Illustrating package cati (Community Assembly by Traits: Individuals and beyond) using Darwin finches data

Adrien Taudiere*and Cyrille Violle

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

December 19, 2014

Abstract

1) The description of species by functional traits and the subsequent quantification of the functional structure of ecological communities have recently stimulated the field of community ecology. Patterns of trait distribution within communities are increasingly reliable indicator of community assembly processes. Myriad functional diversity indices and analytical tools – including the implementation of null models of community assembly – have been developed to detect and quantify different assembly process. With increased methodological complexities, such as the incorporation of intraspecific trait variation, the construction of a unified R package to test community assembly using functional traits, and its scale-dependency, is imperative and timely particularly to take into account the role of intraspecific variation in community ecology.

2) We introduce the R package cati, available on CRAN, specifically dedicated to community assembly analyses using functional traits. The package includes: (i) the use of single-trait and multi-trait indices to characterize the functional structure of communities; (ii) the partitioning of functional trait variation across spatial scales and organizational levels, noticeably to account for individual differences and their implications for community assembly and the maintenance of species coexistence; (iii) the implementation of flexible null models to detect and quantify environmental filters.

3) cati offers a comprehensive tool to detect and quantify assembly processes and can be applied to any kind of community (plant, animal, fungi. . .). Moreover, the possibility to import any type of ecological distances (notably phylogenetic and genetic distances) in cati allows a test of the response of different facets of biodiversity to assembly processes.

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

The up to date version of this tutorial is available here.

*adrien.taudiere@cefe.cnrs.fr

Contents

1 Introduction **3**

2 Getting started **3**
2.1 Installing the package `cati` . . . 3
2.2 Getting help . . . 3
2.3 Data presentation: Darwin finches in Galapagos Island . . . 4

3 Description of traits distributions **5**
3.1 Plot the kernel density of traits . . . 5

4 Decomposition of variances **13**
4.1 Decomposition of within/among species variances using rao diversity . . . 13
4.1.1 Multitraits analysis . . . 13
4.1.2 Unitraits analysis . . . 15
4.2 Decomposition of community trait response to environment into intraspecific trait variability, variability due to species turnover and their covariation. . . . 16
4.3 Decomposition of traits variances using nested factors . . . 19
4.4 Plot the relation between populational trait means and sites traits means. . . . 21

5 Test of community assembly **26**
5.1 Ratio of variances: T-statistics . . . 26
5.1.1 S3 methods for class Tstats . . . 27
5.1.2 Plot T-statistics correlations . . . 36
5.2 Others univariates or multivariates metrics: function `ComIndex` and `ComIndexMulti` . . . 41
5.2.1 S3 methods for class ComIndex and ComIndexMulti . . . 42
5.2.2 Plot Tstats and other uni/multivariates metrics together . . . 43
5.3 More information on multivariates index . . . 45

6 Others graphical functions **57**

7 Multivariate analysis of metrics **61**
Conclusion . . . 62
References . . . 63

8 Conclusion **63**

9 Acknowledgment **63**

1 Introduction

This vignette present the `cati` package for R (Community Assembly by Traits: Individuals and beyond) using Darwin finches data.

2 Getting started

2.1 Installing the package `cati`

Before going further, we shall make sure that `cati` is well installed on the computer. The current version of the package is 0.93. Make sure you have a recent version of R ($\geq$ 3.0.2) by typing:

```
R.version.string

## [1] "R version 3.1.1 (2014-07-10)"
```

Then, install `cati` with dependencies using:

```
#install.packages("cati", repos = "http://cran.us.r-project.org",                    dependen
library("cati")

## Loading required package:  nlme
## Loading required package:  ade4
## Loading required package:  ape
```

We can now load the package alongside other useful packages:

```
library("mice")

## Loading required package:  Rcpp
## Loading required package:  lattice
## mice 2.22 2014-06-10

library("hypervolume")

## Loading required package:  rgl
```

You can make sure that the right version of the package is installed using:

```
packageDescription("cati", fields = "Version")

## [1] "0.93"
```

`cati` version should read 0.93.

2.2 Getting help

To get help for a given function, use `?foo` where `foo` is the function of interest. For instance:

```
?Tstats
```

will open up the manpage of T-statistics function (`Tstats`). An 'example' section will shows how to use a function at the end of the manpage.

Note that you can also browse help pages as html pages, using:

```
help.start()
```

To go to the `cati` man page on Rstudio, click 'packages' in the lower right windows, then clik 'cati', and 'cati-package'.

2.3 Data presentation: Darwin finches in Galapagos Island

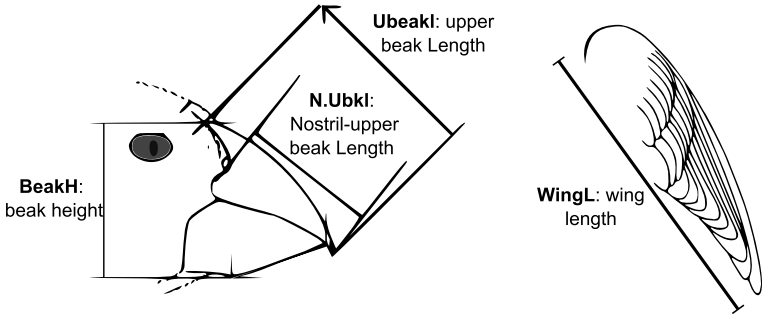
First we need to load the data.

```
data(finch.ind)

#Save default parameters
old.par<-par(no.readonly = TRUE)
```

Now we can see 3 objects: a traits matrix `traits.finch`, a vector of species names `sp.finch` and a vector of sites names `ind.plot`.

```
dim(traits.finch)
#the trait matrix contains 2513 individuals values for 4 traits
table(sp.finch)
#the species names vector contains
#2513 individuals belonging to 12 species
table(ind.plot.finch)
#the sites names vector contains
#2513 individuals belonging to 6 sites (Here 6 Island)
```

The four traits correspond to three beak traits and one wing trait.

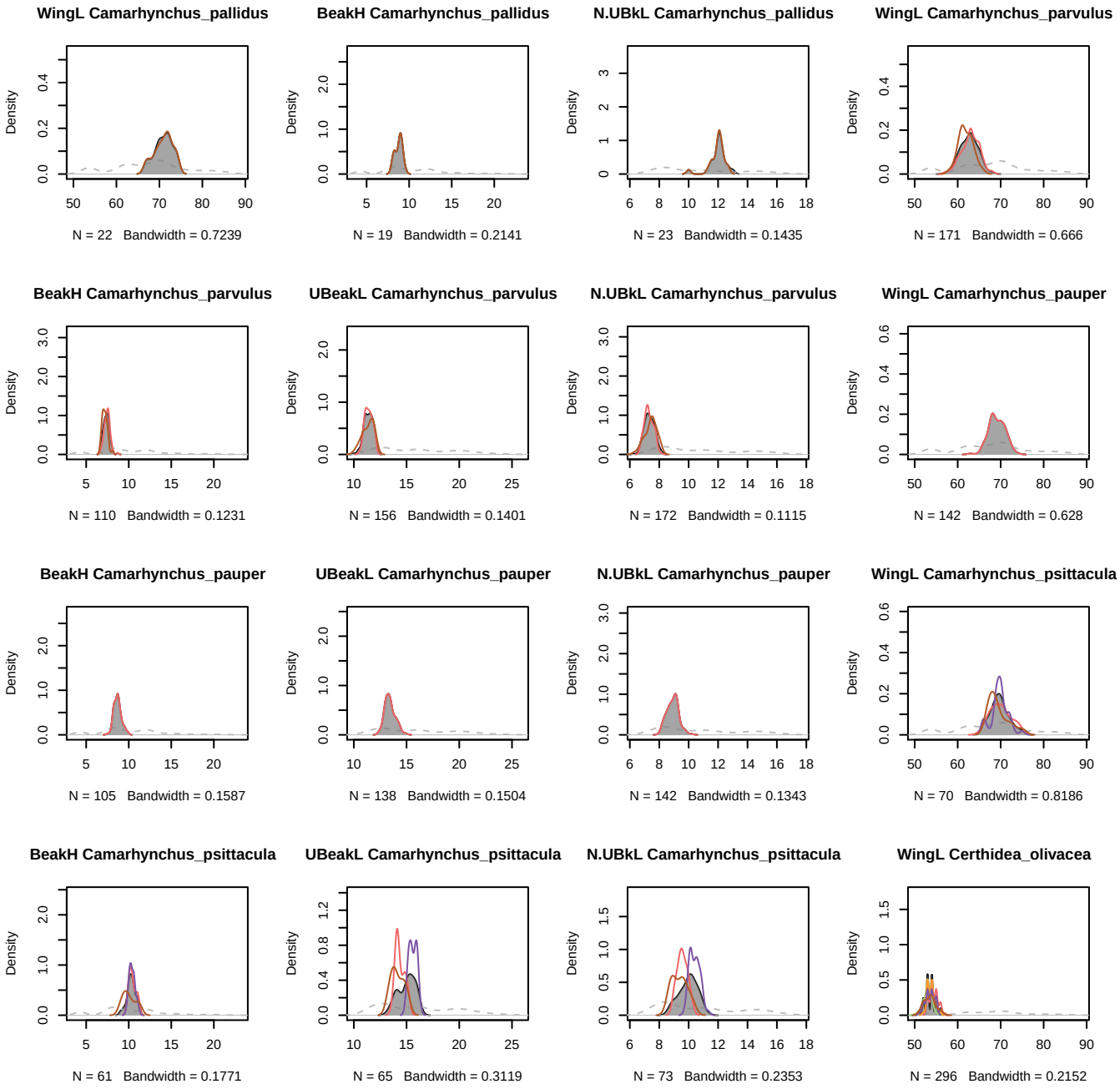


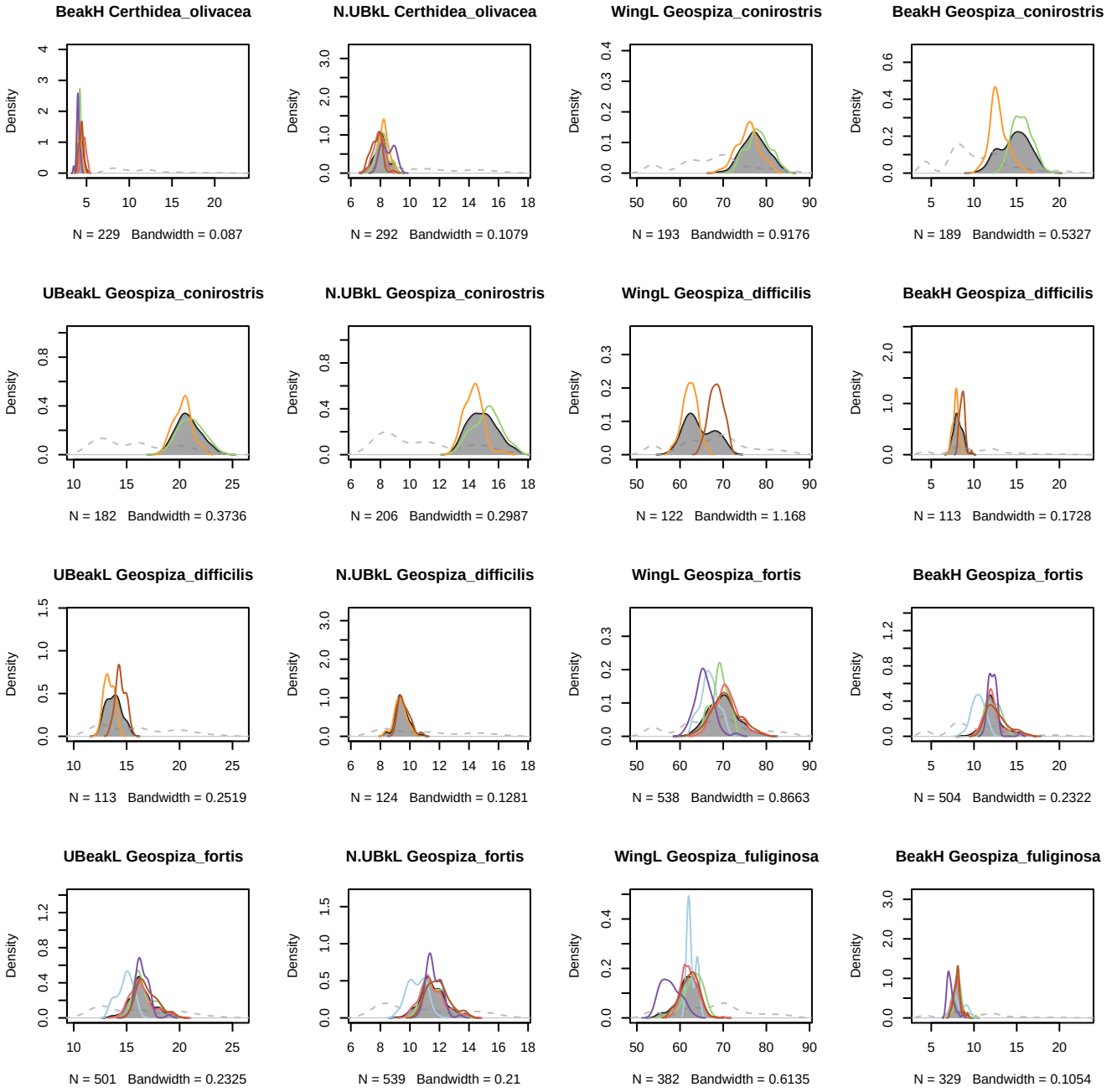
3 Description of traits distributions

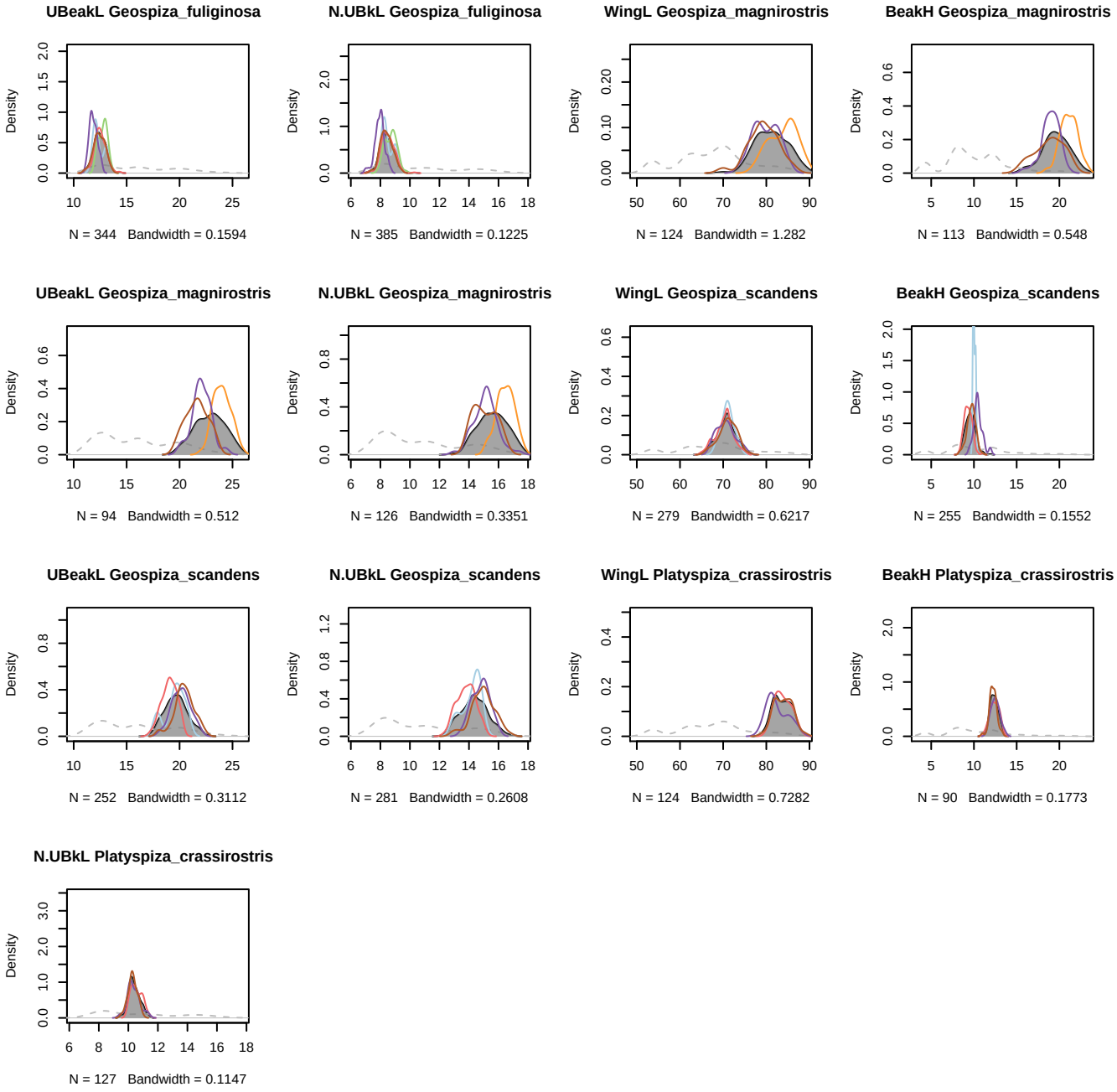
3.1 Plot the kernel density of traits

Plot the distribution of traits values for populations, species, sites and regional scales. First, let try the distribution for all populations of Darwin finches. In R, FALSE and TRUE can be written respectively `F` and `T`.

```
par(mfrow = c(4,4), cex = 0.5)
plotDistri(traits.finch, sp.finch, ind.plot.finch,
     ylim.cex = 3, plot.ask = F, multipanel = F, leg = F)
```



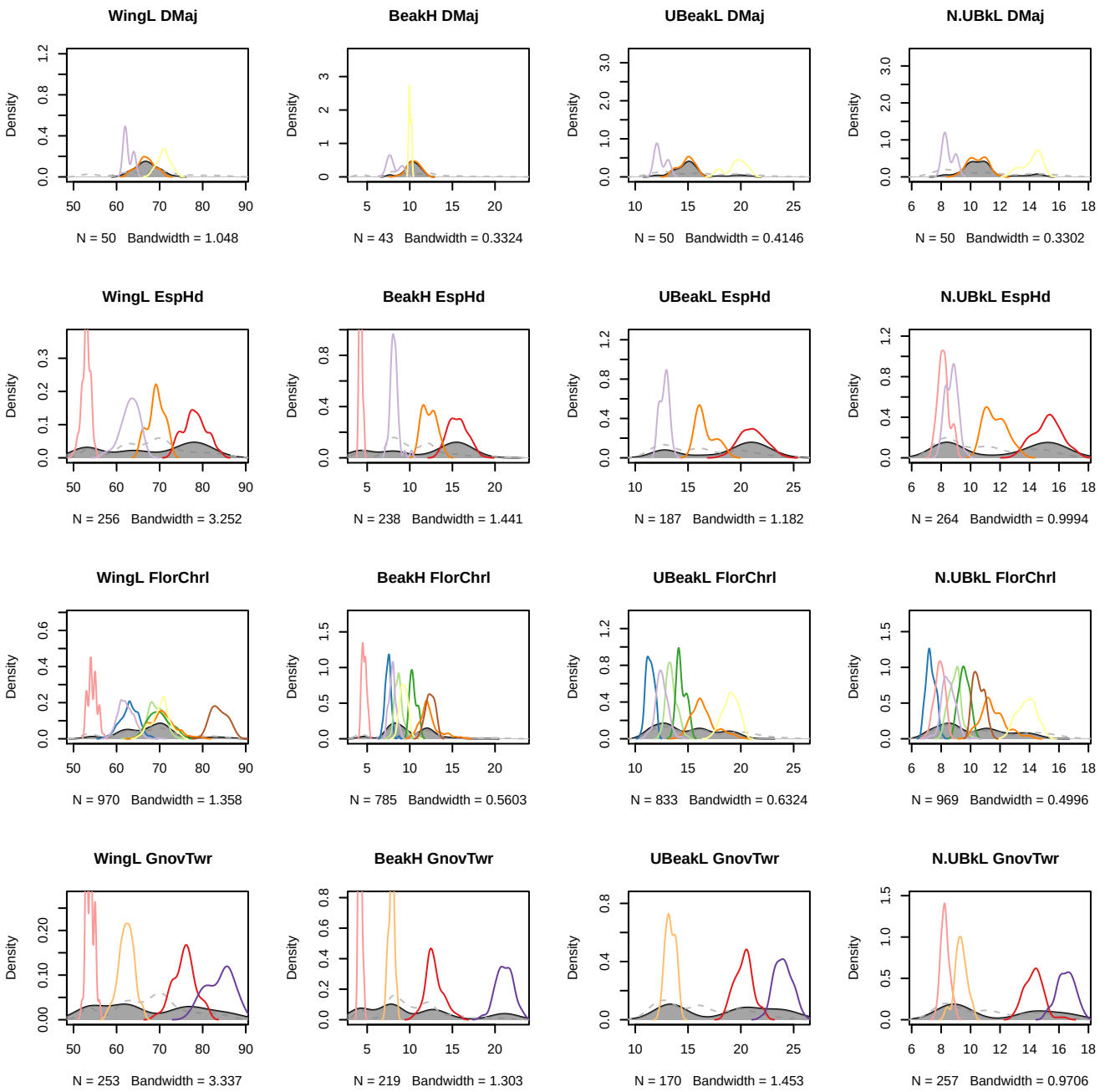

BeakH Certhidea_olivacea
N = 229 Bandwidth = 0.087
N.UBkL Certhidea_olivacea
N = 292 Bandwidth = 0.1079
WingL Geospiza_conirostris
N = 193 Bandwidth = 0.9176
BeakH Geospiza_conirostris
N = 189 Bandwidth = 0.5327
UBeakL Geospiza_conirostris
N = 182 Bandwidth = 0.3736
N.UBkL Geospiza_conirostris
N = 206 Bandwidth = 0.2987
WingL Geospiza_difficilis
N = 122 Bandwidth = 1.168
BeakH Geospiza_difficilis
N = 113 Bandwidth = 0.1728
UBeakL Geospiza_difficilis
N = 113 Bandwidth = 0.2519
N.UBkL Geospiza_difficilis
N = 124 Bandwidth = 0.1281
WingL Geospiza_fortis
N = 538 Bandwidth = 0.8663
BeakH Geospiza_fortis
N = 504 Bandwidth = 0.2322
UBeakL Geospiza_fortis
N = 501 Bandwidth = 0.2325
N.UBkL Geospiza_fortis
N = 539 Bandwidth = 0.21
WingL Geospiza_fuliginosa
N = 382 Bandwidth = 0.6135
BeakH Geospiza_fuliginosa
N = 329 Bandwidth = 0.1054
Density

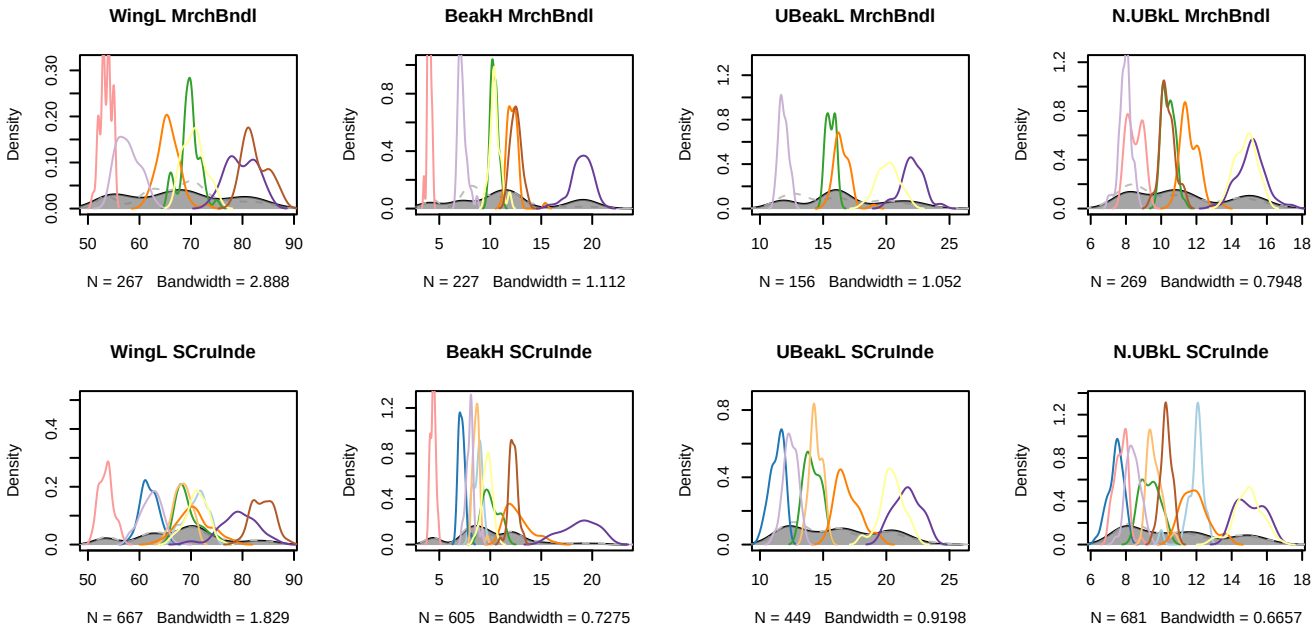


The argument `ylim.cex` define the magnification to be used for range of y axe. To understand the other argument, type `?plotDistri`.

Then we can inverse the second and the third arguments to plot the distribution for all finches species.

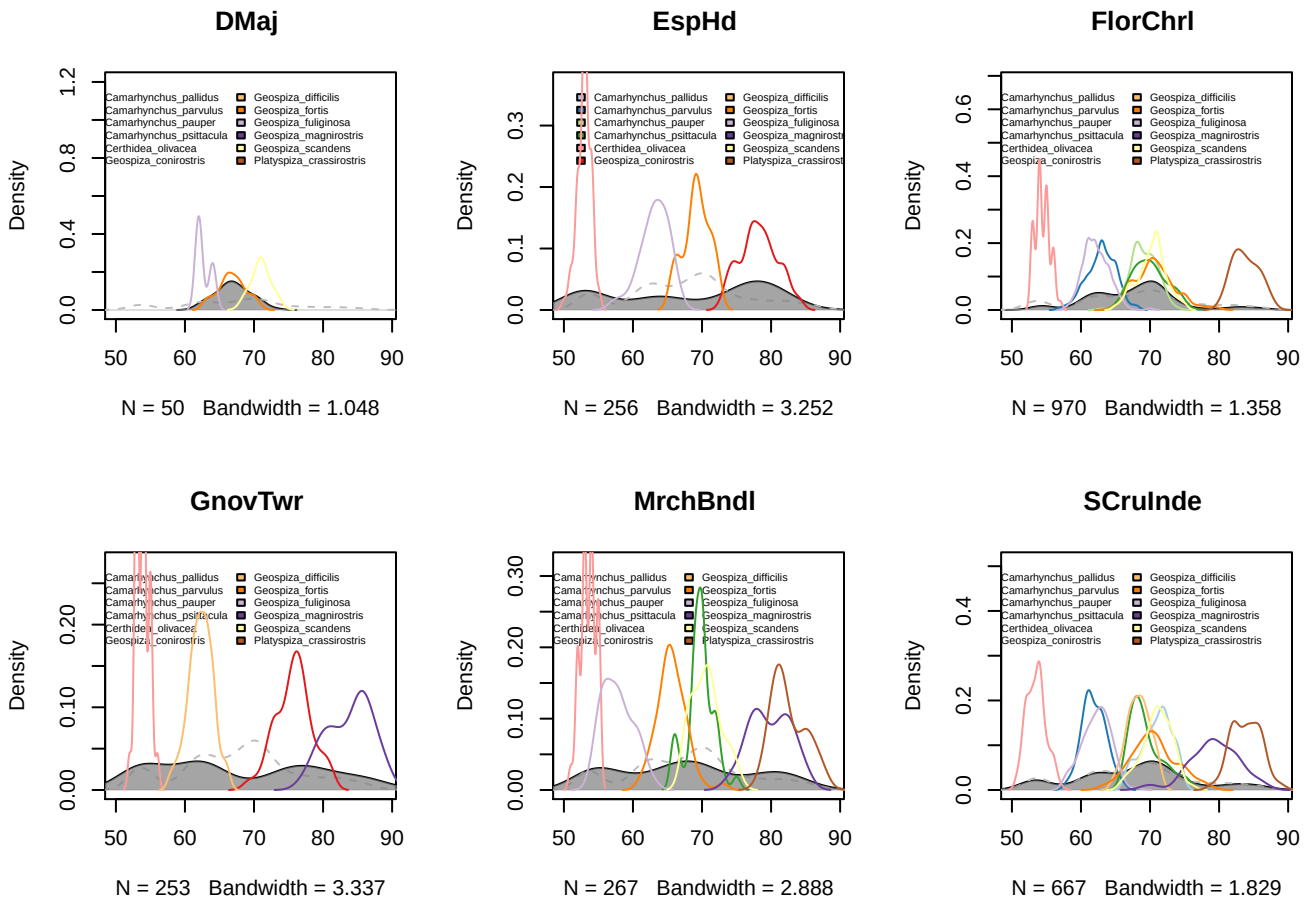
```
par(mfrow = c(4,4), cex = 0.5)
plotDistri(traits.finch, ind.plot.finch, sp.finch,
      ylim.cex = 8, plot.ask = F, multipanel = F, leg = F)
```





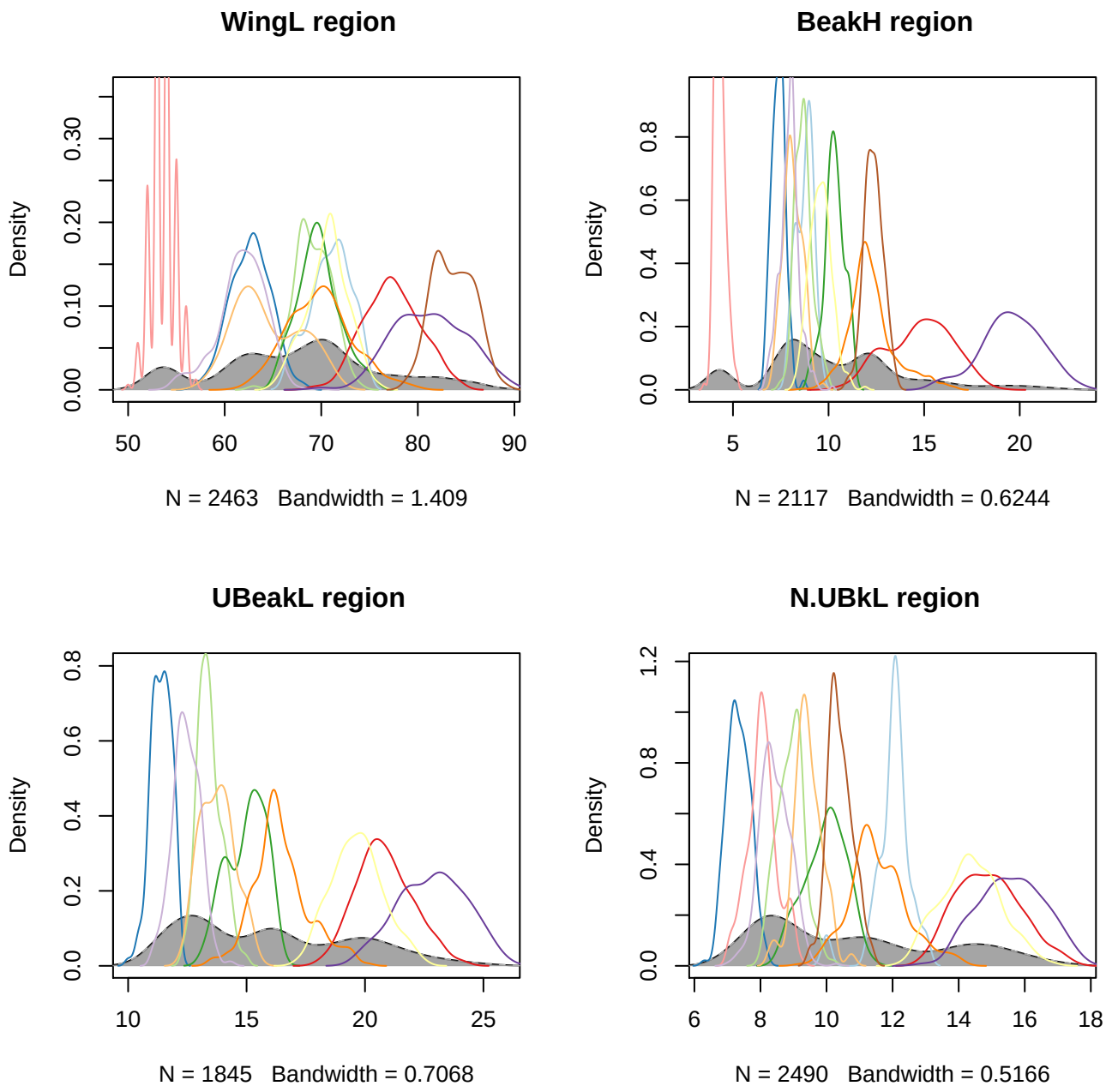
Now, see the result for only one trait with the legend (`leg = TRUE`). `cex.leg` specify the magnification of legend.

```
par(mfrow = c(2,3))
plotDistri(as.matrix(traits.finch[,1]), ind.plot.finch, sp.finch,
     ylim.cex = 8, plot.ask = F, multipanel = F, leg = T, cex.leg = 0.5)
```

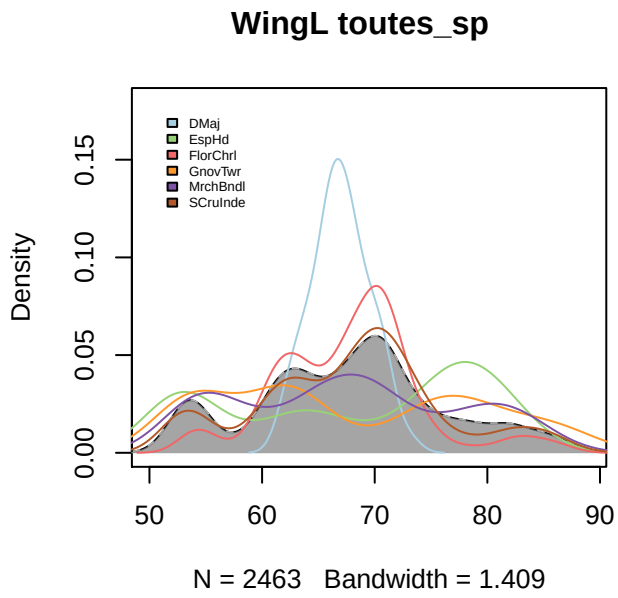


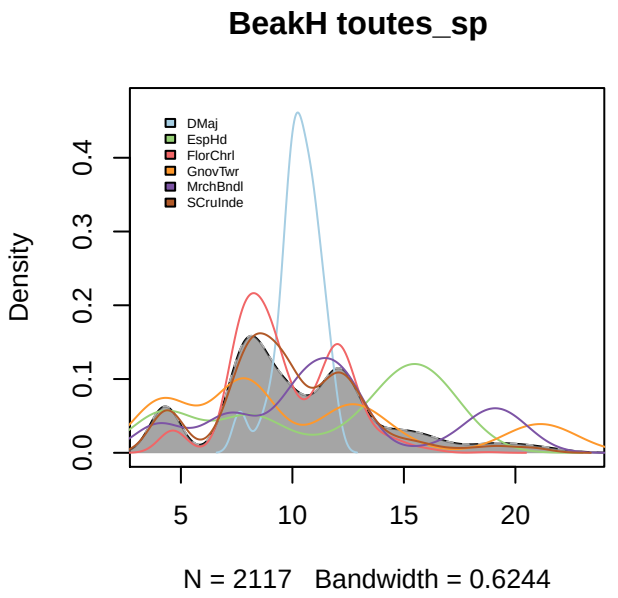
If we want to plot all the sites (regional distribution) or all the species: we can use the following code:

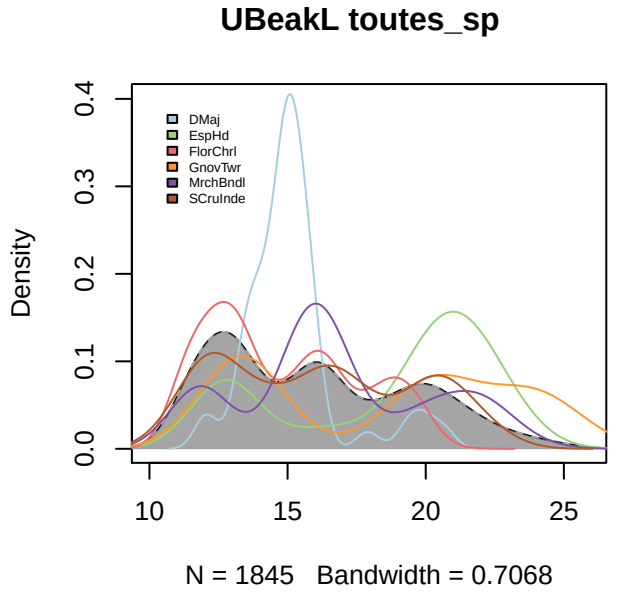
```
par(mfrow = c(4,4), cex = 0.5)
plotDistri(traits.finch, rep("region", times = dim(traits.finch)[1]),
     sp.finch, ylim.cex = 6, plot.ask = F, leg = F)
```

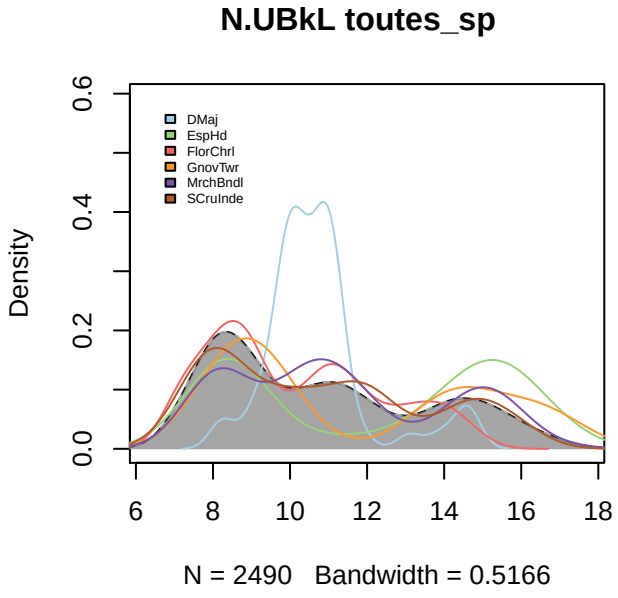



```
plotDistri(traits.finch, rep("toutes_sp", times = dim(traits.finch)[1]),
     ind.plot.finch, ylim.cex = 3, plot.ask = F, cex.leg = 0.5)
```









Now we can reset the default graphical parameter:

```
par(old.par)
```

4 Decomposition of variances

4.1 Decomposition of within/among species variances using rao diversity

The Rao function computes α, β and γ -components for taxonomic, functional and/or phylogenetic diversity with:

$$\gamma = mean(\alpha) + \beta$$

¡¡¡¡¡¡¡ HEAD Where: γ is the diversity of the regional pool, α is the diversity of the local community and β is the turn over between local communities; diversity is estimated using the Rao quadratic entropy indices. Note that this method uses the additive framework of diversity. See the paper of Jost in 2010 (partitioning diversity into independent alpha and beta components) for more details on the differences between additive and multiplicative partitioning of diversity.

=======

¡¡¡¡¡¡¡ HEAD Where: γ is the diversity of the regional pool, α is the diversity of the local community and β is the turn over between local communities; diversity is estimated using the Rao quadratic entropy indices. Note that this method uses the additive framework of diversity. See the paper of Jost in 2010 (partitioning diversity into independent alpha and beta components) for more details on the differences between additive and multiplicative partitioning of diversity.

=======

Where: γ is the diversity of the regional pool, α is the diversity of the local community and β is the turn over between local communities; diversity is estimated using the Rao quadratic entropy indices. Note that this method uses the additive framework of diversity. See the paper of Jost in 2010 (partitioning diversity into independent alpha and beta components) for more details on the differences between additive and multiplicative partitioning of diversity.

Where: $\gamma is the diversity of the regional pool, \alpha is the diversity of the local community and \beta is the turn over between 5f89dfbb4bbfa61805f8e116f94cee418c14cfee >>>>>>> d0b211331074d6462608d116c3317bf39e66e950$

Reference: de Bello, F., Lavorel, S., Albert, C.H., Thuiller, W., Grigulis, K., Dolezal, J., Janecek, S. and Leps, J. (2011) Quantifying the relevance of intraspecific trait variability for functional diversity. Methods in Ecology and Evolution, 2, 163-174.

4.1.1 Multitraits analysis

First, rao diversity can be calculated on the functional space (i.e. considering all traits together).

```
#create individuals community matrix
comm<-t(table(ind.plot.finch,1:length(ind.plot.finch)))
#create species community matrix
comm.sp<-table(sp.finch, ind.plot.finch)
class(comm.sp)<-"matrix"

traits.finch.sp<-apply( apply(traits.finch, 2, scale ), 2,
            function(x) tapply(x, sp.finch, mean, na.rm = T))

mat.dist<-as.matrix(dist(traits.finch.sp))^2

res.rao<-RaoRel(sample = as.matrix(comm.sp),
        dfunc = mat.dist, dphyl = NULL,
```

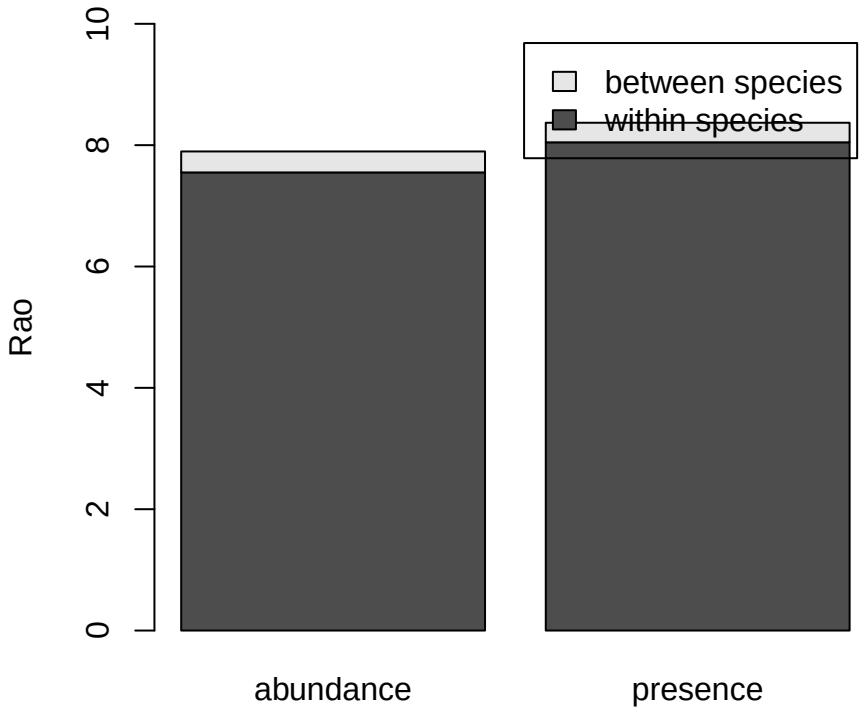
```
        weight = F, Jost = F, structure = NULL)

witRao<-res.rao$FD$Mean_Alpha #overall within species variance
betRao<-res.rao$FD$Beta_add  #between species variance
totRao<-res.rao$FD$Gamma    #the total variance

#Check that the total variance is equal
#to between species + within species variances
witRao+betRao

## [1] 8.37

totRao

## [1] 8.37
```

Now let's take the abundance into account to calculate Rao diversity.

```
res.rao.w<-RaoRel(sample = as.matrix(comm.sp),
         dfunc = mat.dist, dphyl = NULL,
         weight = T, Jost = F, structure = NULL)

witRao.w<-res.rao.w$FD$Mean_Alpha #overall within species variance
betRao.w<-res.rao.w$FD$Beta_add  #between species variance
totRao.w<-res.rao.w$FD$Gamma    #the total variance

witRao.w

## [1] 7.551

betRao.w

## [1] 0.3458
```

Plot the results.

```
barplot(cbind(c(witRao.w, betRao.w), c(witRao, betRao)),
    names.arg = c("abundance" ,"presence"),
    legend.text = c("within species", "between species"),
    ylab = "Rao", ylim = c(0,10))
```



4.1.2 Unitraits analysis

We can also do this analysis for each trait separately. We need to replace (or exclude) NA values. For this example, we use the package `mice` to complete the data.

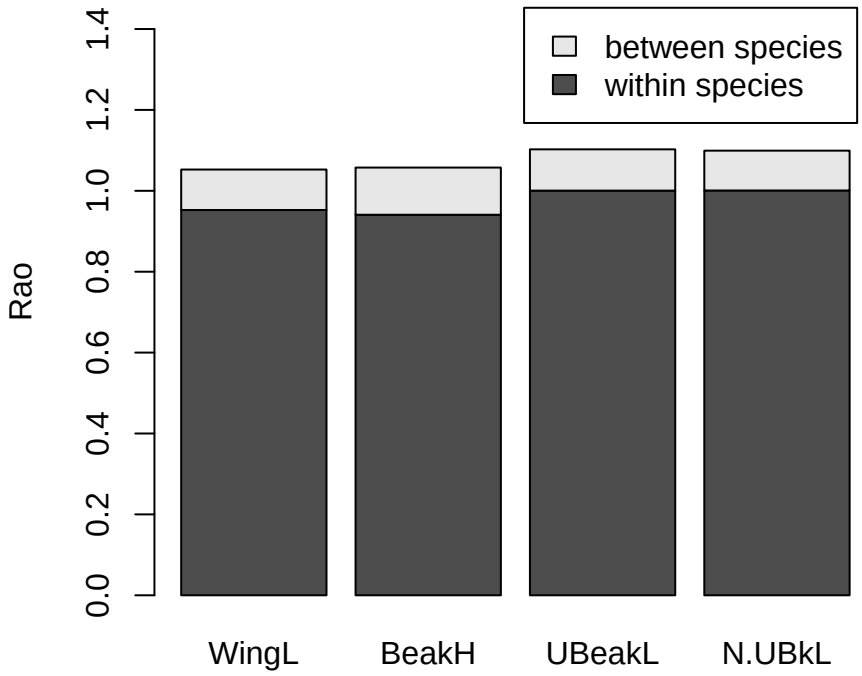
```
comm<-t(table(ind.plot.finch,1:length(ind.plot.finch)))

require(mice)
traits = traits.finch
mice<-mice(traits.finch)
traits.finch.mice<-complete(mice)
```

```
#Calculate the mean traits value by population using the mice dataset
traits.finch.mice.sp<-apply(apply(traits.finch.mice, 2, scale ), 2,
              function(x) tapply(x, sp.finch, mean, na.rm = T))

trait.rao.w<-list()
witRao.w.bytrait<-c()
betRao.w.bytrait<-c()
for(t in 1 : 4){
 trait.rao.w[[t]]<-RaoRel(sample = as.matrix(comm.sp),
              dfunc = dist(traits.finch.mice.sp[,t]),
```

```
              dphyl = NULL, weight = T, Jost = F, structure = NULL)
 witRao.w.bytrait<-c(witRao.w.bytrait, trait.rao.w[[t]]$FD$Mean_Alpha)
 betRao.w.bytrait<-c(betRao.w.bytrait, trait.rao.w[[t]]$FD$Beta_add)
}
```

Plot the results by traits.

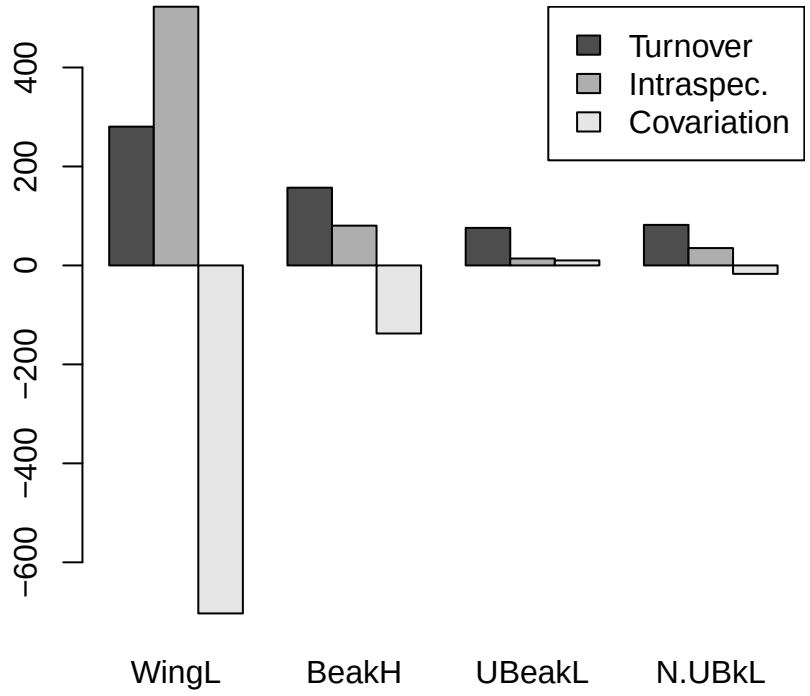
```
barplot(t(cbind( witRao.w.bytrait, betRao.w.bytrait)),
    names.arg = colnames(traits.finch),
    legend.text = c("within species", "between species"),
    ylab = "Rao", ylim = c(0,1.5))
```



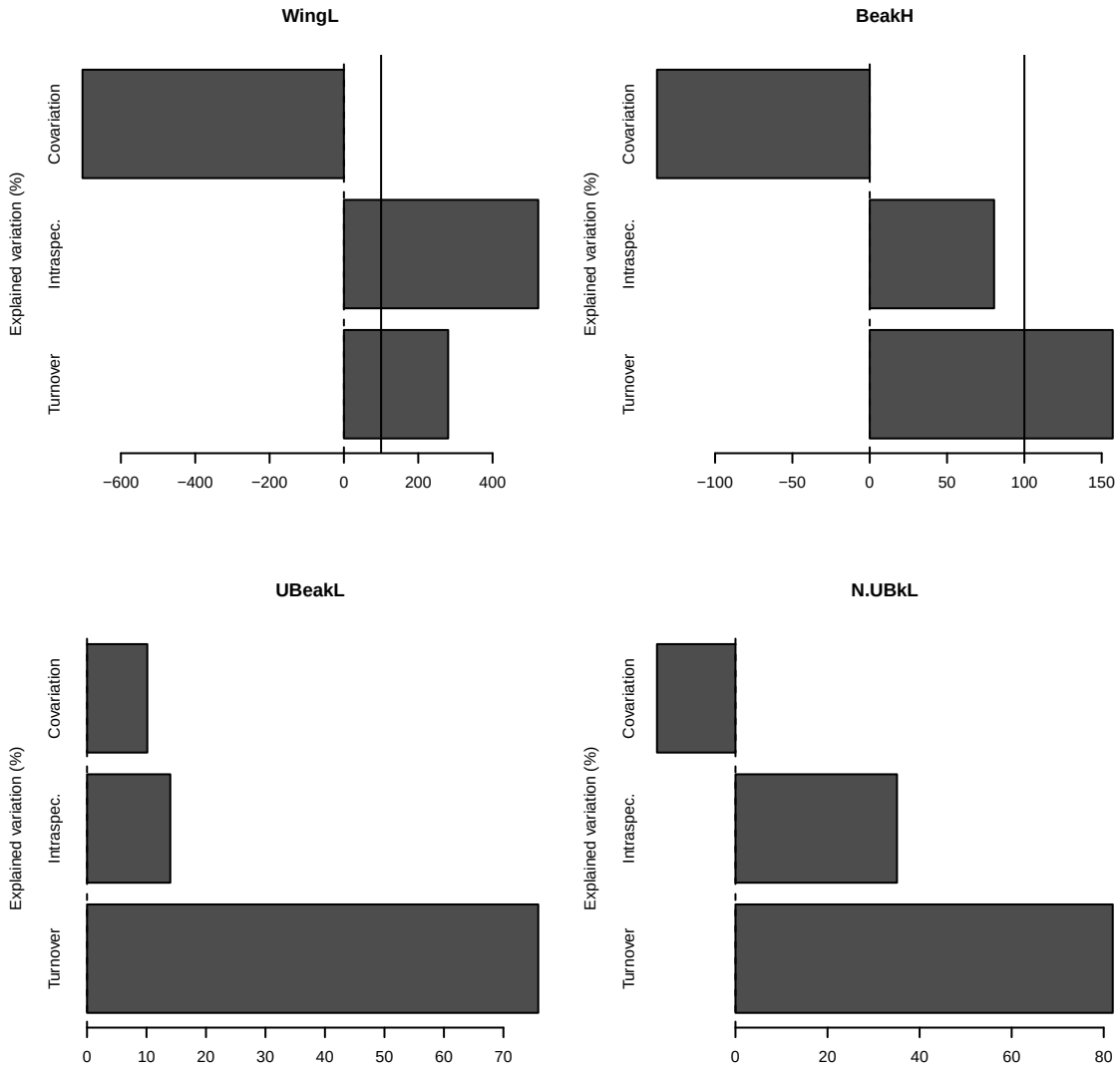
4.2 Decomposition of community trait response to environment into intraspecific trait variability, variability due to species turnover and their covariation.

Reference: Leps, J., de Bello, F., Smilauer, P. and Dolezal, J. (2011) Community trait response to environment: disentangling species turnover vs intraspecific trait variability effects. Ecography, 34, 856-863.

```
res.decomp<-decompCTRE(traits = traits.finch, sp = sp.finch,
              ind.plot = ind.plot.finch, print = FALSE)

barplot(res.decomp)
```




```
par(mfrow = c(2,2))
barplot(res.decomp, resume = F)
```




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

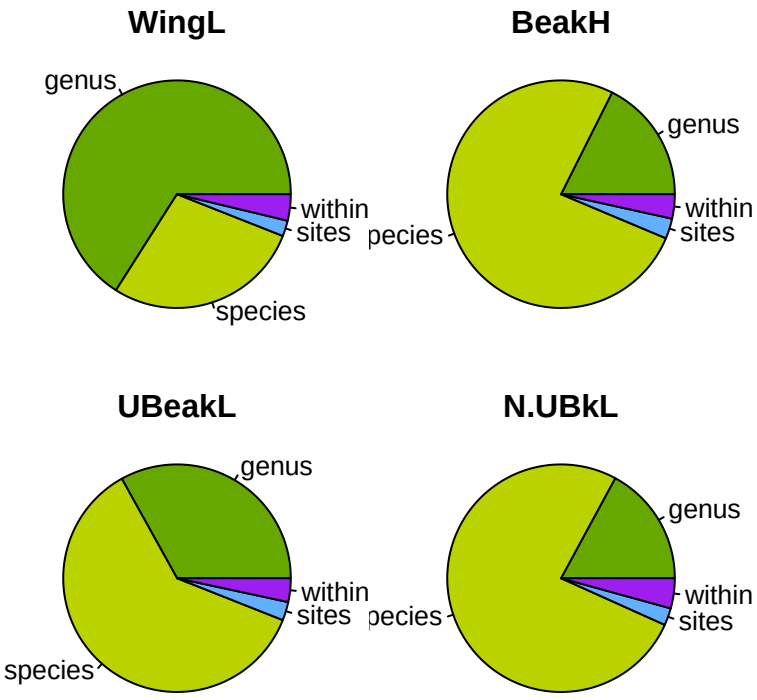
4.3 Decomposition of traits variances using nested factors

Variance partitioning across nested scales using the decomposition of variance on restricted maximum likelihood (REML) method (lme function).

Reference: Messier, J., McGill, B. and Lechowicz, M. (2010) How do traits vary across ecological scales? A case for trait-based ecology. Ecology Letters, 13, 838-848.

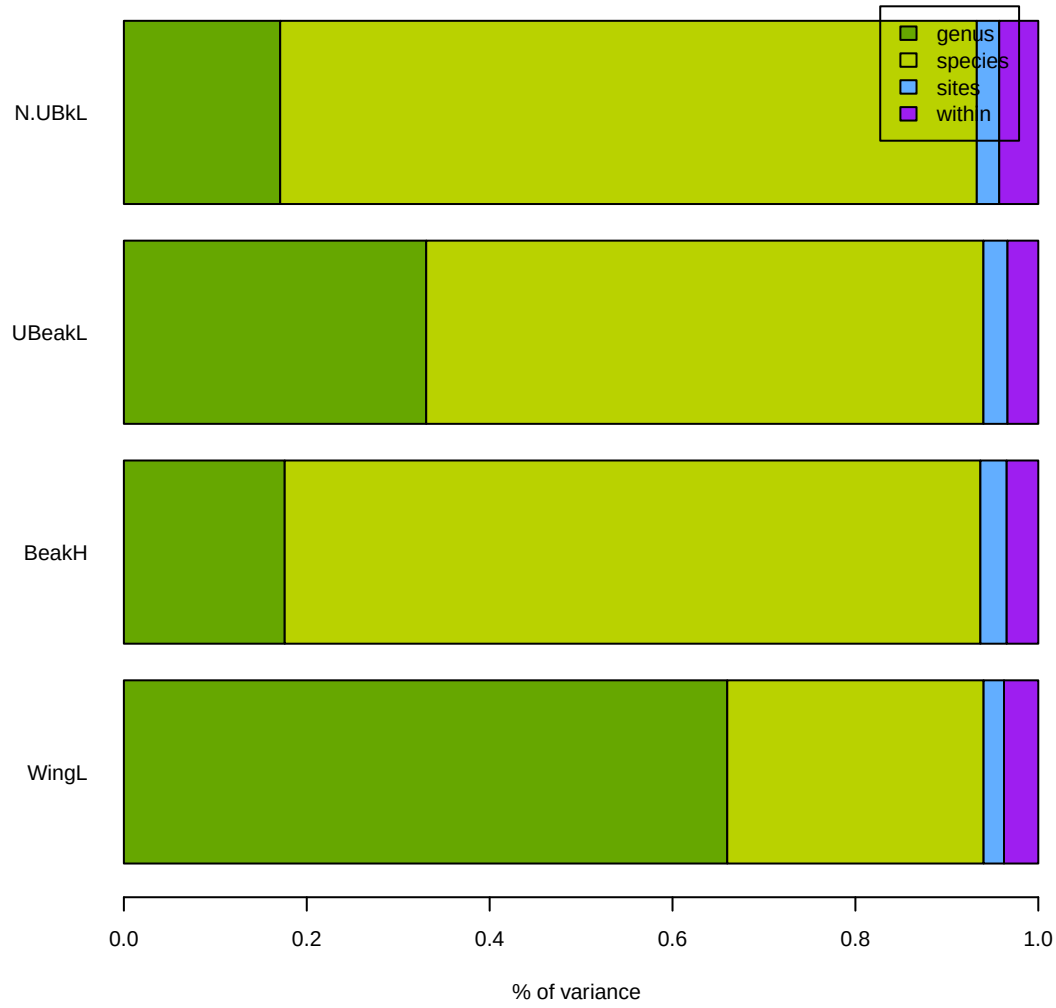
```
vec<- seq(1,length(sp.finch)*2, by = 2)
genus<-as.vector(unlist(strsplit(as.vector(sp.finch),"_"))[vec])
fact<-cbind(genus = as.factor(genus),
      species = as.factor(as.vector(sp.finch)),
      sites = as.factor(as.vector(ind.plot.finch)))

res.partvar.finch<-partvar(traits = traits.finch, factors = fact)

res.partvar.finch
```

```
par(mfrow = c(2,2), mai = c(0.2,0.2,0.2,0.2)) #save graphical parameters
colors<-c(rgb(102,167,0, maxColorValue = 255),
     rgb(185,210,0, maxColorValue = 255),
     rgb(98,174,255, maxColorValue = 255),
     rgb(158,30,240, maxColorValue = 255))

piePartvar(res.partvar.finch, col = colors)
```



```
par(old.par) #reset old graphical parameters
```

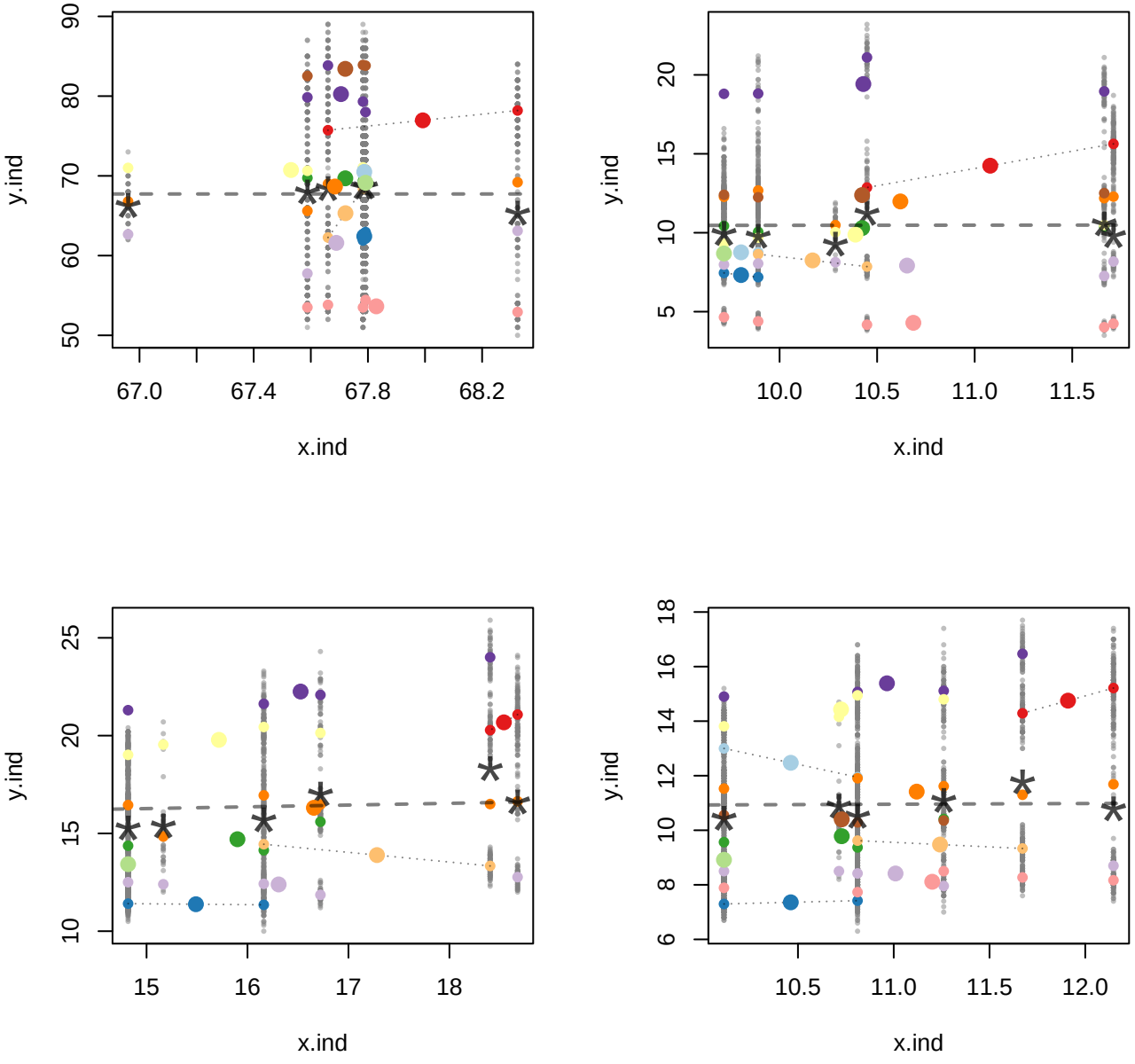
```
barPartvar(res.partvar.finch, col = colors, leg = TRUE)
```



4.4 Plot the relation between populational trait means and sites traits means.

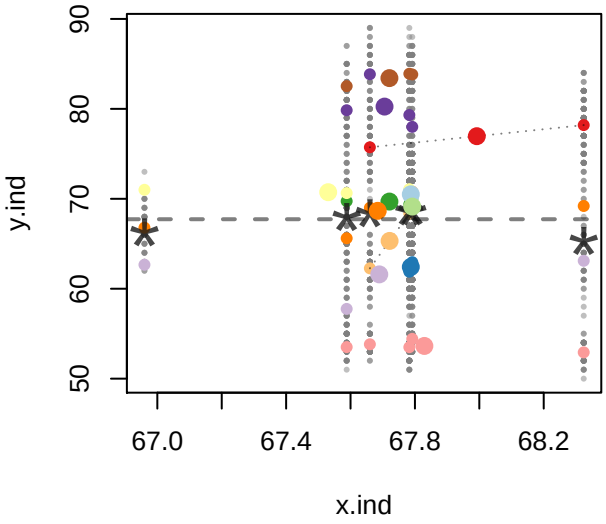
For an example of utilisation see: Cornwell, W.K. and Ackerly, D.D., 2009. Community assembly and shifts in plant trait distributions across an environmental gradient in coastal California. Ecological Monographs, 79, 109-126.

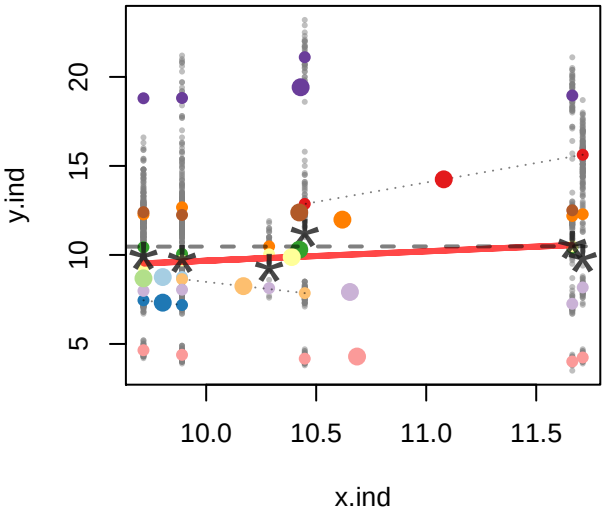
```
plotSpPop(traits.finch, ind.plot.finch, sp.finch, silent = TRUE)
```

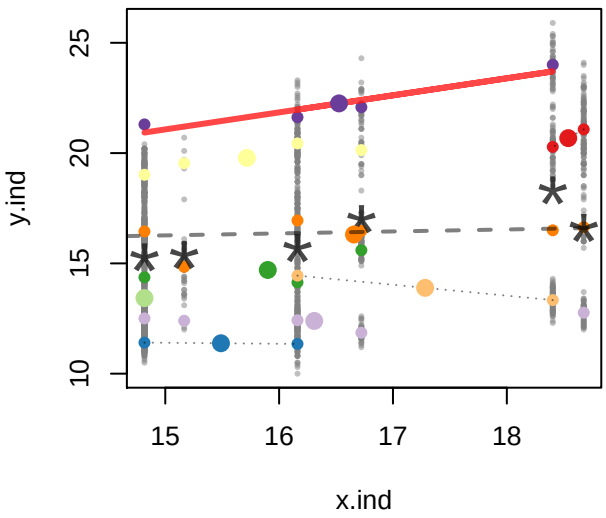


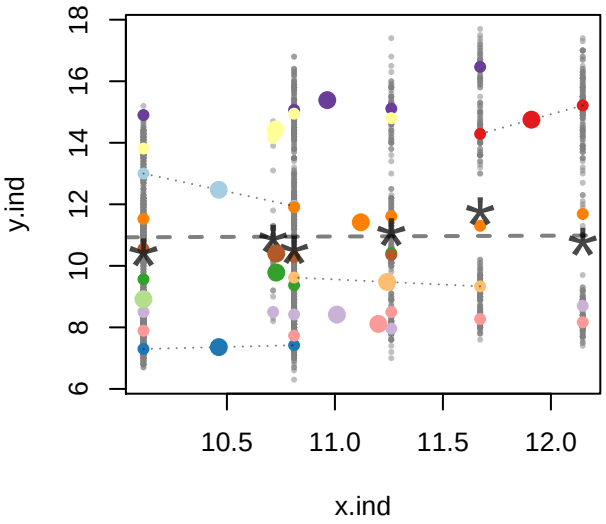
If we change the value of two arguments we can see some significant relationships. Here let's try a more permissive threshold: alpha = 10% instead of 5% (`p.val`) and define a lower minimum of values to represent significance fixed to 3 instead of 10 by default (`min.ind.signif`).

```
plotSpPop(traits.finch, ind.plot.finch, sp.finch,
      p.val = 0.1, min.ind.signif = 3, silent = TRUE)
```



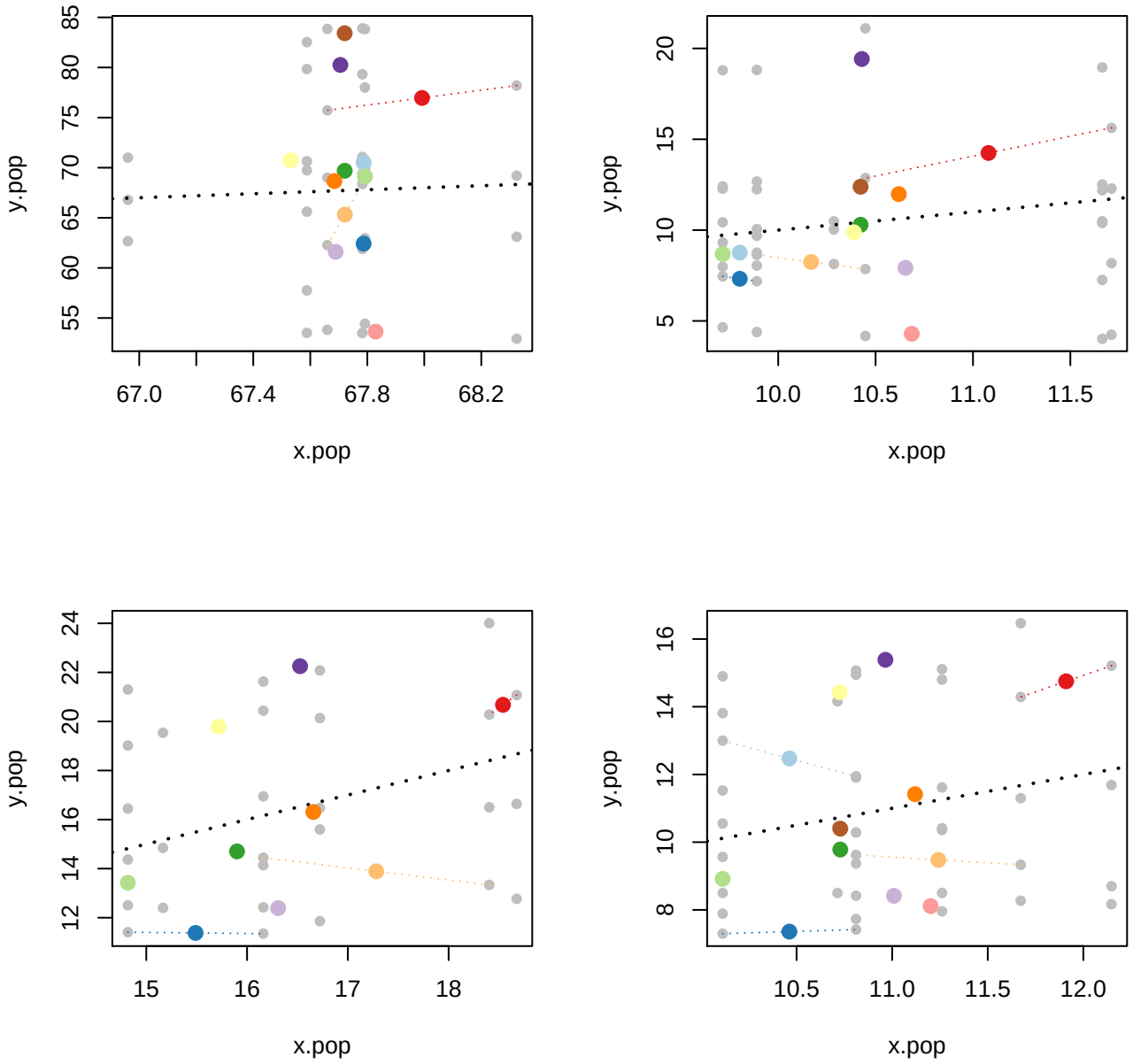






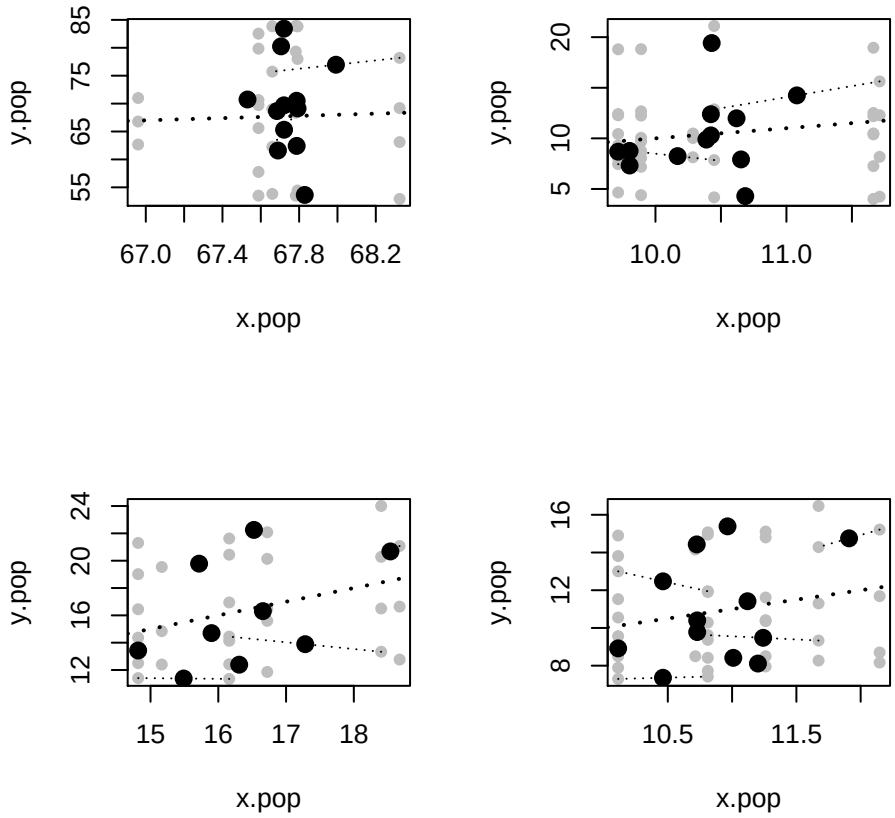
For a more simple figure, add the option resume = TRUE.

```
plotSpPop(traits.finch, ind.plot.finch, sp.finch,
      silent = TRUE, resume = TRUE, col.pop = "grey")
```



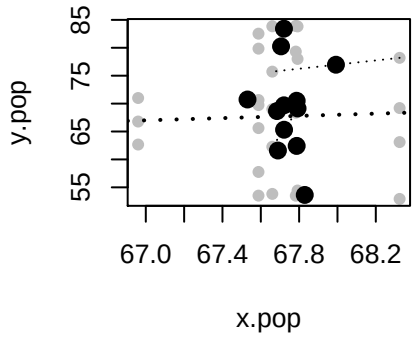
If you are fed up with colors, try this:

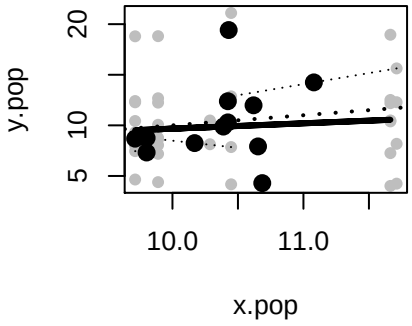
```
plotSpPop(traits.finch, ind.plot.finch, sp.finch,
      silent = TRUE, resume = TRUE, col.pop = "grey", col.sp = "black")
```

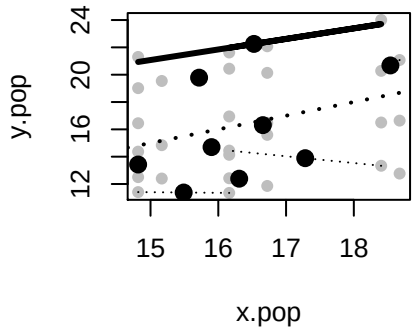


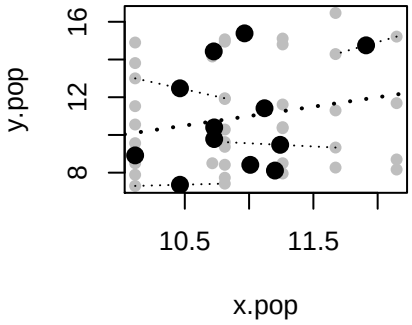
Again if we change the value of the threshold (`p.val` = 0.1 and `min.ind.signif` = 3) we can see some significant relationships.

```
plotSpPop(traits.finch, ind.plot.finch, sp.finch,
      silent = TRUE, resume = TRUE, col.pop = "grey", col.sp = "black",
      p.val = 0.1, min.ind.signif = 3)
```


y.pop
55
65
75
85
67.0
67.4
67.8
68.2
x.pop


y.pop
5
10
20
10.0
11.0
x.pop


y.pop
12
16
20
24
15
16
17
18
x.pop


y.pop
8
12
16
10.5
11.5
x.pop

5 Test of community assembly

5.1 Ratio of variances: T-statistics

The function `Tstat` computes observed T-statistics (T for Traits; Violle et al (2012)) as three ratios of variance, namely $T_{IP/IC}$, $T_{IC/IR}$ and $T_{PC/PR}$. This function can also return the distribution of these three statistics under the three associated null models (respectively **local**, **regional.ind** and **regional.pop**).

Reference: Violle, C., Enquist, B.J., McGill, B.J., Jiang, L., Albert, C., Hulshof, C., Jung, V. and Messier, J. (2012) The return of the variance: intraspecific variability in community ecology. Trends in Ecology and Evolution, 27, 244-252.

```
res.finch<-Tstats(traits.finch, ind.plot = ind.plot.finch, sp = sp.finch,
         nperm = 9, print = FALSE)
res.finch

##  ##################
##  # T-statistiques #
##  ##################
## class: Tstats
## $call: Tstats(traits = traits.finch, ind.plot = ind.plot.finch, sp = sp.finch,
##     nperm = 9, printprogress = FALSE)
##
## ###############
## $Tstats: list of observed and null T-statistics
##
## Observed values
##  $T_IP.IC: ratio of within-population variance to total within-community variance
##  $T_IC.IR: community-wide variance relative to the total variance in the regional pool
##  $T_PC.PR: inter-community variance relative to the total variance in the regional pool
##
## Null values, number of permutation: 9
##  $T_IP.IC_nm: distribution of T_IP.IC value under the null model local
##  $T_IC.IR_nm: distribution of T_IC.IR value under the null model regional.ind
##  $T_PC.PR_nm: distribution of T_PC.PR value under the null model regional.pop
##
## ###############
## $variances: list of observed and null variances
##
## ###############
## data used
##   data      class      dim
## 1 $traits   data.frame 2513,4
## 2 $ind.plot factor     2513
## 3 $sp       factor     2513
##   content
## 1 traits data
## 2 name of the plot in which the individual is
```

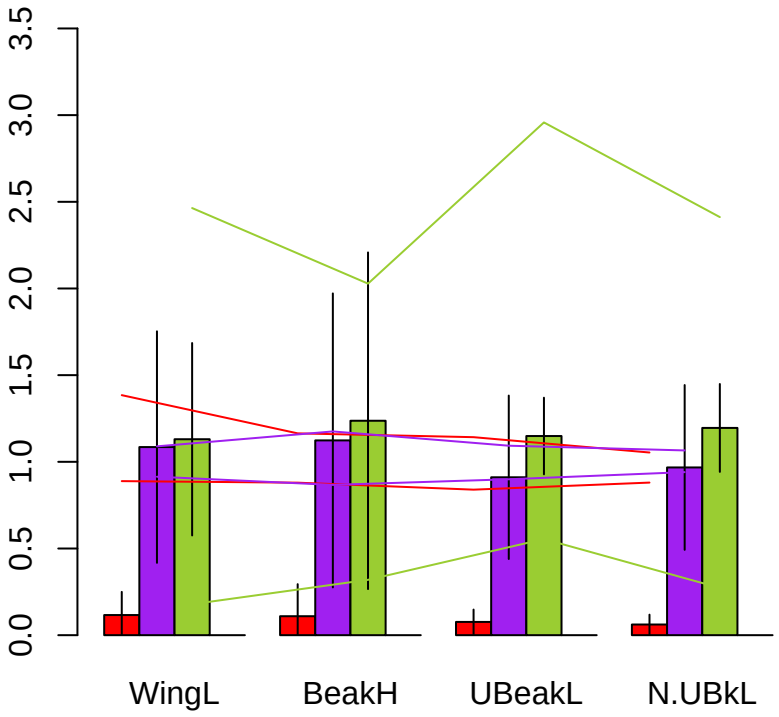
```
## 3 groups (e.g. species) which the individual belong to
##
## ###############
## others
##  $namestraits: 4 traits
## [1] "WingL"  "BeakH"  "UBeakL" "N.UBkL"
##
##  $sites_richness:
##  ind.plot
##     DMaj    EspHd FlorChrl  GnovTwr MrchBndl SCruInde
##       50      267      981      258      270      687
```

5.1.1 S3 methods for class Tstats

Tstats class is associated to S3 methods plot, barplot, print and summary. We have already used the print function in the above script line. Now, how to plot the result of the function `Tstats`?

We can represent observed values thanks to the `barplot` function.

```
barplot(res.finch, ylim = c(0,3.5))
```



One can be more interested in the significance and the effect size available thanks to null model. In that case, the standardized effect size can be easily plot. Note that the function `ses` can be use directly to calculate standardized effect size without plotting. The Standardized Effect Size (ses) is define as :

$$SES = (I_{obs} – I_s im)/\delta_{sim}$$

¡¡¡¡¡¡¡ HEAD ======= ¡¡¡¡¡¡¡ HEAD =======

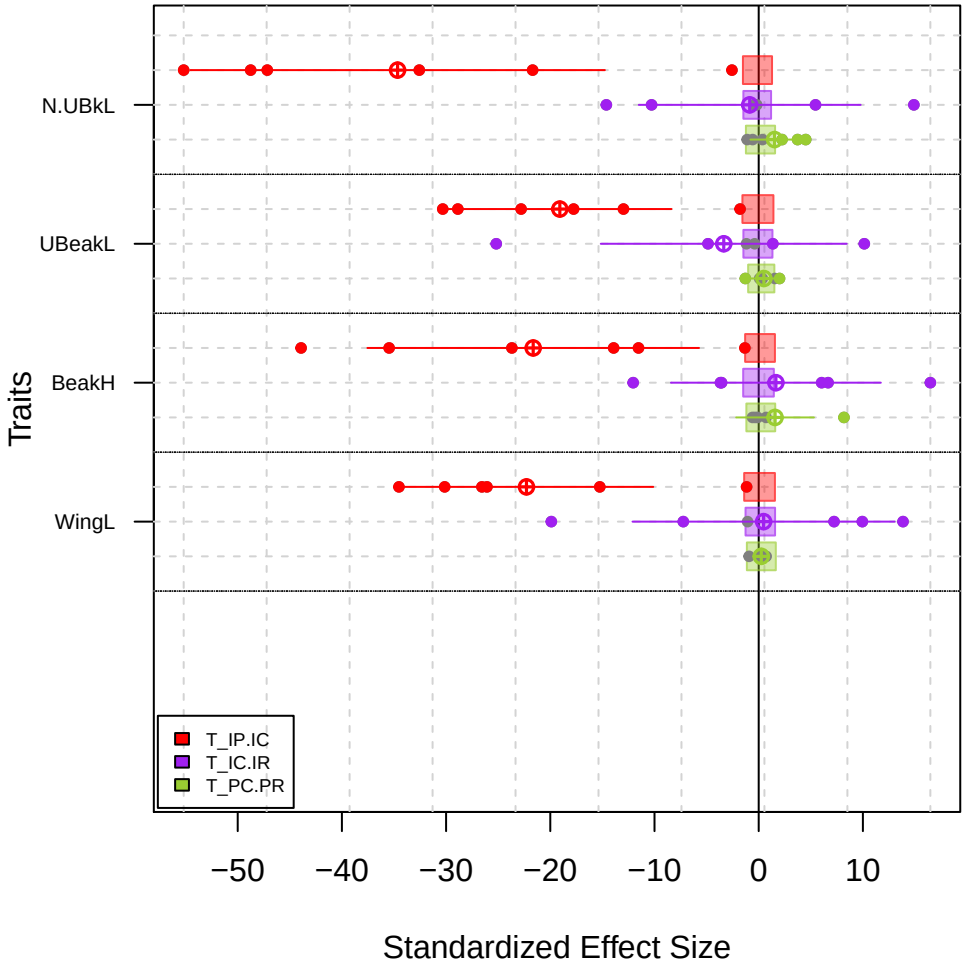
¿¿¿¿¿¿¿ 5f89dfbb4bbfa61805f8e116f94cee418c14cfee

¡¡¡¡¡¡¡ HEAD where I_{obs} is the observed value, I_{sim} the mean of values calculated from the null model and δ_{sim} the standard deviation of these simulated values.

======= ¿¿¿¿¿¿¿ 5f89dfbb4bbfa61805f8e116f94cee418c14cfee ¿¿¿¿¿¿¿ d0b211331074d6462608d116c3317bf39e66e950

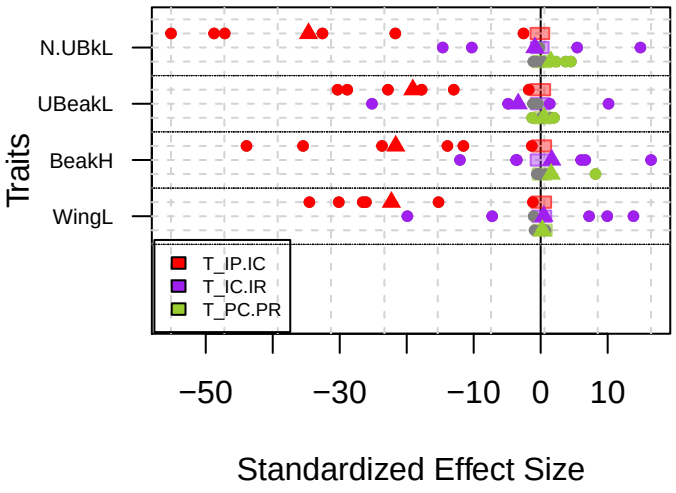
where I_obs is the observed value, I_sim the mean of values calculated from the null model and δ_{sim} the standard deviation of these simulated values.

```
plot(res.finch)
```

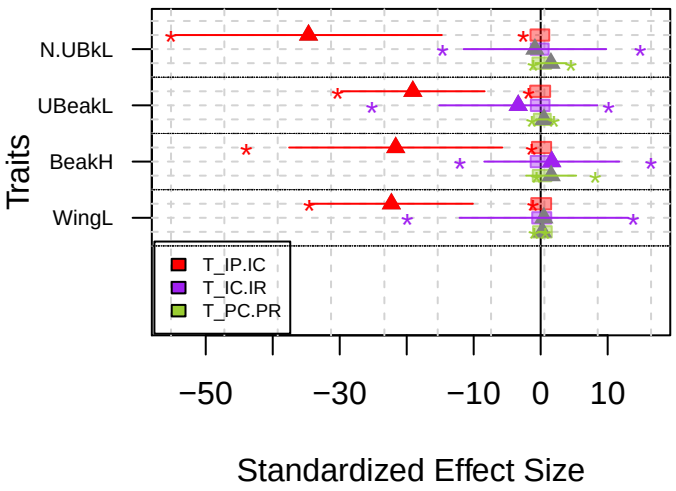


There is multiple kind of representation avaible.

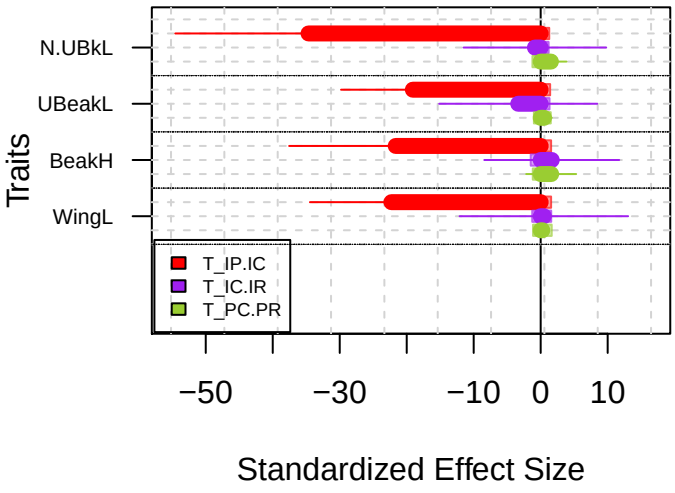
```
plot(res.finch, type = "simple")
```




```
plot(res.finch, type = "simple_range")
```

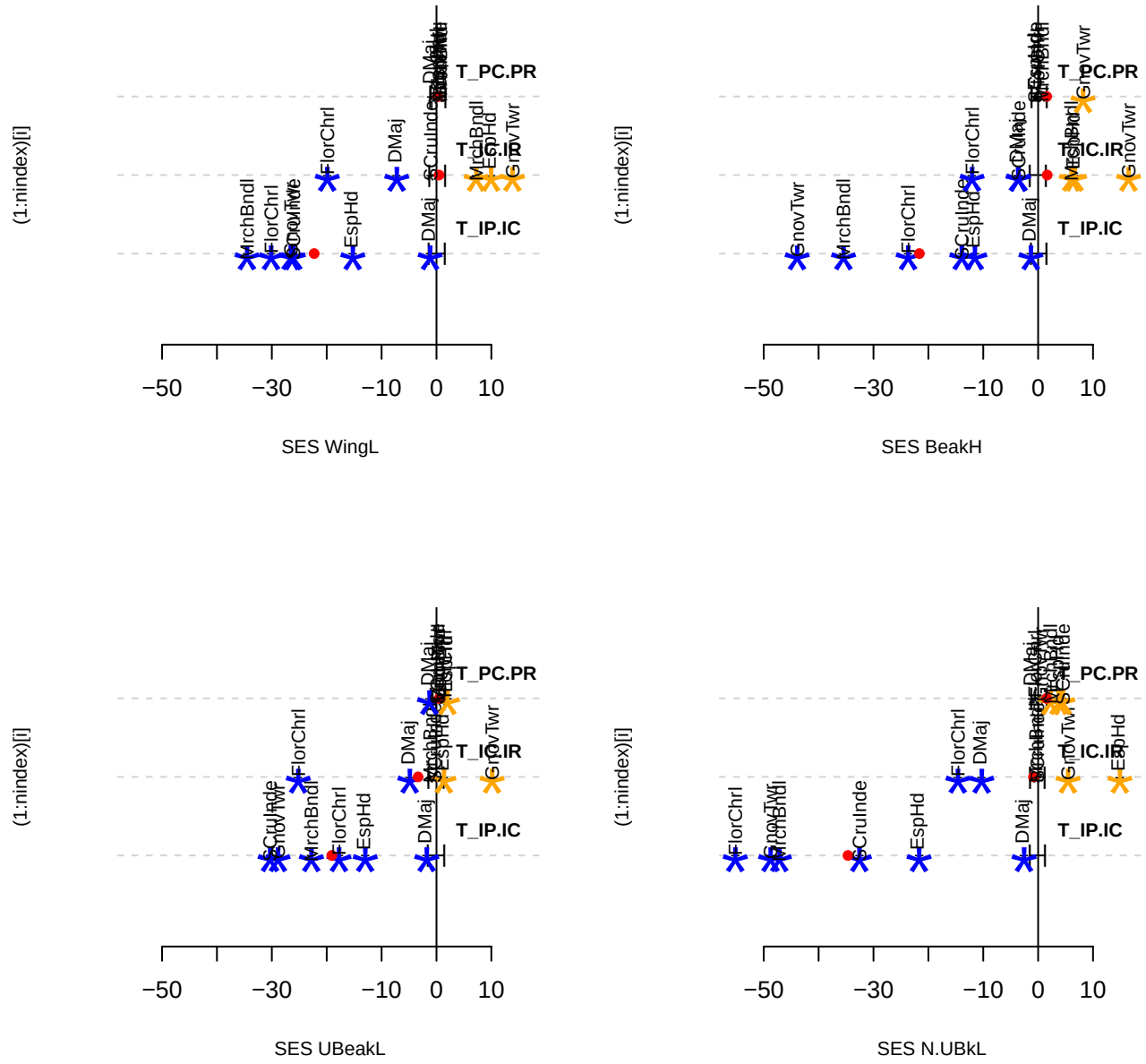


```
plot(res.finch, type = "barplot")
```

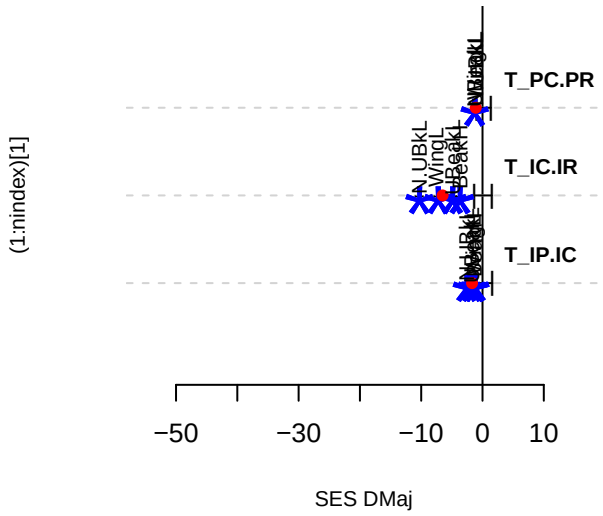


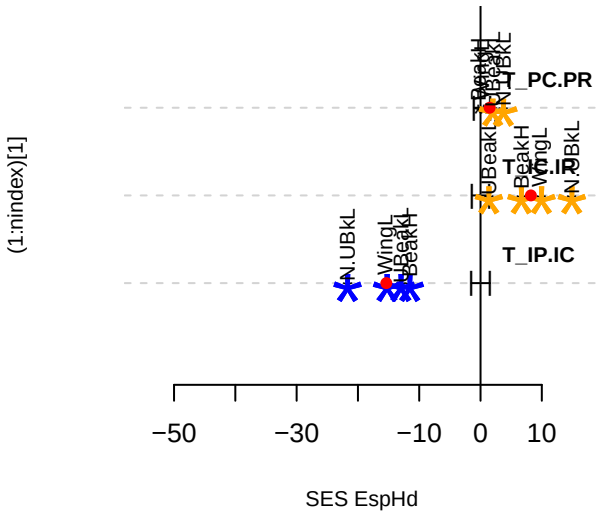
If you want to specifically look at traits or sites statistics, use the argument `type = "bytraits"` or `"bysites"`.

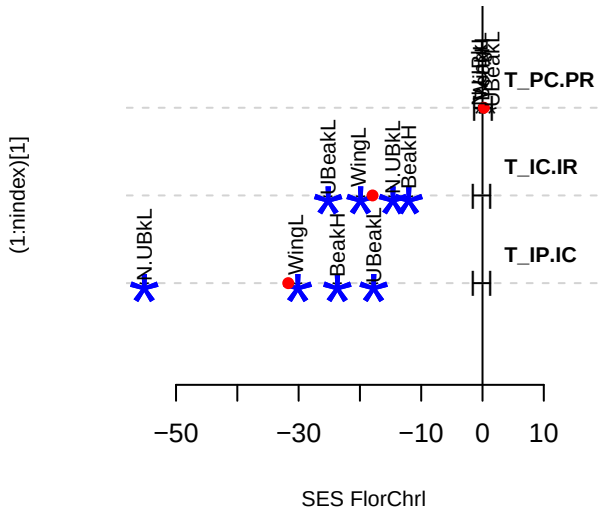
```
par(mfrow=c(2,2))

plot(res.finch, type = "bytraits")
```

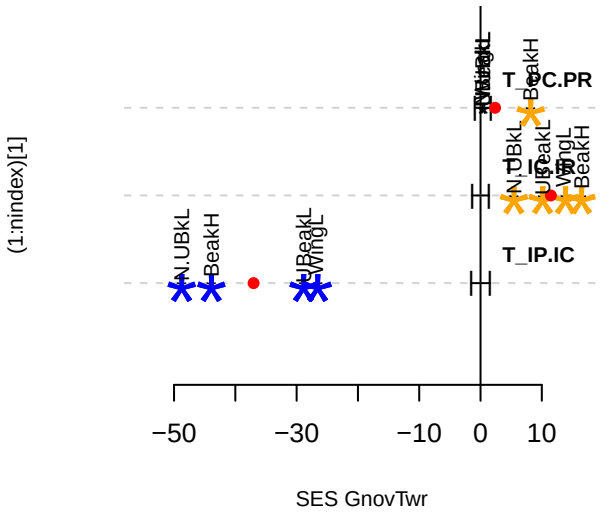


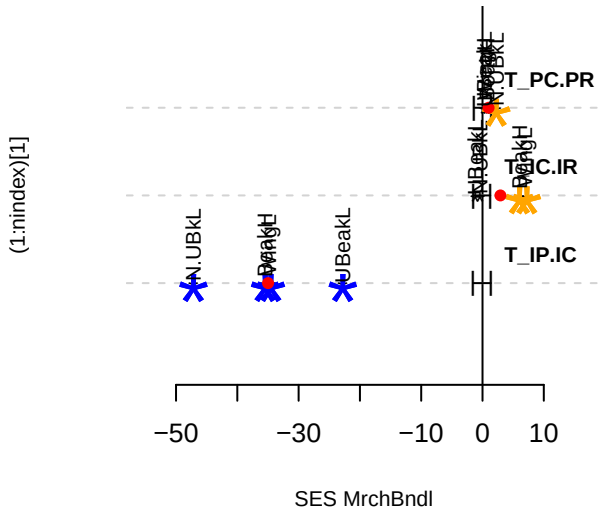

```
plot(res.finch, type = "bysites")
```

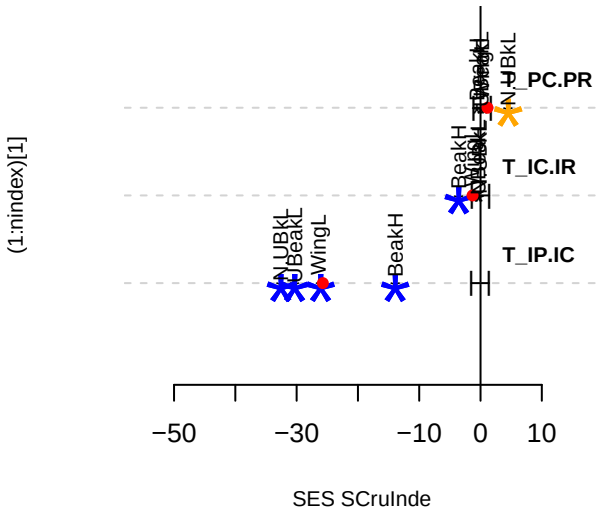

(1:nindex)[1]
T_PC.PR
T_IC.IR
T_IP.IC
N.UBkL
WingL
BeakH
−50
−30
−10
0
10
SES DMaj


(1:nindex)[1]
T_PC.PR
T_IC.IR
T_IP.IC
N.UBkL
WingL
BeakH
−50
−30
−10
0
10
SES EspHd


(1:nindex)[1]
T_PC.PR
T_IC.IR
T_IP.IC
N.UBkL
WingL
BeakH
−50
−30
−10
0
10
SES FlorChrl


(1:nindex)[1]
T_PC.PR
T_IC.IR
T_IP.IC
N.UBkL
BeakH
WingL
−50
−30
−10
0
10
SES GnovTwr






```
par(old.par) # reset default graphical parameters
```

```
summary(res.finch) #S3 summary method for class Tstats
```

```
## [1] "Observed values"
## $T_IP.IC
##      WingL             BeakH             UBeakL            N.UBkL
##  Min.   :0.0343   Min.   :0.0153   Min.   :0.0267   Min.   :0.0238
##  1st Qu.:0.0436   1st Qu.:0.0191   1st Qu.:0.0417   1st Qu.:0.0367
##  Median :0.0645   Median :0.0400   Median :0.0544   Median :0.0403
##  Mean   :0.1161   Mean   :0.1094   Mean   :0.0764   Mean   :0.0615
##  3rd Qu.:0.1012   3rd Qu.:0.0580   3rd Qu.:0.0629   3rd Qu.:0.0494
##  Max.   :0.3831   Max.   :0.4852   Max.   :0.2196   Max.   :0.1764
##
## $T_IC.IR
##      WingL             BeakH             UBeakL           N.UBkL
##  Min.   :0.0925   Min.   :0.0632   Min.   :0.246   Min.   :0.257
##  1st Qu.:0.6739   1st Qu.:0.4913   1st Qu.:0.668   1st Qu.:0.724
##  Median :1.1707   Median :1.1831   Median :0.944   Median :0.980
##  Mean   :1.0852   Mean   :1.1236   Mean   :0.911   Mean   :0.968
##  3rd Qu.:1.6257   3rd Qu.:1.6242   3rd Qu.:1.080   3rd Qu.:1.314
##  Max.   :1.7916   Max.   :2.2802   Max.   :1.629   Max.   :1.525
##
## $T_PC.PR
##      WingL            BeakH             UBeakL           N.UBkL
##  Min.   :0.226   Min.   :0.0868   Min.   :0.933   Min.   :0.936
##  1st Qu.:0.898   1st Qu.:0.8625   1st Qu.:0.983   1st Qu.:1.039
##  Median :1.223   Median :1.0589   Median :1.125   Median :1.099
##  Mean   :1.130   Mean   :1.2370   Mean   :1.149   Mean   :1.196
##  3rd Qu.:1.474   3rd Qu.:1.3286   3rd Qu.:1.215   3rd Qu.:1.351
##  Max.   :1.762   Max.   :3.0016   Max.   :1.530   Max.   :1.585
##
## [1] "null values"
## $T_IP.IC_nm
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.422   0.961   0.992   1.000   1.020   3.230
##
## $T_IC.IR_nm
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.678   0.962   1.000   1.000   1.040   1.520
##
## $T_PC.PR_nm
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   0.331   0.729   1.070   1.340   7.960      19
```

```
attributes(sum_Tstats(res.finch)) #Another mean to summarize Tstatistics
```

```
## $names
## [1] "p.value" "percent" "sites"   "binary"
```

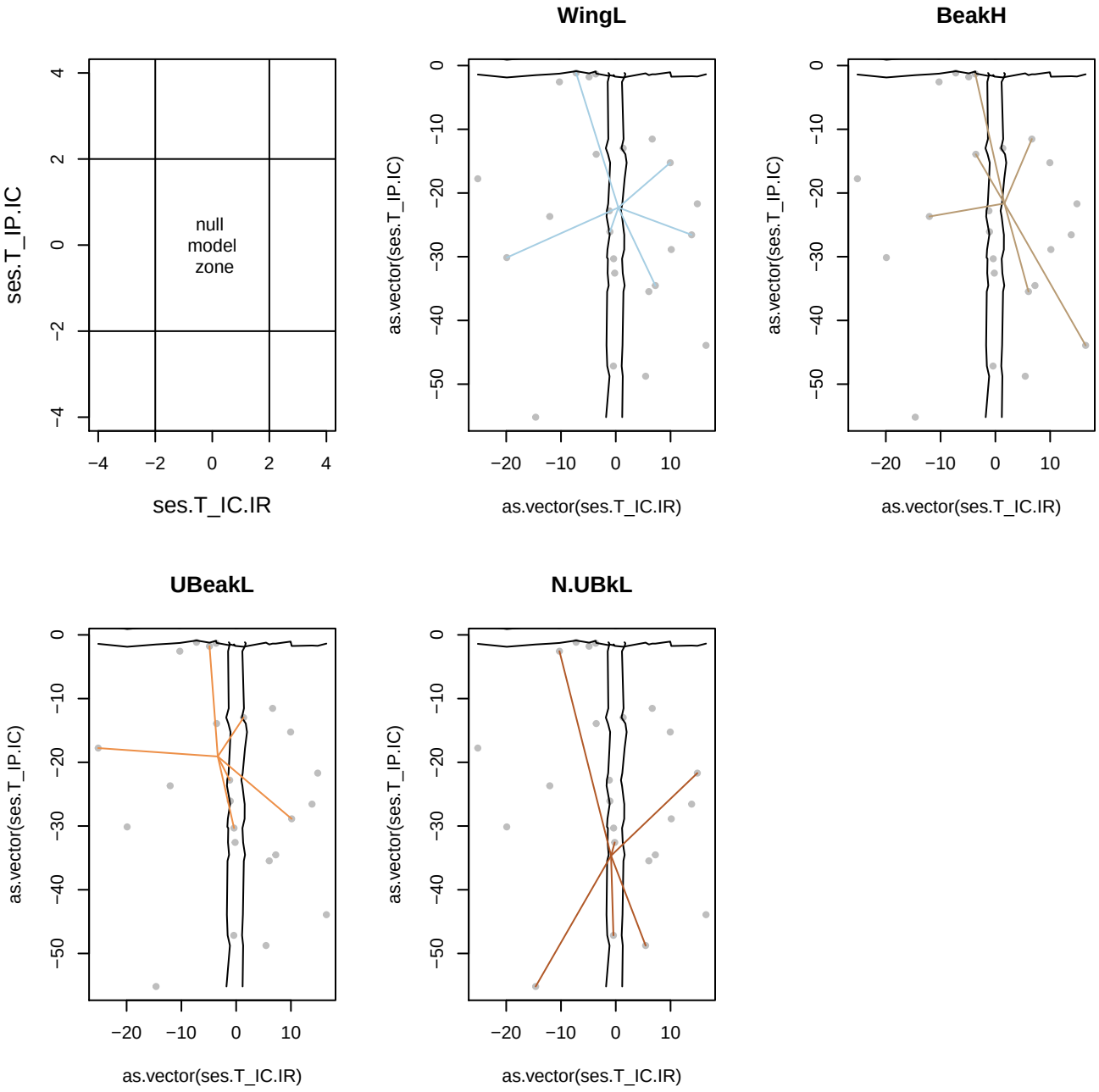
```r
head(sum_Tstats(res.finch)$p.value, 10)
```

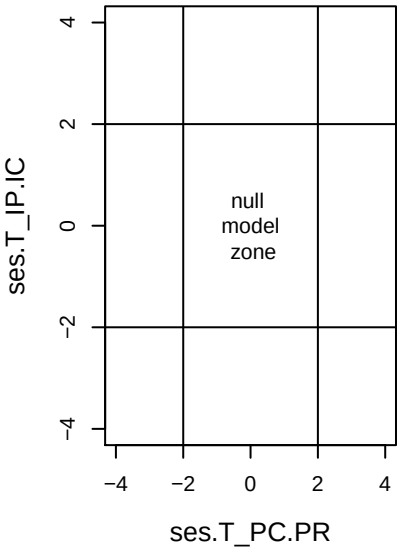
```
##              WingL BeakH UBeakL N.UBkL
## T_IP.IC.inf    0.1   0.1    0.1    0.1
## T_IP.IC.inf    0.1   0.1    0.1    0.1
## T_IP.IC.inf    0.1   0.1    0.1    0.1
## T_IP.IC.inf    0.1   0.1    0.1    0.1
## T_IP.IC.inf    0.1   0.1    0.1    0.1
## T_IP.IC.inf    0.1   0.1    0.1    0.1
## T_IP.IC.sup    1.0   1.0    1.0    1.0
## T_IP.IC.sup    1.0   1.0    1.0    1.0
## T_IP.IC.sup    1.0   1.0    1.0    1.0
## T_IP.IC.sup    1.0   1.0    1.0    1.0
```

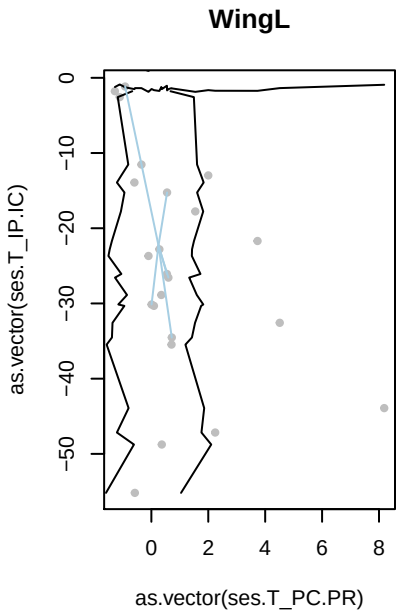
5.1.2 Plot T-statistics correlations

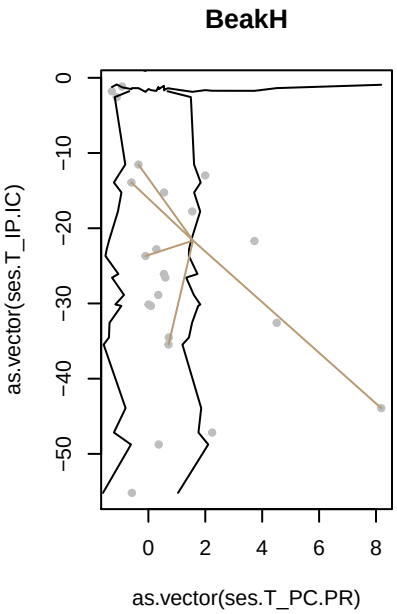
We can also see T-statistics correlations and theirs correlation with others variables (e.g. a gradient variable, or the species richness).

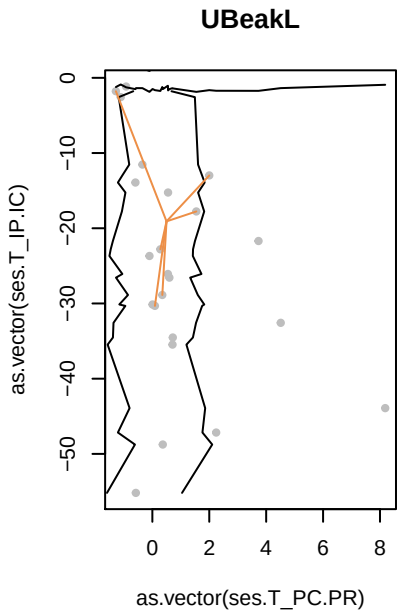
```
par(mfrow = c(2,3))
plotCorTstats(res.finch, plot.ask = FALSE, multipanel = F)
```

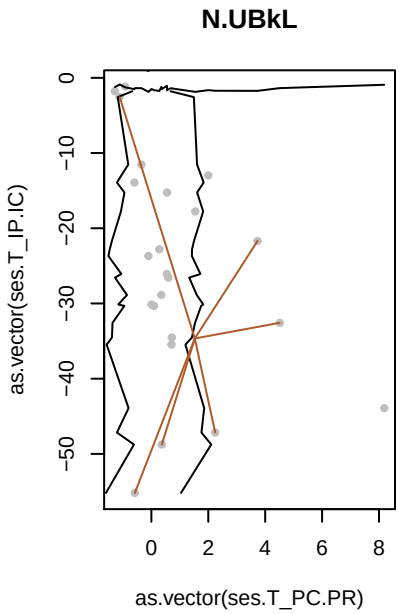


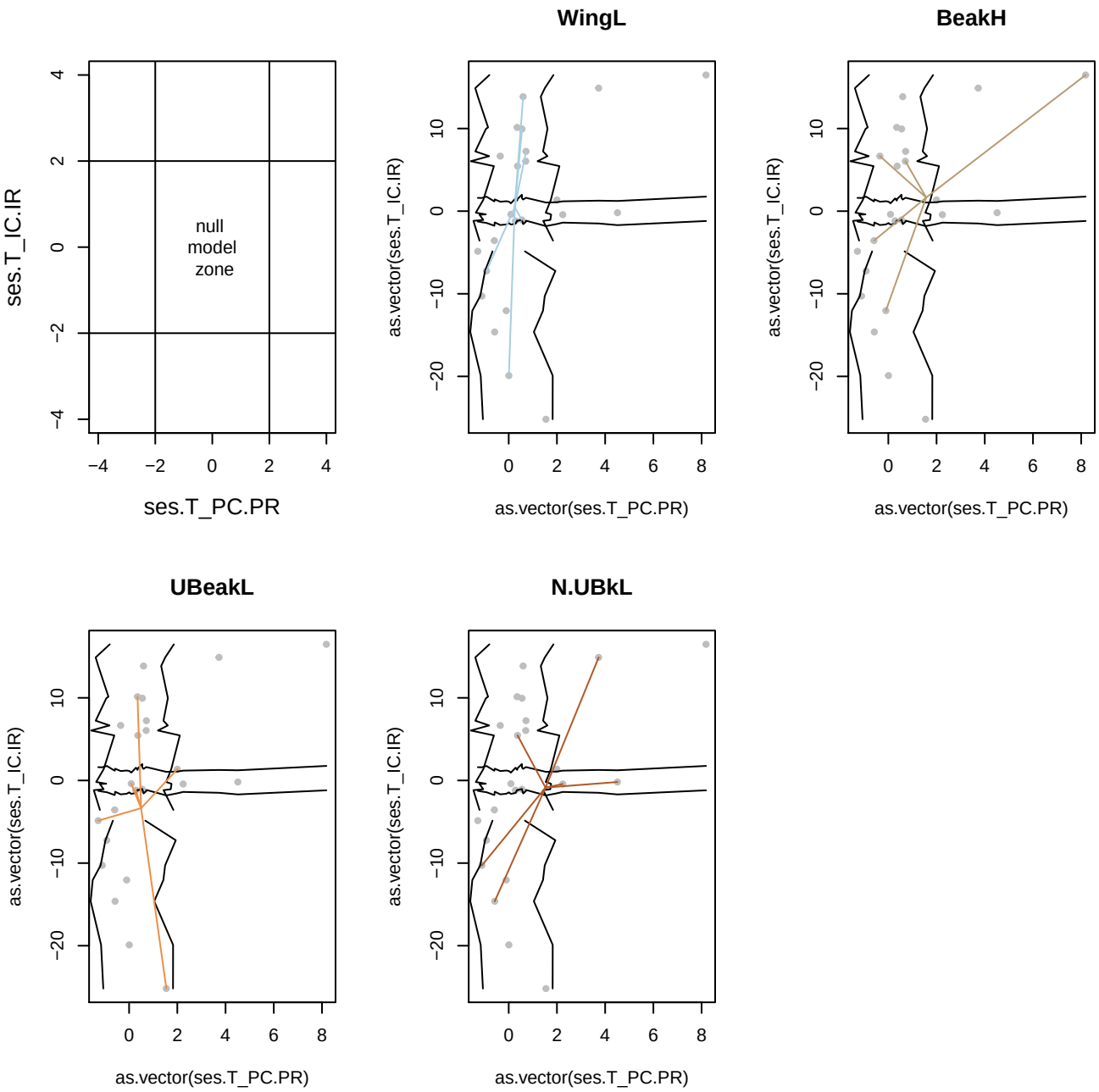

ses.T_IP.IC
ses.T_PC.PR
null
model
zone


WingL
as.vector(ses.T_IP.IC)
as.vector(ses.T_PC.PR)


BeakH
as.vector(ses.T_IP.IC)
as.vector(ses.T_PC.PR)

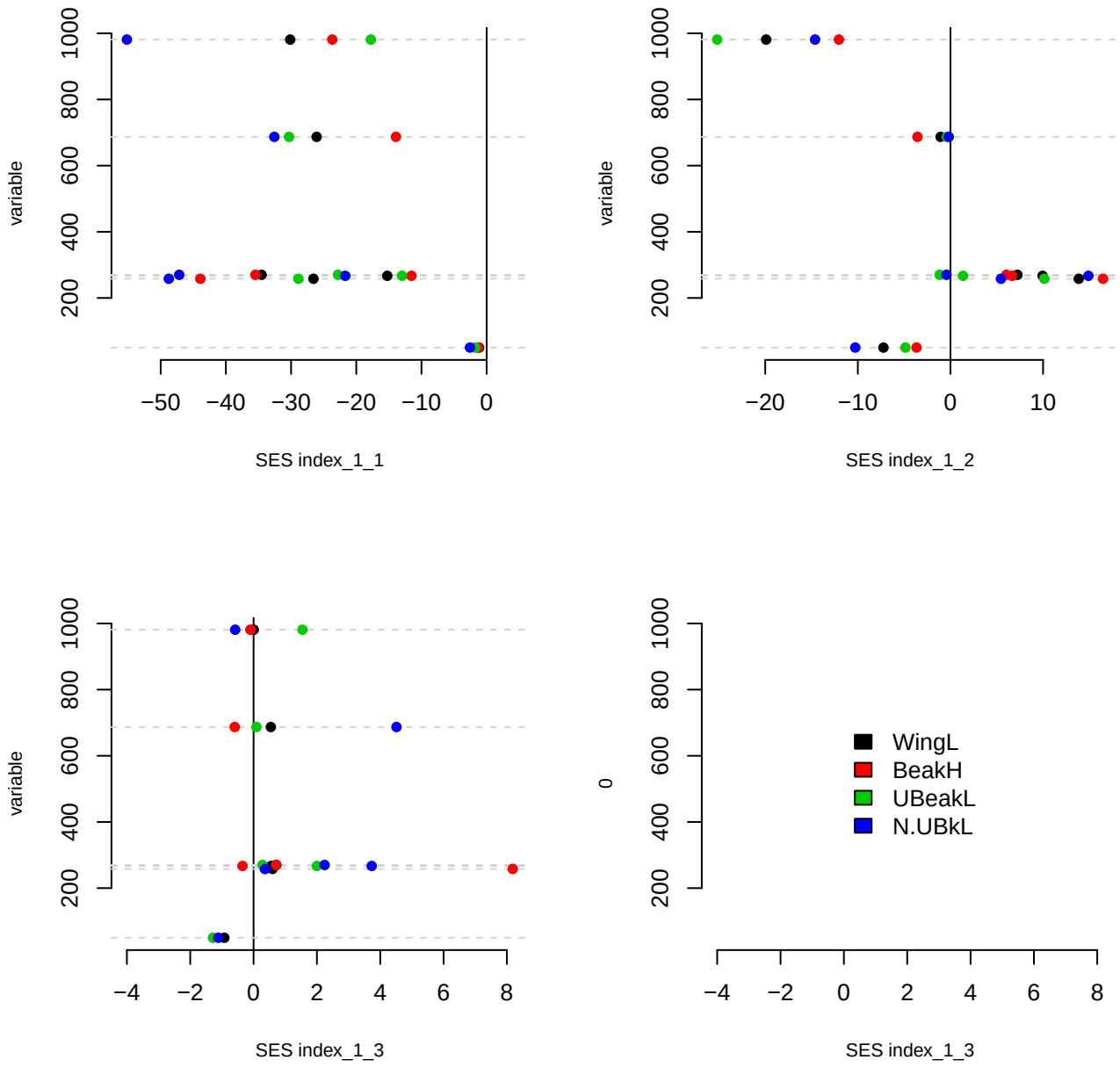

UBeakL
as.vector(ses.T_IP.IC)
as.vector(ses.T_PC.PR)


N.UBkL
as.vector(ses.T_IP.IC)
as.vector(ses.T_PC.PR)



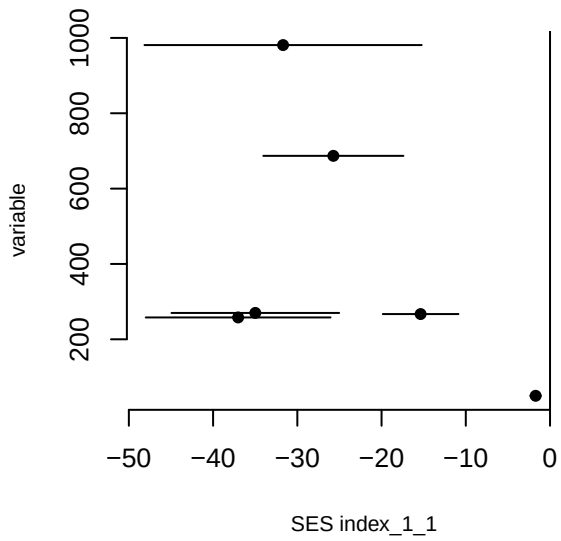
Here we plot T-statistics (in the standardized effect size SES form) in function of species richness by sites.

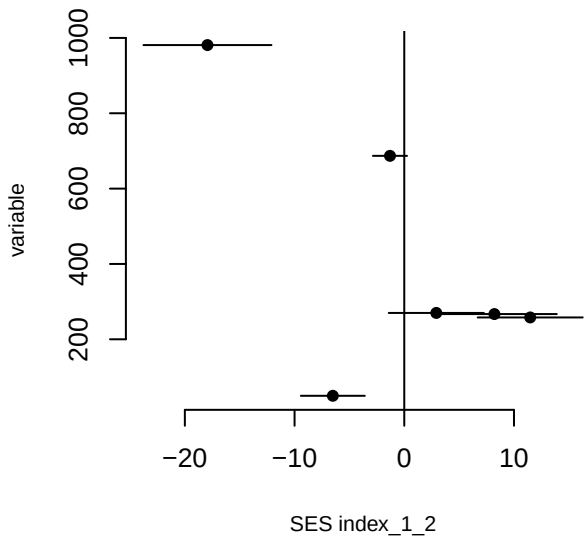
```
par(mfrow = c(2,2))
species.richness<-table(ind.plot.finch)
plotSESvar(as.listofindex(list(res.finch)), species.richness,
       multipanel = F)
```

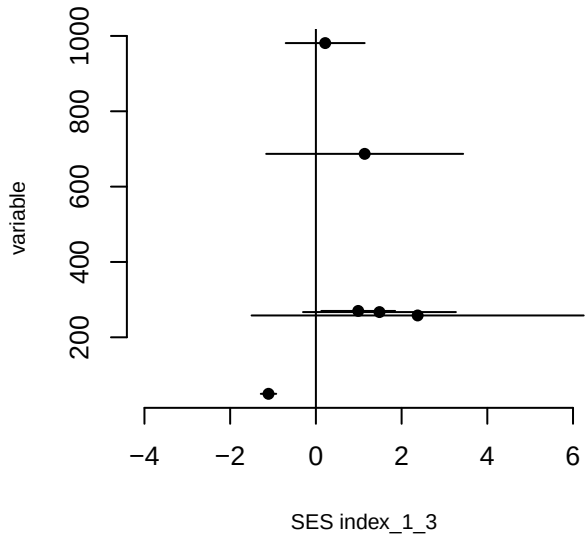


Same plot with `resume = TRUE`.

```
par(mfrow = c(2,2))
plotSESvar(as.listofindex(list(res.finch)), species.richness,
       resume = T, multipanel = F)
```








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

5.2 Others univariates or multivariates metrics: function `ComIndex` and `ComIndexMulti`

The function `ComIndex` allow choosing your own function (like mean, range, variance, ...) to calculate customize metrics. Here `CVNND` refers to the Coefficient of Variation of the Nearest Neighborhood Distance. `ComIndexMulti` do the same things for multivariate metrics.

```
#Define the function s to calculate
funct<-c("mean(x, na.rm = T)", "kurtosis(x, na.rm = T)",
     "max(x, na.rm = T) - min(x, na.rm = T)", "CVNND(x)" )

#Test against the null model regional.ind
res.finch.sp_regional.ind<-ComIndex(traits = traits.finch, index = funct, sp = sp.finch,
                            nullmodels = "regional.ind", ind.plot = ind.plot.finch,
                             nperm = 9, print = FALSE)

## Error:  objet 's' introuvable

#Test against the null model regional.pop
#Individuals values are transformed in populational values
res.finch.sp_regional.pop<-ComIndex(traits = traits.finch, index = funct, sp = sp.finch,
               nullmodels = "regional.pop", ind.plot = ind.plot.finch,
               nperm = 9, print = FALSE)
```

These two functions allows to calculate index by sites for example using `"tapply(x, sites, mean)"`.

```
funct.1<-c("tapply(x, ind.plot.finch, function(x) mean(x, na.rm = T))",
     "tapply(x, ind.plot.finch, function(x) kurtosis(x, na.rm = T))",
     "tapply(x, ind.plot.finch, function(x) max(x, na.rm = T)-min(x, na.rm = T))",
     "tapply(x, ind.plot.finch, function(x) CVNND(x))" )

#The function IndexByGroups permit to easily obtain the above lines
IndexByGroups(funct, "ind.plot.finch")

## [1] "tapply(x, ind.plot.finch, function(x) mean(x, na.rm = T))"
## [2] "tapply(x, ind.plot.finch, function(x) kurtosis(x, na.rm = T))"
## [3] "tapply(x, ind.plot.finch, function(x) max(x, na.rm = T) - min(x, na.rm = T))"
## [4] "tapply(x, ind.plot.finch, function(x) CVNND(x))"

##Null model local is trivial for these functions
##because randomization is within community only

res.finch.ind_loc<-ComIndex(traits = traits.finch, index = funct.1, sp = sp.finch,
               nullmodels = "local", ind.plot = ind.plot.finch,
               nperm = 9, print = FALSE)
res.finch.ind_reg<-ComIndex(traits = traits.finch, index = funct.1, sp = sp.finch,
               nullmodels = "regional.ind", ind.plot = ind.plot.finch,
               nperm = 9, print = FALSE)

## Error:  objet 's' introuvable
```

We can calculate index with or without intraspecific variance.

```
#calculate  of means by population (name_sp_site is a name of a population)
#determine the site for each population (sites_bypop)

name_sp_sites = paste(sp.finch, ind.plot.finch, sep = "_")
traits.by.pop<-apply(traits.finch, 2 ,
           function (x) tapply(x, name_sp_sites, mean , na.rm = T))

sites_bypop<-lapply(strsplit(paste(rownames(traits.by.pop), sep = "_"), split = "_"),
          function(x) x[3])

fact<-unlist(sites_bypop)

funct.2<-c("tapply(x, fact, function(x) mean(x, na.rm = T))",
          "tapply(x, fact, function(x) kurtosis(x, na.rm = T))",
          "tapply(x, fact, function(x) max(x, na.rm = T)-min(x, na.rm = T))",
          "tapply(x, fact, function(x) CVNND(x))")
```

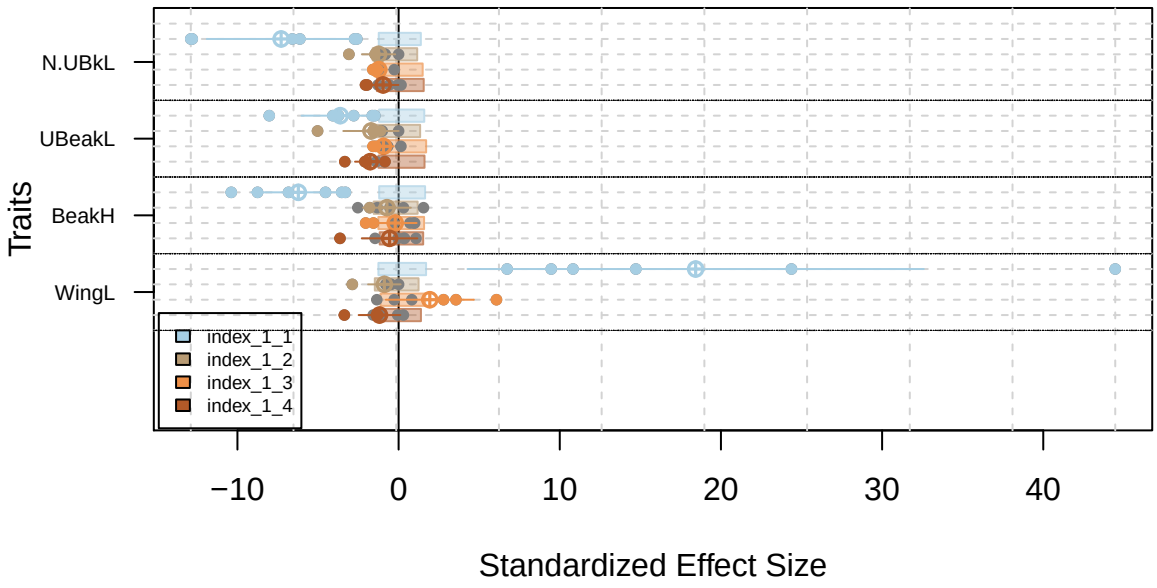
Now calculate index with or without intraspecific variance thanks to function `ComIndex`.

```
res.finch.withIV<-ComIndex(traits = traits.finch, index = funct.1,
               sp = sp.finch, nullmodels = "regional.ind",
               ind.plot = ind.plot.finch, nperm = 9, print = FALSE)

## Error:  objet 's' introuvable

res.finch.withoutIV<-ComIndex(traits = traits.finch, index = funct.2,
               sp = sp.finch, nullmodels = "regional.pop",
               ind.plot = ind.plot.finch, nperm = 9, print = FALSE)
```

5.2.1 S3 methods for class ComIndex and ComIndexMulti

ComIndex and ComIndexMulti class are associated to S3 methods plot, print and summary.

```
res.finch.withIV

## Error:  objet 'res.finch.withIV' introuvable

summary(res.finch.withIV)

## Error:  objet 'res.finch.withIV' introuvable

plot(res.finch.withIV)

## Error:  objet 'res.finch.withIV' introuvable

plot(res.finch.withoutIV)
```



Now plot the two analysis together.

```
plot(as.listofindex(list(res.finch.withIV, res.finch.withoutIV)))

## Error:  objet 'res.finch.withIV' introuvable
```

5.2.2 Plot Tstats and other uni/multivariates metrics together

The class listofindex permits to stock different metrics computed using `Tstats`, `ComIndex` and `ComIndexMulti` and compared to different null model. To do that we can use the Standardized Effect Size (ses) define as :

$$SES = (I_obs – I_sim)/\delta_{sim}$$

where I_obs is the observed value, I_sim the mean of values calculated from the null model and δ_{sim} the standard deviation of these simulated values.

```
list.ind1<-list(res.finch.withIV, res.finch.withoutIV)
index.list1<-as.listofindex(list.ind1)

plot(index.list1)
```

```
list.ind<-list(res.finch.withIV, res.finch.withoutIV, res.finch)

## Error:  objet 'res.finch.withIV' introuvable
```

```
namesindex.i.l1 = c("mean", "kurtosis", "range", "CVNND",
         "mean.pop", "kurtosis.pop", "range.pop", "CVNND.pop",
         "T_IP.IC", "T_IC.IR", "T_PC.PR")

i.l1<-as.listofindex(list.ind, namesindex = namesindex.i.l1)

## Error:  objet 'list.ind' introuvable

class(i.l1)

## Error:  objet 'i.l1' introuvable
```

The plot type `bytraits` allows plotting all SES traits values for all sites or all traits

```
par(mfrow = c(2,3))
plot(i.l1,type = "bysites")

## Error:  objet 'i.l1' introuvable

par(mfrow = c(2,2))
plot(i.l1,type = "bytraits")

## Error:  objet 'i.l1' introuvable

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

The other plot types are the same as plot.Tstats.

```
par(mfrow = c(2,2))

plot(i.l1)

## Error:  objet 'i.l1' introuvable

plot(i.l1,type = "simple_range")

## Error:  objet 'i.l1' introuvable

plot(i.l1,type = "normal")

## Error:  objet 'i.l1' introuvable

plot(i.l1,type = "barplot")

## Error:  objet 'i.l1' introuvable

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

5.3 More information on multivariates index

For most multivariate functions we need to replace (or exclude) NA values. For this example, we use the package `mice` to complete the data.

```
comm<-t(table(ind.plot.finch,1:length(ind.plot.finch)))

require(mice)
traits = traits.finch
mice<-mice(traits.finch)
traits.finch.mice<-complete(mice)
```

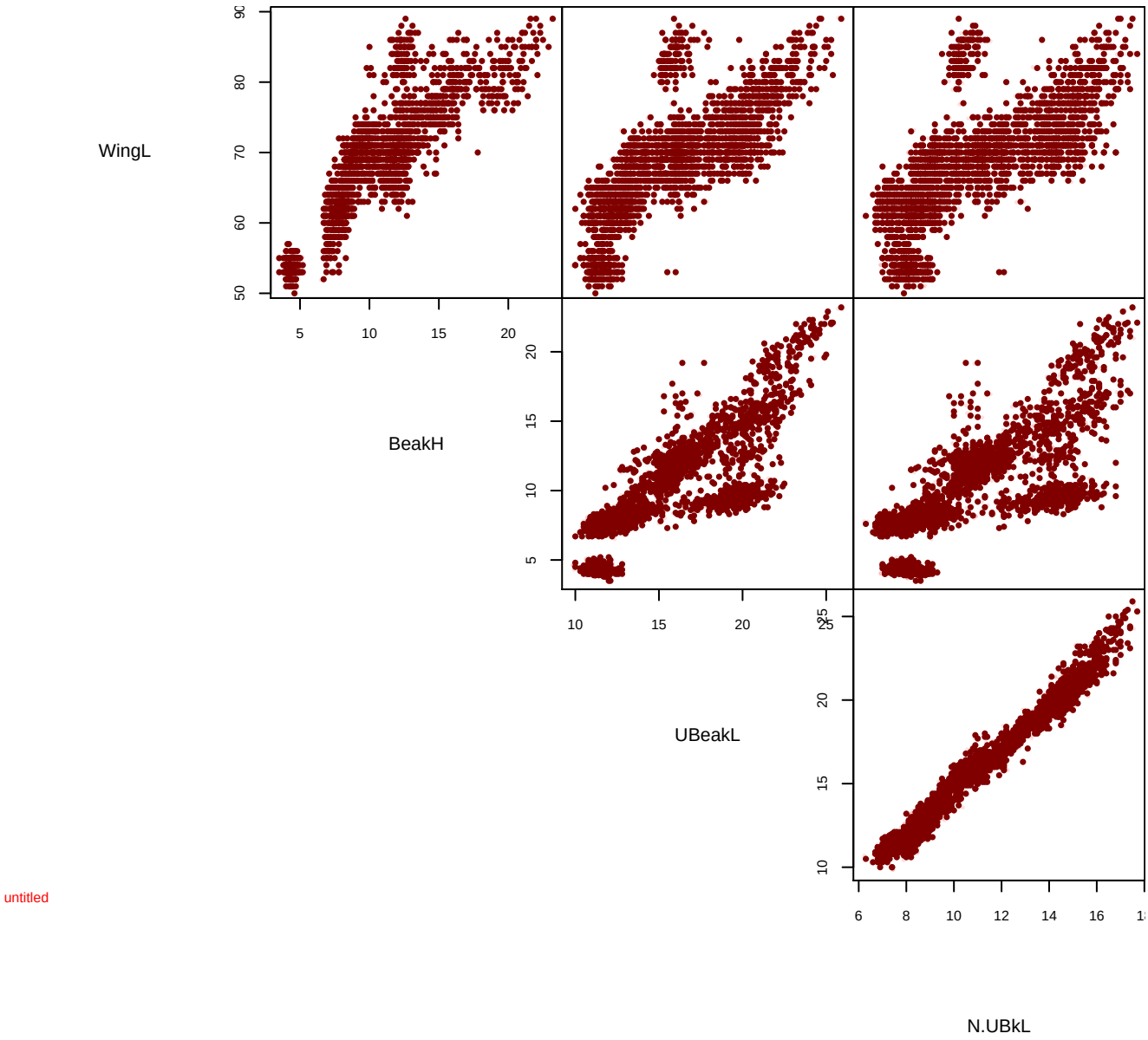
A simple example to illustrate the concept of the function `ComIndexMulti`

```
n_sp_plot<-as.factor(paste(sp.finch, ind.plot.finch, sep = "_"))
res.sum.1<-ComIndexMulti(traits.finch,
              index = c("sum(scale(x), na.rm = T)", "sum(x, na.rm = T)"),
              by.factor = n_sp_plot, nullmodels = "regional.ind",
              ind.plot = ind.plot.finch, nperm = 9, sp = sp.finch)

## [1] "creating null models"

## Error:  objet 's' introuvable

res.sum.1

## Error:  objet 'res.sum.1' introuvable
```

A more interesting example using the function `hypervolume` from the package ... `hypervolume` (Blonder et al., 2014). We show here several results which differed in there factor that delimit the group to calculate different hypervolume (argument `byfactor`).

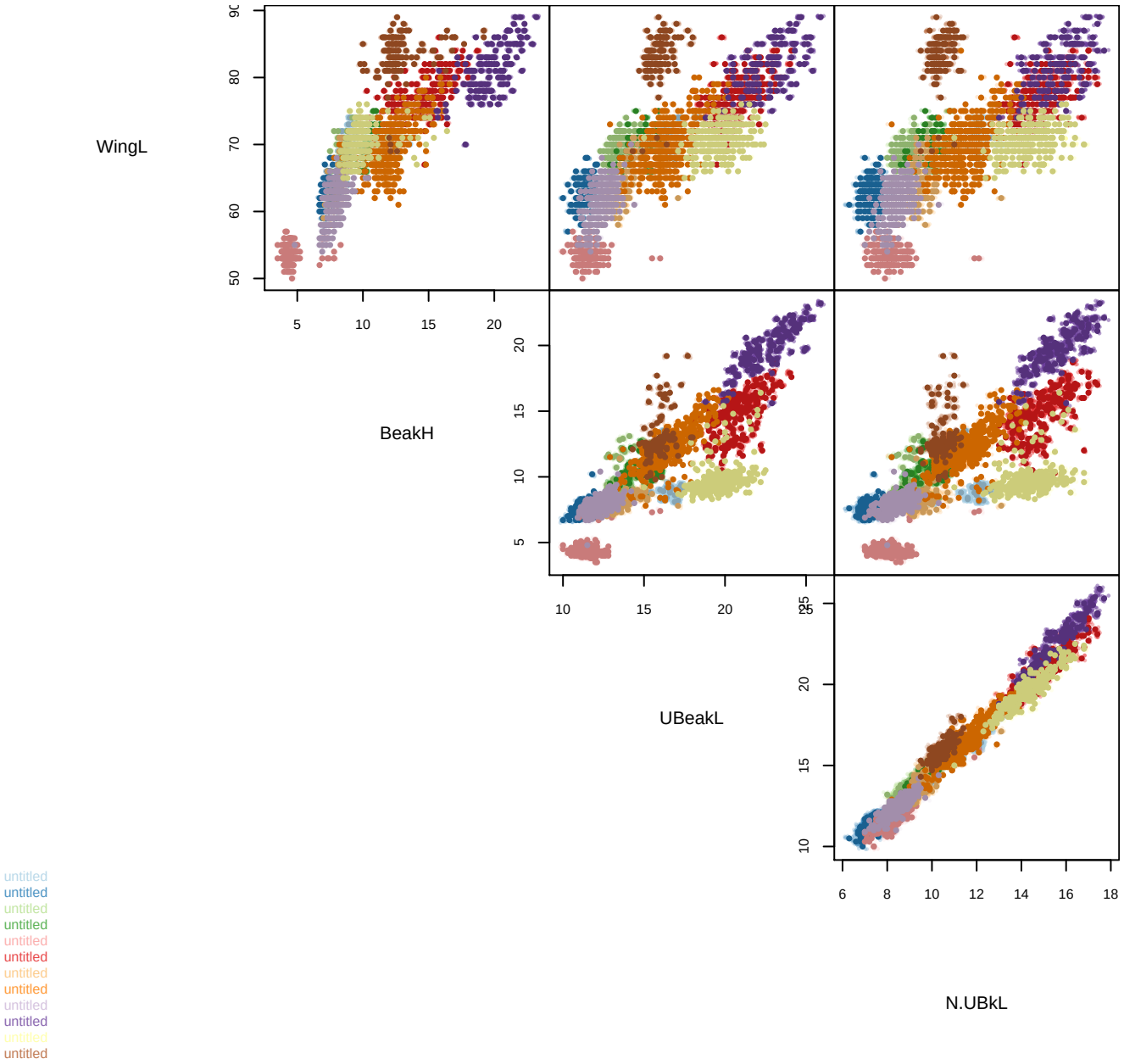
First, let's try the `hypervolume` function one finch data.

```
hv<-hypervolume(traits.finch.mice,
        reps = 100,bandwidth = 0.2,
        verbose = F, warnings = F)
plot(hv)
```

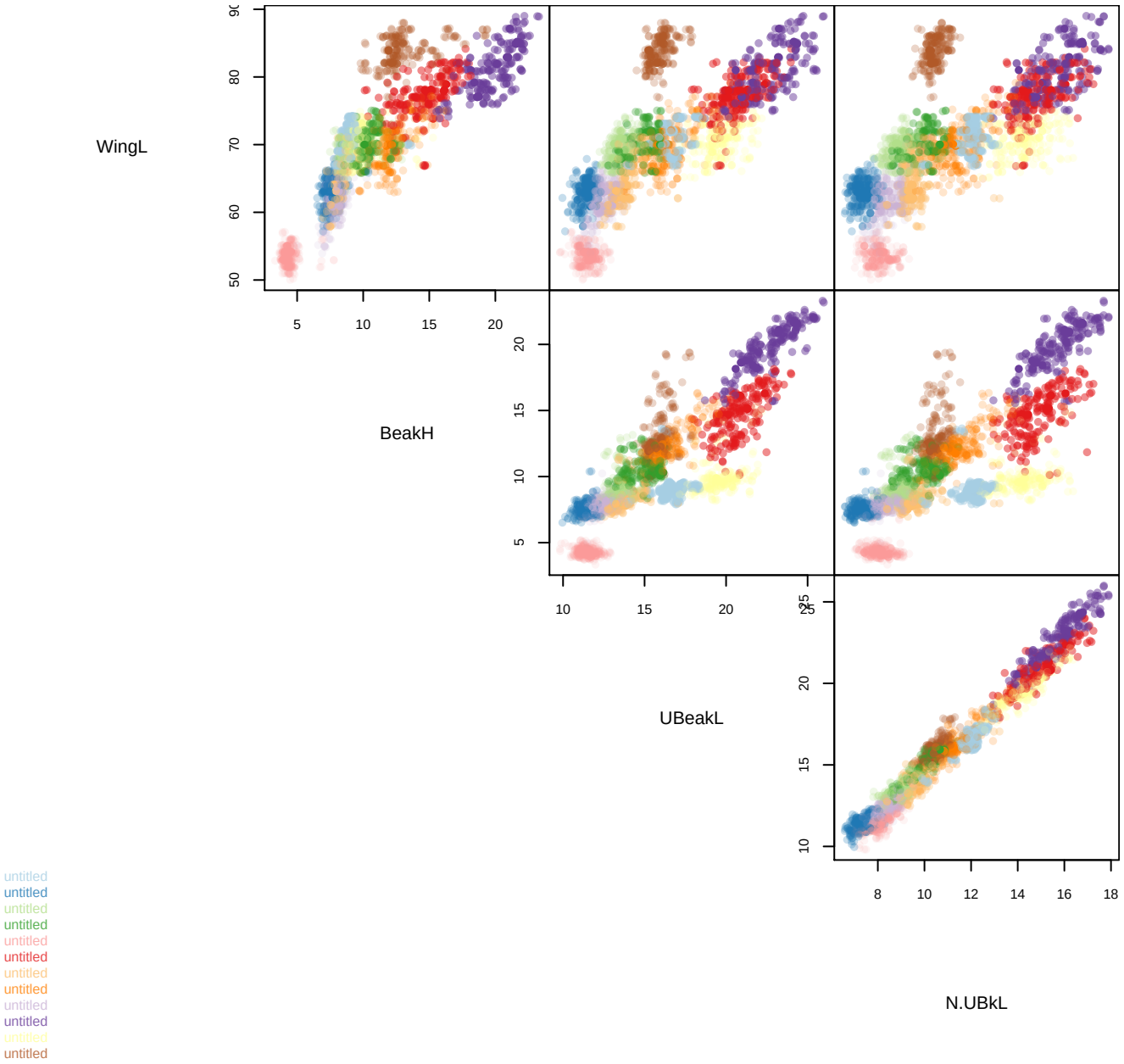


Now, we can do the same analysis for each species.

```
hv.list<-new("HypervolumeList")
hv.list2<-list()

for(i in 1: length(table(sp.finch))) {
 hv.list2[[i]]<-hypervolume(traits.finch.mice[sp.finch == levels(sp.finch)[i], ],
        reps = 1000,bandwidth = 0.2,
        verbose = F, warnings = F)
}

hv.list@HVList<-hv.list2
require(adegenet)

## Loading required package:  adegenet
##    ==========================
##    adegenet 1.4-2 is loaded
##    ==========================
##
##  - to start, type '?adegenet'
##  - to browse adegenet website, type 'adegenetWeb()'
##  - to post questions/comments:  adegenet-forum@lists.r-forge.r-project.org

colorhv<-transp(funky(nlevels(sp.finch)), alpha = 0.8)

plot(hv.list, colors = colorhv, darkfactor = 0.8)
```




```
plot(hv.list, colors = colorhv, darkfactor = 0.8, showdata = F, npmax = 200, cex.random = 1)
```




```
summary(hv.list)
```

The standard example of the **hypervolume** package also use finch data but at the species level.

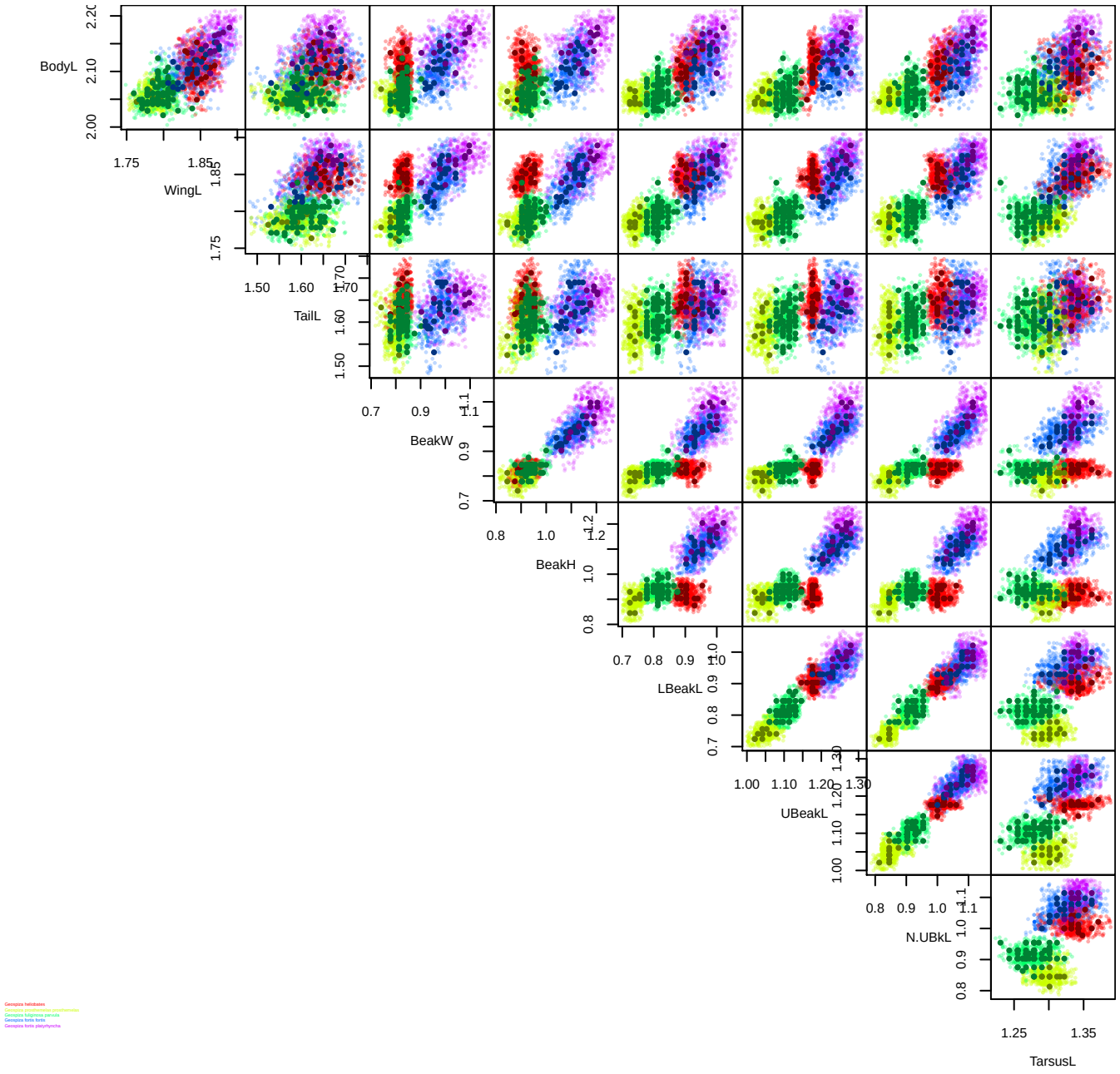
```
doHypervolumeFinchDemo=TRUE
demo('finch', package = 'hypervolume')

##
##
##  demo(finch)
##  ---- ~~~~~
##
## > if (exists('doHypervolumeFinchDemo')==TRUE)
```

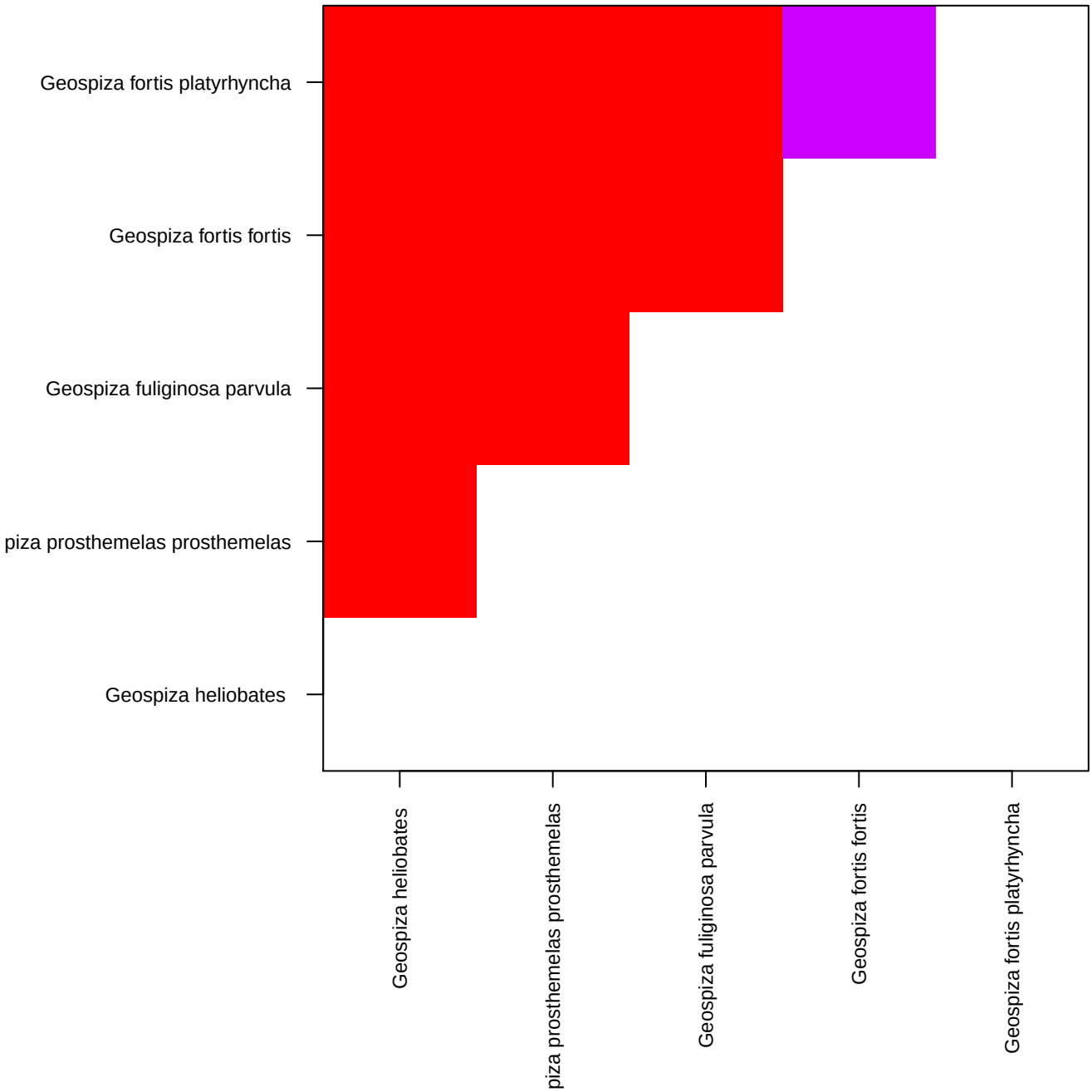
```
## + {
## +   data(finch)
## +
## +   species_list = unique(finch$Species)
## +   num_species = length(species_list)
## +
## +   hv_finches_list = new("HypervolumeList")
## +   hv_finches_list@HVList = vector(mode="list",length=num_species)
## +
## +   # compute hypervolumes for each species
## +   for (i in 1:num_species)
## +   {
## +     this_species = subset(finch, Species==species_list[i])
## +     # keep the trait data
## +     this_species_log <- log10(this_species[,2:ncol(this_species)])
## +     # make a hypervolume using auto-bandwidth
## +     hv_finches_list@HVList[[i]] <- hypervolume(this_species_log, bandwidth=estimate_band
## +                                                reps=10000, quantile=0, name=as.character
## +   }
## +
## +   # compute all pairwise overlaps
## +   overlap = matrix(NA, nrow=num_species, ncol=num_species)
## +   dimnames(overlap)=list(species_list, species_list)
## +   for (i in 1:num_species)
## +   {
## +     for (j in i:num_species)
## +     {
## +       if (i!=j)
## +       {
## +         # compute set operations on each pair
## +         this_set = hypervolume_set(hv_finches_list@HVList[[i]], hv_finches_list@HVList[[
## +         # calculate a Sorensen overlap index (2 x shared volume / sum of |hv1| + |hv2|)
## +         overlap[i,j] = 2 * this_set@HVList$Intersection@Volume / (hv_finches_list@HVList
## +       }
## +     }
## +   }
## +
## +
## +
## +   # show all hypervolumes
## +   plot(hv_finches_list,npmax=500,darkfactor=0.5,cex.legend=0.25,cex.names=0.75)
## +
## +   # show pairwise overlaps - note that actually very few species overlap in nine dimensi
## +   op <- par(mar=c(10,10,1,1))
## +   image(x=1:nrow(overlap), y=1:nrow(overlap), z=overlap,axes=F,xlab='',ylab='',col=rainb
## +   box()
## +   axis(side=1, at=1:(length(dimnames(overlap)[[1]])),dimnames(overlap)[[1]],las=2,cex.ax
## +   axis(side=2, at=1:(length(dimnames(overlap)[[2]])),dimnames(overlap)[[2]],las=1,cex.ax
```

```
## +   par(op)
## +
## +   rm(doHypervolumeFinchDemo)
## + } else
## + {
## +   message('Demo does not run by default to meet CRAN runtime requirements.')
## +   message('This demo requires approximately 3 minutes to run.')
## +   message('To run the demo, type')
## +   message('\tdoHypervolumeFinchDemo=TRUE')
## +   message('\tdemo(finch)')
## +   message('at the R command line prompt.')
## + }
## Evaluating probability density...
## Building tree... done.
## Querying tree... 5.26316e-006  0.0526368  0.105268  0.1579  0.210532  0.263163  0.315795
## Finished evaluating probability density.
## Beginning volume calculation... done.
## Quantile requested: 0.00   obtained: 0.00
## Evaluating probability density...
## Building tree... done.
## Querying tree... 7.69231e-006  0.0769308  0.153854  0.230777  0.3077  0.384623  0.461546
## Finished evaluating probability density.
## Beginning volume calculation... done.
## Quantile requested: 0.00   obtained: 0.00
## Evaluating probability density...
## Building tree... done.
## Querying tree... 1.23457e-006  0.0123469  0.0246926  0.0370383  0.049384  0.0617296  0.074
## Finished evaluating probability density.
## Beginning volume calculation... done.
## Quantile requested: 0.00   obtained: 0.00
## Evaluating probability density...
## Building tree... done.
## Querying tree... 4.54545e-006  0.0454591  0.0909136  0.136368  0.181823  0.227277  0.27273
## Finished evaluating probability density.
## Beginning volume calculation... done.
## Quantile requested: 0.00   obtained: 0.00
## Evaluating probability density...
## Building tree... done.
## Querying tree... 9.09091e-006  0.0909182  0.181827  0.272736  0.363645  0.454555  0.545464
## Finished evaluating probability density.
## Beginning volume calculation... done.
## Quantile requested: 0.00   obtained: 0.00
## Retaining 30679 points in hv1 and 39646 points in hv2.
## Beginning ball queries...
## Building tree... done.
## Querying tree... 2.52232e-005  0.252257  0.50449  0.756722  done.
## Building tree... done.
## Querying tree... 3.25956e-005  0.325988  0.651944  0.9779  done.
```

```
## Finished ball queries.
## Retaining 57564 points in hv1 and 56061 points in hv2.
## Beginning ball queries...
## Building tree... done.
## Querying tree... 1.78377e-005  0.178395  0.356772  0.535149  0.713526  0.891903  done.
## Building tree... done.
## Querying tree... 1.7372e-005  0.173737  0.347457  0.521176  0.694896  0.868616  done.
## Finished ball queries.
## Retaining 1265 points in hv1 and 61641 points in hv2.
## Beginning ball queries...
## Building tree... done.
## Querying tree... 1.6223e-005  0.162246  0.324476  0.486705  0.648935  0.811165  0.973394
## Building tree... done.
## Querying tree... 0.000790514  done.
## Finished ball queries.
## Retaining 428 points in hv1 and 31427 points in hv2.
## Beginning ball queries...
## Building tree... done.
## Querying tree... 3.18198e-005  0.31823  0.636427  0.954625  done.
## Building tree... done.
## Querying tree... 0.00233645  done.
## Finished ball queries.
## Retaining 39646 points in hv1 and 29878 points in hv2.
## Beginning ball queries...
## Building tree... done.
## Querying tree... 3.34694e-005  0.334728  0.669422  done.
## Building tree... done.
## Querying tree... 2.52232e-005  0.252257  0.50449  0.756722  done.
## Finished ball queries.
## Retaining 1635 points in hv1 and 61641 points in hv2.
## Beginning ball queries...
## Building tree... done.
## Querying tree... 1.6223e-005  0.162246  0.324476  0.486705  0.648935  0.811165  0.973394
## Building tree... done.
## Querying tree... 0.000611621  done.
## Finished ball queries.
## Retaining 553 points in hv1 and 31427 points in hv2.
## Beginning ball queries...
## Building tree... done.
## Querying tree... 3.18198e-005  0.31823  0.636427  0.954625  done.
## Building tree... done.
## Querying tree... 0.00180832  done.
## Finished ball queries.
## Retaining 1232 points in hv1 and 61641 points in hv2.
## Beginning ball queries...
## Building tree... done.
## Querying tree... 1.6223e-005  0.162246  0.324476  0.486705  0.648935  0.811165  0.973394
## Building tree... done.
```

```
## Querying tree... 0.000811688  done.
## Finished ball queries.
## Retaining 416 points in hv1 and 31427 points in hv2.
## Beginning ball queries...
## Building tree... done.
## Querying tree... 3.18198e-005  0.31823  0.636427  0.954625  done.
## Building tree... done.
## Querying tree... 0.00240385  done.
## Finished ball queries.
## Retaining 20849 points in hv1 and 31427 points in hv2.
## Beginning ball queries...
## Building tree... done.
## Querying tree... 3.18198e-005  0.31823  0.636427  0.954625  done.
## Building tree... done.
## Querying tree... 4.79639e-005  0.479687  0.959327  done.
## Finished ball queries.
```


BodyL
WingL
TailL
BeakW
BeakH
LBeakL
UBeakL
N.UBkL
TarsusL



End of Hypervolume finch demo

`ComIndexMulti` takes the same arguments as `ComIndex` and an argument by factor to apply the index on different factors.

```
#all individual are put in the same group: calculate the hypervolume without by.factor
hv.1<-ComIndexMulti(traits.finch.mice,
             index = c("as.numeric(try(hypervolume(na.omit(x), reps = 100,
                 bandwidth = 0.2, verbose = F, warnings = F)@Volume))"),
             by.factor = rep(1,length(n_sp_plot)), nullmodels = "regional.ind",
             ind.plot = ind.plot.finch, nperm = 9, sp = sp.finch)

## [1] "creating null models"

## Error:  objet 's' introuvable
```

```
hv.2<-ComIndexMulti(traits.finch.mice,
             index = c("as.numeric(try(hypervolume(na.omit(x), reps = 100,
                 bandwidth = 0.2, verbose = F, warnings = F)@Volume))"),
             by.factor = n_sp_plot, nullmodels = "regional.ind",
             ind.plot = ind.plot.finch, nperm = 9, sp = sp.finch, print = FALSE)
```

```
## Error:  objet 's' introuvable
```

```
hv.3<-ComIndexMulti(traits.finch.mice,
             index = c("as.numeric(try(hypervolume(na.omit(x), reps = 100,
                 bandwidth = 0.2, verbose = F, warnings = F)@Volume))"),
             by.factor = ind.plot.finch, nullmodels ="regional.ind",
             ind.plot = ind.plot.finch, nperm = 9, sp = sp.finch, print = FALSE)
```

```
## Error:  objet 's' introuvable
```

```
hv.4<-ComIndexMulti(traits.finch.mice,
             index = c("as.numeric(try(hypervolume(na.omit(x), reps = 100,
                 bandwidth = 0.2, verbose = F, warnings = F)@Volume))"),
             by.factor = sp.finch, nullmodels = "regional.ind",
             ind.plot = ind.plot.finch, nperm = 9, sp = sp.finch, print = FALSE)
```

```
## Error:  objet 's' introuvable
```

```
list.ind.multi<-as.listofindex(list(hv.2, hv.3, hv.4))
```

```
## Error:  objet 'hv.2' introuvable
```

```
ses.list.multi<-ses.listofindex(list.ind.multi)
```

```
## Error:  objet 'list.ind.multi' introuvable
```

```
ses.list.multi
```

```
## Error:  objet 'ses.list.multi' introuvable
```

```
plot(list.ind.multi)
```

```
## Error:  erreur d'évaluation de l'argument 'x' lors de la sélection d'une méthode
pour la fonction 'plot' :  Erreur :  objet 'list.ind.multi' introuvable
```

```
#Try a zoom on the area near zero
plot(list.ind.multi, xlim = c(-200,20))
```

```
## Error:  erreur d'évaluation de l'argument 'x' lors de la sélection d'une méthode
pour la fonction 'plot' :  Erreur :  objet 'list.ind.multi' introuvable
```

Compare hypervolume to Villéger's metrics convex hull.

```
require("geometry")
```

```
## Loading required package:  geometry
## Loading required package:  magic
## Loading required package:  abind
```

```
FA<-as.character("FA")
funct<-c("round(convhulln(x,FA)$vol,6)")

##Null model local is trivial for this function
##because randomization is within community only
Fdis.finch<-ComIndexMulti(traits.finch.mice,
             index = funct,
             by.factor = ind.plot.finch, nullmodels = "local",
             ind.plot = ind.plot.finch, nperm = 9, sp = sp.finch)
```

```
## [1] "creating null models"
## [1] "local 25 %"
## [1] "local 50 %"
## [1] "local 75 %"
## [1] "local 100 %"
## [1] "calculation of null values using null models"
## [1] "round(convhulln(x,FA)$vol,6) 100 %"
## [1] "calculation of observed values"
## [1] "100 %"
```

```
list.ind.multi2<-as.listofindex(list(hv.3, Fdis.finch))
```

```
## Error:  objet 'hv.3' introuvable
```

```
ses.list.multi2<-ses.listofindex(list.ind.multi2)
```

```
## Error:  objet 'list.ind.multi2' introuvable
```

```
plot(list.ind.multi2)
```

```
## Error:  erreur d'évaluation de l'argument 'x' lors de la sélection d'une méthode
pour la fonction 'plot' :  Erreur :  objet 'list.ind.multi2' introuvable
```

6 Others graphical functions

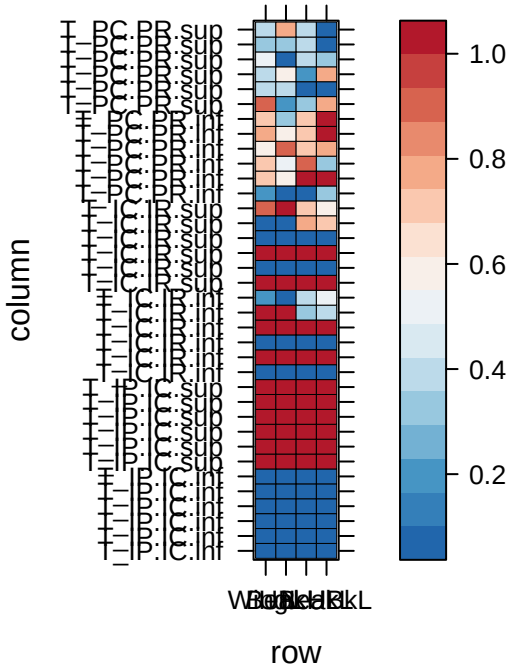
Use rasterVis to obtain more color schemes.

```
## Loading required package:  rasterVis
## Loading required package:  raster
## Loading required package:  sp
##
## Attaching package:  'raster'
##
```

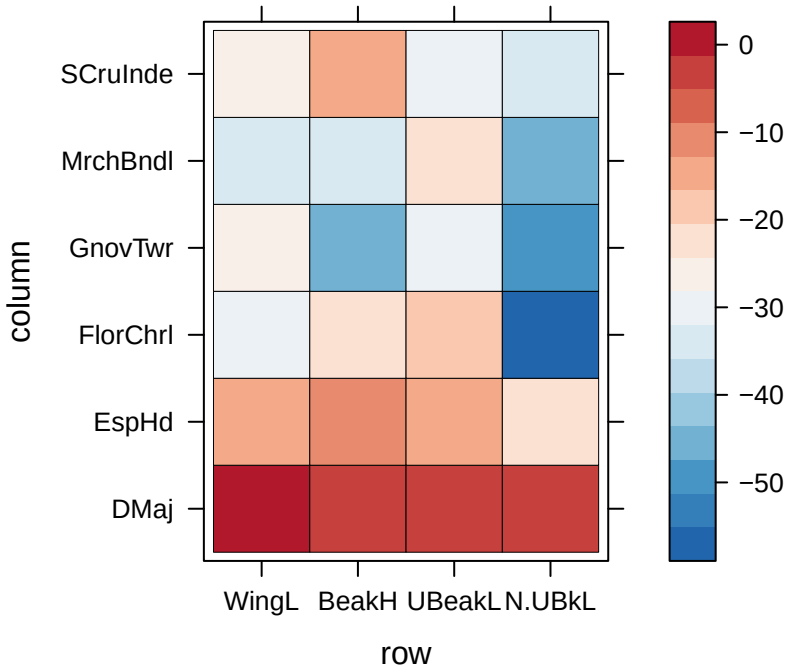
```
## The following object is masked from 'package:magic':
##
##     shift
##
## The following objects are masked from 'package:ape':
##
##     rotate, zoom
##
## The following object is masked from 'package:nlme':
##
##     getData
##
## Loading required package:  latticeExtra
## Loading required package:  RColorBrewer
## Loading required package:  hexbin
```

Plot the p-value or the ses values using the function `levelplot`.

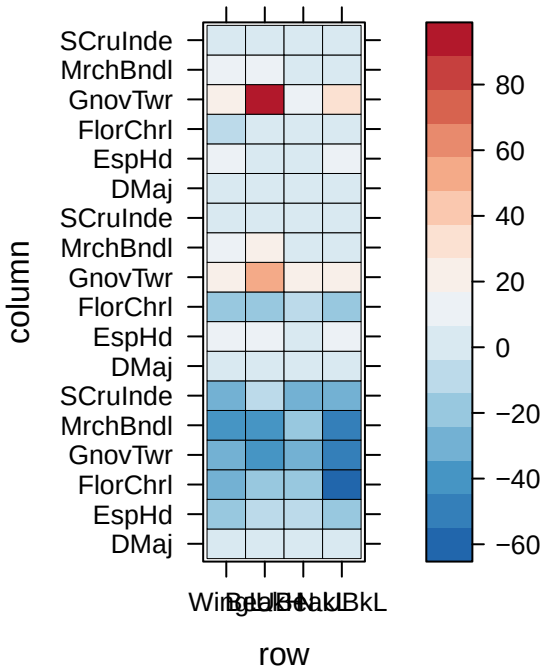
```
levelplot(t(sum_Tstats(res.finch)$p.value),
     colorkey = my.ckey, par.settings = my.theme,border = "black")
```




```
levelplot(t(ses(res.finch$Tstats$T_IP.IC, res.finch$Tstats$T_IP.IC_nm)$ses),
     colorkey = my.ckey, par.settings = my.theme,border = "black")
```




```
levelplot(cbind(t(ses(res.finch$Tstats$T_IP.IC, res.finch$Tstats$T_IP.IC_nm)$ses),
        t(ses(res.finch$Tstats$T_IC.IR, res.finch$Tstats$T_IP.IC_nm)$ses),
        t(ses(res.finch$Tstats$T_PC.PR, res.finch$Tstats$T_IP.IC_nm)$ses))
     , colorkey = my.ckey, par.settings = my.theme,border = "black")
```



Another example using `ses.listofindex`. The first plot represent "ses" values and the second one the result of a test with H0: observed index value are greater than what we can expect using the null model (alpha = 2.5%).

```
ses.list<-ses.listofindex(i.l1)

## Error:  objet 'i.l1' introuvable

levelplot(t(rbind(ses.list[[1]]$ses, ses.list[[2]]$ses,
         ses.list[[3]]$ses, ses.list[[4]]$ses)),
     colorkey = my.ckey, par.settings = my.theme,border = "black")

## Error:  erreur d'évaluation de l'argument 'x' lors de la sélection d'une
méthode pour la fonction 'levelplot' :  Erreur dans t(rbind(ses.list[[1]]$ses,
ses.list[[2]]$ses, ses.list[[3]]$ses,  :
##  erreur d'évaluation de l'argument 'x' lors de la sélection d'une méthode
pour la fonction 't' :  Erreur dans rbind(ses.list[[1]]$ses, ses.list[[2]]$ses,
ses.list[[3]]$ses,  :
##  objet 'ses.list' introuvable

levelplot(t(rbind(ses.list[[1]]$ses>ses.list[[1]]$ses.sup,
         ses.list[[2]]$ses>ses.list[[2]]$ses.sup,
         ses.list[[3]]$ses>ses.list[[3]]$ses.sup,
         ses.list[[4]]$ses>ses.list[[4]]$ses.sup)),
     colorkey = my.ckey, par.settings = my.theme,border = "black")

## Error:  erreur d'évaluation de l'argument 'x' lors de la sélection d'une
méthode pour la fonction 'levelplot' :  Erreur dans t(rbind(ses.list[[1]]$ses >
ses.list[[1]]$ses.sup, ses.list[[2]]$ses >  :
##  erreur d'évaluation de l'argument 'x' lors de la sélection d'une méthode pour
la fonction 't' :  Erreur dans rbind(ses.list[[1]]$ses > ses.list[[1]]$ses.sup,
ses.list[[2]]$ses >  :
##  objet 'ses.list' introuvable
```

Compare metrics calculate on individual against metrics calculate after populationnal meaning

```
ses.ind<-t(rbind(ses.list[[1]]$ses,
      ses.list[[2]]$ses,
      ses.list[[3]]$ses,
      ses.list[[4]]$ses))

## Error:  erreur d'évaluation de l'argument 'x' lors de la sélection d'une méthode
pour la fonction 't' :  Erreur dans rbind(ses.list[[1]]$ses, ses.list[[2]]$ses,
ses.list[[3]]$ses,  :
##  objet 'ses.list' introuvable

ses.sp<-t(rbind(ses.list[[5]]$ses,
     ses.list[[6]]$ses,
     ses.list[[7]]$ses,
     ses.list[[8]]$ses))

## Error:  erreur d'évaluation de l'argument 'x' lors de la sélection d'une méthode
pour la fonction 't' :  Erreur dans rbind(ses.list[[5]]$ses, ses.list[[6]]$ses,
ses.list[[7]]$ses,  :
##  objet 'ses.list' introuvable
```

```
levelplot(ses.ind, colorkey = my.ckey,
     par.settings = my.theme,border = "black")
```

```
## Error:  erreur d'évaluation de l'argument 'x' lors de la sélection d'une méthode
pour la fonction 'levelplot' :  Erreur :  objet 'ses.ind' introuvable
```

```
levelplot(ses.sp, colorkey = my.ckey,
     par.settings = my.theme,border = "black")
```

```
## Error:  erreur d'évaluation de l'argument 'x' lors de la sélection d'une méthode
pour la fonction 'levelplot' :  Erreur :  objet 'ses.sp' introuvable
```

7 Multivariate analysis of metrics

To finish, we can do a multivariate analysis of the metrics obtain during this tutorial using the package `ade4`. Analysis dudi 1 puts together all traits by meaning the SES values for each metrics in each sites whereas analysis dudi 2 analyses all combination of traits / sites / metrics.

```
library(ade4)

 matfordudi<-matrix(nrow = length(colMeans(ses.list[[1]]$ses)), ncol = length(names(ses.list)
```

```
## Error:  objet 'ses.list' introuvable
```

```
 for(i in 1: length(names(ses.list))){
  matfordudi[,i]<-colMeans(ses.list[[i]]$ses)
 }
```

```
## Error:  objet 'ses.list' introuvable
```

```
 colnames(matfordudi)<-names(ses.list)
```

```
## Error:  objet 'ses.list' introuvable
```

```
 rownames(matfordudi)<-colnames(traits.finch)
```

```
## Error:  objet 'matfordudi' introuvable
```

```
 matfordudi2<-matrix(nrow = length(as.vector(ses.list[[1]]$ses)), ncol = length(names(ses.lis
```

```
## Error:  erreur d'évaluation de l'argument 'x' lors de la sélection d'une méthode
pour la fonction 'as.vector' :  Erreur :  objet 'ses.list' introuvable
```

```
 for(i in 1: length(names(ses.list))){
  matfordudi2[,i]<-as.vector(ses.list[[i]]$ses)
 }
```

```
## Error:  objet 'ses.list' introuvable
```

```
 colnames(matfordudi2)<-names(ses.list)
```

```
## Error:  objet 'ses.list' introuvable
```

```
#Use mice for the purpose of this example
matfordudi<-complete(mice(matfordudi))

## Error:  objet 'matfordudi' introuvable

matfordudi2<-complete(mice(matfordudi2))

## Error:  objet 'matfordudi2' introuvable
```

```
res.dudi<-dudi.pca(t(matfordudi), scan = F, nf = 2)

## Error:  erreur d'évaluation de l'argument 'x' lors de la sélection d'une méthode
pour la fonction 't' :  Erreur :  objet 'matfordudi' introuvable

s.corcircle(res.dudi$co)

## Error:  objet 'res.dudi' introuvable

s.label(res.dudi$li, add.plot = T, clabel = 0, pch = 16)

## Error:  objet 'res.dudi' introuvable

s.label(res.dudi$li+0.05, add.plot = T, boxes = F)

## Error:  objet 'res.dudi' introuvable

res.dudi2<-dudi.pca(matfordudi2, scan = F, nf = 2)

## Error:  objet 'matfordudi2' introuvable

scatter(res.dudi2)

## Error:  objet 'res.dudi2' introuvable

s.corcircle(res.dudi2$co)

## Error:  objet 'res.dudi2' introuvable

s.class(res.dudi2$li, as.factor(c(rep("WingL",6), rep("BeakH",6),
                                  rep("UBeakL",6), rep("N.UBkL",6))),
        col = funky(4))

## Error:  objet 'res.dudi2' introuvable

s.class(res.dudi2$li, as.factor(rep(c("DMaj","EspHd","FlorChrl","GnovTwr",
                                      "MrchBndl","SCruInde"),4 )),
        col = funky(6))

## Error:  objet 'res.dudi2' introuvable
```

Conclusion

This is the end of the tutorial. The up to date version of this tutorial is available here.

References

Acknowledgment

Great thanks to Thibault Jombart for his help on building R packages and writing tutorial with knitR.

8 Conclusion

This is the end of the tutorial. The up to date version of this tutorial is available http://sourceforge.net/p/cati-r/code/ci/master/tree/tutorial/vignettes/vignette.pdf.

9 Acknowledgment

Great thanks to Thibault Jombart for his help on building R packages and writing tutorial with knitR.