Vignette of Using FEEr Package

Tao Zhang, Grant M. Domke, Matthew B. Russell, and Jeremy W. Lichstein

2021-06-22

Introduction

Functional extension and evenness (FEE) is an index to quantify the functional diversity of species assemblies (communities). The FEE index calculated in the **FEEr** package presents the following key features:

- it allows a fair comparison of functional diversity across different species richness levels;
- it preserves the essence of single-facet indices while overcoming some of their limitations;
- it standardizes comparisons among communities by taking into consideration the trait space of the shared species pool;
- it has the potential to distinguish among different community assembly processes.

A detailed description of the FEE index can be found in Zhang et al. (2021). This vignette demonstrates how to use the **FEEr** package to calculate this index under different settings.

Data Preparation

The major data inputs for FEE calculations are species traits and species composition in the studied communities. Here, we generate artificial data for the purpose of demonstration.

```
library(FEEr)
```

```
set.seed(321)
d_trait <- cbind(tr1 = rnorm(10, 0, 1), tr2 = rnorm(10, 3, 2))
d_comm <- rbind(comm1 = sample(0:4, 10, replace = TRUE, prob = c(8, 4:1)),
                comm2 = sample(0:4, 10, replace = TRUE, prob = c(8, 4:1)),
                comm3 = sample(0:4, 10, replace = TRUE, prob = c(8, 4:1)),
                comm4 = sample(0:4, 10, replace = TRUE),
                comm5 = sample(0:4, 10, replace = TRUE),
                comm6 = sample(0:4, 10, replace = TRUE))
d_trait
```

```
##              tr1      tr2
##  [1,]  1.7049032 3.695403
##  [2,] -0.7120386 5.969184
##  [3,] -0.2779849 3.376651
##  [4,] -0.1196490 7.886520
##  [5,] -0.1239606 0.693121
##  [6,]  0.2681838 1.390657
```

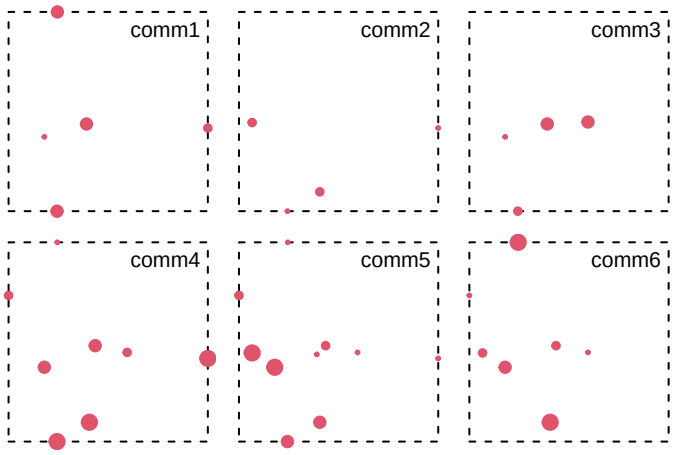
```
##  [7,]  0.7268415 3.912138
##  [8,]  0.2331354 3.840665
##  [9,]  0.3391139 4.155169
## [10,] -0.5519147 3.892712
```

```
d_comm
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## comm1    2    0    1    3    3    0    0    3    0     0
## comm2    1    0    0    0    1    2    0    0    0     2
## comm3    0    0    1    0    2    0    3    3    0     0
## comm4    4    2    3    1    4    4    2    0    3     0
## comm5    1    2    4    1    3    3    1    1    2     4
## comm6    0    1    3    4    0    4    1    0    2     2
```

In this example, the trait data `d_trait` contains two traits (`tr1` and `tr2`) of 10 species (row numbers of `d_trait`). The community composition data `d_comm` describes the abundance of these 10 species in six communities (`comm1` to `comm6`). In the following plot, species presented in these communities are shown as red dots in the trait space (dashed lines), where dot size indicates species abundance.

```
par(mfrow = c(2, 3), mar = rep(0.3, 4))
for(i in 1:6) {
  plot(d_trait, axes = F, type = "n")
  polygon(rbind(c(min(d_trait[,1]), min(d_trait[,2])),
                c(max(d_trait[,1]), min(d_trait[,2])),
                c(max(d_trait[,1]), max(d_trait[,2])),
                c(min(d_trait[,1]), max(d_trait[,2]))), lty=2)
  points(d_trait[which(d_comm[i,] > 0),], cex = 0.4*d_comm[i, which(d_comm[i,] > 0)],
         col = 2, pch = 19)
  legend("topright", legend = paste0("comm", i, " "), bty = "n")
}
```



Index Calculation

The function `computeFEE` is the wrapper of FEE calculation. In its simplest form, the function only requires the above two data as inputs:

```
fee1 <- computeFEE(d_trait, d_comm)
```

```
fee1
```

```
## [1] 0.8769 0.5796 0.1451 0.8448 0.3226 0.4519
```

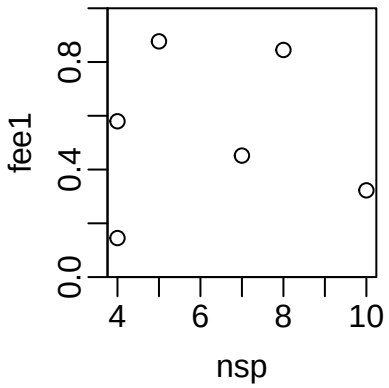
Species richness of a community is often used as a measurement of biodiversity (taxonomic diversity). As a measurement of functional diversity, the FEE index is not necessarily correlated with species richness. Indeed, one motivation of developing the FEE index was to remove the possible intrinsic dependence with species richness from a functional diversity index (refer to Zhang et al. 2021 for details).

The `computeNSP` function can be used to find out species richness (number of species) of the communities. The following scatter plot indicates species richness does not align with the FEE index in our example.

```
nsp <- computeNSP(d_comm)
nsp
```

```
## comm1 comm2 comm3 comm4 comm5 comm6
##     5     4     4     8    10     7
```

```
par(mar = c(2.8, 2.8, 0.2, 0.2), mgp = c(1.8,0.5,0))
plot(nsp, fee1, ylim = 0:1, yaxs = "i")
```



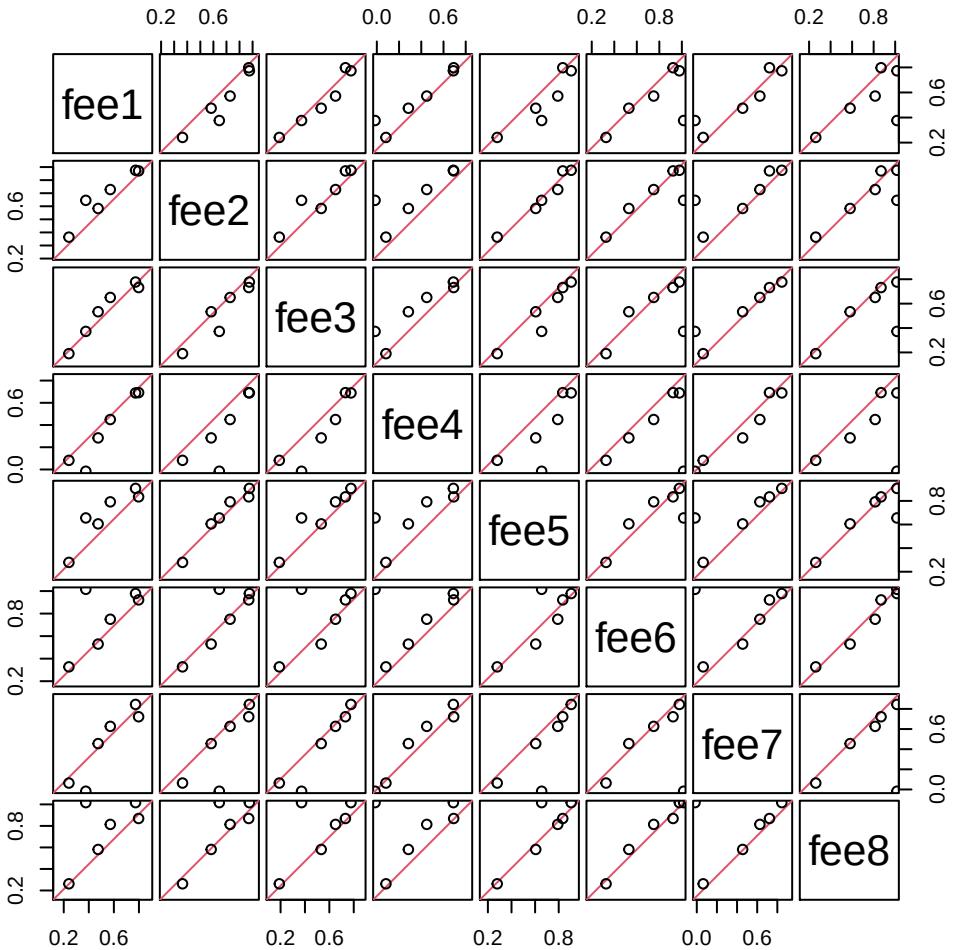
A user might consider the following arguments (and questions) in calculating the FEE index:

- **`abundWeighted`.** Do I consider abundance of the species in the calculation (default), or focus simply on species presence vs. absence (`abundWeighted = FALSE`)?
- **`dis_metric`.** How do I quantify the distance (or dissimilarity) between two species in trait space? In the current version, traits can be continuous or ordinal. In the function, each trait is standardized into the 0-1 range and species distances are calculated based on the standardized trait values. The function currently supports two distance metric options: `euclidean` (default) and `manhattan`. In case of a single trait (i.e., one-dimensional trait space), the two options are equivalent. Refer to `?dist` for more information about the two options.
- **`poolBased`.** Do I use traits of pooled species (`pool_traits`) as the prior knowledge of trait distribution (`poolBased = TRUE`), or assume no prior knowledge about species trait distribution (default)?

Different considerations in these arguments could result in different FEE results, as indicated in the following calculation and pair-wise scatter plots:

```
fee2 <- computeFEE(d_trait, d_comm, abundWeighted = FALSE)
fee3 <- computeFEE(d_trait, d_comm, dis_metric = "manhattan")
fee4 <- computeFEE(d_trait, d_comm, poolBased = TRUE)
fee5 <- computeFEE(d_trait, d_comm, dis_metric = "manhattan", abundWeighted = FALSE)
fee6 <- computeFEE(d_trait, d_comm, abundWeighted = FALSE, poolBased = TRUE)
fee7 <- computeFEE(d_trait, d_comm, dis_metric = "manhattan", poolBased = TRUE)
fee8 <- computeFEE(d_trait, d_comm, dis_metric = "manhattan", abundWeighted = FALSE,
                   poolBased = TRUE)
```

```
# put together FEE results under different argument settings
fees <- cbind(fee1, fee2, fee3, fee4, fee5, fee6, fee7, fee8)
panel_ln <- function(x,y) {   # show the 1:1 line in the scatter plots
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(-0.02, 1.02, -0.02, 1.02))
  abline(0, 1, col = 2)
  points(x, y)
}
pairs(fees, gap = 0.3, panel = panel_ln)
```



Other Options

There are three other arguments in the `computeFEE` function. Their settings will not affect the calculation of results, but they might affect the calculation speed (`user_ecdf` and `doParallel`) or expand output content (`doFEE0`).

- **`doFEE0.`** The FEE calculation depends on the result of raw functional extension and evenness metric (FEE_0). This argument determines whether or not to include this intermediate result in the output (see the example below). The default is `FALSE`.

```
# calculate both FEE0 and FEE for comm2 and comm3
fee1b <- computeFEE(d_trait, d_comm[2:3,], doFEE0 = TRUE)
```

```
fee1b
```

```
## $FEE0
## [1] 0.9630356 0.6254542
##
## $FEE
## [1] 0.5796 0.1451
```

- **`user_ecdf.`** The FEE calculation depends on the empirical cumulative distribution functions (eCDF) of FEE_0. An eCDF is derived from null communities based on either the species pool (when `poolBased = TRUE`) or the assumption of no prior knowledge about trait distribution (refer to Zhang et al. 2021 for details).

  Some pre-calculated eCDFs under the assumption of no prior knowledge about trait distribution are enclosed in the **FEEr** package. They cover the combination scenarios of species richness from 2 to 90, trait space dimension from 1 to 6, and two distance metrics (euclidean and manhattan). For other scenarios, especially species-pool-based calculation, eCDFs are not pre-calculated. The argument `user_ecdf` is used to specify the file where the additional eCDFs are saved. If a file path is provided there, the eCDFs that are not pre-calculated will be saved in that file after they are calculated for the first time. Thus, when calculating FEE for other communities under same eCDF conditions, the time spent generating eCDFs could be saved since the eCDFs in that file can be loaded and reused:

```
if(file.exists("ecdf_exp")) file.remove("ecdf_exp")
str_fee4b <- "computeFEE(d_trait, d_comm, poolBased = TRUE, user_ecdf = 'ecdf_exp')"
str_fee6b <- "computeFEE(d_trait, d_comm, abundWeighted = FALSE, poolBased = TRUE,
              user_ecdf = 'ecdf_exp')"
t_fee4b <- system.time(eval(parse(text = str_fee4b))) # save eCDFs to file
t_fee6b <- system.time(eval(parse(text = str_fee6b))) # use saved eCDFs
```

  As shown below, the second calculation is much faster because it reuses the eCDFs generated in the first calculation:

```
rbind(summary(t_fee4b), summary(t_fee6b))
```

```
##      user system elapsed
## [1,] 2.25   0.04    3.38
## [2,] 0.17   0.01    0.40
```

  However, be aware that if `poolBased = TRUE`, eCDFs will be species-pool-specific. So a user needs to be careful in managing the files of eCDFs. Assigning a wrong file of eCDFs to `user_ecdf` (e.g.,

confusing a file for X species pool with a file for Y species pool) would result in false results of FEEs. And if `pool_traits` (traits of species pool) has any change (e.g., adding or removing species, including new traits or excluding old traits, and updating trait values for some species), the file for old eCDFs cannot be reused.

- **`doParallel`.** When the size of species pool is very large, the number of communities is very large, and/or the number of new eCDFs is very large, the calculation might take time if it is conducted sequentially (`doParallel = FALSE`, default). A user could save time by setting `doParallel = TURE` to conduct parallel computation. However, the parallel backends available will be system-specific. The parallel computation in this function depends on `foreach` function and %dopar% operator in R's `foreach` package. Refer to `?foreach::foreach` and `?parallel` for more details.

Other Functions

Besides the wrapper function `computeFEE`, the **FEEr** package includes the following functions, which are called in the `computeFEE` function or are relevant to biodiversity measurement.

- **`communityMST`.** Minimum spanning tree (MST) plays an important role in calculating the FEE index. MST characterizes the species' distribution of a community in trait space. Its branch lengths are used to calculate FEE_0 (refer to Zhang et al. 2021 for details). The `communityMST` function returns branch lengths of MST for a community:

```
# MST branch lengths of comm1, with and without species abundance considered
br1a <- communityMST(d_comm[1,], d_trait)
br1b <- communityMST(d_comm[1,], d_trait, abundWeighted = FALSE)
rbind(br1a,br1b)
```

```
##          [,1]     [,2]     [,3]      [,4]
## br1a 3.248965 2.534189 1.478919 0.6903282
## br1b 4.061206 2.687947 1.478919 0.6903282
```

- **`computeNSP`.** As shown in the example at the beginning, this function returns species richness of communities.

- **`FEE0_eCDF`.** Using a null model approach, this function builds the eCDF of FEE_0 by calculating FEE_0 of a large number of null communities. These null communities all share the same species richness, trait space, and species pool (or distribution of trait values). If `pool_traits` is not provided (default), the null models are generated under the assumption of no prior knowledge about the trait values (species traits of null models are sampled from the uniform distribution). If `pool_traits` is specified, the null models are generated by randomly sampling species from the pool.

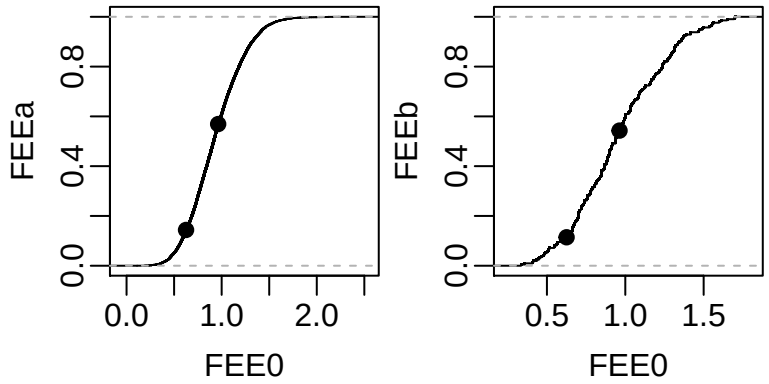
```
# build the eCDF for the case of 4 species in 2D trait space
ecdf_sp4a <- FEE0_eCDF(4, 2)    # no prior knowledge of trait values
ecdf_sp4b <- FEE0_eCDF(4, 2, pool_traits = d_trait)   # species pool is specified
```

FEE_0's eCDF is served as a bridge to translate a raw FEE_0 metric into FEE index:

```
# comm2 and comm3 have 4 species, so use them to illustrate the calculation
fee0 <- fee1b$FEE0    # FEE0 of comm2 and comm3, see "fee1b" above
cbind(FEE0 = fee0, FEEa = ecdf_sp4a(fee0), FEEb = ecdf_sp4b(fee0))
```

```
##           FEE0   FEEa      FEEb
## [1,] 0.9630356 0.5691 0.5428571
## [2,] 0.6254542 0.1437 0.1142857
```

```
par(mfrow = c(1,2), mar = c(2.8, 2.8, 0.2, 0.2), mgp = c(1.8,0.5,0))
plot(ecdf_sp4a, xlab = "FEE0", ylab = "FEEa", main = "", pch = ".")
points(fee0, ecdf_sp4a(fee0), pch = 19)
plot(ecdf_sp4b, xlab = "FEE0", ylab = "FEEb", main = "", pch = ".")
points(fee0, ecdf_sp4b(fee0), pch = 19)
```




```
fee1b
```

```
## $FEE0
## [1] 0.9630356 0.6254542
##
## $FEE
## [1] 0.5796 0.1451
```

It is worth pointing out that the outputs of `FEEa` and `FEEb` values might not be exactly same as the above ones. This is because when running the `FEE0_eCDF` function, the factor of randomness in building null models could result in slightly different eCDFs every time.

Acknowledge

Aditya Gaydhani participated in the early-stage development of this package.