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

August 27, 2014

Abstract

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

*adrien.taudiere@cefe.cnrs.fr

Contents

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 0.91. 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", dep=TRUE)
install.packages("C:/Users/taudiere/Desktop/cati/pkg/cati_0.91.zip", repos=NULL)

## Installing package into 'C:/Users/taudiere/Documents/R/win-library/3.1'
## (as 'lib' is unspecified)

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] "0.91"
```

cati version should read 0.91.

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* page, click 'packages', 'cati', and 'cati-package'.

2.3 Data presentation: Darwin finches in Galapagos Island

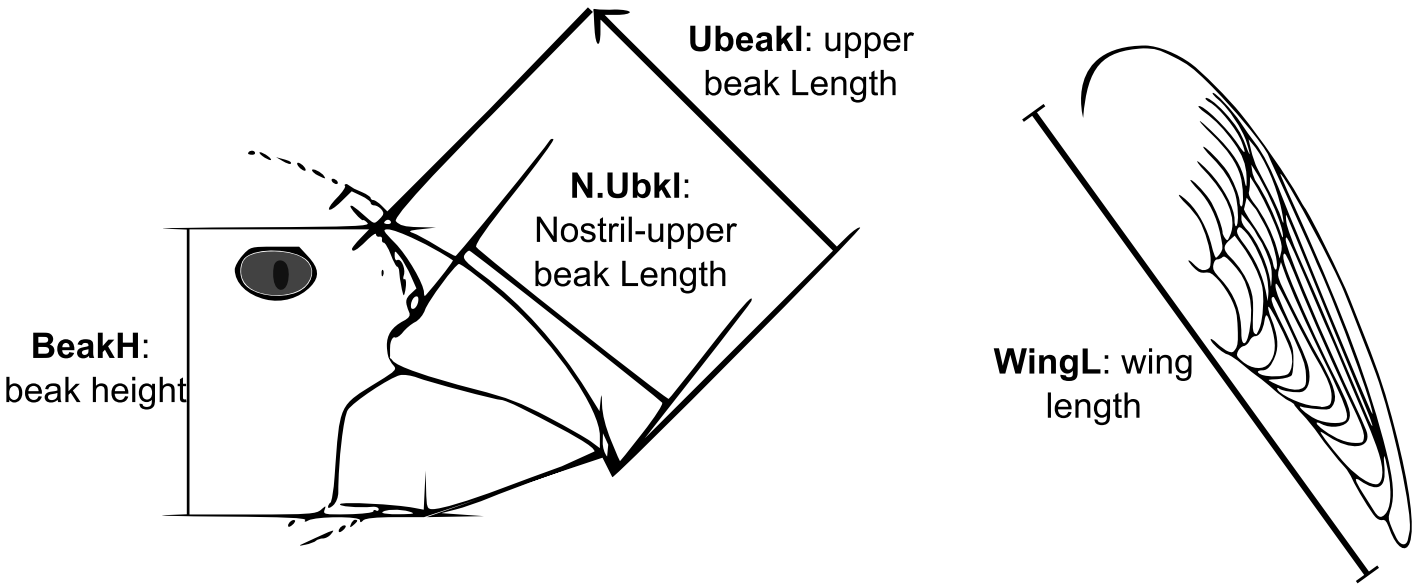
First we need to load the data.

```
data(finch.ind)

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

Now we can see 3 objects: a traits matrix

```
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 site
```

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

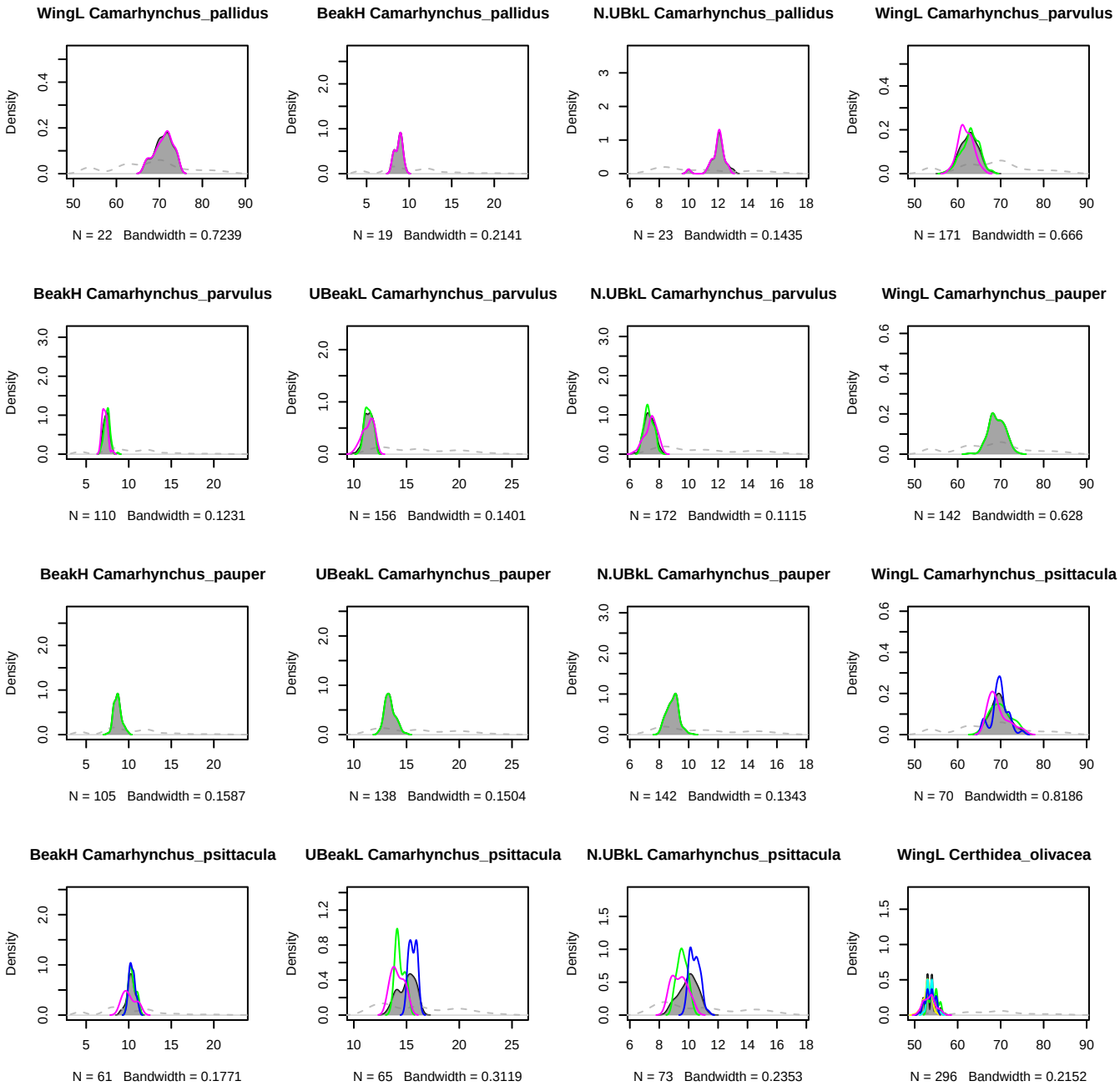


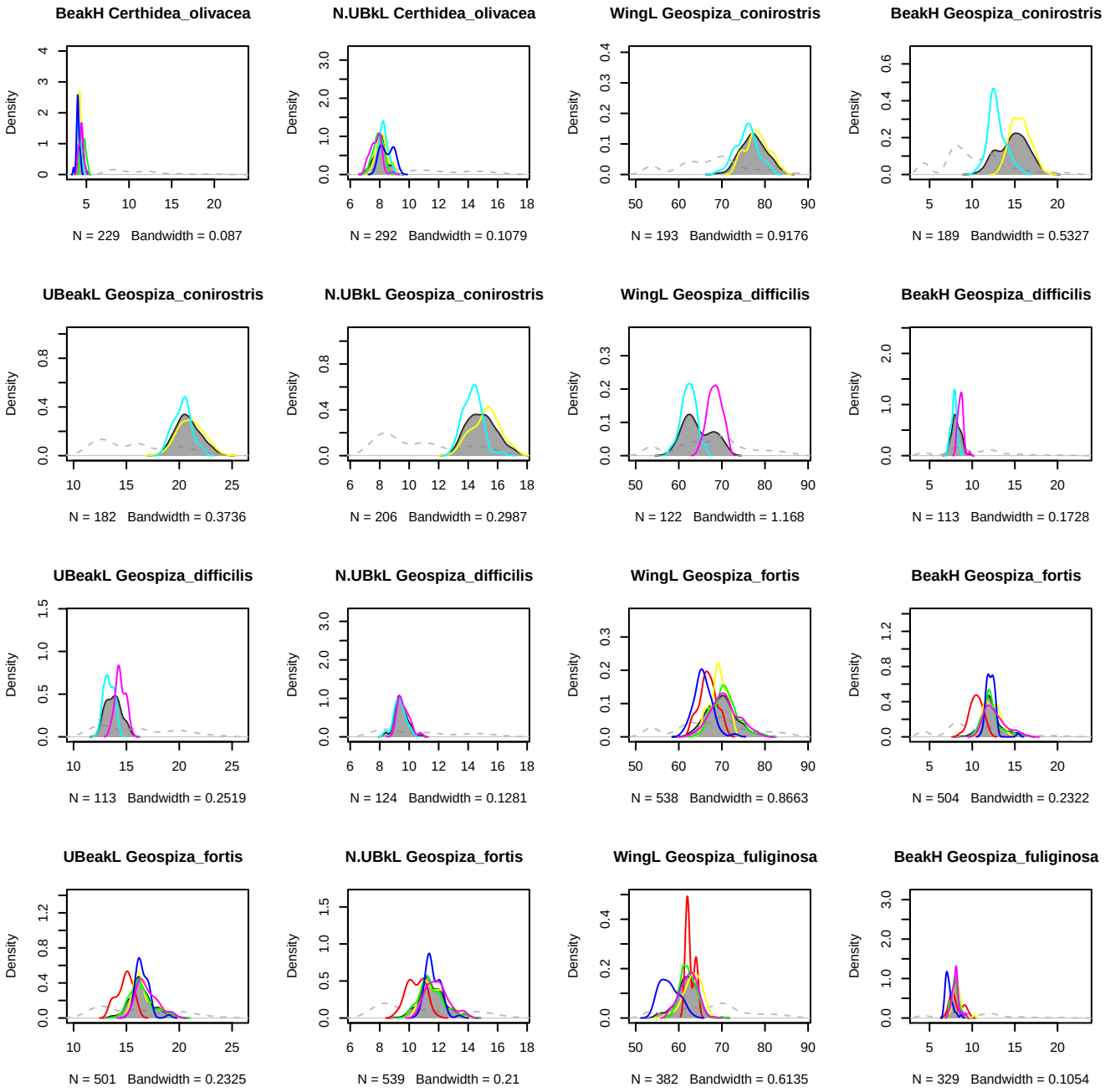
3 Description of traits distributions

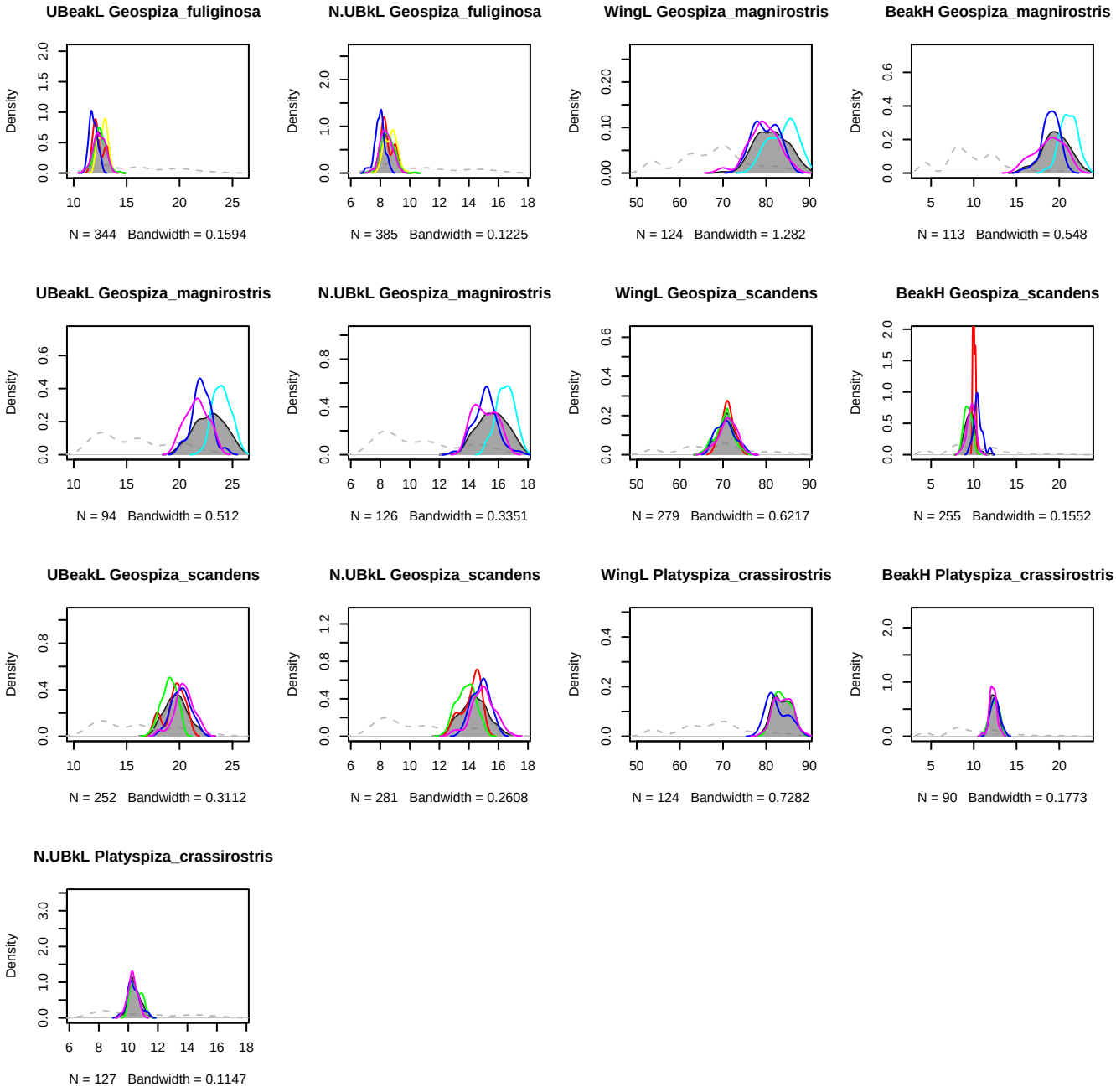
3.1 Plot the kernel density of traits

Plot the distribution of traits values for populations, species, sites and regional scales. First, let try the distribution for all populations of Darwin finches.

```
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)
```

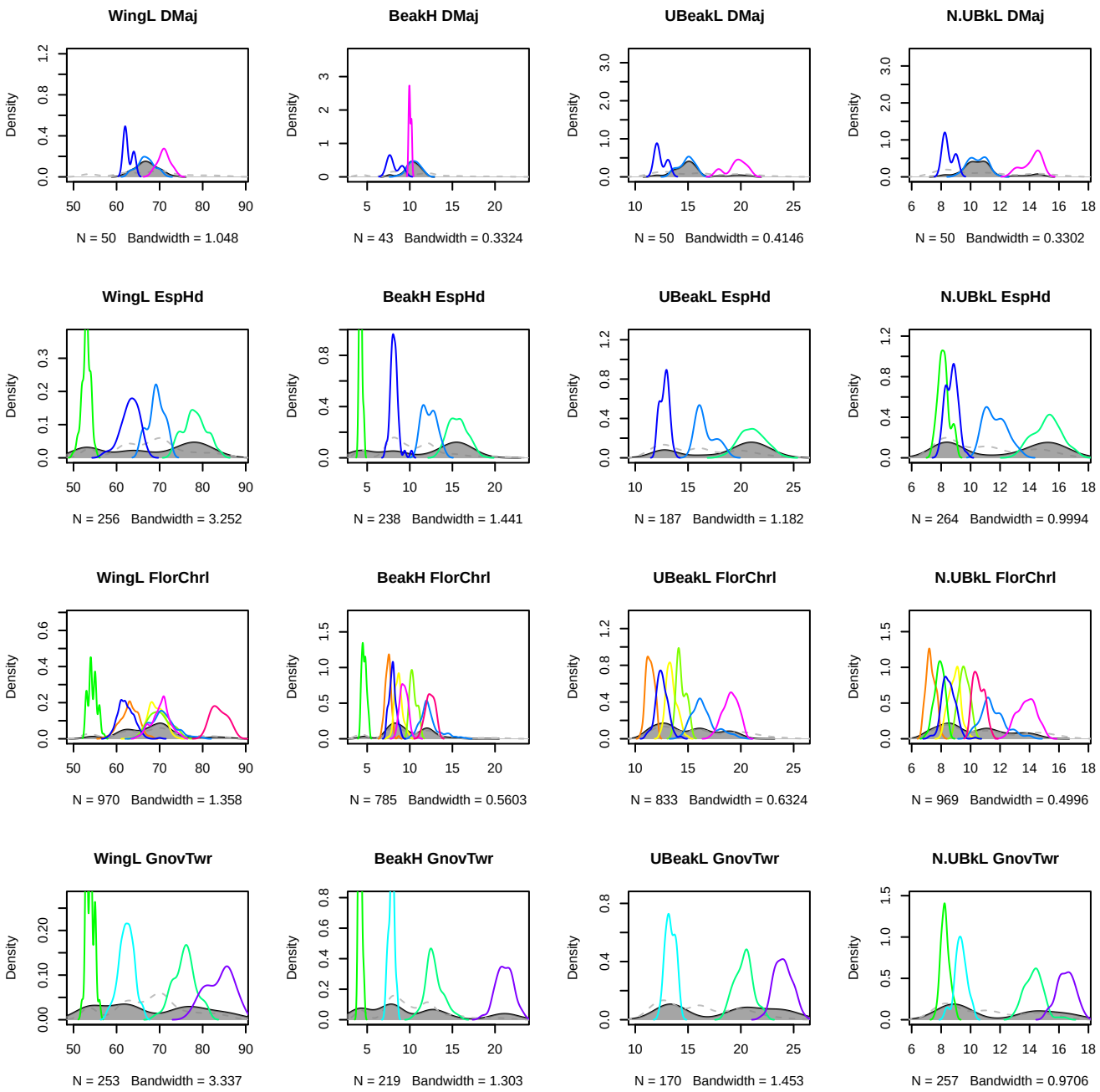


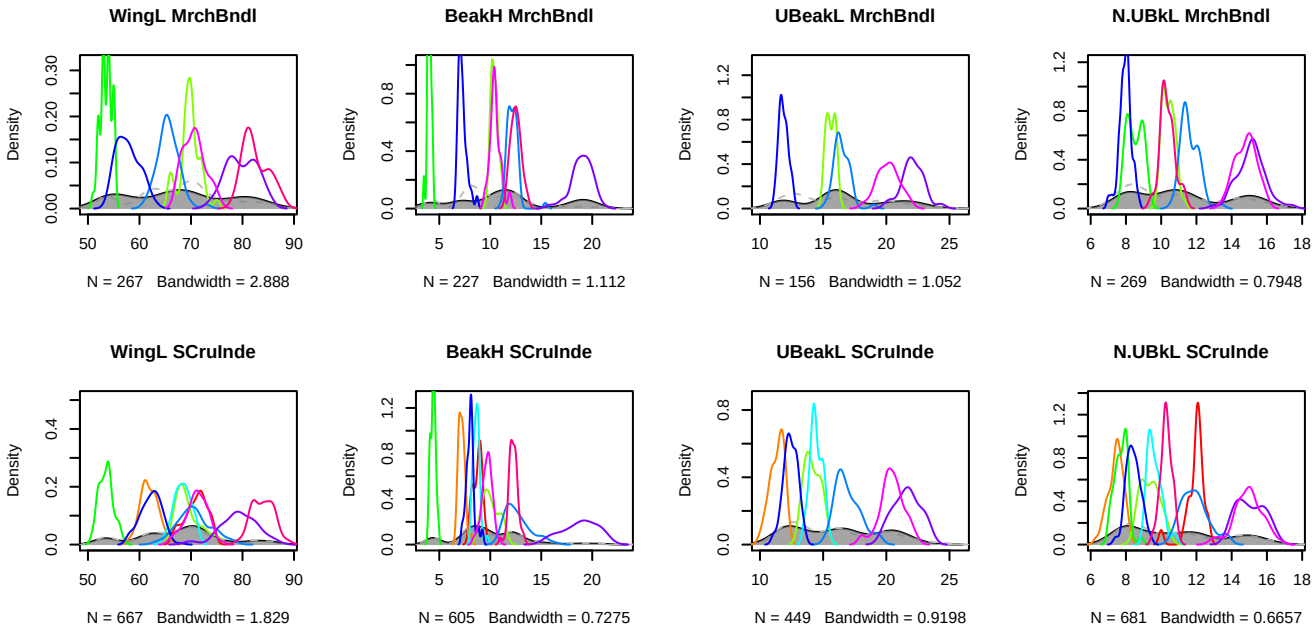

BeakH Certhidea_olivacea
N = 229 Bandwidth = 0.087
N.UBkL Certhidea_olivacea
N = 292 Bandwidth = 0.1079
WingL Geospiza_conirostris
N = 193 Bandwidth = 0.9176
BeakH Geospiza_conirostris
N = 189 Bandwidth = 0.5327
UBeakL Geospiza_conirostris
N = 182 Bandwidth = 0.3736
N.UBkL Geospiza_conirostris
N = 206 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


UBeakL Geospiza_fuliginosa
Density
N = 344 Bandwidth = 0.1594
N.UBkL Geospiza_fuliginosa
Density
N = 385 Bandwidth = 0.1225
WingL Geospiza_magnirostris
Density
N = 124 Bandwidth = 1.282
BeakH Geospiza_magnirostris
Density
N = 113 Bandwidth = 0.548
UBeakL Geospiza_magnirostris
Density
N = 94 Bandwidth = 0.512
N.UBkL Geospiza_magnirostris
Density
N = 126 Bandwidth = 0.3351
WingL Geospiza_scandens
Density
N = 279 Bandwidth = 0.6217
BeakH Geospiza_scandens
Density
N = 255 Bandwidth = 0.1552
UBeakL Geospiza_scandens
Density
N = 252 Bandwidth = 0.3112
N.UBkL Geospiza_scandens
Density
N = 281 Bandwidth = 0.2608
WingL Platyspiza_crassirostris
Density
N = 124 Bandwidth = 0.7282
BeakH Platyspiza_crassirostris
Density
N = 90 Bandwidth = 0.1773
N.UBkL Platyspiza_crassirostris
Density
N = 127 Bandwidth = 0.1147

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

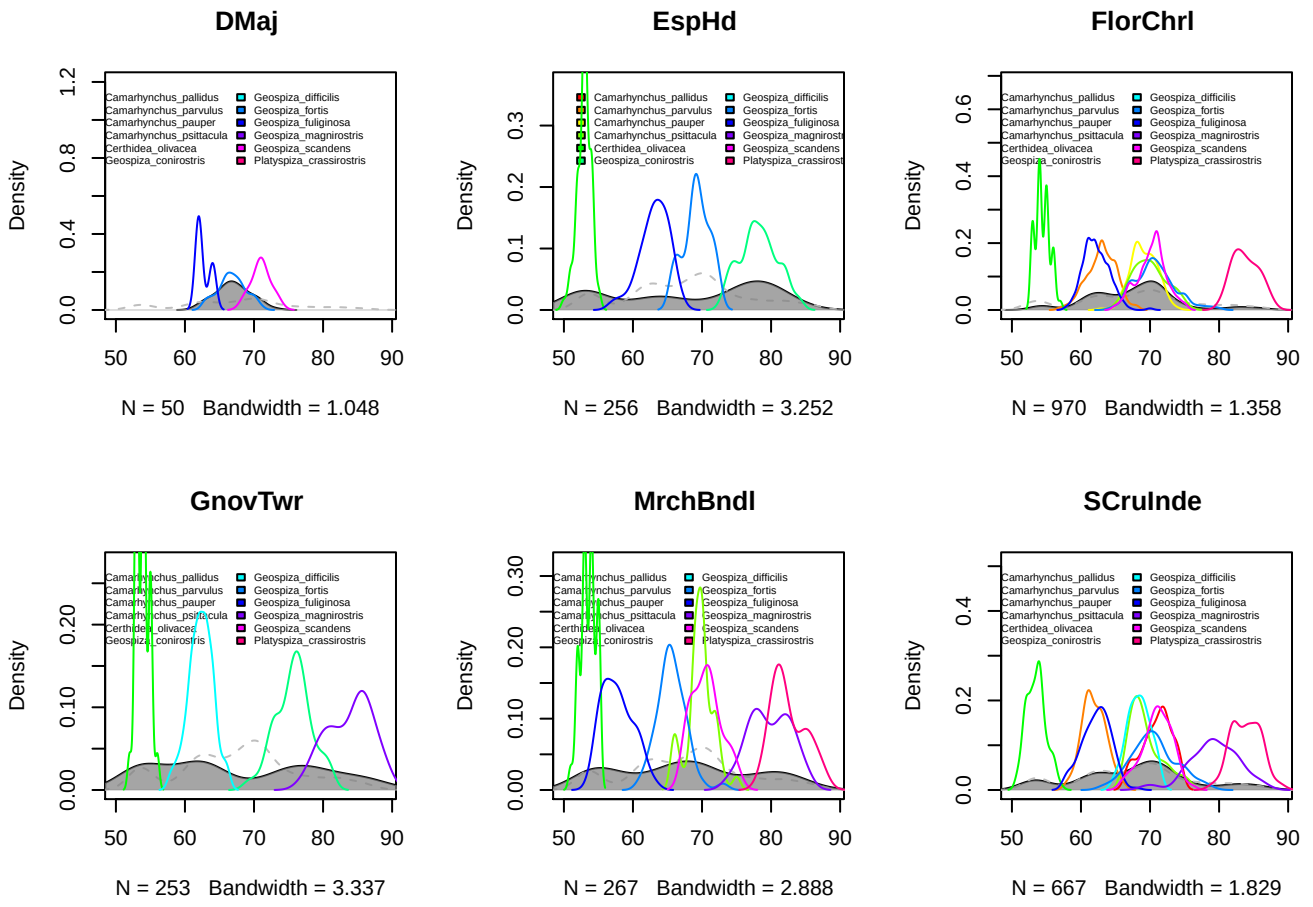
```
par(mfrow=c(4,4), cex=0.5)
plotDistri(traits.finch, ind.plot.finch, sp.finch,
           ylim.cex=8, plot.ask=F, multipanel=F, leg=F)
```





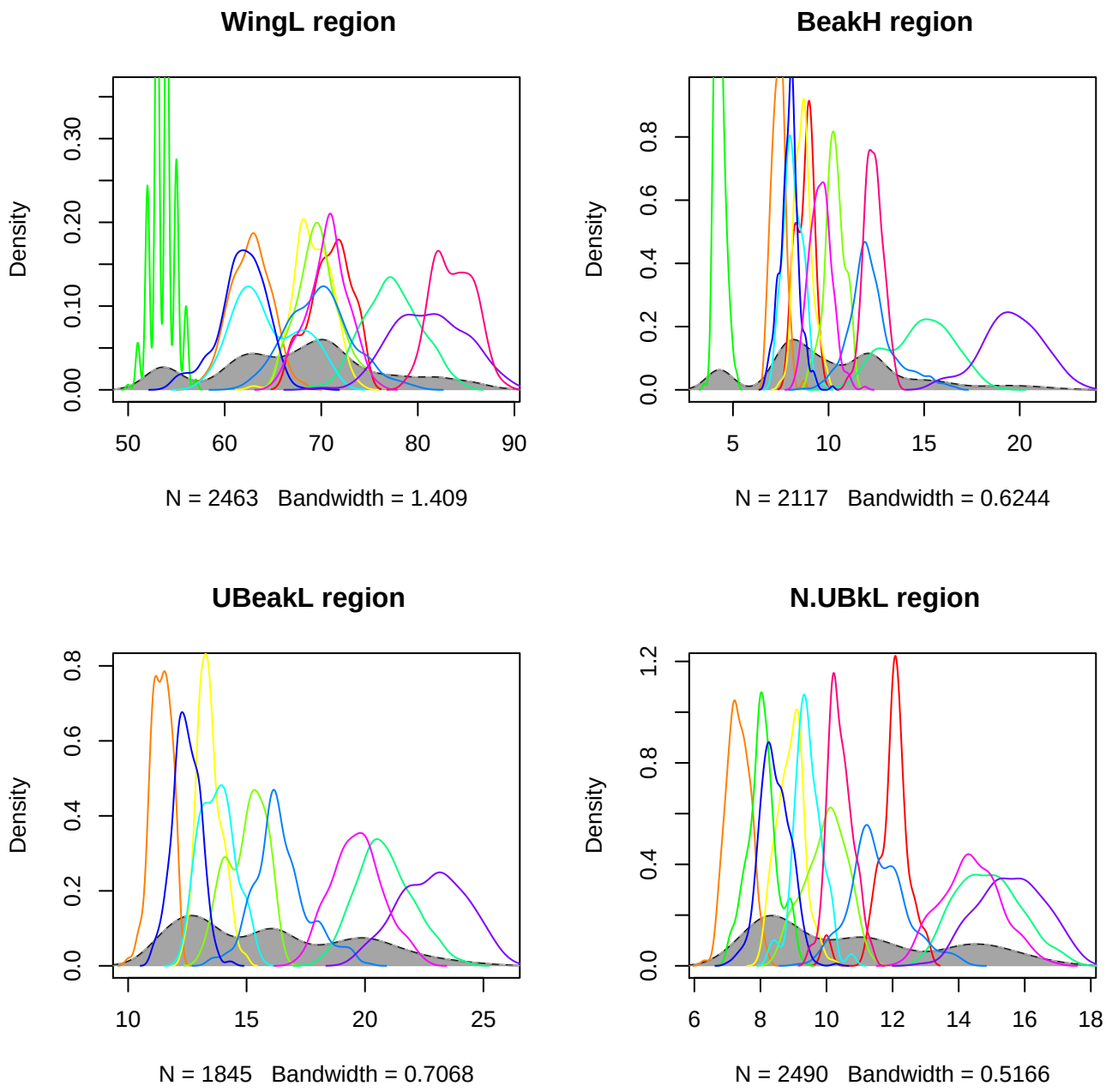
Only one trait to plot using leg=T to plot the legend.

```
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)
```

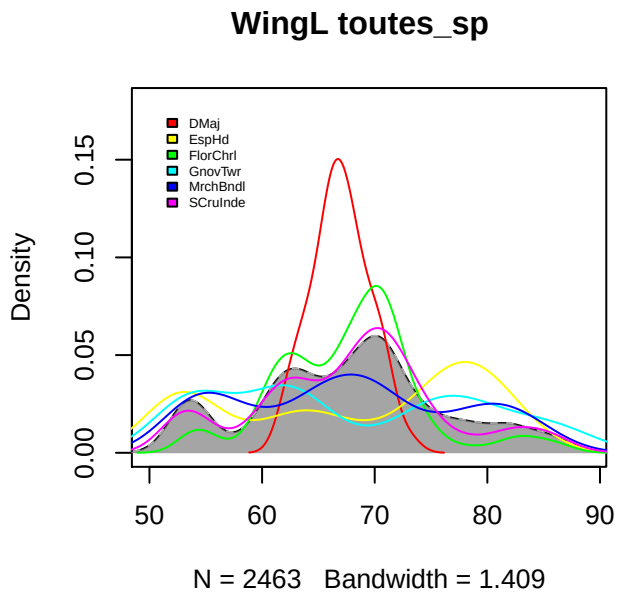


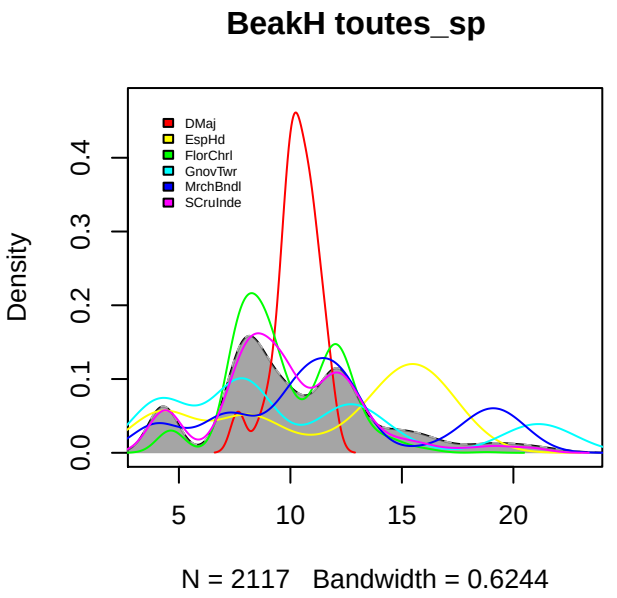
If we want to plot all the sites (regional distribution) or all the species: we can use the folowing code:

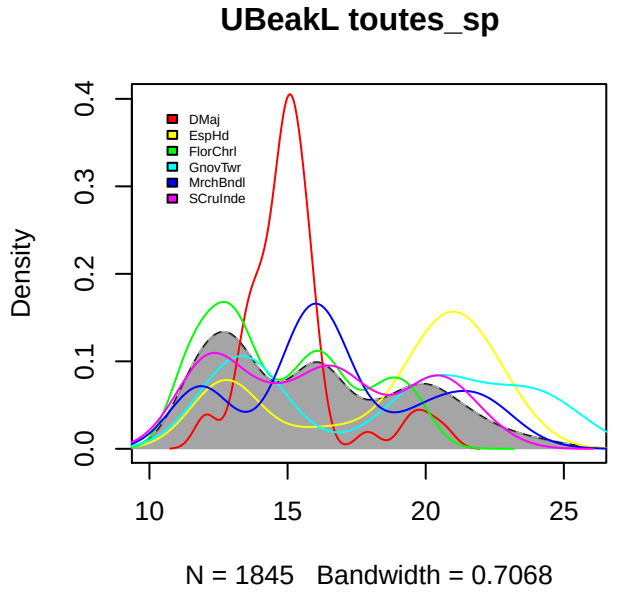
```
par(mfrow=c(4,4), cex=0.5)
plotDistri(traits.finch, rep("region", times=dim(traits.finch)[1]),
           sp.finch, ylim.cex=6, plot.ask=F, leg=F)
```

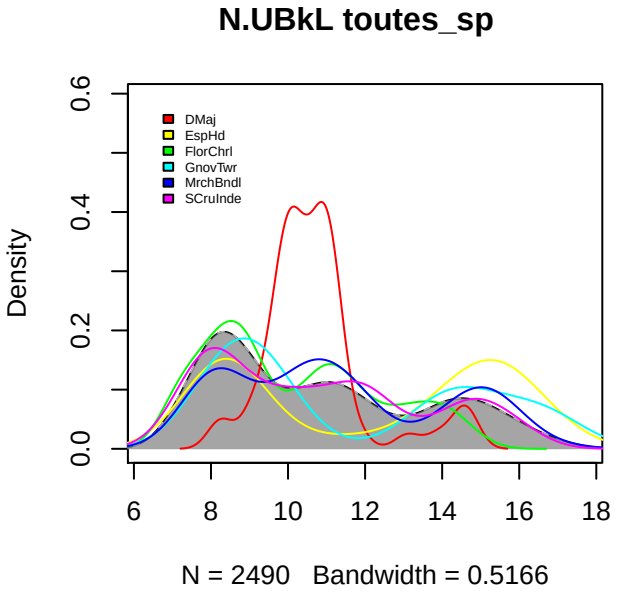



```
plotDistri(traits.finch, rep("toutes_sp", times=dim(traits.finch)[1]),
          ind.plot.finch, ylim.cex=3, plot.ask=F, cex.leg=0.5)
```









Reset the default parameter

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

4 Decomposition of variances

4.1 Decomposition of within/among species variances using rao diversity

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

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

Where: $\gamma is the diversity of the regional pool, \alpha is the diversity of the local community and \beta is the turnover between$

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 functionnal space (i.e. considering all traits together).

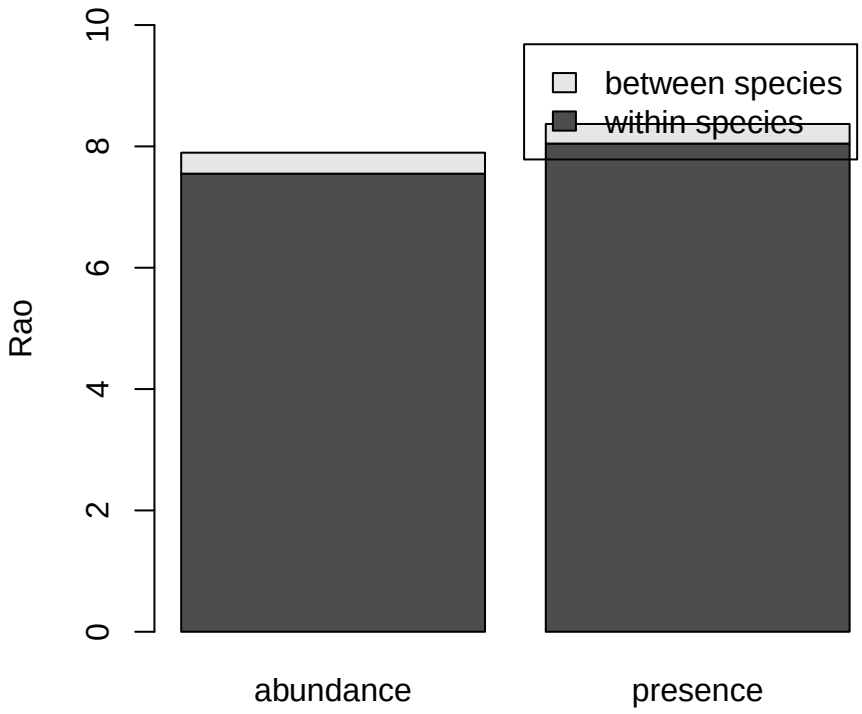
```
#create individuals community matrix
comm<-t(table(ind.plot.finch,1:length(ind.plot.finch)))
#create species community matrix
comm.sp<-table(sp.finch, ind.plot.finch)
class(comm.sp)<-"matrix"

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

witRao+betRao

## [1] 8.37

totRao

## [1] 8.37
```

Now let's take the abundance 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

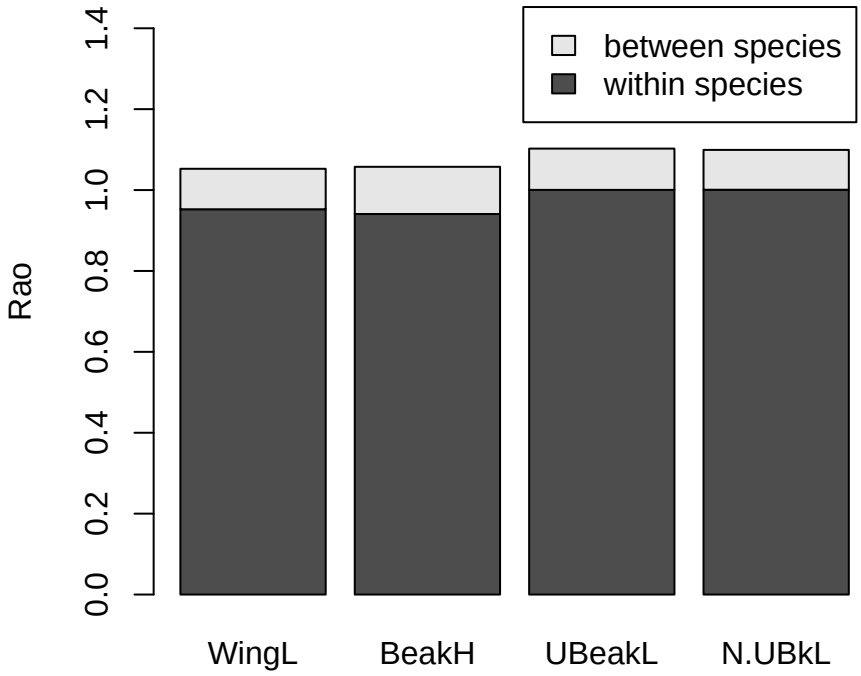
We can also do this analysis for each trait separately. We need to replace (or exclude) NA values. For this example, we use the package mice to complete the data.

```
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.

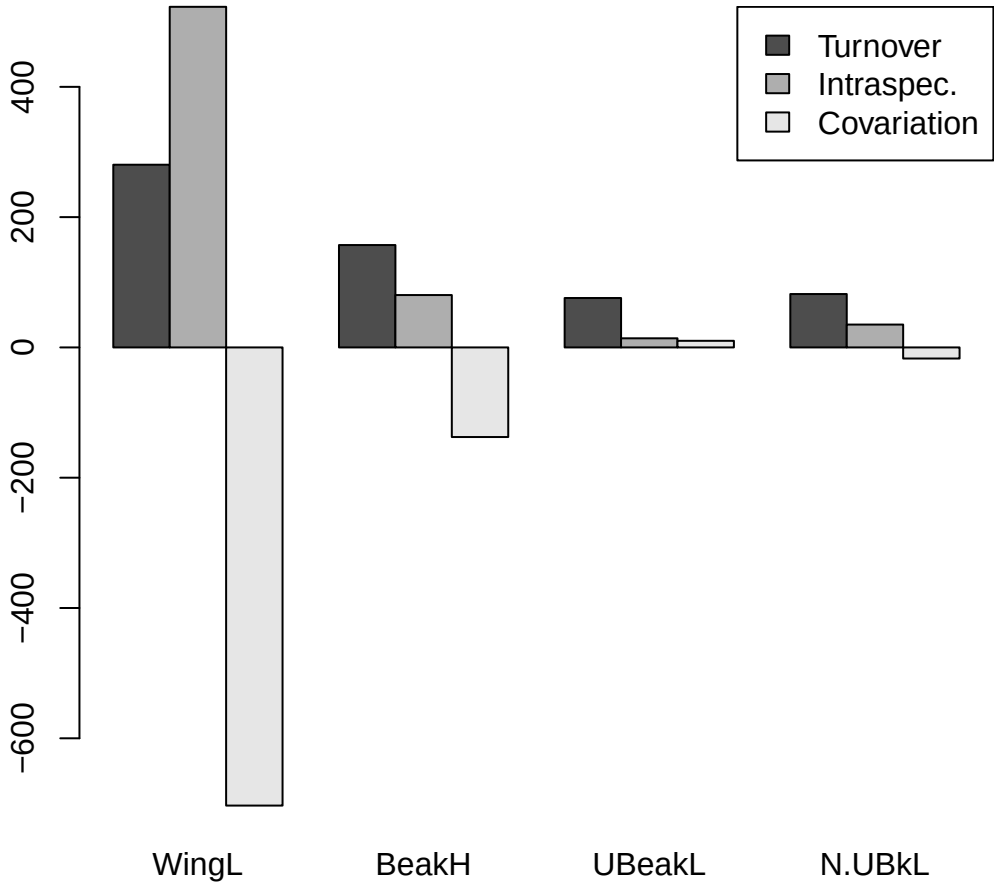
```
barplot(t(cbind( witRao.w.bytrait, betRao.w.bytrait)),
        names.arg = colnames(traits.finch),
        legend.text=c("within species", "between species"),
        ylab="Rao", ylim=c(0,1.5))
```



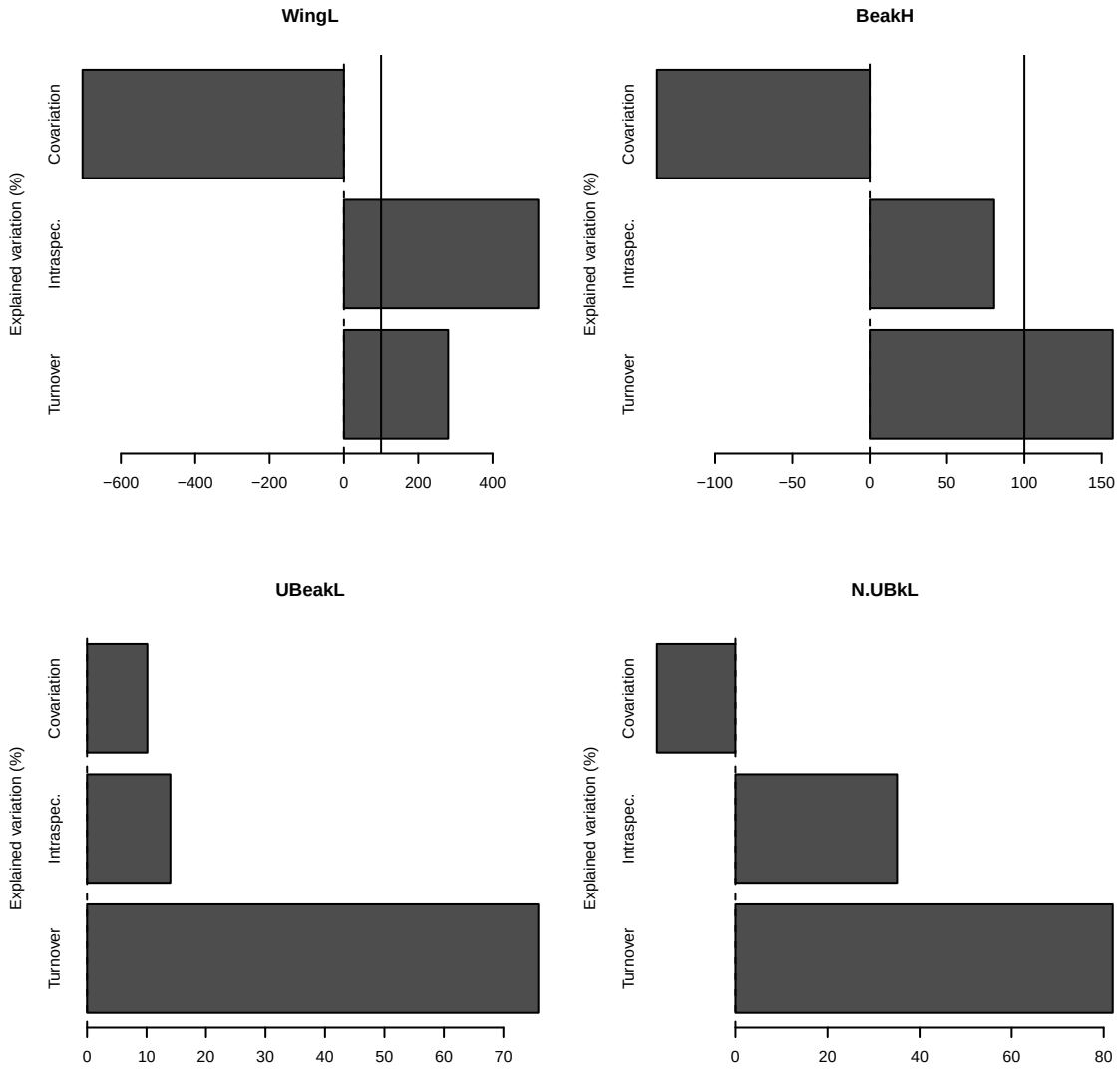
4.2 Decomposition of community trait response to environment into intraspecific trait variability, variability due to species turnover and their covariation.

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))
```

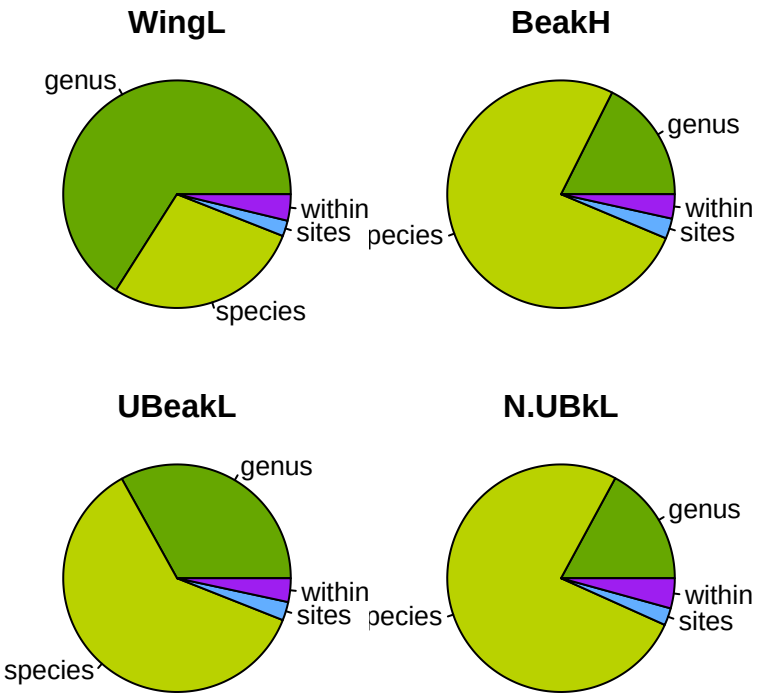
4.3 Decomposition of traits variances using nested factors

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

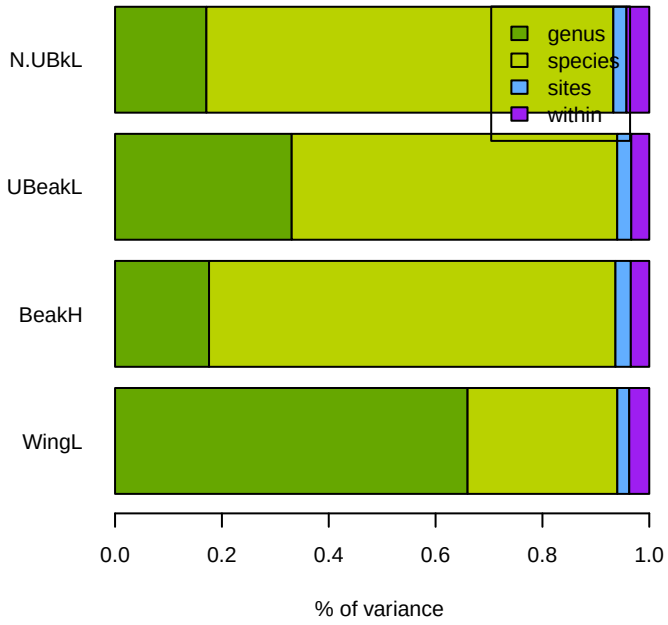
Reference: Messier, J., McGill, B. and Lechowicz, M. (2010) How do traits vary across ecological scales? A case for trait-based ecology. Ecology Letters, 13, 838-848.

```
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))
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)
```



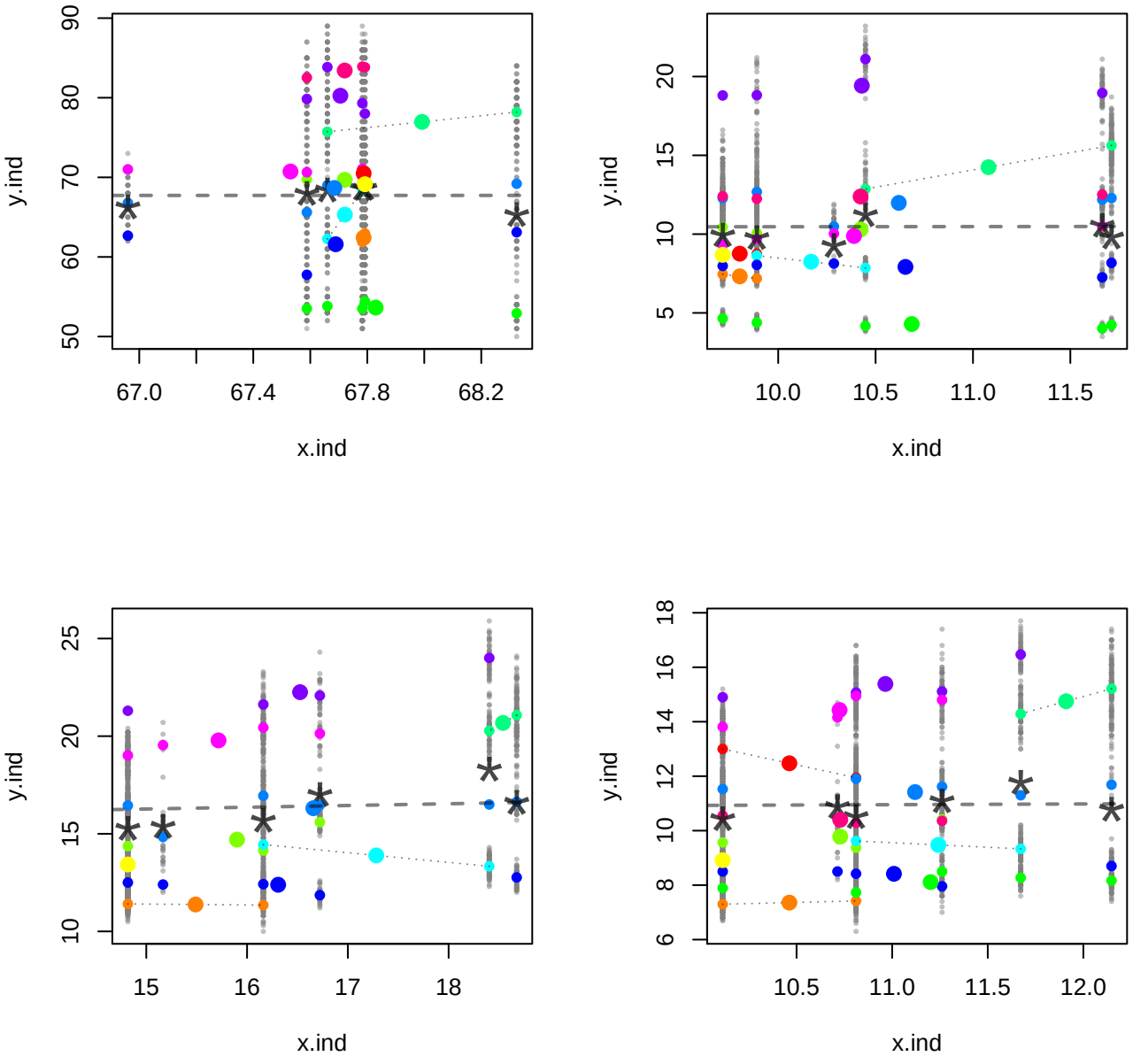
```
par(old.par)
barPartvar(res.partvar.finch, col=colors,
           leg=TRUE)
```



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

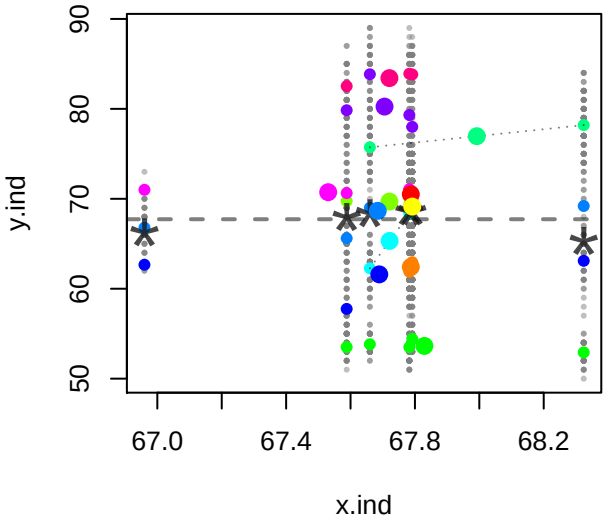
For an example of utilisation see: Cornwell, W.K. and Ackerly, D.D., 2009. Community assembly and shifts in plant trait distributions across an environmental gradient in coastal California. Ecological Monographs 79, 109-126.

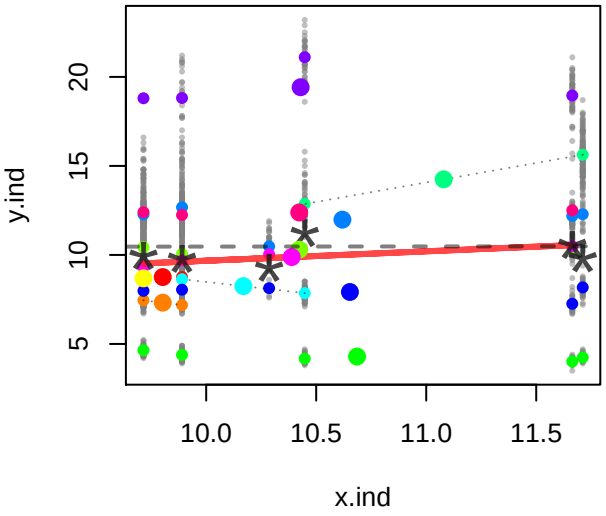
```
plotSpPop(traits.finch, ind.plot.finch, sp.finch, silent=TRUE)
```

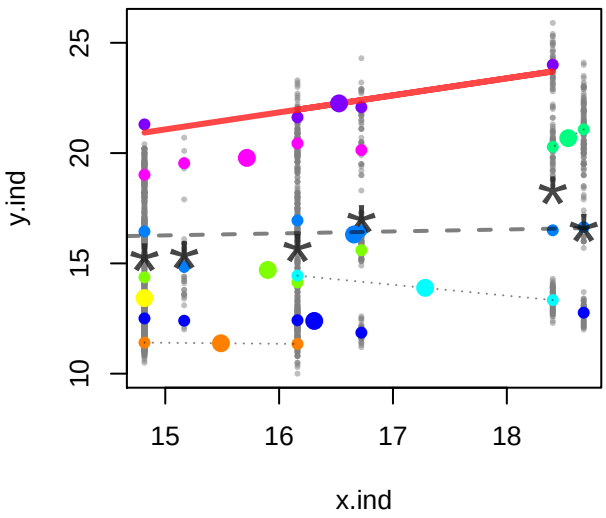


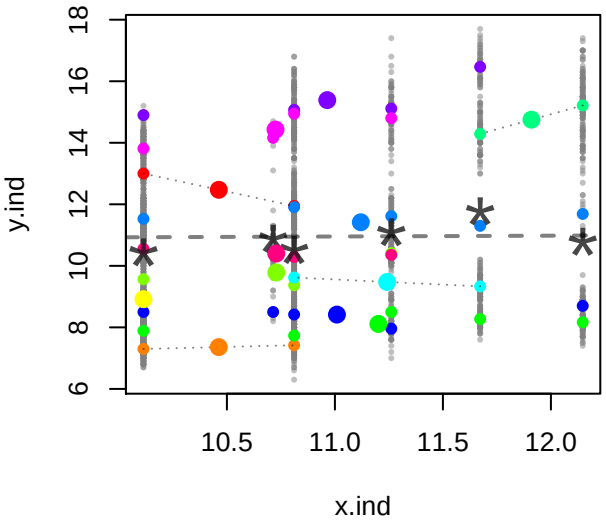
If we change the value of the threshold (alpha=10% instead of 5% and the minimum individual to represent singificativity fixed to 3 instead of 10 by default) we can see some significant relationships.

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



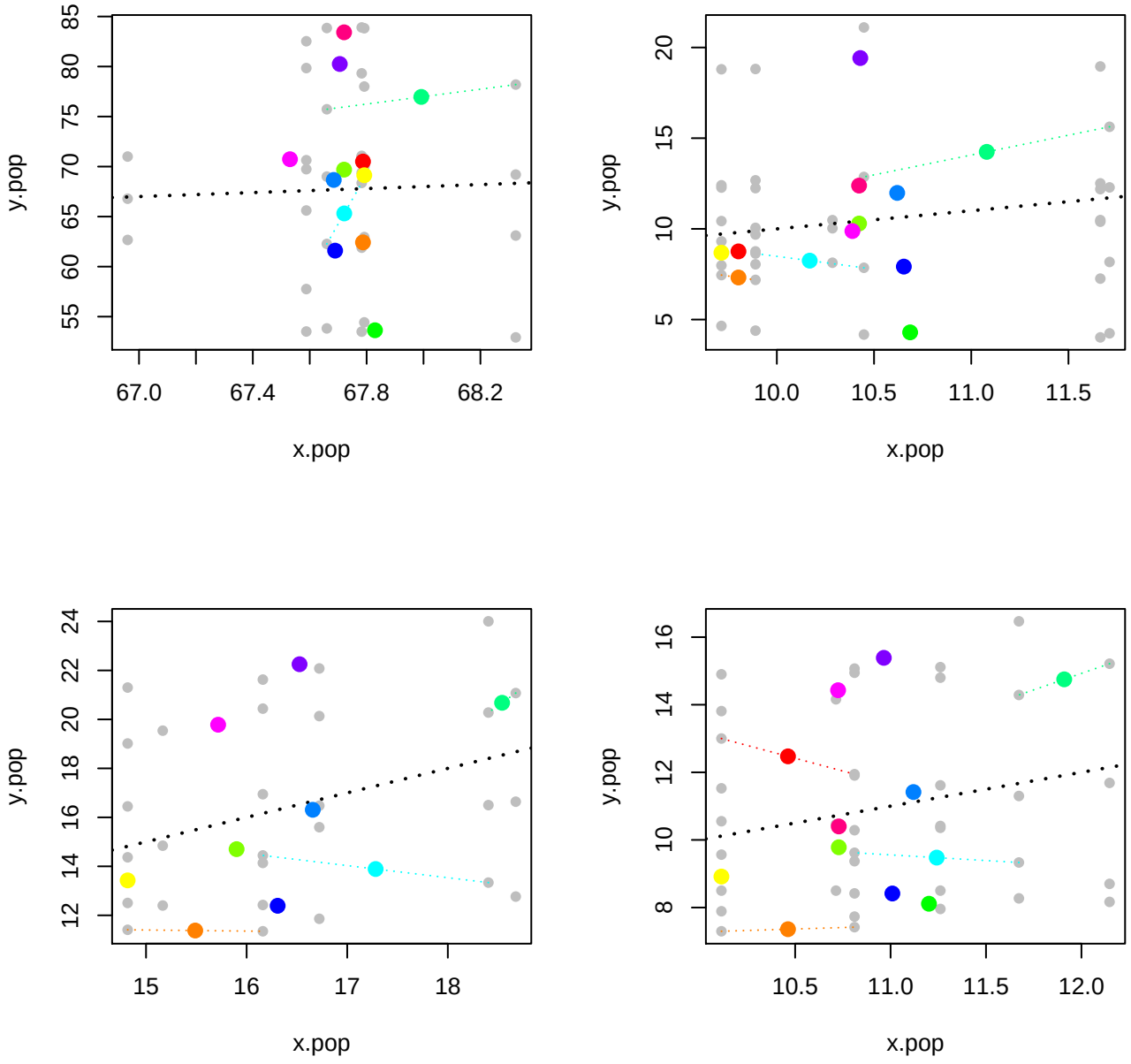




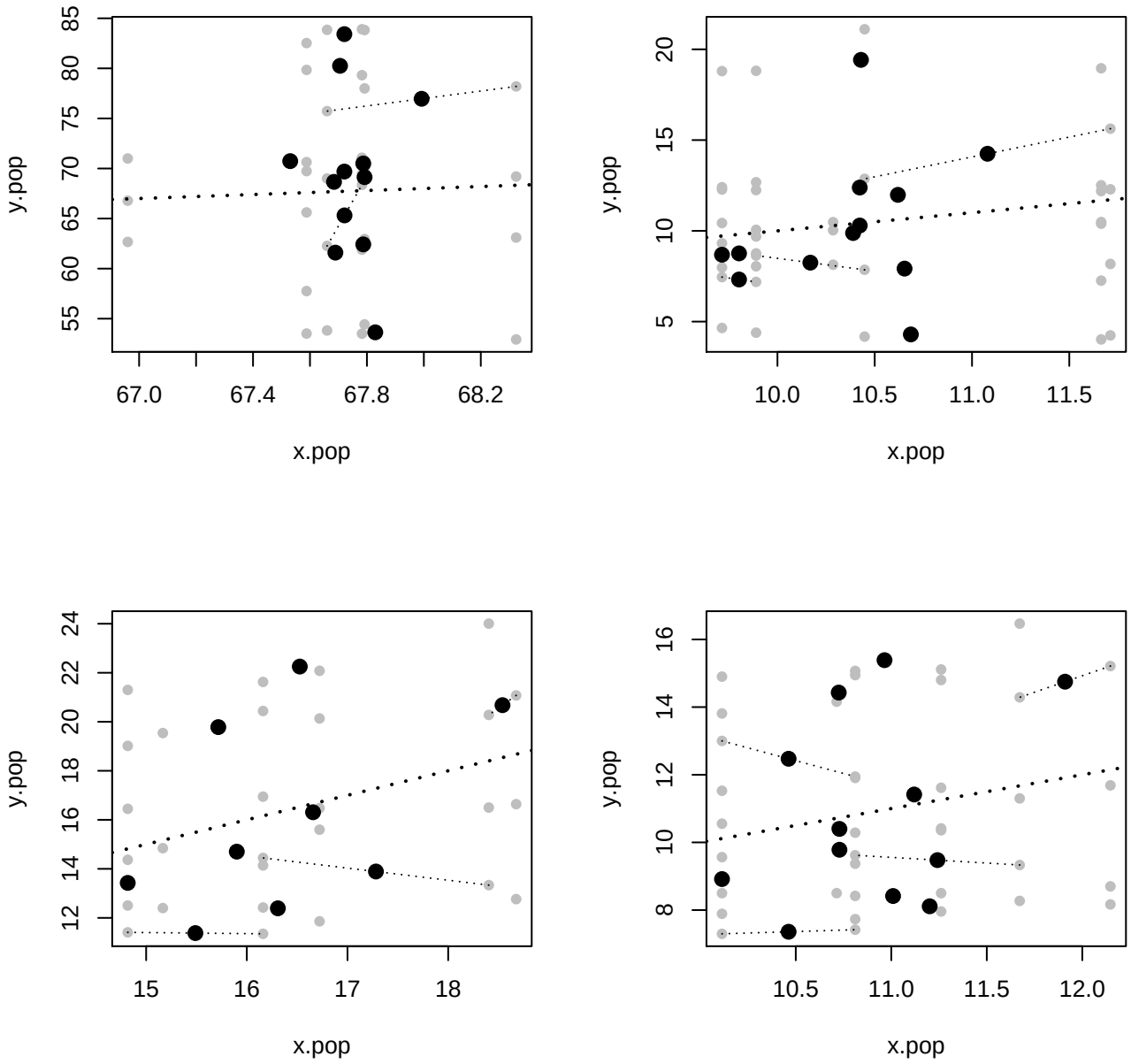


For a more simple figure, add the option resume=TRUE. Again if we change the value of the threshold (alpha=10% instead of 5% and the minimum individual to represent singificativity fixed to 3 instead of 10 by default) we can see some significant relationships.

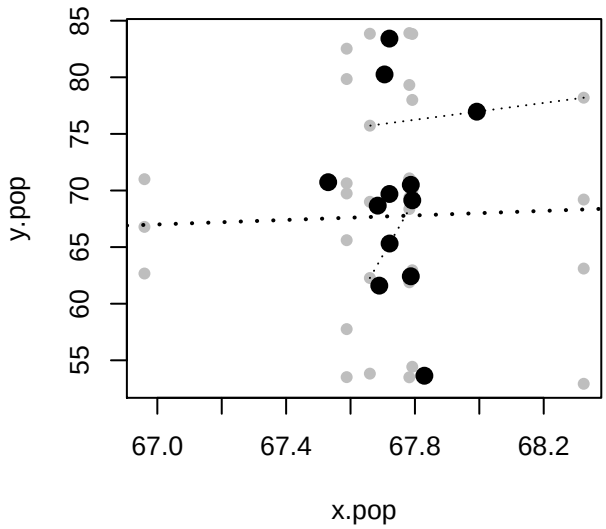
```
plotSpPop(traits.finch, ind.plot.finch, sp.finch,
          silent=TRUE, resume=TRUE, col.pop="grey")
```

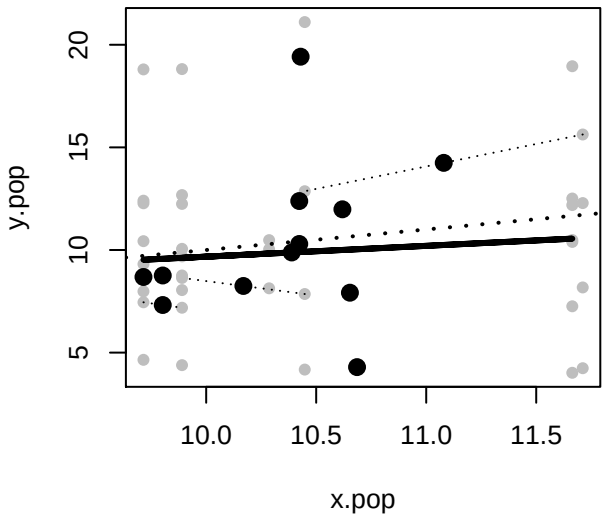


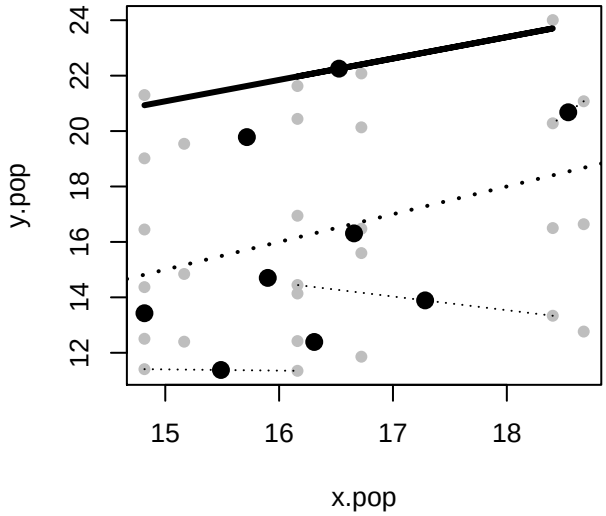

```
plotSpPop(traits.finch, ind.plot.finch, sp.finch,
          silent=TRUE, resume=TRUE, col.pop="grey", col.sp="black")
```

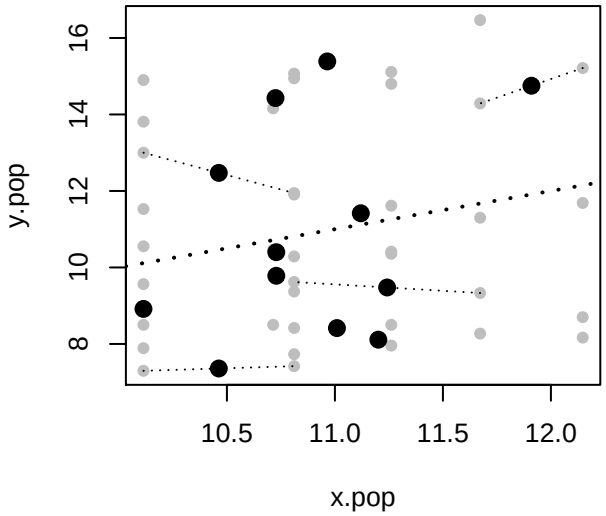



```
plotSpPop(traits.finch, ind.plot.finch, sp.finch,
          silent=TRUE, resume=TRUE, col.pop="grey", col.sp="black",
          p.val=0.1,  min.ind.signif=3)
```


y.pop
55
60
65
70
75
80
85
67.0
67.4
67.8
68.2
x.pop


y.pop
5
10
15
20
10.0
10.5
11.0
11.5
x.pop


y.pop
12
14
16
18
20
22
24
15
16
17
18
x.pop


y.pop
8
10
12
14
16
10.5
11.0
11.5
12.0
x.pop

5 Test of community assembly

5.1 Ratio of variances: T-statistics

The funcion Tstats 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 null models.

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.

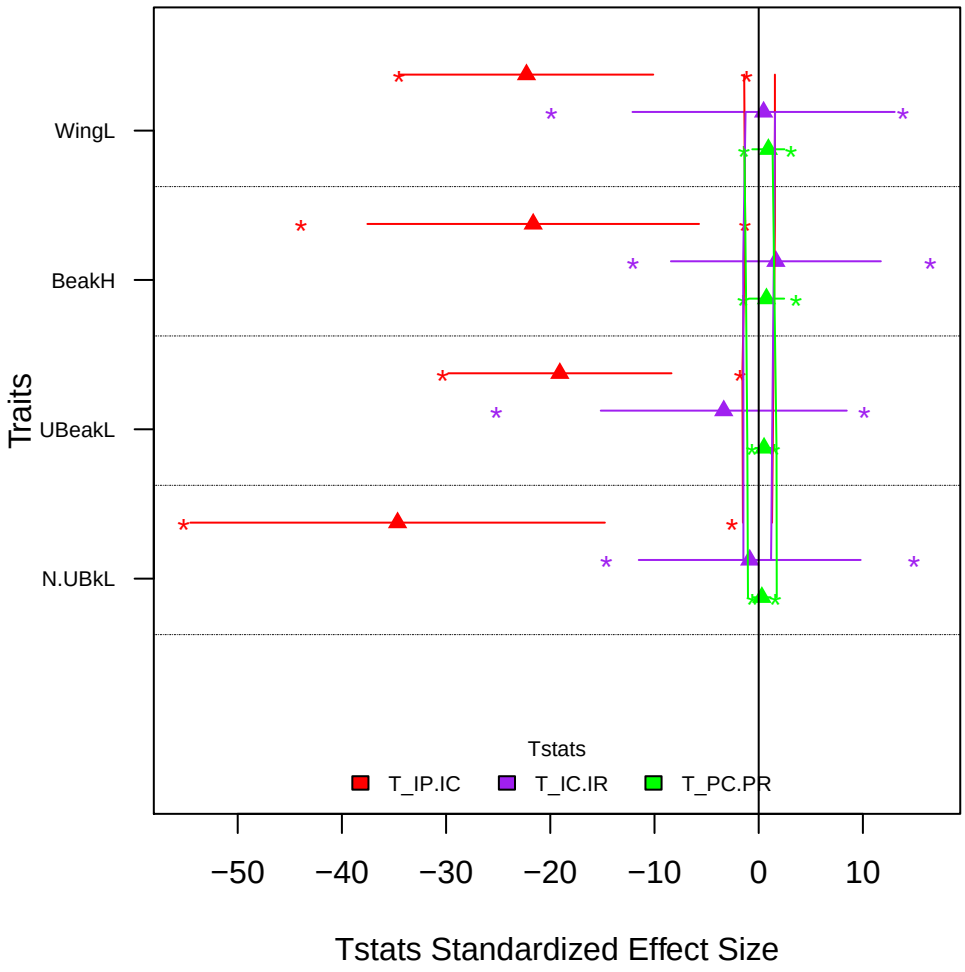
```
res.finch<-Tstats(traits.finch, ind.plot=ind.plot.finch, sp=sp.finch,
                  nperm=9, print=FALSE)
res.finch

## $Tstats
##            Length Class  Mode
## T_IP.IC     24    -none- numeric
## T_IC.IR     24    -none- numeric
## T_PC.PR     24    -none- numeric
## T_IP.IC_nm 216    -none- numeric
## T_IC.IR_nm 216    -none- numeric
## T_PC.PR_nm 216    -none- numeric
##
## $variances
##               Length Class  Mode
## var_IP         156   -none- numeric
## var_PC          24   -none- numeric
## var_CR           4   -none- numeric
## var_IC          24   -none- numeric
## var_PR           4   -none- numeric
## var_IR           4   -none- numeric
## var_IP_nm1    1404   -none- numeric
## var_PC_nm2sp   216   -none- numeric
## var_IC_nm1     216   -none- numeric
## var_IC_nm2     216   -none- numeric
## var_PR_nm2sp    36   -none- numeric
## var_IR_nm2      36   -none- numeric

attributes(res.finch)

## $names
## [1] "Tstats"    "variances"
##
## $class
## [1] "Tstats"

str(res.finch)
```

```
## List of 2
##  $ Tstats   :List of 6
##   ..$ T_IP.IC   : num [1:6, 1:4] 0.3831 0.0343 0.1084 0.0417 0.0496 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:6] "DMaj" "EspHd" "FlorChrl" "GnovTwr" ...
##   .. .. ..$ : chr [1:4] "WingL" "BeakH" "UBeakL" "N.UBkL"
##   ..$ T_IC.IR   : num [1:6, 1:4] 0.0925 1.7103 0.5752 1.7916 1.3718 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:6] "DMaj" "EspHd" "FlorChrl" "GnovTwr" ...
##   .. .. ..$ : chr [1:4] "WingL" "BeakH" "UBeakL" "N.UBkL"
##   ..$ T_PC.PR   : num [1:6, 1:4] 0.226 1.468 0.871 1.762 1.476 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:6] "DMaj" "EspHd" "FlorChrl" "GnovTwr" ...
##   .. .. ..$ : chr [1:4] "WingL" "BeakH" "UBeakL" "N.UBkL"
##   ..$ T_IP.IC_nm: num [1:9, 1:4, 1:6] 0.948 1 3.226 2.291 2.216 ...
##   .. ..- attr(*, "dimnames")=List of 3
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:4] "WingL" "BeakH" "UBeakL" "N.UBkL"
##   .. .. ..$ : NULL
##   ..$ T_IC.IR_nm: num [1:9, 1:4, 1:6] 0.948 1.183 0.875 1.067 1.067 ...
##   .. ..- attr(*, "dimnames")=List of 3
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:4] "WingL" "BeakH" "UBeakL" "N.UBkL"
##   .. .. ..$ : NULL
##   ..$ T_PC.PR_nm: num [1:9, 1:4, 1:6] 0.707 0.357 5.311 7.249 4.839 ...
##   .. ..- attr(*, "dimnames")=List of 3
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:4] "WingL" "BeakH" "UBeakL" "N.UBkL"
##   .. .. ..$ : NULL
##  $ variances:List of 12
##   ..$ var_IP      : num [1:39, 1:4] NA 4.6 4.5 2.94 3.54 ...
##   ..$ var_PC      : num [1:6, 1:4] 17.4 112.8 66.9 135.3 113.3 ...
##   ..$ var_CR      : num [1:4] 1.941 0.473 1.376 0.235
##   ..$ var_IC      : num [1:6, 1:4] 6.49 120.02 40.37 125.72 96.26 ...
##   ..$ var_PR      : num [1:4] 76.8 17.89 13.97 7.36
##   ..$ var_IR      : num [1:4] 70.17 14.01 12.49 7.52
##   ..$ var_IP_nm1  : num [1:9, 1:4, 1:39] NA NA NA NA NA NA NA NA NA NA ...
##   ..$ var_PC_nm2sp: num [1:9, 1:4, 1:6] 58.7 32.8 557.5 896.8 334.2 ...
##   ..$ var_IC_nm1  : num [1:9, 1:4, 1:6] 6.49 125.4 43.19 25.3 22.78 ...
##   ..$ var_IC_nm2  : num [1:9, 1:4, 1:6] 67.1 81.6 61.8 77.4 76.5 ...
##   ..$ var_PR_nm2sp: num [1:9, 1:4] 83.1 91.9 105 123.7 69.1 ...
##   ..$ var_IR_nm2  : num [1:9, 1:4] 70.8 69 70.6 72.5 71.8 ...
##  - attr(*, "class")= chr "Tstats"
```

5.1.1 S3 methods for class Tstats

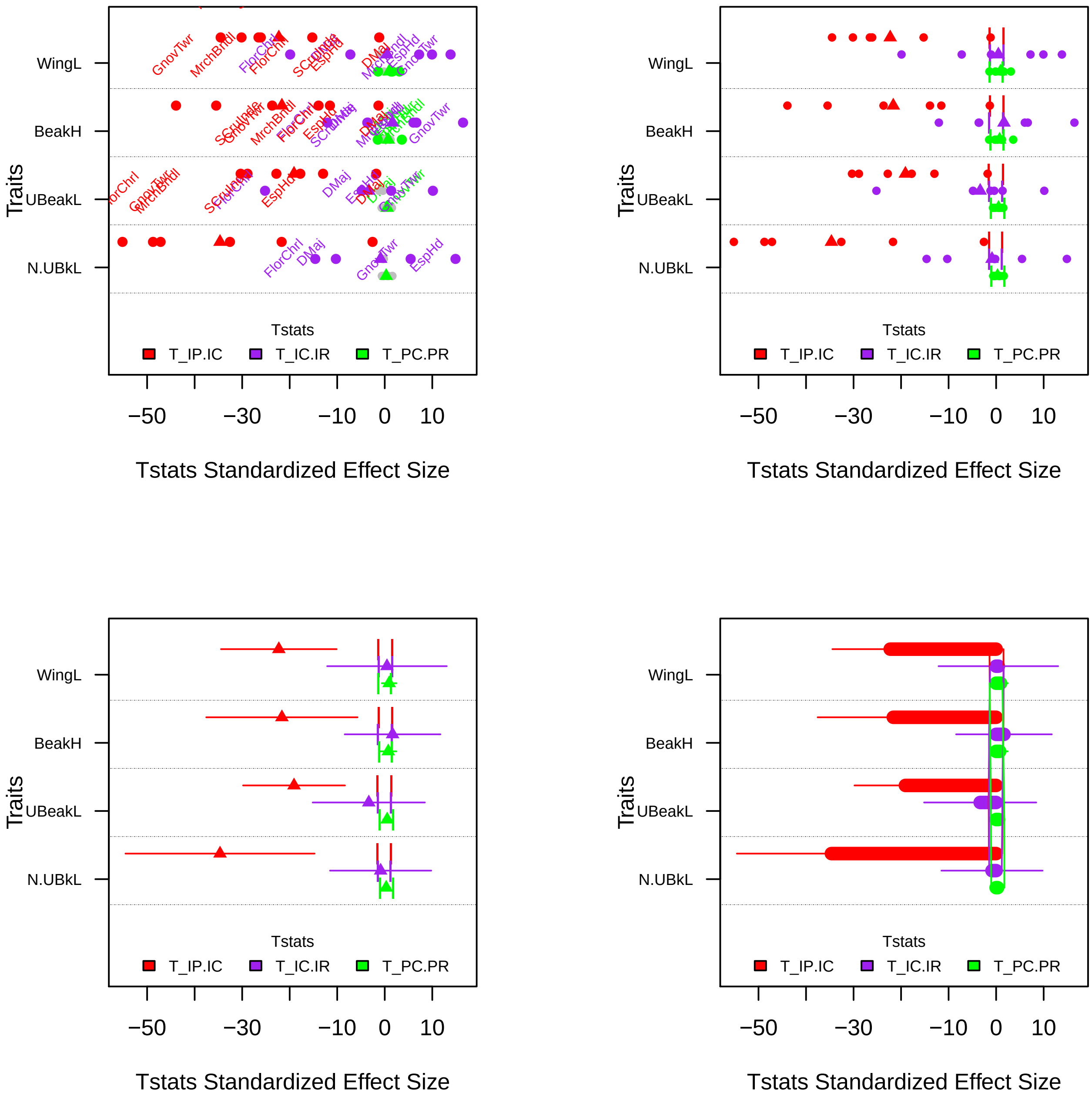
Tstats class is associated to S3 methods plot, barplot, print and summary

```
plot(res.finch)
abline(v=0)
```



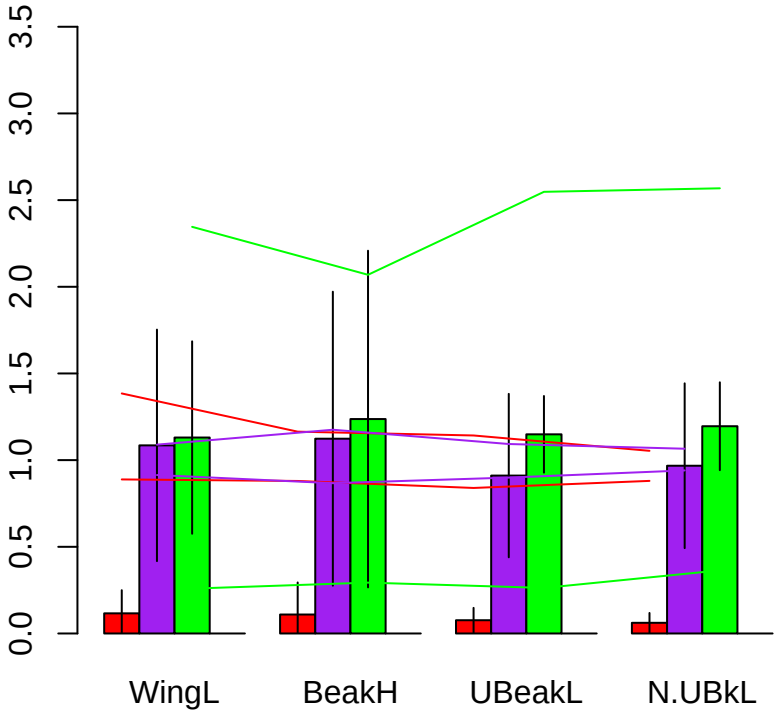
There is multiple kind of representation avaible

```
par(mfrow=c(2,2))
plot(res.finch, type="color_cond")
plot(res.finch, type="simple")
plot(res.finch, type="simple_sd")
plot(res.finch, type="barplot")
```


WingL
BeakH
UBeakL
N.UBkL
Traits
Tstats
T_IP.IC
T_IC.IR
T_PC.PR
−50
−30
−10
0
10
Tstats Standardized Effect Size

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

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




```
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
##   0.003   0.438   0.838   1.160   1.390   7.460       2
```

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

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

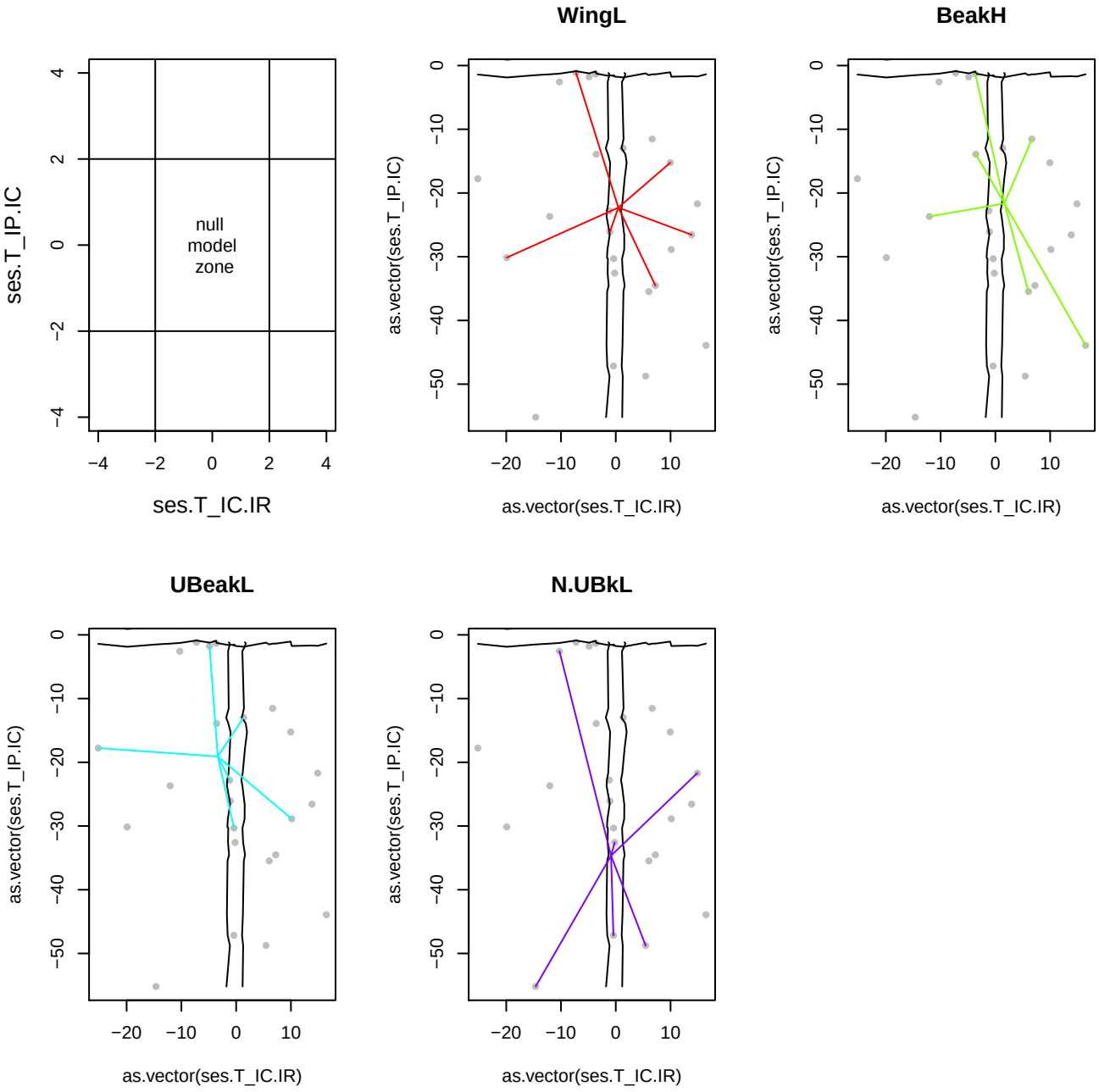
```
head(sum_Tstats(res.finch)$p.value, 10)
```

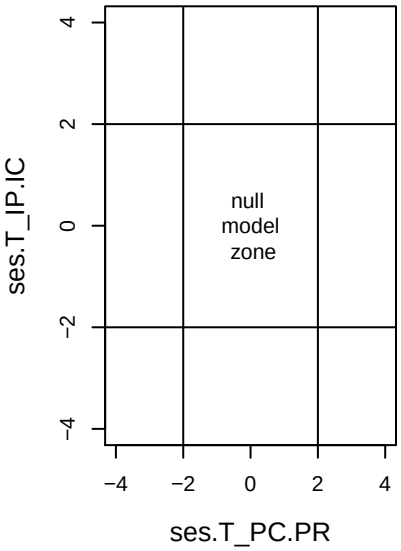
```
##             WingL BeakH UBeakL N.UBkL
## T_IP.IC.inf   0.1   0.1    0.1    0.1
## T_IP.IC.inf   0.1   0.1    0.1    0.1
## T_IP.IC.inf   0.1   0.1    0.1    0.1
## T_IP.IC.inf   0.1   0.1    0.1    0.1
## T_IP.IC.inf   0.1   0.1    0.1    0.1
## T_IP.IC.inf   0.1   0.1    0.1    0.1
## T_IP.IC.sup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

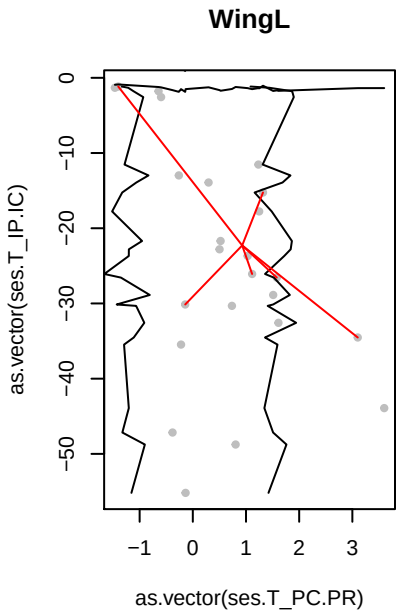
5.1.2 Plot T-statistics correlations

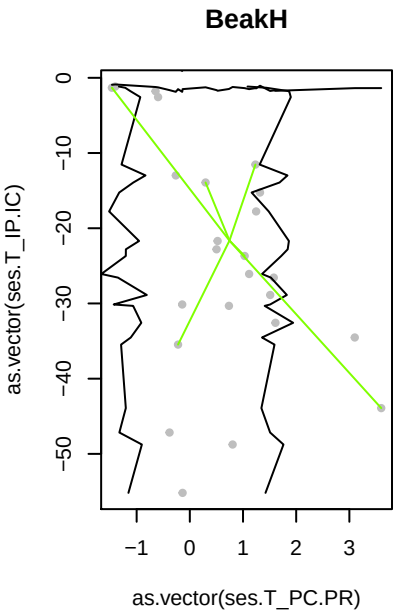
We can also see T-statistics correlations and theirs correlation with others variables (e.g. a gradient variable, or the species richness).

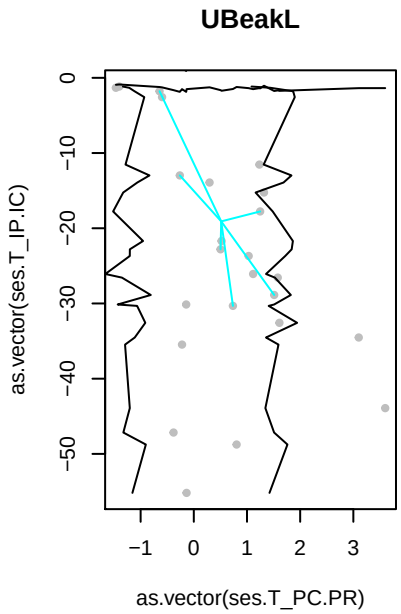
```
par(mfrow=c(2,3))
plotCorTstats(res.finch, plot.ask=FALSE, multipanel=F)
```

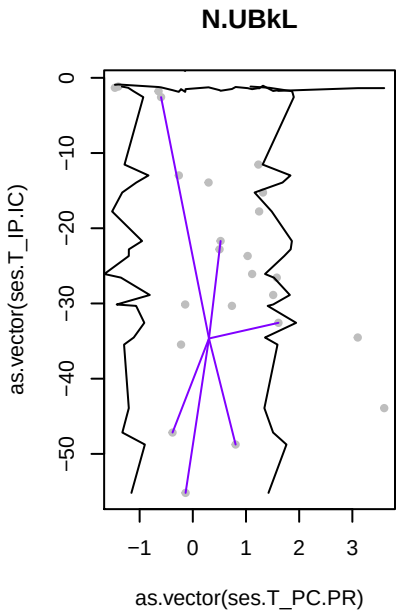


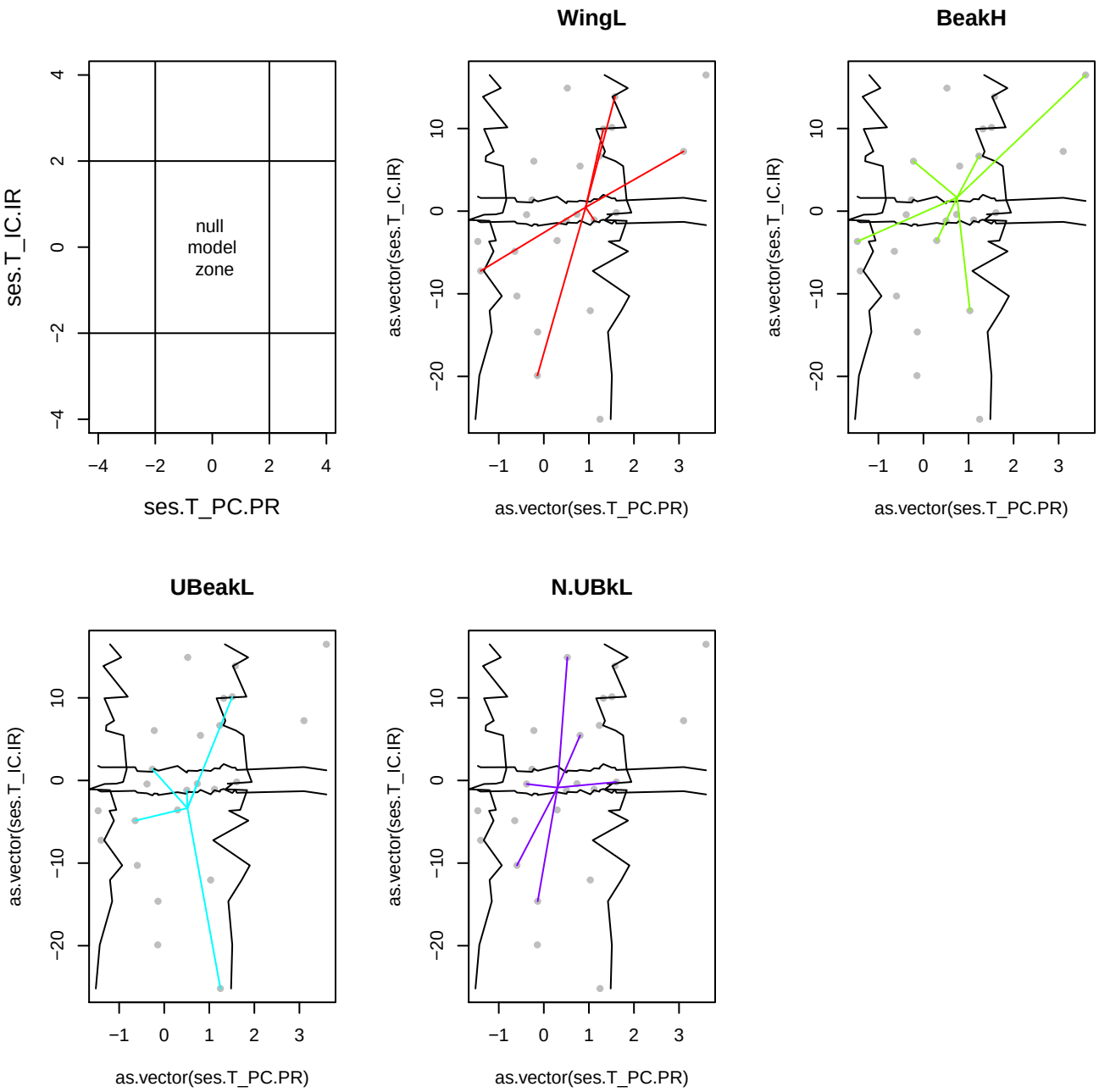

ses.T_IP.IC
ses.T_PC.PR
null
model
zone


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)

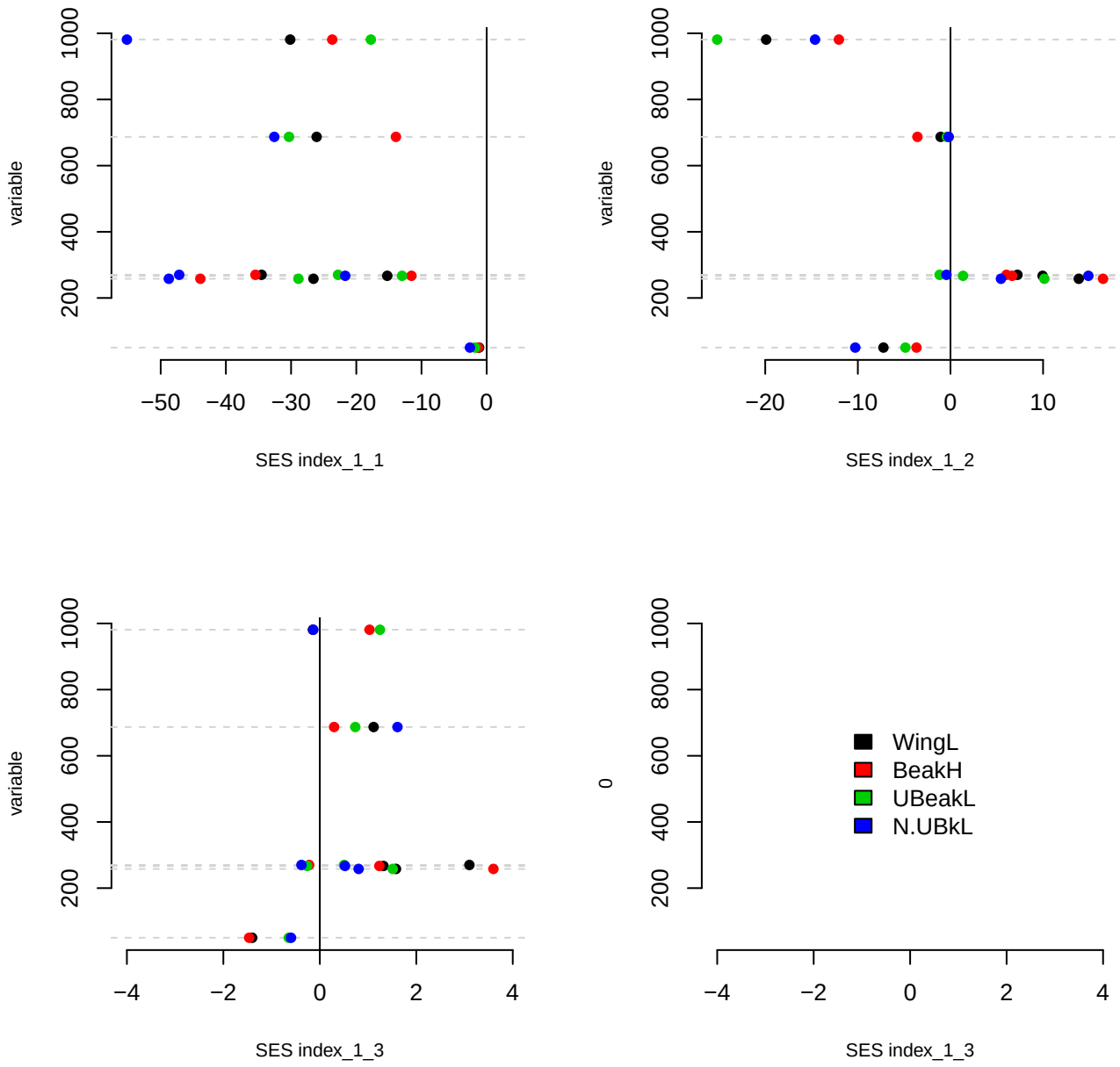

UBeakL
as.vector(ses.T_IP.IC)
as.vector(ses.T_PC.PR)


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



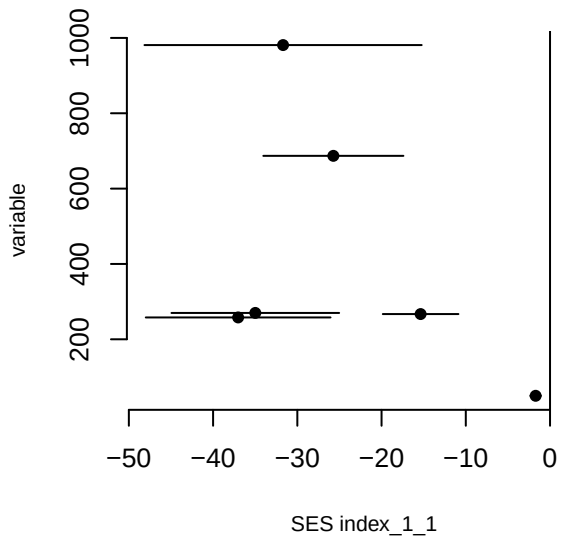
Here we plot T-statistics (in the standardized effect size SES form) in function of species richness by sites.

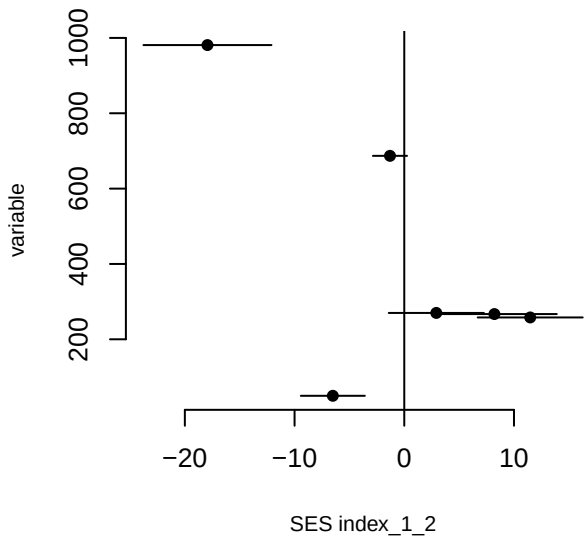
```
par(mfrow=c(2,2))
species.richness<-table(ind.plot.finch)
plotSESvar(as.listofindex(list(res.finch)), species.richness,
            multipanel=F)
```

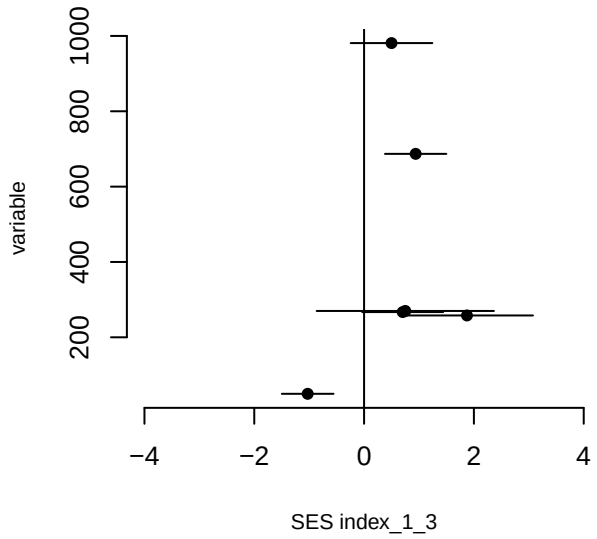


Same plot with `resume=TRUE`.

```
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 metrics: function ComIndex and ComIndexMulti

The function ComIndex allow to choose 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 funcions 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 2
res.finch.sp_mn2<-ComIndex(traits=traits.finch, index=funct, sp=sp.finch,
                           nullmodels=c(2,2,2,2), ind.plot=ind.plot.finch,
                           nperm=9, print=FALSE)

#Test against the null model 2sp
res.finch.sp_mn2sp<-ComIndex(traits=traits.finch, index=funct, sp=sp.finch,
                             nullmodels=rep("2sp",4), ind.plot=ind.plot.finch,
                             nperm=9, print=FALSE)
```

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

```
funct<-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))"  )

##Null model 1 is trivial for this function
##because randomisation is within community only

res.finch.ind_mn1<-ComIndex(traits=traits.finch, index=funct, sp=sp.finch,
                            nullmodels=c(1,1,1,1), ind.plot=ind.plot.finch,
                            nperm=9, print=FALSE)
res.finch.ind_mn2<-ComIndex(traits=traits.finch, index=funct, sp=sp.finch,
                            nullmodels=c(2,2,2,2), ind.plot=ind.plot.finch,
                            nperm=9, print=FALSE)
```

We can calcul index with or without intraspecific variance.

```
#Calcul of means by population (name_sp_site is a name of a population)
#like in the function ComIndex and 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])

#We use the precedent list of function "funct"
funct.withIV<-funct

fact<-unlist(sites_bypop)
funct.withoutIV<-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))")


res.finch.withIV<-ComIndex(traits=traits.finch, index=funct.withIV,
                           sp=sp.finch, nullmodels=c(2,2,2,2),
                           ind.plot=ind.plot.finch, nperm=9, print=FALSE)

res.finch.withoutIV<-ComIndex(traits=traits.finch, index=funct.withoutIV,
                           sp=sp.finch, nullmodels=rep("2sp",4),
                           ind.plot=ind.plot.finch, nperm=9, print=FALSE)
```

5.2.1 S3 methods for class ComIndex and ComIndexMulti

Tstats class is associated to S3 methods plot, print and summary

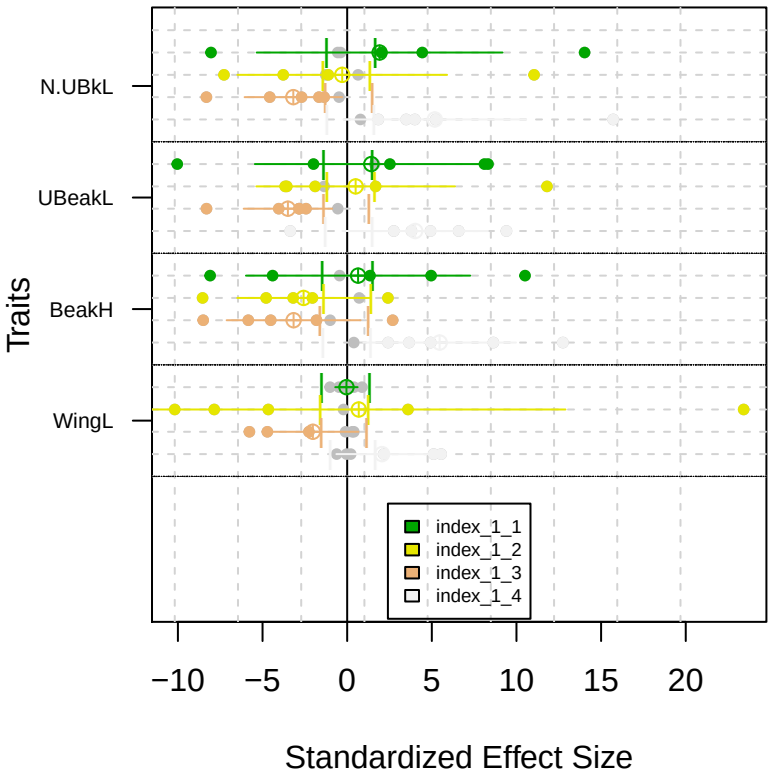
```
res.finch.withIV

## List of 1
##  $ obs:List of 4
##   ..$ tapply(x, ind.plot.finch, function(x) mean(x, na.rm=T))               : num [1:6, 1:
##   .. ..- attr(*, "dimnames")=List of 2
##   ..$ tapply(x, ind.plot.finch, function(x) kurtosis(x, na.rm=T))           : num [1:6, 1:
##   .. ..- attr(*, "dimnames")=List of 2
##   ..$ tapply(x, ind.plot.finch, function(x) max(x, na.rm=T)-min(x, na.rm=T)): num [1:6, 1:
##   .. ..- attr(*, "dimnames")=List of 2
##   ..$ tapply(x, ind.plot.finch, function(x) CVNND(x))                       : num [1:6, 1:
##   .. ..- attr(*, "dimnames")=List of 2
## NULL
## List of 5
##  $ Null          :List of 4
##  $ list.index    :List of 8
##  $ list.index.t  :List of 8
##  $ sites_richness: Named num [1:6] NULL ...
##   ..- attr(*, "names")= chr [1:6]  ...
##  $ namestraits   : chr [1:4]  ...
## NULL

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.   :67.0   Min.   : 9.71   Min.   :14.8   Min.   :10.1
##  1st Qu.:67.6   1st Qu.: 9.99   1st Qu.:15.4   1st Qu.:10.7
##  Median :67.7   Median :10.37   Median :16.4   Median :11.0
##  Mean   :67.7   Mean   :10.62   Mean   :16.7   Mean   :11.1
##  3rd Qu.:67.8   3rd Qu.:11.36   3rd Qu.:18.0   3rd Qu.:11.6
##  Max.   :68.3   Max.   :11.71   Max.   :18.7   Max.   :12.1
##
## $`tapply(x, ind.plot.finch, function(x) kurtosis(x, na.rm=T))`
##      WingL            BeakH            UBeakL           N.UBkL
##  Min.   :-1.466   Min.   :-1.360   Min.   :-1.600   Min.   :-1.752
##  1st Qu.:-1.252   1st Qu.:-0.642   1st Qu.:-1.185   1st Qu.:-1.449
##  Median :-0.821   Median :-0.297   Median :-1.089   Median :-1.083
##  Mean   :-0.622   Mean   :-0.110   Mean   :-0.564   Mean   :-0.740
##  3rd Qu.:-0.274   3rd Qu.: 0.646   3rd Qu.:-0.876   3rd Qu.:-0.946
##  Max.   : 0.865   Max.   : 1.087   Max.   : 2.414   Max.   : 1.950
##
## $`tapply(x, ind.plot.finch, function(x) max(x, na.rm=T)-min(x, na.rm=T))`
##      WingL          BeakH          UBeakL         N.UBkL
##  Min.   :11.0   Min.   : 4.3   Min.   : 8.7   Min.   : 6.50
##  1st Qu.:34.2   1st Qu.:14.6   1st Qu.:11.1   1st Qu.: 8.88
##  Median :35.5   Median :16.1   Median :12.6   Median :10.05
##  Mean   :31.8   Mean   :14.7   Mean   :11.9   Mean   : 9.33
##  3rd Qu.:36.8   3rd Qu.:17.5   3rd Qu.:13.2   3rd Qu.:10.32
##  Max.   :38.0   Max.   :19.4   Max.   :13.6   Max.   :10.50
##
## $`tapply(x, ind.plot.finch, function(x) CVNND(x))`
##      WingL          BeakH          UBeakL         N.UBkL
##  Min.   :2.09   Min.   :1.66   Min.   :1.77   Min.   :1.89
##  1st Qu.:2.62   1st Qu.:3.00   1st Qu.:2.17   1st Qu.:2.17
##  Median :2.67   Median :3.27   Median :2.74   Median :2.61
##  Mean   :3.21   Mean   :3.37   Mean   :2.61   Mean   :2.64
##  3rd Qu.:3.73   3rd Qu.:3.47   3rd Qu.:2.91   3rd Qu.:3.10
##  Max.   :5.17   Max.   :5.58   Max.   :3.46   Max.   :3.41
##
## [1] "Null values"
## $`tapply(x, ind.plot.finch, function(x) mean(x, na.rm=T))`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9.66   10.60   13.30   26.20   29.20   70.30
##
## $`tapply(x, ind.plot.finch, function(x) kurtosis(x, na.rm=T))`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -1.360  -0.951  -0.653  -0.445  -0.176   1.590
##
## $`tapply(x, ind.plot.finch, function(x) max(x, na.rm=T)-min(x, na.rm=T))`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

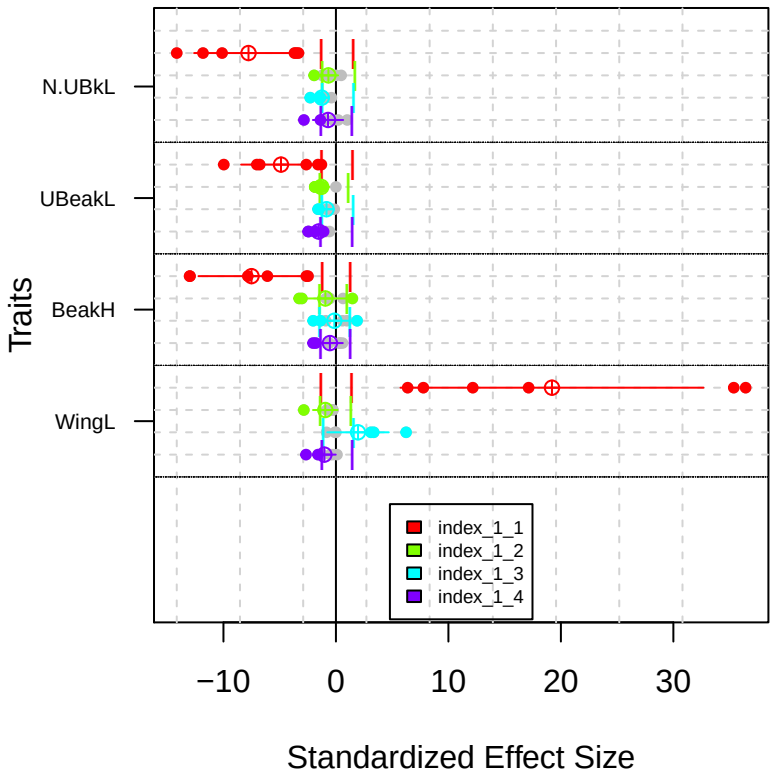
```
##     8.7    11.4    15.9    19.8    20.8    39.0
##
## $`tapply(x, ind.plot.finch, function(x) CVNND(x))`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.796   1.610   2.180   2.300   2.680   4.990

summary(res.finch.withoutIV)

## [1] "Observed values"
## $`tapply(x, fact, function(x) mean(x, na.rm=T))`
##      WingL          BeakH           UBeakL         N.UBkL
##  Min.   :65.9   Min.   : 9.55   Min.   :15.5   Min.   :10.6
##  1st Qu.:67.2   1st Qu.:10.06   1st Qu.:15.7   1st Qu.:10.7
##  Median :68.7   Median :10.15   Median :16.4   Median :11.0
##  Mean   :68.1   Mean   :10.37   Mean   :16.6   Mean   :11.1
##  3rd Qu.:69.1   3rd Qu.:10.68   3rd Qu.:17.1   3rd Qu.:11.2
##  Max.   :69.2   Max.   :11.50   Max.   :18.5   Max.   :11.9
##
## $`tapply(x, fact, function(x) kurtosis(x, na.rm=T))`
##      WingL            BeakH             UBeakL          N.UBkL
##  Min.   :-2.333   Min.   :-2.3333   Min.   :-2.33   Min.   :-2.33
##  1st Qu.:-1.954   1st Qu.:-2.0761   1st Qu.:-2.27   1st Qu.:-2.04
##  Median :-1.760   Median :-1.4834   Median :-1.98   Median :-1.87
##  Mean   :-1.559   Mean   :-1.2609   Mean   :-1.99   Mean   :-1.84
##  3rd Qu.:-1.033   3rd Qu.:-0.4232   3rd Qu.:-1.77   3rd Qu.:-1.60
##  Max.   :-0.693   Max.   : 0.0779   Max.   :-1.60   Max.   :-1.37
##
## $`tapply(x, fact, function(x) max(x, na.rm=T)-min(x, na.rm=T))`
##      WingL           BeakH           UBeakL          N.UBkL
##  Min.   : 8.33   Min.   : 2.35   Min.   : 7.14   Min.   :5.66
##  1st Qu.:26.21   1st Qu.:12.08   1st Qu.: 8.71   1st Qu.:7.07
##  Median :29.21   Median :14.29   Median :10.06   Median :7.38
##  Mean   :25.41   Mean   :12.36   Mean   : 9.42   Mean   :7.22
##  3rd Qu.:29.87   3rd Qu.:14.81   3rd Qu.:10.26   3rd Qu.:7.64
##  Max.   :30.41   Max.   :16.93   Max.   :10.67   Max.   :8.20
##
## $`tapply(x, fact, function(x) CVNND(x))`
##      WingL            BeakH           UBeakL           N.UBkL
##  Min.   :0.0162   Min.   :0.106   Min.   :0.0954   Min.   :0.382
##  1st Qu.:0.1530   1st Qu.:0.530   1st Qu.:0.1835   1st Qu.:0.398
##  Median :0.4492   Median :0.910   Median :0.4489   Median :0.556
##  Mean   :0.5157   Mean   :0.784   Mean   :0.3828   Mean   :0.663
##  3rd Qu.:0.9333   3rd Qu.:1.108   3rd Qu.:0.5212   3rd Qu.:0.909
##  Max.   :1.0306   Max.   :1.198   Max.   :0.6635   Max.   :1.110
##
## [1] "Null values"
## $`tapply(x, fact, function(x) mean(x, na.rm=T))`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    15.6    24.7    26.7    27.0    28.6    40.3
```

```
##
## $`tapply(x, fact, function(x) kurtosis(x, na.rm=T))`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -2.380  -2.000  -1.420  -1.200  -0.856   3.270
##
## $`tapply(x, fact, function(x) max(x, na.rm=T)-min(x, na.rm=T))`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.98   10.00   14.10   18.90   21.40   75.30
##
## $`tapply(x, fact, function(x) CVNND(x))`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0199  0.6750  0.9220  0.9960  1.2400  2.4500
```

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




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



5.2.2 Plot Tstats and other uni/multivariates metrics together

The class listofindex permit to stock differents 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.

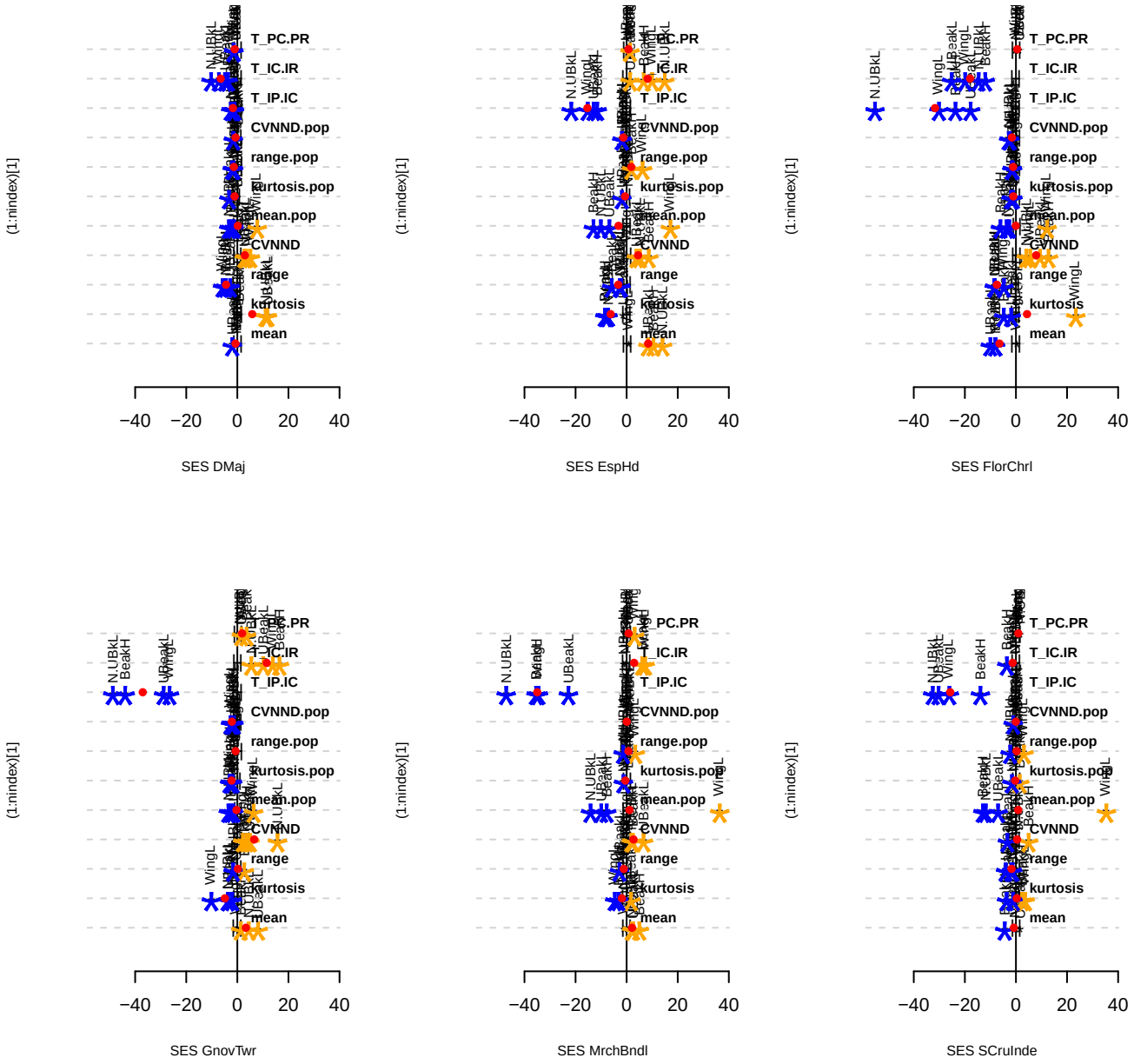
```
list.ind1<-list(res.finch.withIV, res.finch.withoutIV)
index.list1<-as.listofindex(list.ind1)

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)
```

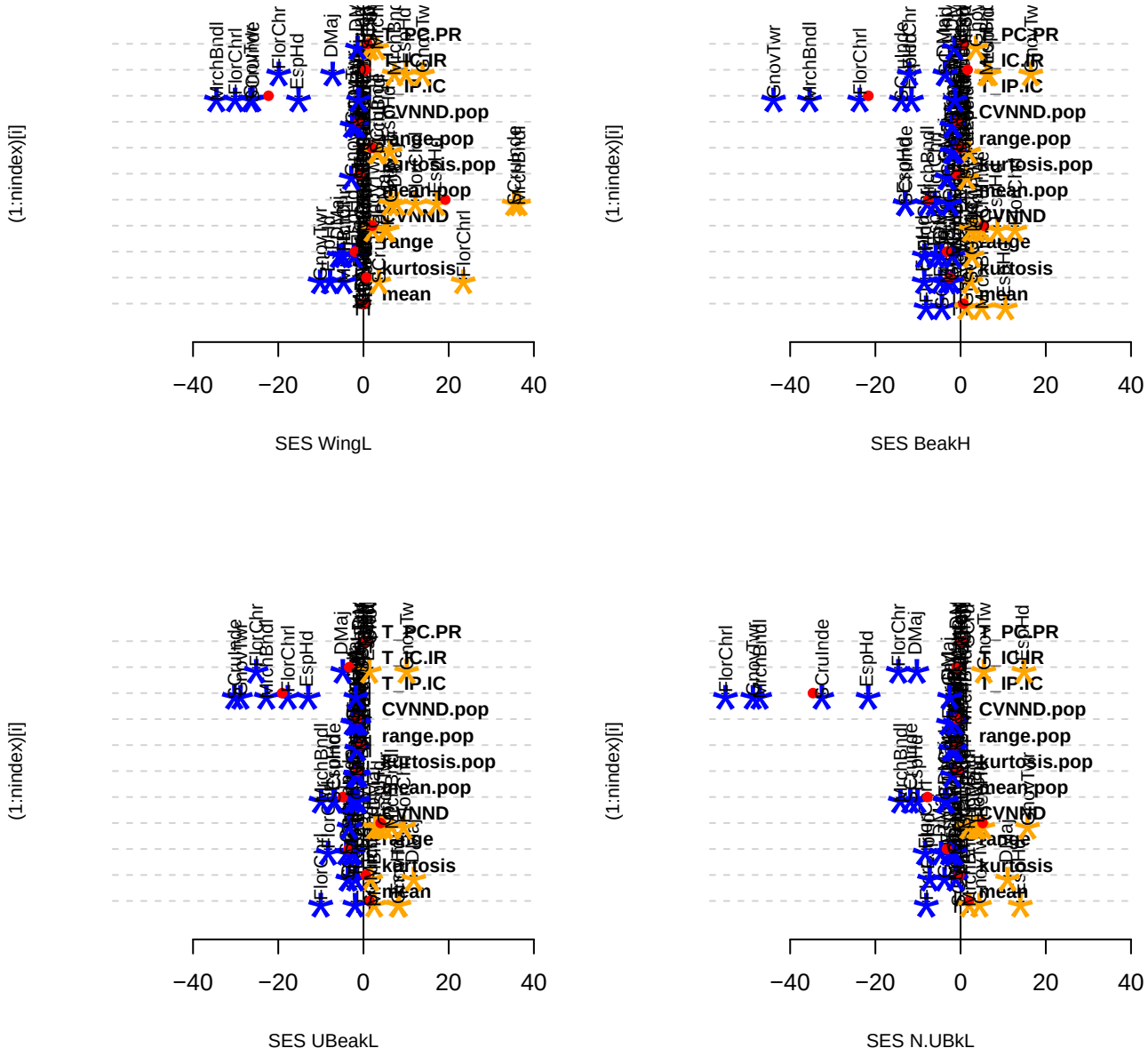
```
class(i.l1)

## [1] "listofindex"
```

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

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



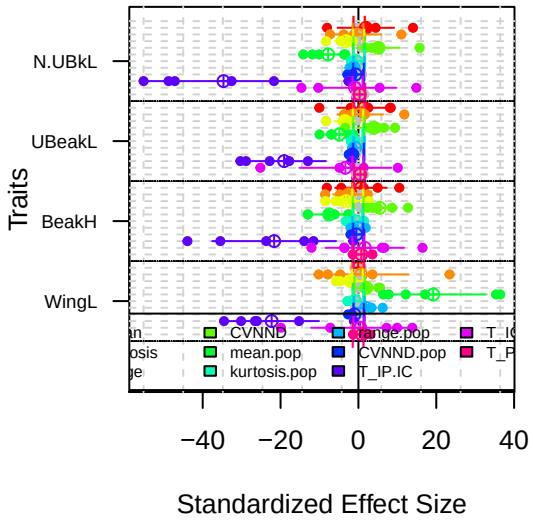

```
par(mfrow=c(2,2))
plot(i.l1,type="bytraits")
```



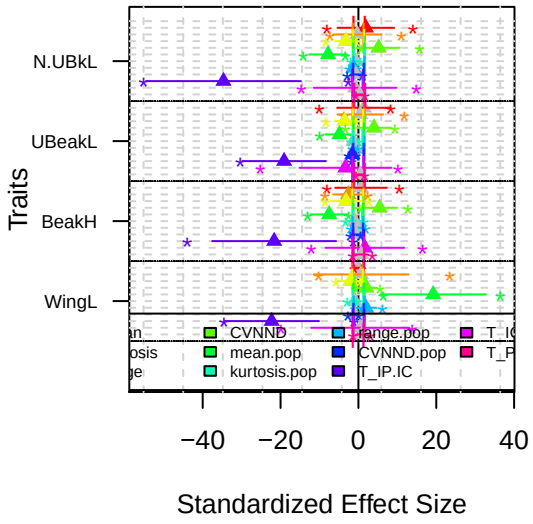

```
par(mfrow=c(1,1))
```

The other plot type are the same as plot.Tstats

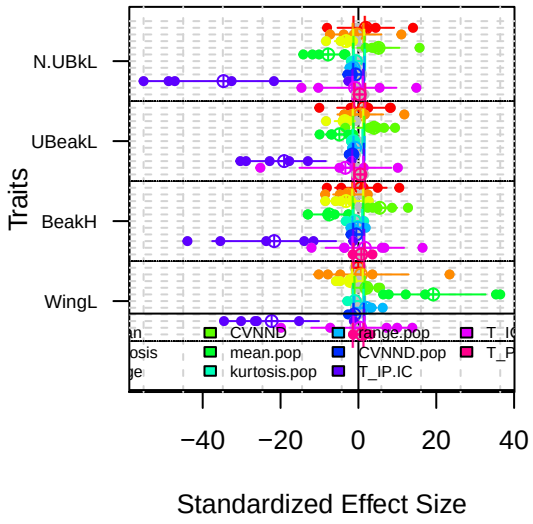
```
par(mfrow=c(2,2))
plot(i.l1)
```



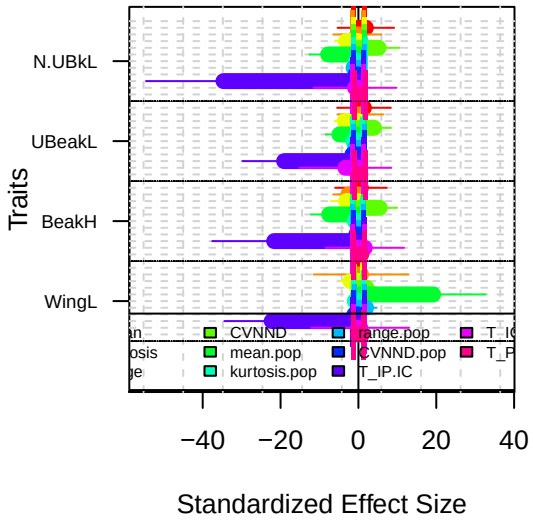

```
plot(i.l1,type="simple_range")
```




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




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




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

5.3 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)
```

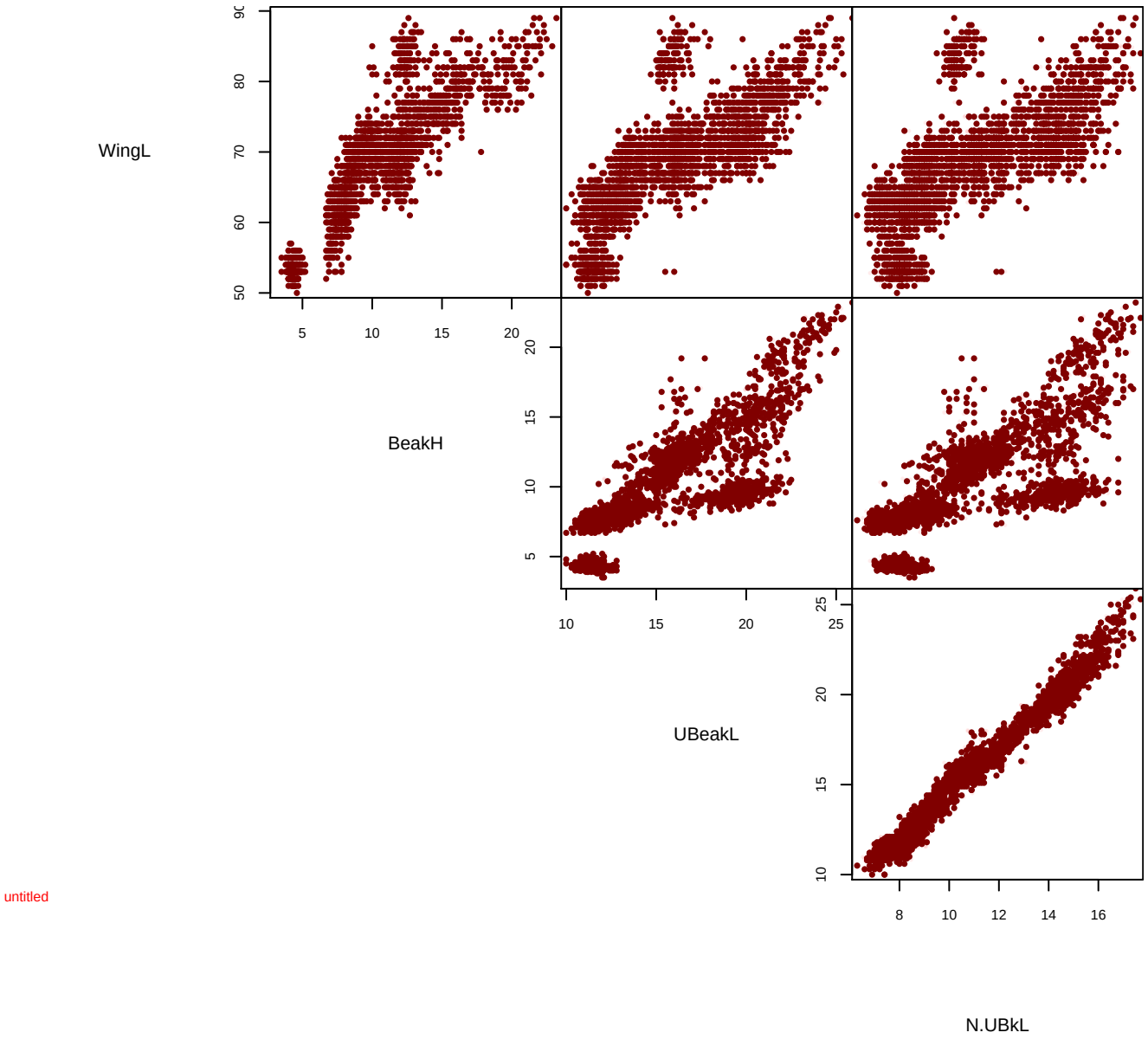
A simple example to illustrate the concept of the function ComIndexMulti

```
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=c(2,2),
                         ind.plot=ind.plot.finch, nperm=9, sp=sp.finch)

## [1] "creating null models"
## [1] "nm.2 25 %"
## [1] "nm.2 50 %"
## [1] "nm.2 75 %"
## [1] "nm.2 100 %"
## [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 %"

attributes(ses.listofindex(as.listofindex(list(res.sum.1))))

## $names
## [1] "index_1_1" "index_1_2"
##
## $class
## [1] "ses.list"
```

A more interesting example using the function hypervolume from the package ... hypervolume. We show here several results which differe in there factor that delimit the group to calculate different hypervolume (argument "byfactor").

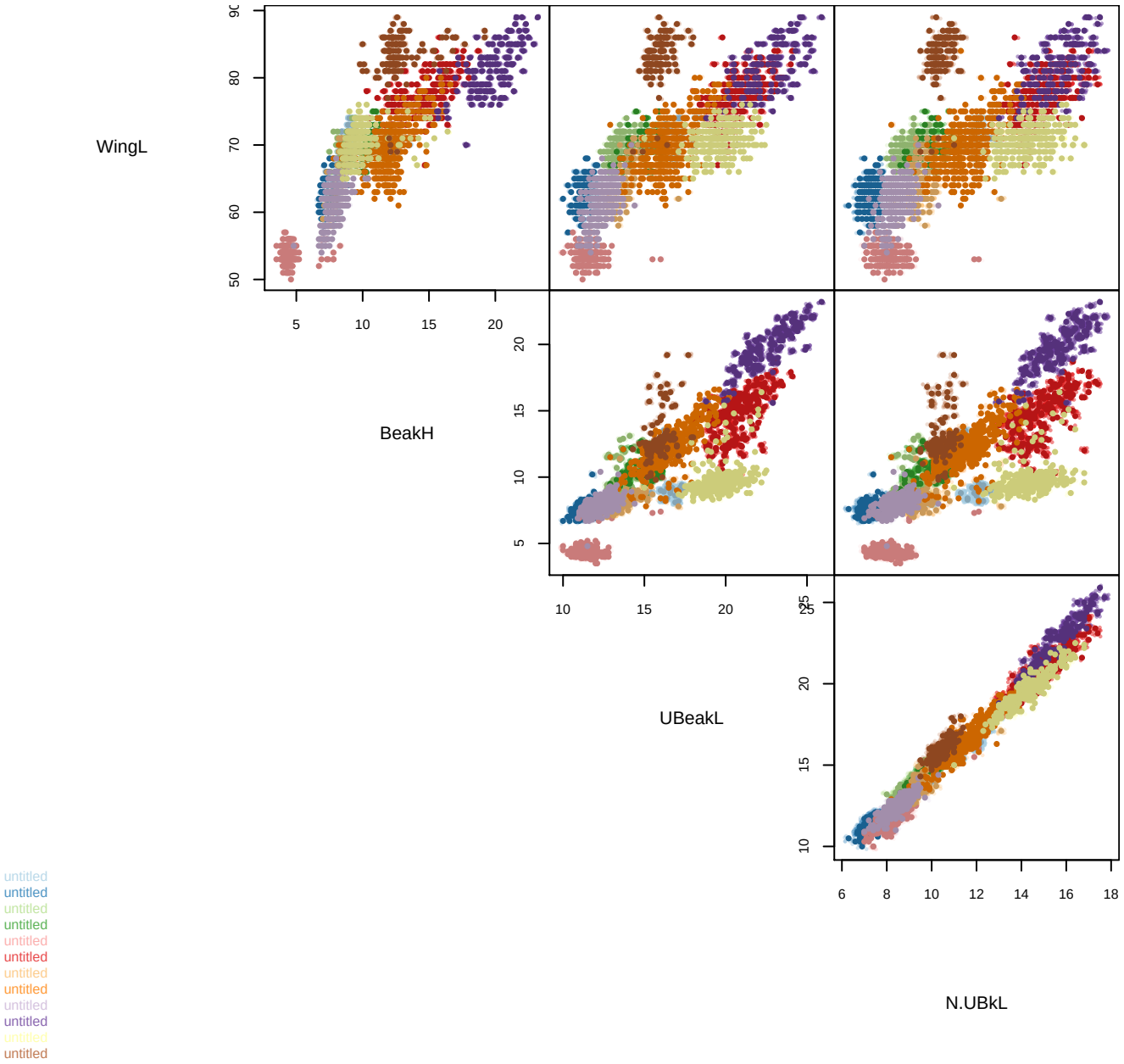
First, let's try the hypervolume function one finch data.

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

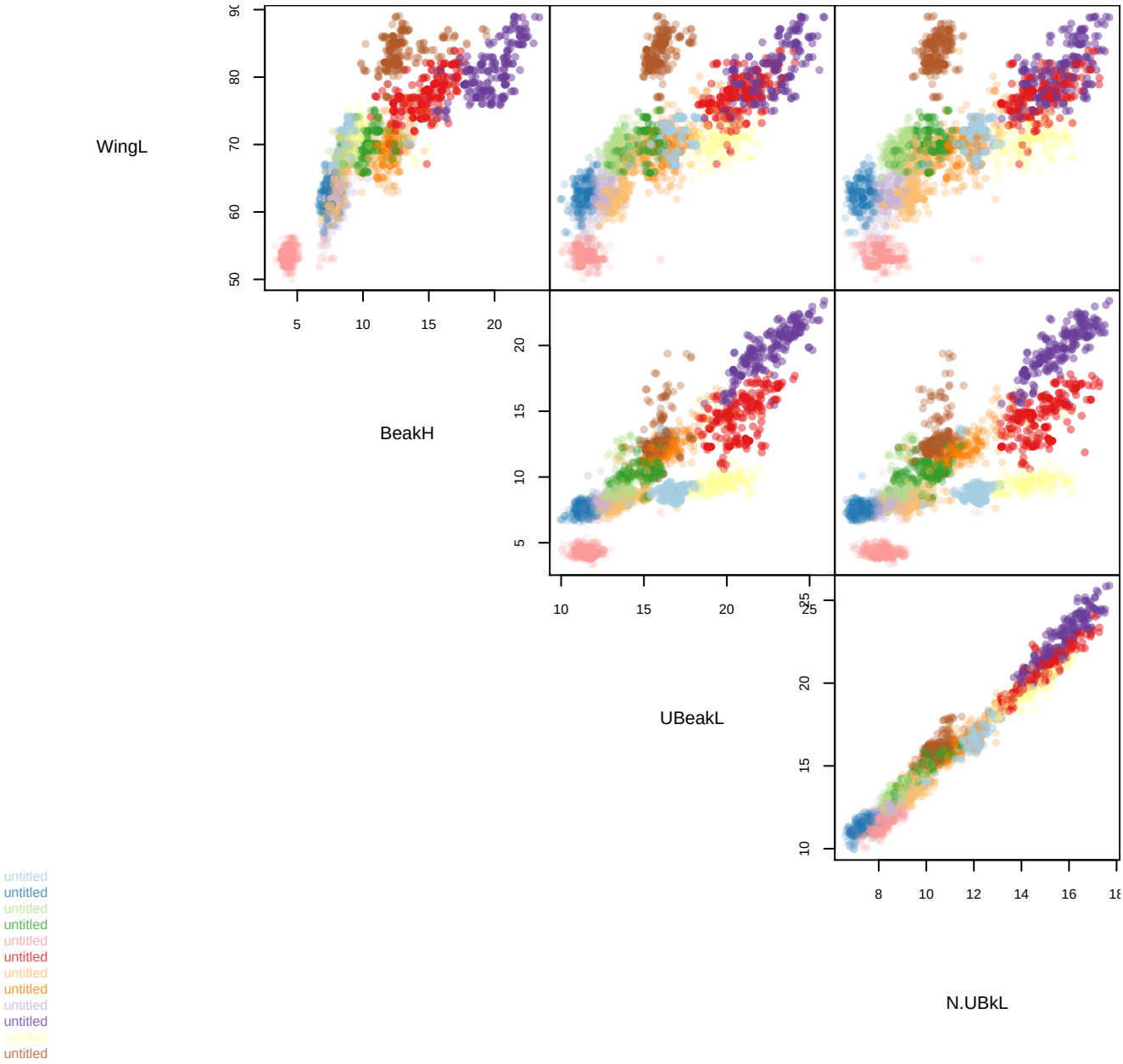


Now, we can do the same analysis for each species.

```r
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)

## HypervolumeList with 12 elements:
##
## Hypervolume
##  Name: untitled
##  Nr. of observations: 23
##  Dimensionality: 4
##  Volume: 0.581274
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Disjunct factor: 0.987217
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
```

```
##  Nr. of repetitions per point: 1000
##  Number of random points: 14385
## Hypervolume
##  Name: untitled
##  Nr. of observations: 172
##  Dimensionality: 4
##  Volume: 3.508501
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Disjunct factor: 0.796807
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 91104
## Hypervolume
##  Name: untitled
##  Nr. of observations: 142
##  Dimensionality: 4
##  Volume: 2.957552
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Disjunct factor: 0.813587
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 76385
## Hypervolume
##  Name: untitled
##  Nr. of observations: 73
##  Dimensionality: 4
##  Volume: 1.729685
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Disjunct factor: 0.925559
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 43564
## Hypervolume
##  Name: untitled
##  Nr. of observations: 299
##  Dimensionality: 4
##  Volume: 4.127026
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Disjunct factor: 0.539170
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 114079
## Hypervolume
##  Name: untitled
```

```
##  Nr. of observations: 206
##  Dimensionality: 4
##  Volume: 5.070588
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Disjunct factor: 0.961504
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 126546
## Hypervolume
##  Name: untitled
##  Nr. of observations: 125
##  Dimensionality: 4
##  Volume: 2.734825
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Disjunct factor: 0.854633
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 69799
## Hypervolume
##  Name: untitled
##  Nr. of observations: 548
##  Dimensionality: 4
##  Volume: 11.785994
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Disjunct factor: 0.840128
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 302349
## Hypervolume
##  Name: untitled
##  Nr. of observations: 386
##  Dimensionality: 4
##  Volume: 6.293490
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Disjunct factor: 0.636890
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 170580
## Hypervolume
##  Name: untitled
##  Nr. of observations: 126
##  Dimensionality: 4
##  Volume: 3.180147
##  Bandwidth: 0.2 0.2 0.2 0.2
```

```
##  Disjunct factor: 0.985909
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 78794
## Hypervolume
##  Name: untitled
##  Nr. of observations: 284
##  Dimensionality: 4
##  Volume: 6.331289
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Disjunct factor: 0.870831
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 161250
## Hypervolume
##  Name: untitled
##  Nr. of observations: 129
##  Dimensionality: 4
##  Volume: 2.871520
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Disjunct factor: 0.869525
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 72987
```

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

```
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.')
## + }

## Demo does not run by default to meet CRAN runtime requirements.
## This demo requires approximately 3 minutes to run.
## To run the demo, type
## doHypervolumeFinchDemo=TRUE
## demo(finch)
## at the R command line prompt.
```

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: calcul 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=c(2,2),
                          ind.plot=ind.plot.finch, nperm=9, sp=sp.finch)

## [1] "creating null models"
## [1] "nm.2 25 %"
## [1] "nm.2 50 %"
## [1] "nm.2 75 %"
## [1] "nm.2 100 %"
## [1] "calculation of null values using null models"
## [1] "as.numeric(try(hypervolume(na.omit(x), reps=100, \n
## [1] "calculation of observed values"
## [1] "100 %"

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=c(2,2),
                          ind.plot=ind.plot.finch, nperm=9, sp=sp.finch)

## [1] "creating null models"
## [1] "nm.2 25 %"
## [1] "nm.2 50 %"
## [1] "nm.2 75 %"
## [1] "nm.2 100 %"
## [1] "calculation of null values using null models"
## [1] "as.numeric(try(hypervolume(na.omit(x), reps=100, \n
## [1] "calculation of observed values"
## [1] "100 %"

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=c(2,2),
                          ind.plot=ind.plot.finch, nperm=9, sp=sp.finch)
```

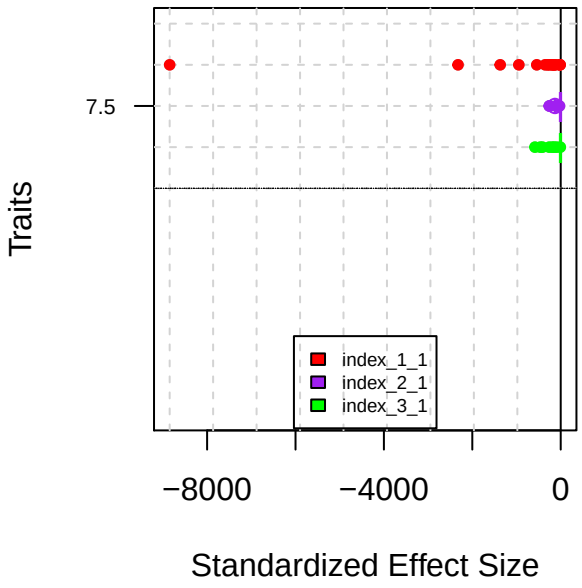
```
## [1] "creating null models"
## [1] "nm.2 25 %"
## [1] "nm.2 50 %"
## [1] "nm.2 75 %"
## [1] "nm.2 100 %"
## [1] "calculation of null values using null models"
## [1] "as.numeric(try(hypervolume(na.omit(x), reps=100,\n                                                                    b
## [1] "calculation of observed values"
## [1] "100 %"
```

```
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=c(2,2),
                          ind.plot=ind.plot.finch, nperm=9, sp=sp.finch)
```

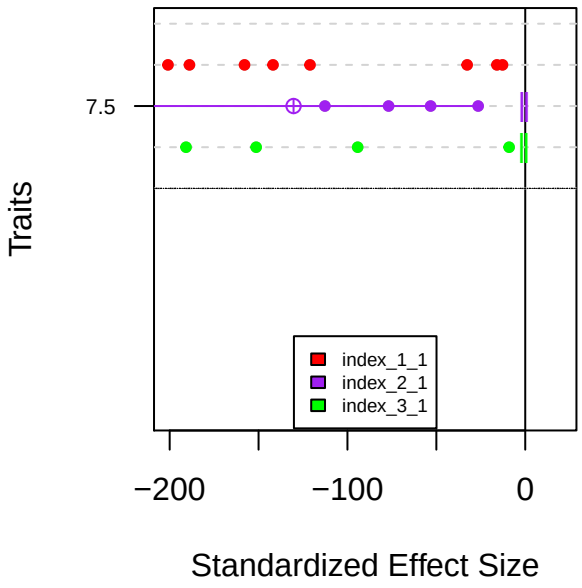
```
## [1] "creating null models"
## [1] "nm.2 25 %"
## [1] "nm.2 50 %"
## [1] "nm.2 75 %"
## [1] "nm.2 100 %"
## [1] "calculation of null values using null models"
## [1] "as.numeric(try(hypervolume(na.omit(x), reps=100, \n
## [1] "calculation of observed values"
## [1] "100 %"
```

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

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

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




```
plot(list.ind.multi, xlim=c(-200,20))
```



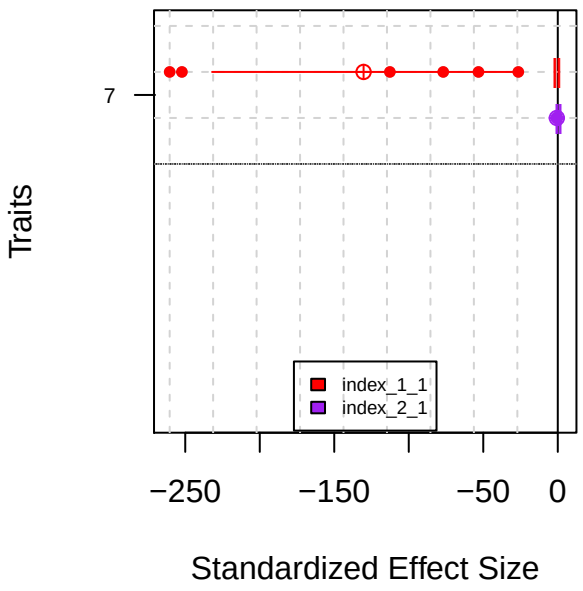
Compare hypervolume to Villéger metrics convex hull.

```
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 1 is trivial for this function
##because randomisation is within community only
Fdis.finch<-ComIndexMulti(traits.finch.mice,
                          index=funct,
                          by.factor=ind.plot.finch, nullmodels=c(2,2),
                          ind.plot=ind.plot.finch, nperm=9, sp=sp.finch)

## [1] "creating null models"
## [1] "nm.2 25 %"
## [1] "nm.2 50 %"
## [1] "nm.2 75 %"
## [1] "nm.2 100 %"
## [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)
```



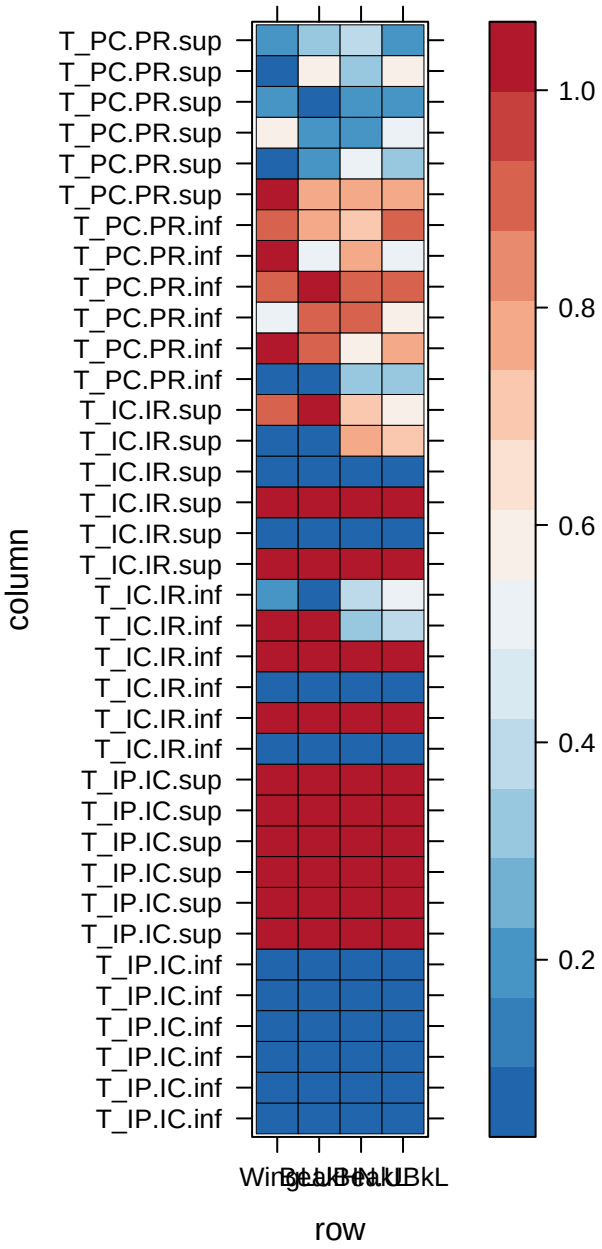
6 Others graphics functions

Using rasterVis to obtain more color schemes.

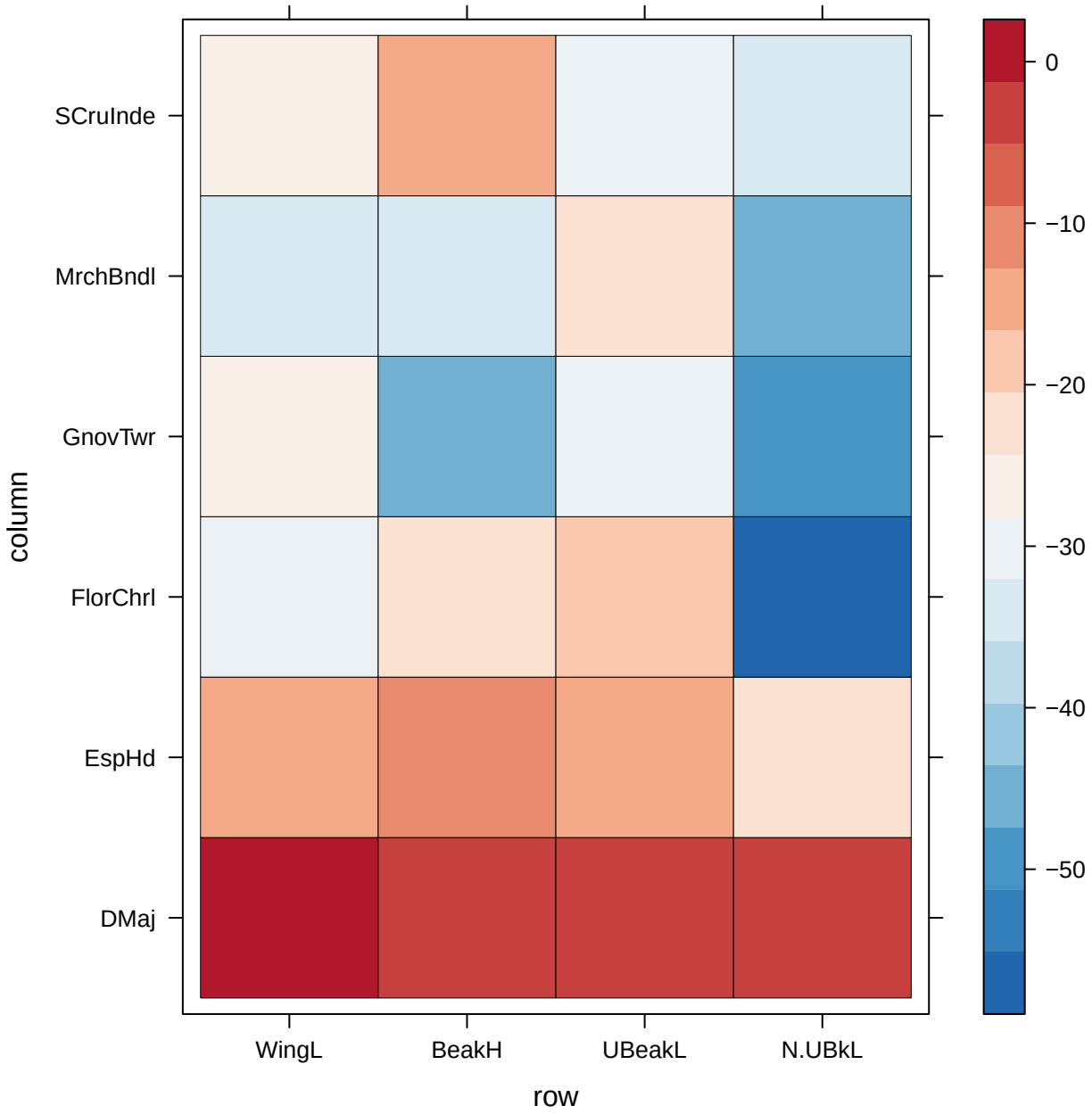
```
## Loading required package:  rasterVis
## Loading required package:  raster
## Loading required package:  sp
##
## Attaching package:  'raster'
##
## The following object is masked from 'package:magic':
##
##    shift
##
## The following objects are masked from 'package:ape':
##
##    edges, rotate, 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.

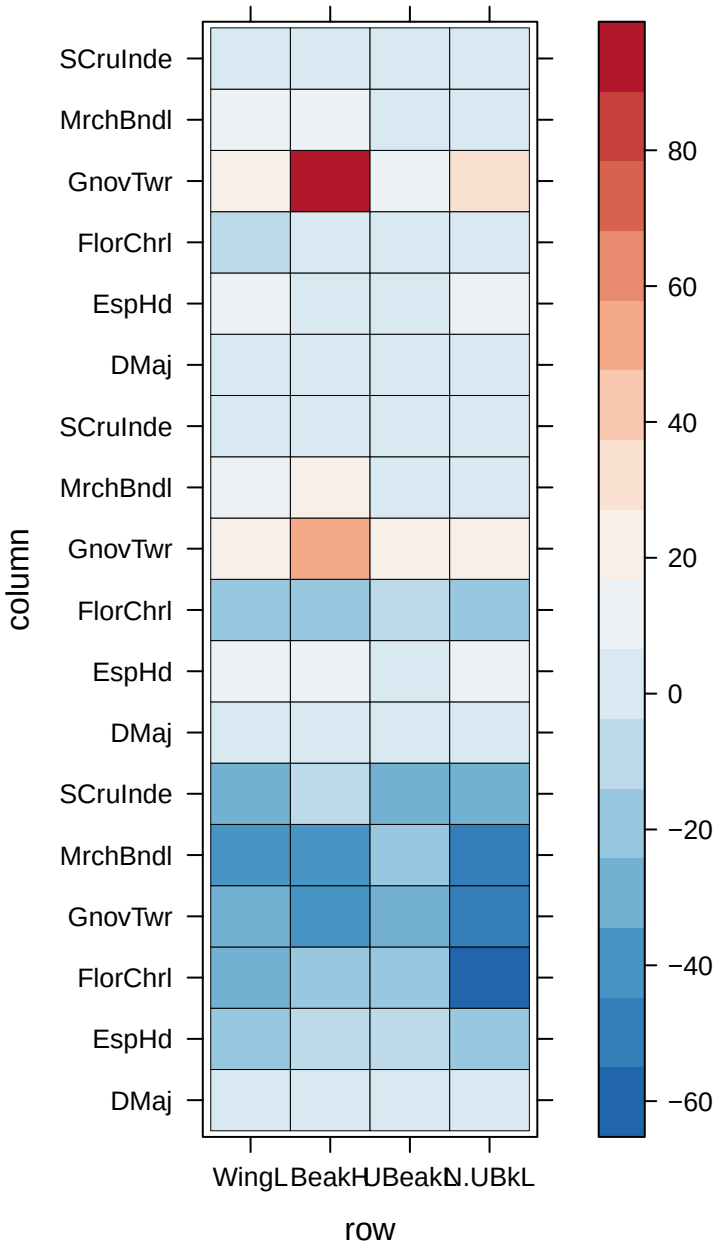
```
levelplot(t(sum_Tstats(res.finch)$p.value),
          colorkey=my.ckey, par.settings=my.theme,border="black")
```




```
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")
```

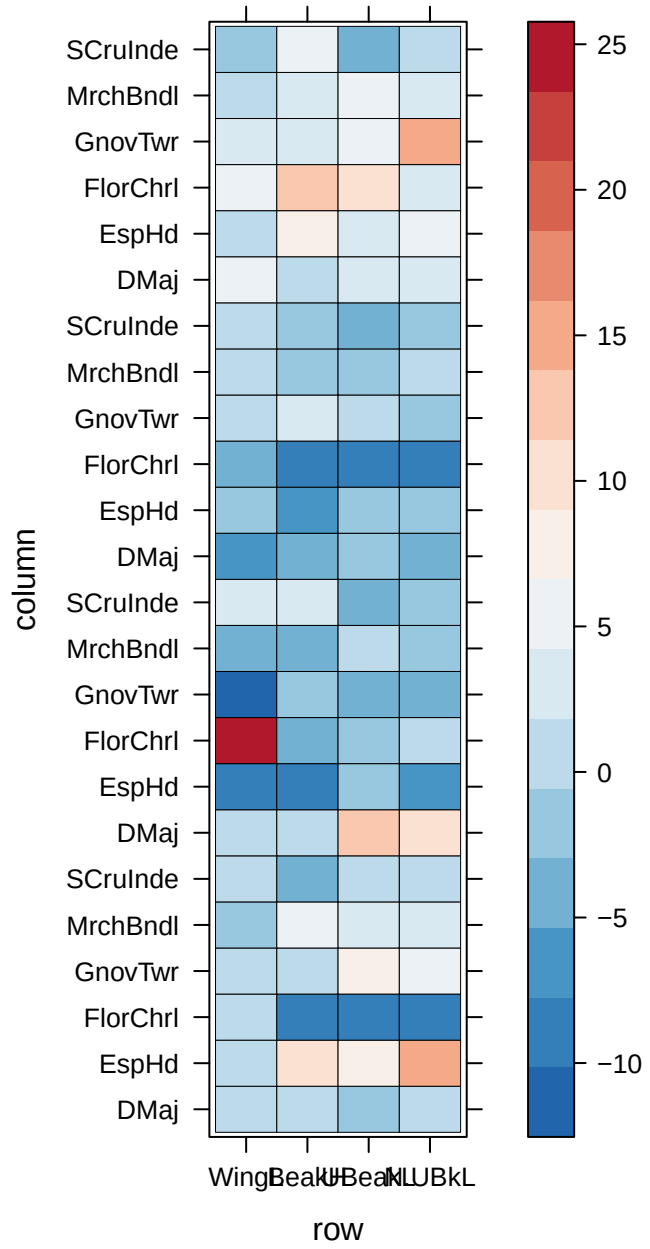



```
levelplot(cbind(t(ses(res.finch$Tstats$T_IP.IC, res.finch$Tstats$T_IP.IC_nm)$ses),
                t(ses(res.finch$Tstats$T_IC.IR, res.finch$Tstats$T_IP.IC_nm)$ses),
                t(ses(res.finch$Tstats$T_PC.PR, res.finch$Tstats$T_IP.IC_nm)$ses))
          , colorkey=my.ckey, par.settings=my.theme,border="black")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

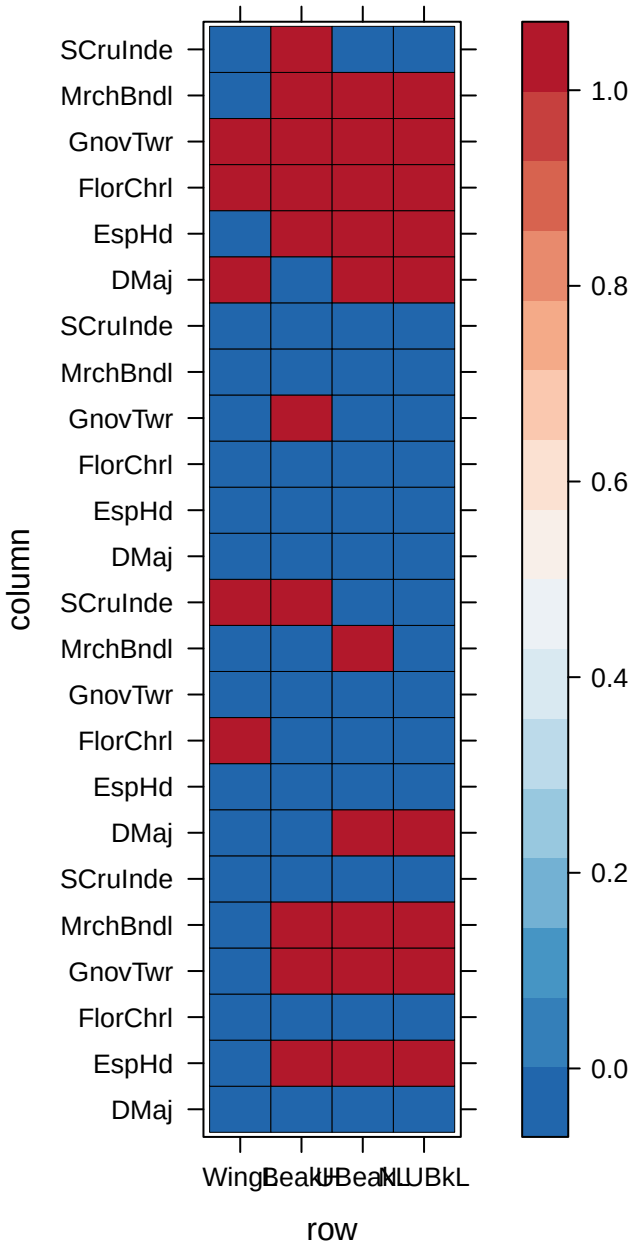


An other example using ses.listofindex. The first plot represent "ses" values and the second one the result of a test with H0: observed index value are greater than what we can expect using the null model (alpha=2.5%).

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

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")
```



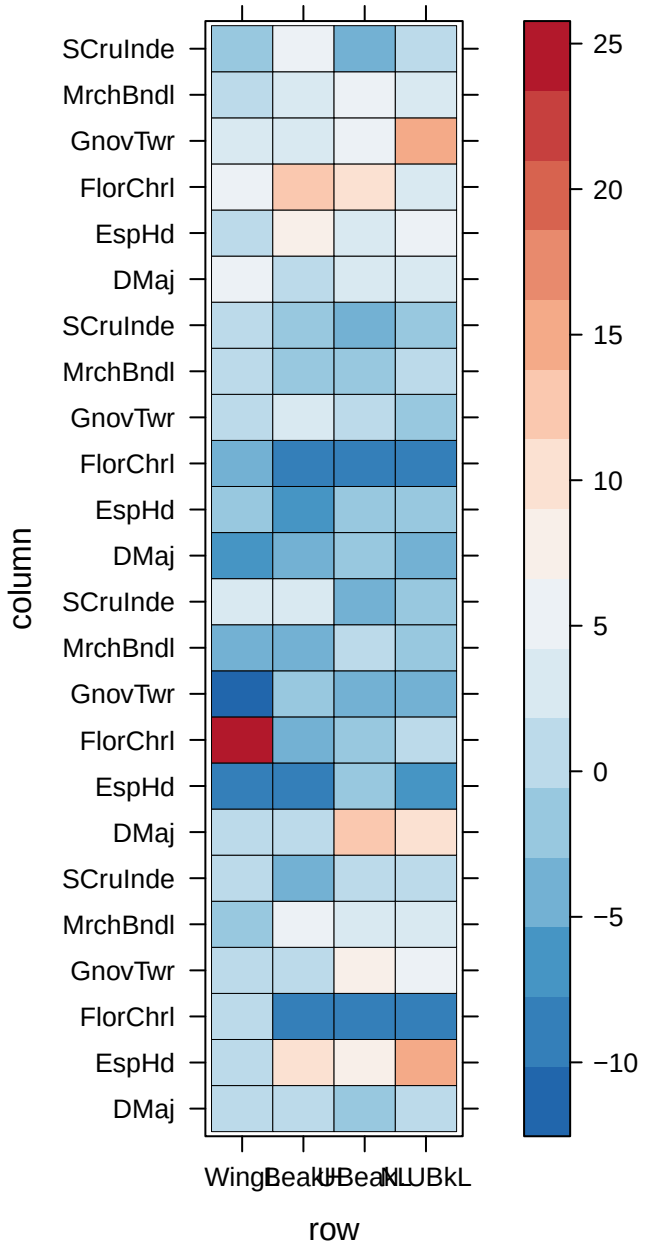

```
levelplot(t(rbind(ses.list[[1]]$ses>ses.list[[1]]$ses.sup,
                  ses.list[[2]]$ses>ses.list[[2]]$ses.sup,
                  ses.list[[3]]$ses>ses.list[[3]]$ses.sup,
                  ses.list[[4]]$ses>ses.list[[4]]$ses.sup)),
          colorkey=my.ckey, par.settings=my.theme,border="black")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



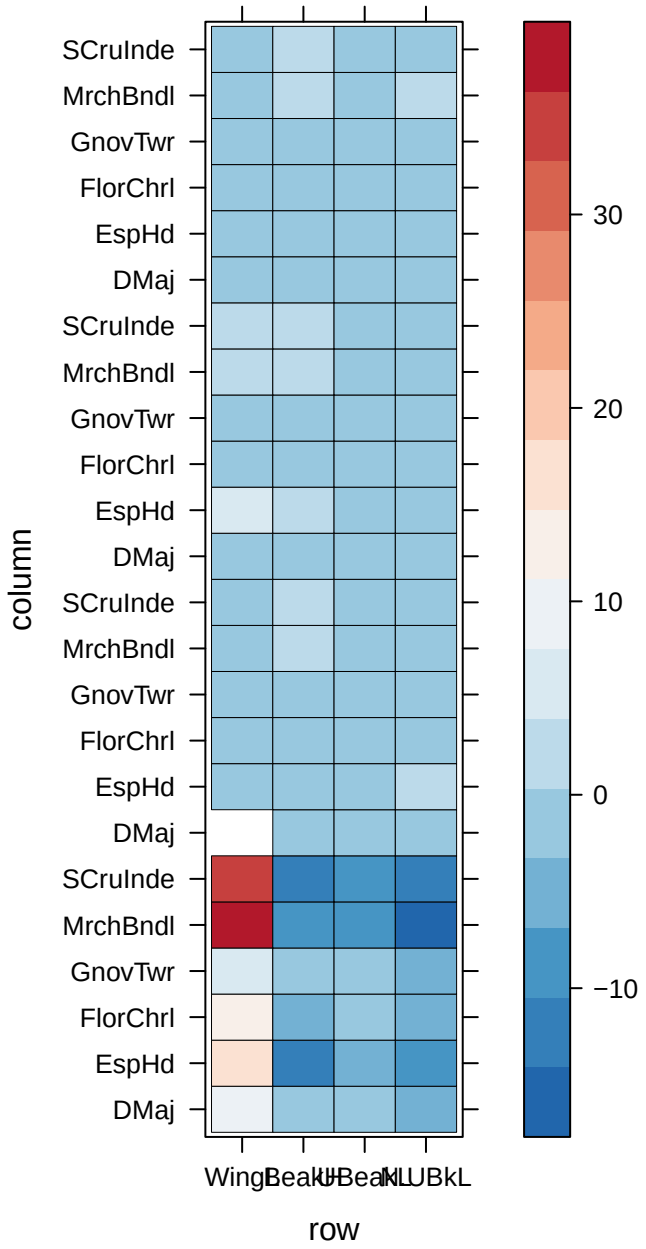
Compare metrics calculate on individual against metrics calculate after populationnal meaning

```
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 Conclusion

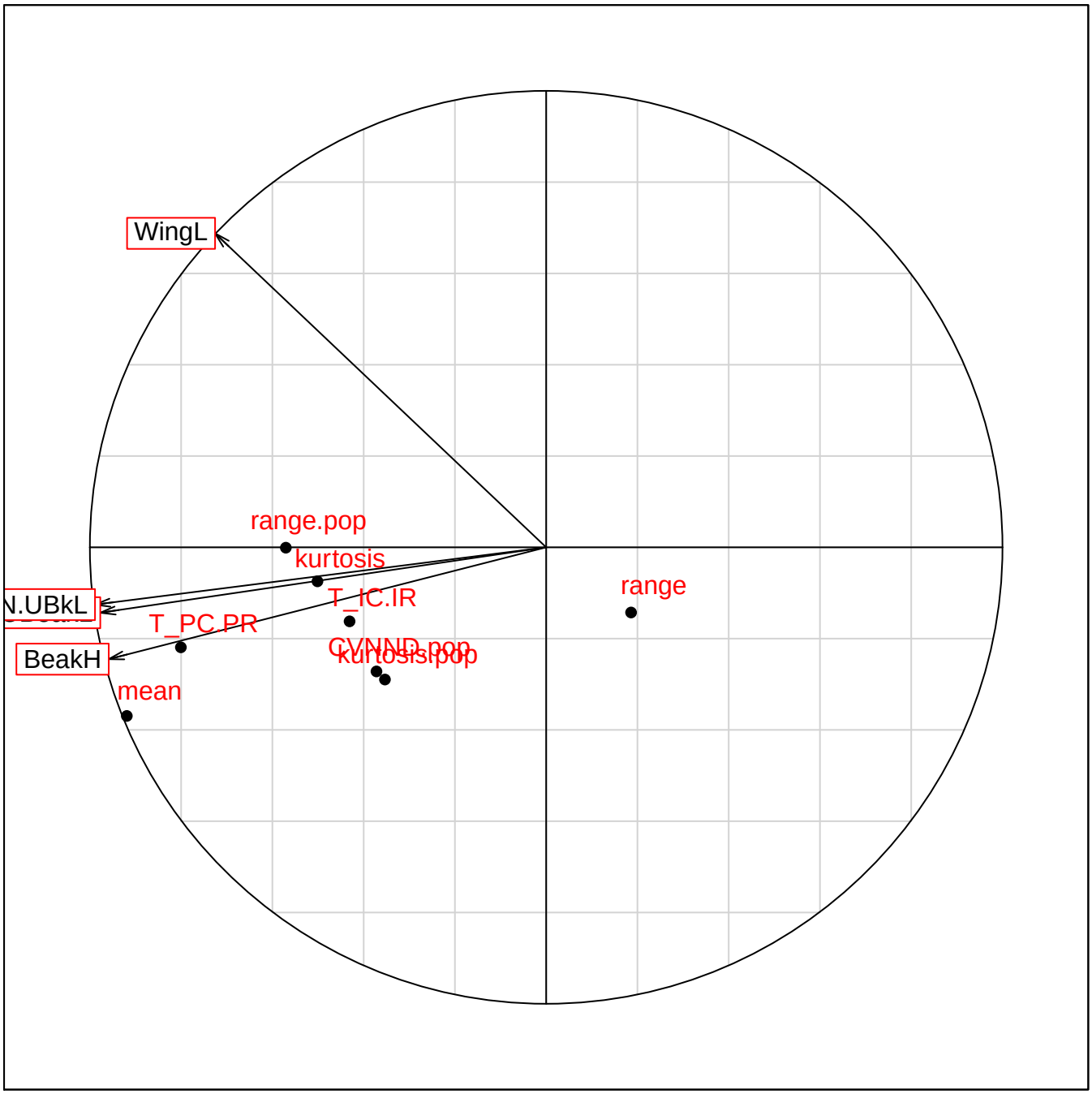
To finish, we can do a multivariate analysis of the metrics obtain during this tutorial. Analysis dudi 1 put together all traits by meaning the SES values for each metrics in each sites whereas analysis dudi 2 analyse all combination of traits / sites / metrics

```
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)
```

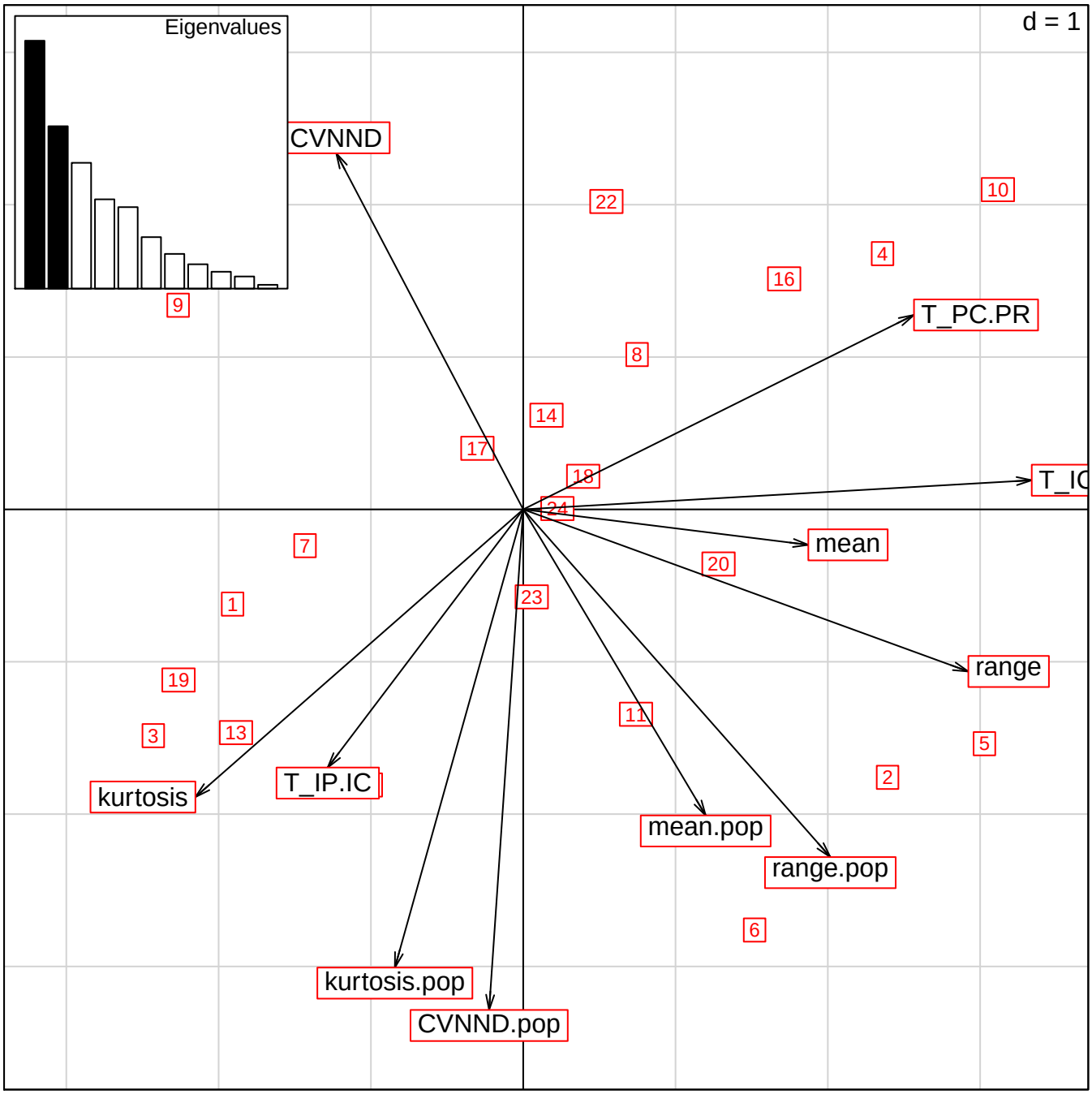
```
 }
 colnames(matfordudi)<-names(ses.list)
 rownames(matfordudi)<-colnames(traits.finch)

 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)

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

##
##  iter imp variable
##   1   1  kurtosis.pop
##   1   2  kurtosis.pop
##   1   3  kurtosis.pop
##   1   4  kurtosis.pop
##   1   5  kurtosis.pop
##   2   1  kurtosis.pop
##   2   2  kurtosis.pop
##   2   3  kurtosis.pop
##   2   4  kurtosis.pop
##   2   5  kurtosis.pop
##   3   1  kurtosis.pop
##   3   2  kurtosis.pop
##   3   3  kurtosis.pop
##   3   4  kurtosis.pop
##   3   5  kurtosis.pop
##   4   1  kurtosis.pop
##   4   2  kurtosis.pop
##   4   3  kurtosis.pop
##   4   4  kurtosis.pop
##   4   5  kurtosis.pop
##   5   1  kurtosis.pop
##   5   2  kurtosis.pop
##   5   3  kurtosis.pop
##   5   4  kurtosis.pop
##   5   5  kurtosis.pop

matfordudi2<-complete(mice(matfordudi2))

##
##  iter imp variable
##   1   1  kurtosis.pop
##   1   2  kurtosis.pop
##   1   3  kurtosis.pop
##   1   4  kurtosis.pop
```

```
##   1   5  kurtosis.pop
##   2   1  kurtosis.pop
##   2   2  kurtosis.pop
##   2   3  kurtosis.pop
##   2   4  kurtosis.pop
##   2   5  kurtosis.pop
##   3   1  kurtosis.pop
##   3   2  kurtosis.pop
##   3   3  kurtosis.pop
##   3   4  kurtosis.pop
##   3   5  kurtosis.pop
##   4   1  kurtosis.pop
##   4   2  kurtosis.pop
##   4   3  kurtosis.pop
##   4   4  kurtosis.pop
##   4   5  kurtosis.pop
##   5   1  kurtosis.pop
##   5   2  kurtosis.pop
##   5   3  kurtosis.pop
##   5   4  kurtosis.pop
##   5   5  kurtosis.pop
```

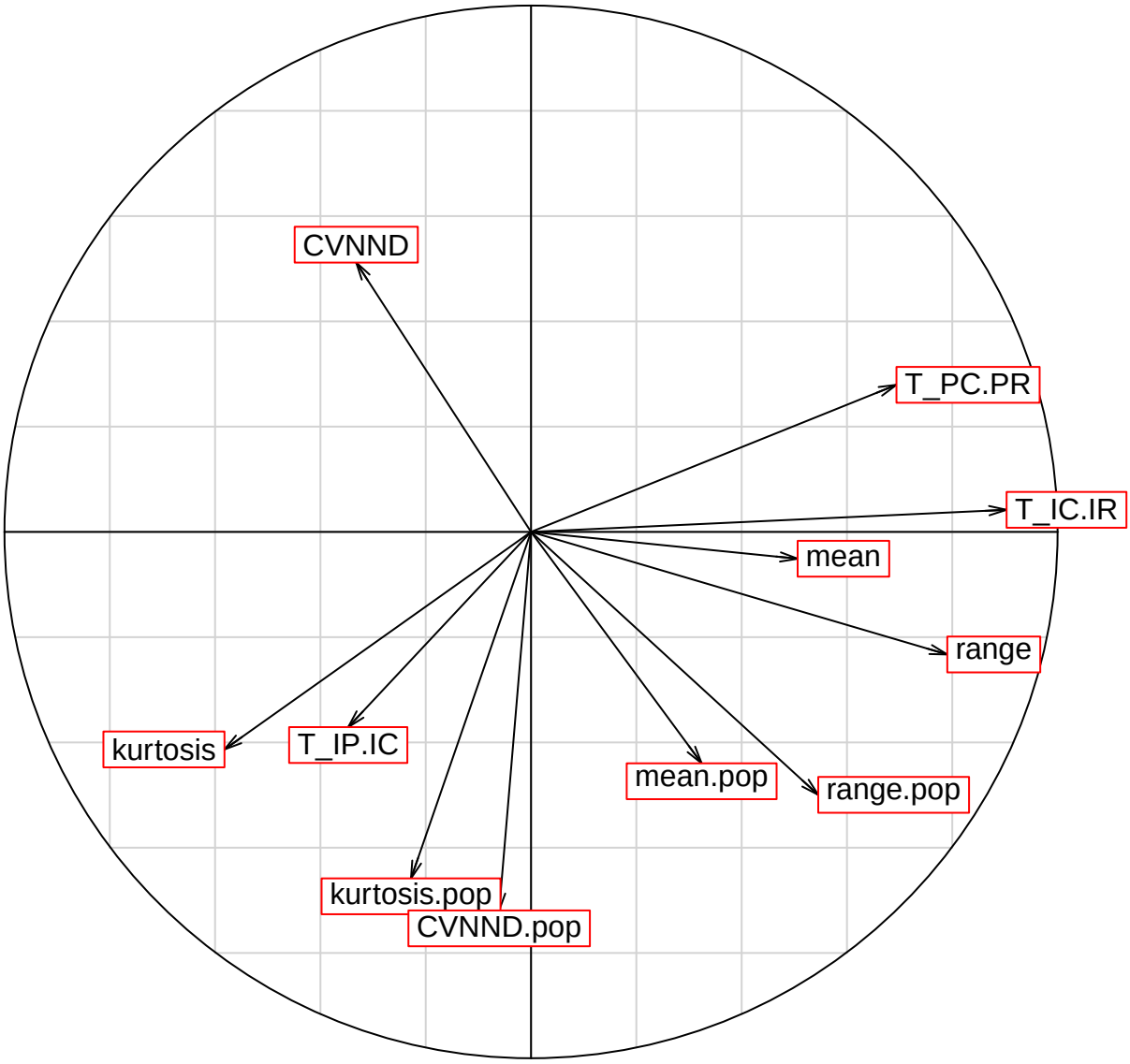
```
res.dudi<-dudi.pca(t(matfordudi), scan=F, nf=2)
s.corcircle(res.dudi$co)
s.label(res.dudi$li, add.plot=T, clabel = 0, pch=16)
s.label(res.dudi$li+0.05, add.plot=T, boxes=F)
```



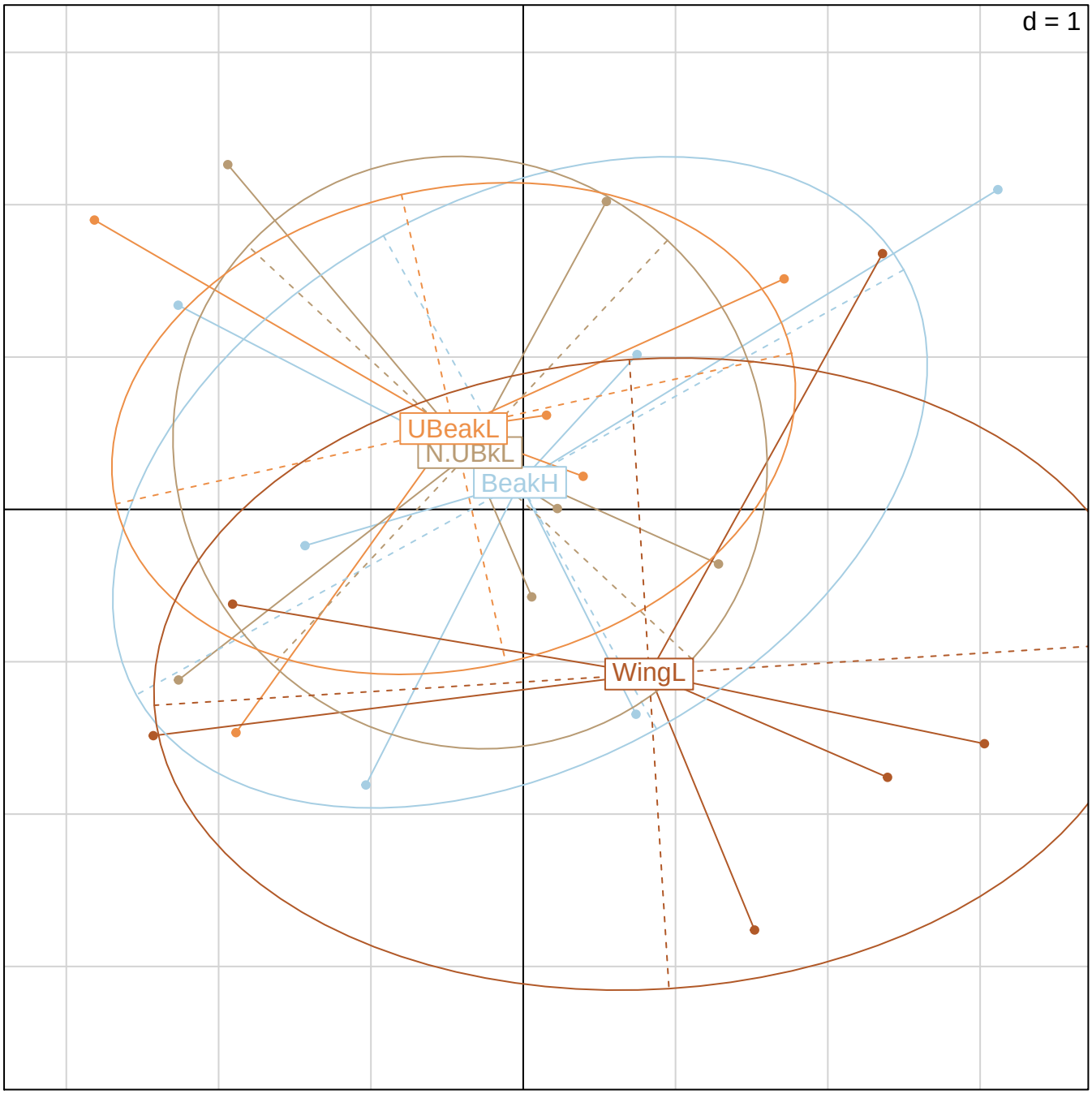

```
res.dudi2<-dudi.pca(matfordudi2, scan=F, nf=2)
scatter(res.dudi2)
```



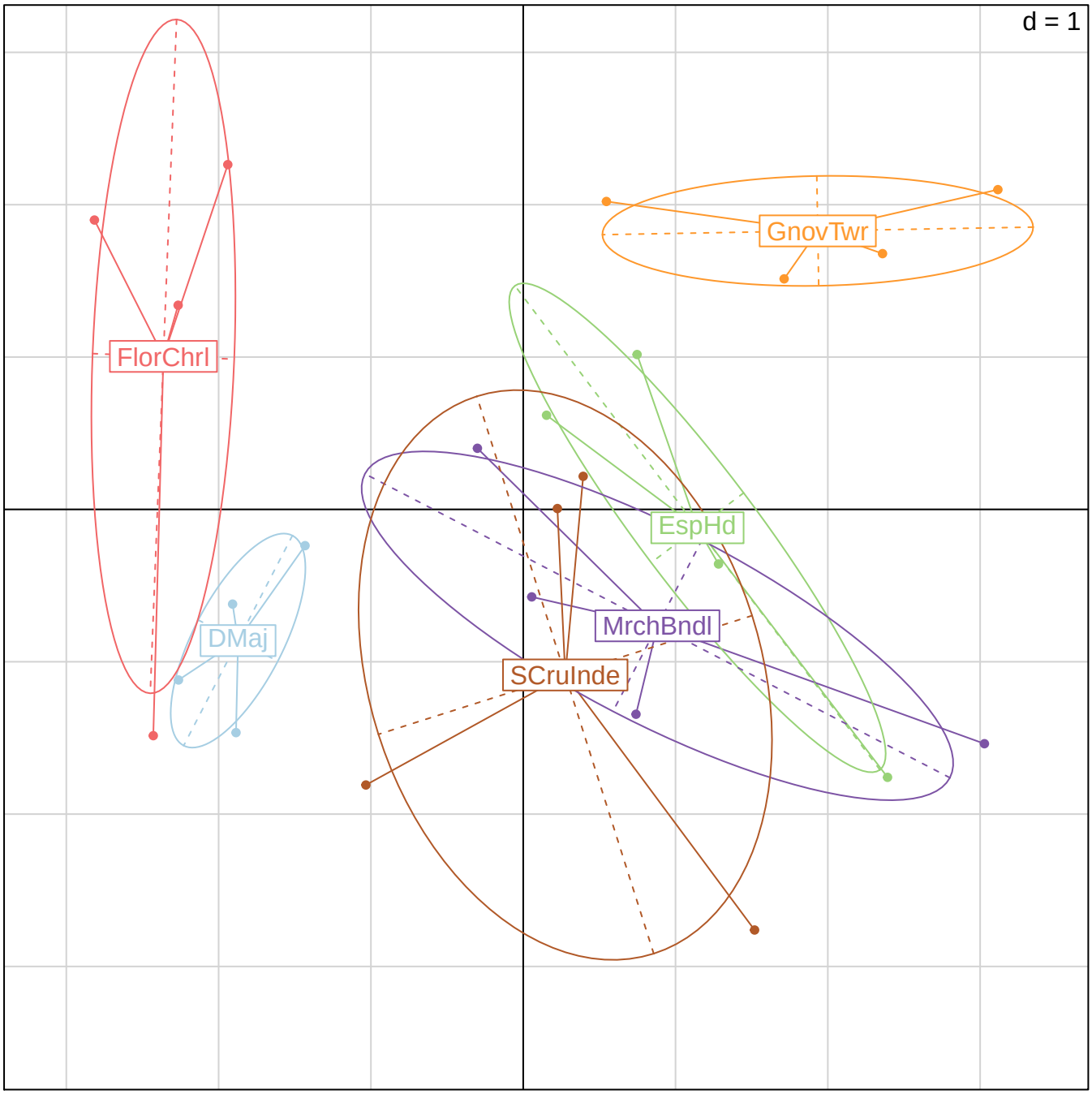

```
s.corcircle(res.dudi2$co)
```




```
s.class(res.dudi2$li, as.factor(c(rep("WingL",6), rep("BeakH",6), rep("UBeakL",6), rep("N.UBk
```




```
s.class(res.dudi2$li, as.factor(rep(c("DMaj","EspHd","FlorChrl","GnovTwr","MrchBndl","SCruInd
```



8 Conclusion

9 Acknowledgment