mefa4 Design Decisions and Performance

Peter Solymos

January 06, 2017

Contents

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 can not have, and most importantly, it stores the cross-tabuted 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**.

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 separately 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 Comparisons

2.1 S3 and S4 classes

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

```
library(mefa)
```

```
## mefa 3.2-7    2016-01-11
```

```
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)])
s <- stcs(x)
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 cross-tab made by the function `mefa()` creates a proper cross-tab by eliminating the column that is only a placeholder for empty samples, etc.:

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

```
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 `stats::xtabs` function with `sparse = TRUE`. This function is adapted to some extent, so it can subset the cross-tabulated 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:

```
library(mefa4)
```

```
## Loading required package: Matrix
```

```
## Loading required package: pbapply
```

```
## mefa4 0.3-4   2016-10-12
```

```
##
## Attaching package: 'mefa4'
```

```
## The following objects are masked from 'package:mefa':
##
##     samp, taxa, xtab
```

```
x0  <- Xtab(~ sample + species, x)
x1  <- Xtab(count ~ sample + species, x)
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:

```
x2  <- Xtab(count ~ sample + species, x, cdrop=FALSE, rdrop=TRUE)
x21 <- Xtab(count ~ sample + species, x, cdrop=TRUE, rdrop=FALSE)
(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:

```
(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 cross-tab 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):

```
(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:

```
(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
```

```
(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:

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

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

The equivalent of the melt method of the **mefa** package is the Melt function in **mefa4**. It can be used to reverse the side effects of the cross-tabulation, thus making a data frame from a matrix, sparse matrix, list of sparse matrices, a mefa or a Mefa object:

```
Melt(x1)
```

```
##       rows      cols value
## 1 Sample.1 Species.1     3
## 2 Sample.2 Species.1    10
## 3 Sample.2 Species.2     3
## 4 Sample.3 Species.3     4
```

```
Melt(x11)
```

```
##       rows      cols segm value
## 1 Sample.1 Species.1    f     1
## 2 Sample.2 Species.1    f    10
## 3 Sample.3 Species.3    f     4
## 4 Sample.1 Species.1    m     2
## 5 Sample.2 Species.2    m     3
```

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.

2.2 Coercions back and forth

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

```
as.stcs(x1)
```

```
##       samp        taxa count        segm
## 1 Sample.1   Species.1     3   undefined
## 2 Sample.2   Species.1    10   undefined
## 3 Sample.2   Species.2     3   undefined
## 4 Sample.3   Species.3     4   undefined
## 5 Sample.4 zero.pseudo     0 zero.pseudo
```

```
as.mefa(x1)
```

```
##
## An object of class 'mefa' containing
##
##  $ xtab: 20 individuals of 4 taxa in 4 samples,
##  $ segm: 1 (all inclusive) segment,
##  $ samp: table for samples not provided,
##  $ taxa: table for taxa not provided.
```

```
as.stcs(x3)
```

```
##       samp        taxa count        segm
## 1 Sample.1   Species.1     3   undefined
## 2 Sample.2   Species.1    10   undefined
## 3 Sample.2   Species.2     3   undefined
## 4 Sample.3   Species.3     4   undefined
## 5 Sample.4 zero.pseudo     0 zero.pseudo
```

```
a <- as.mefa(x3)
xtab(a)
```

```
##          Species.1 Species.2 Species.3 zero.pseudo
## Sample.1         3         0         0           0
## Sample.2        10         3         0           0
## Sample.3         0         0         4           0
## Sample.4         0         0         0           0
```

```
samp(a)
```

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

```
taxa(a)
```

```
##               specnames var2
## Species.1     Species.1    b
## Species.2     Species.2    a
## Species.3     Species.3    b
## zero.pseudo zero.pseudo    a
```

```
segm(a)
```

```
##          Species.1 Species.2 Species.3 zero.pseudo
## Sample.1         3         0         0           0
## Sample.2        10         3         0           0
## Sample.3         0         0         4           0
## Sample.4         0         0         0           0
```

```
segm(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         .         .         .           .
```

```
as.Mefa(a)
```

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

```
as.Xtab(a)
```

```
## 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         .         .         .           .
```

```
s <- melt(a)
as.Xtab(s)
```

```
## 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         .         .         .           .
```

```
as.Mefa(s)
```

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

```
melt(x1)
```

```
##       samp        taxa count        segm
## 1 Sample.1   Species.1     3   undefined
## 2 Sample.2   Species.1    10   undefined
## 3 Sample.2   Species.2     3   undefined
## 4 Sample.3   Species.3     4   undefined
## 5 Sample.4 zero.pseudo     0 zero.pseudo
```

```
melt(x3)
```

```
##       samp        taxa count        segm
## 1 Sample.1   Species.1     3   undefined
## 2 Sample.2   Species.1    10   undefined
## 3 Sample.2   Species.2     3   undefined
## 4 Sample.3   Species.3     4   undefined
## 5 Sample.4 zero.pseudo     0 zero.pseudo
```

2.3 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 `xtab` slot of an S4 `"Mefa"` object):

```
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         .         .         .           .
```

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

```
samp(x3)
```

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

```
## Sample.4 Sample.4    2
```

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

```
## NULL
```

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

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

```
taxa(x3)
```

```
##               specnames var2
## Species.1     Species.1    b
## Species.2     Species.2    a
## Species.3     Species.3    b
## zero.pseudo zero.pseudo    a
```

```
taxa(x3) <- NULL
taxa(x3)
```

```
## NULL
```

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

```
##             specnames var2
## Species.1   Species.1    b
## Species.2   Species.2    a
## Species.3   Species.3    b
## zero.pseudo      <NA> <NA>
```

Replacing parts of these attribute tables can be done as

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

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

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

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

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

```
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
```

```
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
```

```
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
```

3 Methods for S4 classes

Simple methods are provided for convenience:

```
dim(x5)
```

```
## [1] 3 4
```

```
dimnames(x5)
```

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

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

4 Utilities

4.1 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:

```
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
```

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

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

```
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           .
```

```
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
```

```
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
```

```
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:

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

```
##   Species.1 Species.2 Species.3 zero.pseudo
## 1       6.5       1.5         0           0
## 2       0.0       0.0         4           0
```

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

```
##            1 2
## Sample.1 1.5 0
## Sample.2 6.5 0
## Sample.3 0.0 2
```

```
groupMeans(x2, 1, c(1,1,2))
```

```
## 2 x 4 sparse Matrix of class "dgCMatrix"
##   Species.1 Species.2 Species.3 zero.pseudo
## 1       6.5       1.5         .           .
## 2       .         .           4           .
```

```
groupMeans(x2, 2, c(1,1,2,2))
```

```
## 3 x 2 sparse Matrix of class "dgCMatrix"
##            1 2
## Sample.1 1.5 .
## Sample.2 6.5 .
## Sample.3 .   2
```

```
groupMeans(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
```

```
groupMeans(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
```

4.2 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.

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

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

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

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

```
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):

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

5 Performance 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):

```
library(mefa)
library(mefa4)
data(abmibirds)
```

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

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

```
b4 <- abmibirds
b4$Label <- with(b4, paste(ABMI.Site, Year,
    Point.Count.Station, sep="_"))
```

```
x4 <- b4[!duplicated(b4$Label), c("Label", "ABMI.Site",
    "Year", "Field.Date", "Point.Count.Station",
    "Wind.Conditions", "Precipitation")]
rownames(x4) <- x4$Label
z4 <- b4[!duplicated(b4$Scientific.Name), c("Common.Name",
    "Scientific.Name", "Taxonomic.Resolution",
    "Unique.Taxonomic.Identification.Number")]
rownames(z4) <- z4$Scientific.Name
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))
t42 <- system.time(m40 <- Mefa(s4))
t43 <- system.time(m41 <- Mefa(s4, x4, z4))
y40 <- as.matrix(m40@xtab)
t44 <- system.time(m42 <- Mefa(y40, x4, z4))
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
```

```
sum(m42@xtab)
```

```
## [1] 59098
```

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

```
res <- cbind("SIZE, *=3"=c("b*"=object.size(b3),
    "s*"=object.size(s3),
    "y*0"=object.size(y30),
    "m*0"=object.size(m30),
    "m*1"=object.size(m31),
    "m*2"=object.size(m32)),
"SIZE, *=4"=c("b*"=object.size(b4),
    "s*"=object.size(s4),
    "y*0"=object.size(y40),
    "m*0"=object.size(m40),
    "m*1"=object.size(m41),
    "m*2"=object.size(m42)),
"TIME, *=3"=c("b*"=NA,
    "s*"=t31[3],
    "y*0"=NA,
    "m*0"=t32[3],
    "m*1"=t33[3],
    "m*2"=t34[3]),
```

```
"TIME, *=4"=c("b*"=NA,
    "s*"=t41[3],
    "y*0"=NA,
    "m*0"=t42[3],
    "m*1"=t43[3],
    "m*2"=t44[3]))
(res <- cbind(res, "SIZE"=res[,2]/res[,1], "TIME"=res[,4]/res[,3]))
```

```
##     SIZE, *=3 SIZE, *=4 TIME, *=3 TIME, *=4      SIZE        TIME
## b*    6480216   5771200        NA        NA 0.8905876          NA
## s*    1428600    674304     0.864     0.050 0.4720034 0.057870370
## y*0   6293248   6292984        NA        NA 0.9999581          NA
## m*0   6294744    675432     1.920     0.002 0.1073009 0.001041667
## m*1   6910160   1290448     1.630     0.002 0.1867465 0.001226994
## m*2   6910160   1290448     0.055     0.017 0.1867465 0.309090909
```

The compressed sparse matrix representation is 47.2% of the `stcs` object in size. `"Mefa"` object sizes are maximum of 18.7% of their S3 representatives. Processing time speed-up is enormous with sparse matrices (0.1%), and still quite high by standard matrices (30.9%).

Check that objects are the same:

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

The aggregation also improved quite a bit with sparse matrices:

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

```
##    user  system elapsed
##   0.360   0.010   0.375
```

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

```
##    user  system elapsed
##   0.006   0.000   0.005
```

6 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.