Bioversity International

Handbooks for Genebanks No. 9

Statistical Analysis

for Plant Genetic Resources: Clustering and indices in R made simple

Mikkel Grum and Fredrick Atieno

Handbooks for Genebanks No. 9

Statistical Analysis

for Plant Genetic Resources: Clustering and indices in R made simple

Mikkel Grum and Fredrick Atieno

Bioversity, c/o ICRAF, P.O. Box 30677, 00100 Nairobi, Kenya

Bioversity International is an independent international scientific organization that seeks to improve the well-being of present and future generations of people by enhancing conservation and the deployment of agricultural biodiversity on farms and in forests. It is one of 15 centres supported by the Consultative Group on International Agricultural Research (CGIAR), an association of public and private members who support efforts to mobilize cutting-edge science to reduce hunger and poverty, improve human nutrition and health, and protect the environment. Bioversity has its headquarters in Maccarese, near Rome, Italy, with offices in more than 20 other countries worldwide. The Institute operates through four programmes: Diversity for Livelihoods; Understanding and Managing Biodiversity; Global Partnerships; and Commodities for Livelihoods.

The international status of Bioversity is conferred under an Establishment Agreement which, by January 2007, had been signed by the Governments of Algeria, Australia, Belgium, Benin, Bolivia, Brazil, Burkina Faso, Cameroon, Chile, China, Congo, Costa Rica, Côte d'Ivoire, Cyprus, Czech Republic, Denmark, Ecuador, Egypt, Greece, Guinea, Hungary, India, Indonesia, Iran, Israel, Italy, Jordan, Kenya, Malaysia, Mali, Mauritania, Morocco, Norway, Pakistan, Panama, Peru, Poland, Portugal, Romania, Russia, Senegal, Slovakia, Sudan, Switzerland, Syria, Tunisia, Turkey, Uganda and Ukraine.

Financial support for Bioversity's research is provided by more than 150 donors, including governments, private foundations and international organizations. For details of donors and research activities please see Bioversity's Annual Reports, which are available in printed form on request from bioversity-publications@cgiar.org or from Bioversity's Web site (www.bioversityinternational.org).

Citation: Grum M and Atieno F. 2007. Statistical analysis for plant genetic resources: clustering and indices in **R** made simple. Handbooks for Genebanks, No. 9. Bioversity International, Rome, Italy.

ISBN: 978-92-9043-735-2

Bioversity International
Via dei Tre Denari, 472/a
00057 Maccarese
Rome, Italy

TABLE OF CONTENTS

Introduction

INTRODUCTION

This handbook aims to give researchers tools and methods to analyse mathematically complex morphological characterization data.

R is a powerful software language and environment for statistical computing and graphics, developed by volunteers from around the world, working together over the Internet. The core development team consists of fewer than twenty programmers who maintain and develop the general **R** environment, with contributions from many others. A much larger number of contributors write individual statistical packages for **R** that address specific needs.

Statisticians and researchers write many of these statistical packages for their own research purposes, ensuring that **R** is in the forefront of developments in statistics. Another of **R**'s strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed.

On the down side, if you are used to point-and-click interfaces, you may find that **R** has a relatively steep learning curve. As one of the developers of **R** once put it, **R** is not designed to make easy things simple, but to make difficult things possible. Nonetheless, several projects are working on making **R** easier to use, and interfaces should improve in the near future.

We would not propose **R** had we not found that it has many facilities useful to the analysis of plant genetic resources data, facilities not commonly found in other statistical packages. We also believe that with a step-by-step guide, providing examples of typical analyses required in plant genetic resources research, most researchers will be able to implement such analyses quite simply on their own data.

This tutorial provides such a guide, beginning with simple examples that soon demonstrate the power of **R**, and allows users to gradually become acquainted with the more complex aspects. Just enough explanation of the statistics is given for readers with a basic understanding of the subject to understand the output of the analyses.

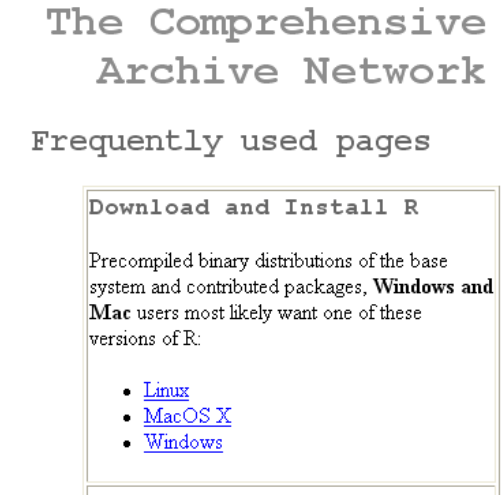
For broader and more complete introductions to **R**, or manuals on specific aspects, or reference works, the reader is referred to the wide range of freely available manuals that can be downloaded by going to www.r-project.org and clicking on the link “Contributed” under the heading “Documentation” (left hand column of the page).

Chapter 1

CHAPTER 1 DOWNLOADING AND INSTALLING R

R can be downloaded from the main website http://www.cran.r-project.org

Under the heading **Download and Install R**, the Windows bullet is the hypertext link for downloading the binary version of **R** for a PC running Windows. The **base** subdirectory has the main **R** self-extracting executable file in it:


The Comprehensive R Archive Network

Frequently used pages

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Linux
- MacOS X
- Windows

Clicking the hypertext link **base** gives the screen:

R-2.6.1 for Windows

This directory contains a Windows binary distribution of R-2.6.1.

Patches to this release are incorporated in the r-patched snapshot build.

A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.

In this directory:

README.R-2.6.1	Installation and other instructions.
CHANGES	New features of this Windows version.
NEWS	New features of all versions.
R-2.6.1-win32.exe	Setup program (about 29 megabytes). Please download this from a mirror near you.
old	Previous releases.
md5sum.txt	md5sum output for the setup program. A Windows GUI version of md5sum is available at http://www.md5summer.org/; a Windows command line version is available at http://www.etree.org/md5com.html.

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information, including upgrade advice.

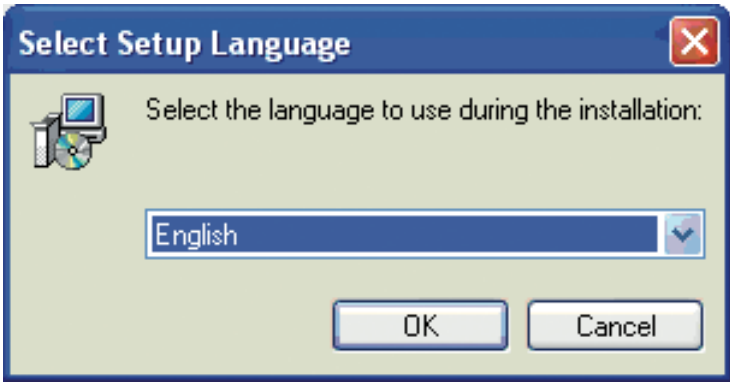
Note to webmasters: A stable link which will redirect to the current Windows binary release is <CRAN MIRROR>/bin/windows/base/release.htm.

Last change: 2007-11-26. by Duncan Murdoch

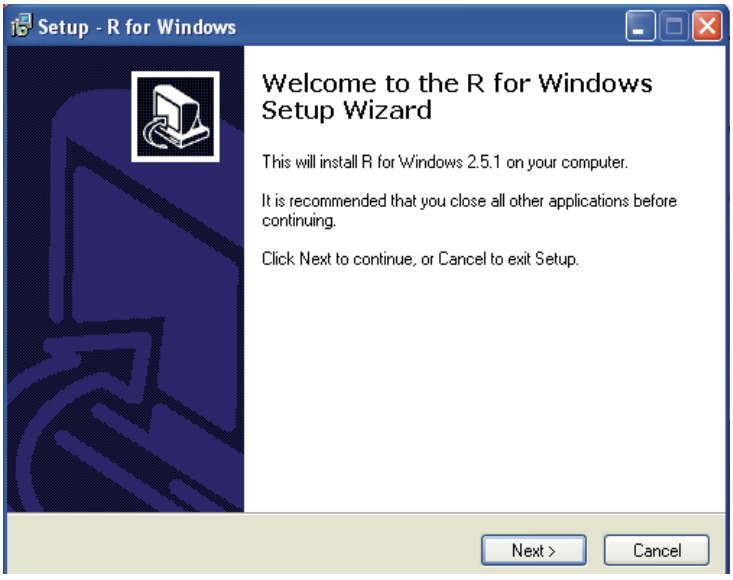
The R-X.X.X-win32.exe file can be saved to your local desktop by a "right mouse-click" on the R-X.X.X-win32.exe hypertext link. In the example above, the file is R-2.6.1-win32.exe, but new versions of **R** are released at least every six months and the number changes with each release. Select "Save Target As ..."; browse the file system to find the Desktop and then "Save".

Installation

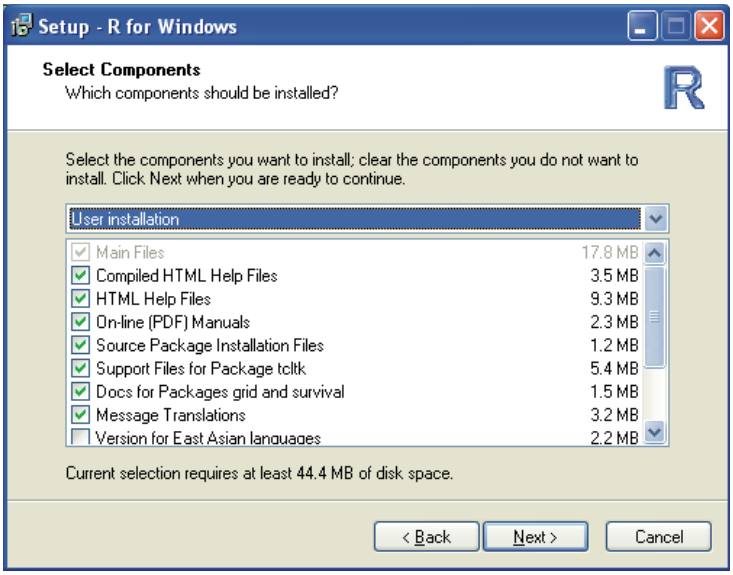
Double clicking this downloaded file should give the following series of installation screens:



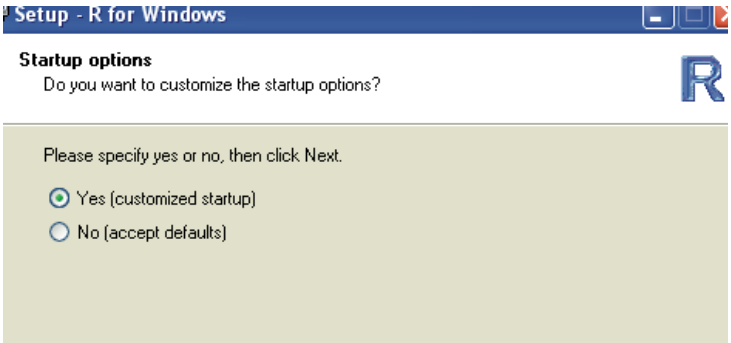
Choose the language you would like to use during installation and press OK. You will be taken to the next screen.



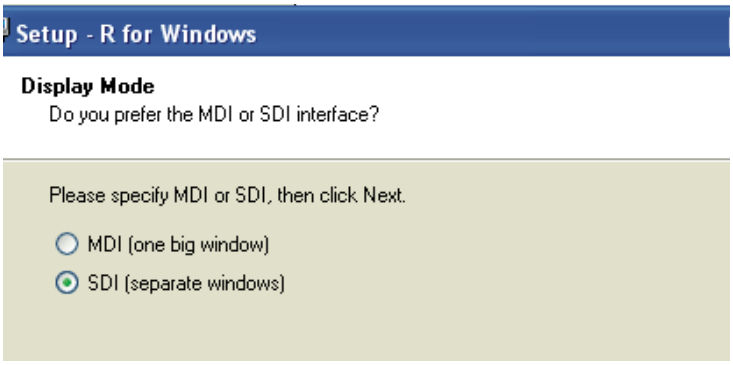
Select the **Next** button. Continue to choose the default selections in the installation window. If installation is proceeding properly, you will see the installation screen below, from which you can select the components of **R** to install. However, we recommend simply choosing the default selections. Once installation is finished, an **R** icon will be placed on the Desktop.



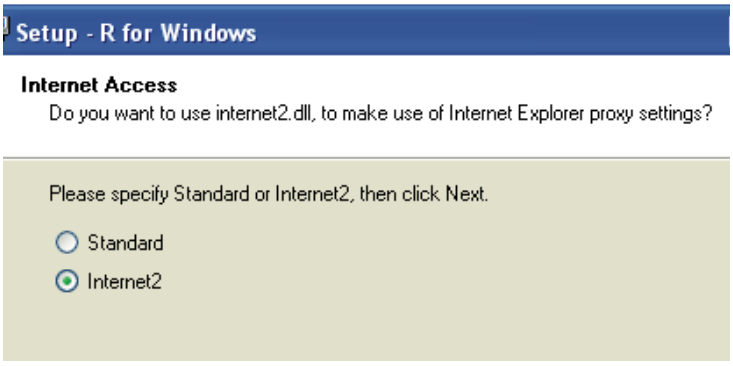
Clicking **Next** will give you the window below, which allows you to either choose to customize the start up or not. We recommend you choose **YES**, for customized start up. This will enable you to configure access to Help and the Internet for installing contributed packages.



Choose **Next**, which will give you the window below to select the **R** interface. While it is safe to go with the default installation option (MDI), we recommend **SDI**, which will give you separate windows for graphics and the console, making your work easier.



In the next window for **R** setup choose **Default** and press **Next**, which brings you to the window below, where you are to choose the type of Internet connection to use. We recommend you choose **Internet2**, which will allow you to use the proxy settings of your local server in case you are working behind a firewall.



From here onwards, just press **Next** at every step, choosing the defaults until the installation is complete. Then press **FINISH**. An **R** icon will be placed on the desktop. You are now ready to use **R**. Congratulations!!

Chapter 2

CHAPTER 2
HIERARCHICAL CLUSTERING

Basics

One of the most frequently used statistical analyses in plant genetic resources work is hierarchical cluster analysis. Based on passport data, characterization data or evaluation data, this analysis produces a dendrogram showing the approximate relationships among a set of accessions.

R has a very wide range of options for cluster analysis. This section looks at the most commonly used and straightforward ones.

We will start with an example using Anderson's classical data set on Iris (Anderson, 1935). This data set comes as a standard with **R**. Try typing the following at the prompt to load and view the data (press "Enter" to submit each line)[1] (Figure 1):

```
data(iris)
iris
```

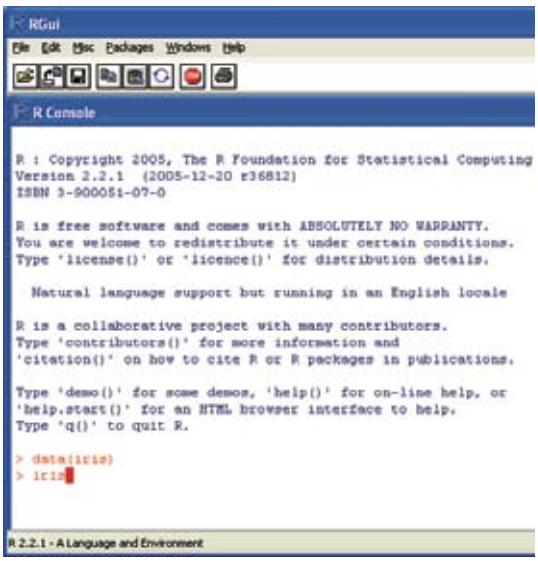


Figure 1. The **R** interface.

[1] Note that **R** is case sensitive and you must therefore use lower-case and capital letters consistently.

Why cluster?

Cluster analysis may serve a number of purposes in plant genetic resources work:

Where at least some characterization data exists for an entire collection, a genebank curator may use clustering to decide the order in which the plots are laid out in the field, so that morphologically similar accessions are next to each other. This facilitates the comparison of similar varieties in the field and the detection of duplicates. The clustering may also indicate accessions where molecular markers may be necessary to detect duplicates.

Clustering can be used in selecting a core collection. Morphological data alone would be an inadequate basis for determining a core collection, but can be combined with geographical data to create a robust procedure. One approach for this is to divide the collection into geographical units and sample one accession from each morphological cluster within each geographical unit. The desired size of the core collection determines the number of geographical units and clusters to use. This is consistent with the approaches recommended by van Hintum *et al*. (2000).

For the breeder, clustering gives a sense of the relationships among accessions and can help the selection of parents needed to maintain adequate diversity in the breeding programme. It can also help determine which varieties to evaluate in which localities, as information is gathered on which morphotypes do best in which areas.

For the researcher, clustering can help understand the structure of diversity in relation to a large number of factors, e.g. farmer variety names; distribution of diversity among farmers, villages, districts or other units; and the levels of similarity, dissimilarity or diversity of the accessions found in different units. Finally, clustering may be used in calculating diversity indices, as described in the next chapter. This provides an understanding that is essential for the development of conservation strategies, prioritizing interventions and facilitating the use of germplasm.

Nonetheless, it is worth mentioning a few caveats. Clustering is only an exploratory technique, and one that is highly dependent on the subjective choice of methods for calculating distances and agglomeration. It does not directly show you which variables contribute to the observed structure. Ordination methods may be more useful in understanding the underlying variables in clusters, though posterior tabulation and regression may also help to identify contributing variables.

You will see that there are 150 accessions belonging to 3 different species. Each accession has been characterized for four different descriptors: sepal length, sepal width, petal length and petal width. These are all straightforward numerical variables; so to do the most common hierarchical cluster analysis just type:

```
distiris<-dist(iris[,1:4])
hclustiris<-hclust(distiris)
plot(hclustiris)
```

The three lines of code above illustrate the three steps of a hierarchical cluster analysis: creating a distance or dissimilarity matrix [`dist()`]; clustering or agglomeration [`hclust()`]; and plotting [`plot()`]. The terms `distiris` and `hclustiris` are names that we have given the outputs of the each of the analyses. We have tried to make these names descriptive, so that we can easily remember them, but we could have used any name. In **R**, these outputs are known as objects.

The command `iris[,1:4]` indicates that you are analysing the data in columns one to four of the iris dataset. The square brackets are used to select a subset of the dataset, with numbers before the comma (in this case, none) indicating rows, and numbers after the comma indicating the columns to select.

This analysis will give you Figure 2. You can click the maximize button in the top right corner of the screen to stretch the graph horizontally. To print the graph legibly, you will need to change to Landscape mode under "Properties . . ." in the "Print" dialogue box.

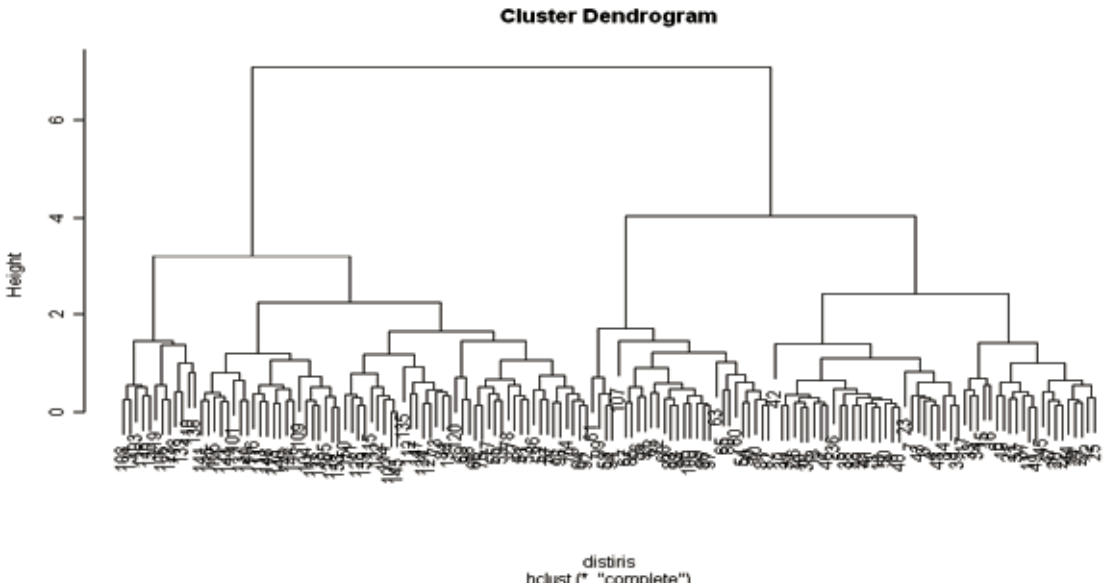


Figure 2. Default cluster dendrogram for Anderson's Iris data.

In this example. we used the default options for hierarchical cluster analysis in **R**. However, to get the analysis that we really want, we would need to change some of these options. The `hclust()` function

in **R** uses "`complete`" linkage as its default method. We prefer to use either the "`average`" (akin to the popular UPGMA or Unweighted pair-group method, arithmetic average) or "`ward`" (Ward's method is an alternative approach based on minimizing the sum of squares).

We would also like the labels on the dendrogram to be more informative and the writing smaller, so that the labels do not overlap. To do this, use the arrow-up key to recall the commands you previously typed, and edit the text to match the lines below. Use the left arrow key to get to the part you want to edit. Here we have changed the method to "`average`", added the species names as labels and reduced their font size with the option `cex = 0.5`:

```
hclustiris<-hclust(distiris,method="average")
plot(hclustiris,labels=iris$Species,cex=0.5)
```

Remember that capitalization and accurate spelling are important. If you write `iris$species` you will not get your labels. You need the capital S in Species, because that is how it is spelt in the data set. Even worse, if you write "`Average`" with a capital A or make a spelling mistake in that option, **R** will use the default "`complete`" option and you may never notice the mistake!

This will give you Figure 3 below.

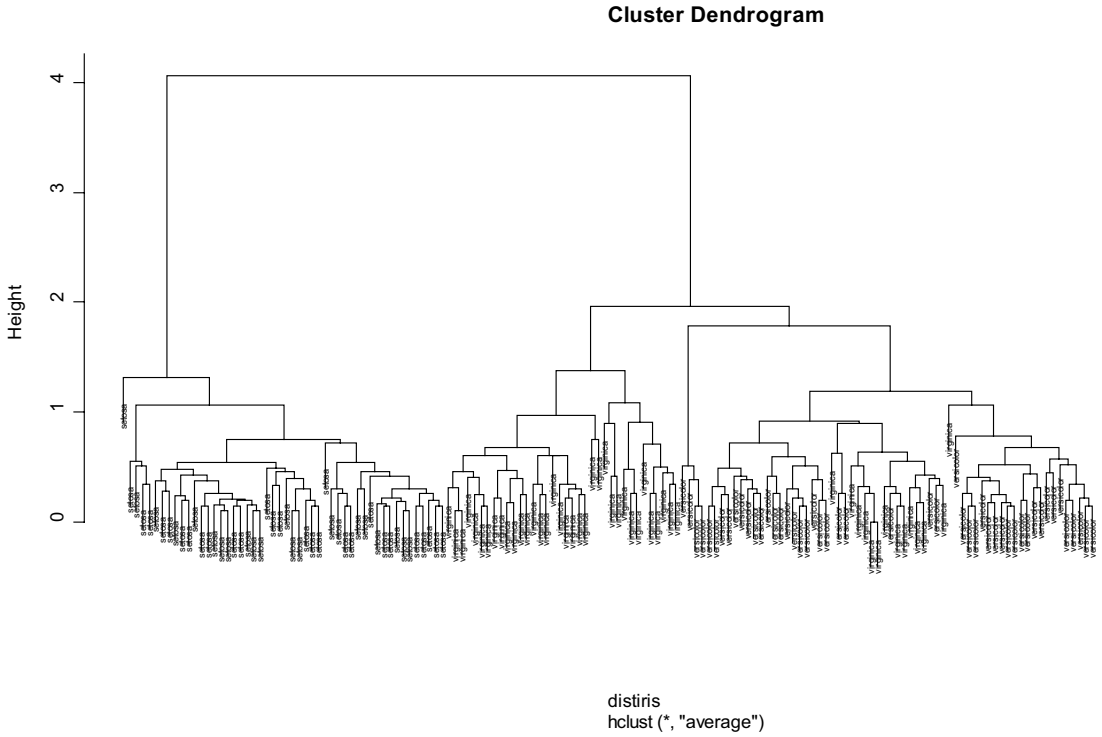


Figure 3. Cluster dendrogram for Anderson's Iris data, using the "average" method of agglomeration. The labels are species names.

We can now see that the cluster analysis separates the accessions of *Iris setosa* clearly from the other two species, but that with the limited data we have, we are not able to separate the accessions of *I. versicolor* and *I. virginica* effectively.

Try adding the option `hang=-1` to the plot function (just do not ask why –1), by using the upward arrow on your keyboard to recall the line with the `plot()` function, adding `hang=-1` as indicated below and pressing 'Enter':

```
plot(hclustiris,labels=iris$Species,cex=0.5,
  hang=-1)
```

We also sneaked in the $ sign in `iris$Species`. This is used when you want to select a specific component of the dataset, so in this case `iris$Species` indicates the variable `Species` in the dataset iris.

Notice that the labels are now all aligned. This is a purely cosmetic change, which some prefer. It does not change the analysis in any way. You can also try changing the `cex` value, which determines relative font sizes.

Now try doing the analysis using the option `method="ward"` by using the upward arrow on your keyboard to recall the line with the `hclust()` function, replacing the word "`average`" with "`ward`", and pressing 'Enter'. Use the upward arrow again to recall the line with the `plot()` function and press 'Enter'. How did the figure change?

Interpreting a hierarchical cluster dendrogram

The hierarchical cluster dendrogram represents the relationships among the accessions in terms of approximate distances, based on morphological traits. The distance between two accessions is approximately proportional to the height of the horizontal line that joins the two groups that the accessions are in. So looking at Figure 2, produced earlier, the distance between accessions 63 and 42 is approximately 4. Note that the number of accessions 'between' two accessions is not at all important. Thus, the distance between accessions 107 and 23 is also approximately 4.

What is the approximate difference between accessions 107 and 63?

In hierarchical cluster displays, the program needs to decide at each join to specify which sub-tree should go on the left and which on the right. The algorithm used in '`hclust`' is to order the sub-tree so that the tighter cluster is on the left (the last, i.e. most recent, join of the left sub-tree is at a lower value than the last join of the right sub-tree).

It is possible in **R** to re-order the clusters to give a more intuitive display, where clusters are ordered so that the accession in one cluster that has the smallest distance to the accessions in the next cluster is the accession that is placed adjacent to the next cluster (see Figure 5). In order to do this, you require the package `gclus`, which does not automatically come with **R**, but is what is called a "contributed" package.

If you are connected to the Internet it is easy enough to install `gclus`. Just click "Packages|Install package(s)…", select `gclus` from the menu and click OK (Figure 4).

If you are not connected to the Internet, but have the software package on a CD, the procedure is almost the same. Just click "Install package(s) from local zip file…" and navigate to the folder containing the downloaded packages.

R is written by a large number of people around the globe that each maintains a package (or library) of functions (or procedures). Some of the functions that you typed above, like `hclust()` and `plot()` are base functions and can be called without loading any library, but the function `reorder.hclust()` is in the package `gclus`, which therefore needs to be downloaded and loaded before the function can be used.

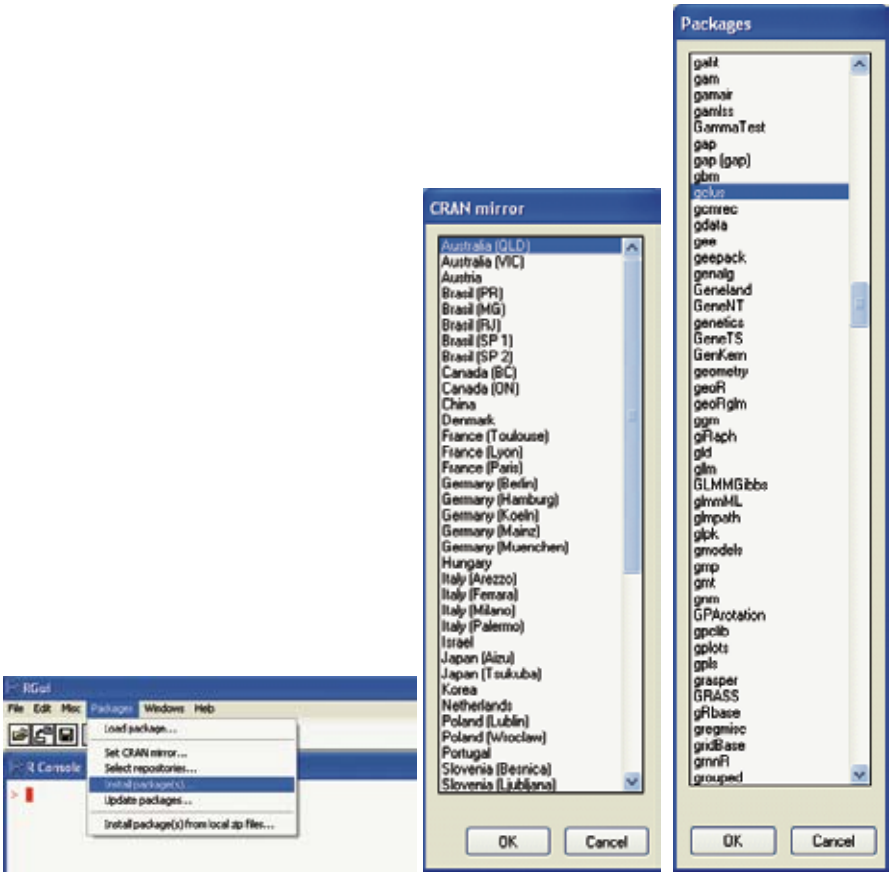


Figure 4. Installing a new package from the Internet

When you have downloaded the package, compare Figures 2 and 3 with the following plot (Figure 5):

```
library(gclus)
hclustreordered<-reorder.hclust(hclustiris,distiris)
plot(hclustreordered,cex=0.5)
```

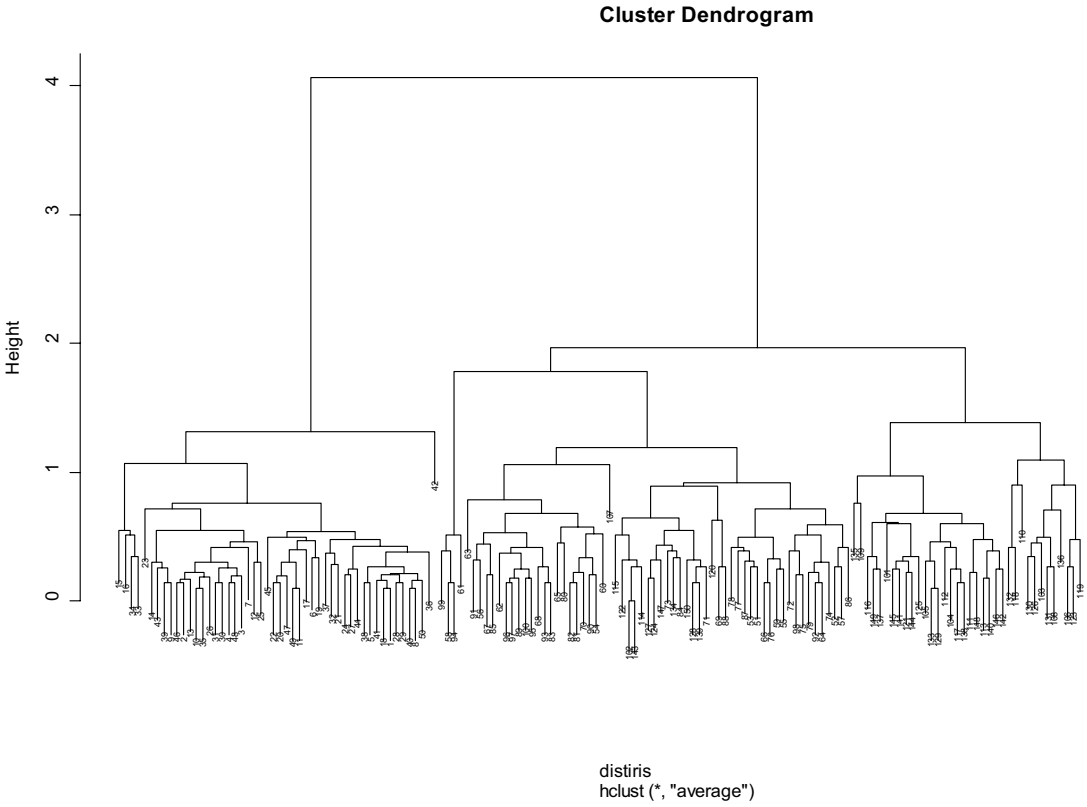


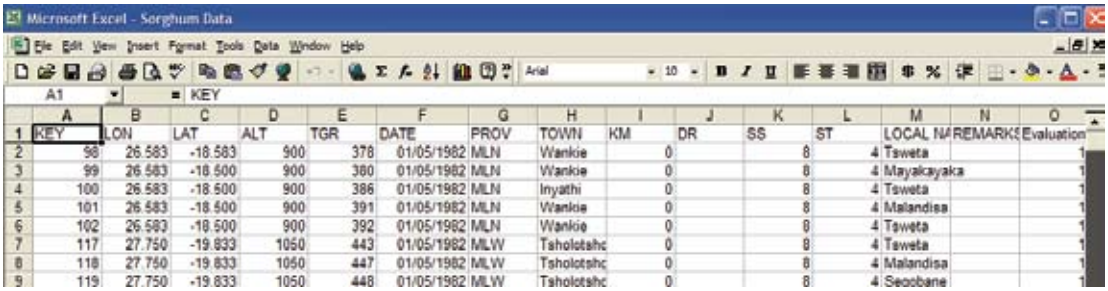
Figure 5. The dendrogram re-ordered so that more-similar accessions are closer to each other, even when they are in different clusters.

With this re-ordering, you can see that accession 99 is more similar to accession 42 than 107 is to 42. Since the dendrogram still shows both distances as approximately Height 4, you have no way of telling how much closer the first two accessions are. This display could be very useful for ordering accessions in demonstration plots or for further characterization, where there is advantage in having similar accessions closer to each other.

Importing your own data

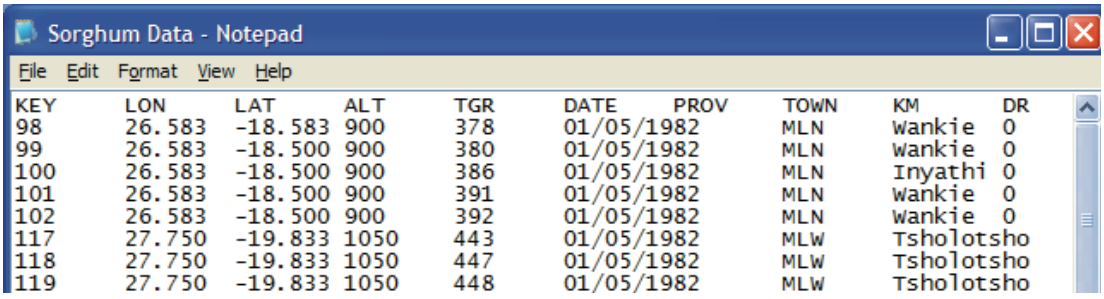
Entering your data directly into **R** is a tedious process, to say the least; it is simpler to enter your data into a program like Excel or Access first, in rows and columns, and then export/save it as a tab-delimited text file. This tutorial assumes that you have one (and only one) row at the top that gives the name of each variable (see Figure 6). For the following exercises we will use a data set from Appa Rao and Mushonga (1987) that contains both passport and morphological characterization data (see Table 1 for details). This data set can be downloaded from http://www.bioversityinternational.org/Publications/Sorghum_Data.xls.

A word of **warning** about Excel: if some cells outside the data area contain, or have previously contained, information this can make it impossible to import the exported text file into **R**. The solution is always to copy just the part of the spreadsheet containing the data that you want onto a new, clean spreadsheet before saving this as a tab-delimited text file. I recommend using tabs rather than a comma or a space to separate variables, because tabs are rarely found in normal text.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	KEY	LON	LAT	ALT	TGR	DATE	PROV	TOWN	KM	DR	SS	ST	LOCAL NA	REMARKS	Evaluation
2	98	26.583	-18.583	900	378	01/05/1982	MLN	Wankie	0		8	4	Tsweta		1
3	99	26.583	-18.500	900	380	01/05/1982	MLN	Wankie	0		8	4	Mayakayaka		1
4	100	26.583	-18.500	900	386	01/05/1982	MLN	Inyathi	0		8	4	Tsweta		1
5	101	26.583	-18.500	900	391	01/05/1982	MLN	Wankie	0		8	4	Malandisa		1
6	102	26.583	-18.500	900	392	01/05/1982	MLN	Wankie	0		8	4	Tsweta		1
7	117	27.750	-19.833	1050	443	01/05/1982	MLW	Tsholotsho	0		8	4	Tsweta		1
8	118	27.750	-19.833	1050	447	01/05/1982	MLW	Tsholotsho	0		8	4	Malandisa		1
9	119	27.750	-19.833	1050	448	01/05/1982	MLW	Tsholotsho	0		8	4	Seoobane		1

Figure 6. The file Sorghum Data in Excel.




```
KEY     LON      LAT      ALT   TGR   DATE        PROV   TOWN       KM       DR
98      26.583   -18.583  900   378   01/05/1982  MLN    Wankie     0
99      26.583   -18.500  900   380   01/05/1982  MLN    Wankie     0
100     26.583   -18.500  900   386   01/05/1982  MLN    Inyathi    0
101     26.583   -18.500  900   391   01/05/1982  MLN    Wankie     0
102     26.583   -18.500  900   392   01/05/1982  MLN    Wankie     0
117     27.750   -19.833  1050  443   01/05/1982  MLW    Tsholotsho
118     27.750   -19.833  1050  447   01/05/1982  MLW    Tsholotsho
119     27.750   -19.833  1050  448   01/05/1982  MLW    Tsholotsho
```

Figure 7. The tab-delimited file `Sorghum_Data.txt` opened in notepad.

Table 1. Variables in the sorghum data set.

Variable	Descriptor	R mode
KEY	Key	Integer
LON	Latitude	Numeric
LAT	Longitude	Numeric
ALT	Altitude	Integer
TGR	Collector no.	Factor
DATE	Collecting date	Factor
PROV	Province	Factor
TOWN	Town/village	Factor
KM	Distance from town	Integer
DR	Direction from town	Factor
SS		Integer
ST		Integer
LOCAL.NAME	Farmer variety name	Factor
REMARKS	Remarks	Factor
EVALUATION.DATA	Presence of evaluation data	Integer
PC	Plant colour at harvest	Factor
MC	Leaf midrib colour	Factor
SJ	Stalk juiciness	Factor
JQ	Juice quality	Factor
LD	Lodging	Ordered factor
SN	Synchrony of flowering	Factor
EN	Head exertion	Numeric
BD	Bird damage	Ordered factor
PA	Overall plant aspect	Ordered factor
EV	Early vigour	Ordered factor
FL	Days to 50% flowering	Integer
HT	Plant height	Integer
HL	Head length	Integer
HW	Head width	Integer
PT	Productive tillers (#)	Integer
KW	Kernel weight	Integer
EX	Endosperm texture	Factor
EC	Endosperm colour	Factor
ET	Endosperm type	Factor
KL	Kernel lustre	Factor
SC	Sub-coat	Factor
KP	Kernel plumpness	Factor
F1-F12	Mean monthly rainfall	Numeric
F13-F24	Mean temperature	Numeric
F25-F36	Mean diurnal temperature fluctuation	Numeric

In **R**, select "File|Change dir…|Browse" from the menu to set your working directory and point **R** towards the directory where you have your files (Figure 8).

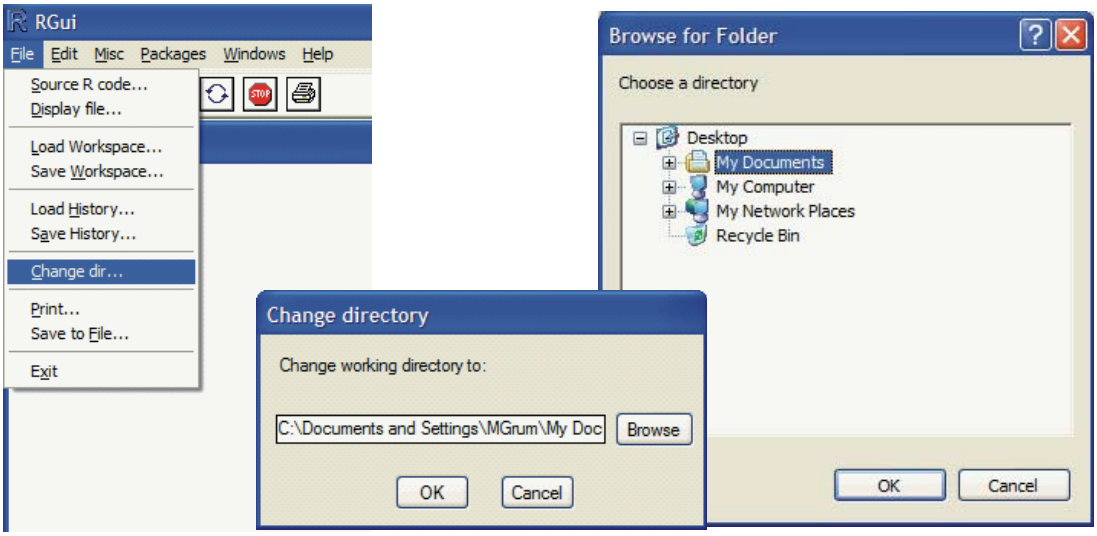


Figure 8. Changing your working directory in **R** for Windows.

Now import your data file to **R** with:

```
sorghum<-read.table("Sorghum_Data.txt",
  header=T,sep="\t")
```

substituting the names `Sorghum_Data.txt` and `sorghum` with your own choice of file names. You have just created a data object ('`sorghum`' or whatever name you gave it) of the type **R** calls a dataframe. The option `header=T` or `header=TRUE` indicates that you have a header row with variable names. The option `sep="\t"` tells the program that you have delimited your variables using tabs.

R has a long list of other ways to import data, including linking directly to and from almost any database, e.g. Access. For more on these other options, click "Help|Manuals|**R** Data Import/Export" and read the manual, though in most cases I would stick with the approach suggested above.

Light data validation

Before going on to analyse your data, you should look closely at your new dataframe to search for trends and errors in the data. The commands below will help you do this.

```
dim(sorghum)
```

will give you:

```
[1] 137 74
```

which is the number of rows and columns you have.

Now try:

```
sorghum[1:10,1:10]
sorghum[128:137,65:74]
```

This will give you the data in the top left corner (the top ten rows and the first ten columns) and the bottom right corner (the last ten rows and the last ten columns) of the table respectively.

```
str(sorghum)
```

will tell you how many observations and variables you have, list the variables in your data set, say what the variable type is, and even show you samples of the data in the variable. We will return to the variable type in the next section, because if you want to do this analysis right, there are a few things that you will need to do.

```
summary(sorghum)
```

is an extremely useful command for spotting trends and errors in your data. Look carefully at the maximum and minimum values for each numerical variable. Are they within the expected range? For factor variables with only a few levels, you will see how many entries there are for each level. For factor variables with many levels, e.g. a list of variety names with the variable name LOCAL.NAME, the `table()` command may be more useful (remember capitalization!):

```
table(sorghum$LOCAL.NAME)
```

There are several examples of inconsistently spelled names. Can you spot some of them?

Bedhlana	Bimba	Bimbe	Chetacheya	Chibedlani
1	1	1	2	1
Chibedyane	Chibuku	Chibulanye	Chifumbata	Chikombe
1	1	1	8	2
Chikondoma	Chikota	Chikrochani	Chimhondoro	Chimondo
1	7	1	1	1
Chimutode	Chinzetu	Chiota	Chiponda	Chitate
1	1	1	1	2
Chiunda	Chivedyana	Chivendaw	Compact head	Dewe
1	2	1	1	1
Fumbata	Gangara	Gangare	Gwusi	Impala
1	4	1	1	1
Isifumbata	Jayta	Kabhuruwayo	Kanzwono	Loose head
1	1	2	1	1
Maboyana	Mabumbe	Malandisa	Mashava	Masintiri
1	1	3	2	1
Matipa	Matserendende	Mavoyana	Mayakayaka	Mayere
2	1	1	1	1
Mbwende	Mhando	Mhonda	Mhonde	Mososo
1	1	1	1	1
Muchaini	Muise	Mungonellengo	Mungore	Murara
2	1	2	2	1
Mururu	Musoso	Mutanda	Mutode	Ngaima
1	2	1	3	1
Ngumane	Nzende	Nzetu	Pumbata White	Red Swazi
4	1	2	1	1
Rongwe	Rudebare	Rugare	Rundende	Rundewda
1	2	1	3	1
Rundunde	Rushutura	Rusutura	Segobane	Tswedani
1	1	1	1	2
Tsweta	Udayakayaka	Umkumbe	Vedyana	Zangewaya
17	1	1	3	1

Data types – critical distinctions

We indicated earlier that we would talk more about data types, so here goes. The states that form a descriptor may be of qualitative or quantitative nature, as such descriptors take several mathematical types. In the IPGRI/Bioversity descriptor lists, the descriptors are classified as biological (characterization and evaluation descriptors), taxonomic (species, sub-species), geographical (longitude, latitude, topography), ethnobotanical (ethnic group, vernacular name, plant uses), etc. However, the mathematical classification of descriptors is independent of these discipline-based groupings.

The mathematical type of a descriptor determines the statistical functions that can be employed for its analysis. For example, parametric correlations (Pearson's r) may be calculated between quantitative descriptors, while non-parametric correlations (such as Kendall's τ) may be used on ordered descriptors that are not quantitative, and chi-squared for qualitative descriptors. This is a critical point. It is not just that, for example, correlation does not work with factors, but that it does not make sense. The three main mathematical types of descriptors that we will deal with here are classified as Numeric, Factor and Ordered factors in **R**. Table 2 shows how these relate to some common morphological descriptors.

It is regrettably common to see unordered qualitative (nominal or categorical) descriptors analysed as if they were numeric, simply by replacing, for example, each colour with a number, i.e. red, blue and green with 1, 2 and 3. This is tantamount to saying that red is closer to blue than red is to green, which can obviously distort the results of a statistical analysis that is made based on this premise.

Table 2. Different mathematical types of descriptors and how they are defined in **R**.

R-mode	Descriptor types	Examples
Factor	Binary (two states, presence-absence)	Plant type (indeterminate, determinate)
		Nodulation activity (absent, present)
		Prominence of leaf vein (yes, no)
	Multi-state (many states)	
	Non-ordered (qualitative, nominal, attributes)	Leaf pubescence (upper surface of leaf, lower surface, both upper and lower surfaces, only leaf margin)
		Seed shape (oblate, triangular, rhomboid, square, obtriangular, spherical)
		Seed coat colour (grey, brown, yellow-green, pink, red-purple, grey-mottled)
Ordered factor	Ordered	
	Semi-quantitative (rank-ordered, ordinal)	Plant growth habit (prostrate, spreading, semi-erect, erect)
		Leaf colour (light green, green, dark green)
		Soil particle size classes (clay, fine silt, coarse silt, very fine sand, fine sand, medium sand, coarse sand, very coarse sand)
Numeric or Integer	Quantitative (metric, measurement)	
	Discontinuous (meristic, discrete)	Number of seed per pod (integers)
	Continuous	Maturation period (days)
		Peduncle length (cm)
		Seed volume (cm^3)
		Seed yield per plant (g)

A bit of good practice: solve the problem at its root and do not code levels of qualitative factors as 1, 2, 3. In the example above use "red", "blue" and "green". That way you avoid forgetting to recode the variables and avoid forgetting what the coding means!

R will automatically identify alphanumeric variables (variables with letters, or letters and numbers) as Factors and will identify numeric variables as either numeric (if they have decimals) or integers (if they have no decimals). However, when Factor variables are coded using numbers, **R** can obviously not detect this by itself. **R** can also not detect if a variable is an ordered factor, or detect the order of unnumbered variable states. In these cases, we must tell **R** what variable types we have before we proceed with the analysis.

In the case of our sorghum data, the variables were described in Table 1. Variables PC to KP are morphological descriptors and contain the data that we are interested in analysing in the first instance.

However, some of these variables are coded with numbers, though they are in fact factors or ordered factors. We need to let **R** know this, which is done with the following commands:

```
sorghum$LD<-ordered(sorghum$LD)
sorghum$SN<-factor(sorghum$SN)
sorghum$EN<-ordered(sorghum$EN)
sorghum$BD<-ordered(sorghum$BD)
sorghum$PA<-ordered(sorghum$PA)
sorghum$EX<-ordered(sorghum$EX)
sorghum$EC<-factor(sorghum$EC)
sorghum$ET<-factor(sorghum$ET)
sorghum$KL<-factor(sorghum$KL)
sorghum$SC<-factor(sorghum$SC)
sorghum$KP<-factor(sorghum$KP)
```

Now try looking at the structure once more:

```
str(sorghum)
```

Saving your work, exiting R and getting back

At this point you may want to save your work, close **R** and take a rest. Click "File|Save history…" and all your commands, including every mistake that you made on the way, get saved to a text file. If you give the file a name with a .**R** extension, e.g. temp.R, you can open this file in **R**'s text editor with "File|Open sript …". You might want to edit your errors or delete certain lines.

You can exit **R** with "File|Exit". When prompted to save your workspace image (Figure 9), my advice is to click "No". That way you avoid building up garbage in your workspace.

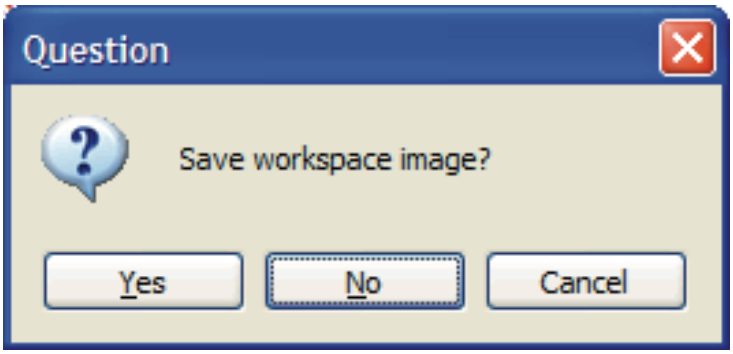


Figure 9. Make sure that you have saved the commands that you want to use again before clicking "No" on exit.

When you start **R** again, you simply set the working directory and open your saved script. You run the commands in **R** by highlighting them in **R**'s editor and pressing Ctrl-R. This will get you right back to where you left off. You can run the whole file, or you can use just the lines that you need to get you where you want to be.

Selecting a subset of your data to analyse

For almost all analyses that you do, you will want to select a subset of the data. In this case, you want to select the variables with quantitative morphological characterization data.

```
quantsorghum<-subset(sorghum,select=EV:KW)
```

See what this gave with:

```
str(quantsorghum)
```

You can get more information and further examples by typing:

```
?subset
```

Now analyse the quantitative data as with the iris data set:

```
distquant<-dist(quantsorghum)
hclustquant<-hclust(distquant)
plot(hclustquant)
```

Though you get a pretty nice dendrogram, it was based on only one-third of your variables, and even they should probably have been standardized so that a variable does not become a thousand times more important because you measure it in millimetres rather than metres.

The function `dist()` calculates the Euclidian distance (the square root of the sum of the squared distances along each dimension/variable) by default, but can also calculate maximum, Manhattan, Canberra or binary distances; none of them are suitable for Factor variables. In practical terms, this means that we cannot use the function `dist()`. But **R** has solutions, so read on!

Cluster analysis with mixed data types

It is disturbingly common to see a cluster analysis of mixed data types carried out as though all the data was numeric by simply giving a number to each variable state, e.g. red=1, blue=2, green=3. Mathematically, this is like saying that red is closer to blue than it is to green. This is of course nonsense that can seriously distort your analysis.

As an alternative, **R** provides a function `daisy()` in the library `cluster` (Struyff *et al*., 1996) based on a general coefficient of dissimilarity proposed by Gower (1971). This combines different types of descriptors, and processes each one according to its own mathematical type.

So this time round you want to select *all* the morphological variables. The first of these is PC and the last is KP and includes all the variables in between, so can be written:

```
morphsorghum<-subset(sorghum,select=PC:KP)
```

Since the `daisy()` function is in the cluster library, you must first load the cluster library. To do this, type:

```
library(cluster)
daisymorph<-daisy(morphsorghum)
hclustmorph<-hclust(daisymorph)
plot(hclustmorph)
```

Before going on, select 'History|Recording' on the menu in the graphics window. This will store the graphs that you create for the rest of the session and you can move back and forth through the session's graphs by using the page up and page down keys.

What does `daisy()` do?

The function slightly extends Gower's definition so as to cover ordinal and ratio variables in addition to quantitative and multi-state variables. The definition of the dissimilarity in `daisy()`, *d(i,j)*, is

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)}\, d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{f}} \in [0,1]$$

where

$d_{ij}^{(f)}$ = contribution of variable *f* to *d(i,j)*, which depends on its type:

- *f* binary or nominal: $d=0$ if $x_{if} = x_{jf}$, and $d_{ij}^{(f)}=1$ otherwise,
- *f* interval scaled: $d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{\max_h x_{hf} - \min_{hf} x_{jf}}$,
- *f* ordinal or ratio-scaled: compute ranks r_{if} and $z_{if} = \frac{r_{if} - 1}{\max_h r_{hf} - 1}$ and treat these z_{if} as interval-scaled,

and $\delta_{ij}^{(f)}$ = weight of variable *f*, where: $\delta_{ij}^{(f)} = 0$, if x_{if} or x_{jf} is missing; $\delta_{ij}^{(f)} = 0$, if $x_{if} = x_{jf} = 0$ and variable *f* is asymmetric binary; otherwise $\delta_{ij}^{(f)}=1$.

Now try embellishing the last two lines a bit to change the method of agglomeration, add variety names as labels and reduce the font size. There is no need to execute the first two lines again as you have already loaded the library `cluster` and created the object `daisymorph`. The option `ylab=` gives the label for the y-axis.

```
hclustmorph<-hclust(daisymorph,method="average")
plot(hclustmorph,labels=sorghum$LOCAL.NAME,cex=0.5,
  ylab="Dissimilarity")
library(gclus)
hclustreord<-reorder.hclust(hclustmorph,daisymorph)
plot(hclustreord,labels=sorghum$LOCAL.NAME,cex=0.5,
  ylab="Dissimilarity")
```

Have a careful look at how the farmer variety names are spread across the cluster dendrogram in Figure 10. This shows just how loosely farmer variety names are used in Zimbabwe. The names often refer to one to three specific traits, and all other aspects of the plant can vary considerably from one site to another. Have a look at each accession's town of origin, by changing the last line to:

```
plot(hclustreord,labels=sorghum$TOWN,cex=0.5,
  ylab="Dissimilarity")
```

How good is the cluster dendrogram?

With so many different ways of doing cluster analyses, and each one giving a different result, it is natural to ask which is the best method? The answer is that there really is no way of knowing for certain, but there are a few ways in which you can attempt to answer part of the

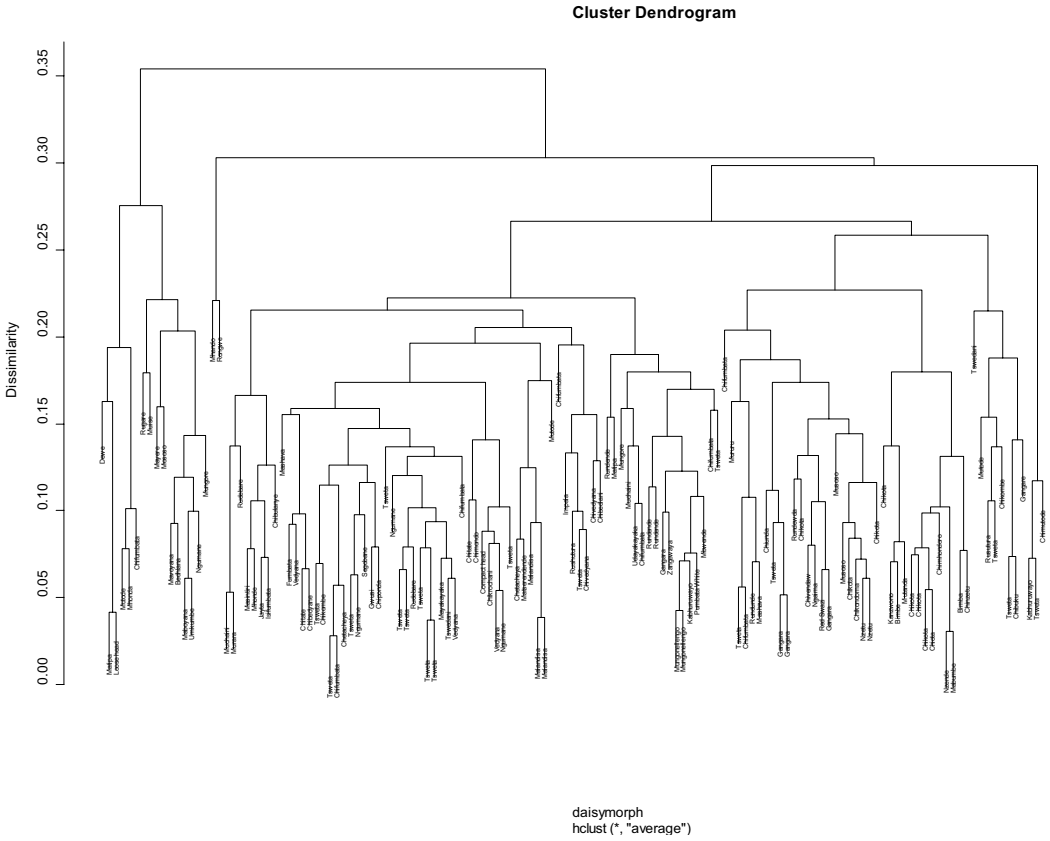


Figure 10. The dendrogram showing farmer variety names.

question with the help of **R**.

The first part of the question is whether the measure of dissimilarity or distance is the most appropriate. Though this can influence your results considerably, we are not going to enter into this discussion here beyond the points we have already made about using measures appropriate to the type of data that you have. A good introductory discussion of this issue and other aspects of the analysis of genetic diversity can be found in Mohammadi and Prasanna (2003) and in more detail in Legendre and Legendre (1998).

The second part of the question relates to how well the distance or dissimilarity matrix is represented graphically. One way of looking at this is that the distances represented on the diagram should be as close as possible to the distances in the distance or dissimilarity matrix. The following test extracts the distances (height) in the diagram using the function `cophenetic()` and then correlates this with the original distance or dissimilarity matrix:

```
cophenmorph<-cophenetic(hclustreord)
```

First download the package `vegan` as described earlier for the package `gclus`, and load the library.

```
library(vegan)
mantel(cophenmorph,daisymorph)
```

The mantel test will give you the correlation between two matrices. By going back and reconstructing the object `hclustmorph` using different methods of clustering or agglomeration, we can compare the ability of these methods to reflect the distances or dissimilarities that we have calculated, e.g.:

```
hclustward<-hclust(daisymorph,method="ward")
hclustreward<-reorder.hclust(hclustward,daisymorph)
cophenreward<-cophenetic(hclustreward)
mantel(cophenreward,daisymorph)
```

When we compare the coefficients obtained from the two mantel tests, we see that in this case the method "average", commonly known as UPGMA (Unweighted Pair Group Method with Arithmetic mean), has by far the highest correlation coefficient, which makes it easy to interpret and probably goes a long way to explaining its popularity. This does not necessarily imply that it correctly addresses a third question, namely "How well does it cluster or agglomerate the accessions?" This is a much more difficult question to answer, as there is no correct result against which to test different methods.

Pruning dendrograms

With larger data sets, dendrograms become unwieldy and can be impossible to display on one page. It can then be useful to show just the upper part of the dendrogram. This can be done with a number of different functions in **R**, but my own preference is the function `clip.clust()` from the package `maptree`. As with vegan, `maptree` is a contributed package and needs to be downloaded and installed prior to use. Since some of the commands in `maptree` depend on functions in the package `combinat`, this package is downloaded automatically with `maptree`.

How do you decide where to prune a hierarchical cluster tree? Kelley *et al*. (1996) proposed a method that can help you decide by calculating a penalty function for a hierarchical cluster tree. This function compares the mean across all clusters with the mean within clusters of the dissimilarity measure.

The **R** function is `kgs()` and also comes in the package `maptree`.

but in addition requires that the package `combinat` is loaded. To calculate the penalty function, type:

```
library(maptree)
kgsmorph<-kgs(hclustmorph,daisymorph,maxclust=50)
plot(names(kgsmorph),kgsmorph)
```

The lowest value shows you the optimum number of clusters, in this case 9 clusters (see Figure 11)[2]. Like with so much in cluster analysis, do not take this as the definitive answer. You may well have other reasons to select a different number, such as a requirement to select a specific number of accessions that represent a collection as well as possible. There are almost certainly other methods that will give you other conclusions. Nonetheless, this is a useful guide.

So we have chosen to cut the dendrogram into 9 clusters (Figure 12):

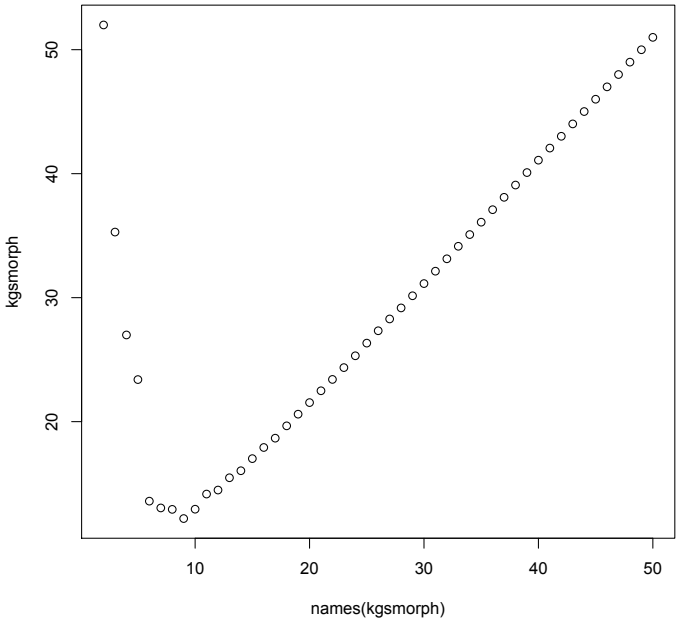


Figure 11. Kelley-Gardner-Sutcliffe penalty function for the sorghum data showing that the optimum number of clusters is nine.

[2] The option '`maxclust=`' sets the maximum number of clusters for which to compute the measure, as it is not useful to calculate it for very high numbers and the calculations take a long time.

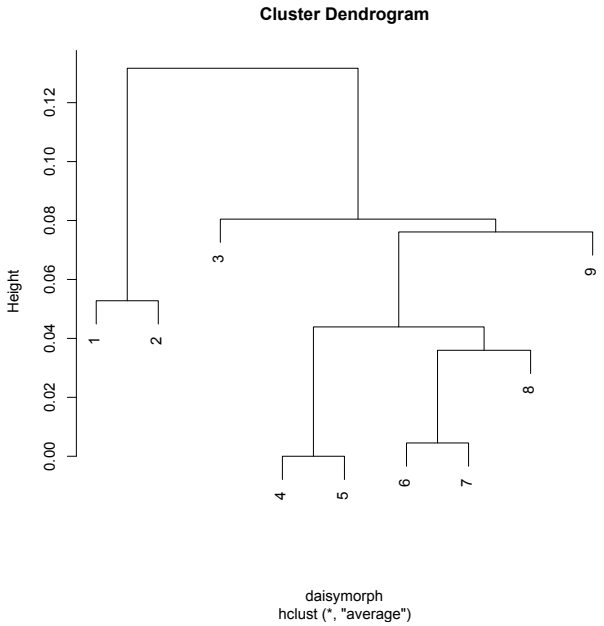


Figure 12. Cluster dendrogram pruned to eight clusters using `clip.clust()`.

```
clipreord<-clip.clust(hclustreord,sorghum,k=9)
plot(clipreord)
```

Now, it would be useful to know how many accessions belong to each of the groups. The following code:

```
groupk9<-group.clust(hclustreord,k=9)
table(groupk9)
```

gives the number of accessions in each group:

```
groupk9
 1  2  3  4  5  6  7  8  9
 6 10  2 55 17 23 13  8  3
```

Typing the object name `groupk9` will give you group membership for each accession in the order that the accessions appear in the dataframe:

```
groupk9
```

The output can be a bit difficult to decipher, so try binding the groups to the original data file so that you can see clearly which accessions belong to which group:

```
sorghum<-cbind(sorghum,groupk9)
str(sorghum)
```

You can see group membership against each accession number with:

```
subset(sorghum,select=c(TGR,groupk9))
```

The `select` statement selects the variables to display. This arrangement of the dataframe now opens up table views that can give a better idea of the properties of each cluster. Using the function `table()`, you can now look at how different qualitative traits are spread over the different clusters, e.g.:

```
table(sorghum$MC,sorghum$groupk9)
```

which will give you:

```
   1   2   3   4   5   6   7   8   9
C  1   0   1  52  17  23  13   8   3
D  4  10   1   3   0   0   0   0   0
Y  1   0   0   0   0   0   0   0   0
```

Try other variables too to look for interesting patterns in the dataset.

Adding molecular markers

This tutorial focuses on the analysis of morphological characterization data, as there are many other guides to the analysis of molecular data. However, there may be times when you would want to do a combined analysis of your morphological and molecular data. The function `daisy()` allows you to do this by defining your molecular variables as asymmetric binary (see the box on `daisy()` above). This is equivalent to the simple matching coefficient, which is frequently used with molecular data.

Contrary to the way that other variables are defined in the dataframe itself, asymmetric binary variables are defined when executing `daisy()`. For example, if the data set `sorghum` had contained data for twenty molecular marker variables called `mm1` to `mm20`, we would have had:

```
morphmm<-subset(sorghum,select=c(PC:KP,mm1:mm20)
  daimorpmm<-
daisy(morphmm,type=list(asymm=c(mm1:mm20)))
```

Advanced clustering

The examples of hierarchical clustering that we have gone through above are in reality only a few of very many different ways of doing clustering. For example, because the size of the dissimilarity matrix rises exponentially with the number of accessions, hierarchical clustering quickly runs into problems with very large datasets (approximately over 500 accessions). **R** offers a very large number of other possibilities, being perhaps best known for the functions in the package `cluster`, described by Struyff *et al*. (1996).

To see what other functions the package contains try typing:

```
library(help=cluster)
```

Another package with a more eclectic range of functions that perform what is called model-based clustering is the package `mclust`, described by Fraley and Raftery (2002).

R is developing so quickly that it is not possible to give an up-to-date overview of all the functions for clustering, but a complete list can be obtained by downloading all the contributed packages and using the HTML help: click 'Help|HTML help', then click on the link 'Search Engine and Keywords' and move down the page to the link 'cluster'.

References

Anderson E. 1935. The irises of the Gaspe Peninsula. *Bulletin of the American Iris Society*, 59: 2–5.

Appa Rao S, Mushonga JN. 1987. A catalogue of passport and characterization data of sorghum, pearl millet and finger millet germplasm from Zimbabwe. IBPGR, Rome, Italy.

Fraley C, Raftery AE. 2002. MCLUST: Software for model-based clustering, density estimation and discriminant analysis. Technical Report, Department of Statistics, University of Washington, Seattle, Washington, USA. Available from: http://www.stat.washington.edu/mclust Date accessed: 19 July 2007.

Gower JC. 1971. A general coefficient of similarity and some of its properties. *Biometrics* 27: 857–871.

Kelley LA, Gardner SP, Sutcliffe MJ. 1996. An automated approach for clustering an ensemble of NMR-derived protein structures into conformationally-related subfamilies. *Protein Engineerin*g, 9: 1063–1065.

Legendre P, Legendre L. 1998. Numerical ecology. 2nd English edition. Elsevier Science BV, Amsterdam, The Netherlands. xv + 853 p.

Mohammadi SA, Prasanna BM. 2003. Analysis of genetic diversity in crop plants – salient statistical tools and considerations. *Crop Science*, 43: 1235–1248.

Struyff A, Hubert M, Rousseeuw PJ. 1996. Clustering in an object-oriented environment. *Journal of Statistical Software* 1(4) [on-line]. Available from: http://www.jstatsoft.org/v01/i04/paper/clus.pdf. Date accessed: 19 July 2007.

Van Hintum ThJL, Brown AHD, Spillane C, Hodgkin T. 2000. Core collections of plant genetic resources. IPGRI *Technical Bulletin* No. 3. Bioversity International, Rome, Italy.

Chapter 3

CHAPTER 3
DIVERSITY INDICES

By Mikkel Grum

The following commands from the previous chapter will get you where you need to be to carry out the exercises in this chapter.

```
sorghum<-read.table("Sorghum_Data
  .txt",header=T,sep="\t")
  sorghum$LD<-ordered(sorghum$LD)
  sorghum$SN<-factor(sorghum$SN)
  sorghum$EN<-ordered(sorghum$EN)
  sorghum$BD<-ordered(sorghum$BD)
  sorghum$PA<-ordered(sorghum$PA)
  sorghum$EX<-ordered(sorghum$EX)
  sorghum$EC<-factor(sorghum$EC)
  sorghum$ET<-factor(sorghum$ET)
  sorghum$KL<-factor(sorghum$KL)
  sorghum$SC<-factor(sorghum$SC)
  sorghum$KP<-factor(sorghum$KP)
morphsorghum<-subset(sorghum,
   select=PC:KP)
library(cluster)
daisymorph<-daisy(morphsorghum)
hclustmorph<-hclust(daisymorph,
   method="average")
library(gclus)
hclustreord<-reorder.hclust(hclust
   morph,daisymorph)
library(maptree)
groupk9<-group.
clust(hclustreord,k=9)
sorghum<-cbind(sorghum,groupk9)
```

Ready!

Richness, Shannon-Weaver and Simpson

In ecological research, a number of indices are often used to measure species diversity in different communities. After 'richness', which is simply the number of species represented in the community, the two most common indices are probably the Shannon-Weaver index and the Simpson index.

The Shannon-Weaver index combines a measure of richness with a measure of evenness. Higher values indicate more diversity. The Simpson index is between zero and one. It measures evenness of species or group membership. It can also be interpreted as the likelihood of two randomly selected accessions being different from each other.

When looking at intra-specific diversity we could use morphological clusters instead of species. Magurran (1998) and Kindt and Coe (2005) provide more detailed descriptions of diversity indices and are good places to continue for the reader who is interested in working further with the measurement of biodiversity.

The output table from the command:

```
provk9<-table(sorghum$PROV,sorghum$groupk9)
provk9
```

gives us the number of accessions collected from each group in each province. The columns represent the groups and the rows the provinces:

```
     1  2  3  4  5  6  7  8  9
MDL  0  7  0 19  2  4  0  5  1
MLC  2  1  0  2  3  3  0  0  1
MLN  0  0  0  8  1  1  0  0  0
MLW  0  0  1  4  0  4  4  0  0
MNL  1  1  0  3  3  7  9  1  1
MSV  3  1  1 19  8  4  0  2  0
```

This type of table, with the numbers or frequencies of each species or group membership for each site or area, is known as a community data matrix. The matrix can be transposed to give a view that is more easily compared to the plots that follow using the command:

```
t(provk9)
```

Richness, defined here as the number of groups represented in each province, can be calculated using:

```
specnumber(provk9)
```

with the results:

```
MDL MLC MLN MLW MNL MSV
  6   6   3   4   8   7
```

We can get to see the code behind these calculations by typing:

```
specnumber
```

which gives us:

```
function (x, MARGIN = 1)
{
```

```
    apply(x > 0, MARGIN, sum)
}
```

The function simply applies the function `sum` along the '`margin 1`', i.e. in our example, it calculates the sum of varieties found (more than 0 times) in each province in the table `provk9`.

For comparison, the number of accessions collected in each province can be obtained using:

```
table(sorghum$PROV)
MDL  MLC  MLN  MLW  MNL  MSV
 38   12   10   13   26   38
```

The package `vegan` has a function, `diversity()` that can calculate both the Shannon-Weaver index and the Simpson index on a community data matrix. The default is the Shannon-Weaver index (Figure 13):

```
library(vegan)
swi<-diversity(provk9)
barplot(swi)
```

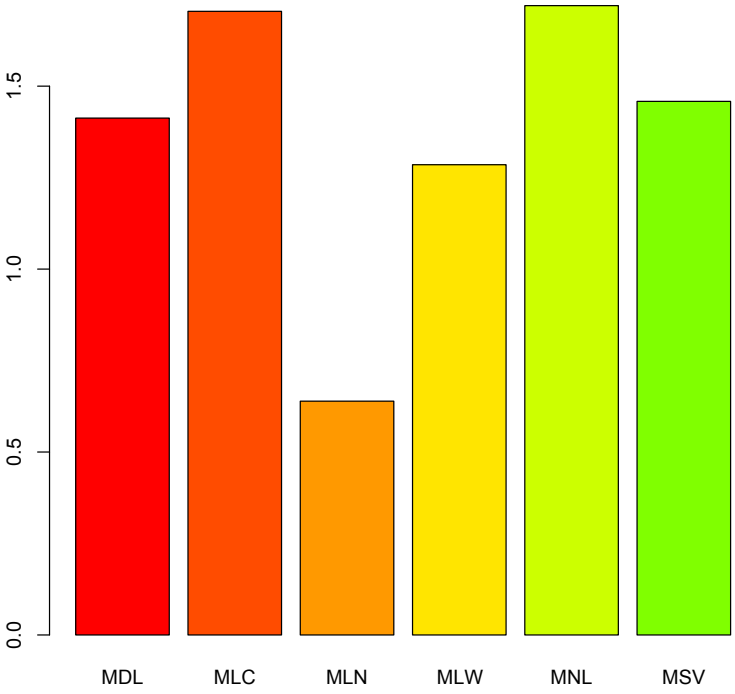


Figure 13. Shannon-Weaver index for intra-specific sorghum diversity based on nine morphological groups.

Add colours using the following commands:

```
barplot(swi,col=rainbow(20))
barplot(swi,col="lightblue")
```

The first command will give your bars a range of colours from the rainbow, and the second will make your bars light blue.

To get the Simpson index, you need to specify it:

```
simp<-diversity(provk9,index="simpson")
barplot(simp)
```

You can get the correlation coefficient between the Shannon-Weaver and Simpson indices with:

```
cor(swi,simp)
```

What is the correlation coefficient between the two if you calculate them for each town, rather than province? What does this tell you about the two indices?

Rolling your own function

If your objective were to evaluate a town's contribution to Zimbabwe's conservation effort, then the richness, diversity and evenness measures above do not take account of how common the clusters are in Zimbabwe as a whole.

Since I could not find any function in any software that tackled this question to my satisfaction, I developed my own index. This is an index that weights each cluster for rareness, by dividing the number of observations of that cluster in the province (or town, farm, or whatever other unit you might want to do the analysis on) with the total number of observations of that cluster in your study area (in this case Zimbabwe). The square root of the value[3] for each town is then summed across all varieties as in:

$$CI = \sum^{n} \sqrt{\frac{a_{ij}}{A_i}}$$

where

a_{ij} is the number of observations of variety i in province j, and
A_i is the total number of observations of variety i in the study area.

We write this as a function with the following script:

[3] The square root of the value is used, because this gives an almost linear correlation with richness in most cases and richness is the diversity measure that most people feel most comfortable with.

```
ci <- function(x, MARGIN = 2)
{
  x <- as.matrix(x)
  Ai <- apply(x, MARGIN, sum)
  qq <- function (aij)
    {
    sqrt(aij/Ai)
  }
  Q <- apply(x, 1, qq)
  P <- apply(Q, 2, sum)
  P
}
```

Once the function is created by running the lines above, it can be used to calculate our new index in the same way that the diversity function was used.

```
ci(provk9)
```

Plotting these values against richness using

```
plot(specnumber(provk9),ci(provk9),
  type="n")
text(specnumber(provk9),ci(provk9),row.
  names(provk9))
```

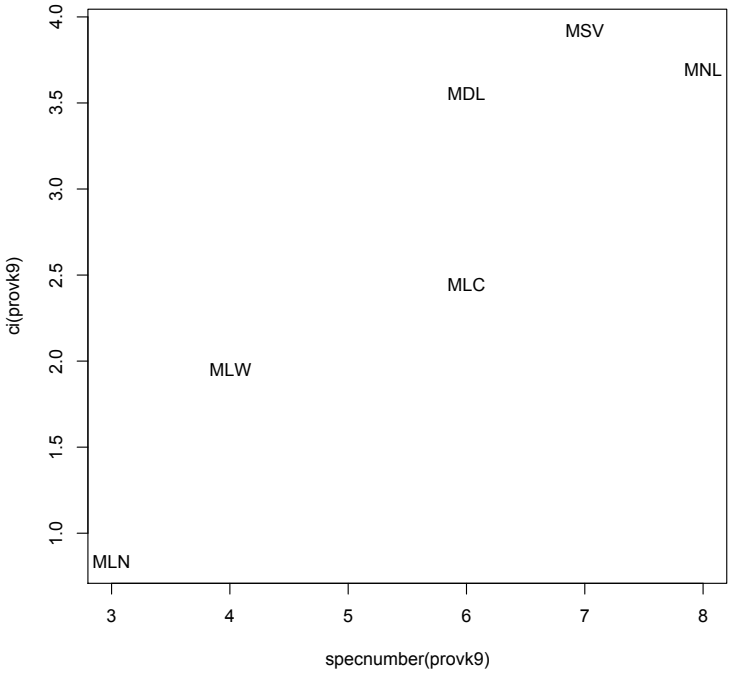


Figure 14. Plot of CI scores against richness in each province in Zimbabwe.

we see that although both MDL and MLC have six clusters each, MDL scores higher on the CI scale, because the clusters in the province are rarer and therefore contribute more to the overall level of sorghum diversity in Zimbabwe. This would seem like valuable information in prioritizing conservation areas!

For code comparison and a better understanding of how you might construct your own function, you might want to take a look at the code for the diversity() function with:

```
diversity
```

Frequent use of the possibility of seeing the code used in functions is one of the best ways of learning how to make your own functions.

Dissimilarity measures for estimating diversity

Most of these indices do not take genetic distances into account, so two closely related groups are considered as diverse as two relatively unrelated groups. This is a fairly acceptable approach when used for species diversity, because barriers to gene flow among species result in fairly discrete units. However, when used for the analysis of intra-specific diversity the assumption of distinct classes within species is less well-founded, partly because there may be quite significant gene flow within species and partly because of difficulties in fixing criteria for intra-specific classification. Therefore a better approach may be to make direct use of the dissimilarity measures to obtain estimates of diversity.

One analysis in **R** that gives a useful picture of diversity across a number of units is the function `anosim()` in the package `vegan`. This analysis ranks all the dissimilarities among accessions and produces a boxplot of the ranks of dissimilarities within a given unit, e.g. provinces. The commands:

```
anomorph<-anosim(daisymorph,sorghum$PROV)
plot(anomorph)
```

will give the plot in Figure 15. This plot is very informative, although its interpretation can be a bit challenging. The boxes show the minimum, first quartile, median, third quartile and maximum *rank* of the dissimilarities within each province, e.g. the median dissimilarity within the Manicaland Province (MNL) is ranked 3837th and the highest dissimilarity is ranked 9295th of the total 9316 dissimilarity values in the matrix. The boxes are drawn with widths proportional to the square-roots of the number of observations in the groups.

One observation that we make in Figure 15 is that the median dissimilarity rank for MLC is substantially higher than any of the

others, indicating that diversity may in fact be high in MLC. With fewer accessions collected, 12 accessions in MLC against 26 accessions in MNL, the lower Shannon-Weaver index for MLC might simply be a result of a small sample size.

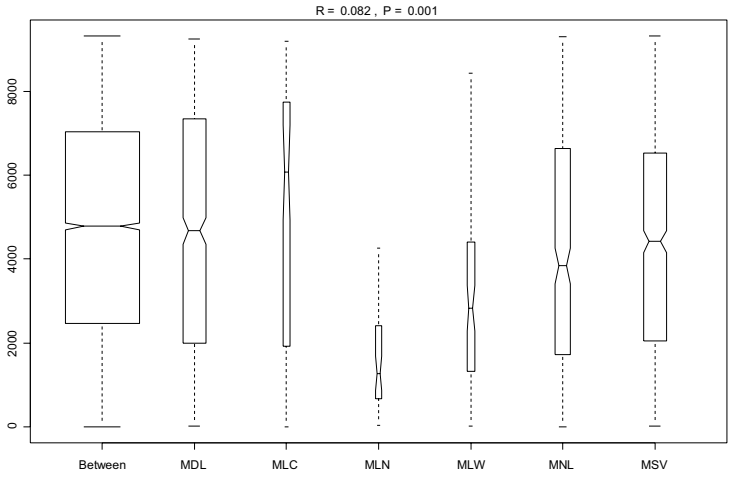


Figure 15. Boxplot of the rank order of dissimilarities by province.

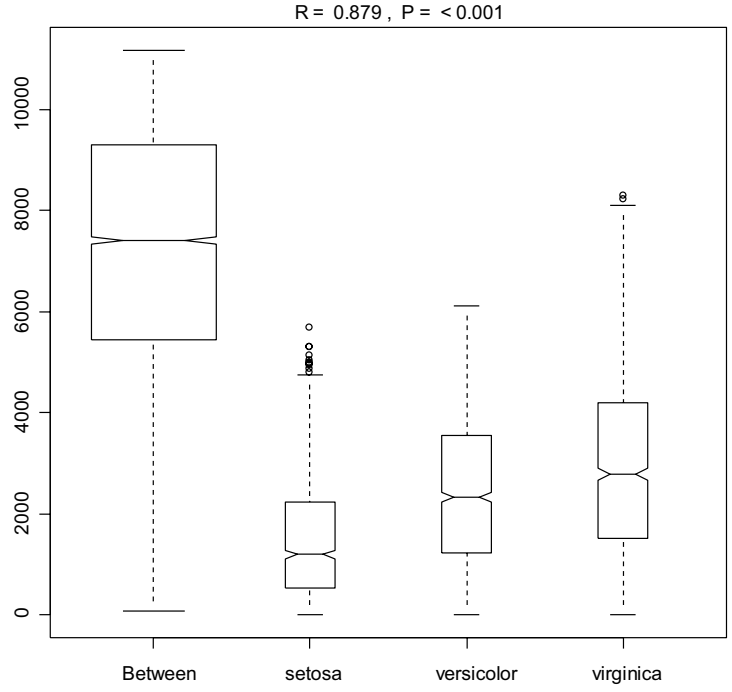


Figure 16. Boxplot of the rank order of dissimilarities by species in Anderson's iris dataset.

Another quick example shows how different the results can be:

```
data(iris)
distiris<-dist(iris[,1:4])
anoiris<-anosim(distiris,iris$Species)
plot(anoiris)
```

In this example, Figure 16 shows, not surprisingly, that most of the dissimilarity is between species, with much less dissimilarity within species. Most of the samples of the species *Iris setosa* are very similar to each other, but there are a few outliers that are morphologically distinct from the rest of the samples of the species. Their higher dissimilarity is shown as dots above the whiskers of the box.

Summary

In this chapter we extended the work from the previous chapter to look at how we could use some of the common diversity indices with intra-specific diversity based on morphological data. We briefly looked at how we might develop our own function, in this case to take account of each region's contribution to overall diversity in the study area. We also looked at how we could deal with the continuous nature of intra-specific diversity, which rarely exists in discrete units.

This tutorial has been a stepwise guide through the analysis of morphological characterization data and the specific problems that their mathematical properties present. I have seen users with no experience of **R** go through these steps one line at a time, modify the code to deal with their own data and get the results that they want. At the same time, this is not a complete introduction to **R**, and those users that would like to move on to other analyses will need additional references. The software **R** comes with a beginners manual entitled "An introduction to R" that can be found in the menu under "Help|Manuals (in PDF)". Many other references can be found on the **R** Project website: www.r-project.org/.

References

Magurran AE. 1988. *Ecological diversity and its measurement*. Croom Helm, London, UK.

Kindt R, Coe R. 2005. Tree diversity analysis: a manual and software for common statistical methods for ecological and biodiversity studies. ICRAF, Nairobi, Kenya. 203 p.

Chapter 4

CHAPTER 4 TROUBLESHOOTING

This section lists some of the common problems and error messages we came across in using this tutorial. For other error messages encountered when using **R**, the best option is to use any one of the search engines found on the **R** project website for help (http://www.r-project.org/).

1. “Error: Object " . . . " not found”

You have tried to execute a function without first loading the library or package that contains the function. Solution: load the library or package.

2. I can’t read the text in the cluster diagrams; it’s too small and cramped.

Have you maximized the graphics window? Double-clicking on the blue bar across the top of the graphics window will do this. For printing from **R**, you can set the page to landscape by clicking on the properties button in the print dialogue box and selecting ’landscape‘. You can also use the option `cex=0.5` or another number to reduce the size of the font.

3. Capitalization

R is sensitive to capitalization; if the variable is called `iris$Species`, then `iris$species` will not work. If options are capitalized incorrectly, e.g. writing `method=Average`, or `method=Ward`, the default method is used. In both the examples used, the method should have been written without capital letters, so the default `method=complete` is used instead. No warning is given.

4. “Error in file(file, "r") : unable to open connection”

You forgot to set the working directory. Clicking ’File|Change dir…|Browse‘ from the menu will lead you through this (see Figure 8).

5. My data won't import (a)

Serves you right! You are using Excel and did not believe me when I said that you should cut and paste your data, and only the data, to a new spreadsheet before saving it as a tab-delimited file.

6. My data won't import (b)

The only other problem I know of that messes up importing of a simple data set of rows and columns is when you have apostrophes in some of your variables, e.g. names like n'goni. As far as I know, you have to remove the apostrophes before importing the data. Accents and most other symbols present no problem.

IPGRI and INIBAP
operate under the name
Bioversity International

Supported by the CGIAR

ISBN 978-92-9043-735-2