Characterising communities and their ecotones with the EcotoneFinder package

Antoine Bagnaro

2023-02-16

Abstract

This document covers and explains the use of the different functions regrouped in the `EcotoneFinder` package. Attention is given to the interdependence between analyses, and their presentation follows a sequence deemed appropriate for studies on ecotones. The theoretical framework and set of assumptions on which this package is built are thus detailed, and the outputs of the analyses – Detrended Correspondence Analyses, Fuzzy clusters, and Networks – are discussed. This document should prove a comprehensive and helpful guide for any who would like to get started with the `EcotoneFinder` package.

Contents

Introduction

The aim of this vignette is mainly to serve as a tutorial for the use of the `EcotoneFinder` package. This package implements methods for the localisation and characterizations of both ecotones and ecological communities along gradients – environmental or biotic – in ecosystem-wide representations.The initial motivation to put together the set of analyses presented below was an attempt at addressing one of the shortcomings presented by [29] in their review on ecotone research, namely that there was no consistent framework for the exploration of edge characteristics at the ecosystem/community level. Alongside with proposing such a framework, we tried to make the outputs of the `EcotoneFinder` package as compatible as possible with the single-species framework presented earlier in [28].
The set of functions that constitute this package provide a workflow from an initial data set (typically containing field observations) to the extraction of relevant ecotone and community information, and publishable graphical outputs. The basic organisation of the `EcotoneFinder` package is given in Figure 1. The rest of this tutorial will be structured function-by-function, as highlighted by the roman numbers in parenthesis in Figure 1.

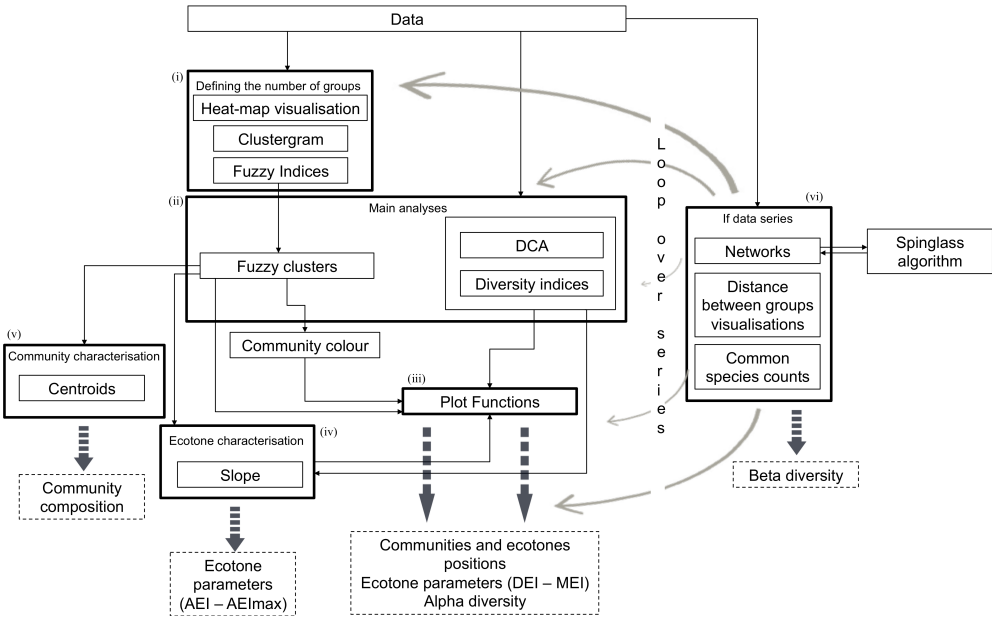

Figure 1: General organisation chart of the R-package, from raw data to typical outputs. It may be subdivided in several main sections: (i) functions to explore the internal structure of the raw data, (ii) main data analyses for ecotones and communities characterisation, (iii) plotting functions, (iv) functions to extract ecotone parameters, (v) functions to access community compositions and (vi) extensions of these functions for data series.

First, we start by loading the package:

```
library(EcotoneFinder)
```

Artificial data generation:

The use of artificial data – with known patterns – is a powerful evaluation method for statistical approaches [1], as it is not hindered by the unknown nature of the actual relationship between variables, and by the high level of variability of field data [1, 34]. As a commodity, the EcotoneFinder package implements two functions to generate artificial datasets (`SyntheticData` and `SyntheticDataSeries`). The use of artificial data will also provide practical examples for the presentation of this R-package.

The `SyntheticData` function generates gaussian-shaped species abundance curves, as equivalent to species response curves [41], along the x-axis (which corresponds to the "distance" gradient, or transect).

Species are grouped in community types according to the coincidence of their response curves along the x-axis, and a number of arguments allow for the control of the number of communities to be generated, the number of species per community, and the distinctiveness of these communities along the axis (Figure 2).

```
# 2 Communities - 26 species:
Community2 <- SyntheticData(SpeciesNum=26, CommunityNum=2, SpCo=NULL, Length = 500,
                            Parameters=list(a=c(60, 60), b=c(0, 500), c=c(0.009, 0.009)),
                            dev.c = .0024, dev.a = 25, dev.b = 30,
                            pal=c("#008585", "#C7522B"),
                            title = "2 communities")

# 3 Communities - 39 species:
Community3 <- SyntheticData(SpeciesNum=39, CommunityNum=3, SpCo=NULL, Length = 500,
                            Parameters=list(a=c(60, 60, 60), b=c(0, 250, 500),
                                            c=c(0.015, 0.015, 0.015)),
                            dev.c = .0024, dev.a = 35, dev.b = 25,
                            pal=c("#008585", "#B8CDAE", "#C7522B"),
                            title = "3 communities")
```

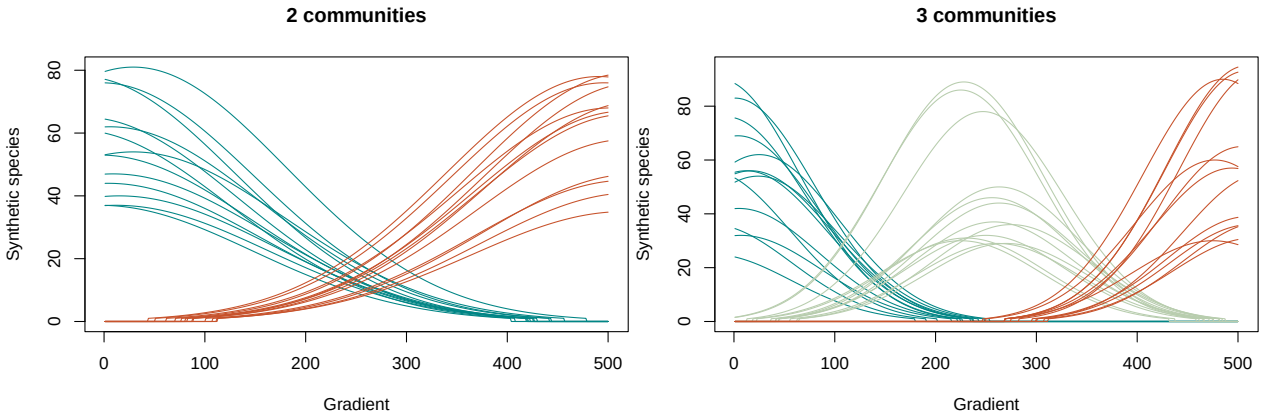

Figure 2: Examples of artificial data with 2 and 3 communities

`SpeciesNum` controls the number of curves, and `CommunityNum` the number of communities. Note that the community matrices are stored in the `Community2` and `Community3` objects (dataframe format). Species curves belonging to the same communities share the same sets of parameters for their gaussian response curves (the `Parameters` argument). The internal gaussian formula is of the form:

$$ae^{-\frac{(x-b)^2}{2c^2}}$$

This simple formula allows for an easy manipulation of the shapes of the curves. The three parameters, `a`, `b` and `c`, respectively control the height of the curve, the position of its centre, and the steepness of the slopes on both sides (as shown in Figure 3).

```
# 2 Communities - a = 80 and a = 40:
Community2 <- SyntheticData(SpeciesNum=26, CommunityNum=2, SpCo=NULL, Length = 500,
                            Parameters=list(a=c(80, 40), b=c(0, 500), c=c(0.009, 0.009)),
                            dev.c=.0024, dev.a = 25, dev.b = 30,
                            pal=c("#008585", "#C7522B"),
                            title = "a = 80 and a = 40")

# 2 Communities - b = 150 and b = 350:
```

```
Community2 <- SyntheticData(SpeciesNum=26, CommunityNum=2, SpCo=NULL, Length = 500,
                            Parameters=list(a=c(60, 60), b=c(100, 400),
                                            c=c(0.009, 0.009)),
                            dev.c=.0024, dev.a = 25, dev.b = 30,
                            pal=c("#008585", "#C7522B"),
                            title = "b = 150 and b = 350")

# 2 Communities - c = 0.005 and c = 0.012:
Community2 <- SyntheticData(SpeciesNum=26, CommunityNum=2, SpCo=NULL, Length = 500,
                            Parameters=list(a=c(60, 60), b=c(0, 500), c=c(0.005, 0.012)),
                            dev.c=.0024, dev.a = 25, dev.b = 30,
                            pal=c("#008585", "#C7522B"),
                            title = "c = 0.005 and c = 0.012")
```

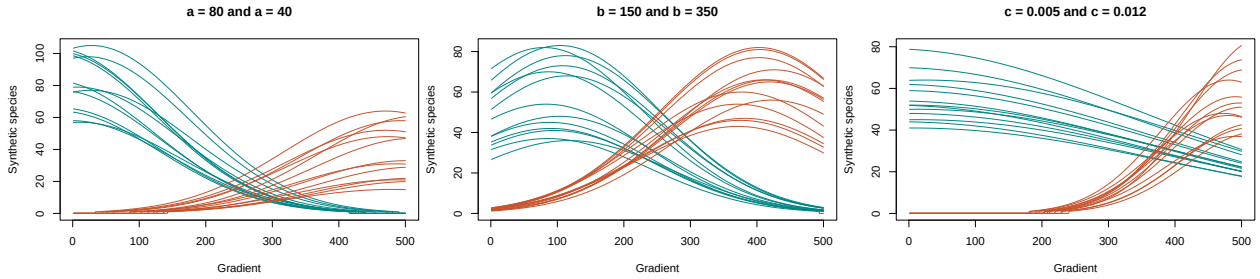

Figure 3: Examples of artificial data with different gaussian parameters

So far, the `SpCo` argument has been left `NULL`. By default, the `SyntheticData` function equally partition the species in the different communities (in our example it means that – out of 26 species – 13 are put in the first community, and 13 in the second). The `SpCo` argument (standing for Species per Community) allows for user-defined species partitioning.
Another set of arguments, `dev.a`, `dev.b`, and `dev.c`, controls for a certain amount of deviation around the given parameters values, thus producing more or less identical species curves for each community. The gaussian parameters for each curves are then chosen randomly (via the `sample()` function) in the interval $[Parameter - dev.value; Parameter + dev.value]$. The greater these parameters, the less homogeneous the artificial communities will be.
Playing with these different arguments allows for the generation of diverse sets of artificial data (Figure 4).

```
# 2 Communities with unequal species numbers:
Community2 <- SyntheticData(SpeciesNum=30, CommunityNum=2, SpCo=c(10,20), Length = 500,
                            Parameters=list(a=c(60, 60), b=c(0, 500), c=c(0.009, 0.009)),
                            dev.c=.0024, dev.a = 25, dev.b = 30,
                            pal=c("#008585", "#C7522B"),
                            title = "Unequal species numbers")

# Adding parameter deviations:
Community2 <- SyntheticData(SpeciesNum=30, CommunityNum=2, SpCo=NULL, Length = 500,
                            Parameters=list(a=c(60, 60), b=c(0, 500), c=c(0.009, 0.009)),
                            dev.c=.005, dev.a = 50, dev.b = 60,
                            pal=c("#008585", "#C7522B"),
                            title = "Parameter deviation")

# Adding complexity:
Community2 <- SyntheticData(SpeciesNum=60, CommunityNum=4, SpCo=c(20,10,10,20),
                            Length = 500,
```

```
                                Parameters=list(a=c(60, 30, 40, 70), b=c(0,0,500,500),
                                                c=c(0.015, 0.008, 0.007, 0.01)),
                                dev.c=.0024, dev.a = 25, dev.b = 30,
                                pal=c("#008585", "#AAC1A8", "#D58545", "#C7522B"),
                                title = "Pools of species inside community types")
```

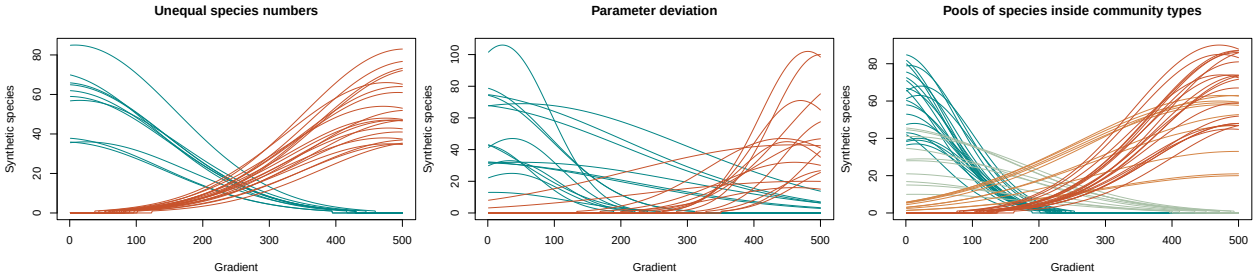

Figure 4: Control of the number of species per communities, of the gaussian parameter deviations and of particular groups of species in community types.

Data series

Lastly, the `SyntheticDataSeries` function simplifies the generation of artificial data with common characteristics, along a series (be it a time series, or replicates over the spatial extent of the sampling). This function essentially loops over the `SyntheticData` function, while controlling (to a certain extent) the kind of changes that affect the data over the series. Particularly, the number of species per artificial community can vary inside a specified interval (for instance, to account for the loss or gain of seasonal species over a time series, see Figure 5), and the position of the successive communities can be set to move along the gradient (to mimic, for instance, the gradual displacement of an hypothetical environmental gradient). This is done through the `displacement` parameter, and a numeric matrix specifying the desired changes (direction and values) along the x-axis for each community (Figure 6).

```
### Series:
Series <- SyntheticDataSeries(CommunityPool = 30, CommunityNum = 3, Length = 500,
                              SeriesNum = 6, replacement = TRUE, range.repl = 20,
                              #SpCo = c(30,30,30),
                              Parameters = list(a = c(60,60,60),
                                                b = c(100,250,400),
                                                c = c(0.015, 0.02, 0.02)),
                              dev.a=30, dev.b=40, dev.c=0.005,
                              pal = c("#008585", "#E6C186", "#C7522B"))
```

```
## Displacement matrix:
disp <- matrix(data=c(0,0,0,
                      0,35,-0.0007,
                      0,10,0), nrow = 3, ncol = 3)

### Series:
Series <- SyntheticDataSeries(CommunityPool = 60, CommunityNum = 3, Length = 500,
                              SeriesNum = 6, replacement = FALSE, SpCo = c(15,15,30),
                              Parameters = list(a = c(60,60,60),
                                                b = c(-50,-50,400),
                                                c = c(0.01, 0.01, 0.01)),
                              dev.a=30, dev.b=40, dev.c=0,
```

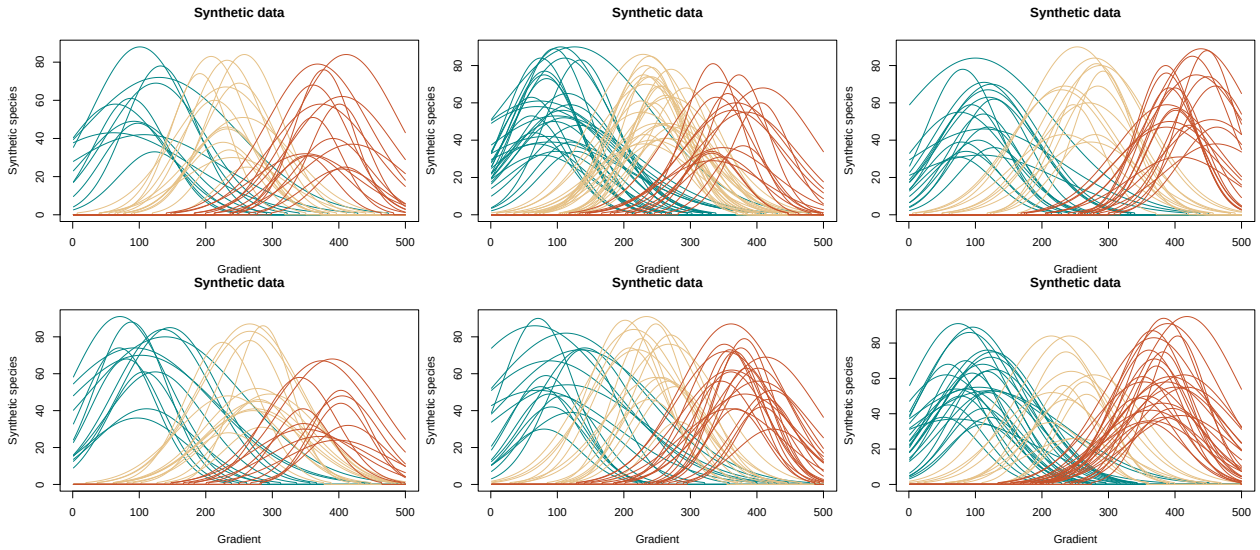

Figure 5: Time series with varying number of species par community types and per series.

```
                displacement = disp,
                pal = c(rep("#008585",15), rep("#E6C186",15),
                        rep("#C7522B",30)))
```

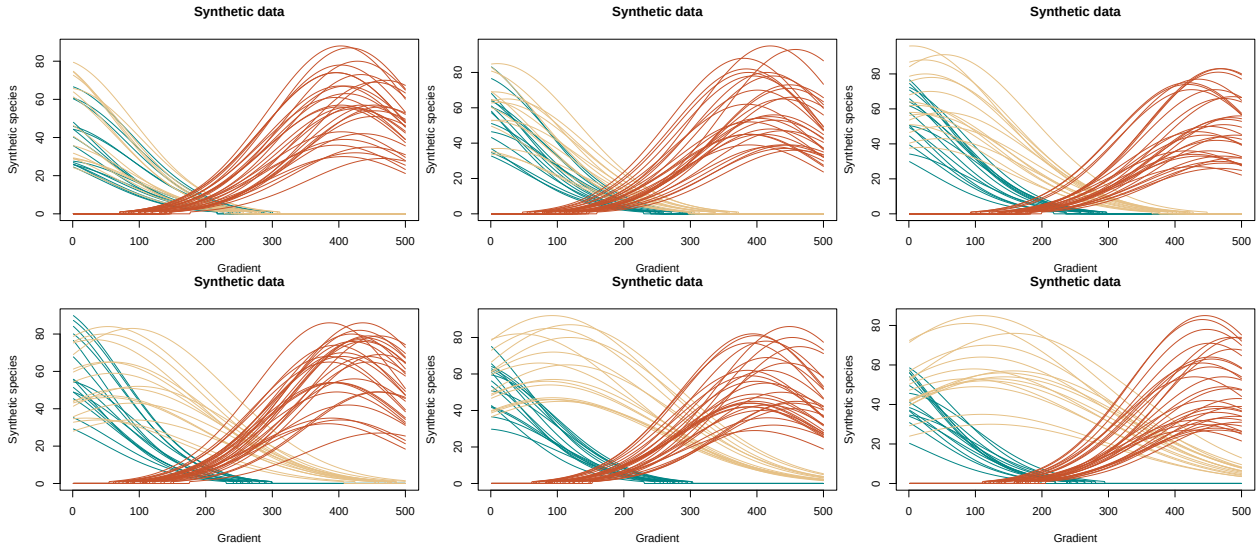

Figure 6: Time series with displacement: part of the first community (left) advances toward the right-end of the gradient by 35 gradient units per series, as if reacting to a moving environmental variable. The last community also withdraw even further to the right, but slower (10 gradient units per series).

Now that we have generated some artificial community data, we can proceed to their analyses.

Internal structure of the data:

One of the recurrent (and vexing) problem encountered by community ecologists is the determination of the "correct" number of groups in their data, i.e. the actual number of communities that have been sampled. In our case, this means being able to correctly recover the number of communities that have been generated when producing the artificial data (and we know this number: we chose it!). With field

observations, however, this can be assessed *à-priori* when delineations between community types are visually obvious (e.g. between forests and agricultural lands), or when the number of groups are decided based on standard protocols (e.g. to fit data into pre-existing categories). Many situations, however, are harder to resolve, either due to the size of the organisms (making visual assessments impossible), the existence of closely related groups (making the classification of near-boundary objects unsatisfactory), or the existence of important mismatches between pre-existing classifications and the reality (driving the necessity for re-evaluation).

A number of approaches have been proposed over the years to address these issues, with contrasted results when confronted with the high internal variability of ecological field data. Recent years have seen promising methodological advances, notably with the use of machine learning, and new and more effective approaches will undoubtedly become available in the near future. The `EcotoneFinder` package – through the `DistEco` function – only implements basic methods to explore the internal structure of data (based on distance matrix computation and network analyses, controlled by the `plot` argument), to help the decision process. We will review them in this section, but it is advised to reach out for more recent approaches if problems persist.

Distance matrices can be easily visualized with heat-maps. In such representations, groups of more closely related data-points appear along the diagonal of the heat-map, forming similarly colored squares (Fig. 7). The main branches of the hierarchical trees on the sides of the heat-maps also segregate the data in similar ways.They are often easier to interpret than the heat-maps themselves when the internal variability of the data is high. In these situations, the computation of networks from the raw community data may help considerably – particularly as robust statistical tools exist to test for the existence of statistical network communities (i.e. groups of nodes that are consistently related, independently of network topology). In our case, these network communities may correspond to actual ecological communities (Fig. 7).

To test for the existence of statistical communities in the networks, the `DistEco` function makes use of the spinglass algorithm implemented in the `igraph` package (if the `spinglass` argument is set to `TRUE`). The test itself is run a given number of times (chosen by the `run` argument, here set to 5 only, to reduce computation time.), and the average community score over the total number of runs is returned. The results of the test can then either be given as rounded numbers (i.e. 1, 2, 3, etc.), or "raw" (i.e. with decimals), which highlights with more clarity the nodes that tends to swap from one community to another depending on the spinglass test run. This can give more accuracy to the results – and already point out to "ecotonal" samples – but may complicate the interpretation of the outputs. More details about this process is given towards the end of this document (see Statistical groups in networks).

Lastly, the `transpose` argument controls whether to compare sites (usually as rows) or species (usually as columns) in the provided community matrix.

For finer control of the graphical outputs, the function also accept additional arguments from the `stats::heatmap` function – if `plot` is set to "heatmap" – and the `qgraph::qgraph` function – if `plot` is set to "network".

```
# 2 Communities - 26 species:
Community2 <- SyntheticData(SpeciesNum=26, CommunityNum=2, SpCo=NULL, Length = 500,
                            Parameters=list(a=c(60, 60), b=c(0, 500), c=c(0.009, 0.009)),
                            dev.c=.0024, dev.a = 25, dev.b = 30,
                            pal=c("#008585", "#C7522B"),
                            title = "2 communities")

# 3 Communities - 39 species:
Community3 <- SyntheticData(SpeciesNum=39, CommunityNum=3, SpCo=NULL, Length = 500,
                            Parameters=list(a=c(60, 60, 60), b=c(0, 250, 500),
                                            c=c(0.015, 0.015, 0.015)),
                            dev.c=.0024, dev.a = 35, dev.b = 25,
                            pal=c("#008585", "#B8CDAE", "#C7522B"),
                            title = "3 communities")
```

```
# Heat maps:
HeatMap2 <- DistEco(Community2[,2:ncol(Community2)], transpose = T, symm = F,
                    plot = c("heatmap"))
HeatMap3 <- DistEco(Community3[,2:ncol(Community3)], transpose = T, symm = F,
                    plot = c("heatmap"))

# Networks (with spinglass algorithm for statistical communities):
Network2 <- DistEco(Community2[,2:ncol(Community2)], transpose = T, symm = F,
                    plot = c("network"), spinglass = T, run = 5,
                    spinglass.groups = c("rounded"))
Network3 <- DistEco(Community3[,2:ncol(Community3)], transpose = T, symm = F,
                    plot = c("network"), spinglass = T, run = 5,
                    spinglass.groups = c("rounded"))
```

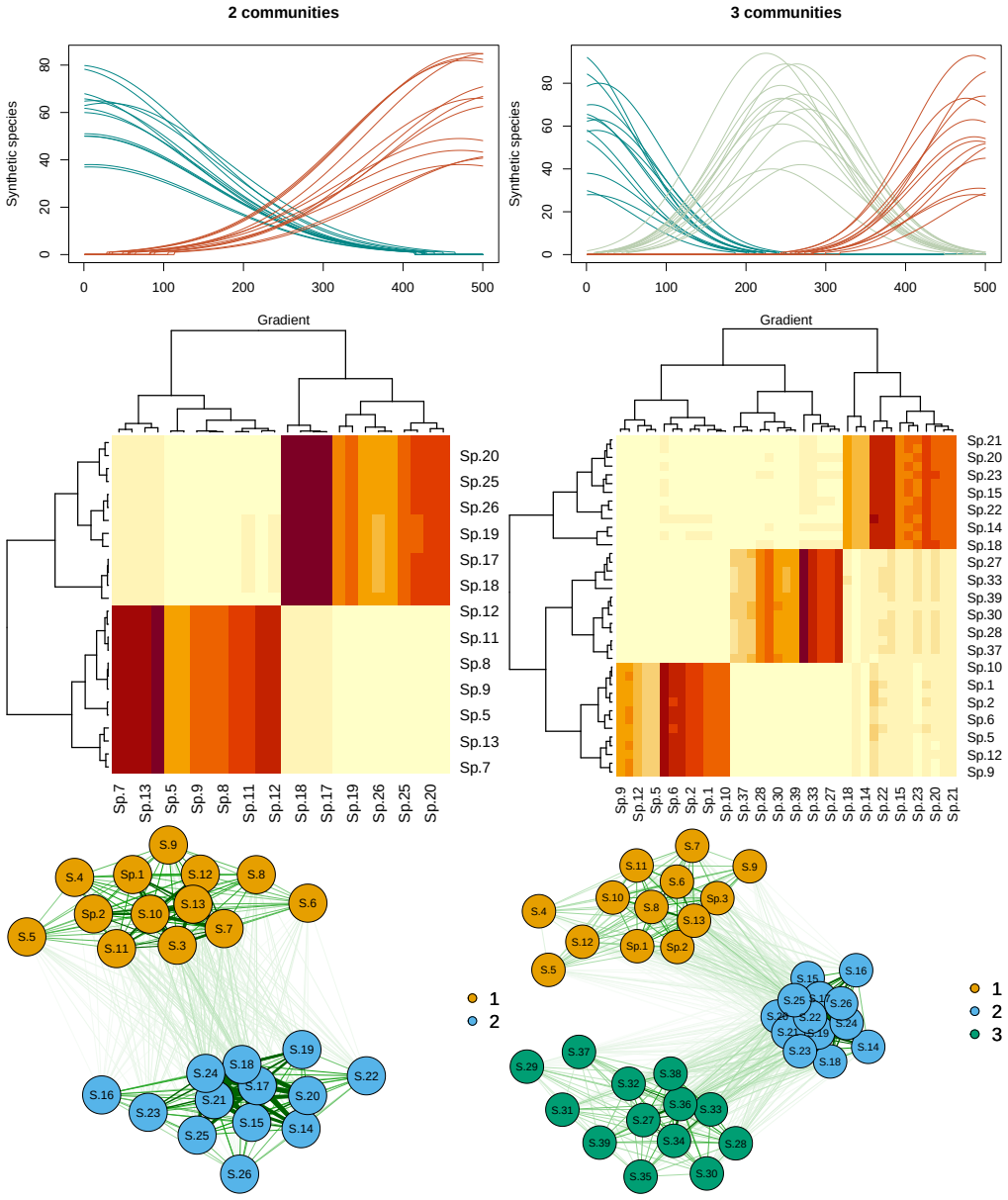

Figure 7: Heatmaps and Networks, based on the example artificial communities presented above in Fig. 2

[33] proposed the use of particular plots – clustergrams – to explore the internal structure of data. The analyses was initially intended for hierarchical and k-means clustering. Their use have been extended to additionally accept fuzzy cluster algorithms, to better fit the scope of this package (see Fuzzy clustering. The reader is referred to the original publication of [33] for comprehensive details on the meaning of these plots. Only a few examples will be reproduced here (Fig. 8). In simple terms, though, they may be read as a graphical representation of the evolution of the distance between the k cluster centroids computed by the fuzzy cluster algorithm (red dots), over increasing number of clusters (controlled by the `k.range` argument).
For completeness, the `clustergramInd` function in the `EcotoneFinder` package may also return the evolution of the fuzzy indices implemented in the `vegclust` package, and initially proposed by [4]. Both these indices (the Normalized partition coefficient (PCN) – that should be maximized – and the Normalized partition entropy – that should be minimized) can be used as criteria to choose the number of clusters. To be noted, low number of clusters (e.g. when $k = 2$) are often more stable solutions than higher number of clusters. When the number of actual groups in the data is high, it may thus appear as local minima/maxima rather than overall minima/maxima.

```
######## clustergram plots (on fuzzy c-means algorithm):
## with the previous '3 communities' example:
clustergramInd(as.matrix(Community3[,2:ncol(Community3)]),
                          clustering.function = clustergram.vegclust.Ind,
                          clustergram.plot = clustergram.plot.matlines,
                          FuzzyIndice.plot = FuzzyIndice.plot.matlines,
                          k.range = 2:10, line.width = .2)

######## clustergram plots (on fuzzy c-means algorithm):
## with 6 artificial communities:
Community6 <- SyntheticData(SpeciesNum=60, CommunityNum=6, SpCo=NULL, Length = 500,
                            Parameters=list(a=rep(60, times = 6), b=seq(0,500, by = 100),
                                            c=rep(0.03, times = 6)),
                            dev.c=.0024, dev.a = 35, dev.b = 25,
                            pal=rep(c("#008585", "#B8CDAE", "#C7522B"), times =2),
                            title = "6 communities",
                            plot = FALSE)

clustergramInd(as.matrix(Community6[,2:ncol(Community6)]),
                          clustering.function = clustergram.vegclust.Ind,
                          clustergram.plot = clustergram.plot.matlines,
                          FuzzyIndice.plot = FuzzyIndice.plot.matlines,
                          k.range = 2:10, line.width = .1)
```

Once the number of communities in the data has been chosen (or – more realistically – once the number of groups in the data has been reduced to a few good candidates), we can move on to the more accurate description of these communities, and of their interceding ecotones.

Community detection along gradients:

The `EcotoneFinder` function regroups a number of approaches designed for the detection and characterization of ecological communities and ecotones along spatial or environmental gradients. So far, the implemented methods are Detrended Correspondence Analyses (DCA), and different fuzzy cluster algorithms. The visualization of the outputs can then be handled by the `PlotEcotone` function (for base-R plot grammar) or the `ggEcotone` function (for ggplot grammar).
Both of these functions are designed to explore the variability of the DCA or clustering results over the spatial or temporal dimensions of the sampling – in other words, they put the *distance* or *time* on the x-axis. A separate section will be dedicated to the use of the `ggEcotone` function (see Visualisation with ggolot

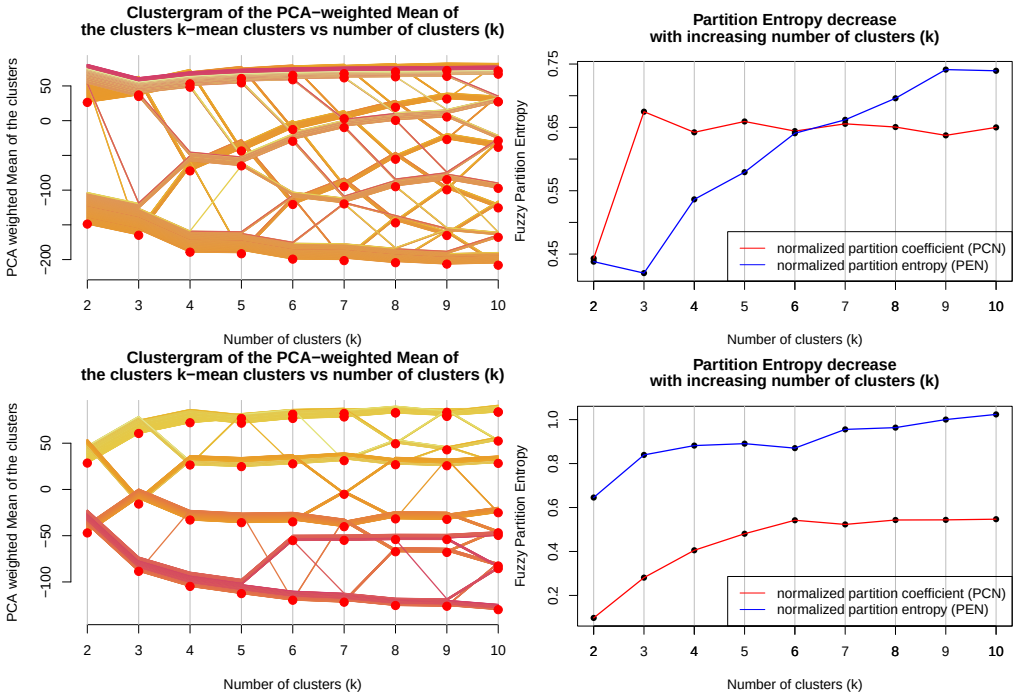

Figure 8: Clustergram plots on fuzzy c-means algorithms, with associated fuzzy partition indices, for 3 and 6 artificial communities. The normalised partition coefficient (red) should be maximized, whereas the normalised partition entropy (blue) should be minimized.

grammar).
The `PlotEcotone` function essentially facilitates the extraction of elements from the `EcotoneFinder` list, for plotting purposes (the full list of arguments it accepts can be found on its dedicated help page, see `?EcotoneFinder::plotEcotone`). It provides a quick way of comparing the outputs of different methods (by choosing them in the `plot.method` argument), either with each other, with the actual species abundance curves in the community matrix, with environmental data or clusters (that may also be computed with the `EcotoneFinder` function), or with their derivatives – to highlight rates of change (see slopes, and derivatives of community structure).
We will make use of this function in the following section to illustrate the use of the `EcotoneFinder` function.

Detrended Correspondance Analyses (DCA):

Detrended Correspondence Analyses (DCA) were developed by Hill and Gauch (1980)[15], to circumvent some of the shortcomings of Correspondence Analyses (CA) – in particular, the distortion of the ecological gradients by the arch effect. As the name indicates, this is done through detrending, a method that centers the second axis of the CA on zero (described in more details in Gauch (1982)[11]). More relevant for ecotonal studies, the distortion of the gradient is additionally corrected by rescaling, leading to constant beta diversity change (per axis unit) along the ordination space. In other words, for three samples A, B, and C, if the distance (in the ordination space) between A and C is twice as much as the distance between A and B, then the beta diversity is effectively twice as much between A and C than between A and B – regardless of the position of the samples over geographical space. This property of DCA enables it to consistently capture community variability along spatial gradients [10].

DCA has thus been used in several studies on ecotone characteristics, as a technique to statistically locate ecotone positions [22][39][5]. These investigations relied on the plotting of the first axis of the DCA (site scores) against the actual spatial dimension of the gradient. In such representations, sites that are neighbors both in the physical space and in the ordination space appear as plateaux (community cores), whereas sites that are close in the physical space but not in the ordination space form steep slopes (ecotones). The

`EcotoneFinder` function internally uses the `decorana` function of the `vegan` package ([24]), if `method = "dca"`, as this is the most standard procedure. This can be seen on Figure 9. For convenience, the entirety of the results produced by the `decorana` function are returned by the `EcotoneFinder` function.
Standardization of ecological data prior to multivariate analyses is commonplace, and usually ensures that differences in species abundances – either rare or common – do not disproportionately skew the output. Legendre & De Cáceres, 2013 [20], recommend the use of the hellinger transformation for studies on beta diversity, and it is thus the default method in the `EcotoneFinder` function (`standardize` argument). Any other transformation method provided by the `decostand` function of the `vegan` package may be used instead, to fit the specific needs of the data under consideration.

```
# DCA analyses:
DCA2 <- EcotoneFinder(Community2[,2:ncol(Community2)], dist = Community2$Distance,
                      method = "dca", standardize = "hellinger")
DCA3 <- EcotoneFinder(Community3[,2:ncol(Community3)], dist = Community3$Distance,
                      method = "dca", standardize = "hellinger")

# Visualistation:
plotEcotone(data = DCA2, plot.data = FALSE, plot.method = c("dca"), axis.number = 1,
            col.method = "#26A63A", ylab = "DCA first axis")
plotEcotone(data = DCA3, plot.data = FALSE, plot.method = c("dca"), axis.number = 1,
            col.method = "#26A63A", ylab = "DCA first axis")
```

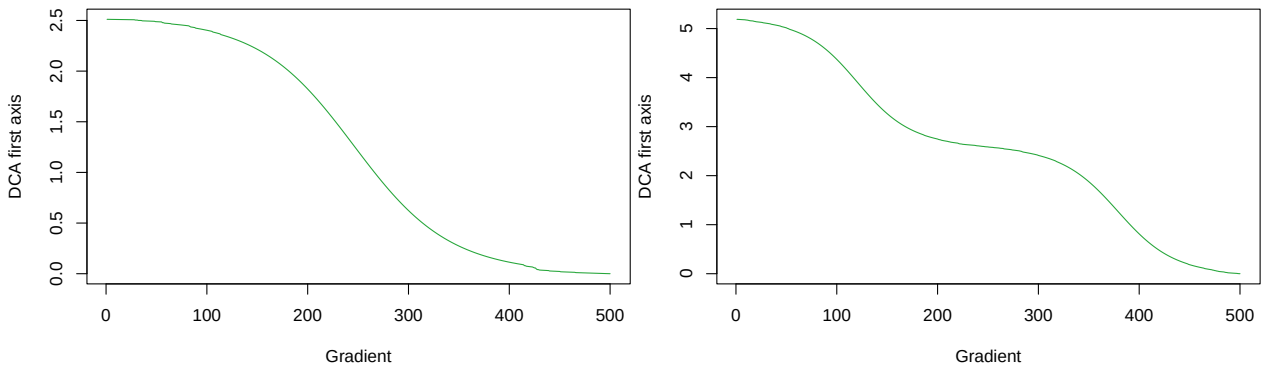

Figure 9: Evolution of the first axis of Detrended Correspondance Analyses along the artificial gradient. Plateaux and slopes are indicative of the positions of communities and ecotones, respectively.

Interestingly, the variations of the second axis of the DCA may also provide information to refine the position of the centre of the ecotone. Their intersection correctly pinpoint the ecotone at 250 gradient units for the left panel, and around 125 and 375 for the right panel, respectively (Fig. 10), as expected given the structure of our artificial data. For that reason, the `axis number` argument can be set up to four (which is the number of axis that the `decorana` function finds, see [24]), which allow for the exploration of the different axis produced by DCA analyses.

```
plotEcotone(data = DCA2, plot.data = FALSE, plot.method = c("dca"), axis.number = 2,
            col.method = c("#26A63A", "#FFC183"), ylab = "DCA axis")
plotEcotone(data = DCA3, plot.data = FALSE, plot.method = c("dca"), axis.number = 2,
            col.method = c("#26A63A", "#FFC183"), ylab = "DCA axis")
```

Fuzzy clustering:

A possible method to safeguard the gradualness of ecological gradients (necessary for ecotone characterization) and the patchiness of ecological communities (necessary for ecosystem classification) is the use of fuzzy clustering techniques. Fuzzy set theory has been formalized by [43] as an alternative to Boolean (or

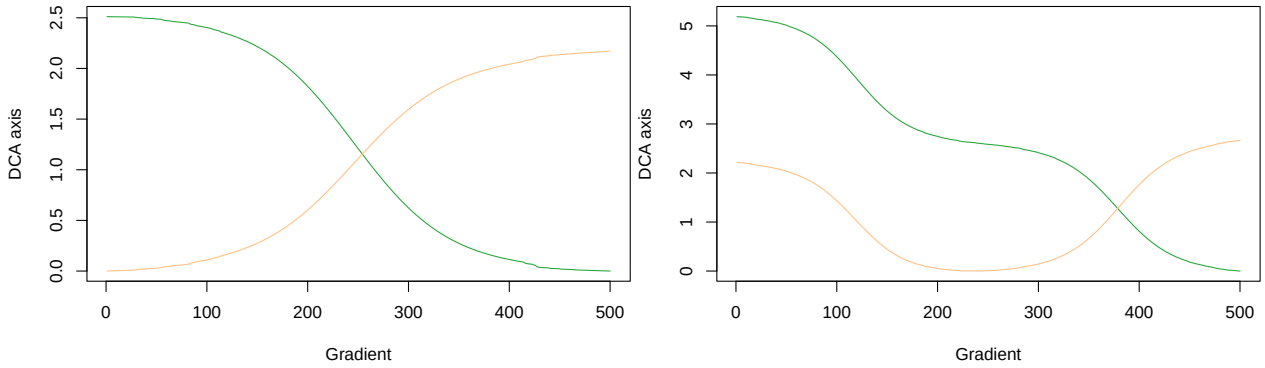

Figure 10: Evolution of both the the first and second axis of Detrended Correspondance Analyses along the artificial gradient. The interaction (and particularly intersetion) between the two axis brings some more light on the position of the ecotones and communities.

crisp) set theory. They latter attracted the attention of landscape ecologists for their ability to account for the "inherent fuzziness of biological spatial structuring observed in the real world" [3].
In fuzzy set theory, observations (usually sample sites in ecology) are assigned membership values in each fuzzy cluster in the range [0, 1], which expresses the degree to which a given observation meets the definition of each of the cluster centroid [31]. In other words, the fuzzy cluster centroids correspond to the archetypal composition of the different community types [3], and each site is given membership values in each cluster, depending on how well the site composition reflects the centroid composition. Once plotted against the spatial dimensions of the gradient, the membership grades of a given cluster usually form a plateau surrounded by two declining edges, corresponding to a community core surrounded by two ecotones. Successive overlapping clusters further produce two distinct edges for every ecotone. Differences in the value of some, or all, of the edge characteristics may provide additional information on the geometry (e.g. symmetry, regularity, abruptness) of the ecotone. This point will be developped later (see Characterising ecotones).
The `EcotoneFinder` function regroups three different common algorithms for the computation of fuzzy clusters on R: the `fanny` algorithm, from the `cluster` package; the `vegclust` algorithm, from the `vegclust` package; and the `cmeans` algorithm, from the `e1071` package. These three functions are mostly equivalent, but may produce slightly different outputs (Fig. 11). Choice is left to the user as to which one performs best, given the particular set of data under consideration.

```
# Fuzzy cluster analyses:
Fuzzy2 <- EcotoneFinder(Community2[,2:ncol(Community2)], dist = Community2$Distance,
                        method = c("fanny", "vegclust", "cmeans"), groups = 2, m.exp = 2,
                        standardize = "hellinger")
Fuzzy3 <- EcotoneFinder(Community3[,2:ncol(Community3)], dist = Community3$Distance,
                        method = c("fanny", "vegclust", "cmeans"), groups = 3, m.exp = 2,
                        standardize = "hellinger", seed = 11)

# Visualistation:
plotEcotone(data = Fuzzy2, plot.data = FALSE, plot.method = c("fanny"),
            col.method = c("#D33F6A", "#E2E6BD"), ylab = "Membership grades",
            title = "Fanny algorithm (2 clusters)")
plotEcotone(data = Fuzzy2, plot.data = FALSE, plot.method = c("vegclust"),
            col.method = c("#D33F6A", "#E2E6BD"), ylab = "Membership grades",
            title = "Vegclust algorithm (2 clusters)")
plotEcotone(data = Fuzzy2, plot.data = FALSE, plot.method = c("cmeans"),
            col.method = c("#D33F6A", "#E2E6BD"), ylab = "Membership grades",
            title = "Cmeans algorithm (2 clusters)")
```

```
plotEcotone(data = Fuzzy3, plot.data = FALSE, plot.method = c("fanny"),
            col.method = c("#D33F6A", "#E99A2C", "#E2E6BD"), ylab = "Membership grades",
            title = "Fanny algorithm (3 clusters)")
plotEcotone(data = Fuzzy3, plot.data = FALSE, plot.method = c("vegclust"),
            col.method = c("#D33F6A", "#E99A2C", "#E2E6BD"), ylab = "Membership grades",
            title = "Vegclust algorithm (3 clusters)")
plotEcotone(data = Fuzzy3, plot.data = FALSE, plot.method = c("cmeans"),
            col.method = c("#D33F6A", "#E99A2C", "#E2E6BD"), ylab = "Membership grades",
            title = "Cmeans algorithm (3 clusters)")
```

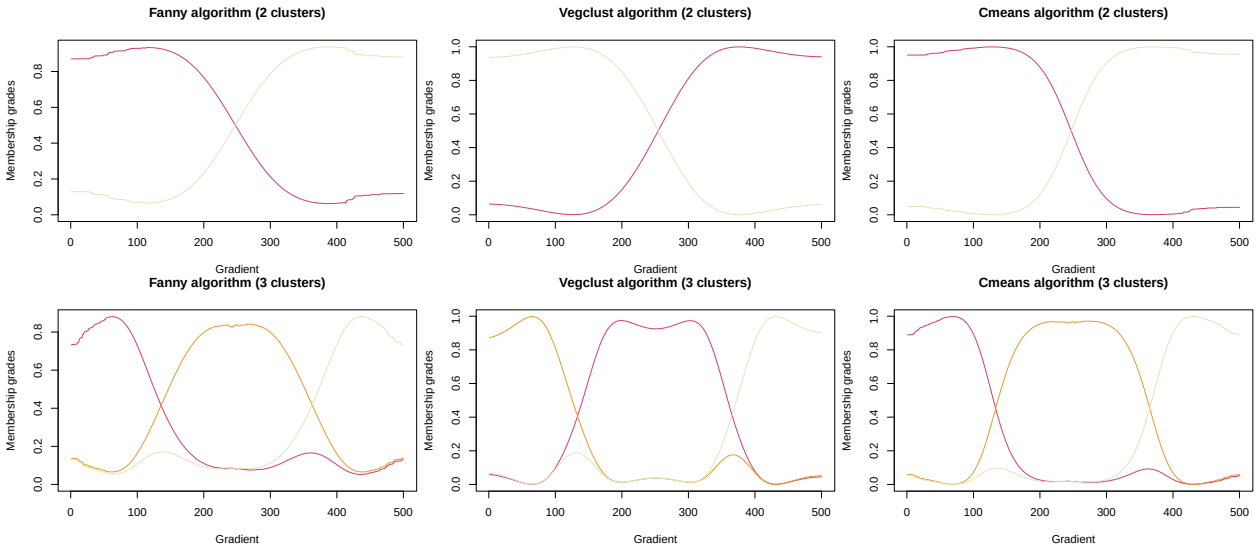

Figure 11: Evolution of the fuzzy clusters along the gradient, depending on the selected fuzzy algorithm.

Fuzzy clusters require some additional parameters:

Number of groups: First, the number of desired groups needs to be specified (through the `groups` argument), the selection of which we discussed in Internal structure of the data. Fuzzy cluster algorithms typically fail – or may produce unstable results – for $k > (n-1)/2$, where n is the number of observations in the data. This is an important limitation to acknowledge, particularly when the sampling effort has been limited by practical constraints, and only a small number of observations are available.

Membership exponent: Second, the membership exponent needs to be specified (the `m.exp` argument). This exponent controls the degree of fuzzification of the clusters, with higher values leading to increased fuzziness. Most of the time, it is defaulted at `m.exp = 2`, and there is no objective method – to our knowledge – to choose this value. Ecological data with high internal variability may require lower values in order to produce clear outputs [6]. Although this may provide some indirect information on the level of internal variability of the ecological communities under consideration, this also introduces an important bias when comparing studies or ecosystems. The membership exponent value indeed influences the shape of the clusters. It is therefore advised to keep `m.exp` at a standard values, if cluster shapes are to be compared between gradients (e.g. if several transects were sampled for a study, and need to be compared).

Seed: As can be seen on the Fig. 11, the order of the fuzzy clusters changes along the gradient between the `cmeans`, `fanny`, and `vegclust` algorithms (the order of the colored lines). This is due to the random selection of vectors of seeds for the initial clusters in the algorithm – i.e. although the elements in each cluster remain the constant, the algorithm does not necessarily start by the same "first" elements. Assigning a specified seed to the function solves the issue (Fig. 12). The seed is internally used by the `EcotoneFinder`

function (encapsulated within `withr::with_seed`) and will not interfere with R global environment. Setting a seed may be of interest when series of similar gradients have been sampled (either as time series or as field replication). It indeed facilitates the automated comparison between different portions of ecological gradients.

```
# Fuzzy cluster analyses:
Fuzzy3 <- EcotoneFinder(Community3[,2:ncol(Community3)], dist = Community3$Distance,
                        method = c("fanny", "vegclust", "cmeans"), groups = 3, m.exp = 2,
                        standardize = "hellinger", seed = 12)

# Visualistation:
plotEcotone(data = Fuzzy3, plot.data = FALSE, plot.method = c("fanny"),
            col.method = c("#D33F6A", "#E99A2C", "#E2E6BD"), ylab = "Membership grades",
            title = "Fanny algorithm (3 clusters)")
plotEcotone(data = Fuzzy3, plot.data = FALSE, plot.method = c("vegclust"),
            col.method = c("#D33F6A", "#E99A2C", "#E2E6BD"), ylab = "Membership grades",
            title = "Vegclust algorithm (3 clusters)")
plotEcotone(data = Fuzzy3, plot.data = FALSE, plot.method = c("cmeans"),
            col.method = c("#D33F6A", "#E99A2C", "#E2E6BD"), ylab = "Membership grades",
            title = "Cmeans algorithm (3 clusters)")
```

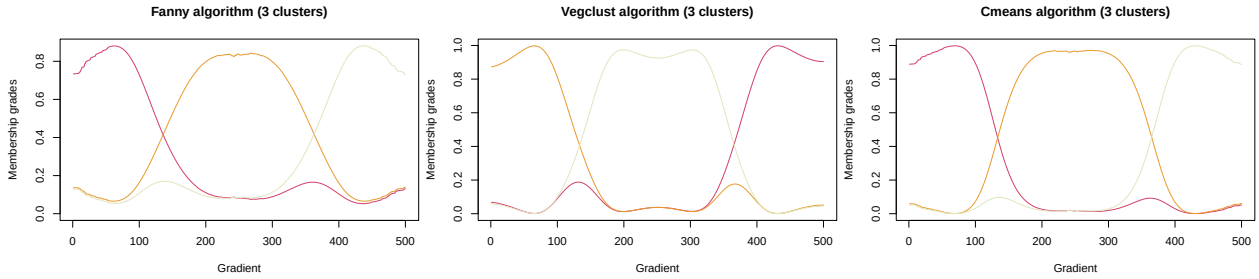

Figure 12: Controling the order of the clusters along the gradient by setting a seed value.

Diversity indices:

Ecotones have been historically linked to the occurrence of peaks of biodiversity across landscapes (see [23]), although this assumption has undergone much debate in recent literature [29]. For convenience – and in order to allow for the easy testing of this relation – common biodiversity indices have been implemented in the `EcotoneFinder` function (the `diversity` argument). At the moment, the implemented indices include the species richness, the Shannon-Wiener index and the Pielou evenness (all through the use of the `vegan` package). The exponential of the Shannon-Wiener index ($exp(H')$) is also provided, and corresponds to the transformation of the Shannon-Wiener index into the Hill's numbers family of indices (see [17]), to access "effective number of species".
The variation of these indices is presented in Fig. 13 for the case of our artificial data. Note that the "expected" increase in alpha diversity around the ecotone is mostly a result of the theoretical assumptions that have been used to create the artificial data (chiefly the symmetry of the species response curves produced by the Gaussian formula, and the internal homogeneity of the artificial communities – idealistic compared to most field observations – that causes all of the species of one community to overlap with all of the species of the other). This type of patterns is by no means a general finding in the ecotone literature.

```
# Diversity indices:
Diver2 <- EcotoneFinder(Community2[,2:ncol(Community2)], dist = Community2$Distance,
                        method = c("diversity"), diversity = "all")
```

```
# Visualistation:
plotEcotone(data = Diver2, plot.data = FALSE, plot.method = c("diversity"),
            diversity = c("SpeciesRichness", "ExpShannon"),
            col.method = c("#88C3C8", "#2A5676"), ylab = "Effective number of species",
            title = "Species richness and Exponential of Shannon-Wiener index")
plotEcotone(data = Diver2, plot.data = FALSE, plot.method = c("diversity"),
            diversity = c("Shannon"),
            col.method = c("#419F44"), ylab = "Index value",
            title = "Shannon-Wiener index")
plotEcotone(data = Diver2, plot.data = FALSE, plot.method = c("diversity"),
            diversity = c("Pielou"),
            col.method = c("#8346A1"), ylab = "Index value",
            title = "Pielou evenness")
```

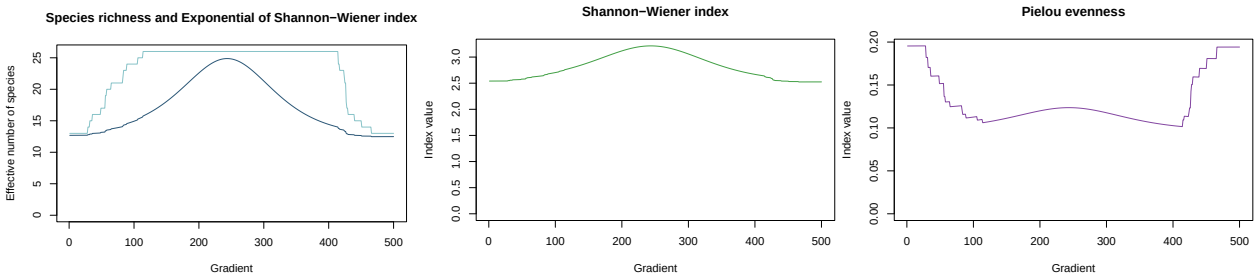

Figure 13: Variations of the species richness, the Shannon-Wiener index, the exponential of the Shannon-Wiener index and the Pielou's eveness along the gradient.

For convenience – and before continuing – let's create an object that contains all of the implemented analyses (setting the `method` argument to `all`, and the `diversity` argument to `all`), so that it can be re-used throughout this document.

```
# Fuzzy cluster analyses:
EcoFind <- EcotoneFinder(Community3[,2:ncol(Community3)], dist = Community3$Distance,
                         method = "all", groups = 3, m.exp = 2,
                         diversity = "all",
                         standardize = "hellinger", seed = 12)
```

Visualisation with ggplot grammar:

Due to the widespread use of the `ggplot2` package, and in order to (i) facilitate the preparation of publishable figures and (ii) add more flexibility to the plot rendering, a ggplot compatible function has been implemented in the `EcotoneFinder` package.
The `ggEcotone` function can accept additional ggplot layers, through the use of the `+` sign (Fig. 14). In a way, the `ggEcotone` function internally creates the first layers of a ggplot object, so that the user may concentrate on the clarity, readability, and design of the graphical output. Similarly to the aforementioned `PlotEcotone` function, it extracts elements of the `EcoFind` list produced by the `EcotoneFinder` function to produce its graphical outputs. The `method` argument selects which analyses to plot, and the `plot.data` argument allows for the plotting of the unmodified species abundance curves. Arguments like `col`, `title`, `xlab`, and `ylab` have been passed inside the `ggEcotone` function, and should not be specified through other ggplot layers.
Some examples below:

```
# Basic plot:
Plot <- ggEcotone(EcoFind, plot.data = FALSE,
```

```
                  method = c("cmeans"),
                  col = c("#D33F6A", "#E99A2C", "#E2E6BD"),
                  title = "fuzzy clusters (cmeans algorithm)",
                  xlab = "Gradient", ylab = "Membership grades")
Plot

# Adding other ggplot layers:
require(ggplot2)
Plot <- Plot + theme_bw() +
  theme(plot.title = element_text(hjust = 0.5, face="bold"))
Plot

# Add smoothing function:
for (i in 2:ncol(Plot$data)) {
  Plot <- Plot + geom_smooth(data = Plot$data,
                             aes_string(x = Plot$data[,1], y = Plot$data[,i]),
                             col = "grey60", lty = 2)
}
Plot
```

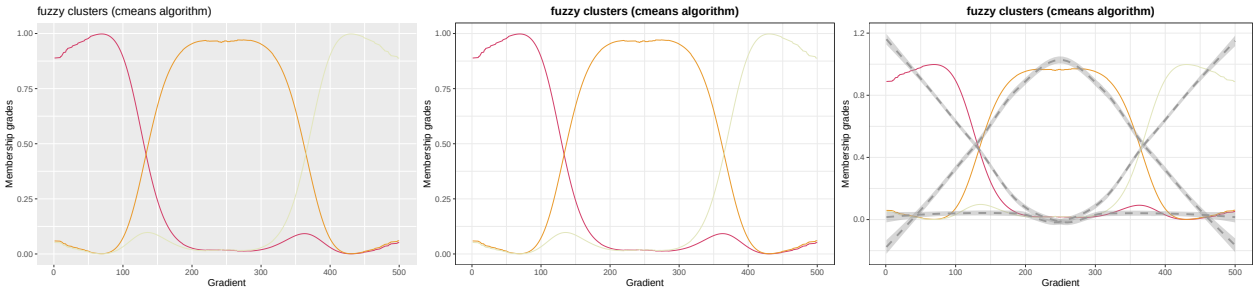

Figure 14: Basic use of the ggEcotone function for ecotone visualisation.

The ggEcotone function also provides some faceting options (Fig. 15), to facilitate comparisons between methods.

```
# Comparing the fuzzy algorithms:
Plot <- ggEcotone(EcoFind, plot.data = FALSE,
                  method = c("fanny", "cmeans"),
                  facet = c("fanny", "cmeans"),
                  col = c("#D33F6A", "#E99A2C", "#E2E6BD"),
                  title = "fuzzy clusters",
                  xlab = "Gradient", ylab = "Membership grades") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5, face="bold"))
Plot

# This can be used in conjunction with the species response curves:
Plot <- ggEcotone(EcoFind, plot.data = TRUE,
                  method = c("cmeans", "diversity"),
                  col = colorspace::diverging_hcl(39, palette = "Berlin"),
                  diversity = c("SpeciesRichness"),
                  facet = list(c("data"), c("cmeans"), c("diversity")),
                  title = "Species distribution, fuzzy clusters \n and Species richness",
                  xlab = "Gradient", ylab = "Membership grades") +
```

```
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5, face="bold"))
Plot
```

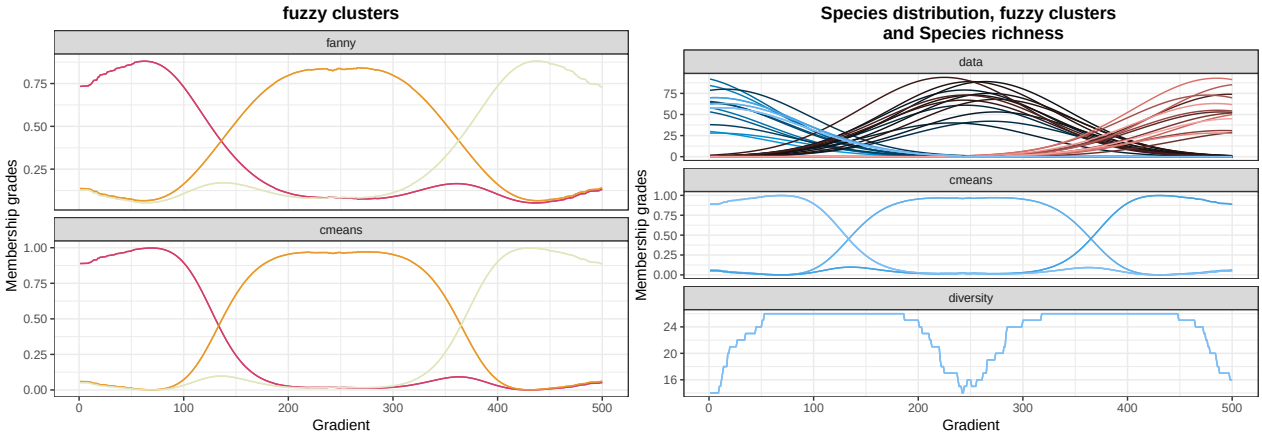

Figure 15: Examples of ggEcotone outputs with facets

As one can see, the coloring of the different facets can be difficult to control. The colors are indeed internally recycled by `ggplot` for each facet. The use of individual plots on a grid layout is therefore encouraged for improved control over the plot rendering (Fig. 16).

```
# Multiplot layout:
Plot1 <- ggEcotone(EcoFind, plot.data = TRUE,
                   method = c("none"),
                   col = colorspace::diverging_hcl(39, palette = "Berlin"),
                   facet = NULL,
                   title = "Species distributions", xlab = NULL,
                   ylab = "Abundances") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5, face="bold"))

Plot2 <- ggEcotone(EcoFind, plot.data = FALSE,
                   method = c("cmeans"), col = c("#023FA5", "#BEC1D4", "#D6BCC0"),
                   facet = NULL, title = "Fuzzy clusters", xlab = NULL,
                   ylab = "Membership grades") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5, face="bold"))

Plot3 <- ggEcotone(EcoFind, plot.data = FALSE,
                   method = c("diversity"),
                   col = c("#26A63A", "#B4B61A"), facet = NULL,
                   diversity=c("SpeciesRichness", "ExpShannon"),
                   title = "diversity indices", xlab = "Gradient",
                   ylab = "Index scores") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5, face="bold"))

require(Rmisc)
Rmisc::multiplot(Plot1, Plot2, Plot3)
```

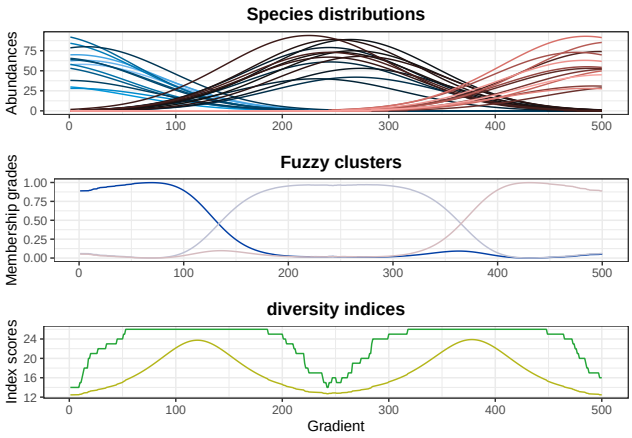

Figure 16: Arranging plots on a grid instead of facetting

Colouring community types on a plot:

The previous figure makes it obvious that the coloring of the species response curves can be a little bit random – particularly if we now assume that we do not know *à priori* the partitioning of the recorded species into the detected community types, as it would be the case if we were dealing with real field data.
The `CommunityColor` function is designed for this purpose. It essentially extracts information from the fuzzy clusters (either of the implemented algorithms chosen with the `method` argument), de-fuzzifies the sets and assigns each species to their highest-ranking cluster. It then assigns a color to each cluster (Fig. 17), which can either be user defined (with the `col` argument) or chosen among the `colorspace` palettes.

```
# Colouring function:
ComColour <- CommunityColor(EcoFind, method = "cmeans", palette = "Berlin")

# Species response curve plot:
Plot1 <- ggEcotone(EcoFind, plot.data = TRUE,
                   method = c("none"), col = ComColour,
                   facet = NULL,
                   title = "Species distributions", xlab = NULL,
                   ylab = "Abundances") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5, face="bold"))
Plot1
```

Characterising communities:

Community composition:

So far, the focus has been on the position of communities and ecotones along gradients. This first step is necessary, and particularly useful in cryptic environments where the delineation of ecosystem types is unknown (e.g. many aquatic environments or microbial ecosystems). Most ecological studies, however, need to get past purely observational results, and provide information for species or habitat conservation strategies, or improve the current understanding of ecosystem functions and other more theoretical ecological questions.
The centroid composition of the fuzzy clusters corresponds to a probabilistic way of looking at ecological community composition [3][6]. This prove valuable, not only conceptually, but also because both species that are strongly linked with a particular community cluster (i.e. they are key in defining the community in which they occur), and species that are not linked to any particular cluster (i.e. they may be ubiquitous at the spatial scale under consideration) are being treated.

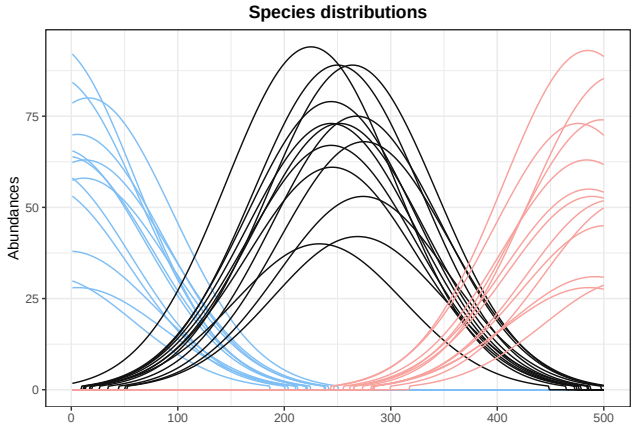

Figure 17: Partitioning species into their dominant community type, given fuzzy cluster classification

The `ExtractCentroid` function provide an easy way to visualize the composition of fuzzy cluster centroids. To facilitate interpretation, different methods for the standardization of the fuzzy centroids are proposed (through the `normalized` argument, Fig. 18), and can be used in synergy for a better understanding of the species composition of each community.
The `ExtractCentroid` function returns the fuzzy cluster composition as a data frame (after the normalization step), and a `ggplot` object (if `return.plot = TRUE`). The `ggplot` object can either be directly plotted by the `ExtractCentroid` function (if `plot = TRUE`), or reused externally in conjunction with other `ggplot` layers.

```
# Extracting centroids (raw):
Centroids <- ExtractCentroid(EcoFind, method = "cmeans", normalized = "none",
                             plot = FALSE, cex.x = 9, col = unique(ComColour))
Centroids$GGplot +
  theme(legend.position = "none")

# Extracting centroids (normalised by species):
Centroids <- ExtractCentroid(EcoFind, method = "cmeans", normalized = "species",
                             plot = FALSE, cex.x = 9, col = unique(ComColour))
Centroids$GGplot +
  theme(legend.position = "none")

# Extracting centroids (normalised by clusters):
Centroids <- ExtractCentroid(EcoFind, method = "cmeans", normalized = "cluster",
                             plot = FALSE, cex.x = 9, col = unique(ComColour))
Centroids$GGplot +
  labs(fill = "Clusters")
```

With these simple artificial data, the segregation of the different species in the three fuzzy clusters is clear and straightforward – the only interesting pattern being the slightly more mixed centroid values for species belonging to the middle cluster. This community indeed overlaps on both sides with the other communities, meaning that its typical species composition is less homogeneous than both its bordering communities. In a probabilistic sense – for any new sampling event – the probability of finding *species 1* would be about 85% in the left inside community, 15% in the middle community, and nearly 0% in the right-hand cluster (where it does not occur in our artificial set). *Species 15*, however, could be found in its main community (i.e. middle cluster: 62%), but also in the two other communities (left cluster: 13%, right cluster: 25%).
Now let's create a slightly more intricate example (Fig. 19), to better explore the interpretation of fuzzy cluster centroids:

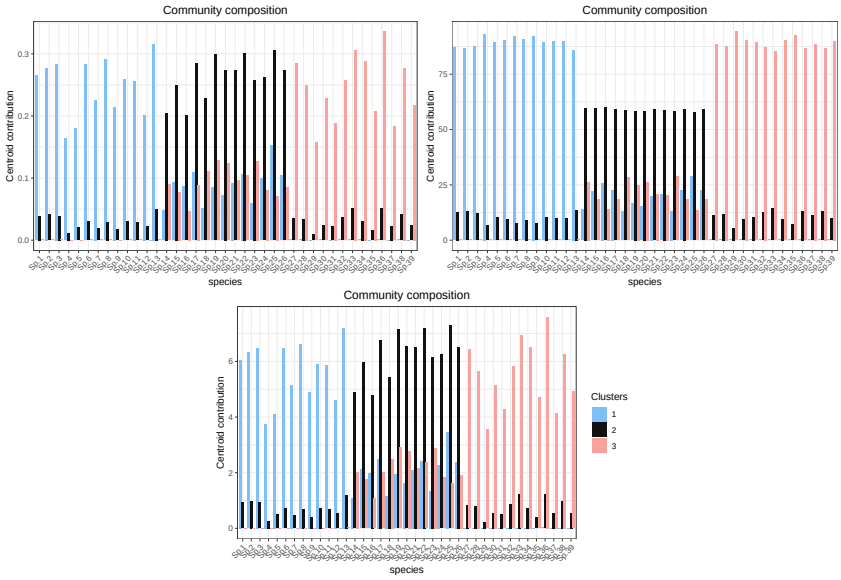

Figure 18: Barcharts representing the centroid composition of the different fuzzy clusters. Note that normalising centroids per cluster produces equivalent results than the raw values, but it sums up all the values to a hundred for each cluster to facilitate interpretation. When normalised by species, the sum for each species is brought to a hundred. The function may also returns the ggplot object, so that it can be used as a basic ggplot. As the function internally use 'scale_fill_manual', the 'ggplot2::labs' function should be used with the 'fill' argument.

```
# Artificial data:
Community <- SyntheticData(SpeciesNum=40, CommunityNum=5, SpCo=c(12,12,5,7,4),
                           Length = 500,
                           Parameters=list(a=c(60, 60, 20, 15, 60),
                                           b=c(0, 500, 50, 450, 250),
                                           c=c(0.01, 0.01, 0.02, 0.02, 0.0001)),
                           dev.c=.0024, dev.a = 20, dev.b = 30,
                           pal=c("#023FA5", "#8E063B", "#A1A6C8", "#CA9CA4", "grey20"),
                           title = "Communities with localised and ubiquituous species")
```

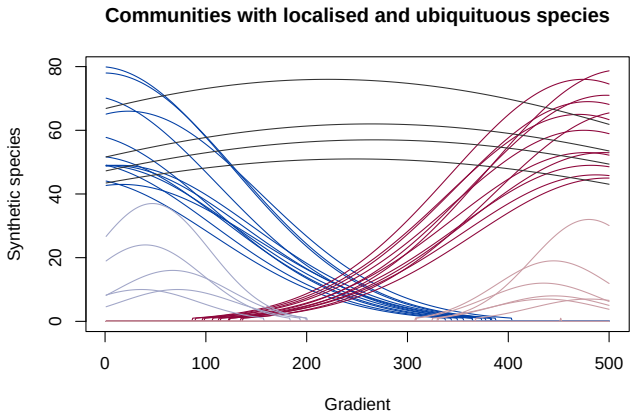

Figure 19: Artificial data with 2 communities containing overlapping species and non-overlaping species, as well as some species that occur throughout the gradient

```
## Fuzzy clusters and centroids:
# Fuzzy cluster analyses:
```

```
Fuzzy <- EcotoneFinder(Community[,2:ncol(Community)], dist = Community$Distance,
                       method = c("fanny", "vegclust", "cmeans"), groups = 2, m.exp = 2,
                       standardize = "hellinger", seed = 12)

## Centroids:
# Normalised by clusters:
Centroids <- ExtractCentroid(Fuzzy, method = "cmeans", normalized = "cluster",
                             plot = FALSE, cex.x = 9, col=c("#A1A6C8", "#CA9CA4"))
Centroids$GGplot +
  theme(legend.position = "none")

# Normalised by species:
Centroids <- ExtractCentroid(Fuzzy, method = "cmeans", normalized = "species",
                             plot = FALSE, cex.x = 9, col=c("#A1A6C8", "#CA9CA4"))
Centroids$GGplot +
  labs(fill = "Clusters")
```

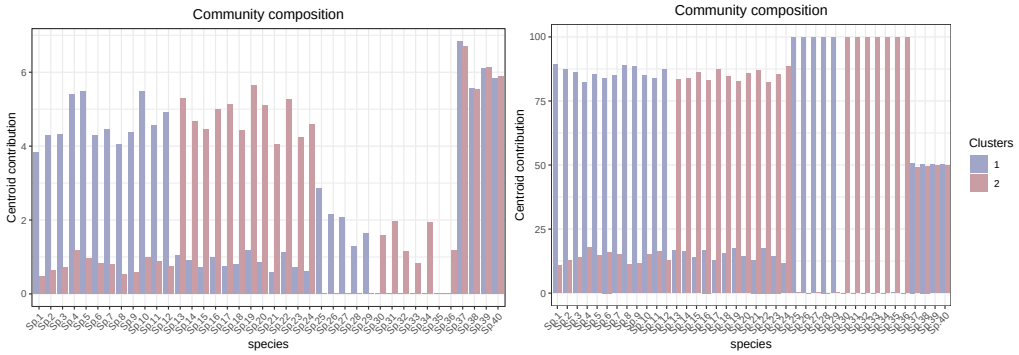

Figure 20: Fuzzy centroids

Some distinctions can be made regarding the distribution of the different species into the two main communities, on Fig. 20:

. The few ubiquitous species (in our example: the one numbered 37 to 40) have nearly identical centroid values in both clusters, indicating that they equally occur in all communities.

. The species that are localized in a single community (from 25 to 29 and 30 to 36 respectively, for the two communities) reach a 100% on the normalized-by-species plot, indicating that they do not overlap with any other community. They also reach lower centroid values on the normalized-by-cluster plot due to their overall low abundances. If we had chosen to create artificial data with localized species presenting higher abundance peak values, their centroid contribution would also have been higher.

. As previously, the overlapping species of each communities share mixed centroid values, with one cluster being majoritary.

These distinctions may be quite informative, when dealing with data containing many species with unknown distributions and spreads in potential community types.
The sensibility of fuzzy centroids to overall species abundance means that rare species are easily underrepresented in normalized-by-cluster (or raw) centroid values. This bias may nevertheless be overcome by choosing to normalize by species instead. If the focus is on abundant species only, the `threshold` argument can be set to be > 0 in order to discard rare species from the graphical output. This can be useful for readability purposes, particularly when the number of rare species is high (e.g. microbial communities).
Be mindful, however, that using `threshold = 0` or `threshold = 1` may lead to non-sense results.

Networks:

In pretty much the same way than the networks presented in the Internal structure of the data section, networks can be drawn out of the centroid compositions of the fuzzy clusters. The main difference is that the networks are not based on the raw species data anymore, but on transformed (i.e. clustered) data. This added level of abstraction means that the comparison now takes place between the archetypal community compositions obtained after statistical analyses, instead of spatial coincidence between species.
To do so, The `NetworkEco` function (see Fig. 21) directly recycles the centroids of the fuzzy clusters, either 'as is' (if `dist = "raw"`), or normalized per clusters (i.e. normalized per community types, if `dist = "relative"`). It can also draw networks based on a count of common species between community types (if `dist = "count"`) – that is: the function extract the archetypal species composition of a community type based on a threshold in each species centroid scores, and count the number of shared species per community. This gives a very straightforward measure of the relatedness between two communities, in terms of number of species.
The `plot` argument can then either be set to "community" – to present the links between the species composition of the cluster centroids (Fig. 21 B, C, D and H, I, J) – or to "species" – to highlight the similarities between species, depending on their scores in each cluster centroids (Fig. 21 E, F and K, L). In the latter case, only "raw" or "relative" distance may be computed. More ample discussion on the use of this type of network will be given in the section on data series (Data series and networks).

```
## Artificial communities with different spatial dispositions:
# Communities 1 & 2 closer to each other (by sharing species):
Community12 <- SyntheticData(SpeciesNum=40, CommunityNum=4, SpCo=c(10,10,10,10),
                             Length = 500,
                             Parameters=list(a=c(60, 60, 60, 60),
                                             b=c(0, 250, 500, 100),
                                             c=c(0.012, 0.015, 0.012, 0.009)),
                            dev.c=.0024, dev.a = 20, dev.b = 30,
                            pal=c("#008585","#B8CDAE","#C7522B", "#72E2AD"),
                            title = "A: Communities 1 & 2 share species")

## Analyses (first case):
Fuzzy12 <- EcotoneFinder(Community12[,2:ncol(Community12)], dist = Community12$Distance,
                         method = c("cmeans"), groups = 3, m.exp = 2,
                         standardize = "hellinger")
## Networks:
NetworkEco(Fuzzy12, plot.type = "network", method = "cmeans",
           dist.method = "inner_product",
           plot = "community", dist = "raw", layout = "spring")
title(main  = "B: Raw distances", adj = 0, cex = .5)
NetworkEco(Fuzzy12, plot.type = "network", method = "cmeans",
           dist.method = "inner_product",
           plot = "community", dist = "relative", layout = "spring")
title(main  = "C: Relative distances", adj = 0, cex = .5)
NetworkEco(Fuzzy12, plot.type = "network", method = "cmeans",
           dist.method = "inner_product",
           plot = "community", dist = "count", threshold = 0.25, layout = "spring")
title(main  = "D: Number of shared species", adj = 0, cex = .5)

###### Now comparing species with each other:
plot(NULL, xlim=c(0, 1), ylim=c(0, 1), bty ="n",axes=F,
     frame.plot=F, xaxt='n', ann=FALSE, yaxt='n')
NetworkEco(Fuzzy12, plot.type = "network", method = "cmeans",
           dist.method = "inner_product",
```

```
           plot = "species", dist = "raw", layout = "spring",
           network.group = as.factor(rep(c("C1", "C2", "C3", "Shared C1/C2"),
                                         each = 10)),
           color = c("#008585","#B8CDAE","#C7522B", "#72E2AD"))
title(main  = "E: Raw distances", adj = 0, cex = .5)
NetworkEco(Fuzzy12, plot.type = "network", method = "cmeans",
           dist.method = "inner_product",
           plot = "species", dist = "relative", layout = "spring",
           network.group = as.factor(rep(c("C1", "C2", "C3", "Shared C1/C2"),
                                         each = 10)),
           color = c("#008585","#B8CDAE","#C7522B", "#72E2AD"))
title(main  = "F: Relative distances", adj = 0, cex = .5)
plot(NULL, xlim=c(0, 1), ylim=c(0, 1), bty ="n",axes=F,
     frame.plot=F, xaxt='n', ann=FALSE, yaxt='n')

# Communities 2 & 3 closer to each other:
Community23 <- SyntheticData(SpeciesNum=40, CommunityNum=4, SpCo=c(10,10,10,10),
                             Length = 500,
                             Parameters=list(a=c(60, 60, 60, 60),
                                             b=c(0, 250, 500, 400),
                                             c=c(0.012, 0.015, 0.012, 0.009)),
                            dev.c=.0024, dev.a = 20, dev.b = 30,
                            pal=c("#008585","#B8CDAE","#C7522B", "#F9C29C"),
                            title = "G: Communities 2 & 3 share species")
## Analyses (second case):
Fuzzy23 <- EcotoneFinder(Community23[,2:ncol(Community23)], dist = Community23$Distance,
                         method = c("cmeans"), groups = 3, m.exp = 2,
                         standardize = "hellinger")

## Networks:
NetworkEco(Fuzzy23, plot.type = "network", method = "cmeans",
           dist.method = "inner_product",
           plot = "community", dist = "raw", layout = "spring")
title(main  = "H: Raw distances", adj = 0, cex = .5)
NetworkEco(Fuzzy23, plot.type = "network", method = "cmeans",
           dist.method = "inner_product",
           plot = "community", dist = "relative", layout = "spring")
title(main  = "I: Relative distances", adj = 0, cex = .5)
NetworkEco(Fuzzy23, plot.type = "network", method = "cmeans",
           dist.method = "inner_product",
           plot = "community", dist = "count", threshold = 0.25, layout = "spring")
title(main  = "J: Number of shared species", adj = 0, cex = .5)

###### Now comparing species with each other:
plot(NULL, xlim=c(0, 1), ylim=c(0, 1), bty ="n",axes=F,
     frame.plot=F, xaxt='n', ann=FALSE, yaxt='n')
NetworkEco(Fuzzy23, plot.type = "network", method = "cmeans",
           dist.method = "inner_product",
           plot = "species", dist = "raw", layout = "spring",
           network.group = as.factor(rep(c("C1", "C2", "C3", "Shared C2/C3"),
                                         each = 10)),
           color = c("#008585","#B8CDAE","#C7522B", "#F9C29C"))
title(main  = "K: Raw distances", adj = 0, cex = .5)
```

```
NetworkEco(Fuzzy23, plot.type = "network", method = "cmeans",
           dist.method = "inner_product",
           plot = "species", dist = "relative", layout = "spring",
           network.group = as.factor(rep(c("C1", "C2", "C3", "Shared C2/C3"),
                                         each = 10)),
           color = c("#008585","#B8CDAE","#C7522B", "#F9C29C"))
title(main  = "L: Relative distances", adj = 0, cex = .5)
plot(NULL, xlim=c(0, 1), ylim=c(0, 1), bty ="n",axes=F,
     frame.plot=F, xaxt='n', ann=FALSE, yaxt='n')
```

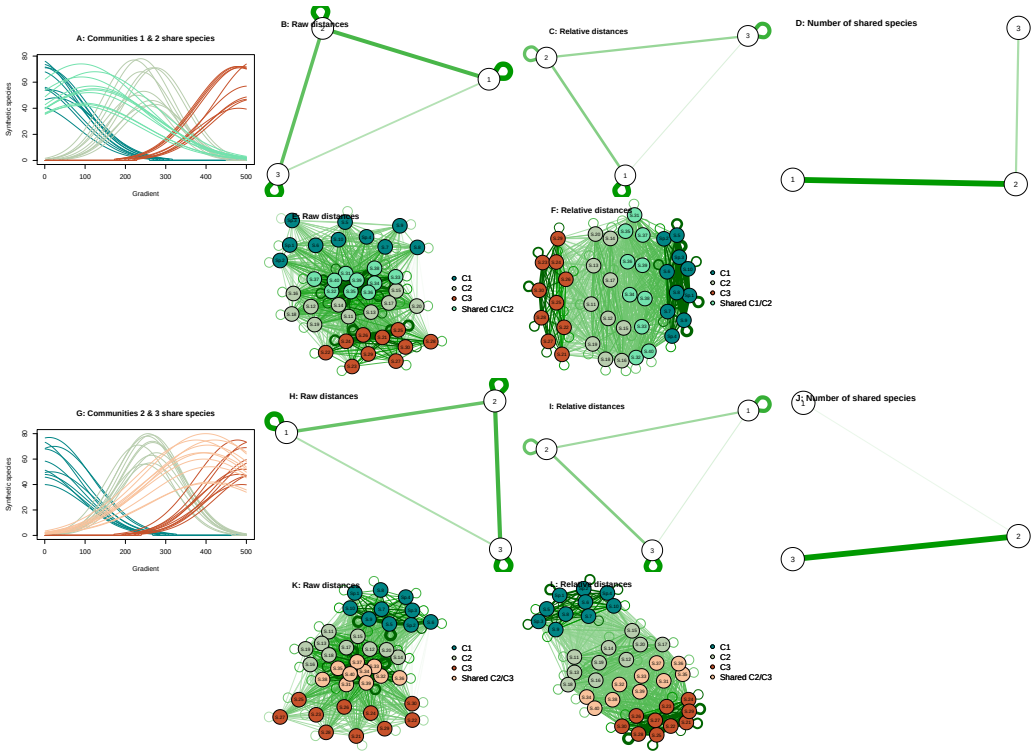

Figure 21: Network visualisations of two different relatedness scenarios for three community types (A: Community 1 and 2 are more related, G: Community 2 and 3 are more related). In both cases, the difference in the nerworks is most obvious when selection a threshold for the count of common species (D and J). Networks of species relations within the cluster centroids correctly place the species shared by community types in between the 'defined' community types (after clustering). Actual field data, where species commonly have more complex distribution patterns, are usually more responsive to this type of representations.

The `NetworkEco` function can also return heat-maps (if `plot.type = "heatmap"`) or correlation plots (if `plot.type = "corrplot"`), to directly access the values corresponding to the edges of the networks (Fig. 22). The `method.corr` argument allow the user to chose what type of outputs the correlation plot should contain (here, numbers). It also accept additional arguments from the `corrplot::corplot` function. Is this case, `is.corr = FALSE`, as we are not working with actual correlation matrices.

```
## First case:
NetworkEco(Fuzzy12, plot.type = "heatmap", method = "cmeans",
           dist.method = "inner_product",
           plot = "community", dist = "raw")
NetworkEco(Fuzzy12, plot.type = "corrplot", method = "cmeans",
           dist.method = "inner_product",
           plot = "community", dist = "count", threshold = 0.25, method.corr = "number",
```

```
            is.corr = FALSE)

# Second case:
NetworkEco(Fuzzy23, plot.type = "heatmap", method = "cmeans",
            dist.method = "inner_product",
            plot = "community", dist = "raw")
NetworkEco(Fuzzy23, plot.type = "corrplot", method = "cmeans",
            dist.method = "inner_product",
            plot = "community", dist = "count", threshold = 0.25, method.corr = "number",
            is.corr = FALSE)
```

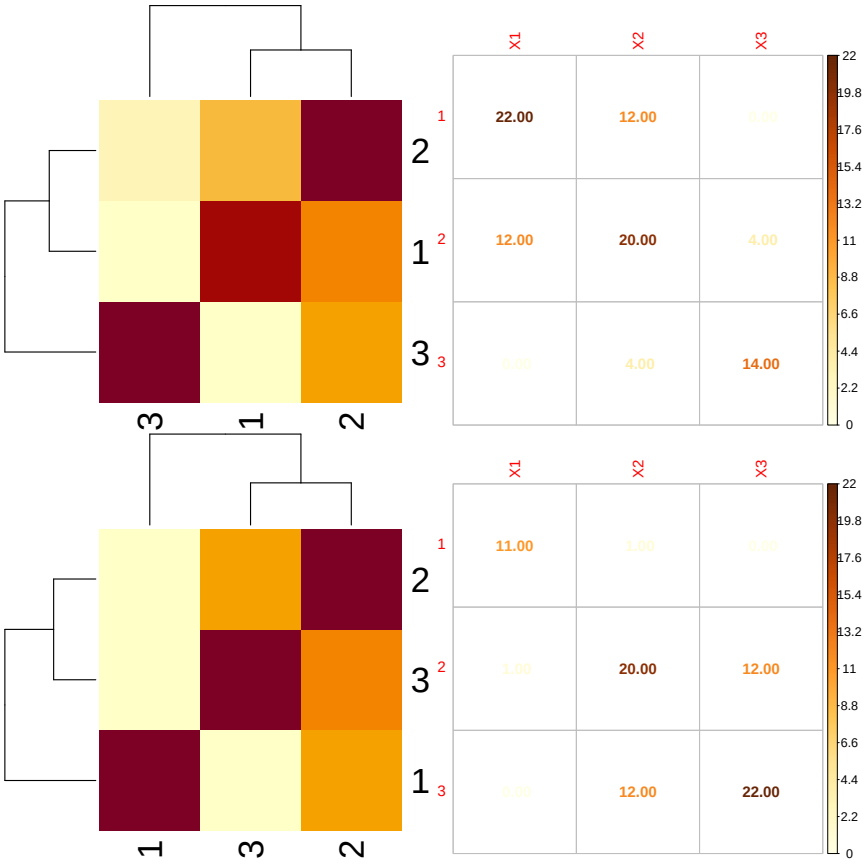

Figure 22: Heatmaps and corrplot visualisations. It can be seen on the hierarchical tree of the heatmaps that the external group changes (either 3 or 1), and that the number of shared species between communities is correctly represented on the corrplots.

Now that we covered how the `EcotoneFinder` package works for characterizing communities, it is time to focus on how to characterize ecotones.

Characterising ecotones:

Ecotones metrics:

We already saw in previous sections how DCA and fuzzy clusters could be used to pinpoint the locations of ecotones along gradients. Now let's go one step further and explore what type of metrics can be extracted from these ecotones. Accessing standardized metrics on ecotone characteristics is crucial to improve the categorization of these landscape features, but also to provide the common ground for cross-scales and cross-ecosystem studies.
A number of such metrics have already been proposed in the literature (see [29]). The schematic view in

Figure 23 summaries the main ones:

. The **Magnitude of Edge Influence** (MEI) – corresponding to the difference of level between two successive plateau of community types.

. The **Depth of Edge Influence** (DEI) – corresponding to the total spatial extent of the ecotonal area along the gradient.

. The **Amplitude of Edge Influence** (AEI) – corresponding to the steepness, or slope, of the ecotone. This can be extracted from the two previous metrics, but discrepancies may exist between the averaged value over the entire ecotone, and the maximum value it takes at a given point of the ecotone (hereafter: AEImax). These differences may be informative as to certain processes occurring in the ecotonal area, and should thus be considered.

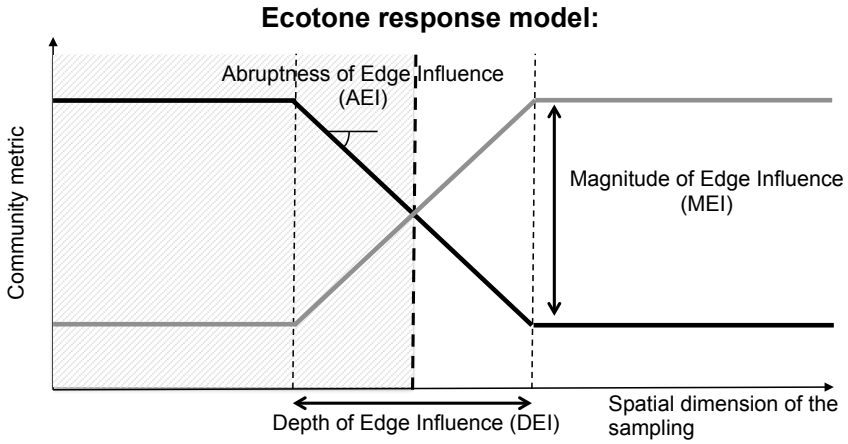

Figure 23: Schematic ecotone, with the graphical representation of its Magnitude of depth influence (MEI), Depth of Edge Influence (DEI), and Amplitude of Edge Influence (AEI).

Fuzzy clusters provide an interesting community metric for that purpose, as their membership grades are in $[0, 1]$, meaning that the MEI, DEI and AEI can be expressed in standardized units regardless of the type of ecological community under consideration. The approach bellow may nevertheless be applied to any other metric of interest – and was initially applied to single species response curves across landscape boundaries [9][26].

Slopes, and derivatives of community structure:

The `EcotoneFinder` package provide a `Slope` function that will return the derivative of a continuous variable. It does so by applying a Moving Split Window (MSW) approach, which provide a step-by-step way of exploring variations along continuous irregular functions (reviewed in [8]). MSW relies on the breaking of the continuous variable into segments (or windows) that are then compared using a dissimilarity function.
The graphical outputs typically present sharp peaks where rapid changes in the measured variable occur, and near-zero plateau where the measured variable remains stable. The width of the windows is user defined and can vary. It typically corresponds to a single sample unit, but several sample units may be pooled together to achieve greater smoothing (e.g. to reduce noise), or to access larger scale patterns.
The `Slope` function internally extract the slope coefficient obtained from linear models (with the use of `stats::lm`) over a user-defined window of observations. The default window size is set to $window = 3$, so that sample points are taken by groups of three. The window then slides by one point at a time and the collected coefficients are stored as the derivative of the initial variable.
This procedure can compute the derivatives of any continuous variable, even if it presents chaotic patterns. One drawback, however, needs to be mentioned: the extremities of the gradients are shortened accordingly to the width of the chosen window (e.g. if $window = 3$, the two most extreme points are discarded on both sides of the gradient, and if $window = 21$, the ten most extreme points are discarded on both sides of the

gradient). This should not have any major consequences if the number of observations is high enough, and if the ecotone relatively centered along the gradient, but may be problematic otherwise.
The `Slope` function accepts results from the `Ecotonefinder` function, in its `method` argument. It would technically accept any other list, so long the chosen `method` appears in the names of the list elements.
The plotting functions presented in previous sections both accept `Slope` outputs on top of the initial `EcotoneFinder` outputs, which enables overlaying plots containing both the initial results of the analyses and their derivatives (Fig. 24).

```
# Slope calculation:
EcoSlope <- Slope(EcoFind, method = c("cmeans", "diversity"), window = 3,
                  diversity = "richness")

# Fuzzy clusters and derivatives (ggplot grammar):
Plot <- ggEcotone(EcoFind, slope = EcoSlope, plot.data = FALSE,
                  method = c("cmeans", "cmeans_slope"),
                  facet = c("cmeans", "cmeans_slope"),
                  col = c("#D33F6A", "#E99A2C", "#E2E6BD"),
                  title = "fuzzy clusters and their derivatives",
                  xlab = "Gradient", ylab = "Membership grades") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5, face="bold"))
Plot

# Fuzzy clusters and derivatives (basic plot):
plotEcotone(data = EcoFind, slope = EcoSlope, plot.data = FALSE,
            plot.method = c("cmeans", "cmeans_slope"),
            col.method = c("#D33F6A", "#E99A2C", "#E2E6BD"),
            col.slope = c("#008585","#B8CDAE","#C7522B"),
            ylab = "Memberships & derivatives",
            magnification.slope = 100)
```

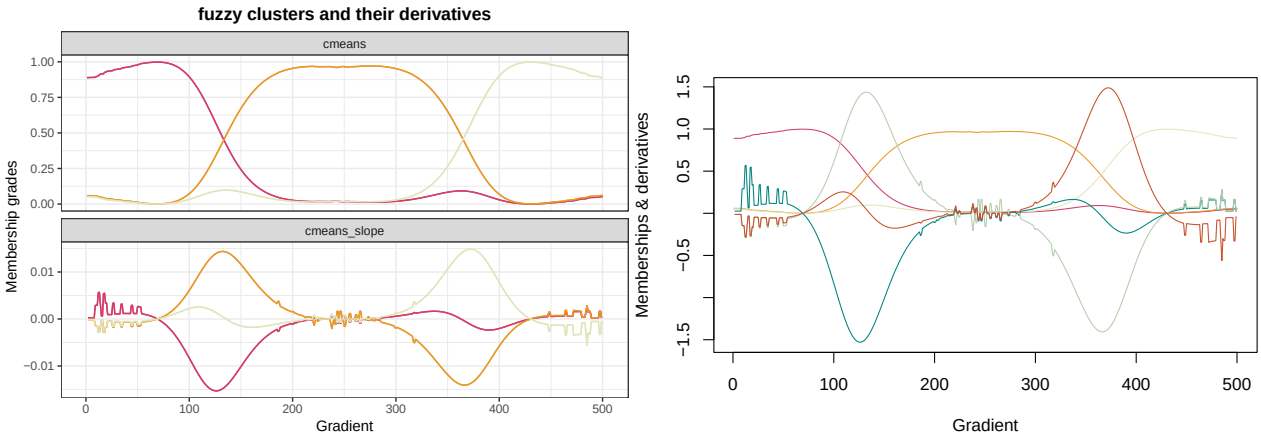

Figure 24: Comparison of the cmeans fuzzy algorythm membership variations and their derivatives.

As it is apparent on the faceted plot, the scale of the derivatives may be orders of magnitude lower than the scale of the analyses' outputs – particularly if the spatial extent of the gradient is orders of magnitude higher than 1. The `magnification.slope` argument of the `plotEcotone` function can circumvent this issue by simply applying a multiplier to the derivative values – making them more visible on the graphical outputs.
The actual derivative values are still directly accessible in the `Slope` object (Table 1):

```
# Derivative values:
pander::pandoc.table(head(EcoSlope$cmeans_slope), keep.line.breaks = T, justify = "lccc",
                     style = 'rmarkdown', row.names = c(1:6),
                     caption = "Derivative values for the three clusters.
                                Only the first few values are being presented")
```

Table 1: Derivative values for the three clusters. Only the first few values are being presented

	1	2	3
1	0.0002638	-0.0001324	-0.0001314
2	0.0002634	-0.0001321	-0.0001313
3	0.000263	-0.0001318	-0.0001312
4	0.0002626	-0.0001315	-0.0001311
5	0.0002622	-0.0001312	-0.000131
6	0.0002618	-0.0001309	-0.0001309

Extracting ecotone metrics:

From there on, the extraction of the aforementioned ecotone metrics (MEI, DEI, AEI and AEImax) is rather straightforward, and only require the use of some base R functions.
The AEImax is accessible through the `max` (or `min`) functions, applied to the derivatives of the corresponding clusters. The MEI requires the selection of an interval of interest (user-defined), but is similarly accessible by subtracting the minimum membership value from the maximum membership value of two successive clusters.
The DEI – corresponding to the spatial extent of the ecotone along the gradient – may be harder to assess, as the beginning and end of clusters (and, arguably, of ecological communities) are often characterized by high levels of variability. This is evidenced by the wavelet patterns in Figure 24. Such patterns can be even more variable and chaotic with real-world data. The starting (and ending) points for the DEI may nevertheless be obtained from the maxima/minima of the second derivatives (use the `Slope` function on the results of the first derivative), or just visually from the variations of the fuzzy clusters themselves. Ultimately, though, the decision regarding whether the *wavelets* are in or out of the ecotonal region is left to the user.
The code bellow provides an example of how to extract these metrics from the first ecotone (on the left side of the gradient) of the artificial communities of Figure 24. The fuzzy clusters effectively draw edges on both sides of the ecotonal region – as one community fades and the other emerges. There is thus two set of values for the AEI and MEI.

```
## Ecotone parameters:
# AEImax and AEImin:
AEImin <- min(EcoSlope$cmeans_slope[,1]) # Cluster 1
AEImax <- max(EcoSlope$cmeans_slope[,2]) # Cluster 2
AEImin
[1] -0.01529607
AEImax
[1] 0.01437219

# MEI (the interval needs to be specified, to target the first ecotone, but can be broad):
MEI_1 <- max(EcoFind$cmeans$membership[0:250,1]) - min(EcoFind$cmeans$membership[0:250,2])
MEI_2 <- min(EcoFind$cmeans$membership[0:250,1]) - max(EcoFind$cmeans$membership[0:250,2])

# DEI:
# Visually: from 90 to 200.
```

```
# With second derivatives:
EcoSlopeSecond <- Slope(EcoSlope$cmeans_slope,
                        method = c("custom"), window = 21)

# The ecotone would be between the minimum and maximum values:
DEI_start <- which(EcoSlopeSecond$Slope_1 == min(EcoSlopeSecond$Slope_1[70:250,1]))
DEI_end <-which(EcoSlopeSecond$Slope_1 == max(EcoSlopeSecond$Slope_1[70:250,1]))
DEI_start
[1] 90
DEI_end
[1] 139
# Length along the gradient:
DEI_1 <- DEI_end - DEI_start
DEI_1
[1] 49

# Or alternatively, at the cross-point between the derivatives of
# the two successive clusters (rounding may be necessary):
Interval <- as.numeric(which(round(EcoSlope$cmeans_slope[0:250,1], digits = 3) ==
                      round(EcoSlope$cmeans_slope[0:250,2], digits = 3)))
Interval
 [1]   1   2   3   4   5   6   7  64  65  66  67  68  69  70  71 211 212 213 214
[20] 215 216 217 218 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235
[39] 239 244 247 248
```

What is of interest here, in the computed interval, are the first two intersect points (i.e. 64 and 211), as they correspond to the beginning and end of the ecotonal region. In this particular case, we can consider that the lowest values on the the left of the interval are artefacts only.
We can use those values to extract the DEI:

```
DEI_2 <- Interval[which(Interval > 120)[1]] - Interval[which(Interval > 50)[1]]
DEI_2
[1] 147
```

Two points may be worth making here:

. First, the extraction of the DEI does require some user-defined criteria (e.g. how to extract it, on which interval of interest to extract it...), which is not optimal for an automated detection of ecotones, though the use of derivatives reduces associated biases.

. Second, the DEI interval differs depending on the procedure that has been chosen to extract it. The cross-points between first derivatives will always give wider intervals than the extremes of the second derivatives, and – to my knowledge – there is no bulletproof reason to select one method over the other. Visual extraction of the DEI interval is yet another valid approach that can produce acceptable results, when confronted to the natural variability of real-world data (instead of artificial data).

Importantly, there is no conceptual reasons to expect symmetry between the two edges of adjacent communities. In our present case, the apparent symmetry between the two edges is a direct consequence of the artificial data generation.
For the moment, however, let's continue with the average value for the Abruptness of Edge Influence (AEI). Indeed, once the DEI has been determined, this average value for the Abruptness of Edge Influence (AEI) can be calculated – this time corresponding to the average rate of change over the ecotonal region, instead of the maximum rate of change (as previously given by the AEImax).
This is given by the simple formula $AEI = MEI/DEI$, and thus depends on the selected method for DEI extraction.

In our case:

```
# If calculated from the second derivatives:
AEI <- mean(c(MEI_1, MEI_2))/DEI_1
AEI
[1] 0.0004539329

# If calculated from the first derivatives:
AEI <- mean(c(MEI_1, MEI_2))/DEI_2
AEI
[1] 0.000151311

# If estimated visually:
AEI <- mean(c(MEI_1, MEI_2))/(200 - 90)
AEI
[1] 0.0002022065
```

Obviously, the results depends on the DEI values. The AEI is nevertheless consistently orders of magnitude lower than the AEImax.
Such difference is not a simple result of the averaging process. When dealing with real-world data, this ratio between the average AEI and the AEImax can be informative as to the regularity of the transition itself, i.e. whether the passing between one community type to the next is gradual throughout, or if there are peaks of rate of change inside the ecotonal region.

Shapes of community edges and ecotone properties: It is not in the scope of this package vignette to discuss the variety of shapes and forms ecotones may take. Reviews on the topic may be found in [37][30] [9][5][26][29], to cite a few. Such discussion is also much better informed by real-world data than by artificial data. It is nevertheless encouraged to pay attention to details, such as patterns of asymmetry, edge irregularities, mosaicity, or any other deviation from regularity around ecotonal regions.
As we deviate from single species response to community response to edges, these patterns are linked to community assembly/dissassembly processes (see [40][18][36]). They may provide insights as to the spatial synchrony of community emergence along ecological/environmental gradients (i.e. if a large number of the constituent species of a given community present similar ecotonal response types – either due to similarities in their environmental requirements, or due to facilitation mechanisms – or if species of a community tend to present various response types to the ecotones, thus coming together as a community in a more Gleaonian sense, [12]. Also see the landmark paper of [38]).
This brings us to considerations on the response of particular species to the ecotone (the later having been defined at the community level).

Ecotonal species and species response to ecotones:

We presented, so far, a community approach to ecotones – with the purpose of filling a gap in methodological approaches relative to these particular landscape features. There was, nevertheless, valid reasons for the usual focus of ecotonal studies on singular species response, in particular, its relevance for conservation strategies. It is no surprise that the refinement of the spatial distribution of some species of interest, and indeed of their response to increasingly fragmented landscapes (another way to term the increase of ecotonal areas, [25]) has been central to ecotonal research in the context of the current ecological crisis.
The `EcotoneFinder` package makes investigation at the species level straightforward. whether it concerns species that were considered as a part of the community of interest (to investigate the role of each species in shaping community response), or "outsider" species (from different trophic levels, typically, for which the communities and ecotonal regions may represent a habitat through which to navigate).
Let's take, again, the example of two artificial communities (Figure 25), this time focusing on the behavior of some individual species.

```
# Two communities with a reasonable amount of variability in species responses:
Community3 <- SyntheticData(SpeciesNum = 30, CommunityNum = 3, SpCo = c(13,4,13),
                            Length = 500, Parameters = list(a = c(60,30,60),
                                                            b = c(0,250,500),
                                                            c = c(0.009,0.015,0.009)),
                            dev.c = .005,
                            dev.a = c(50,20,50), dev.b = 60,
                            pal = c("#008585", "green", "#C7522B"))

# And corresponding analyses:
Fuzzy2 <- EcotoneFinder(Community3[,2:ncol(Community3)], dist = Community3$Distance,
                        method = c("cmeans"), groups = 2, m.exp = 2,
                        standardize = "hellinger")
```

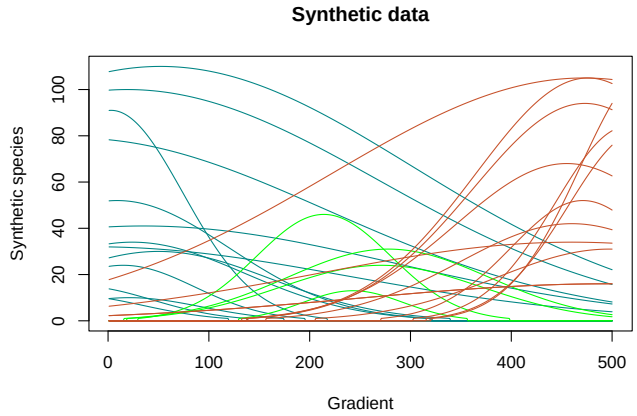

Figure 25: Artificail communities, with some variation in the species response around the ecotone. A 'third' community has been created to introduce some ecotonal species, in the sense that their abundance maxima are centred on the ecotone.

First, let's plot some individual species curves together with the fuzzy clusters (Figure 26). Compared to crisp or arbitrary representations of ecotones, the abundance curve of singular species can now be weighted directly against community clusters, thus refining the interpretation of their response types. This includes the comparison (i) of the rate of change along species abundance curves and (ii) of the overall rate of change of a community type.

```
### Fuzzy clusters and individual species:
# Base plot:
Plot <- ggEcotone(Fuzzy2, plot.data = FALSE,
                  method = c("cmeans"),
                  col = c("#D33F6A", "#E2E6BD"),
                  title = "fuzzy clusters",
                  xlab = "Gradient", ylab = "Membership grades") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5, face="bold"))

## Selecting interesting cases:
FirstCom <- Community3[,1:13]
# trespassing the ecotone:
Crossings <- which(apply(FirstCom[250:500,-1], MARGIN = 2, sum) > 50)
# Stoping short:
Negatives <- which(apply(FirstCom[250:500,-1], MARGIN = 2, sum) < 10)
```

```
# Adding species (dividing by their maximum abundances,
# to match the [0,1] range of the fuzzy clusters:
Plot1 <- Plot +
  geom_line(aes(x = Community3$Distance,
            y = Community3[,names(Crossings[1])]/max(Community3[,names(Crossings[1])])),
            col = "blue")
Plot1

Plot2 <- Plot +
  geom_line(aes(x = Community3$Distance,
            y = Community3[,names(Negatives[1])]/max(Community3[,names(Negatives[1])])),
            col = "darkgreen")
Plot2
```

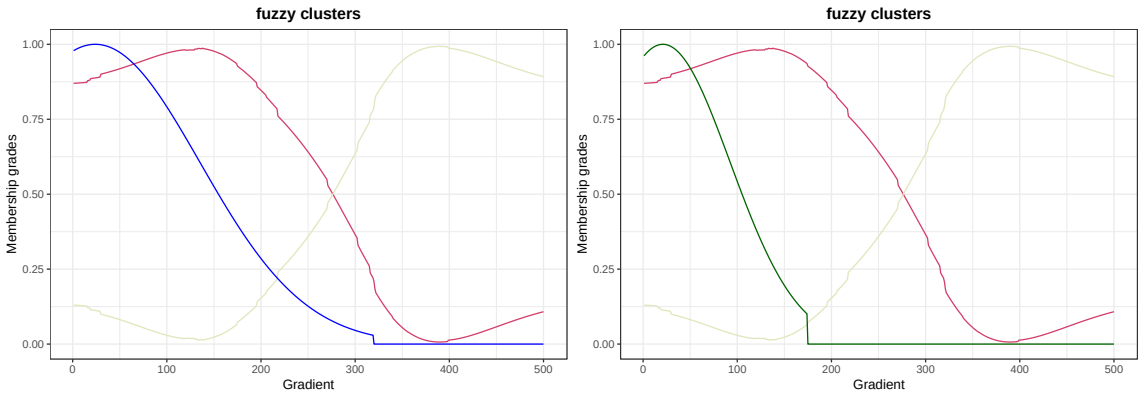

Figure 26: Example of two species respose type, one (on the left) crossing the community-wide demarcation of the ecotone, thus termed 'crossing' response, and another (on the right) falling short from it, thus termed 'negative' response.

In the artificial community example that we just presented, several species (four, in green) were specifically created to display ecotonal preferences, or – if this was the only known occurrence of these species – to be truly ecotonal. There is little evidence so far in the literature for strictly ecotonal species (i.e. whose distribution is restricted to ecotones), but ecotonal preferences have been reported widely (i.e. species that are either more frequent in ecotones, or locally restricted to particular ecotones, see [22]).
Using a varying number of fuzzy clusters in the `EcotoneFinder` function, in combination with the `ExtractCentroid` function, may help identify such species (Figure 27).

```
## Forcing a third 'ecotonal' community:
Fuzzy3 <- EcotoneFinder(Community3[,2:ncol(Community3)], dist = Community3$Distance,
                        method = c("cmeans"), groups = 3, m.exp = 2,
                        standardize = "hellinger")

## Comparing clusters:
PlotF2 <- ggEcotone(Fuzzy2, plot.data = FALSE,
                    method = c("cmeans"),
                    col = c("#D33F6A", "#E2E6BD"),
                    title = "2 fuzzy clusters",
                    xlab = "Gradient", ylab = "Membership grades") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5, face="bold"))

PlotF3 <- ggEcotone(Fuzzy3, plot.data = FALSE,
```

```
                  method = c("cmeans"),
                  col = c("#D33F6A", "green", "#E2E6BD"),
                  title = "3 fuzzy clusters",
                  xlab = "Gradient", ylab = "Membership grades") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5, face="bold"))

PlotF2
PlotF3

## Corresponding centroids:
CentroidsF2 <- ExtractCentroid(Fuzzy2, method = "cmeans", normalized = "cluster",
                               plot = FALSE, cex.x = 9, col=c("#A1A6C8", "#CA9CA4"))
CentroidsF2$GGplot +
  theme(legend.position = "none")

CentroidsF3 <- ExtractCentroid(Fuzzy3, method = "cmeans", normalized = "cluster",
                               plot = FALSE, cex.x = 9,
                               col=c("#A1A6C8", "green", "#CA9CA4"))
CentroidsF3$GGplot +
  theme(legend.position = "none")
```

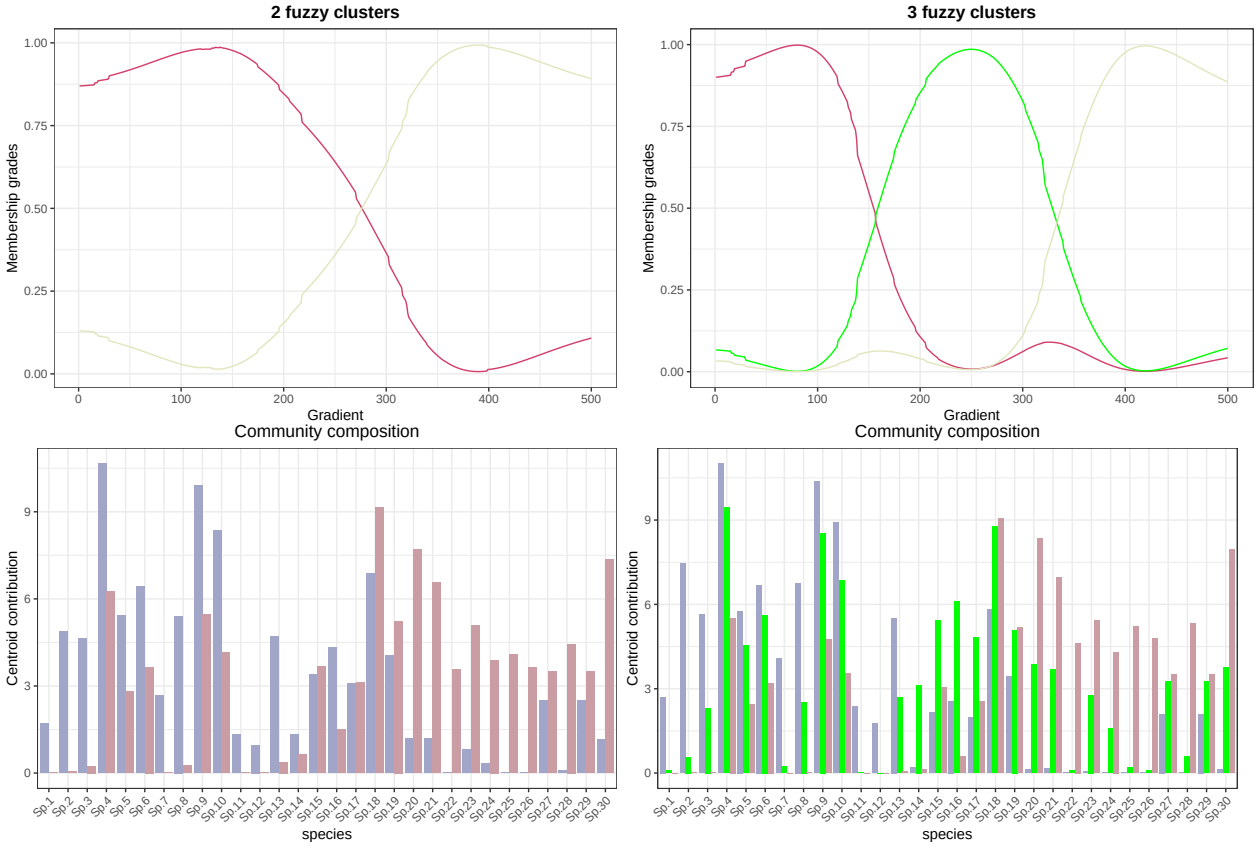

Figure 27: Community cluster shapes depending on the number of clusters. If a third cluster is drawn, it captures the particular ecotonal community structure, which is mostly driven by variations in individual species abundances along the gradient, but also by the potential presence of ecotonal species.

Let's focus on the case of the "green" species (sp. 14 to sp. 17), that is, the species we know are displaying

ecotonal preferences. The centroid values gives us little information on these species when only the two main community types are computed (left). When an additional "ecotonal" community is drawn, however, these four species clearly belong to it – i.e. they are the only species maximizing their membership values inside the middle community.

Data series and networks:

Note: The following section applies equally to time series and to site replication – i.e. space series – as they are similar in terms of data structures. The interpretation of those may remain, however, very different despite the many parallels that have been drawn between space and time in ecology (e.g. [21][16][19]).

We already saw in previous sections how to generate artificial space-time data series (see artificial data series) and community networks from fuzzy cluster centroids (see Community networks). The `EcotoneFinderSeries` and the `NetworkEcoSeries` functions of this package has been specifically implemented to directly performs some of these analyses directly on data series.
Both these functions recognize space-time series data if they are stored either in a list (named or not) containing the community dataframes corresponding to the different sampling events, or in a single community dataframe containing a factor column specifying the sampling events (e.g. a site column or a date column). When present, the names of the provided list (e.g. site names or dates), or the levels of the factor column, will be re-used to label the outputs of the analyses.
Let's start by reproducing some artificial data series (Fig. 28):

```
## Displacement matrix:
disp <- matrix(data=c(0,0,0,
                      0,35,-0.0007,
                      0,10,0), nrow = 3, ncol = 3)

### Series:
Series <- SyntheticDataSeries(CommunityPool = 60, CommunityNum = 3, Length = 500,
                              SeriesNum = 6, replacement = FALSE, SpCo = c(15,15,30),
                              Parameters = list(a = c(60,60,60),
                                                b = c(-50,-50,400),
                                                c = c(0.01, 0.01, 0.01)),
                              dev.a=c(30,50,30),
                              dev.b=c(40,10,30),
                              dev.c=c(0,.0001,0),
                              displacement = disp,
                              pal = c(rep("#008585",15), rep("#E6C186",15),
                                      rep("#C7522B",30)))
```

The corresponding analyses, are done using an extension of the `EcotoneFinder` function presented above (see Community detection along gradients), called `EcotoneFinderSeries` to keep it consistent with the `SyntheticDataSeries` function – being the extension of the `SyntheticData` function.
The `EcotoneFinderSeries` function is roughly analogous to the `EcotoneFinder` function – and accepts the same arguments – with the notable exception of an additional `series` argument. When a single dataframe is provided to the function, this argument will be used to break down the initial dataframe into its different sampling events. If a list of dataframes has been provided to the function, this argument is ignored and the names of the initial list will be kept for the output list.
To note, whereas the `dist` argument of the `EcotoneFinder` function accepted entire columns (such as `dataframe$distance`). The `dist` argument of the `EcotoneFinderSeries` function accept names of columns only (as characters).

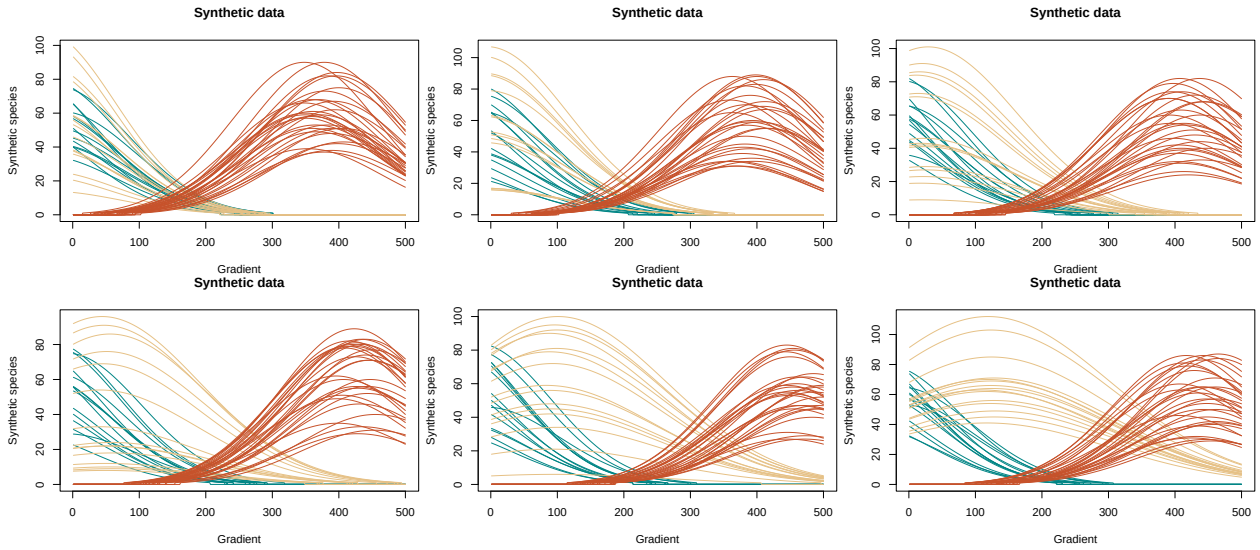

Figure 28: Time series with displacement: part of the first community (left) advances toward the right-end of the gradient by 35 gradient units per series, as if reacting to a moving environmental variable or gradient. The last community also withdraw further to the right, but more slowly (10 gradient units per series).

```
## All analyses (considering 3 groups):
EcoSeries <- EcotoneFinderSeries(data = Series,
                                 dist = "Distance",
                                 method = c("cmeans", "dca", "diversity"),
                                 diversity = c("richness", "expShannon"),
                                 series = "Time", groups = 3,
                                 standardize = "hellinger", na.rm = TRUE)

## Plotting only requires a loop:
Plot <- list()
for (i in names(EcoSeries)) {
  Plot[[i]] <- ggEcotone(EcoSeries[[i]], plot.data = FALSE,
                         method = c("cmeans", "dca"),
                         col = c("#D33F6A", "#E99A2C", "#E2E6BD"),
                         facet = c("cmeans", "dca"),
                         title = paste(i), xlab = "Gradient",
                         ylab = "Membership grades and first DCA axis") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5, face="bold"))
}

Plot[[1]]
Plot[[2]]
Plot[[3]]
Plot[[4]]
Plot[[5]]
Plot[[6]]
```

To provide a quick commentary on the outputs (Figure 29) – as part of the initially condensed species of the left-hand community shift their distributions towards the other end of the gradient, the middle cluster (arguably ecotonal, but we will come back to that shortly) accordingly shift towards the right-end of the gradient, or simply widen as the corresponding region also spread over the gradient.

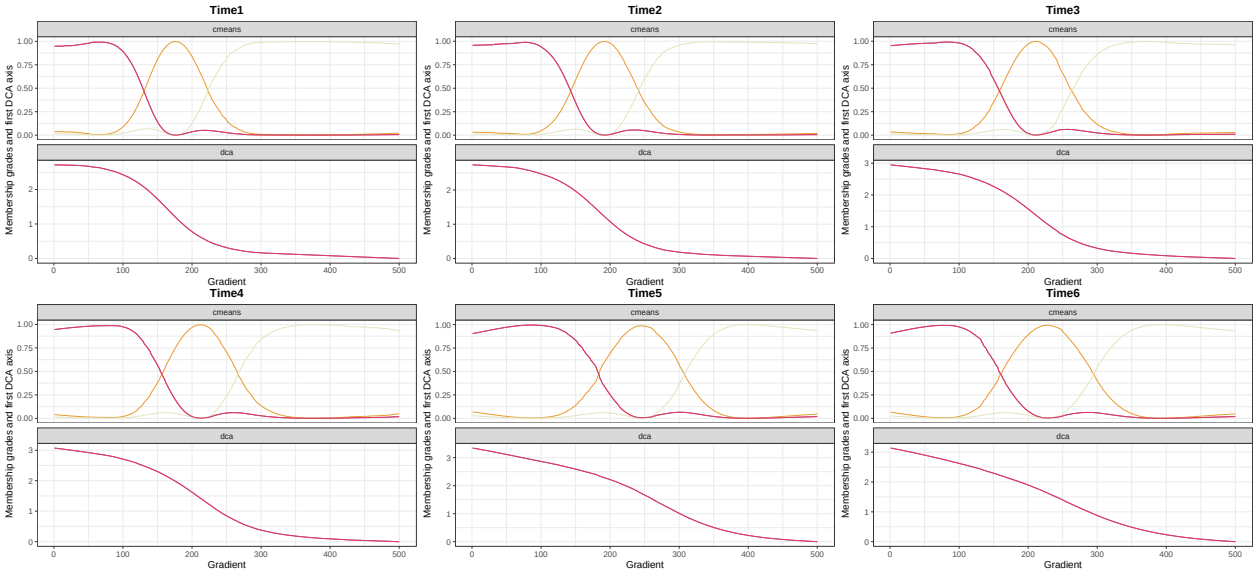

Figure 29: Evolution of the fuzzy clusters and of the first axis of the DCA along the gradient for each sampling event in the artificial series.

The first axis of the DCA becomes less steep as the series progress (the overall community transition is indeed more gradual at `Time 6` than at `Time 1`). The range of the axis also slightly increases, suggesting an overall increase of beta diversity over the gradient.
We can also have a look at diversity patterns and evolution over the series (Figure 30).

```
## Ploting only requires a loop:
# colour gradient:
Colours <- c("grey60", "cadetblue", "gold")
Plot <- list()
for (i in names(EcoSeries)) {
  Plot[[i]] <- ggEcotone(EcoSeries[[i]], plot.data = TRUE,
                         method = c("cmeans", "diversity"),
                         col = Colours,
                         facet = c("data", "cmeans", "diversity"),
                         title = paste(i), xlab = "Gradient",
                         ylab = "Species richness and shannon entropy (exp.)") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5, face="bold"))
}

Plot[[1]]
Plot[[2]]
Plot[[3]]
Plot[[4]]
Plot[[5]]
Plot[[6]]
```

Two aspects may be worth noting about these graphical outputs: first, the gradual disconnect between the species richness curve and the Shannon entropy curve, and second, the gradual disconnect between the ecotonal region/cluster and the peaks of biodiversity. This illustrates how quickly one can deviates from the initial assumption that ecotones would be characterized by peaks of biodiversity. This assumption relied primarily on an organismal (or holistic) view of ecological communities (the paper of Tansley, 1935 [38]

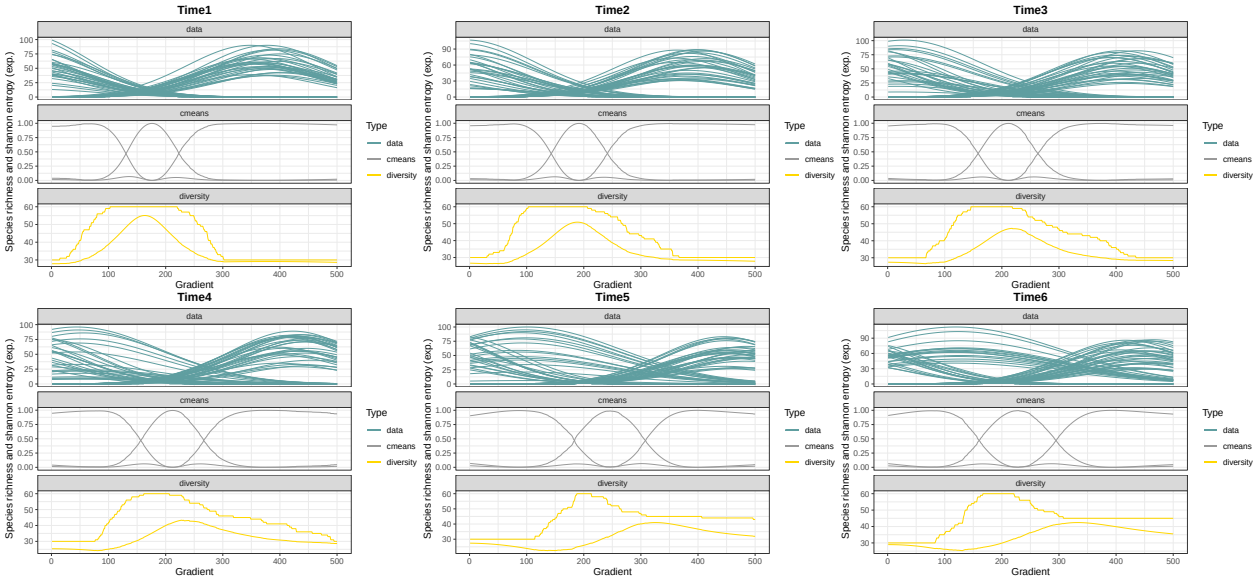

Figure 30: Evolution of the species richness and the exponential of the Shannon entropy along the gradient for each of hte sampling event of the series. The generated species distribution data and the evolution of the fuzzy clusters are given again, for comparison purposes. This example easily illustrates how diversity patterns and statistical community structures can be disjoinct, even with simple ecological data.

already discusses this point very well).
Although the existence of 'ecosystems' as integrated biological/environmental entities is now generally accepted, there is still a divide, at least in practice, between an organismal conception of ecological communities, and a gradual conception of ecological communities (usually taking the form of a choice in representation strategies, such as clustering technics in the one hand, or ordination methods – nMDS, DCA, etc. – in the other). The difference is presented here by super-imposing the fuzzy clusters and the first axis of the DCA – the later identifying a single, albeit expending, gradient instead of a displacement of community types.
Fuzzy clusters are an obvious attempt at reconciling both organismal and gradient approaches, but all representations are correct, give complementary information, and should thus be considered in combination.

Networks across data series:

Now that the basic patterns in the provided example of data series have been covered, let's explore what additional information could be extracted from it.
The `NetworkEcoSeries` function (again, an extension of the previously presented `NetworkEco` function) is designed to draw the links – not only between communities extracted from a single gradient – but among communities along series of gradients. This typically allow for the disentangling of spatial and temporal patterns in data series (i.e. are the community types stable through space or time – indicating a certain consistency along the series, or do they group more according to each sampling event – denoting high spatial or temporal turnover?).
Networks of association can help answering the question.
Although there are a number of possible options regarding the identities of the different nodes in the networks – and regarding the computation strategies for the edges that link them – the overall topology remains equivalent. Let's start by building a few of them, with the aim of answering the previous question (Fig. 31).

```
NetworkSeries <- NetworkEcoSeries(EcoSeries,
                                  method = "cmeans", plot.type = "network",
                                  plot = "community", dist = "raw",
                                  network.group = "cluster",
                                  dist.method = "inner_product",
```

```
                                    no.plot = FALSE, layout = "spring",
                                    shape = "ellipse",
                                    palette = "colorblind")
title(main = "A: By common species counts \n coloured by clusters",
      adj = .9, cex.main = .7, outer = FALSE)

NetworkSeries <- NetworkEcoSeries(EcoSeries,
                                    method = "cmeans", plot.type = "network",
                                    plot = "community", dist = "raw",
                                    network.group = "site",
                                    dist.method = "inner_product",
                                    no.plot = FALSE, layout = "spring",
                                    shape = "ellipse",
                                    palette = "colorblind")
title(main = "B: By common species counts \n coloured by sampling events",
      adj = .9, cex.main = .7, outer = FALSE)
```

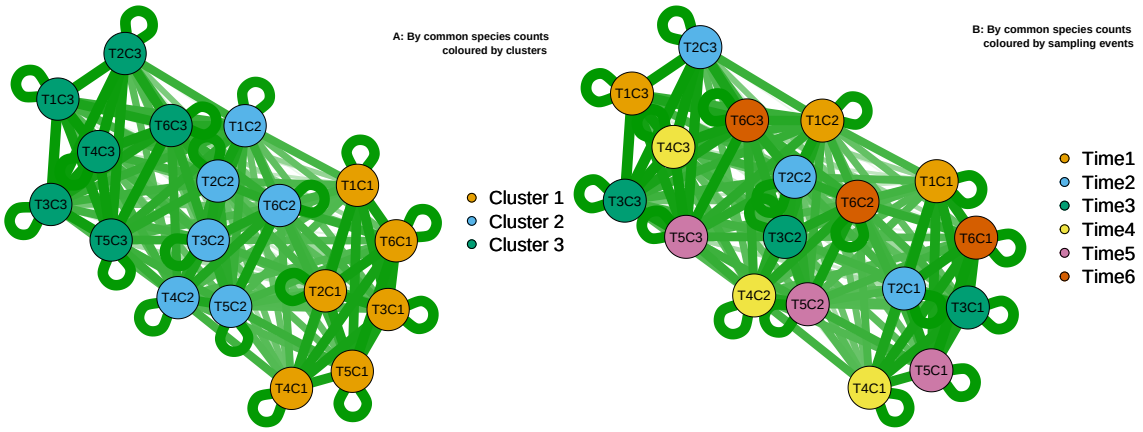

Figure 31: Networks of associations between main community types (based on fuzzy cluster centroid compositions). The centroid species scores are unmodified (raw) and the topology of the network – the relative position of the nodes and length of the edges – relies on the inner product similarity measure. The colouring of the nodes either follows (A) the positions of the communities along the gradient or (B) the sampling event, here chosen to be 'Times'

These two identical networks only differ by the coloring of their nodes – controlled by the `network.group` argument. Their comparison clearly highlights that the consistency of the cluster species composition along this artificial time series.

Before continuing on the different types of networks that can be generated with this function, let's review the many arguments it contains – most of them equivalent to the ones of the `NetworkEco` function.

The `plot.type` – currently set to 'network' – indicates the type of graphical output. It can also be set to 'heatmap' or 'corrplot' (similarly to the `NetworkEco` function).

The `method` – in this case "cmeans" – specifies the fuzzy cluster algorithm to pick in the `EcoSeries` object (accordng to the choice made in the `EcotoneFinderSeires` function). The cluster centroids are then extracted, and will be used to build the network.

From there, a number of arguments control how to use and manipulate the cluster centroids. The `plot` argument ('community' or 'species') specify the identity of the nodes – which can either be the recorded species in the data, or the fuzzy clusters, i.e. the community types. The `network.group` argument ('site' or 'cluster') does not modify the network, but controls the coloring of the nodes. It corresponds to the comparison given above (Fig. 31) between a grouping per community types or per sampling events.

The `dist` argument ('raw', 'relative' or 'count', as in the `NetworkEco` function) informs a first-level transformation of the fuzzy cluster centroid matrix, which can either be kept untouched ('raw'), be standardized to a margin total of 1, following the `method = 'total'` of the `decostand` function in the

`vegan` package ('relative' – per species or clusters, depending on the `plot` argument), or – once standardized – be transformed to a count of common species per community clusters, according to a given `threshold` (between 0 and 1). The rationale behind this threshold is that a species scoring higher than 0.5 in a community cluster may be considered representative, or iconic, of this community cluster. The number of shared representative species between sampling events across a series may be a reasonably good indicator of temporal turn-over in species compositions. *If `dist = 'count'`, the `plot` argument is thus automatically set to 'community'*
Finally, the `dist.method` argument (to be chosen among the distance methods of the *phillentropy* package) controls the distance calculations to be used on the fuzzy clusters centroids matrix in order to draw the networks. Since a count of shared species between clusters is in itself a distance calculation, this argument is ignored if `dist = 'count'`.
The remaining arguments are additional specification that are recycled internally by the `qgraph` function of the `qgraph` package (see [7]).

Let's see some other network examples based the same data.
So far in this package, the distance calculations for cluster centroid matrices has been chosen among the `philentropy` package options (it offers a wide array of similarity/dissimilarity measures). The effect of this choice on the topologies of the networks is obvious (Fig. 32). For the purposes of comparing fuzzy cluster centroids, similarity measures (like 'inner_product' or 'intersection') are better suited than dissimilarity calculations (such as 'jaccard'), as we want similar clusters (i.e. nodes) to be positioned closer to one another.

```
NetworkSeries <- NetworkEcoSeries(EcoSeries,
                                  method = "cmeans", plot.type = "network",
                                  plot = "community", dist = "raw",
                                  network.group = "cluster",
                                  dist.method = "intersection",
                                  no.plot = FALSE, layout = "spring",
                                  shape = "ellipse",
                                  palette = "colorblind")
title(main = "A: By common species counts \n coloured by clusters",
      adj = .9, cex.main = .7, outer = FALSE)

NetworkSeries <- NetworkEcoSeries(EcoSeries,
                                  method = "cmeans", plot.type = "network",
                                  plot = "community", dist = "raw",
                                  network.group = "cluster",
                                  dist.method = "jaccard",
                                  no.plot = FALSE, layout = "spring",
                                  shape = "ellipse",
                                  palette = "colorblind")
title(main = "B: By common species counts \n coloured by sampling events",
      adj = .9, cex.main = .7, outer = FALSE)
```

The type of transformation applied to the cluster centroids ('raw', 'relative' or 'count') also impacts the network outputs. The transformation of the cluster centroids to a count of shared representative species requires some additional steps, and will thus be treated later. For now, we can compare the two other types (Fig. 33).

```
NetworkSeries <- NetworkEcoSeries(EcoSeries,
                                  method = "cmeans", plot.type = "network",
                                  plot = "community", dist = "raw",
                                  network.group = "cluster",
                                  dist.method = "inner_product",
                                  no.plot = FALSE, layout = "spring",
```

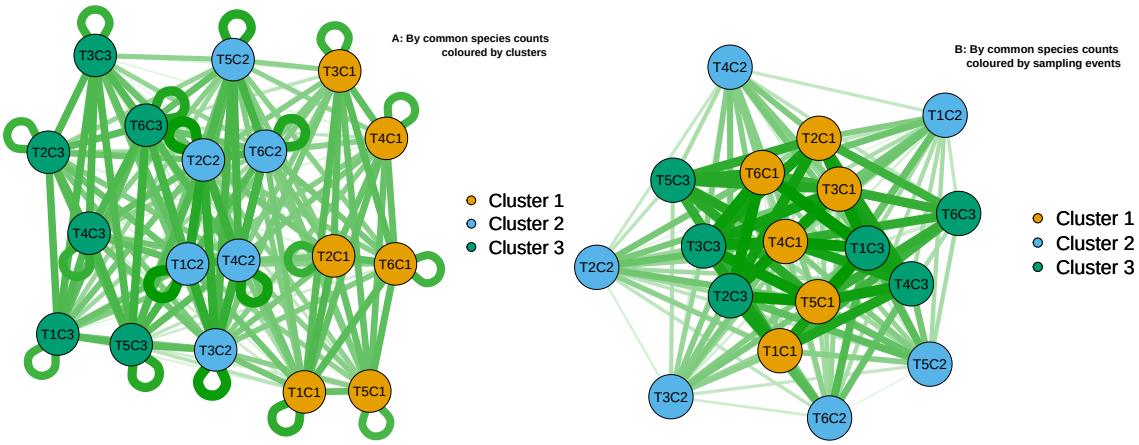

Figure 32: Networks of associations between main community types (based on fuzzy cluster centroid compositions). The centroid species scores are unmodified (raw) and the topology of the network – the relative position of the nodes and length of the edges – either relies on (A) the inner product similarity measure or (B) the jaccard dissimilarity measure. Similarity calculations are more adapted to network of this kind, as they put similar nodes closer to one another.

```
                                  shape = "ellipse",
                                  palette = "colorblind")
title(main = "A: By common species counts \n coloured by clusters",
      adj = .9, cex.main = .7, outer = FALSE)

NetworkSeries <- NetworkEcoSeries(EcoSeries,
                                  method = "cmeans", plot.type = "network",
                                  plot = "community", dist = "relative",
                                  network.group = "cluster",
                                  dist.method = "inner_product",
                                  no.plot = FALSE, layout = "spring",
                                  shape = "ellipse",
                                  palette = "colorblind")
title(main = "B: By common species counts \n coloured by sampling events",
      adj = .9, cex.main = .7, outer = FALSE)
```

The nodes of the networks can also correspond to relations between species in the data (Fig. 34). In this case, it is better to specify the coloring of the nodes manually (with the internal `colour` argument of the `qgraph` function). In these networks, the colors that have been used for the artificial communities are re-used – highlighting the relations between species belonging to the same community types.

```
NetworkSeries <- NetworkEcoSeries(EcoSeries, threshold = .5,
                                  method = "cmeans", plot.type = "network",
                                  plot = "species", dist = "raw",
                                  network.group = NULL,
                                  dist.method = "inner_product",
                                  no.plot = FALSE, layout = "spring",
                                  shape = "ellipse",
                                  color = c(rep("#008585",15), rep("#E6C186",15),
                                      rep("#C7522B",30)))
title(main = "A: By common species counts \n coloured according to
      the artificial data structure",
      adj = .9, cex.main = .7, outer = FALSE)
```

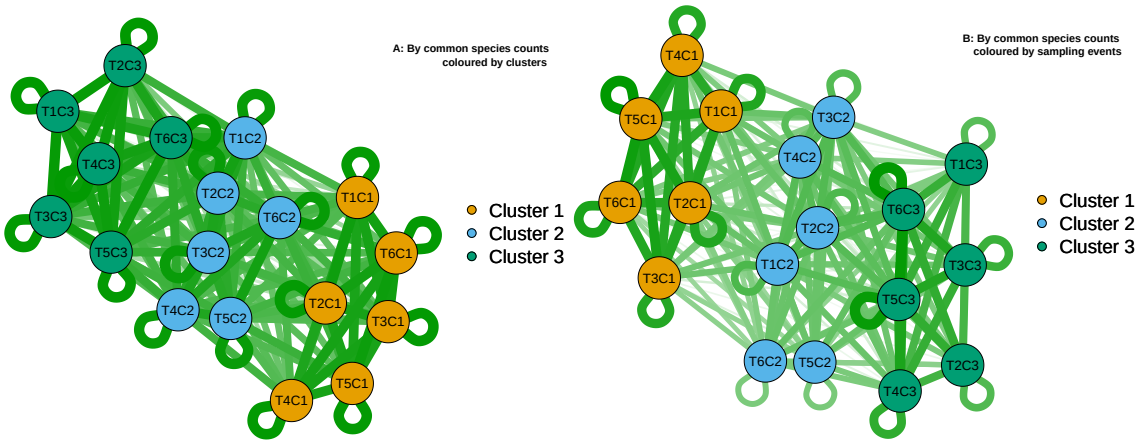

Figure 33: Networks of associations between main community types (based on fuzzy cluster centroid compositions). The centroid species scores are either (A) unmodified ('raw') or (B) standardized (divided by margin total). The corresponding network topology are mostly equivalent, but stadardized centroid values weaken the links between the intermediate clusters in the series (in blue), as their compositions are less stable over the artificial data series – a difference overlooked by raw cluster centroid inputs.

```
NetworkSeries <- NetworkEcoSeries(EcoSeries, threshold = .5,
                                  method = "cmeans", plot.type = "network",
                                  plot = "species", dist = "relative",
                                  network.group = NULL,
                                  dist.method = "inner_product",
                                  no.plot = FALSE, layout = "spring",
                                  shape = "ellipse",
                                  color = c(rep("#008585",15), rep("#E6C186",15),
                                      rep("#C7522B",30)))
title(main = "A: By common species counts \n coloured according to
      the artificial data structure",
      adj = .9, cex.main = .7, outer = FALSE)
```

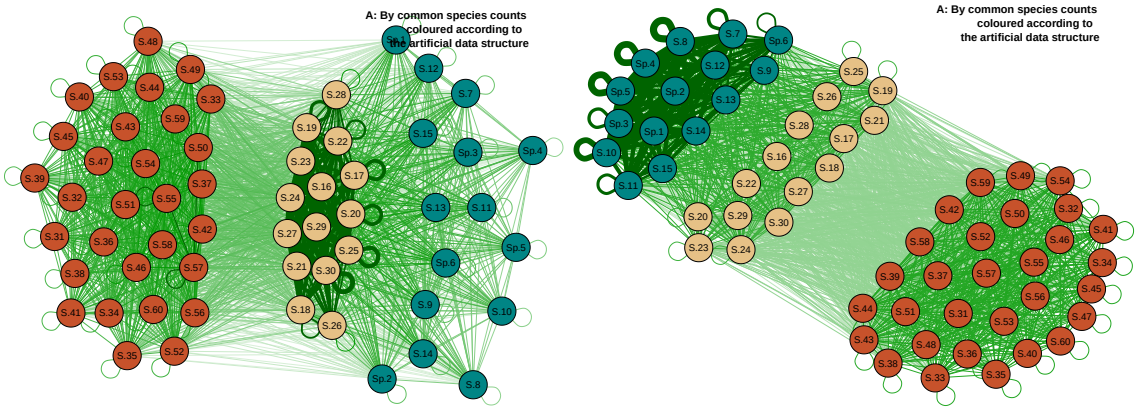

Figure 34: Networks of associations between species in the artificial data series (based on fuzzy cluster centroid compositions). The centroid species scores are either (A) unmodified ('raw') or (B) standardized (divided by margin total). With such simple data structure, Tthe main community types may be identified back due to the pooling of groups of species together.

Statistical groups in networks:

So far we have been using pre-defined groups to pool nodes together (sampling events, position of communities along gradients, or any factorial aspects of the initial data). Network topology is, however, often the result of compromises between equivalent or near equivalent topologies, and thus only represent one possibility – among others – as to the relative position of the nodes and length of the edges. Visual interpretation of groups in the networks may thus be misleading, and insufficient to reach solid conclusions.
There exist a number of algorithms for community detection in networks (see [42]), a community in a network being defined as group of nodes sharing more edges with one another than with any other nodes in the network. Both due to easiness of implementation and overall good performance, the `EcotoneFinder` package includes the spinglass algorithm of the `igraph` package in the `NetworkCommunity` function. This function is specifically designed to work alongside with the networks created by the `NetworkEcoSeries` function. The spinglass algorithm was already presented with the `DistEco` function above, (see data structure). It is not in the scope of this package vignette to explain the functioning of this algorithm in too much details. Suffice to say that – drawing parallels with statistical physics, and the spins of particles – it aims at discovering the number of spins states in the system, with each spin states corresponding to a community of nodes in the network (see [27], for more details). Like many other clustering approaches, this algorithm selects a random node as a starting point. It will thus often lead to different results every time it is being run. The `run` argument of the `NetworkCommunity` function lets the user define how many times the algorithm should be run (selecting a random seed for each run) before returning the outputs (both as rounded integers, or as overall means, leaving some nodes with in between groups averages – in accordance with the general philosophy of fuzzy clusters and ecotones).
It is important to remind, however, that community detection algorithms will find communities even in random networks [27]. In such cases, equally sized communities tend to be generated, i.e. the total number of nodes is divided equally between the N number of detected communities. When dealing with complex ecological data, it is thus recommended to remain suspicious of equally sized network community outputs.
The `NetworkCommunity` function will return a vector of the same length than the number of nodes in the network, and can thus be directly recycled by the `group` argument of the `NetworkEcoSeries` function (Figure 35).

```
# Using a previous community network (not plotting it this time):
NetworkSeries <- NetworkEcoSeries(EcoSeries,
                                  method = "cmeans", plot.type = "network",
                                  plot = "community", dist = "relative",
                                  network.group = "cluster",
                                  dist.method = "inner_product",
                                  no.plot = TRUE, layout = "spring",
                                  shape = "ellipse",
                                  palette = "colorblind")

# Making use of the previous network:
CommunityID <- NetworkCommunity(NetworkSeries, run = 10)

NetworkSeriesSpin <- NetworkEcoSeries(EcoSeries,
                            method = "cmeans", plot.type = "network",
                            plot = "community", dist = "relative",
                            network.group = as.factor(CommunityID$Memberships$RoundedMean),
                            dist.method = "inner_product",
                            no.plot = FALSE, layout = "spring",
                            shape = "ellipse")
title(main = "A: By common species counts \n coloured according to
      \n spinglass algoritm outputs (rounded)",
      adj = .9, cex.main = .7, outer = FALSE)
```

```
NetworkSeriesSpin <- NetworkEcoSeries(EcoSeries, threshold = .5,
                          method = "cmeans", plot.type = "network",
                          plot = "community", dist = "relative",
                          network.group = as.factor(CommunityID$Memberships$Mean),
                          dist.method = "inner_product",
                          no.plot = FALSE, layout = "spring",
                          shape = "ellipse")
title(main = "A: By common species counts \n coloured according to
      \n spinglass algorithm outputs (not rounded)",
      adj = .9, cex.main = .7, outer = FALSE)
```

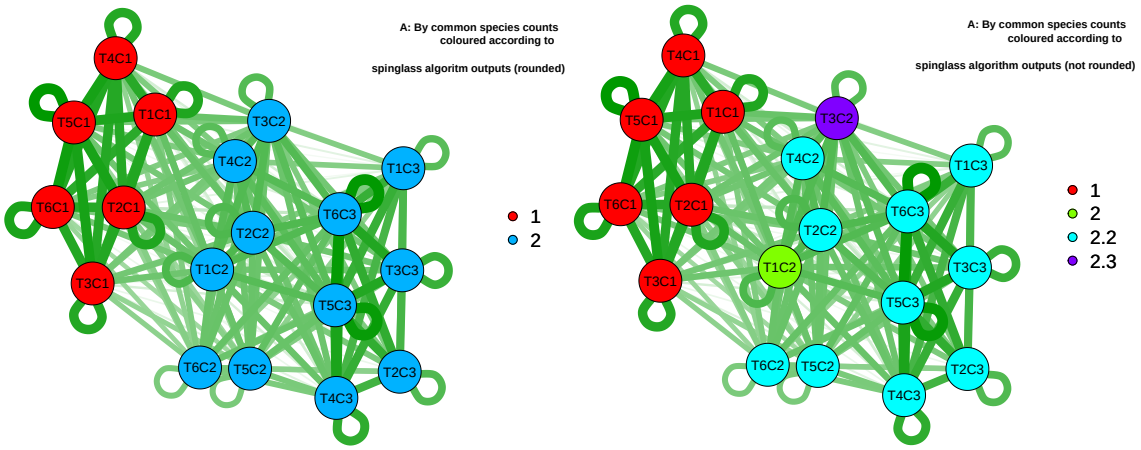

Figure 35: Networks of associations between species in the artificial data series (based on fuzzy cluster centroid compositions). The centroid species scores are either (A) unmodified ('raw') or (B) standardized (divided by margin total). With such simple data structure, Tthe main community types may be identified back due to the pooling of groups of species together.

Concluding remarks:

The aim of this package was to propose a community-level framework for the study of ecotones (Fig.23), and to provide the associated workflow (Fig.1), in the form of a sequence of functions.
As such, the `EcotoneFinder` package has been designed to be relatively self-contained. It notably includes functions for the generation of artificial datasets to facilitate the testing of hypotheses, function to explore the structure of ecological data to support decisions regarding further clustering analyses, and functions to easily produce relevant graphical outputs.
The core of the analyses, however, relies on a combination (a comparison) of ecological gradients analyses (e.g. DCA), and ecosystem classification methods (fuzzy clusters). Provided adequate sampling intensity, this method combination proves helpful and efficient at pinpointing the location of ecotones and ecosystems in complex environments (see [2]). Expanding on the single-species response frameworks that have been discussed in [9][29][26], key metrics such as the Abruptness of Edge Influence, the Magnitude of Edge Influence, and the Depth of Edge influence were adapted to continuous metrics summarizing the variability of entire community types. Using the same initial analyses, functions are provided to explore the composition and relatedness of ecosystems and ecotones – to go beyond purely observational studies.

The present workflow – at the limited scale of ecological gradients – proposes a solution to one of the persistent gaps in edge ecology, to paraphrase [29]. We believe it provides a comprehensive starting point for studies on ecotones along ecological gradients.
The set of analyses regrouped in this package are, however, by no means exhaustive. There exists a number of other approaches that have been successfully used in ecotonal research, and many more will undoubtedly follow.
It is customary to finish by offering future research directions. A few shall be presented here:

. First, the present framework could be – and probably should be – extended to landscape as whole, instead of isolated gradients. Satellite imagery could typically provide the necessary data, and fuzzy clustering algorithms are already implemented on GIS software, such as QGIS. The extension of the proposed ecotonal metrics (AEI, DEI, MEI) to surfaces instead of one-dimensional curves still remains to be done.

. Decisions regarding the number of fuzzy clusters, in the one hand, and on the exact start point of ecotones (for instance, for DEI calculations), on the other hand, could be improved. Any steps towards the standardization of these two aspects will greatly facilitate future comparative studies on ecotones across scale and ecosystem types.

. The exact meaning – in ecological sense – of the shape of the ecotones is still mostly unresolved. Patterns at the edges of communities could indicate fragmentation processes, highlight the effect of changing environmental conditions on communities, serve as indicator of the future evolution of ecosystem types, and generally provide insights on the rules leading to the assembly/disassembly of ecological communities. Among possible patterns, one could think of irregularities (e.g. wavelets) along edges, asymmetry between successive edges, mosaicity, and the coincidence or mismatches of community edges with particular environmental conditions.

. The application of this framework to practical conservation strategies and restoration efforts still demand much work, and – as a starting point – the integration of ecotones into mainstream ecological investigations. We hope the present document made a step in that direction.

Although using different methods, recent work have already been conducted along these lines (e.g. [35][32][13][14]), with promising results. Readers are refereed to their works for more information.

References

[1] Mike Austin. "Species distribution models and ecological theory: A critical assessment and some possible new approaches". In: *Ecological Modelling* 200.1-2 (Jan. 2007), pp. 1–19.

[2] Antoine Bagnaro et al. "Reducing the arbitrary: fuzzy detection of microbial ecotones and ecosystems – focus on the pelagic environment". In: *Environmental Microbiome* 15:16 (Aug. 2020), pp. 1–14.

[3] Vinko Bandelj et al. "Fuzziness and Heterogeneity of Benthic Metacommunities in a Complex Transitional System". In: *PLoS ONE* 7.12 (Dec. 2012), e52395.

[4] James C Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Ed. by Morton Nadler. Plenum Press, New York, May 1981.

[5] G Brownstein et al. "Functional traits shed new light on the nature of ecotones: a study across a bog-to-forest sequence". In: *Community ecology …* 14.1 (June 2013), pp. 31–40.

[6] Miquel De Cáceres, Xavier Font, and Francesc Oliva. "The management of vegetation classifications with fuzzy clustering". In: *Journal of Vegetation Science* 21.6 (Aug. 2010), pp. 1138–1151.

[7] Sacha Epskamp et al. "qgraph: Network Visualizations of Relationships in Psychometric Data". In: *Journal of Statistical Software* 48.4 (May 2012), pp. 1–18.

[8] László Erdős et al. "The Moving Split Window (MSW) analysis in vegetation science – an overview". In: *Applied Ecology and Environmental Research* 12.3 (Aug. 2014), pp. 787–805.

[9] Robert M Ewers and Raphael K Didham. "Continuous response functions for quantifying the strength of edge effects". In: *Journal of Applied Ecology* 43.3 (June 2006), pp. 527–536.

[10] Hannes Feilhauer and Sebastian Schmidtlein. "Mapping Continuous Fields of Forest Alpha and Beta Diversity". In: *Applied Vegetation Science* 12 (Mar. 2009), pp. 429–439.

[11] Hugh G Jr Gauch. *Multivariate Analysis In Community Ecology*. Ed. by E Beck, H J B Birks, and E F Connor. Cambridge University Press, Cambridge, Aug. 1982.

[12] H A Gleason. "The Individualistic Concept of the Plant Association". In: *The American Midland Naturalist* 21 (May 1939), pp. 92–110.

[13] Wenkai Guo and Gareth Rees. "Correlation between the dynamics and spatial configuration of the circumarctic latitudinal forest- tundra ecotone". In: *International Journal of Remote Sensing* 42.4 (Dec. 2020), pp. 1250–1274.

[14] Wenkai Guo, Gareth Rees, and Annika Hofgaard. "Delineation of the forest-tundra ecotone using texture-based classification of satellite imagery". In: *International Journal of Remote Sensing* 41.16 (May 2020), pp. 6384–6408.

[15] M O Hill and H G Gauch JJr. "Detrended correspondence analysis: An improved ordination technique". In: *Vegetatio* 42 (June 1980), pp. 47–58.

[16] Sven E Jørgensen, Søren Nors Nielsen, and Brian D Fath. "Recent progress in systems ecology". In: *Ecological Modelling* (Dec. 2016), pp. 1–7.

[17] Lou Jost. "Partitioning Diversity Into Independent Alpha And Beta Components". In: *Ecology* 88 (Oct. 2007), pp. 2427–2439.

[18] Nathan J B Kraft et al. "Community assembly, coexistence and the environmental filtering metaphor". In: *Functional Ecology* 29.5 (Oct. 2015), pp. 592–599.

[19] Pierre Legendre. "A temporal beta-diversity index to identify sites that have changed in exceptional ways in space–time surveys". In: *Ecology and Evolution* 9.6 (Feb. 2019), pp. 3500–3514.

[20] Pierre Legendre and Miquel De Cáceres. "Beta diversity as the variance of community data: dissimilarity coefficients and partitioning". In: *Ecology Letters* 16.8 (July 2013), pp. 951–963.

[21] S A Levin. "The problem of pattern and scale in ecology". In: *Ecology* 73.6 (1992), pp. 1943–1967.

[22] Kelvin M Lloyd et al. "Evidence on ecotone concepts from switch, environmental and anthropogenic ecotones". In: *Journal of Vegetation Science* 11 (Dec. 2000), pp. 903–910.

[23] Eugene P Odum. *Fundamentals of Ecology*. W. B. Saunders Company, Philadelphia, Apr. 1953.

[24] J Oksanen et al. "vegan: Community Ecology Package". In: *Community ecology …* (2013).

[25] Debra P C Peters et al. "Integrating Patch and Boundary Dynamics to Understand and Predict Biotic Transitions at Multiple Scales". In: *Landscape Ecology* 21.1 (Jan. 2006), pp. 19–33.

[26] M Pfeifer et al. "Creation of forest edges has a global impact on forest vertebrates". In: *Nature* 551.7679 (Nov. 2017), pp. 187–191.

[27] Jörg Reichardt and Stefan Bornholdt. “Statistical mechanics of community detection”. In: *Physical Review E* 74.1 (July 2006), p. 291.
[28] Leslie Ries and Thomas D Sisk. “A predictive model of edge effects ”. In: *Ecology* 85.11 (Mar. 2004), pp. 2917–2926.
[29] Leslie Ries et al. “Closing Persistent Gaps in Knowledge About Edge Ecology”. In: *Curr Landscape Ecol Rep* (Mar. 2017), pp. 1–12.
[30] Leslie Ries et al. “Ecological Responses to Habitat Edges: Mechanisms, Models, and Variability Explained”. In: *Annual Review of Ecology, Evolution, and Systematics* 35.1 (Dec. 2004), pp. 491–522.
[31] David W Roberts. “Comparison of multidimensional fuzzy set ordination with CCA and DB-RDA”. In: *Ecology* 90 (Aug. 2009), pp. 2622–2634.
[32] G M Wardle E J Leitch F A McInerney A J Lowe S Caddy-Retalic. “Vegetation change along a Mediterranean to arid zone bioclimatic gradient reveals scale-dependent ecotone patterning”. In: *Australian Journal of Botany* (Dec. 2020), pp. 1–13.
[33] M Schonlau. “The clustergram: A graph for visualizing hierarchical and nonhierarchical cluster analyses”. In: *The Stata Journal* 2 (May 2002), pp. 391–402.
[34] Andreas H Schweiger et al. “Optimizing sampling approaches along ecological gradients”. In: *Methods in Ecology and Evolution* 7.4 (Nov. 2016), pp. 463–471.
[35] Monika E Shea et al. “Identifying ecotone location using the co-occurrence property”. In: *Journal of Vegetation Science* (Sept. 2020), pp. 1–13.
[36] Eric R Sokol, Bryan L Brown, and J E Barrett. “A simulation-based approach to understand how metacommunity characteristics influence emergent biodiversity patterns”. In: *Oikos* 126.5 (Nov. 2017), pp. 723–737.
[37] David L Strayer et al. “A Classification of Ecological Boundaries”. In: *BioScience* 53 (Aug. 2003), pp. 723–729.
[38] A G Tansley. “The Use and Abuse of Vegetational Concepts and Terms”. In: *Ecology* 16 (July 1935), pp. 284–307.
[39] Susan Walker et al. “Properties of ecotones: Evidence from five ecotones objectively determined from a coastal vegetation gradient”. In: *Journal of Vegetation Science* 14 (Mar. 2003), pp. 579–590.
[40] Evan Weiher and Paul Keddy, eds. *Ecological assembly rules: perspectives, advances, retreats*. Cambridge University Press, UK, July 1999.
[41] R H Whittaker. “Gradient Analysis Of Vegetation”. In: *Biological Reviews* 49 (Nov. 1967), pp. 207–264.
[42] Zhao Yang, René Algesheimer, and Claudio J Tessone. “A Comparative Analysis ofCommunity Detection Algorithmson Artificial Networks”. In: *Scientific Reports* (June 2016), pp. 1–18.
[43] L A Zadeh. “Fuzzy Sets”. In: *Information and Control* 8 (Nov. 1965), pp. 338–353.