fundiversity: a modular R package to compute functional diversity indices

Matthias Grenié[a,b,c,1], Hugo Gruson[c,d,2]

[a]*German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig Puschstraße 4 04103 Leipzig Germany*
[b]*Leipzig University Ritterstraße 26 04109 Leipzig Germany*
[c]*CEFE Université de Montpellier CNRS EPHE IRD Université Paul Valéry Montpellier 3 Montpellier France*
[d]*Centre for Mathematical Modelling of Infectious Diseases London School of Hygiene \& Tropical Medicine London UK*

Abstract

(max 350 words)

1. Functional diversity is widely used and widespread. However, the main package used to compute functional diversity indices `FD` is not flexible and not adapted to the volume of data used in modern ecological analyses.
2. We here present `fundiversity`, an R package that eases the computation of classical functional diversity indices. It leverages parallelization and memoization (caching results in memory) to maximize efficiency with data with thousands of columns and rows.
3. In addition to being more flexible we did a performance comparison with packages that provide analog functions. `fundiversity` was always an order of magnitude quicker than alternative packages.
4. `fundiversity` aims to be a lightweight efficient tool to compute functional diversity indices, that can be used in a variety of contexts. Because it has been designed following clear principles, it is easy to extend. We hope the wider community will adopt it and we welcome all contributions.

Keywords: biodiversity;, diversity facet;, R package;, functional biogeography;, functional ecology;, community ecology

Running title (max 45 char.): fundiversity: functional diversity in R
Word count (3000-4000 incl. refs & captions): 3453

Introduction

Functional diversity, the diversity of traits across scales, is a major facet of biodiversity (Pavoine & Bonsall, 2011). It has been widely used across ecological contexts (Cadotte et al., 2011) and has been shown to relate to ecosystem functioning (Díaz & Cabido, 2001; Leps et al., 2006). Many indices exist to characterize it across its three dimensions: richness, evenness, and divergence (Pavoine & Bonsall, 2011). To compute these indices in reproducible ways ecologists rely on computational tools able to crunch the numbers for thousands of species and thousands of sites. In the last few years, R has been the programming language of choice for ecologists (Lai et al., 2019; R Core Team, 2021). The main tool available to compute functional diversity indices has been the `FD` package which has now accumulated more than 1200 citations (Laliberté et al., 2014). But `FD` has been released in 2009 and has since then only received minor updates that stopped in 2015. At the same time, best practices in software development have changed dramatically and new, higher-performance tools have emerged in the R ecosystem. Additionally, since 2009, the size of

*Corresponding author
Email addresses: `matthias.grenie@idiv.de` (Matthias Grenié), `hugo.gruson@normalsup.org` (Hugo Gruson)
[1]Corresponding Author
[2]ORCID-ID <0000-0002-4659-7522>; ORCID-ID <0000-0002-4094-1476>

ecological datasets has grown exponentially (Farley et al., 2018; Wüest et al., 2020) and high performance computing (HPC) environments have become standard. There is therefore a dire need for a modern package to compute functional indices using state-of-the-art software development techniques and tools.

The main function of the `FD` package `dbFD()` lets users compute a dozen functional diversity indices in a single call from raw trait data (Laliberté et al., 2014). While great for exploratory analyses this can increase computation time when only a single index is needed. Furthermore, it does not enforce good practice in choosing beforehand the appropriate functional diversity index for the question(s) asked (Legras et al., 2018; Mason et al., 2013; Schleuter et al., 2010). It encourages the user to fish the functional diversity index that matches their predicted relationships (a form of p-hacking). This can lead the users to report all computed functional diversity indices even when there are no clear expectations on different functional diversity facets and/or to report correlated indices (Legras et al., 2018; Mason et al., 2013; McPherson et al., 2018; Schleuter et al., 2010). Computing all indices in a single function also makes long-term maintenance and addition of new indices harder. Finally, it adds an extra performance burden in the case where not all indices are needed.

The average size of datasets analyzed in ecology increased several folds in the last years (Wüest et al., 2020). Considering that most analyses on functional diversity rely on null models that increase the data size by two or three orders of magnitude (Gotelli & Graves, 1996), the need for efficient computation is paramount. First, any improvement of the algorithmic efficiency to compute functional diversity indices could save sensible amounts of time as it is repeated many times. For example, we noted that many R packages that compute functional diversity indices do not leverage matrix algebra with its libraries available that can cut the number of operations by orders of magnitude compared to using a loop directly in R. Second, functional diversity indices are generally computed over many mathematically independent sites. With the rise of multi-core computers, parallelization, i.e. splitting independent computations between different Computing Processor Units (CPUs), is becoming the norm. Very few functional diversity R packages propose parallelization which leaves the burden of implementing it to the user. There have been formidable new developments in this area in R over the last few years with the release `future` framework (Bengtsson, 2020) that allows the user to seamlessly parallelize computations on multiple cores on a single machine or across several machines, or even on a remote cluster without changing execution code. Third, computations on the same input can be cached through a process called memoization (Wickham et al., 2021). This avoids wasting computing power on previously seen inputs. Several functional diversity indices rely on the computation of convex hulls across a multi-dimensional space (Cornwell et al., 2006; Villéger et al., 2008). Caching the results of this costly computation could save time and computing power when measuring the diversity across similar sets, such as sites across a given region.

Increasing discussions are held regarding scientific software robustness and reliability in ecology (Mislan et al., 2016; Poisot, 2015; White, 2015; Wilson et al., 2017). Mainly because most ecologists are self-trained in programming (Farrell & Carey, 2018), these virtuous practices are rarely applied in ecology (Barraquand et al., 2014). For example, unit tests use predefined inputs to compare the software's outputs to expectations (Poisot, 2015). Unit tests have also become standard in R packages since the release of packages streamlining this process, such as `testthat` and `tinytest`. In part because of the relative recentness of the testing frameworks, very few R functional diversity packages provide unit tests to assess that the functions behave expectedly. Automatic tests of one's code are crucial when developing a tool for a wider audience as it may be used across different contexts.

We here propose a modern alternative to `FD` called `fundiversity` that benefits from modern development practices, necessary features for large-sized datasets (modularity, parallelization, and memoization), and greater flexibility. The package can be easily extended to accommodate additional diversity indices not covered by following a clear design pattern detailed in the next section. We then go through a use case to show how it can be used. We then compare the performance of `fundiversity` against similar packages.

Main features of `fundiversity`

To ensure the consistency of its functions and to make it user-friendly, `fundiversity` follows clear design principles. In this section, we expose its distinctive features and principles.

To give maximum flexibility to the users, we tried to build `fundiversity` as modular as possible. Each function in `fundiversity` computes a single functional diversity index, as such if the user is interested in computing a single index, they only need to use a single function. All functions in `fundiversity` are prefixed with `fd_` to avoid conflict with similarly named functions in other packages, as it's becoming standard practice in newer R packages (rOpenSci et al., 2021). In line with its modularity, we focused on making the inputs and outputs of functions coherent. The functions compute functional diversity indices using two main information: a species by traits matrix and a site by species matrix, all functions accept these two objects as first arguments. Because the function outputs one diversity value per site the outputs are always structured similarly: one `site` column that contains the name of its sites and one column named as the computed index (such as `FRic` when computing functional richness). The shape of the output is predictable and easy to be combined with other data.

Parallelization can be an easy way to vastly decrease computation speed why leveraging the architecture of modern computers. Almost all functions in `fundiversity` can be parallelized out of the box. `fundiversity` provides parallelization through the `future` backend (Bengtsson, 2020). Parallelization is toggled through a single function call using `future::plan()` before using fundiversity functions. Thanks to the flexibility given by the future backend, the code to use won't change whether parallelizing across several cores on a single computer, across multiple computers, or on a remote high-performance cluster. The user has only to update the call to `future::plan()` to distribute computations on another infrastructure. Furthermore, the future backend provides load balancing so that no cores/units stay idle for too long and the parallelized tasks are split evenly. The packages contains a dedicated "Parallelization" vignette to guide the users through transforming unparallelized to parallelized code (accessible through `vignette("parallel", package = "fundiversity")`).

Because functional diversity indices can be computed repeatedly on the same data subset, such as in null models, we can leverage these repeated computations to reuse already computed indices. For example to compute functional richness (FRic) the convex hull of the input data has first to be identified, then the program needs to compute the volume of this convex hull. The first step, identifying the convex hull, takes the most time and as such, storing the results of each computed convex hull across a subset of data can vastly cut computation time for a little memory footprint. Memoization consists in doing exactly that, it trades a little of computer memory (keeping the convex hulls stored) for more computation speed. `fundiversity` leverages memoization for all complex computations such as convex hulls. By default, memoization is turned on for FRic, the intersection of convex hulls, and FDiv. However, it can sometimes create a memory bottleneck which slows down the overall computation. The default behavior can always be overridden through a change in the option `fundiversity.memoise`.

Packages depend on one another to avoid reinventing the wheel and thus reuse already developed functions. A higher number of dependencies means that a package requires more packages to be installed before its installation. While a high number of dependencies minimizes code replication, it also comes with a high price, because if a single dependency breaks then the whole package cannot be installed anymore (Cox, 2019). Inflated dependencies have been identified as a major risk in software and especially scientific software development (Claes et al., 2014; Cox, 2019). `FD` only has four dependencies but other functional diversity packages have many more dependencies. This renders them quite brittle for the users after years of not being actively developed. `fundiversity` has been designed to only have minimal external dependencies, it currently depends on only four external packages: `future.apply` which depends only on two other packages, `Matrix` which is shipped with R, `geometry` and `vegan` on which `FD` also depends.

Because user flexibility is key, `fundiversity` has minimal assumptions on the input data structure. All its functions work with data frames, matrices, or sparse matrices alike. Sparse matrices are a different formalization of matrices that do not store explicitly the cells that contain zero. They offer a reduced memory footprint and optimized algebra library for computation (Bates & Maechler, 2021). These matrices are thus specifically relevant for occurrence/abundance matrices that contain many zeros. If the used data have a high proportion of zeros, using sparse matrices can vastly decrease computational time in `fundiversity`.

As we underlined in the introduction, automatic software testing, while not 100% foolproof, is needed to increase the confidence in the behavior of functions. It is widespread in computer science but less in scientific software development. This means that software behavior is never assessed against known inputs to make sure it behaves in expected ways. It does not mean that the software is of poor quality, but rather that some simple errors could introduce unnoticed changes in the behavior of functions. Most packages that compute functional diversity indices do not include any form of automatic testing. We do want to point out that most ecologists never received formal training in software development hence the lack of tests (Farrell & Carey, 2018). We designed `fundiversity` with many unit tests from the beginning, executing at least every single line of code once (i.e. achieving coverage of 100%).

`fundiversity` only computes alpha functional diversity indices, because other recent packages exist to compute other types of functional diversity indices [Hill numbers, Li (2018); beta-diversity indices, Baselga_betapart_2012]. We focused on indices available through the `dbFD()` function in the `FD` package and on indices that could benefit from faster implementation. fundiversity contains the following alpha functional diversity indices: functional richness (FRic), functional dispersion (FDis), functional divergence (FDiv), functional evenness (FEve), and Rao's quadratic entropy (Q). `fundiversity` also contains a beta-diversity index as it can be useful to compare functional richness between sites.

Table 1: List of functions available in `fundiversity` to compute functional diversity indices. The two last columns specify which functions are parallelizable and memoizable.

Function Name	Index Name	Source	Parallelizable	Memoizable
`fd_fdis()`	Functional Dispersion (FDis)	Laliberté & Legendre (2010)	Yes	No
`fd_fdiv()`	Functional Divergence (FDiv)	Villéger et al. (2008)	Yes	Yes
`fd_feve()`	Functional Evenness (FEve)	Villéger et al. (2008)	Yes	No
`fd_fric()`	Functional Richness (FRic)	Villéger et al. (2008)	Yes	Yes
`fd_fric_intersect()`	Functional β-diversity	Villéger et al. (2013)	Yes	Yes
`fd_raoq()`	Rao's Quadratic Entropy (Q)	Rao (1982)	No	No

Case Study

Now that we described the main features of fundiversity, we are going to show how to use it in practice when computing functional diversity indices. As an example dataset, we included in `fundiversity` site-species and trait data from Nowak et al. (2019). It is accessible through the use of the `data(..., package = "fundiversity")` function. This dataset describes the presence of bird species in South America at different elevations and four morphological traits.

```r
knitr::include_graphics(
  here::here("inst", "manuscript", "figures", "fundiversity_conceptual_diagram.svg")
)
```

[Figure 1 about here.]

The trait values show species in rows (species are specified as row names) and traits in columns with trait names as column names. Similarly, the site-species matrix contains sites as rows (site names are row names) and species as columns (species names are column names).

```
data("traits_birds",  package = "fundiversity")
data("site_sp_birds", package = "fundiversity")

head(traits_birds)
```

```
##                        Bill.width..mm. Bill.length..mm. Kipp.s.index Bodymass..g.
## Aburria_aburri                   18.35            35.48         0.18       1407.5
## Amazona_farinosa                 26.50            38.81         0.29        626.0
## Amazona_mercenaria               17.51            26.30         0.33        340.0
## Amazona_ochrocephala             20.17            31.40         0.26        440.0
## Ampelioides_tschudii             16.53            24.58         0.24         78.4
## Ampelion_rufaxilla               16.97            21.89         0.28         73.9
```

```
head(site_sp_birds)[, 1:3]
```

```
##           Aburria_aburri Amazona_farinosa Amazona_mercenaria
## elev_250               0                1                  0
## elev_500               0                1                  0
## elev_1000              1                1                  1
## elev_1500              1                0                  1
## elev_2000              0                0                  1
## elev_2500              0                0                  1
```

Now we obtained trait and occurrence data we need to compute the trait dissimilarity between each pair of species. As all traits are quantitative we first Z-score them, then we compute the Euclidean distance between pairs of species.

```
z_traits = scale(traits_birds, center = TRUE, scale = TRUE)

trait_distance = as.matrix(dist(z_traits))
```

We can then compute the functional richness of each index at each location. To do so we are using the `fd_fric()` function. It expects quantitative trait values as the first argument and a site-species matrix as the second argument.

```
library("fundiversity")

birds_fric = fd_fric(z_traits, site_sp_birds)

head(birds_fric)
```

```
##        site      FRic
## 1  elev_250 66.048816
## 2  elev_500 71.465678
## 3 elev_1000 43.354008
## 4 elev_1500 25.466685
## 5 elev_2000  7.725843
## 6 elev_2500  7.046431
```

All other functions in `fundiversity` use a similar structure, the first input is trait data the second one is a site-species matrix. For Rao's quadratic entropy computed through `fd_raoq()` functional dissimilarities can be specified as the third argument:

```r
# With functional dissimilarity
birds_raoq = fd_raoq(traits = NULL, site_sp_birds, dist_matrix = trait_distance)

# With trait values
birds_raoq_2 = fd_raoq(z_traits, site_sp_birds)

# Both options give the same results
identical(birds_raoq, birds_raoq_2)
```

```
## [1] TRUE
```

If all traits are not quantitative it is possible to transform them back into independent traits through the use of Gower's distance [Gower (1971); and its extensions: Podani (1999), Pavoine et al. (2009);] then applying multivariate analysis to obtain orthogonal dimensions (Maire et al., 2015).

Performance Comparison

To test the actual performance improvements realized by `fundiversity`, we compared computation time on standardized datasets across similar functions in other packages. We only compared "original" packages that provide actual functions and not wrappers that depend on other packages to compute functional diversity indices. We identified 6 packages that computed similar indices to `fundiversity`. Most indices are computed by the `FD::dbFD()` function but the comparison would be unfair as the function computes many indices in a single call while functions in `fundiversity` only compute single indices. We considered functions from: `adiv` (Pavoine, 2020), `BAT` (Cardoso et al., 2015), `betapart` (Baselga & Orme, 2012), `hillR` (Li, 2018), `mFD` (Magneville et al., 2022), and `FD` (Laliberté et al., 2014) (see Table 2 for the correspondence between packages). A continuously updated version of this section can be found in the performance comparison vignette within the `fundiversity` package with `vignette("performance", package = "fundiversity")`.

Table 2: List of functions available in `fundiversity` to compute functional diversity indices and corresponding functions in other packages. The name of the package is indicated before the `::` while the name of the functions (including specified arguments) follows.

Index Name	`fundiversity` Functions	Functions in other packages

Functional Dispersion (FDis)	`fd_fdis()`	`BAT::dispersion()`
		`FD::fdisp()`
		`mFD::alpha.fd.multidim(..., ind_vect = "fdis")`

Functional Divergence (FDiv)	`fd_fdiv()`	`mFD::alpha.fd.multidim(..., ind_vect = "fdiv")`
Functional Evenness (FEve)	`fd_feve()`	`mFD::alpha.fd.multidim(..., ind_vect = "feve")`
Functional Richness (FRic)	`fd_fric()`	`BAT::alpha()` (tree) `BAT::hull.alpha()` (hull) `mFD::alpha.fd.multidim(..., ind_vect = "fric")`

Index Name	`fundiversity` Functions	Functions in other packages

Rao's Quadratic Entropy (Q)	`fd_raoq()`	`adiv::QE()`
		`BAT::rao()`
		`hillR::hill_func()`
		`mFD::alpha.fd.hill(..., q = 2, tau = "max")`

Functional β-diversity	`fd_fric_intersect()`	`betapart::functional.beta.pair()`
		`hillR::hill_func_parti_pairwise()`

For testing purposes, we used datasets of increasing size with the number of species being 200, 500, or 1000; the number of traits 2, 4, or 10; the number of sites 50, 100, or 500. For each set of parameters, we generated a fictional site-species matrix and site-trait matrix, comprised only of continuous trait data. We used these simulated data to perform benchmarks across comparable functions among the selected packages. The benchmark ran 30 times through the `bench` package (Hester & Vaughan, 2021). A summary of the results of the benchmark can be seen in Fig. 2. Full results detailing the timings for each combination of parameters and functions are available in the Supplementary Material (Fig. S4).

[Figure 2 about here.]

We see that for all the indices and functions tested, `fundiversity` is at least an order of magnitude faster than alternative packages. For functional dispersion, `fundiversity` is two orders of magnitude faster compared to `BAT` and `mFD`. For functional divergence, `fundiversity` is one order of magnitude faster than `mFD`. For functional evenness, `fundiversity` is two orders of magnitude faster than `mFD` with sequential and parallelized versions having similar performances. For Rao's quadratic entropy, `fundiversity` is one order of magnitude faster than `hillR` and `mFD`, two orders faster than `adiv`, and three orders of magnitude faster than `BAT`. For functional richness, `fundiversity` is half an order of magnitude faster than the hull version of `BAT`, as well as one and a half order of magnitude faster than its tree version and `mFD`. For functional richness intersection (beta functional diversity), `fundiversity` is two orders of magnitude faster than `betapart` and `hillR`.

[Figure 3 about here.]

As shown on Fig. 2, on average the parallelized versions of `fundiversity` functions executed an order of magnitude faster than the sequential versions. For functional richness we even observed a difference of two orders of magnitude. However, for functional dispersion, parallelization increased overall computation time. This may be due to inherent parallelization issues: there is an overhead cost when splitting tasks across multiple cores of a computer. The efficiency of parallelization depends on the difficulty of the tasks that are split between cores. In the case of functional richness, the main task is computing the convex hull, which is computationally costly, that is why parallelization increase performance in this case. While computing functional dispersion is simpler, and as such, does not benefit from being split across different cores. Different values for the number of cores, species, traits or sites produce qualitatively the same results (full results in Fig. S5).

One important note regarding parallelization in `fundiversity`, is that it is important to avoid doing both memoization and parallelization simultaneously. Memoization creates a cache to avoid recomputing results, and the cache may be corrupted if several cores access the same results at the same time. We noticed that toggling memoization while performing parallelization severely increase total computational time, compared to sequential performance.

Note that these benchmarks only assess the packages computation speed and in no way any package intrinsic quality or usefulness. We're comparing `fundiversity`, a package whose one of the main goals is performance, with other packages that may have other primary goals and offer other benefits. Most other packages offer more features than simply computing functional diversity indices. For example, several packages offer nice default visualization functions to plot the different diversity indices, while we explicitly considered that visualization functions were not part of `fundiversity` and let the users decide how they want to plot their indices.

Conclusion

We proposed a modern alternative R package to compute functional diversity indices. This package follows current best development practices and leverages modern features like parallelization and memoization to increase its performance. This is only made possible by recent developments which were for the most part not available at the time when alternative packages came out. `fundiversity` does not propose to replace the entire toolkit for the researcher interested in functional diversity (including the upstream selection of the traits and building of a functional space) but instead focuses on improving the most computationally costing step: computing functional diversity indices. We hope it will be a useful contribution to this toolkit. This package aims to always be a work in progress and we welcome contributions from interested users and developers.

Acknowledgements

MG gratefully acknowledges the support of iDiv funded by the German Research Foundation (DFG–FZT 118, 202548816).

Authors' Contributions

Our authors' statement follows the Contributor Roles Taxonomy (CRediT, `https://casrai.org/credit/`). **Matthias Grenié**: Conceptualization, Methodology, Software, Resources, Writing - Original Draft, Writing - Review & Editing, Supervision. **Hugo Gruson**: Conceptualization, Methodology, Software, Writing - Review & Editing.

Data Availability

`fundiversity` is available on CRAN through `install.packages("fundiversity")` as well as on GitHub at `https://github.com/Bisaloo/fundiversity`, for archival all releases are available on Zenodo at `https://doi.org/10.5281/zenodo.4761754`. The data used in this article are available from the package, through `data(package = "fundiversity")` call.

Supplementary Information

[Figure 4 about here.]

[Figure 5 about here.]

References

Barraquand, F., Ezard, T. H. G., Jørgensen, P. S., Zimmerman, N., Chamberlain, S., Salguero-Gómez, R., Curran, T. J., & Poisot, T. (2014). Lack of quantitative training among early-career ecologists: A survey of the problem and potential solutions. *PeerJ*, *2*, e285. `https://doi.org/10.7717/peerj.285`

Baselga, A., & Orme, C. D. L. (2012). Betapart: An R package for the study of beta diversity. *Methods in Ecology and Evolution*, *3*(5), 808–812. `https://doi.org/10.1111/j.2041-210X.2012.00224.x`

Bates, D., & Maechler, M. (2021). *Matrix: Sparse and dense matrix classes and methods* [Manual].

Bengtsson, H. (2020). *A unifying framework for parallel and distributed processing in r using futures.* `https://arxiv.org/abs/2008.00553`

Cadotte, M. W., Carscadden, K., & Mirotchnick, N. (2011). Beyond species: Functional diversity and the maintenance of ecological processes and services. *Journal of Applied Ecology*, *48*(5), 1079–1087. `https://doi.org/10.1111/j.1365-2664.2011.02048.x`

Cardoso, P., Rigal, F., & Carvalho, J. C. (2015). BAT – Biodiversity Assessment Tools, an R package for the measurement and estimation of alpha and beta taxon, phylogenetic and functional diversity. *Methods in Ecology and Evolution*, *6*(2), 232–236. `https://doi.org/10.1111/2041-210X.12310`

Claes, M., Mens, T., & Grosjean, P. (2014). On the maintainability of CRAN packages. *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, 308–312. `https://doi.org/10.1109/CSMR-WCRE.2014.6747183`

Cornwell, W. K., Schwilk, D. W., & Ackerly, D. D. (2006). A Trait-Based Test for Habitat Filtering: Convex Hull Volume. *Ecology*, *87*(6), 1465–1471. `https://doi.org/10.1890/0012-9658(2006)87%5B1465:ATTFHF%5D2.0.CO;2`

Cox, R. (2019). Surviving software dependencies. *Communications of the ACM*, *62*(9), 36–43. `https://doi.org/10.1145/3347446`

Díaz, S., & Cabido, M. (2001). Vive la DiffÃ©rence: Plant functional diversity matters to ecosystem processes. *Trends in Ecology & Evolution*, *16*(11), 646–655. `https://doi.org/10.1016/S0169-5347(01)02283-2`

Farley, S. S., Dawson, A., Goring, S. J., & Williams, J. W. (2018). Situating Ecology as a Big-Data Science: Current Advances, Challenges, and Solutions. *BioScience*, *68*(8), 563–576. `https://doi.org/10.1093/biosci/biy068`

Farrell, K. J., & Carey, C. C. (2018). Power, pitfalls, and potential for integrating computational literacy into undergraduate ecology courses. *Ecology and Evolution*, *8*(16), 7744–7751. `https://doi.org/10.1002/ece3.4363`

Gotelli, N. J., & Graves, G. R. (1996). *Null models in ecology*. Smithsonian Institution Press.

Gower, J. C. (1971). A general coefficient of similarity and some of its properties. *Biometrics*, 857–871.

Hester, J., & Vaughan, D. (2021). *Bench: High precision timing of r expressions*. `https://CRAN.R-project.org/package=bench`

Lai, J., Lortie, C. J., Muenchen, R. A., Yang, J., & Ma, K. (2019). Evaluating the popularity of R in ecology. *Ecosphere*, *10*(1), e02567. `http://esajournals-onlinelibrary-wiley-com/doi/abs/10.1002/ecs2.2567`

Laliberté, E., & Legendre, P. (2010). A distance-based framework for measuring functional diversity from multiple traits. *Ecology*, *91*(1), 299–305. `https://doi.org/10.1890/08-2244.1`

Laliberté, E., Legendre, P., & Shipley, B. (2014). *FD: Measuring functional diversity from multiple traits, and other tools for functional ecology*.

Legras, G., Loiseau, N., & Gaertner, J. -C. (2018). Functional richness: Overview of indices and underlying concepts. *Acta Oecologica*, *87*, 34–44. `https://doi.org/10.1016/j.actao.2018.02.007`

Leps, J., Bello, F., Lavorel, S., & Berman, S. (2006). Quantifying and interpreting functional diversity of natural communities: Practical considerations matter. *Preslia*, *78*, 481–501.

Li, D. (2018). hillR: Taxonomic, functional, and phylogenetic diversity and similarity through Hill Numbers. *Journal of Open Source Software*, *3*(31), 1041. `https://doi.org/10.21105/joss.01041`

Magneville, C., Loiseau, N., Albouy, C., Casajus, N., Claverie, T., Escalas, A., Leprieur, F., Maire, E., Mouillot, D., & Villéger, S. (2022). mFD: An R package to compute and illustrate the multiple facets of functional diversity. *Ecography*, *2022*(1). `https://doi.org/10.1111/ecog.05904`

Maire, E., Grenouillet, G., Brosse, S., & Villéger, S. (2015). How many dimensions are needed to accurately assess functional diversity? A pragmatic approach for assessing the quality of functional spaces. *Global Ecology and Biogeography*, *24*(6), 728–740. `https://doi.org/10.1111/geb.12299`

Mason, N. W. H., de Bello, F., Mouillot, D., Pavoine, S., & Dray, S. (2013). A guide for using functional diversity indices to reveal changes in assembly processes along ecological gradients. *Journal of Vegetation Science*, *24*(5), 794–806. `https://doi.org/10.1111/jvs.12013`

McPherson, J. M., Yeager, L. A., & Baum, J. K. (2018). A simulation tool to scrutinise the behaviour of functional diversity metrics. *Methods in Ecology and Evolution*, *9*(1), 200–206. `https://doi.org/10.1111/2041-210X.12855`

Mislan, K. A. S., Heer, J. M., & White, E. P. (2016). Elevating The Status of Code in Ecology. *Trends in Ecology & Evolution*, *31*(1), 4–7. `https://doi.org/10.1016/j.tree.2015.11.006`

Nowak, L., Kissling, W. D., Bender, I. M. A., Dehling, D. M., Töpfer, T., Böhning-Gaese, K., & Schleuning, M. (2019). Data from: Projecting consequences of global warming for the functional diversity of fleshy-fruited plants and frugivorous birds along a tropical elevational gradient. In *Data Dryad Digital Repository* (pp. 264849 bytes). `https://doi.org/10.5061/DRYAD.C0N737B`

Pavoine, S. (2020). Adiv: An r package to analyse biodiversity in ecology. *Methods in Ecology and Evolution*, *11*(9), 1106–1112. `https://doi.org/10.1111/2041-210X.13430`

Pavoine, S., & Bonsall, M. B. (2011). Measuring biodiversity to explain community assembly: A unified approach. *Biological Reviews*, *86*(4), 792–812. `https://doi.org/10.1111/j.1469-185X.2010.00171.x`

Pavoine, S., Vallet, J., Dufour, A.-B., Gachet, S., & Daniel, H. (2009). On the challenge of treating various types of variables: Application for improving the measurement of functional diversity. *Oikos*, *118*(3), 391–402. `https://doi.org/10.1111/j.1600-0706.2008.16668.x`

Podani, J. (1999). Extending Gower's general coefficient of similarity to ordinal characters. *Taxon*, 331–340.

Poisot, T. (2015). Best publishing practices to improve user confidence in scientific software. *Ideas in Ecology and Evolution*, *8*. `https://doi.org/10.4033/iee.2015.8.8.f`

R Core Team. (2021). *R: A language and environment for statistical computing* [Manual]. R Foundation for Statistical Computing. `https://www.R-project.org/`

Rao, C. R. (1982). Diversity and dissimilarity coefficients: A unified approach. *Theoretical Population Biology*, *21*(1), 24–43. `https://doi.org/10.1016/0040-5809(82)90004-1`

rOpenSci, Anderson, B., Chamberlain, S., DeCicco, L., Gustavsen, J., Krystalli, A., Lepore, M., Mullen, L., Ram, K., Ross, N., Salmon, M., & Vidoni, M. (2021). *rOpenSci packages: Development, maintenance, and peer review* (Version 0.6.0) [Computer software]. Zenodo. `https://doi.org/10.5281/zenodo.4554776`

Schleuter, D., Daufresne, M., Massol, F., & Argillier, C. (2010). A user's guide to functional diversity indices. *Ecological Monographs*, *80*(3), 469–484. `https://doi.org/10.1890/08-2225.1`

Villéger, S., Grenouillet, G., & Brosse, S. (2013). Decomposing functional β-diversity reveals that low functional β-diversity is driven by low functional turnover in European fish assemblages. *Global Ecology and Biogeography*, *22*(6), 671–681. `https://doi.org/10.1111/geb.12021`

Villéger, S., Mason, N. W. H., & Mouillot, D. (2008). New Multidimensional Functional Diversity Indices for a Multifaceted Framework in Functional Ecology. *Ecology*, *89*(8), 2290–2301. `https://doi.org/10.1890/07-1206.1`

White, E. (2015). Some thoughts on best publishing practices for scientific software. *Ideas in Ecology and Evolution*, *8*. `https://doi.org/10.4033/iee.2015.8.9.c`

Wickham, H., Hester, J., Chang, W., Müller, K., & Cook, D. (2021). *Memoise: Memoisation of functions* [Manual]. `https://CRAN.R-project.org/package=memoise`

Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., & Teal, T. K. (2017). Good enough practices in scientific computing. *PLOS Computational Biology*, *13*(6), e1005510. `https://doi.org/10.1371/journal.pcbi.1005510`

Wüest, R. O., Zimmermann, N. E., Zurell, D., Alexander, J. M., Fritz, S. A., Hof, C., Kreft, H., Normand, S., Cabral, J. S., Szekely, E., Thuiller, W., Wikelski, M., & Karger, D. N. (2020). Macroecology in the age of Big Data – Where to go from here? *Journal of Biogeography*, *47*(1), 1–12. `https://doi.org/10.1111/jbi.13633`

List of Figures

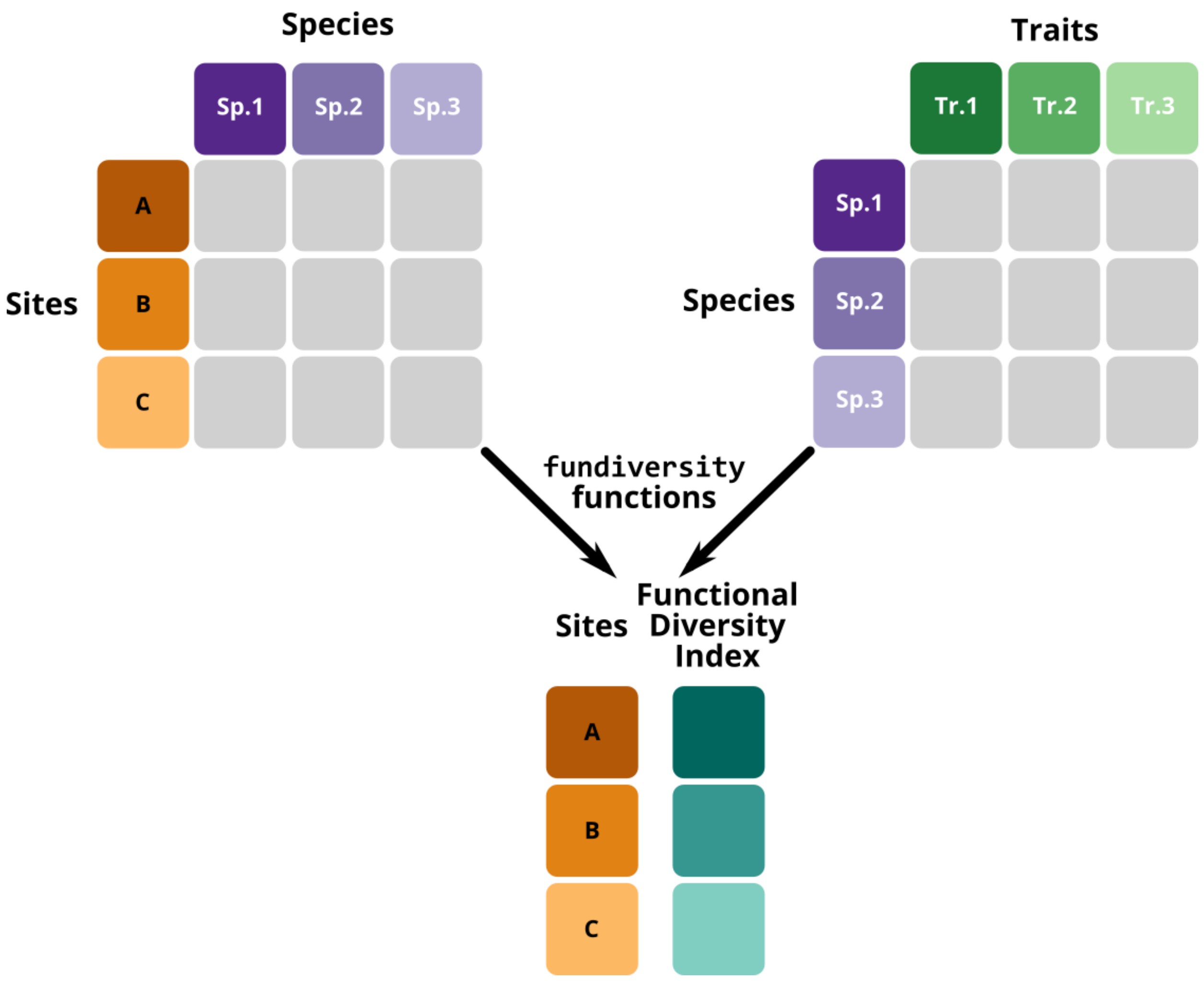

Figure 1: Conceptual diagram showing the input and typical ouput data from ‘fundiversity‘ functions. Input data are generally a site-species table and a species-traits table, and the output gives back a table of functional diversity index per site.

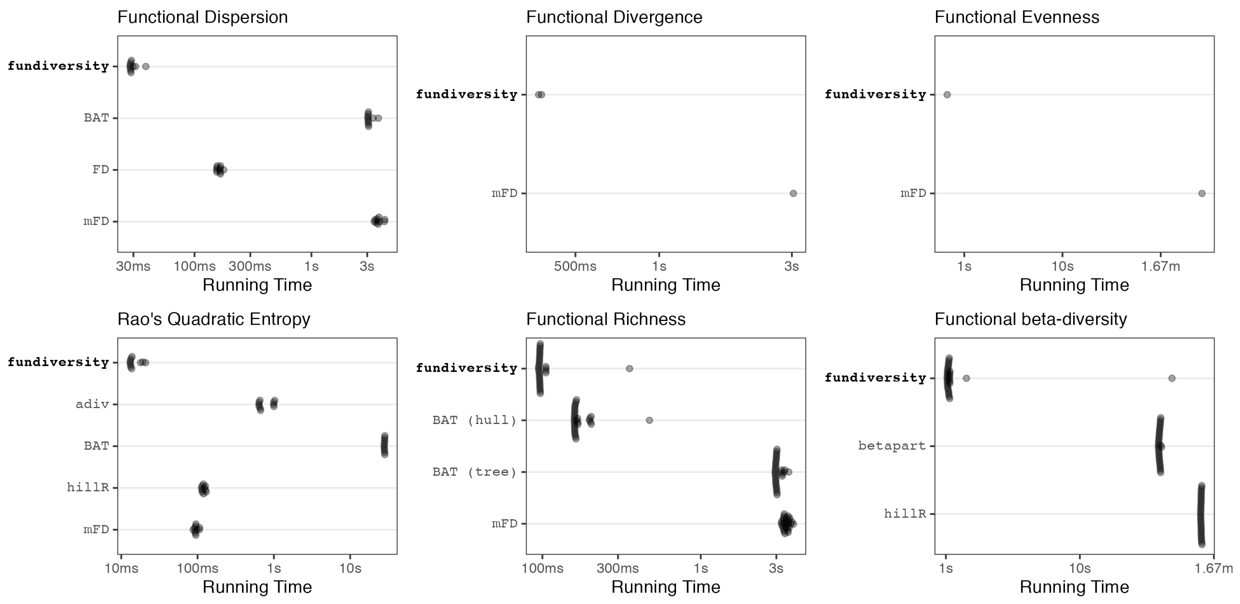

Figure 2: Timing comparison across functional diversity indices between packages. Each point represents the execution time of one run using a simulated dataset, the points are transparent and jittered to avoid overplotting. We here show the performance results considering only a single set of parameters with 4 traits, 500 species, and 100 sites, repeated 30 times.

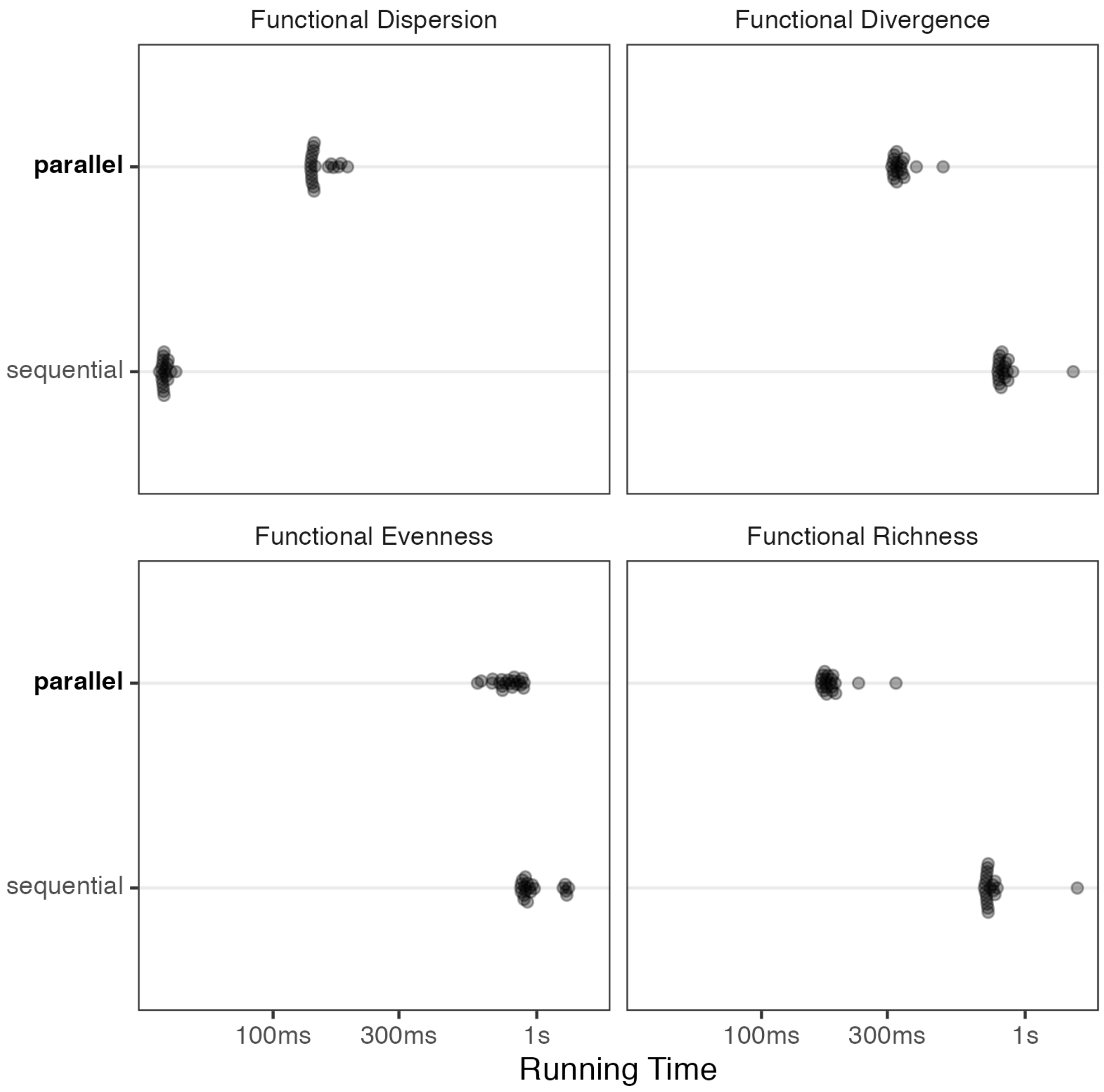

Figure 3: Timing comparison between parallel and sequential version of fundiversity functions across functional diversity indices. Each point represents the execution time of one run using simulated datasets with fixed properties (4 traits, 100 sites, 500 species), the points are transparent and jittered to avoid overplotting. The parallel version ran across 6 cores.

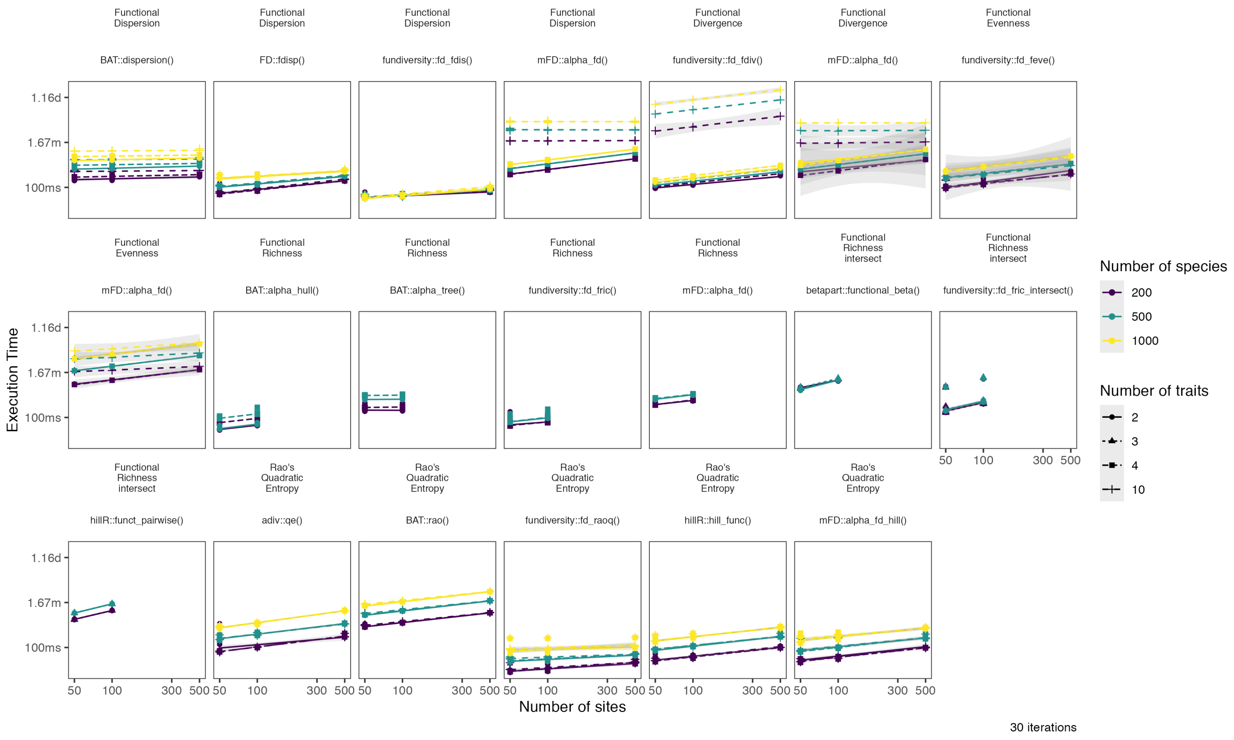

Figure 4: Performance comparison across functions of different packages over a range of parameters (number of traits, species, and sites). Note that each combination of parameters ran 30 iterations. The lines show trends of execution time in function of number of sites of the input dataset.

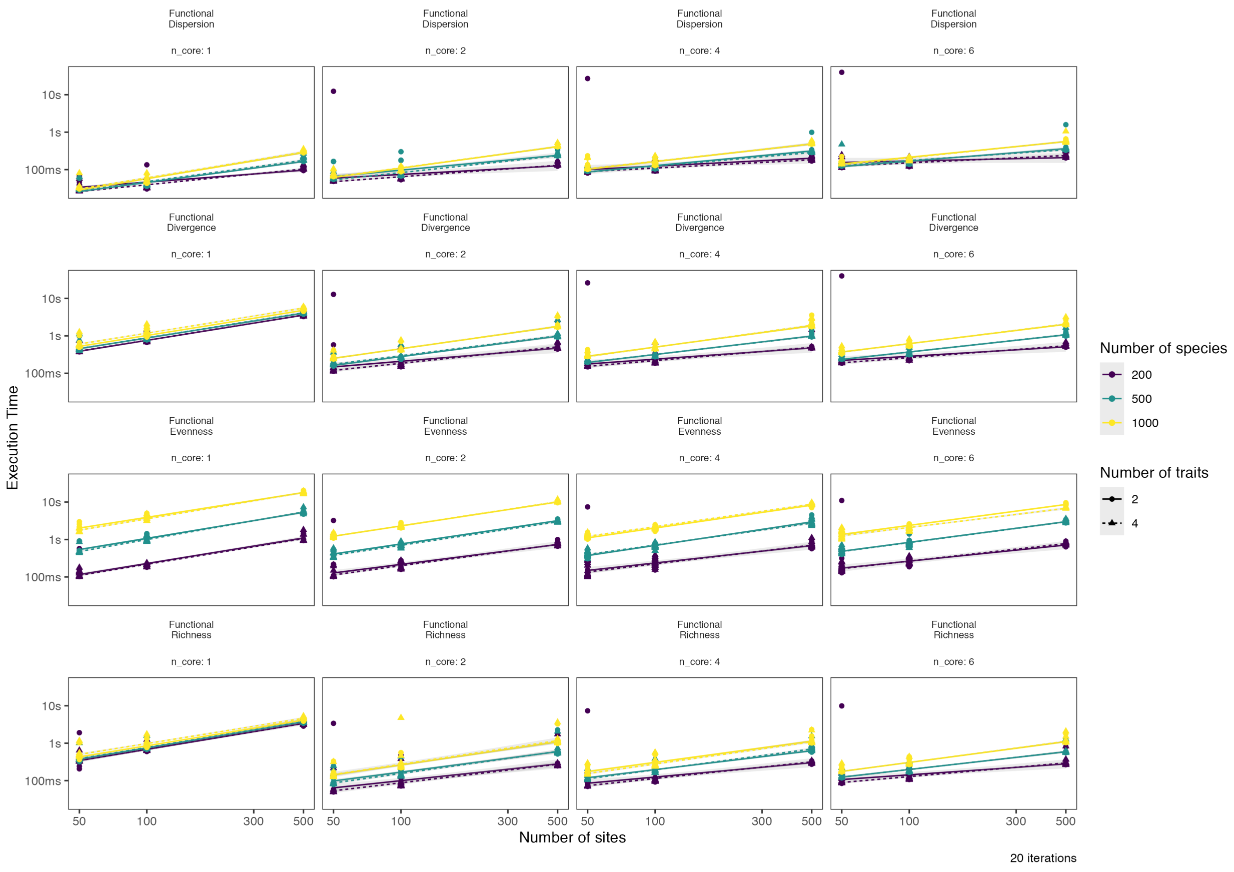

Figure 5: Performance comparison across internal functions over a range of parameters (number of traits, species, and sites) and different parallelization parameters. Note that each combination of parameters ran 20 iterations. The lines show trends of execution time in function of number of sites of the input dataset.