Demo hypervolume overlap

Quentin D. Read

10/21/2020

This document shows some work I've done to get a preliminary working version of multivariate community overlap statistic. It is based on the R package `hypervolume` created by Ben Blonder and colleagues. Previously, our `Ostats` package did not support multivariate overlap; instead, it calculated the community median pairwise overlap statistic separately for each trait. I created a branch called `hypervolumetest` on our repo where I have developed an "alpha" version of a multivariate overlap function.

I mainly modified the `pairwise_overlap()` function to determine whether its trait input is univariate or multivariate. If univariate, it returns the single-variable overlap statistic as before, using `density()` to get the smoothed density functions for each species and finding their overlap. But if the input is multivariate, it will use functions from the `hypervolume` package to get smoothed hypervolumes for each species and find their overlaps. Below is a snippet of the important part of the code where those computations are done. As you can see it really just runs a few basic functions from the `hypervolume` package.

```
# Convert each of the input matrices to hypervolume.
hv_a <- do.call(hypervolume::hypervolume, args = c(list(data = a), density_args))
hv_b <- do.call(hypervolume::hypervolume, args = c(list(data = b), density_args))

# Calculate hypervolume set operations
# This uses default arguments except for verbose.
hv_set_ab <- hypervolume::hypervolume_set(hv_a, hv_b,
                                          num.points.max = NULL,
                                          verbose = density_args[['verbose']],
                                          check.memory = FALSE,
                                          distance.factor = 1)

# Calculate hypervolume overlap statistic
hv_overlap_ab <- hypervolume::hypervolume_overlap_statistics(hv_set_ab)
```

After that, the output of `pairwise_overlap` is just a single value between 0 and 1 just the same as for the univariate case. So the rest of the calculation of community-weighted stuff is the same as before! I had to slightly modify `community_overlap_merged()` to take different types of input. For now I didn't modify `Ostats()` to take different input; instead, I just made a different function called `Ostats_multivariate()`.

Here I run the `Ostats_multivariate()` function on the classic `iris` example dataset provided with base R.

Load package and data

The version of the `Ostats` package loaded here is the one in the development branch `hypervolumetest`.

We are using the four iris morphological traits for the three species, considering the entire iris dataset as a single "plot" or community.

```
library(Ostats)
```

```
iris_traits <- iris[,1:4]
iris_sp <- iris[,5]
iris_plots <- rep('plot1', nrow(iris))
```

Calculate overlaps

First calculate the four individual overlap statistics with associated null models for each trait separately.

```
set.seed(333)

iris_overlap_separate <- Ostats(traits = as.matrix(iris_traits),
                                sp = as.factor(iris_sp),
                                plots = as.factor(iris_plots),
                                nperm = 99)
```

```
## [1] "Calculating observed local O-stats for each community . . ."
##   |                                                                                                  |
## [1] "Calculating null distributions of O-stats . . . "
##   |                                                                                                  |
```

```
set.seed(222)

iris_overlap_multivariate <- Ostats_multivariate(traits = as.matrix(iris_traits),
                                                 sp = as.factor(iris_sp),
                                                 plots = as.factor(iris_plots),
                                                 nperm = 99,
                                                 hypervolume_args = list(verbose = FALSE, method = 'box
```

```
## [1] "Calculating observed local O-stats for each community . . ."
##   |                                                                                                  |
## [1] "Calculating null distributions of O-stats . . . "
##   |                                                                                                  |
```

```
write.table(t(signif(iris_overlap_separate$overlaps_norm,2)), col.names = FALSE)
```

```
## "Sepal.Length" 0.31
## "Sepal.Width" 0.51
## "Petal.Length" 8.2e-08
## "Petal.Width" 2.8e-05
```

The four univariate null model z-scores are:

```
write.table(t(signif(iris_overlap_separate$overlaps_norm_ses$ses,3)), col.names = FALSE)
```

```
## "Sepal.Length" -16.9
## "Sepal.Width" -11.2
## "Petal.Length" -23.3
```

```
## "Petal.Width" -21.7
```

The multivariate overlap statistic for all traits combined is NA. Its corresponding z-score is NA.

Plots

add some plots

To do

The `Ostats_multivariate` function is sufficiently different from the `Ostats` function that it may be OK to keep them as two separate functions. However, they could be merged into one function. If there are multiple traits, the user would have to specify whether they want to calculate multiple univariate overlap statistics, or a single multivariate overlap statistic.

Obviously I haven't done a lot of tests on these functions yet, nor have I written any documentation. That would be something to do as well.