Appendix 4: Test of the robustness of the T-statistics using simulations

Adrien Taudiere*and Cyrille Violle

CEFE - Centre d'Ecologie Fonctionnelle et Evolutive, Montpellier: France

February 21, 2015

Abstract

Community ecologists are actively describing species by their functional traits, quantifying the functional structure of plant and animal assemblages, and inferring community assembly processes with null model analyses of trait distributions and functional diversity indices. Intraspecific variation in traits and effects of spatial scale are potentially important in these analyses.

We introduce the R package cati, available on CRAN, for the analysis of community assembly with functional traits. The cati package (i) calculates a variety of single-trait and multi-trait indices from interspecific and intraspecific trait measures; (ii) partitions functional trait variation among spatial and taxonomic levels; (iii) implements a palette of flexible null models for detecting non-random patterns of functional traits. These patterns can be used to draw inferences about hypotheses of community assembly such as environmental filtering and negative species interactions.

The basic input for cati is a data frame in which columns are traits, rows are species or individuals, and entries are the measured trait values. The cati package can also incorporate into analyses a square distance matrix, which could include phylogenetic or genetic distances among individuals or species. Users select from a variety of functional trait metrics and analyze them relative to a null model that specifies trait distributions in a regional source pool.

Key words: Functional space, functional structure, community assembly, ecological niche, environmental filter, individual differences, intraspecific variation, null model, trait, variance decomposition

To read a summary of this appendix, see directly section Summary 7. An up to date version of the cati tutorial is available here.

*adrien.taudiere@cefe.cnrs.fr

Contents

1 Introduction

This document presents the lack of bias in the functions `Tstats`, `partvar`, `decompCTRE` and a test of robustness for the T-statistics. A summary of the main results is available at the end of each section.

1.1 R requirements

First we need to install and load the package.

```
install.packages("cati", repos = "http://cran.rstudio.com/")
library("cati")
library("xtable")

# Save the graphical parameter
oldpar <- par(no.readonly = TRUE)
```

1.2 System and session informations

This document was created with R version 3.1.2 (2014-10-31) on Linux. See below for more information. The speed of computations of `cati` main functions are given in the cati tutorial.

```
sessionInfo()

## R version 3.1.2 (2014-10-31)
## Platform: x86_64-pc-linux-gnu (64-bit)
##
## locale:
##  [1] LC_CTYPE=fr_FR.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=fr_FR.UTF-8        LC_COLLATE=fr_FR.UTF-8
##  [5] LC_MONETARY=fr_FR.UTF-8    LC_MESSAGES=fr_FR.UTF-8
##  [7] LC_PAPER=fr_FR.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=fr_FR.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] xtable_1.7-4 cati_0.96    ape_3.2      ade4_1.6-2   nlme_3.1-119
## [6] knitr_1.9
##
## loaded via a namespace (and not attached):
##  [1] class_7.3-11        cluster_2.0.1       codetools_0.2-10
##  [4] digest_0.6.8        e1071_1.6-4         evaluate_0.5.5
##  [7] FD_1.0-12           formatR_1.0         geometry_0.3-5
## [10] grid_3.1.2          hexbin_1.27.0       highr_0.4
## [13] hypervolume_1.1.2   lattice_0.20-29     latticeExtra_0.6-26
## [16] magic_1.5-6         MASS_7.3-37         Matrix_1.1-5
## [19] mgcv_1.8-3          mice_2.22           nnet_7.3-8
## [22] parallel_3.1.2      pdist_1.2           permute_0.8-3
```

```
## [25] randomForest_4.6-10 raster_2.3-24       rasterVis_0.32
## [28] RColorBrewer_1.1-2  Rcpp_0.11.4         rgl_0.93.996
## [31] rpart_4.1-8         sp_1.0-17           stringr_0.6.2
## [34] tools_3.1.2         vegan_2.2-1         zoo_1.7-11
```

1.3 T-statistics and associated null models

Table 1: The four types of null models implemented in cati, their related null and alternative hypotheses, randomization design and associated T-statistics

	Null hypothesis	Randomization procedure	Unilateral alternative hypothesis	T-statistics
local	There is no internal filtering: the distribution of trait values of all individuals within a given community does not depend on species identity	Randomization of individual trait values within the community	Internal filtering significantly impacts the distribution of trait values within a given community: two individuals belonging to a population have more similar trait values than two individuals randomly-drawn in the community	$T_{IP/IC}$
regional.ind	There is no external filtering: the distribution of trait values of individuals within a given community is a random drawing from the regional pool	Draw without replacement of individual trait values belonging to the regional pool (keeping the actual number of individuals in each community)	Two individuals belonging to a community have more similar trait values than two individuals randomly-drawn in the regional pool	$T_{IC/IR}$
regional.pop & regional.pop.prab	There is no species-based external filtering: the distribution of mean trait values of species within a given community is a random draw from the regional pool	(i) Assignment of a population-level value to each individual and (ii) Draw without replacement of population-level trait values belonging to the regional pool (keeping the actual number of individuals in each community (regional.pop) or not (regional.pop .prab).	Two individuals belonging to a community have more similar population-based trait values than two individuals randomly-drawn in the regional pool with (regional.pop) or without (regional.pop .prab) taking abundance into account.	$T_{PC/PR}$ (regional.pop)

1.4 Simulations parameters

Here we set the number of permutations for the entire analysis.

```
#Number of permutations for the analysis without filter
npermut <- 1000

#Number of permutations for each parameter values in the analysis with filter(s)
N_repet_Param <- 500

#Number of values for each parameters
nb_param_val <- 10
```

Now we set the number of individuals, species and communities. The argument sd log allows to modify the species abundance distribution within communities.

```
#Total number of individuals
Nind <- 1000

#Ten communities named A, B, C ... J
com <- c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J")
Ncom <- length(com)

#Number of species (if Nsp > 676 you need to defined additional name)
Nsp <- 20
sp <- as.factor(paste("sp", paste(sort(rep(LETTERS, 26)), rep(LETTERS, 26),  sep="")
                      [seq(1:Nsp)], sep ="_"))
Nsp <- length(sp)

#Parameter of the log normal distribution for species abundances
#distribution within communities
sdlog = 1.5
```

We use toys models of 1000 individuals belonging to 20 species occuring in 10 communities (sites). Each community contains 100 individuals and species abundances in each community follow a lognormal distribution of mean 0 and standard deviation `sdlog` of 1.5. Figure 1 depicts the general outline of the next fourth sections dealing with the statistical test of T-statistics from Violle *et al.* (2012).

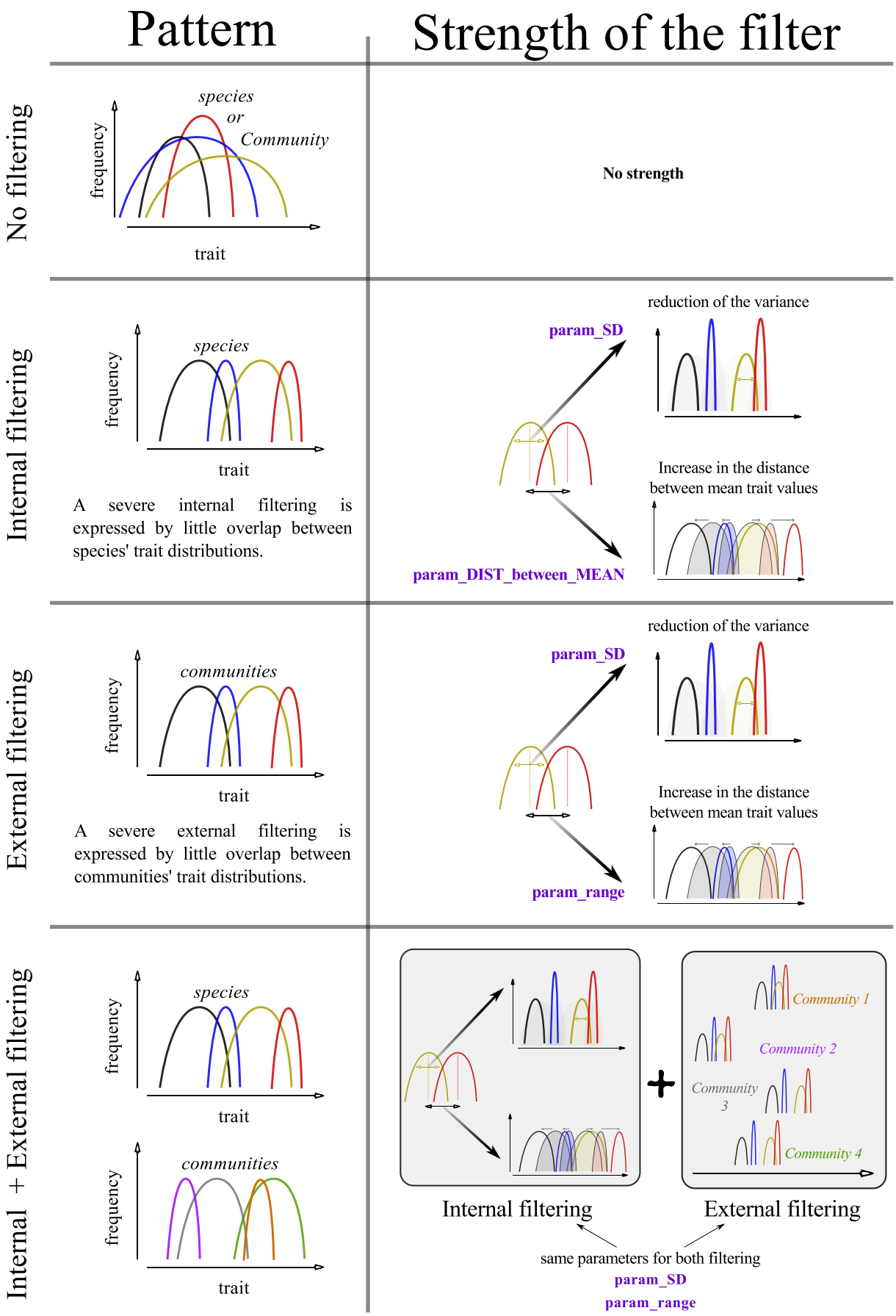


Figure 1: An approach by simulations to evaluate the power of the T-statistics to detect internal and external filtering. (Continue on next page)

Figure 1 (continue): The **lack of filtering** allows to test the type I error (alpha-error) of the three Tstatistics. In the absence of filtering the expected pattern is a random distribution of individual traits values among species and communities. **Internal filter** includes all of the assembly processes internal to the community. The alternative hypothesis is the non-overlap of trait distribution curves between species, as measured by the ratio $T_{IP/IC}$. We modulate the strength of the filter in the simulation using the variance of species' trait distributions (`param_SD`) and the distance between species trait means (`param_DIST_between_MEAN`). **External filter** includes all of the assembly processes external to the community. In this document we specifically treat the case of an environnemental gradient influencing the individual traits values. Thus the expected pattern is a non-overlap between communities' trait distribution curves, as measured by the ratio $T_{IC/IR}$. We modulate the strength of external filter in the simulation using the variance of communities's trait distributions (`param_SD`) and the range of communities' trait means (the "extent" of the environnemental gradient; (`param_range`)). Finally, we mix external and internal filter. First, we simulate the effect of internal filter alone. Second, we add to each individual trait value a number depending on its community. Again two parameters are used to modulate the strength of the filters (*cf* section 5 for more details).

2 No filter: calculation of error

2.1 Randomization outline

To calculate the **alpha error** (*i.e.* the probability to reject the null hypothesis H0 while H0 is true), we draw 1000 random trait matrices irrespective of the species and community attributes of each individual.

```
#Start simulation
res.simu1 <- list() ; res.simu1.pval <- list()

for(n in 1:npermut){#for each permutation

 ex.traits1 <- array(NA, dim = c(Nind, 2))
 colnames(ex.traits1) <- paste("trait", c("a", "b"), sep = " ")

 #trait a: normal distribution
 ex.traits1[, 1] <- rnorm(Nind, rlnorm(Nind, 0, 1), rlnorm(Nind, 0, 1))

 #trait b: uniform distribution
 ex.traits1[, 2] <- runif(Nind)

 # Draw communities using lognormal distribution of abundances
 ex.sp1 <- c()
 ex.com1 <- matrix(0, nrow = Ncom, ncol = Nsp)
 for(c in 1: Ncom){
  ex.com.interm <- table(sample(sp, size = Nind / Ncom,
                                prob = rlnorm(Nsp, 0, sdlog), replace = T))

  ex.com1[c, sp %in% names(ex.com.interm)] <- ex.com.interm

  ex.sp1 <- c(ex.sp1, rep(sp, times = ex.com1[c,]))
 }
 ex.indplot1 <- sort(as.factor(rep(com, Nind / Ncom)))

 #Stock the results
 res.simu1[[n]] <- Tstats(ex.traits1, ex.indplot1, ex.sp1)
 res.simu1.pval[[n]] <- sum_Tstats(res.simu1[[n]], type = "p.value")
 print(paste("---", round(n/npermut, 2) * 100, "%", sep = " "))
}#End of simulations
```

2.2 Results of the simulations without filtering

Let's see the result for one randomization. We can plot the distribution of traits values within species and/or communities thanks to the function `plotDistri` (Fig. 2). We can also plot the result of the T-statistics for one of the 1000 permutations (Fig. 3).

```
par(mfrow=c(2, 2))
plotDistri(ex.traits1, rep("all_sp", times = dim(ex.traits1)[1]), ex.indplot1,
           plot.ask = F, multipanel = F, leg =c(T, F), main = c("a", "b"))
plotDistri(ex.traits1, rep("region", times = dim(ex.traits1)[1]), ex.sp1,
           plot.ask = F, multipanel = F, leg =c(T, F), main = c("c", "d"))
```

```r
par(mfrow=c(1, 1))
```

```r
plot(res.simu1[[1]])
```

2.3 Assessing the statistical type I error of local T-statistics

Here we call local p-values the p-values [1] corresponding to one index for one trait in one community [2]. Now let's plot the ordered p.value for each T-statistics and each trait (Fig. 4)). In that case we have 20 p-values [3] per T-statistics and per trait for each permutation. Using the defined parameters, we plot 2×10^4 points per T-statistics by trait (20 p.values * 1000 permutations).

The alpha error on the figure are the proportion of p.values which are inferior to 0.025 (because performed a bilateral test).

```r
par(mfrow = c(3, 2))
par(mar = c(3, 3.5, 2, 0.2))

xx <- log10(sort(unlist(lapply(res.simu1.pval, function(x) x[1:20, 1]))))
plot(xx, type = "l", main = "T_IP.IC norm", ylab = NA , xlab = NA)
abline(h = log10(0.025))
nbre_alpha_error_norm_loc_Tipic <- round((sum(xx < log10(0.025)) +1 ) / (length(xx)+1), 5)
text(0, -0.3, labels = paste("alpha error", nbre_alpha_error_norm_loc_Tipic, sep = " = "),
     pos = 4)
mtext("log10 pvalue", 2, line = 2)

xx <- log10(sort(unlist(lapply(res.simu1.pval, function(x) x[1:20, 2]))))
plot(xx, type = "l", main = "T_IP.IC Uni", ylab = NA, xlab = NA)
abline(h = log10(0.025))
nbre_alpha_error_uni_loc_Tipic <- round((sum(xx < log10(0.025)) +1 ) / (length(xx)+1), 5)
text(0, -0.3, labels = paste("alpha error", nbre_alpha_error_uni_loc_Tipic, sep = " = "),
     pos = 4)

xx <- log10(sort(unlist(lapply(res.simu1.pval, function(x) x[21:40, 1]))))
plot(xx, type = "l", main = "T_IC.IR norm", ylab = NA, xlab = NA)
abline(h = log10(0.025))
nbre_alpha_error_norm_loc_Ticir <- round((sum(xx < log10(0.025)) +1 ) / (length(xx)+1), 5)
text(0, -0.3, labels = paste("alpha error", nbre_alpha_error_norm_loc_Ticir, sep = " = "),
     pos = 4)
mtext("log10 pvalue", 2, line = 2)

xx <- log10(sort(unlist(lapply(res.simu1.pval, function(x) x[21:40, 2]))))
plot(xx, type = "l", main = "T_IC.IR Uni", ylab = NA, xlab = NA)
abline(h = log10(0.025))
nbre_alpha_error_uni_loc_Ticir <- round((sum(xx < log10(0.025)) +1 ) / (length(xx)+1), 5)
text(0, -0.3, labels = paste("alpha error", nbre_alpha_error_uni_loc_Ticir, sep = " = "),
     pos = 4)
```

[1] Computed using the `Tstats` function

[2] In contrast with global statistics which aggregate the values of the metrics of all communities

[3] 10 p-value, corresponding to 10 communities, multiplied by 2 because we performed a bilateral test

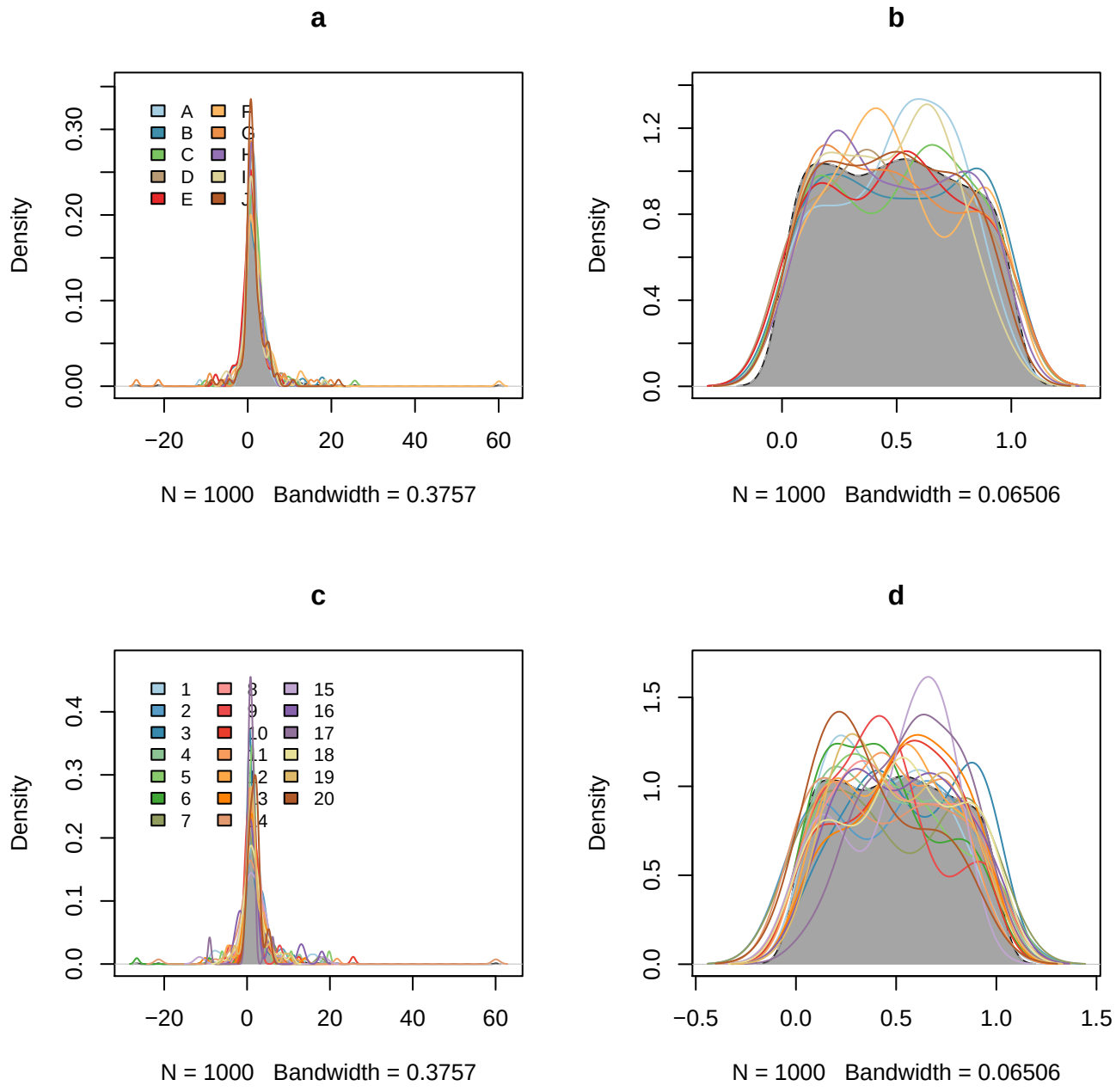


Figure 2: Distribution of trait values for one randomization without filtering: (a) Community-level trait distributions for the trait a (normal distribution); (b) Community-level trait distributions for the trait b (uniform distribution). In panels a and b, each color represents one community (site). (c) Species' trait distributions for the trait a; (d) Species' trait distributions for the trait b. In panels c and d, each color represents one species.

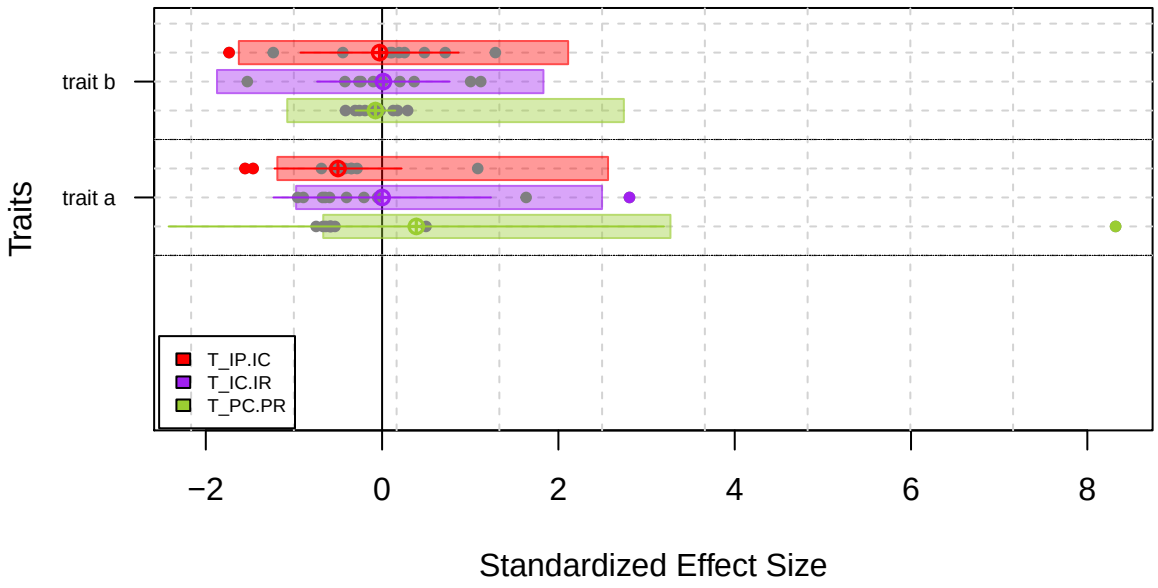


Figure 3: T-statistics for the first randomization without filtering: $T_{IP/IC}$ in red, $T_{IC/IR}$ in purple and $T_{PC/PR}$ in green. All but three community-level metrics (colored dots) depart significantly from the null model. As expected under no filtering, the mean Standardized Effect Size (SES) values are closed to zero for all three T-statistics and are included in the area of the null model (colored boxes). Thus this plot is typically what we expected under the lack of both external and internal filtering.

```
xx <- log10(sort(unlist(lapply(res.simu1.pval, function(x) x[41:60, 1]))))
plot(xx, type = "l", main = "T_PC.PR norm", ylab = NA, xlab = NA)
abline(h = log10(0.025))
nbre_alpha_error_norm_loc_Tpcpr <- round((sum(xx < log10(0.025)) +1 ) / (length(xx)+1), 5)
text(0, -0.3, labels = paste("alpha error", nbre_alpha_error_norm_loc_Tpcpr, sep = " = "),
     pos = 4)
mtext("log10 pvalue", 2, line = 2)
mtext("rank", 1, line = 2)

xx <- log10(sort(unlist(lapply(res.simu1.pval, function(x) x[41:60, 2]))))
plot(xx, type = "l", main = "T_PC.PR Uni", ylab = NA, xlab = NA)
abline(h = log10(0.025))
nbre_alpha_error_uni_loc_Tpcpr <- round((sum(xx < log10(0.025)) +1 ) / (length(xx)+1), 5)
text(0, -0.3, labels = paste("alpha error", nbre_alpha_error_uni_loc_Tpcpr, sep = " = "),
     pos = 4)
mtext("rank", 1, line = 2)

par(oldpar)
```

2.4 Assessing the statistical type I error of global T-statistics

In contrast with local p-values, we call global p-values the p-values corresponding to one index for a given trait across all the communities.

We test for the **alpha error** of global p-values is similar to the test for local p-values. We plot the

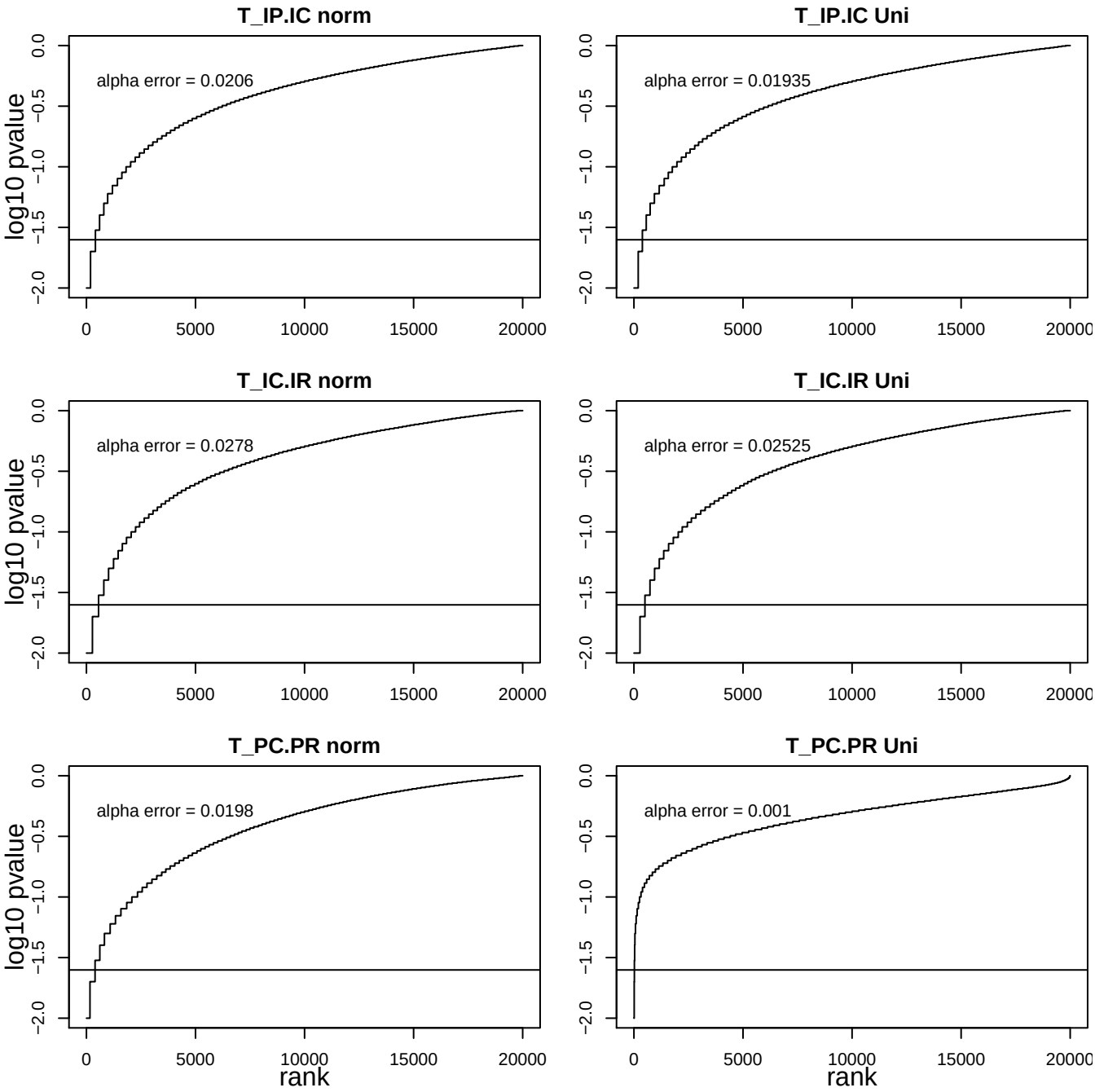


Figure 4: Local alpha errors: local p.values are log-transformed and ordered before plotting. The black line represent the value of 0.025. Consequently, p-values under this line are false-positive cases. Alpha-error is the proportion of these false-positive cases. Norm: normal distribution, Uni: uniform distribution.

ordered mean [4] standardized effect size (SES)[5] and the 95% confidence interval in grey.

In that case we have one p-value by T-statistics by traits for each of the 1000 permutations.

```
meanSES.1.T_IP.IC.distriNorm <- lapply(res.simu1, function(x)
          mean(ses.listofindex(as.listofindex(x))
           $index_1_1$ses[,1], na.rm = T))

meanSES.1.T_IC.IR.distriNorm <- lapply(res.simu1, function(x)
          mean(ses.listofindex(as.listofindex(x))
           $index_1_2$ses[,1], na.rm = T))

meanSES.1.T_PC.PR.distriNorm <- lapply(res.simu1, function(x)
          mean(ses.listofindex(as.listofindex(x))
           $index_1_3$ses[,1], na.rm = T))

meanSES.1.T_IP.IC.distriUni <- lapply(res.simu1, function(x)
          mean(ses.listofindex(as.listofindex(x))
           $index_1_1$ses[,2], na.rm = T))

meanSES.1.T_IC.IR.distriUni <- lapply(res.simu1, function(x)
          mean(ses.listofindex(as.listofindex(x))$
            index_1_2$ses[,2], na.rm = T))

meanSES.1.T_PC.PR.distriUni <- lapply(res.simu1, function(x)
          mean(ses.listofindex(as.listofindex(x))
           $index_1_3$ses[,2], na.rm = T))
```

Now, plot the result as ordered SES values (Fig. 5). Horizontal lines represent the confidence area at 95%.

```
par(mfrow = c(3, 2))
par(mar = c(3, 3.5, 2, 0.2))

####   T_IP.IC
#T_IP.IC.distriNorm
xx <- sort(unlist(meanSES.1.T_IP.IC.distriNorm))
xx_lim.inf <- unlist(lapply(res.simu1, function(x) mean(ses.listofindex(as.listofindex(x))
              $index_1_1$ses.inf[,1], na.rm = T)))
xx_lim.sup <- unlist(lapply(res.simu1, function(x) mean(ses.listofindex(as.listofindex(x))
              $index_1_1$ses.sup[,1], na.rm = T)))
alpha1 <- round((sum(xx<xx_lim.inf | xx>xx_lim.sup)+1) / (length(xx)+1), 3)

plot(xx, type = "l", main = "T_IP.IC norm", ylim = c(-3, 3),
     ylab = NA, xlab = NA)
rect(-100, mean(xx_lim.inf), npermut*1.2, mean(xx_lim.sup), col = rgb(0, 0, 0, 0.2))
text(3, 0.8, labels = paste("alpha error", alpha1, sep = " = "), cex = 0.7, pos = 4)
mtext("Standardized Effect Size", 2, line = 2)

#T_IP.IC.distriUni
```

[4] mean of the 10 communities values

[5] compute as (the observed value - the mean value among simulation) / the standard error among simulations

```
xx <- sort(unlist(meanSES.1.T_IP.IC.distriUni))
xx_lim.inf <- unlist(lapply(res.simu1, function(x) mean(ses.listofindex(as.listofindex(x))
              $index_1_1$ses.inf[,2], na.rm = T)))
xx_lim.sup <- unlist(lapply(res.simu1, function(x) mean(ses.listofindex(as.listofindex(x))
              $index_1_1$ses.sup[,2], na.rm = T)))
alpha2 <- round((sum(xx<xx_lim.inf | xx>xx_lim.sup)+1) / (length(xx)+1), 3)

plot(xx, type = "l", main = "T_IP.IC uni", ylim = c(-3, 3),
     ylab = NA, xlab = NA)
rect(-100, mean(xx_lim.inf), npermut*1.2, mean(xx_lim.sup), col = rgb(0, 0, 0, 0.2))
text(3, 0.8, labels = paste("alpha error", alpha2, sep = " = "), cex = 0.7, pos = 4)

#__________
####   T_IC.IR
#T_IC.IR.distriNorm
xx <- sort(unlist(meanSES.1.T_IC.IR.distriNorm))

xx_lim.inf <- unlist(lapply(res.simu1, function(x) mean(ses.listofindex(as.listofindex(x))
              $index_1_2$ses.inf[,1], na.rm = T)))
xx_lim.sup <- unlist(lapply(res.simu1, function(x) mean(ses.listofindex(as.listofindex(x))
              $index_1_2$ses.sup[,1], na.rm = T)))
alpha3 <- round((sum(xx<xx_lim.inf | xx>xx_lim.sup)+1) / (length(xx)+1), 3)

plot(xx, type = "l", main = "T_IC.IR norm", ylim = c(-3, 3),
     ylab = NA, xlab = NA)
rect(-100, mean(xx_lim.inf), npermut*1.2, mean(xx_lim.sup), col = rgb(0, 0, 0, 0.2))
text(3, 0.8, labels = paste("alpha error", alpha3, sep = " = "), cex = 0.7, pos = 4)
mtext("Standardized Effect Size", 2, line = 2)

#T_IC.IR.distriUni
xx <- sort(unlist(meanSES.1.T_IC.IR.distriUni))

xx_lim.inf <- unlist(lapply(res.simu1, function(x) mean(ses.listofindex(as.listofindex(x))
              $index_1_2$ses.inf[,2], na.rm = T)))
xx_lim.sup <- unlist(lapply(res.simu1, function(x) mean(ses.listofindex(as.listofindex(x))
              $index_1_2$ses.sup[,2], na.rm = T)))
alpha4 <- round((sum(xx<xx_lim.inf | xx>xx_lim.sup)+1) / (length(xx)+1), 3)

plot(xx, type = "l", main = "T_IC.IR uni", ylim = c(-3, 3),
     ylab = NA, xlab = NA)
rect(-100, mean(xx_lim.inf), npermut*1.2, mean(xx_lim.sup), col = rgb(0, 0, 0, 0.2))
text(3, 0.8, labels = paste("alpha error", alpha4, sep = " = "), cex = 0.7, pos = 4)

#__________
####   T_PC.PR
#T_PC.PR.distriNorm
xx <- sort(unlist(meanSES.1.T_PC.PR.distriNorm))

xx_lim.inf <- unlist(lapply(res.simu1, function(x) mean(ses.listofindex(as.listofindex(x))
              $index_1_3$ses.inf[,1], na.rm = T)))
xx_lim.sup <- unlist(lapply(res.simu1, function(x) mean(ses.listofindex(as.listofindex(x))
```

```
              $index_1_3$ses.sup[,1], na.rm = T)))
alpha5 <- round((sum(xx<xx_lim.inf | xx>xx_lim.sup)+1) / (length(xx)+1), 3)

plot(xx, type = "l", main = "T_PC.PR norm", ylim = c(-3, 3),
     ylab = NA, xlab = NA)
rect(-100, mean(xx_lim.inf), npermut*1.2, mean(xx_lim.sup), col = rgb(0, 0, 0, 0.2))
text(3, 0.8, labels = paste("alpha error", alpha5, sep = " = "), cex = 0.7, pos = 4)
mtext("Standardized Effect Size", 2, line = 2)
mtext("rank", 1, line = 2)

#T_PC.PR.distriUni
xx <- sort(unlist(meanSES.1.T_PC.PR.distriUni))
xx_lim.inf <- unlist(lapply(res.simu1, function(x) mean(ses.listofindex(as.listofindex(x))
              $index_1_3$ses.inf[,2], na.rm = T)))
xx_lim.sup <- unlist(lapply(res.simu1, function(x) mean(ses.listofindex(as.listofindex(x))
              $index_1_3$ses.sup[,2], na.rm = T)))
alpha6 <- round((sum(xx<xx_lim.inf | xx>xx_lim.sup)+1) / (length(xx)+1), 3)

plot(xx, type = "l", main = "T_PC.PR uni", ylim = c(-3, 3),
     ylab = NA, xlab = NA)
rect(-100, mean(xx_lim.inf), npermut*1.2, mean(xx_lim.sup), col = rgb(0, 0, 0, 0.2))
text(3, 0.8, labels = paste("alpha error", alpha6, sep = " = "), cex = 0.7, pos = 4)
mtext("rank", 1, line = 2)

par(oldpar)
```

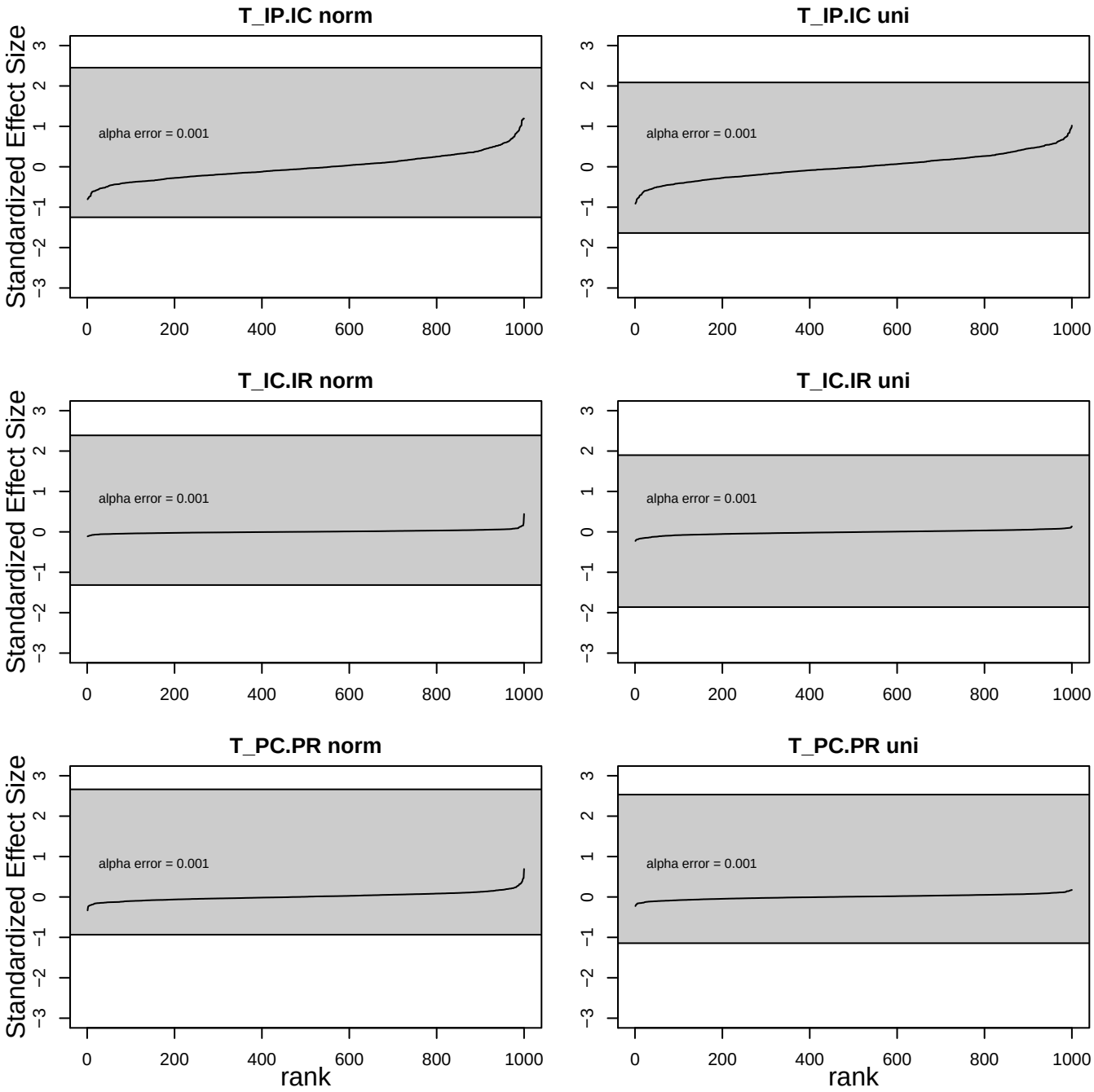


Figure 5: Global alpha errors: global Standardized Effect Sizes are ordered before plotting. The grey area represents the confidence interval at 0.025 on each side. Consequently, p-values outside this area are false-positive cases. Alpha-error is the proportion of these false-positive cases. Norm: normal distribution, Uni: uniform distribution.

2.5 Conclusion on the type I error of T-statistics

The 1000 permutations allow the calculation of alpha error for two traits (the first normally distributed and the second uniformally distributed), three indices (the three T-statistics) and using local indices (one by community) or global indices (averaged across communities).

Table 2: Alpha-errors for T-statistics

Traits	Indices	Averaged across communities?	alpha-error
Trait normally distributed	$T_{IP/IC}$	no (local)	0.021
		yes (global)	0.001
	$T_{IC/IR}$	no (local)	0.028
		yes (global)	0.001
	$T_{PC/PR}$	no (local)	0.02
		yes (global)	0.001
Trait uniformally distributed	$T_{IP/IC}$	no (local)	0.019
		yes (global)	0.001
	$T_{IC/IR}$	no (local)	0.025
		yes (global)	0.001
	$T_{PC/PR}$	no (local)	0.001
		yes (global)	0.001

The results of table 2 show of a good behavior of the T-statistics regardless of the distributions of the traits. Alpha-error of the local indices are always near the desired alpha-error = 2.5%. Logically the global indices which averaged local indices are far more robust. Note that the higher alpha-error is obtained using $T_{PC/PR}$ on the trait normally distributed (0.02).

3 Internal filter: assessing the statistical power of $T_{IP/IC}$

3.1 Randomization outline

We decide to modulate the strength of the internal filter thanks to two parameters (*cf* figure 1): we define (i) a gradient of distance between species' mean trait distributions and (ii) a gradient of the mean variance in species' mean trait distribution. It is strongly linked to the definition of limiting similarity by May and Mac Arthur (1972) who defined a coefficient of competition as

$$a = exp(-d^2/4w^2)$$

with `d` corresponds to the distance between the species' mean traitvalues and `w` corresponds to the averaged variance of species' trait distributions.

```
# Number of values for each parameter: param_DIST_between_MEAN.init and param_SD
nb_param_val

## [1] 10

# Number of permutations is the product of the number of values for each parameter
#(nb_param_val) and the number of permutations by parameter (N_repet_Param)
nperm <- nb_param_val * N_repet_Param

# Maximum mean value for traits
max.value_traits <- 250

# Parameter for the distance between species' mean trait values
param_DIST_between_MEAN.init <- round(sort(seq(10, 200, length.out = nb_param_val)), 2)
param_DIST_between_MEAN <- rep(param_DIST_between_MEAN.init, N_repet_Param)
mean_sd.param <- rep(10, nperm)

# Parameter for the variance in species' mean trait values
param_SD <- seq(10, 100,length.out = nb_param_val)
sd_mean.param <- sample (rep(param_SD, N_repet_Param),
       size = length(rep(param_SD, N_repet_Param)), replace = F)
sd_sd.param <- rep(10, nperm)
```

Practically, we defined 10 values for two parameters:

(i) `param_DIST_between_MEAN.init` is a vector of length 10 which defines the minimum value for the mean trait values (the maximum value is 250). In decreasing the range of traits values while keeping the same number of species, there is a decrease in the mean distance between species' means of trait values. Thus the mean for each species is drawn from a normal distribution with standard deviation of 10 (parameter `mean_sd.param`) and means evenly distributed between `max.value_traits - param_DIST_between_MEAN` and `max.value_traits`.

(ii) `param_SD` is a vector of length 10 which defined the standard deviation of trait distributions for each species. In order to decorrelate the mean and standard deviation of species' trait distributions, `param_SD` is permuted before the analysis.

For the trait "a" the trait value for each individual is drawn from a normal distribution with the mean depending on it species' attribute. For the trait "b" the trait value for each individual is drawn

from a uniform distribution with the range depending on it species' attribute. The range is defined as: *species' mean value - species' sd value* for the minimum and *species' mean value + species' sd value* for the maximum.

Now we can start the simulations with 10 parameter values and 5000 permutations.

```
mean.sp_stock2 <- list() ; sd.sp_stock2 <- list() ; res.simu2 <- list()
res.simu2.pval <- list() ; res.simu.traits2 <- list() ;  mean_dist_sp <- list()

for(n in 1:nperm){#for each permutation

 # Draw communities using lognormal distribution of abundances
 ex.sp2 <- c()
 ex.com2 <- matrix(0, nrow = Ncom, ncol = Nsp)
 for(i in 1: Ncom){
  ex.com.interm <- table(sample(sp, size = Nind/Ncom, prob = rlnorm(Nsp, 0, sdlog),
                                replace = T))
  ex.com2[i, sp%in% names(ex.com.interm)] <- ex.com.interm
  ex.sp2 <- c(ex.sp2, rep(sp, times = ex.com2[i,]))
 }

 ex.indplot2 <- sort(as.factor(rep(com, Nind/Ncom)))

 # Defining trait mean and sd per species
 mean_mean.param.interm <- seq(max.value_traits - param_DIST_between_MEAN[n],
                               max.value_traits, length.out =
                               length(unique(param_DIST_between_MEAN)))

 mean_mean.param <- rep(round(sort(mean_mean.param.interm), 2), N_repet_Param)

 mean.sp <- rnorm(length(unique(sp)), mean = mean_mean.param, sd = mean_sd.param[n])
 sd.sp <- rnorm(length(unique(sp)), mean = sd_mean.param[n], sd = sd_sd.param[n])

 ex.traits2 <- array(NA, dim = c(Nind, 2))
 colnames(ex.traits2) <- paste("trait", c("a", "b"), sep = " ")

 # Draw the individual traits depending on species' attributes
  for(s in unique(ex.sp2)){
   #trait a : normal distribution
   ex.traits2[ex.sp2 == s, 1] <-
   rnorm(5*Nind/Ncom, rep(mean.sp[unique(ex.sp2) == s], 5*Nind/Ncom),
         rep(sd.sp[unique(ex.sp2) == s], 5*Nind/Ncom))[1:sum(ex.sp2 == s)]

   #trait b : uniform distribution
   ex.traits2[ex.sp2 == s, 2] <-
   runif(5*Nind/Ncom, min = rep(mean.sp[unique(ex.sp2) == s], 5*Nind/Ncom) -
         rep(sd.sp[unique(ex.sp2) == s], 5*Nind/Ncom),
         max = rep(mean.sp[unique(ex.sp2) == s], 5*Nind/Ncom) +
         rep(sd.sp[unique(ex.sp2) == s], 5*Nind/Ncom))[1:sum(ex.sp2 == s)]
  }

 #stock results
 mean.sp_stock2[[n]] <- mean.sp
```

```
 sd.sp_stock2[[n]] <- sd.sp

 mean_dist_sp[[n]] <- c(mean(tapply(ex.traits2[, 1], ex.indplot2, function(x)
                                    mean(dist(x), na.rm=T)), na.rm=T),
                        mean(tapply(ex.traits2[, 2], ex.indplot2, function(x)
                                    mean(dist(x), na.rm=T)), na.rm=T))

 res.simu.traits2[[n]] <- ex.traits2
 res.simu2[[n]] <- Tstats(ex.traits2, ex.indplot2, ex.sp2, printprogress = FALSE)
 res.simu2.pval[[n]] <- sum_Tstats(res.simu2[[n]], type = "p.value")
 print(paste(round(n/nperm, 2) * Nind/Ncom, "%", sep = " "))
}#End of simulations
```

3.2 Results of the simulations of internal filtering

Let's see the result for one randomization. We can plot the distribution of trait values within species and/or communities thanks to the function `plotDistri` (Fig. 6).

```
par(mfrow=c(2, 2))
plotDistri(ex.traits2, rep("all_sp", times = dim(ex.traits2)[1]), ex.indplot2,
           plot.ask = F, multipanel = F, cex.leg = 0.6, main= c("a", "b"))
plotDistri(ex.traits2, rep("region", times = dim(ex.traits2)[1]), ex.sp2, plot.ask = F,
           multipanel = F, ylim = c(0,0.04), cex.leg = 0.6, main= c("c", "d"))
par(mfrow=c(1, 1))
```

We can also plot the result of the T-statistics for two contrasting cases (Fig. 7).

```
par(mfrow=c(2, 1))
plot(res.simu2[[1]], main="a")
plot(res.simu2[[n]], main="b")
par(mfrow=c(1, 1))
```

For the power analysis of internal filtering we present the results in two steps: (i) the test of robustness of each T-statistics values using local p-values and (ii) the test of community-aggregated T-statistics using global p-values.

3.2.1 Results with local p-values

Here we call local p-values the p-values corresponding to one index for one trait in one community. We can plot the local p-values using different ways. First, we can represent the ordered p.value in decimal logarithm and calculate the power of the $T_{IP/IC}$ index using the beta-error[6]. The less the beta-error, the more the index is powerful. Contrary to the test without filter, we choose a unilateral test because we only want to test if $T_{IP/IC}$ is smaller than it should be under the associated null hypothesis.

3.2.1.1 Calculation of beta-error mixing all strengths of filtering

[6]The proportion of cases for which we do not reject the null hypothesis while this hypothesis is wrong.

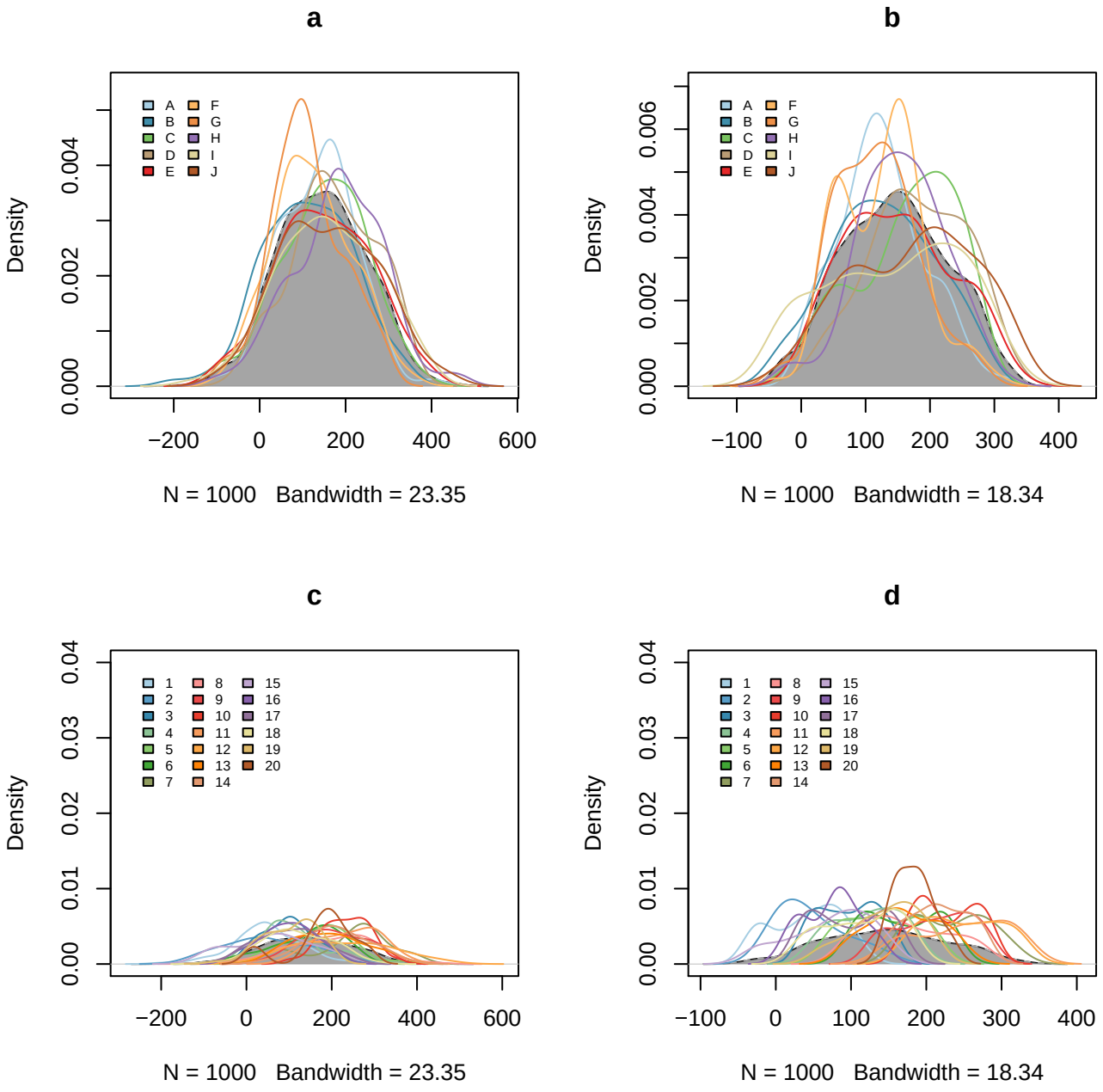


Figure 6: Distribution of trait values for one randomization with internal filtering: (a) Community-level trait distributions for the trait a (normal distribution); (b) Community-level trait distributions for the trait b (uniform distribution). In panels a and b, each color represents one community (site). (c) Species' trait distributions for the trait a; (d) Species' trait distributions for the trait b. In panels c and d, each color represents one species.

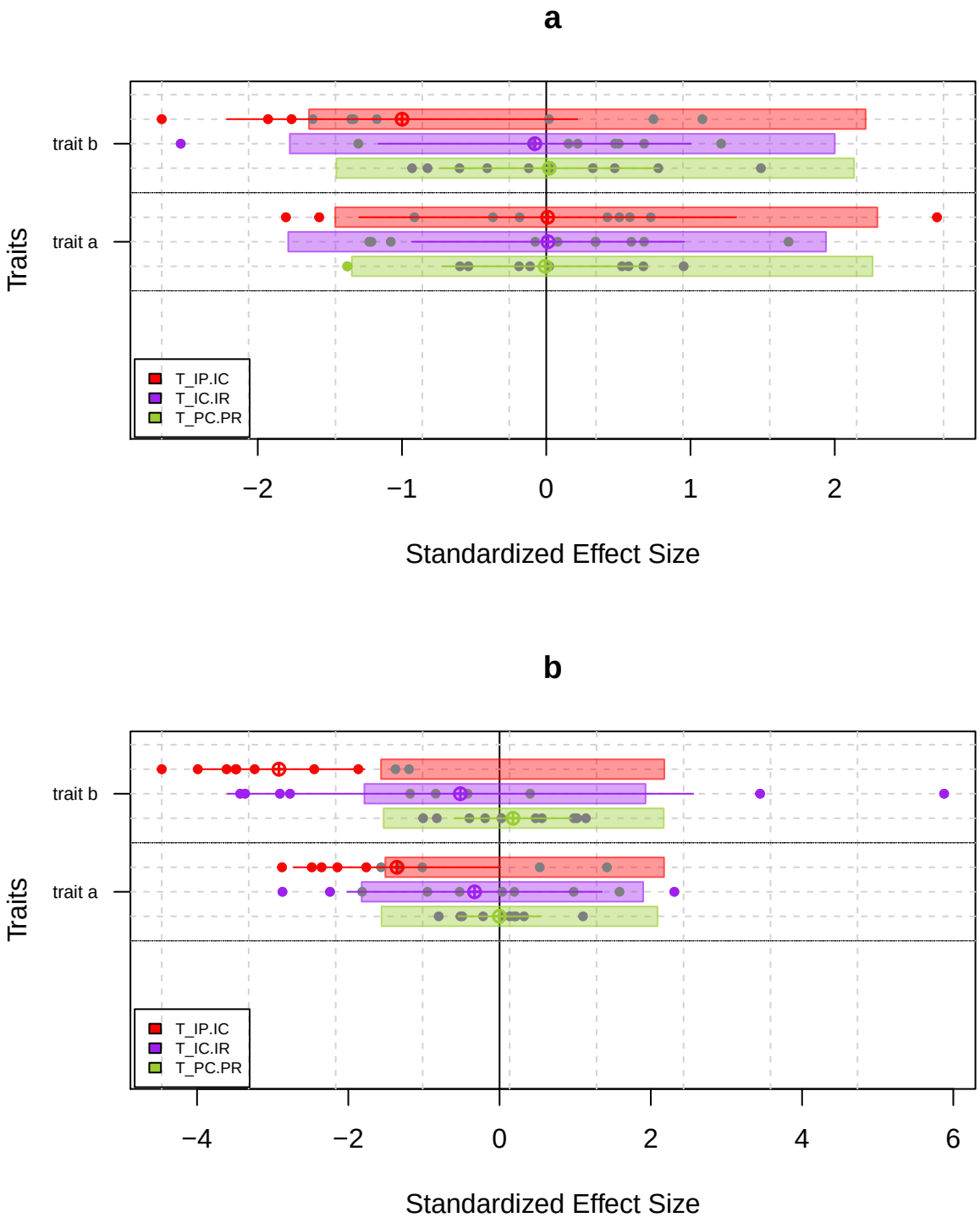


Figure 7: T-statistics for two randomizations with contrasted strengths of internal filtering: $T_{IP/IC}$ in red, $T_{IC/IR}$ in purple and $T_{PC/PR}$ in green. (a) Lower strength of internal filtering and (b) higher strength of internal filtering.

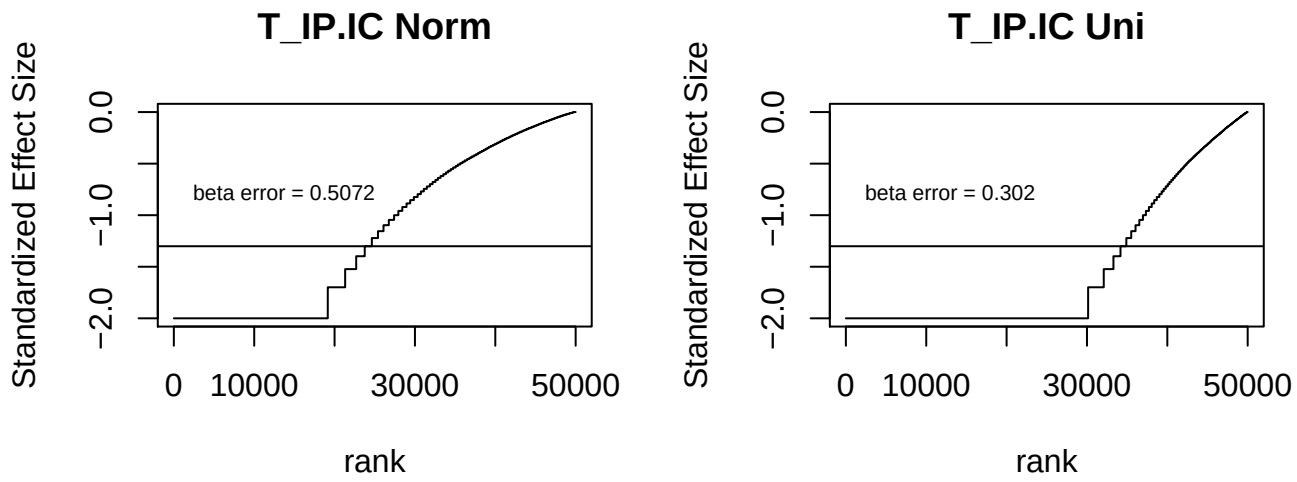


Figure 8: Beta-error mixing all strengths of internal filtering: Standardized effect sizes of $T_{IP/IC}$ are ordered and plotted. The black line represents the value of 0.05. Consequently, p-values above this line are false-negative cases. Norm: normal distribution, Uni: uniform distribution.

```
par(mfrow = c(1, 2))
xx <- log10(sort(unlist(lapply(res.simu2.pval, function(x) x[1:10, 1]))))
plot(xx, type = "l", main = "T_IP.IC Norm",
     xlab = "rank", ylab = "Standardized Effect Size")
abline(h = log10(0.05))
nbre_beta_error <- round((sum(xx>log10(0.05))+1)/(length(xx)+1), 4)
text(0, -0.8, labels = paste("beta error", nbre_beta_error, sep = " = "),
  cex = 0.7, pos = 4)

xx <- log10(sort(unlist(lapply(res.simu2.pval, function(x) x[1:10, 2]))))
plot(xx, type = "l", main = "T_IP.IC Uni",
     xlab = "rank", ylab = "Standardized Effect Size")
abline(h = log10(0.05))
nbre_beta_error <- round((sum(xx>log10(0.05))+1)/(length(xx)+1), 4)
text(0, -0.8, labels = paste("beta error", nbre_beta_error, sep = " = "),
  cex = 0.7, pos = 4)
par(oldpar)
```

The beta-error is very high and thus the power of $T_{IP/IC}$ can be *prima facie* seen as problematic (Fig. 8). But here we mingle the p-value for several parameter values. Thus it is far more informative to see the power of the test in relation to the initial parameter values. We represent standardized effect sizes values (hereafter called SES) in relation to the strength of internal filter to determine the parameter values which allow a satisfactory power of $T_{IP/IC}$.

As we add stochasticity around the initial parameter values, we can measure the strength of internal filtering by (i) the initial parameter values or (ii) by calculating the modeled parameter values[7].

3.2.1.2 Local $T_{IP/IC}$ SES values against initial parameter values

```
meanSES.2loc.norm_Tipic <- unlist(lapply(res.simu2, function(x)
            ses.listofindex(as.listofindex(x))
```

[7] *i.e.* the parameter values after adding stochasticity

```
            $index_1_1$ses[, 1]))

meanSES.2loc.uni_Tipic <- unlist(lapply(res.simu2, function(x)
            ses.listofindex(as.listofindex(x))
            $index_1_1$ses[, 2]))

SES.inf.MEAN.norm_Tipic <- unlist(lapply(res.simu2, function(x)
            ses.listofindex(as.listofindex(x))
            $index_1_1$ses.inf[, 1]))

SES.inf.MEAN.uni_Tipic <- unlist(lapply(res.simu2, function(x)
            ses.listofindex(as.listofindex(x))
            $index_1_1$ses.inf[, 2]))
```

Now, we can plot SES of $T_{IP/IC}$ (Fig. 9). The trait "a" normally distributed is in black and the uniform traits "b" is in purple. The colored rectangles represent the null model area with alpha = 5%. Thus, when a dot is outside these area, the modeled parameters are strong enough to detect the internal filter with a high power (beta-error < 0.05).

```
init_param <- param_DIST_between_MEAN / sd_mean.param
init_param.loc <- rep(init_param, each = Ncom)

plot(meanSES.2loc.norm_Tipic, init_param.loc, pch = 16, col = rgb(0, 0, 0, 0.6),
     xlim = c(min(c(unlist(meanSES.2loc.norm_Tipic), unlist(meanSES.2loc.uni_Tipic)),
     na.rm = T), 0),
     main = "Local T_IP.IC",
     xlab = "standardized effect size of T_IP.IC",
     ylab = "initial parameters: range/sd")
abline(v = mean(SES.inf.MEAN.norm_Tipic, na.rm = T))

points(meanSES.2loc.uni_Tipic, init_param.loc, pch = 16, col = rgb(0.5, 0, 1, 0.6))
abline(v = mean(SES.inf.MEAN.uni_Tipic, na.rm = T), col = "purple")
rect(mean(SES.inf.MEAN.norm_Tipic, na.rm = T), -1, 0, max(init_param) + 1,
     col = rgb(0, 0, 0, 0.3), border = NA)
rect(mean(SES.inf.MEAN.uni_Tipic, na.rm = T), 0, 0, max(init_param),
     col = rgb(0.5, 0, 1, 0.3), border = NA)
```

For each initial parameter values we can calculate the proportion of cases for which we do not reject the null hypothesis while this hypothesis is wrong (*i.e.* the Beta-error). The next chunk only shows the calculation for the $T_{IC/IR}$ metrics for the trait "a".

```
beta_sd_norm_Tipic <-
 (unlist(lapply(by(cbind(meanSES.2loc.norm_Tipic, SES.inf.MEAN.norm_Tipic),
     rep(sd_mean.param, each = Ncom), function(x) x[,1] > x[,2]),
     function(x) sum(x, na.rm = T)))+1)/(10 * N_repet_Param + 1)
beta_range_norm_Tipic <-
 (unlist(lapply(by(cbind(meanSES.2loc.norm_Tipic, SES.inf.MEAN.norm_Tipic),
     rep(param_DIST_between_MEAN, each = Ncom), function(x) x[,1] > x[,2]),
     function(x) sum(x, na.rm = T)))+1)/(10 * N_repet_Param + 1)

beta_sd_uni_Tipic <-
```

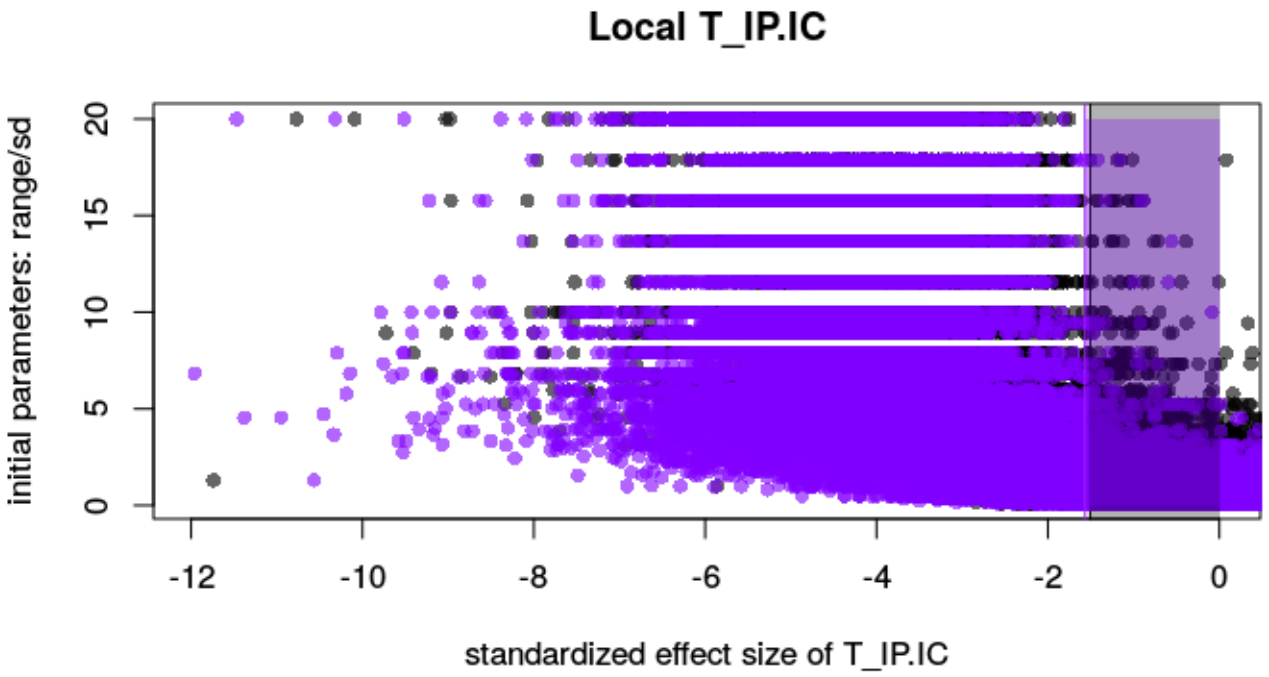


Figure 9: Local $T_{IP/IC}$ SES and initial parameter values: Standardized effect size of $T_{IP/IC}$ as a function of the strength of internal filtering defined as the ratio of the initial range parameter by the initial standard error parameter. Colored area represents the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and trait b (uniform) in purple.

```
 (unlist(lapply(by(cbind(meanSES.2loc.uni_Tipic, SES.inf.MEAN.uni_Tipic),
     rep(sd_mean.param, each = Ncom), function(x) x[,1] > x[,2]),
     function(x) sum(x, na.rm = T)))+1)/(10 * N_repet_Param + 1)
beta_range_uni_Tipic <-
 (unlist(lapply(by(cbind(meanSES.2loc.uni_Tipic, SES.inf.MEAN.uni_Tipic),
     rep(param_DIST_between_MEAN, each = Ncom), function(x) x[,1] > x[,2]),
     function(x) sum(x, na.rm = T)))+1)/(10 * N_repet_Param + 1)

res_beta_Tipic <- rbind(rev(beta_sd_norm_Tipic), rev(beta_sd_uni_Tipic),
      beta_range_norm_Tipic, beta_range_uni_Tipic)
colnames(res_beta_Tipic) <- paste("str", 1:ncol(res_beta_Tipic))
rownames(res_beta_Tipic) <- c("effect of sd (a)", "effect of sd (b)",
       "effect of range (a)", "effect of range (b)")
```

```
print(xtable(res_beta_Tipic, caption = 'Local beta-error of $T_{IP/IC}$ as a function
             of the strength of internal filtering. str: strength. Trait a is
             normally distributed and trait b is uniformally distributed.',
             label = "tab:local_beta_error_xtable"),
      caption.placement = "top", size = "small")
```

Table 3: Local beta-error of $T_{IP/IC}$ as a function of the strength of internal filtering. str: strength. Trait a is normally distributed and trait b is uniformally distributed.

	str 1	str 2	str 3	str 4	str 5	str 6	str 7	str 8	str 9	str 10
effect of sd (a)	0.82	0.79	0.74	0.69	0.65	0.55	0.44	0.32	0.22	0.15
effect of sd (b)	0.59	0.53	0.47	0.43	0.39	0.32	0.24	0.15	0.09	0.05
effect of range (a)	0.87	0.82	0.75	0.66	0.59	0.47	0.39	0.31	0.28	0.23
effect of range (b)	0.75	0.67	0.57	0.41	0.32	0.22	0.13	0.09	0.06	0.04

Now we can see the beta-error for each strength in table 3. A high value of range and a low value of standard error defined a high strength of filtering. For example, in the case of a trait normally distributed, the proportion of false negative is 82.26% for the higher value of standard errors and thus for the lower strength of internal filtering. The good behavior of the function `Tstats` is ascertained by the decrease in the beta-error when increasing the strength of the filter.

3.2.1.3 Local $T_{IP/IC}$ SES values against modeled parameter values

First we need to calculate the modeled parameter values. Here again we calculate the strength of internal filtering by dividing the range of trait values in a community by its standard error.

```
mean_dist_sp.interm <- t(matrix(unlist(lapply(mean_dist_sp, function(x) x)), nrow = 2))
mean_sd_of_com <- unlist(lapply(sd.sp_stock2, function(x) mean(x)))

modeled_param_norm <- mean_dist_sp.interm[, 1] / mean_sd_of_com
modeled_param_uni <- mean_dist_sp.interm[, 2] / mean_sd_of_com
modeled_param_norm.loc <- rep(modeled_param_norm, each = Ncom)
modeled_param_uni.loc <- rep(modeled_param_uni, each = Ncom)
```

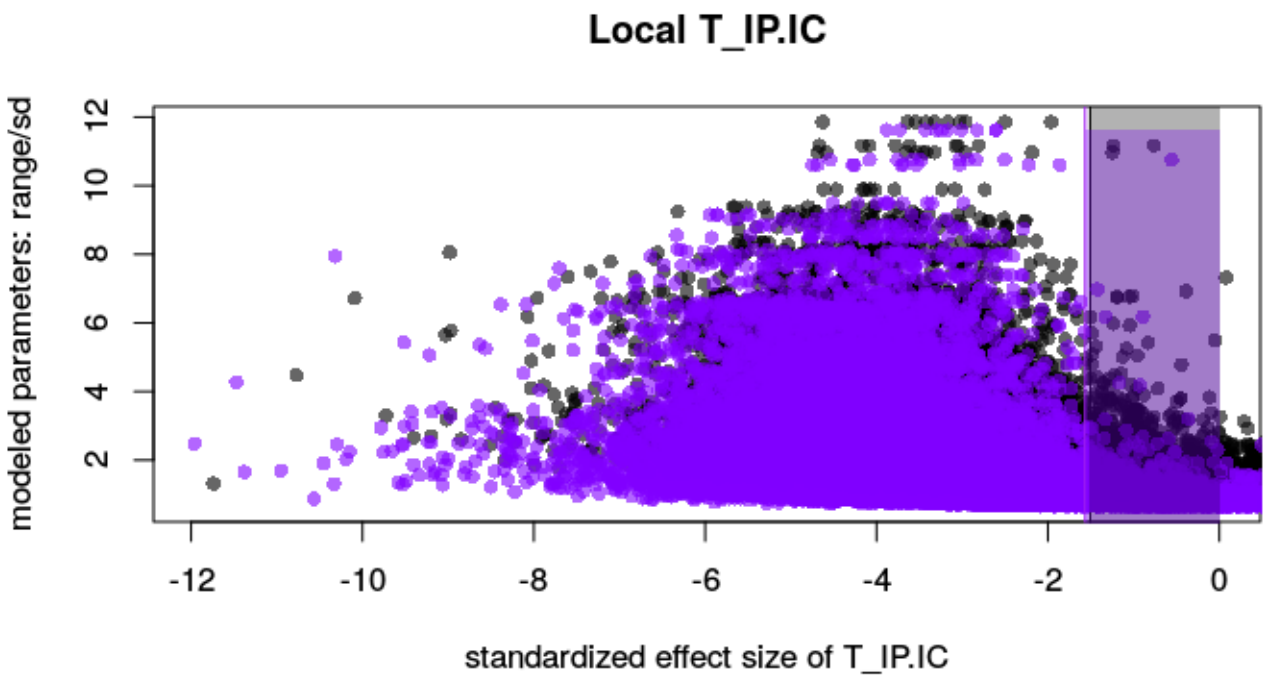


Figure 10: Local $T_{IP/IC}$ SES and modeled parameter values: Standardized effect size of $T_{IP/IC}$ as a function of the strength of internal filtering defined as the ratio of the modeled range parameter by the modeled standard error parameter. The colored rectangles represent the mean confidence interval for SES values ($alpha = 0.95$). Trait a (normal) in black and trait b (uniform) in purple.

```
plot(meanSES.2loc.norm_Tipic, modeled_param_norm.loc, pch = 16, col = rgb(0, 0, 0, 0.6),
     xlim = c(min(c(unlist(meanSES.2loc.norm_Tipic), unlist(meanSES.2loc.uni_Tipic)),
              na.rm = T), 0),
     ylim = c(min(c(modeled_param_norm.loc, modeled_param_uni.loc), na.rm = T),
              max(c(modeled_param_norm.loc, modeled_param_uni.loc), na.rm = T)),
     main = "Local T_IP.IC",
     xlab = "standardized effect size of T_IP.IC",
     ylab = "modeled parameters: range/sd")
abline(v = mean(SES.inf.MEAN.norm_Tipic, na.rm = T))

points(meanSES.2loc.uni_Tipic, modeled_param_uni.loc, pch = 16, col = rgb(0.5, 0, 1, 0.6))
abline(v = mean(SES.inf.MEAN.uni_Tipic, na.rm = T), col = "purple")
rect(mean(SES.inf.MEAN.norm_Tipic, na.rm = T), -1, 0, max(modeled_param_norm.loc) + 1,
     col = rgb(0, 0, 0, 0.3), border = NA)
rect(mean(SES.inf.MEAN.uni_Tipic, na.rm = T), 0, 0, max(modeled_param_uni.loc),
     col = rgb(0.5, 0, 1, 0.3), border = NA)
```

There is a very hich correlation coefficient between initial and modeled parameter either for the range (trait a: Pearson correlation = 0.337; trait b: Pearson correlation = 0.374) and for the standard error(trait a and b: Pearson correlation = 0.997). Consequently the consistency between the figures 9 and 10 is not suprising.

3.2.2 Global p-values results for $T_{IP/IC}$

In contrast with local p-values, we call global p-values the p-values corresponding to one index for one trait across all the communities.

3.2.2.1 Global $T_{IP/IC}$ SES values against initial parameter values

Again, these first results mix different initial parameter values. Now, we can plot the SES values as a function of the strength of the internal filtering assessed by the two parameters (either the initial values: `mean_range_between_com` and `mean_sd_of_com` or the modeled values `mean_range_com` and `sd.com_stock2` [8]).

First, we need to compute the SES values from simulations.

```
meanSES.2glob.norm_Tipic <- unlist(lapply(res.simu2, function(x)
           mean(ses.listofindex(as.listofindex(x))
             $index_1_1$ses[,1], na.rm = T)))
meanSES.2glob.uni_Tipic <- unlist(lapply(res.simu2, function(x)
           mean(ses.listofindex(as.listofindex(x))
             $index_1_1$ses[,2], na.rm = T)))

meanSES.INF_glob.norm_Tipic <- unlist(lapply(res.simu2, function(x)
           mean(ses.listofindex(as.listofindex(x))
             $index_1_1$ses.inf[, 1], na.rm = T)))
meanSES.INF_glob.uni_Tipic <- unlist(lapply(res.simu2, function(x)
           mean(ses.listofindex(as.listofindex(x))
             $index_1_1$ses.inf[, 2], na.rm = T)))
```

Plot the result against initial parameters (Fig. 11).

```
plot(meanSES.2glob.norm_Tipic, init_param, pch = 16, col = rgb(0, 0, 0, 0.6),
     main = "Global T_IP.IC",
     xlim = c(min(c(meanSES.2glob.norm_Tipic, meanSES.2glob.uni_Tipic),
            na.rm = T), 0),
     xlab = "standardized effect size of T_IP.IC",
     ylab = "initial parameters: range/sd")
points(meanSES.2glob.uni_Tipic, init_param, pch = 16, col = rgb(0.5, 0, 1, 0.6),)
abline(v = mean(meanSES.INF_glob.norm_Tipic, na.rm = T))
abline(v = mean(meanSES.INF_glob.uni_Tipic, na.rm = T), col = "purple")
rect(mean(meanSES.INF_glob.norm_Tipic, na.rm = T), -1, 0, max(modeled_param_norm.loc) + 1,
     col = rgb(0, 0, 0, 0.3), border = NA)
rect(mean(meanSES.INF_glob.uni_Tipic, na.rm = T), 0, 0, max(modeled_param_uni.loc),
     col = rgb(0.5, 0, 1, 0.3), border = NA)
```

3.2.2.2 Global $T_{IP/IC}$ SES values against modeled parameter values

We can verify the consistency of our results by plotting SES values against the modeled parameter values (Fig. 12).

[8] These modeled values are stochastic versions of the initial values.

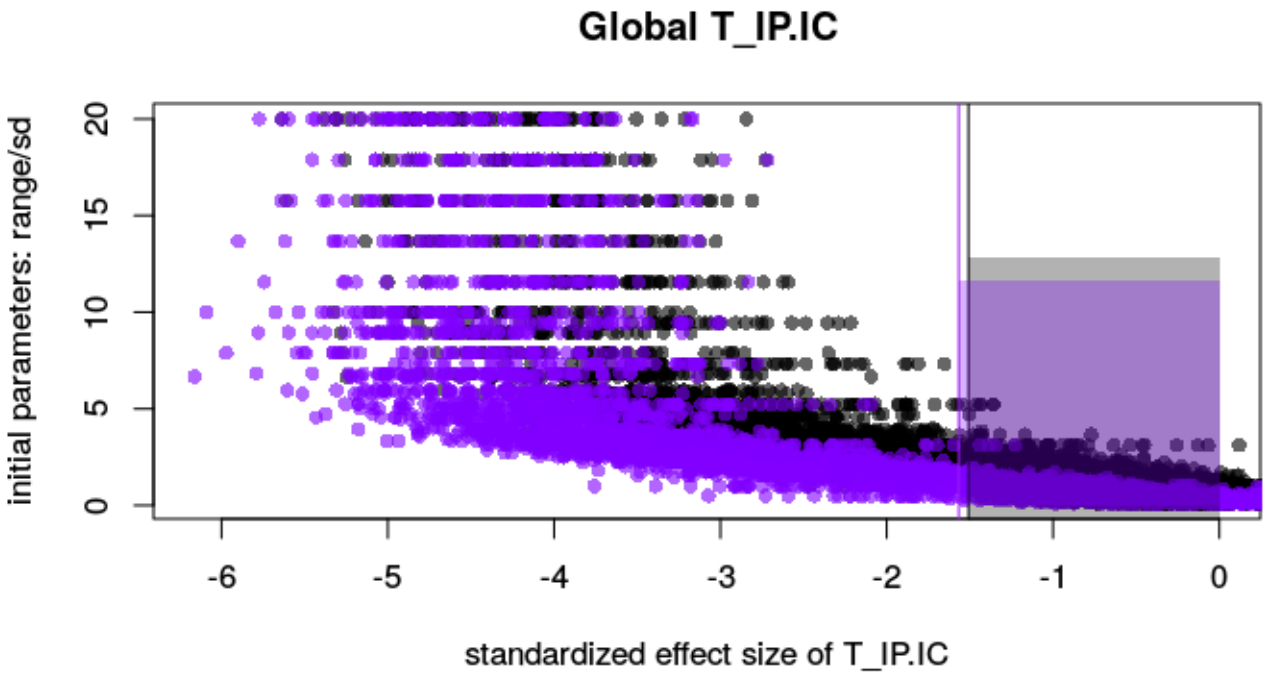


Figure 11: Global $T_{IP/IC}$ SES and initial parameter values: Standardized effect size of $T_{IP/IC}$ as a function of the strength of internal filtering defined as the ratio of the inital range parameter by the initial standard error parameter. Colored area represents the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and trait b (uniform) in purple.

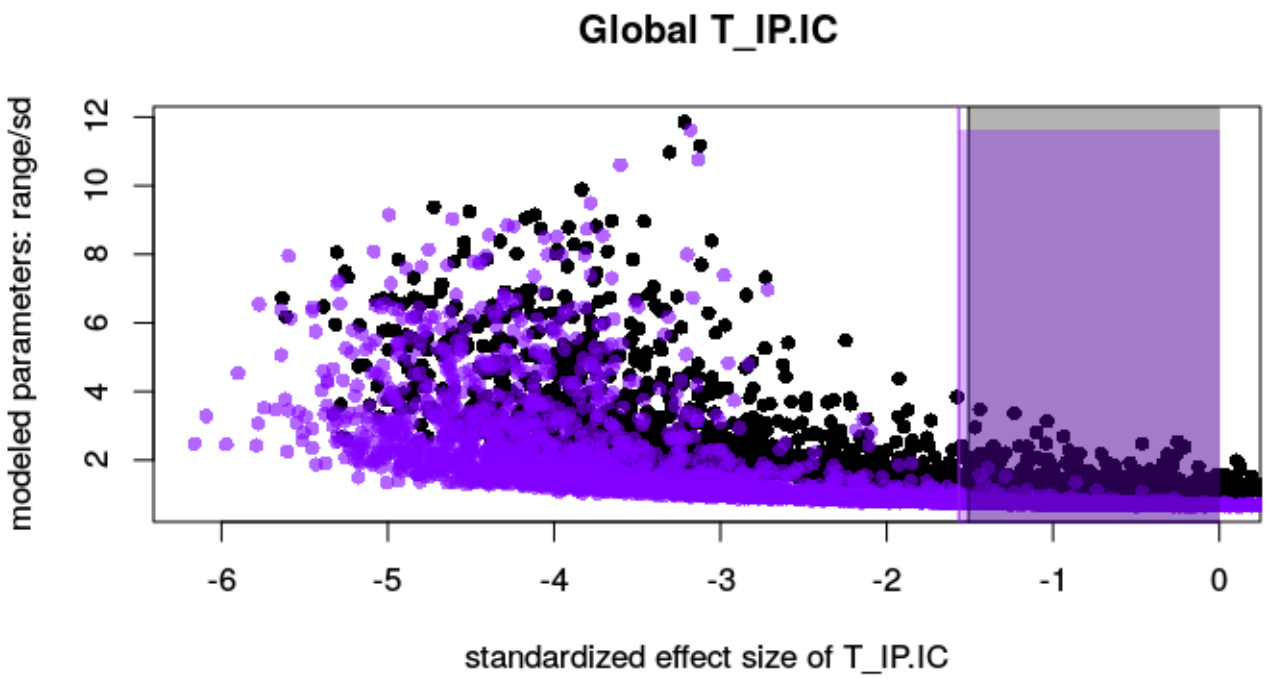


Figure 12: Global $T_{IP/IC}$ SES and modeled parameter values: Standardized effect size of $T_{IP/IC}$ as a function of the strength of internal filtering defined as the ratio of the modeled range parameter by the modeled standard error parameter. Colored area represents the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and trait b (uniform) in purple.

```
mean_dist_sp.interm <- t(matrix(unlist(lapply(mean_dist_sp, function(x) x)), nrow = 2))
mean_sd_of_com <- unlist(lapply(sd.sp_stock2, function(x) mean(x)))

modeled_param_norm <- mean_dist_sp.interm[, 1] / mean_sd_of_com
modeled_param_uni <- mean_dist_sp.interm[, 2] / mean_sd_of_com

plot(meanSES.2glob.norm_Tipic, modeled_param_norm, pch = 16,
     main = "Global T_IP.IC",
     xlim = c(min(c(meanSES.2glob.norm_Tipic, meanSES.2glob.uni_Tipic),
              na.rm = T), 0),
     ylim = c(min(c(modeled_param_norm, modeled_param_uni), na.rm = T),
     max(c(modeled_param_norm, modeled_param_uni), na.rm = T)),
     xlab = "standardized effect size of T_IP.IC",
     ylab = "modeled parameters: range/sd")
points(meanSES.2glob.uni_Tipic, modeled_param_uni, pch = 16, col = rgb(0.5, 0, 1, 0.6))
abline(v = mean(meanSES.INF_glob.norm_Tipic, na.rm = T))
abline(v = mean(meanSES.INF_glob.uni_Tipic, na.rm = T), col = "purple")
rect(mean(meanSES.INF_glob.norm_Tipic, na.rm = T), -1, 0, max(modeled_param_norm) + 1,
     col = rgb(0, 0, 0, 0.3), border = NA)
rect(mean(meanSES.INF_glob.uni_Tipic, na.rm = T), 0, 0, max(modeled_param_uni),
     col = rgb(0.5, 0, 1, 0.3), border = NA)
```

3.3 Conclusion on the power of $T_{IP/IC}$ to detect internal filtering

To conclude on the power of $T_{IP/IC}$ to detect internal filtering, we apply an exponential linear model and identify the strength from which the beta-error is inferior to 0.05. This strength is defined as the ratio of the initial parameters `param_DIST_between_MEAN` and `sd_mean.param`.

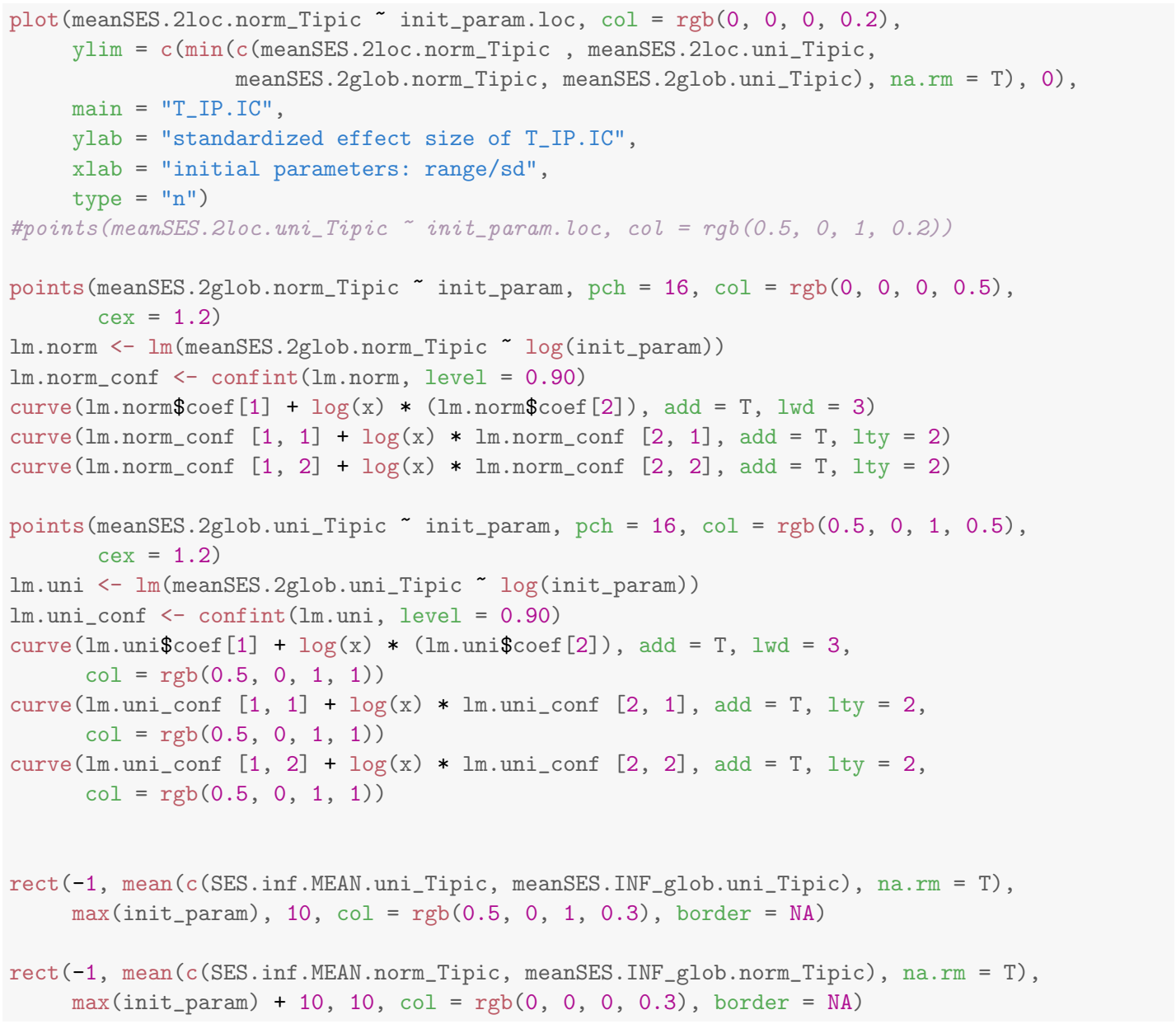
```
plot(meanSES.2loc.norm_Tipic ~ init_param.loc, col = rgb(0, 0, 0, 0.2),
     ylim = c(min(c(meanSES.2loc.norm_Tipic , meanSES.2loc.uni_Tipic,
                    meanSES.2glob.norm_Tipic, meanSES.2glob.uni_Tipic), na.rm = T), 0),
     main = "T_IP.IC",
     ylab = "standardized effect size of T_IP.IC",
     xlab = "initial parameters: range/sd",
     type = "n")
#points(meanSES.2loc.uni_Tipic ~ init_param.loc, col = rgb(0.5, 0, 1, 0.2))

points(meanSES.2glob.norm_Tipic ~ init_param, pch = 16, col = rgb(0, 0, 0, 0.5),
       cex = 1.2)
lm.norm <- lm(meanSES.2glob.norm_Tipic ~ log(init_param))
lm.norm_conf <- confint(lm.norm, level = 0.90)
curve(lm.norm$coef[1] + log(x) * (lm.norm$coef[2]), add = T, lwd = 3)
curve(lm.norm_conf [1, 1] + log(x) * lm.norm_conf [2, 1], add = T, lty = 2)
curve(lm.norm_conf [1, 2] + log(x) * lm.norm_conf [2, 2], add = T, lty = 2)

points(meanSES.2glob.uni_Tipic ~ init_param, pch = 16, col = rgb(0.5, 0, 1, 0.5),
       cex = 1.2)
lm.uni <- lm(meanSES.2glob.uni_Tipic ~ log(init_param))
lm.uni_conf <- confint(lm.uni, level = 0.90)
curve(lm.uni$coef[1] + log(x) * (lm.uni$coef[2]), add = T, lwd = 3,
      col = rgb(0.5, 0, 1, 1))
curve(lm.uni_conf [1, 1] + log(x) * lm.uni_conf [2, 1], add = T, lty = 2,
      col = rgb(0.5, 0, 1, 1))
curve(lm.uni_conf [1, 2] + log(x) * lm.uni_conf [2, 2], add = T, lty = 2,
      col = rgb(0.5, 0, 1, 1))


rect(-1, mean(c(SES.inf.MEAN.uni_Tipic, meanSES.INF_glob.uni_Tipic), na.rm = T),
     max(init_param), 10, col = rgb(0.5, 0, 1, 0.3), border = NA)

rect(-1, mean(c(SES.inf.MEAN.norm_Tipic, meanSES.INF_glob.norm_Tipic), na.rm = T),
     max(init_param) + 10, 10, col = rgb(0, 0, 0, 0.3), border = NA)
```



```
yy.norm <- mean(c(SES.inf.MEAN.norm_Tipic, meanSES.INF_glob.norm_Tipic), na.rm = T)
yy.uni <- mean(c(SES.inf.MEAN.uni_Tipic, meanSES.INF_glob.uni_Tipic), na.rm = T)

param_beta0.05_Tipic.norm <- exp((yy.norm - lm.norm_conf [1, 2]) / lm.norm_conf [2, 2])
param_beta0.05_Tipic.uni <- exp((yy.uni - lm.uni_conf [1, 2]) / lm.uni_conf [2, 2])
```

For the trait "a" normally distributed, the power of $T_{IP/IC}$ is satisfactory if the ratio of the initial parameter is superior to 1.86. For the trait "b" uniformally distributed, this value is 0.88. Further investigation on real data with different strengths of internal filtering are needed to complete these simulated values.

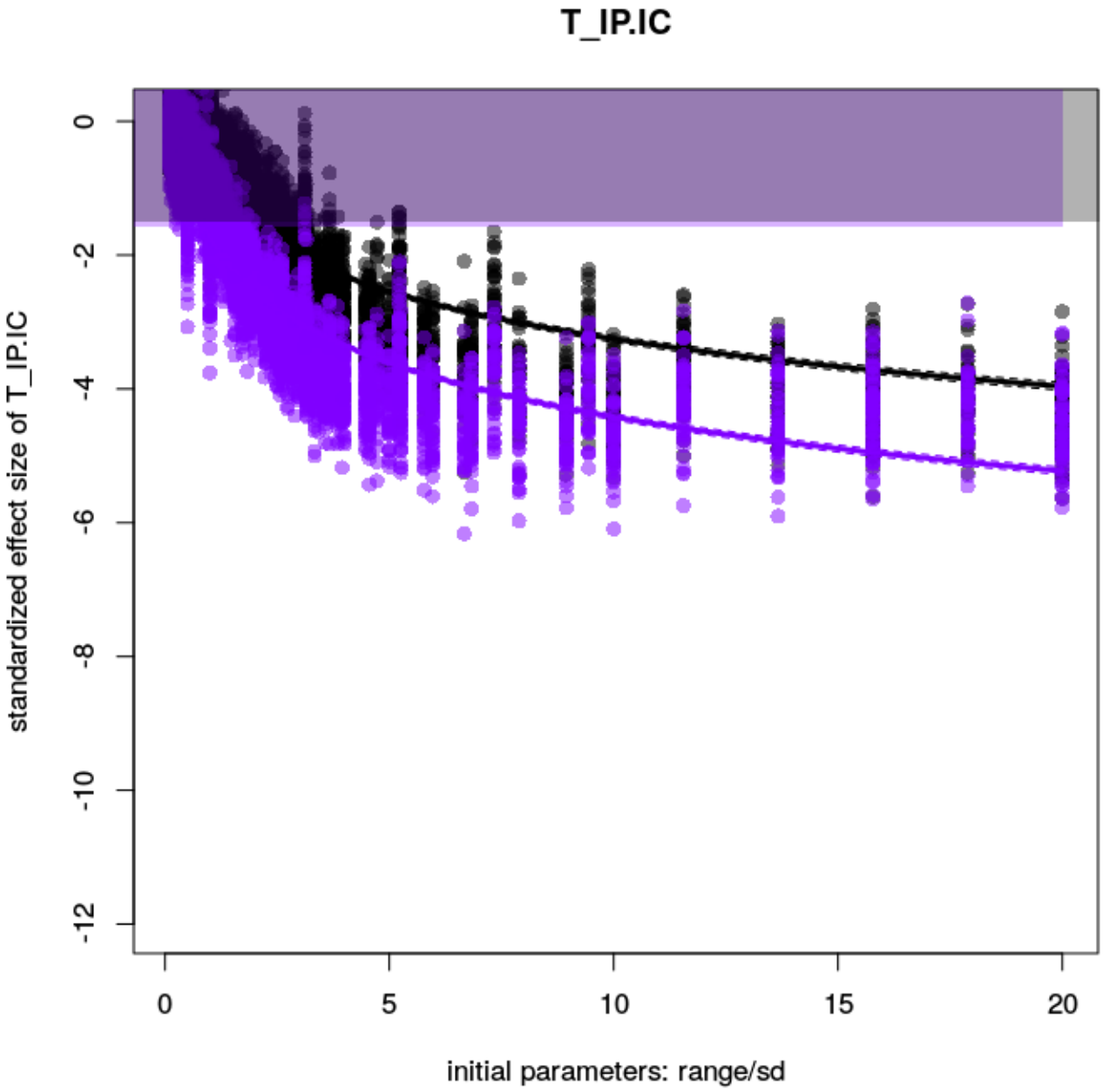


Figure 13: Power of $T_{IP/IC}$ to detect internal filtering: Standardised Effect Size (SES) of $T_{IP/IC}$ as a function of the strength of internal filtering. Trait 'a' normally distributed is shown in black, trait 'b' uniformally distributed in purple. Dots represent global SES values. The exponential linear model is presented with a 0.05 confidence interval on both sides. Colored area represents the mean confidence interval of SES values ($alpha = 0.95$)

4 External Filter

4.1 Randomization outline

We modulate the strength of the external filter (here considered as a functional turnover between communities) thanks to two parameters: we define (i) a gradient of distances between communities' mean trait distributions and (ii) a gradient of variance in communities' trait distributions. We expect the external filtering to increase (i) when the distance between communities' means increase (*i.e.* the environmental gradient is larger) and (ii) when the variance in communities' trait distributions decreases (*i.e.* the environmental gradient is stronger).

In analogy with the internal filtering modelling framework, we defined 10 values for two parameters:

(i) `param_RANGE.init` is a vector of length 10 which defined the distance between communities' mean trait values (the maximum value for the mean is set to 250). In decreasing the range of traits values [9] while keeping the same number of communities, there is a increase in the overlap of the communities' trait distributions and consequently an increase in the impact of external filtering. The mean for each communitiy is drawn from a normal distribution with standard deviation of 10 (parameter `mean_sd.param`) and means evenly [10] distributed between `max.value_traits - param_RANGE` and `max.value_traits`.

(ii) `param_SD` is a vector of length 10 which defined the standard deviation of trait distributions for each community. In order to decorrelate the mean and standard deviation of community's trait distributions, `param_SD` is permuted before the analysis.

Thus for the trait "a" the trait value for each individual is drawn from a normal distribution with the mean depending on its community attribute. For the trait "b" the trait value for each individual is drawn from a uniform distribution with the range depending on its community attribute.

```
# Parameter for the range of communities trait distributions
param_range.init <- round(sort(seq(10, 200, length.out = nb_param_val)), 2)
param_range <- rep(param_range.init, N_repet_Param)
mean_sd.param <- rep(10, nperm)

# Parameter for the variance in species' mean trait values
param_SD <- seq(10, 100,length.out = nb_param_val)
sd_mean.param <- sample (rep(param_SD, N_repet_Param),
       size = length(rep(param_SD, N_repet_Param)), replace = F)
sd_sd.param <- rep(10, nperm)
```

```
nperm <- N_repet_Param*length(param_SD)

mean_range_com <- list() ; sd.com_stock3 <- list() ; res.simu3 <- list()
res.simu3.pval <- list() ; res.simu.traits3 <- list()

for(n in 1:nperm){#for each permutation
```

[9] thus decreasing the distance between communities' mean trait values

[10] In fact, we draw these values in a normal distribution with means evenly distributed and with a standard error equal to `mean_sd.param`

```
ex.sp3 <- c()
ex.com3 <- matrix(0, nrow = Ncom, ncol = Nsp)
for(i in 1: 10){
 ex.com.interm <- table(sample(sp, size = Nind/Ncom, prob = rlnorm(Nsp, 0, sdlog), replace = T))
 ex.com3[i, sp%in% names(ex.com.interm)] <- ex.com.interm
 ex.sp3 <- c(ex.sp3, rep(sp, times = ex.com3[i,]))
}

ex.indplot3 <- sort(as.factor(rep(com, Nind/Ncom)))

#Defining mean and sd for each community
mean_mean.param.interm <- seq(max.value_traits - param_range[n], max.value_traits,
                              length.out = length(unique(param_range)))

mean_mean.param <- rep(round(sort(mean_mean.param.interm), 2), N_repet_Param)

mean.com <- rnorm(length(unique(com)), mean = mean_mean.param, sd = mean_sd.param[n])
sd.com <- rnorm(length(unique(com)), mean = sd_mean.param[n], sd = sd_sd.param[n])

ex.traits3 <- array(NA, dim = c(Nind, 2))
colnames(ex.traits3) <- paste("trait", c("a", "b"), sep = " ")


for(c in unique(ex.indplot3)){
 #trait a : normal distribution
 ex.traits3[ex.indplot3 == c, 1] <-
 rnorm(500, rep(mean.com[unique(ex.indplot3) == c], 500),
   rep(sd.com[unique(ex.indplot3) == c], 500))[1:sum(ex.indplot3 == c)]

 #trait b : uniform distribution
 ex.traits3[ex.indplot3 == c, 2] <-
 runif(500, min = rep(mean.com[unique(ex.indplot3) == c], 500) -
       rep(sd.com[unique(ex.indplot3) == c], 500),
       max = rep(mean.com[unique(ex.indplot3) == c], 500) +
       rep(sd.com[unique(ex.indplot3) == c], 500))[1:sum(ex.indplot3 == c)]
}

#stock results
mean_range_com[[n]] <- c(max(ex.traits3[, 1], na.rm = T) - min(ex.traits3[, 1], na.rm = T),
                         max(ex.traits3[, 2], na.rm = T) - min(ex.traits3[, 2], na.rm = T))
sd.com_stock3[[n]] <- sd.com

res.simu.traits3[[n]] <- ex.traits3
res.simu3[[n]] <- Tstats(ex.traits3, ex.indplot3, ex.sp3)
res.simu3.pval[[n]] <- sum_Tstats(res.simu3[[n]], type = "p.value")
print(paste("---", round(n/nperm, 2) * 100, "%", sep = " "))
}#End of simulations
```

4.2 Results of the simulations of external filtering

Let's see the results for one randomization. We can plot the distribution of trait values within species and/or communities thanks to the function `plotDistri` (Fig. 14).

```
par(mfrow=c(2, 2))
plotDistri(ex.traits3, rep("all_sp", times = dim(ex.traits3)[1]), ex.indplot3,
           plot.ask = F, multipanel = F, main = c("a", "b"), leg = c(T, F))
plotDistri(ex.traits3, rep("region", times = dim(ex.traits3)[1]), ex.sp3, plot.ask = F,
           multipanel = F, main = c("c", "d"), leg = c(T, F))
par(mfrow=c(1, 1))
```

We can also plot the result of the T-statistics for two contrasting cases (Fig. 15).

```
par(mfrow=c(2, 1))
plot(res.simu3[[1]], main = "a")
plot(res.simu3[[n]], main = "b")
par(mfrow=c(1, 1))
```

4.2.1 Local $T_{IC/IR}$ and $T_{PC/PR}$ results

As for internal filtering, we mingle the p-values for different parameter values. Consequently we plot the standard effect size (SES) of $T_{IC/IR}$ and $T_{PC/PR}$ as a function of the initial parameter values.

4.2.1.1 Local $T_{IC/IR}$ and $T_{PC/PR}$ SES values against initial parameter values

```
#T_IC.IR
meanSES.3loc.norm_Ticir <- unlist(lapply(res.simu3, function(x)
            ses.listofindex(as.listofindex(x))
            $index_1_2$ses[, 1]))

meanSES.3loc.uni_Ticir <- unlist(lapply(res.simu3, function(x)
            ses.listofindex(as.listofindex(x))
            $index_1_2$ses[, 2]))

SES.inf.MEAN.norm_Ticir <- unlist(lapply(res.simu3, function(x)
            ses.listofindex(as.listofindex(x))
            $index_1_2$ses.inf[, 1]))

SES.inf.MEAN.uni_Ticir <- unlist(lapply(res.simu3, function(x)
            ses.listofindex(as.listofindex(x))
            $index_1_2$ses.inf[, 2]))

#T_PC.PR
meanSES.3loc.norm_Tpcpr <- unlist(lapply(res.simu3, function(x)
            ses.listofindex(as.listofindex(x))
            $index_1_3$ses[, 1]))

meanSES.3loc.uni_Tpcpr <- unlist(lapply(res.simu3, function(x)
            ses.listofindex(as.listofindex(x))
```

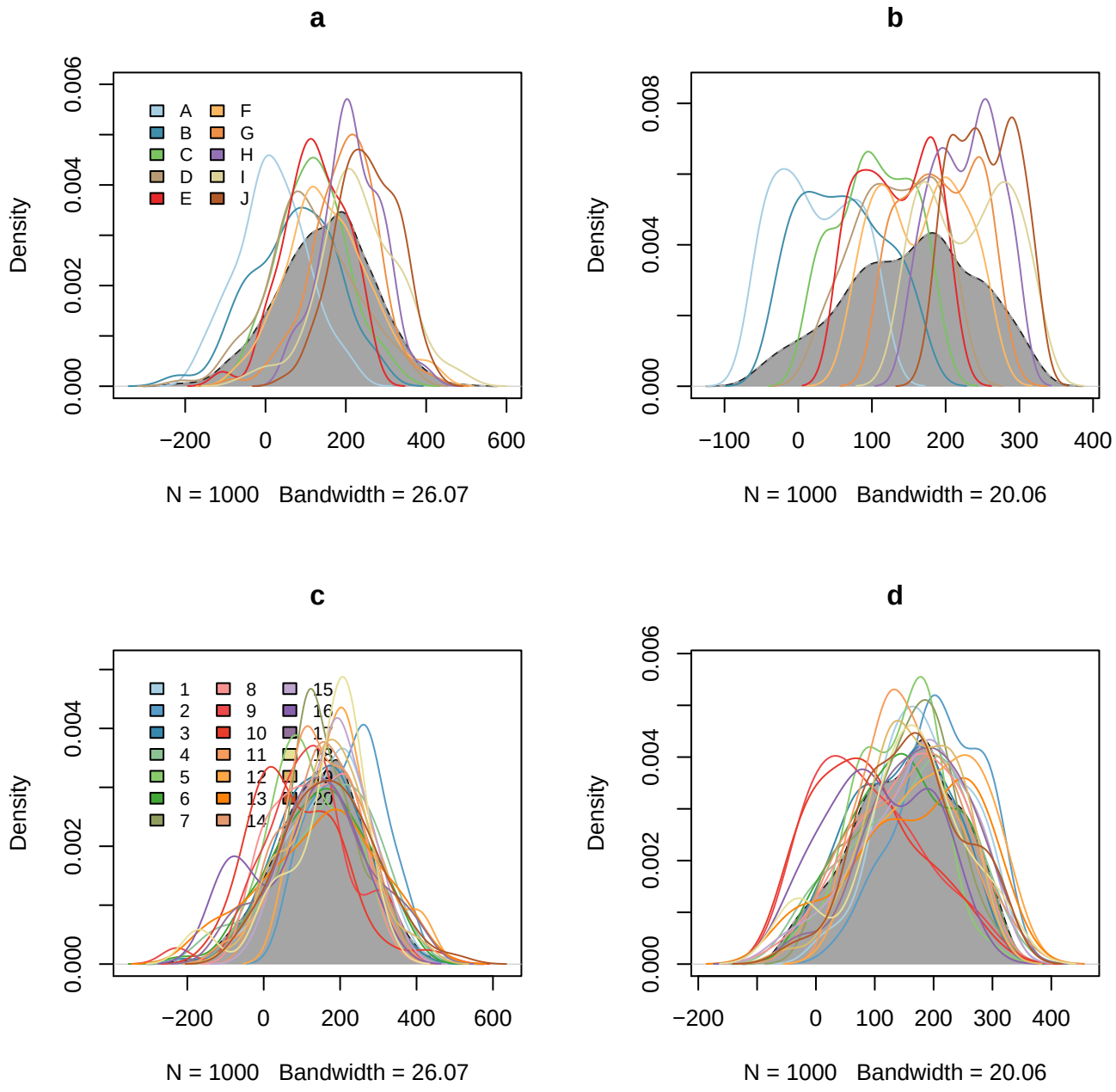


Figure 14: Distribution of trait values for one randomization with external filtering: (a) Community-level trait distributions for the trait a (normal distribution); (b) Community-level trait distributions for the trait b (uniform distribution). In panels a and b, each color represents one community (site). (c) Species' trait distributions for the trait a; (d) Species' trait distributions for the trait b. In panels c and d, each color represents one species.

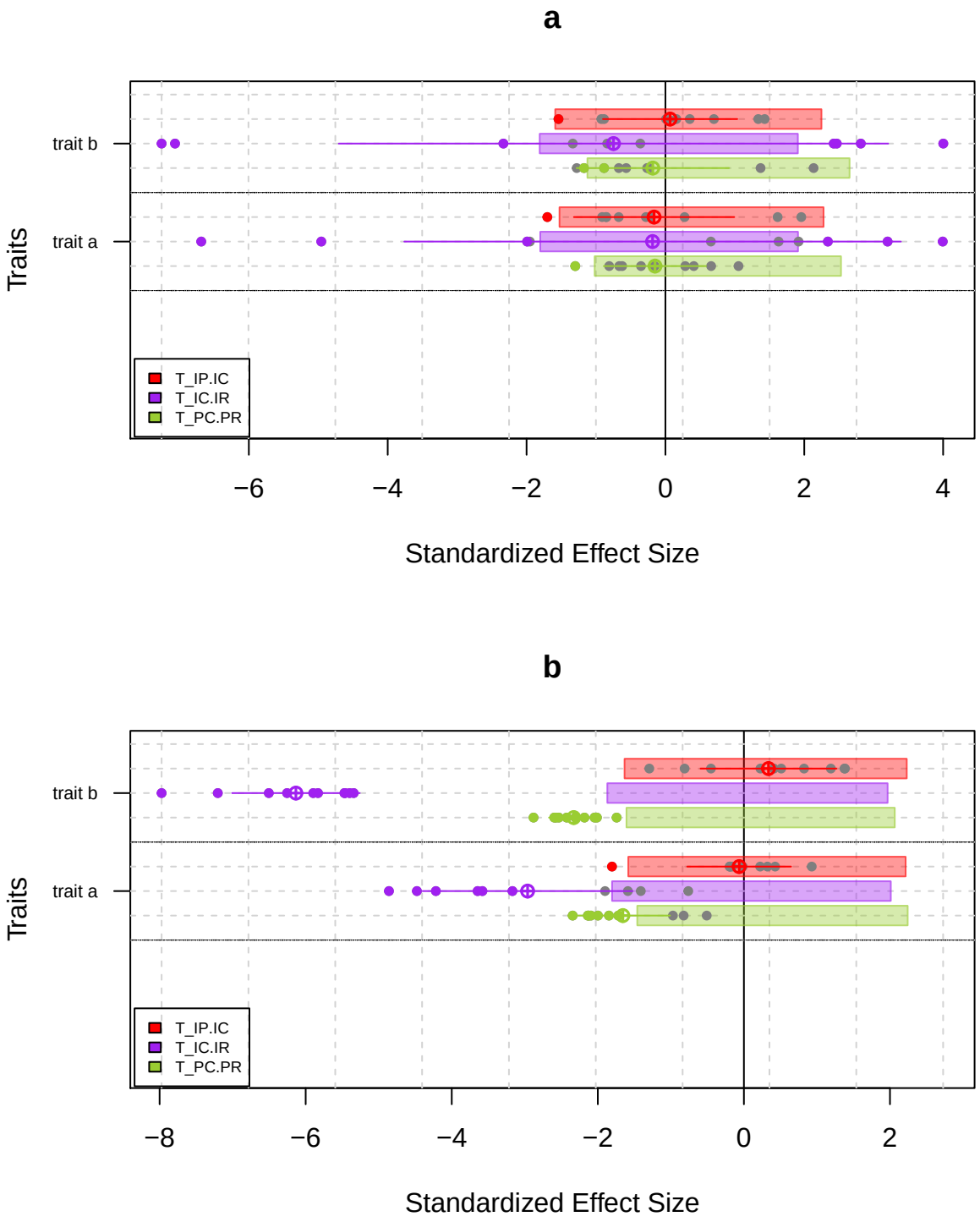


Figure 15: T-statistics for two randomizations with contrasted strengths of external filtering: $T_{IP/IC}$ in red, $T_{IC/IR}$ in purple and $T_{PC/PR}$ in green. (a) Lower strength of external filtering and (b) higher strength of external filtering.

```
            $index_1_3$ses[, 2]))

SES.inf.MEAN.norm_Tpcpr <- unlist(lapply(res.simu3, function(x)
            ses.listofindex(as.listofindex(x))
            $index_1_3$ses.inf[, 1]))

SES.inf.MEAN.uni_Tpcpr <- unlist(lapply(res.simu3, function(x)
            ses.listofindex(as.listofindex(x))
            $index_1_3$ses.inf[, 2]))
```

Now, we can plot SES of $T_{IC/IR}$ (Fig. 16) and $T_{PC/PR}$ (Fig. 17). The trait "a" normally distributed is in black and the uniform trait "b" is in purple. The colored rectangles represent the null model area with alpha = 5%. Thus, when a point is outside this area, the modeled parameters are strong enough to detect the external filter with a high power (beta-error < 0.05).

```
init_param <- param_range / sd_mean.param
#each value is replicated Ncom times (there are Ncom communities for each parameter values)
init_param.loc <- rep(init_param, each = Ncom)

plot(meanSES.3loc.norm_Ticir, init_param.loc, pch = 16, col = rgb(0, 0, 0, 0.6),
     xlim = c(min(c(unlist(meanSES.3loc.norm_Ticir), unlist(meanSES.3loc.uni_Ticir)),
     na.rm = T), 0),
     main = "Local T_IC.IR",
     xlab = "standardized effect size of T_IC.IR",
     ylab = "initial parameters: range/sd")
abline(v = mean(SES.inf.MEAN.norm_Ticir, na.rm = T))

points(meanSES.3loc.uni_Ticir, init_param.loc, pch = 16, col = rgb(0.5, 0, 1, 0.6))
abline(v = mean(SES.inf.MEAN.uni_Ticir, na.rm = T), col = "purple")
rect(mean(SES.inf.MEAN.norm_Ticir, na.rm = T), -1, 0, max(init_param) + 1,
     col = rgb(0, 0, 0, 0.3), border = NA)
rect(mean(SES.inf.MEAN.uni_Ticir, na.rm = T),0, 0, max(init_param),
     col = rgb(0.5, 0, 1, 0.3), border = NA)
```

```
plot(meanSES.3loc.norm_Tpcpr, init_param.loc, pch = 16, col = rgb(0, 0, 0, 0.6),
     xlim = c(min(c(unlist(meanSES.3loc.norm_Tpcpr), unlist(meanSES.3loc.uni_Tpcpr)),
     na.rm = T), 0),
     main = "Local T_PC.PR",
     xlab = "standardized effect size of T_PC.PR",
     ylab = "initial parameters: range/sd")
abline(v = mean(SES.inf.MEAN.norm_Tpcpr, na.rm = T))

points(meanSES.3loc.uni_Tpcpr, init_param.loc, pch = 16, col = rgb(0.5, 0, 1, 0.6))
abline(v = mean(SES.inf.MEAN.uni_Tpcpr, na.rm = T), col = "purple")
rect(mean(SES.inf.MEAN.norm_Tpcpr, na.rm = T), -1, 0, max(init_param) + 1,
     col = rgb(0, 0, 0, 0.3), border = NA)
rect(mean(SES.inf.MEAN.uni_Tpcpr, na.rm = T), 0, 0, max(init_param),
     col = rgb(0.5, 0, 1, 0.3), border = NA)
```

For each initial parameter values we can calculate the proportion of cases for which we do not reject the null hypothesis while this hypothesis is wrong (*i.e.* the Beta-error). The next chunk only shows the calculation for the $T_{IC/IR}$ metrics for the trait "a".

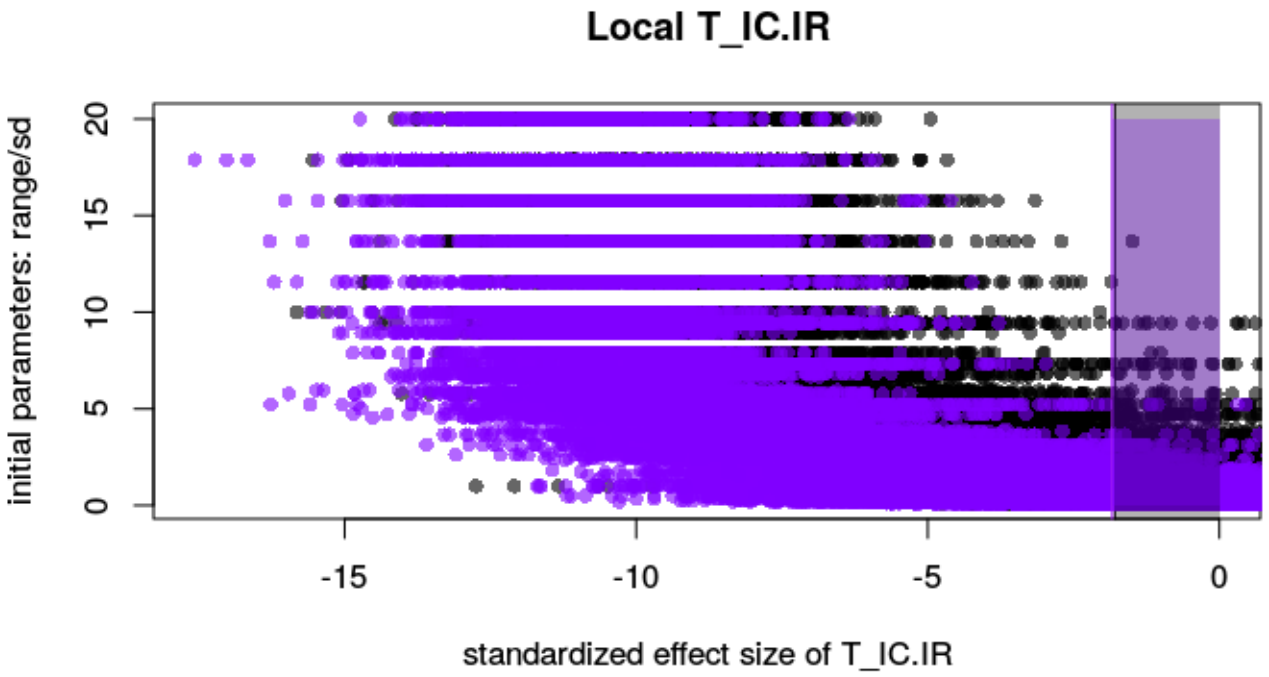


Figure 16: Local $T_{IC/IR}$ SES and initial parameter values: Standardized effect size of $T_{IC/IR}$ as a function of the strength of external filtering defined as the ratio of the initial range parameter by the initial standard error parameter. Colored area represents the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and trait b (uniform) in purple.

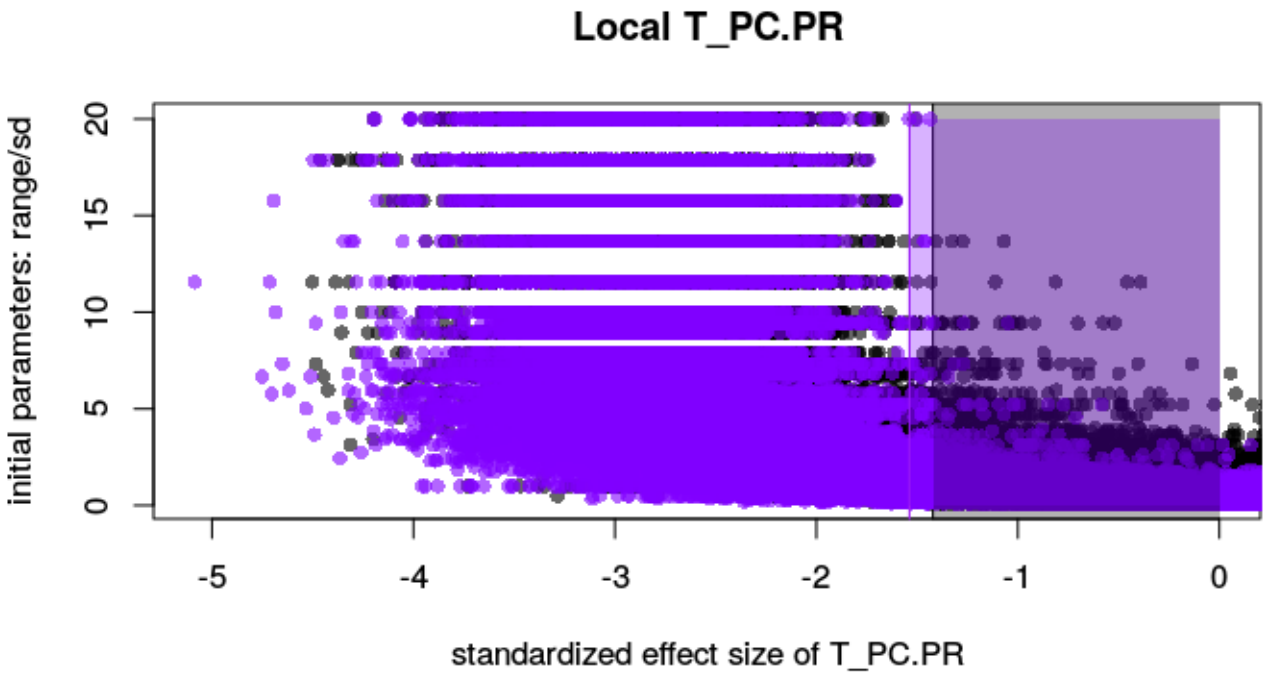


Figure 17: Local $T_{PC/PR}$ SES and initial parameter values: Standardized effect size of $T_{PC/PR}$ as a function of the strength of external filtering defined as the ratio of the initial range parameter by the initial standard error parameter. Colored area represents the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and trait b (uniform) in purple.

```
beta_range_norm_Ticir <-
 (unlist(lapply(by(cbind(meanSES.3loc.norm_Ticir, SES.inf.MEAN.norm_Ticir),
     rep(param_range, each = Ncom), function(x) x[,1] > x[,2]),
     function(x) sum(x, na.rm = T)))+1)/(10 * N_repet_Param + 1)
```

```
print(xtable(res_beta_Ticir, caption = 'Local beta-error of $T_{IC/IR}$ as a function
             of the strength of external filtering. str: strength. Trait a is
             normally distributed and trait b is uniformally distributed.',
             label = "tab:local_beta_error_xtable_ticir"),
      caption.placement = "top", size = "small")
```

Table 4: Local beta-error of $T_{IC/IR}$ as a function of the strength of external filtering. str: strength. Trait a is normally distributed and trait b is uniformally distributed.

	str 1	str 2	str 3	str 4	str 5	str 6	str 7	str 8	str 9	str 10
effect of sd (a)	0.65	0.61	0.53	0.46	0.41	0.36	0.28	0.22	0.13	0.07
effect of sd (b)	0.36	0.30	0.25	0.21	0.19	0.16	0.14	0.09	0.06	0.02
effect of range (a)	0.66	0.63	0.57	0.47	0.41	0.32	0.25	0.19	0.13	0.09
effect of range (b)	0.53	0.47	0.35	0.22	0.12	0.06	0.02	0.01	0.00	0.00

```
print(xtable(res_beta_Ticir, caption = 'Local beta-error of $T_{PC/PR}$ as a function
             of the strength of external filtering. str: strength. Trait a is
             normally distributed and trait b is uniformally distributed.',
             label = "tab:local_beta_error_xtable_tpcpr"),
      caption.placement = "top", size = "small")
```

Table 5: Local beta-error of $T_{PC/PR}$ as a function of the strength of external filtering. str: strength. Trait a is normally distributed and trait b is uniformally distributed.

	str 1	str 2	str 3	str 4	str 5	str 6	str 7	str 8	str 9	str 10
effect of sd (a)	0.65	0.61	0.53	0.46	0.41	0.36	0.28	0.22	0.13	0.07
effect of sd (b)	0.36	0.30	0.25	0.21	0.19	0.16	0.14	0.09	0.06	0.02
effect of range (a)	0.66	0.63	0.57	0.47	0.41	0.32	0.25	0.19	0.13	0.09
effect of range (b)	0.53	0.47	0.35	0.22	0.12	0.06	0.02	0.01	0.00	0.00

Now we can see the beta-error for each strength (Tables 4 and 5). A high value of range and a low value of standard error defined a high strength of filtering. For example, in the case of a trait normally distributed, the proportion of false negative is 65.11% for the higher value of standard errors and thus for the lower strength of external filtering.

4.2.1.2 Local $T_{IC/IR}$ and $T_{PC/PR}$ SES values against modeled parameter values

First we need to calculate the modeled parameter values. Here again we calculate the strength of external filtering by dividing the range of values between communities by their standard error.

```
mean_range_com.interm <- t(matrix(unlist(lapply(mean_range_com, function(x) x)), nrow = 2))
mean_sd_of_com <- unlist(lapply(sd.com_stock3, function(x) mean(x)))

modeled_param_norm <- mean_range_com.interm[, 1] / mean_sd_of_com
modeled_param_uni <- mean_range_com.interm[, 2] / mean_sd_of_com
modeled_param_norm.loc <- rep(modeled_param_norm, each = Ncom)
modeled_param_uni.loc <- rep(modeled_param_uni, each = Ncom)
```

```
plot(meanSES.3loc.norm_Ticir, modeled_param_norm.loc, pch = 16, col = rgb(0, 0, 0, 0.6),
  xlim = c(min(c(unlist(meanSES.3loc.norm_Ticir), unlist(meanSES.3loc.uni_Ticir)),
    na.rm = T), 0),
    ylim = c(0, min(max(c(modeled_param_norm.loc, modeled_param_uni.loc), na.rm = T),
                    100)),  main = "Local T_IC.IR",
  xlab = "standardized effect size of T_IC.IR",
  ylab = "modeled parameters: range/sd")
abline(v = mean(SES.inf.MEAN.norm_Ticir, na.rm = T))

points(meanSES.3loc.uni_Ticir, modeled_param_uni.loc, pch = 16, col = rgb(0.5, 0, 1, 0.6))
abline(v = mean(SES.inf.MEAN.uni_Ticir, na.rm = T), col = "purple")

rect(mean(SES.inf.MEAN.norm_Ticir, na.rm = T), -1, 0, max(modeled_param_norm.loc) + 1,
     col = rgb(0, 0, 0, 0.3), border = NA)
rect(mean(SES.inf.MEAN.uni_Ticir, na.rm = T),0, 0, max(modeled_param_uni.loc),
     col = rgb(0.5, 0, 1, 0.3), border = NA)
```

```
plot(meanSES.3loc.norm_Tpcpr, modeled_param_norm.loc, pch = 16, col = rgb(0, 0, 0, 0.6),
     xlim = c(min(c(unlist(meanSES.3loc.norm_Tpcpr), unlist(meanSES.3loc.uni_Tpcpr)),
              na.rm = T), 0),
     ylim = c(0, min(max(c(modeled_param_norm.loc, modeled_param_uni.loc), na.rm = T),
                     100)),
     main = "Local T_PC.PR",
     xlab = "standardized effect size of T_PC.PR",
     ylab = "modeled parameters: range/sd")
abline(v = mean(SES.inf.MEAN.norm_Tpcpr, na.rm = T))

points(meanSES.3loc.uni_Tpcpr, modeled_param_uni.loc, pch = 16, col = rgb(0.5, 0, 1, 0.6))
abline(v = mean(SES.inf.MEAN.uni_Tpcpr, na.rm = T), col = "purple")
rect(mean(SES.inf.MEAN.norm_Tpcpr, na.rm = T), -1, 0, max(modeled_param_norm.loc) + 1,
     col = rgb(0, 0, 0, 0.3), border = NA)
rect(mean(SES.inf.MEAN.uni_Tpcpr, na.rm = T), 0, 0, max(modeled_param_uni.loc),
     col = rgb(0.5, 0, 1, 0.3), border = NA)
```

There is a very hich correlation coefficient between initial and modeled parameters either for the range (trait a: Pearson correlation = 0.225; trait b: Pearson correlation = 0.341) and for the standard error(trait a and b: Pearson correlation = 0.994). Consequently the consistency between the figures 16 - 17 and 18 - 19 is not suprising.

4.2.2 Global $T_{IC/IR}$ and $T_{PC/PR}$ results

In contrast with local p-values, we call global p-values the p-values corresponding to one index for one trait across all the communities.

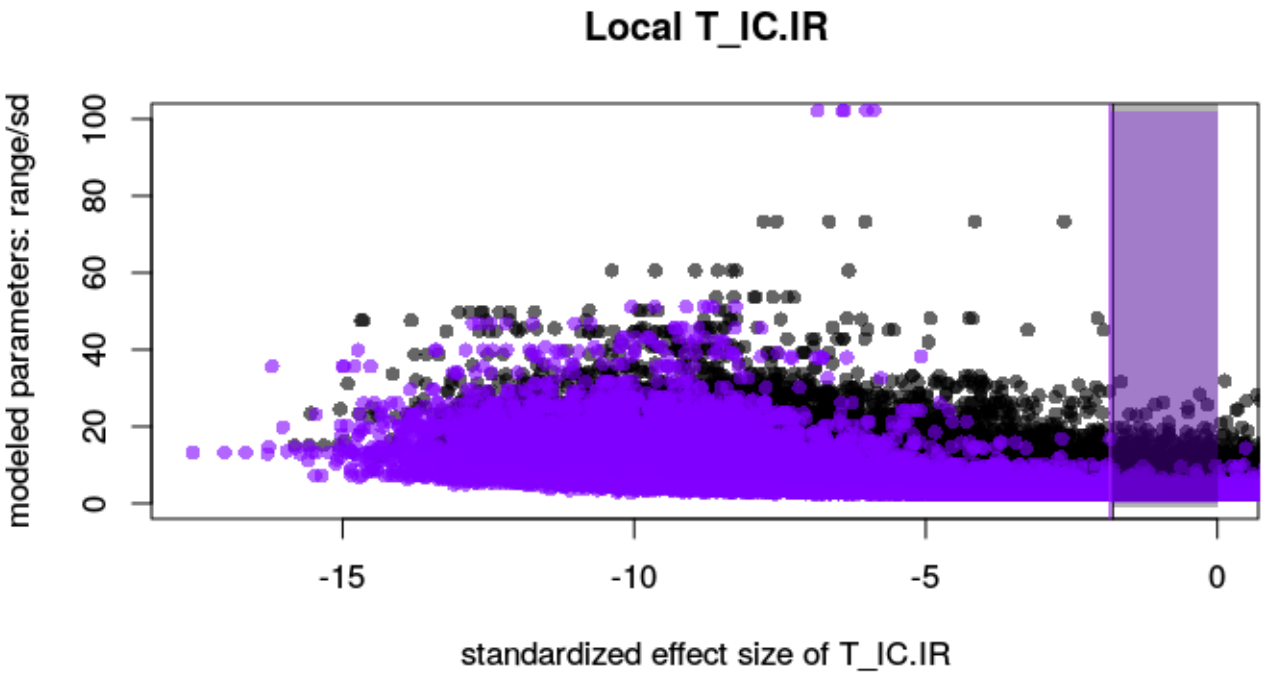


Figure 18: Local $T_{IC/IR}$ SES and modeled parameter values: Standardized effect size of $T_{IC/IR}$ as a function of the strength of external filtering defined as the ratio of the modeled range parameter by the modeled standard error parameter. Colored area represents the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and trait b (uniform) in purple.

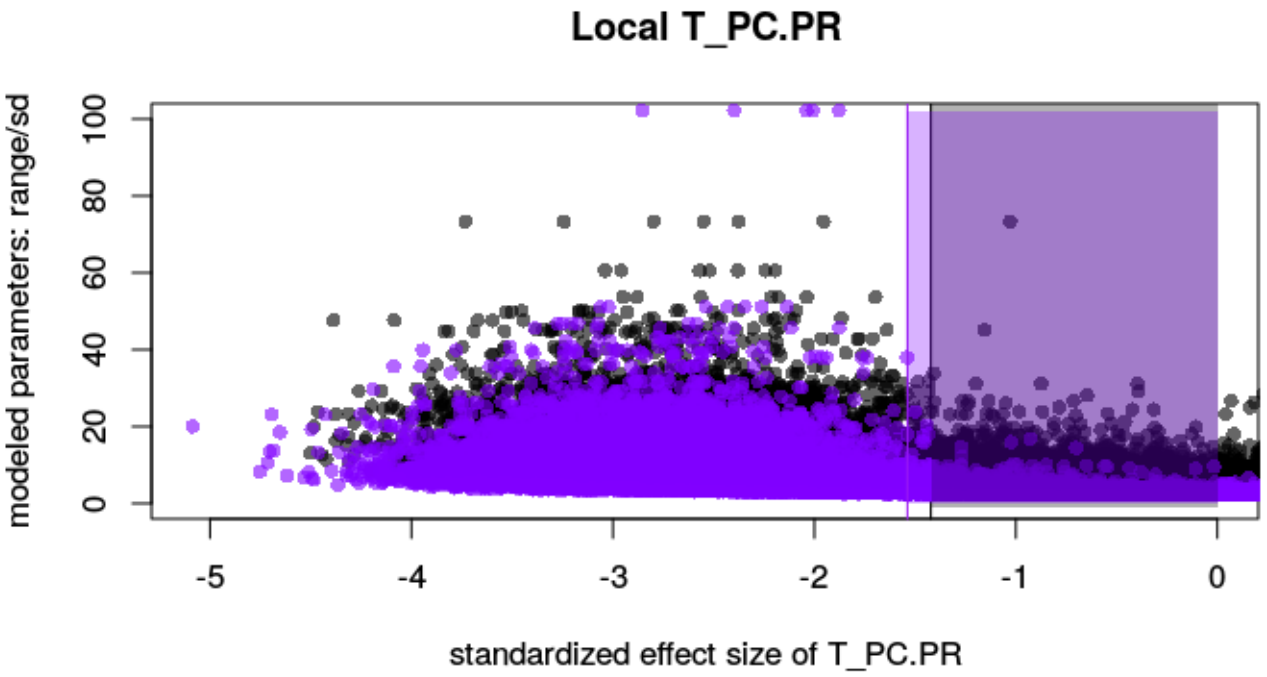


Figure 19: Local $T_{PC/PR}$ SES and modeled parameter values: Standardized effect size of $T_{PC/PR}$ as a function of the strength of external filtering defined as the ratio of the modeled range parameter by the modeled standard error parameter. Colored area represents the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and trait b (uniform) in purple.

4.2.2.1 Calculation of beta-error mixing all strengths of external filtering

To illustrate the global power of $T_{IC/IR}$ and $T_{PC/PR}$ you can plot the ordered p-values and calculate a beta-error thanks to the code below. It is just an illustration because we mingle very different strengths of external filter with different initial parameter values. Plotted results are not shown in this document.

```
par(mfrow = c(2, 2))
xx <- log10(sort(unlist(lapply(res.simu3.pval, function(x) x[21:30, 1]))))
plot(xx, type = "l", main = "T_IC.IR_distriNorm")
abline(h = log10(0.05))
nbre_beta_error <- round((sum(xx>log10(0.05))+1)/(length(xx)+1), 4)
text(0, -0.8, labels = paste("beta error", nbre_beta_error, sep = " = "), cex = 0.7, pos = 4)

xx <- log10(sort(unlist(lapply(res.simu3.pval, function(x) x[41:50, 1]))))
plot(xx, type = "l", main = "T_PC.PR_distriNorm")
abline(h = log10(0.05))
nbre_beta_error <- round((sum(xx>log10(0.05))+1)/(length(xx)+1), 4)
text(0, -0.8, labels = paste("beta error", nbre_beta_error, sep = " = "), cex = 0.7, pos = 4)


xx <- log10(sort(unlist(lapply(res.simu3.pval, function(x) x[21:30, 2]))))
plot(xx, type = "l", main = "T_IC.IR_distriUni")
abline(h = log10(0.05))
nbre_beta_error <- round((sum(xx>log10(0.05))+1)/(length(xx)+1), 4)
text(0, -0.8, labels = paste("beta error", nbre_beta_error, sep = " = "), cex = 0.7, pos = 4)

xx <- log10(sort(unlist(lapply(res.simu3.pval, function(x) x[41:50, 2]))))
plot(xx, type = "l", main = "T_PC.PR_distriUni")
abline(h = log10(0.05))
nbre_beta_error <- round((sum(xx>log10(0.05))+1)/(length(xx)+1), 4)
text(0, -0.8, labels = paste("beta error", nbre_beta_error, sep = " = "), cex = 0.7, pos = 4)

par(mfrow = c(1, 1))
```

4.2.2.2 Global $T_{IC/IR}$ and $T_{PC/PR}$ SES values against initial parameter values

Again, these first results mix different initial parameter values. Now, we can plot the SES values in relation to the strength of the external filter assessed by the two parameters (either the initial values: `mean_range_between_com` and `mean_sd_of_com` or the modeled values `mean_range_com` and `sd.com_stock3` [11]).

First, we need to compute the SES values from simulations.

```
meanSES.3glob.norm_Ticir <- unlist(lapply(res.simu3, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_2$ses[,1], na.rm = T)))
meanSES.3glob.uni_Ticir <- unlist(lapply(res.simu3, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_2$ses[,2], na.rm = T)))
meanSES.3glob.norm_Tpcpr <- unlist(lapply(res.simu3, function(x)
```

[11] These modeled values are stochastic versions of the initial values.

```
          mean(ses.listofindex(as.listofindex(x))
            $index_1_3$ses[,1], na.rm = T)))
meanSES.3glob.uni_Tpcpr <- unlist(lapply(res.simu3, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_3$ses[,2], na.rm = T)))

meanSES.INF_glob.norm_Ticir <- unlist(lapply(res.simu3, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_2$ses.inf[, 1], na.rm = T)))
meanSES.INF_glob.uni_Ticir <- unlist(lapply(res.simu3, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_2$ses.inf[, 2], na.rm = T)))
meanSES.INF_glob.norm_Tpcpr <- unlist(lapply(res.simu3, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_3$ses.inf[, 1], na.rm = T)))
meanSES.INF_glob.uni_Tpcpr <- unlist(lapply(res.simu3, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_3$ses.inf[, 2], na.rm = T)))
```

Plot the result against initial parameters (Fig. 20 and 21).

```
init_param <- param_range / sd_mean.param

plot(meanSES.3glob.norm_Ticir, init_param, pch = 16, col = rgb(0, 0, 0, 0.6),
     main = "Global T_IC.IR",
     xlim = c(min(c(meanSES.3glob.norm_Ticir, meanSES.3glob.uni_Ticir),
              na.rm = T), 0),
     xlab = "standardized effect size of T_IC.IR",
     ylab = "initial parameters: range/sd")
points(meanSES.3glob.uni_Ticir, init_param, pch = 16, col = rgb(0.5, 0, 1, 0.6),)
abline(v = mean(meanSES.INF_glob.norm_Ticir, na.rm = T))
abline(v = mean(meanSES.INF_glob.uni_Ticir, na.rm = T), col = "purple")
rect(mean(meanSES.INF_glob.norm_Ticir, na.rm = T), -1, 0, max(modeled_param_norm.loc) + 1,
     col = rgb(0, 0, 0, 0.3), border = NA)
rect(mean(meanSES.INF_glob.uni_Ticir, na.rm = T), 0, 0, max(modeled_param_uni.loc),
     col = rgb(0.5, 0, 1, 0.3), border = NA)
```

```
plot(meanSES.3glob.norm_Tpcpr, init_param, pch = 16, col = rgb(0, 0, 0, 0.6),
     main = "Global T_PC.PR",
     xlim = c(min(c(meanSES.3glob.norm_Tpcpr, meanSES.3glob.uni_Tpcpr),
              na.rm = T), 0),
     xlab = "standardized effect size of T_PC.PR",
     ylab = "initial parameters: range/sd")
points(meanSES.3glob.uni_Tpcpr, init_param, pch = 16, col = rgb(0.5, 0, 1, 0.6))
abline(v = mean(meanSES.INF_glob.norm_Tpcpr, na.rm = T))
abline(v = mean(meanSES.INF_glob.uni_Tpcpr, na.rm = T), col = "purple")
rect(mean(meanSES.INF_glob.norm_Tpcpr, na.rm = T), -1, 0, max(modeled_param_norm.loc) + 1,
     col = rgb(0, 0, 0, 0.3), border = NA)
rect(mean(meanSES.INF_glob.uni_Tpcpr, na.rm = T), 0, 0, max(modeled_param_uni.loc),
     col = rgb(0.5, 0, 1, 0.3), border = NA)
```

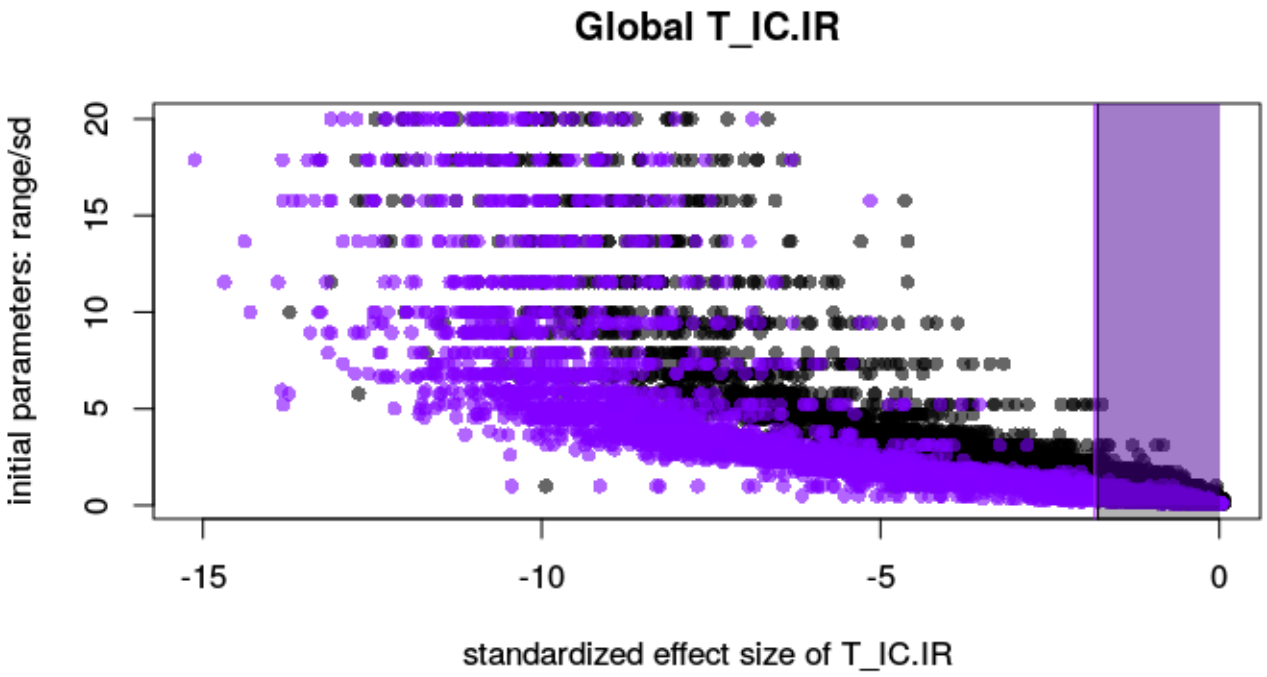


Figure 20: Global $T_{IC/IR}$ SES and initial parameter values: Standardized effect size of $T_{IC/IR}$ as a function of the strength of external filtering defined as the ratio of the inital range parameter by the initial standard error parameter. Colored area represents the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and trait b (uniform) in purple.

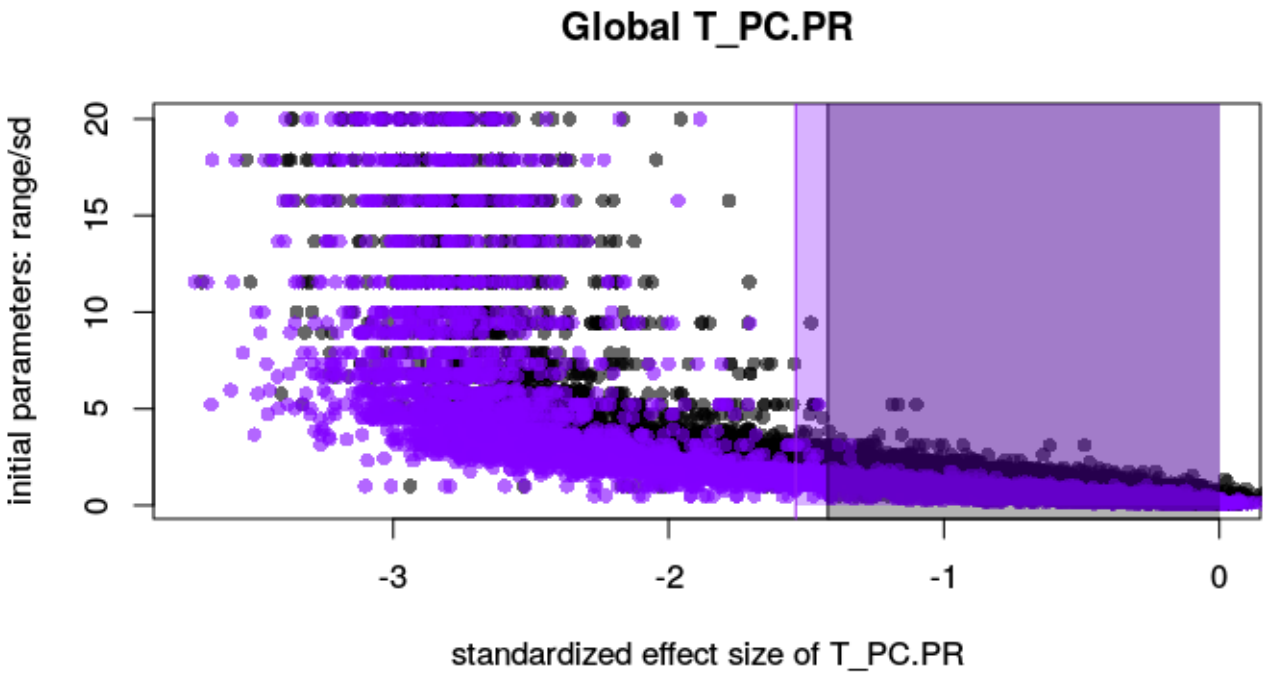


Figure 21: Global $T_{PC/PR}$ SES and initial parameter values: Standardized effect size of $T_{PC/PR}$ as a function of the strength of external filtering defined as the ratio of the inital range parameter by the initial standard error parameter. Colored area represents the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and trait b (uniform) in purple.

4.2.2.3 Global $T_{IC/IR}$ and $T_{PC/PR}$ SES values against modeled parameter values

We can verify the consistency of our results by plotting SES values against the modeled parameter values (Fig. 22 and 23).

```
mean_range_com.interm <- t(matrix(unlist(lapply(mean_range_com, function(x) x)), nrow = 2))
mean_sd_of_com <- unlist(lapply(sd.com_stock3, function(x) mean(x)))

modeled_param_norm <- mean_range_com.interm[, 1] / mean_sd_of_com
modeled_param_uni <- mean_range_com.interm[, 2] / mean_sd_of_com

plot(meanSES.3glob.norm_Ticir, modeled_param_norm, pch = 16,
  main = "Global T_IC.IR",
  xlim = c(min(c(meanSES.3glob.norm_Ticir, meanSES.3glob.uni_Ticir), na.rm = T), 0),
  ylim = c(0, min(max(c(modeled_param_norm, modeled_param_uni), na.rm = T), 100)),
  xlab = "standardized effect size of T_IC.IR",
  ylab = "modeled parameters: range/sd")
points(meanSES.3glob.uni_Ticir, modeled_param_uni, pch = 16, col = rgb(0.5, 0, 1, 0.6))
abline(v = mean(meanSES.INF_glob.norm_Ticir, na.rm = T))
abline(v = mean(meanSES.INF_glob.uni_Ticir, na.rm = T), col = "purple")
rect(mean(meanSES.INF_glob.norm_Ticir, na.rm = T), -1, 0, max(modeled_param_norm) + 1,
  col = rgb(0, 0, 0, 0.3), border = NA)
rect(mean(meanSES.INF_glob.uni_Ticir, na.rm = T), 0, 0, max(modeled_param_uni),
  col = rgb(0.5, 0, 1, 0.3), border = NA)
```

```
plot(meanSES.3glob.norm_Tpcpr, modeled_param_norm, pch = 16,
  main = "Global T_PC.PR",
  xlim = c(min(c(meanSES.3glob.norm_Tpcpr, meanSES.3glob.uni_Tpcpr),
    na.rm = T), 0),
  ylim = c(0, min(max(c(modeled_param_norm, modeled_param_uni), na.rm = T), 100)),
  xlab = "standardized effect size of T_PC.PR",
  ylab = "modeled parameters: range/sd")
points(meanSES.3glob.uni_Tpcpr, modeled_param_uni, pch = 16, col = rgb(0.5, 0, 1, 0.6))
abline(v = mean(meanSES.INF_glob.norm_Tpcpr, na.rm = T))
abline(v = mean(meanSES.INF_glob.uni_Tpcpr, na.rm = T), col = "purple")
rect(mean(meanSES.INF_glob.norm_Tpcpr, na.rm = T), -1, 0, max(modeled_param_norm) + 1,
  col = rgb(0, 0, 0, 0.3), border = NA)
rect(mean(meanSES.INF_glob.uni_Tpcpr, na.rm = T), 0, 0, max(modeled_param_uni),
  col = rgb(0.5, 0, 1, 0.3), border = NA)
```

4.3 Conclusion on the power of $T_{IC/IR}$ to detect external filtering

To conclude on the power of $T_{IC/IR}$ to detect external filtering, we apply an exponential linear model and identify the strength from which the beta-error is inferior to 0.05. This strength is defined as the ratio of the initial parameters `param_range` and `sd_mean.param`.

```
plot(meanSES.3loc.norm_Ticir ~ init_param.loc, col = rgb(0, 0, 0, 0.2),
     ylim = c(min(c(meanSES.3loc.norm_Ticir , meanSES.3loc.uni_Ticir,
                    meanSES.3glob.norm_Ticir, meanSES.3glob.uni_Ticir), na.rm = T), 0),
     main = "Global T_IC.IR",
```

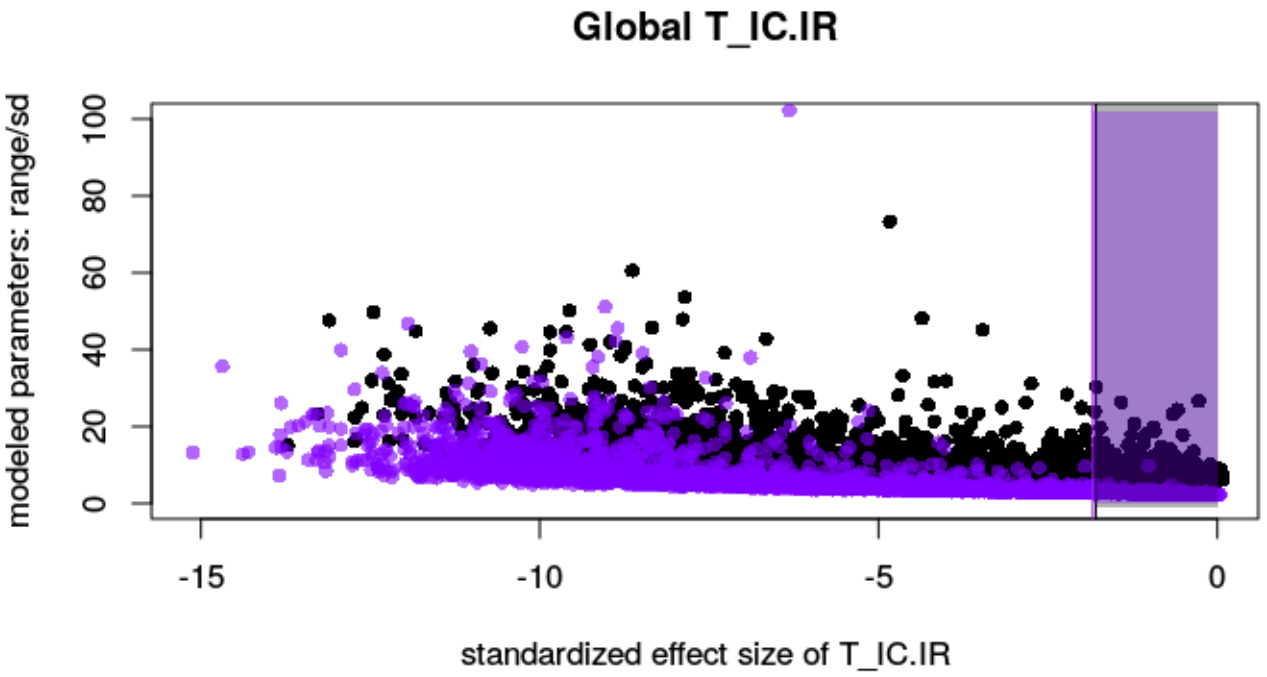


Figure 22: Global $T_{IC/IR}$ SES and modeled parameter values: Standardized effect size of $T_{IC/IR}$ as a function of the strength of external filtering defined as the ratio of the modeled range parameter by the modeled standard error parameter. Colored area represents the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and trait b (uniform) in purple.

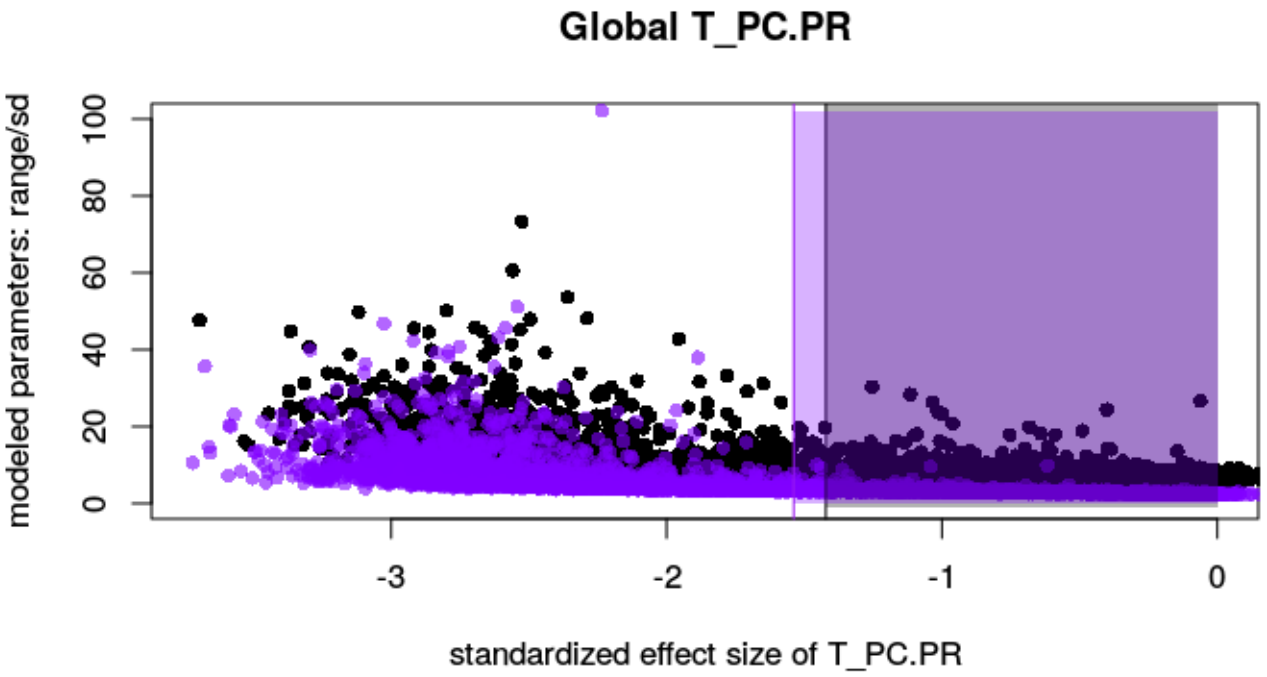


Figure 23: Global $T_{PC/PR}$ SES and modeled parameter values: Standardized effect size of $T_{PC/PR}$ as a function of the strength of external filtering defined as the ratio of the modeled range parameter by the modeled standard error parameter. Colored area represents the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and trait b (uniform) in purple.

```
     ylab = "standardized effect size of T_IC.IR",
     xlab = "initial parameters: range/sd",
     type = "n")
#points(meanSES.3loc.uni_Ticir ~ init_param.loc, col = rgb(0.5, 0, 1, 0.2))

points(meanSES.3glob.norm_Ticir ~ init_param, pch = 16, col = rgb(0, 0, 0, 0.5),
       cex = 1.2)
lm.norm <- lm(meanSES.3glob.norm_Ticir ~ log(init_param))
lm.norm_conf <- confint(lm.norm, level = 0.90)
curve(lm.norm$coef[1] + log(x) * (lm.norm$coef[2]), add = T, lwd = 3,
      col = rgb(0, 0, 0, 1))
curve(lm.norm_conf[1, 1] + log(x) * lm.norm_conf[2, 1], add = T, lty = 2,
      col = rgb(0, 0, 0, 1))
curve(lm.norm_conf[1, 2] + log(x) * lm.norm_conf[2, 2], add = T, lty = 2,
      col = rgb(0, 0, 0, 1))

points(meanSES.3glob.uni_Ticir ~ init_param, pch = 16, col = rgb(0.5, 0, 1, 0.5),
       cex = 1.2)
lm.uni <- lm(meanSES.3glob.uni_Ticir ~ log(init_param))
lm.uni_conf <- confint(lm.uni, level = 0.90)
curve(lm.uni$coef[1] + log(x) * (lm.uni$coef[2]), add = T, lwd = 3,
      col = rgb(0.5, 0, 1, 1))
curve(lm.uni_conf[1, 1] + log(x) * lm.uni_conf[2, 1], add = T, lty = 2,
      col = rgb(0.5, 0, 1, 1))
curve(lm.uni_conf[1, 2] + log(x) * lm.uni_conf[2, 2], add = T, lty = 2,
      col = rgb(0.5, 0, 1, 1))


rect(-1, mean(c(SES.inf.MEAN.uni_Ticir, meanSES.INF_glob.uni_Ticir), na.rm = T),
     max(init_param), 10, col = rgb(0.5, 0, 1, 0.3), border = NA)

rect(-1, mean(c(SES.inf.MEAN.norm_Ticir, meanSES.INF_glob.norm_Ticir), na.rm = T),
     max(init_param) + 10, 10, col = rgb(0, 0, 0, 0.3), border = NA)
```

```
yy.norm <- mean(c(SES.inf.MEAN.norm_Ticir, meanSES.INF_glob.norm_Ticir), na.rm = T)
yy.uni <- mean(c(SES.inf.MEAN.uni_Ticir, meanSES.INF_glob.uni_Ticir), na.rm = T)

param_beta0.05_Ticir.norm <- exp((yy.norm - lm.norm_conf[1, 2]) / lm.norm_conf[2, 2])
param_beta0.05_Ticir.uni <- exp((yy.uni - lm.uni_conf[1, 2]) / lm.uni_conf[2, 2])
```

For the trait "a" normally distributed, the power of $T_{IC/IR}$ is satisfactory if the ratio of the initial parameter is superior to 1. For the trait "b" uniformally distributed, this value is 0.49. Further investigation on real data with different strengths of external filtering are needed to extend these simulated values.

4.4 Conclusion on the power of $T_{PC/PR}$ to detect external filtering

To conclude on the power of $T_{PC/PR}$ to detect external filtering, we apply an exponential linear model and identify the strength from which the beta-error is inferior to 0.05. This strength is defined as the ratio of the initial parameters `param_range` and `sd_mean.param`.

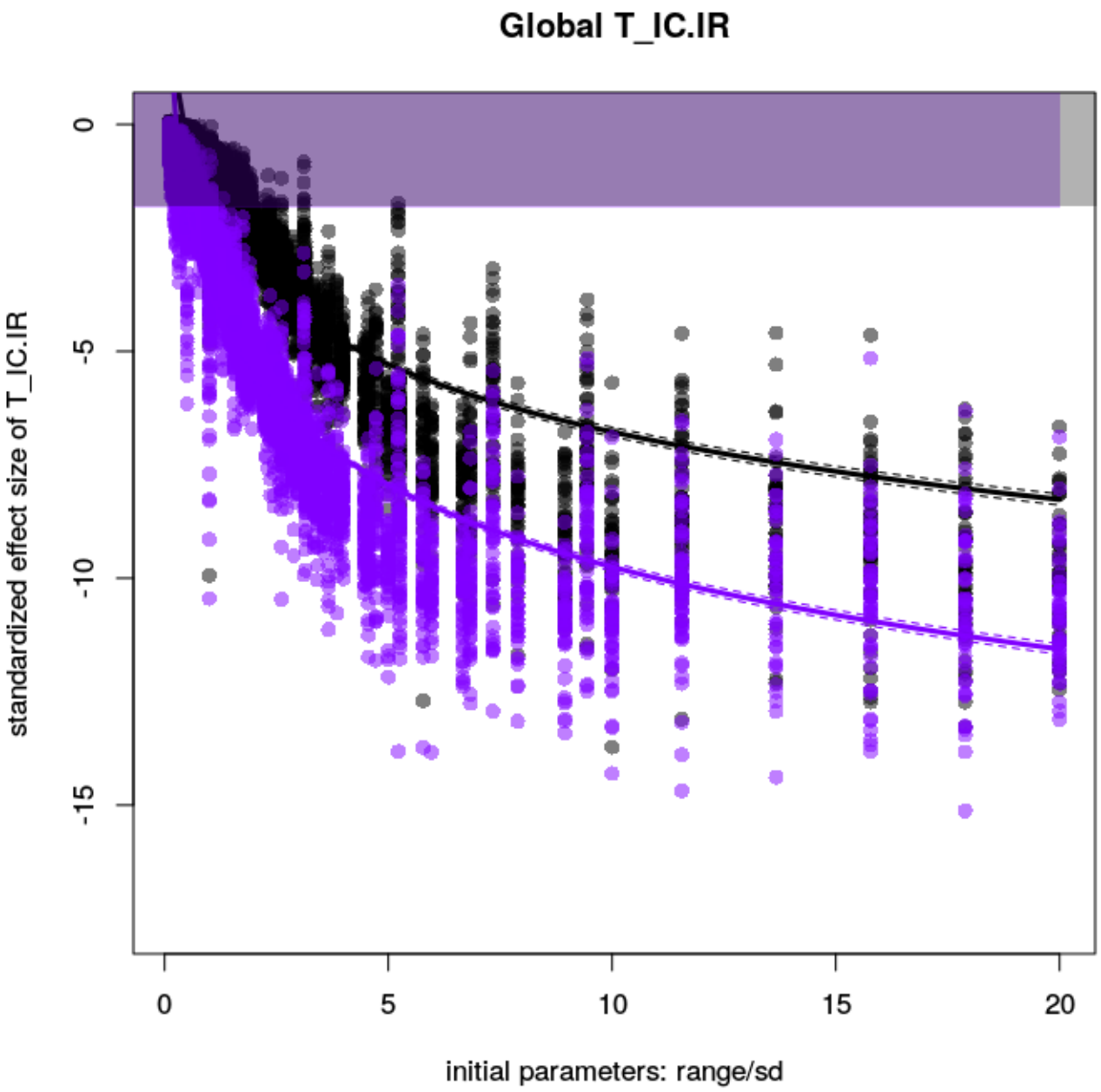


Figure 24: Power of $T_{IC/IR}$ to detect external filtering: Standardised Effect Size (SES) of $T_{IC/IR}$ as a function of the strength of external filtering. Trait 'a' normally distributed is shown in black, trait 'b' uniformally distributed in purple. Dots represent global SES values. The exponential linear model is presented with 0.05 confidence interval on both sides. Colored area represents the mean confidence interval of SES values ($alpha = 0.95$)

```
plot(meanSES.3loc.norm_Tpcpr ~ init_param.loc, col = rgb(0, 0, 0, 0.2),
     ylim = c(min(c(meanSES.3loc.norm_Tpcpr , meanSES.3loc.uni_Tpcpr,
                    meanSES.3glob.norm_Tpcpr, meanSES.3glob.uni_Tpcpr), na.rm = T), 0),
     main = "Global T_PC.IR",
     ylab = "standardized effect size of T_PC.IR",
     xlab = "initial parameters: range/sd",
     type = "n")
#points(meanSES.3loc.uni_Tpcpr ~ init_param.loc, col = rgb(0.5, 0, 1, 0.2))

points(meanSES.3glob.norm_Tpcpr ~ init_param, pch = 16, col = rgb(0, 0, 0, 0.5),
       cex = 1.2)
lm.norm <- lm(meanSES.3glob.norm_Tpcpr ~ log(init_param))
lm.norm_conf <- confint(lm.norm, level = 0.90)
curve(lm.norm$coef[1] + log(x) * (lm.norm$coef[2]), add = T, lwd = 3,
      col = rgb(0, 0, 0, 1))
curve(lm.norm_conf [1, 1] + log(x) * lm.norm_conf [2, 1], add = T, lty = 2,
      col = rgb(0, 0, 0, 1))
curve(lm.norm_conf [1, 2] + log(x) * lm.norm_conf [2, 2], add = T, lty = 2,
      col = rgb(0, 0, 0, 1))

points(meanSES.3glob.uni_Tpcpr ~ init_param, pch = 16, col = rgb(0.5, 0, 1, 0.5),
       cex = 1.2)
lm.uni <- lm(meanSES.3glob.uni_Tpcpr ~ log(init_param))
lm.uni_conf <- confint(lm.uni, level = 0.90)
curve(lm.uni$coef[1] + log(x) * (lm.uni$coef[2]), add = T, lwd = 3,
      col = rgb(0.5, 0, 1, 1))
curve(lm.uni_conf [1, 1] + log(x) * lm.uni_conf [2, 1], add = T, lty = 2,
      col = rgb(0.5, 0, 1, 1))
curve(lm.uni_conf [1, 2] + log(x) * lm.uni_conf [2, 2], add = T, lty = 2,
      col = rgb(0.5, 0, 1, 1))


rect(-1, mean(c(SES.inf.MEAN.uni_Tpcpr, meanSES.INF_glob.uni_Tpcpr), na.rm = T),
     max(init_param), 10, col = rgb(0.5, 0, 1, 0.3), border = NA)

rect(-1, mean(c(SES.inf.MEAN.norm_Tpcpr, meanSES.INF_glob.norm_Tpcpr), na.rm = T),
     max(init_param) + 10, 10, col = rgb(0, 0, 0, 0.3), border = NA)
```

```
yy.norm <- mean(c(SES.inf.MEAN.norm_Tpcpr, meanSES.INF_glob.norm_Tpcpr), na.rm = T)
yy.uni <- mean(c(SES.inf.MEAN.uni_Tpcpr, meanSES.INF_glob.uni_Tpcpr), na.rm = T)

param_beta0.05_Tpcpr.norm <- exp((yy.norm - lm.norm_conf [1, 2]) / lm.norm_conf[2, 2])
param_beta0.05_Tpcpr.uni <- exp((yy.uni - lm.uni_conf [1, 2]) / lm.uni_conf[2, 2])
```

For the trait "a" normally distributed, the power of $T_{PC/PR}$ is satisfactory if the ratio of the initial parameter is superior to 2.31. For the trait "b" uniformally distributed, this value is 1.27. Further investigation on real data with different strengths of external filtering are needed to extend these simulated values. As expected, $T_{PC/PR}$ is far less powerful than $T_{IC/IR}$ to detect external filtering for both traits "a" (2.31 *vs* 1) and "b" (1.27 *vs* 0.49). This is due to the loss of information in $T_{PC/PR}$ when we compute the mean by population and exclude intra-populationnal variation from the test of external filtering.

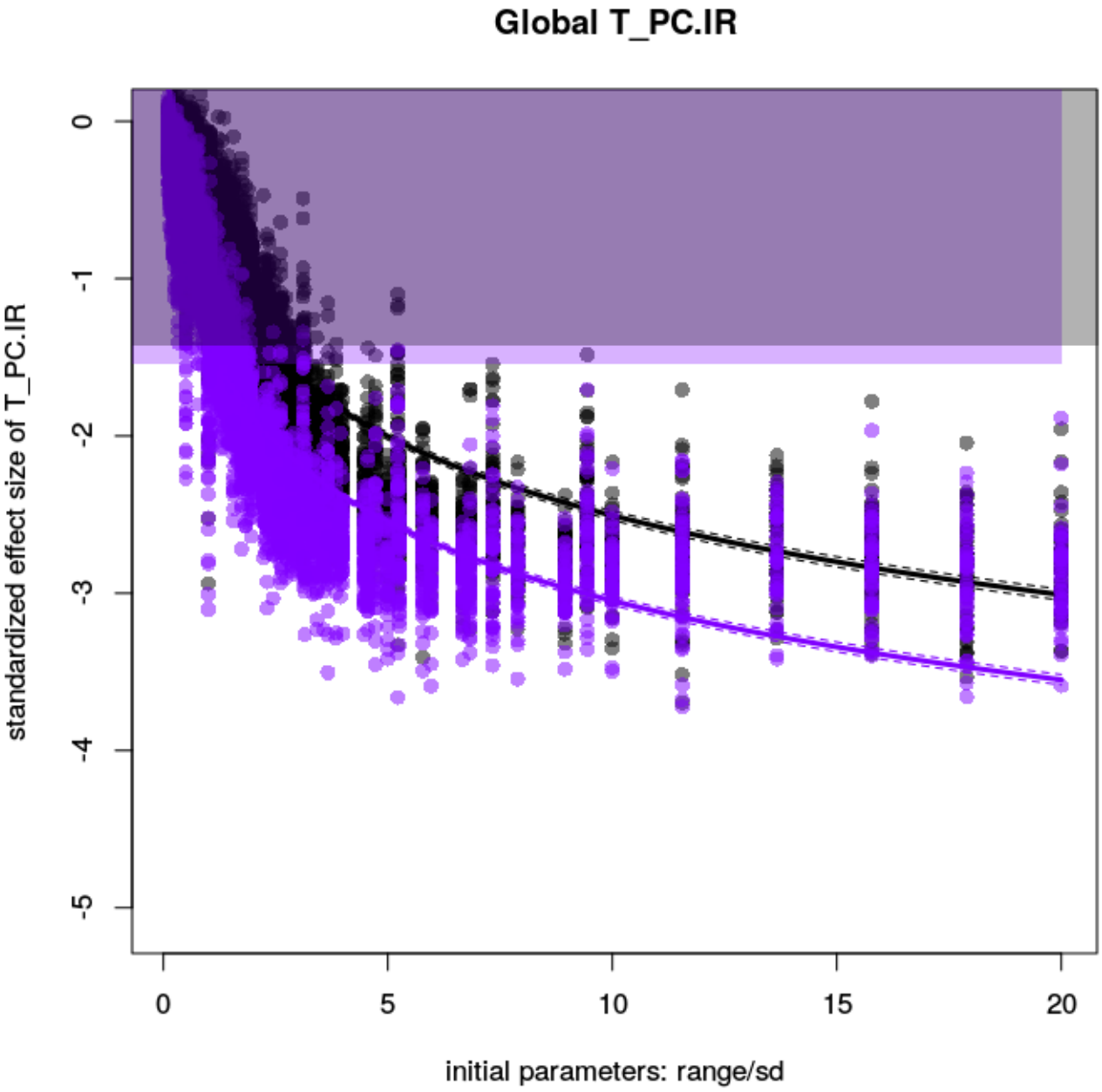


Figure 25: Power of $T_{PC/PR}$ to detect external filtering: Standardised Effect Size (SES) of $T_{PC/PR}$ as a function of the strength of external filtering. Trait 'a' normally distributed is shown in black, trait 'b' uniformally distributed in purple. Dots represent global SES values. The exponential linear model is represented with 0.05 confidence interval on both sides. Colored area represents the mean confidence interval of SES values ($alpha = 0.95$)

5 Combining internal and external filtering

5.1 Randomization outline

We use the same two parameters for both internal and external filtering analyses. The internal filtering is computed in the section dedicated to internal filter. To add the effect of external filtering, we add a value for each individual belonging to a given community, these value differing between communities.

```
# Parameter for the distance between species' mean trait values
param_DIST_between_MEAN.init <- round(sort(seq(10, 200, length.out = nb_param_val)), 2)
param_DIST_between_MEAN <- rep(param_DIST_between_MEAN.init, N_repet_Param)
mean_sd.param <- rep(10, nperm)

# Parameter for the variance in species' mean trait values
param_SD <- seq(10, 100,length.out = nb_param_val)
sd_mean.param <- sample (rep(param_SD, N_repet_Param),
       size = length(rep(param_SD, N_repet_Param)), replace = F)
sd_sd.param <- rep(10, nperm)

nperm <- N_repet_Param*length(param_SD)
```

```
#Start simulation
mean.sp_stock4 <- list()
sd.sp_stock4 <- list()
mean.com_stock4 <- list()
sd.com_stock4 <- list()
res.simu4 <- list()
res.simu4.pval <- list()
res.simu.traits4 <- list()

for(n in 1:nperm){#for each permutation

ex.sp4 <- c()
ex.com4 <- matrix(0, nrow = Ncom, ncol = Nsp)
for(i in 1: 10){
 ex.com.interm <- table(sample(sp, size = Nind/Ncom, prob = rlnorm(Nsp, 0, sdlog),
                               replace = T))
 ex.com4[i, sp%in% names(ex.com.interm)] <- ex.com.interm
 ex.sp4 <- c(ex.sp4, rep(sp, times = ex.com4[i,]))
}

ex.indplot4 <- sort(as.factor(rep(com, Nind/Ncom)))


#Defining trait mean and sd for each species
mean_mean.param.interm <- seq(max.value_traits - param_DIST_between_MEAN[n],
        max.value_traits, length.out =
        length(unique(param_DIST_between_MEAN)))

mean_mean.param <- rep(round(sort(mean_mean.param.interm), 2),
        N_repet_Param)
```

```
mean.sp <- rnorm(length(unique(sp)), mean = mean_mean.param, sd = mean_sd.param[n])
sd.sp <- rnorm(length(unique(sp)), mean = sd_mean.param[n], sd = sd_sd.param[n])

ex.traits4 <- array(NA, dim = c(Nind, 2))
colnames(ex.traits4) <- paste("trait", c("a", "b"), sep = " ")

for(s in unique(ex.sp4)){
 #trait a : normal distribution
 ex.traits4[ex.sp4 == s, 1] <- rnorm(500, rep(mean.sp[unique(ex.sp4) == s], 500),
         rep(sd.sp[unique(ex.sp4) == s], 500))[1:sum(ex.sp4 == s)]

 #trait b : uniform distribution
 ex.traits4[ex.sp4 == s, 2] <- runif(500, min = rep(mean.sp[unique(ex.sp4) == s], 500) -
         rep(sd.sp[unique(ex.sp4) == s], 500),
         max = rep(mean.sp[unique(ex.sp4) == s], 500) +
         rep(sd.sp[unique(ex.sp4) == s], 500))[1:sum(ex.sp4 == s)]
}

mean.com.add <- rnorm(length(unique(com)), mean = mean_mean.param, sd = mean_sd.param[n])
sd.com.add <- rnorm(length(unique(com)), mean = sd_mean.param[n], sd = sd_sd.param[n])

for(c in unique(ex.indplot4)){
 #trait a : normal distribution
 ex.traits4[ex.indplot4 == c, 1] <-
 ex.traits4[ex.indplot4 == c, 1] +
 rnorm(500, rep(mean.com.add[unique(ex.indplot4) == c], 500),
 rep(sd.com.add [unique(ex.indplot4) == c], 500))[1:sum(ex.indplot4 == c)]

 #trait b : uniform distribution
 ex.traits4[ex.indplot4 == c, 2] <-
 ex.traits4[ex.indplot4 == c, 2] +
 runif(500, min = rep(mean.com.add[unique(ex.indplot4) == c], 500) -
 rep(sd.com.add[unique(ex.indplot4) == c], 500),
 max = rep(mean.com.add [unique(ex.indplot4) == c], 500) +
 rep(sd.com.add [unique(ex.indplot4) == c], 500))[1:sum(ex.indplot4 == c)]
}

#stock results
mean.sp_stock4[[n]] <- mean.sp
sd.sp_stock4[[n]] <- sd.sp

mean.com_stock4[[n]] <- mean.com.add
sd.com_stock4[[n]] <- sd.com.add

res.simu.traits4[[n]] <- ex.traits4
res.simu4[[n]] <- Tstats(ex.traits4, ex.indplot4, ex.sp4)
res.simu4.pval[[n]] <- sum_Tstats(res.simu4[[n]], type = "p.value")
print(paste("---", round(n/nperm, 2) * 100, "%", sep = " "))
}#End of simulations
```

5.2 Results

```
par(mfrow=c(2, 2))
plotDistri(ex.traits4, rep("all_sp", times = dim(ex.traits1)[1]), ex.indplot4,
           plot.ask = F, multipanel = F, leg =c(T, F), main = c("a", "b"))
plotDistri(ex.traits4, rep("region", times = dim(ex.traits1)[1]), ex.sp4,
           plot.ask = F, multipanel = F, leg =c(T, F), main = c("c", "d"),
           ylim = c(0, 0.01))
par(mfrow=c(1, 1))
```

```
par(mfrow=c(2, 1))
plot(res.simu4[[1]], main = "a")
plot(res.simu4[[nperm]], main = "b")
par(mfrow=c(1, 1))
```

```
meanSES.4loc.norm_Tipic <- unlist(lapply(res.simu4, function(x)
             ses.listofindex(as.listofindex(x))
             $index_1_1$ses[, 1]))
meanSES.4loc.uni_Tipic <- unlist(lapply(res.simu4, function(x)
             ses.listofindex(as.listofindex(x))
             $index_1_1$ses[, 2]))
SES.inf.MEAN.norm_Tipic <- unlist(lapply(res.simu4, function(x)
             ses.listofindex(as.listofindex(x))
             $index_1_1$ses.inf[, 1]))
SES.inf.MEAN.uni_Tipic <- unlist(lapply(res.simu4, function(x)
             ses.listofindex(as.listofindex(x))
             $index_1_1$ses.inf[, 2]))

meanSES.4loc.norm_Ticir <- unlist(lapply(res.simu4, function(x)
             ses.listofindex(as.listofindex(x))
             $index_1_2$ses[, 1]))
meanSES.4loc.uni_Ticir <- unlist(lapply(res.simu4, function(x)
             ses.listofindex(as.listofindex(x))
             $index_1_2$ses[, 2]))
SES.inf.MEAN.norm_Ticir <- unlist(lapply(res.simu4, function(x)
             ses.listofindex(as.listofindex(x))
             $index_1_2$ses.inf[, 1]))
SES.inf.MEAN.uni_Ticir <- unlist(lapply(res.simu4, function(x)
             ses.listofindex(as.listofindex(x))
             $index_1_2$ses.inf[, 2]))

meanSES.4loc.norm_Tpcpr <- unlist(lapply(res.simu4, function(x)
             ses.listofindex(as.listofindex(x))
             $index_1_3$ses[, 1]))
meanSES.4loc.uni_Tpcpr <- unlist(lapply(res.simu4, function(x)
             ses.listofindex(as.listofindex(x))
             $index_1_3$ses[, 2]))
SES.inf.MEAN.norm_Tpcpr <- unlist(lapply(res.simu4, function(x)
             ses.listofindex(as.listofindex(x))
```

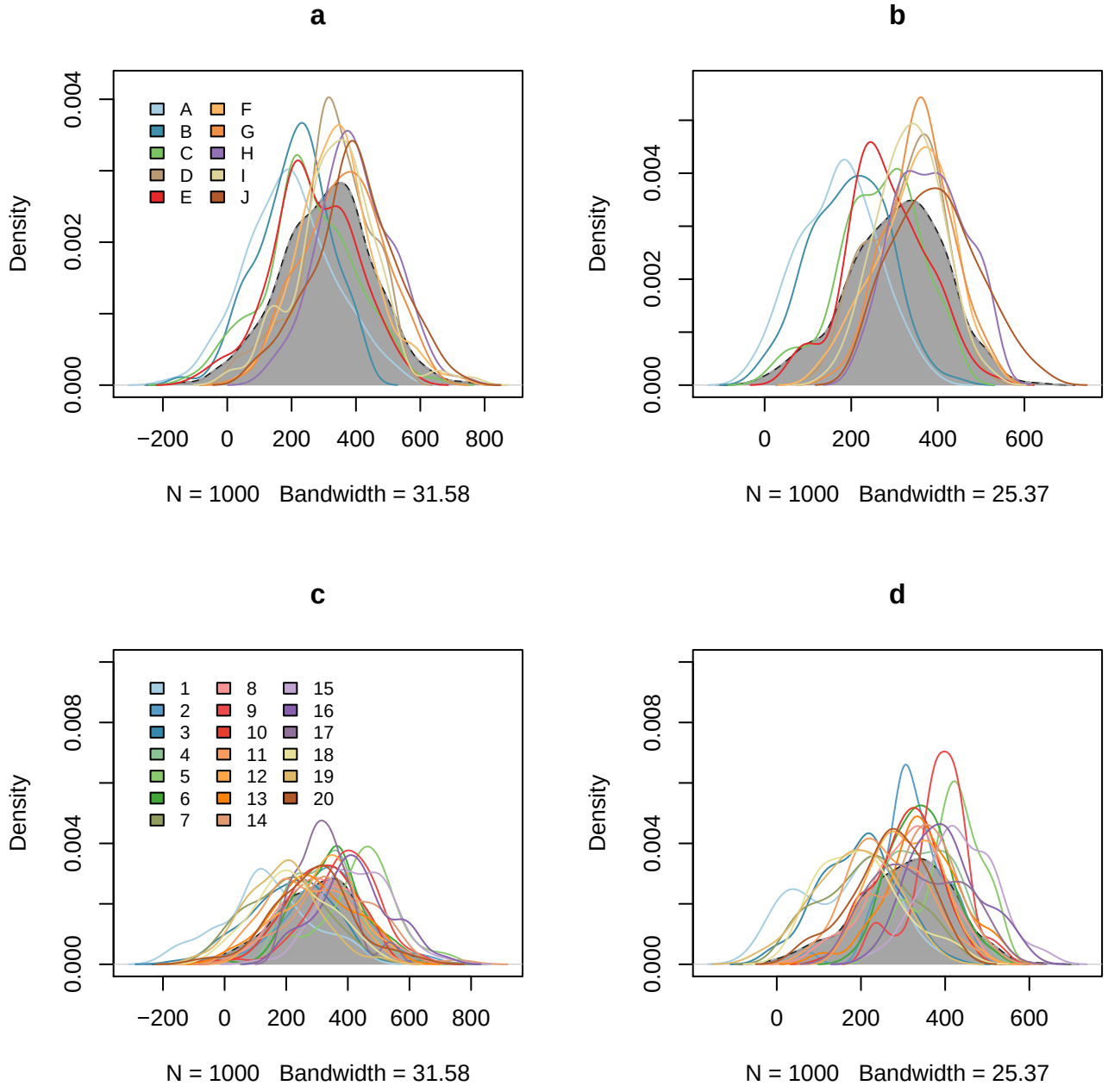


Figure 26: Distribution of trait values for one randomization with both internal and external filtering: (a) Community-level trait distributions for the trait a (normal distribution); (b) Community-level trait distributions for the trait b (uniform distribution). In panels a and b, each color represents one community (site). (c) Species' trait distributions for the trait a; (d) Species' trait distributions for the trait b. In panels a and b, each color represents one species.

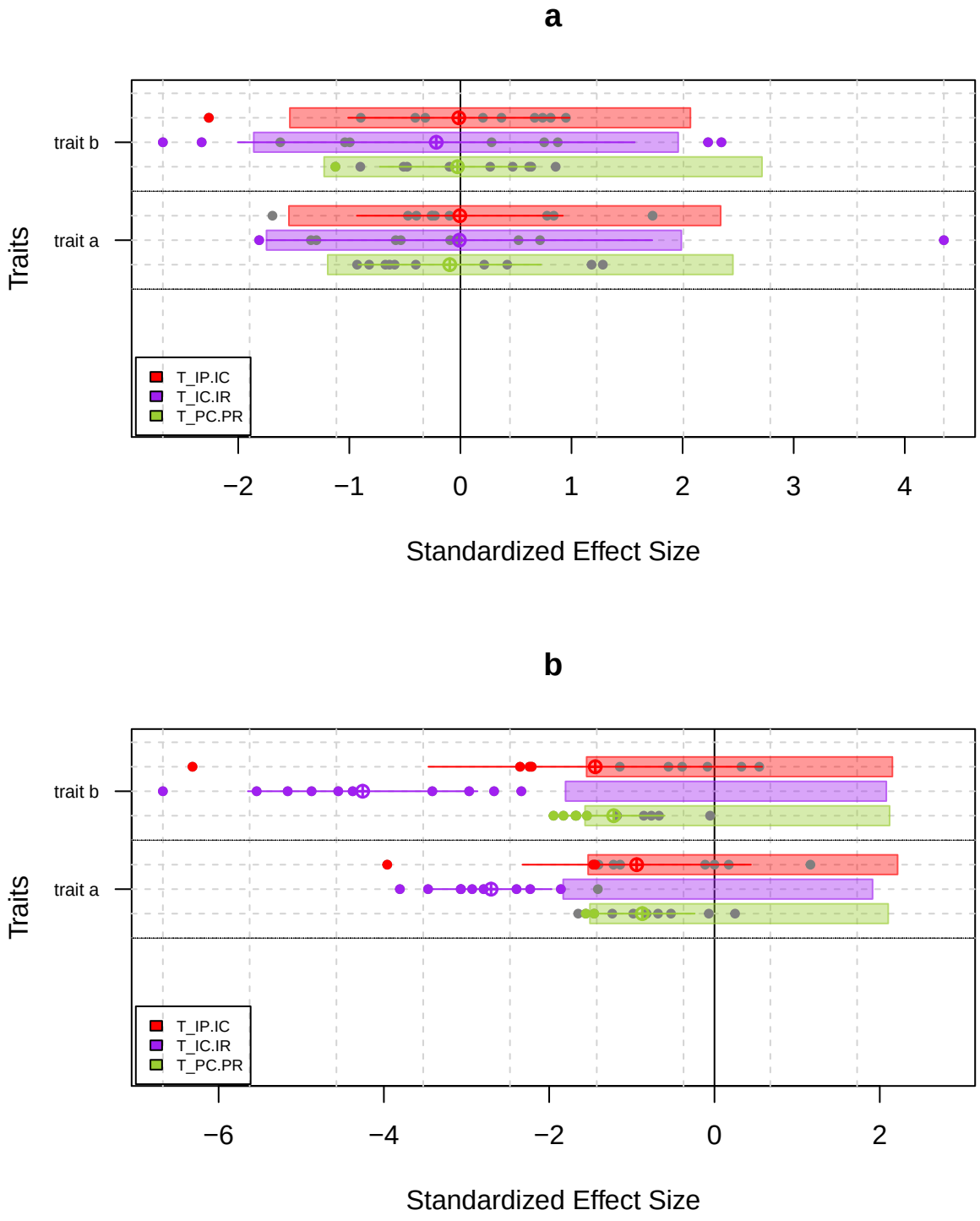


Figure 27: Results of T-statistics with both internal and external filtering: $T_{IP/IC}$ in red, $T_{IC/IR}$ in purple and $T_{PC/PR}$ in green. (a) Lower strength of filtering (b) higher strength of filtering.

```
            $index_1_3$ses.inf[, 1]))
SES.inf.MEAN.uni_Tpcpr <- unlist(lapply(res.simu4, function(x)
            ses.listofindex(as.listofindex(x))
            $index_1_3$ses.inf[, 2]))
```

```
meanSES.4glob.norm_Tipic <- unlist(lapply(res.simu4, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_1$ses[,1], na.rm = T)))
meanSES.4glob.uni_Tipic <- unlist(lapply(res.simu4, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_1$ses[,2], na.rm = T)))
meanSES.INF_glob.norm_Tipic <- unlist(lapply(res.simu4, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_1$ses.inf[, 1], na.rm = T)))
meanSES.INF_glob.uni_Tipic <- unlist(lapply(res.simu4, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_1$ses.inf[, 2], na.rm = T)))

meanSES.4glob.norm_Ticir <- unlist(lapply(res.simu4, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_2$ses[,1], na.rm = T)))
meanSES.4glob.uni_Ticir <- unlist(lapply(res.simu4, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_2$ses[,2], na.rm = T)))
meanSES.INF_glob.norm_Ticir <- unlist(lapply(res.simu4, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_2$ses.inf[, 1], na.rm = T)))
meanSES.INF_glob.uni_Ticir <- unlist(lapply(res.simu4, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_2$ses.inf[, 2], na.rm = T)))

meanSES.4glob.norm_Tpcpr <- unlist(lapply(res.simu4, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_3$ses[,1], na.rm = T)))
meanSES.4glob.uni_Tpcpr <- unlist(lapply(res.simu4, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_3$ses[,2], na.rm = T)))
meanSES.INF_glob.norm_Tpcpr <- unlist(lapply(res.simu4, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_3$ses.inf[, 1], na.rm = T)))
meanSES.INF_glob.uni_Tpcpr <- unlist(lapply(res.simu4, function(x)
          mean(ses.listofindex(as.listofindex(x))
            $index_1_3$ses.inf[, 2], na.rm = T)))
```

```
# trait a
lm_glob_norm_Tipic <- lm(meanSES.4glob.norm_Tipic ~ log(init_param))
lm_glob_norm_Tipic.conf <- confint(lm_glob_norm_Tipic , level = 0.90)
lm_glob_norm_Ticir <- lm(meanSES.4glob.norm_Ticir ~ log(init_param))
lm_glob_norm_Ticir.conf <- confint(lm_glob_norm_Ticir , level = 0.90)
lm_glob_norm_Tpcpr <- lm(meanSES.4glob.norm_Tpcpr ~ log(init_param))
```

```
lm_glob_norm_Tpcpr.conf <- confint(lm_glob_norm_Tpcpr , level = 0.90)
yy.glob_norm_Tipic <- mean(meanSES.INF_glob.norm_Tipic, na.rm = T)
yy.glob_norm_Ticir <- mean(meanSES.INF_glob.norm_Ticir, na.rm = T)
yy.glob_norm_Tpcpr <- mean(meanSES.INF_glob.norm_Tpcpr, na.rm = T)
param_B_glob_norm_Tipic <- exp( (yy.glob_norm_Tipic - lm_glob_norm_Tipic.conf [1, 2]) /
                                lm_glob_norm_Tipic.conf [2, 2] )
param_B_glob_norm_Ticir <- exp( (yy.glob_norm_Ticir - lm_glob_norm_Ticir.conf [1, 2]) /
                                lm_glob_norm_Ticir.conf [2, 2] )
param_B_glob_norm_Tpcpr <- exp( (yy.glob_norm_Tpcpr - lm_glob_norm_Tpcpr.conf [1, 2]) /
                                lm_glob_norm_Tpcpr.conf [2, 2] )

lm_loc_norm_Tipic <- lm(meanSES.4loc.norm_Tipic ~ log(init_param.loc))
lm_loc_norm_Tipic.conf <- confint(lm_loc_norm_Tipic, level = 0.90)
lm_loc_norm_Ticir <- lm(meanSES.4loc.norm_Ticir ~ log(init_param.loc))
lm_loc_norm_Ticir.conf <- confint(lm_loc_norm_Ticir, level = 0.90)
lm_loc_norm_Tpcpr <- lm(meanSES.4loc.norm_Tpcpr ~ log(init_param.loc))
lm_loc_norm_Tpcpr.conf <- confint(lm_loc_norm_Tpcpr, level = 0.90)
yy.loc_norm_Tipic <- mean(SES.inf.MEAN.norm_Tipic, na.rm = T)
yy.loc_norm_Ticir <- mean(SES.inf.MEAN.norm_Ticir, na.rm = T)
yy.loc_norm_Tpcpr <- mean(SES.inf.MEAN.norm_Tpcpr, na.rm = T)
param_B_loc_norm_Tipic <- exp( (yy.loc_norm_Tipic - lm_loc_norm_Tipic.conf [1, 2]) /
                                lm_loc_norm_Tipic.conf [2, 2] )
param_B_loc_norm_Ticir <- exp( (yy.loc_norm_Ticir - lm_loc_norm_Ticir.conf [1, 2]) /
                                lm_loc_norm_Ticir.conf [2, 2] )
param_B_loc_norm_Tpcpr <- exp( (yy.loc_norm_Tpcpr - lm_loc_norm_Tpcpr.conf [1, 2]) /
                                lm_loc_norm_Tpcpr.conf [2, 2] )

# trait b
lm_glob_uni_Tipic <- lm(meanSES.4glob.uni_Tipic ~ log(init_param))
lm_glob_uni_Tipic.conf <- confint(lm_glob_uni_Tipic , level = 0.90)
lm_glob_uni_Ticir <- lm(meanSES.4glob.uni_Ticir ~ log(init_param))
lm_glob_uni_Ticir.conf <- confint(lm_glob_uni_Ticir , level = 0.90)
lm_glob_uni_Tpcpr <- lm(meanSES.4glob.uni_Tpcpr ~ log(init_param))
lm_glob_uni_Tpcpr.conf <- confint(lm_glob_uni_Tpcpr , level = 0.90)
yy.glob_uni_Tipic <- mean(meanSES.INF_glob.uni_Tipic, na.rm = T)
yy.glob_uni_Ticir <- mean(meanSES.INF_glob.uni_Ticir, na.rm = T)
yy.glob_uni_Tpcpr <- mean(meanSES.INF_glob.uni_Tpcpr, na.rm = T)
param_B_glob_uni_Tipic <- exp( (yy.glob_uni_Tipic - lm_glob_uni_Tipic.conf [1, 2]) /
                                lm_glob_uni_Tipic.conf [2, 2] )
param_B_glob_uni_Ticir <- exp( (yy.glob_uni_Ticir - lm_glob_uni_Ticir.conf [1, 2]) /
                                lm_glob_uni_Ticir.conf [2, 2] )
param_B_glob_uni_Tpcpr <- exp( (yy.glob_uni_Tpcpr - lm_glob_uni_Tpcpr.conf [1, 2]) /
                                lm_glob_uni_Tpcpr.conf [2, 2] )

lm_loc_uni_Tipic <- lm(meanSES.4loc.uni_Tipic ~ log(init_param.loc))
lm_loc_uni_Tipic.conf <- confint(lm_loc_uni_Tipic, level = 0.90)
lm_loc_uni_Ticir <- lm(meanSES.4loc.uni_Ticir ~ log(init_param.loc))
lm_loc_uni_Ticir.conf <- confint(lm_loc_uni_Ticir, level = 0.90)
lm_loc_uni_Tpcpr <- lm(meanSES.4loc.uni_Tpcpr ~ log(init_param.loc))
lm_loc_uni_Tpcpr.conf <- confint(lm_loc_uni_Tpcpr, level = 0.90)
yy.loc_uni_Tipic <- mean(SES.inf.MEAN.uni_Tipic, na.rm = T)
```

```
yy.loc_uni_Ticir <- mean(SES.inf.MEAN.uni_Ticir, na.rm = T)
yy.loc_uni_Tpcpr <- mean(SES.inf.MEAN.uni_Tpcpr, na.rm = T)
param_B_loc_uni_Tipic <- exp( (yy.loc_uni_Tipic - lm_loc_uni_Tipic.conf [1, 2]) /
                                lm_loc_uni_Tipic.conf [2, 2] )
param_B_loc_uni_Ticir <- exp( (yy.loc_uni_Ticir - lm_loc_uni_Ticir.conf [1, 2]) /
                                lm_loc_uni_Ticir.conf [2, 2] )
param_B_loc_uni_Tpcpr <- exp( (yy.loc_uni_Tpcpr - lm_loc_uni_Tpcpr.conf [1, 2]) /
                                lm_loc_uni_Tpcpr.conf [2, 2] )
```

For the analysis with both internal and external filtering we only present the type II error using the initial parameter ratio allowing a beta-error < 0.05.

Table 6: Initial parameter ratios allowing beta-error < 0.05 for T-statistics under both external and internal filtering

Traits	Indices	Average between communities?	Initial parameter ratio
Trait a (normal)	$T_{IP.IC}$	no (local)	5.826
		yes (global)	5.865
	$T_{IC.IR}$	no (local)	2.399
		yes (global)	2.416
	$T_{PC.PR}$	no (local)	296.695
		yes (global)	316.073
Trait b (uniform)	$T_{IP.IC}$	no (local)	1.107
		yes (global)	1.093
	$T_{IC.IR}$	no (local)	0.584
		yes (global)	0.575
	$T_{PC.PR}$	no (local)	84.112
		yes (global)	91.881

6 Test of variance decomposition functions

In addition to the `cati` vignette which tests this function on the Darwin's finches data, we test the behavior of the function `partvar` and `decompCTRE` by building two toy models associated to each function.

6.1 Behavior of the function `partvar`

First to test the behavior of the `partvar` function we assemble the same data as in the test of the T-statistics. 1000 individuals belonging to 20 species occuring in 10 communities. But here we define the trait value for one individual as the addition of three values, each one drawn in a normal distribution of mean 0 and standard error 1: (i) a value depending on the species the individual belong to, (ii) a value depending on the community the individual belong to and (iii) an independent value measuring the individual variance.

```
nperm <- npermut

sdlog
table(com)
table(sp)

res.partvar_toymodel <- list()
for(n in 1:nperm){#for each permutation

 # Draw communities using lognormal distribution of abundances
 ex.sp5 <- c()
 ex.com5 <- matrix(0, nrow = Ncom, ncol = Nsp)
 for(i in 1: Ncom){
 ex.com.interm <- table(sample(sp, size = Nind/Ncom, prob = rlnorm(Nsp, 0, sdlog),
                               replace = T))
 ex.com5[i, sp%in% names(ex.com.interm)] <- ex.com.interm
 ex.sp5 <- c(ex.sp5, rep(sp, times = ex.com5[i,]))
 }


 x1 <- c()
 ex.sp5 <- as.factor(ex.sp5)
 for(s in 1:Nsp){
 x1[ex.sp5 == levels(ex.sp5)[s]] <- rnorm(1)
 }

 ex.indplot5 <- sort(as.factor(rep(com, Nind/Ncom)))
 x2 <- c()
 x3 <- c()
 for(c in 1:Ncom){
 x2[ex.indplot5 == levels(ex.indplot5 )[c]] <- rnorm(1)
 }
 X <- rnorm(Nind) + x1 + x2

 res.partvar_toymodel[[n]] <- partvar(X, factors = cbind(sites = ex.indplot5,
              species = ex.sp5))
```

```
}
```

Now we can plot the result as the density of the three components of variation in individual trait distributions. We expect a mean of approximately 0.33 for each component in view of our model.

```
res.simu.partvar_toymodel <- lapply(res.partvar_toymodel, function (x) x[, 1])
res.simu.partvar_toymodel <- t(matrix(unlist(res.simu.partvar_toymodel), nrow = 3))
colnames(res.simu.partvar_toymodel) <- c("sites", "species", "within")
col.funk <- funky.col(3)

plot(density(res.simu.partvar_toymodel[, 3]), col = col.funk[3],
  pch = 16, xlim = c(0, 0.9), lwd = 2)
lines(density(res.simu.partvar_toymodel[, 1]), col = col.funk[1], lwd = 2)
lines(density(res.simu.partvar_toymodel[, 2]), col = col.funk[2], lwd = 2)
abline(v = 0.33)
points(apply(res.simu.partvar_toymodel, 2, mean),
   c(max(density(res.simu.partvar_toymodel[, 1])$y)/2,
   max(density(res.simu.partvar_toymodel[, 2])$y)/2,
   max(density(res.simu.partvar_toymodel[, 3])$y)/2),
   col = col.funk)
segments(apply(res.simu.partvar_toymodel, 2, mean) - apply(res.simu.partvar_toymodel, 2, sd),
   c(max(density(res.simu.partvar_toymodel[, 1])$y)/2,
   max(density(res.simu.partvar_toymodel[, 2])$y)/2,
   max(density(res.simu.partvar_toymodel[, 3])$y)/2),
   apply(res.simu.partvar_toymodel, 2, mean) + apply(res.simu.partvar_toymodel, 2, sd),
   c(max(density(res.simu.partvar_toymodel[, 1])$y)/2,
   max(density(res.simu.partvar_toymodel[, 2])$y)/2,
   max(density(res.simu.partvar_toymodel[, 3])$y)/2),
   col = col.funk)
legend(0.6, 4, legend = c("sites", "species", "within"), fill = col.funk)
text(0.33, 0.5, pos = 4, "0.33")
```

The mean contribution of each component is close to 0.33. Consequently the function `partvar` does not show apparent dysfunction.

6.2 Behavior of the function `decompCTRE`

To test the function `decompCTRE`, we adopt the same model but we add a new step to control the strength of the turnover in defining the community weighted mean. We define several values of turnover strength by deleting individuals from the data set if their species-components trait and community-component trait are too different. To be more precise, when `turnover_strength` $= 1$, we delete all individuals whose the square of the species value minus the community value is superior to one.

```
sdlog
turnover_strength <- rep(c(0.1, 0.25, 0.5, 1, 2, 5, 7.5, 10, 12.5, 15),
                         N_repet_Param)
nperm <- length(turnover_strength)
res.decompCTRE_toymodel <- list()
for(n in 1:nperm){#for each permutation

 # Draw communities using lognormal distribution of abundances
```

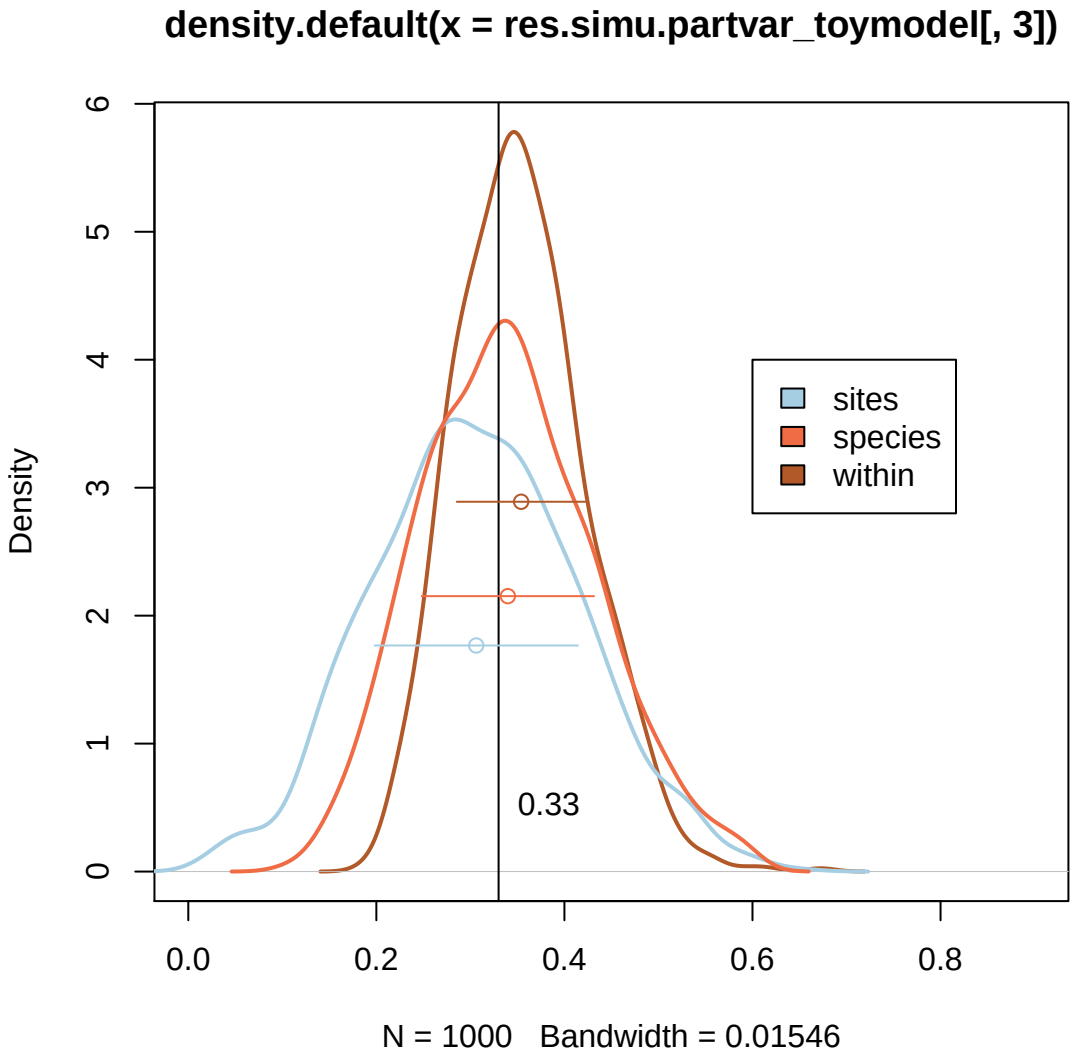


Figure 28: Variance partitioning accross nested scales: each colored line represents the density of permutation results (sites in blue, species in orange and the remaining variance in brown). Dots represent the mean and segments the standard deviation. Vertical line figures the value of 0.33.

```
ex.sp5 <- c()
ex.com5 <- matrix(0, nrow = Ncom, ncol = Nsp)
for(i in 1: Ncom){
ex.com.interm <- table(sample(sp, size = Nind/Ncom, prob = rlnorm(Nsp, 0, sdlog),
                              replace = T))
ex.com5[i, sp%in% names(ex.com.interm)] <- ex.com.interm
ex.sp5 <- c(ex.sp5, rep(sp, times = ex.com5[i,]))
}


x1 <- c()
ex.sp5 <- as.factor(ex.sp5)
for(s in 1:Nsp){
x1[ex.sp5 == levels(ex.sp5)[s]] <- rnorm(1)
}

ex.indplot5 <- sort(as.factor(rep(com, Nind/Ncom)))
x2 <- c()
for(c in 1:Ncom){
x2[ex.indplot5 == levels(ex.indplot5)[c]] <- rnorm(1)
}

X <- rnorm(Nind) + x1 + x2

#We delete the individuals whose species values and communities values are too different
#For this we use the argument turnover_strength
X2 <- X[(x1-x2)^2 < turnover_strength[n]]
ex.sp5.bis <- ex.sp5[(x1-x2)^2 < turnover_strength[n]]
ex.indplot5.bis <- ex.indplot5[(x1-x2)^2 < turnover_strength[n]]

res.decompCTRE_toymodel[[n]] <- decompCTRE(traits = cbind(X2, X2), sp = ex.sp5.bis,
           ind.plot = ex.indplot5.bis)
}#End of simulations
```

Now we can plot the relationship between the modeled strength of the turnover and the resulting contribution of turnover from the function `decompCTRE` (Fig. 29).

```
par(mfrow=c(1, 2))
plot(res.decompCTRE_toymodel[[1]]$X2)
plot(res.decompCTRE_toymodel[[length(res.decompCTRE_toymodel)]]$X2)
par(mfrow=c(1, 1))
```

```
res.sim.decompCTRE_toymodel <- unlist(lapply(res.decompCTRE_toymodel, function(x)
                                             x$X2$RelSumSq[1]))
plot(res.sim.decompCTRE_toymodel, -turnover_strength, pch = 16,
     col = rgb(0, 0, 0, 0.5),
     ylab = "Strength of the turnover (maximum strength = 0)",
     xlab = "Turnover relative contribution to community mean traits")
```

The correlation between the modeled strength of the turnover and the turnover relative contribution result is very high. Consequently the function `decompCTRE` does not show apparent dysfunction (Fig. 30).

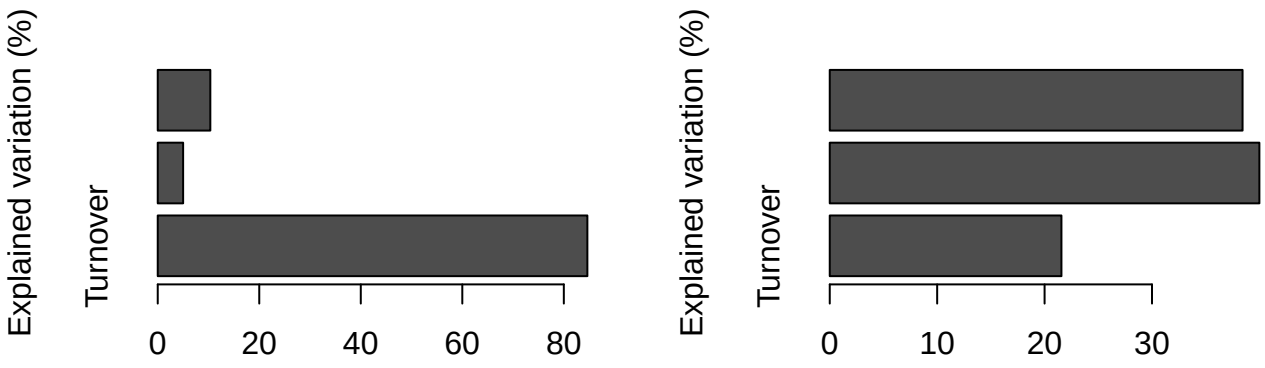


Figure 29: Decomposition of the variation in community trait composition for two contrasted cases: on the left almost all the variation is due to turnover whereas on the rigth case intraspecific variation explain most of the variation.

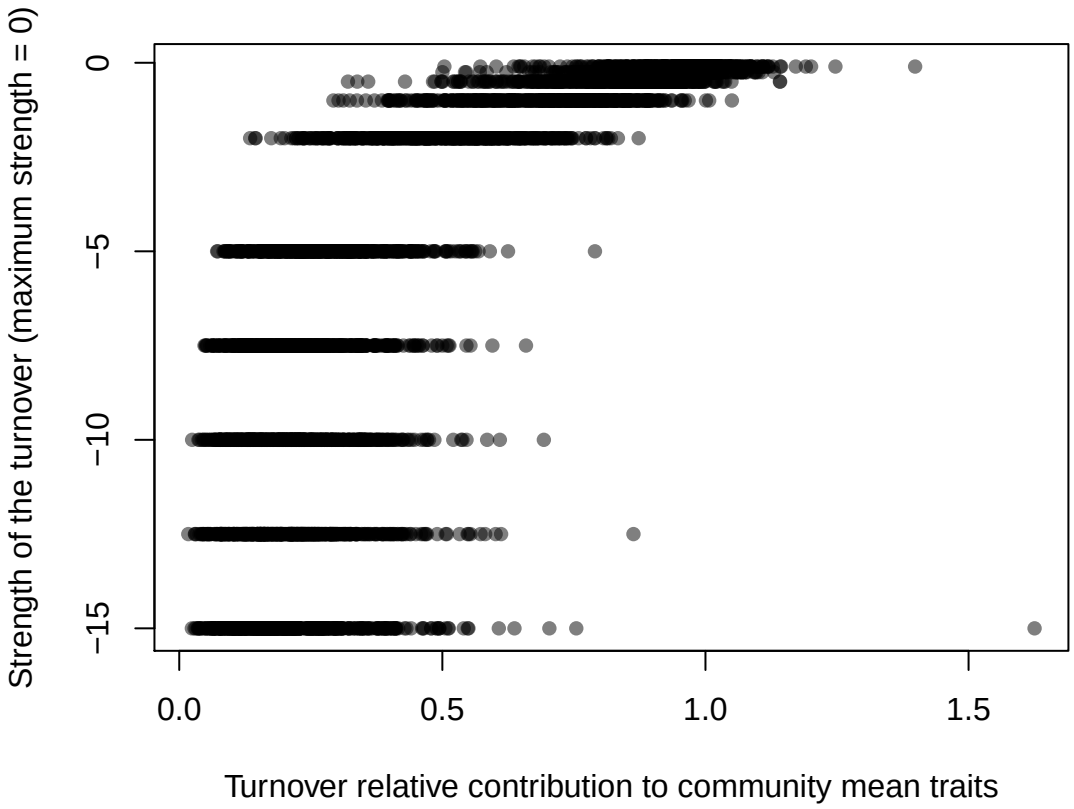


Figure 30: Modeled against obtained turnover contribution in community trait variance

7 Summary

7.1 Type I error

Table 2 summarizes the alpha error for the T-statistics. The three global [12] T-statistics are very robust regardless of the distribution of the trait (alpha error ranging from 0.001 to 0.001). Local [13] T-statistics present lower but nevertheless satifactory alpha-error with a maximum of 0.0198 for $T_{PC/PR}$ on a trait normally distributed. We therefore recommand to use $T_{IC/IR}$ to detect external filtering and more generally to prefer global T-statistics which are more robust and more powerful.

7.2 Type II error

Table 7: Summary of initial parameter ratios allowing beta-error < 0.05 for global T-statistics

Traits	Indices	Initial parameter ratio with internal filtering	Initial parameter ratio with external filtering	Initial parameter ratio with internal and external filtering
Trait a (normal)	$T_{IP.IC}$	1.86		5.87
	$T_{IC.IR}$		1	2.42
	$T_{PC.PR}$		2.31	316.07
Traits	Indices	Initial parameter ratio with internal filtering	Initial parameter ratio with external filtering	Initial parameter ratio with internal and external filtering
Trait b (uniform)	$T_{IP.IC}$	0.88		1.09
	$T_{IC.IR}$		0.49	0.57
	$T_{PC.PR}$		1.27	91.88

Conclusion

In this document, we bring out a low type I error (alpha-error is inferior to 0.05 for all T-statistics; Table 2). We also studied the type II error in relation to the type of hypotheses specified in the models. The power to detect the external filter at the individual level ($T_{IC/IR}$) predominates over the power to detect it at the populational level ($T_{PC/PR}$) in all simulation cases. We therefore encourage cati's users to be very careful when they compare $T_{IC/IR}$ and $T_{PC/PR}$. Finally, we detected no bias in the functions partvar and decomCTRE. This appendix can be rerun with different parameters using the text document at https://github.com/adrientaudiere/cati/blob/Package-cati/Documentation/Appendix4/Appendix4.Rnw.

[12] one index for all the dataset
[13] one index for each community of the dataset

References

1. May RM, Mac Arthur RH (1972) Niche overlap as a function of environmental variability. Proceedings of the National Academy of Sciences 69: 1109-1113.

2. Violle C, Enquist BJ, McGill BJ, Jiang L, Albert CH, et al. (2012) The return of the variance: intraspecific variability in community ecology. Trends in Ecology and Evolution 27: 244-252.

List of Figures

List of Tables