{Pearson}(\mathbf{x}_1,\mathbf{x}_2)$	One complement of Pearson correlation computed between rows.

Similarity	Formula	Short description
Jaccard (Ellenberg)	$s_{Jac-Ellen}(\mathbf{x}_1,\mathbf{x}_2)=\frac{C/2}{(A+B+C/2)}$	Jaccard similarity index when applied to binary data. Ellenberg similarity index when applied to quantitative data.
Sörensen (Steinhaus)	$s_{Sor-Stein}(\mathbf{x}_1,\mathbf{x}_2)=\frac{2W}{(x_{1+}+x_{2+})}$	Sörensen similarity index when applied to binary data. Steinhaus similarity index when applied to quantitative data.
Gleason index	$s_{Gleason}(\mathbf{x}_1,\mathbf{x}_2)=\frac{C}{(A+B+C)}$	Gleason similarity index. It is very similar but gives double weight to common species. For PA data it behaves like the Sorensen index. However with quantitative data it is different from Motyka index, because it does not double the minimum of values. Instead, it sums both.
Simple Matching Coefficient	$s_{SMC}(\mathbf{x}_1,\mathbf{x}_2)=\frac{a}{a+b+c+d}$	SMC is a binary coefficient also known as the Sokal&Michener index.
Spätz index	$s_{Spath}(\mathbf{x}_1,\mathbf{x}_2)=\frac{1}{(a+b+c)}\cdot\sum_{j=1}^{p}\frac{\min(x_{1j},x_{2j})}{\max(x_{1j},x_{2j})}\cdot s_{Gleason}(\mathbf{x}_1,\mathbf{x}_2)$	Spätz similarity index. This is an alternative quantification of the Jaccard index. Its formula has two components; the left part is an expression of the relative similarity of stands, while the right part is the Gleason index.
Kulczynski index	$s_{Kulc}(\mathbf{x}_1,\mathbf{x}_2)=\frac{1}{2}\left(\frac{W}{x_{1+}}+\frac{W}{x_{2+}}\right)$	Kulczynski similarity index. The sum of minima is compared to the grand total of each site.
Pearson correlation	$s_{Pearson}(\mathbf{x}_1,\mathbf{x}_2)=r_{Pearson}(\mathbf{x}_1,\mathbf{x}_2)$	Computes Pearson correlation among objects instead of the normal correlation among variables (descriptors).
Chi-square similarity	$s_{\chi^2}(\mathbf{x}_1,\mathbf{x}_2)=1-\sqrt{\sum_{j=1}^{p}\frac{1}{x_{+j}}\left(\frac{x_{1j}}{x_{1+}}-\frac{x_{2j}}{x_{2+}}\right)^2}$	Chi-square similarity index. This is the complement of Chi-Square Metric. As the metric excludes double-zeros this index does the same. It needs positive data matrices because it uses values as frequencies.
Similarity Ratio	$s_{SR}(\mathbf{x}_1,\mathbf{x}_2)=\frac{\sum_{j=1}^{p}x_{1j}x_{2j}}{\sum_{j=1}^{p}x_{1j}^2+\sum_{j=1}^{p}x_{2j}^2-\sum_{j=1}^{p}x_{1j}x_{2j}}$	An angular measure of similarity
Simpson index	$s_{Simpson}(\mathbf{x}_1,\mathbf{x}_2)=\frac{a}{a+\min(b,c)}$	A binary index of similarity

2.3.2 Translating similarities to distances.

Many of the multivariate methods cannot work with similarities. If one wants to use the object relations provided by a similarity measure in distance based computations it is necessary to transform similarity values into distance values. Transformations are launched by selecting the **Transform D-S** menu item on the **Resemblance** menu (or pressing ***Ctrl-F8***). The following transforms are available:

$$d_{hl}=1-s_{hl} \tag{2.2a}$$

$$d_{hl}=\sqrt{s_{hh}+s_{ll}-2s_{hl}} \tag{2.2b}$$

$$d_{hl}=\sqrt{1-s_{hl}^2} \tag{2.2c}$$

Chapter 3: Ordination methods

3.1 Ordination concepts

In view of the obvious difficulty involved in graphically displaying the information contained in multivariate matrices when the number of objects and/or descriptor is high, several multivariate methods have been developed to assist in this task. Ordination methods build vector spaces linearly or non-linearly related to the original variables. A usual objective is to accumulate the variability of data in the first dimensions of the ordination space, in order to maximize the amount of information displayed in 2D or 3D ordination scatter diagrams. In general, the more dimensions of the original space there are, the less information will be shown in ordination diagrams. An important aspect to consider in this regard is the representativeness of the representation in reduced space.

Traditionally, ordination methods have been divided into canonical and non-canonical methods. If only one data matrix is used, the method is referred to as 'non-canonical' (or indirect gradient analysis). In a non-canonical ordination method the interpretation of the resulting axes is given by the contribution of the original variables. On the other hand, if the analysis needs two or more matrices it is broadly called 'canonical' (or direct gradient analysis in case of ecological data). In canonical methods, which are usually based on multiple regressions, interpretation of ordination is done by considering the contribution of explicative variables to the canonical axes.

Ordination methods available in GINKGO are displayed as menu items in the **Ordinate** menu of the main menu bar. If the user wants to launch non-canonical ordinations, a data matrix must be selected in the Data Editor. The type of data selected in the Data Editor's tree determines which analysis can be done. When selecting one analysis in the **Ordinate** menu, a dialog will be prompted to request analysis options. In most non-canonical ordination methods, the user can restrict the analysis to a submatrix of the selected matrix by clicking on **Set objects** or **Set descriptors**. When running ordination analysis on symmetric matrices only **Set objects** will be available. If the **Use all** checkbox is selected then all objects/variables will be used for the ordination.

Canonical methods are run similarly, though data matrices are selected in the prompted dialog. No restriction of objects or descriptors can be done (matrix cropping have been performed previously, though).

After selecting the desired parameterization and clicking on **Ok** the analysis will start, and a new item will be posted on the Analysis Manager tree.

3.2 Principal Components Analysis (PCA)

3.2.1 The method

Under a multinormal distribution, the first principal axis is the line that goes through the greatest dimension of the concentration hyperellipsoid describing the distribution. Extending this idea, the **principal axes** are a set of axes resulting from a rigid rotation of the original coordinate system, and corresponding to the successive directions of maximum variance of the scatter of points. The **principal components** give the positions of the objects in the new system of coordinates.

Principal Components Analysis (PCA, Hotelling 1933) is the simplest of the ordination methods. It finds the principal axes through the eigenvalue decomposition of the dispersion matrix **S**, i.e. solving:

$$(\mathbf{S} - \lambda_k \mathbf{I})\mathbf{u}_k = \mathbf{0} \tag{3.1}$$

where the dispersion matrix is computed as

$$\mathbf{S} = \frac{1}{n-1}\left[\mathbf{x} - \bar{\mathbf{x}}\right]'\left[\mathbf{x} - \bar{\mathbf{x}}\right]. \tag{3.2}$$

Eigenvectors $\mathbf{u}_k$ are the principal axes of dispersion matrix **S**. A maximum of $c = p$ principal axes may be derived from a data table containing p variables. They are normalized to unit length before computing the principal components, which give the coordinates of the objects on the successive principal axes. Since any dispersion matrix **S** is symmetric, its principal axes $\mathbf{u}_k$ are orthogonal to one another (i.e. they correspond to linearly independent directions in the concentration ellipsoid of the distribution of objects). On the other hand the eigenvalues λ_k of a dispersion matrix give the amount of variance corresponding to the successive principal axes. Because of the above properties, PCA can summarize in a few dimensions most of the variability of a dispersion matrix of a large number of descriptors.

The elements u_{jk} of the eigenvectors are also weights, or **loadings**, of the original descriptors in the linear combination of descriptors from which the principal components are computed. Thus, the position of object ω_i on the first principal axis is given by the linear combination:

$$f_{i1} = \left[\mathbf{x} - \bar{\mathbf{x}}\right]_i \cdot \mathbf{u}_1 \tag{3.3}$$

and the position of all objects in the new system of c axes, $\mathbf{F}_{(n\ x\ c)}$, i.e. the matrix of principal components (or scores), is computed using the matrix of eigenvectors ($\mathbf{U}_{(p\ x\ c)}$):

$$\mathbf{F} = \left[\mathbf{x} - \bar{\mathbf{x}}\right] \cdot \mathbf{U}\,. \tag{3.4}$$

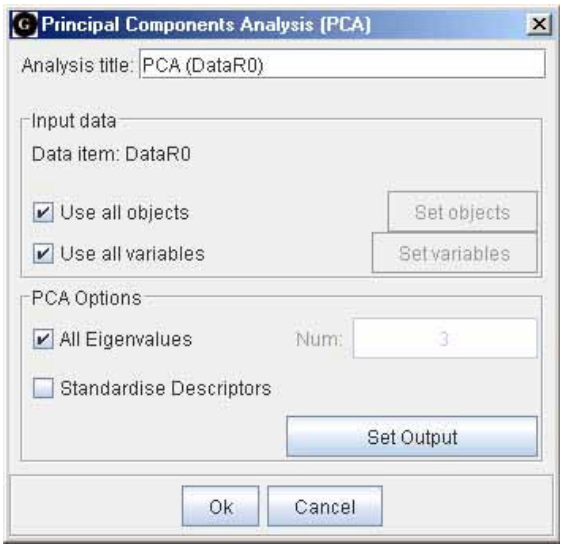
As can be seen from the previous equations, the system of principal axes is centered with respect to the scatter of point-objects. The quality of the representation in a reduced Euclidean space with m dimensions only ($c \geq m$) may be assessed by the ratio:

$$\%\ \text{var} = \left(\sum_{k=1}^{m} \lambda_k\right) \Big/ \left(\sum_{k=1}^{c} \lambda_k\right) \tag{3.5}$$

Note that the denominator is actually equal to the trace of **S**.

3.2.2 Running PCA

PCA can only be executed on an object-descriptor matrix. Starting from this matrix (**X**), the procedure computes the dispersion matrix **S** and performs eigenvalue decomposition on it. Apart from cropping the matrix to be analyzed, there are two other PCA options that can be changed by the user. First, the user can restrict the number of ordination axes to be found (by default, all axes will be computed). Second, original descriptors can be standardized prior to PCA. This causes eigenanalysis to be based on correlation matrix instead of dispersion matrix. It may be advisable to use this option when original descriptors are in different units.



3.2.3 PCA standard output

The PCA procedure outputs a text with a list of eigenvalues and their corresponding percentage of variance (i.e. the proportion of **S** trace). Cumulative percentages of explained variance are also given. On the other hand two matrices are given, the object principal components (matrix **F**) and the variable loadings (matrix **U**).

3.2.4 PCA scalings, plots and preserved distances

As mentioned in subsection 3.2.1 eigenvectors are usually scaled to unit length. Under this scaling, Euclidean distances calculated between objects using all principal components are equal to the Euclidean distances on the original space. If only the first principal components are used, to the Euclidean distances on the original space are only approximated. Thus, distances between objects plotted in a scatter diagram of principal components should be taken cautiously when the cumulative percentage of explained variance is not close or equal to 100%.

It is also possible to plot variable loadings on a scatter diagram. The higher the value of a variable on an axis, the greater weight it has on the axis definition. Normalizing the eigenvectors to unit length also normalizes the axes of descriptors in the ordination space. Thus, taking into account all ordination space, the directions of original descriptors are orthogonal to each other (since **U** is an orthonormal square matrix). This does not mean that, in a scatter diagram of variable loadings, descriptor-axes are shown as orthogonal (since some ordination axes may be lacking). Note that angles among descriptor-axes in such a diagram do not approximate their correlation values. The aim of building such plots is to visualize the weight of the original

variables on the construction of the ordination axes. When both principal components and variable loadings are plotted in the same diagram, it is called a **distance biplot**.

There is a second approach to the study of the relationships among descriptors. It consists in scaling the eigenvectors in such a way that the cosines of the angles between descriptor-axes are proportional to their covariances. In this approach, the angles between descriptor-axes are between 0º (maximum positive covariance), and 180º (maximum negative covariance). An angle of 90º indicates a null covariance (orthogonality). This approach is obtained by scaling the eigenvectors to a length equal to their standard deviation. That is, for all $\mathbf{u}_k$ multiply by $\sqrt{\lambda_k}$. In matrix form, we can write this new scaling as $\mathbf{U} \cdot \mathbf{\Lambda}^{1/2}$. Under this second scaling, the projection of a descriptor-axis on a principal axis shows its covariance and, thus, its positive or negative contribution. Correlations among original descriptors are given by the angle between descriptor axes (on the full space), not by the proximity between apices of axes. If the representation is not on full space, the angles between axes of descriptors only approximate their correlations.

Objects can also be plotted onto this new scaling. Applying the same transform to axes, $\mathbf{G} = [\mathbf{x} - \bar{\mathbf{x}}] \cdot \mathbf{U} \cdot \mathbf{\Lambda}^{1/2}$, does this. Euclidean distances between objects on this scaling do not approximate Euclidean distances of the original space, but rather approximate Mahalanobis distances on the original space. Again, principal components and variable loadings can be plotted together on this scaling. Such a biplot is called a **correlation biplot**.

3.2.5 Meaningful components

The successive principal components correspond to successively smaller fractions of variance. One problem is therefore to determine how many components are meaningful. The best approach may be to study the representativeness of projections in reduced space for two, three or more dimensions, using Shepard diagrams. This consists in plotting the Euclidean distances on the reduced ordination space against the Euclidean distances on the full ordination space. If the object distances are equal or proportional in both projections, the points in the Shepard diagram will form a straight line. However, the more elliptical the configuration of points in the Shepard diagram is, the less similar the two projections will be; and therefore the less representative the projection in reduced space will be. Alternatively, it may also be helpful to draw a **scree plot**, where eigenvalues are plotted in descending order.

There is an empirical rule (also known as the Kaiser-Guttman criterion) suggesting that one should only interpret a principal component if the corresponding eigenvalue is larger than the mean of the eigenvalues. If standardized descriptors are used in PCA (**S** is actually a correlation matrix) the mean of eigenvalues is 1.

3.3 Metric Multidimensional Scaling (classical MDS)

3.3.1 Introduction

Classical MDS (also called Principal Coordinates Analysis) was created by Gower (1966) as a method to obtain Euclidean representations (i.e. in a Cartesian space) from a set of objects whose relations are obtained by using any dissimilarity measure. Thus, it is performed on a dissimilarity matrix **D**. One interesting aspect of classical MDS is that it may be used with all types of original **X** variables, provided that a coefficient appropriate to the data has been used to compute the distance matrix.

Principal components in PCA are linear combinations of the original descriptors. In contrast, principal coordinates of classical MDS are complex functions of the original descriptors mediated through a dissimilarity measure. However, both analyses yield Euclidean spaces displayable in scatter diagrams. When classical MDS is run from a Euclidean distance matrix, the principal coordinates obtained are equal to the principal components of PCA. However, if other dissimilarity measures are used to measure object relations this equivalence between methods is lost.

When using classical MDS on a dissimilarity matrix not embeddable on a Euclidean space, negative eigenvalues are obtained and thus some ordination axes are imaginary (see below). Non-metric Multidimensional Scaling, NMDS, also obtains ordinations from any dissimilarity (or similarity) matrix (see section 3.4). One advantage of NMDS over classical MDS is that the former always yields real Euclidean axes no matter what the original dissimilarity measure used is. However, NMDS can be computationally very expensive.

3.3.2 Classical MDS algorithm

MDS starts from a symmetric distance matrix (**D**)[3]. The classical MDS algorithm includes the following steps:

1. Matrix **D** is transformed to **A** applying the following equation for each pair of objects:

$$a_{lh} = -(1/2)d_{lh}^2 . \tag{3.6}$$

2. Matrix **A** is then centered to the coordinates' origin, yielding the *inner product matrix* $\Delta = [\delta_{lh}]$, whose elements are:

$$\delta_{lh} = a_{lh} - \overline{a}_l - \overline{a}_h + \overline{a} , \tag{3.7}$$

   where the column and row means are subtracted from each a_{lh} element and the grand mean is added.

3. Compute an eigenvalue decomposition Δ, obtaining a diagonal matrix of eigenvalues (Λ) and a matrix of eigenvectors (**U**). Scale eigenvectors to have a length equal to the square root of the corresponding eigenvalue. These scaled eigenvectors correspond to the principal coordinates of objects. That is, assuming that **U** is initially column-scaled to unit length:

$$\mathbf{X} = \mathbf{U}\mathbf{\Lambda}^{1/2} . \tag{3.8}$$

[3] If one has a symmetric similarity matrix **S**, its values must be transformed into distances (subsection 2.3.2).

Due to the centering (step 2), matrix Δ has always at least one zero eigenvalue. Note that, at most, (n – 1) axes are necessary for representing n points in Euclidean space. There may be more than one zero eigenvalue in the case where the distance matrix is degenerate (i.e. when the n objects can be represented in less than n – 1 dimensions). If matrix **D** is the Euclidean Distance computed from a rectangular matrix with more objects than descriptors ($n > p$), then the maximum number of non-zero eigenvalues is p. Other resemblance coefficients may produce fewer or more axes. The transformation of **A** into Δ (step 2) is not essential. It is simply meant to eliminate one of the eigenvalues (turn it into 0), concretely the eigenvalue that would only account for the distance between the centroid and the origin.

By construction, classical MDS preserves the original distances, regardless of the formula used to compute them. The representation of principal coordinates in reduced space forms the best Euclidean approximation of the original distances, in the sense that the sum of squares of the projection distances of the objects onto the selected subspace is minimum. As with PCA, the quality of the representation can be assessed by means of a Shepard diagram or by a scree plot.

As previously mentioned, running a classical MDS starting from a matrix of the Pythagorean Euclidean Distance is equivalent to running a PCA. However, MDS does not produce the relation between original descriptors and the principal coordinates, as PCA does, because it does not start from a rectangular data matrix. However, one could *a posteriori* compute covariances and correlations with original descriptors.

3.3.3 Negative eigenvalues

Some symmetric dissimilarity matrices do not allow a full real Euclidean representation. This happens in two situations: 1) If dissimilarities do not fulfill the triangle inequality (measure d is not a metric); or 2) If the measure is a metric but its resemblance values are not fully embeddable in a Euclidean space. When transforming similarities we must be aware of the Euclidean properties of the transformed measure. For example, transform (2.2a) could yield metric but non-Euclidean relations. On the other hand, (2.2c) will normally yield fully embeddable spaces.

A dissimilarity not fully embeddable in a Euclidean space will yield negative eigenvalues in the eigenvalue decomposition phase of classical MDS. This means there will be imaginary principal coordinates, which cannot be represented along with the real part. To solve this, some corrections for non-euclideanarity have been proposed.

The Lingoes (1971) correction method is the one implemented in GINKGO. In this method, the largest negative eigenvalue (m) yielded by classical MDS is used to correct the original dissimilarity matrix as follows:

$$d_{lh} = \sqrt{d_{lh}^2 + 2m}\ . \qquad (3.9)$$

After this correction of distances, MDS is run again and then no negative eigenvalues are produced. However, corrections for non-Euclideanarity can be potentially distorting, because small distance values will be incremented more than large distance values. This has an overall effect of make possible cluster structures present in data more fuzzy.

3.3.4 Running classical MDS in GINKGO.

Classical MDS can only be run when a dissimilarity matrix is selected on the Data Editor. Once the MDS dialog appears, the user can change some analysis options. First, some elements from the original dissimilarity matrix may be excluded from the computations by unselecting the checkbox '**use all objects**'. Second, the user may ask for the Lingoes correction of negative eigenvalues by selecting checkbox '**Negative EV correction**'. Finally, extra output matrices can be requested from the analysis (see below).

Similarly to PCA, the routine for classical MDS lists eigenvalues and their associated % of variances, as well as the cumulative percentages. The user must be aware that, whenever negative eigenvalues are produced, cumulative percentages of real eigenvalues can exceed 100% since proportions are computed using overall variance. Therefore, these numbers must be regarded with caution. After adding variance of imaginary axes, the cumulative percentage at the end is 100% (note that variance of imaginary axes is negative).

As negative eigenvalues may be produced, GINKGO can be asked to output both real and imaginary principal coordinates. It also outputs the corresponding distance spaces (real and imaginary space). If no correction has been done, the initial dissimilarity matrix can be recovered by subtracting the squared imaginary distance elements from the squared real distance elements: $d^2_{lh} = d^2_{lh(real)} - d^2_{lh(imag)}$ (the program can also be asked to produce the resulting matrix). On the other hand, if correction for negative eigenvalues is done, the dissimilarity matrix computed with this formula will differ from the initial dissimilarity matrix.

A final additional output option for classical MDS is to compute the position of cluster centroids in the space of principal coordinates. Obviously, the user is then asked to choose one classification matrix among those available in the analysis manager. Ideally, this classification should have been obtained from the same dissimilarity matrix being used in MDS. Plotting cluster centroids in MDS diagrams is a good way to combine results from ordination with those of classification.

3.4 Non-metric Multidimensional Scaling (NMDS)

3.4.1 Introduction

All space-ordination methods attempt to represent objects in a reduced space while preserving, as well as possible, the distance relationships among them. Sometimes it is not of primary importance to preserve the exact distance values. Instead, one can preserve *the ordering relationships* of objects and be able to represent them in a small and specified number of dimensions. The method that does so is called **non-metric multidimensional scaling** (NMDS). The NMDS method and first algorithm were proposed by Kruskal (1964a, 1964b), although many other algorithms have been proposed. Kruskal's NMDS is the algorithm implemented in GINKGO.

Like its classical metric counterpart, NMDS is not limited to Euclidean distance matrices. It can produce ordinations of objects from any dissimilarity matrix. The method can also handle missing values as long as there are enough distance measures left. Contrary to PCA, classical MDS or CA, which are eigenvector methods, NMDS calculations do not maximize the variability associated with individual axes of the ordination. NMDS axes are arbitrary, so that plots may arbitrarily rotated, centered or inverted. Thus, it is common to perform, after NMDS, a final rotation (PCA) to maximize the variance contained on the first axes.

NMDS is an iterative search for a ranking and placement of *n* entities on *m* dimensions (axes) that minimizes the **stress** of the *m*-dimensional configuration. **Stress** is a measure of departure from monotonicity in relationship between the dissimilarity in the original *p*-dimensional space and distance in the reduced *m*-dimensional ordination space.

3.4.2 Kruskal's NMDS algorithm

Consider a general dissimilarity matrix $[\delta_{hl}]$ computed using a measure appropriate to the data at hand. The user must specify the number of *m* dimensions for scaling the objects. The output will provide coordinates of the *n* objects on *m* axes. NMDS then proceeds as follows:

1. Construct an initial $\mathbf{X}_{n \times m}$ configuration of the *n* objects in *m* dimensions to be used as a starting point for the iterative algorithm.
2. Normalize configuration.
3. Calculate a matrix **D** of fitted distances d_{hl} in the ordination space, using one of the Minkowski's metrics (2^{nd} order metric, the Euclidean distance, in the case of GINKGO). In the first iteration distances d_{hl} are computed from the initial *m* space configuration, and in the others they are the modifications of those distances.
4. One regression must then be chosen to regress d_{hl} on δ_{hl}. Values forecasted by the regression line are called $\hat{d}_{hl}$. In case of GINKGO a monotone regression is implemented. Monotone regression is a step-function, which is constrained to always increase from left to right. It is equivalent to a linear regression performed after monotonic transformation of the original distances, so as to maximize the linear relationship between d_{hl} and δ_{hl}. The regression is fitted by least squares (stress function).
5. Measure the goodness-of-fit of the regression using an objective function. This is based on the sum of squared differences between fitted values d_{hl} and the forecasted $\hat{d}_{hl}$ values by the regression function:

$$Stress = \sqrt{\frac{S*}{T*}} = \sqrt{\frac{\sum_{h,l}^{n}(d_{hl} - \hat{d}_{hl})^2}{\sum_{h,l}^{n} d_{hl}^2}} \tag{3.10}$$

6. Improve the configuration by moving it slightly in a direction of decreasing stress. This is done by methods of numerical optimization based on gradient, such as *steepest descent*. The direction of steepest descent is the direction in the space of solutions along which stress is decreasing more rapidly. We first calculate the *negative gradient of stress* for each point *i* (i.e. the direction of movement of each point *i* in the *k*th dimension of the *m*-space in which stress decreases faster). The gradient vector *g* is calculated by:

$$g_{ik} = Stress \cdot \sum_{h=1}^{n}\sum_{l=1}^{n}(\delta^{ih} - \delta^{il})\left[\frac{d_{hl} - \hat{d}_{hl}}{S*} - \frac{d_{hl}}{T*}\right]\frac{(x_{hk} - x_{jk})}{d_{hl}} \tag{3.11}$$

The letter *h* indexes one of the *n* points, *l* indexes another of the *n* points, *k* indexes a particular dimension, and *i* indexes a third point, the point of interest which is moved on dimension *k*. So g_{ik} indicates a shift for point *i* along dimension *k*. Here δ^{ih} and δ^{il} are Kronecker deltas which have the value 1 if *i* and *h* or *i* and *l* indices are equal, otherwise they have value 0. The amount of movement in the direction of the negative gradient is set by the step size, α, which is initially set at 0.2 but can change during the iterations. The next configuration, **X'**, is obtained by:

$$x'_{ik} = x_{ik} + \frac{\alpha}{mag(g)} g_{ik} \tag{3.12}$$

where *mag*(*g*) means the relative magnitude of *g* and is given by (assuming that *x* is normalized):

$$mag(g) = \sqrt{\frac{1}{n}\sum_{i}^{n}\sum_{k}^{m} g_{ik}^2}\ . \tag{3.13}$$

7. We repeat steps 3-6 until convergence is achieved (a minimum). The last step coordinates of step 6 are the ordination axes of the *n* objects in the *m* dimensions.

3.4.3 Running NMDS in GINKGO

NMDS options

NMDS is run starting from a symmetric dissimilarity matrix (if the starting matrix is a symmetric similarity matrix, a transform should be applied first). As with other symmetric starting methods, changing the objects included in the analysis results in using a submatrix for the computation. The other NMDS options are:

Number of dimensions: The number of dimensions to be searched. The lower the number of dimensions selected, the higher the *stress* will be.

Starting mode: As with K-means (section 4.6), starting mode is very important in NMDS. This starting point is critical, as it is in most optimization algorithms (like K-means, space may contain several minima besides the overall minimum). Two options are provided in GINKGO as starting options:

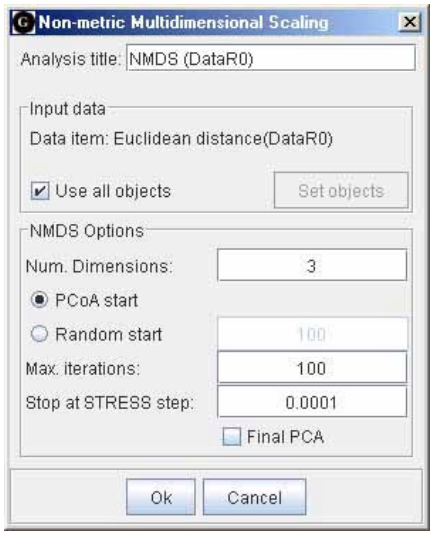
- *Random start*: Run the program several times, starting from different random placement of objects.
- *PCoA start*: Initiate the run from a classical MDS ordination.

The recommended procedure is to perform a metric MDS (classical MDS) first, and to select the desired number of axes. Those axes are used as starting coordinates for NMDS[4]. Alternatively, randomly generated coordinates can be chosen as starting point. In such cases, the number of random startings can be specified.

Maximum number of iterations: The user can specify a number of iterations to be done before stopping. Normally this option is not changed

Stop at stress step: As *stress* will be reduced with each step, a threshold for stopping the iterations must be chosen. Normally this option is not changed.

Final PCA: NMDS does not maximize the variance in his axis. It is a common practice to perform a final PCA (i.e. a rotation) to cumulate variance in first axis (see section 3.2).



NMDS output

NMDS output is different depending on the starting mode and whether a final PCA is performed or not. However, NMDS procedure always yields the final object coordinates, and the (Euclidean) distance matrix on the NMDS space. The procedure also outputs the number of NMDS iterations used and the final NMDS STRESS value.

[4] See section 3.3 for a description of how metric scaling works.

3.5 Correspondence Analysis (CA)

3.5.1 Introduction

Correspondence analysis was developed independently by several authors. It was first proposed for analysing two-way contingency tables, where the states of a first descriptor (rows) are compared to the states of a second descriptor (columns). Data in each contingency table cell are frequencies, i.e. numbers of objects coded with a combination of states of the two descriptors. These frequencies are thus positive integers or zeros.

In ecology, CA is often used to analyze species data at different sampling sites, where values are also zero or positive numbers of species abundances. The rows and columns then correspond to sites and species, respectively. Such a table can be treated analogously to a contingency table. CA can also be used on contingency tables that compare two groups of descriptors. In general, CA can be applied to any data table that is dimensionally homogeneous and only contains positive integers or zero values.

CA is primarily a method of ordination. As such it is similar to PCA. It preserves, in the space of principal axes, the Euclidean distance between profiles of weighted conditional probabilities. In other words, the χ^2 distance is used to quantify the relationships between rows and columns, which in this analysis are treated symmetrically. Thus, the Euclidean distance between objects on the ordination axes equals the χ^2 distance on the original contingency table.

Besides its role as an ordination method, CA may be used to study the proximities between rows or columns of the contingency table, as well as the correspondence between rows and columns.

3.5.2 The method

Let **X** be a contingency table with r rows and c columns. Therefore, their cell values (x_{ij}) should be regarded as frequencies ($x_{ij} = f_{ij}$). Then, a matrix $\mathbf{P} = [p_{ij}]$ of relative frequencies can be computed. Assume that the table is written in such a way that $r \geq c$ (it may be transposed, since rows and columns are treated symmetrically in CA). The steps in Correspondence Analysis are as follows:

1. Remember that Pearson χ^2 statistic is a sum of squared χ_{ij} values, computed for every cell ij of the contingency table. Each χ_{ij} value is the standardized residual of a frequency f_{ij} after fitting a null model to the contingengy table. The null model states that there is no relationship between rows and columns of the table. CA is based upon a matrix **Q** (r x c), whose cells are defined very similarly to χ_{ij} values. Thus **Q** is a matrix of departures (residuals) from the null model:

$$\mathbf{Q} = [q_{ij}] = \left[\frac{p_{ij} - p_{i+}p_{+j}}{\sqrt{p_{i+}p_{+j}}} \right] = \left[\frac{x_{ij}x_{++} - x_{i+}x_{+j}}{x_{++} \cdot \sqrt{x_{i+}x_{+j}}} \right] \qquad (3.14)$$

where $x_{i+} = \sum_{j=1}^{c} x_{ij}$ and $x_{+j} = \sum_{i=1}^{r} x_{ij}$ are the row and column marginals, respectively (the same for relative frequencies p_{ij}); and $x_{++} = \sum_{i=1}^{r} \sum_{j=1}^{c} x_{ij}$ is the grand total. The sum of squares of all values in matrix **Q** ($\sum q_{ij}^2$) measures the total inertia in **Q**.

2. Singular value decomposition is applied to matrix **Q**, with the following result:

$$\mathbf{Q}_{(rxc)} = \hat{\mathbf{U}}_{(rxc)} \mathbf{W}_{(rxc)} \mathbf{U}'_{(cxc)} \tag{3.15}$$

where both **U** and $\hat{\mathbf{U}}$ are column-orthonormal matrices and **W** is a diagonal matrix of singular values. Since $\hat{\mathbf{U}}$ is orthonormal, $\hat{\mathbf{U}}'\hat{\mathbf{U}} = \mathbf{I}$ and:

$$\mathbf{Q}'\mathbf{Q} = \mathbf{U}\mathbf{W}'\hat{\mathbf{U}}'\hat{\mathbf{U}}\mathbf{W}\mathbf{U}' = \mathbf{U}\mathbf{W}'\mathbf{W}\mathbf{U}' \tag{3.16}$$

The diagonal matrix **W'W** which contains squared singular values on its diagonal, is the diagonal matrix of eigenvalues of **Q'Q**. In addition **U** is equal to the matrix of eigenvectors of **Q'Q**. Therefore, identical results can be obtained applying eigenanalysis to **Q'Q**. Both singular value decomposition of **Q** and eigenvalue analysis of **Q'Q** always yield one null eigenvalue, due to the centering step implicit in the construction of **Q**. Thus, there are (c – 1) positive eigenvalues or singular values when $r \geq c$.

3. Matrices **U** and $\hat{\mathbf{U}}$ may be used as scores to plot the positions of the row and column vectors in two separate scatter diagrams. For joint plots various scalings have been proposed. First, matrices **U** and $\hat{\mathbf{U}}$ can be weighted by the inverse of the square roots of the column and row scores:

$$\mathbf{V}_{(cxc)} = \mathbf{D}(p_{+j})^{-1/2}\mathbf{U} \tag{3.17}$$

$$\hat{\mathbf{V}}_{(cxc)} = \mathbf{D}(p_{i+})^{-1/2}\hat{\mathbf{U}}\,. \tag{3.18}$$

Matrix **F**, which gives the positions of the rows of the contingency table in the correspondencey analysis space, is obtained from the transformed matrix of eigenvectors **V**, which gives the positions of the columns in that space. This is done by applying the usual equation for component scores to **Q**, including a division by the row weights:

$$\mathbf{F}_{(rxc)} = \mathbf{D}(p_{i+})^{-1}\mathbf{Q}\mathbf{V}\,. \tag{3.19}$$

In the same way, matrix $\hat{\mathbf{F}}$, which gives the positions of the columns of the contingency table in the correspondence analysis space, is obtained from the transformed matrix of eigenvectors $\hat{\mathbf{V}}$, which gives the positions of the rows in that space:

$$\hat{\mathbf{F}}_{(cxc)} = \mathbf{D}(p_{+j})^{-1}\mathbf{Q}'\hat{\mathbf{V}}\,. \tag{3.20}$$

From the last CA computation step, it should be clear that there are two possible scalings to draw biplots of rows and columns:

- *Scaling type 1*: Draw a joint plot with the rows (**F**) at the centroids of the columns (**V**). In this plot, distances between rows (i.e. sites in ecology) preserve their χ^2 distances. Note that positions of the centroids are calculated using weights equal to the relative frequencies of the columns (species), so the ordination of rows (sites) is meaningful.
- *Scaling type 2*: Draw a joint plot with the columns ($\hat{\mathbf{F}}$) at the centroids of the rows ($\hat{\mathbf{V}}$). In this plot distances between columns (i.e. species in ecology) preserve their χ^2 distances.

3.5.3 Running CA

CA can only be run from a rectangular object-descriptor matrix containing positive or zero values. As in PCA, some objects or descriptors can be excluded from computations. The user can also limit the number of axes to be kept, unselecting the '**All eigenvalues**' checkbox and entering a suitable number.

As text output, GINKGO gives a table of singular values, along with their corresponding eigenvalues and their contribution to the total inertia of **Q**.

By default, the CA routine gives matrices for scaling 1 (**V** and **F**), but can be asked to also give the matrices for scaling 2 ($\hat{\mathbf{V}}$ and $\hat{\mathbf{F}}$). Depending on the final available matrices, the CA plots that can be built (on the **Plot** menu) are:

- *Scree plot*: Plot of eigenvalues (see 3.2.5)
- *Object's space plot*: Scatter plot of matrix **F**.
- *Character's space*: Scatter plot of matrix $\hat{\mathbf{F}}$.
- *CA biplot scaling 1*: Scatter biplot of matrices **V** and **F**.
- *CA biplot scaling 2*: Scatter biplot of matrices $\hat{\mathbf{V}}$ and $\hat{\mathbf{F}}$.

3.6 Redundancy Analysis (RDA)

3.6.1 Introduction to canonical analysis

Canonical analysis is the simultaneous analysis of two, or eventually several, data tables. In ecology, one may typically be interested in the relationship between a first table describing species composition and a second table consisting of environmental descriptors, observed at the same locations. In some canonical analyses the two tables are treated symmetrically, so they could be interchanged without effect (RMDS of section 3.8 is an example of this). In other canonical analyses, where regression techniques are involved, the two tables are treated asymmetrically. One of the matrices is used as an **explanatory matrix** (from now on we will call this matrix **X**) and the other as a **response matrix** (from now on we will call this matrix **Y**). Redundancy analysis and Canonical Correspondence Analysis are examples of the latter type of canonical analysis. In those analyses, matrix **X** intervenes in the ordination of **Y**, forcing the ordination vectors to be maximally related (through multiple regression) to combinations of the variables in **X**. This makes it possible to bring out all the variance of **Y** that is related to **X**, and to test some *a priori* hypotheses. Further examination of the unexplained variability of **Y** may help to generate new hypotheses to be tested using new observations.

3.6.2 Introduction to Redundancy analysis

Redundancy Analysis (RDA, van den Wollenberg 1977) is the direct extension of multiple regression to the modeling of multivariate response data. The term 'redundancy' is synonymous here with 'explained variance'. The analysis is asymmetric. In RDA, each canonical ordination axis corresponds to a direction in the response space of **Y**, which is maximally related to a **linear combination** of the explanatory variables **X**. In fact, three ordinations of the objects are obtained in RDA:

(1) Ordination of objects in the space of variables **Y**. Scores are linear combinations of the **Y** variables, as in PCA, but using canonical axes as loadings.
(2) Ordination of objects in the space of variables **X** or in the space of fitted values $\hat{\mathbf{Y}}$. Scores are linear combinations of the fitted $\hat{\mathbf{Y}}$ variables, which are also linear combinations of the **X** variables.
(3) Ordination of objects in the space of residuals ($\mathbf{Y} - \hat{\mathbf{Y}}$).

RDA preserves the Euclidean distance between objects in matrix $\hat{\mathbf{Y}}$ containing values of **Y** fitted by regression to the explanatory variables **X**.

3.6.3 Algebra of RDA

Let **X** (*n x m*) be the explanatory matrix of *m* explanatory descriptors on *n* objects, and let **Y** (*n x p*) be the response matrix of *p* descriptors on the same *n* objects. For convenience, these matrices are assumed to be column-centered. Problems of canonical analysis may be presented by the following partitioned covariance matrix, resulting from the fusion of the **Y** and **X** data sets; the joint dispersion matrix $\mathbf{S}_{\mathbf{Y+X}}$ contains blocks that are indentified as follows for convenience:

$$\mathbf{S}_{Y+X} = \begin{bmatrix} \mathbf{S}_{YY} & \mathbf{S}_{YX} \\ \mathbf{S}_{XY} & \mathbf{S}_{XX} \end{bmatrix} = \begin{bmatrix} \mathbf{S}_{YY} & \mathbf{S}_{YX} \\ \mathbf{S'}_{YX} & \mathbf{S}_{XX} \end{bmatrix} \tag{3.21}$$

where submatrices $\mathbf{S_{YY}}$ (p *x* p) and $\mathbf{S_{XX}}$ (m *x* m) concern each of the two sets of descriptors, whereas $\mathbf{S_{YX}}$ (p *x* m) and its transpose $\mathbf{S'_{YX}} = \mathbf{S_{XY}}$ (m *x* p) account for the covariances among the descriptors of the two groups.

The eigenanalysis equation for redundancy analysis is:

$$(\mathbf{S_{YX}}\mathbf{S_{XX}^{-1}}\mathbf{S'_{YX}} - \lambda_i \mathbf{I})\mathbf{u}_i = \mathbf{0} \tag{3.22}$$

and may be derived through multiple linear regression, followed by principal component decomposition (PCA). This can be proven as follows. If we do a multiple linear regression for each variable in table **Y** on all variables of X, the matrix of regression coefficients corresponding to the whole set of regressions (i.e. for all response variables) is

$$\mathbf{B} = [\mathbf{X'X}]^{-1}\mathbf{X'Y} \tag{3.23}$$

and the matrix of fitted values, which we will call $\hat{\mathbf{Y}}$, is:

$$\hat{\mathbf{Y}} = \mathbf{XB} = \mathbf{X}[\mathbf{X'X}]^{-1}\mathbf{X'Y} \tag{3.24}$$

Because variables in **X** and **Y** are centered on their means, there is no intercept parameter in the vectors of **B**. [If $m = n$, **X** is square; in that case, the multiple regression will always explain the variables in matrix **Y** entirely, so that $\hat{\mathbf{Y}} = \mathbf{Y}$]. The covariance matrix corresponding to the fitted values $\hat{\mathbf{Y}}$ is computed from:

$$\mathbf{S}_{\hat{\mathbf{Y}}'\hat{\mathbf{Y}}} = (1/(n-1))\hat{\mathbf{Y}}'\hat{\mathbf{Y}} = (1/(n-1))\mathbf{Y}'\mathbf{X}[\mathbf{X'X}]^{-1}\mathbf{X'X}[\mathbf{X'X}]^{-1}\mathbf{X'Y} = \mathbf{S_{YX}}\mathbf{S_{XX}^{-1}}\mathbf{S'_{YX}} \tag{3.25}$$

so thus it is shown that RDA applies PCA on the covariance matrix of the table of fitted values $\hat{\mathbf{Y}}$:

$$(\mathbf{S}_{\hat{\mathbf{Y}}'\hat{\mathbf{Y}}} - \lambda_i \mathbf{I})\mathbf{u}_i = \mathbf{0}. \tag{3.26}$$

The matrix containing normalized canonical eigenvectors is called **U** (p *x* p). They give the contributions of the descriptors in $\hat{\mathbf{Y}}$ to the various canonical axes. It contains only min[p, m, n – 1] eigenvectors with non-zero eigenvalues. They should be interpreted as in PCA. The ordination of objects in the space of the response variables can be obtained directly from the centered matrix **Y** by projecting on the canonical axes:

$$\mathbf{F} = \mathbf{YU} \tag{3.27}$$

Ordination vectors are sometimes called "site scores". They have variances that are close but not equal to the corresponding eigenvalues (since eigenvalue decomposition has been performed on the matrix $\hat{\mathbf{Y}}$ of fitted values and not on the original response matrix). The ordination of objects in space **X** (or $\hat{\mathbf{Y}}$) is obtained similarly:

$$\mathbf{Z} = \hat{\mathbf{Y}}\mathbf{U} = \mathbf{XBU} \tag{3.28}$$

These ordination vectors are sometimes called "fitted site scores". They have variances equal to the corresponding eigenvalues.

The contribution of explanatory variables (**C**) to the formation of canonical ordination axes can be known by computing **C** = **BU**.

Redundancy analysis does not completely explain the variation in the response variables (matrix **Y**). During the regression step, regression residuals may be computed for each variable **y**. The residuals are obtained as the difference between observed values and the corresponding fitted values. The matrix of residuals, $\mathbf{Y}_{res}$ (*n x p*) is thus obtained by:

$$\mathbf{Y}_{res} = \mathbf{Y} - \hat{\mathbf{Y}} \tag{3.29}$$

Residuals may be analyzed by Principal Components Analysis, leading to min[*p*, *n* – 1] non-canonical eigenvalues and eigenvectors ($\mathbf{U}_{res}$). Non-canonical object scores are then obtained by projecting $\mathbf{Y}_{res}$ on the non-canonical axes:

$$\mathbf{F}_{res} = \mathbf{Y}_{res}\mathbf{U}_{res} \tag{3.30}$$

When the variables in **X** are good predictors of the variables in **Y**, the canonical eigenvalues may be larger than the first non-canonical eigenvalues, but this is not always the case. If the variables in **X** are not good predictors of **Y**, the first non-canonical eigenvalues, computed on the residuals, may be larger than their canonical counterparts.

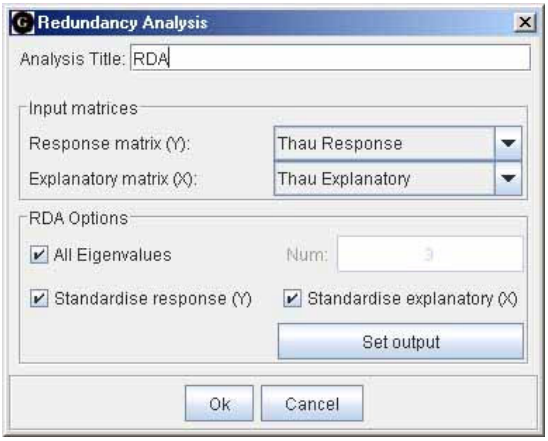
Note that the "site scores" approximate the observed data, which contains residuals ($\mathbf{Y} = \hat{\mathbf{Y}} + \mathbf{Y}_{res}$). On the other hand, the "fitted site scores" approximate only the fitted data. Either may be used in biplots.

In the case where **Y** contains a single response variable, redundancy analysis is nothing but multiple linear regression analysis.

3.6.4 Running RDA

When running RDA in GINKGO the user must specify which data items in the Data Editor will be used as explanatory and response. Note that if there are less than two object descriptor matrices RDA cannot be computed. Some other options are available:

- *All eigenvalues*: This option is selected by default, so all eigenvalues are sought. The user can specify how many eigenvalues are to be kept in the final results.
- *Standardize explanatory* (**X**): Centering of **X** is always done. If the explanatory variables are standardized, the possible effects of their different physical dimensions are removed. This turns regression coefficients into standard regression coefficients, which makes them comparable to one another.
- *Standardize response* (**Y**): Centering of **Y** is always done, but the user can standardize columns of the response matrix, which is advisable when the different variables are not dimensionally homogeneous.



Eigenvalue output information is listed in two separate blocks, one for canonical axes and the other for the non-canonical ones. The partition of the total variance of **Y** between the variance explained by regression and the residual variance is also given. The available RDA output matrices are:

- *Canonical Eigen Vectors* (**U**): Eigenvectors resulting from the decomposition of fitted response values covariance matrix.
- *Canonical Scores* (**F** = **YU**): Object scores (also called "site scores") resulting from projecting original response values on canonical axes.
- *Fitted Canonical Scores* ($\mathbf{Z} = \hat{\mathbf{Y}}\mathbf{U} = \mathbf{XBU}$): Object scores (also called "fitted site scores") resulting from projecting fitted response values on canonical axes.
- *Non-Canonical Eigen Vectors* ($\mathbf{U_{res}}$): Eigenvectors resulting from the Decomposition of Residual Response Values Covariance Matrix.
- *Non-Canonical Scores* ($\mathbf{F_{res}} = \mathbf{Y_{res}U_{res}}$): Object scores resulting from projecting residual response values on non-canonical axes.
- *Fitted Response Values* ($\hat{\mathbf{Y}} = \mathbf{XB}$): Fitted Values resulting from Multiple Regressions.
- *(Standardized) Regression Coefficients* (**B**): (Standardized) regression coefficients for (standardized) explanatory variables resulting from multiple regressions.
- *Canonical Coefficients* (**C** = **BU**): Weights of the explanatory variables **X** on the formation of the canonical axes.
- *Correlations* **X** *vs.* **F**: Pearson correlations between explanatory variables and canonical scores.

3.6.5 RDA plots and biplots

As in PCA, RDA plots and biplots could be based on two types of scaling, because canonical axes can be scaled to length equal to 1 or to length equal to its standard deviation ($\sqrt{\lambda_i}$). The same applies to the non-canonical axes. In GINKGO scaling to length 1 is the only scaling available for RDA.

Canonical ordination plots can be built in GINKGO using either **F**, **Z** or **U**. Additionally, **F** and **U** or **Z** and **U** can be used together in biplots because the products of the eigenvectors with the object score matrices reconstruct the original matrices (either **Y** or $\hat{\mathbf{Y}}$). These biplots are called *distance biplots* (in the scaling to length 1), and allow the interpretation to focus on the ordination of objects because Euclidean distances approximate their Euclidean distances in the space of response variables. Similarly, non-canonical ordination plots can be built using $\mathbf{F_{res}}$ or $\mathbf{U_{res}}$, and a non-canonical distance biplot is available showing the information on both matrices.

3.7 Canonical Correspondence Analysis (CCA)

3.7.1 Introduction to Canonical correspondence analysis

Canonical correspondence analysis (CCA, ter Braak 1986, 1995) is the canonical form of correspondence analysis (section 3.5). Any data table that could be subjected to correspondence analysis forms a suitable response matrix **Y** for CCA (see 3.6.1).

The mathematics of CCA is essentially the same as in redundancy analysis (see subsection 3.6.3). Thus CCA is essentially a multiple regression followed by an ordination analysis. However, there are some differences. The dependent data table in CCA is not matrix **Y** centered by variables as in RDA. CCA uses matrix **Q** of contributions to the χ^2 statistic (see correspondence analysis section). Relative frequencies in **Y** are also used in the scaling operations. Another difference is that weighted multiple regression is used instead of a conventional linear multiple regression. CCA approximates χ^2 distances between the rows of the dependent data matrix, subject to the constraint that the canonical ordination vectors have to be maximally related to weighted linear combinations of the explanatory variables.

3.7.2 Method and algebra of CCA

Here **Y** must be regarded as a contingency table with *n* rows and *p* columns. Therefore, their cell values (y_{ij}) should be regarded as frequencies ($y_{ij} = f_{ij}$). Then, a matrix **P** = [p_{ij}] of relative frequencies can be computed. In addition, the marginals have to be calculated:

$y_{i+} = \sum_{j=1}^{p} y_{ij}$ and $y_{+j} = \sum_{i=1}^{n} y_{ij}$ are the row (object) and column (variable) marginals.

$y_{++} = \sum_{i=1}^{n} \sum_{j=1}^{p} y_{ij}$ is the grand total.

The CCA method is as follows:

1. Compute **Q**, the matrix of departures (residuals) of the response matrix **Y** from the null model:

$$\mathbf{Q} = [q_{ij}] = \left[\frac{p_{ij} - p_{i+}p_{+j}}{\sqrt{p_{i+}p_{+j}}} \right] = \left[\frac{y_{ij}y_{++} - y_{i+}y_{+j}}{y_{++} \cdot \sqrt{y_{i+}y_{+j}}} \right] \qquad (3.31)$$

   As in correspondence analysis, the sum of squares of all values in matrix **Q** ($\sum q_{ij}^2$) measures the total inertia in **Q**.

2. The explanatory matrix **X** is standardized using weights **D**(y_{i+}).
3. Weighted multiple regression is used instead of the conventional multiple regression. To do a weighted multiple regression, the weights, given by diagonal matrix **D**$(p_{i+})^{1/2}$, must be applied to matrix **X** everywhere it occurs. The multiple regression coefficients of RDA are in CCA computed as:

$$\mathbf{B} = [\mathbf{X}'\mathbf{D}(p_{i+})\mathbf{X}]^{-1}\mathbf{X}'\mathbf{D}(p_{i+})^{1/2}\mathbf{Q} \qquad (3.32)$$

   and the equation for computing $\hat{\mathbf{Y}}$ is:

$$\hat{\mathbf{Y}} = \mathbf{D}(p_{i+})^{1/2}\,\mathbf{XB} \tag{3.33}$$

4. Eigenvalue decomposition is carried out on the covariance matrix $\mathbf{S}_{\hat{\mathbf{Y}}'\hat{\mathbf{Y}}}$, which in this case is simply:

$$\mathbf{S}_{\hat{\mathbf{Y}}'\hat{\mathbf{Y}}} = \hat{\mathbf{Y}}'\hat{\mathbf{Y}}. \tag{3.34}$$

   Eigenvalue decomposition produces a diagonal matrix of eigenvalues Λ and a matrix $\mathbf{U}$ of eigenvectors.

5. The normalized matrix $\hat{\mathbf{U}}$ is obtained using

$$\hat{\mathbf{U}} = \mathbf{QU}\boldsymbol{\Lambda}^{1/2} \tag{3.35}$$

   This matrix does not contain the loadings of the rows of $\hat{\mathbf{Y}}$ on the canonical axes, but the loadings of the rows of $\mathbf{Q}$ on the canonical axes, as in correspondence analysis. It is used to find the "site scores" in the space of the original variables. Canonical column and row scores for the two CA scalings (matrices $\mathbf{V}_{\mathbf{Can}}$, $\hat{\mathbf{V}}_{\mathbf{Can}}$, $\mathbf{F}_{\mathbf{Can}}$ and $\hat{\mathbf{F}}_{\mathbf{Can}}$) are obtained using the transformations of (3.17) (3.18), (3.19) and (3.20).

6. The "fitted site scores", linear combinations of the environmental variables, are found from $\hat{\mathbf{Y}}$ using the following equations:

$$\mathbf{Z}_{scaling\ 1} = \mathbf{D}(p_{i+})^{-1/2}\,\hat{\mathbf{Y}}\mathbf{U} \tag{3.36}$$

$$\mathbf{Z}_{scaling\ 2} = \mathbf{D}(p_{i+})^{-1/2}\,\hat{\mathbf{Y}}\mathbf{U}\boldsymbol{\Lambda}^{-1/2} \tag{3.37}$$

7. Residuals ($\mathbf{Y}_{\mathbf{res}} = \mathbf{Q} - \hat{\mathbf{Y}}$) can be analyzed by applying eigenvalue decomposition to covariance matrix $\mathbf{S}_{\mathbf{Y}_{\mathbf{res}}'\mathbf{Y}_{\mathbf{res}}} = \mathbf{Y}_{\mathbf{res}}'\mathbf{Y}_{\mathbf{res}}$, producing matrices of non-canonical eigenvalues and eigenvectors. Then a non-canonical matrix $\hat{\mathbf{U}}$ is obtained using (3.35) and the non-canonical scores for both rows and columns are computed again using equations (3.17), (3.18) (3.19) and (3.20). On these equations the row and column sums $\mathbf{D}(p_{i+})^{-1/2}$ and $\mathbf{D}(p_{+j})^{-1/2}$ from the original matrix $\mathbf{Y}$ have to be used.

3.7.3 Running CCA

As with its linear counterpart, when running CCA in GINKGO the user must specify which two data items in the Data Editor will be used as explanatory and response matrices. Note that if there are less than two object descriptor matrices CCA cannot be computed. Other options include whether all eigenvalues have to be kept and which output matrices are desired.

Eigenvalue output information is listed in two separate blocks, one for canonical axes and the other for the non-canonical ones. The partition of the total inertia of $\mathbf{Q}$ between the inertia explained by regression and the residual inertia is also given.

The available output matrices of CCA are:

- *Canonical site scores scaling 1* ($\mathbf{F_{Can}}$): Row scores on the canonical axes for scaling 1.
- *Canonical site scores scaling 2* ($\hat{\mathbf{V}}_{\mathbf{Can}}$): Row scores on the canonical axes for scaling 2.
- *Canonical species scores scaling 1* ($\mathbf{V_{Can}}$): Column scores on the canonical axes for scaling 1.
- *Canonical species scores scaling 2* ($\hat{\mathbf{F}}_{\mathbf{Can}}$): Column scores on the canonical axes for scaling 2.
- *Non-canonical site scores scaling 1* ($\mathbf{F_{NonCan}}$): Site Scores on the non-canonical axes for Scaling 1.
- *Non-canonical site scores scaling 2* ($\hat{\mathbf{V}}_{\mathbf{NonCan}}$): Row scores on the non-canonical axes for scaling 2.
- *Non-canonical species scores scaling 1* ($\mathbf{V_{NonCan}}$): Column scores on the non-canonical axes for scaling 1.
- *Non-canonical species scores scaling 2* ($\hat{\mathbf{F}}_{\mathbf{NonCan}}$): Column scores on the non-canonical axes for Scaling 2
- *Canonical eigenvectors* ($\mathbf{U_{Can}}$): Eigenvectors resulting from the decomposition of fitted response values covariance matrix ($\mathbf{S}_{\hat{\mathbf{Y}}'\hat{\mathbf{Y}}} = \hat{\mathbf{Y}}'\hat{\mathbf{Y}}$).
- *Non-canonical eigenvectors* ($\mathbf{U_{NonCan}}$): Eigenvectors resulting from the decomposition of the residual response covariance matrix ($\mathbf{S}_{\mathbf{Y_{res}}'\mathbf{Y_{res}}} = \mathbf{Y_{res}}'\mathbf{Y_{res}}$).
- *Explanatory variable weights* ($\mathbf{C} = \mathbf{BU_{Can}}$): Weights (contribution) of explanatory variables in the formation of the matrix of fitted site scores.
- *Correlation* $\mathbf{X}$ *vs* $\hat{\mathbf{V}}_{\mathbf{Can}}$: Correlations between explanatory variables and site scores.
- *Fitted response values* ($\hat{\mathbf{Y}}$): Fitted values resulting from weighted multiple regressions.
- *Regression coefficients* ($\mathbf{B}$): Variable coefficients resulting from weighted multiple regressions.

3.7.4 CCA plots and biplots

GINKGO allows CCA plots of row (sites, $\mathbf{F_{Can}}$) space or column (species, $\hat{\mathbf{F}}_{\mathbf{Can}}$) space. With scaling type 1, canonical biplots can be drawn using either $\mathbf{F_{Can}}$ and $\mathbf{V_{Can}}$, or $\mathbf{Z}_{\boldsymbol{Scaling\ 1}}$ and $\mathbf{V_{Can}}$. Analogously, with scaling type 2, canonical biplots can be drawn using either $\hat{\mathbf{F}}_{\mathbf{Can}}$ and $\hat{\mathbf{V}}_{\mathbf{Can}}$, or $\mathbf{Z}_{\boldsymbol{Scaling\ 2}}$ and $\hat{\mathbf{F}}_{\mathbf{Can}}$. One-matrix plots of the non-canonical contribution are built using either $\mathbf{F_{NonCan}}$ or $\hat{\mathbf{F}}_{\mathbf{NonCan}}$. Biplots of the non-canonical contribution can be done using either $\mathbf{F_{NonCan}}$ and $\mathbf{V_{NonCan}}$ for scaling 1 or $\hat{\mathbf{F}}_{\mathbf{NonCan}}$ and $\hat{\mathbf{V}}_{\mathbf{NonCan}}$ for scaling 2.

3.8 Related Multidimensional Scaling (RMDS)

3.8.1 Introduction

Whereas redundancy analysis and canonical correspondence analysis use one matrix as an explanatory data matrix and the other as a response matrix, related multidimensional scaling (RMDS, Cuadras & Fortiana 1998) treats both matrices symmetrically (i.e. they can be interchanged without any effect). RMDS combines the resemblance information contained in two dissimilarity matrices to derive a single Euclidean representation. RMDS carries the advantage of being able to filter out redundant resemblance information from the two dissimilarity matrices. Indeed, this ordination method can be viewed as the symmetric canonical counterpart of classical MDS.

3.8.2 RMDS foundations and method description

Suppose that we have two *n x n* distance matrices $\mathbf{D}_A = [d_A(i, j)]$ and $\mathbf{D}_B = [d_B(i, j)]$, which are defined on the same finite set of objects, paired between them. As in classical MDS, our objective is to construct a joint *n x n* distance matrix $\mathbf{D}_{AB} = [d_{AB}(i, j)]$ which allows us to represent the *n* objects in a single plot, relating the plots that one would obtain applying classical MDS on either $\mathbf{D}_A$ or $\mathbf{D}_B$. The problem of constructing $\mathbf{D}_{AB}$ is similar to that of constructing a joint probability distribution given its marginals. The authors (Cuadras & Fortiana 1998) propose the following properties for d_{AB}, with marginal distances d_A and d_B:

If $d_A = 0$ then $d_{AB} = d_B$; If $d_B = 0$ then $d_{AB} = d_A$. (3.38a)

If $d_A = d_B$ then $d_{AB} = d_A = d_B$. (3.38b)

If the principal coordinates obtained from $\mathbf{D}_A$ and those obtained from $\mathbf{D}_B$ are orthogonal, then $d_{AB}^2 = d_A^2 + d_B^2$. (3.38c)

Knowing how classical MDS works, the RMDS method is simply as follows:

1. Let Δ_A and Δ_B be the inner product matrices associated with $\mathbf{D}_A$ and $\mathbf{D}_B$ respectively. These inner product matrices are obtained applying equations (3.6) and (3.7) to matrices $\mathbf{D}_A$ and $\mathbf{D}_B$.
2. The inner product matrix Δ_{AB} associated with the matrix $\mathbf{D}_{AB}$ of joint distances is given by:

$$\mathbf{\Delta}_{AB} = \mathbf{\Delta}_A + \mathbf{\Delta}_B - \frac{1}{2}(\mathbf{\Delta}_A^{1/2}\mathbf{\Delta}_B^{1/2} + \mathbf{\Delta}_B^{1/2}\mathbf{\Delta}_A^{1/2}) \tag{3.39}$$

where $\mathbf{\Delta}_\alpha^{1/2} = \mathbf{X}_\alpha \cdot \mathbf{\Lambda}_\alpha^{1/2} \cdot \mathbf{X}_\alpha'$, being $\mathbf{X}_\alpha$ the principal coordinates, and $\mathbf{\Lambda}_\alpha$ the diagonal matrix of eigenvalues ($\alpha = A, B$). It can be proved that the matrix $\mathbf{D}_{AB}$ of joint distances, whose corresponding inner product matrix is Δ_{AB}, fulfils the properties of (3.38a-c), provided that $\mathbf{D}_A$ and $\mathbf{D}_B$ have the same *geometric variability* (see 4.4.2), i.e. if:

$$\frac{1}{n^2}\sum_{i,j=1}^{n} d_A(i, j) = \frac{1}{n^2}\sum_{i,j=1}^{n} d_B(i, j) \tag{3.40}$$

Note that this condition can always be assumed to hold, since multiplying one of the marginal distances by an appropriate constant amounts to a change of measurement unit.

3. Finally, the related metric scaling solution $\mathbf{X}_{AB}$ is computed from the spectral decomposition of Δ_{AB}, which gives the matrix of eigenvalues $\mathbf{\Lambda}_{AB}$ and the matrix of eigenvectors $\mathbf{U}_{AB}$:

$$\mathbf{X}_{AB} = \mathbf{U}_{AB}\mathbf{\Lambda}_{AB}^{1/2} \tag{3.41}$$

3.8.3 RMDS in GINKGO

Similarly to RDA and CCA to run RMDS in GINKGO the user must specify which two symmetric dissimilarity items in the Data Editor will be used as original distance matrices $\mathbf{D}_A$ and $\mathbf{D}_B$. However, in this case the ordering of the matrices is not important, since the analysis is symmetric. Two checkbox options of the method can be changed:

Negative EV correction: It may happen that either $\mathbf{D}_A$ or $\mathbf{D}_B$, or both dissimilarity matrices are non-Euclidean. In either case, Δ_{AB} would consequently have some negative eigenvalues. Therefore it is advisable to apply the usual Lingoes (1971) correction (3.9) before RMDS.

Make geometric variances equal: In step 2 of RMDS it is assumed that $\mathbf{D}_A$ and $\mathbf{D}_B$ have the same geometric variability. If this option is selected, a correcting factor is applied to $\mathbf{D}_B$ in order to ensure this property.

The RMDS routine outputs a list of eigenvalues ($\mathbf{\Lambda}_{AB}$) and their associated % of variance (of the joint *AB* space), as well as the cumulative percentages. Output matrices are the matrix of eigenvectors ($\mathbf{U}_{AB}$), the RMDS object principal coordinates ($\mathbf{X}_{AB}$) and the RMDS distances in the full ordination space.

Chapter 4: Classification methods

4.1 Classification concepts. Clustering and Discriminant Analysis

The necessity of establishing classes among a set of elements is motivated by the *a priori* belief that the observed variability of data makes it possible to divide the set into categories which are expected to be useful in further studies. However, the way one should tackle the classification problem will be different depending on the *a priori* knowledge of data.

If there is a subset of data samples whose classification is known *a priori*, we are then allowed to use this subset as a training set to mathematically learn how to classify new samples. This is what is done in **discriminant analysis methods** and related techniques. Those methods use the training data to build some numerical criteria or rules, which are expected to be useful in classifying new observations whose class is unknown. The quality of the rules built depends on the quality and quantity of information about the groups available. The best situation is when one not only knows the classification of the training set, but also knows the multivariate distribution of each group and its parameters. In this ideal case the optimal or Bayes classification rule could be applied. If one knows the theoretical distribution of variables but not their parameters, these can be estimated using sample statistics. Unfortunately, it is very common to ignore even the theoretical distribution of data.

An usual step in ecological analysis is to start with an already known grouping of the objects and try to determine to what extent a set of quantitative descriptors can actually explain this grouping. It may be the result of a cluster analysis computed from a different data set, or reflect a hypothesis to be tested. Thus, apart from being able to classify new observations, discriminant analysis helps to interpret already defined groups. Some reference manuals on discriminant analysis are McLachlan (1992), Mardia et al. (1994) or Duda et al. (2001).

In the worst case, the classification of the training set can also be unknown. Then, there is actually no training set and discriminant analysis cannot be used. Luckily there is a second group of classification methods that can be used: **cluster analysis** (or clustering) **methods**. In contrast to discriminant analysis, the lack of *a priori* knowledge about data causes clustering methods to be very heuristic in their approach to classification. Hartigan (1975) stated in reference to those methods: 'they are not yet an accepted inhabitant of the statistical world'. Reference manuals focusing on cluster analysis are Jain & Dubes (1989), Everitt (1993) or Gordon (1999).

One of the fundamental problems of cluster analysis (and sometimes also of discriminant analysis) is the lack of a precise definition of what a cluster is. This has led to a huge amount of different *ad hoc* clustering methods and algorithms. We will focus here mainly on those based on interobject distances. Beyond this, there are many possibilities. One may, for instance, create clusters by connecting close points, by seeking dense regions of points or by partitioning data seeking to minimize group variances. Example methods for these clustering criteria are found in the next sections.

4.2 Canonical linear discriminant analysis

4.2.1 Introduction

Linear (or canonical) discriminant analysis can be seen as a method of linear modeling, like analysis of variance, multiple linear regression, and canonical correlation analysis. It proceeds in two steps. First, one tests for differences in the explanatory variables (**X**), among the predefined groups (this identical to the overall multivariate analysis of variance tests). If the tests support the alternative hypothesis of significant differences among groups, the analysis proceeds to find the linear combinations (also called *discriminant functions*) of the **X** variables that best discriminate among groups. Linear discriminant analysis is also called *canonical variate analysis* (CVA).

Analysis of variance is often used for screening variables prior to linear discriminant analysis. However, any single variable may not discriminate between groups well although it may have high discriminating power in combination with other variables. Therefore, one should be careful when using univariate analysis to eliminate variables. Methods of variable selection (forward, backward or stepwise selection) are a better way to select sets of variables with discriminating power. However, these variable selection methods do not guarantee that the "best" set of explanatory variables is necessarily going to be found.

Concretely the problem of linear discriminant analysis consists in finding linear combinations of the discriminant descriptors in matrix **X** that maximize the differences **among groups**, while minimizing the variation **within groups**. The explanatory descriptors must be quantitative or binary, since they are combined into a linear function. If necessary, they must have already been transformed to meet the condition of multinormality. The discriminant analysis model is robust to departures from this condition but the statistical tests assume *within-group* normality of each variable.

4.2.2 Algebra and method description

Computations are carried out on the matrices of sums of squares and cross products of centered descriptors. Note that these matrices can be constructed with a classification in the form of a crisp or a fuzzy partition (see subsection 4.7.1 for more information on crisp and fuzzy partitions). Given a fuzzy *c*-partition, the following formulas provide the equations to compute the fuzzy sum of squares and cross products matrices for the total, within-group, and among group parts of scatter. These formulas generalize the crisp case (i.e. where vector **u** is not a continuous variable but an indicator of crisp membership).

Within-group sum of squares product $\mathbf{W}_k = \sum_{i=1}^{n} u_{ij}^m (\mathbf{x}_i - \overline{\mathbf{x}}_k)(\mathbf{x}_i - \overline{\mathbf{x}}_k)' \;\; ; \;\; \mathbf{W} = \sum_{k=1}^{c} \mathbf{W}_k$ (4.1)

Among group fuzzy sum of squares product $\mathbf{A}_k = \left(\sum_{i=1}^{n} u_{ij}^m \right) \cdot (\overline{\mathbf{x}}_k - \overline{\mathbf{x}})(\overline{\mathbf{x}}_k - \overline{\mathbf{x}})' \;\; ; \;\; \mathbf{A} = \sum_{k=1}^{c} \mathbf{A}_k$ (4.2)

Total sum of squares product $\mathbf{T} = \mathbf{A} + \mathbf{W}$ (4.3)

Note that if fuzzy memberships are used and the fuzziness exponent is greater than 1, the sum of **W** and **A** does not equal the usual total sum of squares matrix of data.

The solution to the problem of maximizing the variation among groups while minimizing that within groups calls for the eigenvalues and eigenvectors of a matrix corresponding to the ratio

between the among-group and between group sum of squares and cross products matrices. That is, the characteristic equation is:

$$(\mathbf{W}^{-1}\mathbf{A} - \lambda_k \mathbf{I})\mathbf{u}_k = 0 \tag{4.4}$$

Note that the eigenvectors $\mathbf{u}_k$ will not be orthogonal in the reference space of original variables. Thus, when these non-orthogonal eigenvectors are plotted at right angles, they straighten the reference space and the ellipsoids of within-group scatters of objects. If the eigenvectors are normalized in an appropriate manner, the within-group scatters of objects can be made circular, insofar as the within-group cross-product matrices are homogeneous (same dispersion in all groups). These **normalized coefficients** (eigenvectors) are:

$$\mathbf{C} = \mathbf{U}(\mathbf{U}'\mathbf{V}\mathbf{U})^{-1/2} \tag{4.5}$$

where the normalizing factor is $(\mathbf{U}'\mathbf{V}\mathbf{U})^{1/2}$.

Matrices of normalized (**C**) coefficients can then be used to obtain the **object canonical coordinates**, projecting centered **X** values on the canonical space. For each object:

$$\mathbf{F}_i = (\mathbf{x}_i - \bar{\mathbf{x}})'\mathbf{C} \tag{4.6}$$

Similarly, the group centroids can also be projected to the canonical space, for each *k* group:

$$\mathbf{F}_k = (\bar{\mathbf{x}}_k - \bar{\mathbf{x}})'\ \mathbf{C} \tag{4.7}$$

Note that canonical coordinates could also be computed using matrix **U** of non-normalized eigenvectors.

Canonical coordinates are used to build the classification (or identification) rule (or function) used to classify objects: Each object is assigned to the class whose centroid is closest (in terms of Euclidean Distance) to the object on the canonical space. If the number of groups is two ($c = 2$), this rule corresponds to classifying each object to the closest group in the original space, using as proximity measure the Mahalanobis distance from that object to each one of the group.

4.2.3 Running LDF

Input data and training classes

Linear discriminant analysis can only be executed when an object descriptor matrix item is selected on the Data Editor tree. As usual, this matrix can be cropped in order to eliminate those objects or variables not desired for the analysis. In addition, LDF needs a starting classification (partition) of objects. Users can take a selected column of the original matrix as the classification column. Alternatively, a selected classification matrix from the Analysis Manager can also be used.

Advanced options

LDF execution options in GINKGO 1.4 are the following:

Use all B/W dimensions: Keeps all canonical axes resulting from the eigenvalue decomposition or uses only the first indicated axes.

Normalize eigenvectors: Normalize eigenvectors by the square root of $\mathbf{U}'\mathbf{V}\mathbf{U}$. Leaving this checkbox selected is recommended.

Assume equal group sizes: Among group sum of squares are normally weighted by their group sizes. The user may in some cases force the analysis to regard populations as equally sized.

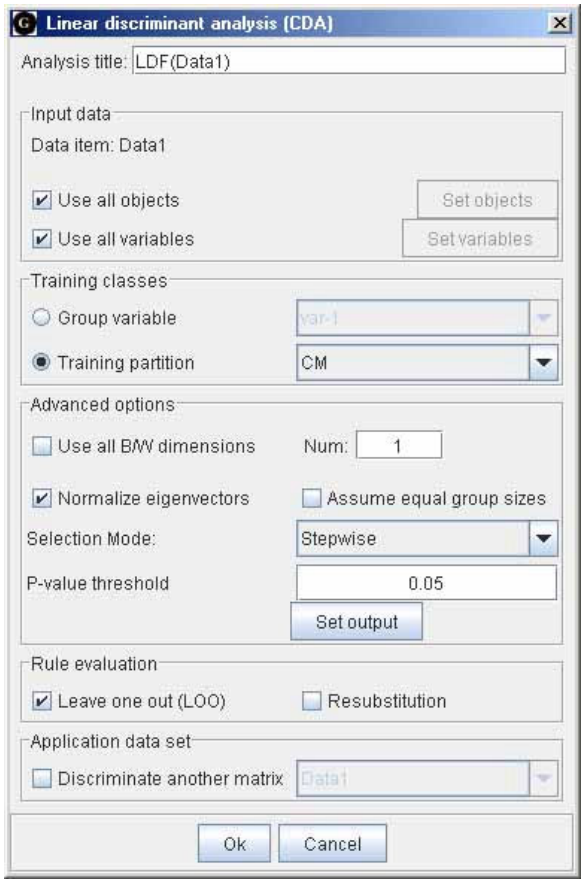
Selection mode: Permits the application of variable selection procedures such as forward, backward or stepwise selection.

P-value threshold: Sets the maximum probability of error for the inclusion of a variable in the LDF model.

Leave one out (LOO): Evaluates the LDF classification rule by classifying each object after extracting it from the model.

Resubstitution: Evaluates the LDF classification rule by reclassifying each object (without extraction).

Discriminate another matrix: Applies the LDF rule to a new set of objects (provided that they have values for the same set of variables).



4.2.4 LDF output

LDF may output a lot of information. To start with, univariate ANOVA test results may be written. If the user requests a variable selection procedure, the different procedure steps are shown in the text output, including the tests done to include or exclude variables.

After selecting the original variables to be included in the LDF analysis, a MANOVA Wilk's test can be performed. Then, the eigenvalues of $\mathbf{W}^{-1}\mathbf{A}$ are presented, as well as their contribution to variance and their associated Wilk's ratio.

The output matrices that GINKGO may output are:

- *Pooled within-group dispersion*: Within-group covariance matrix (pooling within-group sum of squares).
- *Between-group dispersion*: Between-group covariance matrix.
- *Total dispersion*: Overall covariance matrix.
- *Pooled within-group sum of squares*: Pooled within-group sum of squares matrix.
- *Between-group sum of squares*: Between-group sum of squares matrix.
- *Total sum of squares*: Overall sum of squares matrix.
- *Canonical coefficients*: Canonical eigenvectors (normalized if requested) defining canonical axes.
- *Canonical object coordinates*: Object scores on canonical axes.
- *Canonical group centroids*: Centroid coordinates on canonical axes.
- *Canonical centroids interdistances*: Euclidean distances between centroids on canonical space.
- X-F *correlations*: Pearson correlations between canonical coordinates and the original descriptors.

If reclassification of objects was requested by the user, additional classification partitions are presented, obtained from resubstitution or LOO procedures. These classification matrices can be compared with the original classification matrix used to build discriminant analysis afterwards (see section 5.1 Comparing classifications). Percentages of correct reclassification are given in the text output. Finally, an additionally classification matrix will appear if the user asked for the classification of an external data matrix containing new object observations.

The canonical object coordinates matrix can be plotted directly from LDF analysis. In this scatter graphic, groups from either the original classification or reclassifications can be used to group sample points. To display other matrices they must be recopied into the Data Editor (see Graphics chapter for more information).

4.3 Quadratic discriminant

4.3.1 The quadratic discriminant rule

The quadratic discriminant is based on the premise that good classification rules may be obtained by minimizing the probability of erroneous classification. If the probabilistic population model for each population (group) is that of multivariate normal distribution with equal covariance matrices, the rule that minimizes the probability of erroneous classification is that of (canonical) linear discriminant analysis. If, in contrast, the covariance matrices of groups are not equal, the best rule is provided by quadratic discriminant function (QDF). QDF function for object ω_i and for group Ω_k is:

$$QDF(x_i, \Omega_k) = \Delta_k^2(\mathbf{x}_i) + \ln|\Sigma_k| - 2\ln\pi_k \tag{4.8}$$

where π_j is the proportion of group Ω_k and $\Delta_k^2(\mathbf{x}_i)$ is the Mahalanobis distance from object ω_i to group Ω_k :

$$\Delta_k^2(\mathbf{x}_i) = (\mathbf{x}_i - \overline{\mathbf{x}}_j)'\Sigma_j^{-1}(\mathbf{x}_i - \overline{\mathbf{x}}_j) \tag{4.9}$$

Known as QDF, the quadratic discriminant rule can be simply stated as:

"Assign object ω_i to group Ω_k if $QDF(x_i, \Omega_k) = \min_h \{QDF(x_i, \Omega_h)\}$".

4.3.2 Running quadratic discriminant

As in linear discriminant analysis, QDF needs a selected data item to be run. As usual, the analysis dialogs make cropping the input matrix possible, in order to eliminate those objects or variables not desired. QDF again needs a starting classification (i.e. a fuzzy or crisp *c*-partition) of objects. Users can take a selected column of the original matrix as the classification column. Alternatively, a selected classification matrix from the Analysis Manager can also be used.

Rule evaluation can be done by *leave-one-out* (extracting each object before reclassifying it) or by resubstititution (reclassifying the objects used to build the rule without extraction). Also, an additional object descriptor matrix can be used to classify new observations.

Quadratic discriminant output in GINKGO is much simpler than in LDF. No canonical coordinates are produced. Covariance matrices and their determinants are given for each group, because these are necessary steps in the computation of QDF. If reclassification of objects has been requested by the user, additional classification partitions are presented, obtained from resubstitution or LOO procedures. These classification matrices can be compared with the original classification matrix used to build discriminant analysis afterwards (see section 5.1 Comparing classifications). Percentages of correct reclassification are given in the text output. Finally, an additional classification matrix will appear if the user has asked for the classification of an external data matrix containing new object observations.

4.4 Distance-based discriminant analysis

4.4.1 Introduction

Both in linear and quadratic discriminant analysis the input data matrix (**X**) is always a rectangular data matrix. In addition, the assumed theoretical probability distribution of variables in each group is the multivariate normal distribution (with either equal or unequal group variances). In contrast, distance-based (DB) discriminant analysis (Cuadras *et al*. 1997) does not suppose a particular distribution of descriptors. In addition, it needs a symmetric dissimilarity matrix as input, thus allowing the application of many dissimilarity indices to measure object relations in **X**.

4.4.2 Theoretical foundations

Let **X** be a *p*-dimensional random vector, defined on a probability space (Π, A, ***P***) and taking values in $S \subset \Re^P$ with a density probability function *f* with reference to an adequate measure λ. Furthermore, let $d(\cdot,\cdot)$ be a dissimilarity function defined over pairs of elements in Π, so that its square is integrable in *S*. Then, the *geometric variability of* **X** *with respect to* $d(\cdot,\cdot)$ is defined as (Cuadras & Fortiana 1995):

$$V_d(\mathbf{X}) = \frac{1}{2} E\left[d^2(\mathbf{X}_1, \mathbf{X}_2) \,\middle|\, \mathbf{X}_1, \mathbf{X}_2 \in S\right] = \frac{1}{2} \int_{S \times S} d^2(\mathbf{x}_1, \mathbf{x}_2) f(\mathbf{x}_1) f(\mathbf{x}_2) \lambda(d\,\mathbf{x}_1) \lambda(d\,\mathbf{x}_2) \quad (4.10)$$

Given $\mathbf{x}_o \in \Re^P$, we define the *proximity of* $\mathbf{x}_o$ *to the population* Π *with respect to* $d(\cdot,\cdot)$ as (Cuadras *et al*. 1997):

$$\phi_d^2(\mathbf{x}_o, \Pi) = \int_S d^2(\mathbf{x}_o, \mathbf{x}) f(\mathbf{x}) \lambda(d\,\mathbf{x}) - V_d(\mathbf{X}) \quad (4.11)$$

If a representation of $d(\cdot,\cdot)$ exists, i.e. if there is a function $\psi : R^P \to L$ (where $L, \langle\cdot,\cdot\rangle$ symbolizes an Euclidean or Hilbert space filled with a scalar product $\langle\cdot,\cdot\rangle$, such as $\| \mathbf{u} \|^2 = \langle \mathbf{u}, \mathbf{u} \rangle$ is the natural norm for every $\mathbf{u} \in L$), and assuming that $E(\psi(\mathbf{X}))$ and $E(\| \psi(\mathbf{X}) \|^2)$ are finite, then the geometric variability of **X** and the proximity of $\mathbf{x}_o$ to the population Π are:

$$V_d(\mathbf{X}) = E(\| \psi(\mathbf{X}) - E(\psi(\mathbf{X})) \|^2) \quad (4.12a)$$

$$\phi_d^2(\mathbf{x}_o) = \left\| \psi((\mathbf{x}_o) - E(\psi(\mathbf{X})) \right\|^2 \quad (4.12b)$$

4.4.3 The DB rule

The key of the distance-based (DB) approach is to compute the distance from an object to a group from a matrix $\mathbf{D}_{(n\,x\,n)} = [d(i, j)]$ of object interdistances. This approach is possible both for crisp and fuzzy partititions (see 4.7.1). We will first enunciate the DB rule for the crisp case and later give the fuzzy generalization. Let $\mathbf{x}(k)_1, \ldots, \mathbf{x}(k)_{n_k}$ be the n_k vectors of objects belonging to a group Ω_k. The squared distance of a given object ω_i to the group Ω_k is (Cuadras *et al*. 1997):

$$d^2(\mathbf{x}_i, \Omega_k) = \frac{1}{n_k} \sum_{h=1}^{n_k} d^2(\mathbf{x}_i, \mathbf{x}(k)_h) - \hat{V}_d(k) \quad (4.13)$$

where $\hat{V}_d(k)$ is the geometric variability of Ω_k :

$$\hat{V}_d(k) = \frac{1}{2n_k^2} \sum_{h,l}^{n_k} d^2(\mathbf{x}(k)_h, \mathbf{x}(k)_l) \tag{4.14}$$

These equations should be computed for each object vector $\mathbf{x}_i$ (i = 1, 2, …, n) and each Ω_k group (k = 1, 2, …, c) until obtaining a matrix of squared distances from all objects to all group centers. Now the distance-based (DB) rule is stated as follows:

"Assign ω_i to the group Ω_k if $d^2(\mathbf{x}_i, \Omega_k) = \min\{d^2(\mathbf{x}_i, \Omega_1), d^2(\mathbf{x}_i, \Omega_2), ..., d^2(\mathbf{x}_i, \Omega_c)\}$" (4.15)

As stated above, equations (4.13) and (4.14) can be generalized to the case of having a fuzzy partition (see 4.7.1). Given a fuzzy partition matrix $\mathbf{U}_{(n\ x\ c)}$ and a fuzziness exponent m, the distance from an object to a fuzzy group Ω_k and the fuzzy group geometrical variability are:

$$d^2(\mathbf{x}_i, \Omega_k) = \frac{1}{\sum_{h=1}^{n} u_{hk}^m} \sum_{h=1}^{n} u_{hk}^m d^2(\mathbf{x}_i, \mathbf{x}_h) - \hat{V}_{fd}(\Omega_k) \tag{4.16}$$

$$\hat{V}_{fd}(\Omega_k) = \frac{1}{2\left(\sum_{i=0}^{n} u_{ik}^m\right)^2} \sum_{h,l}^{n} u_{hk}^m \cdot u_{lk}^m \cdot d^2(\mathbf{x}_h, \mathbf{x}_l) \tag{4.17}$$

Note that these last equations reduce to equations (4.13) and (4.14) when the classification training matrix is a crisp partition.

4.4.4 Running DB discriminant analysis

Unlike LDF and QDF, DB discriminant analysis needs a selected a symmetric dissimilarity item to be run. As usual, the DB discriminant analysis dialog makes it possible to crop the input matrix in order to eliminate those objects. The analysis also needs a starting classification (partition) of objects. In this case only a selected classification matrix from the Analysis Manager can be used. If the selected classification matrix is fuzzy, then equations (4.16) and (4.17) will be used to obtain squared distances to centroids. Otherwise their crisp counterparts will be used in the DB rule.

As output, DB discriminant analysis yields the geometric variability of each group, computed through (4.14) or (4.17). If reclassification of objects has been requested by the user, additional classification partitions are presented, obtained from resubstitution or LOO procedures. These classification matrices can be compared with the original classification matrix used to build discriminant analysis afterwards (see section 5.1 Comparing classifications). Percentages of correct reclassification are given in the text output.

4.5 Agglomerative Hierarchical Clustering

4.5.1 Introduction

Agglomerative hierarchical methods produce dendrograms as graphical output. These plots show hierarchical relations between objects in the form of a tree with branches of equal length. Any agglomerative hierarchical classification can be fully described by means of a ***cophenetic matrix***. A cophenetic similarity (or distance) of two objects is defined as the similarity (or distance) level at which those objects become members of the same cluster or node. Any dendrogram can be uniquely represented by a matrix in which the similarity (or distance) for a pair of objects is their cophenetic similarity (or distance).

Hierarchical agglomerative methods can be very useful for graphically summarizing the relations contained in a resemblance matrix. However, these clustering algorithms become problematic when dealing with many objects because dendrogram trees become difficult to understand. In addition, at the highest levels of the hierarchy the branches are totally dependent on the lower links and can accumulate errors. Generally speaking, almost all hierarchical agglomerative algorithms are safe when the interest is placed on the small, low levels of the hierarchy, and they become unsafe at high levels. Precisely, the different hierarchical agglomerative algorithms normally differ in the solutions given at those high levels. See Legendre & Legendre (1998) for a detailed explanation on this kind of algorithms. The following subsection briefly introduces those algorithms implemented in GINKGO.

4.5.2 Hierarchical algorithms

Single Linkage (the basic model)

The basic agglomerative model is *Single Linkage* (Sneath, 1957), also called *nearest neighbor*. It can start from a resemblance matrix among objects (distance matrix or similarity matrix). It orders object pairs according to similarity in a descending order, and forms clusters hierarchically starting from the most similar pairs.

In *single linkage*, to join an object to an already existing cluster it must have a similarity at least equal to the considered similarity level with ONE of the objects already included in the cluster. Therefore, an element will be grouped to a cluster at the level corresponding to the maximum similarity between it and the elements in the cluster. Due to its nature, *single linkage* presents a property: what is called ***chaining***. As a result, it may be inadequate in many cases. Chaining means that the objects join a cluster easily if there are intermediate objects that bridge the cluster to other objects. As chaining appears when there are intermediate points, single linkage is a good way to detect them. On the other hand, chaining makes single linkage prone to data noise. The bigger the cluster is, the easiest it is to join more and more elements, because the probability of finding intermediate objects increases. Thus, single linkage is said to ***contract the space*** of relations among objects in the nearby of clusters.

Complete Linkage

In *complete linkage* two clusters fuse depending on the most distant pair of objects among them. In other words, an object joins a cluster when its similarity to ALL the elements in that cluster is equal to or higher than the considered level. In a growing cluster, new objects have little

chance to join, because they must be close to all objects in the cluster, so they join later when the level of similarity considered is lower. Thus, when a cluster grows, it moves apart from other objects. Complete linkage is said to ***expand the space*** in each cluster surrounds. This algorithm may be adequate when one wants to inspect clear discontinuities.

All the following methods are said to ***conserve*** the metric relations of the reference space. They are not object-link methods in the sense of previous methods. Alternatively, they average the similarities at each agglomerative step.

UPGMA -Unweighted arithmetic average clustering

As with the previous methods, at each step the highest similarity identifies the new cluster to be formed. This method then modifies the similarities between the rest of elements (objects or clusters) and the newly formed cluster. UPGMA (Unweighted Pair-Group Method using Averages) computes the similarity between an element and the cluster as the **arithmetic average** of the similarities between the element and all the objects in the cluster. If the element to be compared is also a cluster, the similarity between both clusters is the average. In the two-cluster case, the average is computed among all intercluster pairs. It is unweighted because all original objects receive equal weight in the computations. It should be used in connection with simple random or systematic sampling if the results are to be extrapolated to a bigger reference population.

WPGMA -Weighted arithmetic average clustering

It may occur that the different groups of objects are unequally sampled in the reference populations so their proportions in the sample do not reflect their proportions in the population. Eliminating objects would mean losing valuable information. On the other hand, the presence of big groups which are more similar *a priori* due to their common origin can distort the results obtained with UPGMA when fusing with a small group. Sokal & Michener (1958) proposed an algorithmic solution to this question, called *weighted arithmetic average clustering, WPGMA*. It consists in giving equal weight to the two fusing branches, meaning that the original objects are unequally weighted (i.e. the objects in the larger group receive lower weights). Compared to UPGMA, it increases the separation among groups, since the low similarities in large groups are down-weighted.

UPGMC -Unweighted centroid clustering

A cluster *centroid* is the typical object of the cluster, whether exists or not. In the original space, it is built as the point with coordinates corresponding to the average of the cluster object values. UPGMC (Unweighted Pair-Group Method using centroids) is based on a very simple geometric approach. As usual, the highest similarity points to the next fusion to be made. In the computation of the similarities between the newly formed cluster and the rest of the elements, the members of the cluster are represented by the cluster centroid. This centroid is considered a new object for the rest of the steps. The formula proposed by Gower for computing the similarities between the centroid formed by the elements h and i , with a third element g is:

$$S_{(hi,g)} = \frac{w_h}{w_h + w_i} S_{(h,g)} + \frac{w_i}{w_h + w_i} S_{(i,g)} + \frac{w_h w_i}{(w_h + w_i)^2} (1 - S_{(h,i)}) \tag{4.18}$$

where w_i are the weights given to clusters. In classical centroid clustering, the number of objects in each element is taken as the weight value. At the beginning, the weights are all 1 because each element is just one object. In Gower's formula it is guaranteed that the centroid *hi* lies geometrically on the line between *h* and *i*. Note that the main difference between the average method and the centroid method is that in the former the averaged values are the similarities (or distances) themselves, while in the latter the formula implies similarities but the averaged values are conceptually the original coordinates, so it is a geometric average operation.

UPGMC can be used with any similarity measure, but Gower's formula remains valid only for similarities corresponding to metric distances. The sampling assumptions of this model are the same as UPGMA, because all the original objects are given the same weight. UPGMC can yield ***reversals***.

WPGMC-Weighted centroid clustering

WPGMC is the centroid counterpart of WPGMA. When there are two unequal sized clusters that join, the bigger one will drag the new centroid to itself. If the sampling is not random this may be undesirable. To give equal weight to both branches we must use the following formula:

$$S_{(hi,g)} = \frac{1}{2}(S_{(h,g)} + S_{(i,g)}) + \frac{1}{4}(1 - S_{(h,i)}) \tag{4.19}$$

After each step, the newly formed cluster has a centroid that represents the *geometric center* of the line joining the previous centroids. WPGMC can produce *reversals*.

Ward's method or minimum sum of squares method **(Ward 1963)**

This method is related to the two previous ones in the sense that the two centroids play an important role. Ward's method minimizes, at each step, the sum of quadratic errors. Concretely, it fuses those nodes yielding a lower increase in quadratic error. Note that this objective function is similar to that of K-means (see next section).

β-Flexible method.

Lance & Willams (1967) proposed a generalized hierarchical method that includes all the above methods. Let *h* and *i* be the elements of the newly formed group, and *g* an external object. The distance from *g* to the group (*hi*) is computed using:

$$d_{g(ij)} = \alpha_i \cdot d_{gi} + \alpha_j \cdot d_{gj} + \beta \cdot d_{ij} + \lambda \cdot \left| d_{gi} - d_{gj} \right|. \tag{4.20}$$

Assigning different values to the four model parameters of the model, it is possible to obtain all the hierarchical algorithms described so far. In addition, Lance & Willams (1967) proposed to let one of the parameters, β, vary between –1.0 and 1.0 in order to obtain intermediate solutions between *single linkage* and *complete linkage*. This method was later called *β-flexible clustering*. The reference space is conserved for $\beta \approx -0.25$, value recommended by Milligan (1989).

4.5.3 Running an agglomerative clustering method

Agglomerative clustering methods in GINKGO can be run from either similarity or dissimilarity symmetric matrices. After selecting the desired matrix on the Data Editor the menu option '**Agglomerative hierarchical clustering**' must be selected. A dialog is prompted, and the user must select the agglomerative method that is to be run. Some objects in the initial similarity matrix can be excluded from the computations.

The resulting output panel of an agglomerative method contains two matrices: the input symmetric matrix and the *cophenetic matrix*, which is the matrix representation of the resulting hierarchical classification. Three measures of goodness-of-fit between the clustering results (cophenetic matrix) and the original symmetric matrix are also provided:

- *Pearson r Cophenetic Correlation*: Pearson correlation computed by comparing the matrices' ordered cell pairs.
- *Spearman r Cophenetic Correlation*: Spearman rank correlation computed by comparing the matrices' ordered cell pairs.
- *Gower Distance (Stress1)*: Sum of squared differences between values in the cophenetic matrix and the original symmetric matrix.

4.5.4 Plotting hierarchical methods' results

GINKGO enables the user to build two different plots from a hierarchical clustering method result: the **dendrogram** and the **Shepard diagram**. The first graphically displays the cophenetic relations among objects, while the second compares in a two dimensional scatter graphic the original similarities (or distances) with the cophenetic similarities (or distances) obtained. The diagonal represents the equivalence between both values. In a Shepard plot, the user can check two things. First, a graphical idea of the stress is presented. Second, it can be seen whether the reference space has been conserved, expanded or contracted, depending on whether the points fall near, below or above the diagonal respectively.

4.5.5 Building partitions from dendrograms

Partitions can be obtained by 'cutting' a dendrogram at a given similarity (or distance) level. The number of groups in the resulting partition will depend on how many branches are 'cut' at that level in the dendrogram. By selecting '**Create resemblance level partition**' a partition will be created by cutting the dendrogram at a given level, so the number of groups obtained will depend on the dendrogram. By selecting '**Create a number of groups partition**' the dendrogram will be cut at the level that yields the desired number of groups (noted as *c* in the next section). In both cases, the newly formed partition is added to the *Partition* node of the Analysis Manager tree. In this way, dendrograms can be used to generate classifications to be exported, used in further analysis or compared to other classifications.

4.6 K-means

4.6.1 K-means clustering method

K-means (MacQueen 1967) is a classification algorithm which partitions a set of *n* objects in *c* groups or clusters. The K-means criterion for defining the groups is that they must have a minimum dispersion. The functional to be minimized is the Total Error Sum of Squares (*TESS*):

$$TESS_c = \sum_{k=1}^{c} E_{(k)}^2 = \sum_{k=1}^{c} \sum_{i=1}^{n} I[\omega_i \subset \Omega_k] e_{ik}^2 \tag{4.21}$$

where $E_{(k)}^2$ is the sum of square errors (*ESS*) for group Ω_k, and $I[\omega_i \subset \Omega_k] = 1$ if object ω_i belongs to Ω_k, and $I[\omega_i \subset \Omega_k] = 0$ otherwise. e_{ik}^2 is the square distance from the object ω_i to centroid of Ω_k. In other words, e_{ik}^2 is the scatter of object ω_i with respect to the centroid:

$$e_{ik}^2 = d^2(\omega_i, \Omega_k) = \sum_{j=1}^{p} (x_{ij} - \bar{x}_{(k)j})^2 = (\mathbf{x}_i - \bar{\mathbf{x}}_{(k)})'(\mathbf{x}_i - \bar{\mathbf{x}}_{(k)}) \tag{4.22}$$

Note that, as there is a limited number of possible partitions of objects into groups, one could ideally explore them, to end up keeping the partition with lowest corresponding *TESS*. However the number of possible partitions can be extremely high even for moderate values of *n* or *c*. Therefore, algorithms optimizing *TESS* are needed.

The concrete implementation algorithm of K-means can take several forms, but in GINKGO it is as follows. Starting from an initial partition:

1. Calculate the cluster centers (centroids).
2. Calculate square distances from all objects to all centroids with (4.22).
3. Reassign each object to its nearest cluster.
4. Iterate (1-3) until no more moves occur.

By construction, each K-means iteration decreases *TESS* until a minimum is reached. It is important to note that for each cluster configuration there is a distinct TESS value. There exists a solution space that depends on the centroid positions. Depending on the starting point in this solution space, the algorithm may converge to different final solutions. These solutions may correspond to local minima of the functional. As the starting cluster configuration may be critical, we must be careful when running K-means. Different approaches have been proposed (see starting modes below).

K-means can be executed either on a rectangular object-descriptor matrix (**X**) or a symmetric dissimilarity matrix (**D**). This second mode is possible due to the distance-based (DB) approach (Hathaway et al. 1989). Its rationale is similar to Distance-based discriminant analysis (subsection 4.4.2), i.e. squared distances between objects and cluster centers can be computed from a matrix of object interdistances using equations (4.13) and (4.14). These are substitutes in the algorithm for computing cluster centroids and applying (4.22). The equivalence between running K-means on **X** or on **D** is straightforward in the case of Euclidean distance, but can be proved for other dissimilarities by means of classical multidimensional scaling (MDS).

4.6.2 Running K-means

K-means in GINKGO can be run from either a rectangular data matrix or a symmetric dissimilarity matrix, though it cannot be run from a similarity matrix. This has to be first converted to a distance matrix. As with other analyses, some objects and/or some descriptors can be excluded from computations. The other options in the K-means dialog are:

Starting Mode: As explained in the algorithm section, K-means can be started from different partition configurations. This option is critical, so the user must be aware of the advantages and disadvantages of each starting mode. Five starting modes can be used in GINKGO. A good strategy is to run K-means with the same data several times, using a different starting mode each time, and to finally keep the results of the run which yielded a *lower TESS*. The available starting modes for K-means (and Fuzzy C-means) are given here, as well as some recommendations for their use:

1. RANDOM SEEDS: GINKGO will perform several K-means runs each time choosing different objects as starting cluster centers (seeds). The number of random runs and the number clusters can be changed. When looking for a high number of clusters, the number of runs should also be high in order to explore more completely the space of K-means solutions. This option can be very slow and ineffective when the number of objects or clusters is high but it is safe to run it and then compare its results with other starting modes.
2. FARTHEST POINTS: GINKGO will try to determine the points that lie farther and use them as starting seeds. A single run is made with this starting mode.
3. CHOSEN SEEDS: The user can choose the objects that will be used by K-means as starting cluster centers (seeds). The number of clusters will be the number of seeds chosen. A single K-means run is made with this starting mode.
4. CHOSEN PARTITIONS: The user can choose the partitions, from the ones available in the Analysis Manager, which will be used by K-means as starting partitions. The number of clusters will be the number of groups in each chosen partition.
5. HIERARCHICAL STARTING: This option performs an Agglomerative Hierarchical Clustering before running K-means. The dendrogram coming from the hierarchical algorithm is 'cut' at a level which yields the desired number of clusters to be found and the resulting partition is used as starting partition configuration for K-means. The Agglomerative hierarchical clustering methods available are the same as when running them alone. This is recommended when looking for a **high number of clusters**. When the number of clusters to be found is low compared to the number of objects a RANDOM SEEDS starting mode may be more effective.

Number of Clusters: K-means is a partitioning technique, so the number of groups in the partition must be known in advantage. When starting by RANDOM SEEDS, FARTHEST POINTS or HIERARCHICAL AGGLOMERATIVE CLUSTER, the user can specify the number of clusters to be found.

Number of random runs: This option is only functional when the starting mode is RANDOM SEEDS.

Seeds/partitions: When starting from CHOSEN SEEDS one must specify which objects will be used as starting cluster seeds. Analogously, when running K-means starting from CHOSEN PARTITIONS this option must be used to specify which partitions will be used as starting configurations.

Hierarchic algorithm: This option is only functional when starting K-means from HIERARCHICAL AGGLOMERATIVE CLUSTER. Here the user can choose which hierarchical agglomerative method will be used *prior* to k-means.

Dynamic distance mode: Sometimes the number of objects in a data matrix is so great that the symmetric distance matrix cannot be computed due to the lack of free memory. If one is not interested in the distance matrix itself but wants to run K-means on the space provided by a certain dissimilarity measure, it is possible to run K-means in a distance-based mode, while computing object interdistances when needed. Obviously, the drawback of this dynamic calculation of distances is the increase in computational burden. This option is only available when starting K-means from a rectangular matrix.

LOO inside K-means: Includes a Leave One Out validation mechanism inside K-means. This will not change the K-means best solutions, but can be very useful for increasing the effectiveness of random starting. Leave One Out validation consists of extracting the object from a cluster when evaluating its distance to the cluster center (Oliva et al. 2001).

4.6.3 K-means output

K-means text output will vary depending on the K-means starting mode and whether the **detailed output** option has been selected or not. However, it always includes the following information on the partition found:

- *TESS*: Final total sum of squares error.
- *TESS-LOO*: Final total sum of squares error with *LOO* correction (only shown when *LOO* correction has been requested).
- *Total Var*: Total data variance. This will be always greater than TESS. It is the total dispersion of data points.
- *C-H pseudo F*: Calinski-Harabasz pseudo F statistic (see 5.1.2). This measure may prove to be useful for deciding how many clusters could exist in the data analyzed.

Next, a table describing each group is written. It includes:

- *Group*: group number.
- *Freq*: Number of objects (in float format) included in the group.
- *ESS*: Sum of squares error of the given group.
- *MaxD*: Maximum distance between objects belonging to this group.
- *Near*: Nearest group number.
- *NearD*: Nearest group distance.

To help interpret the resulting groups, if K-means was run using a dissimilarity matrix, or if starting was HIERARCHICAL CLUSTER MODE, K-means outputs a **Silhouette analysis** of these groups (see subsection 5.1.2 for more information). If K-means has been run using a rectangular matrix, the user may ask for univariate ANOVAs in order to test for differences among groups for each of the original variables.

K-means available output matrices are the following:

- *Object memberships*: Displays the output partition of the object set. Objects are in rows and groups in columns. Values are either 1.0 (for an object

belonging to the given group column) or 0.0 (for an object not belonging to the given group column). This matrix can be used in partition comparisons or to group points in plots.

- *Prototypes*: Only available when running K-means from a rectangular object-descriptor matrix. Provides the final centroid coordinates for all groups in the partition. Rectangular matrix with groups in rows and the descriptors in columns.
- *Distance to clusters*: Rectangular matrix with objects in rows and clusters in columns as with membership matrix. Matrix cells are the distances from the objects to the cluster centers.
- *Squared distance to clusters*: Rectangular matrix with objects in rows and clusters in columns as with membership matrix. Matrix cells are the squared distances from the objects to the cluster centers.
- *Inter-cluster distances*: Symmetric distance matrix. Cells are distances between pairs of clusters. This matrix reflects the space relations between cluster centers (centroids).

4.7 Fuzzy C-means

4.7.1 Fuzzy sets, fuzzy logic and fuzzy partitions

Fuzzy set theory (FST) is an extension of the classic set theory developed by Zadeh (1965) as a way to deal with vague concepts, such as "Jordi is tall". Classical (hard or crisp) set theory considers an object as a member of a given set (i.e. the set of tall people) or not. We may express this membership using an indicator variable (*I*), which will take value 1 if the object is member of the set and 0 otherwise. In a fuzzy set, the binary indicator variable is extended to a continuous variable (*u*) named **membership**, which can now take intermediate values in the interval [0,1] (i.e. Jordi can have a value of 0.8 of membership to the set of tall people).

A **fuzzy partition** is a partition of objects into groups or classes allowing intermediate memberships. In a fuzzy partition, each object splits its overall membership of 1 between the different clusters. Formally speaking, matrix $\mathbf{U}_{(n \text{ x } c)} = [u_{ik}]$ is a non-degenerate fuzzy *c*-partition if satisfies:

$$0 \leq u_{ik} \leq 1 \text{ for all } i = 1, ..., n \text{ and } k = 1, ..., c; \tag{4.22a}$$

$$\sum_{i=1}^{c} u_{ik} = 1 \text{ for all } i = 1, ..., n; \tag{4.22b}$$

$$\sum_{i=1}^{n} u_{ik} > 0 \text{ for all } k = 1, ..., c; \tag{4.22c}$$

Note that **hard** or **crisp** ***c*-partitions** (like those of K-means) imply the same conditions excepting (4.22a), which has to be replaced by:

$$u_{ij} \in \{0, 1\} \text{ for } i = 1, ..., n \text{ and } j = 1, ..., c; \tag{4.22a'}$$

4.7.2 Fuzzy C-means (FCM)

Fuzzy C-means (FCM) is an extension of classic K-means using the concepts of fuzzy logic. One of the ways to introduce fuzzy logic in K-means functional is (Bezdek 1981, 1987), (keeping *K* to note the number of clusters):

$$FTESS_{c,m} = \sum_{k=1}^{c} J_{k,m}^2 = \sum_{k=1}^{c} \sum_{i=1}^{n} u_{ik}^m \, e_{ik}^2 \tag{4.23}$$

where u_{ik} is the fuzzy membership of object ω_i to the fuzzy set Ω_k, and $m \in (1, \infty)$ is a fuzziness exponent which determines the incidence of fuzzy values on the computations. Here e_{ik}^2 has the same meaning as in K-means, though the centroid coordinates are now computed as:

$$\overline{x}_{kj} = \frac{\sum_{i=1}^{n} u_{ik}^m \cdot x_{ij}}{\sum_{i=1}^{n} u_{ik}^m} \tag{4.24}$$

From the minimization of the functional equation one can obtain the following expression to compute memberships:

$$u_{ik} = \frac{1}{\sum_{l=1}^{c} \left[\frac{e_{ik}}{e_{il}} \right]^{2/(m-1)}} \tag{4.25}$$

Fuzzy C-means algorithm works in essentially the same way as K-means. We must specify the starting conditions and let the algorithm minimize its functional, provided that centroids are computed using (4.24) and objects are reassigned using (4.25). In addition, FCM needs a more complex stopping rule. This is done here by comparing the fuzzy partitions of two successive iterations. It stops when $\mathbf{U}^t$ and $\mathbf{U}^{t-1}$ differ less than a specified value.

4.7.3 Running Fuzzy C-means

As K-means, FCM can be run from both raw data matrices and symmetric dissimilarity matrices. FCM dialog is mainly the same as K-means (see 4.6.2 for detailed information on input). Only one new parameter, the fuzziness exponent *m*, must be supplied. Higher values of fuzziness will cause FCM to yield fuzzier classification matrices. For normal data an $m = 2.0$ can be used most of the time. With ecological community data there is a high degree of noise, and *m* must be lower in order to increase partition hardness ($m = [1.1, 1.2]$). Leave-one-out validation is not available in FCM.

Obviously, output text and matrices produced in FCM are also similar to those of K-means (see 4.6.3 for detailed information on output). In this case, *object memberships* are fuzzy, so both the membership matrix and the cluster number of objects (fuzzy set cardinality) are real numbers.

Some additional measures of fit are provided when running FCM (these are fully explained in section 5.1):

- *Dunn coefficient*: Measures the "crispness" in a fuzzy partition. Its maximum value is 1.0 (crisp).
- *Normalized Dunn coefficient*: Normalizes the previous measure. It is ranged between 0.0 and 1.0.
- *Partition entropy*: An indicator of the information content.
- *Partition efficiency*: Normalizes partition entropy. It is ranged between 0.0 and 1.0.

4.8 K-medians/Fuzzy C-medians

4.8.1 Centroids and medoids

Both K-means and FCM assume that the best way of representing the prototype of a cluster is by means of its centroid (i.e. the average vector), and that distances from every object to the cluster must be computed through the Euclidean Distance to the centroid. Note that this still holds true when running these clustering methods using dissimilarity matrices. Although the dissimilarity may be non-Euclidean, in the Distance-based approach the centroid is implicitly placed in a position assuming Euclidianity.

It is possible to define K-means/FCM counterparts for the L1 norm (instead of L2). Under this norm, the prototypes are defined as the vector of medians (medoids), and the distances to them are computed using the Manhattan metric (i.e. by sum of differences in absolute value). These counterpart methods are called K-medians (K-med) and Fuzzy C-medians (FCMed). The difference between them is the same as in the case of K-means/FCM. No distance-based approach is available in this case.

4.8.2 Running K-medians/Fuzzy C-medians in GINKGO

K-med/FCMed methods are started only from an object-descriptor matrix. As usual, some objects or descriptors may be excluded from the analysis. The available options are similar to K-means or FCM. Both analyses share the same menu item. Therefore, Fuzzy C-medians must be indicated through a checkbox in the dialog. The output is similar to the previous clustering methods, except for the optimization function, which is now a measure of **Median Absolute Deviations** (MAD), and the output prototypes, which are medoids.

4.9 Possibilistic C-means

4.9.1 The PCM method

Constraint (4.22b) on fuzzy partitions causes FCM to generate fuzzy memberships that can be interpreted as degrees of sharing or relative memberships but not as degrees of true typicality (i.e. the compatibility between the object and the class prototype). The most serious disadvantage of using relative memberships is that the performance of FCM can be inadequate when noise and outliers contaminate the data set. This problem motivated the search for more robust clustering methods (Frigui & Krishnapuram 1996, Davé & Krishnapuram 1997).

Krishnapuram & Keller (1993) cast the clustering problem into the framework of possibility theory. These authors presented the *possibilistic c-means* (PCM) algorithm. In PCM memberships values are absolute and can be interpreted in terms of true cluster typicalities. Thus PCM is a mode-seeking algorithm, i.e., each component generated by PCM corresponds to a dense region in the data set. As iterations proceed, prototypes are successively attracted to dense regions in feature space. In PCM, each cluster is in fact independent of the others. Hence, the objective function corresponding to cluster k can be formulated as (Krishnapuram & Keller 1996):

$$J_k = \sum_{i=1}^{n} (u_{ik})^m e_{ik}^2 + \eta_k \sum_{i=1}^{n} (1 - u_{ik})^m \tag{4.26}$$

where η_k is a suitable positive number, a "bandwidth" parameter called *reference distance*; $\mathbf{u}_k$ is the vector containing all the possibilistic membership values associated with cluster Ω_k. Then $\mathbf{u}_k$ may be a local minimum of J_k only if memberships are computed using:

$$u_{ik} = \frac{1}{1 + \left(\frac{e_{ik}^2}{\eta_k} \right)^{1/(m-1)}} \tag{4.27}$$

As equation (4.27) shows, the membership value of object ω_i in cluster Ω_k depends only on the distance from the feature vector $\mathbf{x}_i$ to prototype $\mathbf{x}_k$ and not on the distances from $\mathbf{x}_i$ to other cluster prototypes, as happens in FCM. Therefore, PCM can be seen as c independent runs of a one-cluster possibilistic algorithm. Interpretation of m is also different in FCM and PCM. In the former, increasing values of m represent increased sharing of points among all clusters, whereas in the latter, increasing values of m represent increased possibility of all points completely belonging to a given cluster. The value of m that gives satisfactory results can be different in the two algorithms.

4.9.2 Two PCM improvements

Correction for bounded dissimilarities

Bounded dissimilarities are those dissimilarity measures with an upper bound (maximum value), regardless of their metric or Euclidean properties. Let d_b be the upper dissimilarity bound of one of such measures. In the case of dissimilarity matrices obtained using a bounded dissimilarity, it is easy to see that a problem arises, concerning the possibilistic membership function when running PCM: no matter what the fuzziness level is, the calculated membership

value for the upper distance bound, $u(d_b)$, will always be non-zero. In other words, the usual possibilistic membership function only asymptotically yields null values for large distances.

In order to solve this problem, one can simply correct (De Cáceres et al. *submitted*) function (4.27) by re-scaling it to the interval [0, $u(d_b)$]. That is, we define a new possibilistic membership function, noted u^*, as:

$$u_{ik}^* = (u_{ik} - u(d_b)_k)/(1 - u(d_b)_k) = u_{ik} \cdot a_{ik} \tag{4.28}$$

where the right product member $a_{ik} = 1 - (e_{ik}^2 / d_b^2)^{1/(m-1)}$ is the membership function's correcting factor. The closer e_{ik}^2 gets to d_b^2, the closer to zero factor a_{ik} will be, and so will u^*. When using equation (4.28) instead of (4.27) in the PCM algorithm, one obtains a new algorithmic variant, which minimizes the following objective function:

$$J_k^* = \sum_{i=1}^{n} (u_{ik}^{*m} \cdot e_{ik}^2 / a_{ik}^m) + \eta_k \sum_{i=1}^{n} (a_{ik} - u_{ik}^*)^m / a_{ik}^m \tag{4.29}$$

Estimating suitable reference distances

Reference distance, η_k, can be interpreted as a "scale" or "bandwidth" parameter. In general, it is desirable for η_k to be related to the overall size of the cluster Ω_k. This parameter also determines the zone of influence of a cluster. The larger η_k is in a cluster, the more mobility it has in PCM, since it can "see" more points. Therefore, overestimating η_k for a "small" cluster located beside a "larger" one increases the probability of missing the small one. Thus, it is obvious that good estimations of this parameter are crucial for the success of PCM.

When the nature of clusters is known, values for η_k may be fixed *a priori*. Krishnapuram & Keller (1993) provide two ways of initializing η_k:

$$\eta_k = K \cdot ((\sum_{i=1}^{n} u_{ik}^m \cdot e_{ik}^2)/(\sum_{i=1}^{n} u_{ik}^m)) \tag{4.30}$$

$$\eta_k = (\sum_{x_i \in (\Pi_k)_\alpha} e_{ik}^2) \Big/ \left|(\Omega_k)_\alpha\right| \tag{4.31}$$

where $(\Omega_k)_\alpha$ is an appropriate α-cut of Ω_k. However, estimating reference distance from FCM and using the above equations can overestimate or underestimate this parameter, due to the inclusion of outliers and inliers in the initial partition. In addition, cluster variance may not provide good η_k estimates when clusters exhibit large deviations from multi-normality, which is expected to be the case in many arbitrary dissimilarity matrices.

What is needed is a function to provide suitable values for η_k, that is, values that yield compact and isolated clusters. A heuristic criterion to provide suitable η_k values would be to search for those values that correspond to local minimum values of the partial derivative of cluster variance (see section 4.4.2) with respect to reference distance, $\partial \hat{V}_{fd}(\Omega_k) / \partial \eta_k$. However, cluster variance increases quadratically and not linearly in an isotropic space. The solution to this is to use cluster "standard deviation" ($Std_{fd}(\Omega_k) = \hat{V}_{fd}(\Omega_k)^{1/2}$) instead of cluster variance. The partial derivative of cluster standard deviation with respect to reference distance is:

$$\partial Std_{fd}(\Omega_k)/\partial\eta_k = (\partial\hat{V}_{fd}(\Omega_k)/\partial\eta_k)/(2\cdot Std_{fd}(\Omega_k))$$

where $\partial\hat{V}_{fd}(\Omega_k)/\partial\eta_k = ((\sum_{i=1}^{n} e_{ik}^2\cdot u_{ik}^m\cdot\alpha_{ik})/(\sum_{i=1}^{n} u_{ik}^m)) - \hat{V}_{fd}(\Omega_k)\cdot((\sum_{i=1}^{n} u_{ik}^m\cdot\alpha_{ik})/(\sum_{i=1}^{n} u_{ik}^m))$

and $\alpha_{ik} = (m/(m-1))\cdot\eta_k^{-1}\cdot u_{ik}\cdot(e_{ik}^2/\eta_k)^{1/(m-1)}$. (4.32)

Finally, reference distance initialization strategy is as follows (De Cáceres et al. *submitted*):

- Starting from a suitable initial membership matrix, calculate for each cluster the usual estimate of reference distance using equation (4.30).
- For each cluster, find the closest reference distance that yields a minimum of (4.32). Avoid the trivial solutions $\eta_k = 0$ (and $\eta_k = d_b^2$, in the case of bounded minima).

4.9.3 Running PCM

Like K-means and FCM, possibilistic C-means can be run from both raw data matrices and symmetric dissimilarity matrices. As usual, undesired objects or descriptors can be excluded from computations. The PCM options available are:

Starting classification: The current implementation of PCM in GINKGO needs a starting classification to be run. For each cluster of the starting classification, the PCM procedure finds a suitable reference distance (see 4.8.2) and runs the PCM iterative algorithm, which is attracted to identify a dense region. When two initial clusters converge to the same final PCM cluster, an insight is given against the statistical validity of one or both initial clusters, since they do not constitute true dense regions. Thus, PCM acts as a sort of validation for the clusters of the starting classification.

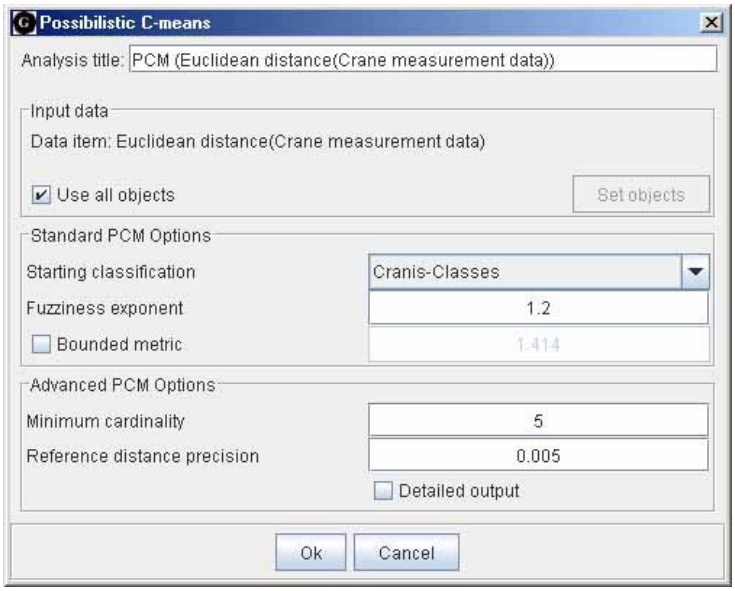
Fuzziness exponent: This has a meaning similar to that found in FCM. Higher values of fuzziness will cause PCM to yield fuzzier classification matrices. Increasing values of *m* represent increases the possibility of all points completely belonging to a given cluster. For normal data an $m = 1.5$ can be used most of the time. With ecological community data there is a high degree of noise, and *m* must usually be very low in order to increase partition hardness ($m = [1.05, 1.1]$). There is, however, an objection against lower values of *m* since they reduce PCM cluster mobility on the space of solutions.

Bounded metric: As explained in subsection (4.8.2) PCM membership function needs a correction when operating on a space of dissimilarities with a distance maximum value. If this is the case with the input matrix, the user should activate this option and indicate the maximum value for dissimilarities.

Minimum cardinality: Minimum cardinality (i.e. the number of objects belonging to the cluster, in fuzzy terms) that a given cluster must have in order to be retained by the PCM procedure. Low cardinality clusters may be considered statistically non-significant as dense regions of space.

Reference distance precision: Reference distance step used by the program in the optimization algorithm that looks for the best reference distance to be used in PCM.

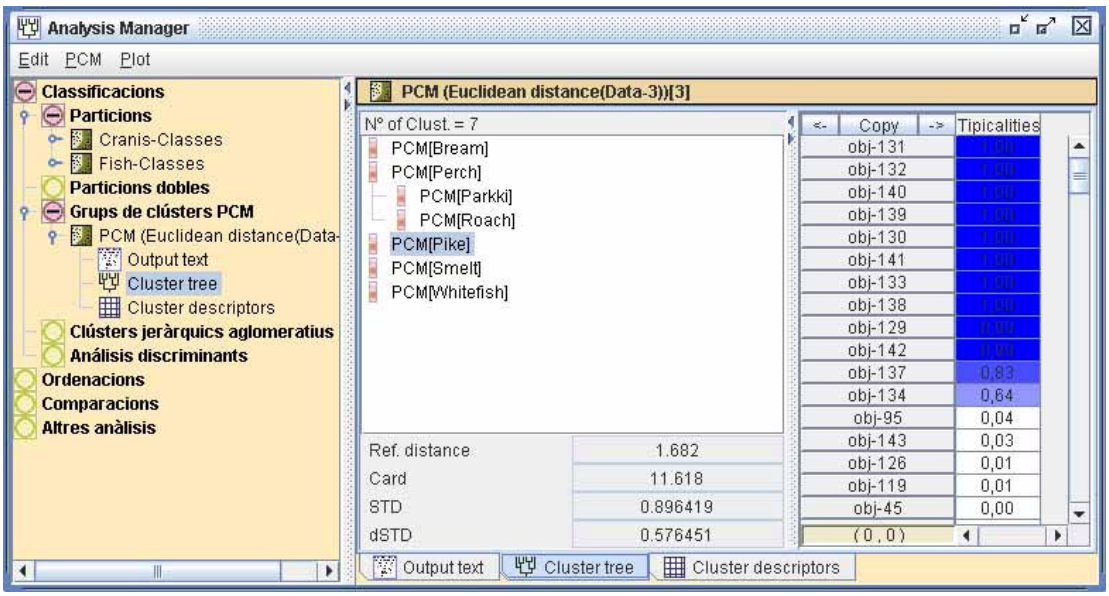
Detailed output: Selecting this option increases the amount of text output produced in PCM.



4.9.4 PCM output

The flexibility of the cluster concept in PCM makes it capable of detecting structures at different clustering levels. However, the same property may be regarded as a drawback of the algorithm, since nested clusters may be produced. Therefore, PCM clusters can be represented in the form of a tree of nested clusters. This is the output form chosen in GINKGO. The PCM output panel has three subpanels:

- *Output text panel*: Panel with all the text output yielded by PCM procedure.
- *Cluster tree panel*: Panel with three parts. On the left part there is a tree of PCM clusters. When selecting a cluster node on the tree, the panel below the tree will show information about the PCM cluster selected (reference distance, cardinality, cluster standard deviation…). At the same time, on the right part the vector of object possibilistic memberships will be shown.
- *Cluster descriptors panel*: Panel showing the information on all PCM clusters, in the form of an object-descriptor matrix.

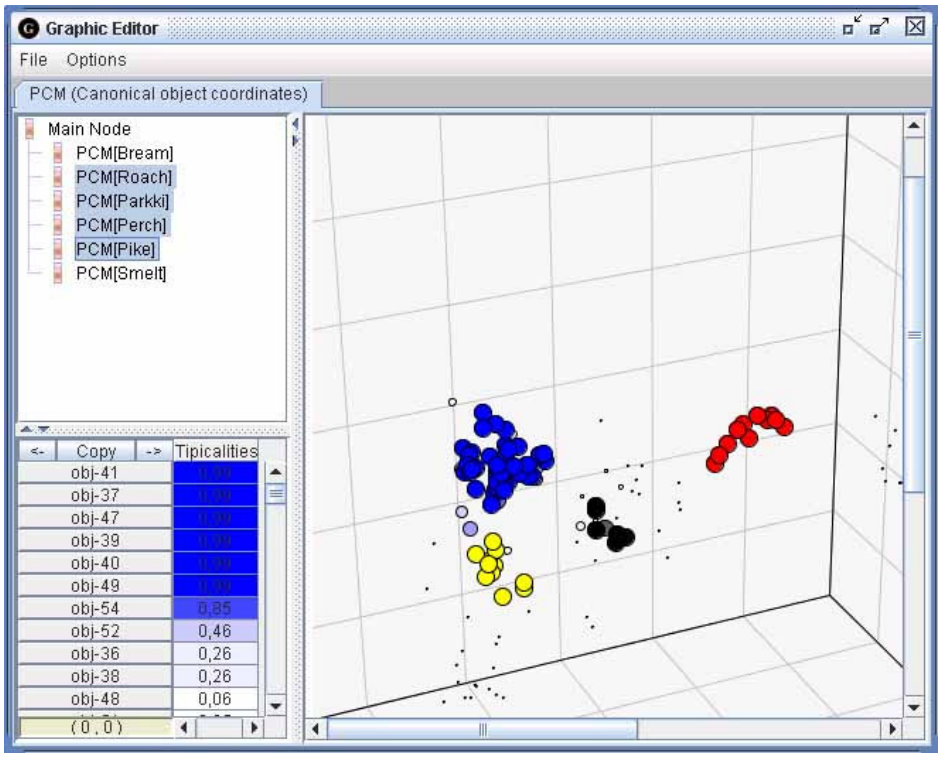


4.9.5 Actions on output and graphics

Some actions are possible after the PCM calculations have been done. The following options are available on the **PCM** menu in the Analysis Manager:

- *(Re)build cluster tree*: Constructs or reconstructs the tree of PCM clusters, analyzing which clusters are nested or overlapped. When some clusters have been deleted, sometimes the cluster tree needs to be rebuilt.
- *(Re)build classification matrix*: Constructs or reconstructs a possibilistic classification matrix merging the different PCM cluster membership vectors.
- *Build fuzzy partition*: Builds a fuzzy partition from the PCM clusters. User must decide: 1) Which combination of clusters, from the ones available in the current cluster tree, will be used to build the partitions. 2) Which level of fuzziness will be used. 3) Whether the ratio between possibilistic memberships or the distance to each cluster center will be used to compute fuzzy membership values.
- *Show intersection matrix*: Builds a symmetric matrix of PCM cluster intersection.
- *Show inclusion matrix*: Builds a square matrix of inclusion between clusters.
- *Show best suited combinations*: Finds which cluster combinations, from the ones available in the current cluster tree, are better suited to partition data.
- *Compare to classification matrix*: Computes Pearson correlation among the membership vectors of the selected classification matrix and the PCM clusters.
- *Delete cluster node*: Deletes the tree selected cluster node.
- *Delete low cardinality nodes*: Deletes clusters having cardinalities lower than a specified threshold value.

GINKGO allows users of PCM to build an interactive plot, through the menu item **fuzzy interactive graphic**. Such a plot is made up of a cluster tree and a scatter diagram. For the latter, the user must indicate which coordinate matrix has to be used as the source for ordination axes. Once built, each tree node selection highlights the corresponding cluster on the scatter diagram. If two or more nodes are selected, the corresponding clusters are indicated in different colors.



4.10 REBLOCK

4.10.1 The method

REBLOCK (Podani & Feoli 1991) is a clustering algorithm that performs a double classification of objects and descriptors. It has been designed in the context of numerical phytosociology, for classifying plant communities and at the same time identifying species groups. In its original conception, it makes it possible to optimize three different functions, but in GINKGO the only function optimized is a χ^2 statistic. The actual aim is to maximize the contingency of blocks (defined by object groups and descriptor groups) through the statistic:

$$\chi^2_{(c_1,c_2)} = \sum_{i=1}^{c_1}\sum_{j=1}^{c_2} \frac{\left(f_{ij} - \dfrac{f_{i\cdot}\cdot f_{\cdot j}}{f_{\cdot\cdot}}\right)^2}{\dfrac{f_{i\cdot}\cdot f_{\cdot j}}{f_{\cdot\cdot}}} \tag{4.33}$$

where c_1 and c_2 are the number of object and descriptor groups, respectively, f_{ij} is the number of presences ($x_{ls} > 0$) in block ij. The statistic measures the divergence of species presence distribution in the blocks from the expected (random) one.

An important drawback of REBLOCK is that it needs the descriptor and object number of groups to be specified *a priori*. Starting from an initial double partition, the algorithm reassigns the variable or object that turns into a larger increase of the functional. As in K-means and related methods, the final solution of REBLOCK depends on the initial double partition used, so several starts are recommended.

To evaluate its results, Podani & Feoli (1991) proposed to use Analysis of Concentrations (AOC, Feoli & Orlóci 1979). In that paper, the $\chi^2_{(c_1,c_2)}$ statistic was used to evaluate the relative sharpness of phytosociological double partitions. Feoli & Orlóci (1979) also proposed the following measure of sharpness:

$$S = \chi^2_{(c_1,c_2)} / \left[f_{\cdot\cdot} \cdot \min(c_1 - 1, c_2 - 1)\right]. \tag{4.34}$$

The greater S was, the better the double partition would be. In our opinion, this measure has a bias because it would be minimum when $c_1=c_2$. In GINKGO, the following modified measure of relative sharpness is given, to be used as a source for evaluations:

$$S = \chi^2_{(c_1,c_2)} \Big/ \left(f_{\cdot\cdot} \cdot \sqrt{(c_1 - 1)\cdot(c_2 - 1)}\right). \tag{4.35}$$

4.10.2 Running REBLOCK

REBLOCK can be started when there is a selected object-descriptor data item in the Data Editor. As usual, objects or descriptors may be excluded. The input matrix is treated as if it were binary (values above 0 are treated as 1). The options that the user may change are:

Object number of groups: The number of object groups to be formed.

Descriptor number of groups: The number of descriptor groups to be formed.

Number of random runs: The number of times REBLOCK will be run starting with a different random double partition.

Detailed output: The user may select this option in order to follow the increase of the functional at each step.

As text output, the REBLOCK procedure yields the chi-square value of each run (4.33). The solution with larger functional is kept. The relative sharpness (4.35) of each solution is also given. Output matrices are the object and descriptor memberships to their respective groups (the two partitions), and the absolute and relative contributions to the chi-square statistic.

Chapter 5 Evaluation and interpretation of classifications

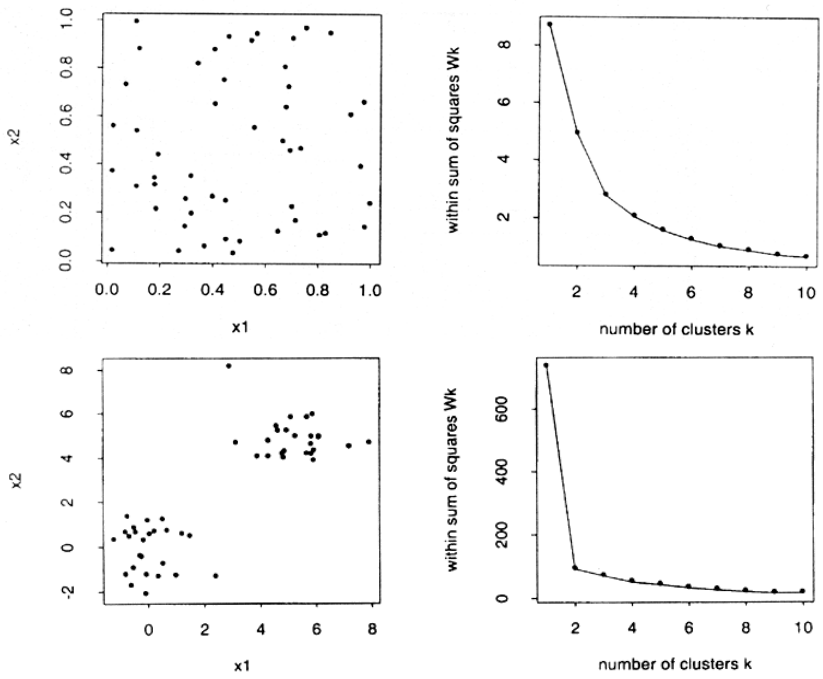
5.1 Evaluation of classifications

5.1.1 Introduction

The validity of a clustering structure can be expressed in terms of three criteria: *internal*, *external* or *relative* (Jain & Dubes 1988). Internal criteria not only assess the fit between the clusters and the original data, but may also be used to determine, for example, the number of groups to be sought. External criteria are used when matching a clustering structure to *a priori* external information (typically another classification structure). Finally, relative criteria determine which of two cluster structures is better in some qualitative or quantitative sense.

5.1.2 Internal criteria of partition evaluation. Determining the number of groups.

One of the annoying aspects of partitioning algorithms is the need to specify *a priori* the number of groups to be found, *c*. This parameter is rarely known before conducting the cluster analysis. A common strategy is to run partitioning methods using different values of *c* and to compare the results by means of suitable internal criteria (i.e. comparing the values of a statistic). It is important to say that the internal criteria used to decide the number of groups contained in a given data item should always be congruent with the clustering method. We will focus here on internal criteria applicable to K-means and FCM results. The figure on the right shows an example from Tibshirani *et al.* (2001), where the evolution of *TESS* (W_k) is shown in a situation of random distribution (above) and a two cluster distribution (below). In both cases, the total sum of squared errors (*TESS*) decreases when *c* increases. The decrease occurs regardless of the existence of cluster structure. However, if a cluster structure exists, the initial decrease is steeper. This sort of "elbow" in the *TESS* plot is what many statistics try to detect (*elbow problem*).



GINKGO incorporates two indices that may be used as validating internal criteria for crisp (and fuzzy) partitions: Calinsky-Harabasz (1974) pseudo-F statistic, and the non-parametric *silhouette* (Rousseuw 1987). The latter can also be used to detect individual object misclassifications.

Pseudo-F

One of the most commonly used criteria for the selection of a number of groups, *c*, is the maximization of a pseudo-F statistic (Calinsky-Harabasz 1974):

$$F(c) = \frac{\mathbf{A}/(c-1)}{\mathbf{W}/(n-c)} \tag{5.1}$$

where **A** and **W** are given by equations (4.2) and (4.1), respectively. This statistic is based on multivariate normal distribution of data. Therefore, it works best with clusters showing this kind of distribution. Departures from normality will tend to lower the performance of this statistic. The higher the value of pseudo-F is, the more variance among groups there will be (compared to within-group variance), and the better the partitition will be. Note that Pseudo-F can be computed using fuzzy partitions.

Silhouettes

Another possibility is to use a non-parametric, geometric, approach. Silhouettes (Rousseuw 1987) try to make a graphical representation of the partition, indicating for each object, whether it is closest to the group it belongs to or is closer to another cluster. For each element, ω_i silhouette s_i, is computed as follows:

1) Let Ω_a be the cluster containing ω_i. We first define a_i as the average dissimilarity between ω_i and the other elements in Ω_a.
2) We next define $d_i(k)$ as the average dissimilarity between ω_i and the elements of Ω_k. We compute for all k=1, ..., c $(k \neq a)$.
3) We select the minimum $b_i = \min_{k \neq a} d_i(k)$. The cluster corresponding to this minimum is the neighbor cluster, Ω_b.
4) s_i is obtained combining a_i and b_i:

$$s_i = \frac{b_i - a_i}{max\{a_i, b_i\}} \tag{5.2}$$

From this definition one can see that $-1 \leq s_i \leq 1$. Values of s_i near zero indicate it is not clear whether object ω_i should be assigned to Ω_a or to Ω_b. Positive values confirm a good classification of the object, whereas negative values are a good indicator of misclassification. The graphical representation of cluster silhouettes is made depicting, for each object, lines whose length is proportional to its s_i value. Arithmetic average of all cluster s_i values is called the **average cluster silhouette**. Finally, averaging s_i for all objects in the data set gives the **average partition silhouette**.

It is important to say that it is possible to extend the measure of silhouette in case of fuzzy sets, by simply redefining a_i and b_i:

$$a_i = \frac{\sum_{j=1}^{n} u_{ja} \cdot d_{ij}}{\sum_{j=1}^{n} u_{ja}} \qquad b_i = \min_{k \neq a} \left[\frac{\sum_{j=1}^{n} u_{jk} \cdot d_{ij}}{\sum_{j=1}^{n} u_{jk}} \right]. \tag{5.3}$$

Fuzziness measures for fuzzy partitions

In parallel to the development of fuzzy partitive clustering algorithms, some partition fuzziness indices have been proposed (Dunn 1976). See Kim *et al*. (2004) for a recent review on the subject. The **partition** or **Dunn coefficient** (Bezdek 1974, 1981), measures the degree of 'hardness' or 'fuzziness' of a partition:

$$PC(c,\mathbf{U}) = \frac{1}{n}\sum_{k=1}^{c}\sum_{i=1}^{n} u_{ik}^2 \tag{5.4}$$

When a partition is completely fuzzy $PC(c,\mathbf{U}) = 1/c$, while for a crisp partition $PC(c,\mathbf{U}) = 1$. A **normalized** version of the **Dunn coefficient** is:

$$PC_{norm}(c,\mathbf{U}) = \frac{PC(c,\mathbf{U}) - (1/c)}{1-(1/c)} = \frac{c\cdot PC(c,\mathbf{U}) - 1}{c-1} \tag{5.5}$$

Another internal measure of fuzziness is **partition entropy** (Ruspini 1969), which measures the information content of the fuzzy partition:

$$H(c,\mathbf{U}) = -\frac{1}{n\cdot \log_a(c)}\sum_{k=1}^{c}\sum_{i=1}^{n} u_{ik}\cdot \log_a(u_{ik}) \tag{5.6}$$

The fuzzier the partition, the more entropy (disorder) will appear. Analogously to the previous coefficient, partition entropy can be also normalized by dividing by its maximum value. This is called **normalized entropy** or **partition efficiency** (Dunn 1976):

$$H_{norm}(c,\mathbf{U}) = \frac{H(c,\mathbf{U})}{(1-c/n)} \tag{5.7}$$

Both the normalized partition coefficient and normalized entropy are statistics that can be used to decide the number of groups in fuzzy partitioning (Dunn 1976, Marsili-Libelli 1989, Equihua 1990). One must choose the *c* value with a highest partition coefficient or lowest partition efficiency. Remember, though, that these coefficients use only the membership values to evaluate partitions, and not the geometrical structure of data.

5.1.3 Comparing classifications

In order to evaluate crisp partitions using external criteria, GINKGO provides the corrected Rand (Rand 1971, Hubert & Arabie 1985) and Fowlkes-Mallows (1983) indices. These indices can assess the level of agreement shown between two crisp partitions. To compare fuzzy partitions using these indices, they must first be "defuzzified". Alternatively, GINKGO boasts a modification of the corrected Rand index, which directly compares fuzzy matrices (*unpublished results*).

Confusion tables

Let $\mathbf{U}_{nxc}$ and $\mathbf{V}_{nxc'}$ be two partitions into c and c' clusters, respectively, of the same data set. There are different approaches to the resemblance between partitions. In a first approach, one builds a **cross-classification table** or confusion **table**, $\mathbf{T}_{cxc'}$, where the two partitions are crossed. Each $t_{kk'}$ element contains the number of objects classified in group k of **U** and in group k' of **V**. The simple inspection of matrix **T** gives a lot of information related to the relation between the two partitions.

Incidence matrices. Rand index

Another approach to measure partition agreement implies the intermediate step of translating each partition matrix into an **incidence matrix** $\mathbf{C}_{nxn}$. In this matrix, each cell c_{ij} has a value 1 if the objects ω_i and ω_j belong to the same cluster, and 0 otherwise. From this starting point, it is possible to study the relation between the two partitions comparing the cells of their respective incidence matrices. One of the most common partition agreement measures is the **Rand index** (Rand 1971). It computes the probability that two randomly chosen objects can be treated in the same manner in the two partitions. Formally, it is defined as:

$$Rand(\mathbf{U},\mathbf{V}) = \frac{\sum_{i}^{n}\sum_{j<i}^{n}\gamma_{ij}}{\binom{n}{2}} \quad \text{, where} \tag{5.8}$$

a. $\gamma_{ij} = 1$ if ω_i and ω_j belong to the same group in both **U** and **V**: $c(\mathbf{U})_{ij} = 1$ and $c(\mathbf{V})_{ij} = 1$.
b. $\gamma_{ij} = 0$ if ω_i and ω_j belong to a different group in **U** and to the same group in **V**: $c(\mathbf{U})_{ij}=0$ and $c(\mathbf{V})_{ij}=1$.
c. $\gamma_{ij} = 0$ if ω_i and ω_j belong to the same group in **U** and to a different group in **V**: $c(\mathbf{U})_{ij}=1$ and $c(\mathbf{V})_{ij}=0$.
d. $\gamma_{ij} = 1$ if ω_i and ω_j belong to a different group in both **U** and **V**: $c(\mathbf{U})_{ij}=0$ and $c(\mathbf{V})_{ij}=0$.

It is rather easy to show that the Rand index can also be computed from the cross-classification table **T**:

$$Rand(\mathbf{T}(\mathbf{U},\mathbf{V})) = \frac{\binom{n}{2} + \sum_{i=1}^{c}\sum_{j}^{c'} t_{ij}^2 - \frac{1}{2}\left(\sum_{i=1}^{c} t_{i\cdot}^2 + \sum_{j=1}^{c'} t_{\cdot j}^2\right)}{\binom{n}{2}}, \text{ where } \binom{x}{2} = x\cdot(x-1)/2\cdot \tag{5.9}$$

Using this last approach, Hubert & Arabie (1985) proposed to correct the Rand index to extract the agreement that would occur only due to chance effects. The **corrected Rand index** is:

$$CorrectedRand(\mathbf{T}(\mathbf{U},\mathbf{V})) = \frac{Rand - Expected(Rand)}{Maximum(Rand) - Expected(Rand)}$$

$$= \frac{\sum_{i=1}^{c}\sum_{j}^{c'}\binom{t_{ij}}{2} - \sum_{i=1}^{c}\binom{t_{i\cdot}}{2}\cdot\sum_{j=1}^{c'}\binom{t_{\cdot j}}{2}\Big/\binom{n}{2}}{\frac{1}{2}\left(\sum_{i=1}^{c}\binom{t_{i\cdot}}{2} + \sum_{j=1}^{c'}\binom{t_{\cdot j}}{2}\right) - \sum_{i=1}^{c}\binom{t_{i\cdot}}{2}\cdot\sum_{j=1}^{c'}\binom{t_{\cdot j}}{2}\Big/\binom{n}{2}} \quad \text{, where } \binom{x}{2} = x\cdot(x-1)/2\cdot \tag{5.10}$$

The corrected Rand index is 0 when the number of coincidences is equal to those expected by chance (null model). In contrast, the value of 1 implies, as in the case of the uncorrected Rand index, a perfect correspondence between the compared partitions. An advantage of the Rand index, whether corrected or not, is that it can be computed even if the number of clusters *c* and *c*' are different.

MINDMT

Another approach to the measure of partition agreement consist in expressing the distance between partitions in terms of the minimum changes needed to transform one partition into the other. This is what the **MINDMT** (*minimum number of divisions, mergences and transfers,* Day 1981, Podani 1986) index does.

Comparing fuzzy partitions

In order to compare fuzzy partitions, the simpler approach is to 'defuzzify' them before applying crisp comparison indices. *Defuzzification* can be done in two basic ways: 1) Creating a crisp partition by choosing, for each object, the group that has a higher membership. 2) Creating a crisp partition from an α-cut, i.e., defining a crisp membership matrix where values are 1 only if the corresponding fuzzy value is above a specified α threshold. As the second method can yield degenerate partitions (not fulfilling condition 4.22b) the most frequently used defuzzification method is the first.

In GINKGO fuzzy partitions may be defuzzified through the menu item **Defuzzify partition** in the Analysis Manager by choosing an α value. However, when several groups have a membership above the threshold non-degeneracy is avoided by choosing the highest membership group. Thus, setting α to 0.0 the first defuzzification method is used.

MINDMT

If one wants to compare two fuzzy partitions without losing the information contained in fuzzy memberships measures of agreement for fuzzy values are needed. In this sense, Podani (1990) proposed to extend *MINDMT* to the fuzzy case. Concretely, all the possible permutations of columns in **U** with respect to **V** are tested to find the permutation that minimizes the distance:

$$d^2(\mathbf{U},\mathbf{V}) = \sum_{i=1}^{n}\sum_{k=1}^{c}(u_{ij} - v_{ij})^2 \tag{5.10}$$

Fuzzy Rand index

The *MINDMT* approach has the drawback of not allowing different numbers of groups in **U** and **V**. In order to compare fuzzy partitions without this limitation we must seek to generalize indices allowing different numbers of groups. This generalization is possible for the Rand index if we express **T** and **C** in matrix notation:

$$\mathbf{T}(\mathbf{U},\mathbf{V}) = \mathbf{U}'\cdot\mathbf{V} \tag{5.11}$$

$$\mathbf{C}(\mathbf{U}) = \mathbf{U}\cdot\mathbf{U}' \tag{5.12}$$

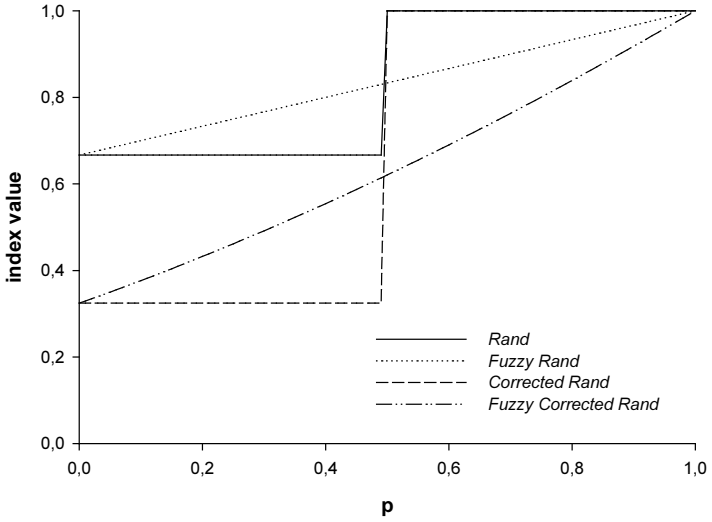
Note that, using matrix notation **U** and **V** can have fuzzy values. Computing **C** as (5.12), one can then compute the four cells (a-d) of the Rand index as:

$$a = \sum_{i=1}^{N}\sum_{j=i+1}^{N} c(\mathbf{U})_{ij} \cdot c(\mathbf{V})_{ij} \qquad b = \sum_{i=1}^{N}\sum_{j=i+1}^{N} \left(1 - c(\mathbf{U})_{ij}\right) \cdot c(\mathbf{V})_{ij}$$
$$c = \sum_{i=1}^{N}\sum_{j=i+1}^{N} c(\mathbf{U})_{ij} \cdot \left(1 - c(\mathbf{V})_{ij}\right) \quad d = \sum_{i=1}^{N}\sum_{j=i+1}^{N} \left(1 - c(\mathbf{U})_{ij}\right) \cdot \left(1 - c(\mathbf{V})_{ij}\right) \tag{5.13}$$

It is easy to show that this definition generalizes the crisp case, and that, in case of non-degenerate partitions, fulfills the equation (a+b+c+d) = (n ·(n - 1)) / 2. Analogously, this generalization is extensible to the corrected Rand index. The figures below show two cases to exemplify the generalization.

Partition U **Partition V**

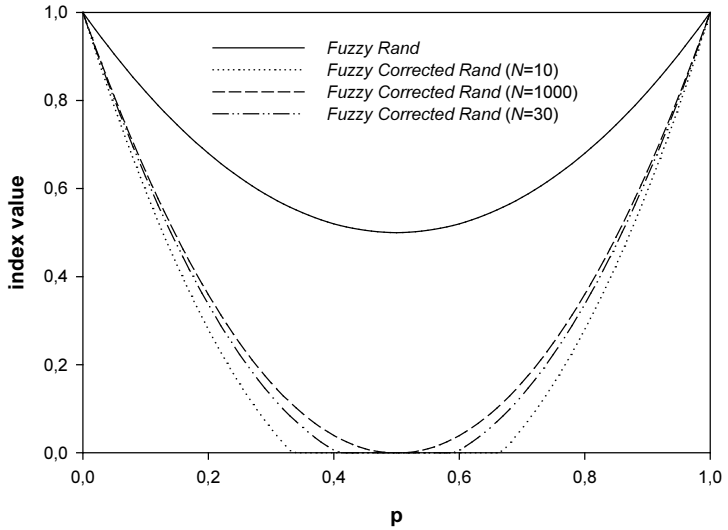
	U_1	U_2		V_1	V_2
ω_1	0	1	ω_1	0	1
ω_2	0	1	ω_2	0	1
ω_3	1	0	ω_3	1	0
ω_4	1	0	ω_4	1	0
ω_5	0	1	ω_5	0	1
ω_6	0	1	ω_6	**1 - p**	**p**



Example 1: Partitions **U** and **V** of dimensions 6x2 are on the left. The only difference between them is the classification of the last object, ω_6, which in **V** depends on the value of parameter **p**. The diagram on the right shows the Rand index values in the cases of combining correction and fuzzy generalization. In the extreme cases (p = 0 and p = 1), the fuzzy and crisp versions are coincident. For intermediate values of **p** the fuzzy membership of ω_6, makes the fuzzy Rand index change gradually. On the other hand, the corrected versions of both indices show, logically, the same pattern, but in lower values.

Partition U **Partition V**

	U_1	U_2		V_1	V_2
ω_1	0	1	ω_1	p	1.0 - p
…	…	…	…	…	…
$\omega_{n/2-1}$	0	1	$\omega_{n/2-1}$	p	1.0 - p
$\omega_{n/2}$	1	0	$\omega_{n/2}$	1.0 - p	p
…	…	…	…	…	…
ω_n	1	0	ω_n	1.0 - p	p



Example 2: Partitions **U** and **V** of dimensions nx2 are on the left. **U** is a two-group partition, each of size n/2. In **V**, the membership to these same groups depends on parameter 'p'. The diagram on the right shows the fuzzy Rand index values. In the extremes (p = 0 and p = 1) the index is 1.0. For the intermediate confusion zone of **V** (p around 0 .5) the uncorrected index is 0.5 and the corrected index is 0.0, independently of the number of objects. This would reflect that the indetermination in **V** corresponds in a crisp sense, to the random assignment of objects.

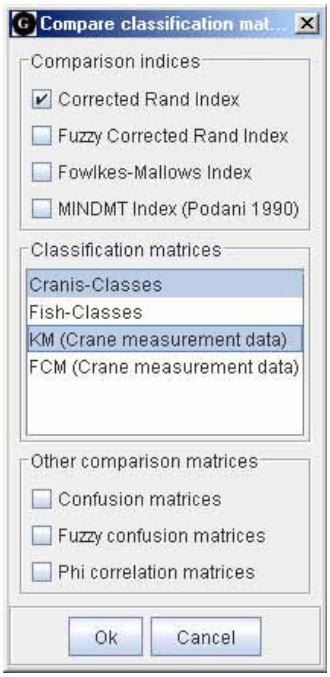
Comparing agglomerative hierarchical trees

Comparisons between hierarchical classifications (i.e. dendrograms) can be made by generating partitions at different cutting levels and then assessing their agreement with any of the aforementioned indices. Moreover, this strategy is applicable to comparisons between dendrograms and partitions.

5.1.4 Comparisons in GINKGO

The GINKGO allows the comparison of classifications through the menu **Compare** available in the main menu bar. First, the program allows the comparison of several classification matrices by means of the Rand index, the fuzzy Rand index, the Fowlkes-Mallows (1983) index or MINDMT. Users can ask for the (fuzzy) confusion matrix of each comparison and the phi (i.e. Pearson) correlation between clusters. Second, hierarchical agglomerative trees can be also compared using the same indices. In this case the index values are computed for each number of groups from k=2 to k=n. The resulting vectors are included in the output and are shown as index profiles in the Graphic Editor window.

In order to facilitate validation by means of external criteria, external classification matrices can be imported into the Analysis' Manager by reading rectangular ASCII text files. Remember that classification matrices can be created from data matrices in the Data Editor (see subsection 2.1.6). In addition, GINKGO is capable of reading TWINSPAN's double partitions (i.e. one partition of relevés and one of species) as generated by the program PC-ORD (McCune & Mefford 1999).



5.2 Further interpretation of clusters

5.2.1 Analyzing cluster properties on a data matrix

Once a classification has been obtained it is possible to check its agreement with a new data set, different from the one used to obtain the classification. This *a posteriori* analysis of clusters can be done in GINKGO through the menu item **Describe cluster in data** from the menu **Group info**. Note that the new data set may be either a rectangular object-descriptor (O) matrix or a symmetric distance (D) matrix. As with other analyses, some objects or descriptors may be excluded from computations. The user must also specify which classification will be used as source for clusters. The prompted dialog shows many other options, available depending on the input matrix used:

- *Variance description* (O/D): Gives the error sum of squares (ESS) of each cluster, as well as its variance and percentage of the total error sum of squares (TESS).
- *Centroid coordinates* (O): Gives the vector of average values for each cluster (a weighted average in the case of fuzzy partitions).
- *Euclidean Distances to Centroids* (O/D): Gives the matrix of (Euclidean) distances between objects and centroids.
- *Medoid coordinates* (O): Gives the vector of median values for each cluster (a fuzzy median in the case of fuzzy partitions).
- *Manhattan Distances to Medoids* (O): Gives the matrix of Manhattan metric distances between objects and medoids.
- *Cluster form* (O/D): Different statistics of cluster sphericity are given (in the following M may be an inner product matrix, for D calculus, or a covariance matrix, for S calculus):
  - det(M) - determinant (squared volume) of M.
  - det(M)^(1/p) - Expected mean trace per dimension of M.
  - tr(M) - trace of M.
  - tr(M)/p - mean trace per dimension of M.
  - [tr(M)/p]^p - expected determinant (squared volume) under sphericity.
  - HyperVolume Ratio = det(M)/([tr(M)/p]^p) - measures the departure from spherical condition as a ratio in p dimensions.
  - Variance Ratio = [det(M)^(1/p)]/[tr(M)/p] - measures the departure from spherical condition as a ratio in one dimension.
- *Isolation statistics* (D): Gives a matrix of cluster isolation statistics computed from each cluster vector on distance to centroid matrix: Average, median, standard deviation, skewness, mean cluster silhouette, ...
- *Intergroup distances* (O/D): Gives a matrix of (Euclidean) distances between cluster centroids.
- *Intergroup adjacency* (O/D): Gives a matrix of adjacency between clusters. Finds the nearest group for each object, excluding the actual group. The same matrix is given in group percentages.
- *Sum of squares group matrices* (O): Gives the sum of squares matrix for each group.
- *Covariance group matrices* (O): Gives the covariance matrix for each group.
- *Correlation group matrices* (O): Gives the correlation matrix for each group.
- *Univariate ANOVAs* (O): Gives a matrix with univariate ANOVA computations for each descriptor (including sum of squares, degrees of freedom, mean squares, F statistics and p-values).

5.2.2 Measurement of ecological diversity

The measurement of ecological diversity is an object of study in several reference manuals (see for instance, Margalef 1974, Magurran 1989 o Legendre & Legendre 1998). We will briefly introduce here some statistics related to diversity. There are two main factors in the measurement of ecological diversity: the number of species (S) and their abundances (n_1, n_2,..., n_S). The entropy or Shannon-Wiener (Shannon & Weaver 1949) diversity index, is one of the commonest ways to measure the biological diversity in a sample:

$$H = -\sum_{j=1}^{S} p_j \log(p_j) \tag{5.14}$$

where p_j is the probability of symbol (i.e. species) j. In practice, probabilities are estimates using relative frequencies in the sample. Taylor (1978) states that if H is obtained for several samples, the values tend to be distributed according to normality. This property makes possible the use of parametric statistics (Magurran 1989). Shannon's entropy is a specific case of Rényi generalized entropy (Rényi 1961):

$$H_a = \frac{1}{1-a} \log \sum_{j=1}^{S} p_j^a \tag{5.15}$$

For each entropy of order a there is a corresponding value of specific richness (*diversity number, DN*) of the same order:

Order (a)	Entropy (H_a)	*Diversity number* (DN_a)
0	$H_0 = \log(S)$	$DN_0 = S$
1	$H_1 = -\sum_{j=1}^{S} p_j \cdot \log(p_j) = H$ (Shannon's H)	$DN_1 = \exp H$
2	$H_2 = -\log \sum_{j=1}^{S} p_j^2$	$DN_2 = \left(\sum_{j=1}^{S} p_j^2 \right)^{-1}$ (Concentration^{-1})

Increasing the order (a) lessens the influence of rare species as compared to the dominant ones. Therefore, dependence of diversity measures with respect to the number of species is higher for low values of a.

If one calculates the division between the observed H_1 value and the value it would have if all species were equiprobable, $H_{max} = log(S) = H_0$, one obtains a measure of **evenness** or **equitability** in the proportion of symbols (Pielou 1966):

$$J = H / H_{\max} = H_1 / H_0 \tag{5.16}$$

Pielou's evenness is too dependent on the sampling effort, since it includes H_0. To overcome this limitation, Hill (1973) proposed a family of uniformity measures:

$$E_{a,b} = DN_a / DN_b, \; a > b. \tag{5.17}$$

When diversity is measured on the species abundance values sampled on a specific site, it is a measure of **alpha** (α) **diversity** of the community. Studying diversity is more difficult when one

wants to pool different samples belonging to a given group or cluster. The simplest option is to average the alpha diversity values of all group site members. Alternatively, it is possible to compute diversity measures taking as sample the group pooled data. When the unit of measurement is a geographical region, one can speak of **gamma** (γ) or **regional diversity**. However, if the group does not have a geographical sense, this nomenclature is erroneous, and it can simply be called **group diversity**.

To sum up, it is important to distinguish: (1) site alpha diversity values; (2) the average diversity values for a group; and (3) the group diversity computed taking group pooled data as sampling unit.

5.2.3 Computing ecological diversity statistics

GINKGO makes it possible to compute several diversity statistics using the menu item **Cluster species diversity** from the menu **Group info**. Note that the data set has to be a rectangular object-descriptor matrix. As with other analyses, some objects or descriptors may be excluded from computations. The user must also specify which classification (partition) will be used as source for clusters.

By default, the diversity procedure performs analysis of variance on three parameters: species richness (*S*), site total abundance and Shannon's entropy (*H*). It first outputs a matrix with the following statistics computed for each group or cluster:

- *N°sp AVERAGE*: Average site species richness within sites of the same group.
- *N°sp SS* : Sum of squares of site species richness within sites of the same group.
- *N°sp STDDEV* : Standard deviation of site species richness within sites of the same group.
- *N°sp CV* : Variation coefficient of site species richness within sites of the same group.
- *Abundance AVERAGE*: Average total site abundance within sites of the same group.
- *Abundance SS* : Sum of squares of total site abundance within sites of the same group.
- *Abundance STDDEV* : Standard deviation of total site abundance within sites of the same group.
- *Abundance CV* : Variation coefficient of total site abundance within sites of the same group.
- *H-Div AVERAGE*: Average site Shannon's H diversity within sites of the same group.
- *H-Div SS* : Sum of squares of site Shannon's H diversity within sites of the same group.
- *H-Div STDDEV* : Standard deviation of site Shannon's H diversity within sites of the same group.
- *H-Div CV* : Variation coefficient of site Shannon's H diversity within sites of the same group.

The ANOVAs results are presented in the output text. Four additional analyses can be performed:

- *Alpha (site) diversity values*: Outputs diversity statistic values computed at each site (object), in form of a matrix with the following descriptors:
  - *N0[N°sp]* : Number of site species. Diversity number of order zero.
  - *H0* : Entropy of order zero = e^(N0).
  - *N1[exp H1]* : Diversity number of order one = e^(H1).
  - *H1[Shannon entropy]*: Entropy of order one. Shannon's entropy measure.
  - *N2[1/Concentration]* : Diversity number of order two=Inverse of species concentration.
  - *H2[-log(Concentration)]*: Entropy of order two = - log(species concentration).
  - *Pielou's J[H1/H0]*: Pielou's measure of Evenness.
  - *N1/N0*: Measure of evenness. Ratio between N1 and N0.
  - *H2/H1* : Measure of evenness. Ratio between H2 and H1.
  - *N2/N1*: Measure of evenness. Ratio between N2 and N1.
  - *VarH1*: Measure of evenness. Variance of site Shannon entropy values.
- *Gamma (group) diversity values*: Group diversity statistics computed taking group pooled data as sampling unit. Pooling can be done by counting species occurrences

(constancy) or summing up abundances (total abundance). The output matrix has the following descriptors:

- *N(k)* : Number of sites in group k.
- *N0(k)* : Number of species in group k.
- *H0(k)* : Entropy of order zero = e^(N0(k)).
- *H1(k)(Constancies)*: Entropy of order one. Shannon's entropy measure, computed using species frequency in data.
- *N2(k)(Constancies)*: Diversity number of order two=Inverse of species concentration, computed using species frequency in data.
- *H2(k)(Constancies)*: Entropy of order two = - log(species concentration), computed using species frequency in data.
- *J=H1/H0(k)(Constancies)*: Pielou's measure of Evenness, computed using species frequency in data.
- *N1/N0(k)(Constancies)*: Measure of evenness. Ratio between N1 and N0, computed using species frequency in data.
- *H2/H1(k)(Constancies)*: Measure of evenness. Ratio between H2 and H1, computed using species frequency in data.
- *N2/N1(k)(Constancies)*: Measure of evenness. Ratio between N2 and N1, computed using species frequency in data.
- *H1(k)(Tot.Ab.)*: Entropy of order one. Shannon's entropy measure, computed using total abundance of species in data.
- *N2(k)(Tot.Ab.)*: Diversity number of order two=Inverse of species concentration, computed using total abundance of species in data.
- *H2(k)(Tot.Ab.)*: Entropy of order two = - log(species concentration), computed using total abundance of species in data.
- *J=H1/H0(k)(Tot.Ab.)*: Pielou's measure of Evenness, computed using total abundance of species in data.
- *N1/N0(k)(Tot.Ab.)*: Measure of evenness. Ratio between N1 and N0, computed using total abundance of species in data.
- *H2/H1(k)(Tot.Ab.)*: Measure of evenness. Ratio between H2 and H1, computed using total abundance of species in data.
- *N2/N1(k)(Tot.Ab.)*: Measure of evenness. Ratio between N2 and N1, computed using total abundance of species in data.
- *VarH1(Constancies)*: Measure of evenness. Variance of site Shannon entropy values, computed using species frequency in data.
- *VarH1(Tot.Ab.)*: Measure of evenness. Variance of site Shannon entropy values, computed using total abundance of species in data.

- *Constancy analysis*: Computes species (variable) constancy values for each group of sites (objects), as well as the absolute and relative frequency of constancy classes (five classes) for each group.
- *Mean abundance analysis*: Computes species (variable) average abundance values for each group of sites (objects), as well as the absolute and relative frequency of mean abundance classes (five classes) for each group.

5.2.4 Computing species fidelities and diagnostic species

Diagnostic species are those species showing a preference for a given vegetation unit. As a consequence, they are useful for pointing to a specific vegetation type among others, thus resembling the concept of indicator species. The degree of preference shown by a species is called 'fidelity' in the European approach to phytosociology. Fidelity was initially defined as the degree of preference of a species for a given phytosociological association but the concept can be extended to mean the degree of preference of a taxon for a given syntaxon.

GINKGO focuses on fidelity measurements suitable for categorical data. In vegetation data sets, species presences/absences give more robust fidelity estimations than covers/abundances, as they are less affected by temporal fluctuations and observer bias (Chytrý et al. 2002). Generally speaking, the degree of fidelity shown by a taxon is mainly assessed by regarding the difference in the degree of presence of a species inside and outside the vegetation unit. We can refer to this difference in frequency as *selectivity*. Besides its influence on selectivity, the degree of presence plays a role by itself in fidelity assessment: A rare species may indeed be very selective but, if its occurrence is very low one may survey the diagnosed vegetation type but fail to sample it, thus losing its indicator information. Therefore, constant selective species are far more useful as indicators for the diagnosed vegetation unit. Whereas, fidelity degrees are calculated on the basis of degree of presence and selectivity, if we want to assess how reliable a particular fidelity degree is we must compute its statistical significance. Let A be a sintaxon or vegetation unit for which taxon diagnostic value is to be assessed. Then we define:

- N = number of relevés in the data set.
- N_A = number of relevés belonging to A.
- n = number of taxon occurrences in the data set.
- n_A = number of taxon occurrences in relevés belonging to A.

With known values for these parameters, several fidelity statistics may be computed. The following are available in GINKGO:

1. *Hypergeometric u-value* (Bruelheide 2000):

$$u_{hyp} = \frac{n_A - N_A \cdot (n/N)}{\sqrt{n \cdot N_A \cdot (N-n) \cdot (N-N_A)/(N^2 \cdot (N-1))}} = \frac{N \cdot n_A - n \cdot N_A}{\sqrt{n \cdot N_A \cdot (N-n) \cdot (N-N_A)/(N-1)}} \quad (5.18)$$

2. *Binomial u-value* (Bruelheide 2000):

$$u_{bin} = \frac{n_A - N_A \cdot (n/N)}{\sqrt{n \cdot (N_A/N) \cdot (1-N_A/N)}} = \frac{N \cdot n_A - n \cdot N_A}{\sqrt{n \cdot N_A \cdot (N-N_A)}} = u_{hyp} \cdot \sqrt{(N-n)/(N-1)} \quad (5.19)$$

3. *Chi-square* (χ^2, Sokal & Rohlf 1995: 736): From the cells of a contingency table 2x2 one can build:

$$\chi^2 = \frac{N(ad-bc)^2}{(a+b)(c+d)(a+c)(b+d)} \quad (5.20)$$

where: $a = n_A$, $b = n - n_A$, $c = N_A - n_A$, and $d = (N-n) - (N_A - n_A)$. By substitution, χ^2 turns out to be:

$$\chi^2 = \frac{N \cdot (N \cdot n_A - n \cdot N_A)^2}{n \cdot N_A \cdot (N-n) \cdot (N-N_A)} = u_{hyp}^2 \cdot N/(N-1) = \Phi^2 \cdot N \quad (5.21)$$

4. *Phi coefficient* (Φ, Sokal & Rohlf 1995:741-743):

$$\Phi = \frac{n_A - N_A \cdot (n/N)}{\sqrt{n \cdot N_A \cdot (N-n) \cdot (N-N_A)/(N^2)}} = \frac{N \cdot n_A - n \cdot N_A}{\sqrt{n \cdot N_A \cdot (N-n) \cdot (N-N_A)}} = u_{hyp} \Big/ \sqrt{(N-1)}, \quad |\Phi| = \sqrt{\frac{\chi^2}{N}} \quad (5.22)$$

5. *Binary indicator value* (Dufrêne & Legendre 1997; Chytrý et al. 2002):

$$IndVal_{bin} = \frac{n_A(N - N_A)}{\sqrt{n \cdot N_A - 2n_A N_A + n_A N}} \cdot \frac{n_A}{N_A} \tag{5.19}$$

6. *Binary frequence fidelity*:

$$FreqFid_{bin} = \frac{2n_A - N}{N} \cdot \sqrt{\frac{n_A}{N_A}} \tag{5.20}$$

7. *Binary Specificity Value*:

$$SpecVal_{bin} = \frac{n_A(N - N_A)}{n \cdot N_A - 2n_A N_A + n_A N} \tag{5.21}$$

The menu item **Cluster species fidelity** from the menu **Group info** makes it possible to compute these indices on the data item currently selected in the Data Editor. Note that the data set has to be a rectangular object-descriptor matrix. As with other analyses, some objects or descriptors may be excluded from computations. The user must also specify which classification (partition) will be used as source for clusters. The other options of this procedure are:

- *Fidelity index*: Statistic to be computed among those presented above. A matrix of dimensions (*p* x c) (number of species by number of groups) will be the output of fidelity values.
- *List ordered fidelities*: Checkbox that permits listing, for each group, of the fidelity values going in order from the more diagnostic (highest value) to the least diagnostic (lowest value).
- *Generate fidelity profiles*: Makes possible computing of fidelity profiles. These are obtained, starting from the sites belonging to the cluster, by adding one site at each step to the reference set of sites used to compute fidelity.
- *Community ordering distance*: Distance measure used to order the external sites before starting the adding sequence in the profile generation process.

Chapter 6: Graphics

6.1 Introduction to Graphics in GINKGO

6.1.1 Overview

The results of ordination methods can be displayed in 2D or 3D scatter diagrams, including ordination biplots and Shepard diagrams (Shepard 1962). In addition, any available partition in the Analyses Manager, whether fuzzy or crisp, may be used to label objects in scatter diagrams. It is therefore possible to interpret data structure by combining the patterns revealed using both classification and ordination methods.

The generated graphics are displayed in GINKGO's Graphics Editor, where each plot is displayed in a different tab. Graphics Editor makes it possible to edit some plot properties, export image files, and send print jobs.

6.1.2 Graphic types

The following graphics are available in GINKGO:

- *Scatter graphics*: Scatter graphics that display points and arrows in a 2D or 3D space using object (and sometimes descriptor) coordinates as input. The user can decide to show labels or cluster results on the display.
- *Connected scatter graphics*: Scatter graphics that display connected points in a 2D or 3D space, using object coordinates as input. The user can decide to show labels onto the display.
- *2D series graphics*: Graphics that display ordered series of values on a 2D plot using object-descriptor coordinates as input. Object labels are used in the abscises' axis and each descriptor vector is used to display a different series of ordinate values.
- *Bar graphics*: Graphics that display one or two series of values in a 2D plot. These are used in GINKGO only to represent variable frequency histograms.
- *Shepard diagrams*: 2D scatter graphic where coordinates are given by values in the same cell of two symmetric matrices.
- *Dendrograms*: Hierarchical plot of cluster relations available only from the output of hierarchical agglomerative clustering methods.
- *Scree plot*: Graphic that displays the ordered series of decreasing eigenvalues coming from the eigenanalysis in many ordination methods.

6.2 Creating graphics

6.2.1 Creating 2D and 3D scatter plots

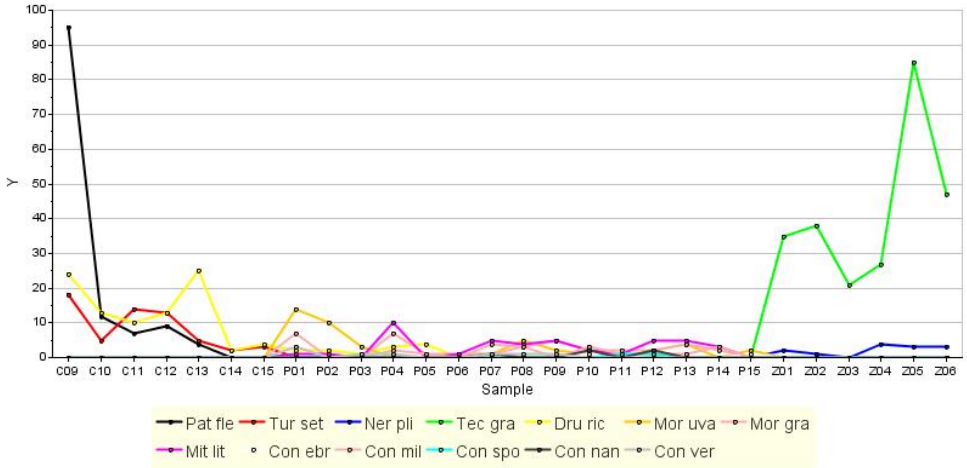
Scatter plots can be created in two ways: 1) From raw data (in the Data Editor), or 2) From the results of a multivariate analysis method yielding object coordinates (in the Analysis Manager). In the first case, the user must select the data matrix where the point coordinates are before selecting **Scatter graphic** in the **Draw** menu. In the case of an analysis, menu items enabling the possible scatter plots will appear on the menu bar of the Analysis Manager. The same dialog prompts in both cases. The following options are available:

- *Title*: The title of the graphic. It will be used to label the corresponding tab in the Graphic Editor window.
- *First axis*: The descriptor (variable) to be used as first axis in the scatter plot.
- *Second axis*: The descriptor (variable) to be used as second axis in the scatter plot.
- *3D plot*: Checkbox to enable 3D plots.
- *Third axis*: The descriptor (variable) to be used as third axis in the scatter plot.
- *Show labels*: Checkbox to enable point labels.
- *Proportional X-Y-Z Axes*: Checkbox to make all scatter axes bounded in the same range.
- *Apply classification matrix*: Checkbox to enable grouping points following the information given by a classification matrix (if any) from the Analysis Manager.
- *Apply fuzzy set*: Checkbox to enable expression of the membership degree shown by the represented objects to a given fuzzy set as a symbol of proportional size and color intensity.

6.2.2 Creating other plots

Connected scatter plots are only drawn by choosing an object-descriptor data item in the Data Editor, and then selecting the menu item **Connected scatter graphic** from the **Draw** menu. In this case, data is expected to be adequately ordered; otherwise the connections drawn will be meaningless. The prompted dialog is similar to that of (unconnected) scatter plots, except for the fact that information from classification matrices or fuzzy sets cannot be represented on the plot.

In order to draw a 2D series graphic an object-descriptor data item must be selected in the Data Editor. The options in this case include specifying the graphic title, as well as the titles for the X and Y-axis, and the variables in the data matrix to be included as series of values.



Bar graphics are created from the **Variable Analysis** menu of the Data Editor menu bar (see subsection 2.2.2).

Shepard diagrams can be built from the Data Editor only when there are at least two symmetric data matrices available in the project. When the user selects the **Shepard diagram** menu item from the **Draw** menu, the prompted dialog asks for the two symmetric matrices to be used as source for coordinates. Shepard diagrams can also be built from the output of some multivariate analysis, such as classical MDS and NMDS ordination, and from a hierarchical agglomerative clustering. In those cases, the menu item can be found on the Analysis Manager menu bar, and no dialog is prompted.

Dendrograms are only available from the results of hierarchical agglomerative clustering methods. When selecting **Dendrogram** in the **Plot** menu of the Analysis Manager menu bar, **t**he information contained in the ultrametric matrix is used to build the dendrogram.

Scree plots are available from the output of ordination methods including eigenanalysis. An item menu named **Scree plot** will appear among the menu items of the Analysis Manager menu bar. No dialog will be prompted. The plot produced is similar to that of a 2D series graphic.

6.3 Graphic manipulation, printing and exporting

6.3.1 Graphic controls

The following options can be used to change the appearance of a certain plot. They are available in the **Options** menu of the Graphics Editor menu bar. Not all options are applicable in all plots.

- *Resize graphic*: Enables changing the graphic size (in pixels). The dialog includes a checkbox option to ensure maintenance of the proportions between the plot's width and height.
- *Change axes range*: Permits changing the range (minimum and maximum) of scatter axes.
- *Show grid*: Checkbox menu item to include the grid in a scatter plot.
- *Edit series*: The prompting dialog of this option allows the user to change the appearance of each series of data being displayed. Changeable options are: symbol type, size and color, labels. A special option is the application multiplicative factor to data values (makes sense when descriptors and objects are displayed in a biplot) in order to rescale the series in relation to the others.
- *Edit titles*: Permits a change in the graphic title, as well as axis titles.
- *Antialiasing*: Checkbox menu item to make the overall graphic appearance softer.

Scatter graphics (whether connected or not) enable the user to 'navigate' inside them. This is done using the mouse and the keys CTRL and SHIFT. In a 2D scatter plot, pressing CTRL+left button and dragging allows the user to select an area to be zoomed in on. Once only a part of the plot it is shown, the user may 'move' the window by using dragging the mouse with the left button pressed. If CTRL+left button is pressed again without dragging it makes the plot zoom out. In a 3D scatter plot the left button on the mouse is used to rotate the plot. The right button can be used to move the plot inside the graphic panel. In the same plot, dragging the mouse with SHIFT+right button allows the user to zoom in and out of the plot.

6.3.2 Graphic printing and exporting

Graphics can be exported as image files selecting the menu option **Export to image file** in the menu **File** of the Graphics Editor. Output formats depend on the capabilities of the operating system (typically GIF, JPG, PNG image formats are allowed). To print the current graphic, select the menu option **Print** from the menu **File** of the Graphics Editor. A dialog will be prompted, asking the user to select a printer (if more than one is available) as well as some other options.

References

Bezdek, J.C. (1974). "Numerical taxonomy with Fuzzy sets". *J. Math. Biol.* 1 (1), 57-71.

Bezdek, J.C. (1981). *Pattern Recognition with Fuzzy Objective Function*. Plenum Press.

Bezdek, J.C. (1987). "Some non-standard clustering algorithms" in: Legendre, P. & Legendre, L. *Developments in Numerical Ecology. NATO ASI Series, Vol. G14*. Springer-Verlag.

Bruelheide,H. (2000). "A new measure of fidelity and its application to defining species groups". *Journal of Vegetation Science*, **11**, 167-178.

Calinski, T. & Harabasz, J. (1974). "A dendrite method for cluster analysis". *Communications in Statistics* 3: 1-27.

Chytrý,M., Tichý,L., Holt,J. & Botta-Dukát,Z. (2002). "Determination of diagnostic species with statistical fidelity measures". *Journal of Vegetation Science*, **13**, 79-90.

Cuadras, C.M, Fortiana, J. (1995). "A continuous metric scaling solution for a random variable". *Journal of Multivariate Analysis* 52, 1-14.

Cuadras, C.M, Fortiana, J. (1998). "Visualizing categorical data with related metric scaling" in: Blasius, J. & Greenacre, M. *Visualization of Categorical Data*. London. Academic Press.

Cuadras, C.M, Fortiana, J, Oliva, F. (1997). "The proximity of an individual to a population with applications in discriminant analysis". *Journal of Classification* 14, 117-136.

Davé, R.N. & Krishnapuram, R. (1997). "Robust clustering methods: A unified view". *IEEE Trans. Fuzzy Syst*. 5: 270-293.

Day, W. H. E. (1981). "The complexity of computing metric distances between partitions". *Mathematical Social Sciences* 1: 269-287.

De Cáceres, M., Oliva, F. & Font, X. (*Submitted*) "On relational possibilistic clustering".

De Cáceres, M., Font, X., García, R. & Oliva, F. (2003). "VEGANA, un paquete de programas para la gestión y análisis de datos ecológicos". *VII Congreso Nacional de la Asociación Española de Ecología Terrestre*. pp. 1484-1497.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern Classification*, 2nd edn. Wiley & Sons, New York.

Dunn, J.C. (1974). "A Fuzzy Relative of the ISODATA Process and its Use in Detecting Compact, Well Separated Clusters". *J. Cybern.* 3, 32-57.

Dunn, J.C. (1976). "Indices of partition fuzziness and the detection of clusters in large data sets" in: Gupta, M. (ed) *Fuzzy automata and decision processes* . Elsevier.

Dufrêne,M. & Legendre,P. (1997). "Species assemblages and indicator species: The need for a flexible asymetrical approach". *Ecological Monographs*, 67, 345-366.

Equihua, M. (1990). "Fuzzy clustering of ecological data". *Journal of ecology* 78: 519-534.

Everitt, B. S. (1993). "Cluster Analysis". Edwald Arnold, London, Melbourne, Auckland.

Feoli, E. & Orlóci, L. (1979). "Analysis of concentration and detection of underlying factors in structured tables". *Vegetatio* 40: 49-54.

Fowlkes, E. B. & Mallows, C. L. (1983) A method for comparing two hierarchical algorithms. *Journal of the American Statistical Association* 78: 553-569.

Frigui, H. & Krishnapuram, R. (1996). "A robust algorithm for automatic extraction of an unknown number of clusters from noisy data". *Pattern Recognition Letters* 17: 1223-1232.

Gordon, A. D. (1999). *Classification*, 2nd edn. Chapman and Hall-CRC, London.

Gower, J.C. (1966). "Some distance properties of latent roots and vector methods used in multivariate analysis". *Biometrika* 53, 325-338.

Hartigan, J. (1975). *Clustering Algorithms*. John Willey and Sons, New York .

Hathaway, R.J., Davenport, J.W., Bezdek, J.C. (1989). "Relational duals of the c-means clustering algorithms". *Pattern Recognition* 22 (2): 205-212.

Hill, M. O. (1973). "Diversity and evenness, a unifying notation and its consequences". *Ecology* 54: 427-432.

Hotelling, H. (1933). "Analysis of complex statistical variables into principal components". *J. Educ. Psychol.* 24: 417-441.

Hubert, L. & Arabie, P. (1985). "Comparing partitions". *Journal of Classification* 2: 193-218.

Jain, A.K. & Dubes, R.C. (1988). *Algorithms for clustering data*. Prentice Hall, Englewood Cliffs, New Jersey.

Kim, D.-W., Lee, K. H., & Lee, D. (2004). "On cluster validity index for estimation of the optimal number of fuzzy clusters". *Pattern Recognition* 37: 2009-2025.

Krishnapuram, R. & Keller, J.M. (1993). "A possibilistic approach to clustering". *IEEE Transactions on Fuzzy Systems* 1 (2): 98-110.

Krishnapuram, R. & Keller, J.M. (1996). "The Possibilistic C-Means Algorithm: Insights and Recommendations". *IEEE Transactions on Fuzzy Systems* 4 (3): 385-393.

Kruskal, J.B. (1964a). "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis". *Psychometrika 29*(1): 1-27.

Kruskal, J.B. (1964b). "Nonmetric Multidimensional scaling: A numerical method". *Psychometrika 29*(2): 115-129.

Lance, G-N. & Williams, W.T.. (1967). "Mixed-data classificatory programs. I. Agglomerative systems". *Austr. Comput. Journal.* 1: 15-20.

Legendre, P. & Legendre, L. (1998). *Numerical Ecology*. Second English Edition. Ed. Elsevier. 853 pp.

Legendre, P. & Gallagher, E.D. (2001). "Ecologically meaningful transformations for ordination of species data". *Oecologia* 129 (2): 271-280.

Lingoes, J. (1971). "Some boundary conditions for a monotone analysis of symmetric matrices". *Psychometrika* 76 (2): 195-203.

Magurran, A. E. (1989). *Diversidad ecológica y su medición*. Ed. Verdra, Barcelona.

Margalef, R. (1974). *Ecologia*. Ed. Omega. Barcelona. 951pp.

Marsili-Libelli, S. (1989). "Fuzzy clustering of ecological data". *Coenoses* 2: 95-106.

Milligan, G. W. (1989). "A study of the beta-flexible clustering method". *Multivariate Behavioral Research* 24: 163-176.

Milligan, G.W., Sokol L.M. (1980). "A two stage clustering algorithm with robust recovery characteristics". *Educational and Psychological Measurement* 40, 755-759.

MacQueen, J. (1967). "Some methods for classification and analysis of multivariate observation". pp. 281-297 in: L.M. Le Cam & J. Neyman (eds.) *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. University of California Press, Berkeley.

Mardia, K. V., Kent, J. T. & Bibby, J. M. (1995). *Multivariate analysis*. Academic Press, London, United Kingdom. 518 pp.

McCune, B & Mefford M.J. (1999). *PC-ORD. Multivariate analysis of ecological data. Version 4.* MjM Software Design, Gleneden Beach, OR. US. URL: http://home.centurytel.net/~mjm/

McLachlan, G. J. (1992). *Discriminant analysis and statistical pattern recognition*. Wiley & Sons, New York.

Oliva, F., Cáceres, M. de, Font, X. & Cuadras, C.M. (2001). "Contribuciones desde una perspectiva basada en proximidades al fuzzy K-means clustering". *XXIV Congreso SEIO*. Úbeda. Spain (pers. comm.).

Orloci, L. (1967). "An agglomerative method for classification of plant communities". *Journal of Ecology* 55: 193-206.

Pielou, E.C. (1966). "The measurement of diversity in different types of biological collections". *Journal of Theoretical Biology* 13: 131-144.

Podani, J. (1986). "Comparison of partitions in vegetation studies". *Abstracta Botanica* 10: 235-290.

Podani, J. (1990). "Comparison of fuzzy classifications". *Coenoses* 1: 17-21.

Podani, J. & Feoli, E. (1991). "A general strategy for the simultaneous classification of variables and objects in ecological data tables". *Journal of Vegetation Science* 2: 435-444.

Rand, W.M. (1971). "Objective Criteria for the Evaluation of Clustering Methods". *Journal of the American Statistical Association* 66: 846-850.

Rao, E. (1995). "A review of canonical coordinates and an alternative to correspondence analysis using Hellinger distance".*Qüestiió (Quaderns d'Estadistica i Investivació Operativa)* 19, 23-63.

Rényi, A. (1961). "On measures of entropy and infomation". 547-561. In: J. Neyman (ed.) *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability.* University of California Press, Berkeley.

Rousseuw, P.J. (1987). "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". *Journal of Computation and Applied Mathematics* 20: 53-65.

Ruspini, E. (1969). "A New Approach to Clustering". *Inf. Control* 15, 22-32.

Shannon, C. E. & Weaver, W. (1949). *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, IL.

Shepard, R. N. (1962). "The analysis of proximities: multidimensional scaling with an unknown distance function". *Psychometrika* 27: 125-139.

Sneath, P. H. A. & Sokal, R. R. (1973). *Numerical taxonomy – The principles and practice of numerical classification*. W. H. Freeman, San Francisco. 573 pp.

Sokal, R. R. & Michener, C. D. (1958). "A statistical method for evaluating systematic relationships". *Univ.Kans.Sci.Bull.* 38: 1409-1938.

Sokal & Rohlf (1995). *Biometry : the principles and practice of statistics in biological research*. 3rd edition. New York, Freeman.

Taylor, L.R. (1978). "Bates, Williams, Hutchinson – a variety of diversities. In: Diversity of Insect Faunas". L.A. Mound & N. Warloff (eds.), *9th Symposium of the Royal Entomological Society*, Blackwell. Oxford. pp. 1-18.

ter Braak, C.J.F. (1986). "Canonical correspondence analysis: A new eigenvector technique for multivariate direct gradient análisis". *Ecology* 67 (5): 1167-1179.

ter Braak, C.J.F. (1995). "Canonical correspondence analysis and related multivariate methods in aquatic ecology". *Aquatic Sciences* 57 (3): 255-289.

ter Braak, C.J.F. & Šmilauer, P. (1988). *CANOCO reference manual and user's guide to Canoco for Windows. Software for Canonical Community Ordination (version 4).* Centre of Biometry, Wageningen. URL: http://www.microcomputerpower.com/

Tibshirani, R., Walther, G., & Hastie, T. (2001). "Estimating the number of clusters in a data set via the gap statistic". *Journal of the Royal Statistical Society B* 63: 411-423.

van den Wollenberg, A.L. (1977). "Redundancy analysis: an alternative for canonical correlation analysis". *Psychometrika* 42(2): 207-219.

Ward, J. H. (1963). "Hierarchical grouping to optimize an objective function". *Jounal of the American Statistical Association* 58: 236-244.

Zadeh, L.A. (1965). "Fuzzy sets". *Information and Control* 8, 338-353.