mefa4 Design Decisions and Performance

Péter Sólymos
solymos@ualberta.ca

Abstract

mefa4 is a reimplementation of the S3 object classes found in the **mefa** R package. The new S4 class `"Mefa"` has all the consistency checks that S3 classes cannot have, and most importantly, it stores the crosstabuted results as a compact sparse matrix (S4 object class `"dgCMatrix"` of the **Matrix** package). The use of sparse matrices speed up computations, and reduces object sizes considerably. This vignette introduces the main functions, classes and methods of the package **mefa4**.

Processed with **mefa4** 0.2-0 and **mefa** 3.2-0 in R version 2.12.1 (2010-12-16) on February 10, 2011.

Keywords: biodiversity, biogeography, data manipulation, ecology, multivariate methods, R, sparse matrices.

1. Introduction

The aim of the **mefa** and **mefa4** packages are to help in storing cross tabulated ecological data tables (community data) together with attributes for rows (samples) and columns (species, taxa). This allows that one can easily subset the relational data object without seperately manipulating 2–3 pieces of R objects. By doing so, the chances of errors are reduced.

As ecological data sets are increasing in size, it is necessary to find more efficient ways of data storage and manipulation. To this end, it was in the air for some time to redesign the **mefa** package and take advantages of sparse matrices from the **Matrix** package. This is done at the costs of some old functionalities not being available for S4 classes at the time being. Here I give an overview so the user can decide how to use the parallel availability of old S3 and newer S4 classes.

2. Comparison of S3 and S4 classes

The S3 classes defined in **mefa** were `stcs` and `mefa`. `stcs` is a data frame with several attributes:

```
R> library(mefa)

mefa 3.2-0          2011-02-07

R> x <- data.frame(
+       sample = paste("Sample", c(1,1,2,2,3,4), sep="."),
+       species = c(paste("Species", c(1,1,1,2,3), sep="."),  "zero.pseudo"),
+       count = c(1,2,10,3,4,0),
+       segment = letters[c(6,13,6,13,6,6)])
R> s <- stcs(x)
R> attributes(s)

$names
[1] "samp"  "taxa"  "count" "segm"

$row.names
[1] 1 2 3 4 5 6

$class
[1] "stcs"       "data.frame"

$call
stcs(dframe = x)

$expand
[1] FALSE

$zero.count
[1] TRUE

$zero.pseudo
[1] "zero.pseudo"
```

These attributes ensure that the crosstab made by the function `mefa()` creates a proper crosstab by eliminating the column that is only a placeholder for empty samples, etc:

```
R> samp <- data.frame(samples=levels(x$sample), var1=1:2)
R> taxa <- data.frame(specnames=levels(x$species), var2=c("b","a"))
R> rownames(samp) <- samp$samples
R> rownames(taxa) <- taxa$specnames
R> (m <- mefa(s, samp, taxa))

An object of class 'mefa' containing

 $ xtab: 20 individuals of 3 taxa in 4 samples,
 $ segm: 2 (non-nested) segments:
         f, m,
 $ samp: table for samples provided (2 variables),
 $ taxa: table for taxa provided (2 variables).

R> m$xtab

          taxa
samp       Species.1 Species.2 Species.3
```

```
  Sample.1         3         0         0
  Sample.2        10         3         0
  Sample.3         0         0         4
  Sample.4         0         0         0
```

The `stcs` step is almost redundant, and inefficient relative to the `Matrix:::xtabs` function with `sparse = TRUE`. This function is adapted to some extent, so it can subset the crosstabulated results before returning the value (`rdrop` and `cdrop` arguments, that is available as the `Xtab` function in the **mefa4** package). This takes a formula, and can be applied directly on a data frame. The formula can have a left-hand side, or the left-hand side can be missing. The right-hand side can contain 2–3 factors, and the result will be a sparse matrix or a list of sparse matrices, respectively:

```
R> library(mefa4)
```

```
mefa4 0.2-0          2011-02-07
```

```
R> x0  <- Xtab(~ sample + species, x)
R> x1  <- Xtab(count ~ sample + species, x)
R> x11 <- Xtab(count ~ sample + species + segment, x)
```

Dropping some rows/columns can be done in several ways. A logical statement implies that all empty rows/columns are dropped, but indices (numeric or character) can also be used:

```
R> x2  <- Xtab(count ~ sample + species, x, cdrop=FALSE, rdrop=TRUE)
R> x21 <- Xtab(count ~ sample + species, x, cdrop=TRUE, rdrop=FALSE)
R> (x22 <- Xtab(count ~ sample + species, x, cdrop="zero.pseudo"))
```

```
4 x 3 sparse Matrix of class "dgCMatrix"
         Species.1 Species.2 Species.3
Sample.1         3         .         .
Sample.2        10         3         .
Sample.3         .         .         4
Sample.4         .         .         .
```

The results here are sparse matrices in compact mode, this means that redundant indices are only kept once, so it is more compact than a long formatted database representation stored in an `stcs` object or in the original data frame, or a triplet representation of a sparse matrix. See vignettes in the **Matrix** package for more details on S4 sparse matrix classes.

The S4 class `"Mefa"` is defined in the **mefa4** package. It can be created by the `Mefa()` function, and the result has 4 slots:

```
R> (x3 <- Mefa(x1, samp, taxa))
```

```
Object of class "Mefa"
  ..@ xtab: 4 x 4 sparse Matrix
  ..@ samp: data frame with 2 variables
  ..@ taxa: data frame with 2 variables
  ..@ join: left
```

The xtab slot stores the crosstab in sparse matrix format. The samp slot stores the row attributes for xtab as data frame or can be NULL. The taxa slot stores the column attributes for xtab as data frame or can be NULL. Validity checks are done to ensure proper object classes to be used and matching dimnames. The option that a column in the attribute tables can be specified to find matching names is not available in the new implementation. Corresponding rownames of the data frames has to match dimnames of xtab. The join slot can be "left" (all rows/columns in the xtab are kept, matching attributes are selected, non matching attributes are excluded, and missing attributes are filled up with NA) or "inner" (only the intersection of corresponding dimnames are used to form the return value).

The call in Mefa() can take any matrix or sparse matrix as argument, but it will be stored in a sparse mode. Here we use a matrix as input, and samp has missing values ("left" join is used by default):

```
R> (x4 <- Mefa(as.matrix(x1), samp[1:2,]))

Object of class "Mefa"
  ..@ xtab: 4 x 4 sparse Matrix
  ..@ samp: data frame with 2 variables
  ..@ taxa: NULL
  ..@ join: left
```

The effect of "inner" join is as follows:

```
R> (x5 <- Mefa(x2, samp, taxa, join="inner"))

Object of class "Mefa"
  ..@ xtab: 3 x 4 sparse Matrix
  ..@ samp: data frame with 2 variables
  ..@ taxa: data frame with 2 variables
  ..@ join: inner

R> (x51 <- Mefa(x2, samp[1:2,], taxa, join="inner"))

Object of class "Mefa"
  ..@ xtab: 2 x 4 sparse Matrix
  ..@ samp: data frame with 2 variables
  ..@ taxa: data frame with 2 variables
  ..@ join: inner
```

A "Mefa" object with only xtab can also be defined:

```
R> (x6 <- Mefa(x1))

Object of class "Mefa"
  ..@ xtab: 4 x 4 sparse Matrix
  ..@ samp: NULL
  ..@ taxa: NULL
  ..@ join: left
```

The structure of the S3 and S4 classes are very similar, and even the accessor methods (`xtab()`, `samp()`, `taxa()`, `segm()`) work properly on both types. The S4 class does not have a slot for a call, and there is no `segm` element/slot either. This means that a `"Mefa"` object cannot have 3 dimensions, only 2. `Xtab` can create 3-dimensional sparse array-like objects (list of sparse matrices of the same dimensions), but there is no formal S4 class that can handle sparse matrix lists as part of a `"Mefa"` object. The `as.mefa` method can convert such a list of sparse matrices into an S3 `"mefa"` object with segments.

3. Back and forth

Coercion methods are defined in both the **mefa** and **mefa4** packages to ensure that S3 and S4 objects are interchangeable:

```
R> as.stcs(x1)
R> as.mefa(x1)
R> as.stcs(x3)
R> a <- as.mefa(x3)
R> xtab(a)
R> samp(a)
R> taxa(a)
R> segm(a)
R> segm(x3)
R> as.Mefa(a)
R> as.Xtab(a)
R> s <- melt(a)
R> as.Xtab(s)
R> as.Mefa(s)
R> melt(x1)
R> melt(x3)
```

4. Subsetting and replacement

Accessing and replacing parts of the `"Mefa"` object is conveniently done by methods `xtab`, `samp`, and `taxa` (the `segm` S3 method only returns the codextab slot of an S4 `"Mefa"` object):

```
R> xtab(x3)

4 x 4 sparse Matrix of class "dgCMatrix"
         Species.1 Species.2 Species.3 zero.pseudo
Sample.1         3         .         .           .
Sample.2        10         3         .           .
Sample.3         .         .         4           .
Sample.4         .         .         .           .

R> x1[3,1] <- 999
R> xtab(x3) <- x1
R> xtab(x3)
```

```
4 x 4 sparse Matrix of class "dgCMatrix"
         Species.1 Species.2 Species.3 zero.pseudo
Sample.1         3         .         .           .
Sample.2        10         3         .           .
Sample.3       999         .         4           .
Sample.4         .         .         .           .
```

Attribute tables can be set to `NULL`, or replaced:

```
R> samp(x3)
```

```
          samples var1
Sample.1 Sample.1    1
Sample.2 Sample.2    2
Sample.3 Sample.3    1
Sample.4 Sample.4    2
```

```
R> samp(x3) <- NULL
R> samp(x3)
```

```
NULL
```

```
R> samp(x3) <- samp[1:3,]
R> samp(x3)
```

```
          samples var1
Sample.1 Sample.1    1
Sample.2 Sample.2    2
Sample.3 Sample.3    1
Sample.4     <NA>   NA
```

```
R> taxa(x3)
R> taxa(x3) <- NULL
R> taxa(x3)
R> taxa(x3) <- taxa[1:3,]
R> taxa(x3)
```

Replacing parts of these attribute tables can be done as

```
R> samp(x3)[1,]
```

```
          samples var1
Sample.1 Sample.1    1
```

```
R> samp(x3)[1,2] <- 3
R> samp(x3)[1,]
```

```
          samples var1
Sample.1 Sample.1    3
```

Subsetting the whole "Mefa" object is done via the [method:

```
R> x3[3:2, 1:2]

Object of class "Mefa"
  ..@ xtab: 2 x 2 sparse Matrix
  ..@ samp: data frame with 2 variables
  ..@ taxa: data frame with 2 variables
  ..@ join: left

R> x3[3:2, ]

Object of class "Mefa"
  ..@ xtab: 2 x 4 sparse Matrix
  ..@ samp: data frame with 2 variables
  ..@ taxa: data frame with 2 variables
  ..@ join: left

R> x3[ ,1:2]

Object of class "Mefa"
  ..@ xtab: 4 x 2 sparse Matrix
  ..@ samp: data frame with 2 variables
  ..@ taxa: data frame with 2 variables
  ..@ join: left
```

5. Methods for S4 classes

Simple methods are provided for convenience:

```
R> dim(x5)

[1] 3 4

R> dimnames(x5)

[[1]]
[1] "Sample.1" "Sample.2" "Sample.3"

[[2]]
[1] "Species.1"   "Species.2"   "Species.3"   "zero.pseudo"
```

```
R> dn <- list(paste("S", 1:dim(x5)[1], sep=""),
+       paste("SPP", 1:dim(x5)[2], sep=""))
R> dimnames(x5) <- dn
R> dimnames(x5)[[1]] <- paste("S", 1:dim(x5)[1], sep="_")
R> dimnames(x5)[[2]] <- paste("SPP", 1:dim(x5)[2], sep="_")
R> t(x5)

Object of class "Mefa"
  ..@ xtab: 4 x 3 sparse Matrix
  ..@ samp: data frame with 2 variables
  ..@ taxa: data frame with 2 variables
  ..@ join: inner
```

6. Grouping rows and columns

The aggregate method was defined for S3 mefa objects. Its equivalent (although it cannot sum the cells simultaneously for rows and columns, but it was done in 2 subsequent steps anyway) is the groupSums method. The MARGIN argument indicates if rows (MARGIN = 1) or columns (MARGIN = 2) are to be added together:

```
R> groupSums(as.matrix(x2), 1, c(1,1,2))

  Species.1 Species.2 Species.3 zero.pseudo
1        13         3         0           0
2         0         0         4           0

R> groupSums(as.matrix(x2), 2, c(1,1,2,2))

          1 2
Sample.1  3 0
Sample.2 13 0
Sample.3  0 4

R> groupSums(x2, 1, c(1,1,2))

2 x 4 sparse Matrix of class "dgCMatrix"
  Species.1 Species.2 Species.3 zero.pseudo
1        13         3         .           .
2         .         .         4           .

R> groupSums(x2, 2, c(1,1,2,2))

3 x 2 sparse Matrix of class "dgCMatrix"
          1 2
Sample.1  3 .
Sample.2 13 .
Sample.3  . 4
```

```
R> groupSums(x5, 1, c(1,1,2))

Object of class "Mefa"
  ..@ xtab: 2 x 4 sparse Matrix
  ..@ samp: NULL
  ..@ taxa: data frame with 2 variables
  ..@ join: inner

R> groupSums(x5, 2, c(1,1,2,2))

Object of class "Mefa"
  ..@ xtab: 3 x 2 sparse Matrix
  ..@ samp: data frame with 2 variables
  ..@ taxa: NULL
  ..@ join: inner
```

A simple extension of this is the `groupMeans` method:

```
R> groupMeans(as.matrix(x2), 1, c(1,1,2))
R> groupMeans(as.matrix(x2), 2, c(1,1,2,2))
R> groupMeans(x2, 1, c(1,1,2))
R> groupMeans(x2, 2, c(1,1,2,2))
R> groupMeans(x5, 1, c(1,1,2))
R> groupMeans(x5, 2, c(1,1,2,2))
```

7. Combining objects

`mbind` can be used to combine 2 matrices (dense or sparse). The 2 input objects are combined in a left join manner, which means that all the elements in the first object are retained, and only non-overlapping elements in the second object are used. Elements of the returning object that are not part of either objects (outer set) are filled up with value provided as `fill` argument.

```
R> x=matrix(1:4,2,2)
R> rownames(x) <- c("a", "b")
R> colnames(x) <- c("A", "B")
R> y=matrix(11:14,2,2)
R> rownames(y) <- c("b", "c")
R> colnames(y) <- c("B", "C")
R> mbind(x, y)

   A  B  C
a  1  3 NA
b  2  4 13
c NA 12 14
```

```
R> mbind(x, y, fill=0)

  A  B  C
a 1  3  0
b 2  4 13
c 0 12 14

R> mbind(as(x, "sparseMatrix"), as(y, "sparseMatrix"))

3 x 3 sparse Matrix of class "dgCMatrix"
   A  B  C
a  1  3 NA
b  2  4 13
c NA 12 14
```

"Mefa" objects can be combined in a similar way, where attribute tables are combined in a left join fashion (S3 "mefa" objects have to be coerced by the as.Mefa method beforehand – this is so because the S3 class does not allow NA values in $xtab, and it is safer to avoid unnecessary complications):

```
R> sampx <- data.frame(x1=1:2, x2=2:1)
R> rownames(sampx) <- rownames(x)
R> sampy <- data.frame(x1=3:4, x3=10:11)
R> rownames(sampy) <- rownames(y)
R> taxay <- data.frame(x1=1:2, x2=2:1)
R> rownames(taxay) <- colnames(y)
R> taxax <- NULL
R> mbind(Mefa(x, sampx), Mefa(y, sampy, taxay))

Object of class "Mefa"
  ..@ xtab: 3 x 3 sparse Matrix
  ..@ samp: data frame with 3 variables
  ..@ taxa: data frame with 2 variables
  ..@ join: left
```

8. Performace comparisons

We compare the performance of the **mefa** and **mefa4** packages. We are using a long formatted raw data file from the Alberta Biodiversity Monitoring Institute database (available at http://www.abmi.ca):

```
R> data(abmibirds)
```

This is the processing with **mefa** and S3 object classes (we are storing the results and processing times):

```
R> b3 <- abmibirds
R> b3 <- b3[!(b3$Scientific.Name %in% c("VNA", "DNC", "PNA")),]
R> levels(b3$Scientific.Name)[levels(b3$Scientific.Name)
+      %in% c("NONE", "SNI")] <- "zero.pseudo"
R> b3$Counts <- ifelse(b3$Scientific.Name == "zero.pseudo", 0, 1)
R> b3$Label <- with(b3, paste(ABMI.Site, Year,
+      Point.Count.Station, sep="_"))
R> x3 <- b3[!duplicated(b3$Label), c("Label",
+      "ABMI.Site", "Year", "Field.Date",
+      "Point.Count.Station", "Wind.Conditions", "Precipitation")]
R> rownames(x3) <- x3$Label
R> z3 <- b3[!duplicated(b3$Scientific.Name), c("Common.Name",
+      "Scientific.Name", "Taxonomic.Resolution",
+      "Unique.Taxonomic.Identification.Number")]
R> rownames(z3) <- z3$Scientific.Name
R> z3 <- z3[z3$Scientific.Name != "zero.pseudo",]
R> t31 <- system.time(s3 <- suppressWarnings(stcs(b3[,
+      c("Label","Scientific.Name","Counts")])))
R> t32 <- system.time(m30 <- mefa(s3))
R> t33 <- system.time(m31 <- mefa(s3, x3, z3))
R> y30 <- m30$xtab
R> t34 <- system.time(m32 <- mefa(y30, x3, z3))
R> m32

An object of class 'mefa' containing

 $ xtab: 59098 individuals of 214 taxa in 3534 samples,
 $ segm: 1 (all inclusive) segment,
 $ samp: table for samples provided (7 variables),
 $ taxa: table for taxa provided (4 variables).
```

The equivalent processing with **mefa4** and S4 object classes:

```
R> b4 <- abmibirds
R> b4$Label <- with(b4, paste(ABMI.Site, Year,
+      Point.Count.Station, sep="_"))
R> x4 <- b4[!duplicated(b4$Label), c("Label", "ABMI.Site",
+      "Year", "Field.Date", "Point.Count.Station",
+      "Wind.Conditions", "Precipitation")]
R> rownames(x4) <- x4$Label
R> z4 <- b4[!duplicated(b4$Scientific.Name), c("Common.Name",
+      "Scientific.Name", "Taxonomic.Resolution",
+      "Unique.Taxonomic.Identification.Number")]
R> rownames(z4) <- z4$Scientific.Name
R> t41 <- system.time(s4 <- Xtab(~ Label + Scientific.Name,
+      b4, cdrop = c("NONE", "SNI"),
+      subset = !(b4$Scientific.Name %in% c("VNA", "DNC", "PNA")),
+       drop.unused.levels = TRUE))
R> t42 <- system.time(m40 <- Mefa(s4))
R> t43 <- system.time(m41 <- Mefa(s4, x4, z4))
R> y40 <- as.matrix(m40@xtab)
R> t44 <- system.time(m42 <- Mefa(y40, x4, z4))
R> m42

Object of class "Mefa"
  ..@ xtab: 3534 x 214 sparse Matrix
  ..@ samp: data frame with 7 variables
  ..@ taxa: data frame with 4 variables
  ..@ join: left

R> sum(m42@xtab)

[1] 59098
```

Let us compare object sizes and processing times, stars indicate similar S3 (`*=3`) and S4 (`*=4`) objects:

```
    SIZE, *=3 SIZE, *=4 TIME, *=3 TIME, *=4       SIZE       TIME
b*    6149312   5439944        NA        NA 0.88464270         NA
s*    1351840    598668      1.59      0.06 0.44285418 0.03773585
y*0   6217976   6217824        NA        NA 0.99997555         NA
m*0   6218828    599280      3.40      0.00 0.09636542 0.00000000
m*1   6670428   1050680      3.36      0.00 0.15751313 0.00000000
m*2   6670428   1050680      0.09      0.04 0.15751313 0.44444444
```

The compressed sparse matrix representation is 44.3% of the `stcs` object in size. `"Mefa"` object sizes are maximum of 15.8% of their S3 representatives. Processing time speed-up is enormous with sparse matrices (0%), and still quite high by standard matrices (44.4%).

Check that objects are the same:

```
R> stopifnot(identical(dim(y30), dim(y40)))
R> stopifnot(identical(setdiff(rownames(y30), rownames(y40)), character(0)))
R> stopifnot(identical(setdiff(rownames(y40), rownames(y30)), character(0)))
R> stopifnot(identical(setdiff(colnames(y30), colnames(y40)), character(0)))
R> stopifnot(identical(setdiff(colnames(y40), colnames(y30)), character(0)))
```

The aggregation also improved quite a bit with sparse matrices:

```
R> system.time(xx3 <- aggregate(m31, "ABMI.Site"))

   user  system elapsed
   4.72    0.00    4.72

R> system.time(xx4 <- groupSums(m41, 1, m41@samp$ABMI.Site))
```

```
 user  system elapsed 
 0.02    0.00    0.01 
```

9. Conclusions

The redesign of the old S3 classes into S4 ones resulted in large savings in computing time and object sizes. Old features are still available due to the free conversion between the two implementations.

10. Session Info

- R version 2.12.1 (2010-12-16), `i386-pc-mingw32`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: Matrix 0.999375-46, lattice 0.19-13, mefa 3.2-0, mefa4 0.2-0
- Loaded via a namespace (and not attached): grid 2.12.1, tools 2.12.1

Affiliation:

Péter Sólymos
Alberta Biodiversity Monitoring Institute
and Boreal Avian Modelling project
Department of Biological Sciences
CW 405, Biological Sciences Bldg
University of Alberta
Edmonton, Alberta, T6G 2E9, Canada
E-mail: `solymos@ualberta.ca`