Sergio Nolazco Plasier

Mestrando em Ecologia - IB/USP

Áreas de Interesse: Ecologia das populações e comunidades. **Sub-áreas:** Efeitos da alteração ambiental sobre as espécies; estimativas de parâmetros populacionais para espécies ameaçadas e endêmicas.

Taxa de preferência: Aves

Meus exercícios

link: exec

Trabalho final

PROPOSTA

Randomization test for functional diversity pairwise comparisons

Functional diversity (FD; Petchey & Gaston 2002) is a distance and dendrogram-based index to examine trends in species diversity traits among assemblages. This is one of the many measures related to functional traits that have been suggested to be closely related to ecosystem processes, but unlike classical diversity indices, commercial softwares do not usually provide ways to test the significance of changes in their values or even calculate them. For FD and other diversity indices, the differences of point estimations between two samples may be a merely reflect of sampling variability, and it becomes essential to test the significance of this difference. Andrew Solow (1993) described a simple way for testing based on a resampling procedure. The method consists on combining two samples of abundance data (species by abundance) into one and then drawn individuals to generate two random partitions with the same number of individuals as the observed samples. For each subsample (partition) the diversity index is calculated and also the difference between both values, and the procedure is repeated n times. Finally, the estimated p-value for a two-sided test is just the proportion of partitions with an absolute index difference (|simulated delta|) larger than the observed (|observed delta|), and for a one-sided test is just the proportion of partitions with a larger delta than the observed.

Details

Inputs: A matrix or data frame (mixed variables) where rows are species and columns are functional traits, and an abundance matrix in which rows are samples and columns are species.

Steps:

1. Create a table with general results such as number of species and individuals by sample and by samples combined (the first step of our null scenario!). Both samples will be selected by indexing them from the abundance matrix (the positions will be defined in the function's arguments).
2. Calculate FD for each sample (a pair of samples from my abundance matrix defined in the function's arguments). Distance method and clustering algorithms will be also defined in the function's arguments.
3. Subtract FD values of sample 1 and sample 2 (observed delta).
4. Start the resampling procedure by extracting from the pooled set (samples combined) 10,000 random partitions with the same number of individuals as sample 1 and other 10,000 random partitions with the same number of individuals as sample 2 (or the number of randomizations defined in the corresponding function's argument).
5. Subtract FD values between the null samples. The result will be a series of simulated delta equal to the number of randomizations.
6. End up with hypothesis testing conditioning the results based on the p-values obtained by two-sided and one-sided tests.

Outputs:

1. A table with observed number of species and individuals by samples and by samples combined.
2. Observed FD values for each sample and the difference between them (observed delta).
3. p-values for two-sided and one-sided tests.
4. A KISS interpretation of the hypothesis testing results (e.g. FD is equal between samples at 0.05 significance level)
5. A graphic device with the functional dendrograms for each observed sample and a histogram with the distribution of simulated delta frequencies along with the observed delta (vertical red line).

References:

Petchey, O.L. & Gaston, K.J. (2002) Functional diversity (FD), species richness, and community composition. Ecology Letters, 5, 402–411.

Solow, A.R. (1993) A simple test for change in community structure. Journal of Animal Ecology 62: 191-193.

PLANO B

A simple method to measure the contribution of each species to abundance-based similarity (or dissimilarity) indices

When calculating abundance-based similarity (or dissimilarity) indices between two samples, it can be useful to identify which species are contributing most to the observed parameter. A simple approach that can be applied to most if not to all known measures is the use of a jackknife resampling procedure. Removing one species at a time from the pool data and calculating the index between samples returns values for each case where a specific species was removed. Then the difference between the index with all the species included and the index with a particular species removed gives a measure of the contribution of that species to the overall measure. The differences can be transformed into a more friendly expression just by summing all the obtained absolute values and calculating the related percentage for each of them. Other methods to look for are Similarity Percentage Analysis (SIMPER; Clarke 1993) and Dufrene-Legendre Indicator Species Analysis (Dufrene

& Legendre 1997).

Details

Inputs: An abundance matrix with two or more samples in columns and species in rows.

Steps:

1. Select the columns (samples) for which the similarity (or dissimilarity) index is going to be calculated.
2. Calculate the measure of interest (maybe with the help of another existing function to make it flexible in terms of calculating any of a bunch just by defining it with only one function argument).
3. Remove one species for the calculation of the measure of interest, and repeat this for all the species observed in the samples. The result is n values as species observed in the samples.
4. Calculate the difference between the measure with all the species included (N) and with N-1 species for every case.
5. Transformed all the differences for each removed species to absolute values and calculate the percentage in relation to the total variation.
6. Sum individuals by species and bind this vector as a new column to a data frame with percentage contribution for each species.

Outputs:

1. The observed similarity(or dissimilarity) index value.
2. A table with the percentage of contribution and the number of observed individuals for each species.

References

Clarke, K. R. (1993). Non-parametric multivariate analysis of changes in community structure. Australian Journal of Ecology 18, 117-143.

Dufrene, M. and Legendre, P. 1997. Species assemblages and indicator species: the need for a flexible asymmetrical approach. Ecol. Monogr. 67(3):345-366.

Hi, Sergio!

I really liked your first idea: it was very well explained and seems to be quite interesting in terms of the analysis of functional diversity data. I've seen in the forum that you have already started to test the viability of your function: how is it going?

I have only two doubts concerning your plan A: a) I couldn't get the difference between the estimated p-values for two-sided and one-sided test; b) have you checked how it works to generate functional dendrogram graphics? — I am asking you this because I've never done similar things and I am not from Biology (:

> About your plan B, it seems to be interesting but a bit simple. If you decide to develop it, then you should add something more to your function. In this case, we can talk to another monitor and ask for suggestions.
>
> — *Viviane Santos 2015/03/24 14:43*

Comentários Vitor Rios

I'd also go with the first proposal. As for the p-values, it is important that the user specifies wether the test is one-sided or two-sided before the test is run, I sugsest it should be a parameter of your function. All in all, the proposal is very clear

RESPONSE Viviane & Vitor: There's no need to include an argument that specifies if the user want to run a one-sided or two sided test! Why ????! Does this look a little bit informal for science? Let me explain why.

Functional analyses are evolving really fast and ways to calculate their measures are rocket science for many students, like me. Not just because of the complex input files and data manipulation, even if you cross that barrier and improve your theoretical ground, R packages for FD-related measures and tests look like trying to decode Sumerian for the amateur. My function in this way is going to run the analysis including the minimal amount of arguments needed to obtain reliable results, supported by the help page. The output includes values with graphics (dendrograms and histogram), citation for the methods used and just a pair of lines that explains what's going on at every stage of the randomization test, don't worry "all in a nutshell" (all for dummies like me). So, the point is that when you get to the part when R gives the p-values for both tests (1 and 2 sided) and a graphic plus one or two text lines (depending on the results) that will clear up all doubts, all with the objective that the user can understand how this thing works. I made a draw that explains why is important to run both tests at a time for the user to understand the basis of the randomization test. In this design it is also neccesary to run the two-sided test first to see if there are differences between samples and then go for the one-sided test, very inneficient to do both things separately, running 10,000 resamplings for each null sample. Further, it will be much easier to understand the logic behind the test with all the output elements in conjunction. What is also important to know is that here we are testing FD differences between samples and not specific point values of FD.

And Viviane if you have some doubts about the method, my objective is that after you ("the user") read the help page and tries the function you are going to understand what's going on empirically and with graphics. If I don't get that one, my function will be a complete fail and makes no difference, no sense (well...just for the course points!). By the way, I am struggling to develop the function.

Comentários Vitor Rios

Sergio, sorry, but it ***it is essential that the test be determined a priori***. The difference between a one-tailed and a two-tailed test is related to the very nature of the analysis being made: what kind of differences am I trying to detect? Any difference or a specific direction of difference? Trying to test for any and all possible differences, and choosing afterwards the one you want is a technique

known as data-dredging, and it is very bad form. Also, it is not necessary to re-run the randomization for each test, since the one- or-two-tailed question only arises when you are calculating the p-value, which is done after the randomizations are run (if I understand it correctly). I understand the relevance of a “for dummies” approach, but the user ***has*** to know a priori what they are doing, otherwise, they cannot interpret the results correctly, and will end up choosing the one with the most significant p-value

RESPOSTA Vitor, I understand your position. The first idea was intended to allow everyone to try the function and understand to some extent the global idea behind the test. So, understanding the possibilities of a significant value for two-sided and not significant value for one directed one-sided test was also a problem but everything depends on the user interpretation, the p-values are for that. However, I am going to follow your advice as it is statistically and scientifically correct, but I am also going to change the string outputs to the real meaning of p-values (rejecting or not rejecting based on evidence at the pre-defined signifance level of 0.05). So, now the arguments are going to be based on the alternative hypothesis to be tested (unequal FDs for two-sided test -1 option-, or more or less FD than the other for one-sided test-2 options-). Now the user have to be very clear of what p-values are (deeply) and is going to be forced to establish an hypothesis before any test is performed. Do you agree now? Besides this: Any other suggestion is welcome! This really makes the function easier to develop

I like the A proposal too! The randomization test proposed is only one of possible null model in ecology. I suggest you take a look in Gotteli's classical book “Null models im Ecology” available online in the author webpage. I like the null model that fixed the number of species and number of individuals (swap algorithm)- Gotelli, N.J. and A.M. Ellison. 2013: “the swap implementation of the fixed-fixed algorithm has proved to be the best choice...” Why we are writing in English? — *Alexandre Adalardo de Oliveira 2015/03/27 15:13*

Comentários Vitor Rios

Sergio, Yes, I agree :) Even though it makes the function a bit leass begginer-friendly, it is the correct way. Go ahead and write the function, if you have any doubts, send an email or post on the foruns :)

RESPOSTA Obrigado Prof. Alexandre pela sugestão. I am writting in english because my portuguese is still unreadable, but everyone else can write in portuguese, I understand. I am going to take a more deeply look at that swap algorithm, I didn't knew that it was also applicable for abundance-based matrices, not just for presence-absence ones. By the way, I am going to try my first randomization procedure as it took me a lot of time to developed it in R, a lot! Then I can try using other algorithms, and see if the “best choice” is appropiate for the kind of data and measures involved in my function.

Página de Ajuda

```
rtFD {unknown}                                                R Documentation

         Randomization test for functional diversity pairwise comparisons


Description:

Randomization test for the statistical comparison of dendrogram-based
measures of
functional diversity between two communities.

Usage:

rtFD(trait, community, nrandom, com1, com2, test = "unequal",  dist.metric =
“gower”,
clust.method = “average”)

Arguments:

trait
        A matrix or data frame where rows are species and columns are
                functional traits.

community
        An abundance matrix where rows are communities and columns are
                species.

nrandom
        Number of randomizations.

com1
                Community one to be specified by the corresponding row
number from
                the abundance matrix.

com2
                Community two to be specified by the corresponding row
number from
                the abundance matrix.

test
                Character string specifying the hypothesis testing in
relation to
                which the alternative hypothesis is going to be, including
"unequal"
                (the default), "less" and "more".
```

```
dist.metric
                Character string specifying the metric to be used, including
"gower"
                (the default), "euclidean" and "manhattan".

clust.method
                Character string specifying the clustering algorithm to be
used,
                including “average" (= UPGMA; the default), "ward.D",
"ward.D2",
                "single", "complete", "mcquitty" (= WPGMA), "median" (=
WPGMC) and
                "centroid" (= UPGMC).

Details:

Function rtFD implements a randomization test for comparing dendrogram-based
functional
diversity (FD; Petchey and Gaston 2002) between two communities. Communities
to be
compared can be selected from a multiple-community abundance matrix just by
using the
com1 and com2 arguments. Notice that the order is important as FD
differences and
hypothesis testing are always performed comparing the first against the
second community.

The randomization procedure consists on pooling the species-by-abundance
sets of both
communities and then drawn individuals without replacement to generate two
random
partitions with the same number of individuals as each corresponding
original
community. The process is repeated n times and FD is calculated for each
partition
along with the difference between them. Lastly, depending on the hypothesis
to be
tested the p-value for a two-sided test (test = "unequal") is just the
proportion
of partitions with absolute FD differences (simulated |delta|) larger than
the
observed (observed |delta|), while for one-sided tests is the proportion of
partitions with larger (test = "more") or smaller (test = "less")
differences
(simulated delta) than the observed (observed delta) (Solow 1993).

For FD calculations, the daisy function (package cluster) is applied first
to
calculate dissimilarity mesures between all species from the trait file.
Euclidean distances which are the root sum-of-squares of differences between
species traits, and manhattan as the sum of absolute differences are both
```

```
available,
as well as an improved implementation of the gower index (the default) for
handling
mixed data of numeric and class variables (nominal, ordinal, and binary
[symmetric/
asymmetric]). Chapter one of Kaufman and Rousseeuw (1990) book describes in
full
detail the daisy function.

The hclust function (package stats) performs a hierarchical cluster analysis
with
the set of dissimilarities from the object created through the daisy
function.
UPGMA clustering algorithm (clust.method = "average"; the default) is
recommended
since it gives the highest cophenetic correlation with the original
multivariate
distances (Podani and Schmera 2006), despite other seven methods are
available
(see Arguments). Finally, FD as defined by Petchey and Gaston (2002, 2006)
is
calculated using the treedive function (package vegan) which returns the sum
of the dendrogram branch lengths excluding the root for each community.
treedive
works reconstructing a dendrogram for each community species composition
based
on the cophenetic distances from the dendrogram that was produced by hclust.

Value:

freq.table
                A data frame with the observed number of species and
individuals for
                each selected community.

FD.table
                A data frame with the observed FD values for each selected
community
                and the difference between them (observed delta).

p.value
                A vector with the p-value for the chosen test.

Warning:

Species row names from the trait matrix (or data frame) must be identical to
the
species column names from the abundance matrix.
At the end of the randomization test a warning message from daisy function
may appear:
```

```
"binary variable(s) treated as interval scaled". This happens when two-
valued numerical
variable(s) do not have a type specified before the function is executed.
The default
type given to these variables is "interval scaled" which makes no difference
in the
analysis, however the warning is intended to persuade the user to think if
it is not
more appropiate to treat some of these as asymmetric binary variables
(asymmetrically
weighted). Just in the case there are asymmetric binary variables or
ordinal, the type
must be change in the daisy function arguments directly (inside rtFD code).

Note(s):

Packages cluster and vegan are needed.

Missing values (NAs) are allowed.

A set of 10000 randomizations normally takes less than 2 minutes for the
function to
complete the task.

See also:

Functions daisy, hclust and treedive.

Author(s):

Sergio Nolazco sergio_atm55@hotmail.com; sergio.nolazco@ib.usp.br

References:

Kaufman, L. and Rousseeuw, P.J. (1990) Finding Groups in Data: An
Introduction to Cluster Analysis. Wiley, New York.

Petchey, O.L. and Gaston, K.J. (2002) Functional diversity (FD), species
richness, and community composition. Ecology Letters 5: 402–411.

Petchey, O.L. and Gaston, K.J. (2006) Functional diversity: back to basics
and looking forward. Ecology Letters 9: 741–758.

Podani, J. and Schmera, D. (2006) On dendrogram-based methods of functional
diversity. Oikos 115: 179–185.

Solow, A.R. (1993) A simple test for change in community structure. Journal
of Animal Ecology 62: 191–193.


Example(s):
```

```
require(cluster)
require(vegan)
## Example for an assemblage of Serpentes.
## Read a file with functional traits by species.
traits <- read.table("trait_set.txt", header = T, row.names = 1, sep = "\t")

## Read a file with species abundance by sites.
community <- read.table("community_set.txt", header = T, row.names = 1, sep
= "t")

## Compare sites 17 and 3 from the community file, testing for the
alternative hypothesis that the former is MORE functionally diverse.
rtFD(traits, community, 10000, 17, 3, "more")
```

Datasets for example(s)

trait_set.txt community_set.txt

Código da Função

```
# Funtion notation, including some character string default arguments.

rtFD <- function(trait, community, nrandom, com1, com2, test = "unequal",
dist.metric = "gower", clust.method = "average")
{

# Define the condition that species names between trait and community must
be identical for the function to execute.

if(identical(as.character(rownames(trait)),as.character(colnames(community))
) == FALSE)
    + stop("Species names are not identical between trait and community
datasets")
    else {
# Combine the information of the two selected communities from the abundance
matrix.

        community.comb <- colSums(community[c(com1, com2), ])
# Count species for each community and for the pooled community.

        s1 <- sum(community[com1, ] > 0)
        s2 <- sum(community[com2, ] > 0)
        S <- sum(community.comb > 0)

# Count individuals for each community.
        n1 <- sum(community[com1, ])
```

```
        n2 <- sum(community[com2, ])

# Create and transpose data frame objects with the number of species and
individuals calculated.
        richness <- t(data.frame(s1, s2, S))
        counts <- t(data.frame(n1, n2, n1 + n2))
# Create a data frame with the calculated frequency information.

        freq.table <- cbind(richness, counts)
        colnames(freq.table) <- c("Species", "Individuals")
        rownames(freq.table) <- c(rownames(community[c(com1,com2),]),
"Combined")
# Create an object of class "dissimilarity" containing the dissimilarities
between rows (species) from the traits file.

        D <- daisy(trait, dist.metric)

# Create an object which describes the dendrogram constructed by the
function hclust.
        tree <- hclust(D, clust.method)

# Replace the third element in the call list by the "clust.method" argument
to fix a bug generated by the update function in treedive that not recognize
"clust. method" as an argument.

        tree$call[[3]] <- clust.method

# Get observed FD values using the function treedive and transform the
corresponding vector object into a data frame for better manipulation.

        obFD <- data.frame(treedive(community, tree))

# Extract from the created object the rows corresponding to the FD values of
the communities of interest and values to three decimal places.
        com1.obFD <- round(obFD[com1, 1], d = 3)

        com2.obFD <- round(obFD[com2, 1], d = 3)

# Calculate the difference between FD of selected communities and round that
values to three decimal places.

        diff.obFD <- round((obFD[com1, 1] - obFD[com2, 1]), d = 3)

# Create a data frame with the calculated FD information.

        FD <- t(data.frame(com1.obFD, com2.obFD))
        delta <- t(data.frame(diff.obFD, 0))
        FD.table <- cbind(FD, delta)
        colnames(FD.table) <- c("FD", "delta")
        rownames(FD.table) <- c(rownames(community[c(com1,com2), ]))
# Create an object with the names of species found in the first community .
```

```
        species.com1 <- colnames(community)[community[com1, ] > 0]

# Get a subset of the trait matrix (or data frame) for the species found in
the first community.

        trait.com1 <- trait[species.com1, ]

# Create a "dissimilarity" object and an "hclust" object with the neccesary
information for plotting a functional dendrogram for the first community.

        distance.com1 <- daisy(trait.com1, dist.metric)
        tree.com1 <- hclust(distance.com1, clust.method)
# Same procedure for the second community.

        species.com2 <- colnames(community)[community[com2, ] > 0]
        trait.com2 <- trait[species.com2, ]
        distance.com2 <- daisy(trait.com2, dist.metric)
        tree.com2 <- hclust(distance.com2, clust.method)
# Create an object to overwrite the result of the randomization algorithm.

        diff.rFD <- rep(NA, nrandom)
# Write a for-loop for the randomization algorithm.

        for (k in 1:nrandom) {

# Create a vector object of individuals for the pooled community (first and
second communities combined).

            ind <- sort(rep(1:length(community.comb), times =
community.comb))

# Take a random partition without replacement for the same number of
individuals as the first community.
            sort.nullcom1 <- sort(sample(ind, n1, replace = FALSE))

# Take a random partition without replacement for the same number of
individuals as the second community.
            sort.nullcom2 <- sort(sample(ind, n2, replace = FALSE))

# Create an object to overwrite one random edited partition with the same
number of individuals as the first community (first null community!).
            sim.nullcom1 <- rep(NA, length(community.comb))

# Write a for-loop for editing a random partition with the same number of
individuals as the first community.
                for (i in 1:length(community.comb)) {

# Sum the number of elements with the same numeric value corresponding to
their equivalent positions in the first null partition.
```

```
                    sim.nullcom1[i] <- sum(sort.nullcom1 == i)
                }

# Change the class of the resulting object to add the corresponding species
row names.
                sim.nullcom1 <- as.table(sim.nullcom1)
                rownames(sim.nullcom1) <- names(community.comb)

# Repeat the procedure for a random partition with the same number of
individuals as the seocnd community (second null community!).

                sim.nullcom2 <- rep(NA, length(community.comb))
                for (j in 1:length(community.comb)) {
                    sim.nullcom2[j] <- sum(sort.nullcom2 == j)
                }
                sim.nullcom2 <- as.table(sim.nullcom2)
                rownames(sim.nullcom2) <- names(community.comb)

# Bind partition objects to calculate FD with the treedive function.

                sim.community<-rbind(sim.nullcom1, sim.nullcom2)
                rFD <- data.frame(treedive(sim.community, tree))

# Calculate the difference between FD of simulated community partitions.

                diff.rFD[k] <- rFD[1, 1] - rFD[2, 1]
        }
# Calculating p values for each hypothesis testing possibilities, two-sided
test and one-sided testing for the alternative hypothesis that first
community is more diverse or less diverse than second comunity.

        twoside.pv <- sum(abs(diff.rFD) > abs(diff.obFD)) / nrandom
        m.oneside.pv <- sum(diff.rFD > diff.obFD) / nrandom
        l.oneside.pv <- sum(diff.rFD < diff.obFD) / nrandom

# Define conditions for which p-value is going to appear in the output
depending on the alternative hypothesis selection (test argument).

            if (test == "unequal") {
            p.value = twoside.pv
            }
            if (test == "more") {
            p.value = m.oneside.pv
            }
            if (test == "less") {
            p.value = l.oneside.pv
            }

# Create a layout for three graphics.

        layout(matrix(c(1, 2, 3, 3), 2, 2, byrow = TRUE), widths = c(1, 1),
```

```
heights = c(1, 2))
# Define the parameters for plotting a functional dendrogram for the first
community.

        par(mar=c(0,4.5,3.5,1))
        plot(tree.com1, labels=FALSE, hang=-1,
main=paste(rownames(community[com1,]),"\n","FD =", round(obFD[com1,1],d=3)),
cex.main=1, ylab="Height")

# Define the parameters for plotting a functional dendrogram for the second
community.

        par(mar=c(0,3,3.5,1))
        plot(tree.com2, labels=FALSE, hang=-1,
main=paste(rownames(community[com2,]),"\n","FD =", round(obFD[com2,1],d=3)),
cex.main=1, ylab="")
# Define the parameters for plotting an histogram of the simulated FD
differences between simulated partitions, including a vertical line for the
observed difference.
        par(mar = c(5.5, 6, 7.5, 4))
        hist(diff.rFD, main = "Histogram of simulation results", cex.main =
1.2, xlab = "FD delta", ylab = "Frequency", col = "lightgray", sub =
paste("Observed delta = ", substitute(diff.obFD)), cex.sub = 0.9, col.sub =
"red")
        obs.line <- abline(v = diff.obFD, col = "red", lwd = 2)

# Return a list of various class objects.
        return(list(Frequency_table = freq.table, Functional_diversity =
FD.table, p_value = p.value))
    }
}
```

Arquivo da Função

rtfd_function.r

For a beginner-friendly output try this code!

fd_friendlyout.r