Appendix: Test the T-statistics using simulations: robustness and lack of bias in the function

Adrien Taudiere*and Cyrille Violle

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

January 14, 2015

Abstract

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 show the lack of bias in the functions `Tstats`, `partvar`, `decompCTRE` and a test of robustness for the T-statistics. This report was created on 2015-01-13 13:20:29. Start directly by the summary section if you want condensate results of this document.

1.1 R requirements

First we need to install and load the package.

```r
# install.packages("cati", repos = "http://cran.us.r-project.org", dependencies = TRUE)
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 Windows. See below for more information. The speed computations of `cati` main functions are develop in the cati tutorial.

```r
sessionInfo()

## R version 3.1.2 (2014-10-31)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
##
## locale:
## [1] LC_COLLATE=French_France.1252  LC_CTYPE=French_France.1252
## [3] LC_MONETARY=French_France.1252 LC_NUMERIC=C
## [5] LC_TIME=French_France.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] xtable_1.7-4 cati_0.94    ape_3.2      ade4_1.6-2   nlme_3.1-119
## [6] knitr_1.8
##
## loaded via a namespace (and not attached):
##  [1] class_7.3-11       codetools_0.2-9    digest_0.6.8
##  [4] e1071_1.6-4        evaluate_0.5.5     formatR_1.0
##  [7] geometry_0.3-5     grid_3.1.2         hexbin_1.27.0
## [10] highr_0.4          hypervolume_1.1.1  igraph_0.7.1
## [13] lattice_0.20-29    latticeExtra_0.6-26 magic_1.5-6
## [16] MASS_7.3-35        mice_2.22          nnet_7.3-8
## [19] randomForest_4.6-10 raster_2.3-12     rasterVis_0.32
## [22] RColorBrewer_1.1-2 Rcpp_0.11.3        rgl_0.95.1201
## [25] rpart_4.1-8        sp_1.0-16          stringr_0.6.2
## [28] tools_3.1.2        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)

6

1.4   Simulations parameters

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

```r
#Number of permutations for the analysis with no filter
npermut <- 1000

#Number of permutations for each parameters values in the analysis with filter(s)
N_repet_Param <- 200

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

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

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

#Twenty species sp1, sp2, sp3, ... sp20
sp <- paste("sp", seq(1:20), sep = "")
Nsp <- length(sp)

#Total number of individuals
Nind <- 1000
```
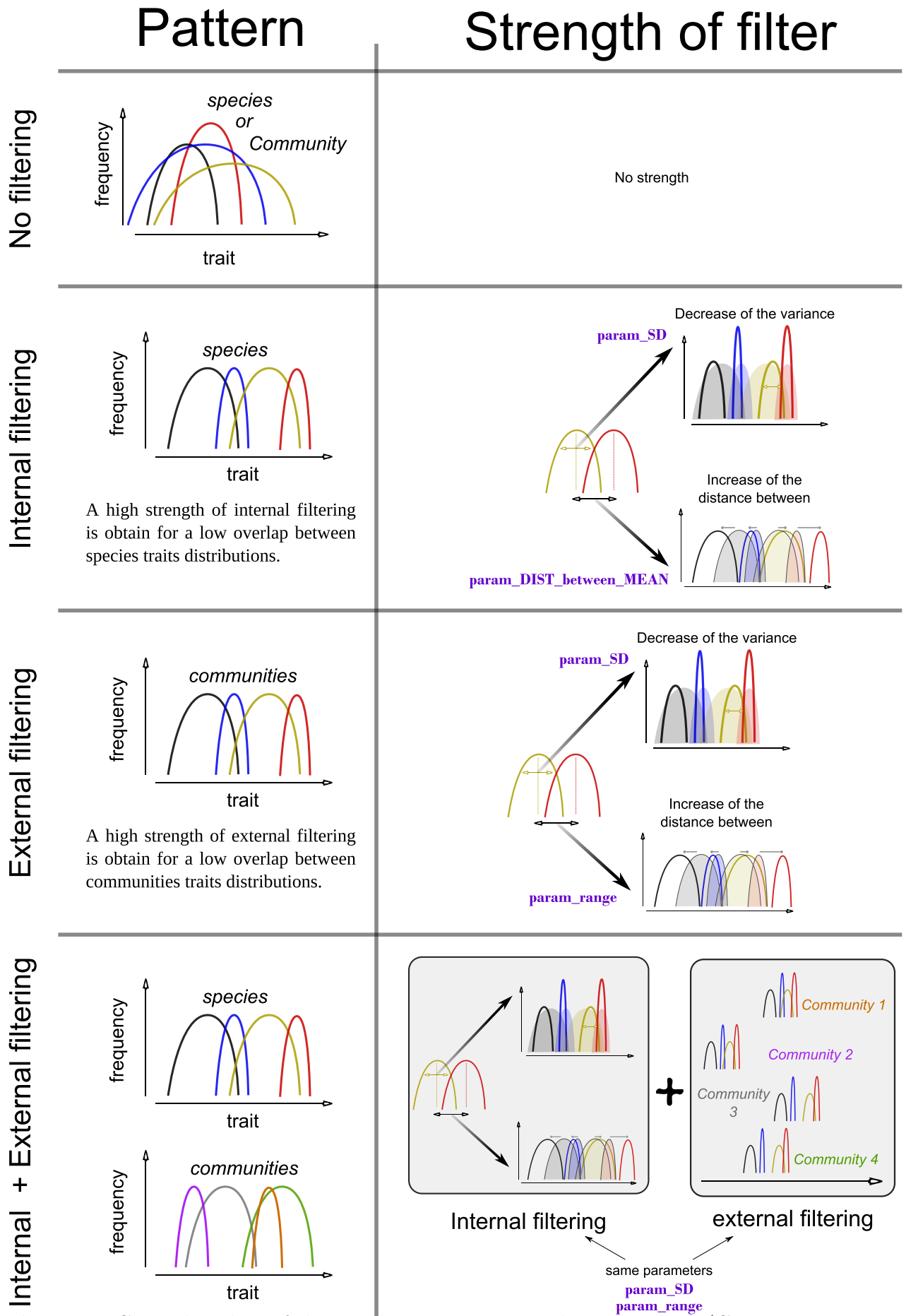
Figure 1: General outline of the simulation concerning the Tstatistics. (Continue on next page)

Figure 1 (continue): The **absence of filtering** is use 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 into species and communities. **Internal filter** includes all of the assembly processes internal to the community. The alternative hypothesis (thus the expected pattern) of habitat filtering is the non overlap between species traits distributions measured by the ratio $T_{IP/IC}$. We modulate the strength of filter in the simulation using the variance of species traits distributions and the distance between species traits means. **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 a non overlap between communities traits distributions measured by the ratio $T_{IC/IR}$ and its populationnal counterparts. We modulate the strength of external filter in the simulation using the variance of communities traits distributions and the range of communities traits means (the "extent" of the environnemental gradient). Finally, we add external filter to internal filter by adding to each individual values of the internal filter simulation a number depending of the community distribution.

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 are defined following a lognormal distribution of mean 0 and standard deviation `sdlog` of 1.5. Figure 1 depict the general outline of the next fourth sections dealing with the statistical test of T-statistics from Violle *et al.* (2012).

2 No filter: calculation of error

2.1 Randomization outline

To calculate the **alpha error** (*i.e.* the probability of rejected the null hypothesis H0 while H0 is true), we draw 1000 random traits matrix irrespective of the species and community attributes of each individuals.

```r
#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 abondances
 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 <- as.factor(rep(com, 100))


 #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 simulation
```

2.2 Results for simulations with no filtering

Let's see the result for one typical randomization. We can plot the distribution of traits 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).

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

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

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

Here we call local p-values the p-values corresponding to one indice for one trait in one community [1].
Now lets plot the results as the ordered p.value for each T-statistics and each traits (Fig. 4)). In that
case we have 20 p-values [2] by T-statistics by traits for each permutation. Using the defined parameters,
we plot 2×10^4 points by T-statistic by trait (20 p.value multiply by 1000 permutation).
   The alpha error on the figure are the proportion of p.value which are inferior to 0.025 (because we
are doing a bilateral test).

```
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]In contrast with global statistics which aggregate the communities values metrics.
[2]10 p-value, corresponding to 10 communities, multiply by 2 because we are doing a bilateral test
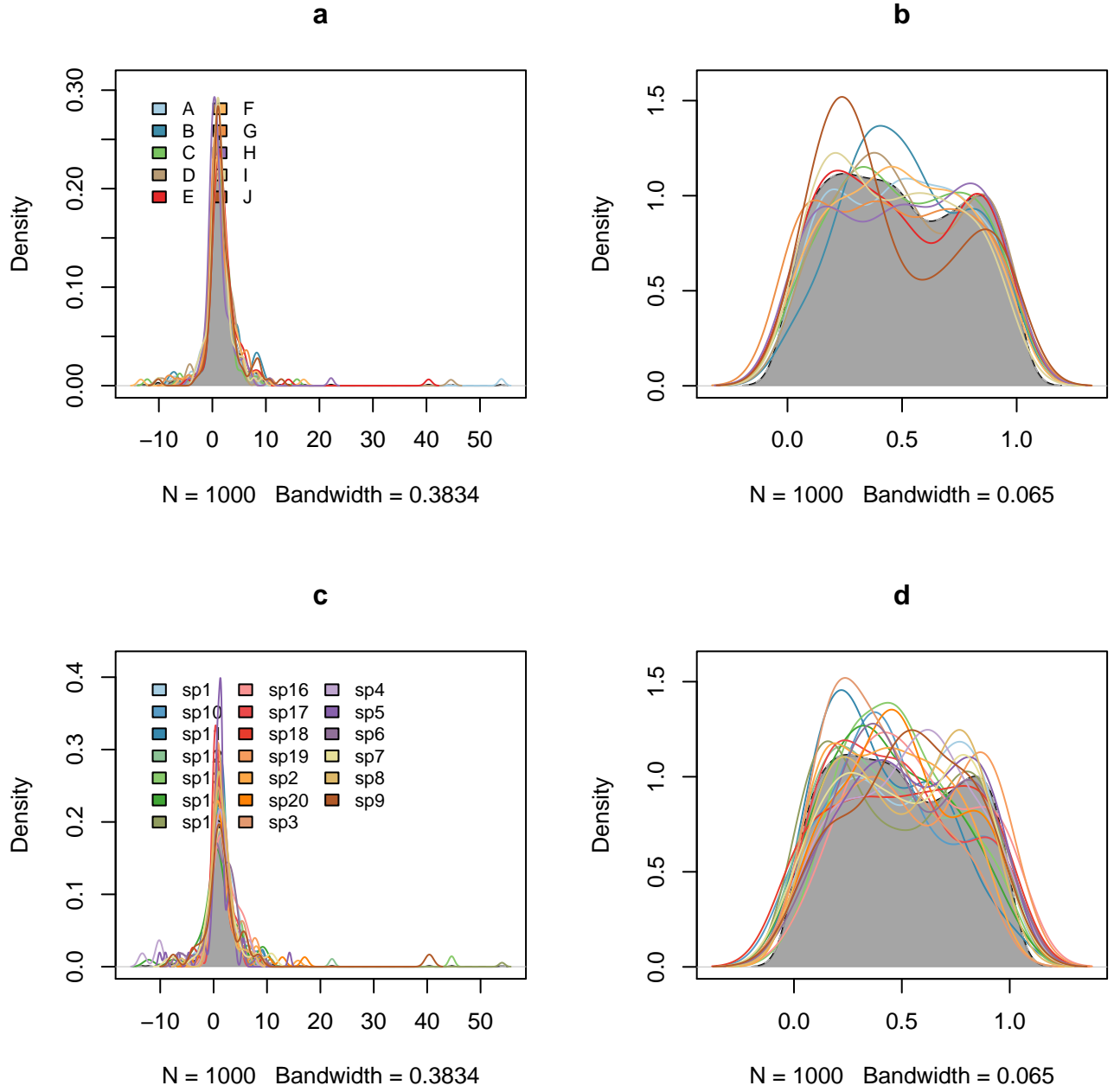
Figure 2: Distribution of traits for one randomization without filter: (a) Communities trait distributions for the trait a (normal distribution); (b) Communities trait distributions for the trait b (uniform distribution); (c) Species trait distributions for the trait a; (d) Species trait distributions for the trait b
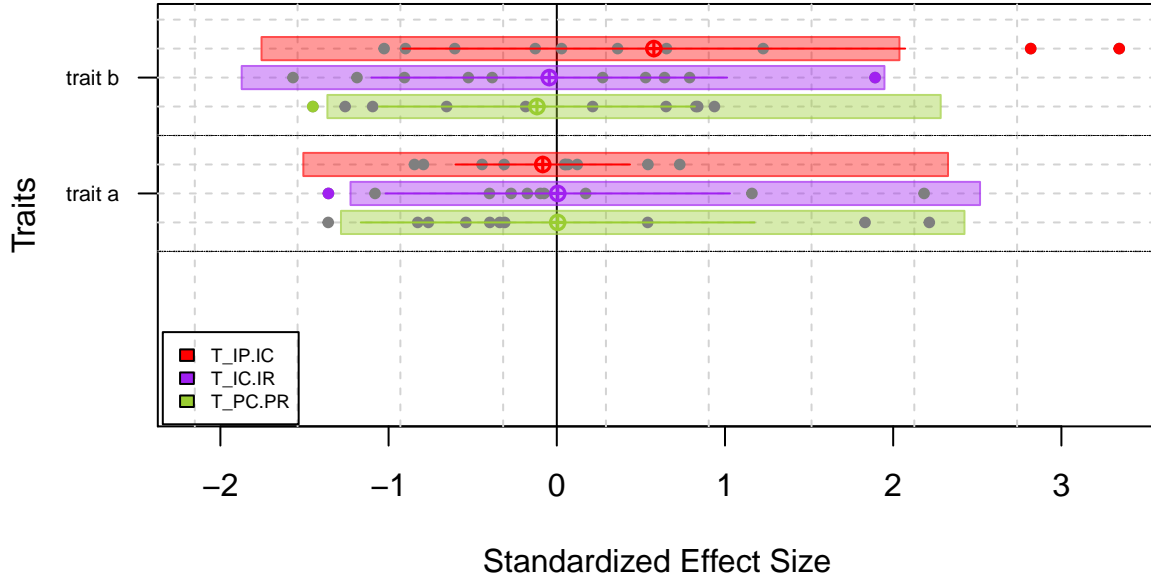
Figure 3: Results of T-statistics for one randomization without filter: $T_{IP/IC}$ in red, $T_{IC/IR}$ in purple and $T_{PC/PR}$ in green.

```
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   Assess 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 indice for one trait across all the communities.

   We test for the **alpha error** of global p-values in the same way as local p-values.

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 this false-positive cases. Norm: normal distribution, Uni: uniforme distribution.
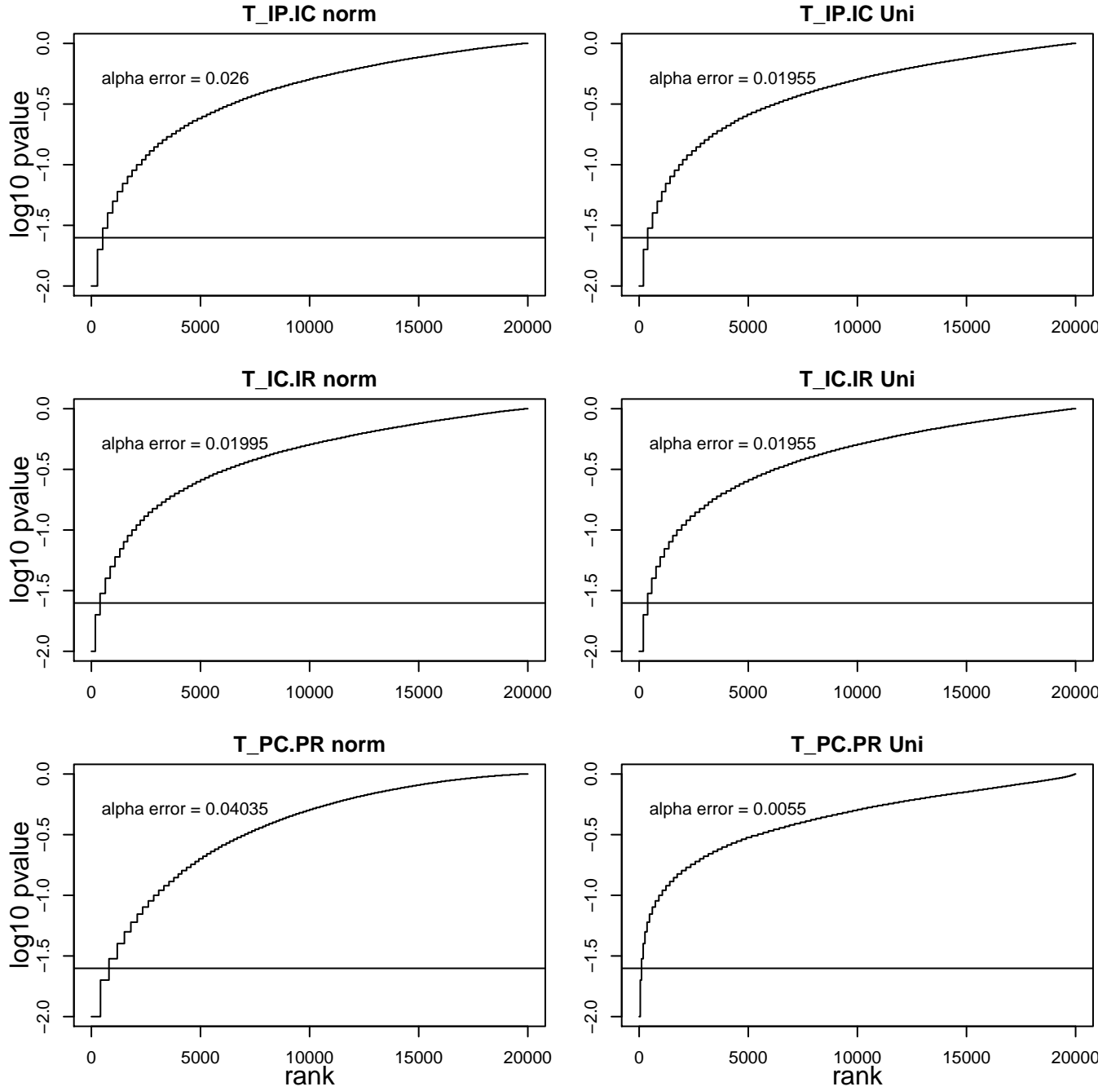
We plot the ordered mean [3] standardized effect size (SES)compute as $(the obsevedvalue - themeanvalueamongsimulation)/thestandarderroramongsimulations$ and the 95% confidence interval in grey.

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)
```

---

[3] mean of the ten communities values

```r
#T_IP.IC.distriUni
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)
```

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 between communities).

Table 2: Alpha-errors for T-statistics

Traits	Indices	Average between communities?	alpha-error
Trait normally distributed	$T_{IP/IC}$	no (local)	0.026
		yes (global)	0.001
	$T_{IC/IR}$	no (local)	0.02
		yes (global)	0.001
	$T_{PC/PR}$	no (local)	0.04
		yes (global)	0.001
Trait uniformly distributed	$T_{IP/IC}$	no (local)	0.02
		yes (global)	0.001
	$T_{IC/IR}$	no (local)	0.02
		yes (global)	0.001
	$T_{PC/PR}$	no (local)	0.006
		yes (global)	0.001
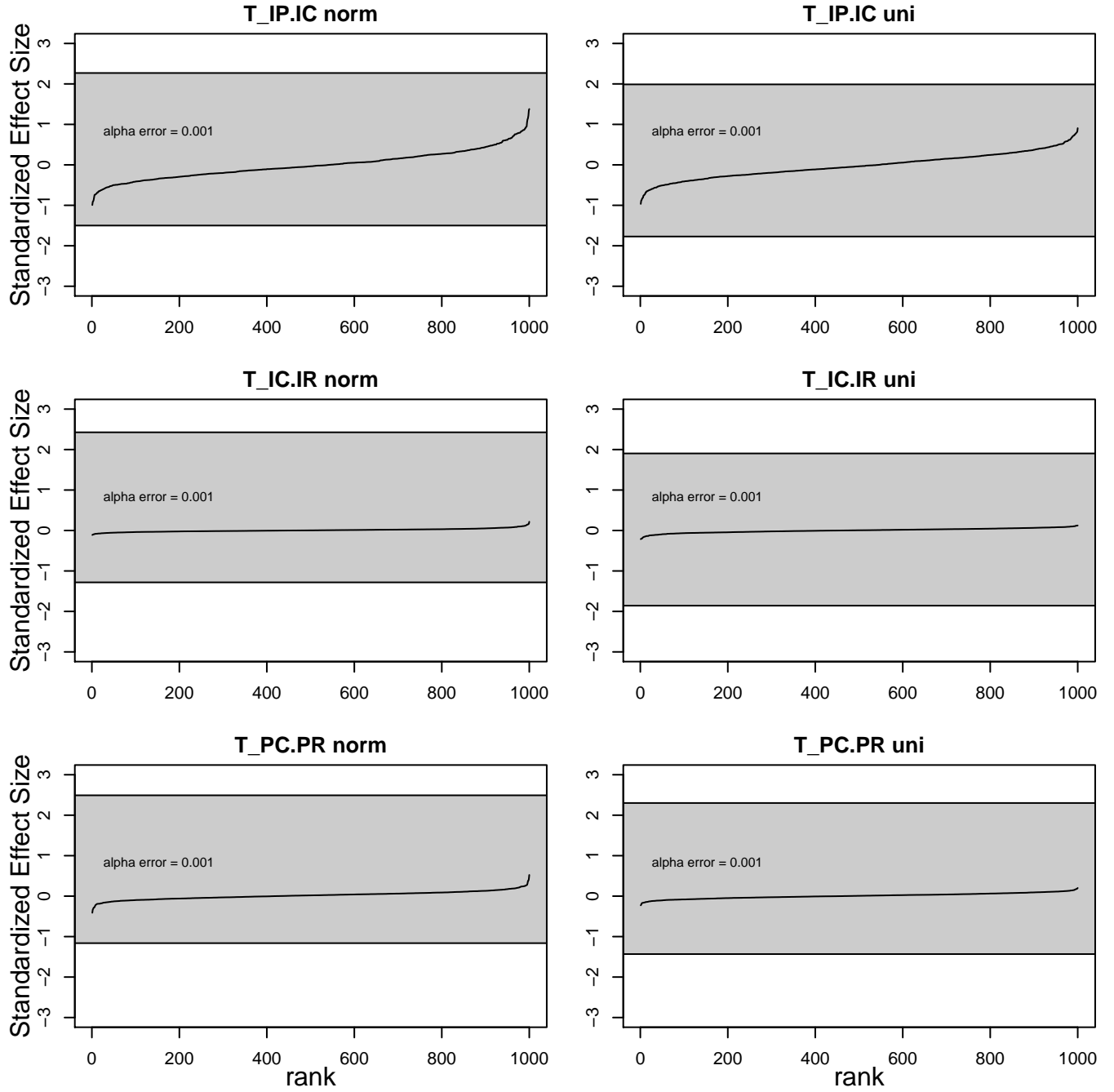
Figure 5: Global alpha errors: global Standardized Effect Size are ordered before plotting. The grey area represent the confidence interval at 2.5 percent on each side. Consequently, p-values outside this area are false-positive cases. Alpha-error is the proportion of this false-positive cases. Norm: normal distribution, Uni: uniforme distribution.

The results of table 2 attest 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 wanted alpha-error of 2.5%. Logically the global indices which averaged local indices are far more robust. Note that the higher alpha-error is obtain using $T_{PC/PR}$ on the trait normally distributed (0.04).

3   Internal filter: assess 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: we define (i) a gradient of distance between species means traits distributions and (ii) a gradient of the mean variance in species traits distributions. It is strongly link 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` correspond to the distance between the species trait means and `w` correspond to the mean variance of species trait distribution.

```
# Number of values for each parameters: 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 defined the minimum values for the mean traits values (the maximum values are defined to 250). In decreasing the range of traits values while keeping the same number of species, there is a decrease of mean distance between species means of traits values. Thus the mean for each species is drawing 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 traits distributions for each species. In order to decorrelating the mean and standard deviation of species traits distributions, `param_SD` is sample before the analysis.

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

from a uniform distribution with the range depending of 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 simulation of 10 parameters values and 2000 permutations.

```r
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 abondances
 # 100 individuals by populations
 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 <- as.factor(rep(com, Nind/Ncom))

 # Defining traits mean and sd by 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 simulation
```

3.2 Results for simulations of internal filtering

Let's see the result for one typical randomization. We can plot the distribution of traits 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 result in two parts: (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 Local p-values results

Here we call local p-values the p-values corresponding to one indice for one trait in one community. First, we can represent the ordered p.value in decimal logarithms and calculate the power of the $T_{IP/IC}$ indice using the beta-error[4]. The less the beta-error, the more the indice is powerful. Contrary to the test with no 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.

We can plot the local p-values in different ways. First, we can represent the ordered p.value in decimal logarithms and calculate the power of the $T_{IP/IC}$ indice using the beta-error. The less the beta-error, the more the indice is powerful. Contrary to the test with no 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.

---

[4]The proportion of case where we do not reject the null hypothesis while this hypothesis is wrong.
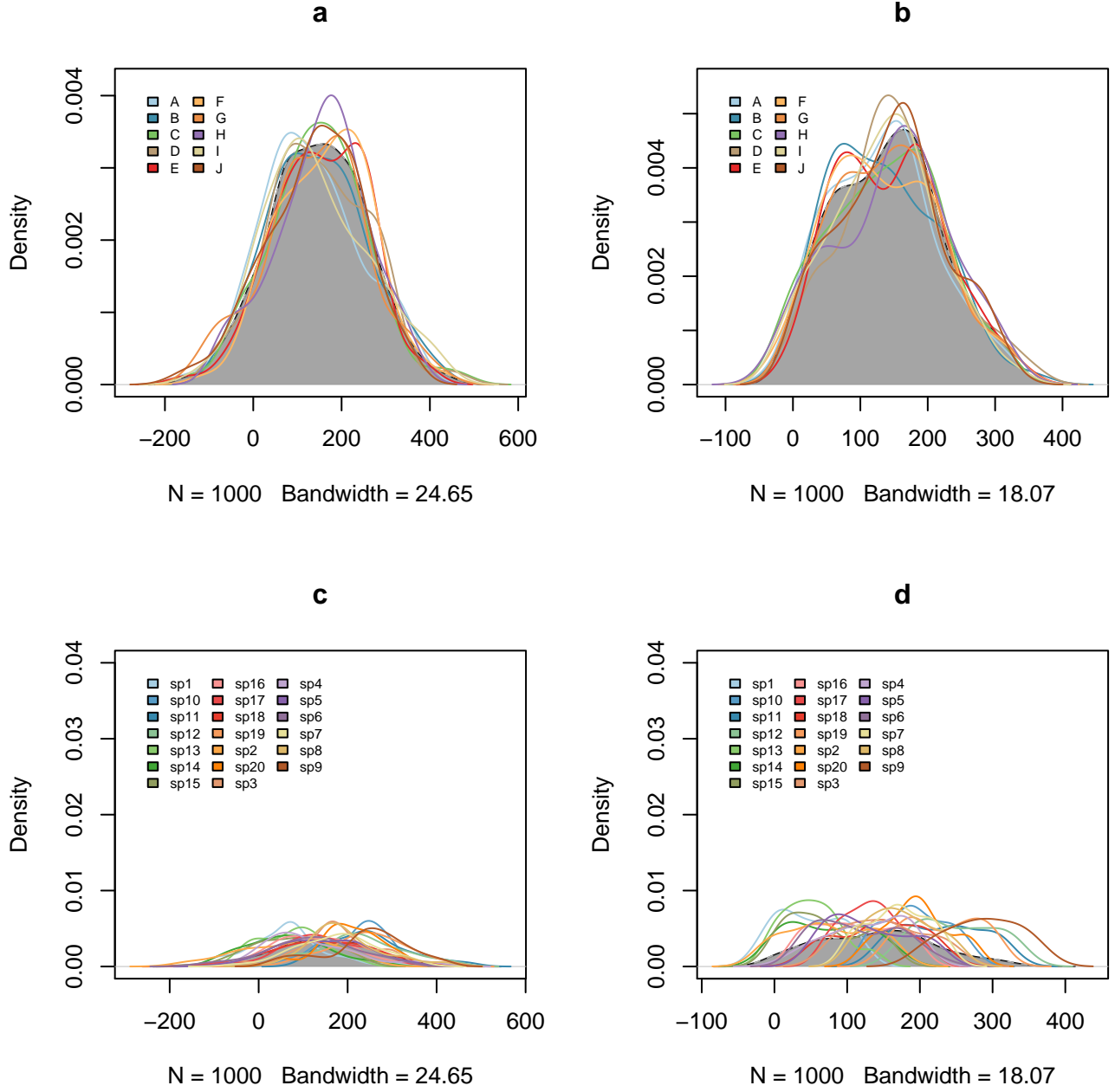
Figure 6: Distribution of traits for one randomization with internal filtering: (a) Communities trait distributions for the trait a (normal distribution); (b) Communities trait distributions for the trait b (uniform distribution); (c) Species trait distributions for the trait a; (d) Species trait distributions for the trait b
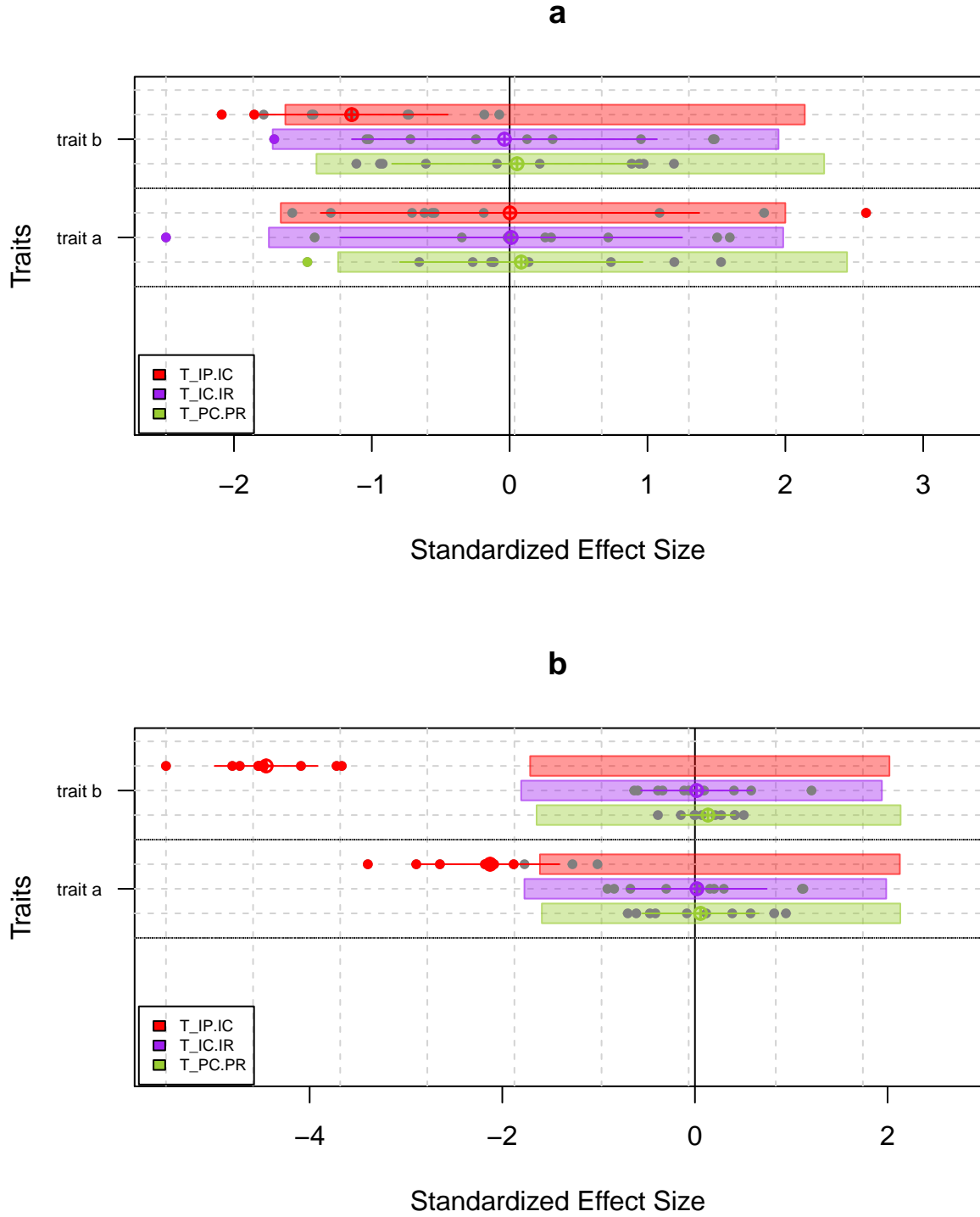
Figure 7: Results of T-statistics for two randomization with contrasted strength of internal filtering: $T_{IP/IC}$ in red, $T_{IC/IR}$ in purple and $T_{PC/PR}$ in green. (a) Lower strength of internal filtering and (b) higher strength of internal filtering.
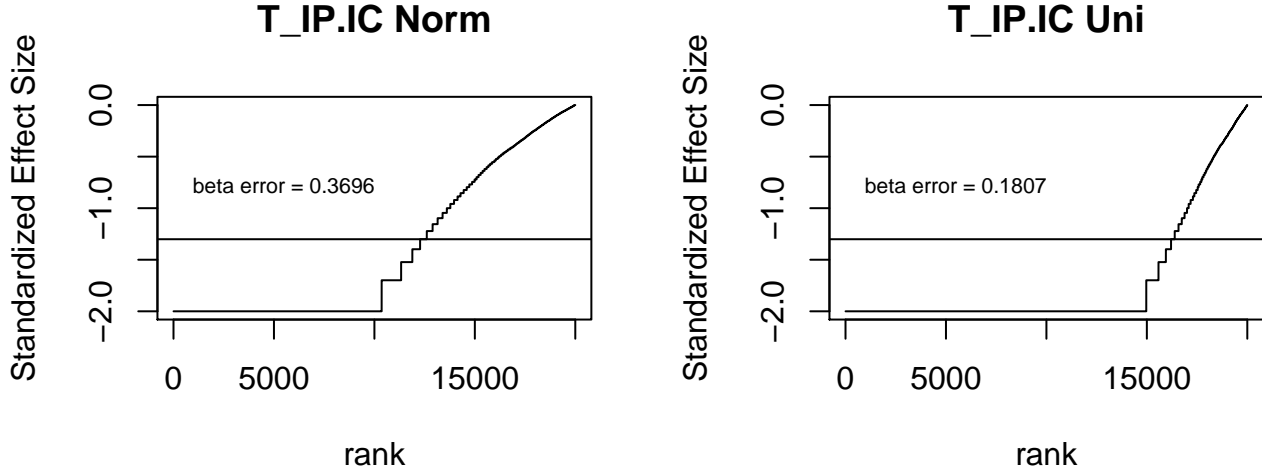
Figure 8: Beta-error mixing all strength of internal filtering: Standardized effect size of $T_{IP/IC}$ are ordered and plot. The black line represent the value of 0.05. Consequently, p-values above this line are false-negative cases. Norm: normal distribution, Uni: uniforme distribution.

3.2.1.1 Calculation of beta-error mixing all strength of filtering

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

As we add stochasticity around the initial parameters values, we can measure the strength of internal filtering by (i) the initial parameter values or (ii) by calculating the modeled parameter values[5].

---

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

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))
            $index_1_1$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 point 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 parameters values we can calculate the proportion of case where we do not reject the null hypothesis while this hypothesis is wrong (*i.e.* the Beta-error). The next chunk only show the calculation for the $T_{IC/IR}$ metrics on 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),
```
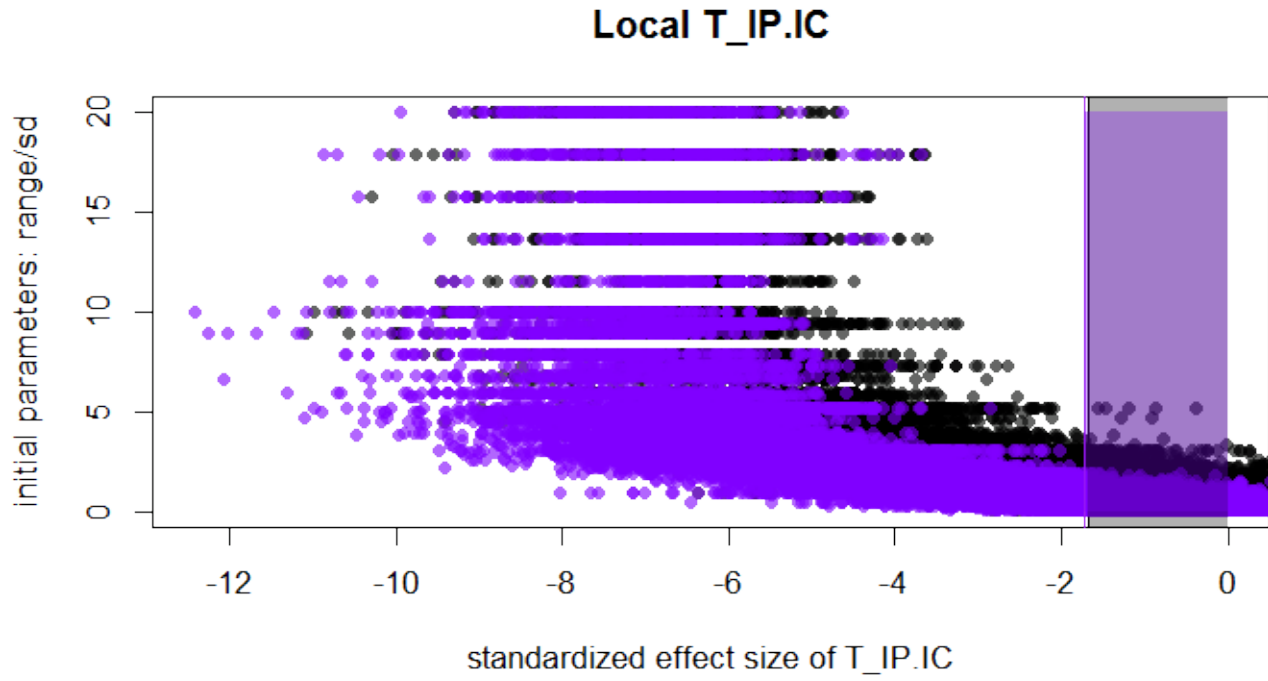
Figure 9: Local $T_{IP/IC}$ SES and initial parameters values: Standardized effect size of $T_{IP/IC}$ in function of the strength of internal filter defined by the ratio of the initial range parameter by the initial standard error parameter. Colored area represent the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and b (uniform) in purple.

```
      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 <-
 (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 $T_{IP/IC}$ in 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 $T_{IP/IC}$ in 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)	0.67	0.66	0.60	0.60	0.41	0.35	0.30	0.20	0.11	0.07
effect of sd (b)	0.37	0.38	0.34	0.33	0.18	0.15	0.13	0.06	0.02	0.01
effect of range (a)	0.82	0.73	0.65	0.51	0.37	0.32	0.22	0.14	0.11	0.09
effect of range (b)	0.66	0.51	0.39	0.24	0.10	0.05	0.02	0.01	0.00	0.00

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 66.87% for the higher value of standard errors and thus for the lower strength of internal filtering. The good behavior of the function `Tstats` is ascertain by the decrease of beta-error along increasing strength of 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 values in communities by their 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
```
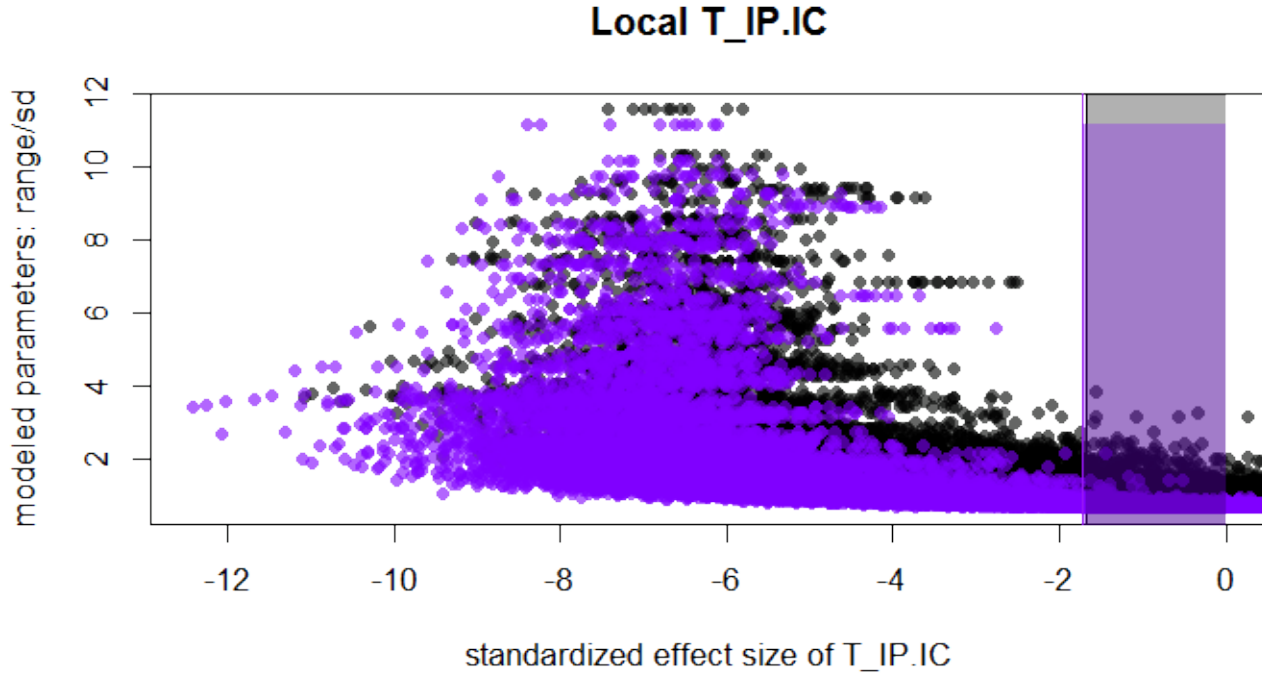
Figure 10: Local $T_{IP/IC}$ SES and modeled parameters values: Standardized effect size of $T_{IP/IC}$ in function of the strength of internal filter defined by the ratio of the modeled range parameter by the modeled standard error parameter. The colored rectangles represent the mean confidence interval for SES values ($alpha = 0.95$). Trait a (normal) in black and b (uniform) in purple.

```r
modeled_param_norm.loc <- rep(modeled_param_norm, each = Ncom)
modeled_param_uni.loc <- rep(modeled_param_uni, each = Ncom)
```

```r
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 = 0.325; trait b: Pearson correlation = 0.36) 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 indice for one trait across all the communities.

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

Again, this first results mix different initial parameter values. Now, we can plot the SES values in relation to the strength of the internal 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_stock2` [6]).

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 parameters values (Fig. 12 and **??**).

---

[6]This modeled values are stochastic versions of the initial values.
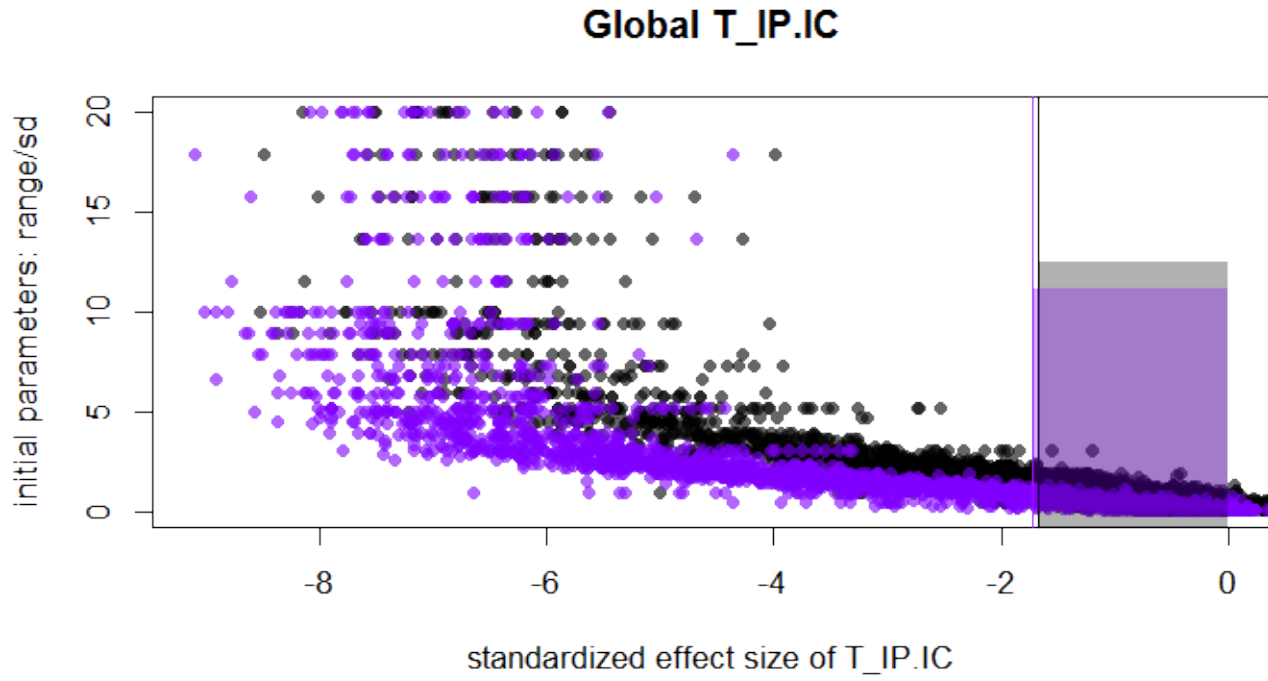
Figure 11: Global $T_{IP/IC}$ SES and initial parameters values: Standardized effect size of $T_{IP/IC}$ in function of the strength of internal filter defined by the ratio of the inital range parameter by the initial standard error parameter. Colored area represent the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and b (uniform) in purple.

Figure 12: Global $T_{IP/IC}$ SES and modeled parameters values: Standardized effect size of $T_{IP/IC}$ in function of the strength of internal filter defined by the ratio of the modeled range parameter by the modeled standard error parameter. Colored area represent the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and b (uniform) in purple.
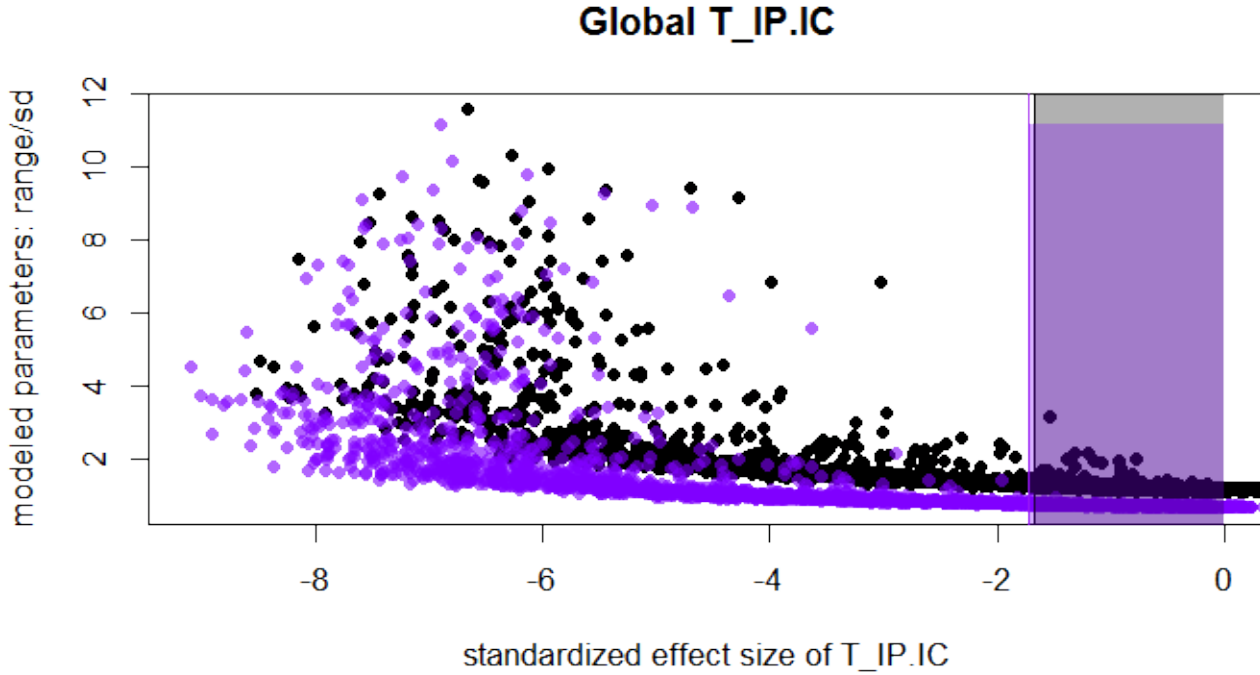
```r
mean_dist_sp.interm <- t(matrix(unlist(lapply(mean_dist_sp, function(x) x)), nrow = 2))
mean_sd_of_com <- unlist(lapply(sd.sp_stock2, function(x) mean(x)))

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 compute exponential linear model and identify the strength from which the beta-error is inferior to 0.05. This strength is define by the two 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 satisfying if the ratio of the initial parameter is superior to 1.05. For the trait "b" uniformly distributed, this value is 0.51. Further investigation on real data with different strength of internal filtering are needed to complete these simulated values.
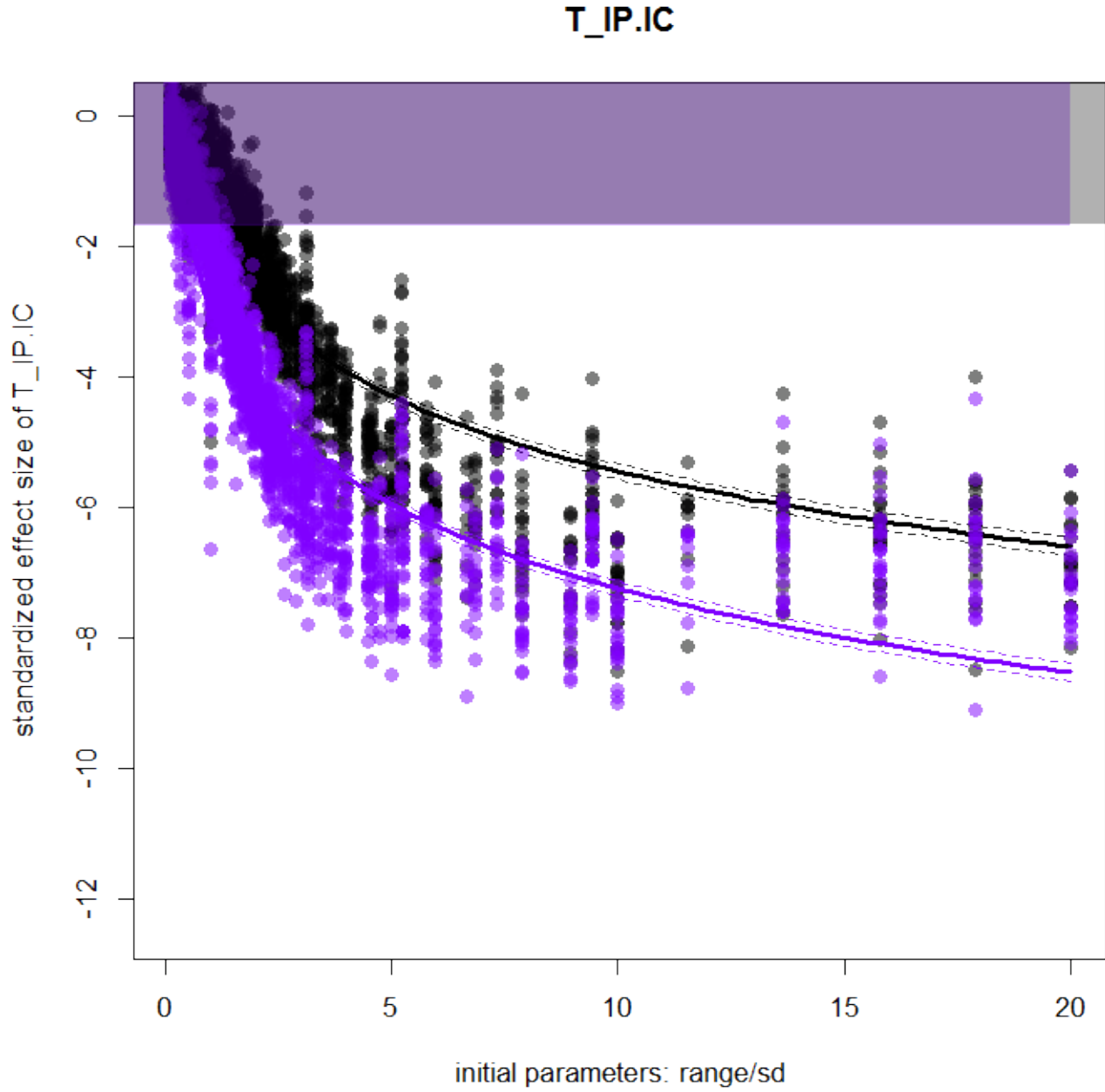
Figure 13: Power of $T_{IP/IC}$ to detect internal filtering: Standardised Effect Size (SES) of $T_{IP/IC}$ in relation to the strength of internal filtering. Trait 'a' normally distributed is shown in black, traits 'b' uniformally distributed in purple. Open circle represent local SES values wheras close one represent global SES values. The exponential linear model is presented with 5 percents confidence interval on both side. Colored area represent the mean confidence interval of SES values ($alpha = 0.95$)

4 External Filter

4.1 Randomization outline

As for internal filter we modulate the strength of the external filter thanks to two parameters: we define (i) a gradient of range between communities means traits distributions and (ii) a gradient of the mean variance in communities traits distributions. We expect the external filtering to increase (i) when the range in communities means increase (*i.e.* the environmental gradient is larger) and (ii) when the variance in the communities traits distributions decrease (*i.e.* the environmental gradient is stronger).

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

(i) `param_RANGE.init` is a vector of length 10 which defined the above range for the mean traits values (the maximum value for the mean is assessed to 250). In decreasing the range of traits values while keeping the same number of communities, there is a decrease in the overlap of the communities traits distributions and consequently an decrease of the impact of external filtering in shaping traits distributions. Thus the mean for each communities is drawing from a normal distribution with standard deviation of 10 (parameter `mean_sd.param`) and means evenly [7] of 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 traits distributions for each species. In order to decorrelating the mean and standard deviation of species traits distributions, `param_SD` is sample before the analysis.

Thus for the trait "a" the trait value for each individual is drawing from a normal distribution with the mean depending of it community attribute. For the trait "b" the trait value for each individual is drawing from a uniform distribution with the range depending of it community attribute.

```
# Parameter for the range of communities traits 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

 ex.sp3 <- c()
```

---

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

```
ex.com3 <- matrix(0, nrow = 10, ncol = 20)
for(i in 1: 10){
 ex.com.interm <- table(sample(sp, size = 100, prob = rlnorm(20, 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 <- as.factor(rep(com, 100))

#Defining mean and sd by 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(1000, 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 simulation
```

4.2   Results for simulations of external filtering

Let's see the result for one typical randomization. We can plot the distribution of traits 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 case (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-value for different parameters values. Consequently we plot the standard effect size (SES) of $T_{IC/IR}$ and $T_{PC/PR}$ in relation to the initial parameters 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))
            $index_1_3$ses[, 2]))

SES.inf.MEAN.norm_Tpcpr <- unlist(lapply(res.simu3, function(x)
            ses.listofindex(as.listofindex(x))
```
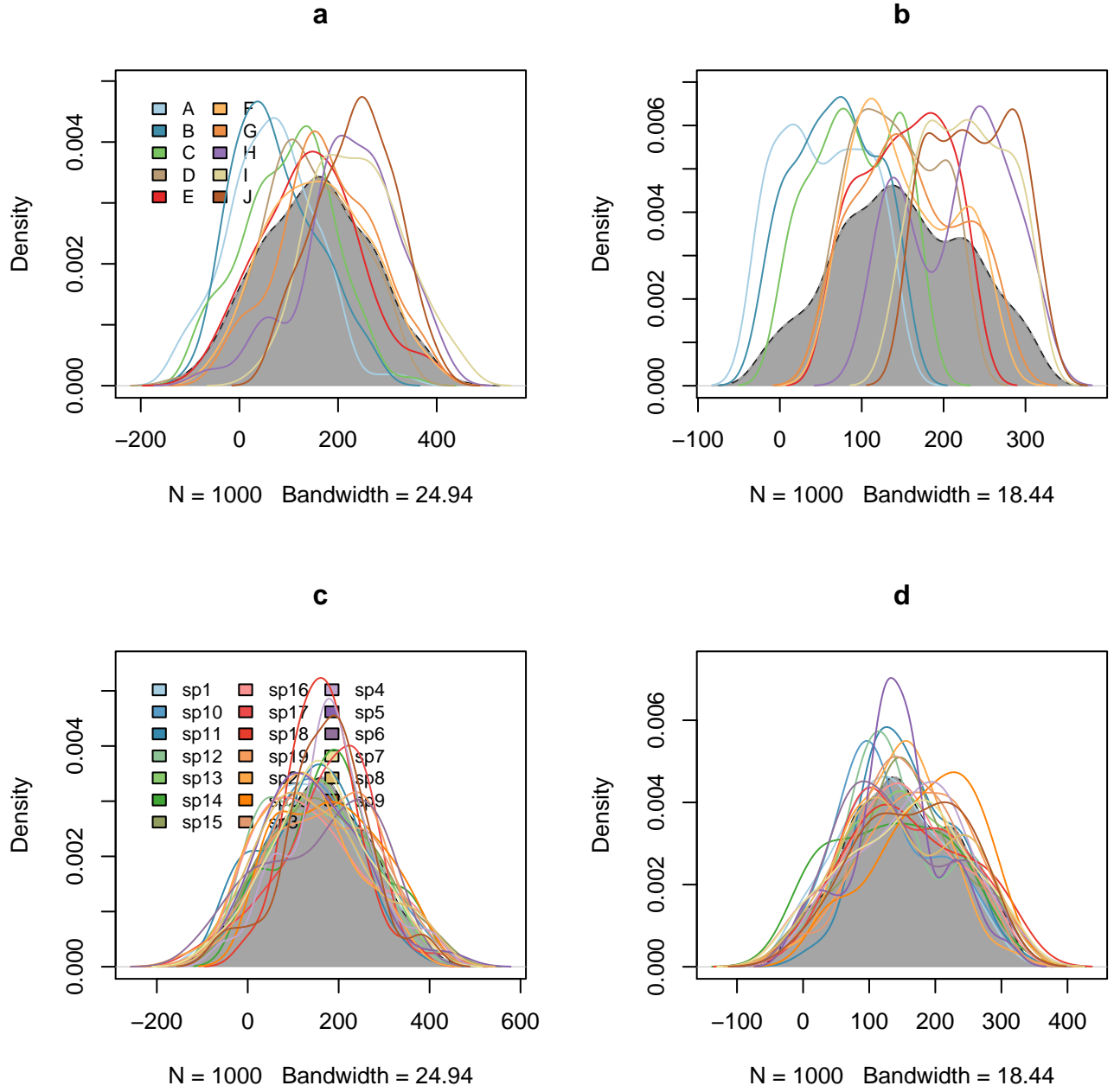
Figure 14: Distribution of traits for one randomization with external filter: (a) Communities trait distributions for the trait a (normal distribution); (b) Communities trait distributions for the trait b (uniform distribution); (c) Species trait distributions for the trait a; (d) Species trait distributions for the trait b
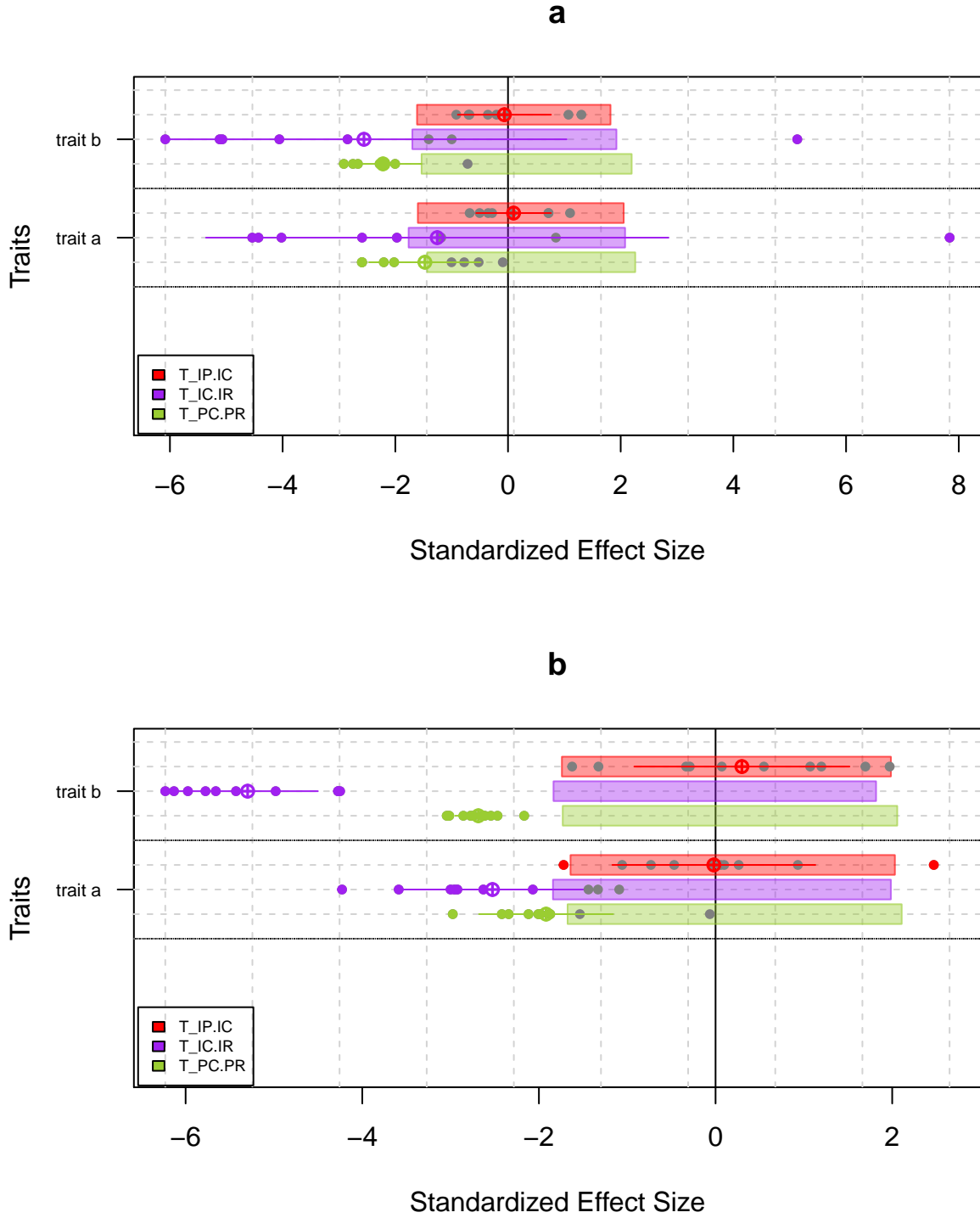
Figure 15: Results of T-statistics of traits for two randomization with contrasted strength of external filter: $T_{IP/IC}$ in red, $T_{IC/IR}$ in purple and $T_{PC/PR}$ in green. (a) Lower strength of external filtering and (b) higher strength of external filtering.

```
              $index_1_3$ses.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 traits "b" is in purple. The colored rectangles represent the null model area with alpha = 5%. Thus, when a point is outside these 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 replicate Ncom times (there is 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 parameters values we can calculate the proportion of case where we do not reject the null hypothesis while this hypothesis is wrong (*i.e.* the Beta-error). The next chunk only show the calculation for the $T_{IC/IR}$ metrics on the trait "a".
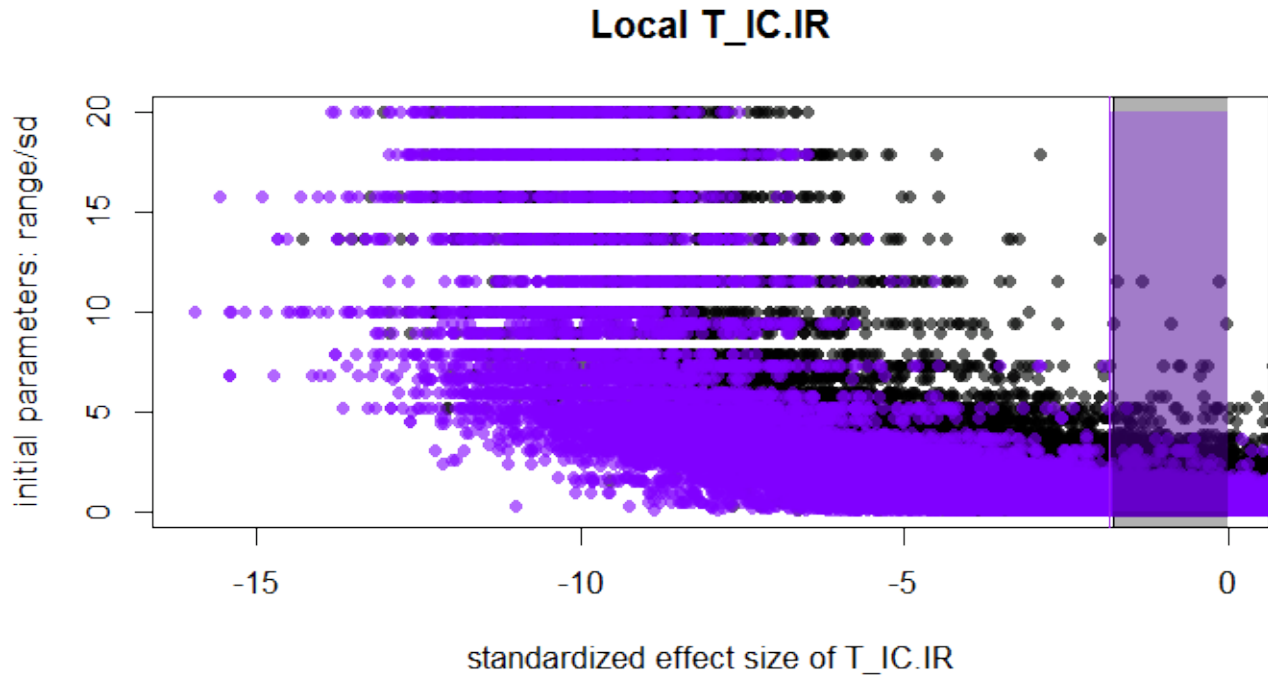
Figure 16: Local $T_{IC/IR}$ SES and initial parameters values: Standardized effect size of $T_{IC/IR}$ in function of the strength of external filter defined by the ratio of the initial range parameter by the initial standard error parameter. Colored area represent the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and b (uniform) in purple.
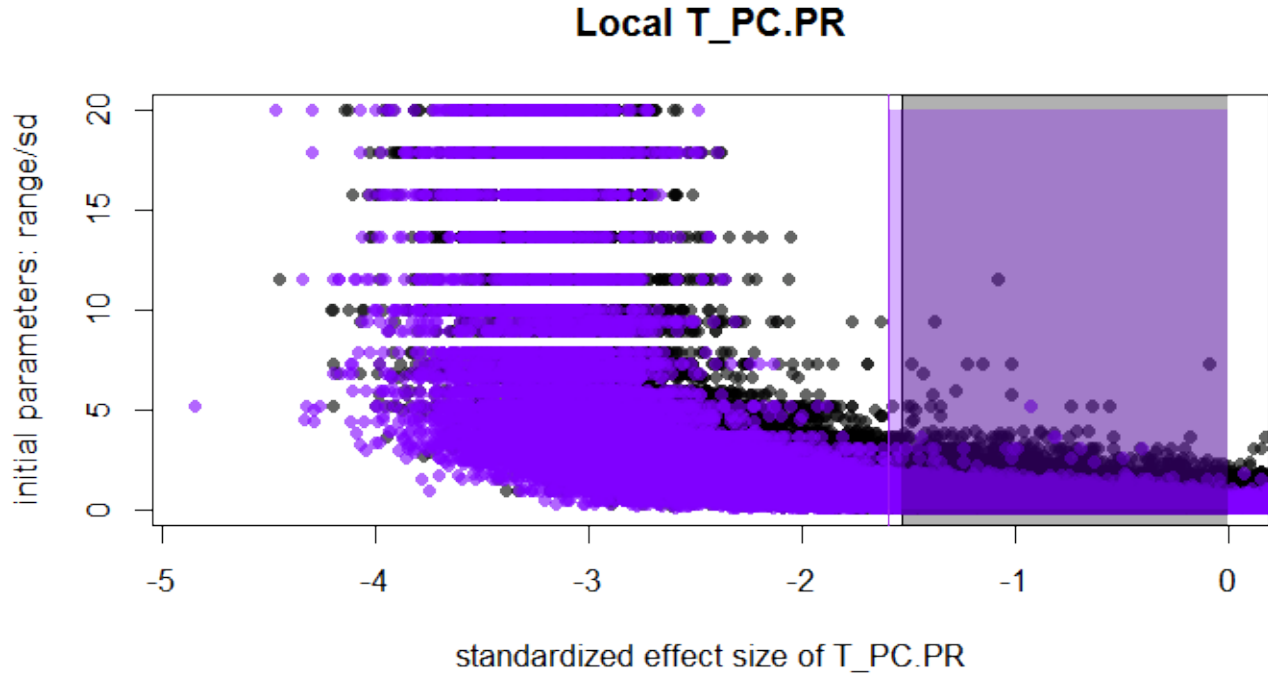
Figure 17: Local $T_{PC/PR}$ SES and initial parameters values: Standardized effect size of $T_{PC/PR}$ in function of the strength of external filter defined by the ratio of the initial range parameter by the initial standard error parameter. Colored area represent the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and 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 $T_{IC/IR}$ in 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 $T_{IC/IR}$ in 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)	0.68	0.61	0.56	0.45	0.40	0.36	0.30	0.21	0.17	0.08
effect of sd (b)	0.36	0.28	0.28	0.19	0.19	0.16	0.14	0.08	0.08	0.03
effect of range (a)	0.66	0.60	0.58	0.48	0.44	0.36	0.26	0.23	0.14	0.09
effect of range (b)	0.53	0.43	0.34	0.22	0.14	0.08	0.02	0.01	0.00	0.00

```
print(xtable(res_beta_Ticir, caption = 'Local beta-error of $T_{PC/PR}$ in 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 $T_{PC/PR}$ in 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)	0.68	0.61	0.56	0.45	0.40	0.36	0.30	0.21	0.17	0.08
effect of sd (b)	0.36	0.28	0.28	0.19	0.19	0.16	0.14	0.08	0.08	0.03
effect of range (a)	0.66	0.60	0.58	0.48	0.44	0.36	0.26	0.23	0.14	0.09
effect of range (b)	0.53	0.43	0.34	0.22	0.14	0.08	0.02	0.01	0.00	0.00

Now we can see the beta-error for each strength (table 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 67.57% 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 in 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 parameter either for the range (trait a: Pearson correlation = 0.245; trait b: Pearson correlation = 0.372) 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 indice for one trait across all the communities.
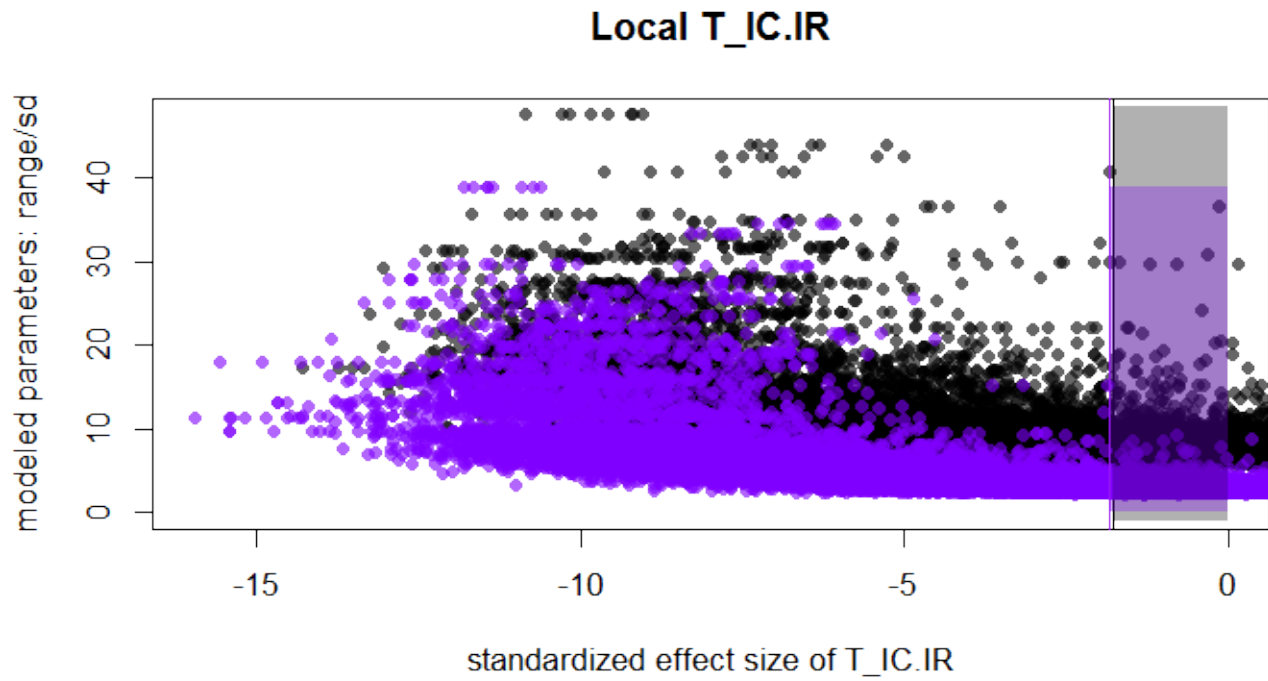
Figure 18: Local $T_{IC/IR}$ SES and modeled parameters values: Standardized effect size of $T_{IC/IR}$ in function of the strength of external filter defined by the ratio of the modeled range parameter by the modeled standard error parameter. Colored area represent the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and b (uniform) in purple.
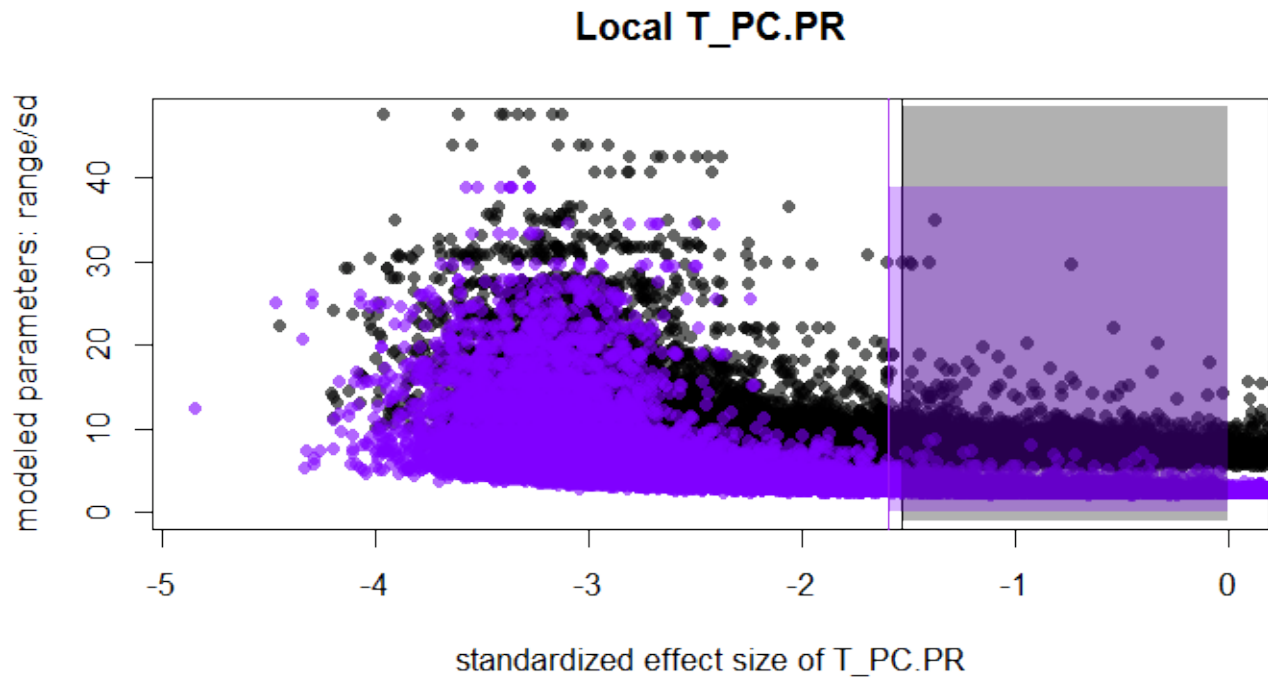
Figure 19: Local $T_{PC/PR}$ SES and modeled parameters values: Standardized effect size of $T_{PC/PR}$ in function of the strength of external filter defined by the ratio of the modeled range parameter by the modeled standard error parameter. Colored area represent the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and b (uniform) in purple.

4.2.2.1 Calculation of beta-error mixing all strength of 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's just an illustration because we mingle very different strength of external filter with different initial parameter values. Plot 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, this 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` [8]).

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)
```

---

[8]This 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)
```
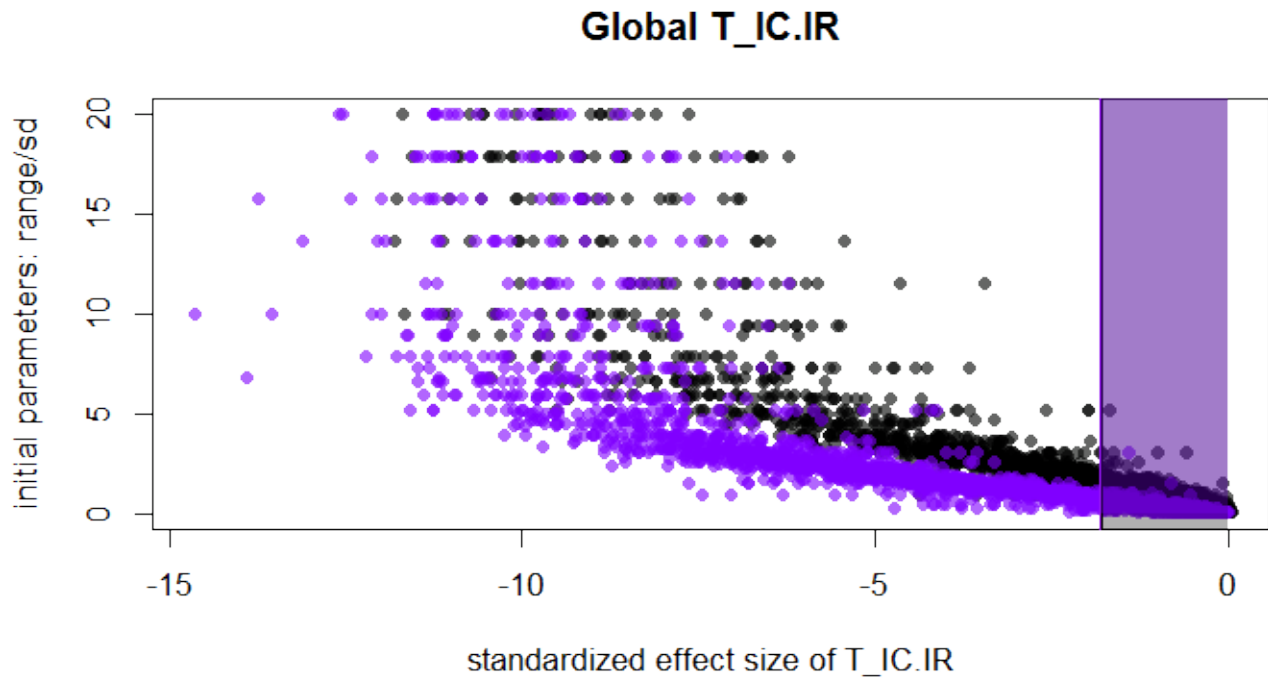
Figure 20: Global $T_{IC/IR}$ SES and initial parameters values: Standardized effect size of $T_{IC/IR}$ in function of the strength of external filter defined by the ratio of the inital range parameter by the initial standard error parameter. Colored area represent the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and b (uniform) in purple.
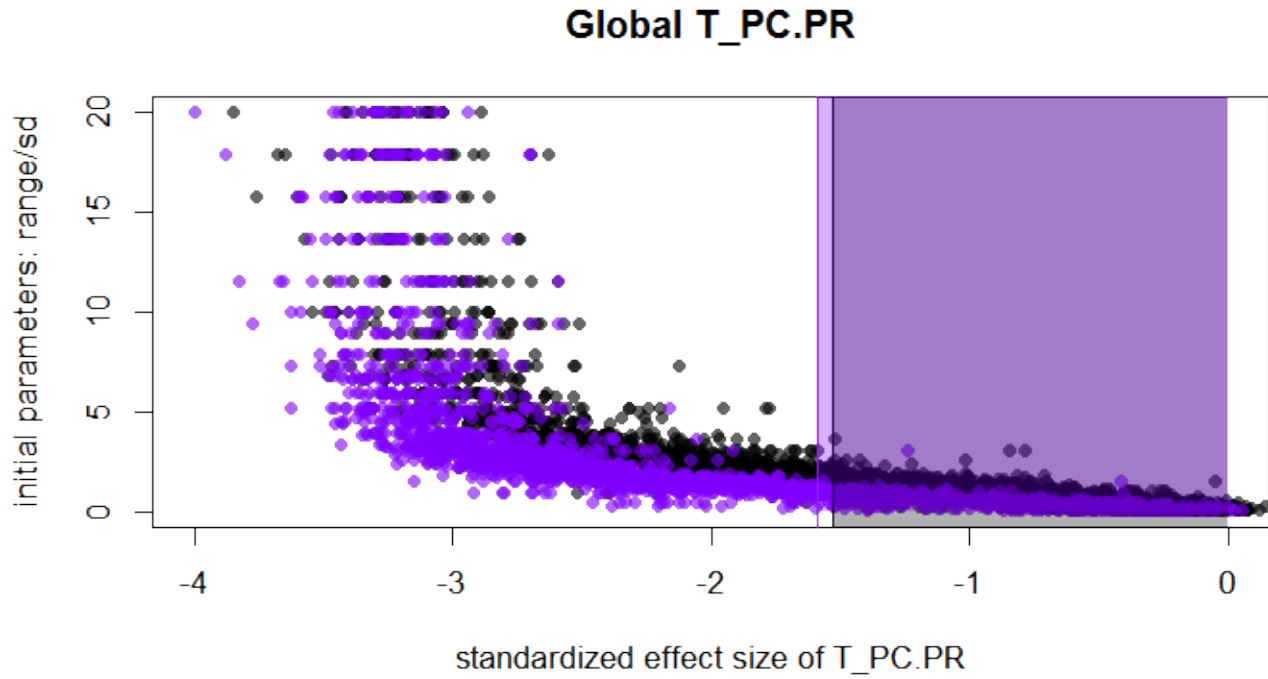
Figure 21: Global $T_{PC/PR}$ SES and initial parameters values: Standardized effect size of $T_{PC/PR}$ in function of the strength of external filter defined by the ratio of the inital range parameter by the initial standard error parameter. Colored area represent the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and 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 parameters values (Fig.).

```r
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)
```

```r
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 compute exponential linear model and identify the strength from which the beta-error is inferior to 0.05. This strength is define by the two ratio of the initial parameters `param_range` and `sd_mean.param`.

```r
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",
```
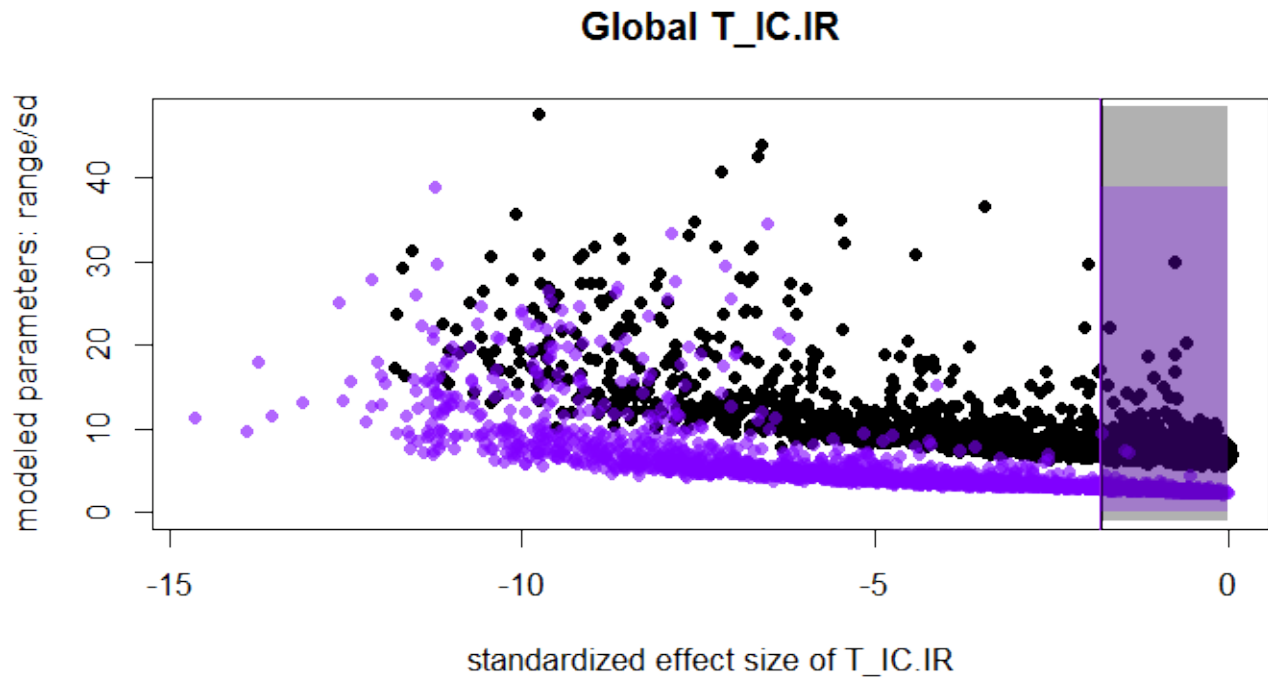
Figure 22: Global $T_{IC/IR}$ SES and modeled parameters values: Standardized effect size of $T_{IC/IR}$ in function of the strength of external filter defined by the ratio of the modeled range parameter by the modeled standard error parameter. Colored area represent the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and b (uniform) in purple.
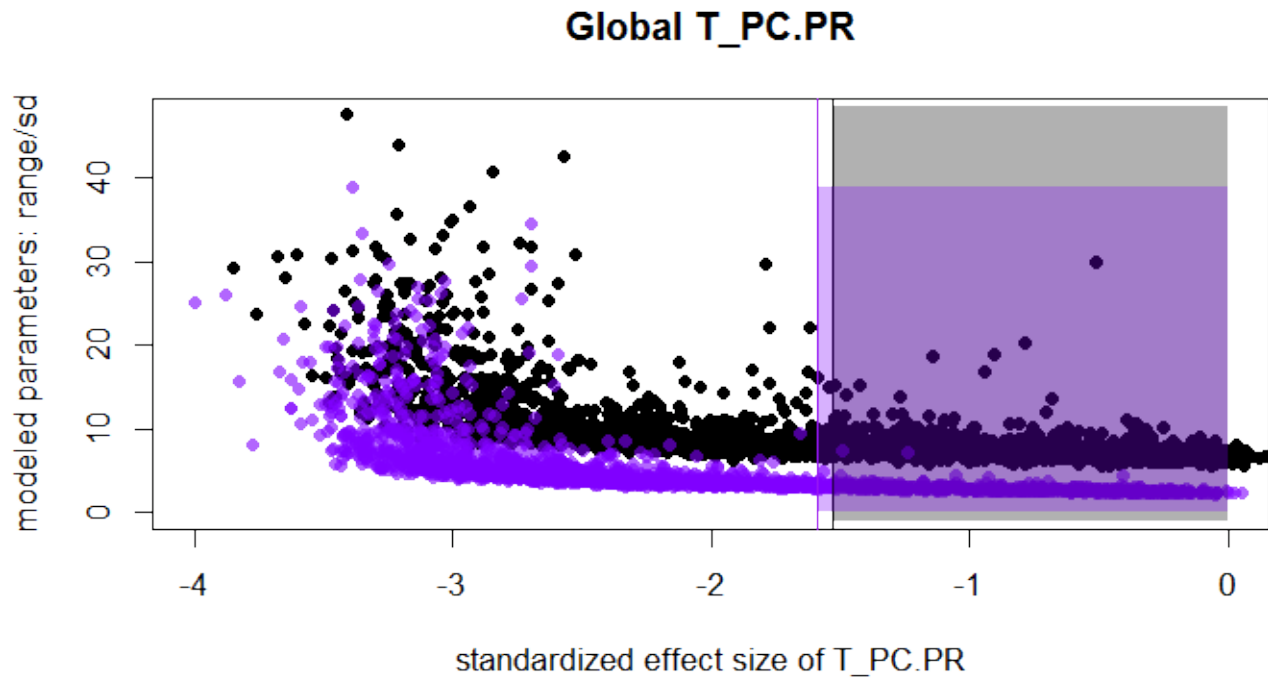
Figure 23: Global $T_{PC/PR}$ SES and modeled parameters values: Standardized effect size of $T_{PC/PR}$ in function of the strength of external filter defined by the ratio of the modeled range parameter by the modeled standard error parameter. Colored area represent the mean confidence interval of SES values ($alpha = 0.95$). Trait a (normal) in black and 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)
  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 satisfying if the ratio of the initial parameter is superior to 1.07. For the trait "b" uniformally distributed, this value is 0.51. Further investigation on real data with different strength of external filtering are needed to complete 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 compute exponential linear model and identify the strength from which the beta-error is inferior to 0.05. This strength is define by the two ratio of the initial parameters `param_range` and `sd_mean.param`.

```
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")
```
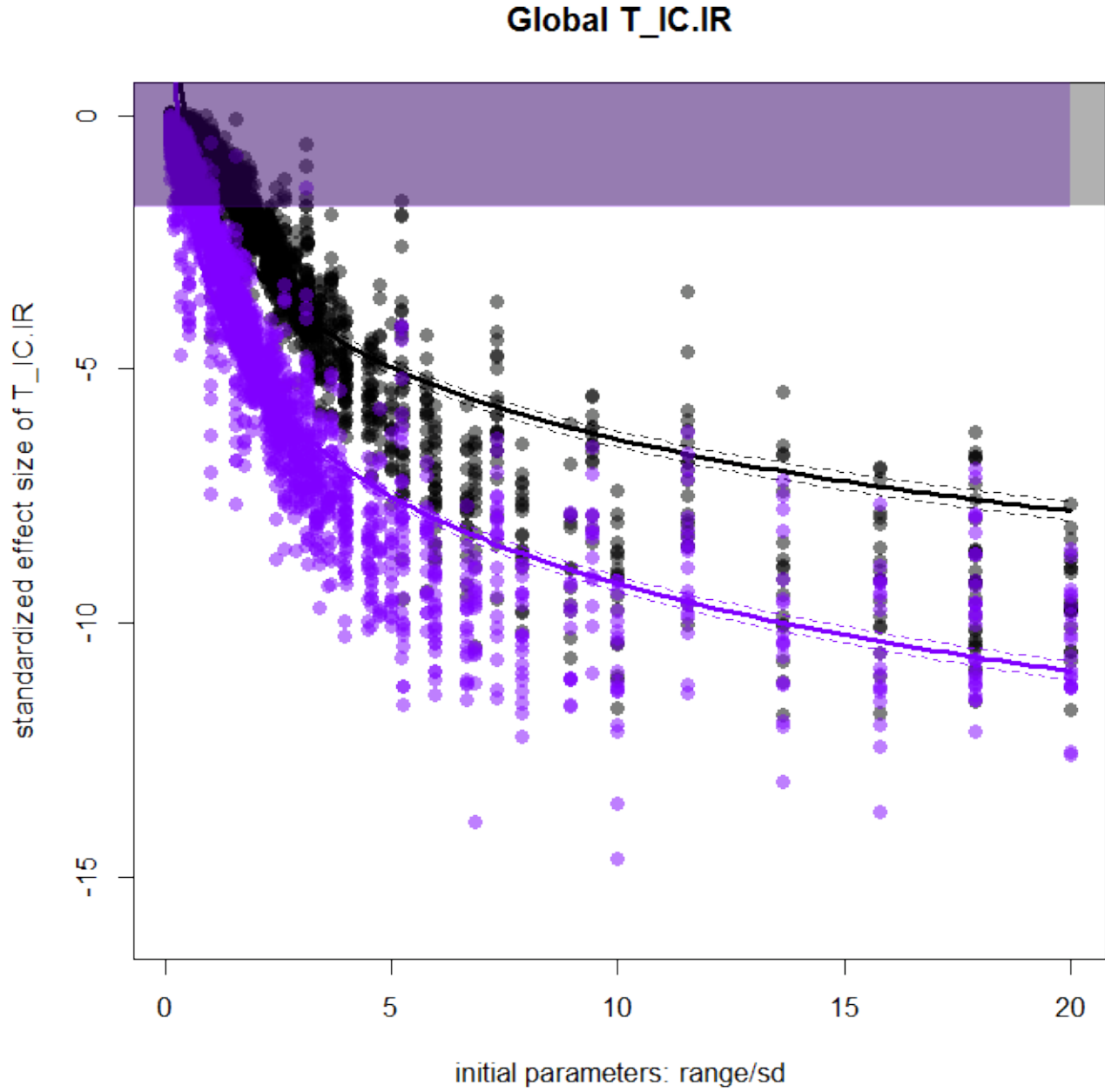
Figure 24: Power of $T_{IC/IR}$ to detect external filtering: Standardised Effect Size (SES) of $T_{IC/IR}$ in relation to the strength of external filtering. Trait 'a' normally distributed is shown in black, traits 'b' uniformally distributed in purple. Close circles represent global SES values. The exponential linear model is presented with 5 percents confidence interval on both side. Colored area represent the mean confidence interval of SES values ($alpha = 0.95$)

```
#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)
   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 satisfying if the ratio of the initial parameter is superior to 1.62. For the trait "b" uniformly distributed, this value is 0.78. Further investigation on real data with different strength of external filtering are needed to complete these simulated values. As expected, $T_{PC/PR}$ is far less powerfull than $T_{IC/IR}$ to detect external filtering either with traits "a" (1.62 *vs* 1.07) and "b" (0.78 *vs* 0.51). 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 filter.
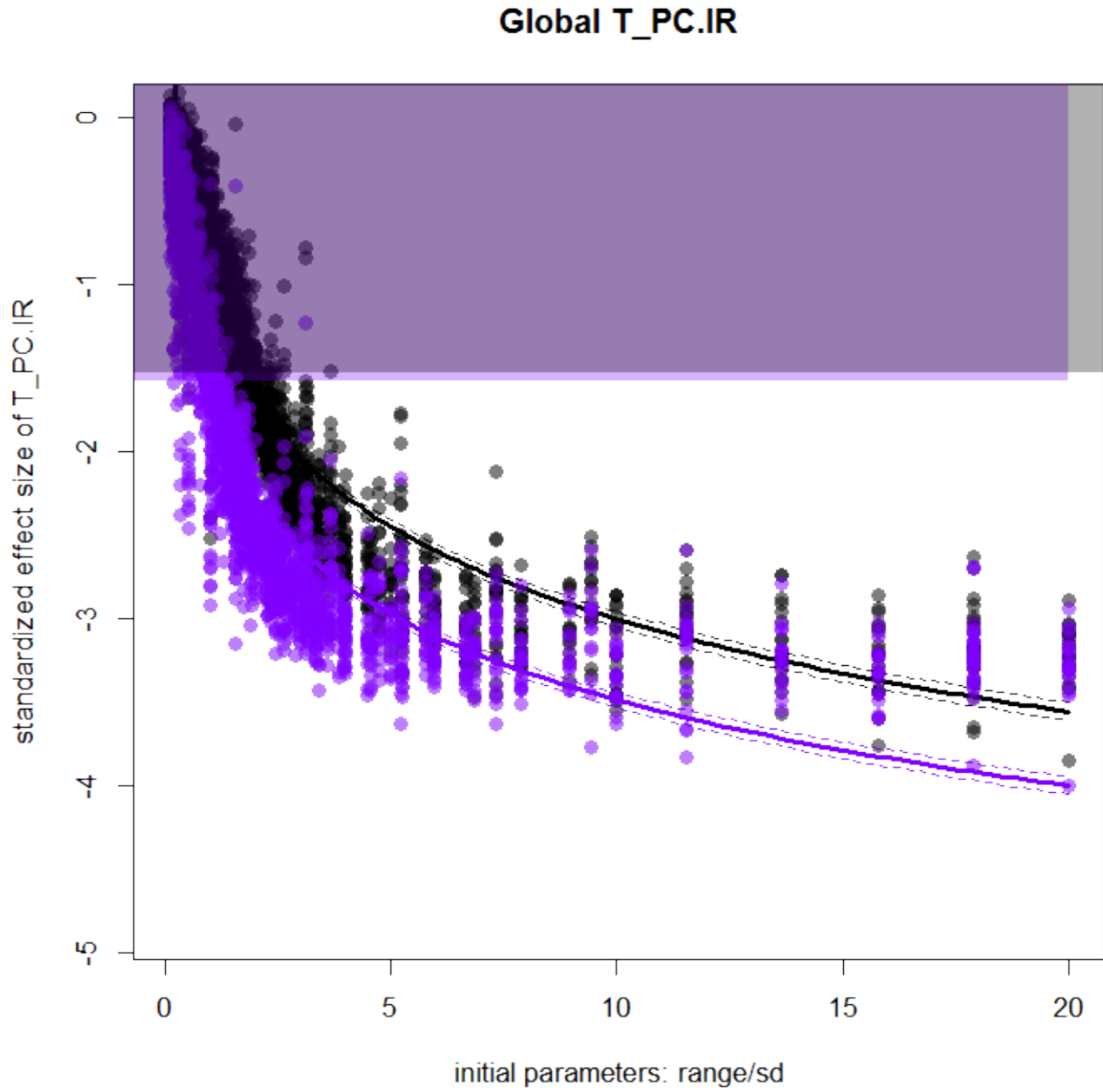
Figure 25: Power of $T_{PC/PR}$ to detect external filtering: Standardised Effect Size (SES) of $T_{PC/PR}$ in relation to the strength of external filtering. Trait 'a' normally distributed is shown in black, traits 'b' uniformally distributed in purple. Close circles represent global SES values. The exponential linear model is represented with 5 percents confidence interval on both side. Colored area represent the mean confidence interval of SES values ($alpha = 0.95$)

5 Internal and external filtering

5.1 Randomization outline

We use the same two parameters as for internal and external filtering analysis. The internal filtering is compute as in the section dedicated to internal filter. To add the effect of external filtering, we add a value to each individual belonging to a community, this values differing among communities.

```r
# 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)
```

```r
#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 = 10, ncol = 20)
for(i in 1: 10){
 ex.com.interm <- table(sample(sp, size = 100, prob = rlnorm(20, 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 <- as.factor(rep(com, 100))


#Defining traits mean and sd by 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(1000, 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 simulation
```

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(3, 1))
plot(res.simu4[[1]], main = "a")
plot(res.simu4[[nperm/2]], main = "b")
plot(res.simu4[[nperm]], main = "c")
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)
```
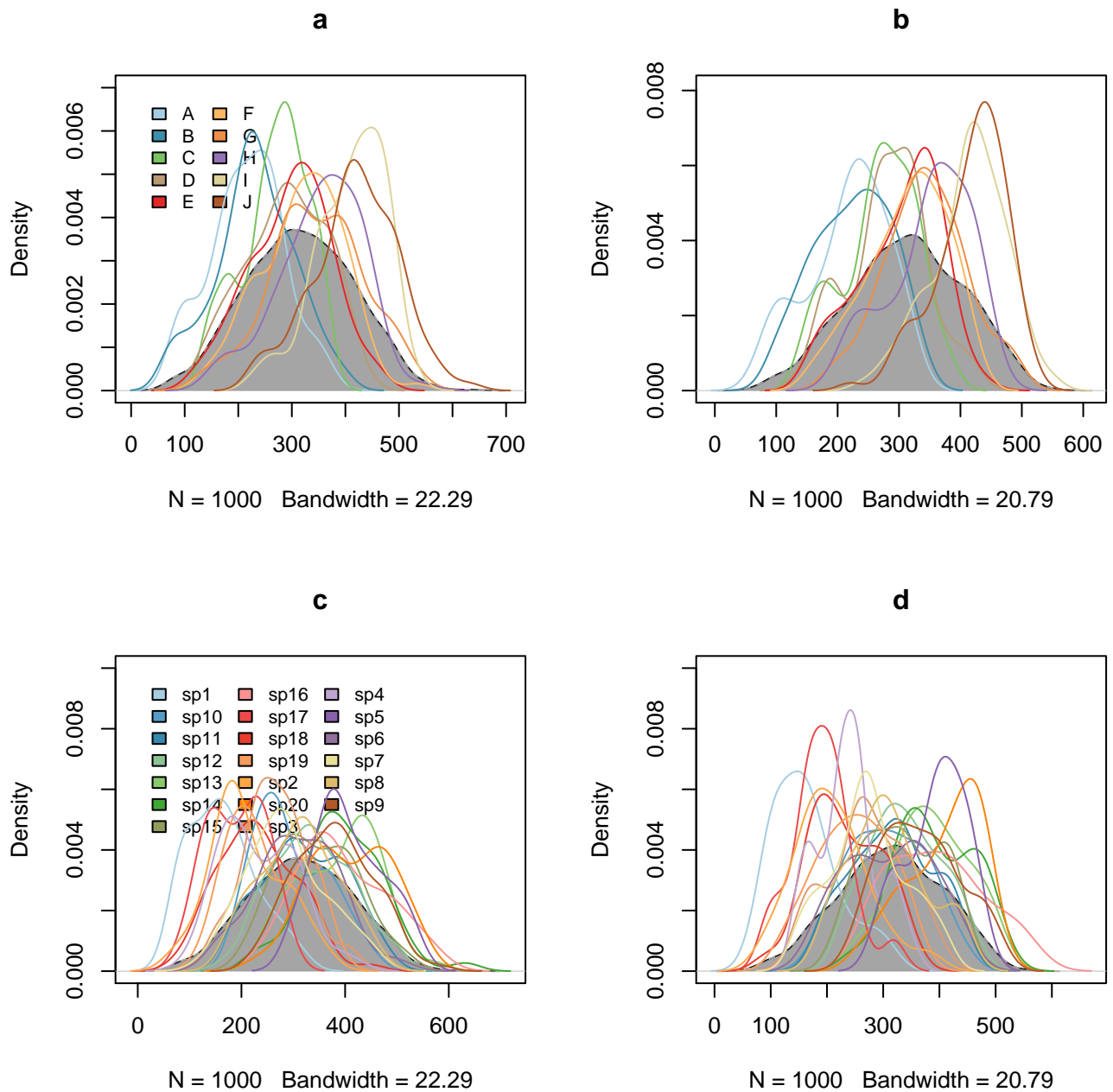
Figure 26: Distribution of traits for one randomization with both internal and external filter: (a) Communities trait distributions for the trait a (normal distribution); (b) Communities trait distributions for the trait b (uniform distribution); (c) Species trait distributions for the trait a; (d) Species trait distributions for the trait b
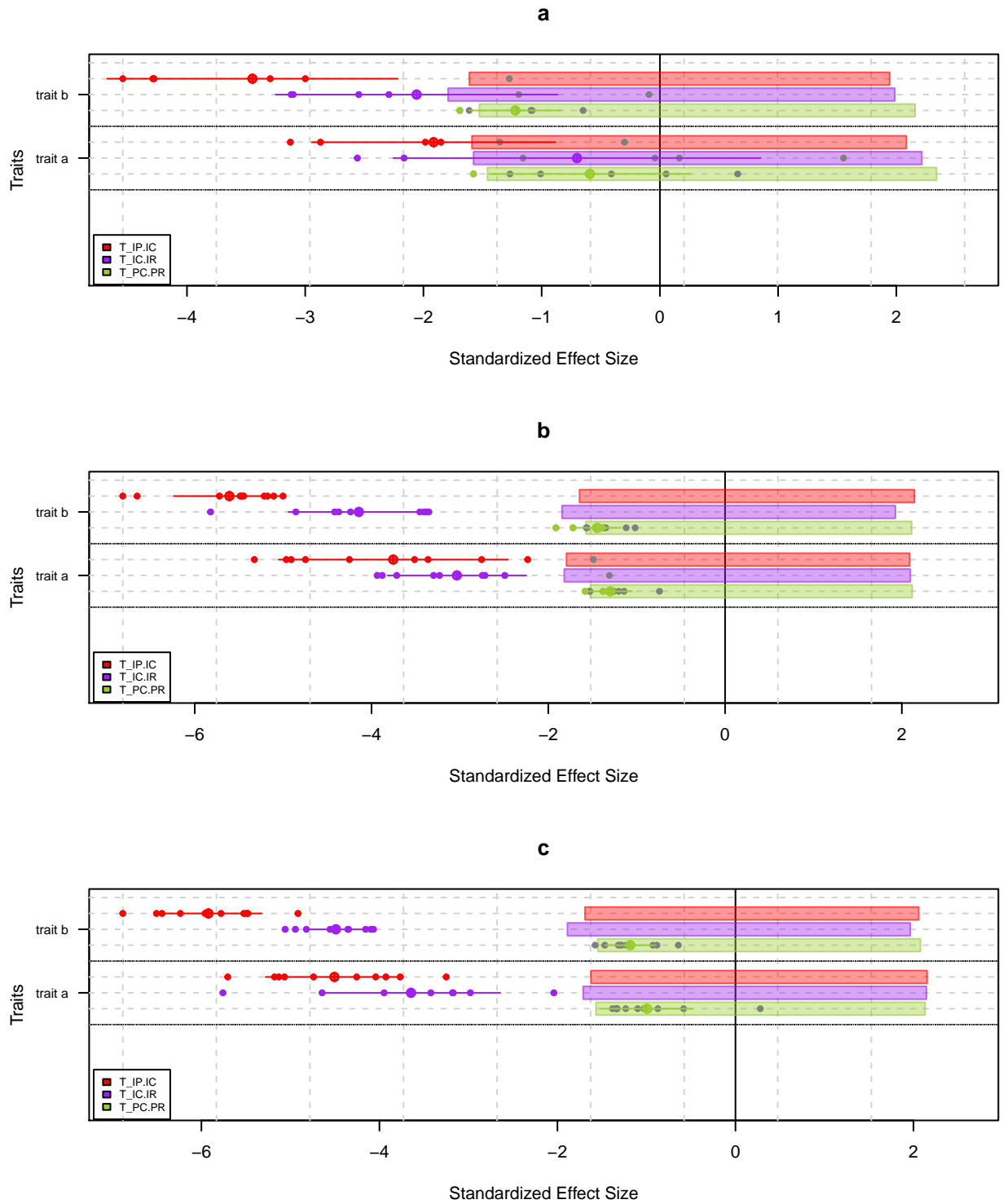
Figure 27: Results of T-statistics with both internal and external filter: $T_{IP/IC}$ in red, $T_{IC/IR}$ in purple and $T_{PC/PR}$ in green. (a) Lower strength of internal filtering 5 (b) higher strength of internal filtering.

```r
                    ses.listofindex(as.listofindex(x))
                    $index_1_3$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)
```

```r
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 filter we only present the type II error using the initial parameter ratio allowing a beta-error < 0.05.

Table 6: Initial parameter ratio 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)	1.515
		yes (global)	1.541
	$T_{IC.IR}$	no (local)	5.224
		yes (global)	5.492
	$T_{PC.PR}$	no (local)	133.91
		yes (global)	150.875
Trait b (uniform)	$T_{IP.IC}$	no (local)	0.413
		yes (global)	0.398
	$T_{IC.IR}$	no (local)	1.01
		yes (global)	1.023
	$T_{PC.PR}$	no (local)	44.917
		yes (global)	52.127

6   Test of variance decomposition functions

In addition to the `cati` vignette that test this function on the Darwin finch data, we test the behavior of the function `partvar` and `decompCTRE` by building two toy models associated to each functions.

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 draw in a normal distribution of mean 0 and standard error 1: (i) a value depending of the species the individual belong to, (ii) a value depending of the community the individual belong to and (iii) a independent value measuring the individual variance.

```
nperm <- npermut

sdlog = 1.5
#Ten communities named A, B, C ... J
com <- c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J")
#Twenty species sp1, sp2, sp3, ... sp20
sp <- paste("sp", seq(1:20), sep = "")

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

 # Draw communities using lognormal distribution of abondances
 # 100 individuals by populations
 ex.sp5 <- c()
 ex.com5 <- matrix(0, nrow = 10, ncol = 20)
 for(i in 1: 10){
 ex.com.interm <- table(sample(sp, size = 100, prob = rlnorm(20, 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 1:20){
 x1[ex.sp5 == levels(ex.sp5)[s]] <- rnorm(1)
 }

 ex.indplot5 <- as.factor(rep(com, 100))
 x2 <- c()
 x3 <- c()
 ex.indplot5 <- as.factor(ex.indplot5 )
 for(c in 1:10){
 x2[ex.indplot5 == levels(ex.indplot5 )[c]] <- rnorm(1)
 }
 X <- rnorm(1000) + 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 component of variation in individual traits distributions. We are waiting for 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 near 0.33. Consequently the function `partvar` don't 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 value of turnover strength by deleting individuals from the data set if there their species-components trait and their community-component trait are to different. To be more precise, when `turnover_strength = 1`, we delete all individual whose the square of the species value minus the community value is superior o one.

```
sdlog = 1.5
turnover_strength <- rep(c(0.01, 0.1, 0.2, 0.5, 1, 2, 3.5, 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
```

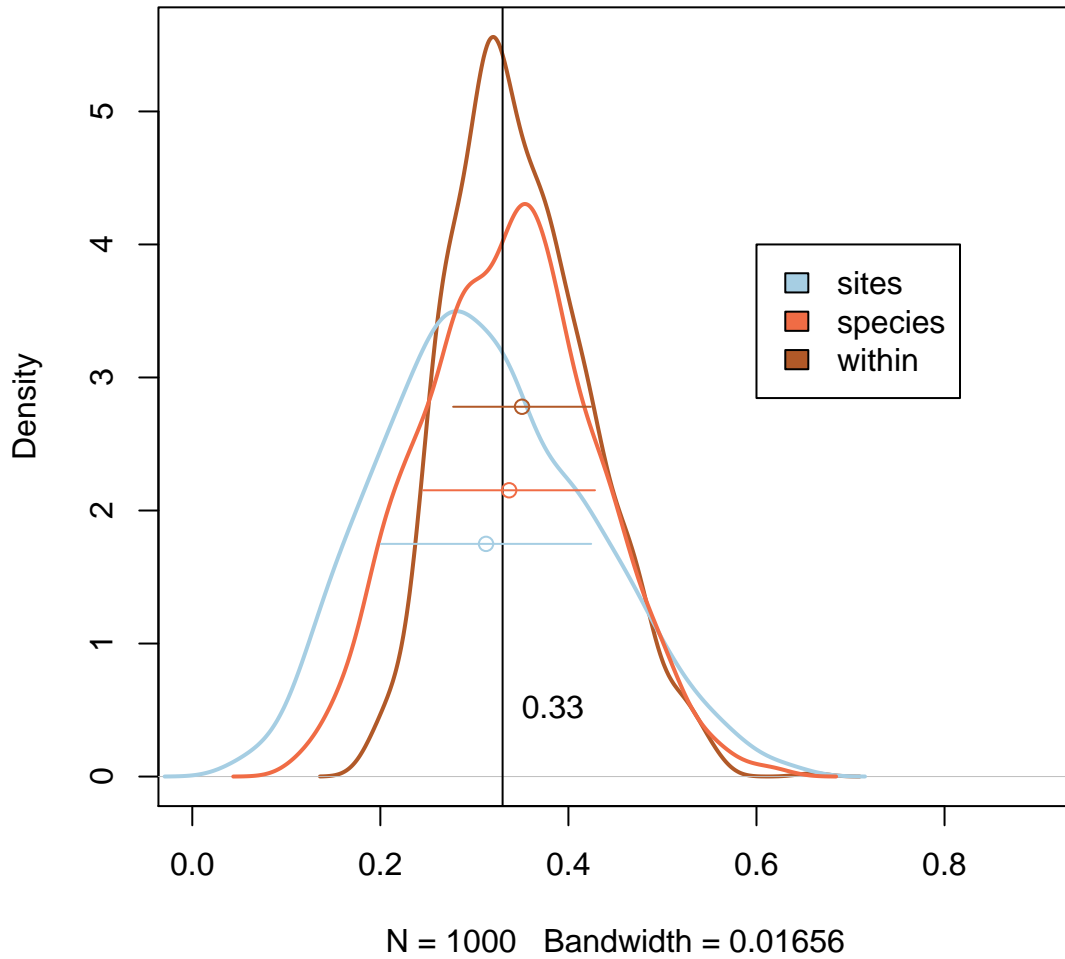**density.default(x = res.simu.partvar_toymodel[, 3])**

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

```
# Draw communities using lognormal distribution of abondances
# 100 individuals by populations
ex.sp5 <- c()
ex.com5 <- matrix(0, nrow = 10, ncol = 20)
for(i in 1: 10){
ex.com.interm <- table(sample(sp, size = 100, prob = rlnorm(20, 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 1:20){
x1[ex.sp5 == levels(ex.sp5)[s]] <- rnorm(1)
}

ex.indplot5 <- as.factor(rep(com, 100))
x2 <- c()
ex.indplot5 <- as.factor(ex.indplot5)
for(c in 1:10){
x2[ex.indplot5 == levels(ex.indplot5)[c]] <- rnorm(1)
}

X <- rnorm(1000) + x1 + x2

#We delete the individual whose species values and communities values are to 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 simulation
```

Now we can plot the relation 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]]$X2)
plot(res.decompCTRE_toymodel[[n]]$X2)
par(mfrow=c(1, 1))


res.simu.decompCTRE_toymodel <- unlist(lapply(res.decompCTRE_toymodel, function(x)
                                       x$X2$RelSumSq[1]))
plot(res.simu.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")
```

69
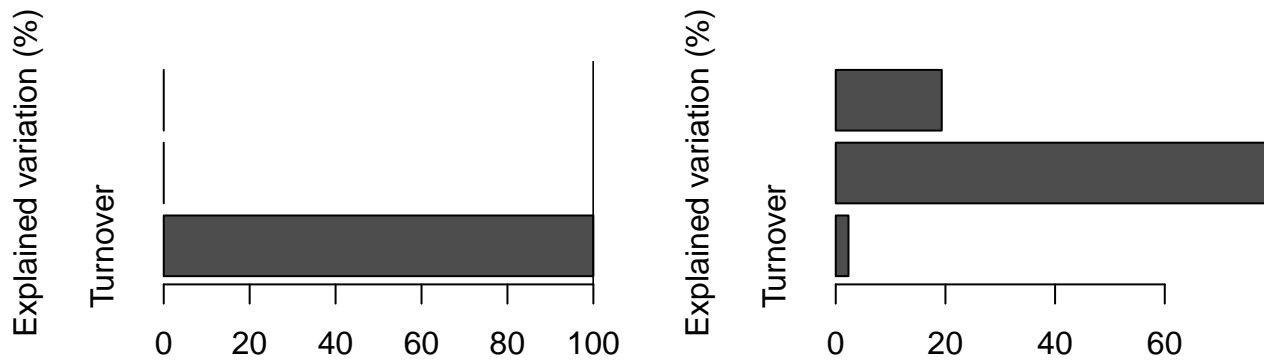
Figure 29: Decomposition of the variation in community trait composition for two contrasted cases: on the left almost all the variation is due to turnover wheras on the rigth case intraspecific variation explain most of the variation.
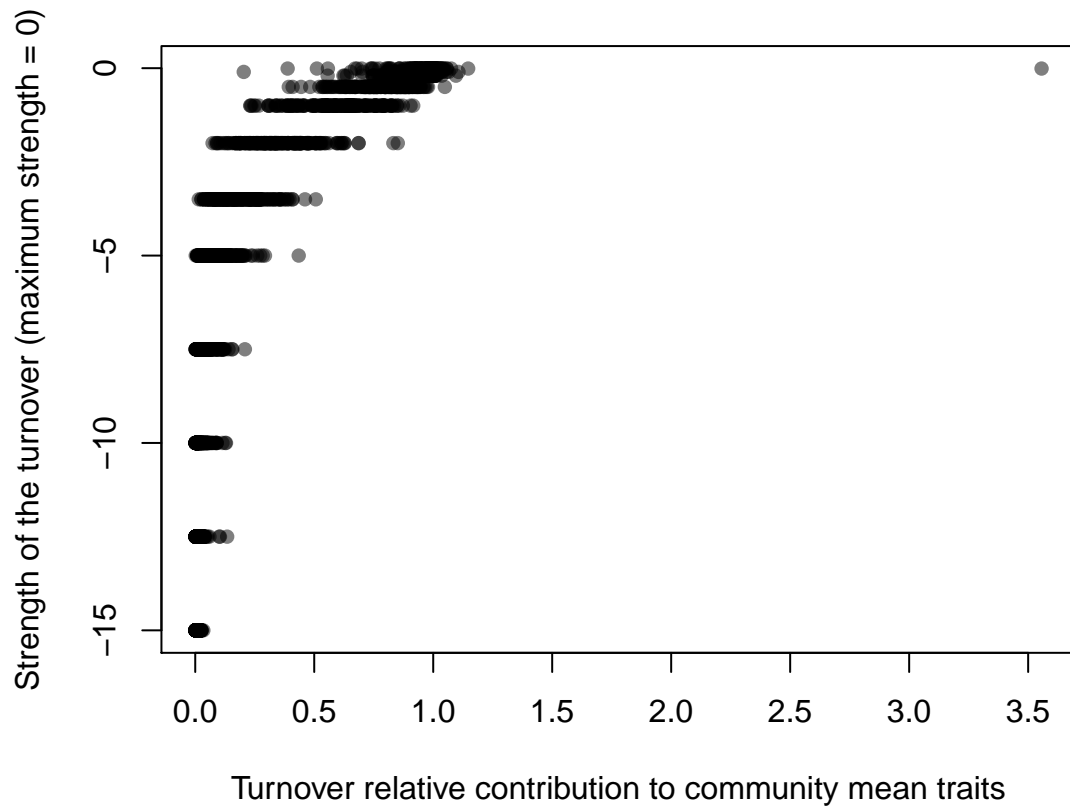
Figure 30: Modeled against obtained turnover contribution in community trait variance

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

7 Summary

7.1 Type I error

Table 2 summarize the alpha error for T-statistics. The three global [9] T-statistics are very robust regardless of the distribution of the trait (alpha error ranging from 0.001 to 0.001). Local [10] T-statistics present lower but nevertheless satifactory alpha-error with a maximum of 0.04035 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 powerfull (see the next section).

7.2 Type II error

Table 7: Summary of initial parameter ratio 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
	$T_{IP.IC}$	1.05		1.54
Trait a (normal)	$T_{IC.IR}$		1.07	5.49
	$T_{PC.PR}$		1.62	150.88
Traits	Indices	Initial parameter ratio with internal filtering	Initial parameter ratio with external filtering	Initial parameter ratio with internal and external filtering
	$T_{IP.IC}$	0.51		0.4
Trait b (uniform)	$T_{IC.IR}$		0.51	1.02
	$T_{PC.PR}$		0.78	52.13

Conclusion

References

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

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 Evolution 27: 244-252.

List of Figures

---

[9]one indice for all the dataset
[10]one indice for each community of the dataset

List of Tables