A beginners tutorial on the *fuzzySim* R package

A. Márcia Barbosa

CIBIO/InBIO – University of Évora (Portugal); barbosa@uevora.pt

updated 2 February 2015

[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 should work if you have the **latest version of R**; otherwise, it may either fail (producing a message like "*package 'fuzzySim' is not available for your R version*") or show a warning and install an older version of *fuzzySim*. To **check the version that you have actually installed**, type `citation(package="fuzzySim")`. To install the latest version of the package, you can either upgrade R *or* download the compressed *fuzzySim* **package source files** to your disk (*.zip* or *.tar.gz* available at the package development page, https://r-forge.r-project.org/R/?group_id=1853) and then install the package from there, e.g. with R menu "*Packages - Install packages from local zip files*" (Windows), or "*Packages & Data - Package installer, Packages repository - Local source package*" (Mac), or "*Tools - Install packages - Install from: Package Archive File*" (RStudio).

You only need to install the package once (unless a new version becomes available), but you need to **load it every time you open a new R session** in which you intend to use *fuzzySim* (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:

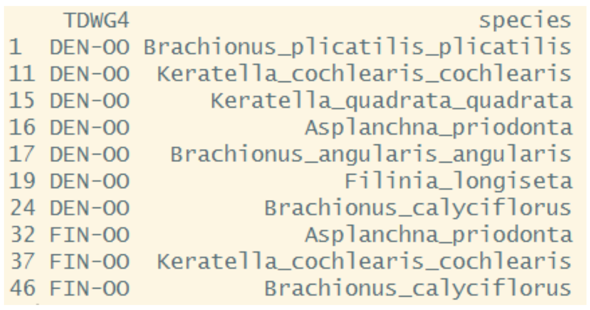
```
data(rotifers)
```

You can get more information on this dataset (the following command should open an *R Documentation* window):

```
help(rotifers)
```

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

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



```
   TDWG4                           species
1  DEN-OO Brachionus_plicatilis_plicatilis
11 DEN-OO  Keratella_cochlearis_cochlearis
15 DEN-OO      Keratella_quadrata_quadrata
16 DEN-OO             Asplanchna_priodonta
17 DEN-OO   Brachionus_angularis_angularis
19 DEN-OO                Filinia_longiseta
24 DEN-OO           Brachionus_calyciflorus
32 FIN-OO              Asplanchna_priodonta
37 FIN-OO  Keratella_cochlearis_cochlearis
46 FIN-OO           Brachionus_calyciflorus
```

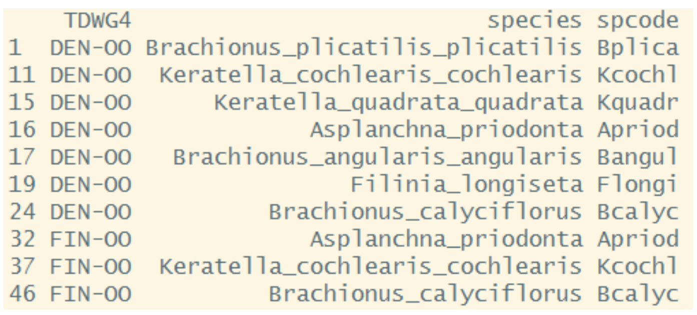
The first column contains the identifiers of the spatial units, which are TDWG level 4 region codes, and the second column contains the (sub)species names. These are a bit long, especially if we intend to use them as column names further on, so use the *spCodes* function to **add to the *rotifers* dataset 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 this function with `help(spCodes)`.

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

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



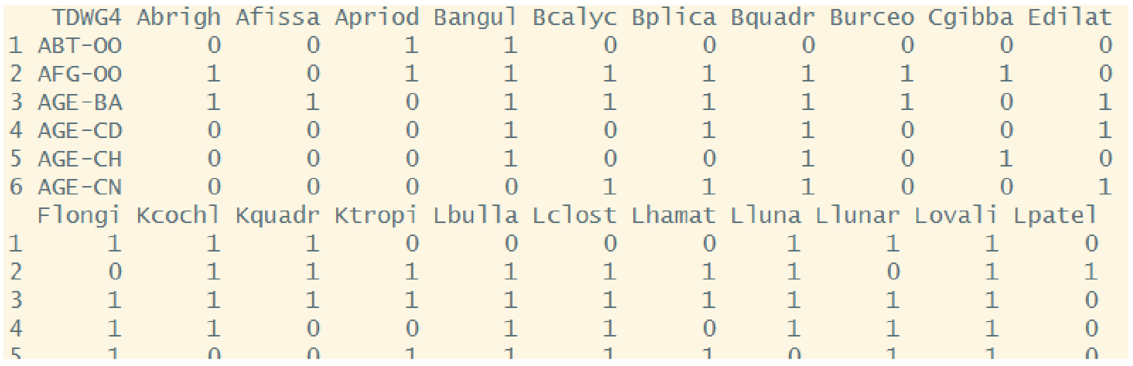
```
   TDWG4                          species spcode
1  DEN-OO Brachionus_plicatilis_plicatilis Bplica
11 DEN-OO  Keratella_cochlearis_cochlearis Kcochl
15 DEN-OO      Keratella_quadrata_quadrata Kquadr
16 DEN-OO             Asplanchna_priodonta Apriod
17 DEN-OO   Brachionus_angularis_angularis Bangul
19 DEN-OO                Filinia_longiseta Flongi
24 DEN-OO           Brachionus_calyciflorus Bcalyc
32 FIN-OO              Asplanchna_priodonta Apriod
37 FIN-OO  Keratella_cochlearis_cochlearis Kcochl
46 FIN-OO           Brachionus_calyciflorus Bcalyc
```

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 species in separate columns (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 = "TDWG4", 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)
```



```
   TDWG4 Abrigh Afissa Apriod Bangul Bcalyc Bplica Bquadr Burceo Cgibba Edilat
1 ABT-OO      0      0      1      1      0      0      0      0      0      0
2 AFG-OO      1      0      1      1      1      1      1      1      1      0
3 AGE-BA      1      1      0      1      1      1      1      1      0      1
4 AGE-CD      0      0      0      1      0      1      1      0      0      1
5 AGE-CH      0      0      0      1      0      0      1      0      1      0
6 AGE-CN      0      0      0      0      1      1      1      0      0      1
  Flongi Kcochl Kquadr Ktropi Lbulla Lclost Lhamat Lluna Llunar Lovali Lpatel
1      1      1      1      0      0      0      0     1      1      1      0
2      0      1      1      1      1      1      1     1      0      1      1
3      1      1      1      1      1      1      1     1      1      1      0
4      1      1      0      0      1      1      0     1      1      1      0
5      1      0      0      1      1      1      1     0      1      1      0
```

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 4 shapefile** into that folder (if you're connected to the internet) **and unzip it**:

```
dir.create("TDWG4_shapefile")

download.file("http://www.kew.org/gis/tdwg/downloads/level4.zip",
  destfile = "TDWG4_shapefile/TDWG_level4.zip", method = "auto")

unzip("TDWG4_shapefile/TDWG_level4.zip", exdir = "TDWG4_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 to download from (choose any).

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

Now **load the *rgdal* package** and **create a map** named *TDWG4shp* by **importing the shapefile** you've downloaded before, using the *readOGR* function of *rgdal*:

```
library(rgdal)

TDWG4shp <- readOGR(dsn = "TDWG4_shapefile", layer = "level4")
```

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

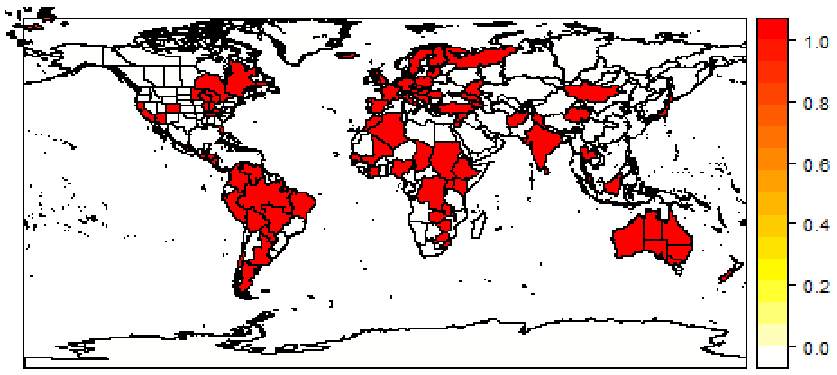
```
unlink("TDWG4_shapefile", recursive = TRUE)
```

Now **add the *rotifers.presabs* data to the map's attribute table** within R, matching them by the name of the column containing the region identifiers in both tables (*TDWG4* in this case):

```
TDWG4shp@data <- data.frame(TDWG4shp@data,
rotifers.presabs[match(TDWG4shp@data$Level4_cod,
rotifers.presabs$TDWG4), ])
```

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, which 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(TDWG4shp, zcol = "Abrigh", col.regions =
rev(heat.colors(256))))
```

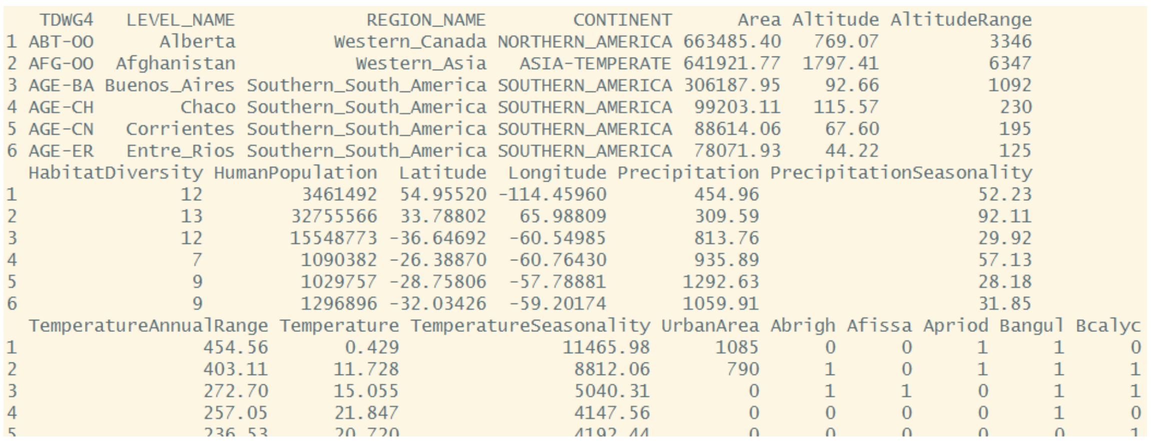


Try this also with *zcol* = other species in `names(TDWG4shp)`. As you can see in the maps and in the *rotifers.presabs* table, these are **binary** (either 0 or 1) **presence-absence data**. You can **get a fuzzy** (continuous between 0 and 1) **version** of such data using e.g. **trend surface analysis** or **inverse distance interpolation**, but these **require the spatial coordinates** of each study unit, which the *rotifers* table doesn't have. So, **load the *rotif.env* sample dataset** that also comes with the *fuzzySim* package, which is already in wide format and contains the geographical coordinates of the centroid of each TDWG3 unit:

```
data(rotif.env)
```

Take a look at its first rows:

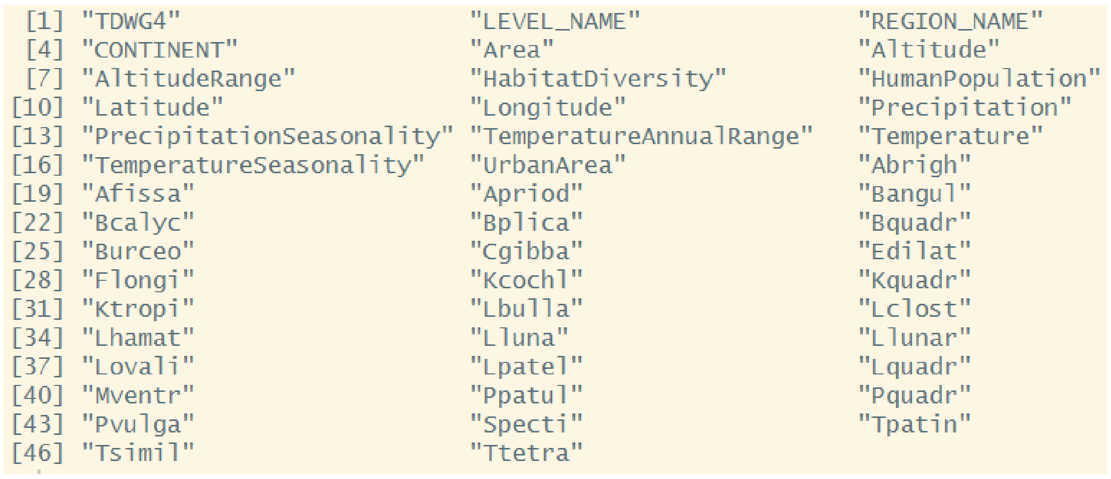
```
head(rotif.env)
```



```
  TDWG4   LEVEL_NAME            REGION_NAME         CONTINENT      Area Altitude AltitudeRange
1 ABT-OO      Alberta         Western_Canada NORTHERN_AMERICA 663485.40   769.07          3346
2 AFG-OO  Afghanistan           Western_Asia   ASIA-TEMPERATE 641921.77  1797.41          6347
3 AGE-BA Buenos_Aires Southern_South_America SOUTHERN_AMERICA 306187.95    92.66          1092
4 AGE-CH        Chaco Southern_South_America SOUTHERN_AMERICA  99203.11   115.57           230
5 AGE-CN   Corrientes Southern_South_America SOUTHERN_AMERICA  88614.06    67.60           195
6 AGE-ER   Entre_Rios Southern_South_America SOUTHERN_AMERICA  78071.93    44.22           125
  HabitatDiversity HumanPopulation  Latitude  Longitude Precipitation PrecipitationSeasonality
1               12         3461492  54.95520 -114.45960        454.96                    52.23
2               13        32755566  33.78802   65.98809        309.59                    92.11
3               12        15548773 -36.64692  -60.54985        813.76                    29.92
4                7         1090382 -26.38870  -60.76430        935.89                    57.13
5                9         1029757 -28.75806  -57.78881       1292.63                    28.18
6                9         1296896 -32.03426  -59.20174       1059.91                    31.85
  TemperatureAnnualRange Temperature TemperatureSeasonality UrbanArea Abrigh Afissa Apriod Bangul Bcalyc
1                 454.56       0.429               11465.98      1085      0      0      1      1      0
2                 403.11      11.728                8812.06       790      1      0      1      1      1
3                 272.70      15.055                5040.31         0      1      1      0      1      1
4                 257.05      21.847                4147.56         0      0      0      0      1      0
5                 236.53      20.720                4192.44         0      0      0      0      0      1
```

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

```
names(rotif.env)
```



```
 [1] "TDWG4"                    "LEVEL_NAME"               "REGION_NAME"
 [4] "CONTINENT"                "Area"                     "Altitude"
 [7] "AltitudeRange"            "HabitatDiversity"         "HumanPopulation"
[10] "Latitude"                 "Longitude"                "Precipitation"
[13] "PrecipitationSeasonality" "TemperatureAnnualRange"   "Temperature"
[16] "TemperatureSeasonality"   "UrbanArea"                "Abrigh"
[19] "Afissa"                   "Apriod"                   "Bangul"
[22] "Bcalyc"                   "Bplica"                   "Bquadr"
[25] "Burceo"                   "Cgibba"                   "Edilat"
[28] "Flongi"                   "Kcochl"                   "Kquadr"
[31] "Ktropi"                   "Lbulla"                   "Lclost"
[34] "Lhamat"                   "Lluna"                    "Llunar"
[37] "Lovali"                   "Lpatel"                   "Lquadr"
[40] "Mventr"                   "Ppatul"                   "Pquadr"
[43] "Pvulga"                   "Specti"                   "Tpatin"
[46] "Tsimil"                   "Ttetra"
```

You can see that species are in columns 18 to 47 and geographical coordinates are in columns 10 and 11. 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(rotif.env, sp.cols = 18:47, coord.cols =
c("Longitude", "Latitude"), 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)
```

```
   TDWG4 Abrigh_TS Afissa_TS  Apriod_TS Bangul_TS Bcalyc_TS Bplica_TS Bquadr_TS
1 ABT-OO 0.1798791 0.1072166 0.56679570 0.1749335 0.2038724 0.1290277 0.1699564
2 AFG-OO 0.4336780 0.4672791 0.41020511 0.6047653 0.6623104 0.4086573 0.5600267
3 AGE-BA 0.2415516 0.1213213 0.06354302 0.1983921 0.5761701 0.2155371 0.4680020
4 AGE-CH 0.3015052 0.1862872 0.06868535 0.2698491 0.5684838 0.2741569 0.4596102
5 AGE-CN 0.2838745 0.1681223 0.07063465 0.2515481 0.5786859 0.2594142 0.4673258
6 AGE-ER 0.2666956 0.1479148 0.06690217 0.2291874 0.5768053 0.2415501 0.4669809
  Burceo_TS Cgibba_TS Edilat_TS Flongi_TS Kcochl_TS Kquadr_TS  Ktropi_TS Lbulla_TS
1 0.1574877 0.4140185 0.4440554 0.4722511 0.8551909 0.5672359 0.01957571 0.4950867
2 0.4654481 0.2779111 0.5078315 0.5350268 0.5613394 0.4210410 0.46405849 0.6196688
3 0.1564815 0.1053790 0.6822418 0.5986828 0.6717552 0.2279913 0.47980394 0.4638878
4 0.1743961 0.2231089 0.5705167 0.5354029 0.5418227 0.1312679 0.41957365 0.4795597
```

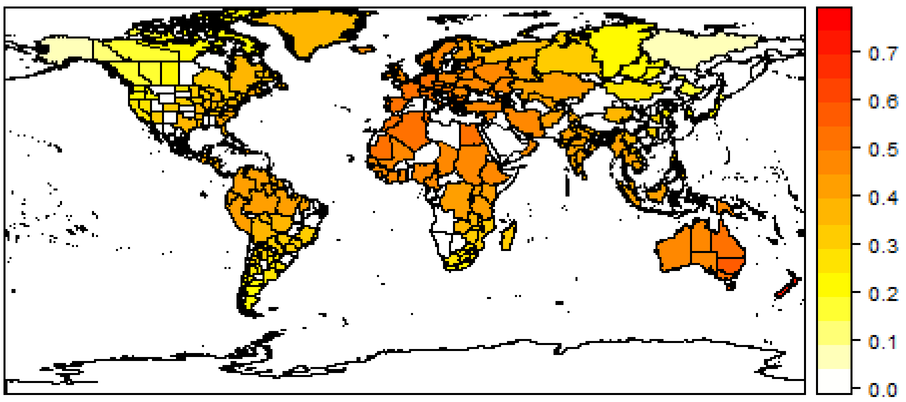
These are **continuous values representing the spatial trend in** each **species' occurrence**. You can **confirm** that **they are bounded between 0 and 1** (so that they can be used in fuzzy logic) by checking the range of values in all columns except the first one (which contains region identifiers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

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

```
[1] 9.408071e-05 9.996004e-01
```

Now add the TSA results to the *TDWG4shp* map table and plot the first species (and then others as you like; check `names(TDWG4shp)` for available *zcol* options):

```
TDWG4shp@data <- data.frame(TDWG4shp@data,
rotifers.tsa[match(TDWG4shp@data$Level4_cod, rotifers.tsa$TDWG4), ])

print(spplot(TDWG4shp, zcol = "Abrigh_TS", col.regions =
rev(heat.colors(256))))
```



The TSA depicts a general spatial trend in a 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(rotif.env, sp.cols = 18:47, coord.cols =
c("Longitude", "Latitude"), 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)
```

```
   TDWG4    Abrigh_D    Afissa_D    Apriod_D  Bangul_D    Bcalyc_D  Bplica_D
1 ABT-OO 0.05181824 0.05126359 1.00000000 1.0000000 0.09385773 0.1040200
2 AFG-OO 1.00000000 0.09194394 1.00000000 1.0000000 1.00000000 1.0000000
3 AGE-BA 1.00000000 1.00000000 0.05007529 1.0000000 1.00000000 1.0000000
4 AGE-CH 0.20166025 0.20166025 0.04244879 1.0000000 0.25938874 0.2593887
5 AGE-CN 0.21368960 0.21368960 0.03821192 0.2136896 1.00000000 1.0000000
6 AGE-ER 0.31123266 0.31123266 0.04038601 0.3112327 1.00000000 0.3112327
    Bquadr_D    Burceo_D    Cgibba_D    Edilat_D    Flongi_D  Kcochl_D  Kquadr_D
1 0.05957269 0.05957269 0.08999225 0.09385773 1.00000000 1.0000000 1.0000000
2 1.00000000 1.00000000 1.00000000 0.09194394 0.09194394 1.0000000 1.0000000
3 1.00000000 1.00000000 0.14290421 1.00000000 1.00000000 1.0000000 1.0000000
4 1.00000000 0.20166025 1.00000000 0.25938874 1.00000000 0.2593887 0.2081774
```

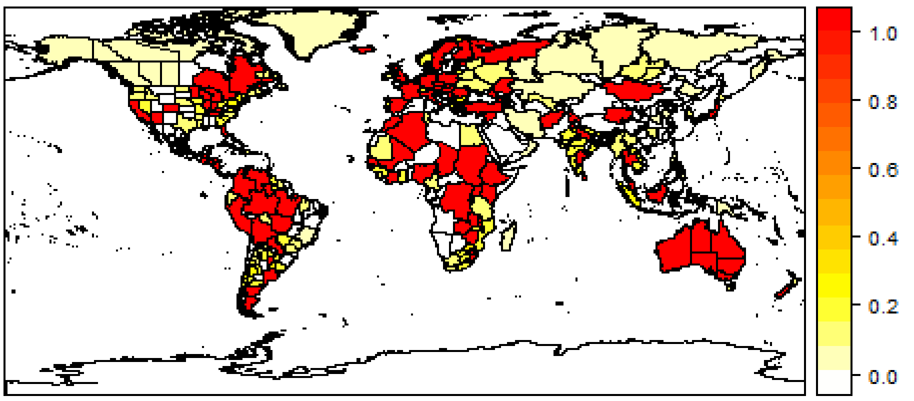
You can check that the values in this table (excluding the first column) also range between 0 and 1, so they can be used with fuzzy logic:

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

```
[1] 0.009745522 1.000000000
```

Note that inverse distance to presence is calculated only for absence localities; *distPres* maintains the value 1 for presences. Now add these distances to the *TDWG4shp* map table and plot the first species (then try other species as well):

```
TDWG4shp@data <- data.frame(TDWG4shp@data, rotifers.invdist
[match(TDWG4shp@data$Level4_cod,rotifers.invdist$TDWG4), ])

print(spplot(TDWG4shp, zcol = "Abrigh_D", col.regions =
rev(heat.colors(256))))
```

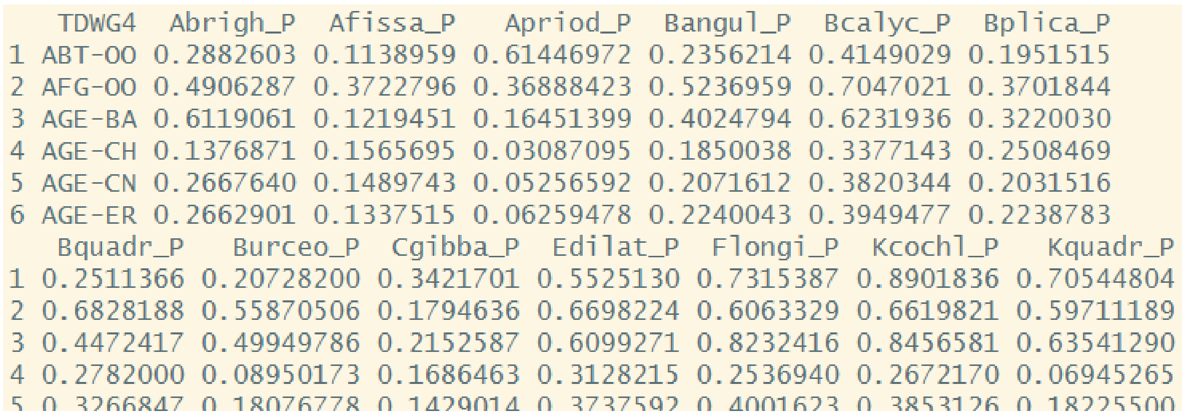


This seems a more faithful portrait of our species' distribution than the TSA. However, note that distance is also not always a good fuzzy representation of a species' occurrence area, as geographical and environmental barriers may cause sharp local variations in species' occurrence patterns. Another way of obtaining a fuzzy versions of species occurrence is to build distribution models based on the relationship between species presence/absence and a set of geographical, environmental and/or human variables. This can be done for multiple species simultaneously with the *multGLM* function and the variables in *rotif.env*. You must specify the name of the data set, the index numbers of the columns containing the species data and the variables, and there are a series of options on how to select variables for the models – read `help(multGLM)` for more details:

```
rotifers.fav <- multGLM(data = rotif.env, sp.cols = 18:47, var.cols =
                  5:17, id.col = 1, Favourability = TRUE)
```

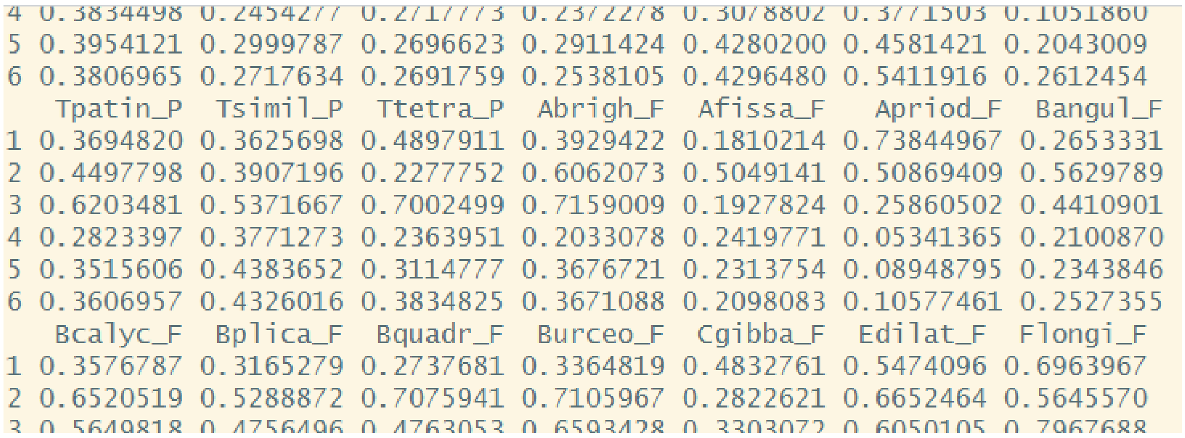
The object output by *multGLM* is a list containing to elements: another list named *models*, and a dataframe with the resulting *predictions*. Check out its first rows:

```
head(rotifers.fav$predictions)
```



```
   TDWG4  Abrigh_P  Afissa_P   Apriod_P  Bangul_P  Bcalyc_P  Bplica_P
1 ABT-OO 0.2882603 0.1138959 0.61446972 0.2356214 0.4149029 0.1951515
2 AFG-OO 0.4906287 0.3722796 0.36888423 0.5236959 0.7047021 0.3701844
3 AGE-BA 0.6119061 0.1219451 0.16451399 0.4024794 0.6231936 0.3220030
4 AGE-CH 0.1376871 0.1565695 0.03087095 0.1850038 0.3377143 0.2508469
5 AGE-CN 0.2667640 0.1489743 0.05256592 0.2071612 0.3820344 0.2031516
6 AGE-ER 0.2662901 0.1337515 0.06259478 0.2240043 0.3949477 0.2238783
   Bquadr_P   Burceo_P  Cgibba_P  Edilat_P  Flongi_P  Kcochl_P   Kquadr_P
1 0.2511366 0.20728200 0.3421701 0.5525130 0.7315387 0.8901836 0.70544804
2 0.6828188 0.55870506 0.1794636 0.6698224 0.6063329 0.6619821 0.59711189
3 0.4472417 0.49949786 0.2152587 0.6099271 0.8232416 0.8456581 0.63541290
4 0.2782000 0.08950173 0.1686463 0.3128215 0.2536940 0.2672170 0.06945265
5 0.3266847 0.18076778 0.1429014 0.3737592 0.4001623 0.3853126 0.18225500
```

[...]



```
4 0.3834498 0.2454277 0.2717773 0.2372278 0.3078802 0.3771503 0.1051860
5 0.3954121 0.2999787 0.2696623 0.2911424 0.4280200 0.4581421 0.2043009
6 0.3806965 0.2717634 0.2691759 0.2538105 0.4296480 0.5411916 0.2612454
   Tpatin_P  Tsimil_P  Ttetra_P  Abrigh_F  Afissa_F   Apriod_F  Bangul_F
1 0.3694820 0.3625698 0.4897911 0.3929422 0.1810214 0.73844967 0.2653331
2 0.4497798 0.3907196 0.2277752 0.6062073 0.5049141 0.50869409 0.5629789
3 0.6203481 0.5371667 0.7002499 0.7159009 0.1927824 0.25860502 0.4410901
4 0.2823397 0.3771273 0.2363951 0.2033078 0.2419771 0.05341365 0.2100870
5 0.3515606 0.4383652 0.3114777 0.3676721 0.2313754 0.08948795 0.2343846
6 0.3606957 0.4326016 0.3834825 0.3671088 0.2098083 0.10577461 0.2527355
   Bcalyc_F  Bplica_F  Bquadr_F  Burceo_F  Cgibba_F  Edilat_F  Flongi_F
1 0.3576787 0.3165279 0.2737681 0.3364819 0.4832761 0.5474096 0.6963967
2 0.6520519 0.5288872 0.7075941 0.7105967 0.2822621 0.6652464 0.5645570
3 0.5649818 0.4756496 0.4763053 0.6593428 0.3303072 0.6050105 0.7967688
```

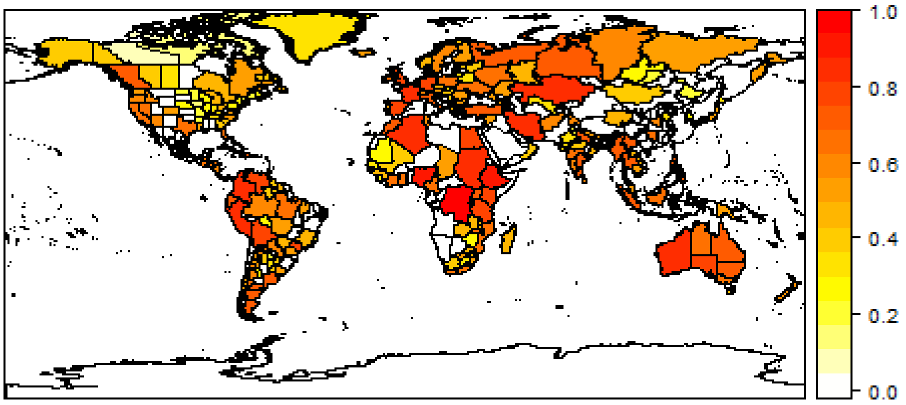
We're interested in the columns with suffix "_F", as those contain favourability values, which are directly comparable among species (whereas probability is affected by species prevalence; type `help(multGLM)` for more info).

```
names(rotifers.fav$predictions)
```

```
 [1] "TDWG4"    "Abrigh_P" "Afissa_P" "Apriod_P" "Bangul_P" "Bcalyc_P" "Bplica_P"
 [8] "Bquadr_P" "Burceo_P" "Cgibba_P" "Edilat_P" "Flongi_P" "Kcochl_P" "Kquadr_P"
[15] "Ktropi_P" "Lbulla_P" "Lclost_P" "Lhamat_P" "Lluna_P"  "Llunar_P" "Lovali_P"
[22] "Lpatel_P" "Lquadr_P" "Mventr_P" "Ppatul_P" "Pquadr_P" "Pvulga_P" "Specti_P"
[29] "Tpatin_P" "Tsimil_P" "Ttetra_P" "Abrigh_F" "Afissa_F" "Apriod_F" "Bangul_F"
[36] "Bcalyc_F" "Bplica_F" "Bquadr_F" "Burceo_F" "Cgibba_F" "Edilat_F" "Flongi_F"
[43] "Kcochl_F" "Kquadr_F" "Ktropi_F" "Lbulla_F" "Lclost_F" "Lhamat_F" "Lluna_F"
[50] "Llunar_F" "Lovali_F" "Lpatel_F" "Lquadr_F" "Mventr_F" "Ppatul_F" "Pquadr_F"
[57] "Pvulga_F" "Specti_F" "Tpatin_F" "Tsimil_F" "Ttetra_F"
```

You see that favourability values are in columns 32 to 61 of the *rotifers.fav$predictions* table. Now add these values to the *TDWG4shp* map table and plot e.g. the first species:

```
TDWG4shp@data <- data.frame(TDWG4shp@data,
rotifers.fav$predictions[match(TDWG4shp@data$Level4_cod,rotifers.fav
$predictions$TDWG4), ])

print(spplot(TDWG4shp, zcol = "Abrigh_F", col.regions =
rev(heat.colors(256))))
```

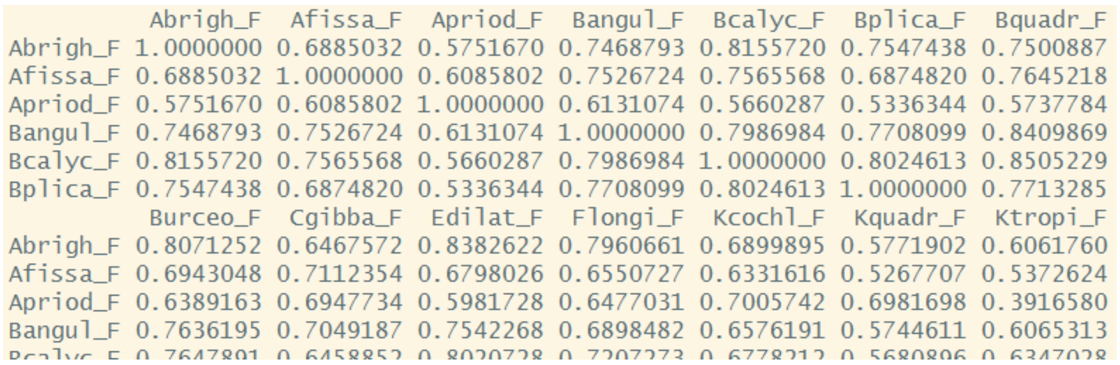


Let’s now use these favourability model predictions as our fuzzy occurrence values. First, **get a matrix of pair-wise fuzzy similarity among these fuzzy species’ distributions**, by comparing these columns with e.g. the fuzzy Jaccard similarity index:

```
fuz.sim.mat <- simMat(rotifers.fav$predictions[ , 32:61], method =
"Jaccard")
```

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

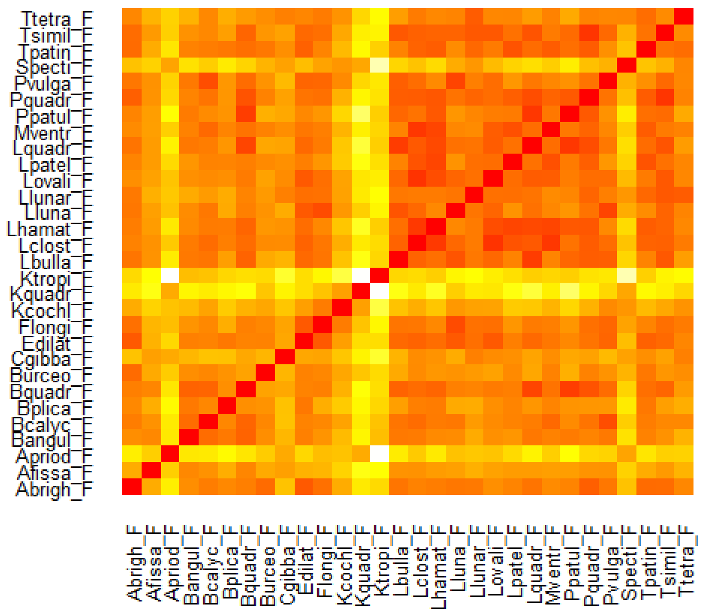
```
head(fuz.sim.mat)
```



```
           Abrigh_F  Afissa_F  Apriod_F  Bangul_F  Bcalyc_F  Bplica_F  Bquadr_F
Abrigh_F 1.0000000 0.6885032 0.5751670 0.7468793 0.8155720 0.7547438 0.7500887
Afissa_F 0.6885032 1.0000000 0.6085802 0.7526724 0.7565568 0.6874820 0.7645218
Apriod_F 0.5751670 0.6085802 1.0000000 0.6131074 0.5660287 0.5336344 0.5737784
Bangul_F 0.7468793 0.7526724 0.6131074 1.0000000 0.7986984 0.7708099 0.8409869
Bcalyc_F 0.8155720 0.7565568 0.5660287 0.7986984 1.0000000 0.8024613 0.8505229
Bplica_F 0.7547438 0.6874820 0.5336344 0.7708099 0.8024613 1.0000000 0.7713285
           Burceo_F  Cgibba_F  Edilat_F  Flongi_F  Kcochl_F  Kquadr_F  Ktropi_F
Abrigh_F 0.8071252 0.6467572 0.8382622 0.7960661 0.6899895 0.5771902 0.6061760
Afissa_F 0.6943048 0.7112354 0.6798026 0.6550727 0.6331616 0.5267707 0.5372624
Apriod_F 0.6389163 0.6947734 0.5981728 0.6477031 0.7005742 0.6981698 0.3916580
Bangul_F 0.7636195 0.7049187 0.7542268 0.6898482 0.6576191 0.5744611 0.6065313
Bcalyc_F 0.7647891 0.6458852 0.8020728 0.7207273 0.6778212 0.5680896 0.6347028
```

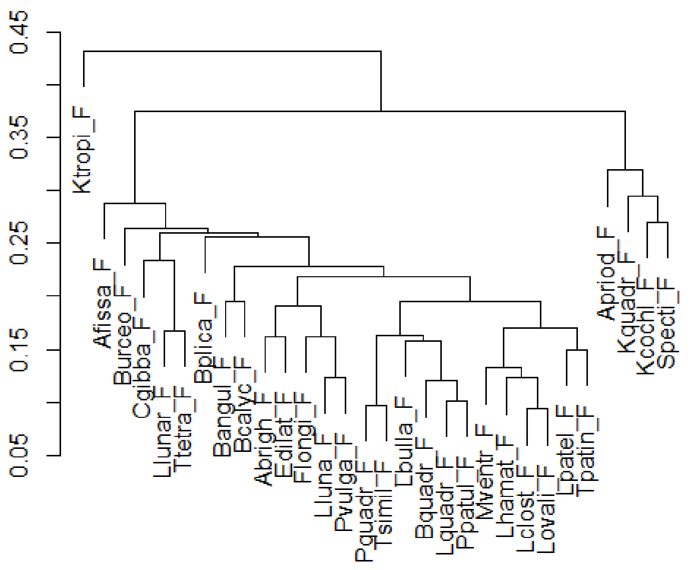
For a quick look at which species pairs are more and less similar in distribution, you can **plot the similarity matrix with a colour scale** (the same used above for the maps), using the *image* function of R and then adding the species as axis labels (the *cex.axis* argument defines the text size):

```
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.axis = 0.8)

axis(side = 2, at = 1:nrow(fuz.sim.mat), tick = FALSE, labels =
    rownames(fuz.sim.mat), las = 2, cex.axis = 0.8)
```



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 *rotif.env*, to see where the species columns are so that you can specify them correctly to the *simMat* function:

```
names(rotif.env)
```

```
 [1] "TDWG4"                      "LEVEL_NAME"                "REGION_NAME"
 [4] "CONTINENT"                  "Area"                      "Altitude"
 [7] "AltitudeRange"              "HabitatDiversity"          "HumanPopulation"
[10] "Latitude"                   "Longitude"                 "Precipitation"
[13] "PrecipitationSeasonality"   "TemperatureAnnualRange"    "Temperature"
[16] "TemperatureSeasonality"     "UrbanArea"                 "Abrigh"
[19] "Afissa"                     "Apriod"                    "Bangul"
[22] "Bcalyc"                     "Bplica"                    "Bquadr"
[25] "Burceo"                     "Cgibba"                    "Edilat"
[28] "Flongi"                     "Kcochl"                    "Kquadr"
[31] "Ktropi"                     "Lbulla"                    "Lclost"
[34] "Lhamat"                     "Lluna"                     "Llunar"
[37] "Lovali"                     "Lpatel"                    "Lquadr"
[40] "Mventr"                     "Ppatul"                    "Pquadr"
[43] "Pvulga"                     "Specti"                    "Tpatin"
[46] "Tsimil"                     "Ttetra"
```

Now calculate the binary similarity matrix:

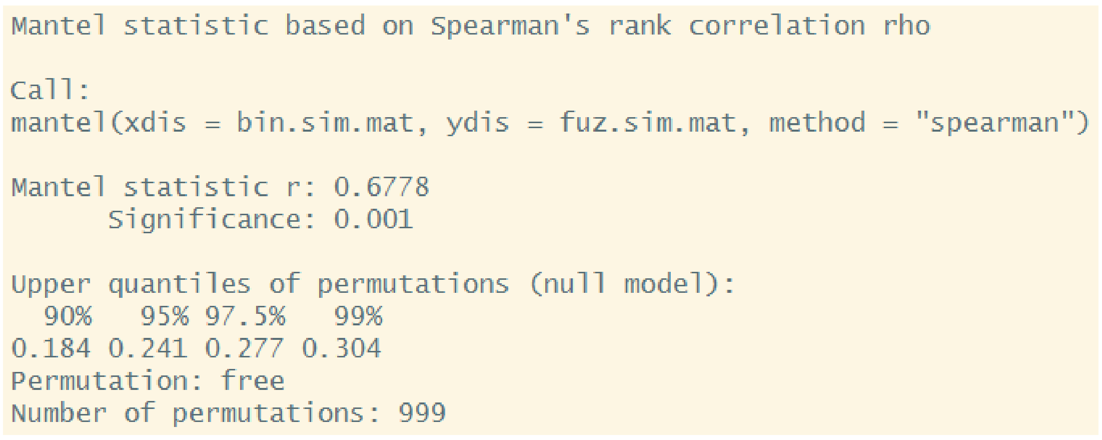
```
bin.sim.mat <- simMat(rotif.env[ , 18:47], method = "Jaccard")
```

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")
```



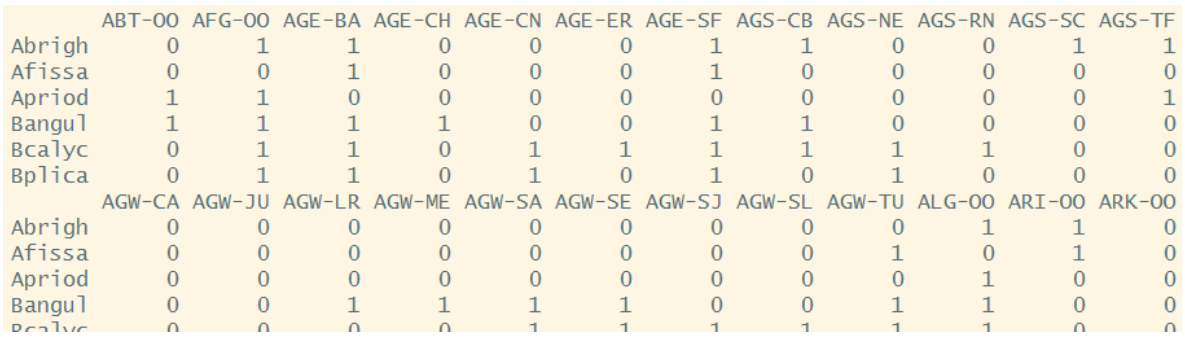
```
Mantel statistic based on Spearman's rank correlation rho

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

Mantel statistic r: 0.6778
      Significance: 0.001

Upper quantiles of permutations (null model):
  90%   95% 97.5%   99% 
0.184 0.241 0.277 0.304 
Permutation: free
Number of permutations: 999
```

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 from the complete tables, **specifying which columns contain the species occurrence data** to transpose in each table, and **which column contains the region names** to use as column names in the transposed table:

```
names(rotif.env)

bin.reg <- transpose(rotif.env, sp.cols = 18:47, reg.names = 1)

names(rotifers.fav$predictions)

fuz.reg <- transpose(rotifers.fav$predictions, sp.cols = 32:61, reg.names
= 1)
```

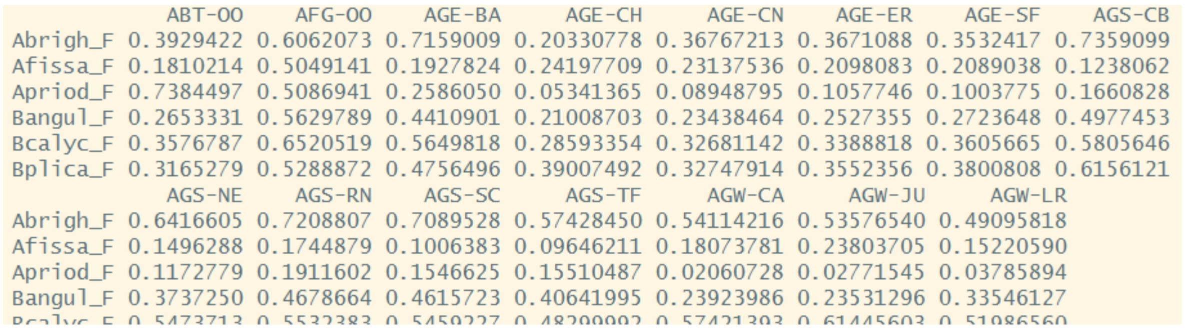
Look at the first rows of the resulting tables:

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



```
       ABT-OO AFG-OO AGE-BA AGE-CH AGE-CN AGE-ER AGE-SF AGS-CB AGS-NE AGS-RN AGS-SC AGS-TF
Abrigh      0      1      1      0      0      0      1      1      0      0      1      1
Afissa      0      0      1      0      0      0      1      0      0      0      0      0
Apriod      1      1      0      0      0      0      0      0      0      0      0      1
Bangul      1      1      1      1      0      0      1      1      0      0      0      0
Bcalyc      0      1      1      0      1      1      1      1      1      1      0      0
Bplica      0      1      1      0      1      0      1      0      1      0      0      0
       AGW-CA AGW-JU AGW-LR AGW-ME AGW-SA AGW-SE AGW-SJ AGW-SL AGW-TU ALG-OO ARI-OO ARK-OO
Abrigh      0      0      0      0      0      0      0      0      0      1      1      0
Afissa      0      0      0      0      0      0      0      0      1      0      1      0
Apriod      0      0      0      0      0      0      0      0      0      1      0      0
Bangul      0      0      1      1      1      1      0      0      1      1      0      0
Bcalyc      0      0      0      0      1      1      1      1      1      1      0      0
```

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



```
              ABT-OO    AFG-OO    AGE-BA     AGE-CH     AGE-CN    AGE-ER    AGE-SF    AGS-CB
Abrigh_F 0.3929422 0.6062073 0.7159009 0.20330778 0.36767213 0.3671088 0.3532417 0.7359099
Afissa_F 0.1810214 0.5049141 0.1927824 0.24197709 0.23137536 0.2098083 0.2089038 0.1238062
Apriod_F 0.7384497 0.5086941 0.2586050 0.05341365 0.08948795 0.1057746 0.1003775 0.1660828
Bangul_F 0.2653331 0.5629789 0.4410901 0.21008703 0.23438464 0.2527355 0.2723648 0.4977453
Bcalyc_F 0.3576787 0.6520519 0.5649818 0.28593354 0.32681142 0.3388818 0.3605665 0.5805646
Bplica_F 0.3165279 0.5288872 0.4756496 0.39007492 0.32747914 0.3552356 0.3800808 0.6156121
              AGS-NE    AGS-RN    AGS-SC     AGS-TF     AGW-CA     AGW-JU     AGW-LR
Abrigh_F 0.6416605 0.7208807 0.7089528 0.57428450 0.54114216 0.53576540 0.49095818
Afissa_F 0.1496288 0.1744879 0.1006383 0.09646211 0.18073781 0.23803705 0.15220590
Apriod_F 0.1172779 0.1911602 0.1546625 0.15510487 0.02060728 0.02771545 0.03785894
Bangul_F 0.3737250 0.4678664 0.4615723 0.40641995 0.23923986 0.23531296 0.33546127
Bcalyc_F 0.5473713 0.5532383 0.5459227 0.48299992 0.57421393 0.61445603 0.51986560
```

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 = "Jaccard")

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

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 text file named *mydata.txt* and 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 *rotif.env* (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, but first remember to check for updates to the package or this tutorial at http://fuzzysim.r-forge.r-project.org.