

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

Adrien Taudiere*and Cyrille Viole

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

February 19, 2015

Abstract:

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

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

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

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

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 <code>cati</code>	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.	22
5	Test of community assembly	27
5.1	Ratio of variances: T-statistics	27
5.1.1	S3 methods for class <code>Tstats</code>	28
5.1.2	Use the standard errors of measure in the analysis (argument <code>SE</code>).	37
5.1.3	Plot T-statistics correlations	38
5.2	Others univariates or multivariates metrics: function <code>ComIndex</code> and <code>ComIndexMulti</code>	44
5.2.1	S3 methods for class <code>ComIndex</code> and <code>ComIndexMulti</code>	45
5.2.2	Plot <code>Tstats</code> and other uni/multivariates metrics together	49
5.3	More information on multivariates index	54
6	Others graphical functions	63
7	Multivariate analysis of metrics	68
8	Speed of computation	74
Conclusion	76
References	76

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 0.96. Make sure you have a recent version of R ($\geq 3.0.2$) by typing:

```
R.version.string  
  
## [1] "R version 3.1.2 (2014-10-31)"
```

Then, install `cati` with dependencies using:

```
#install.packages("cati", repos = "http://cran.us.r-project.org", dependencies = TRUE)  
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  
  
library("e1071")
```

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

```
packageDescription("cati", fields = "Version")  
  
## [1] "0.96"
```

`cati` version should read 0.96.

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 show 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 click 'cati', and 'cati-package'.

2.3 Data presentation: Darwin finches in Galapagos Island

First we need to load the data.

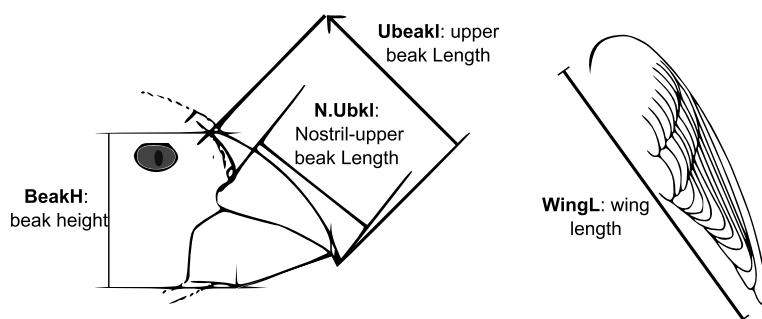
```
data(finch.ind)

#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 (Islands)
```

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

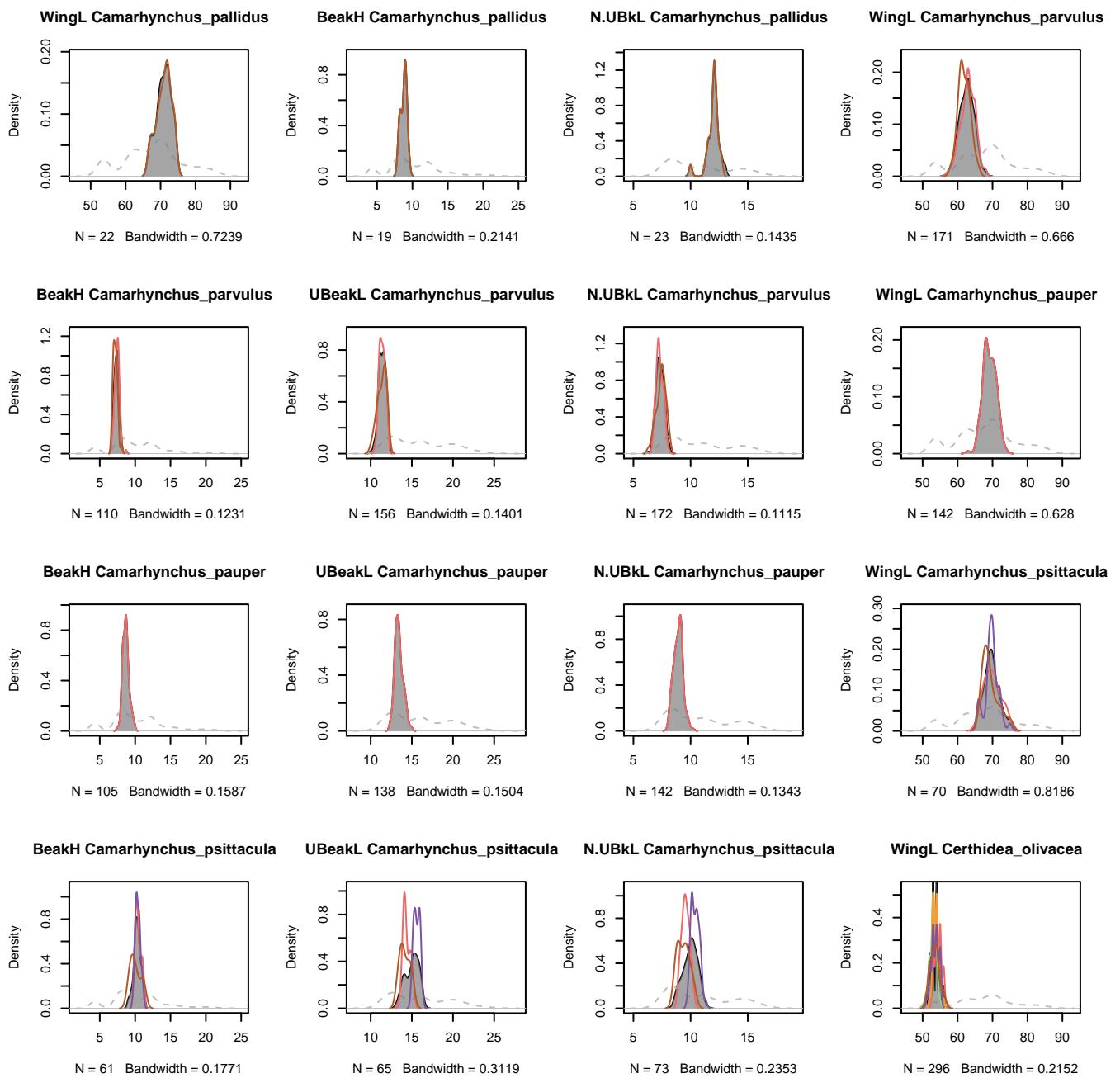


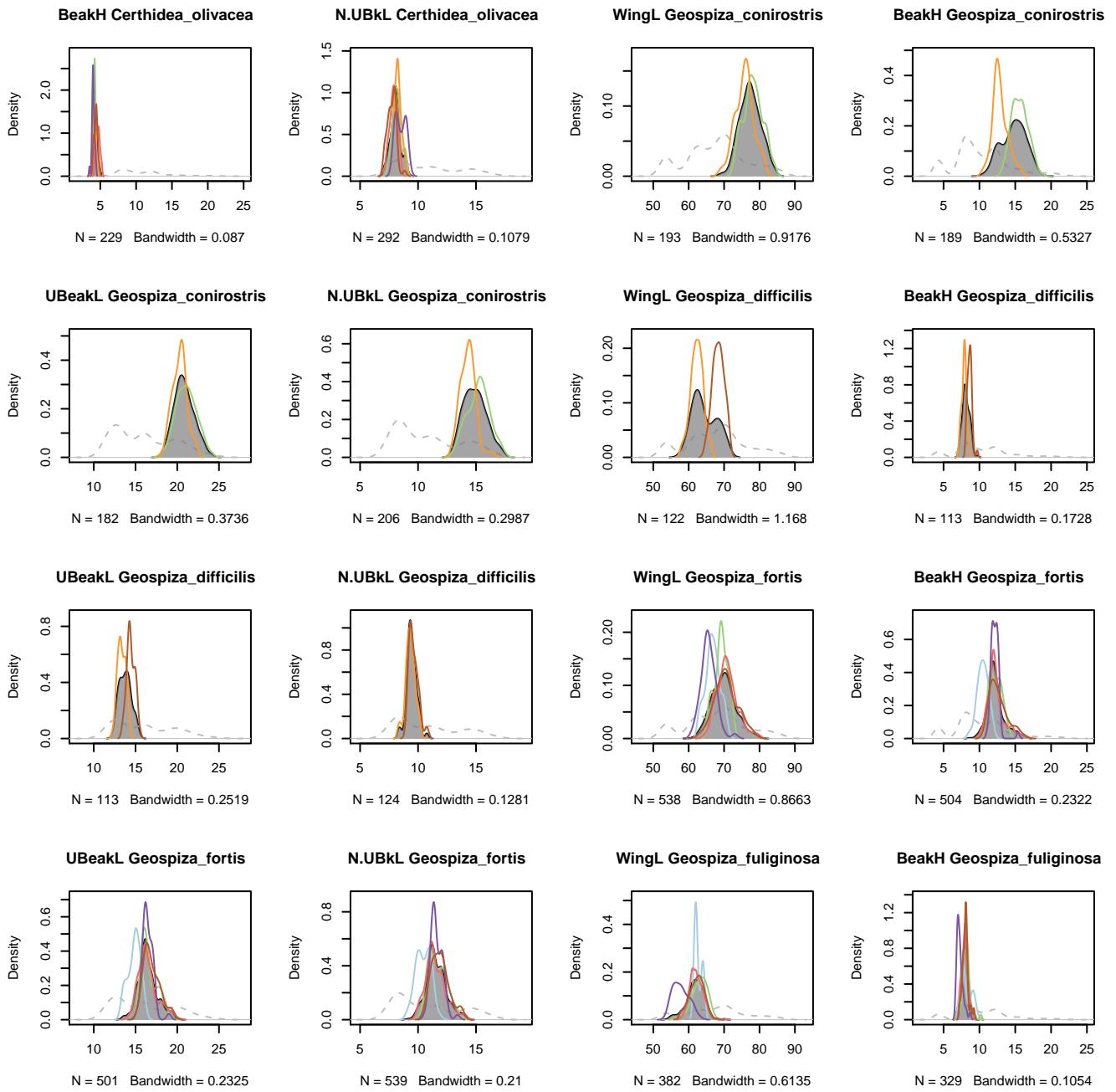
3 Description of traits distributions

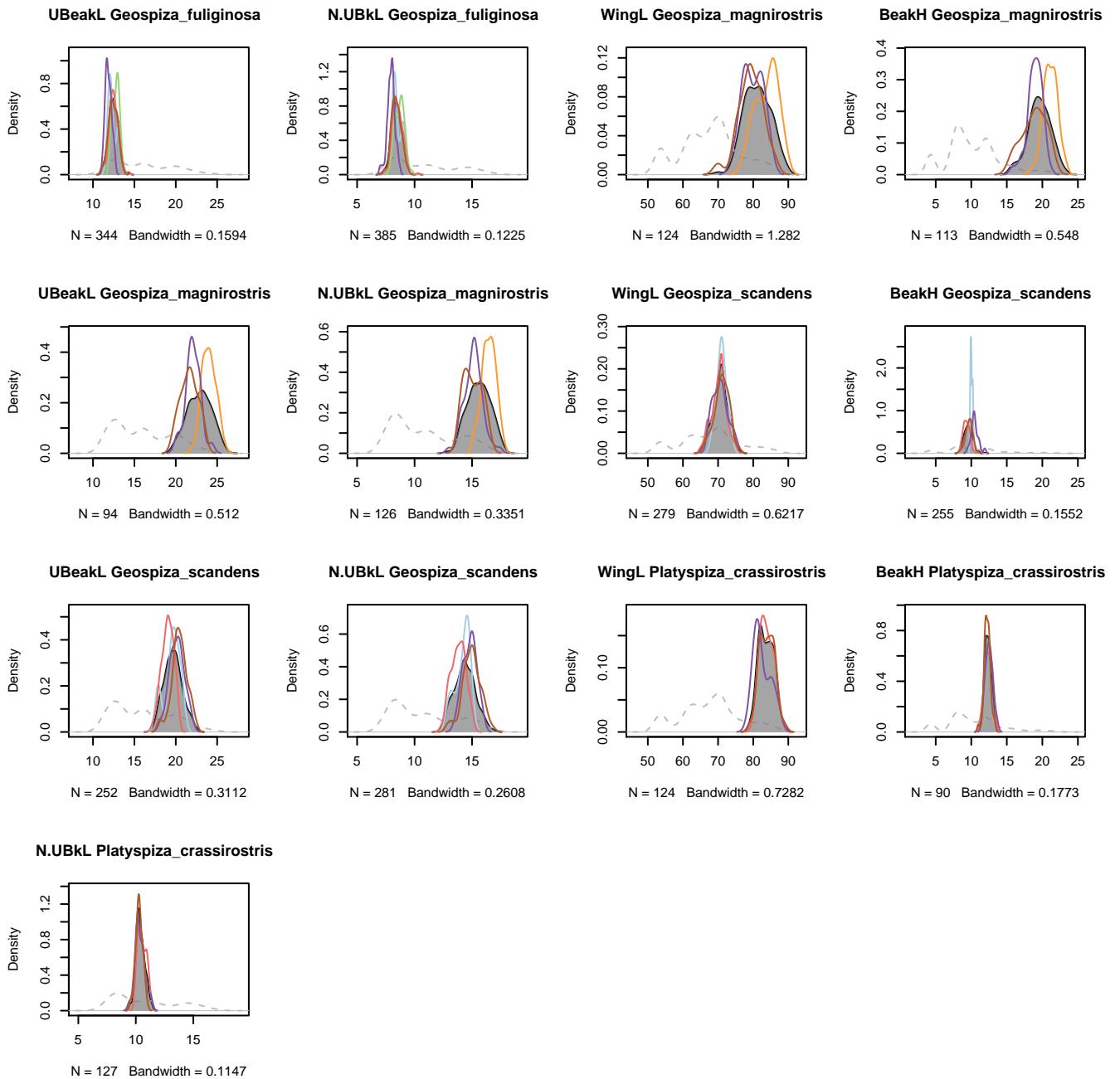
3.1 Plot the kernel density of traits

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)
```



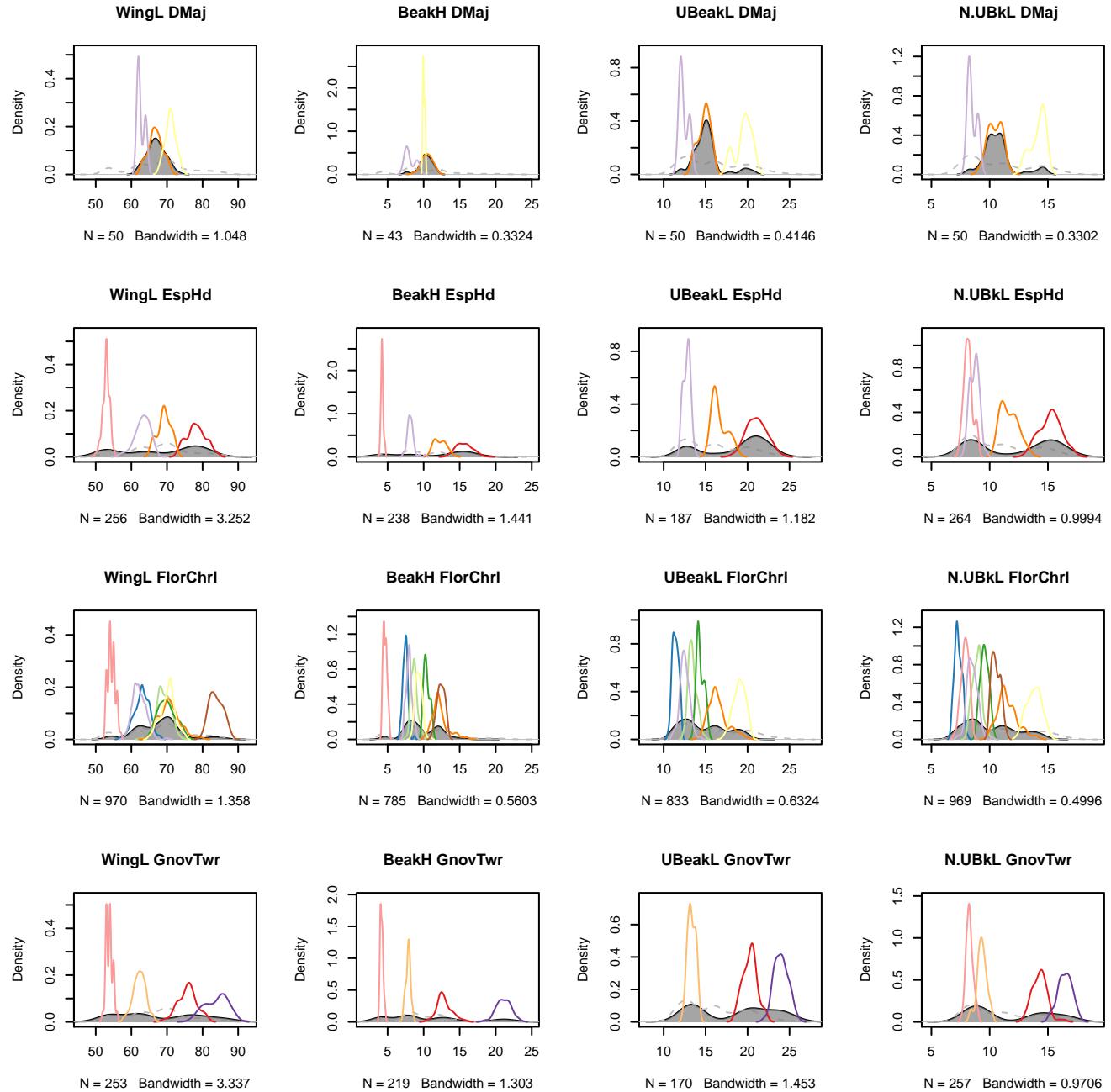


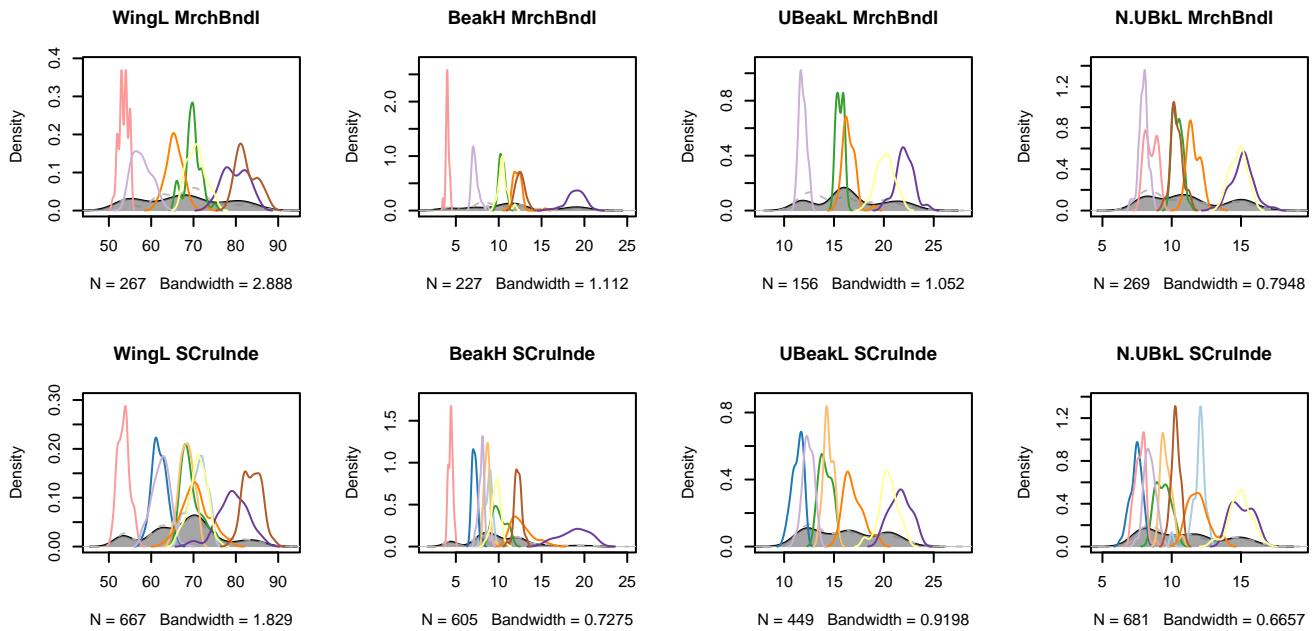


The argument `ylim.cex` define the magnification to be used for range of y axe. To understand the other argument, type `?plotDistri`.

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

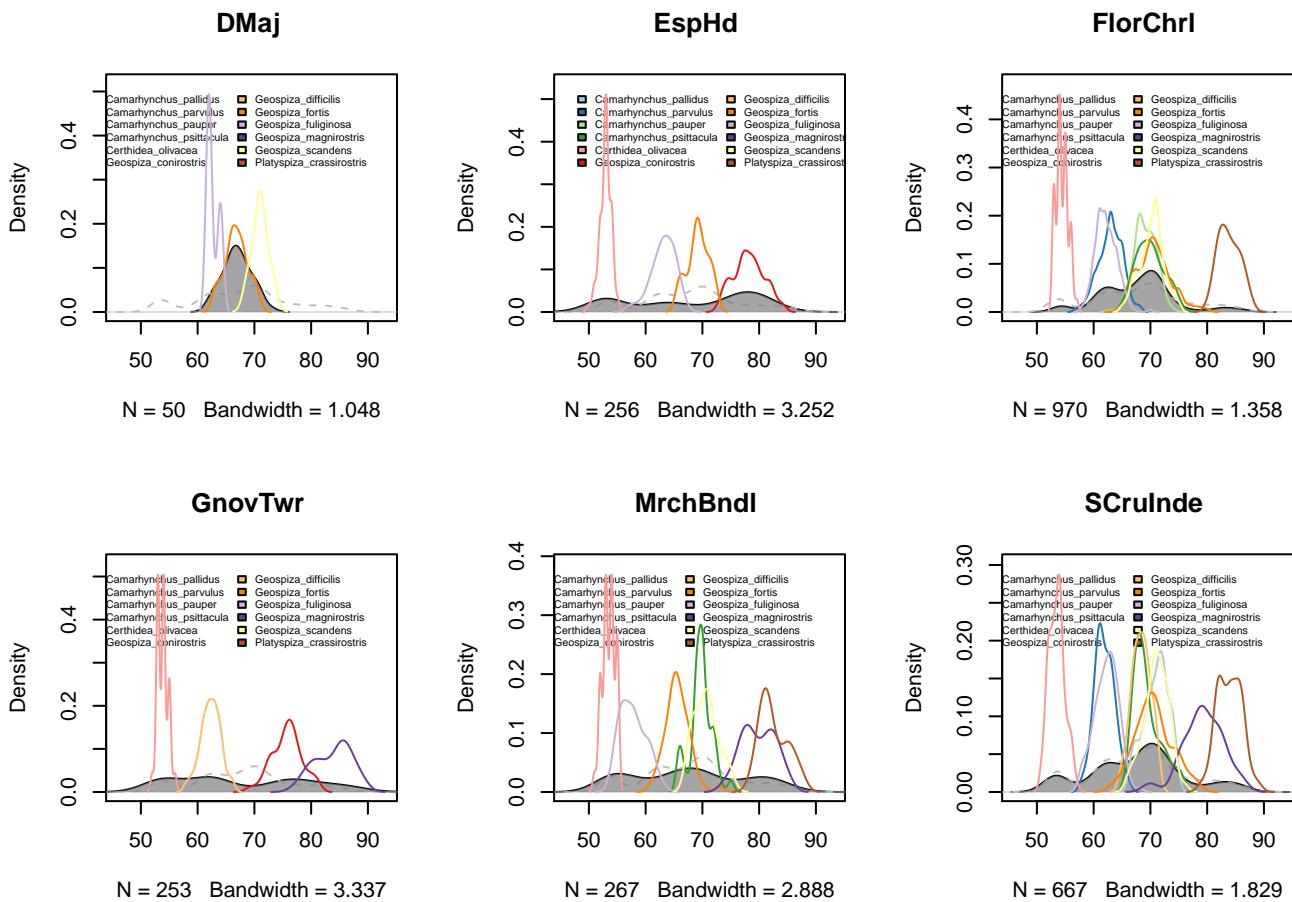
```
par(mfrow = c(4,4), cex = 0.5)
plotDistri(traits.finch, ind.plot.finch, sp.finch,
           ylim.cex = 8, plot.ask = F, multipanel = F, leg = F)
```





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

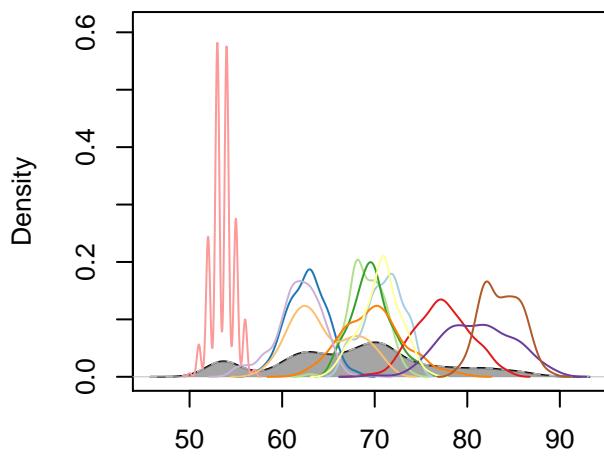
```
par(mfrow = c(2,3))
plotDistri(as.matrix(traits.finch[,1]), ind.plot.finch, sp.finch,
          ylim.cex = 8, plot.ask = F, multipanel = F, leg = T, cex.leg = 0.5)
```



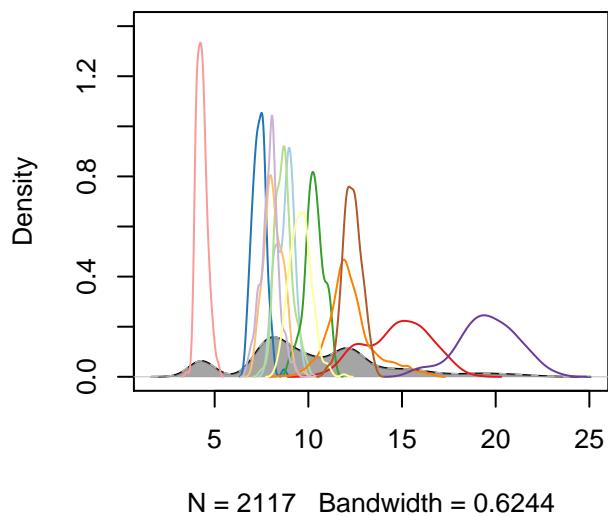
If we want to plot all the sites (regional distribution) or all the species: we can use the following code:

```
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)
```

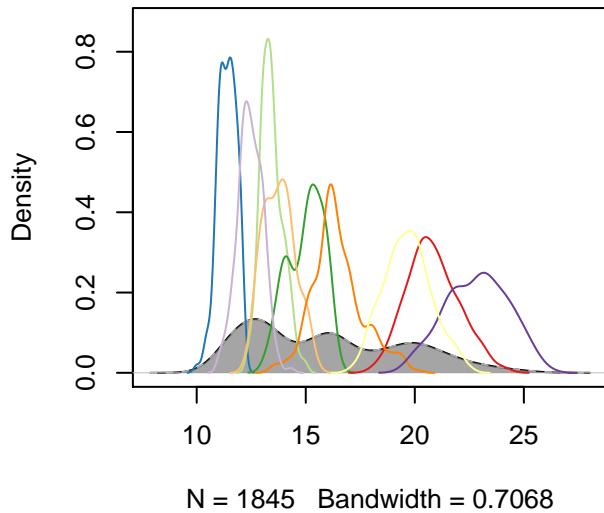
WingL region



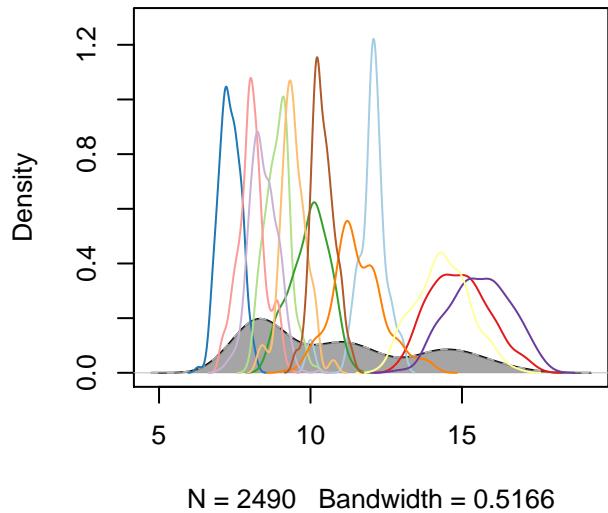
BeakH region



UBeakL region

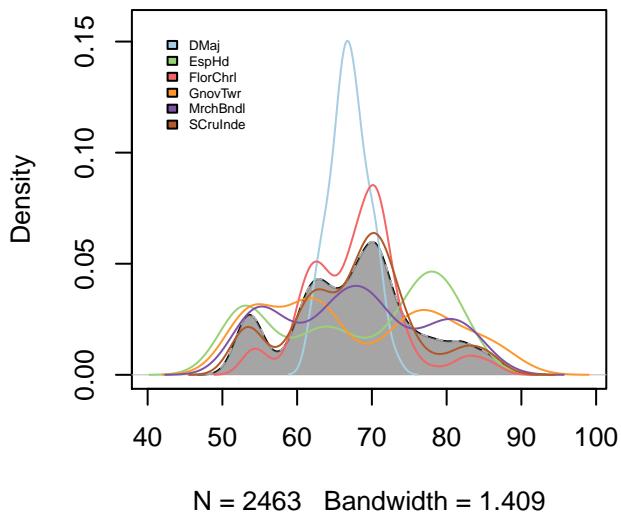


N.UBkL region

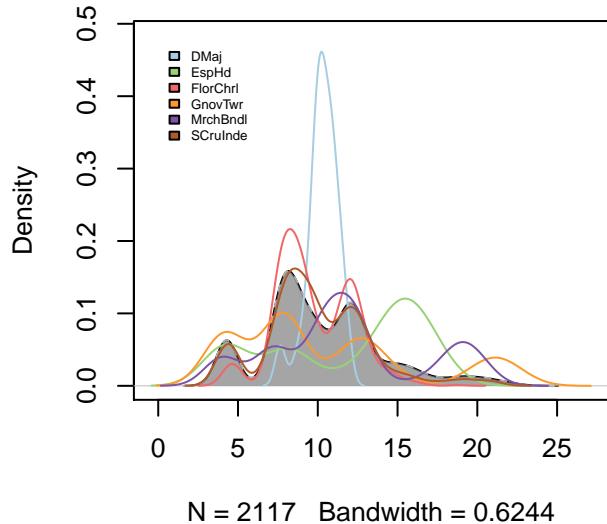


```
plotDistri(traits.finch, rep("all_sp", times = dim(traits.finch)[1]),
           ind.plot.finch, ylim.cex = 3, plot.ask = F, cex.leg = 0.5)
```

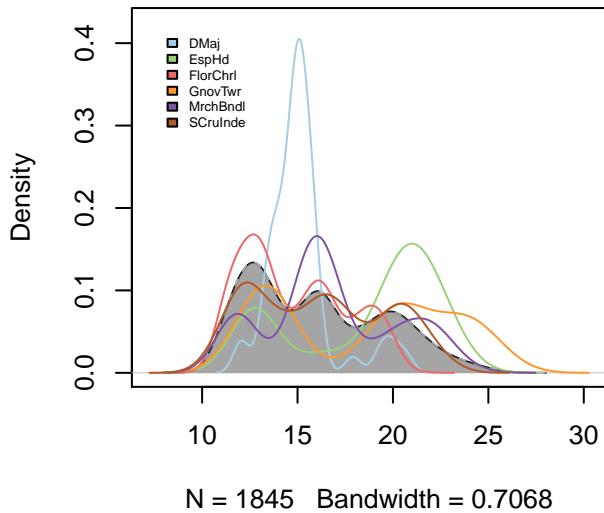
WingL all_sp



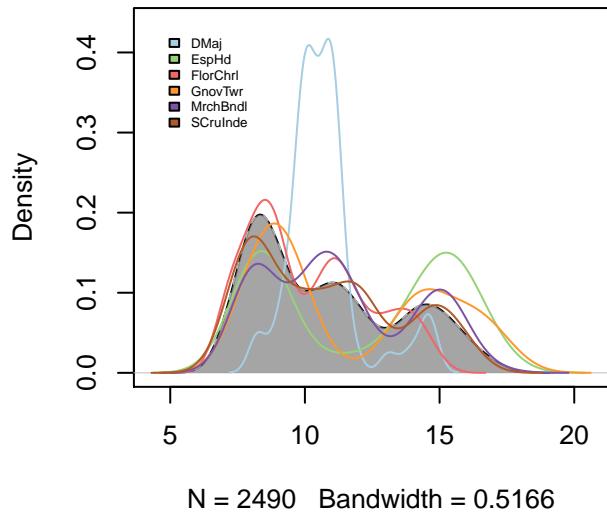
BeakH all_sp



UBeakL all_sp



N.UBkL all_sp



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:

$$\gamma = \text{mean}(\alpha) + \beta$$

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.

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

ptm <- proc.time()

res.rao<-RaoRel(sample = as.matrix(comm.sp),
                 dfunc = mat.dist, dphyl = NULL,
                 weight = F, Jost = F, structure = NULL)
proc.time_RaoRel <- proc.time() - ptm

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] 8.369581

totRao
```

```
## [1] 8.369581
```

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

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

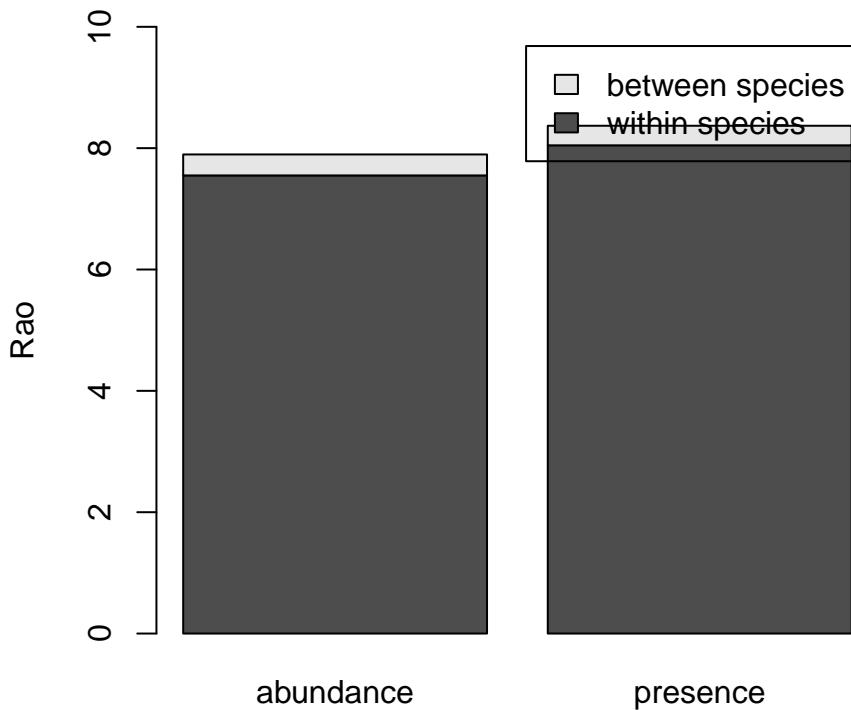
## [1] 7.550591

betRao.w

## [1] 0.3457724
```

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)
}

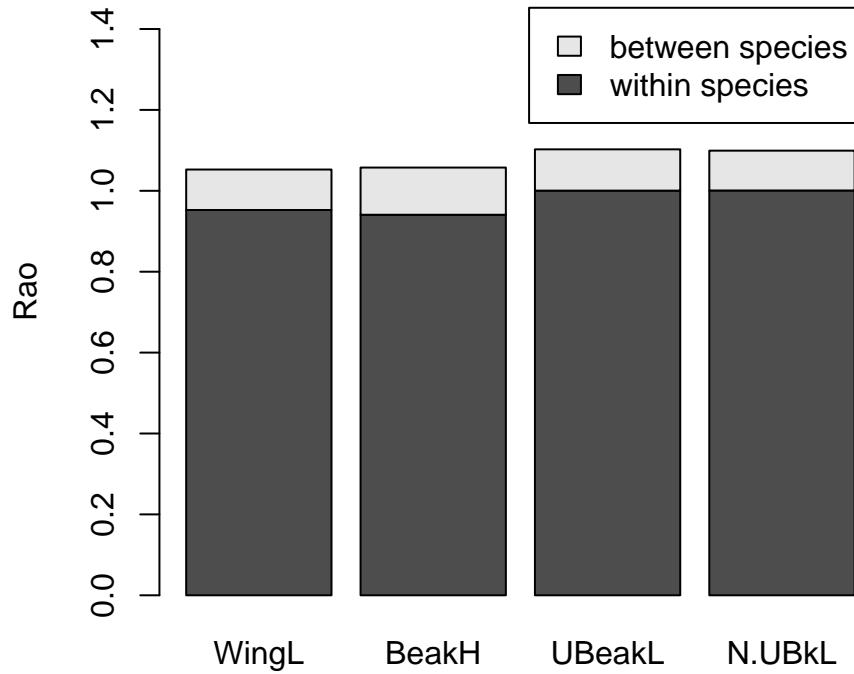
```

Plot the results by traits.

```

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

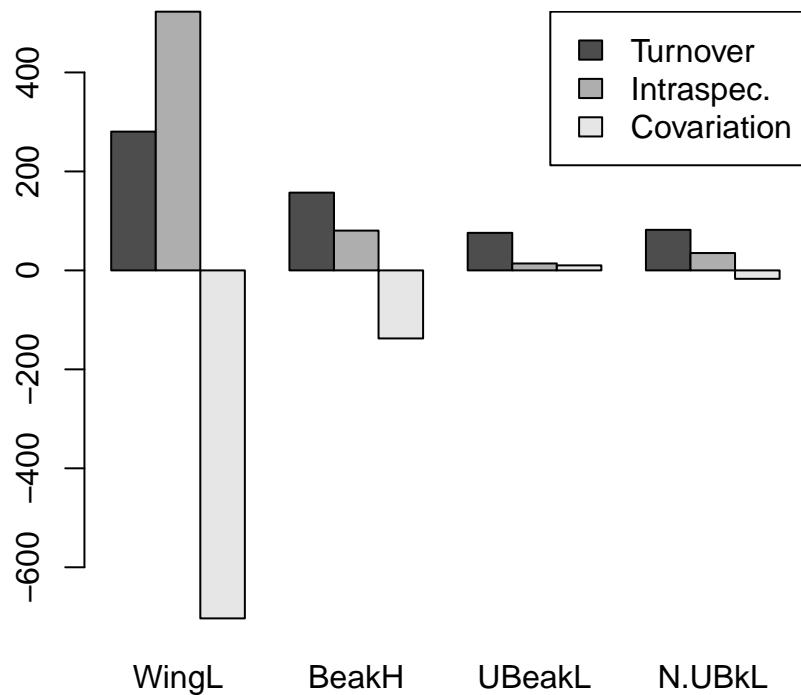
```



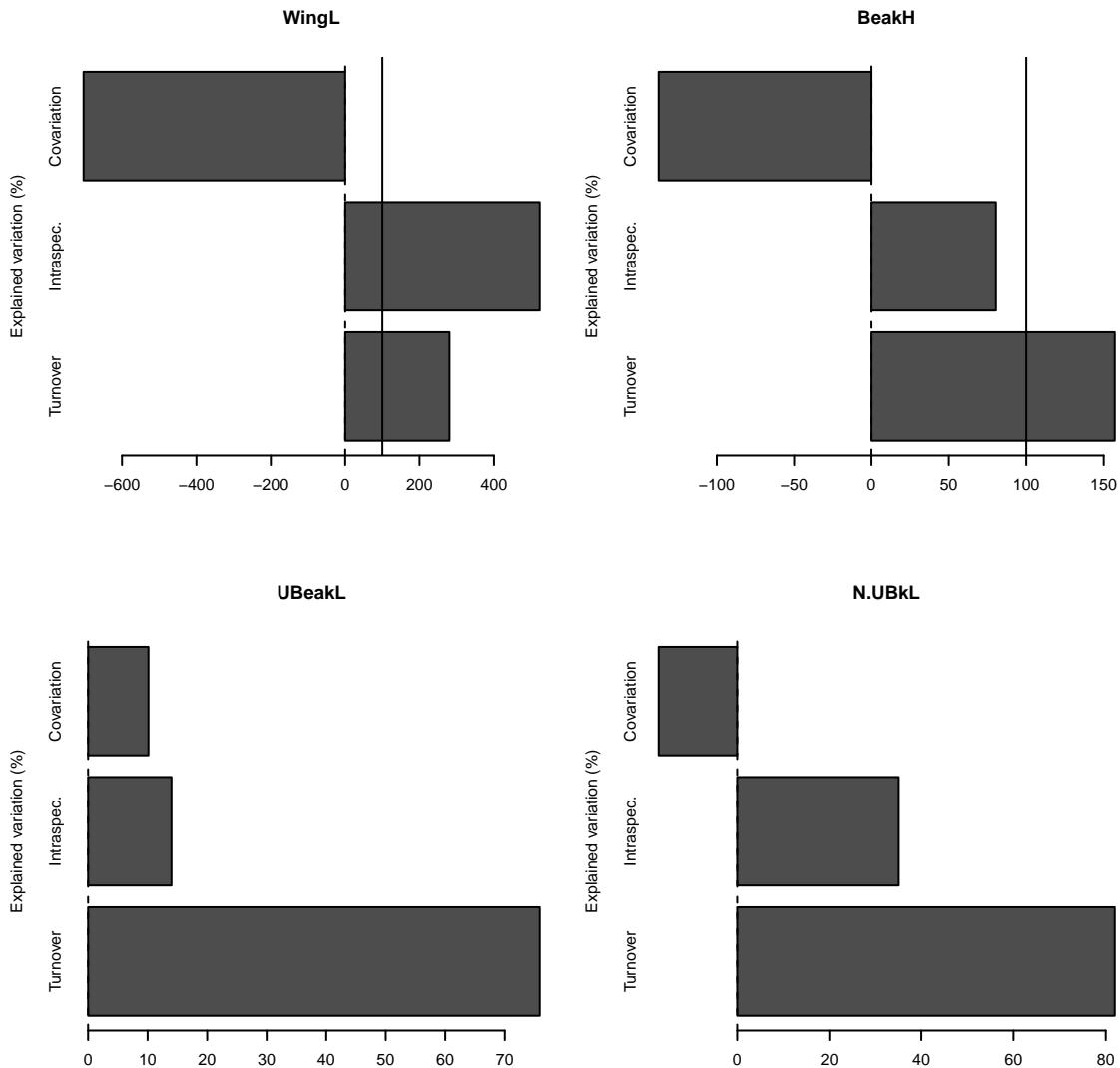
4.2 Decomposition of community trait response to environment into intraspecific trait variability, variability due to species turnover and their covariation.

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.

```
proc.time_decompCTRE <- proc.time() - ptm  
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)))

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

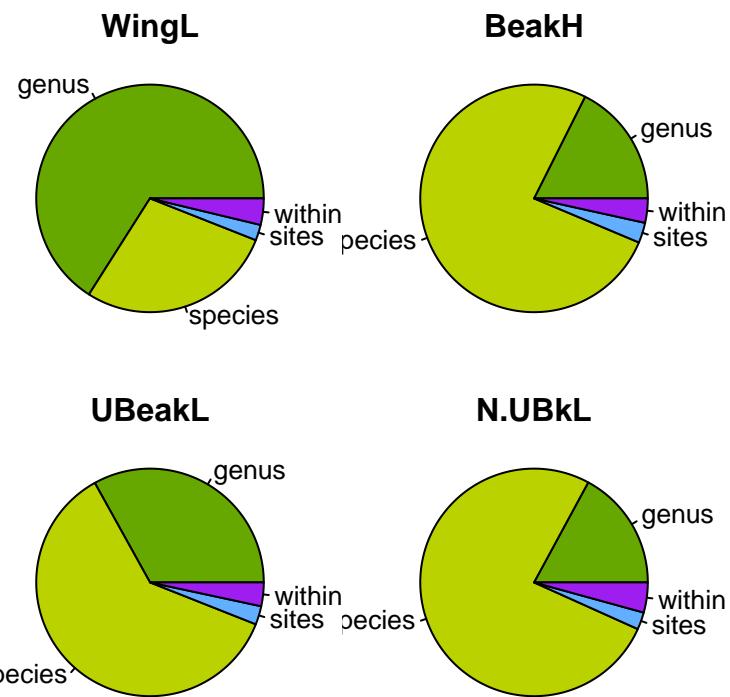
## The partvar function decompose the variance accross nested scales. Thus choose
## the order of the factors very carefully!

proc.time_partvar <- proc.time() - ptm

res.partvar.finch
```

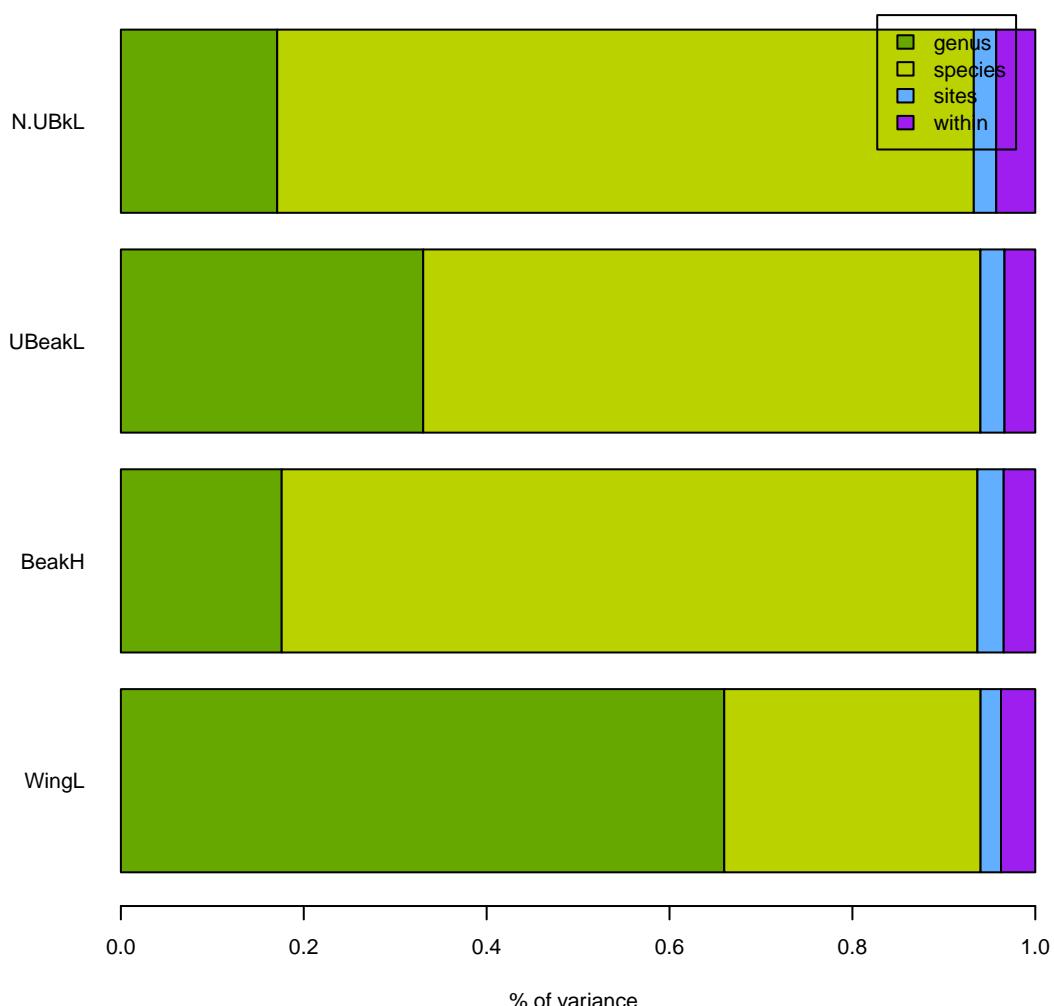
```
par(mfrow = c(2,2), mai = c(0.2,0.2,0.2,0.2)) #save graphical parameters
colors<-c(rgb(102,167,0, maxColorValue = 255),
          rgb(185,210,0, maxColorValue = 255),
          rgb(98,174,255, maxColorValue = 255),
          rgb(158,30,240, maxColorValue = 255))

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



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

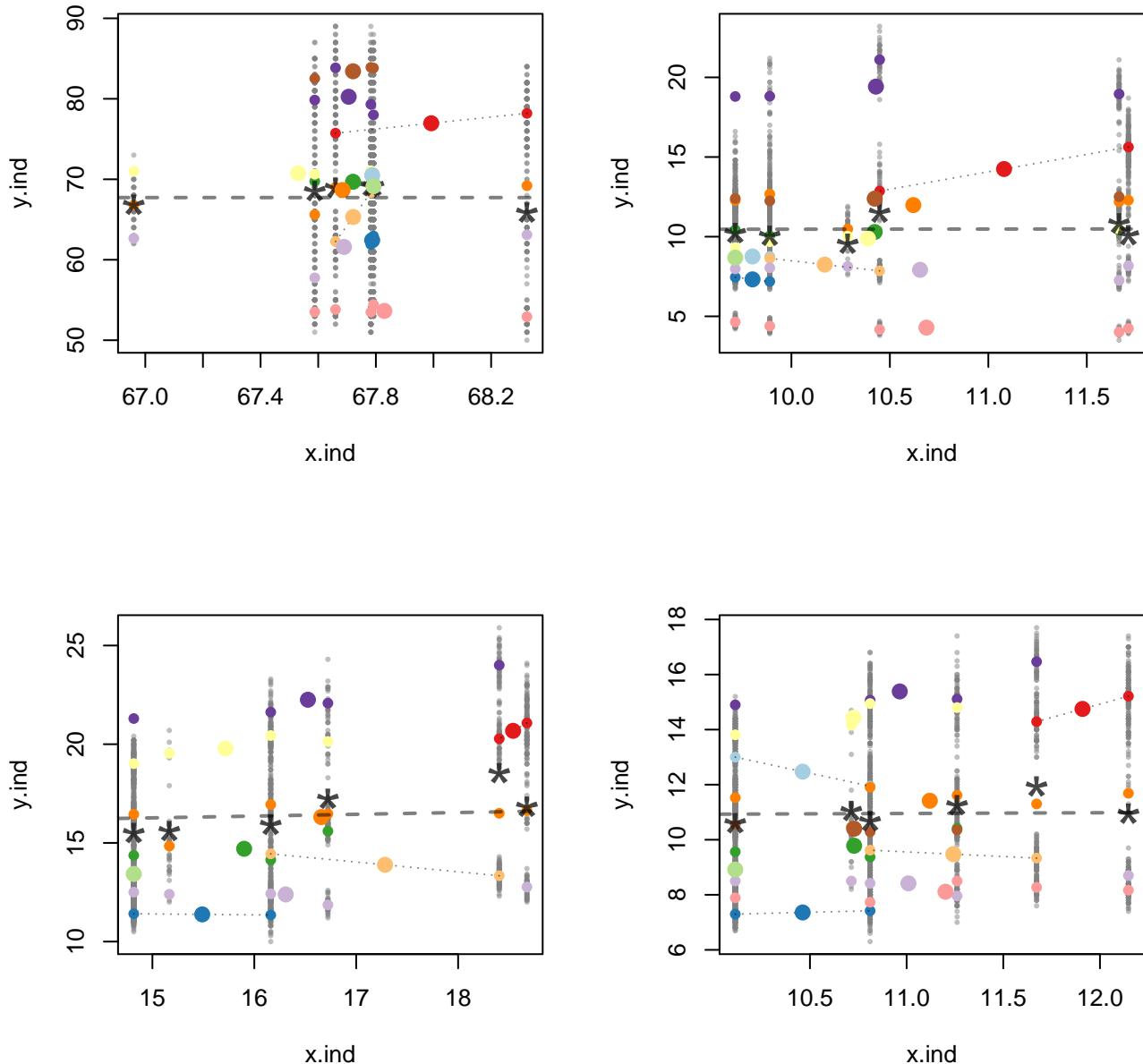
```
barPartvar(res.partvar.finch, col = colors, leg = TRUE)
```



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

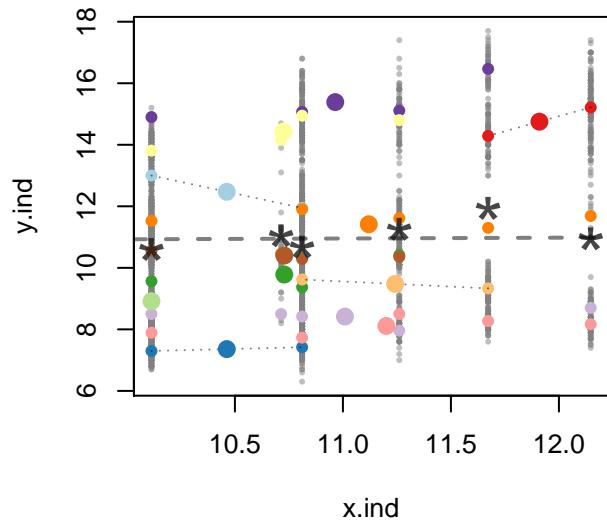
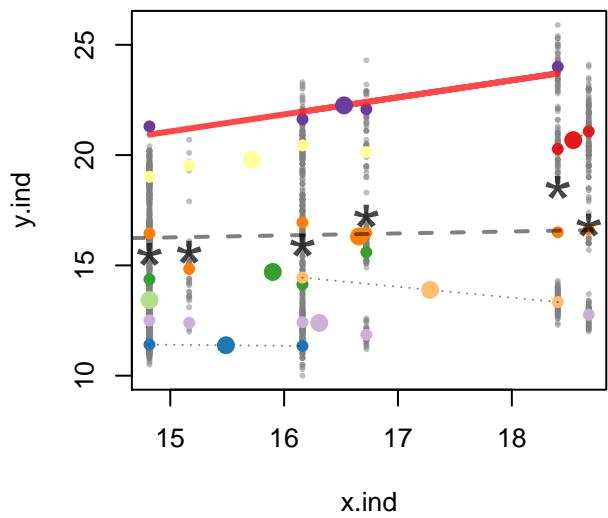
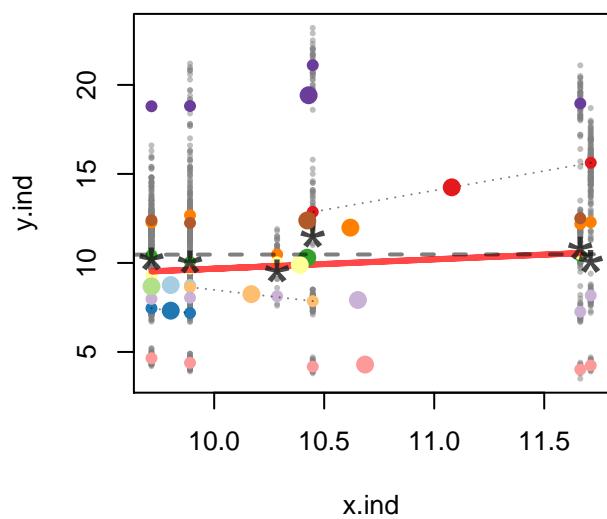
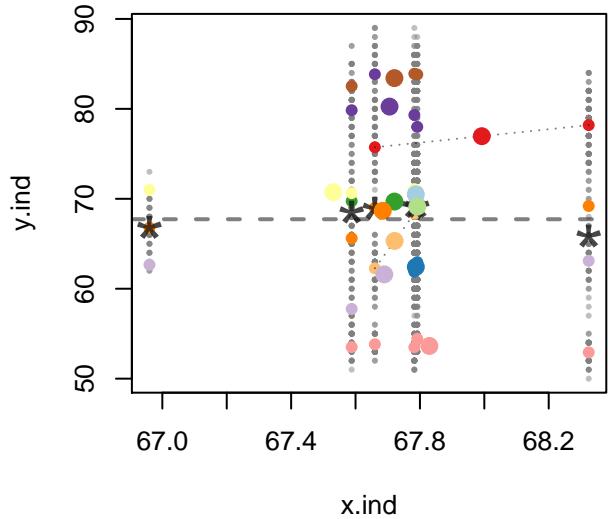
For an example of utilisation see: Cornwell, W.K. and Ackerly, D.D., 2009. Community assembly and shifts in plant trait distributions across an environmental gradient in coastal California. Ecological Monographs, 79, 109-126.

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



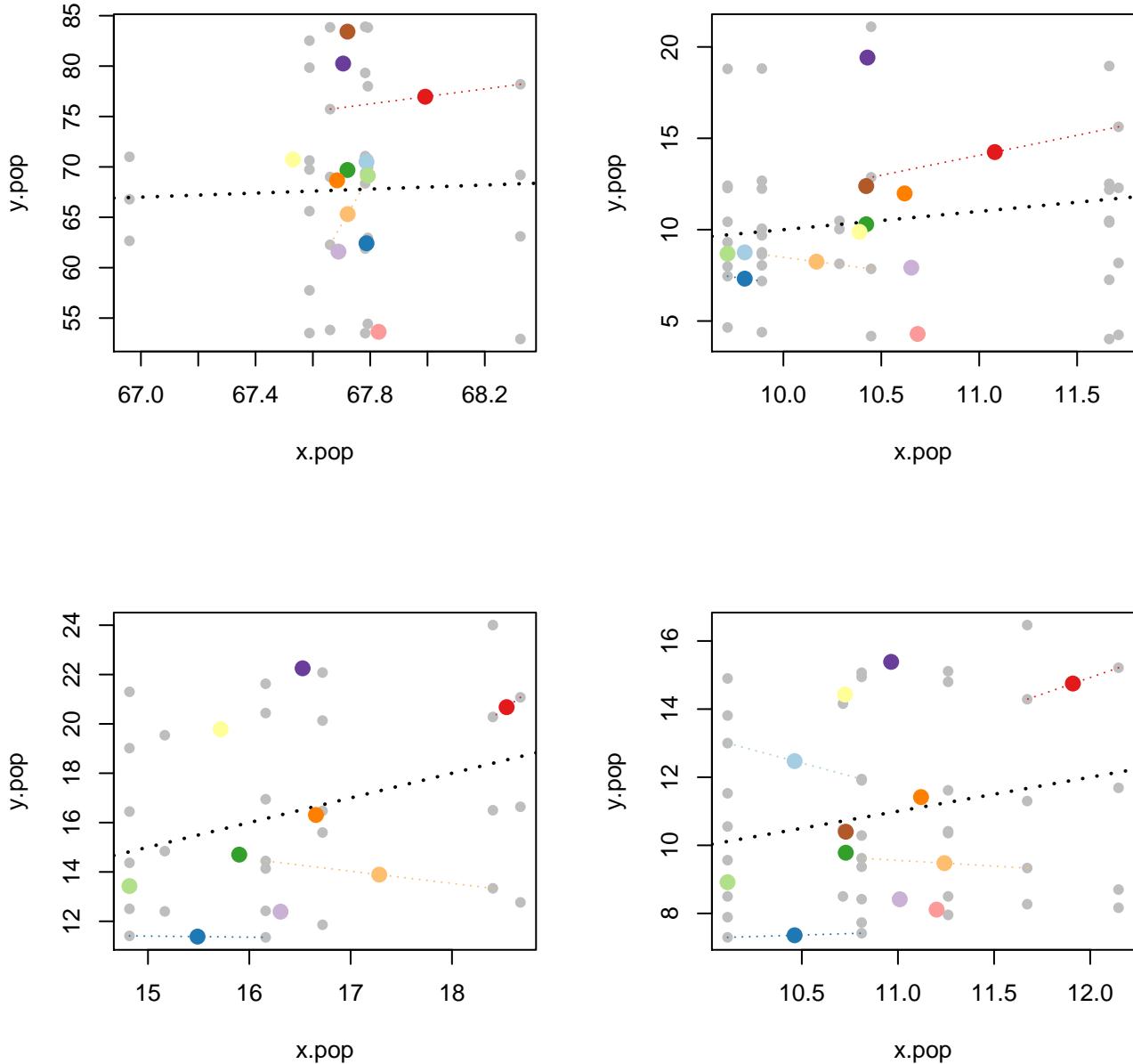
If we change the value of two arguments we can see some significant relationships. Here let's try a more permissive threshold: `alpha = 10%` instead of `5%` (`p.val`) and define a lower minimum of values to represent significance fixed to 3 instead of 10 by default (`min.ind.signif`).

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



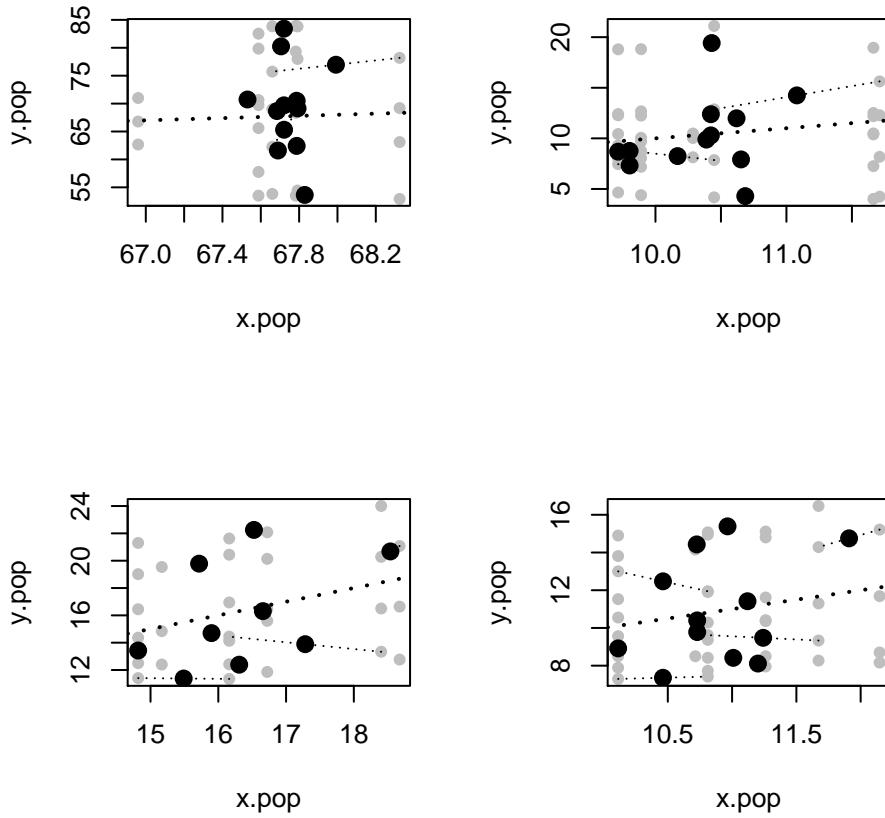
For a more simple figure, add the option resume = TRUE.

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



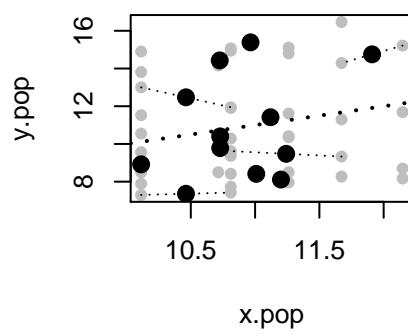
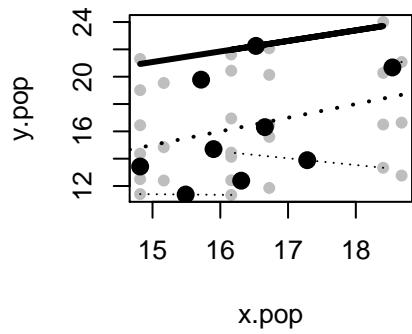
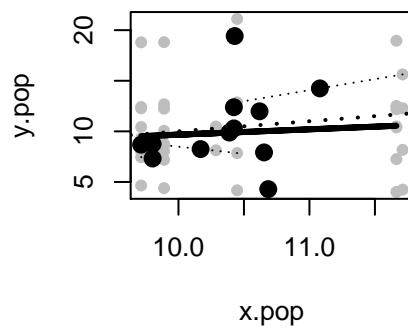
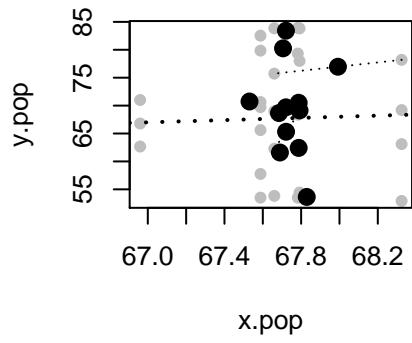
If you are fed up with colors, try this:

```
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)
```



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 = 99, print = FALSE)
res.finch

## #####
## # T-statistiques #
## #####
## class: Tstats
## $call: Tstats(traits = traits.finch, ind.plot = ind.plot.finch, sp = sp.finch,
##               nperm = 99, 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: 99
## $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

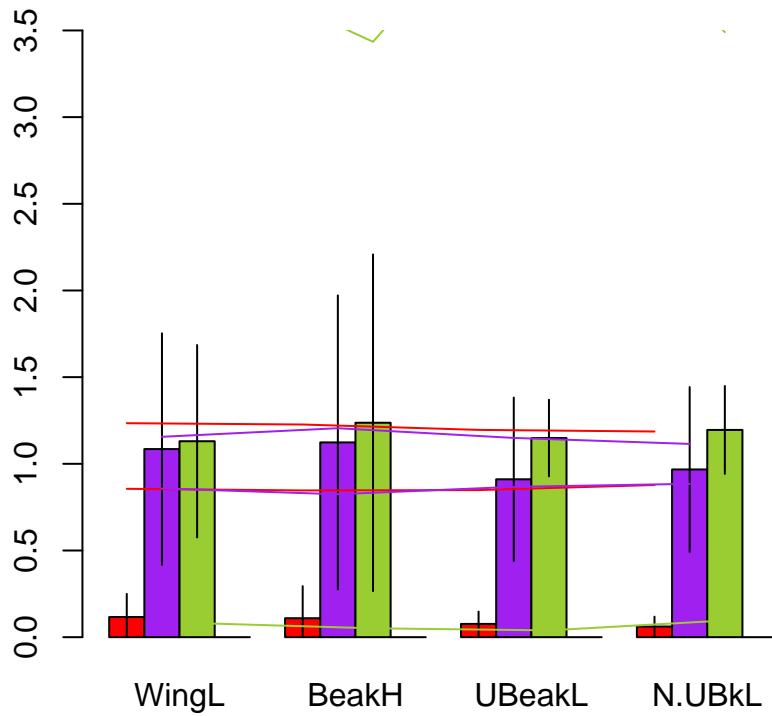
```

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))
```

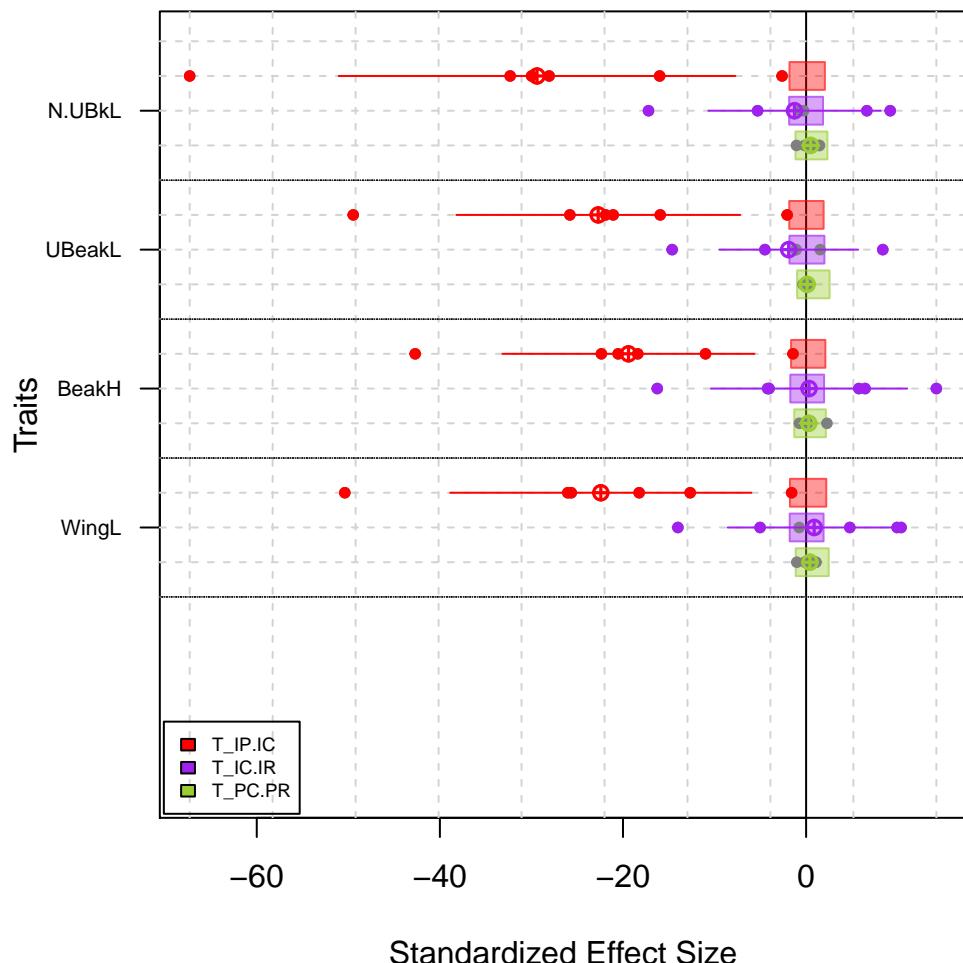


One can be more interested in the significance and the effect size available thanks to null model. In that case, the standardized effect size can be easily plot. Note that the function `ses` can be use directly to calculate standardized effect size without plotting. The Standardized Effect Size (ses)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

$$SES = (I_{obs} - I_{sim}) / \delta_{sim}$$

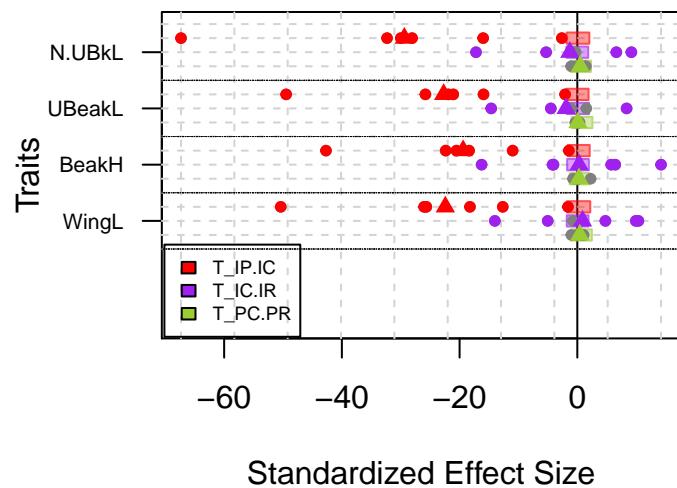
where I_{obs} is the observed value, I_{sim} the mean of values calculated from the null model and δ_{sim} the standard deviation of these simulated values.

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

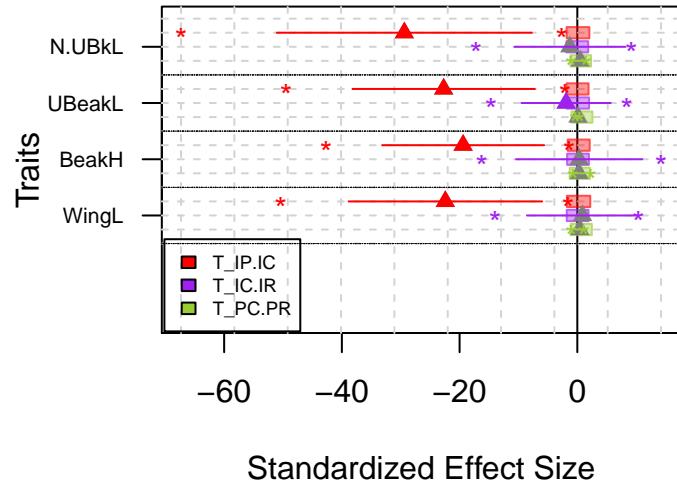


There is multiple kind of representation available.

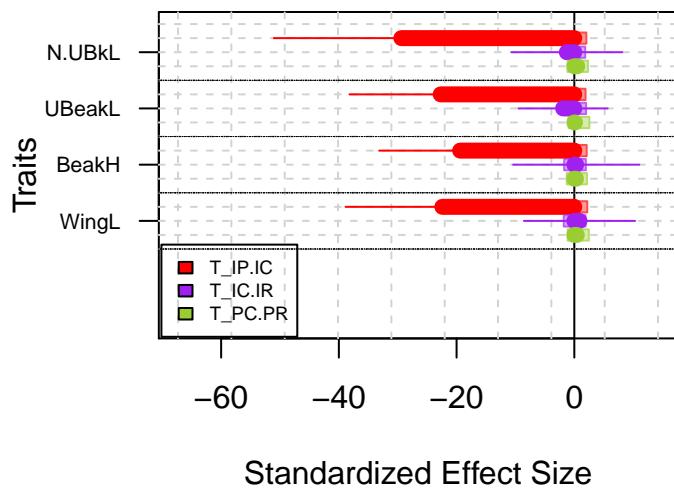
```
plot(res.finch, type = "simple")
```



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



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

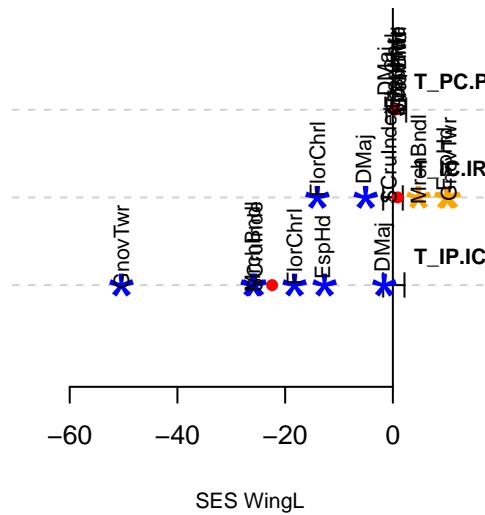


If you want to specifically look at traits or sites statistics, use the argument `type = "bytraits"` or `"bysites"`.

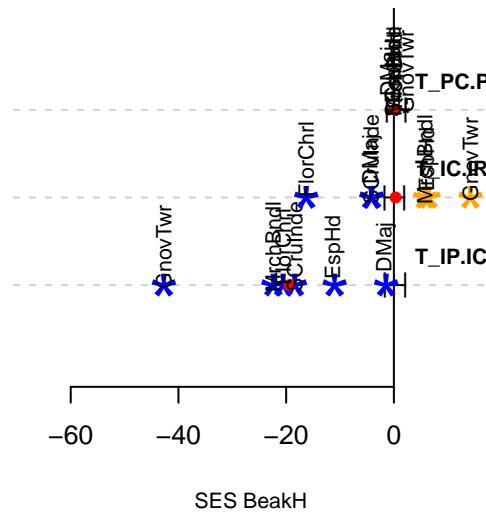
```
par(mfrow=c(2,2))

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

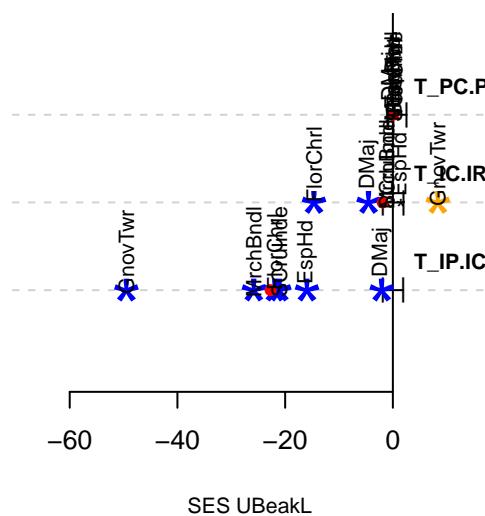
(1:nindex)[i]



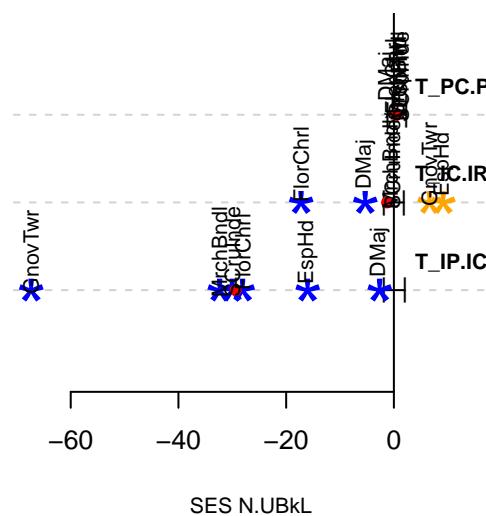
(1:nindex)[i]



(1:nindex)[i]

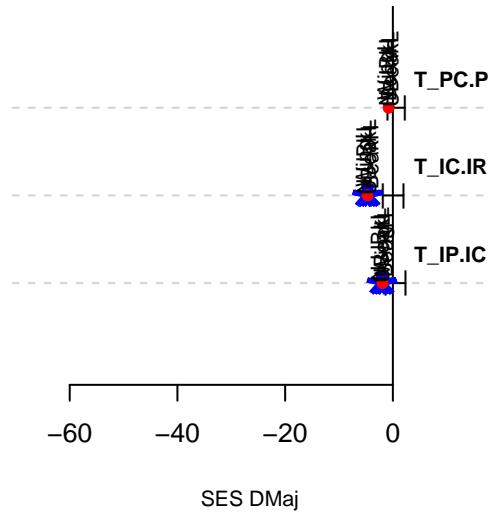


(1:nindex)[i]

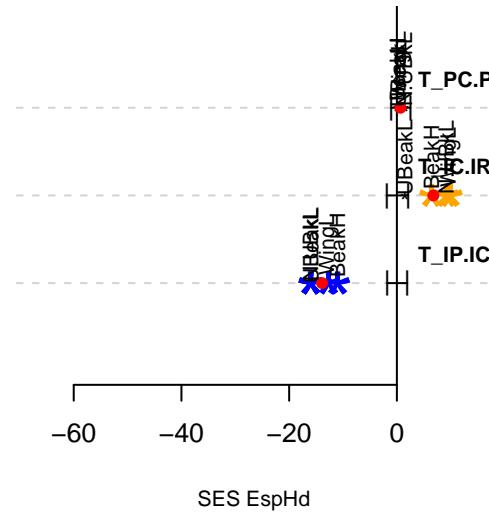


```
plot(res.finches, type = "bysites")
```

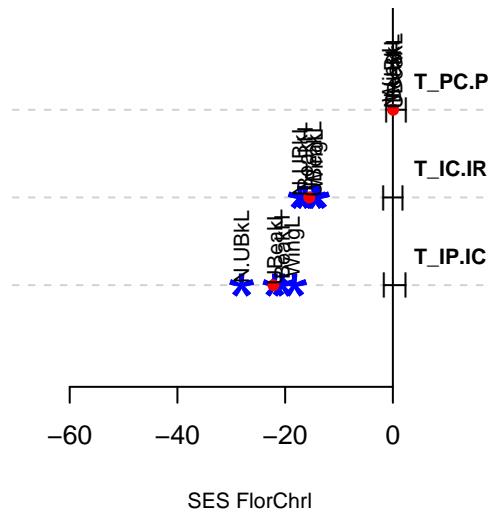
(1:nindex)[1]



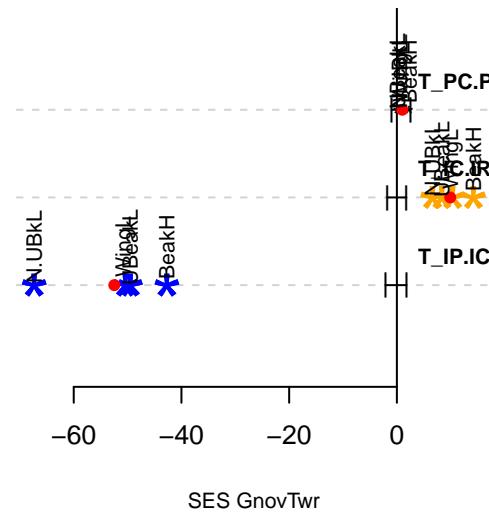
(1:nindex)[1]



(1:nindex)[1]

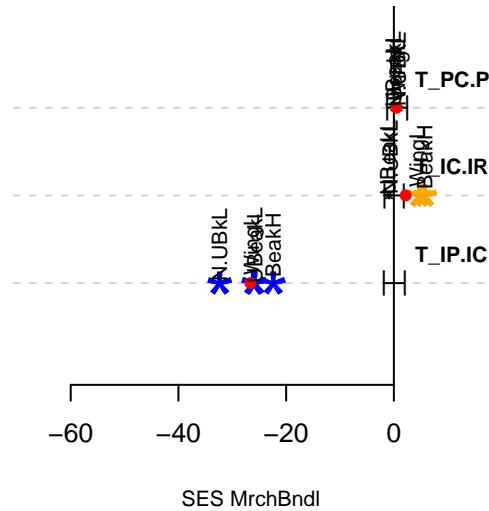


(1:nindex)[1]

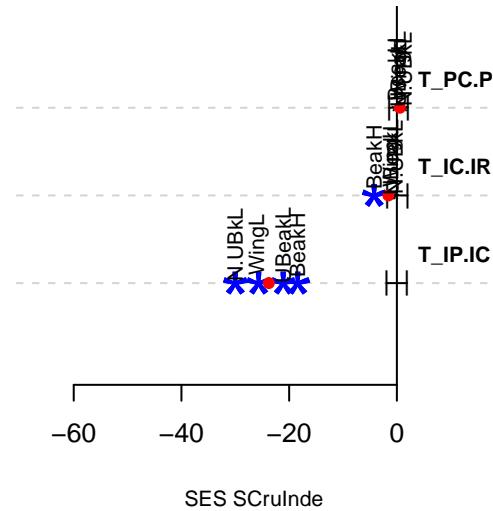


```
par(old.par) # reset default graphical parameters
```

(1:nindex)[1]



(1:nindex)[1]



```

summary(res.finches) #S3 summary method for class Tstats

## [1] "Observed values"
## $T_IP.IC
##      WingL          BeakH          UBeakL        N.UBkL
##  Min.   :0.03435   Min.   :0.01528   Min.   :0.02669   Min.   :0.02385
##  1st Qu.:0.04365  1st Qu.:0.01907  1st Qu.:0.04170  1st Qu.:0.03670
##  Median :0.06448  Median :0.04003  Median :0.05445  Median :0.04035
##  Mean   :0.11608  Mean   :0.10944  Mean   :0.07640  Mean   :0.06147
##  3rd Qu.:0.10115 3rd Qu.:0.05802  3rd Qu.:0.06289  3rd Qu.:0.04939
##  Max.   :0.38311  Max.   :0.48519  Max.   :0.21962  Max.   :0.17637
##
## $T_IC.IR
##      WingL          BeakH          UBeakL        N.UBkL
##  Min.   :0.09246   Min.   :0.06316   Min.   :0.2461   Min.   :0.2569
##  1st Qu.:0.67385  1st Qu.:0.49130  1st Qu.:0.6684  1st Qu.:0.7238
##  Median :1.17073  Median :1.18311  Median :0.9442  Median :0.9800
##  Mean   :1.08518  Mean   :1.12359  Mean   :0.9109  Mean   :0.9677
##  3rd Qu.:1.62569 3rd Qu.:1.62423  3rd Qu.:1.0803  3rd Qu.:1.3144
##  Max.   :1.79158  Max.   :2.28021  Max.   :1.6287  Max.   :1.5250
##
## $T_PC.PR
##      WingL          BeakH          UBeakL        N.UBkL
##  Min.   :0.2261   Min.   :0.08685   Min.   :0.9328   Min.   :0.9356
##  1st Qu.:0.8981  1st Qu.:0.86252  1st Qu.:0.9828  1st Qu.:1.0388
##  Median :1.2233  Median :1.05894  Median :1.1249  Median :1.0992
##  Mean   :1.1303  Mean   :1.23699  Mean   :1.1488  Mean   :1.1955
##  3rd Qu.:1.4737  3rd Qu.:1.32864  3rd Qu.:1.2148  3rd Qu.:1.3513
##  Max.   :1.7622  Max.   :3.00163  Max.   :1.5301  Max.   :1.5852
##
## [1] "null values"
## $T_IP.IC_nm
##      Min. 1st Qu. Median   Mean 3rd Qu.   Max.
##  0.3859  0.9662  0.9987  0.9997  1.0270  2.6290
##
## $T_IC.IR_nm
##      Min. 1st Qu. Median   Mean 3rd Qu.   Max.
##  0.4624  0.9596  0.9992  1.0020  1.0420  1.5160
##
## $T_PC.PR_nm
##      Min. 1st Qu. Median   Mean 3rd Qu.   Max.    NA's
##  0.0000  0.3473  0.7480  1.1690  1.4250 11.6300     189

```

```

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

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

```

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

##          WingL BeakH UBeakL N.UBkL
## T_IP.IC.inf 0.01  0.03   0.01   0.01
## T_IP.IC.inf 0.01  0.01   0.01   0.01
## T_IP.IC.sup 1.00  0.98   1.00   1.00
## T_IP.IC.sup 1.00  1.00   1.00   1.00
## T_IP.IC.sup 1.00  1.00   1.00   1.00
## T_IP.IC.sup 1.00  1.00   1.00   1.00
```

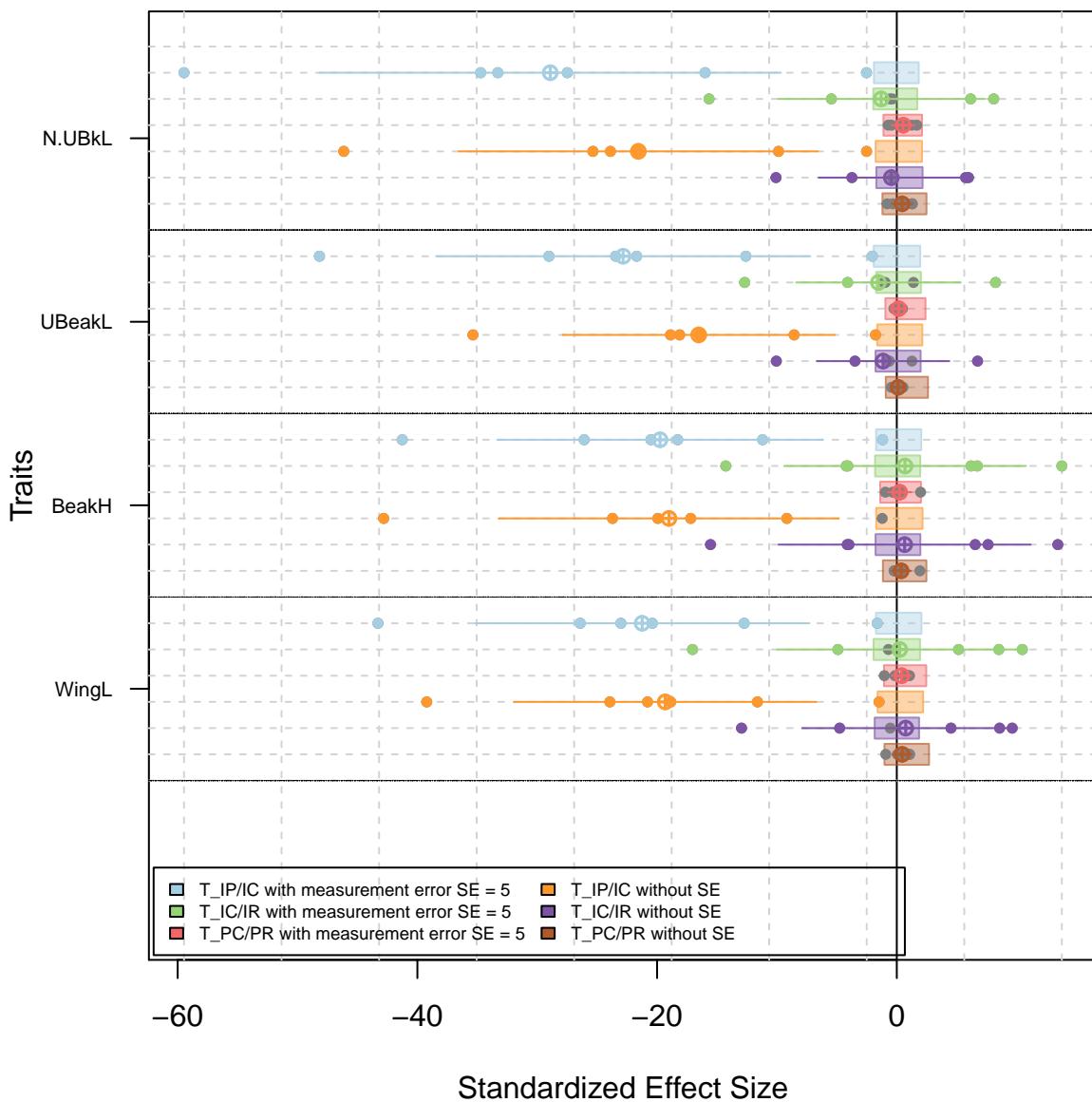
5.1.2 Use the standard errors of measure in the analysis (argument SE).

Here we can take into account the measurement error for each traits in the calculation of Standardized Effect Size of T-statistics. For this exemple, let's say that each trait is subject to a measurement error equal to half of it means.

```
res.finch.SE0 <- Tstats(traits.finch, ind.plot = ind.plot.finch, sp = sp.finch,
                        SE = 0, print = FALSE)

res.finch.SE5 <- Tstats(traits.finch, ind.plot = ind.plot.finch, sp = sp.finch,
                        SE = colMeans(traits.finch, na.rm = T)/2, print = FALSE)

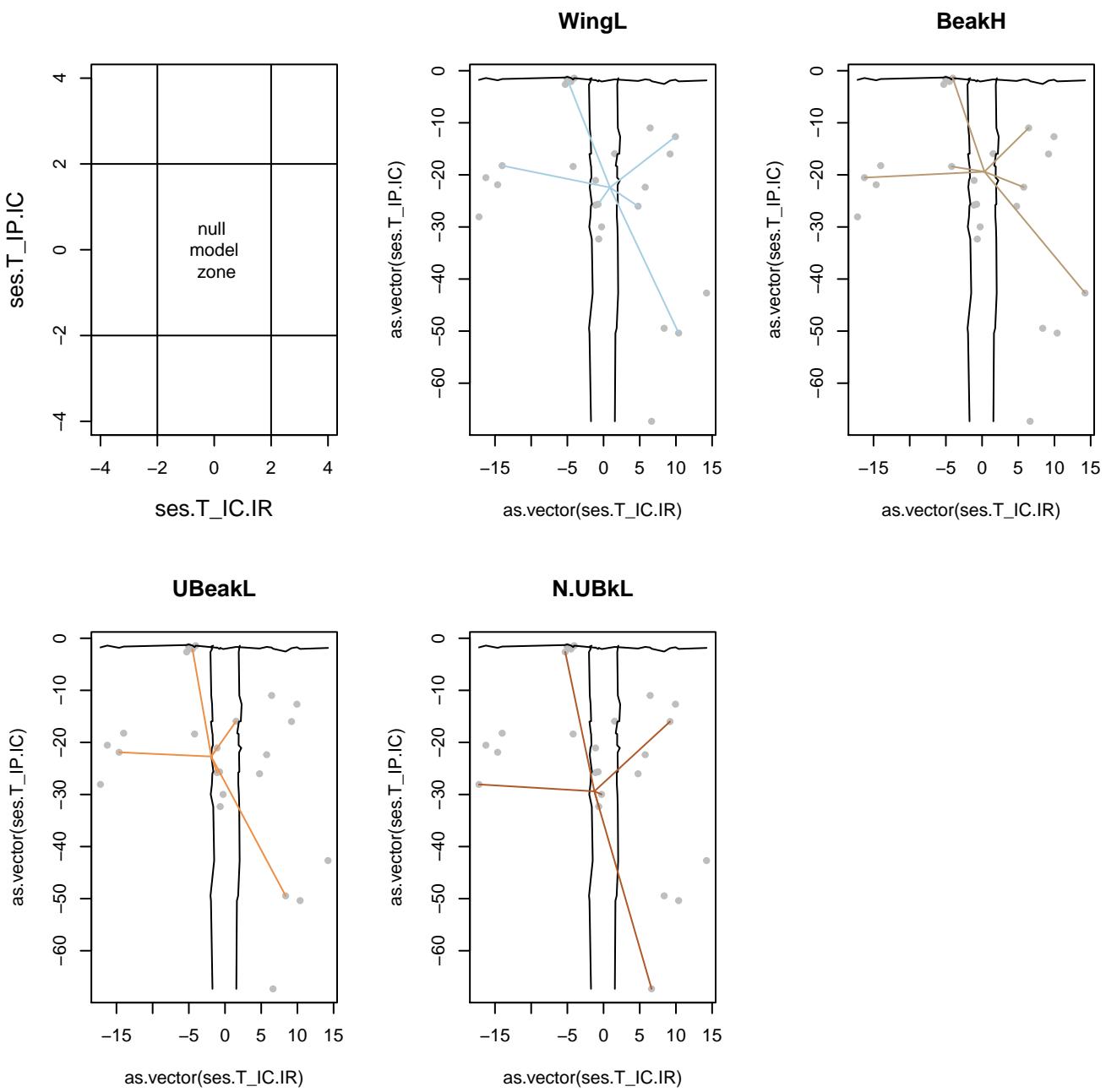
plot(as.listofindex(list(res.finch.SE0, res.finch.SE5),
                    namesindex = c("T_IP/IC with measurement error SE = 5",
                                  "T_IC/IR with measurement error SE = 5",
                                  "T_PC/PR with measurement error SE = 5",
                                  "T_IP/IC without SE", "T_IC/IR without SE",
                                  "T_PC/PR without SE")))
```

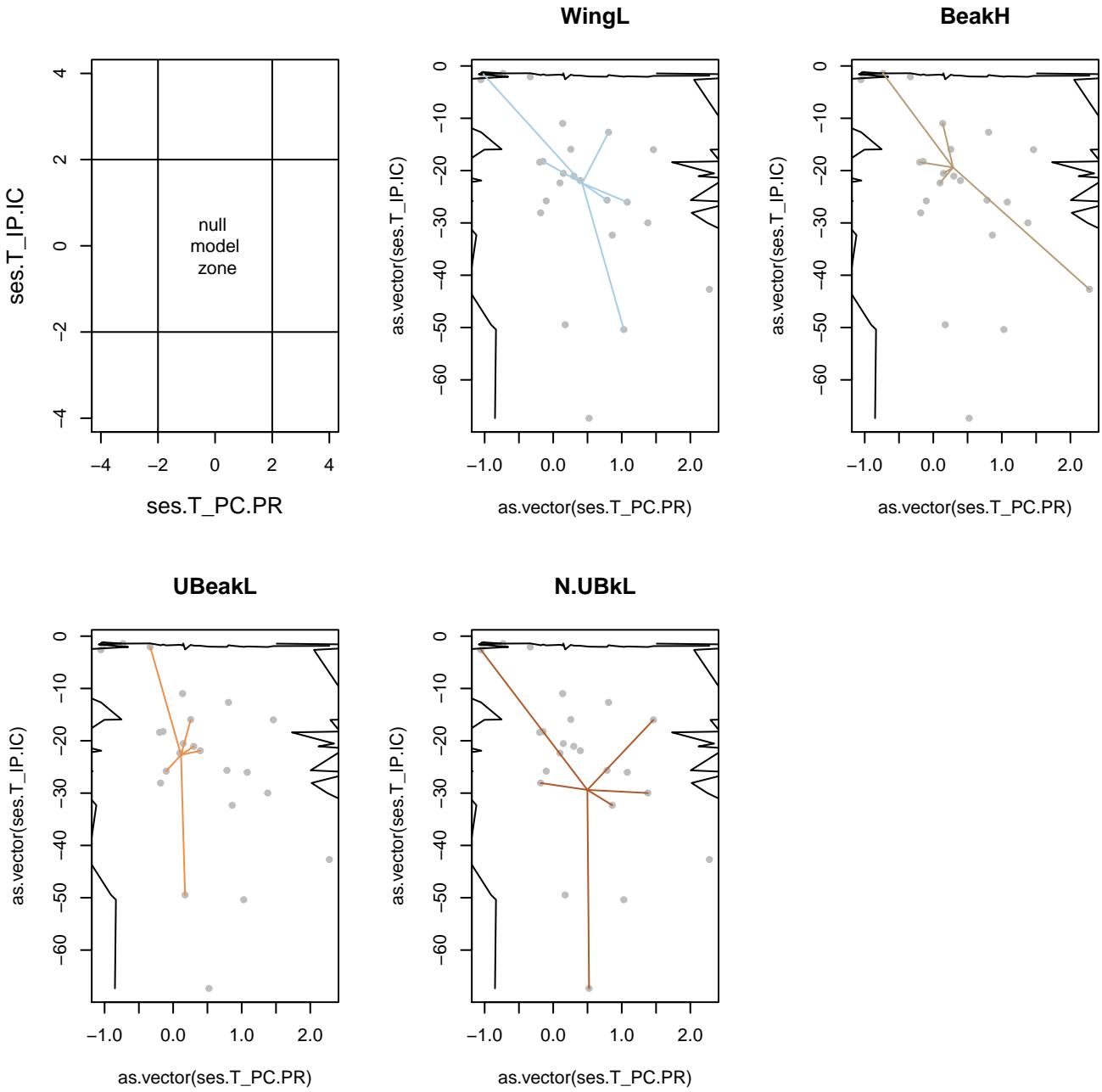


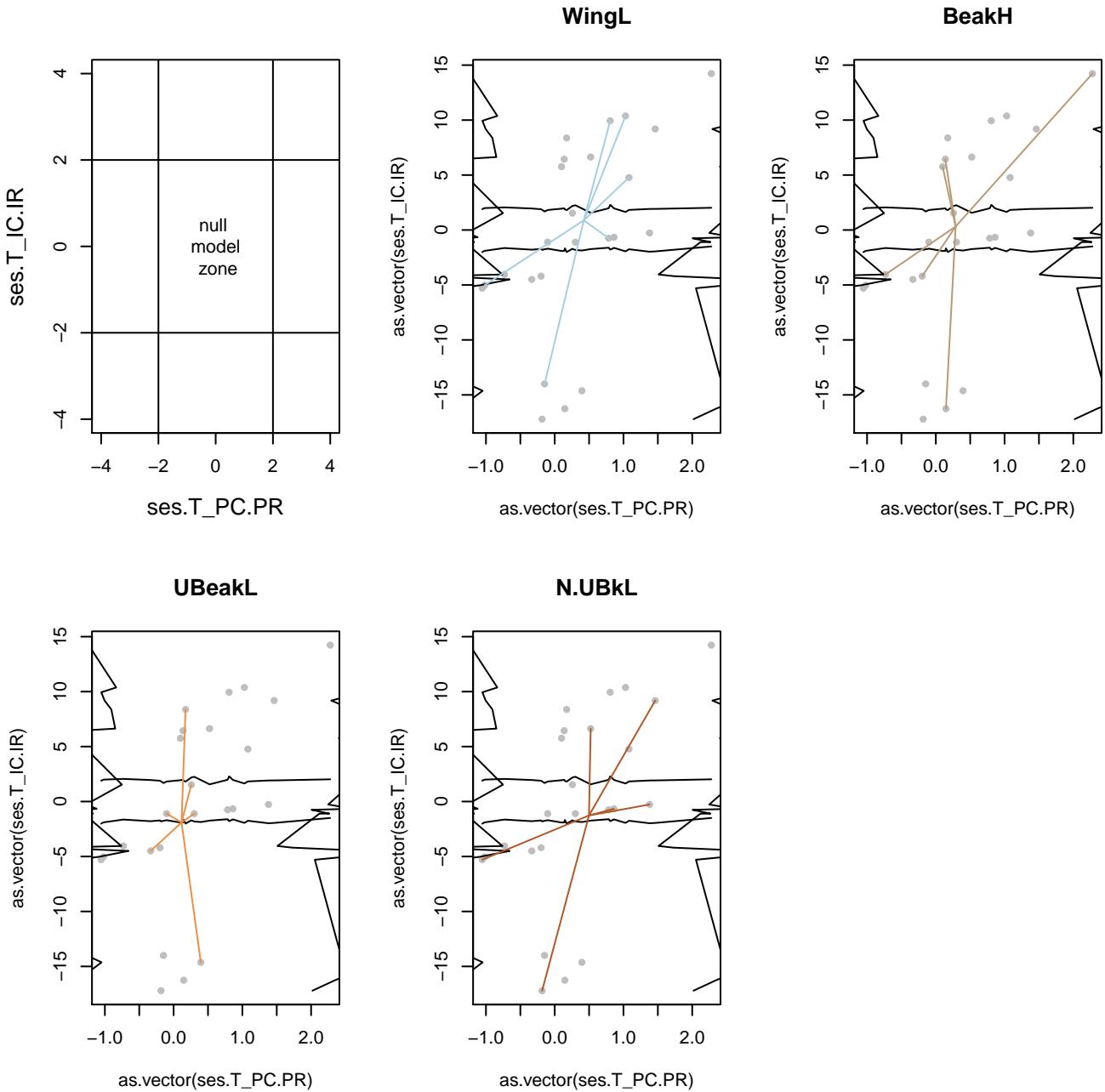
5.1.3 Plot T-statistics correlations

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)
```

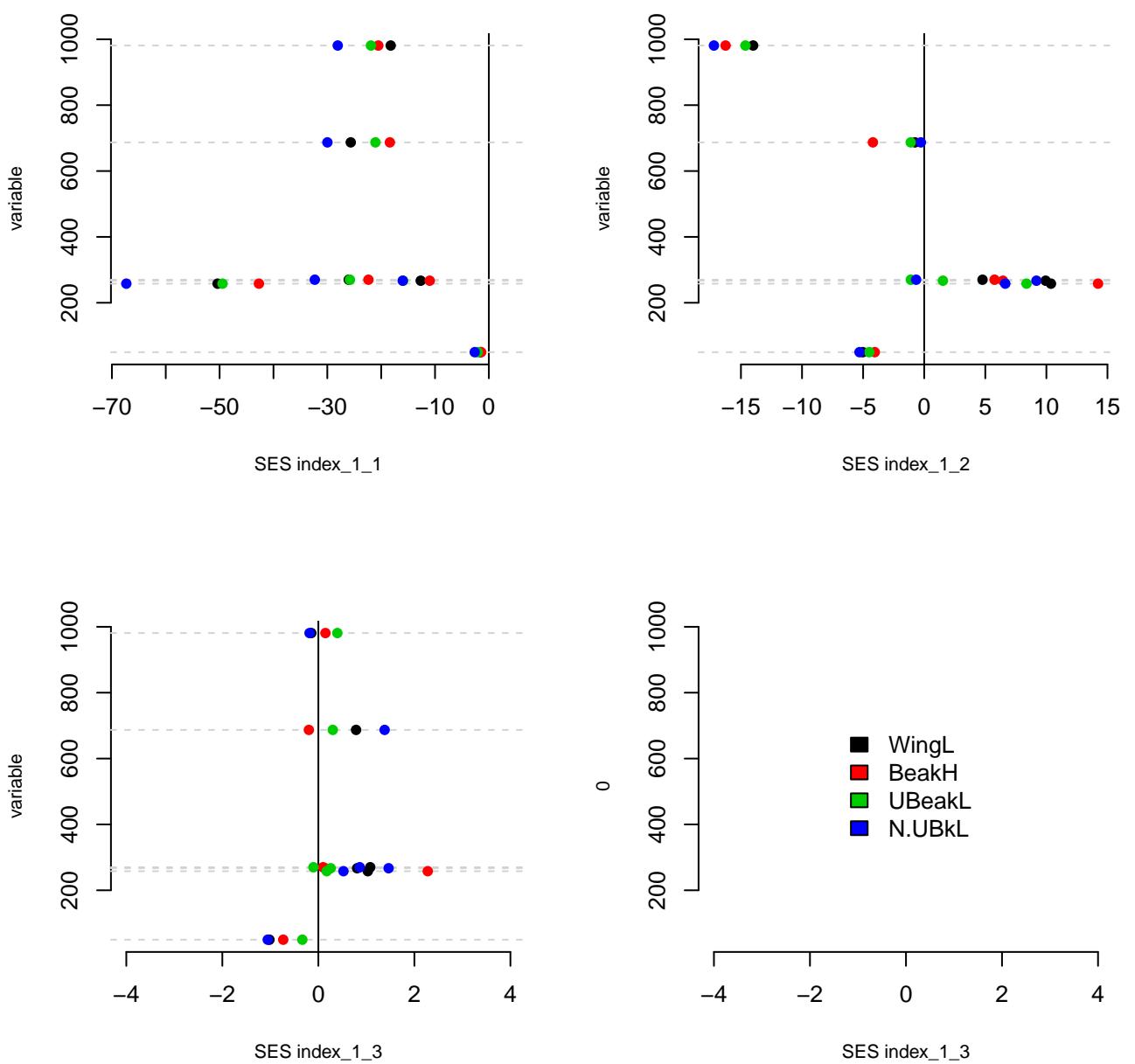






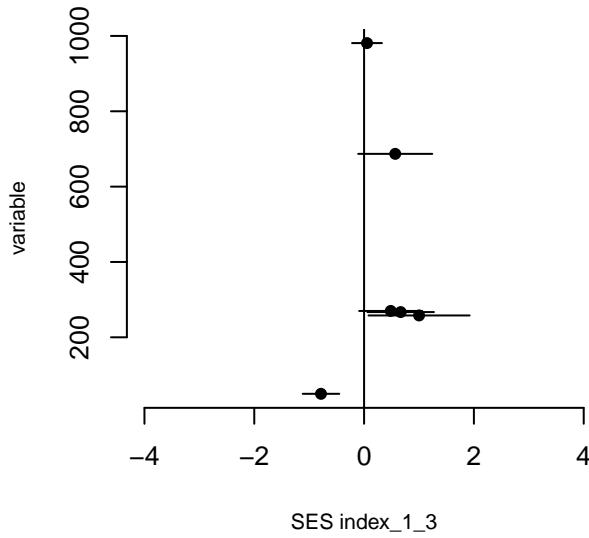
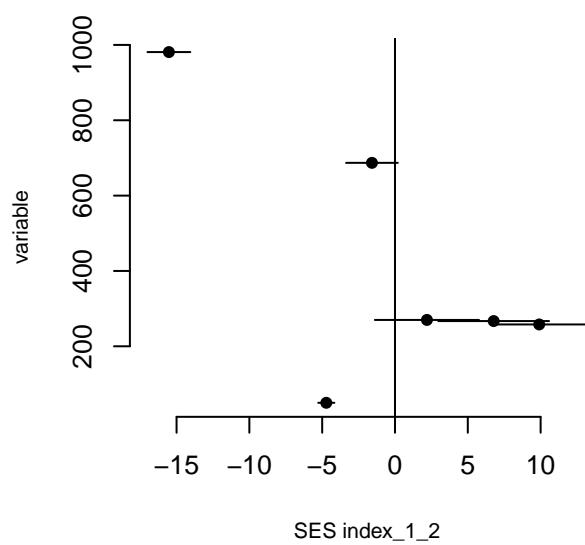
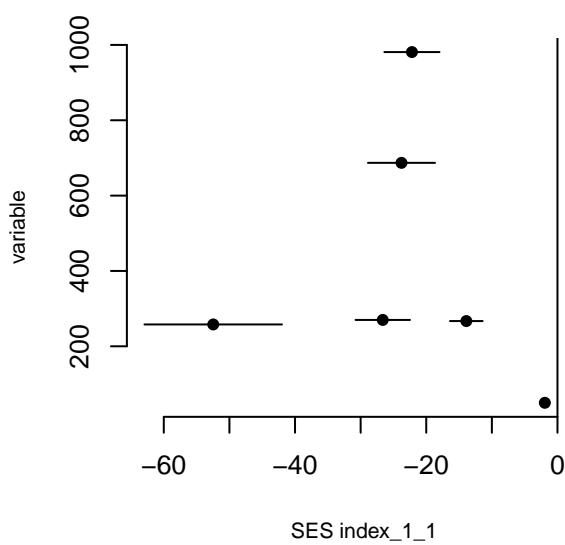
Here we plot T-statistics (in the standardized effect size SES form) in function of species richness by sites.

```
par(mfrow = c(2,2))
species.richness<-table(ind.plot.finches)
plotSESvar(as.listofindex(list(res.finches)), 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 functions you want 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, na.rm = T)" )

#Test against the null model regional.ind
res.finch.spRegional.ind<-ComIndex(traits = traits.finch, index = funct, sp = sp.finch,
                                      nullmodels = "regional.ind", ind.plot = ind.plot.finch,
                                      nperm = 99, print = FALSE)

#Test against the null model regional.pop
#Individuals values are transformed in populational values
res.finch.spRegional.pop<-ComIndex(traits = traits.finch, index = funct, sp = sp.finch,
                                      nullmodels = "regional.pop", ind.plot = ind.plot.finch,
                                      nperm = 99, 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, na.rm = T))" )

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, na.rm = T))"

##because randomization is within community only

ptm <- proc.time()
res.finch.ind_loc<-ComIndex(traits = traits.finch, index = funct.1,
                             sp = sp.finch, nullmodels = "local",
                             ind.plot = ind.plot.finch, nperm = 99,
                             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 = 99,
                             print = FALSE)
proc.time_ComIndex1 <- proc.time() - ptm
```

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, na.rm = T))")

```

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 = 99, 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

## #####
## # Community metrics calculation #
## #####
## class: ComIndex
## $call: ComIndex(traits = traits.finch, index = funct.1, nullmodels = "regional.ind",
##                 ind.plot = ind.plot.finch, sp = sp.finch, nperm = 99, printprogress = FALSE)
##
## #####
## $obs: list of observed values
## $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, na.rm = T))
##
## #####
## $null: list of null values, number of permutation: 99
## $tapply(x, ind.plot.finch, function(x) mean(x, na.rm = T))_nm ... null model = regional.i
## $tapply(x, ind.plot.finch, function(x) kurtosis(x, na.rm = T))_nm ... null model = region
## $tapply(x, ind.plot.finch, function(x) max(x, na.rm = T)-min(x, na.rm = T))_nm ... null m

```

```

## $tapply(x, ind.plot.finch, function(x) CVNND(x, na.rm = T))_nm ... null model = regional.
##
## #####
## 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:
##    DMaj     EspHd FlorChrl  GnovTwr MrchBndl SCruInde
##    50       267      981      258      270      687

summary(res.finch.withIV)

## [1] "Observed values"
## `$tapply(x, ind.plot.finch, function(x) mean(x, na.rm = T))` 
##   WingL        BeakH        UBeakL        N.UBkL
##   Min. :66.96  Min. : 9.715  Min. :14.82  Min. :10.11
##   1st Qu.:67.61 1st Qu.: 9.989  1st Qu.:15.41  1st Qu.:10.74
##   Median :67.72  Median :10.367  Median :16.44  Median :11.04
##   Mean   :67.68  Mean   :10.619  Mean   :16.66  Mean   :11.12
##   3rd Qu.:67.79 3rd Qu.:11.360 3rd Qu.:17.98 3rd Qu.:11.57
##   Max.   :68.32  Max.   :11.711  Max.   :18.68  Max.   :12.15
##
## `$tapply(x, ind.plot.finch, function(x) kurtosis(x, na.rm = T))` 
##   WingL        BeakH        UBeakL        N.UBkL
##   Min. :-1.4661  Min. :-1.3600  Min. :-1.5999  Min. :-1.7517
##   1st Qu.:-1.2523 1st Qu.:-0.6418  1st Qu.:-1.1848  1st Qu.:-1.4487
##   Median :-0.8208  Median :-0.2972  Median :-1.0891  Median :-1.0828
##   Mean   :-0.6218  Mean   :-0.1105  Mean   :-0.5643  Mean   :-0.7396
##   3rd Qu.:-0.2742 3rd Qu.: 0.6464  3rd Qu.:-0.8763  3rd Qu.:-0.9456
##   Max.   : 0.8648  Max.   : 1.0872  Max.   : 2.4142  Max.   : 1.9503
##
## `$tapply(x, ind.plot.finch, function(x) max(x, na.rm = T)-min(x, na.rm = T))` 
##   WingL        BeakH        UBeakL        N.UBkL
##   Min. :11.00  Min. : 4.30  Min. : 8.70  Min. : 6.500
##   1st Qu.:34.25 1st Qu.:14.65  1st Qu.:11.12  1st Qu.: 8.875
##   Median :35.50  Median :16.05  Median :12.60  Median :10.050
##   Mean   :31.83  Mean   :14.67  Mean   :11.93  Mean   : 9.333

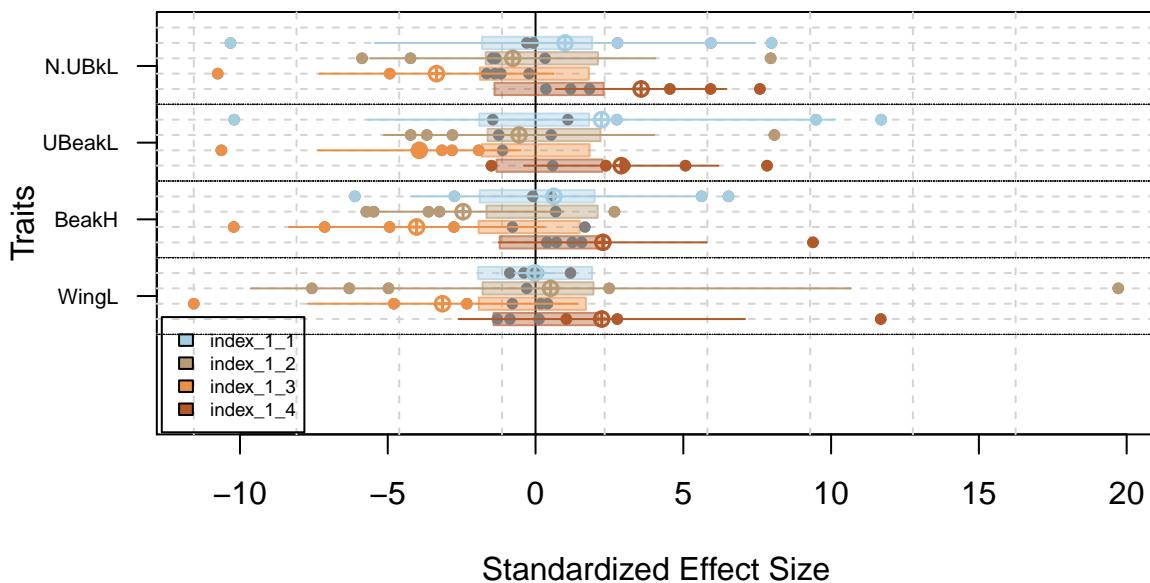
```

```

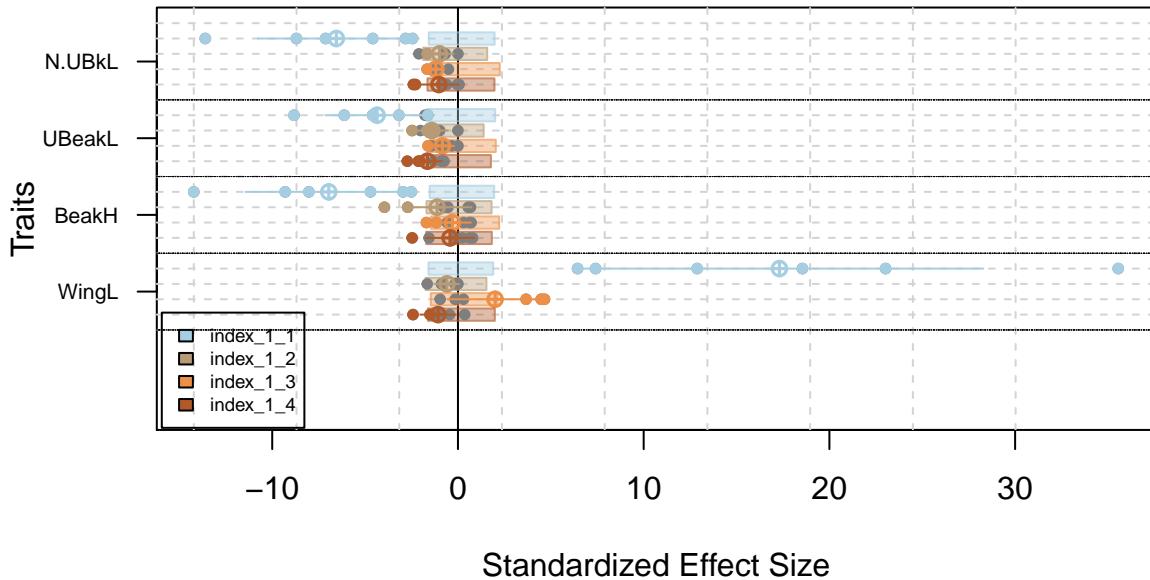
## 3rd Qu.:36.75   3rd Qu.:17.52   3rd Qu.:13.25   3rd Qu.:10.325
## Max.    :38.00   Max.    :19.40    Max.    :13.60    Max.    :10.500
##
## $`tapply(x, ind.plot.finches, function(x) CVNND(x, na.rm = T))`-
##   WingL           BeakH           UBeakL          N.UBkL
##   Min.    : 7.071   Min.    : 2.960   Min.    : 2.689   Min.    : 2.700
##   1st Qu.: 7.611   1st Qu.: 3.322   1st Qu.: 3.231   1st Qu.: 4.034
##   Median  :12.018   Median  : 3.628   Median  : 3.969   Median  : 5.938
##   Mean    :14.282   Mean    : 5.966   Mean    : 6.392   Mean    :10.303
##   3rd Qu.:15.977   3rd Qu.: 5.178   3rd Qu.: 5.052   3rd Qu.:11.021
##   Max.    :31.145   Max.    :16.664   Max.    :19.325   Max.    :31.129
##
## [1] "null values"
## $`tapply(x, ind.plot.finches, function(x) mean(x, na.rm = T))`-
##   Min. 1st Qu. Median  Mean 3rd Qu.   Max.
##   9.075 10.610 13.290 26.240 29.300 70.240
##
## $`tapply(x, ind.plot.finches, function(x) kurtosis(x, na.rm = T))`-
##   Min. 1st Qu. Median  Mean 3rd Qu.   Max.
## -1.5580 -0.9201 -0.6319 -0.4285 -0.1036 2.9990
##
## $`tapply(x, ind.plot.finches, function(x) max(x, na.rm = T)-min(x, na.rm = T))`-
##   Min. 1st Qu. Median  Mean 3rd Qu.   Max.
##   7.90   11.38  15.60  19.89  21.52   39.00
##
## $`tapply(x, ind.plot.finches, function(x) CVNND(x, na.rm = T))`-
##   Min. 1st Qu. Median  Mean 3rd Qu.   Max.   NA's
##   0.7947 2.5340 3.8440 6.1970 7.8770 31.1300      43

plot(res.finches.withIV)

```

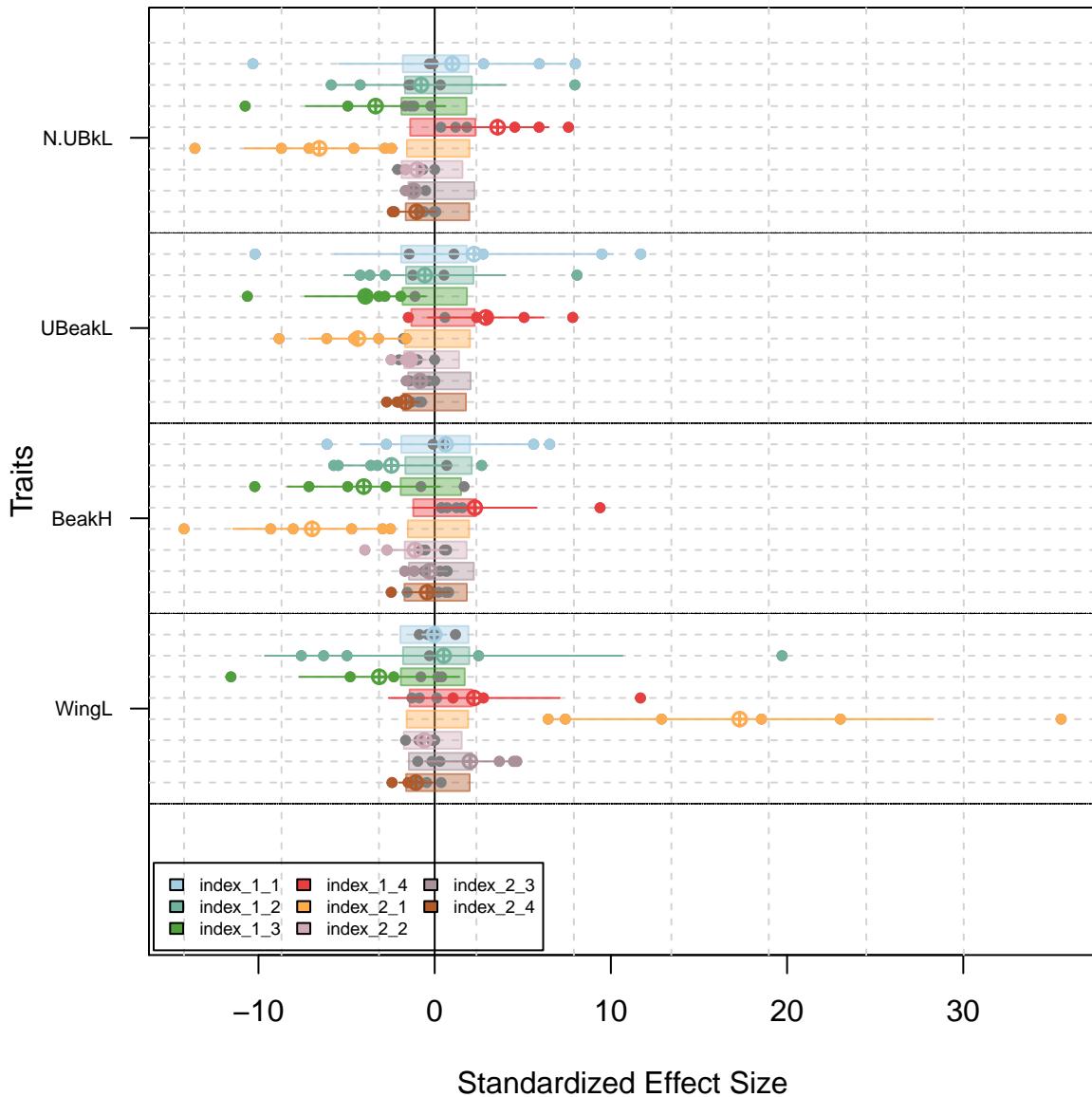


```
plot(res.finch.withoutIV)
```



Now plot the two analysis together.

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



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)
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)

class(i.l1)

## [1] "listofindex"

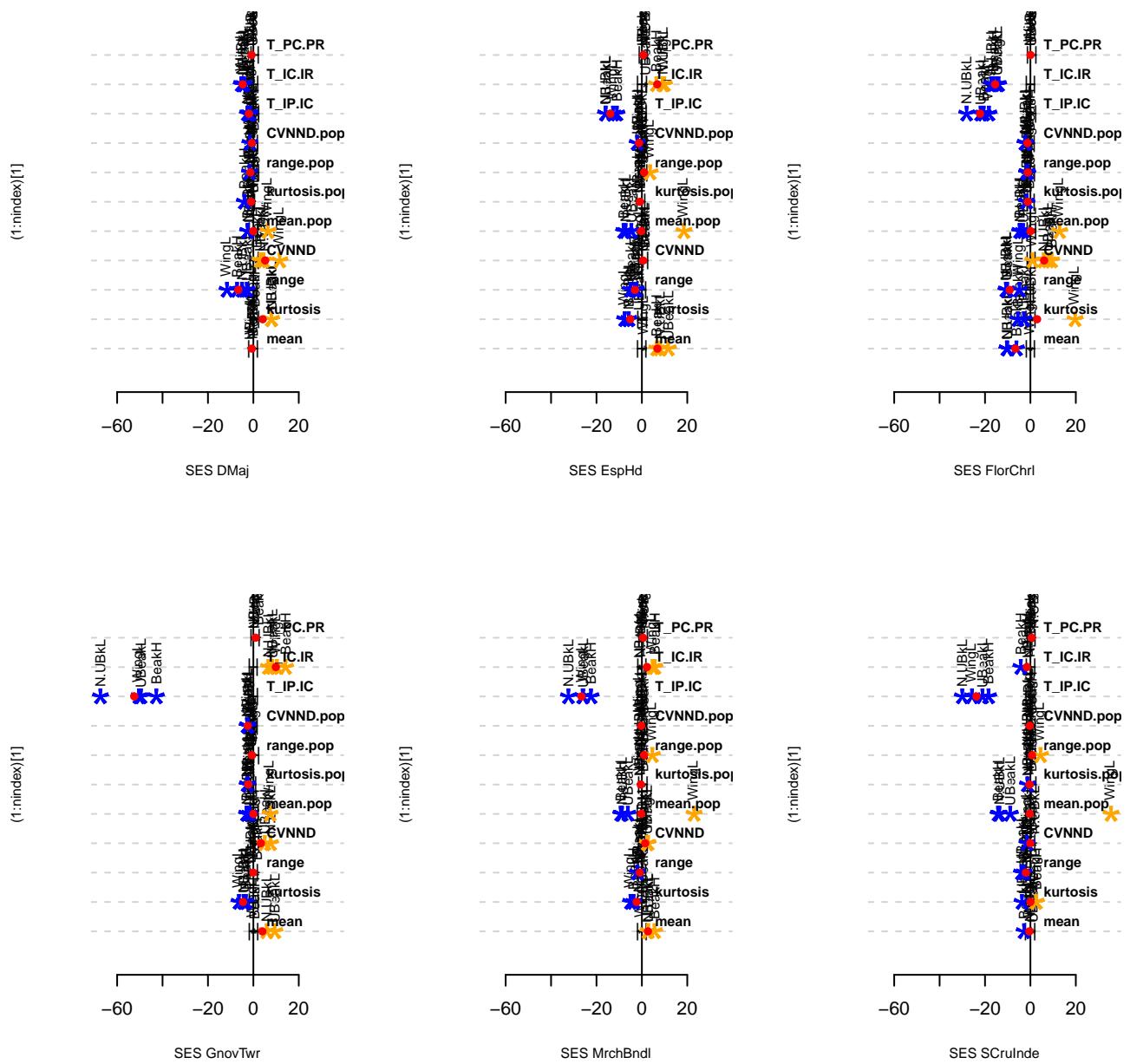
```

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

```

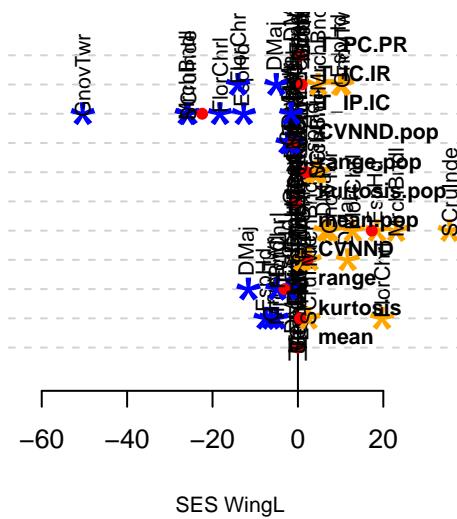
par(mfrow = c(2,3))
plot(i.l1,type = "bysites")

```



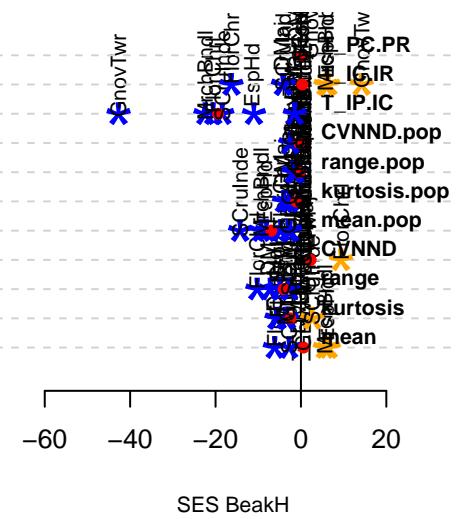
```
par(mfrow = c(2,2))
plot(i.l1,type = "bytraits")
```

(1:nindex)[i]



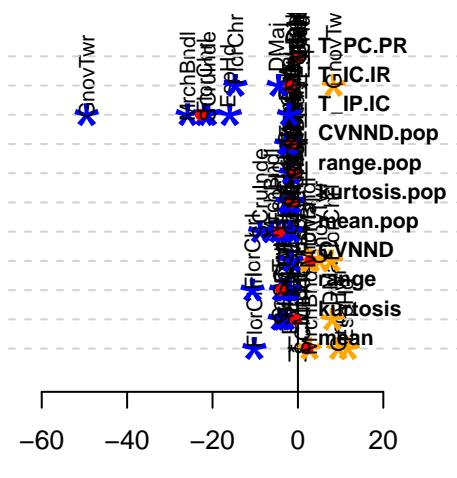
SES WingL

(1:nindex)[i]



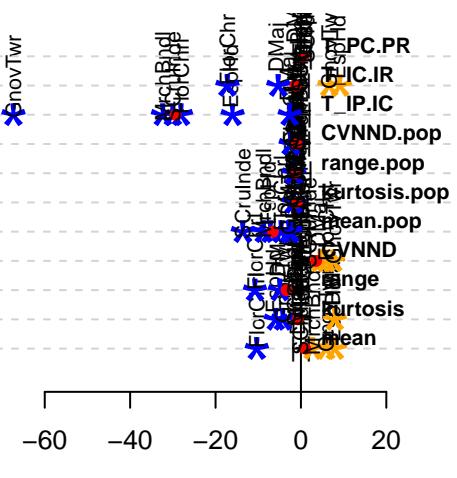
SES BeakH

(1:nindex)[i]



SES UBeakL

(1:nindex)[i]



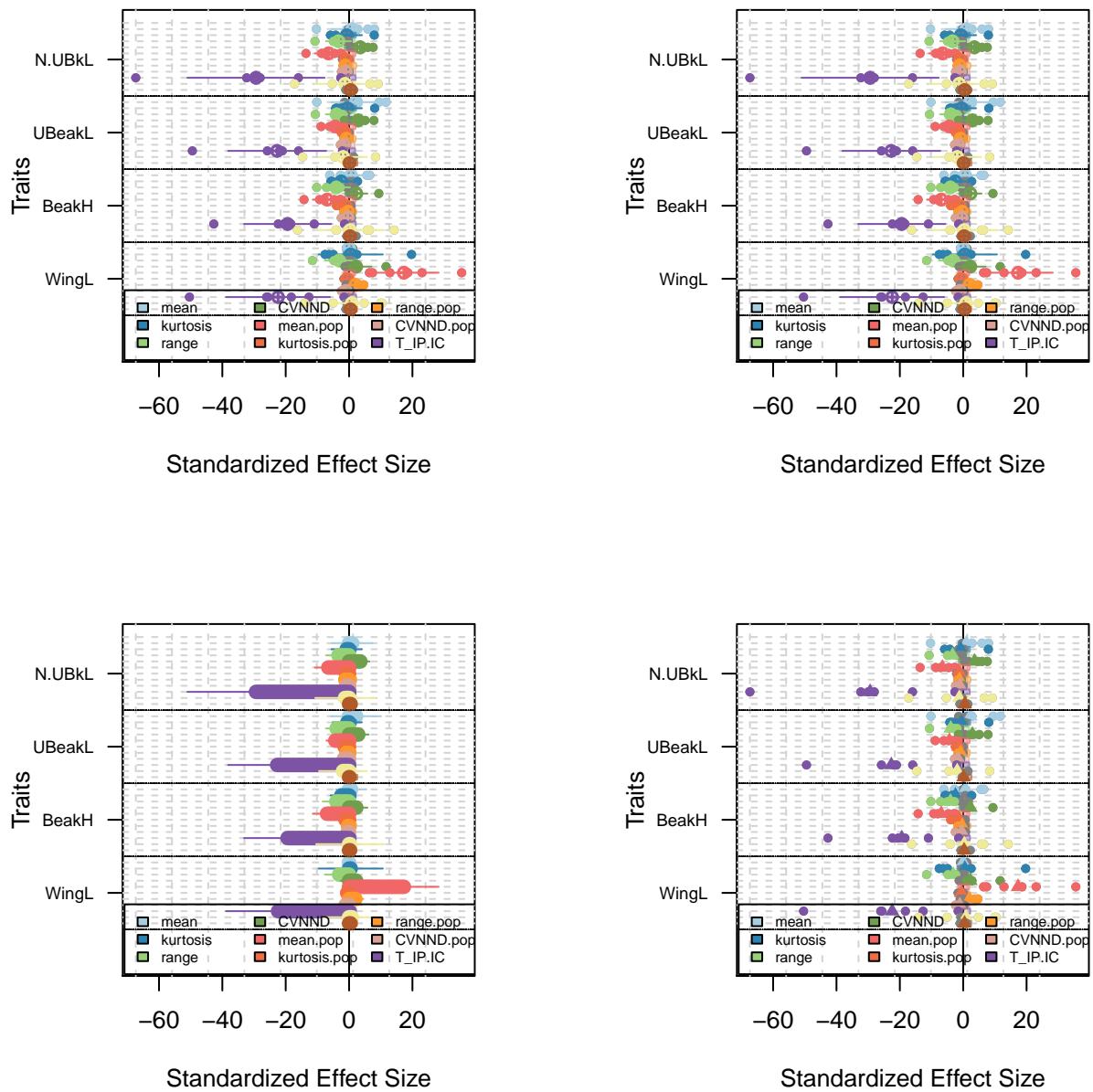
SES N.UBkL

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

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

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

plot(i.11)
plot(i.11,type = "normal")
plot(i.11,type = "barplot")
plot(i.11,type = "simple")
```



```
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 = 99, sp = sp.finch)  
  
## [1] "creating null models"  
## [1] "regional.ind 25 %"  
## [1] "regional.ind 50 %"  
## [1] "regional.ind 75 %"  
## [1] "regional.ind 100 %"  
## [1] "calculation of null values using null models"  
## [1] "sum(scale(x), na.rm = T) 50 %"  
## [1] "sum(x, na.rm = T) 100 %"  
## [1] "calculation of observed values"  
## [1] "50 %"  
## [1] "100 %"  
  
res.sum.1  
  
## #####  
## # Community metrics calculation #  
## #####  
## class: ComIndexMulti  
## $call: ComIndexMulti(traits = 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, sp = sp.finch, nperm = 99)  
##  
## #####  
## $obs: list of observed values  
##   $sum(scale(x), na.rm = T)  
##   $sum(x, na.rm = T)  
##  
## #####  
## $null: list of null values, number of permutation: NA  
##   $sum(scale(x), na.rm = T)_nm ... null model = regional.ind
```

```

## $sum(x, na.rm = T)_nm ... null model = regional.ind
##
## #####
## 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:
##       DMaj     EspHd FlorChrl  GnovTwr MrchBndl SCruInde
##       50       267      981      258      270      687

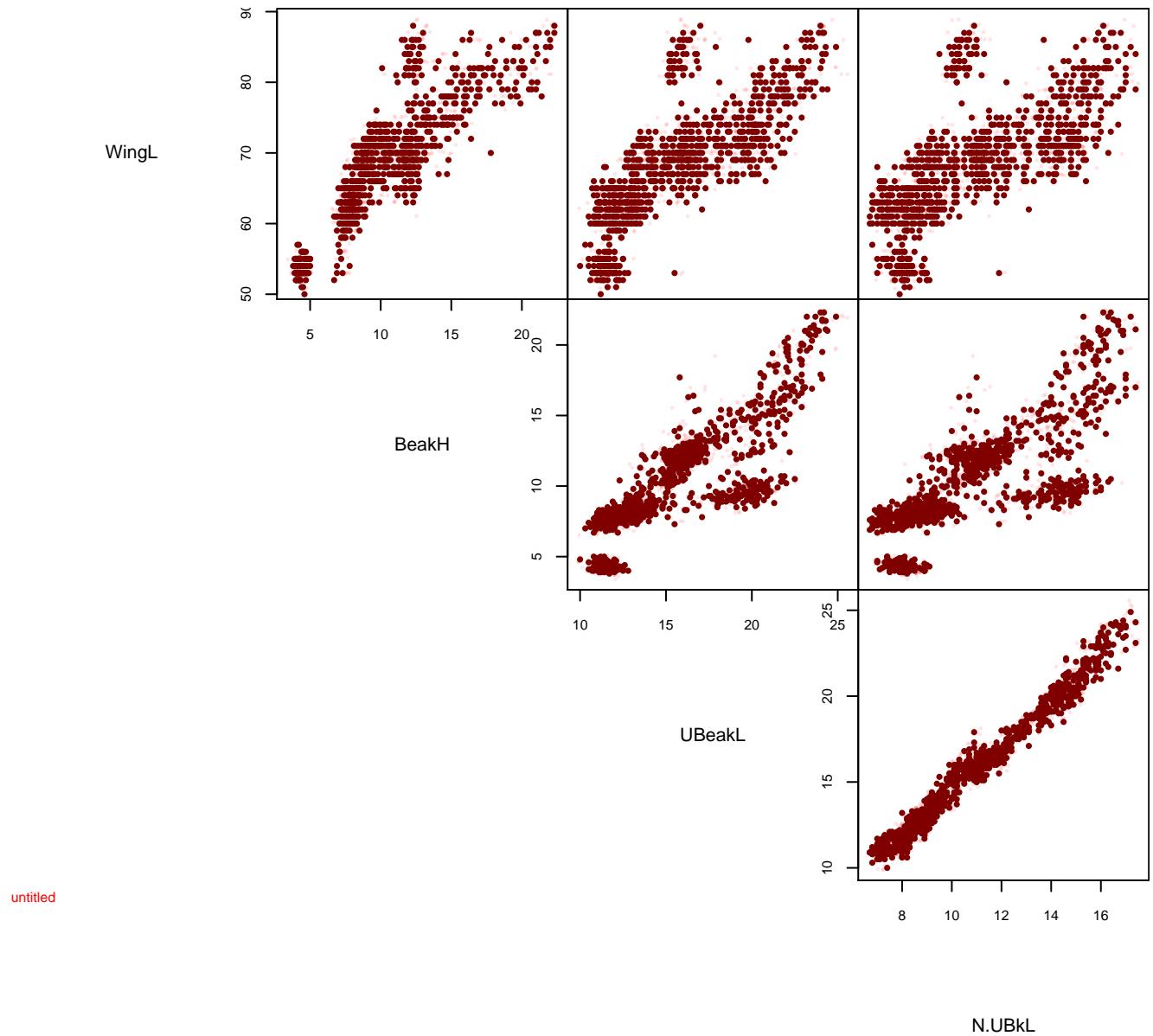
```

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)

## Showing 1000 random points of 129258 for untitled
## Showing 1000 data points of 2513 for untitled
```



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)
}

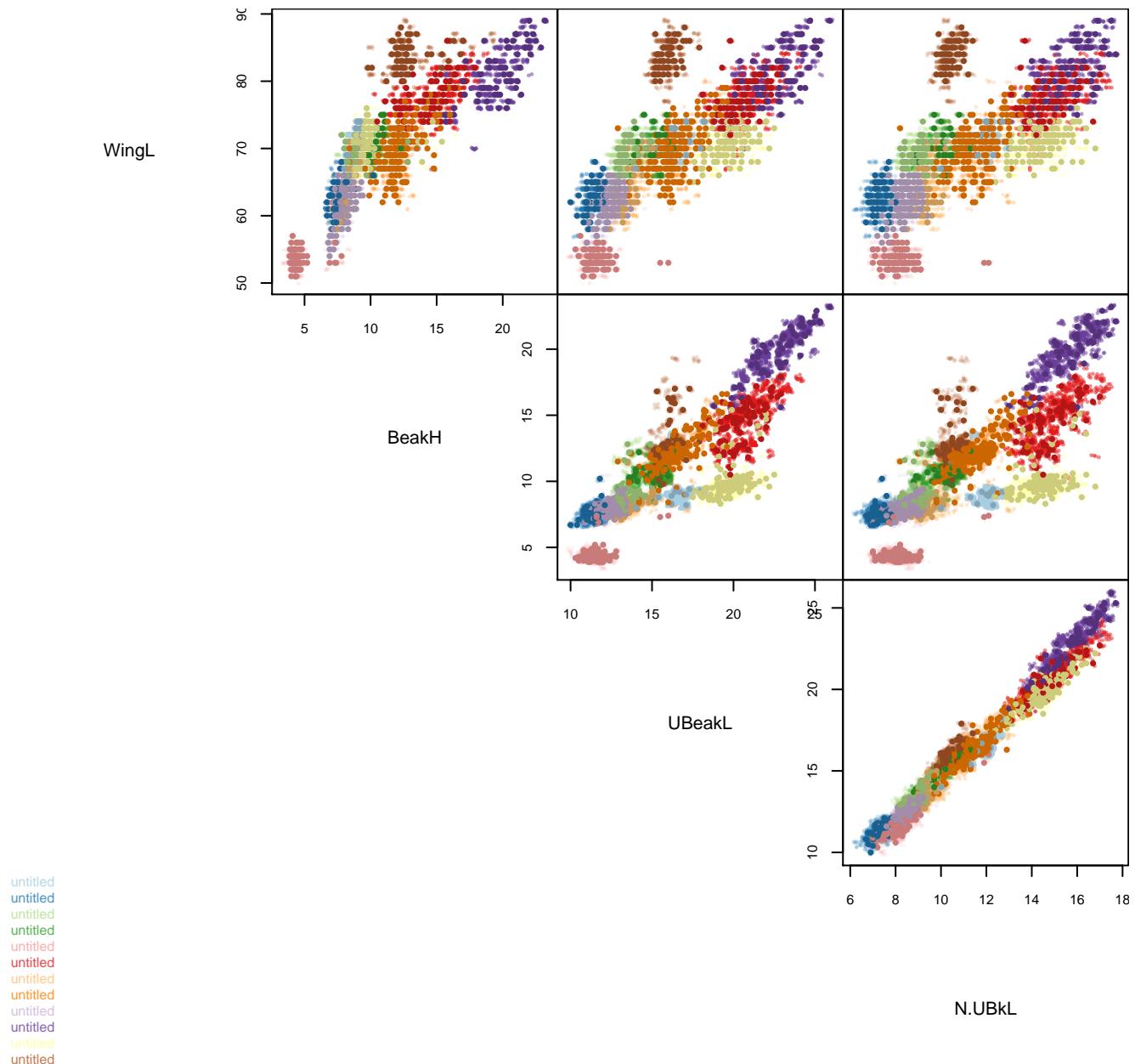
hv.list@HVList<-hv.list2
require(adegenet)

## 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

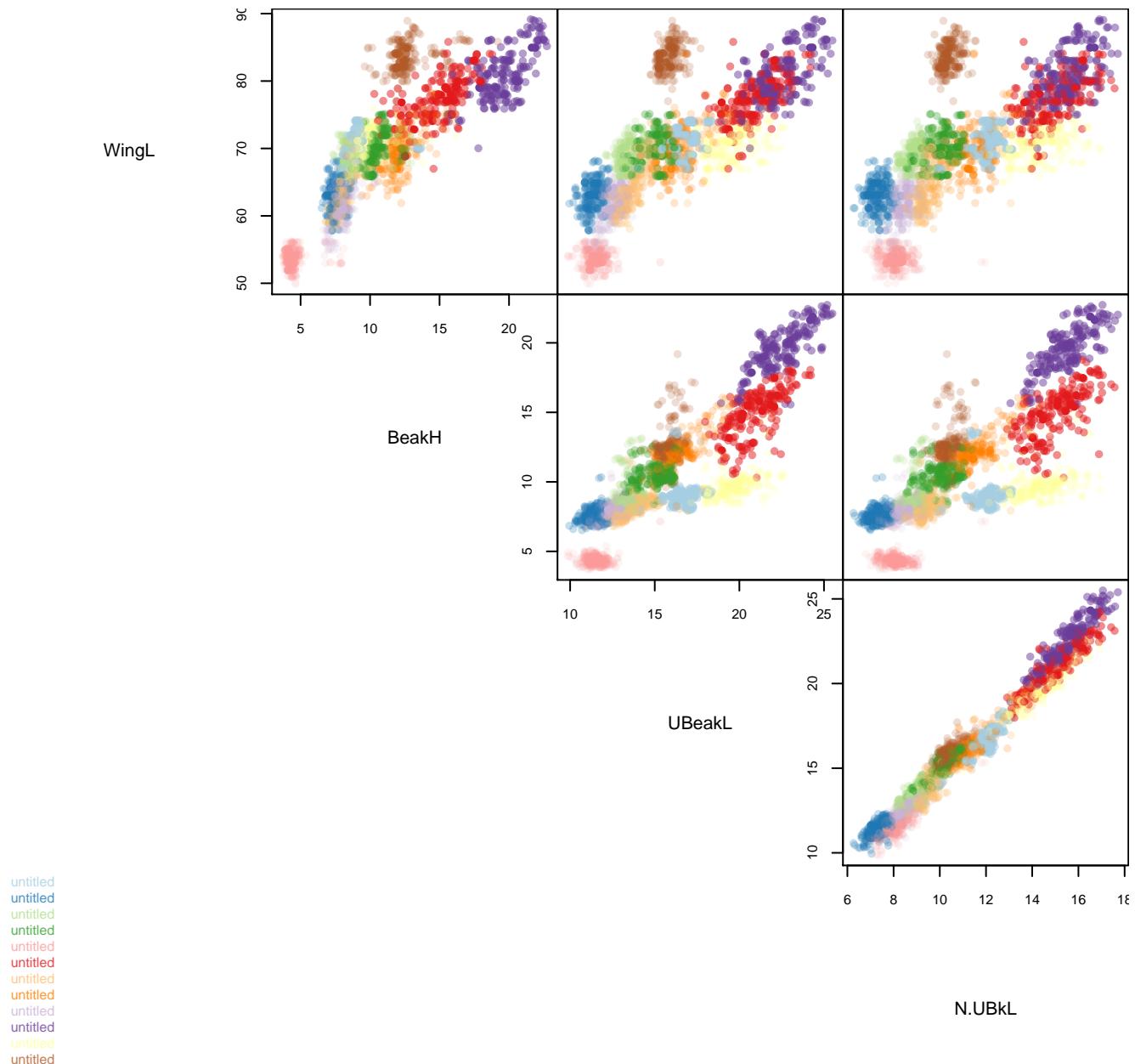
colorhv<-transp(funky(nlevels(sp.finch)), alpha = 0.8)

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

```



```
plot(hv.list, colors = colorhv, darkfactor = 0.8, showdata = F,
  npmax_random = 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')
```

`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.finck, nperm = 99, sp = sp.finck)

dwidth

hv.2<-ComIndexMulti(traits.finck.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.finck, nperm = 99, sp = sp.finck, print = FALSE)

ptm <- proc.time()
hv.3<-ComIndexMulti(traits.finck.mice,
index = c("as.numeric(try(hypervolume(na.omit(x), reps = 100,
bandwidth = 0.2, verbose = F, warnings = F)@Volume))"),
by.factor = ind.plot.finck, nullmodels ="regional.ind",
ind.plot = ind.plot.finck, nperm = 99, sp = sp.finck, print = FALSE)
proc.time_ComIndexMulti <- proc.time() - ptm

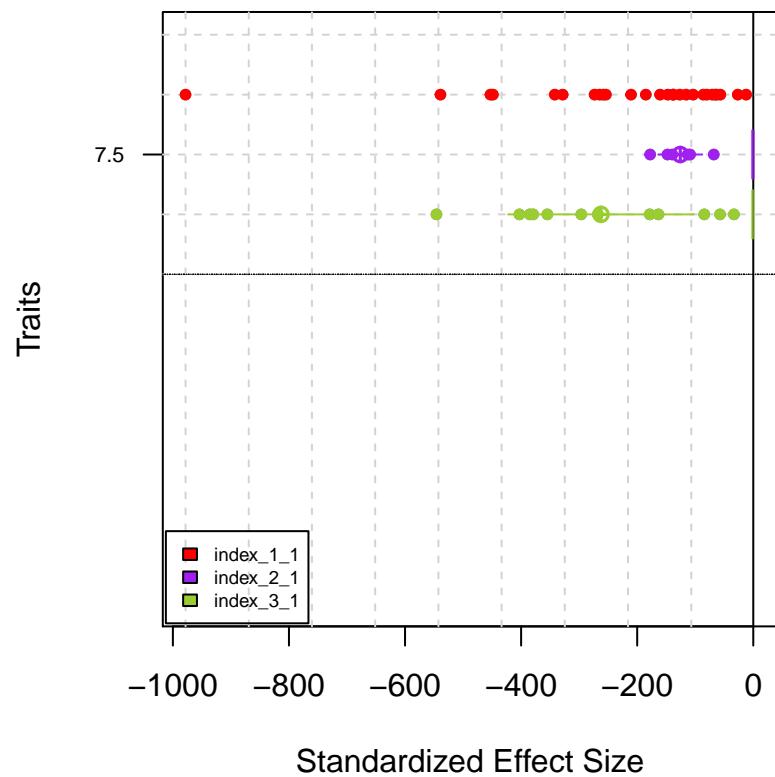
hv.4<-ComIndexMulti(traits.finck.mice,
index = c("as.numeric(try(hypervolume(na.omit(x), reps = 100,
bandwidth = 0.2, verbose = F, warnings = F)@Volume))"),
by.factor = sp.finck, nullmodels = "regional.ind",
ind.plot = ind.plot.finck, nperm = 99, sp = sp.finck, print = FALSE)

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

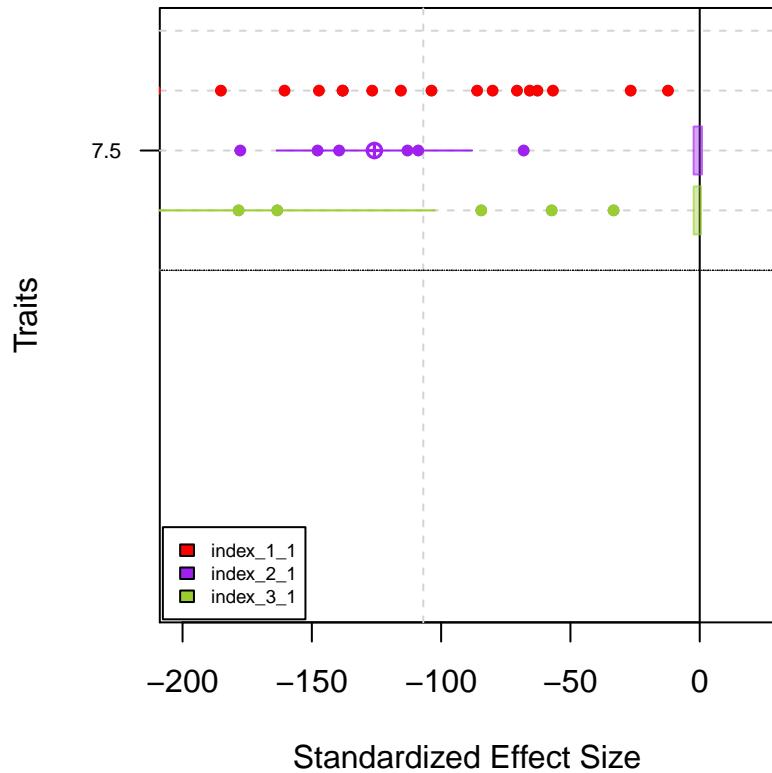
ses.list.multi<-ses.listofindex(list.ind.multi)

plot(list.ind.multi)

```



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



Compare hypervolume to Villeger'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 = 99, 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"

```

```

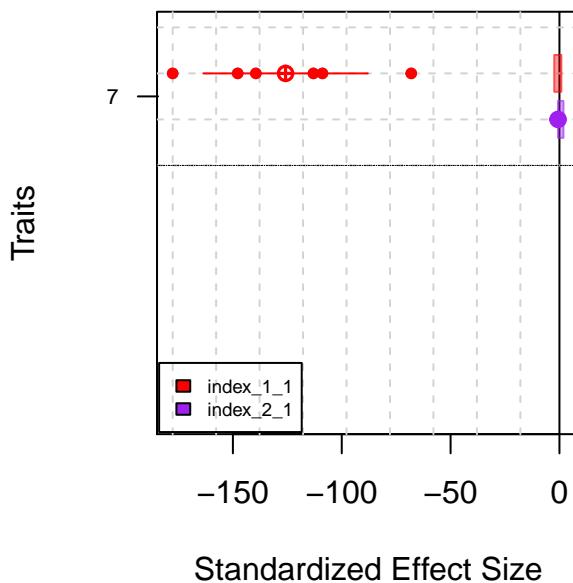
## [1] "round(convhulln(x,FA)$vol,6) 100 %"
## [1] "calculation of observed values"
## [1] "100 %"

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

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

plot(list.ind.multi2)

```



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 object is masked from 'package:e1071':
## 

```

```

##      interpolate
##
## 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

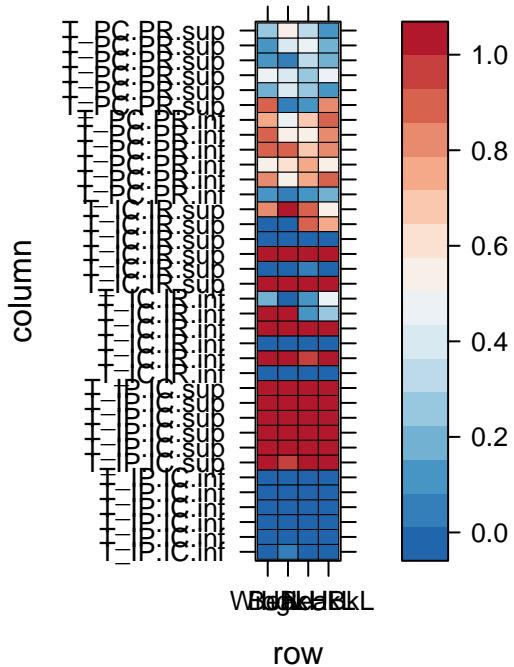
```

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")

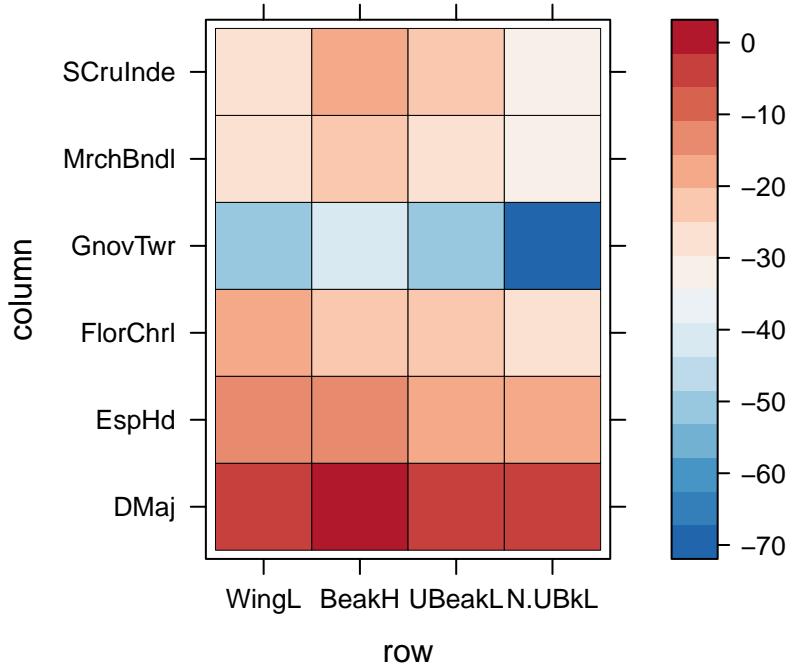
```



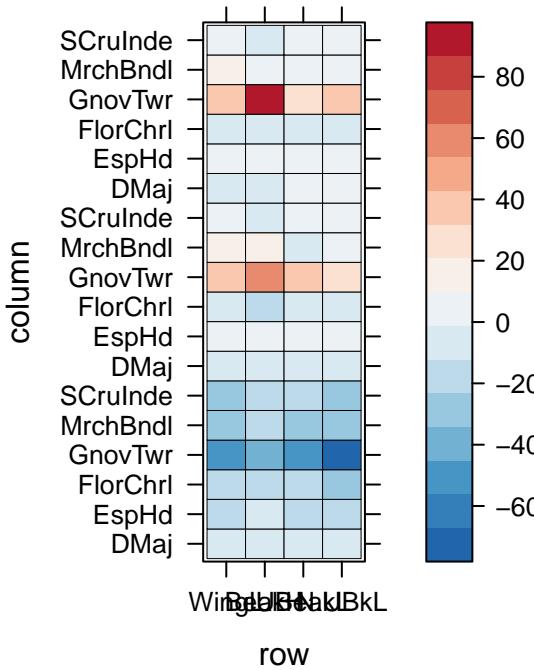
```

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")
```



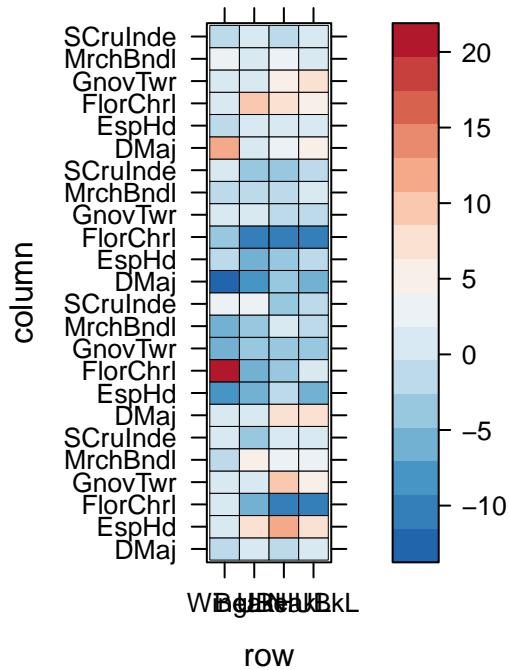
Another example using `ses.listofindex`. The first plot represent "ses" values and the second one the result of a test with H0: observed index value are greater than what we can expect using the null model (alpha = 2.5%).

```

ses.list<-ses.listofindex(i.11)

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")

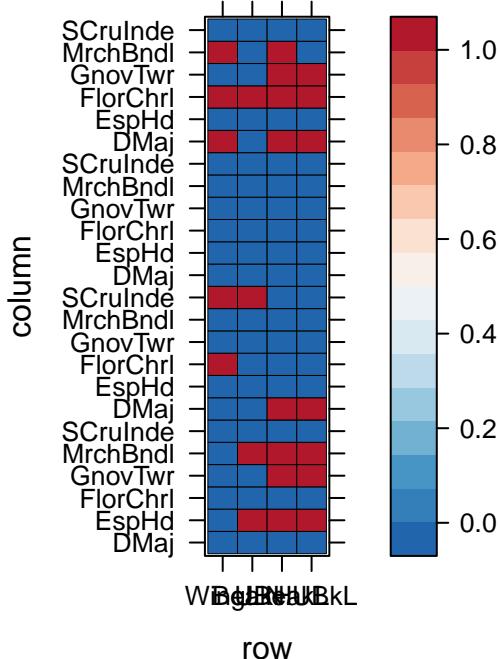
```



```

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")

```



Compare metrics calculate on individual against metrics calculate after populationnal meaning

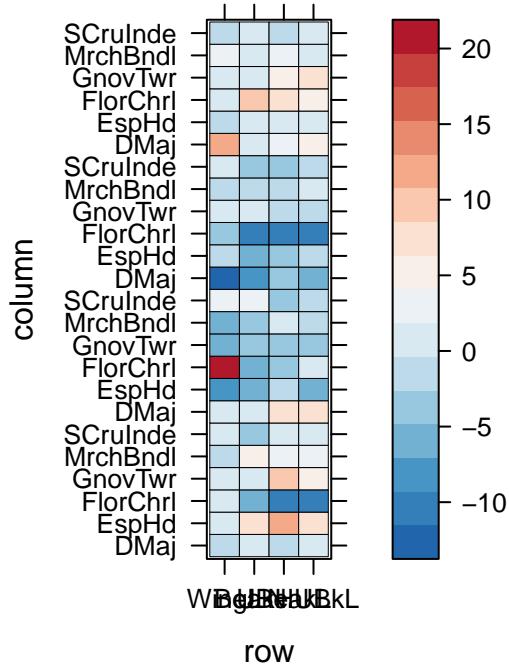
```

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

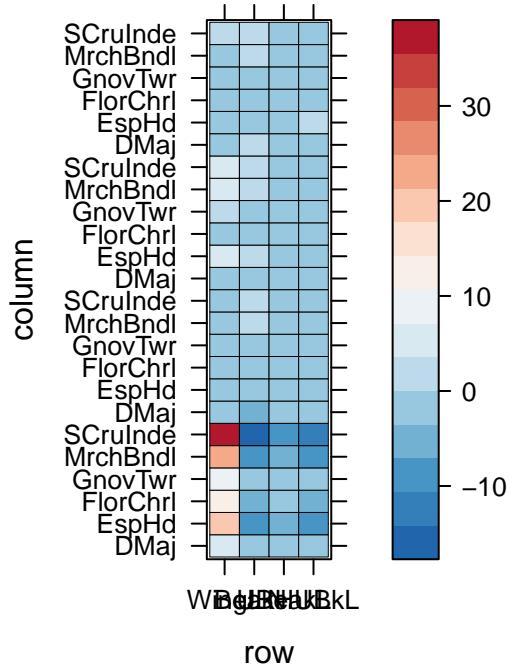
ses.sp<-t(rbind(ses.list[[5]]$ses,
  ses.list[[6]]$ses,
  ses.list[[7]]$ses,
  ses.list[[8]]$ses))

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

```



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



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

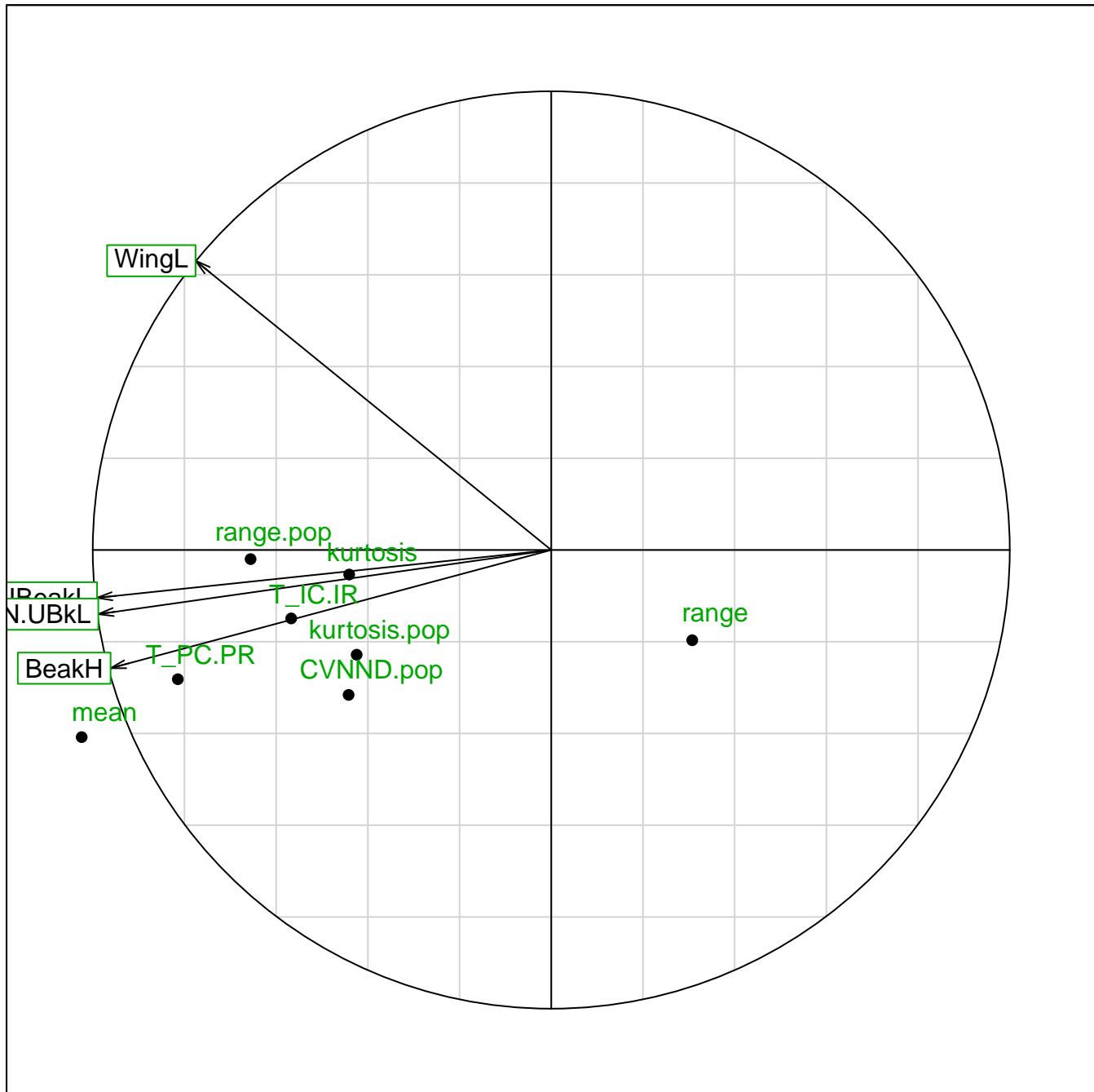
whereas analysis dudi 2 analyses all combination of traits / sites / metrics.

```
library(ade4)

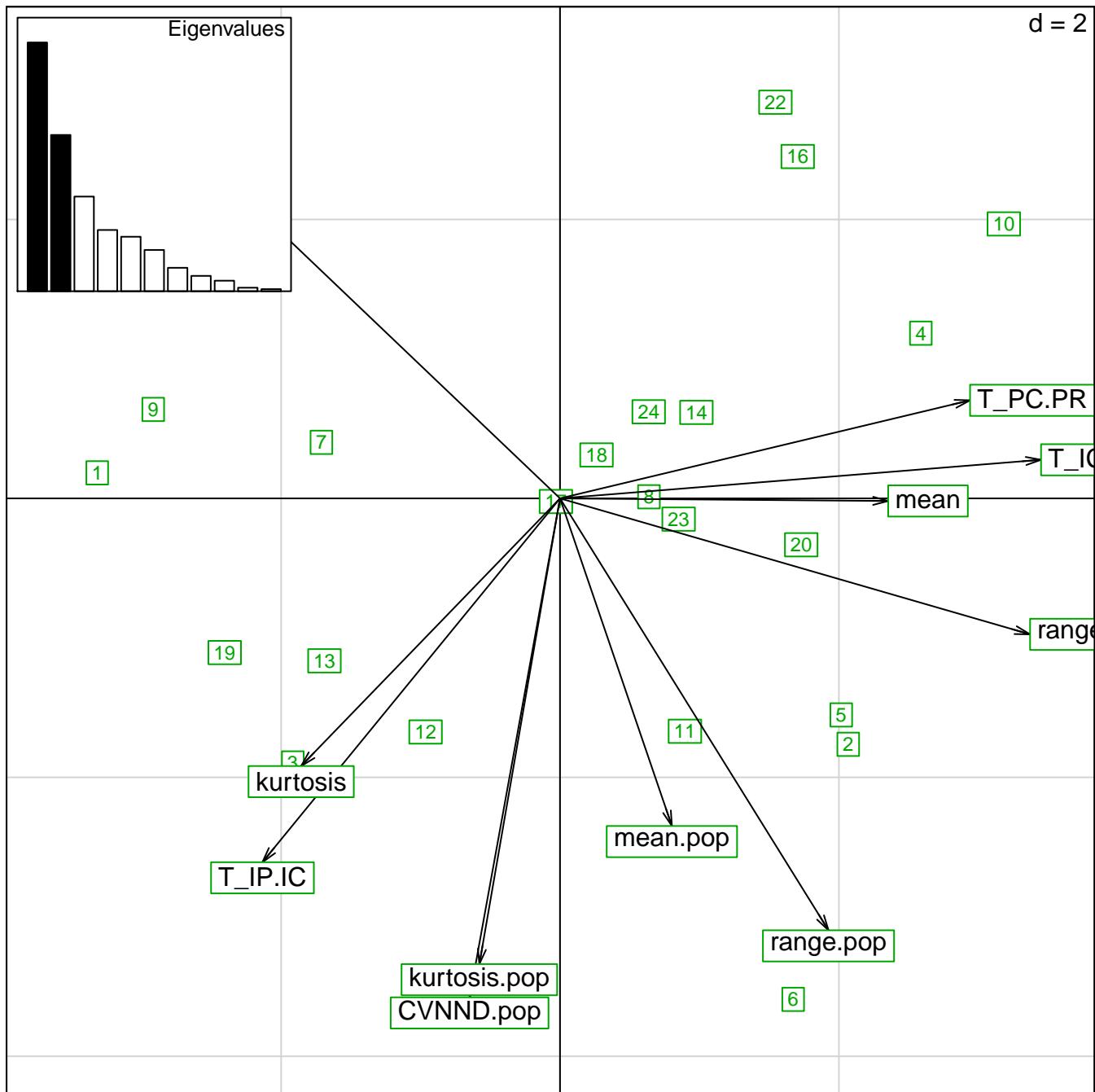
matfordudi<-matrix(nrow = length(colMeans(ses.list[[1]]$ses)), ncol = length(names(ses.list))
for(i in 1: length(names(ses.list))){  
  matfordudi[,i]<-colMeans(ses.list[[i]]$ses)  
}  
colnames(matfordudi)<-names(ses.list)  
rownames(matfordudi)<-colnames(trait.finches)

matfordudi2<-matrix(nrow = length(as.vector(ses.list[[1]]$ses)), ncol = length(names(ses.list))
for(i in 1: length(names(ses.list))){  
  matfordudi2[,i]<-as.vector(ses.list[[i]]$ses)  
}  
colnames(matfordudi2)<-names(ses.list)

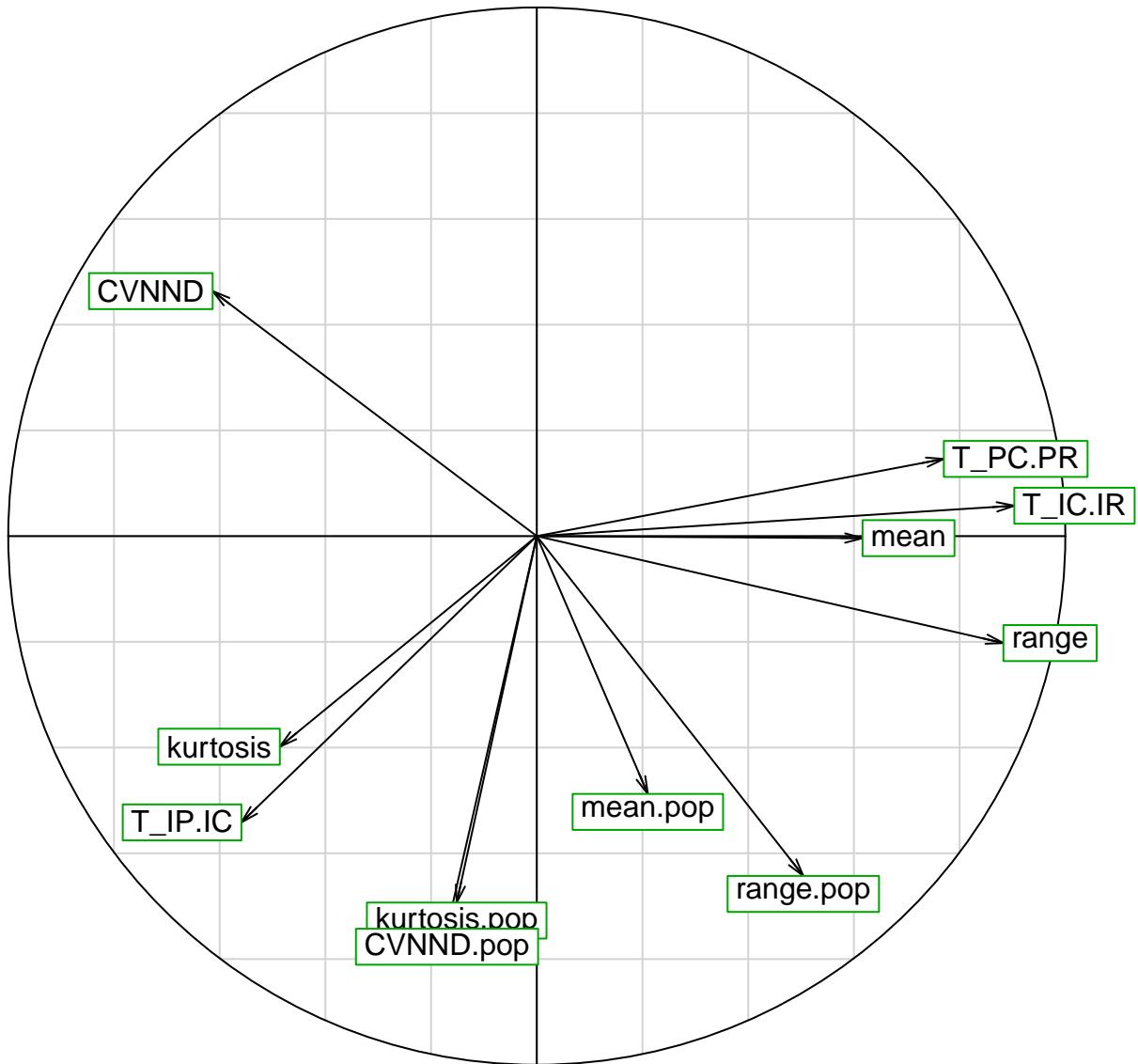
res.dudi<-dudi.pca(t(matfordudi), scan = F, nf = 2)
s.corcircle(res.dudi$co)
s.label(res.dudi$li, add.plot = T, clabel = 0, pch = 16)
s.label(res.dudi$li+0.05, add.plot = T, boxes = F)
```



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

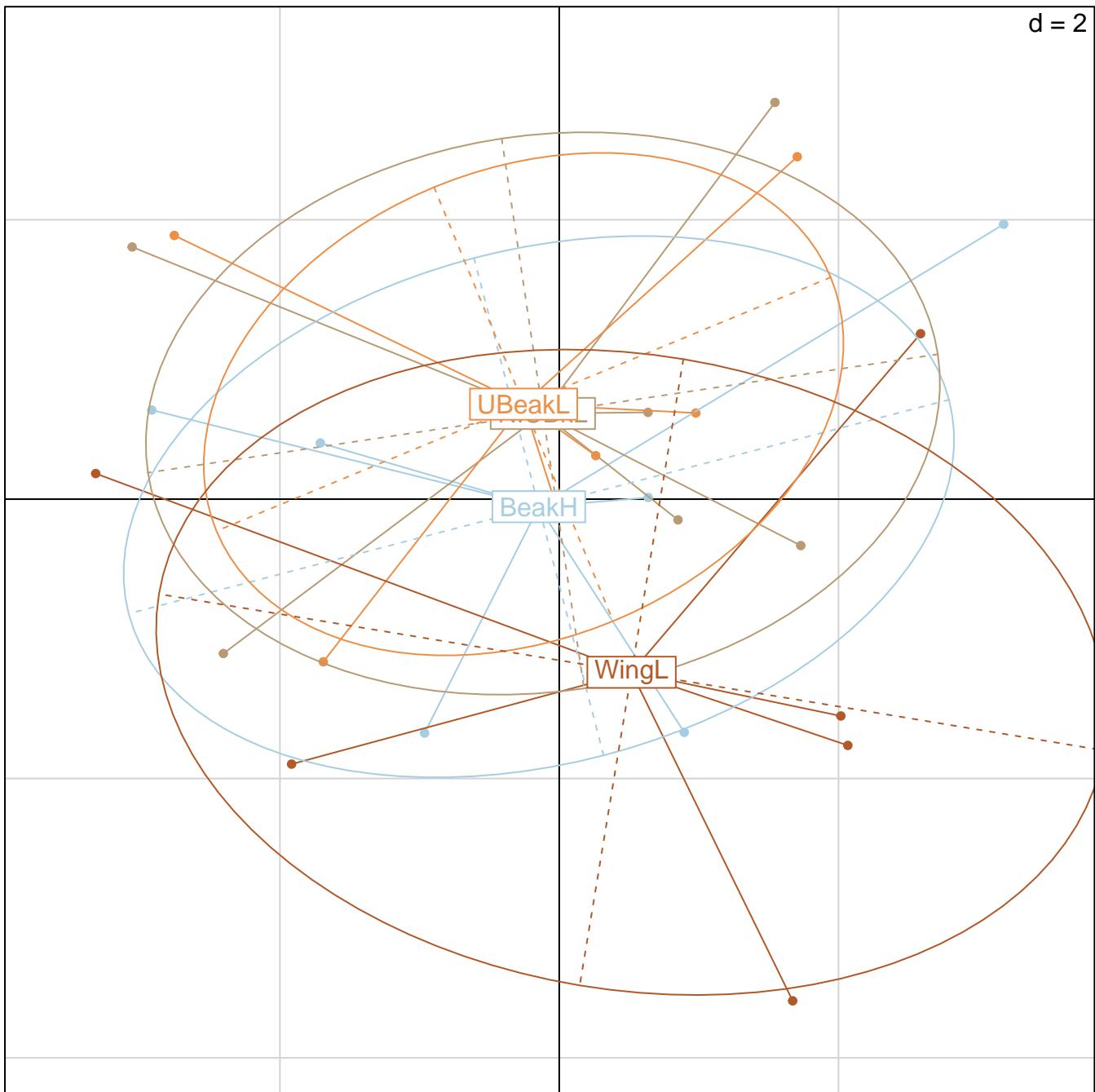


```
s.corcircle(res.dudi2$co)
```

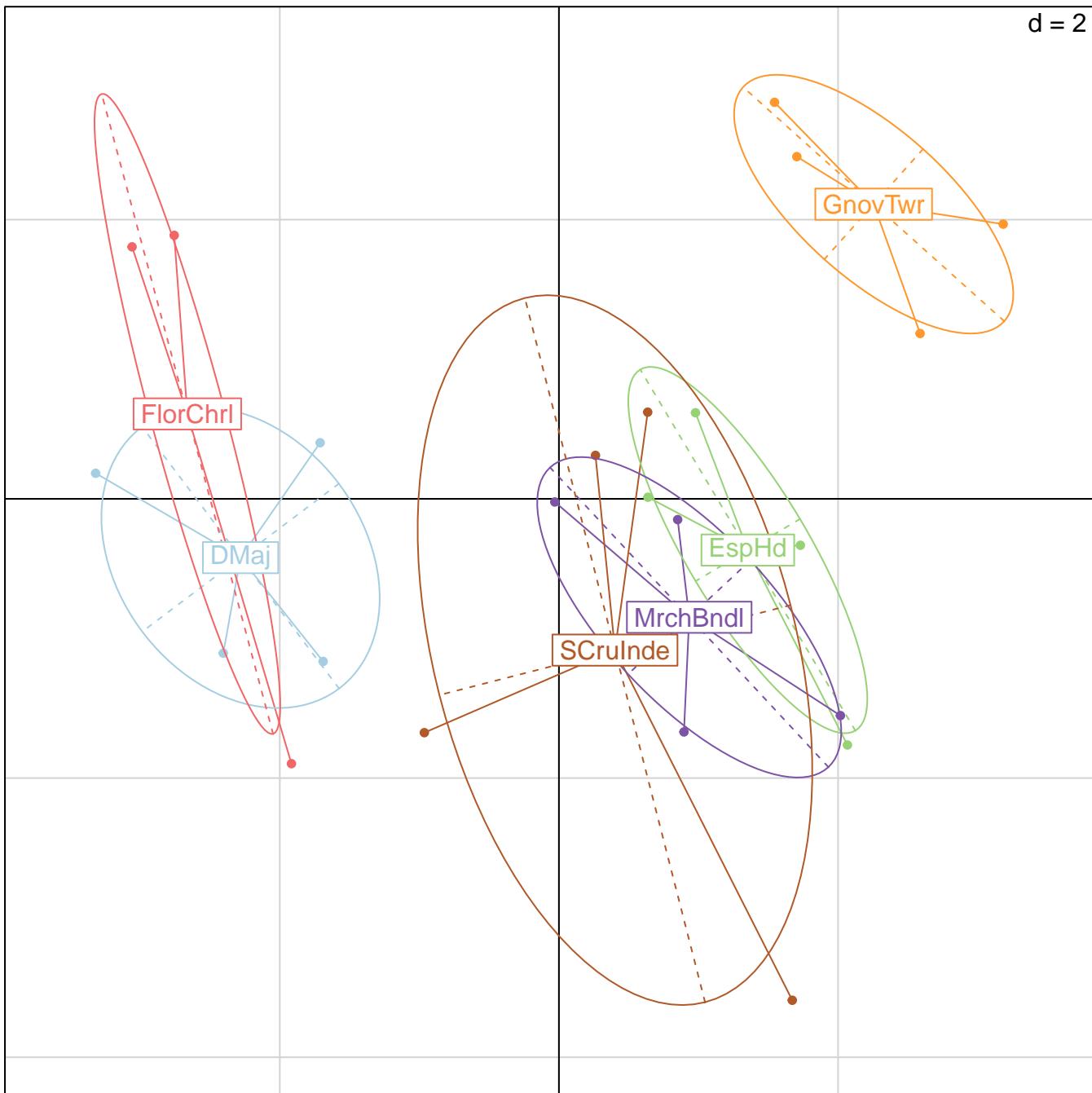


```
s.class(res.dudi2$li, as.factor(c(rep("WingL",6), rep("BeakH",6),
                                rep("UBeakL",6), rep("N.UBkL",6))),
       col = funky(4))
```

$d = 2$



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



8 Speed of computation

Speed of computation are saved throughout the vignette. For the specific system and R session describe above, we can determine the time of computation on darwin finch data with 99 permutations.

- `RaoRel` – > 0.009 s.
- `decompCTRE` – > 0.085 s.
- `partvar` – > 0.338 s.
- `Tstats` – > 19.906 s.
- `ComIndex`

- using four metrics – > 188.161 s.
 - with/without intraspecific variation – > 99.124 s.
- ComIndexMulti: for the calcul of hv.3 see [59](#) – > 215.529 s.

```
sessionInfo()

## R version 3.1.2 (2014-10-31)
## Platform: x86_64-pc-linux-gnu (64-bit)
##
## locale:
## [1] LC_CTYPE=fr_FR.UTF-8          LC_NUMERIC=C
## [3] LC_TIME=fr_FR.UTF-8          LC_COLLATE=fr_FR.UTF-8
## [5] LC_MONETARY=fr_FR.UTF-8       LC_MESSAGES=fr_FR.UTF-8
## [7] LC_PAPER=fr_FR.UTF-8          LC_NAME=C
## [9] LC_ADDRESS=C                  LC_TELEPHONE=C
## [11] LC_MEASUREMENT=fr_FR.UTF-8   LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats      graphics   grDevices utils      datasets  methods   base
##
## other attached packages:
## [1] rasterVis_0.32    hexbin_1.27.0    latticeExtra_0.6-26
## [4] RColorBrewer_1.1-2 raster_2.3-24    sp_1.0-17
## [7] geometry_0.3-5    magic_1.5-6     abind_1.4-0
## [10] adegenet_1.4-2    e1071_1.6-4    hypervolume_1.1.2
## [13] rgl_0.93.996     mice_2.22      lattice_0.20-29
## [16] Rcpp_0.11.4      cati_0.96     ape_3.2
## [19] ade4_1.6-2       nlme_3.1-119   knitr_1.9
##
## loaded via a namespace (and not attached):
## [1] class_7.3-11      cluster_2.0.1    colorspace_1.2-4
## [4] digest_0.6.8      evaluate_0.5.5   FD_1.0-12
## [7] formatR_1.0       ggplot2_1.0.0    grid_3.1.2
## [10] gtable_0.1.2     highr_0.4       htmltools_0.2.6
## [13] httpuv_1.3.2     igraph_0.7.1    MASS_7.3-37
## [16] Matrix_1.1-5     mgcv_1.8-3     mime_0.2
## [19] munsell_0.4.2    nnet_7.3-8      parallel_3.1.2
## [22] pdist_1.2        permute_0.8-3   plyr_1.8.1
## [25] proto_0.3-10    R6_2.0.1       randomForest_4.6-10
## [28] reshape2_1.4.1   RJSONIO_1.3-0   rpart_4.1-8
## [31] scales_0.2.4     shiny_0.11.1   stringr_0.6.2
## [34] tools_3.1.2      vegan_2.2-1    xtable_1.7-4
## [37] zoo_1.7-11
```

```
proc.time_RaoRel  
proc.time_decompCTRE  
proc.time_partvar  
proc.time_Tstats  
proc.time_ComIndex1  
proc.time_ComIndex.with_without  
proc.time_ComIndexMulti
```

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](#).