A beginners tutorial on the *fuzzySim* R package

A. Márcia Barbosa (barbosa@uevora.pt)

updated 28 April 2014

[newer versions may turn up at http://fuzzysim.r-forge.r-project.org/fuzzySim_tutorial.pdf]

The ***fuzzySim*** package works within the free and open-source R statistical software, so you first need to **download, install and open R** (available at http://www.r-project.org). In this tutorial, in `Courier New` font are the **commands that you need to type (or copy and paste) into** the **R** console (and then **press** the '**enter**' key to execute them). Note that all **commands are case-sensitive**, so you must respect upper- and lower-case letters; that you must always use **straight** (', ") rather than curly (', ") **quotes and apostrophes**; and that R is only ready to receive a new command when there's a *prompt* sign (>) at the beginning of the last line in the R console (if not, it's still waiting for an operation to be finished or for you to complete a previous command – watch out for unclosed parentheses or such). **Enter new commands only after a prompt (>) sign**. For commands that generate visible results in R, these are shown below.

Install *fuzzySim* by pasting the command below in the R console (when connected to the internet):

```
install.packages("fuzzySim", repos = "http://R-Forge.R-project.org")
```

This normally requires the latest R version. **Otherwise**, you can **download** the compressed **source files** from the package development page (https://r-forge.r-project.org/R/?group_id=1853) and then **install the package from your disk** (something like "*Packages - Install packages from local zip files*", or "*Tools - Install packages - Install from: Package Archive File*", or "*Packages & Data - Package installer, Packages repository - Local binary package*" ... depending on your R interface).

You only need to install the package **once**, unless a new version becomes available. Then **load *fuzzySim*** (**every time you open a new R session** in which you intend to use the package; no need for an internet connection anymore) by pasting the following command in R:

```
library(fuzzySim)
```

Load the 'rotifers' sample dataset that comes with the *fuzzySim* package, to use as an example:

```
data(rotifers)
```

You can get more information on this dataset (the following command should open a *help* window):

```
help(rotifers)
```

Show the first 10 rows of the *rotifers* dataset:

```
head(rotifers, 10)
```

	LEVEL3_COD	species
1	DEN	Brachionus_plicatilis_plicatilis
2	DEN	Keratella_cochlearis_cochlearis
3	DEN	Keratella_quadrata_quadrata
4	DEN	Asplanchna_priodonta
5	DEN	Brachionus_angularis_angularis
6	DEN	Conochilus_unicornis
7	DEN	Filinia_longiseta
8	DEN	Brachionus_calyciflorus
9	FIN	Asplanchna_priodonta
10	FIN	Keratella_cochlearis_cochlearis

The first column contains the identifiers of the spatial units, which are TDWG level 3 codes (mostly of countries), and the second column contains the (sub)species names. These are a bit long, especially if we intend to use them as column names further on, so use the *spCodes* function of *fuzzySim* to **add to the *rotifers* dataset a column named *spcode* with species name abbreviations**, consisting of the first letter of the genus + the first 5 letters of the specific name. Specify that the character separating words in the input species names is an underscore (_, see above), and that the character you want separating the genus from the specific name code is empty (no separator):

```
rotifers$spcode <- spCodes(rotifers$species, sep.species = "_", nchar.gen
          = 1, nchar.sp = 5, nchar.ssp = 0, sep.spcode = "")
```

You can try the above with different options for *nchar.gen*, *nchar.sp* and *nchar.ssp*; the function will return an error message if the resulting codes are not unique for each species. Find out more details on the *spCodes* function with `help(spCodes)`.

Now show the first 10 rows of the *rotifers* dataset after you've added the *spcode* column:

```
head(rotifers, 10)
```

	LEVEL3_COD	species	spcode
1	DEN	Brachionus_plicatilis_plicatilis	Bplica
2	DEN	Keratella_cochlearis_cochlearis	Kcochl
3	DEN	Keratella_quadrata_quadrata	Kquadr
4	DEN	Asplanchna_priodonta	Apriod
5	DEN	Brachionus_angularis_angularis	Bangul
6	DEN	Conochilus_unicornis	Cunico
7	DEN	Filinia_longiseta	Flongi
8	DEN	Brachionus_calyciflorus	Bcalyc
9	FIN	Asplanchna_priodonta	Apriod
10	FIN	Keratella_cochlearis_cochlearis	Kcochl

The *rotifers* dataset is in long format, listing in the same column the species that are present in each spatial unit. For analyzing distributional relationships with *fuzzySim*, we need a presence-absence table with one species per column (wide format). So, **create a new table called *rotifers.presabs* with the *rotifers* presence-absence data converted to wide format** and using *spcode*s as column names:

```
rotifers.presabs <- splist2presabs(rotifers, sites.col = "LEVEL3_COD",
                   sp.col = "spcode", keep.n = FALSE)
```

Show the first rows of the result (scroll up to see the beginning of the table):

```
head(rotifers.presabs)
```

	LEVEL3_COD	Abrigh	Afissa	Apriod	Bangul	Bcalyc	Bplica	Bquadr	Burceo
1	ABT	0	0	1	1	0	0	0	0
2	AFG	1	0	1	1	1	1	1	1
3	AGE	1	1	0	1	1	1	1	1
4	AGS	1	1	1	1	1	1	1	1
5	AGW	0	1	0	1	1	1	1	1
6	ALG	1	0	1	1	1	1	1	1

	Cgibba	Cobtus	Cunico	Edilat	Flongi	Hmira	Kcochl	Kquadr	Lacumi	Lbulla
1	0	0	1	0	1	1	1	1	0	0
2	1	1	1	0	0	0	1	1	1	1
3	1	0	1	1	1	1	1	1	1	1
4	0	1	1	1	1	0	1	1	1	1
5	1	0	0	1	1	1	1	1	0	1

You can **map these data** if you have a map of the same spatial units and with the same unit identifiers; the TDWG maps are available online. Use the following R commands to **create a folder in your working directory**, **download the TDWG level 3 shapefile** into that folder (if you're connected to the internet) **and unzip it**:

```
dir.create("TDWG_shapefile")

download.file("http://www.kew.org/gis/tdwg/downloads/level3.zip",
  destfile = "TDWG_shapefile/TDWG_level3.zip", method = "auto")

unzip("TDWG_shapefile/TDWG_level3.zip", exdir = "TDWG_shapefile")
```

If everything went well, you should now have these files on your disk, in the working directory (type `getwd()` to find out where it is). Now **import the map to R**. This **requires the *rgdal* package**; the following command will **install *rgdal*** within your R installation if it's not there already (and if you're connected to the internet). You may need to provide additional information if R asks you, such as selecting a CRAN mirror.

```
if (!("rgdal" %in% rownames(installed.packages())))
  install.packages("rgdal")
```

Now **load the *rgdal* package** and **create a map** named *TDWG3shp* by **importing the shapefile** you've downloaded before, using the *readOGR* function of *rgdal*:

```
library(rgdal)

TDWG3shp <- readOGR(dsn = "TDWG_shapefile", layer = "level3")
```

The map is now in your R session. If you want, you can delete the shapefile folder that was saved in the working directory:

```
unlink("TDWG_shapefile", recursive = TRUE)
```

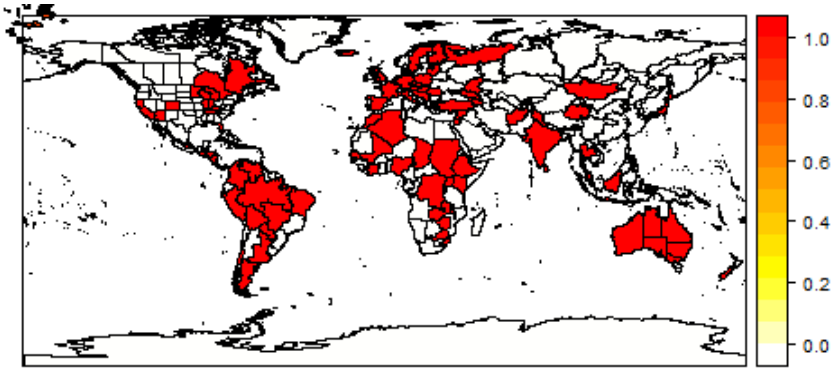
Now **include the *rotifers.presabs* data in the map's attribute table** within R, specifying the name of the column containing the region identifiers in both tables (*LEVEL3_COD* in this case):

```
TDWG3shp@data <- merge(TDWG3shp@data, rotifers.presabs, all.x = TRUE, by
  = "LEVEL3_COD")
```

You are now ready to **map the presence-absence of particular species** from the *rotifers.presabs* table. The *spplot* function in the next command is in the *sp* R package, but this should have been

installed and loaded along with *rgdal* before. Be patient, as this command can be slow to process. The resulting map should pop out in a graphics window within R:

```
print(spplot(TDWG3shp, zcol = "Abrigh", col.regions =
                rev(heat.colors(256))))
```



Try this also with *zcol* = other species in `names(TDWG3shp)`. As you can see in the maps and in the *rotifers.presabs* table, these are **binary** (either 0 or 1) **presence-absence data**. You can **get a fuzzy** (continuous between 0 and 1) **version** of such data using e.g. **trend surface analysis** or **inverse distance interpolation**, but these **require the spatial coordinates** of each study unit, which the *rotifers* table doesn't have. So, **load the *rotifers01* sample dataset** that comes with the *fuzzySim* package, which is already in wide format and contains the geographical coordinates of the centroid of each TDWG3 unit:

```
data(rotifers01)
```

Take a look at its first rows:

```
head(rotifers01)
```

	LEVEL3_COD	ID	LEVEL_NAME	Lati	Long	Abrigh	Afissa
1	ABT	1	Alberta	54.95520	-114.45960	0	0
2	AFG	2	Afghanistan	33.78802	65.98809	1	0
3	AGE	3	Argentina_Northeast	-32.40092	-61.20684	1	1
4	AGS	4	Argentina_South	-44.10649	-68.78644	1	1
5	AGW	5	Argentina_Northwest	-28.99772	-66.31679	0	1
6	ALA	6	Alabama	32.77805	-86.82992	0	0

	Apriod	Bangul	Bcalyc	Bplica	Bquadr	Burceo	Cgibba	Cobtus	Cunico	Edilat
1	1	1	0	0	0	0	0	0	1	0
2	1	1	1	1	1	1	1	1	1	0
3	0	1	1	1	1	1	1	0	1	1
4	1	1	1	1	1	1	0	1	1	1
5	0	1	1	1	1	1	1	0	0	1
6	0	0	0	0	0	0	0	0	0	0

	Flongi	Hmira	Kcochl	Kquadr	Lacumi	Lbulla	Lclost	Lhamat	Lluna	Llunar
1	1	1	1	1	0	0	0	0	1	1

Show the column names of this dataset, to see which columns contain the species data and which contain the coordinates:

```
names(rotifers01)
```

```
 [1] "LEVEL3_COD" "ID"         "LEVEL_NAME" "Lati"       "Long"
 [6] "Abrigh"     "Afissa"    "Apriod"     "Bangul"     "Bcalyc"
[11] "Bplica"     "Bquadr"    "Burceo"     "Cgibba"     "Cobtus"
[16] "Cunico"     "Edilat"    "Flongi"     "Hmira"      "Kcochl"
[21] "Kquadr"     "Lacumi"    "Lbulla"     "Lclost"     "Lhamat"
[26] "Lluna"      "Llunar"    "Lovali"     "Lpatel"     "Lquadr"
[31] "Mventr"     "Pdolic"    "Ppatul"     "Pquadr"     "Pvulga"
[36] "Specti"     "Tpatin"    "Trattu"     "Tsimil"     "Ttetra"
```

You can see that species are in columns 6 to 40 and coordinates are in columns 4 and 5. You can **use either the names or the index numbers of these columns** in the *multTSA* and *distPres* functions below. Beware that the **coordinates must be specified to the function in the correct order**, i.e. ***x, y*** or ***Longitude, Latitude***! First, try a multiple **trend surface analysis** (TSA) for all species using a **3rd-degree** polynomial with stepwise selection of terms:

```
rotifers.tsa <- multTSA(rotifers01, sp.cols = 6:40, coord.cols = c("Long",
                "Lati"), id.col = 1, degree = 3, step = TRUE)
```

You can find out more about trend surface analysis and about the different options of the *multTSA* function by reading the help file that appears if you type `help(multTSA)`. Now look at the first rows of the *rotifers.tsa* created just above:

```
head(rotifers.tsa)
```

```
  LEVEL3_COD AbrighTS3  AfissaTS3  ApriodTS3
1        ABT 0.1289554 0.07358549 0.44422960
2        AFG 0.3133085 0.30193975 0.31963224
3        AGE 0.2572435 0.20913359 0.04583188
4        AGS 0.2478486 0.19681576 0.04077395
5        AGW 0.2590789 0.21110435 0.04505016
6        ALA 0.2451248 0.19572947 0.22928002
  BangulTS3 BcalycTS3  BplicaTS3 BquadrTS3
1 0.1180535 0.1480850 0.08682276 0.1052991
2 0.3800678 0.4244315 0.31291754 0.3555237
3 0.2117665 0.3295177 0.19786590 0.2789177
4 0.1883593 0.3050704 0.16871519 0.2627841
```

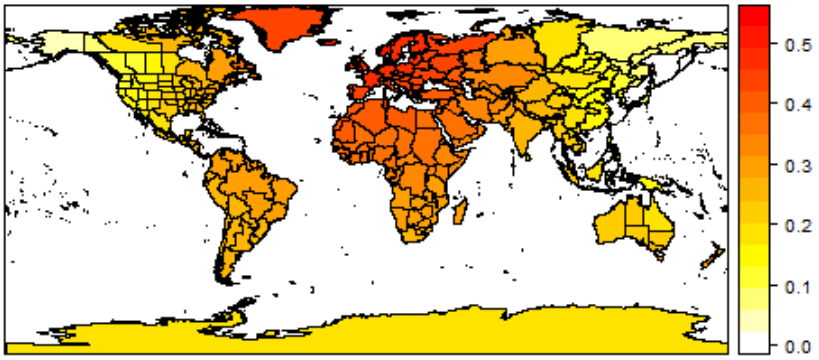
These are **continuous values representing the spatial trend in** each **species' occurrence**. You can **confirm** that **they are bounded between 0 and 1** (so that they can be used in fuzzy logic) by checking the range of values in all columns except the first one (because this one contains the region identifiers rather than species data):

```
range(rotifers.tsa[ , -1])
```

```
[1] 0.003099965 0.980317784
```

Now add the TSA results to the *TDWG3shp* map table and plot the first species (and then others as you like; check `names(TDWG3shp)` for currently available *zcol* options):

```
TDWG3shp@data <- merge(TDWG3shp@data, rotifers.tsa, all.x = TRUE, by =
                       "LEVEL3_COD")

print(spplot(TDWG3shp, zcol = "AbrighTS3", col.regions =
             rev(heat.colors(256))))
```



The TSA depicts a general spatial trend in the species' occurrence, but this may not be a faithful representation of its (fuzzy) occurrence area (compare with the presence-absence map shown before for the same species). You can try *multTSA* again with different polynomial degrees, with or without stepwise selection. Or, you can calculate **inverse distance to presence** to use instead of TSA:

```
rotifers.invdist <- distPres(rotifers01, sp.cols = 6:40, coord.cols =
  c("Long", "Lati"), id.col = 1, p = 1, inv = TRUE, suffix = "_D")
```

You can check `help(distPres)` for more information and options for this function (for example, you may want to **use *p = 2* for** a more conservative **squared distance**, especially if your spatial units are smaller). Check out the first rows of the resulting table:

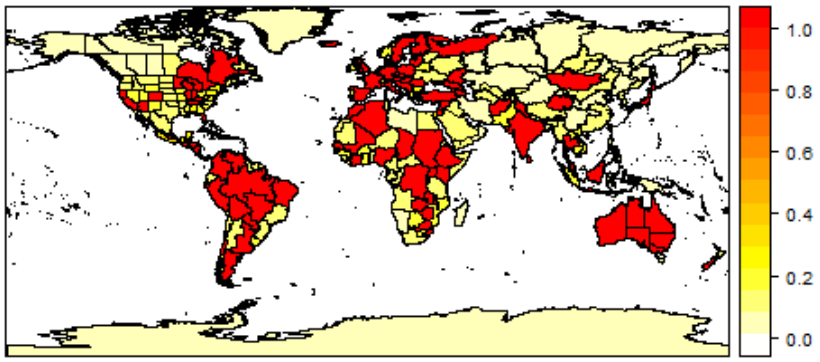
```
head(rotifers.invdist)
```

	LEVEL3_COD	Abrigh_D	Afissa_D	Apriod_D
1	ABT	0.05181824	0.05126359	1.00000000
2	AFG	1.00000000	0.08526539	1.00000000
3	AGE	1.00000000	1.00000000	0.06691079
4	AGS	1.00000000	1.00000000	1.00000000
5	AGW	0.14006602	1.00000000	0.06131478
6	ALA	0.24346099	0.22773074	0.16701188

	Bangul_D	Bcalyc_D	Bplica_D	Bquadr_D
1	1.0000000	0.09385774	0.1040200	0.05957269
2	1.0000000	1.00000000	1.0000000	1.00000000
3	1.0000000	1.00000000	1.0000000	1.00000000

Now add these inverse distances to the *TDWG3shp* map table and plot the first species (then try other species as well):

```
TDWG3shp@data <- merge(TDWG3shp@data, rotifers.invdist, all.x = TRUE, by
                       = "LEVEL3_COD")

print(spplot(TDWG3shp, zcol = "Abrigh_D", col.regions =
                       rev(heat.colors(256))))
```



This seems a more accurate portrait of our species' distribution, so let's use inverse distances as our fuzzy occurrence values. Now **get a matrix of pair-wise fuzzy similarity between these fuzzy species' distributions**, by comparing all columns (except the first, with the region identifiers) in *rotifers.invdist* and using e.g. the fuzzy Baroni similarity index:

```
fuz.sim.mat <- simMat(rotifers.invdist[ , -1], method = "Baroni")
```

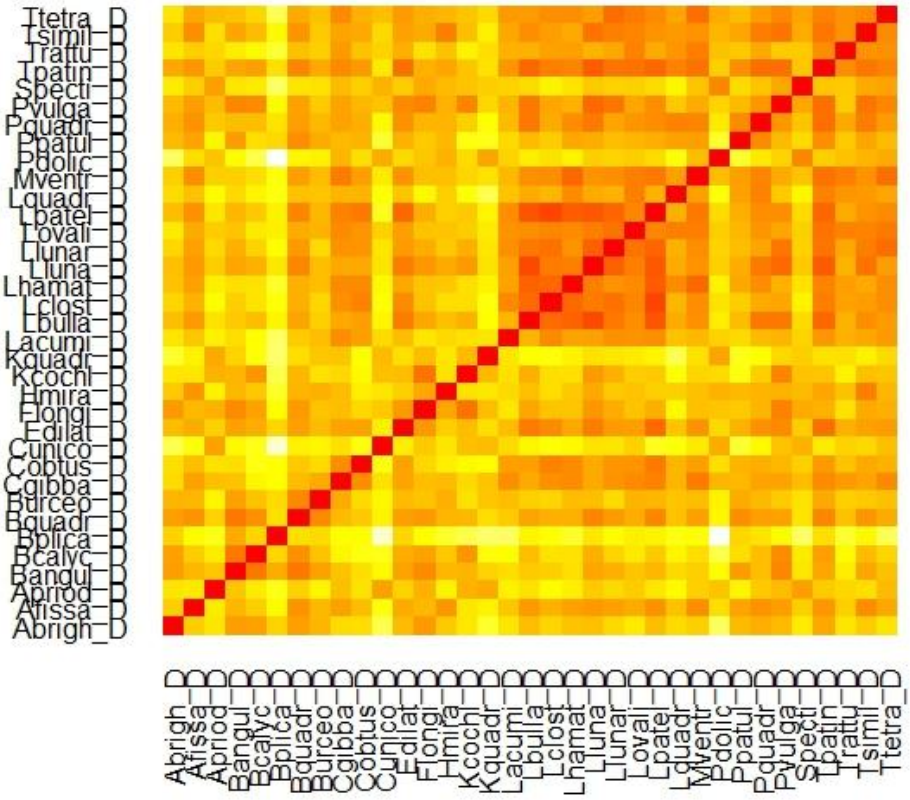
Take a look at the first rows of the resulting fuzzy similarity matrix:

```
head(fuz.sim.mat)
```

```
           Abrigh_D  Afissa_D  Apriod_D  Bangul_D
Abrigh_D 1.0000000 0.7939599 0.7718339 0.8427297
Afissa_D 0.7939599 1.0000000 0.7677697 0.8180130
Apriod_D 0.7718339 0.7677697 1.0000000 0.8020461
Bangul_D 0.8427297 0.8180130 0.8020461 1.0000000
Bcalyc_D 0.8393709 0.7952344 0.7652419 0.8807587
Bplica_D 0.7907104 0.7498984 0.7378773 0.8285949
           Bcalyc_D  Bplica_D  Bquadr_D  Burceo_D
Abrigh_D 0.8393709 0.7907104 0.8404946 0.8198798
Afissa_D 0.7952344 0.7498984 0.8523005 0.8172568
Apriod_D 0.7652419 0.7378773 0.7918134 0.7559885
Bangul_D 0.8807587 0.8285949 0.8808459 0.8386459
Bcalyc_D 1.0000000 0.7843534 0.8569416 0.8066565
```

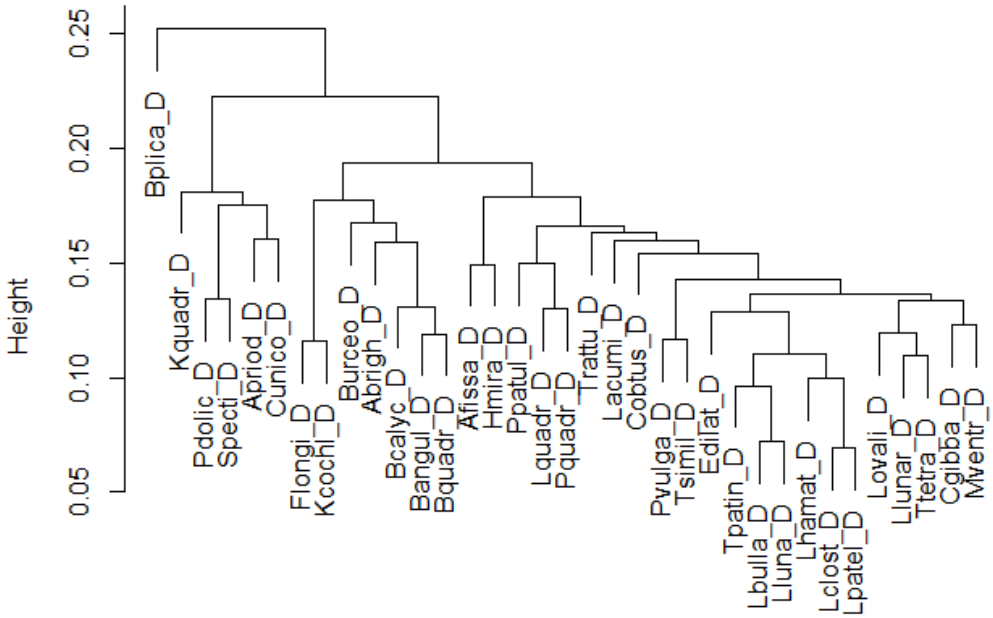
For a quick look at which species pairs are more and less similar in distribution, you can **plot the similarity matrix with a colour scale** (the same one used above for the maps), using the *image* function of R and then adding the species as axis labels (the *cex* option defines the size of the text):

```
image(x = 1:ncol(fuz.sim.mat), y = 1:nrow(fuz.sim.mat), z = fuz.sim.mat,
    col = rev(heat.colors(256)), xlab = "", ylab = "", axes = FALSE)

    axis(side = 1, at = 1:ncol(fuz.sim.mat), tick = FALSE, labels =
            colnames(fuz.sim.mat), las = 2, cex = 0.2)

    axis(side = 2, at = 1:nrow(fuz.sim.mat), tick = FALSE, labels =
            rownames(fuz.sim.mat), las = 2, cex = 0.2)
```



You can also **plot a cluster dendrogram from the similarity matrix**, using the *hclust* R function. The clustering requires a **distance matrix**, so the **similarity matrix** is **subtracted from 1** in the command below. The *method = "average"* option implies that UPGMA is the clustering algorithm, but you can check for other options with `help(hclust)`:

```
plot(hclust(as.dist(1 - fuz.sim.mat), method = "average"))
```



You can also **build a similarity matrix from the original binary presence-absence data**, to **compare with the fuzzy similarity results**. First check again the column names of *rotifers01*, to see where the species columns are so that you can specify them correctly to the *simMat* function:

```
names(rotifers01)
```

```
 [1] "LEVEL3_COD" "ID"         "LEVEL_NAME" "Lati"
 [5] "Long"       "Abrigh"     "Afissa"     "Apriod"
 [9] "Bangul"     "Bcalyc"     "Bplica"     "Bquadr"
[13] "Burceo"     "Cgibba"     "Cobtus"     "Cunico"
[17] "Edilat"     "Flongi"     "Hmira"      "Kcochl"
[21] "Kquadr"     "Lacumi"     "Lbulla"     "Lclost"
[25] "Lhamat"     "Lluna"      "Llunar"     "Lovali"
[29] "Lpatel"     "Lquadr"     "Mventr"     "Pdolic"
[33] "Ppatul"     "Pquadr"     "Pvulga"     "Specti"
[37] "Tpatin"     "Trattu"     "Tsimil"     "Ttetra"
```

Now calculate the binary similarity matrix:

```
bin.sim.mat <- simMat(rotifers01[ , 6:40], method = "Baroni")
```

You can try and **repeat the operations exemplified before**, but **replacing *fuz.sim.mat* with *bin.sim.mat***, to visualize the binary similarity matrix and the resulting dendrogram. You can also **compare the fuzzy and binary similarity matrices using the *mantel* function** of the *vegan* R package. If you want to do this, the command below will install *vegan* within your R installation if you don't already have it:

```
if (!("vegan" %in% rownames(installed.packages())))
                install.packages("vegan")
```

Now **load *vegan*** into the current R session and **calculate the Mantel correlation between the two matrices** (type `help(mantel)` for more info and options):

```
library(vegan)

mantel(bin.sim.mat, fuz.sim.mat, method = "spearman")
```

```
Mantel statistic based on Spearman's rank correlation rho

Call:
mantel(xdis = bin.sim.mat, ydis = fuz.sim.mat, method = "spe
arman")

Mantel statistic r: 0.9935
      Significance: 0.001

Upper quantiles of permutations (null model):
  90%   95% 97.5%   99%
0.125 0.165 0.197 0.229

Based on 999 permutations
```

Besides comparing species according to their (fuzzy) occurrence patterns, you can also compare regions according to their (fuzzy) species composition. For this you only need to **transpose the (fuzzy) species occurrence matrix** so that regions go in columns and species in rows. With the *transpose* function of *fuzzySim*, you can do this directly on the complete tables, **specifying which columns contain the species occurrence data** to transpose and **which column contains the region names** to use as column names in the transposed table:

```
bin.reg <- transpose(rotifers01, sp.cols = 6:40, reg.names = 1)
```

```
fuz.reg <- transpose(rotifers.invdist, sp.cols = 2:36, reg.names = 1)
```

Look at the first rows of the resulting tables:

```
head(bin.reg)
```

	ABT	AFG	AGE	AGS	AGW	ALA	ALB	ALD	ALG	ALT	ALU	AMU	AND	ANG	ANT	ARI	ARK
Abrigh	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0
Afissa	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0
Apriod	1	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
Bangul	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0
Bcalyc	0	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0
Bplica	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0

	ARU	ASC	ASK	ASP	ASS	ATP	AUT	AZO	BAH	BAL	BAN	BEN	BER	BGM	BIS	BKN	BLR
Abrigh	1	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0
Afissa	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0	0	0
Apriod	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
Bangul	1	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0
Bcalyc	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1
Bplica	1	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0

```
head(fuz.reg)
```

	ABT	AFG	AGE	AGS	AGW	ALA
Abrigh_D	0.05181824	1.00000000	1.00000000	1	0.14006602	0.2434610
Afissa_D	0.05126359	0.08526539	1.00000000	1	1.00000000	0.2277307
Apriod_D	1.00000000	1.00000000	0.06691079	1	0.06131478	0.1670119
Bangul_D	1.00000000	1.00000000	1.00000000	1	1.00000000	0.1670119
Bcalyc_D	0.09385774	1.00000000	1.00000000	1	1.00000000	0.2277307
Bplica_D	0.10401999	1.00000000	1.00000000	1	1.00000000	0.1423563

	ALB	ALD	ALG	ALT	ALU
Abrigh_D	0.2342681	0.07318731	1.00000000	0.04843323	0.06661354
Afissa_D	0.2410574	0.09161673	0.07481251	0.04740074	0.06189768
Apriod_D	0.2410574	0.04977166	1.00000000	0.06663151	0.12640487
Bangul_D	0.2410574	0.09161673	1.00000000	0.05448215	0.12640487
Bcalyc_D	0.2410574	0.09161673	1.00000000	0.06663151	0.06820941
Bplica_D	0.2410574	0.21734032	1.00000000	0.05448215	0.29049129

Now create the pair-wise similarity matrices for both binary and fuzzy species composition in these regions. These matrices will take longer to calculate because there are (in this dataset) many more regions than species, so there are many more pair-wise comparisons to make:

```
bin.reg.sim.mat <- simMat(bin.reg, method = "Baroni")
```

```
fuz.reg.sim.mat <- simMat(fuz.reg, method = "Baroni")
```

Then you can proceed as you did before with the species distributional similarity matrices, to plot, compare and build cluster dendrograms of these data. The matrices can also be entered in the **RMACOQUI package**, which is soon to be released, for a systematic analysis of **chorotypes** (significant clusters of species distribution types) or of **biotic regions** (significant clusters of regional species compositions).

If you want to try this out with your own data, get your table (with column names in the first row) in a *.txt* file separated by tabs, save it in your R working directory (type `getwd()` to find out which it is) and then import it to R using the following command:

```
mydata <- read.table("mydata.txt", header = TRUE, sep = "\t")
```

Then reproduce all the operations above, but replacing *rotifers* or *rotifers01* (depending on how your data are organized) with *mydata* (or whatever name you've assigned in the command above) and specifying column names or numbers accordingly.

That's it! You can send me an e-mail if you have any suggestions or concerns.