From biotic records to analysis
Introduction to the mefa package

Péter Sólymos *

June 13, 2008

Abstract

This document provide some insights on the general data model, which can be used effectively in biotic data handling. Than, using a real world example, it gives an overview of S3 object classes and methods provided by the `mefa` package. Reporting options and possible ways of further data analysis are also briefly discussed. This vignette can be used as reerence for the `dolina` demo of the package.

Contents

1 What is `mefa`?

Data resulting from field inventories are often stored primarily in a hard copy notebook format, which are subsequently typed into a spreadsheet. These spreadsheets contain location (samples) and taxon (species) specific information and respective count data. These data can rarely be used directly in a statistical analysis, because one often need sample and species specific attributes to account for a given question or hypothesis.

The manipulation of the data (most commonly subseting and ordering) is time consuming and may lead to mistakes, and mistakes may lead to false conclusions. Thus it is advisable to rigorously check each step in such manipulation

*Péter Sólymos, Department of Ecology, Faculty of Veterinary Science, Szent István University, Rottenbiller Str. 50, 1077 Budapest, Hungary. e-mail: Solymos.Peter@aotk.szie.hu.

process. These problems can be avoided by using sophisticated database servers and database connections prior to analysis, but such instruments need IT skills that are sometimes out of the capabilities of an individual researcher or naturalist. If database connections are used, `mefa` objects may be useful, eg. for analysing different layers, or segments of the results of a query.

Thus, the name `mefa` stands for "metafaunistics", indicating that handling of basic data is only the first, but the most critical and sometimes most time consuming part of data analysis. The aim of the `mefa` package is to give a solution in `R` for the specific requirements needed for biotic data handling in ecology and biogeography, shorten the time spent with data preparation and reduce mistakes during data management. Here I give a practical guide for installing `mefa`, a short overview of the general data model behind `mefa` objects, main functions of the package, reporting options, and examples for further data analysis.

2 How to get `mefa`?

Once you have a working `R` on your machine (if not, go to http://www.r-project.org), you can install the stable version from `CRAN` via the menu of the `R` GUI (Packages/Install package(s)...), or typing `install.packages("mefa")`.

After choosing a repository site, `R` starts to install the package. `CRAN`, however, contains the latest *stable* version of `mefa`. The rather exparimental, developement verion of the package (Nightly Package Snapshot) can be accessed from https://r-forge.r-project.org/projects/mefa/ clicking on the download icon, or typing `install.packages("mefa", repos="http://r-forge.r-project.org")`

The project's Subversion (SVN) repository tree can be checked out through anonymous access with the command `svn checkout svn://svn.r-forge.r-project.org/svnroot/mefa`. For further references (updates, mailing lists, forums, RSS) wisit the site http://mefa.r-forge.r-project.org/

3 How to use `mefa`?

3.1 Data model

General database representation of biotic (faunistic) data may include four modules according to Samu[1]: the project, sample, taxon, and results modules. The project module describes meta-data and background circumstances of a given field experiment. Sample module describes information relevant to the data collecting event and variables that can vary relative to the fixed project design. The taxon module deals with the taxonomic identity of the biological objects. The quantitative outcome of the study is handled in the results module. For a given analysis, only a subset of these modules are necessary, but in an organised format.

The sample and taxon specific attributes, stored in the sample (sometimes in the project) and taxon modules can be arranges into a hierarchy. Eg. samples, nested into layers, layers nested into sites, sites nested into regions, regions nested into continents, and finally, continents nested into the Earth itself. And so, individuals nested into populations, populations nested into species, species

nested into some higher level taxonomic entities or functional groups, are all relevant, and thus might be sored in the background database, or in spreadsheets.

A `mefa` object is a project-oriented representation (subset) of the general database that can be used in subsequent analyses, were the data stored in the results module determines subsets taken from the sample and taxon modules based on the relational links provided by sample and taxon specific identifiers. Project module may also contain information for manipulating sample and results modules in the pre-processing phase (eg. subseting results or combining sample attributes). The concepts of `attribue hierarchy` and `attribue duality` and the use of segments are central to the handling of data via `mefa` objects.

3.2 Workflow in `mefa`

First, we have to load the installed package:

```
> library(mefa)

This is mefa 1.1-2
```

At this point you might want to use the demo by typing `demo(dolina)`.

A basic table of count data resembles a sheet in a notebook, where one takes notes during the identification process, like in the `dolina` example data set (only first ten rows for brevity):

```
> data(dol.count)
> str(dol.count)

'data.frame':        125 obs. of  4 variables:
 $ sample : Factor w/ 16 levels "1A1","1A2","1A3",..: 1 2 NA 3 NA NA NA NA NA NA ...
 $ species: Factor w/ 20 levels "amin","apur",..: 20 1 17 1 NA NA 10 16 17 18 ...
 $ segment: Factor w/ 4 levels "adult","broken",..: 1 3 2 1 3 4 3 3 1 3 ...
 $ count  : int  1 1 1 1 3 1 1 1 1 1 ...

> dol.count[1:10, ]

   sample    species  segment count
1     1A1 zero.count    adult     1
2     1A2       amin    fresh     1
3    <NA>       pvic   broken     1
4     1A3       amin    adult     1
5    <NA>       <NA>    fresh     3
6    <NA>       <NA> juvenile     1
7    <NA>       dper    fresh     1
8    <NA>       pinc    fresh     1
9    <NA>       pvic    adult     1
10   <NA>       tuni    fresh     1
```

This data frame contanis four columns. Samples are the samples taken in the field. Heere we used five minutes search for snails in a one meter radius. Sample identifiers contain two numbers and a letter. First (numeric) character indicates sampling location, second (nonnumeric) character stands for the

investigeted microhabitat (A: litter, H: coarse woody debris, L: living trees, R: rock), third (numeric) character indicates the replicate within microhabitats as strata. Eg. `1A1` is the sample taken from the first location (one out of the studied dolinas), it is a litter microhabitat, and the first replicate out of seven, etc.

Species names are short identifiers (discussed later), segment refers to life stages of the individuals found (`broken`: broken shells of dead animals, `fresh`: intact shells of dead animals, `adult`: adult live animal, `juvenile`: juvenile live animal). Count column indicates the number of specimens that can be characterised by the information in the rorresponding row.

What are `<NA>`-s? This data frame contains sample and taxonomic information in shortened way, sample and species identifiers are shown only for the first cases for easier input. This "notebook-style" data can be filled with respective data by the function `sscount` (see also helper function `fill.count`):

```
> (ssc <- sscount(dol.count, zc = "zero.count", fill = TRUE))

Object of class 'sscount'
Call: sscount(sstable = dol.count, zc = "zero.count", fill = TRUE)
Data type: count
Number of samples:  16
Number of species:  19
Zero count identifier:  zero.count
Number of segments: 4  with levels:
[1] "adult"    "broken"   "fresh"    "juvenile"
```

In this way, "notebook-style" data can be readily converted into an `sscount` (Species/Sample/COUNT) object, with data filled up (first 10 rows):

```
> ssc$data[1:10, ]

   sample    species  segment count
1     1A1 zero.count     <NA>     1
2     1A2       amin    fresh     1
3     1A2       pvic   broken     1
4     1A3       amin    adult     1
5     1A3       amin    fresh     3
6     1A3       amin juvenile     1
7     1A3       dper    fresh     1
8     1A3       pinc    fresh     1
9     1A3       pvic    adult     1
10    1A3       tuni    fresh     1
```

The `zc` flag is important, because it contains information on the (arbitrarily named) "pseudo-species" used in `dol.count` object referring to samples where total count was zero, like in the first row (sample `1A1`). Default for `zc` is `NULL`.

In this `sscount` object, rows represent the basic units (lots) containing sample, species and segment identifiers (character) and the count (numeric). First three columns are treated as factors. Basically, count should be integer, but when the result of an experiment is not an integer vaue (eg. biomass, as a measurement variable), decimal numeric values can be used by specifying the `digits` argument in the `sscount` function (see manual for further references).

`ssc$data` contains the data taken from the filled data frame of `dol.count`. Cross-tabulation of an `sscount` object results in an `xcount` object. Crosstabulation can be done for segments, listed by

```
> ssc$segment.levels
```

```
[1] "adult"    "broken"   "fresh"    "juvenile"
```

or for all segments depending on the value of the segment specifier (default is `segment = "all"` (or equivalently `segment = 0`), meaning all the segments are summed up within lots). A list of segments can also be specified (eg. textttc("adult", "juvenile")). For all segments, simply use:

```
> (xc.all <- xcount(ssc))
```

```
Object of class 'xcount'
Call: xcount(ssc = ssc)
Data type: count
Segment: all
Number of samples:  16
Number of species:  19
Total count : 208
Matrix fill:  0.299 with ( 91  presences)
Samples with zero total count:
[1] "1A1"
```

For broken shells only:

```
> (xc.broken <- xcount(ssc, 2))
```

```
Object of class 'xcount'
Call: xcount(ssc = ssc, segment = 2)
Data type: count
Segment: broken
Number of samples:  16
Number of species:  19
Total count : 108
Matrix fill:  0.164 with ( 50  presences)
Samples with zero total count:
[1] "1A1" "1A7"
Species with zero total count:
[1] "ctri" "druf" "mbor"
```

The pseudo-species for zero count samples is removed from the resulting `xcount` object, but an empty row is placed in the table for sample `1A1`. Empty rows of an `xcount` object can be removed by the function `exclmf`, detailed later.

Data tables can also be converted into `sscount` or `xcount` objects by using the functions `ttsscount` (data without unique row identifiers, use `drtsscount`) or `as.xcount` respectively (for details, see manual). Two `sscount` or `xcount` objects can be merged by functions `msscount` and `mxcount` respectively (for details, see manual).

Plot method is available for `xcount` objects (Figure 1), plotting histograms by default:

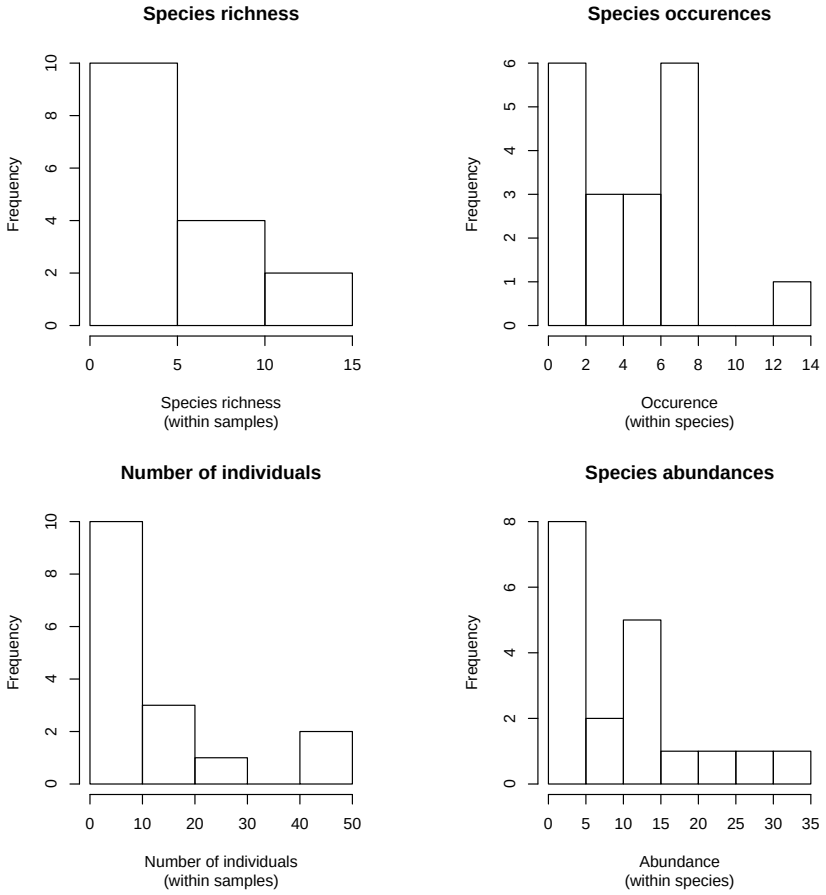

Figure 1: Plotting method for `xcount` objects showing histograms.

```
> plot(xc.all)
```

But `type` can be set to "rank" (Figure 2) or "biplot" as well (for "biplot", see manual):

```
> plot(xc.all, type = "rank", logscale = TRUE)
```

When we have the faunistic data, that is half way of a data inicialization. To have attributes of the samples and the taxa with the same consistency is also desirable. These attributes often are very basic. Eg. for samples, the sample identifier, location, date of collection and name of collector (basic biotic data):

```
> data(dol.sample)
> str(dol.sample)
```

```
'data.frame':        16 obs. of  7 variables:
 $ sample             : Factor w/ 16 levels "1A1","1A2","1A3",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ microhabitat       : Factor w/ 4 levels "dead.wood","litter",..: 2 2 2 2 2 2 2 1 1 1 ..
 $ replicate          : int  1 2 3 4 5 6 7 1 2 3 ...
```

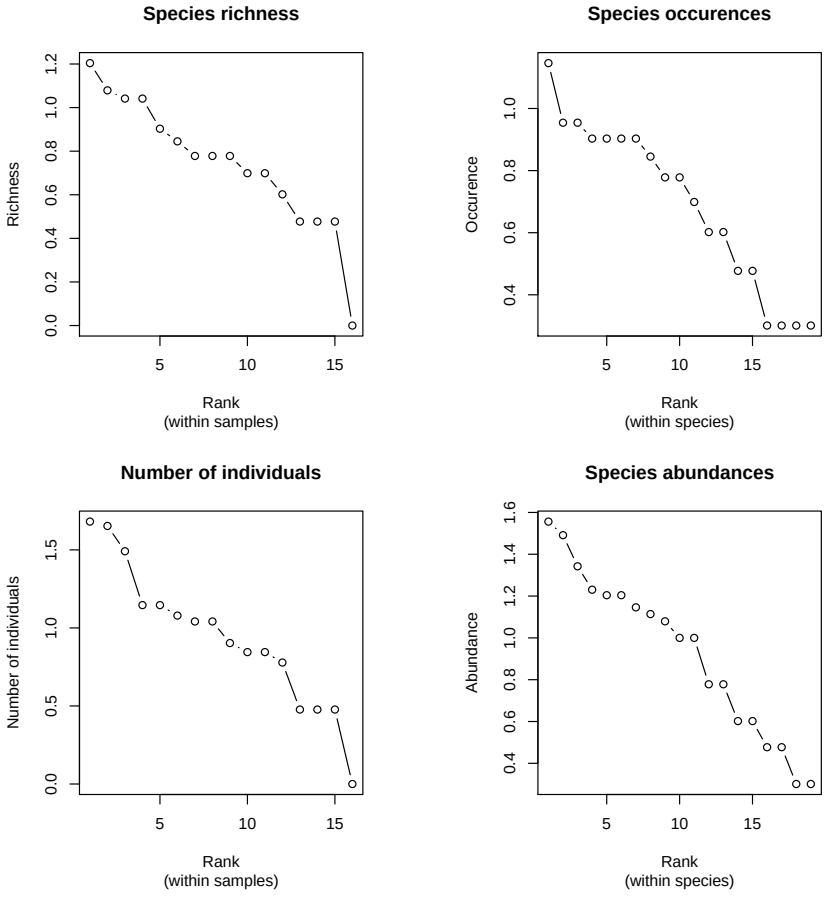

Figure 2: Plotting method for `xcount` objects showing rank-abundance and -occurrence curves.

```
 $ aspect             : Factor w/ 5 levels "eastern","flat",..: 4 4 4 2 3 3 3 4 5 2 ...
 $ depth              : Factor w/ 4 levels "bottom","edge",..: 4 2 3 1 3 2 4 3 3 1 ...
 $ litter.moisture    : num  2 2 2.5 2 2 1 1.5 2 2 1.5 ...
 $ litter.thickness.cm: num  3 3 4 5 3 2 5 4 12 5 ...
```

Once we have the crosstabulated data (`xc`) and attributes for samples and species, we have to checks the compatibility of the sample attribute table and a given `xcount` object:

```
> check.attrib(xc.all, which = "samples", dol.sample, 1)

$call
check.attrib(xc = xc.all, which = "samples", attrib = dol.sample,
    index = 1)

$set.relation
[1] "equal"
```

```
$duplicate
NULL

$missing
NULL

$na
[1] 0
```

If check results indicate full match of sample identifiers (without duplicates and missing elements), an `xorder` object can be created from an attribute table:

```
> (xo1 <- xorder(xc.all, which = "samples", dol.sample, 1))

Object of class 'xorder'
Call: xorder(xc = xc.all, which = "samples", attrib = dol.sample, index = 1)
Attribute table for samples
Match of 'xcount' object and attribute table:  equal
Attribute table contains  7 variables for 16 samples
NA values were not found in the table
```

And this can be done for the species attributes as well:

```
> data(landsnail)
> str(landsnail)

'data.frame':        121 obs. of  8 variables:
 $ order          : int  1 2 3 4 5 6 7 8 9 10 ...
 $ spec.short     : Factor w/ 121 levels "aacu","aarb",..: 6 3 75 80 30 38 90 89 67 28 ...
 $ spec.name      : Factor w/ 121 levels "Acanthinula aculeata",..: 3 2 81 82 16 17 92 93 
 $ author         : Factor w/ 65 levels "(A. Schmidt, 1853)",..: 22 45 36 15 63 40 28 59 4
 $ shell.dimension: num  3.4 5.5 16 16 2.2 2.3 17 8 12 7.5 ...
 $ familia        : Factor w/ 21 levels "Aciculidae","Bradybaenidae",..: 1 1 14 14 6 6 17 
 $ subfamilia     : Factor w/ 31 levels "Acanthinulinae",..: 2 2 23 23 11 11 26 26 26 9 ..
 $ genus          : Factor w/ 57 levels "Acanthinula",..: 2 2 40 40 8 8 48 48 36 14 ...
```

And again, compatibility check:

```
> check.attrib(xc.all, which = "species", landsnail, 2)

$call
check.attrib(xc = xc.all, which = "species", attrib = landsnail,
    index = 2)

$set.relation
[1] "inclusion"

$duplicate
NULL

$missing
```

```
NULL

$na
[1] 5
```

Here, set relation value is `"inclusion"`, that means the original species attribute table `landsnail` contained more cases than the object `xc.all`. To get the object of `xorder` for species attributes:

```
> (xo2 <- xorder(xc.all, which = "species", landsnail, 2))

Object of class 'xorder'
Call: xorder(xc = xc.all, which = "species", attrib = landsnail, index = 2)
Attribute table for species
Match of 'xcount' object and attribute table:  inclusion
Attribute table contains  8 variables for 19 species
Number of NA values in the table:  1
```

In these `xorder` objects (`xo1` and `xo2`), original data are subsetted and ordered according to the row/column names of the `xcount` (`xc`) object. A `mefa` object is a bundle of an `xcount` and one or two `xorder` (one for sample and one for species attributes, one can be `NULL`) objects:

```
> (mf <- mefa(xc.all, xo1, xo2))

Object of class 'mefa'
Call: mefa(xc = xc.all, xorder.samples = xo1, xorder.species = xo2)
Data type: count
Segment: all
Number of samples:  16
Number of species:  19
Total count : 208
Matrix fill:  0.299 with ( 91  presences)
Samples with zero total count:
[1] "1A1"
Both attribute tables are attached:
             check.setrel variables na
sample.attr  "equal"      "7"       "0"
species.attr "inclusion"  "8"       "1"
Variables in the sample attribute table:
[1] "sample"              "microhabitat"        "replicate"
[4] "aspect"              "depth"               "litter.moisture"
[7] "litter.thickness.cm"
Variables in the species attribute table:
[1] "order"           "spec.short"      "spec.name"       "author"
[5] "shell.dimension" "familia"         "subfamilia"      "genus"
```

As you can see, the `mefa` object contains all necessary information for reporting and further analysis. Additional attributes can be added later to a `mefa` object by the function `add.attrib` (for details, see manual).

Plotting method is also available for `mefa` objects (Figure 3):

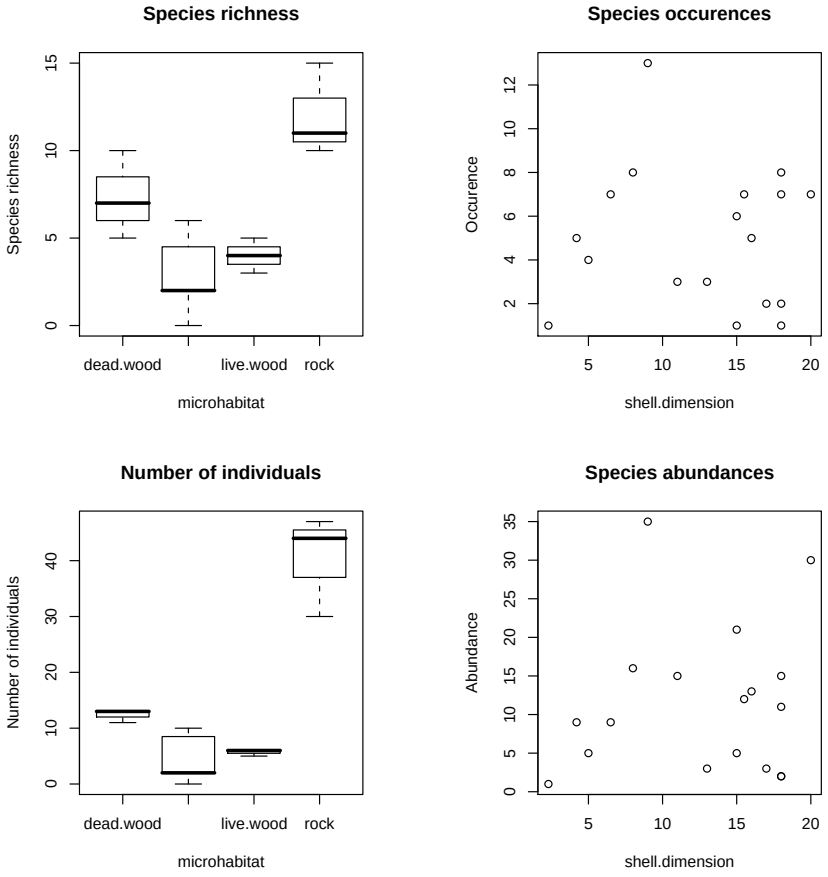

Figure 3: Plotting method for `mefa` objects showing scatterplots or boxplots for variables in the attribute tables.

```
> plot(mf, "microhabitat", "shell.dimension")
```

The `mefa` object (and `xcount` as well) can be stratified according to sample/species groups, eg. in case of hierarchical sampling design:

```
> (mic <- strify(mf, "microhabitat", "samples"))
```

```
Object of class 'mefa'
Call: strify(xc = mf, strata = "microhabitat", which = "samples")
Data type: count
Segment: all
Number of samples:  4
Number of species:  19
Total count : 208
Matrix fill:  0.632 with ( 48  presences)
Only species attribute table is attached:
check.setrel    variables           na
```

```
     "equal"          "8"          "1"
Variables in the species attribute table:
[1] "order"           "spec.short"      "spec.name"       "author"
[5] "shell.dimension" "familia"         "subfamilia"      "genus"
```

The resultant `mefa` object contains the original species attribute table, while its sample attribute table became `NULL`.

Or one can use taxonomic or trophic groups of species for stratification similarly:

```
> (fam <- strify(mf, "familia", "species"))

Object of class 'mefa'
Call: strify(xc = mf, strata = "familia", which = "species")
Data type: count
Segment: all
Number of samples:  16
Number of species:  5
Total count : 208
Matrix fill:  0.55 with ( 44  presences)
Samples with zero total count:
[1] "1A1"
Only sample attribute table is attached:
check.setrel    variables           na
     "equal"          "7"          "0"
Variables in the sample attribute table:
[1] "sample"              "microhabitat"        "replicate"
[4] "aspect"              "depth"               "litter.moisture"
[7] "litter.thickness.cm"
```

or both:

```
> (mic.fam <- strify(as.xcount(mic), mf$species.attr$familia, "species"))

Object of class 'xcount'
Call: strify(xc = as.xcount(mic), strata = mf$species.attr$familia,
    which = "species")
Data type: count
Segment: all
Number of samples:  4
Number of species:  5
Total count : 208
Matrix fill:  0.85 with ( 17  presences)

> mic.fam$data

          Clausiliidae Ellobiidae Endodontidae Helicidae Zonitidae
dead.wood           10          0            2        13        12
litter               1          0            2        16        14
live.wood            2          0            2         4         9
rock                17          1            3        85        15
```

Note, that object `mic.fam` is no more a `mefa` object, but of class `xcount`.

Parts of a `mefa` (or `xcount`) object can be excluded in this way:

```
> (ex1 <- exclmf(mf, which = "samples", empty = TRUE, excl = which(mf$sample.attr$microhab
+     "litter")))

Object of class 'mefa'
Call: mefa(xc = xc.out, xorder.samples = xorder(xc.out, "samples",
    xc$sample.attr), xorder.species = xorder(xc.out, "species",
    xc$species.attr))
Data type: count
Segment: all
Number of samples:  7
Number of species:  9
Total count : 33
Matrix fill:  0.333 with ( 21  presences)
Samples with zero total count:
[1] "1A1"
Both attribute tables are attached:
             check.setrel variables na
sample.attr  "inclusion"  "7"       "0"
species.attr "inclusion"  "8"       "0"
Variables in the sample attribute table:
[1] "sample"              "microhabitat"        "replicate"
[4] "aspect"              "depth"               "litter.moisture"
[7] "litter.thickness.cm"
Variables in the species attribute table:
[1] "order"           "spec.short"      "spec.name"       "author"
[5] "shell.dimension" "familia"         "subfamilia"      "genus"
```

We get samples of the litter microhabitat. Since these were collected along a north-to-south transect (meaning southern-to-northern aspect), we can have a look at the transect (Figure 4):

```
> plot(ex1, "replicate", type = "b")
```

We might exclude some families with low count numbers:

```
> (ex2 <- exclmf(mic.fam, "species", c("Ellobiidae", "Endodontidae"),
+     empty = TRUE))

Object of class 'xcount'
Call: as.xcount(table = matr.ex, segment = xc$segment)
Data type: count
Segment: all
Number of samples:  4
Number of species:  3
Total count : 198
Matrix fill:  1 with ( 12  presences)
```

In the resulting mefa object, respective parts will be excluded for both the crosstabulation and attribute table. Empty rows or columns can be removed by specifying the `empty` argument (see manual). In the end of this subsection, let's have look into the internal structure of a `mefa` object:

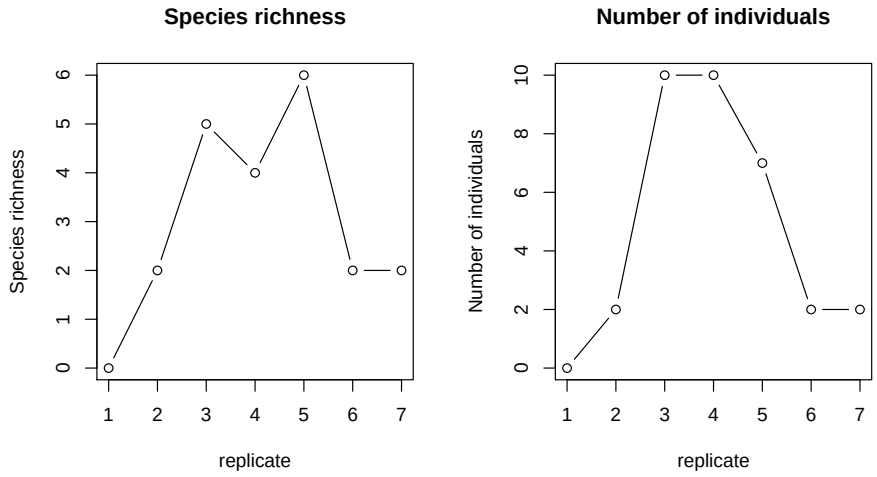

Figure 4: Plotting method for `mefa` objects showing scatterplots or boxplots for variables in the attribute tables.

```
> str(ex2)

List of 12
 $ data      : int [1:4, 1:3] 10 1 2 17 13 16 4 85 12 14 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:4] "dead.wood" "litter" "live.wood" "rock"
  .. ..$ : chr [1:3] "Clausiliidae" "Helicidae" "Zonitidae"
 $ call      : language as.xcount(table = matr.ex, segment = xc$segment)
 $ segment   : chr "all"
 $ digits    : NULL
 $ nsamples  : int 4
 $ nspecies  : int 3
 $ totalcount: int 198
 $ presences : int 12
 $ ninds     : Named int [1:4] 35 31 15 117
  ..- attr(*, "names")= chr [1:4] "dead.wood" "litter" "live.wood" "rock"
 $ srichn    : Named int [1:4] 3 3 3 3
```

```
  ..- attr(*, "names")= chr [1:4] "dead.wood" "litter" "live.wood" "rock"
 $ specabund : Named int [1:3] 30 118 50
  ..- attr(*, "names")= chr [1:3] "Clausiliidae" "Helicidae" "Zonitidae"
 $ specoccur : Named int [1:3] 4 4 4
  ..- attr(*, "names")= chr [1:3] "Clausiliidae" "Helicidae" "Zonitidae"
 - attr(*, "class")= chr "xcount"
```

3.3 Reporting with `mefa`

Parts of the `mefa` object can be reported by the `report.mefa` function into a plain text or preformatted LaTeX file (`tex` operator (argument) containing italicised species names, optionally authors and description dates (`author` specifier), sectioning (according to species or samples, `ordering` argument) and with or without count data (`binary` argument):

```
> library(mefa)
> report.mefa(mf, "dolina-report.tex", ordering = "species", biotic.data = c(1:5),
+     species.name = "spec.name", species.order = 1, author = 0,
+     tex = TRUE, binary = FALSE, sep = c(", ", " (", ")", "; "))
```

The result will look like this:

Carychium tridentatum 1R2, rock, 2, northern, bottom (1).
Cochlodina laminata 1H2, dead.wood, 2, western, middle (2); 1R3, rock, 3, northern, bottom (1).
Cochlodina orthostoma 1L1, live.wood, 1, eastern, edge (1); 1R1, rock, 1, eastern, middle (1); 1R2, rock, 2, northern, bottom (1).
Cochlodina cerata 1R1, rock, 1, eastern, middle (2).
Macrogastra latestriata 1H1, dead.wood, 1, southern, middle (5).
Balea biplicata 1A5, litter, 5, northern, middle (1); 1H2, dead.wood, 2, western, middle (1); 1H3, dead.wood, 3, flat, bottom (1); 1L2, live.wood, 2, western, middle (1); 1R1, rock, 1, eastern, middle (3); 1R2, rock, 2, northern, bottom (3); 1R3, rock, 3, northern, bottom (5).
Bulgarica cana 1H1, dead.wood, 1, southern, middle (1); 1R2, rock, 2, northern, bottom (1).
Discus perspectivus 1A3, litter, 3, southern, middle (1); 1A5, litter, 5, northern, middle (1); 1H1, dead.wood, 1, southern, middle (1); 1H2, dead.wood, 2, western, middle (1); 1L2, live.wood, 2, western, middle (2); 1R1, rock, 1, eastern, middle (1); 1R2, rock, 2, northern, bottom (2).
Vitrea diaphana 1A4, litter, 4, flat, bottom (2); 1H2, dead.wood, 2, western, middle (2); 1H3, dead.wood, 3, flat, bottom (2); 1L3, live.wood, 3, flat, bottom (1); 1R2, rock, 2, northern, bottom (2).
Daudebardia rufa 1H2, dead.wood, 2, western, middle (1).
Aegopinella pura 1H2, dead.wood, 2, western, middle (1); 1L2, live.wood, 2, western, middle (1); 1R2, rock, 2, northern, bottom (2); 1R3, rock, 3, northern, bottom (1).
Aegopinella minor 1A2, litter, 2, southern, edge (1); 1A3, litter, 3, southern, middle (5); 1A4, litter, 4, flat, bottom (5); 1A7, litter, 7, northern, outside (1); 1H1, dead.wood, 1, southern, middle (1); 1H2, dead.wood, 2, western, middle (2); 1H3, dead.wood, 3, flat, bottom (3); 1L1, live.wood, 1, eastern, edge (1); 1L2, live.wood, 2, western, middle (2); 1L3, live.wood, 3, flat, bottom (4); 1R1, rock, 1, eastern, middle (4); 1R2, rock, 2, northern, bottom (1); 1R3, rock, 3, northern, bottom (5).
Helicodonta obvoluta 1H1, dead.wood, 1, southern, middle (1); 1H2, dead.wood, 2, western, middle (1); 1H3, dead.wood, 3, flat, bottom (6); 1R1, rock, 1, eastern, middle (8); 1R2, rock, 2, northern, bottom (1); 1R3, rock, 3, northern, bottom (4).
Euomphalia strigella 1A4, litter, 4, flat, bottom (1); 1A5, litter, 5, northern, middle (1); 1A6, litter, 6, northern, edge (1); 1A7, litter, 7, northern, outside (1); 1H1, dead.wood, 1, southern, middle (1); 1H2, dead.wood, 2, western, middle (1); 1L1, live.wood, 1, eastern, edge (1); 1R2, rock, 2, northern, bottom (4).
Trichia unidentata 1A3, litter, 3, southern, middle (2); 1A4, litter, 4, flat, bottom (2); 1A5, litter, 5, northern, middle (1); 1H2, dead.wood, 2, western, middle (1); 1L1, live.wood, 1, eastern, edge (1); 1R1, rock, 1, eastern, middle (5); 1R2, rock, 2, northern, bottom (3); 1R3, rock, 3, northern, bottom (1).
Perforatella incarnata 1A3, litter, 3, southern, middle (1); 1A5, litter, 5, northern, middle (1); 1R1, rock, 1, eastern, middle (3); 1R2, rock, 2, northern, bottom (5); 1R3, rock, 3, northern, bottom (3).
Perforatella vicina 1A2, litter, 2, southern, edge (1); 1A3, litter, 3, southern, middle (1); 1H1, dead.wood, 1, southern, middle (1); 1H3, dead.wood, 3, flat, bottom (1); 1R1, rock, 1, eastern, middle (4); 1R2, rock, 2, northern, bottom (1); 1R3, rock, 3, northern, bottom (3).
Chilostoma faustinum 1A5, litter, 5, northern, middle (2); 1A6, litter, 6, northern, edge (1); 1L1, live.wood, 1, eastern, edge (1); 1L3, live.wood, 3, flat, bottom (1); 1R1, rock, 1, eastern, middle (7); 1R2, rock, 2, northern, bottom (12); 1R3, rock, 3, northern, bottom (6).
Isognomostoma isognomostomos 1R1, rock, 1, eastern, middle (6); 1R2, rock, 2, northern, bottom (8); 1R3, rock, 3, northern, bottom (1).

Reporting can be useful for making official reports of inventories or presenting data in supporting online material of manuscripts. The LaTeX output file can be easily copied or included into documents eg. in the way presented in the text window by typing `mefadocs("SampleReport")`.

3.4 Further data analysis

The cross-tabulated count data stored in `xcount` and `mefa` objects can be used readily to visualise simple results on abundances, richness, etc., or can be used in subsequent uni- and multivariate analyses. These possibilities of further data analyses are illustrated in the demo script (`demo(dolina)`). (Will be included here later ...)

4 How to cite `mefa`?

Please refer to the actual answer returned by the command (`citation("mefa")`).

References

[1] F. Samu. A general data model for databases in experimental animal ecology. *Acta zool. Acad. Sci. Hung.*, 45:273–292, 1999.