A beginners tutorial on the *fuzzySim* R package

A. Marcia Barbosa, CIBIO/InBIO - University of Evora (Portugal),
barbosa@uevora.pt

updated 30 Apr 2015

fuzzySim is an R package for calculating **fuzzy similarity in species distributions**. It can also **produce** the underlying **fuzzy distribution data**, e.g. through **interpolating or modelling species presence-absence data**.

Installing and loading *fuzzySim*

fuzzySim 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 monospaced 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). For commands that generate visible results in R, these are usually shown below them, preceded by hash marks (##). 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 end of 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.

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* for Windows or *.tar.gz* for Linux and Mac, available at the package development page or at this Dropbox folder and then install the package locally, 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)
```

Preparing the species occurrence data

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 (_, as you've seen in the table 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 *spcodes* as column names:

```
rotifers.presabs <- splist2presabs(rotifers, sites.col = "TDWG4", sp.col = "spcode",
keep.n = FALSE)
```

Show the first rows of the result:

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

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

```
## 6    1    0    0    1    1    1    1    0    0    1
##   Ttetra
## 1     1
## 2     0
## 3     1
## 4     0
## 5     1
## 6     0
```

Mapping the species occurrence data

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 wish, you can delete the shapefile folder that was saved in your working directory:

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

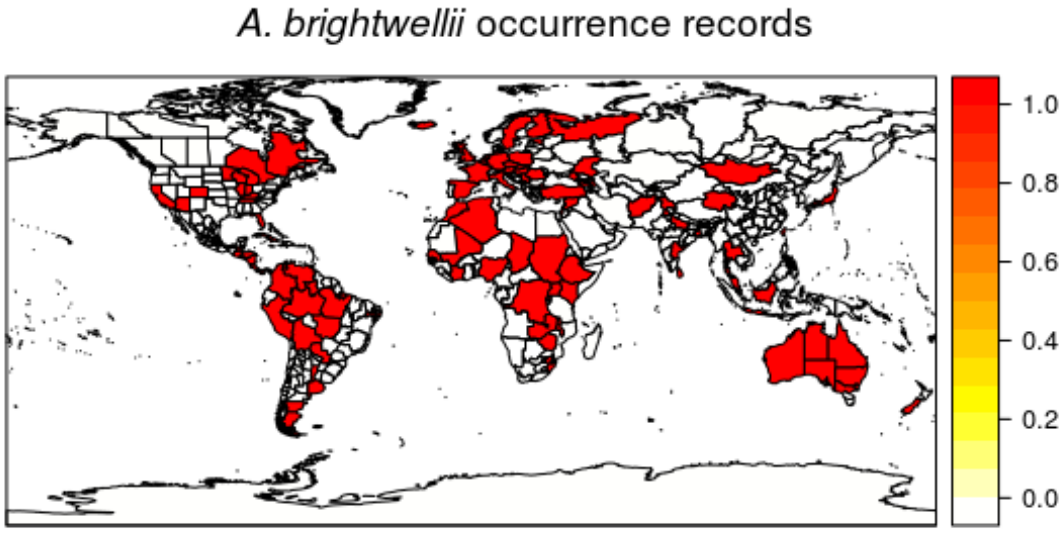
Now add the *rotifers.presabs* data to the map's attribute table within R, matching them by the name of the column containing the region identifiers in each table:

```
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)), main = expression(paste(italic("A. brightwellii"), " occurrence records"))))
```



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.

Converting binary to fuzzy occurrence data

You can get a fuzzy (continuous between 0 and 1) version of presence-absence data using e.g. trend surface analysis or inverse distance interpolation. 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 TDWG4 unit:

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

Take a look at its first rows (results not shown here):

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

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"
##  [3] "REGION_NAME"              "CONTINENT"
##  [5] "Area"                 "Altitude"
```

```
##  [7] "AltitudeRange"           "HabitatDiversity"
##  [9] "HumanPopulation"         "Latitude"
## [11] "Longitude"               "Precipitation"
## [13] "PrecipitationSeasonality" "TemperatureAnnualRange"
## [15] "Temperature"             "TemperatureSeasonality"
## [17] "UrbanArea"               "Abrigh"
## [19] "Afissa"                  "Apriod"
## [21] "Bangul"                  "Bcalyc"
## [23] "Bplica"                  "Bquadr"
## [25] "Burceo"                  "Cgibba"
## [27] "Edilat"                  "Flongi"
## [29] "Kcochl"                  "Kquadr"
## [31] "Ktropi"                  "Lbulla"
## [33] "Lclost"                  "Lhamat"
## [35] "Lluna"                   "Llunar"
## [37] "Lovali"                  "Lpatel"
## [39] "Lquadr"                  "Mventr"
## [41] "Ppatul"                  "Pquadr"
## [43] "Pvulga"                  "Specti"
## [45] "Tpatin"                  "Tsimil"
## [47] "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. Mind 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 the different options of the *multTSA* function by reading the help file that appears when you type help(multTSA). Now look at the first rows of the *rotifers.tsa* created just above:

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

Results (not shown here) 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])
```

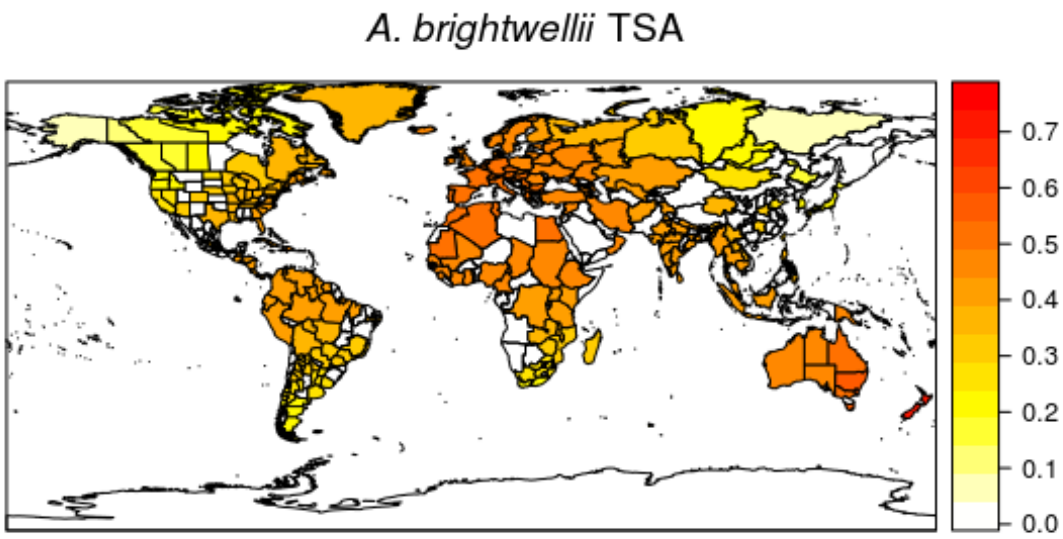
```
## [1] 9.408071e-05 9.996004e-01
```

Now **add the TSA results to the *TDWG4shp* map table**:

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

Plot the results for the first species (and then others as you like; check names(TDWG4shp) for available *zcol* options):

```
print(spplot(TDWG4shp, zcol = "Abrigh_TS", col.regions = rev(heat.colors(256)), main =
expression(paste(italic("A. brightwellii"), " TSA"))))
```



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 (results not shown here):

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

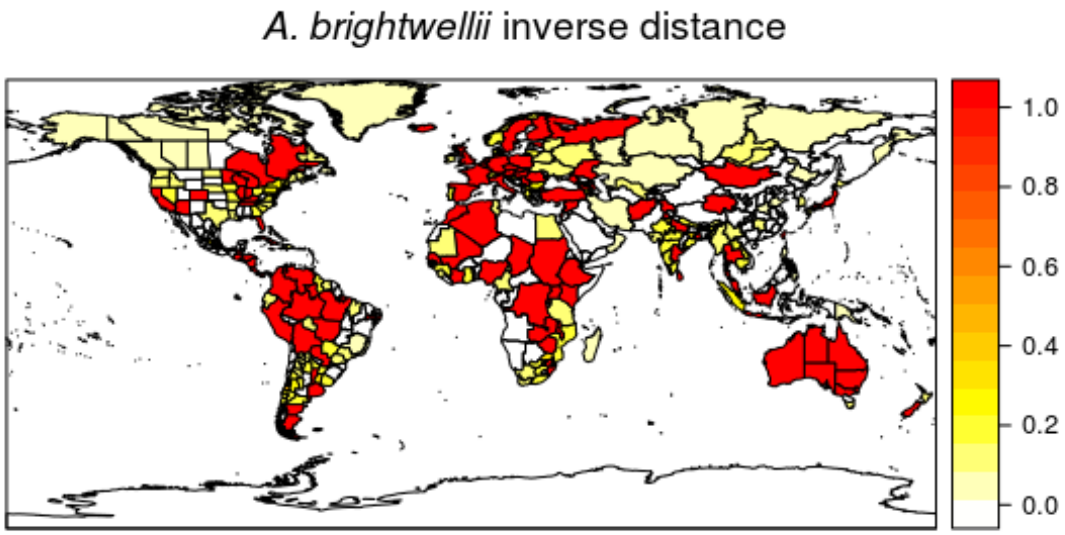
You can check that the values in this table (excluding the first column) also range between 0 and 1, so they can be used with fuzzy logic:

```
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)), main =
expression(paste(italic("A. brightwellii"), " inverse distance"))))
```



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 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 dataset and the index numbers of the columns containing the species data and the variables. There are a number of options on how to select variables for the models -- you can type help(multGLM) or read this modelling tutorial for further details. Here, we'll just see how to create models with the *multGLM* default options:

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

The object output by *multGLM* is a list containing two elements: another list named *models*, and a dataframe with the resulting *predictions*. Check out the first rows of this dataframe (results not shown here):

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

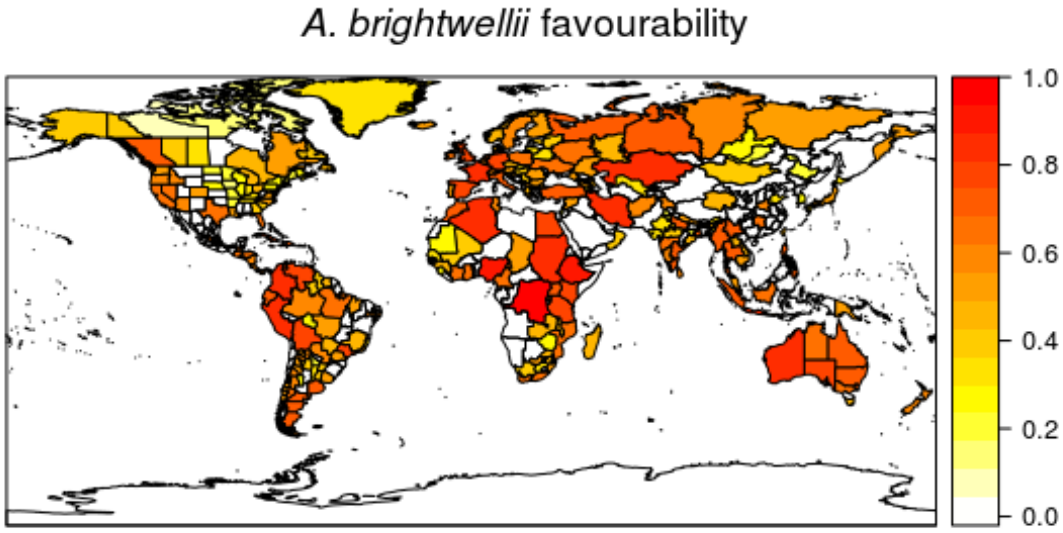
Predicted presence probability is in the columns with suffix *P*, but here we're more interested in the **columns with suffix *F***, containing **environmental favourability** values, which are **directly comparable among species** (whereas probability is affected by species prevalence; see Details in help(Fav) for more info).

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

```
##  [1] "TDWG4"    "Abrigh_P" "Afissa_P" "Apriod_P" "Bangul_P" "Bcalyc_P"
##  [7] "Bplica_P" "Bquadr_P" "Burceo_P" "Cgibba_P" "Edilat_P" "Flongi_P"
## [13] "Kcochl_P" "Kquadr_P" "Ktropi_P" "Lbulla_P" "Lclost_P" "Lhamat_P"
## [19] "Lluna_P"  "Llunar_P" "Lovali_P" "Lpatel_P" "Lquadr_P" "Mventr_P"
## [25] "Ppatul_P" "Pquadr_P" "Pvulga_P" "Specti_P" "Tpatin_P" "Tsimil_P"
## [31] "Ttetra_P" "Abrigh_F" "Afissa_F" "Apriod_F" "Bangul_F" "Bcalyc_F"
## [37] "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"
## [49] "Lluna_F"  "Llunar_F" "Lovali_F" "Lpatel_F" "Lquadr_F" "Mventr_F"
## [55] "Ppatul_F" "Pquadr_F" "Pvulga_F" "Specti_F" "Tpatin_F" "Tsimil_F"
## [61] "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. favourability for 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)), main =
expression(paste(italic("A. brightwellii"), " favourability"))))
```



Calculating (fuzzy) similarity among species occurrence patterns

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:

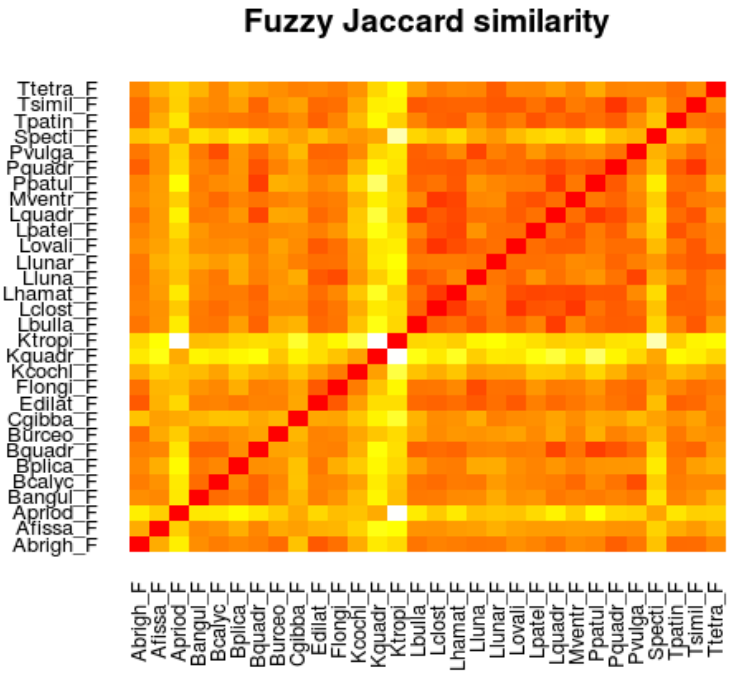
```
fuz.sim.mat <- simMat(rotifers.fav$predictions[ , 32:61], method = "Jaccard")
```

Take a look at the first rows of the resulting fuzzy similarity matrix (results not shown here):

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

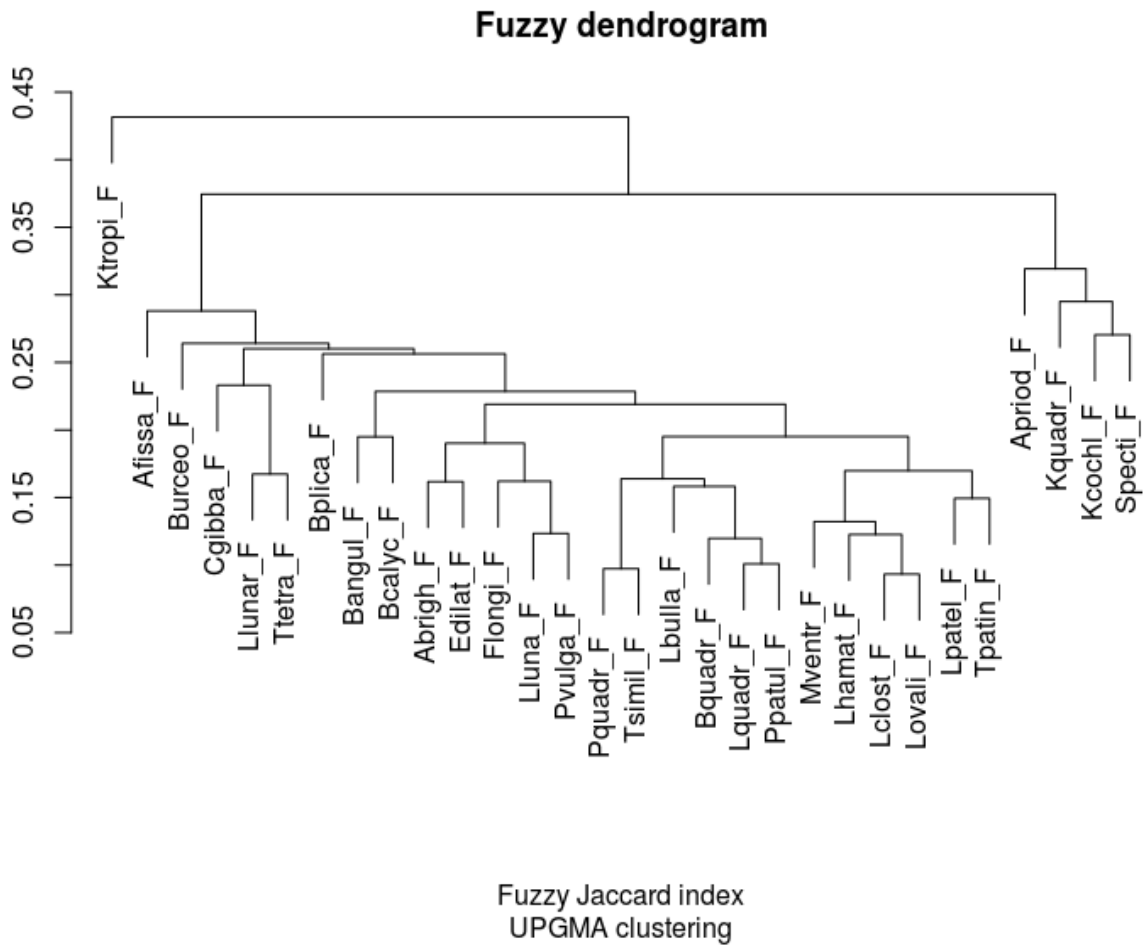
For a quick look at which species pairs are more and less similar in distribution, you can **plot the similarity matrix with a colour scale** (the same used above for the maps), using the *image* function of R and then adding the species as axis labels (the *cex.axis* argument defines the text size):

```
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, main = "Fuzzy Jaccard
similarity")
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 **build a cluster dendrogram from the similarity matrix**, using the *hclust* R function. The clustering method requires a distance matrix, so our similarity matrix is subtracted from 1 in the command below. The *method*="*average*" argument means that UPGMA is the clustering algorithm, but you can check for other options with help(hclust):

```
fuz.dendro <- hclust(as.dist(1 - fuz.sim.mat), method = "average")
plot(fuz.dendro, main = "Fuzzy dendrogram", xlab = "", ylab = "", sub = "Fuzzy Jaccard index\nUPGMA clustering")
```



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"
##  [3] "REGION_NAME"           "CONTINENT"
##  [5] "Area"                  "Altitude"
```

```
##  [7] "AltitudeRange"           "HabitatDiversity"
##  [9] "HumanPopulation"         "Latitude"
## [11] "Longitude"               "Precipitation"
## [13] "PrecipitationSeasonality" "TemperatureAnnualRange"
## [15] "Temperature"             "TemperatureSeasonality"
## [17] "UrbanArea"               "Abrigh"
## [19] "Afissa"                  "Apriod"
## [21] "Bangul"                  "Bcalyc"
## [23] "Bplica"                  "Bquadr"
## [25] "Burceo"                  "Cgibba"
## [27] "Edilat"                  "Flongi"
## [29] "Kcochl"                  "Kquadr"
## [31] "Ktropi"                  "Lbulla"
## [33] "Lclost"                  "Lhamat"
## [35] "Lluna"                   "Llunar"
## [37] "Lovali"                  "Lpatel"
## [39] "Lquadr"                  "Mventr"
## [41] "Ppatul"                  "Pquadr"
## [43] "Pvulga"                  "Specti"
## [45] "Tpatin"                  "Tsimil"
## [47] "Ttetra"
```

Now calculate the binary similarity matrix from the columns containing the species occurrence data:

```
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* package. If you want to do this, the command below will install vegan if you don't already have it within your R installation:

```
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%
## 0.182 0.220 0.268 0.324
##
## Based on 999 permutations
```

Calculating (fuzzy) similarity among regional species pools

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 data to transpose, 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 (results not shown here):

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

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 now:

```
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.

Further steps

Fuzzy similarity matrices can 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/biogeographic regions** (significant clusters of regional species compositions).

Using your own data

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 where it is), and then import it to R using the following command:

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

The sep = "\t" argument above indicates that the columns in the text file are separated by tabulators, but you can specify other separators as necessary, such as spaces (sep = " "), commas (sep = ",") or semicolons (sep = ";").

Then reproduce all the operations above, but replacing *rotifers* or *rotif.env* (depending on if your data are in long or wide format, respectively) with *mydata* (or whatever name you've assigned in the command above) and specifying column names or numbers accordingly. If you use *fuzzySim* in publications, please **cite the paper**:

Barbosa A.M. (2015) fuzzySim: applying fuzzy logic to binary similarity indices in ecology. *Methods in Ecology and Evolution*, in press (DOI: 10.1111/2041-210X.12372).

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. This tutorial was built with *RStudio* + *rmarkdown* + *knitr*. Thanks!