From faunistic records to analysis

Péter Sólymos*

January 6, 2008

Abstract

The document shows the general way to make `mefa` objects. Here I give a short conceptual description highlighting main features that are especially useful for faunistic count data.

Contents

1 Introduction – motivation

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 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.

*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.

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 faunistic data, shorten the time spent with data preparation and reduce mistakes during data management. Here I give a short overview of the (1) general data model behind 'mefa' objects, (2) main functions of the package, and (3) reporting options.

2 General data model

General database representation of biotic (faunistic) data may include four modules according to Samu[**?**]: 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 object. 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. 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 compiling sample attributes).

3 Workflow in `mefa`

3.1 The faunistic part: the catch

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):

```
> library(mefa)

 mefa is loaded

 - to start demo, type demo(dolina)
 - to view ChangeLog, type mefadocs("ChangeLog")
 - to browse help, type help("mefa-package")

> data(dol.count)
> 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. This "notebook-style" data can be filled with respective data by the function `fill.count`:

```
> fill.count(dol.count)[1:10, ]
```

```
   sample    species  segment count
1     1A1 zero.count    adult     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
```

In this way, "notebook-style" data can be readily converted into an `sscount` (species/sample/count) object:

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

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 way, rows represent the basic units (lots) containing sample, species and segment identifiers (treated as characters) and the count (treated as numeric, but for further operations this should preferably be an integer). An `sscount`object is a list of five objects:

```
> str(ssc)

List of 7
 $ data          :'data.frame':        125 obs. of  4 variables:
  ..$ sample : Factor w/ 16 levels "1A1","1A2","1A3",..: 1 2 2 3 3 3 3 3 3 3 ...
  ..$ species: Factor w/ 20 levels "amin","apur",..: 20 1 17 1 1 1 10 16 17 18 ...
  ..$ segment: Factor w/ 4 levels "adult","broken",..: 1 3 2 1 3 4 3 3 1 3 ...
  ..$ count  : num [1:125] 1 1 1 1 3 1 1 1 1 1 ...
 $ call          : language sscount(sstable = fill.count(dol.count), zc = "zero.count")
 $ zc            : chr "zero.count"
 $ digits        : NULL
 $ nsamples      : num 16
 $ nspecies      : num 19
 $ segment.levels: chr [1:4] "adult" "broken" "fresh" "juvenile"
 - attr(*, "class")= chr "sscount"
```

`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 any 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). For all segments, use:

```
> xc <- xcount(ssc)

Samples with ' zero.count ' were detected.

> str(xc)

List of 4
 $ segment : chr "all"
 $ data    : int [1:16, 1:19] 0 1 5 5 0 0 1 1 2 3 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:16] "1A1" "1A2" "1A3" "1A4" ...
  .. ..$ : chr [1:19] "amin" "apur" "bbip" "bcan" ...
 $ nsamples: int 16
 $ nspecies: int 19
 - attr(*, "class")= chr "xcount"
```

For broken shells only:

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

Samples with ' zero.count ' were detected.

> str(xc.broken)
```

```
List of 4
 $ segment : chr "broken"
 $ data    : int [1:16, 1:19] 0 0 0 0 0 0 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:16] "1A1" "1A2" "1A3" "1A4" ...
  .. ..$ : chr [1:19] "amin" "apur" "bbip" "bcan" ...
 $ nsamples: int 16
 $ nspecies: int 19
 - attr(*, "class")= chr "xcount"
```

The pseudo-species for zero count samples is removed from the resulting `xcount` object, but an empty row is placed in the table. 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).

3.2 The meta part: sample and species attributes

When we have goodfaunistic 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 ...
 $ 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 ...
```

For species, eg. full species names, higher level taxonomic information, shell size:

```
> 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 ...
```

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

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

$set.relation
[1] "equal"

$duplicate
NULL

$missing
NULL

$na
[1] 0

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

$set.relation
[1] "inclusion"

$duplicate
NULL

$missing
NULL

$na
[1] 5
```

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

```
> xo1 <- xorder(xc, which = "samples", dol.sample, 1)
> xo2 <- xorder(xc, which = "species", landsnail, 2)
```

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

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

List of 9
 $ segment      : chr "all"
 $ xcount       : int [1:16, 1:19] 0 1 5 5 0 0 1 1 2 3 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:16] "1A1" "1A2" "1A3" "1A4" ...
```

```
  .. ..$ : chr [1:19] "amin" "apur" "bbip" "bcan" ...
 $ sample.attr  :'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 [1:16] 1 2 3 4 5 6 7 1 2 3 ...
  ..$ 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 [1:16] 2 2 2.5 2 2 1 1.5 2 2 1.5 ...
  ..$ litter.thickness.cm: num [1:16] 3 3 4 5 3 2 5 4 12 5 ...
 $ species.attr :'data.frame':        19 obs. of  8 variables:
  ..$ order          : int [1:19] 92 91 59 65 46 151 43 45 6 74 ...
  ..$ spec.short     : Factor w/ 19 levels "amin","apur",..: 1 2 3 4 5 6 7 8 9 10 ...
  ..$ spec.name      : Factor w/ 19 levels "Aegopinella minor",..: 1 2 3 4 7 6 8 9 5 11 ..
  ..$ author         : Factor w/ 16 levels "(A. Schmidt, 1857)",..: 15 2 8 5 12 11 8 7 10
  ..$ shell.dimension: num [1:19] 9 5 18 18 18 20 17 13 2.3 6.5 ...
  ..$ familia        : Factor w/ 5 levels "Clausiliidae",..: 5 5 1 1 1 4 1 1 2 3 ...
  ..$ subfamilia     : Factor w/ 10 levels "Alopiinae","Ariantinae",..: 10 10 3 3 1 2 1 1
  ..$ genus          : Factor w/ 15 levels "Aegopinella",..: 1 1 2 3 5 9 5 5 4 7 ...
 $ nsamples     : int 16
 $ nspecies     : int 19
 $ totalcount   : int 208
 $ nsample.attr : int 7
 $ nspecies.attr: int 8
 - attr(*, "class")= chr "mefa"
```

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).

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

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

List of 4
 $ segment : chr "all"
 $ data    : int [1:4, 1:19] 6 12 7 10 1 0 1 3 2 1 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:4] "dead.wood" "litter" "live.wood" "rock"
  .. ..$ : chr [1:19] "amin" "apur" "bbip" "bcan" ...
 $ nsamples: int 4
 $ nspecies: int 19
 - attr(*, "class")= chr "xcount"

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

List of 4
 $ segment : chr "all"
 $ data    : int [1:16, 1:5] 0 0 0 0 1 0 0 6 3 1 ...
```

```
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:16] "1A1" "1A2" "1A3" "1A4" ...
  .. ..$ : chr [1:5] "Clausiliidae" "Ellobiidae" "Endodontidae" "Helicidae" ...
 $ nsamples: int 16
 $ nspecies: int 5
 - attr(*, "class")= chr "xcount"
```

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

$segment
[1] "all"

$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

$nsamples
[1] 4

$nspecies
[1] 5

attr(,"class")
[1] "xcount"
```

The result is always and xcount object, because, due to stratification, attribute consistency diminishes. By using new attribute tables, a `mefa` new object can be created.

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

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

[1] "dead.wood" "litter"    "live.wood" "rock"

> levels(ex1$sample.attr$microhabitat)

[1] "dead.wood" "live.wood" "rock"

> ex2 <- exclmf(mic.fam, "species", c(2, 3), empty = TRUE)
> ex2

$segment
[1] "all"

$data
          Clausiliidae Helicidae Zonitidae
```

```
dead.wood           10        13        12
litter               1        16        14
live.wood            2         4         9
rock                17        85        15

$nsamples
[1] 4

$nspecies
[1] 3

attr(,"class")
[1] "xcount"
```

In the resulting mefa object, respective parts will be excluded for both the crosstabulation and attribute teble. Empty rows (samples) can be removed by `empty = FALSE`:

```
> mf0 <- exclmf(mf, which = "samples", excl = NULL, empty = FALSE)
> dim(mf$xcount)

[1] 16 19

> dim(mf0$xcount)

[1] 16 19
```

4 Reporting with `mefa`

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

```
> report.mefa("dolina-report.tex", mf, order = "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 LaTeXoutput file can be easily copied or included into documents eg. in the way presented in the text window by typing (`Sweave`-ing is currently not available):

```
> mefadocs("SampleReport")
```

5 Further data analysis

The cross-tabulated count data stored in `xcount` and `mefa` objects can be used readily to calculate sample specific results (abundance, richness, etc.) or can be used in multivariate analyses. These possibilities of further data analyses are illustrated in the demo script:

```
> demo(dolina)
```

References

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