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

Adrien Taudiere *

EPHE

CEFE - Centre d'Ecologie Fonctionnelle et Evolutive

May 23, 2014

Abstract

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

*adrien.taudiere@cefe.cnrs.fr

Contents

1 Introduction

2 Installing the package cati

```
install.packages("C:/Users/taudiere/Desktop/cati/pkg/cati_0.6.zip", repos = NULL)

## Installing package into 'C:/Users/taudiere/Documents/R/win-library/3.0'
## (as 'lib' is unspecified)

# install.packages('cati', repos='http://R-Forge.R-project.org')
```

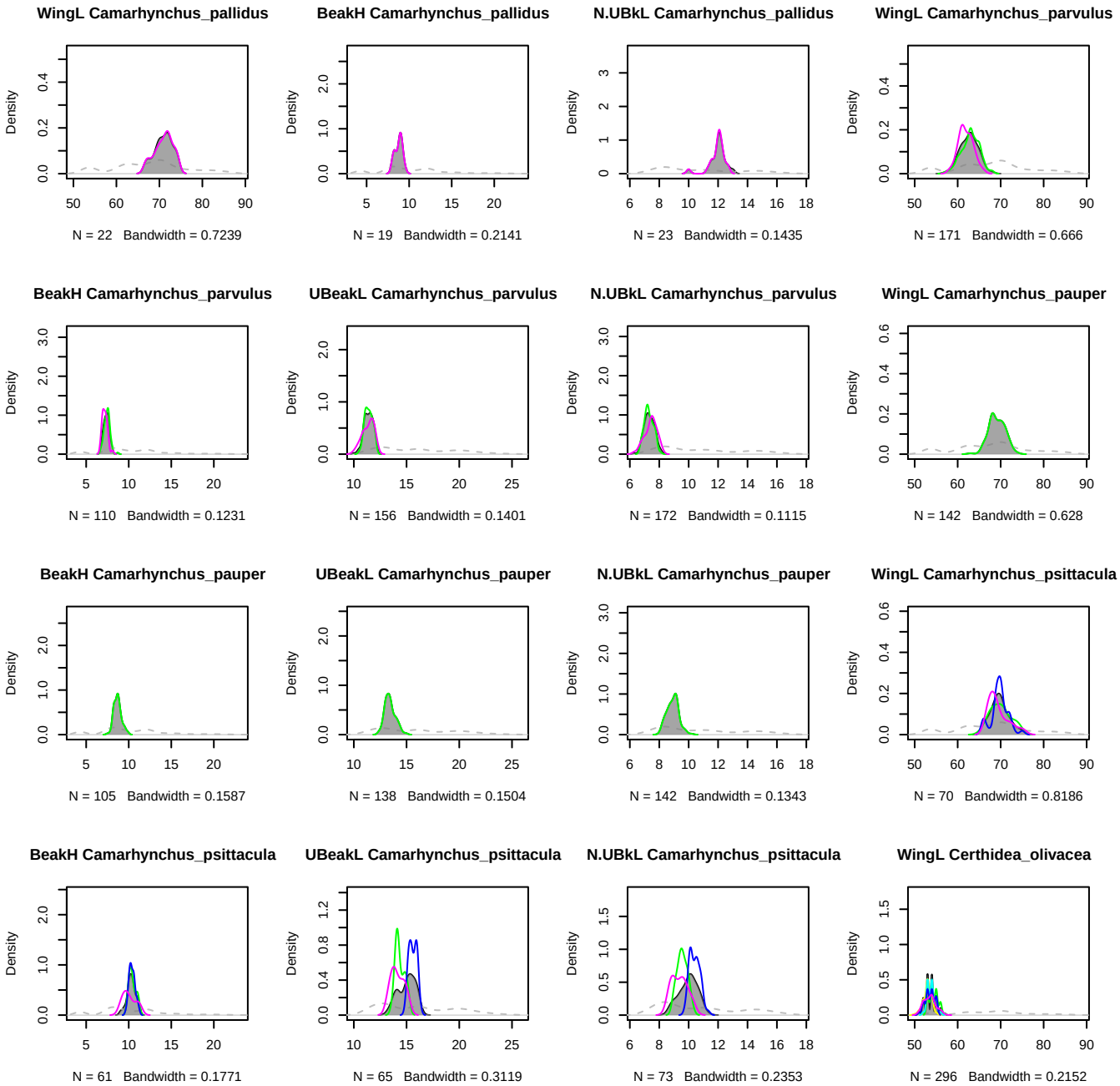
```
library(cati, warn.conflicts = FALSE)
data(finch.ind)

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

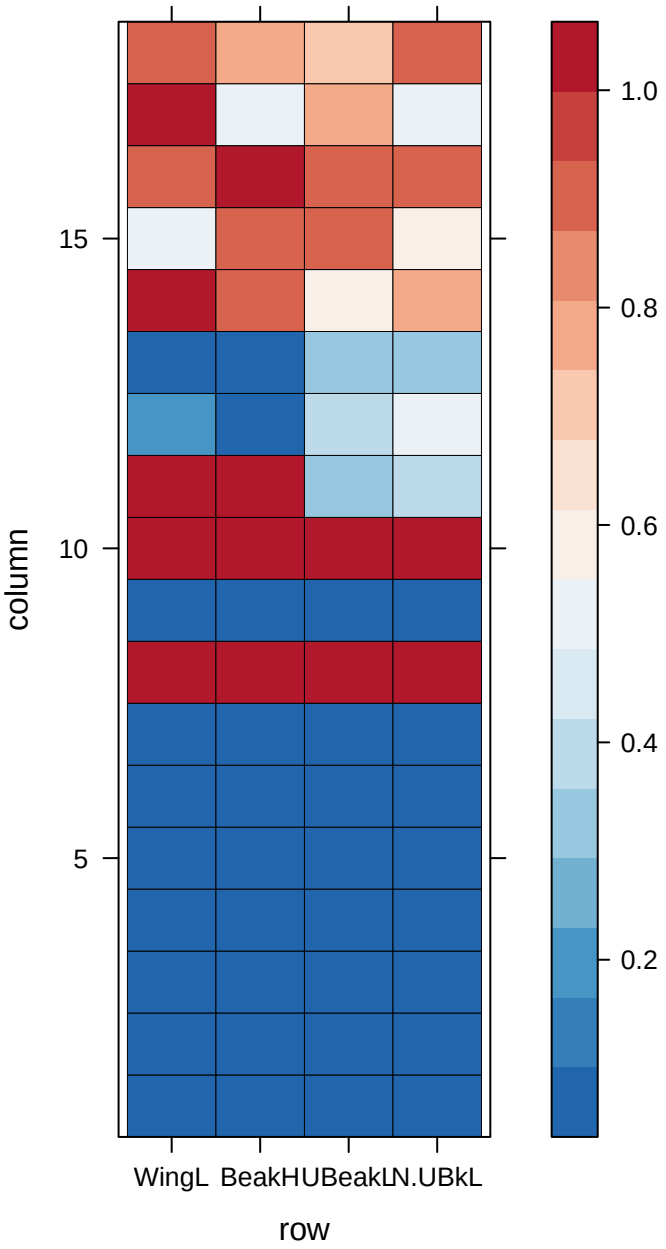
3 Description of distribution

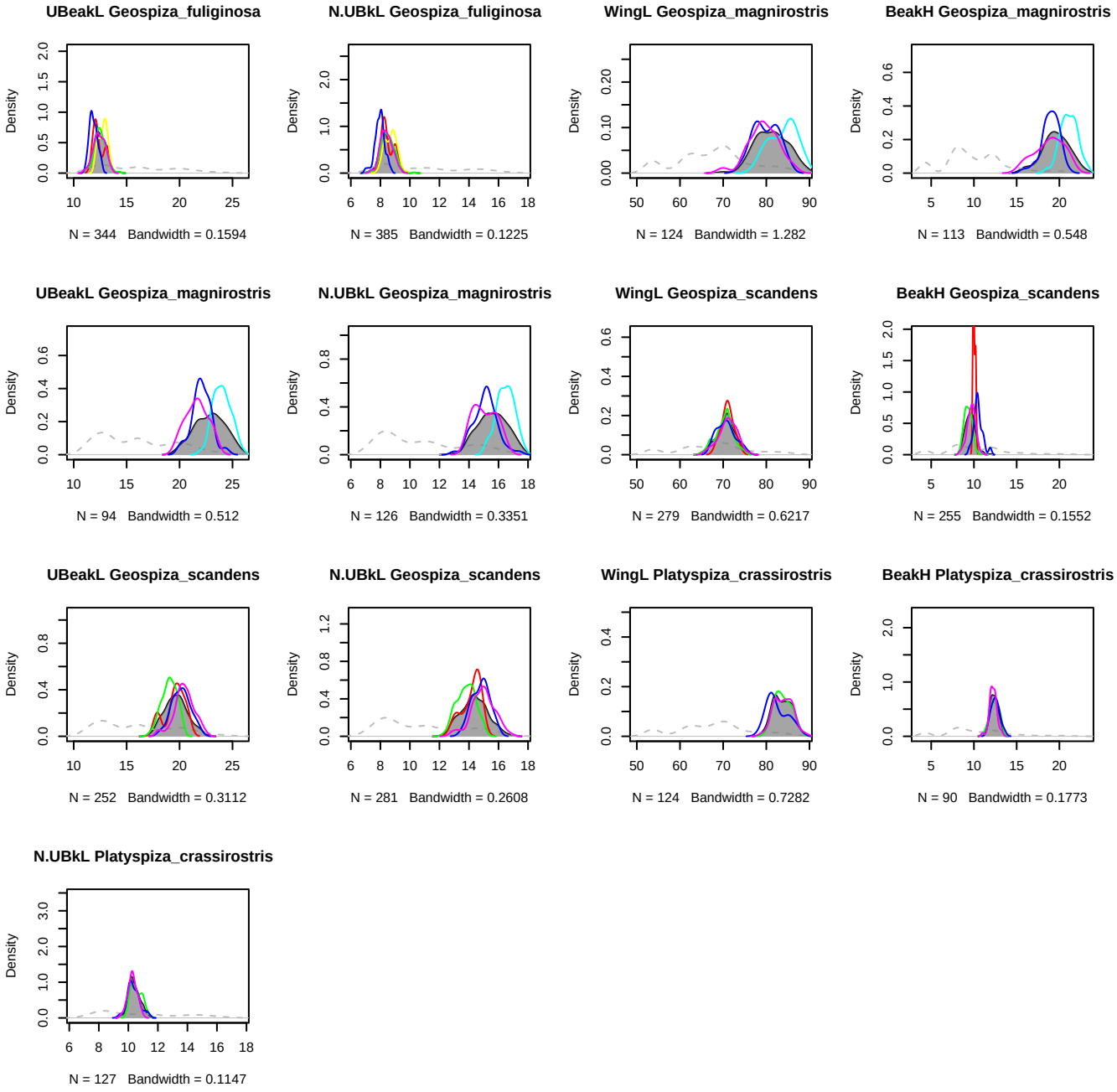
3.1 Plot the density of traits

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)
plot_dens(traits.finch, sp.finch, ind.plot.finch, ylim.cex = 3, plot.ask = F,
    multipanel = F, leg = F)
```

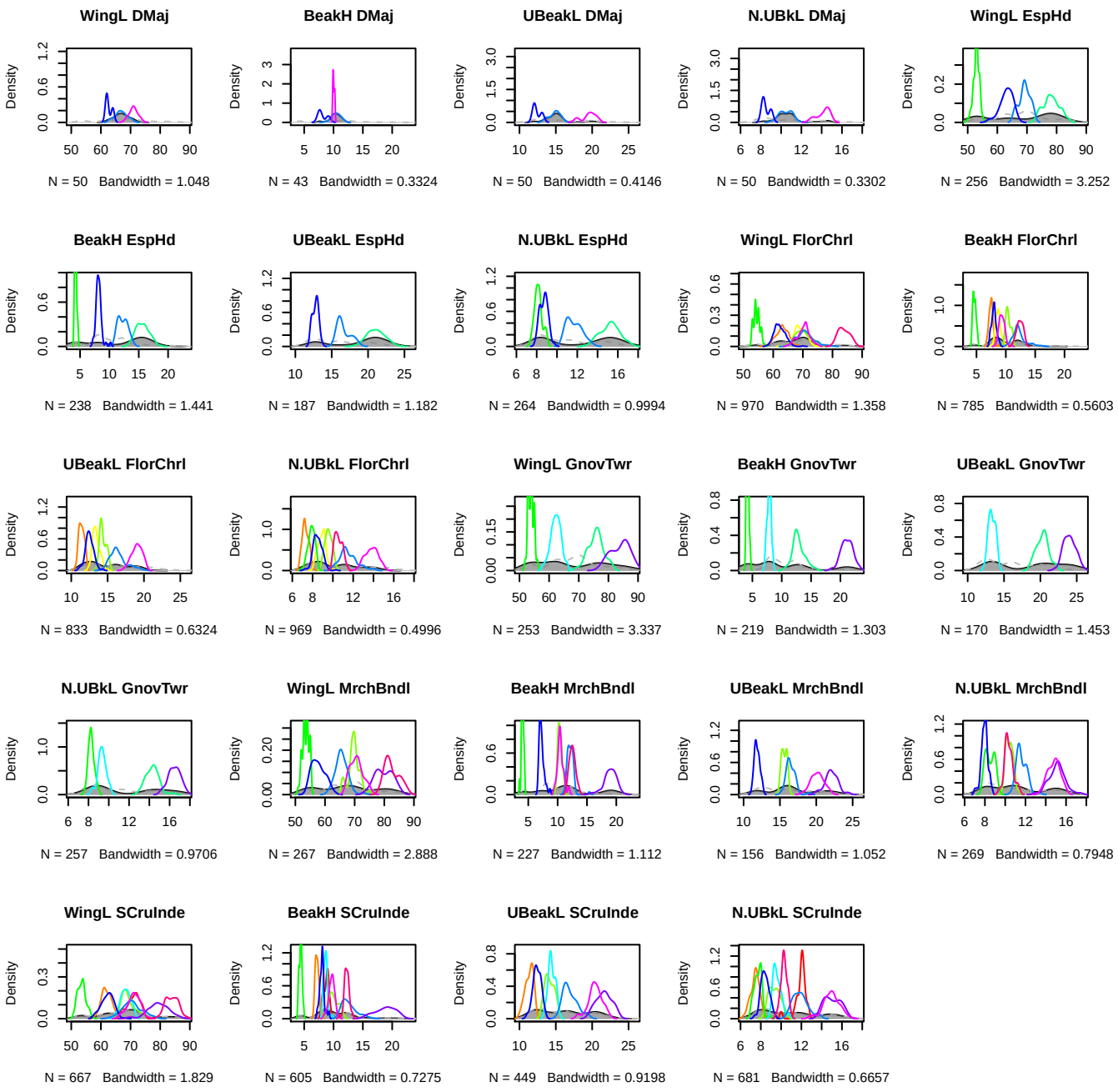



15
10
5
column
WingL
BeakHU
BeakLN.U
BkL
row
1.0
0.8
0.6
0.4
0.2

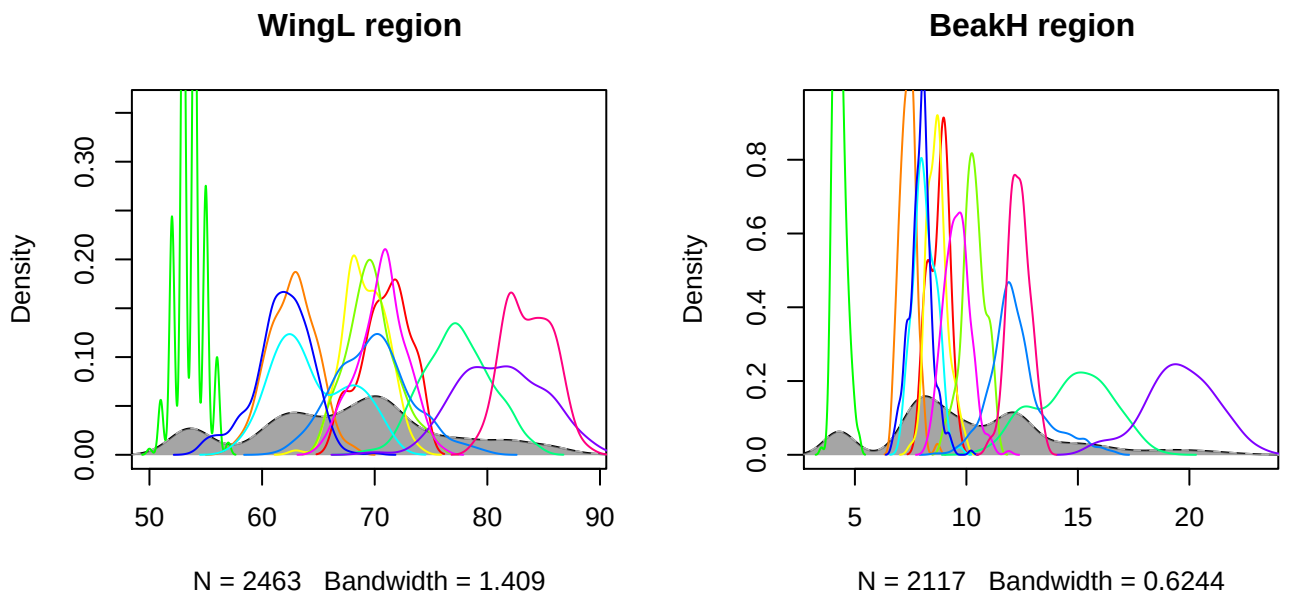

UBeakL Geospiza_fuliginosa
Density
N = 344 Bandwidth = 0.1594
N.UBkL Geospiza_fuliginosa
Density
N = 385 Bandwidth = 0.1225
WingL Geospiza_magnirostris
Density
N = 124 Bandwidth = 1.282
BeakH Geospiza_magnirostris
Density
N = 113 Bandwidth = 0.548
UBeakL Geospiza_magnirostris
Density
N = 94 Bandwidth = 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

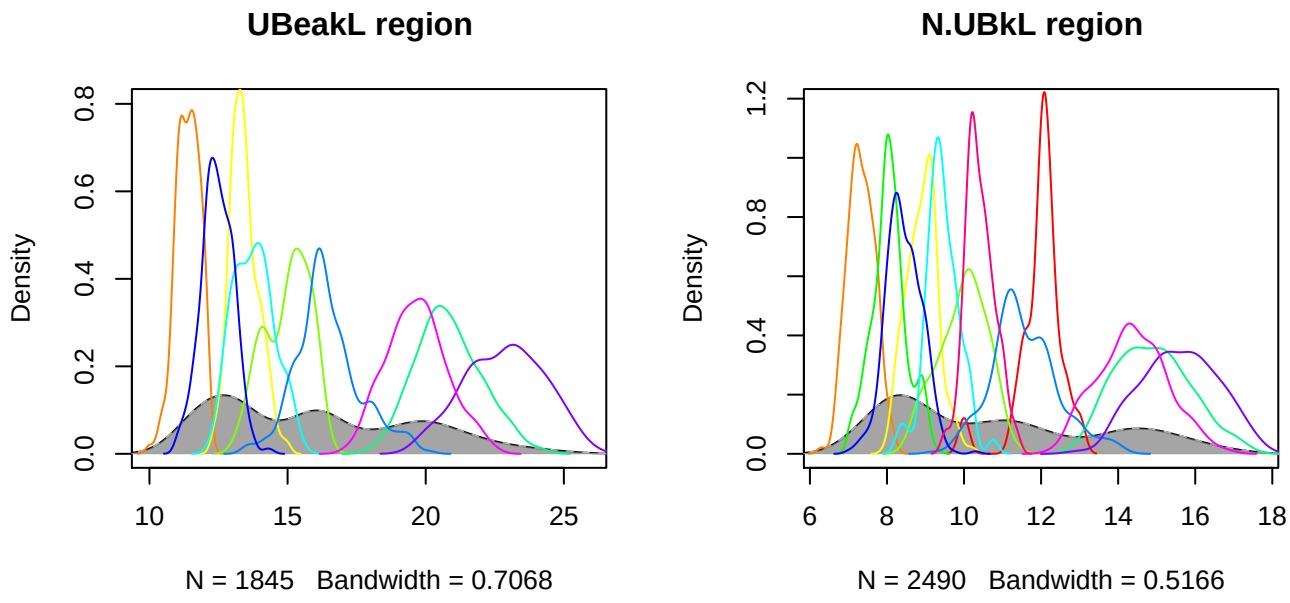
Then we can inverse the second and the third arguments to plot the distribution for all finches species.

```
par(mfrow = c(5, 5), cex = 0.5)
plot_dens(traits.finch, ind.plot.finch, sp.finch, ylim.cex = 8, plot.ask = F,
    multipanel = F, leg = F)
```

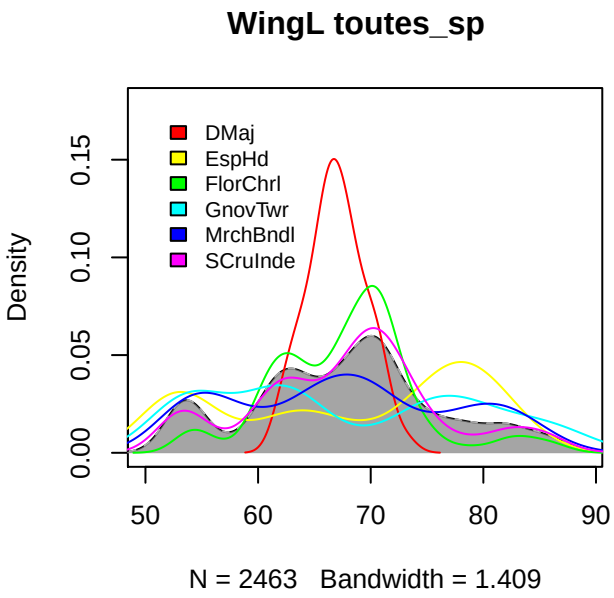


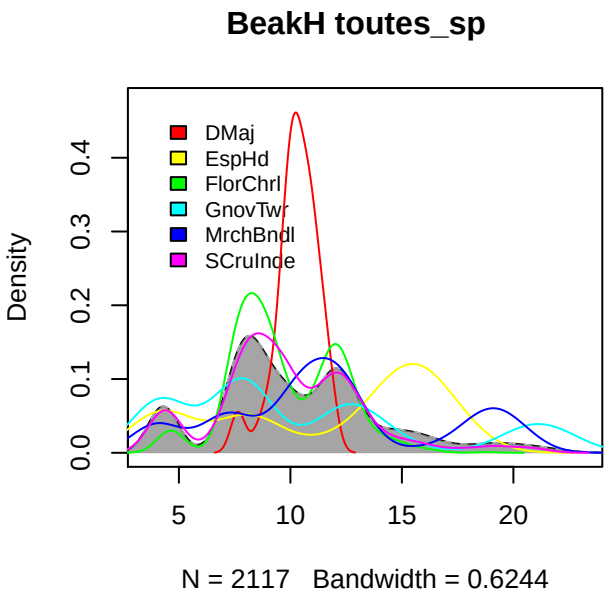

```
plot_dens(traits.finch, rep("region", times = dim(traits.finch)[1]), sp.finch,
    ylim.cex = 6, plot.ask = F, leg = F)
```

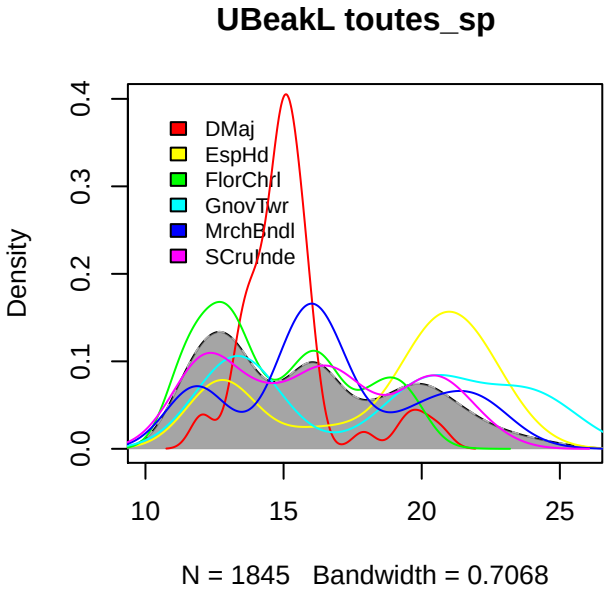


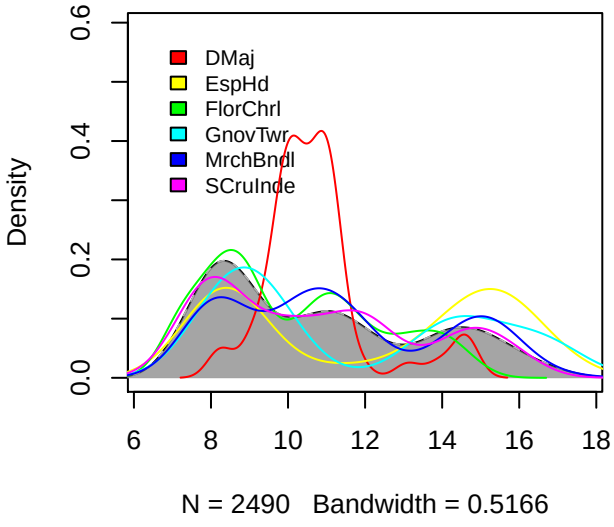



```
plot_dens(traits.finch, rep("toutes_sp", times = dim(traits.finch)[1]), ind.plot.finch,
    ylim.cex = 3, plot.ask = F)
```


WingL toutes_sp
Density
DMaj
EspHd
FlorChrl
GnovTwr
MrchBndl
SCruInde
0.00
0.05
0.10
0.15
50
60
70
80
90
N = 2463 Bandwidth = 1.409


BeakH toutes_sp
Density
DMaj
EspHd
FlorChrl
GnovTwr
MrchBndl
SCruInde
0.0
0.1
0.2
0.3
0.4
5
10
15
20
N = 2117 Bandwidth = 0.6244


UBeakL toutes_sp
Density
DMaj
EspHd
FlorChrl
GnovTwr
MrchBndl
SCruInde
0.0
0.1
0.2
0.3
0.4
10
15
20
25
N = 1845 Bandwidth = 0.7068


N.UBkL toutes_sp
Density
DMaj
EspHd
FlorChrl
GnovTwr
MrchBndl
SCruInde
0.0
0.2
0.4
0.6
6
8
10
12
14
16
18
N = 2490 Bandwidth = 0.5166

4 Decomposition of variances

4.1 Decomposition of within/among species variances

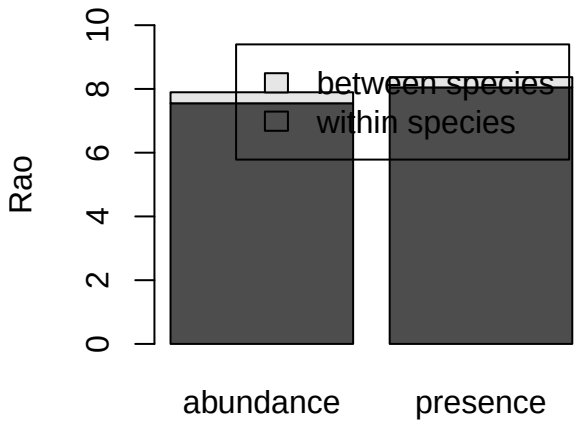
```
comm <- t(table(ind.plot.finch, 1:length(ind.plot.finch)))
comm.sp <- table(sp.finch, ind.plot.finch)
class(comm.sp) <- "matrix"

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))
```



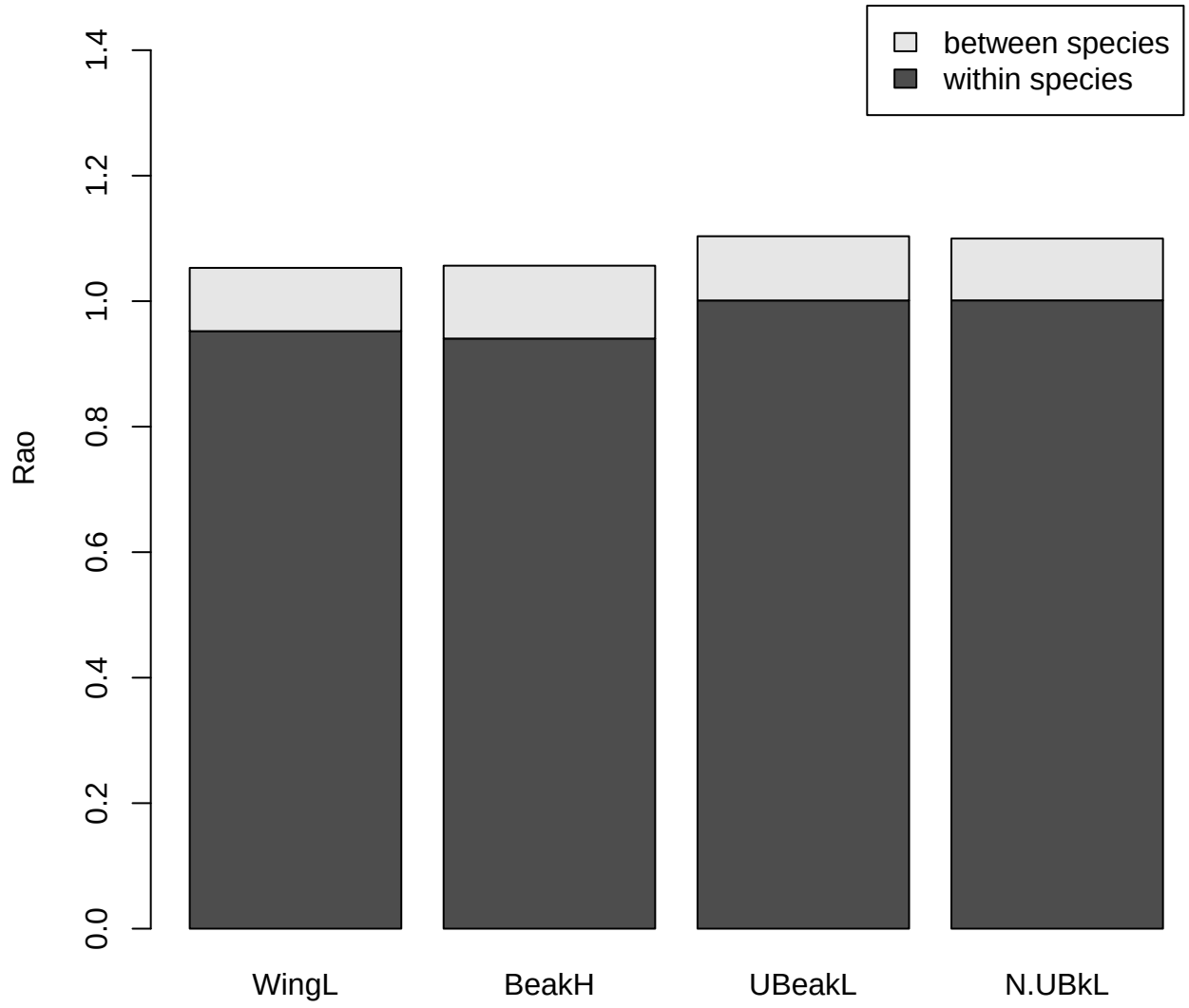
We can do this analysis for each trait separately. We need to replace (or exclude) NA values. For this example, we use the package mice to complete the data.

```
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)
```

```
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)
}
```

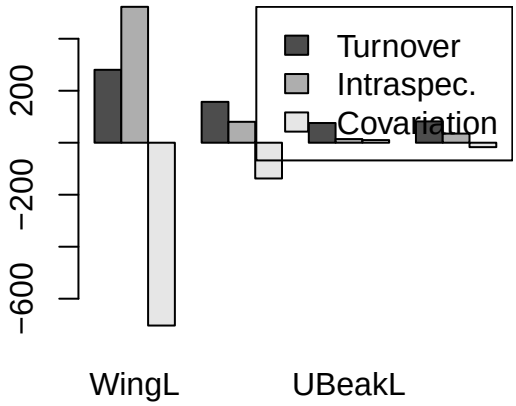
Plot the results by traits.

```
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))
```

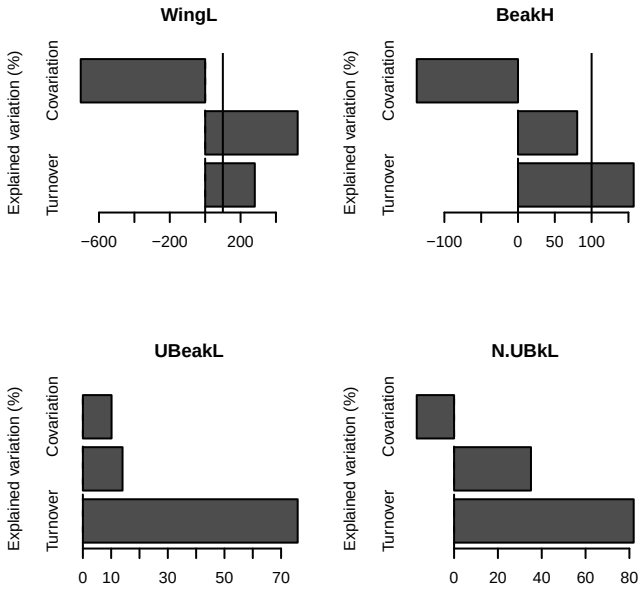


4.2 Decomposition of within/among species variances

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

barplot.decomp_within(res.decomp)
```




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

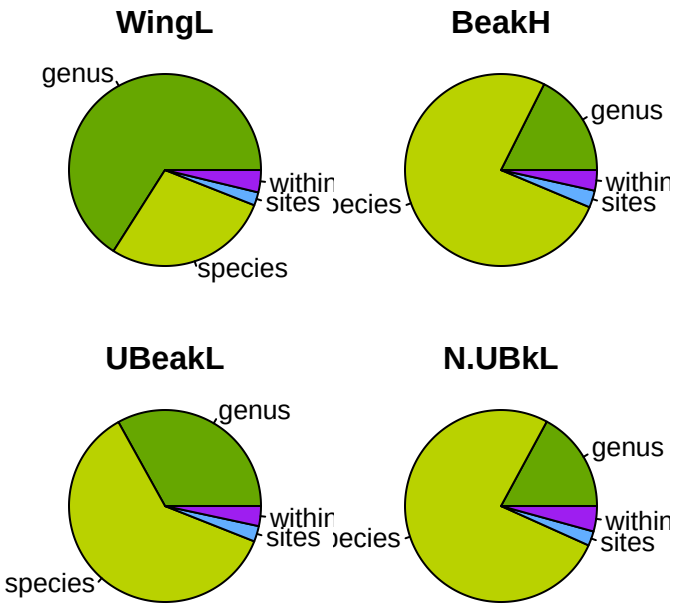



```
par(mfrow = c(1, 1))
```

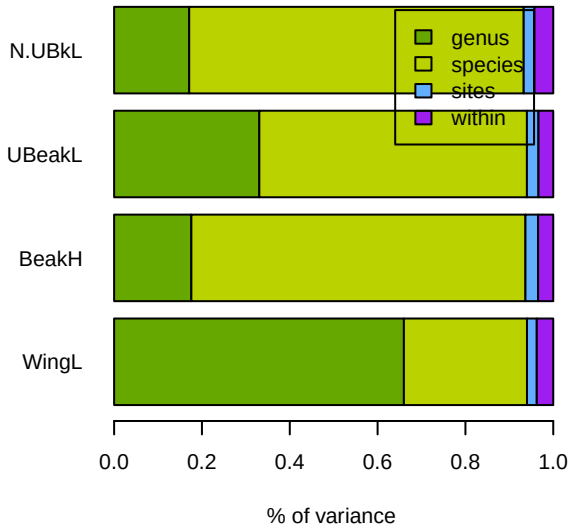
4.3 Decomposition of variances using nested factors

```
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))

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

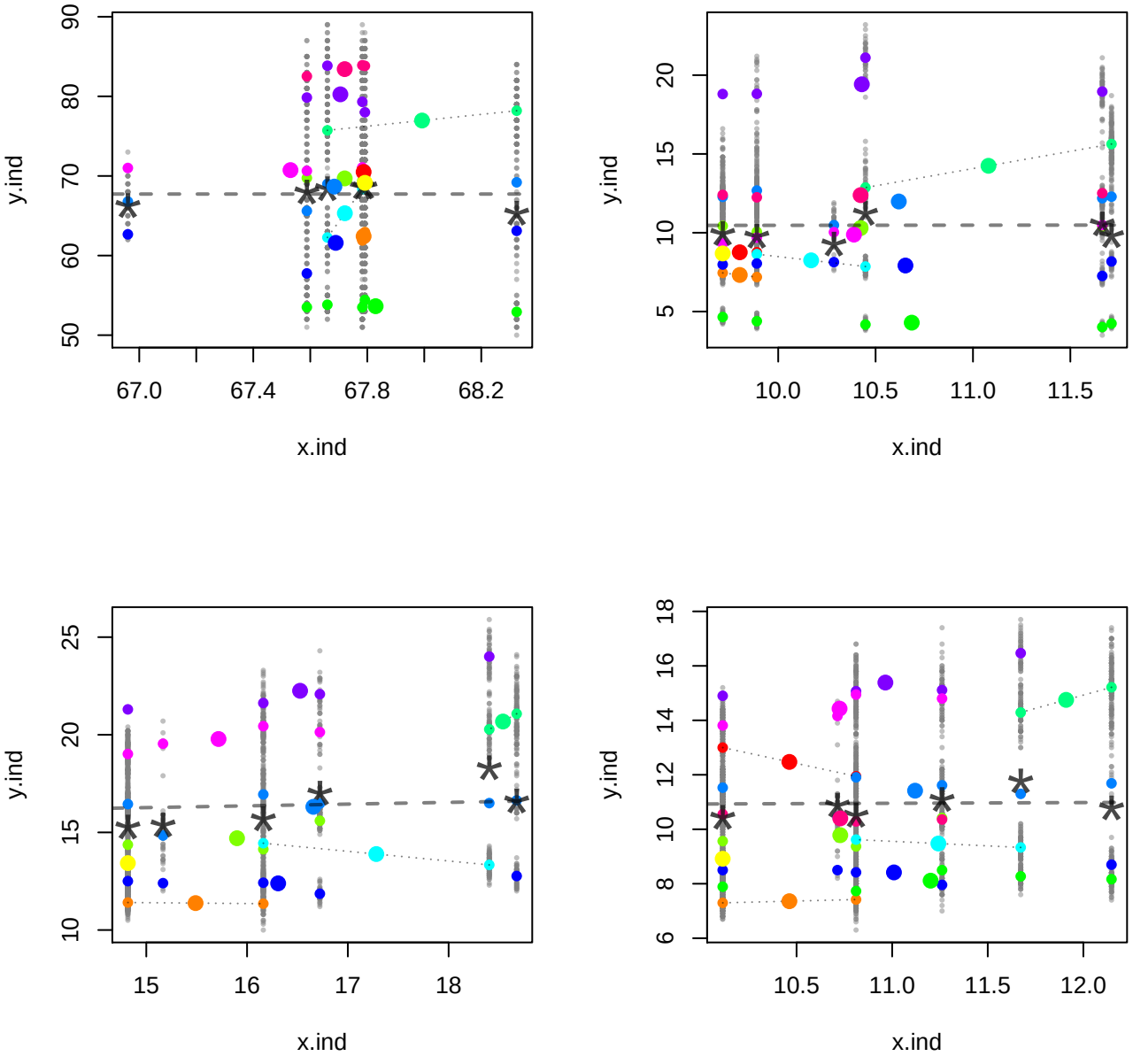



```
par(oldpar)

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


N.UBkL
UBeakL
BeakH
WingL
genus
species
sites
within
0.0
0.2
0.4
0.6
0.8
1.0
% of variance

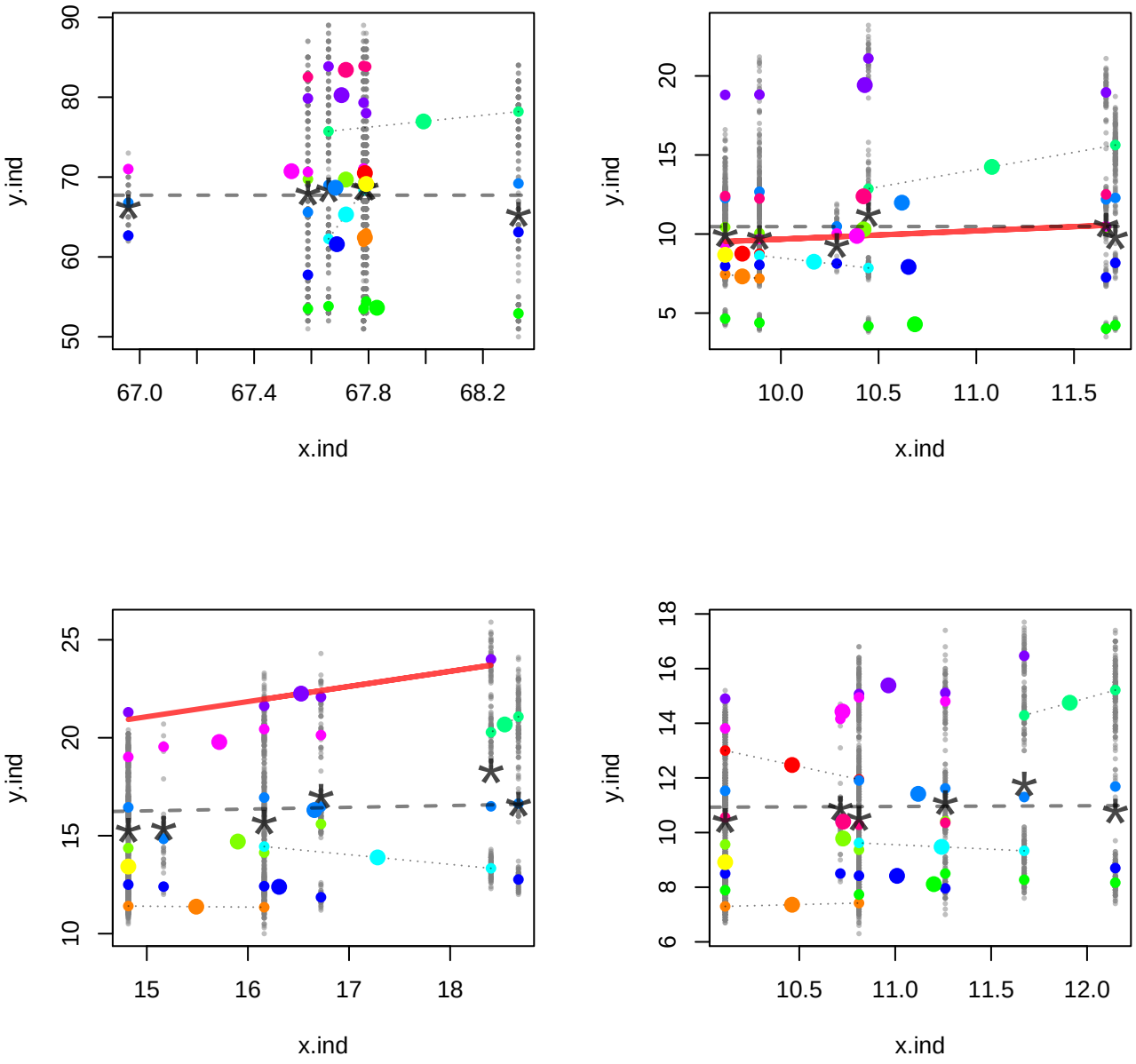
4.4 Plot the relation between populational trait means and sites traits means.

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



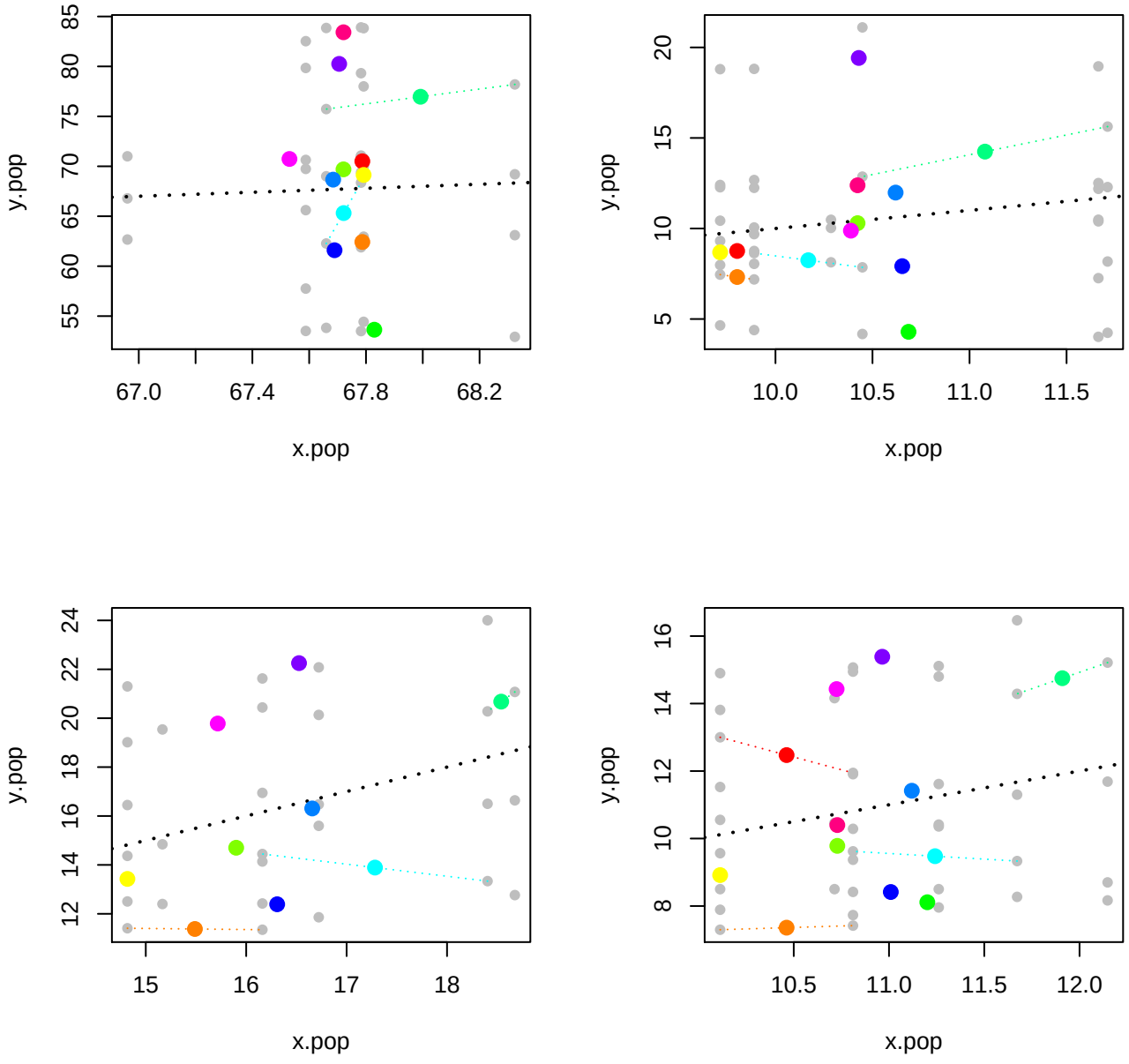
If we change the value of the threshold (alpha=10% instead of 5% and the minimum individual to represent singificativity fixed to 3 instead of 10 by default) we can see some significant relationships.

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

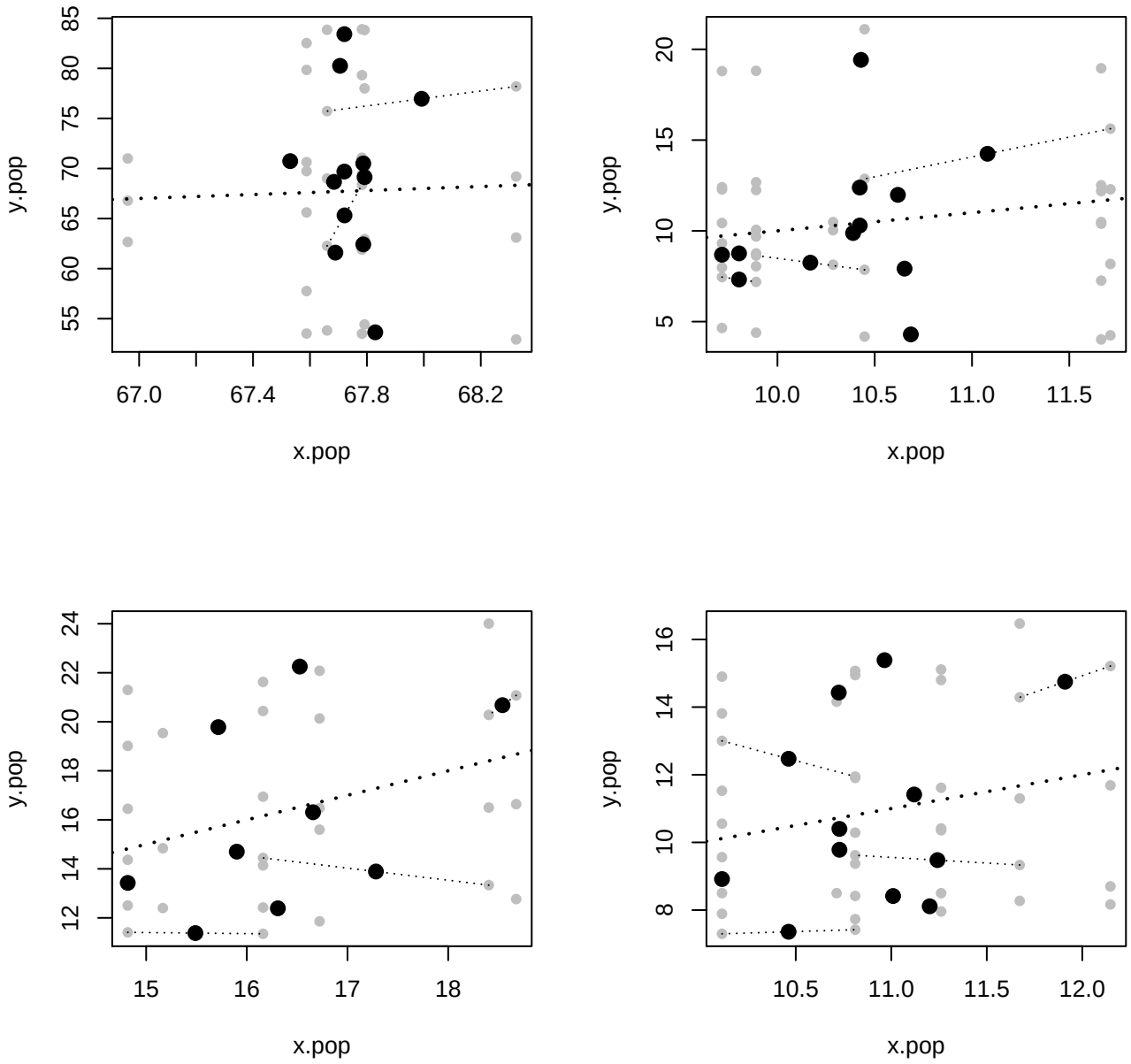


For a more simple figure, add the option resume=TRUE. Again if we change the value of the threshold (alpha=10% instead of 5% and the minimum individual to represent singificativity fixed to 3 instead of 10 by default) we can see some significant relationships.

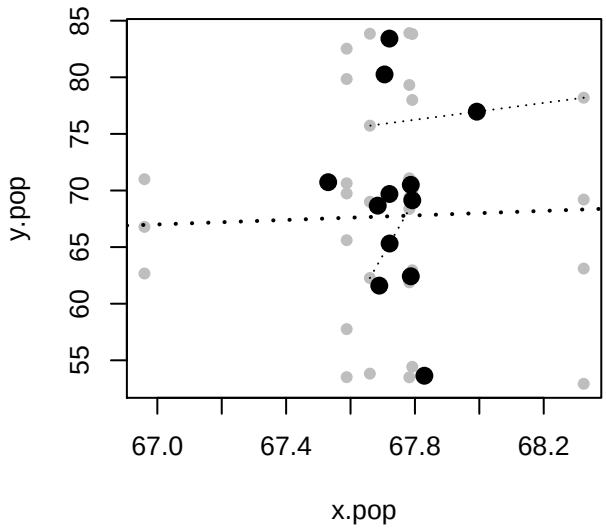
```
plot_sp_pop(traits.finch, ind.plot.finch, sp.finch, silent = TRUE, resume = TRUE,
    col.pop = "grey")
```

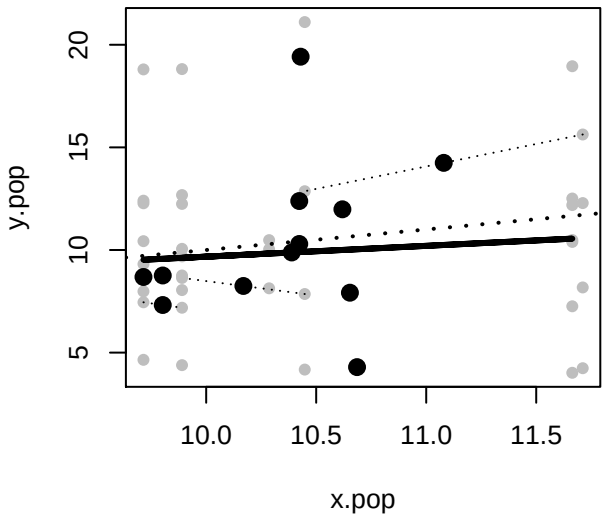


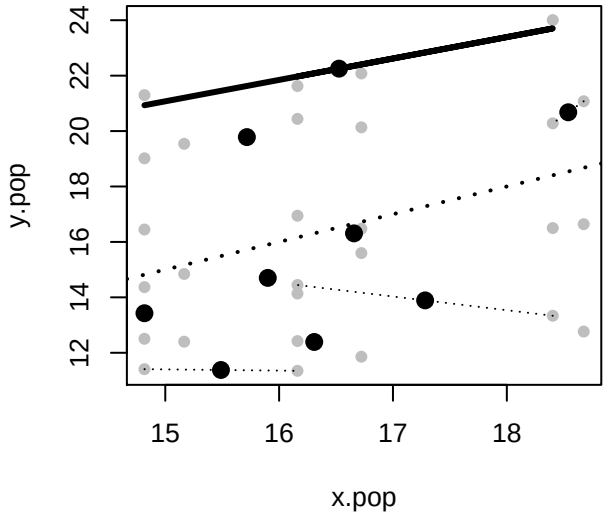

```
plot_sp_pop(traits.finch, ind.plot.finch, sp.finch, silent = TRUE, resume = TRUE,
    col.pop = "grey", col.sp = "black")
```

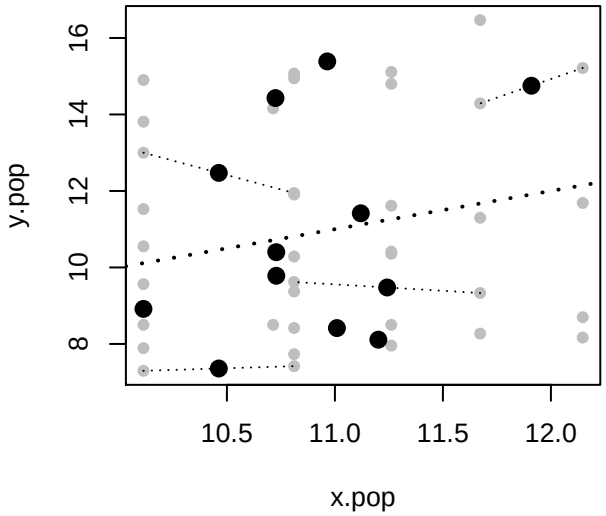



```
plot_sp_pop(traits.finch, ind.plot.finch, sp.finch, silent = TRUE, resume = TRUE,
    col.pop = "grey", col.sp = "black", p.val = 0.1, min.ind.signif = 3)
```


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

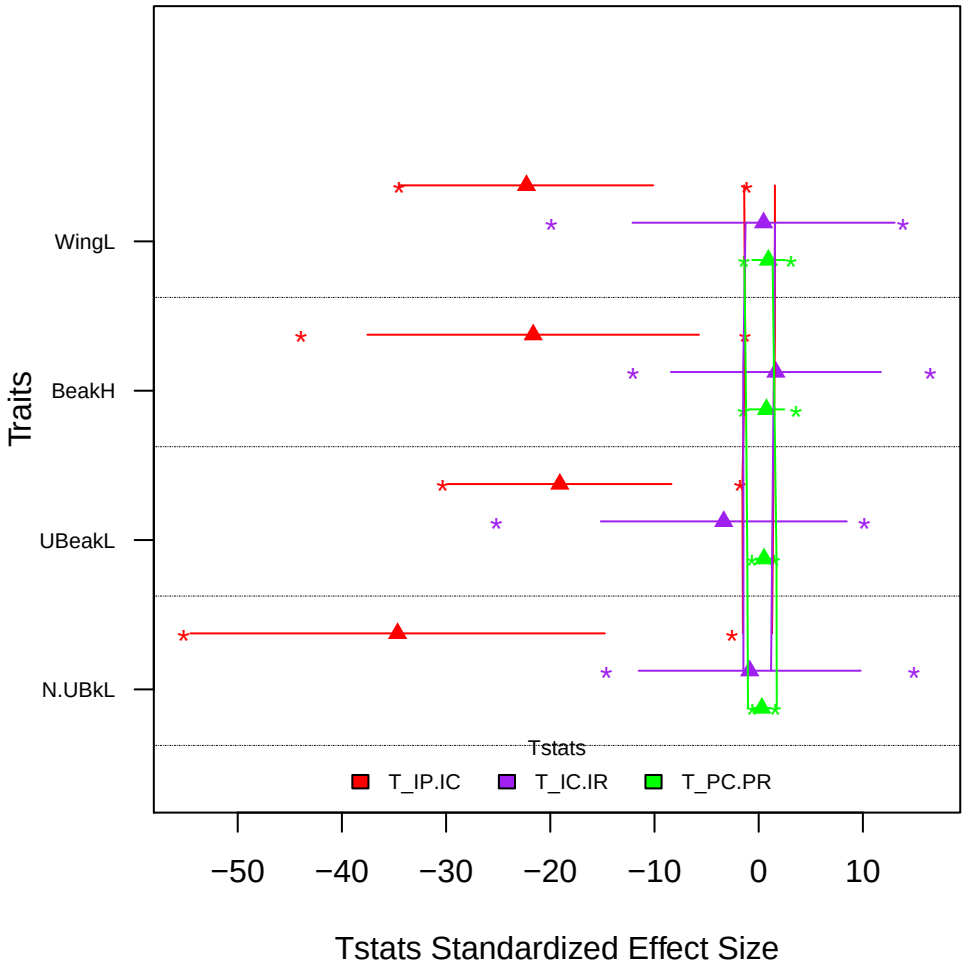
5 Test of community assembly

5.1 Ratio of variances: T-statistics

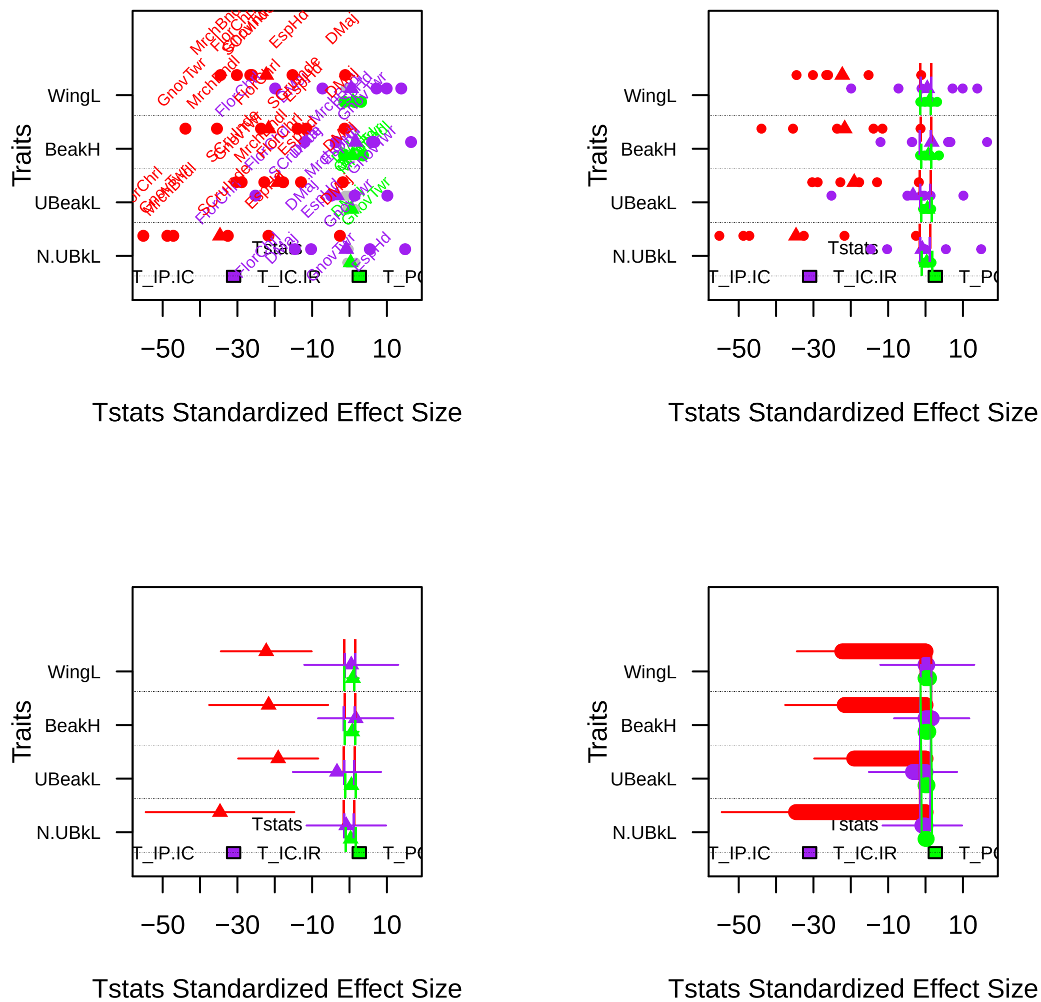
```
res.finch <- Tstats(traits.finch, ind_plot = ind.plot.finch, sp = sp.finch,
    nperm = 9, print = FALSE)
attributes(res.finch)

## $names
## [1] "T_IP.IC"    "T_IC.IR"    "T_PC.PR"    "variances"  "T_IP.IC_nm"
## [6] "T_IC.IR_nm" "T_PC.PR_nm" "pval"
##
## $class
## [1] "Tstats"
```

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

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

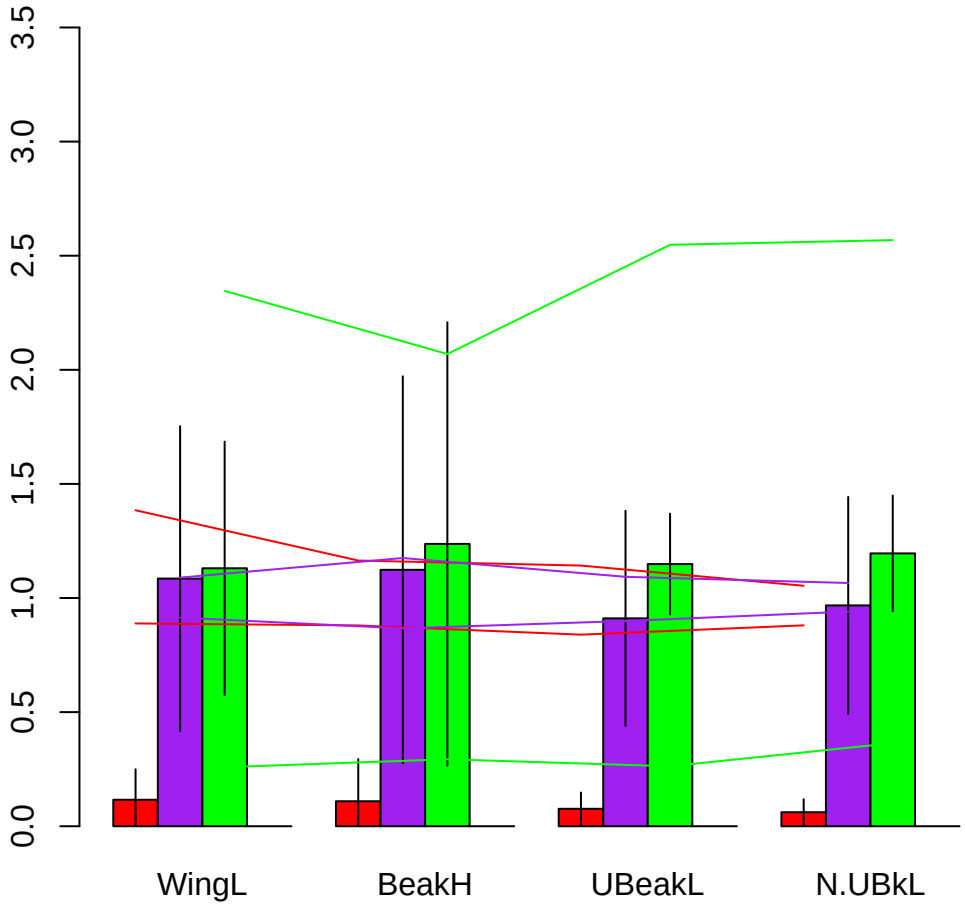


```r
par(mfrow = c(2, 2))
plot(res.finch, type = "color_cond")
plot(res.finch, type = "simple")
plot(res.finch, type = "simple_sd")
plot(res.finch, type = "barplot")
```




```r
par(oldpar)
```

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

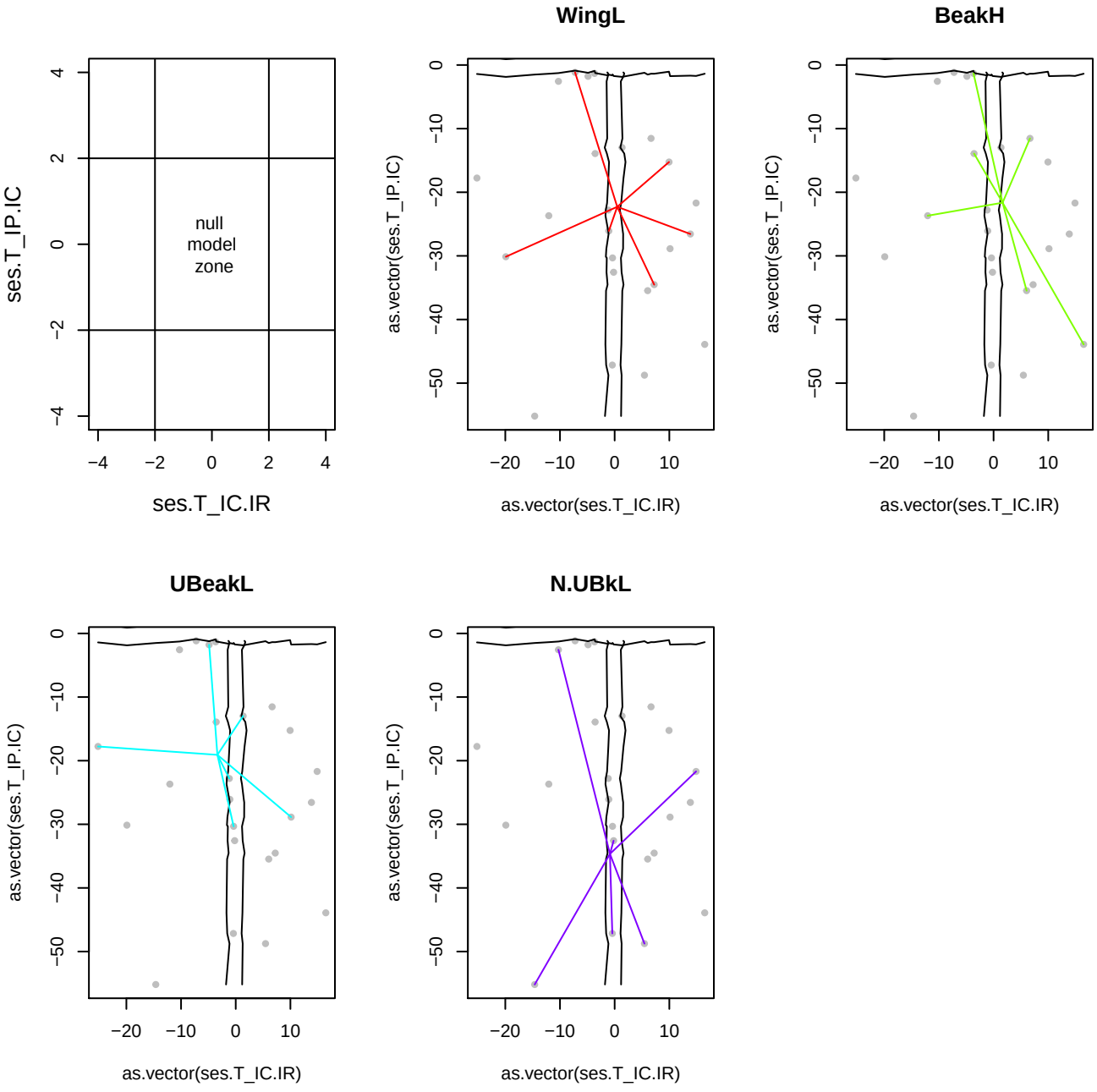



```r
attributes(summary(res.finch))

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

head(summary(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
```

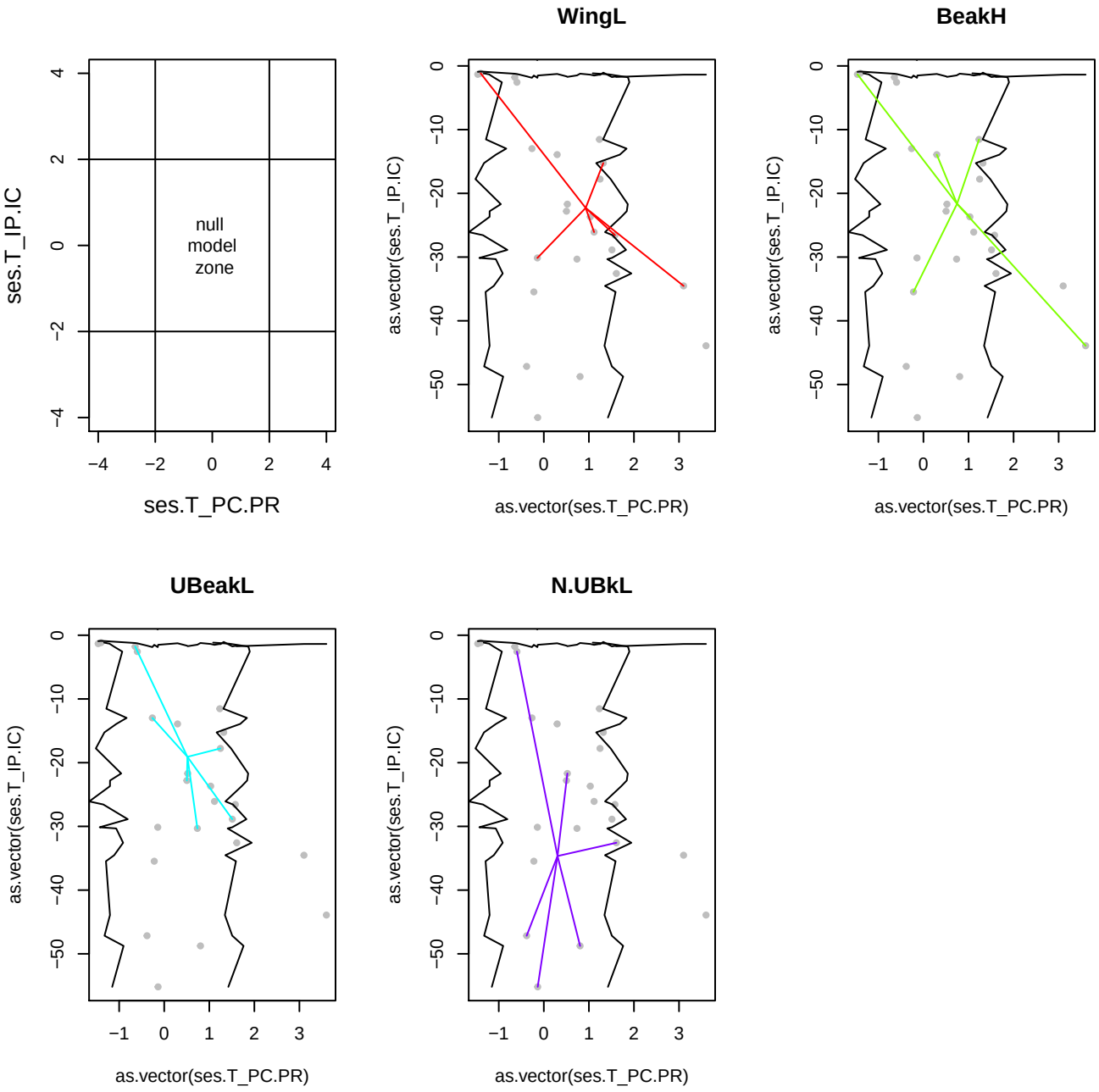
```
## T_IP.IC.sup   1.0   1.0    1.0    1.0
## T_IP.IC.sup   1.0   1.0    1.0    1.0
```

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

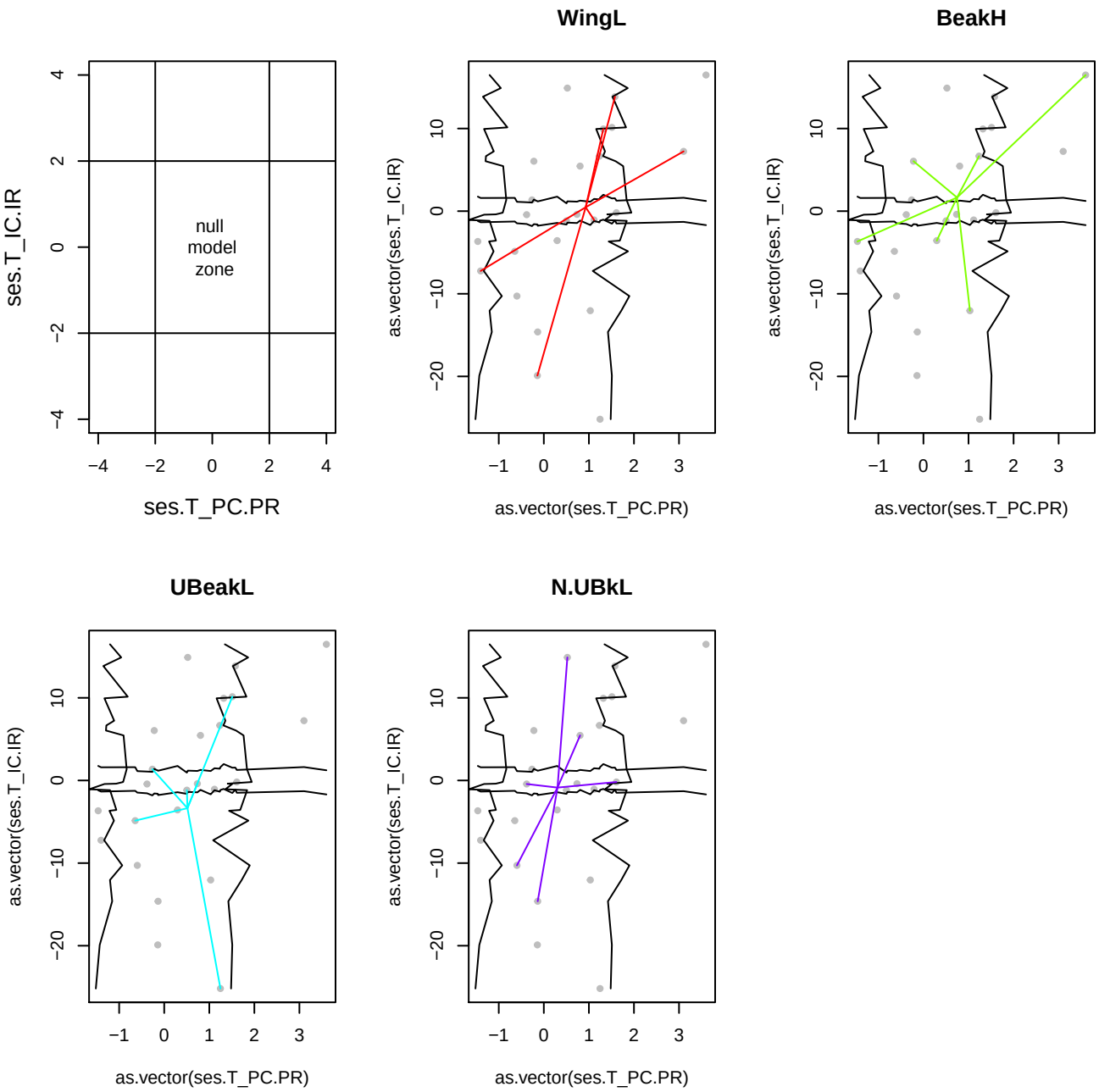
```
par(mfrow = c(2, 3))
plot_cor.Tstats(res.finch, plot.ask = FALSE, multipanel = F)

## Warning:  largeur de police inconnue pour le caractère 0xd
## Warning:  largeur de police inconnue pour le caractère 0xd
```




```
## Warning:  largeur de police inconnue pour le caractère 0xd
## Warning:  largeur de police inconnue pour le caractère 0xd
```

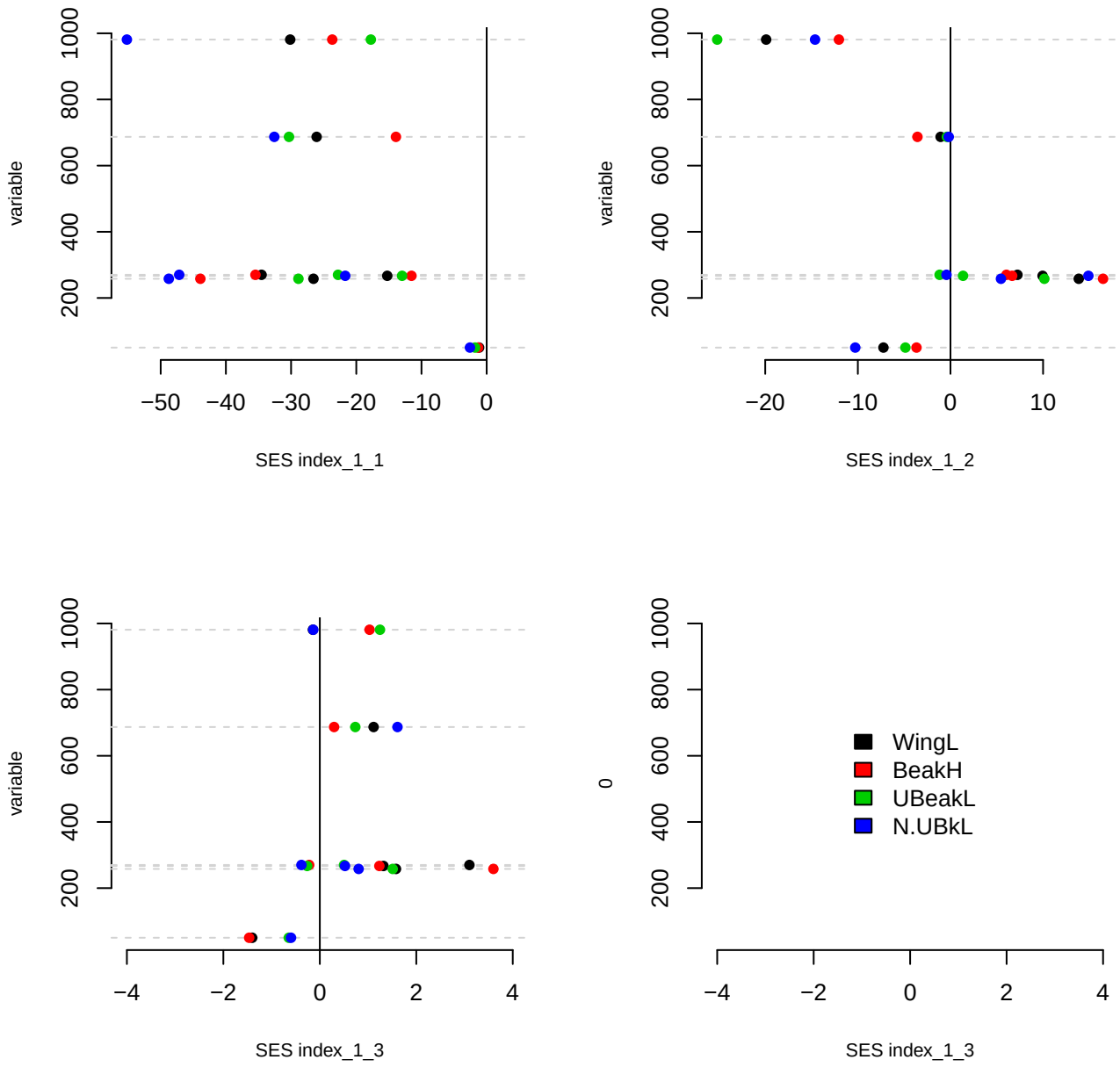



```
## Warning:  largeur de police inconnue pour le caractère 0xd
## Warning:  largeur de police inconnue pour le caractère 0xd
```



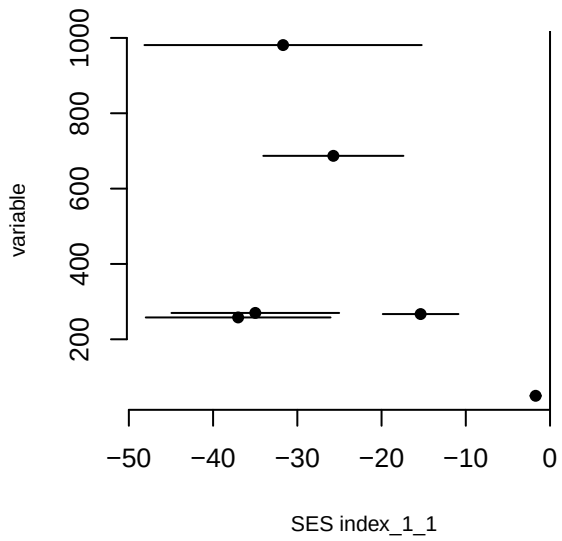
Here we plot T-statistics in function of species richness by sites.

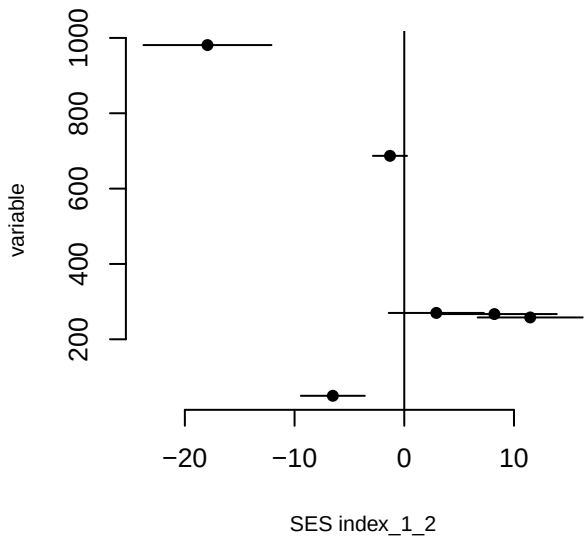
```
par(mfrow = c(2, 2))
species.richness <- table(ind.plot.finch)
plot_ses.var(as.listofindex(list(res.finch)), species.richness, multipanel = F)
```

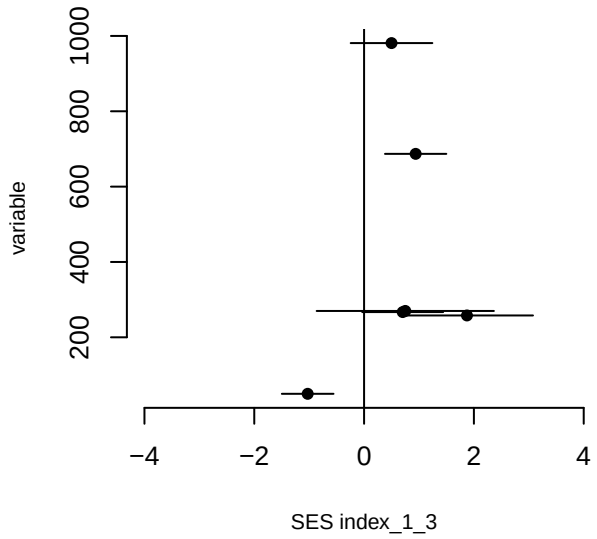


Same plot with (resume=TRUE).

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








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

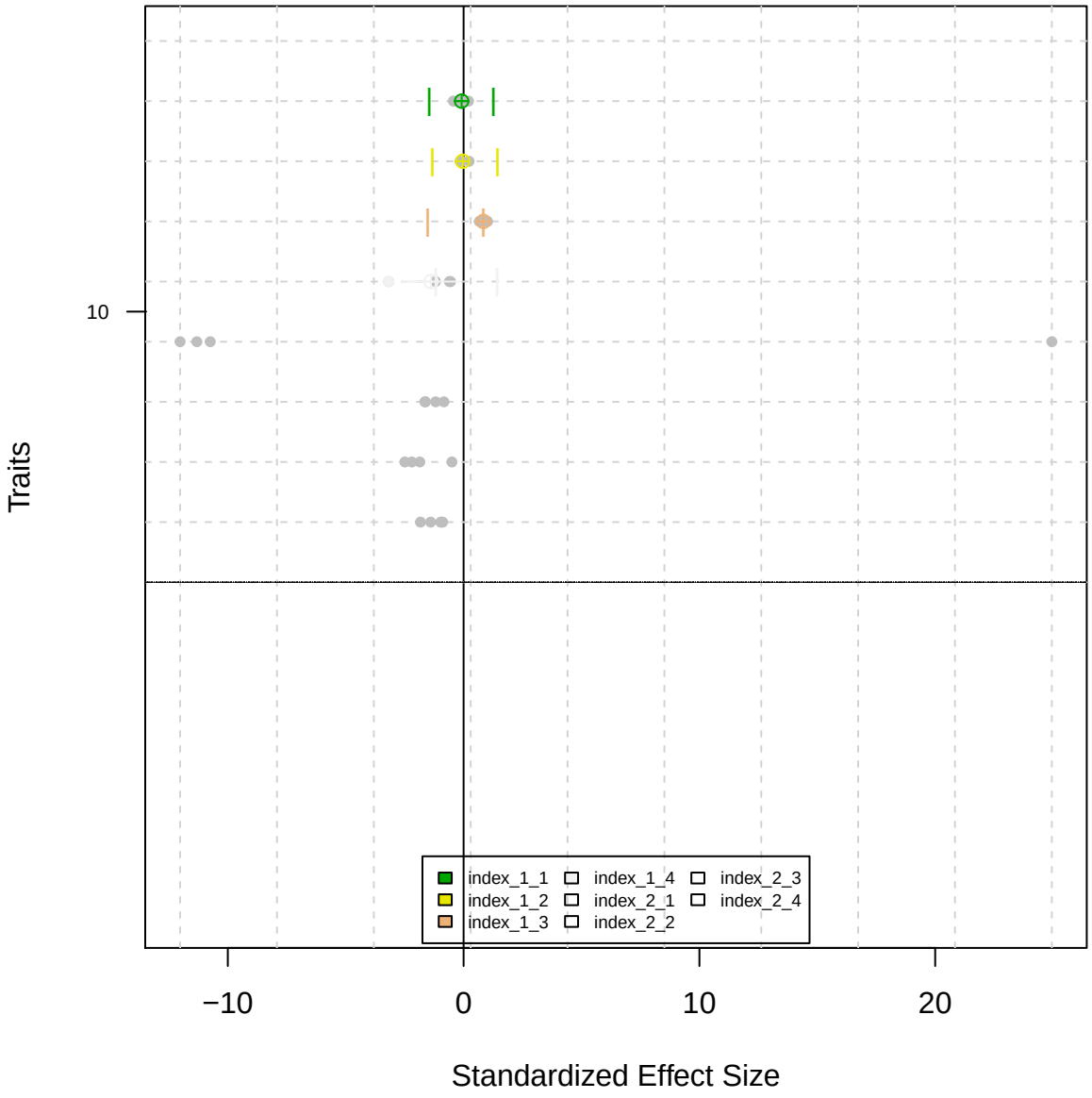
5.2 Others univariates index

The function (com.index) allow to choose your own function (like mean, range, variance...) to calculate customize index.

```
funct <- c("mean(x, na.rm=T)", "kurtosis(x, na.rm=T)", "max(x, na.rm=T) - min(x, na.rm=T)",
    "CVNND(x)")
res.finch.sp_mn2 <- com.index(traits = traits.finch, index = funct, sp = sp.finch,
    nullmodels = c(2, 2, 2, 2), ind.plot = ind.plot.finch, nperm = 9, print = FALSE)
res.finch.sp_mn3 <- com.index(traits = traits.finch, index = funct, sp = sp.finch,
    nullmodels = c(3, 3, 3, 3), ind.plot = ind.plot.finch, nperm = 9, print = FALSE)
```

We can represent Standardized Effect Size (ses) using the function (plot(as.listofindex(list1, list2, list3)))

```
list.ind2 <- list(res.finch.sp_mn2, res.finch.sp_mn3)
index.list2 <- as.listofindex(list.ind2)

plot(index.list2)
```



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

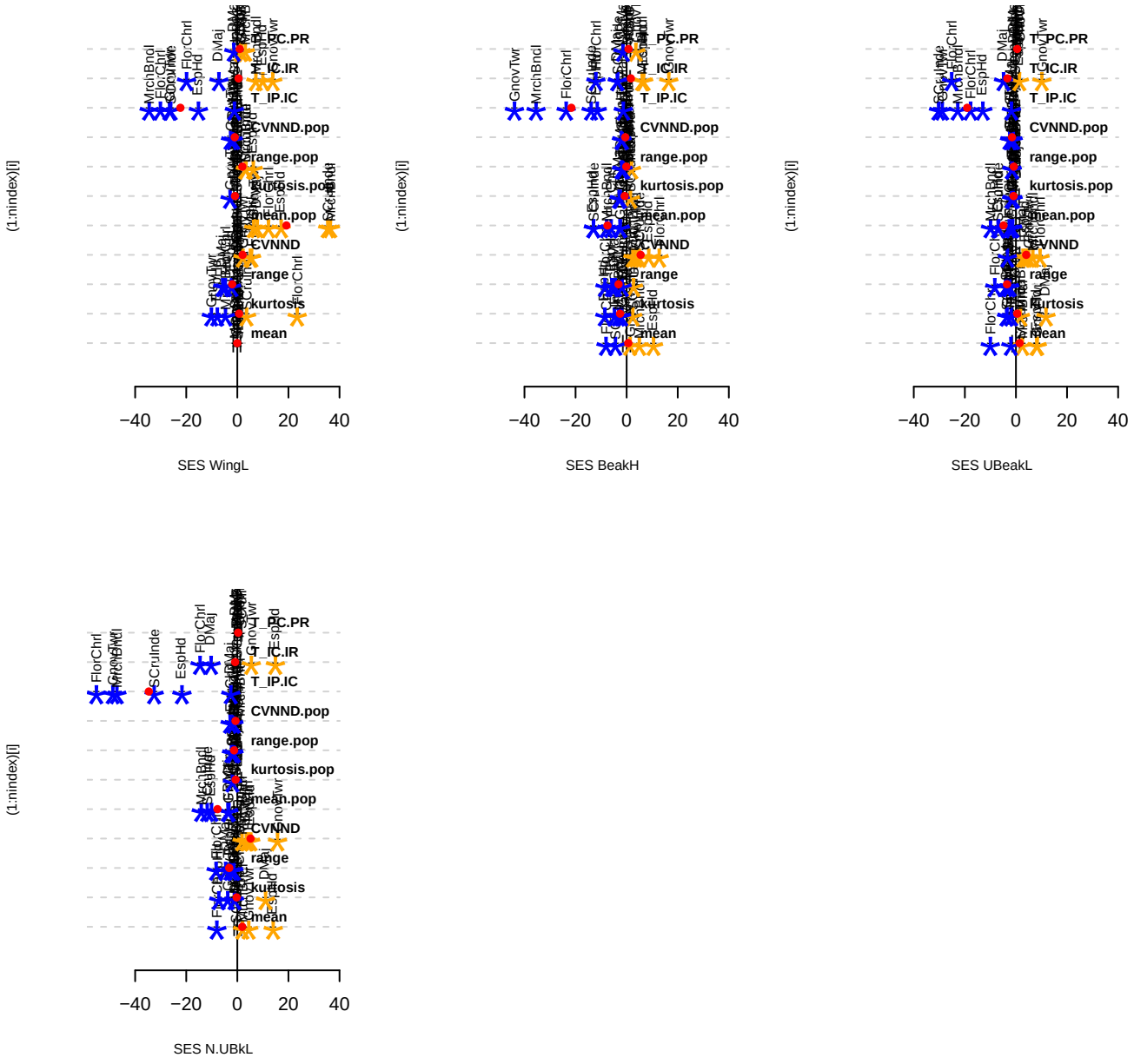
```
funct <- c("tapply(x, ind.plot.finch, function(x) mean(x, na.rm=T))", "tapply(x, ind.plot.fin
    "tapply(x, ind.plot.finch, function(x) max(x, na.rm=T)-min(x, na.rm=T))",
    "tapply(x, ind.plot.finch, function(x) CVNND(x))")

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

res.finch.ind_mn1 <- com.index(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 <- com.index(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.

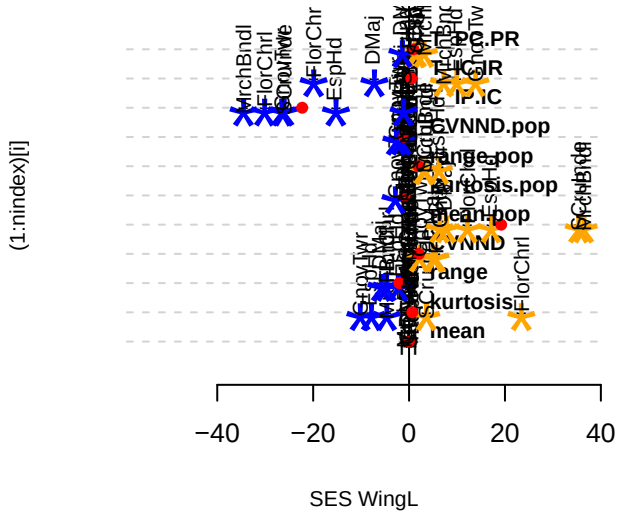
```
# Calcul of means by population (name_sp_site is a name of a population)
# like in the function com.index and determine the site for each population
# (sites_bypop)

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, functi
    "tapply(x, fact, function(x) max(x, na.rm=T)-min(x, na.rm=T))", "tapply(x, fact, function


res.finch.withIV <- com.index(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 <- com.index(traits = traits.finch, index = funct.withoutIV,
    sp = sp.finch, nullmodels = c(3, 3, 3, 3), ind.plot = ind.plot.finch, nperm = 9,
    print = FALSE)
```

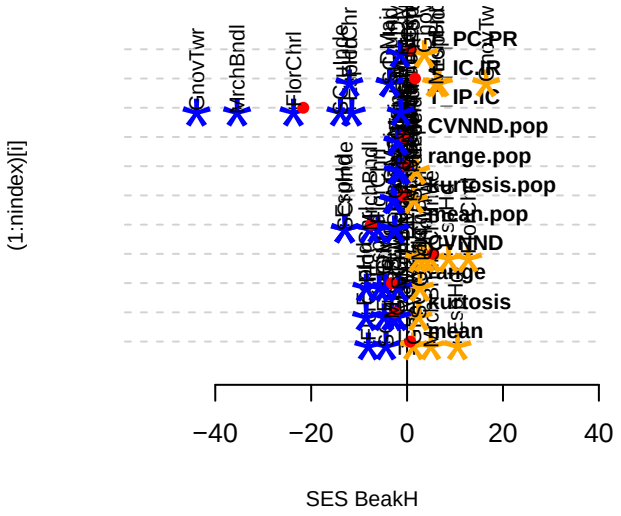
We can also represent T-statistics and custom index thanks to the (plot.listofindex) function.

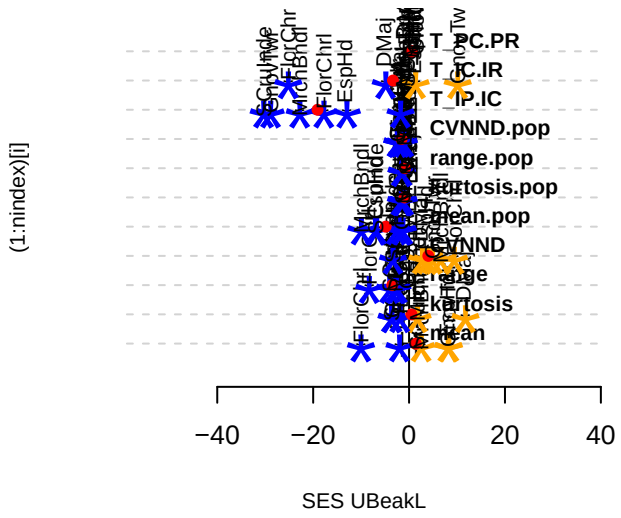
```
list.ind <- list(res.finch.withIV, res.finch.withoutIV, res.finch)
namesindex.i.l1 = c("mean", "kurtosis", "range", "CVNND", "mean.pop", "kurtosis.pop",
    "range.pop", "CVNND.pop", "T_IP.IC", "T_IC.IR", "T_PC.PR")

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

class(i.l1)

## [1] "listofindex"

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

## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
## Warning:  "bysites" is not a graphical parameter
```

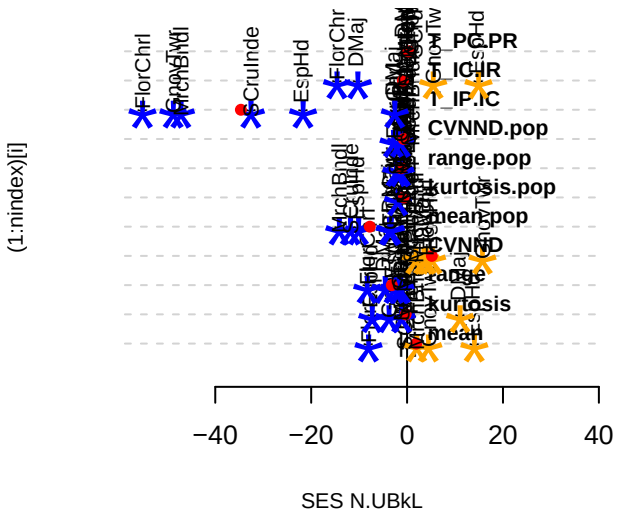



```
par(mfrow = c(2, 2))
plot(i.11, type = "bytraits")
```



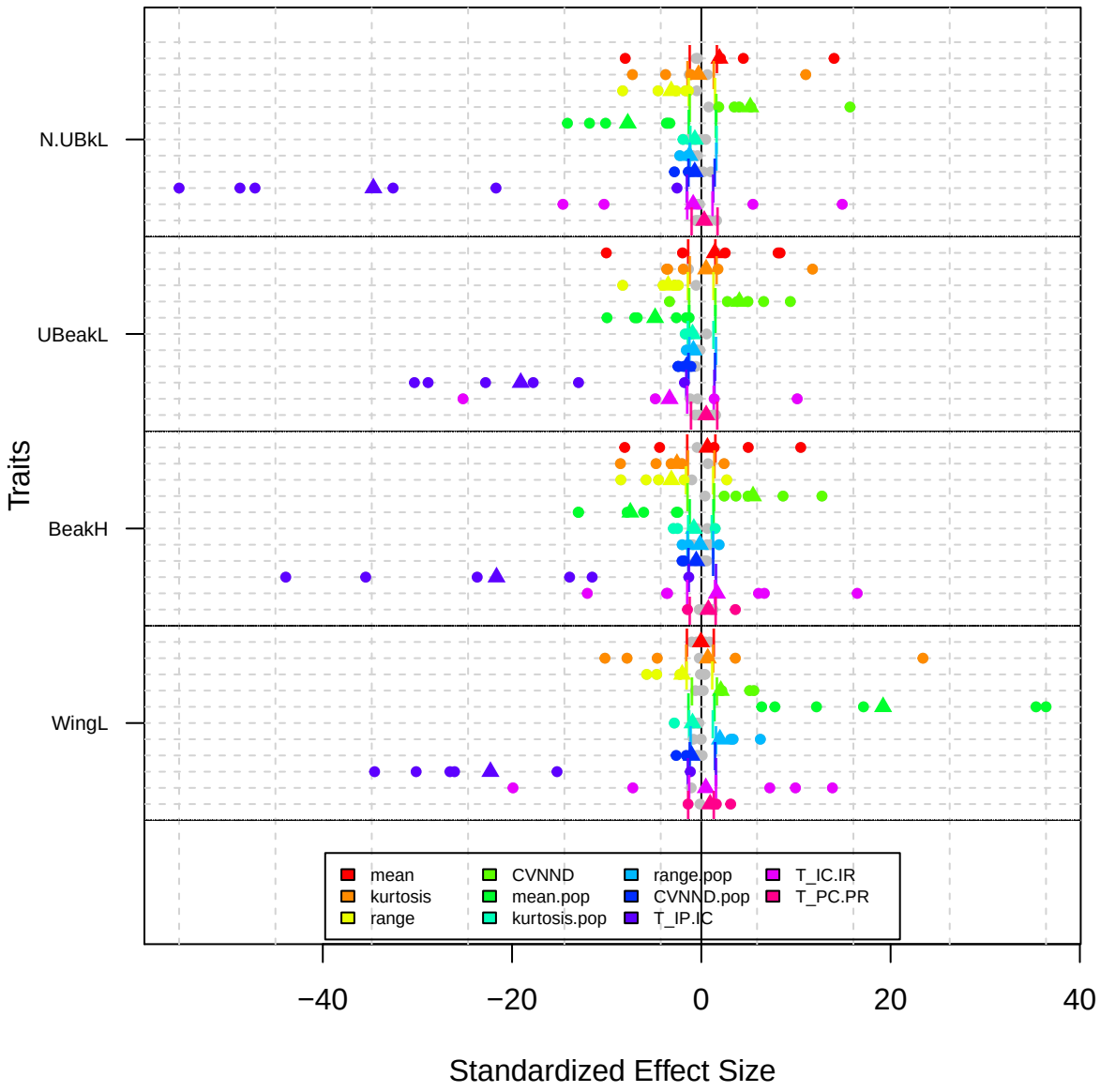




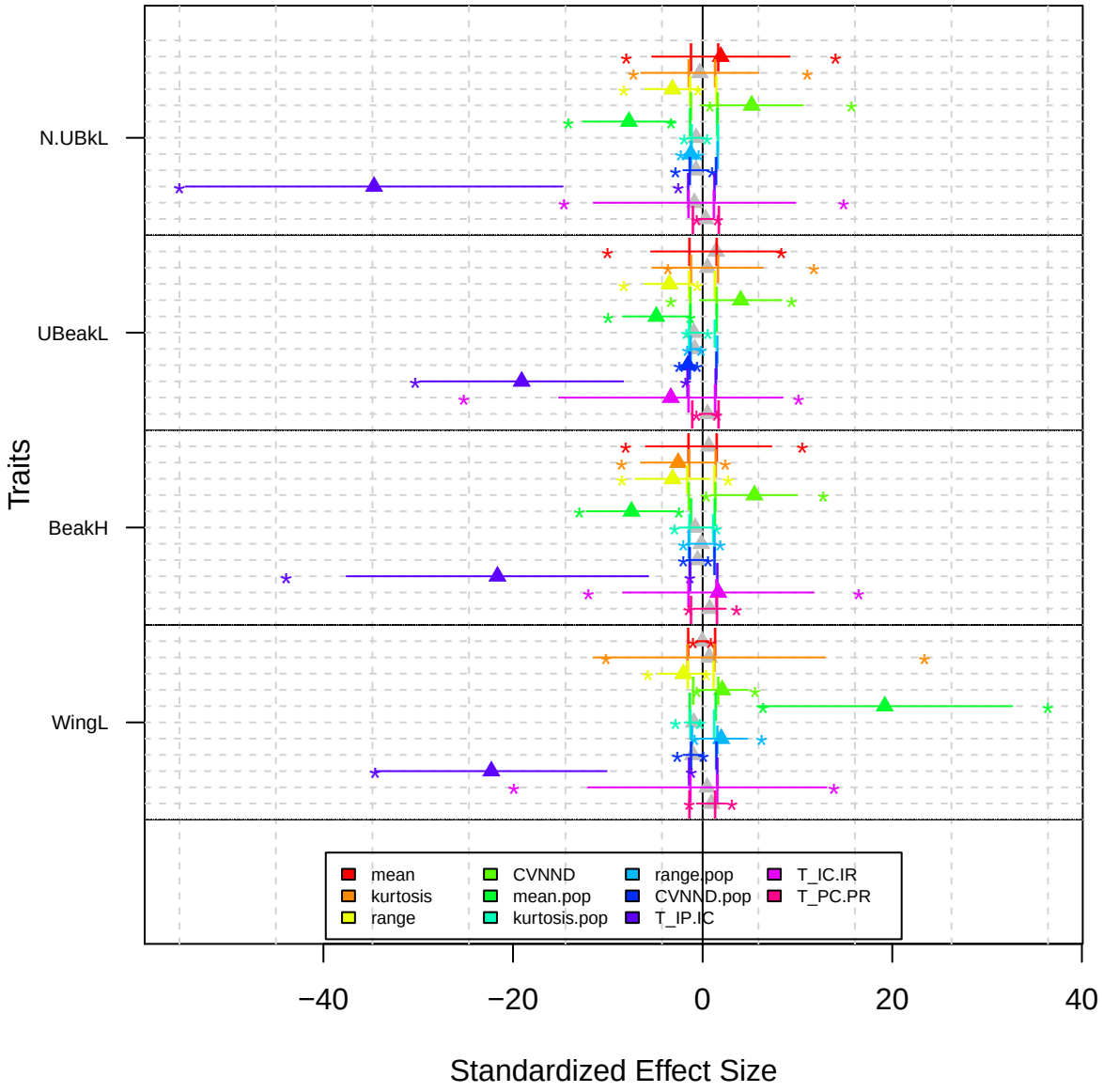



```
par(mfrow = c(1, 1))
```

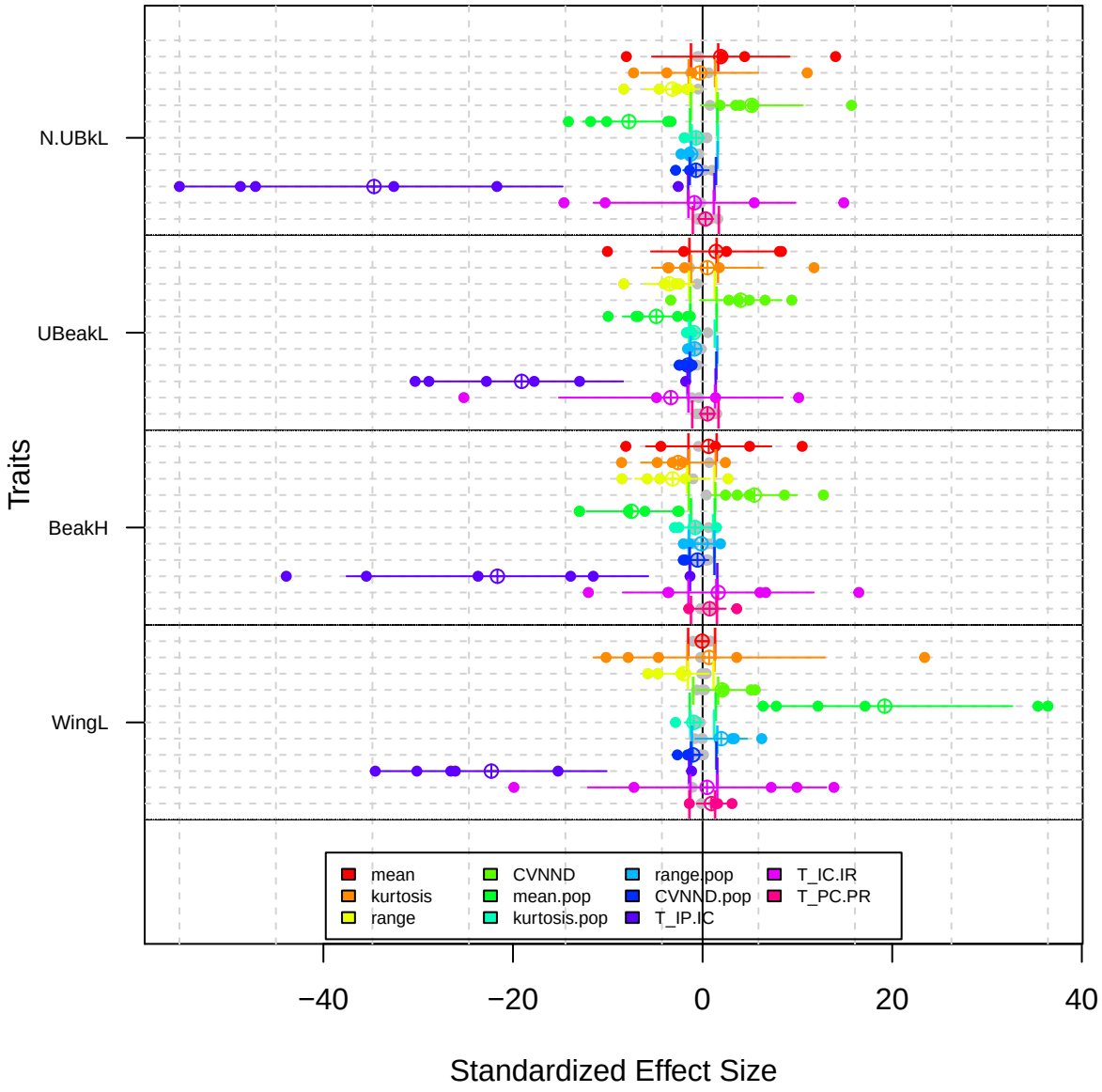
```
plot(i.l1, type = "simple")
```



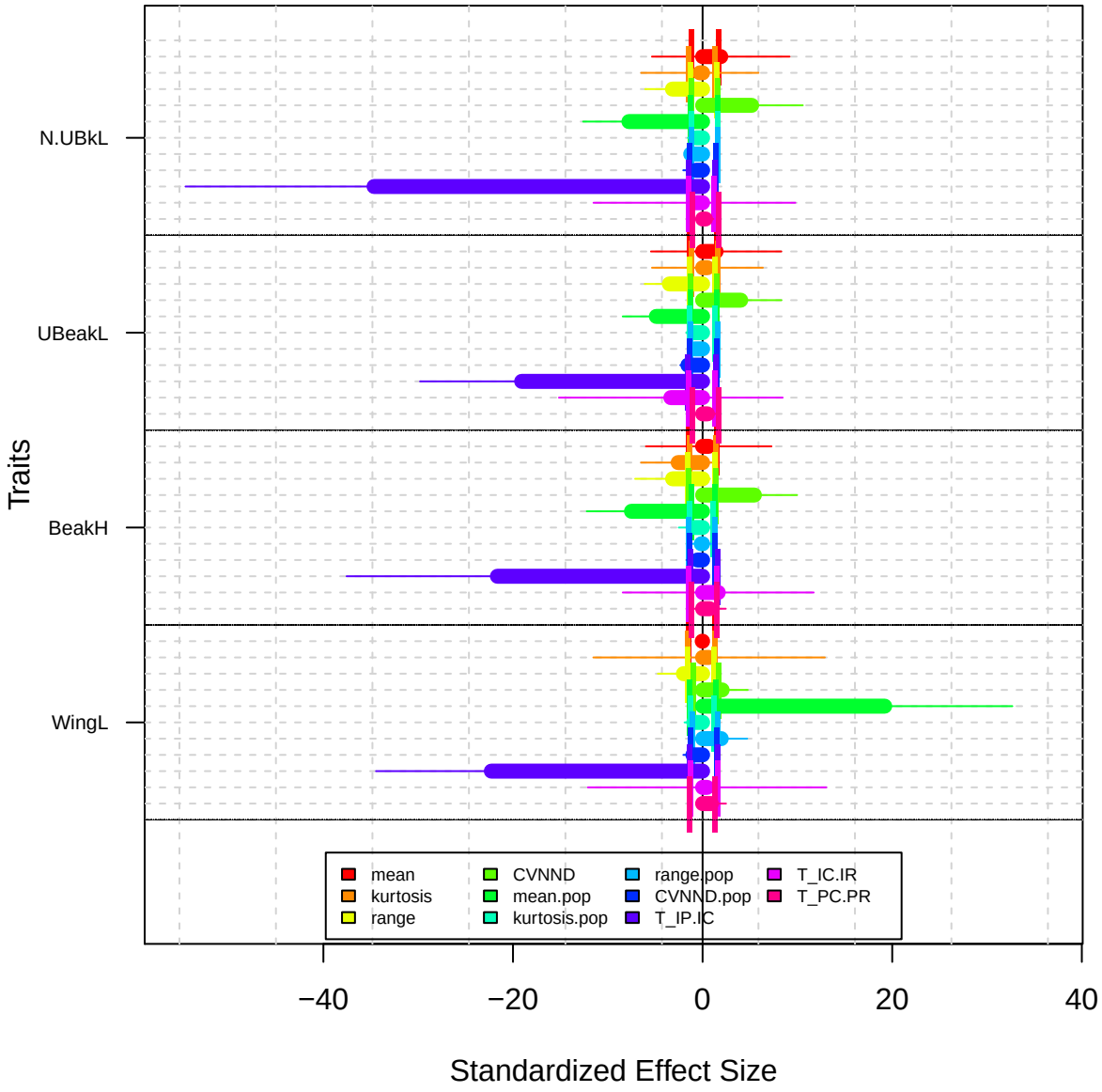

```
plot(i.l1, type = "simple_range")
```




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




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


N.UBkL
UBeakL
BeakH
WingL
Traits
mean
kurtosis
range
CVNND
mean.pop
kurtosis.pop
range.pop
CVNND.pop
T_IP.IC
T_IC.IR
T_PC.PR
−40
−20
0
20
40
Standardized Effect Size

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)
```

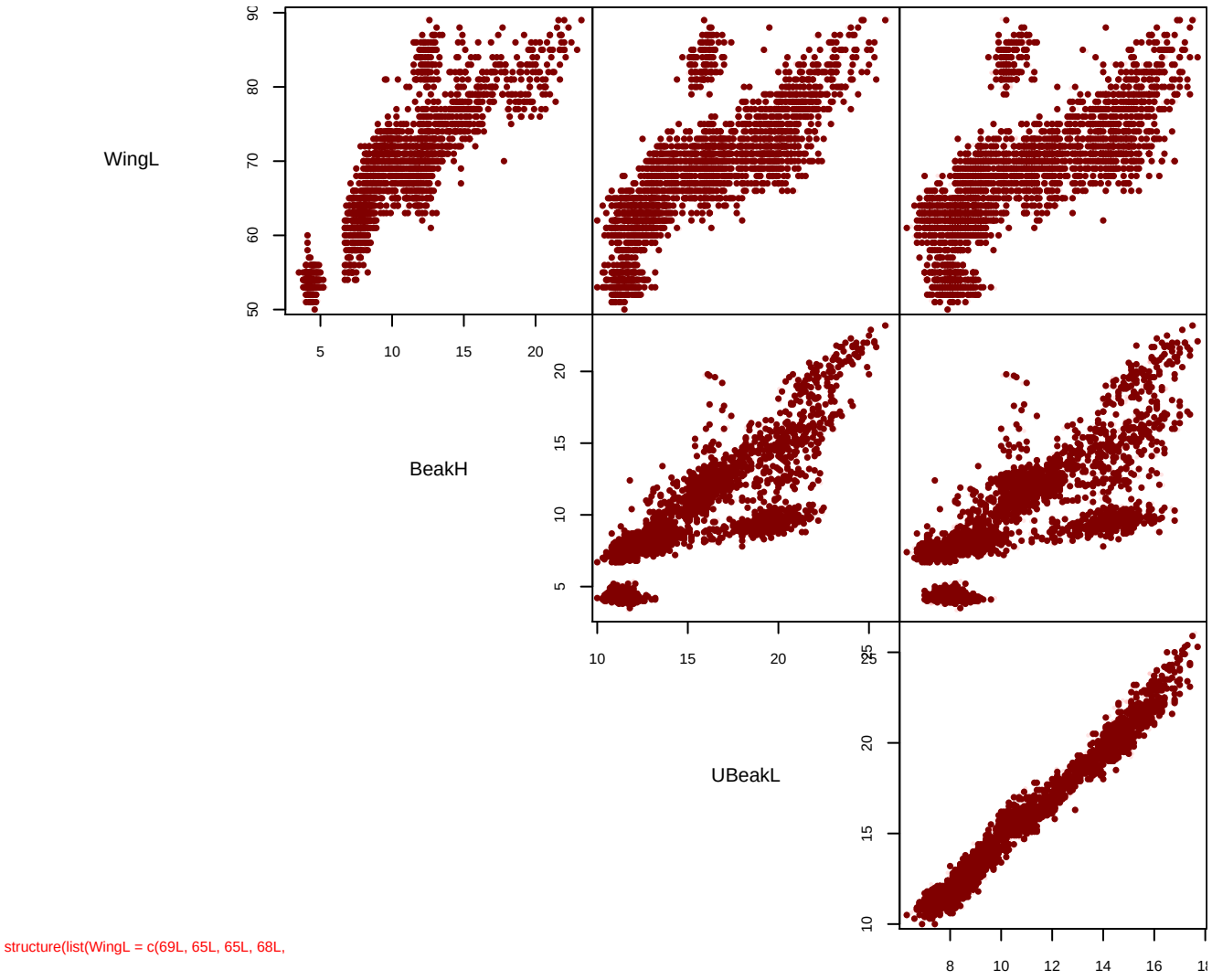
A simple example to illustrate the concept of the function (com.index.multi)

```
n_sp_plot <- as.factor(paste(sp.finch, ind.plot.finch, sep = "_"))
res.sum.1 <- com.index.multi(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.finc
    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] "calcul of null values using null models"
## [1] "sum(scale(x), na.rm=T) 50 %"
## [1] "sum(x, na.rm=T) 100 %"
## [1] "calcul 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.

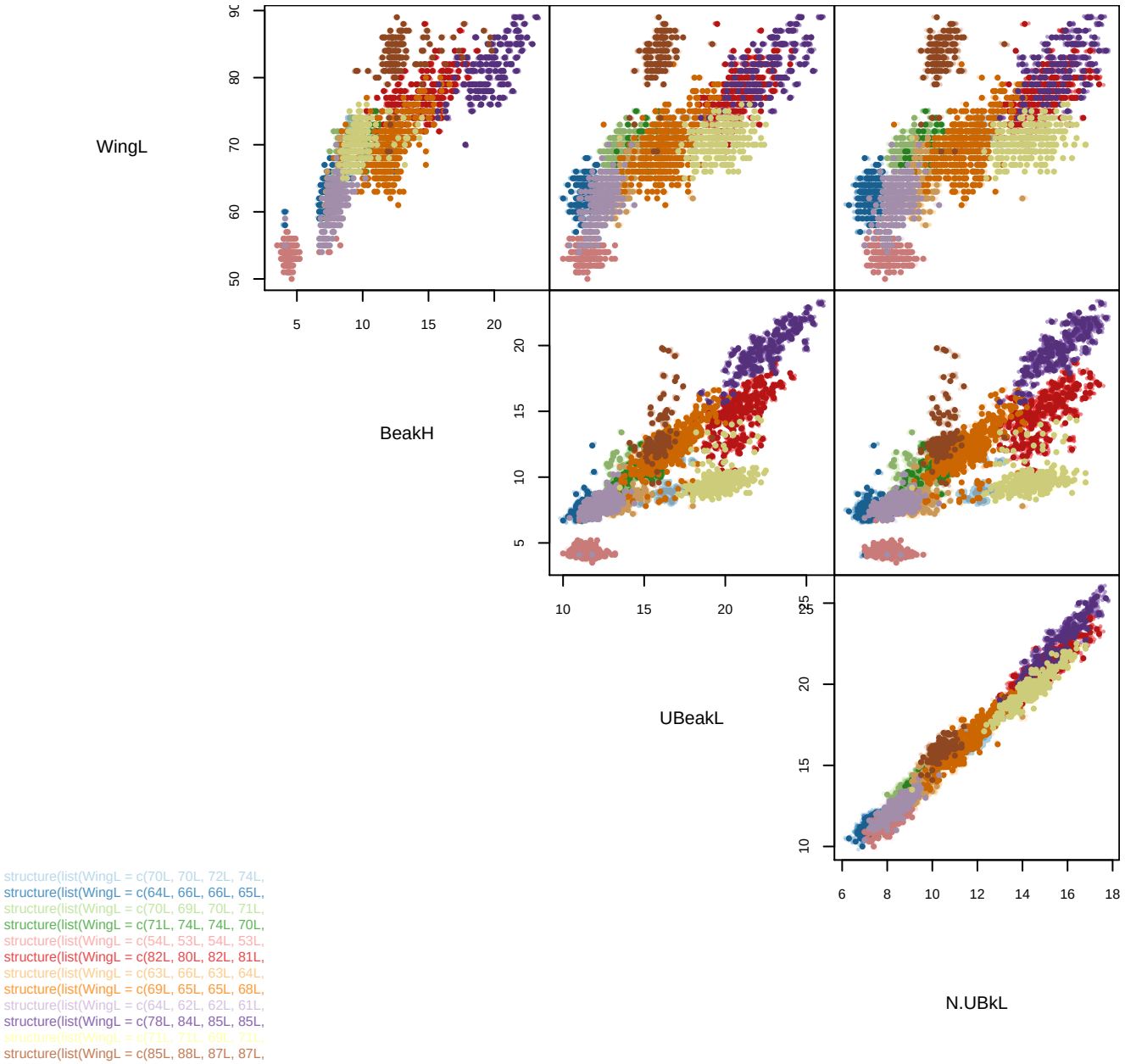
```
hv <- hypervolume(traits.finch.mice, reps = 100, bandwidth = 0.2, verbose = F,
    warnings = F)
plot(hv)
```



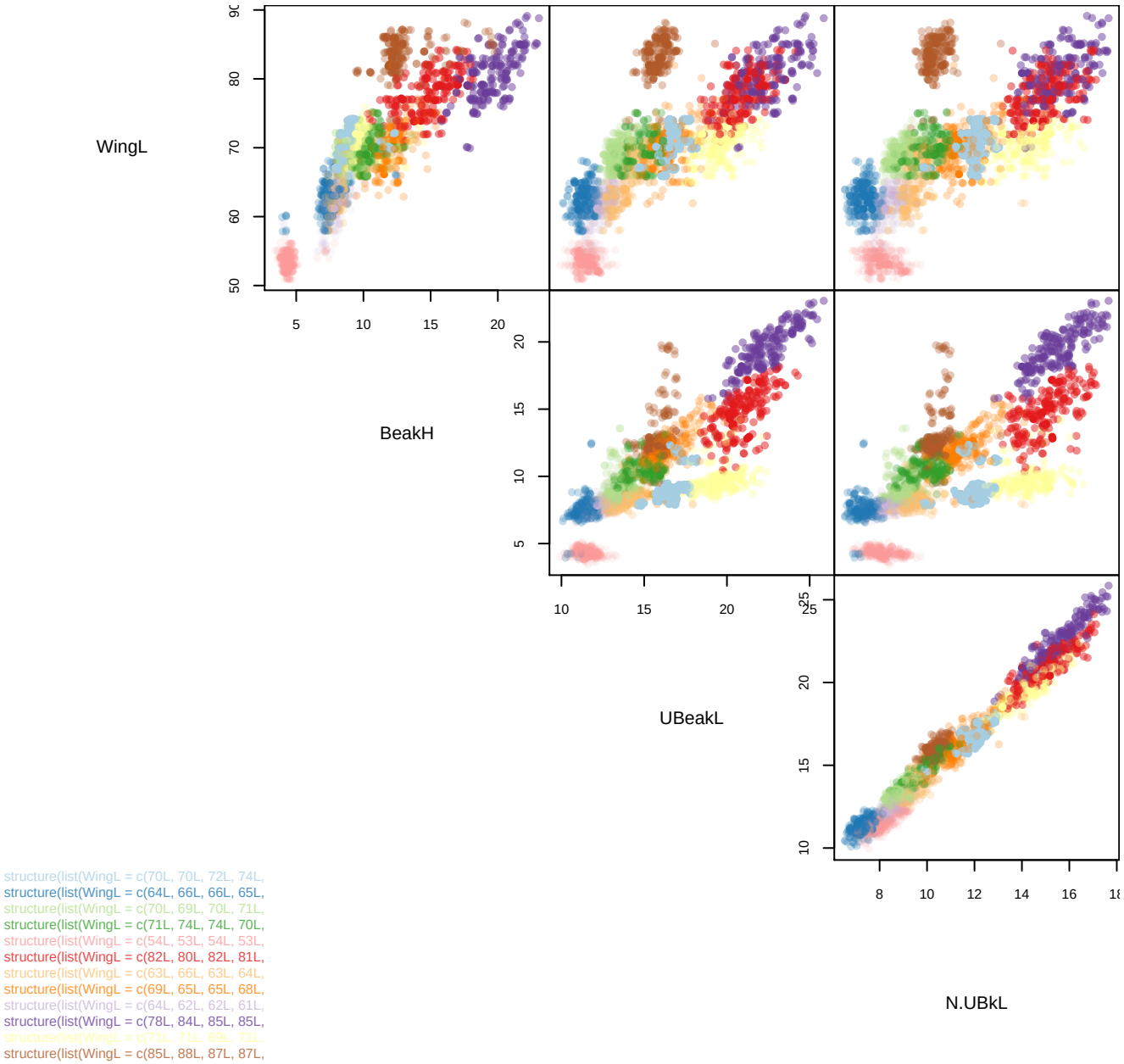
Now, we can do the same analysis for each species.

```
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 1.4-1 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)

## Hypervolume
##  Name: structure(list(WingL = c(70L, 70L, 72L, 74L, 74L, 72L, 73L, 73L,
##  Nr. of observations: 23
##  Dimensionality: 4
##  Volume: 0.588800
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 14617
## Hypervolume
```

```
##  Name: structure(list(WingL = c(64L, 66L, 66L, 65L, 65L, 63L, 63L, 62L,
##  Nr. of observations: 172
##  Dimensionality: 4
##  Volume: 3.512137
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 91290
## Hypervolume
##  Name: structure(list(WingL = c(70L, 69L, 70L, 71L, 72L, 69L, 71L, 68L,
##  Nr. of observations: 142
##  Dimensionality: 4
##  Volume: 3.062729
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 78616
## Hypervolume
##  Name: structure(list(WingL = c(71L, 74L, 74L, 70L, 68L, 70L, 70L, 66L,
##  Nr. of observations: 73
##  Dimensionality: 4
##  Volume: 1.727309
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 43449
## Hypervolume
##  Name: structure(list(WingL = c(54L, 53L, 54L, 53L, 54L, 53L, 53L, 53L,
##  Nr. of observations: 299
##  Dimensionality: 4
##  Volume: 4.132790
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 114703
## Hypervolume
##  Name: structure(list(WingL = c(82L, 80L, 82L, 81L, 82L, 80L, 84L, 83L,
##  Nr. of observations: 206
##  Dimensionality: 4
##  Volume: 5.100971
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
```

```
##  Number of random points: 127193
## Hypervolume
##  Name: structure(list(WingL = c(63L, 66L, 63L, 64L, 63L, 61L, 62L, 65L,
##  Nr. of observations: 125
##  Dimensionality: 4
##  Volume: 2.758810
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 70621
## Hypervolume
##  Name: structure(list(WingL = c(69L, 65L, 65L, 68L, 66L, 68L, 67L, 69L,
##  Nr. of observations: 548
##  Dimensionality: 4
##  Volume: 11.764025
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 302230
## Hypervolume
##  Name: structure(list(WingL = c(64L, 62L, 62L, 61L, 67L, 64L, 64L, 63L,
##  Nr. of observations: 386
##  Dimensionality: 4
##  Volume: 6.372158
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 171917
## Hypervolume
##  Name: structure(list(WingL = c(78L, 84L, 85L, 85L, 87L, 89L, 87L, 86L,
##  Nr. of observations: 126
##  Dimensionality: 4
##  Volume: 3.158656
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 78485
## Hypervolume
##  Name: structure(list(WingL = c(71L, 71L, 69L, 71L, 73L, 71L, 71L, 71L,
##  Nr. of observations: 284
##  Dimensionality: 4
##  Volume: 6.384687
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Quantile desired: 0.000000
```

```
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 162478
## Hypervolume
##  Name: structure(list(WingL = c(85L, 88L, 87L, 87L, 85L, 74L, 86L, 85L,
##  Nr. of observations: 129
##  Dimensionality: 4
##  Volume: 2.897190
##  Bandwidth: 0.2 0.2 0.2 0.2
##  Quantile desired: 0.000000
##  Quantile obtained: 0.000000
##  Nr. of repetitions per point: 1000
##  Number of random points: 73579
```

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.
```

```
hv.1 <- com.index.multi(traits.finch.mice, index = c("as.numeric(try(hypervolume(na.omit(x),
    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] "calcul of null values using null models"
## [1] "as.numeric(try(hypervolume(na.omit(x), reps=100, \n
## [1] "calcul of observed values"
## [1] "100 %"
```

```
hv.2 <- com.index.multi(traits.finch.mice, index = c("as.numeric(try(hypervolume(na.omit(x),
    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] "calcul of null values using null models"
## [1] "as.numeric(try(hypervolume(na.omit(x), reps=100, \n
## [1] "calcul of observed values"
## [1] "100 %"
```

```
hv.3 <- com.index.multi(traits.finch.mice, index = c("as.numeric(try(hypervolume(na.omit(x),
    by.factor = ind.plot.finch, nullmodels = c(2, 2), ind.plot = ind.plot.finch,
    nperm = 9, sp = sp.finch)
```

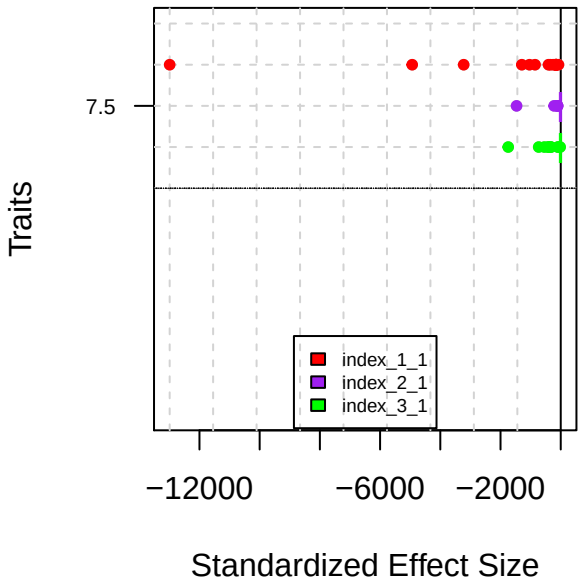
```
## [1] "creating null models"
## [1] "nm.2 25 %"
## [1] "nm.2 50 %"
## [1] "nm.2 75 %"
## [1] "nm.2 100 %"
## [1] "calcul of null values using null models"
## [1] "as.numeric(try(hypervolume(na.omit(x), reps=100,\n                                    h
## [1] "calcul of observed values"
## [1] "100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

```
hv.4 <- com.index.multi(traits.finch.mice, index = c("as.numeric(try(hypervolume(na.omit(x),
    by.factor = sp.finch, nullmodels = c(2, 2), ind.plot = ind.plot.finch, nperm = 9,
    sp = sp.finch)
```

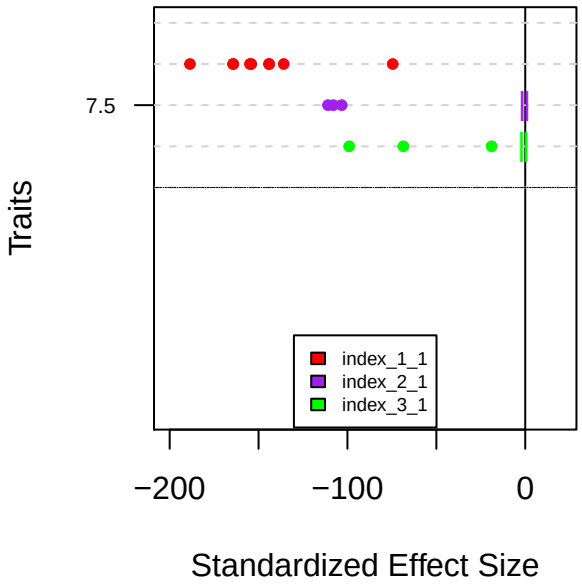
```
## [1] "creating null models"
## [1] "nm.2 25 %"
## [1] "nm.2 50 %"
## [1] "nm.2 75 %"
## [1] "nm.2 100 %"
## [1] "calcul of null values using null models"
## [1] "as.numeric(try(hypervolume(na.omit(x), reps=100, \n
## [1] "calcul of observed values"
## [1] "100 %"
```

```
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))
```



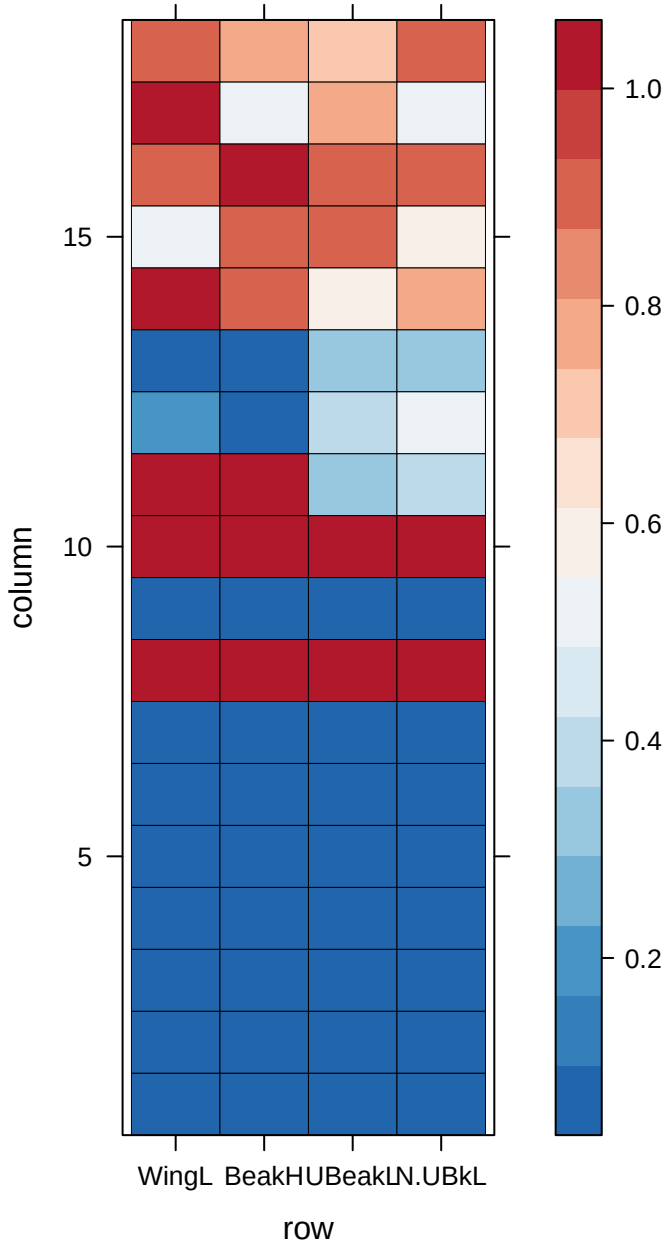
6 Others graphics functions

Using rasterVis to obtain more color schemes.

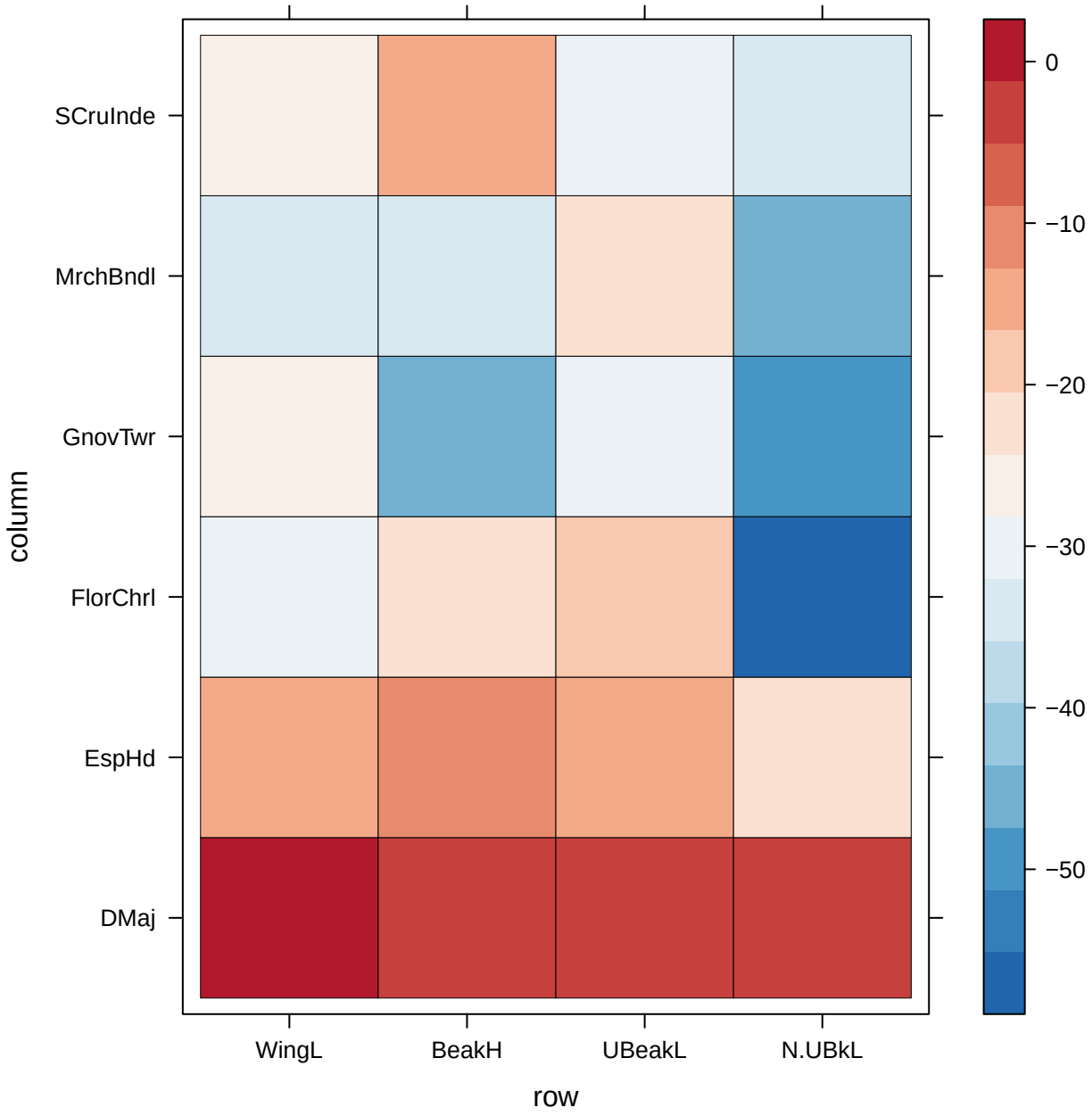
```
require(rasterVis)
# Custom theme (from rasterVis package)
my.theme <- BuRdTheme()
# Customize the colorkey
my.ckey <- list(col = my.theme$regions$col)
```

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

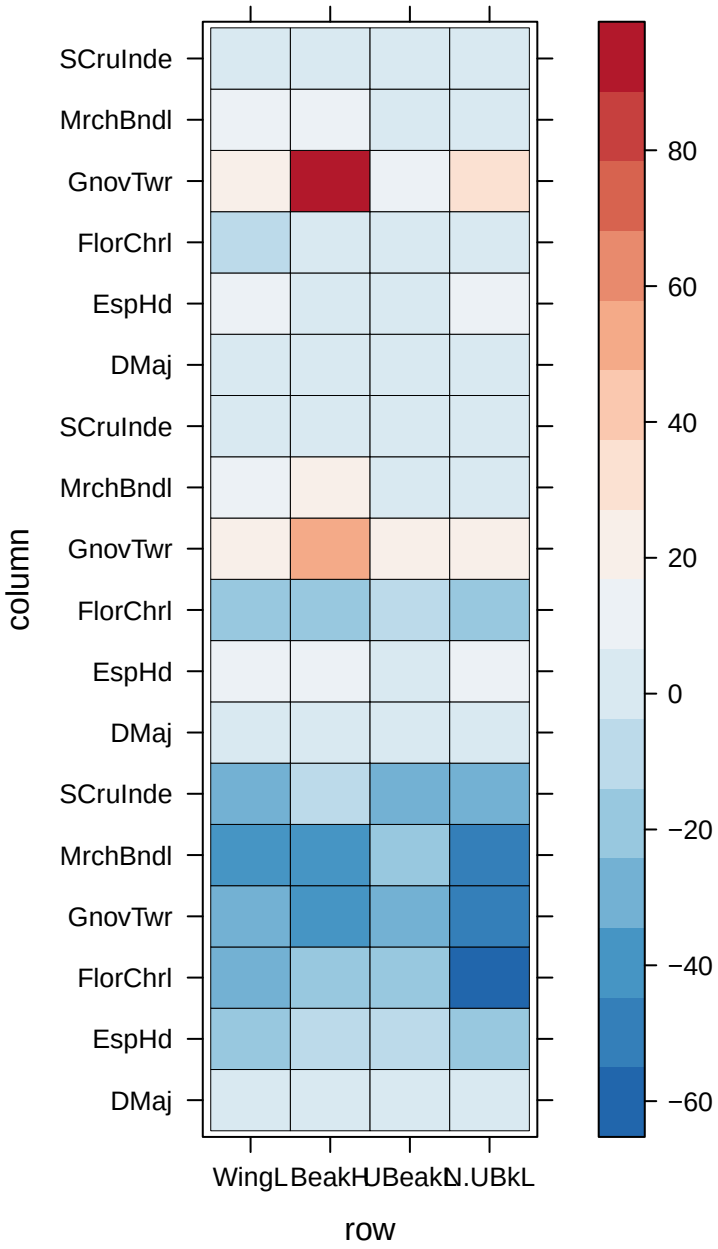
```
levelplot(t(rbind(res.finch$pval$T_IP.IC.inf, res.finch$pval$T_IC.IR.inf, res.finch$pval$T_PC
    colorkey = my.ckey, par.settings = my.theme, border = "black")
```



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

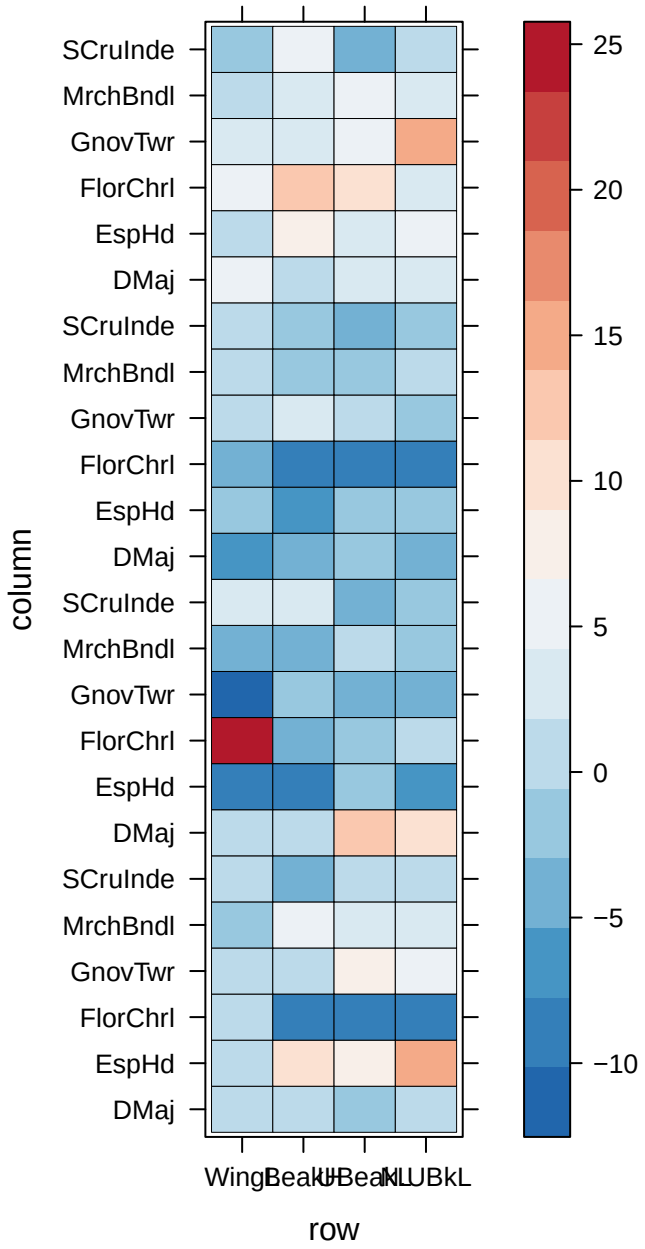



```
levelplot(cbind(t(ses(res.finch$T_IP.IC, res.finch$T_IP.IC_nm)$ses), t(ses(res.finch$T_IC.IR,
    res.finch$T_IP.IC_nm)$ses), t(ses(res.finch$T_PC.PR, res.finch$T_IP.IC_nm)$ses)),
    colorkey = my.ckey, par.settings = my.theme, border = "black")
```

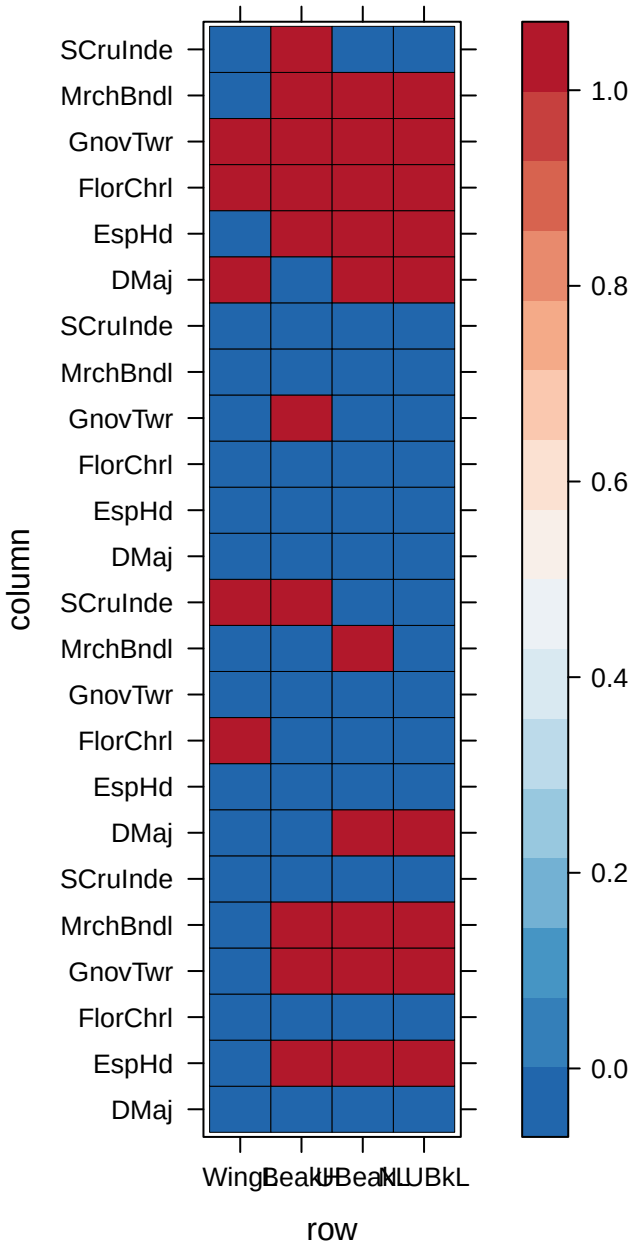


An other example using (ses.listofindex). The first plot represent "ses" values and the second one the result of a test with H0: observed index value are greater than what we can expect using the null model (alpha=2.5%).

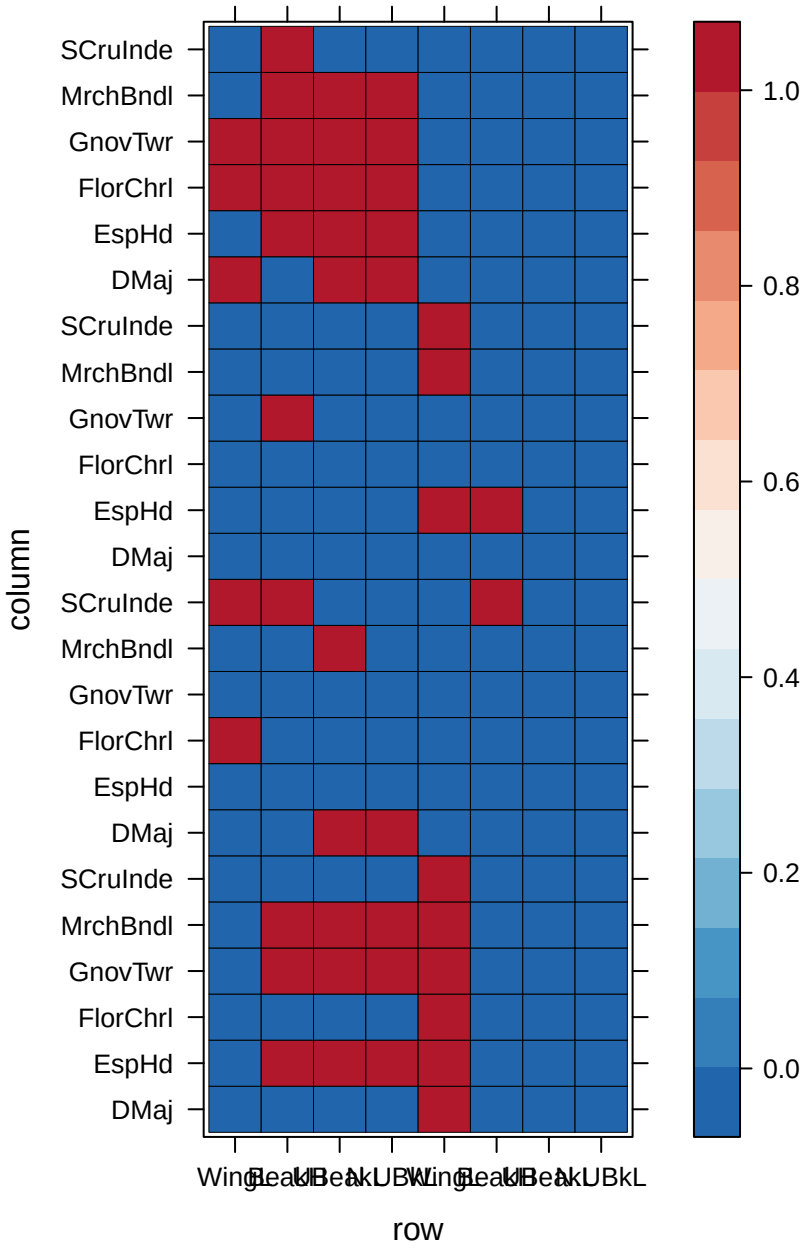
```
ses.list <- ses.listofindex(i.11)

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")
```



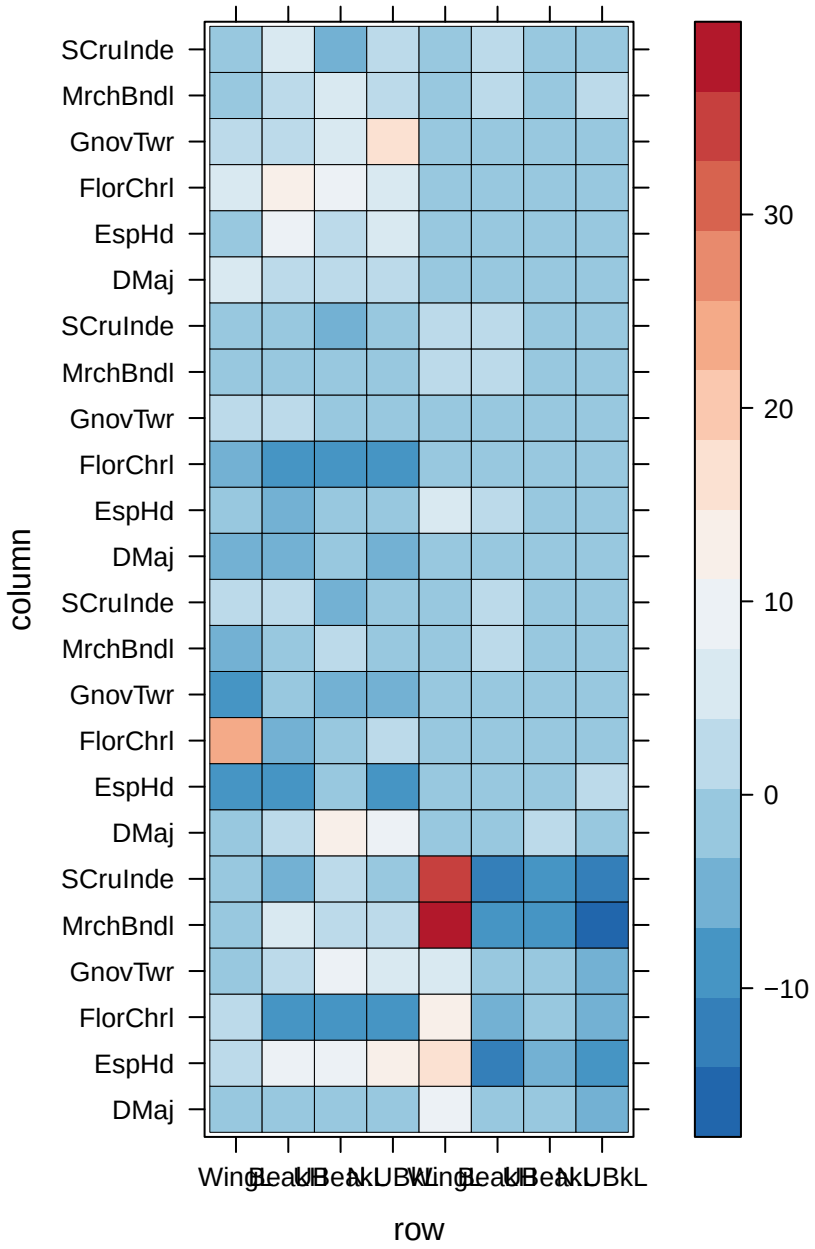

```
levelplot(t(rbind(ses.list[[1]]$ses > ses.list[[1]]$ses.sup, ses.list[[2]]$ses >
    ses.list[[2]]$ses.sup, ses.list[[3]]$ses > ses.list[[3]]$ses.sup, ses.list[[4]]$ses >
    ses.list[[4]]$ses.sup)), colorkey = my.ckey, par.settings = my.theme, border = "black")
```



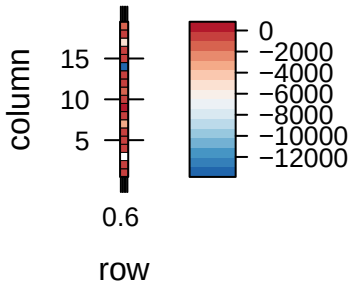

```
obj1 <- t(rbind(ses.list[[1]]$ses > ses.list[[1]]$ses.sup, ses.list[[2]]$ses >
    ses.list[[2]]$ses.sup, ses.list[[3]]$ses > ses.list[[3]]$ses.sup, ses.list[[4]]$ses >
    ses.list[[4]]$ses.sup))

obj2 <- t(rbind(ses.list[[5]]$ses > ses.list[[5]]$ses.sup, ses.list[[6]]$ses >
    ses.list[[6]]$ses.sup, ses.list[[7]]$ses > ses.list[[7]]$ses.sup, ses.list[[8]]$ses >
    ses.list[[8]]$ses.sup))

levelplot(rbind(obj1, obj2), colorkey = my.ckey, par.settings = my.theme, border = "black")
```



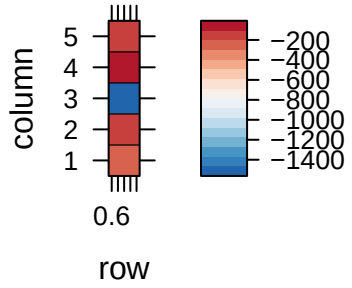

```
obj1.ses <- t(rbind(ses.list[[1]]$ses, ses.list[[2]]$ses, ses.list[[3]]$ses,
    ses.list[[4]]$ses))

obj2.ses <- t(rbind(ses.list[[5]]$ses, ses.list[[6]]$ses, ses.list[[7]]$ses,
    ses.list[[8]]$ses))

levelplot(rbind(obj1.ses, obj2.ses), colorkey = my.ckey, par.settings = my.theme,
    border = "black")
```



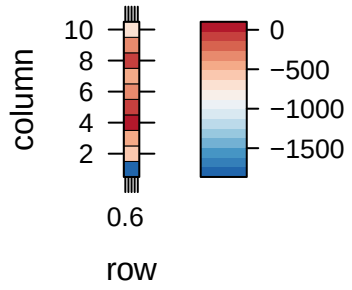

```
ses.list.multi <- ses.listofindex(list.ind.multi)

ses.list.multi[[1]] <- lapply(ses.list.multi[[1]], function(x) x[!is.na(ses.list.multi[[1]][[
ses.list.multi[[2]] <- lapply(ses.list.multi[[2]], function(x) x[!is.na(ses.list.multi[[2]][[
ses.list.multi[[3]] <- lapply(ses.list.multi[[3]], function(x) x[!is.na(ses.list.multi[[3]][[

levelplot(t(as.matrix(ses.list.multi[[1]]$ses)), colorkey = my.ckey, par.settings = my.theme,
    border = "black")
```




```
levelplot(t(as.matrix(ses.list.multi[[2]]$ses)), colorkey = my.ckey, par.settings = my.theme,
    border = "black")
```



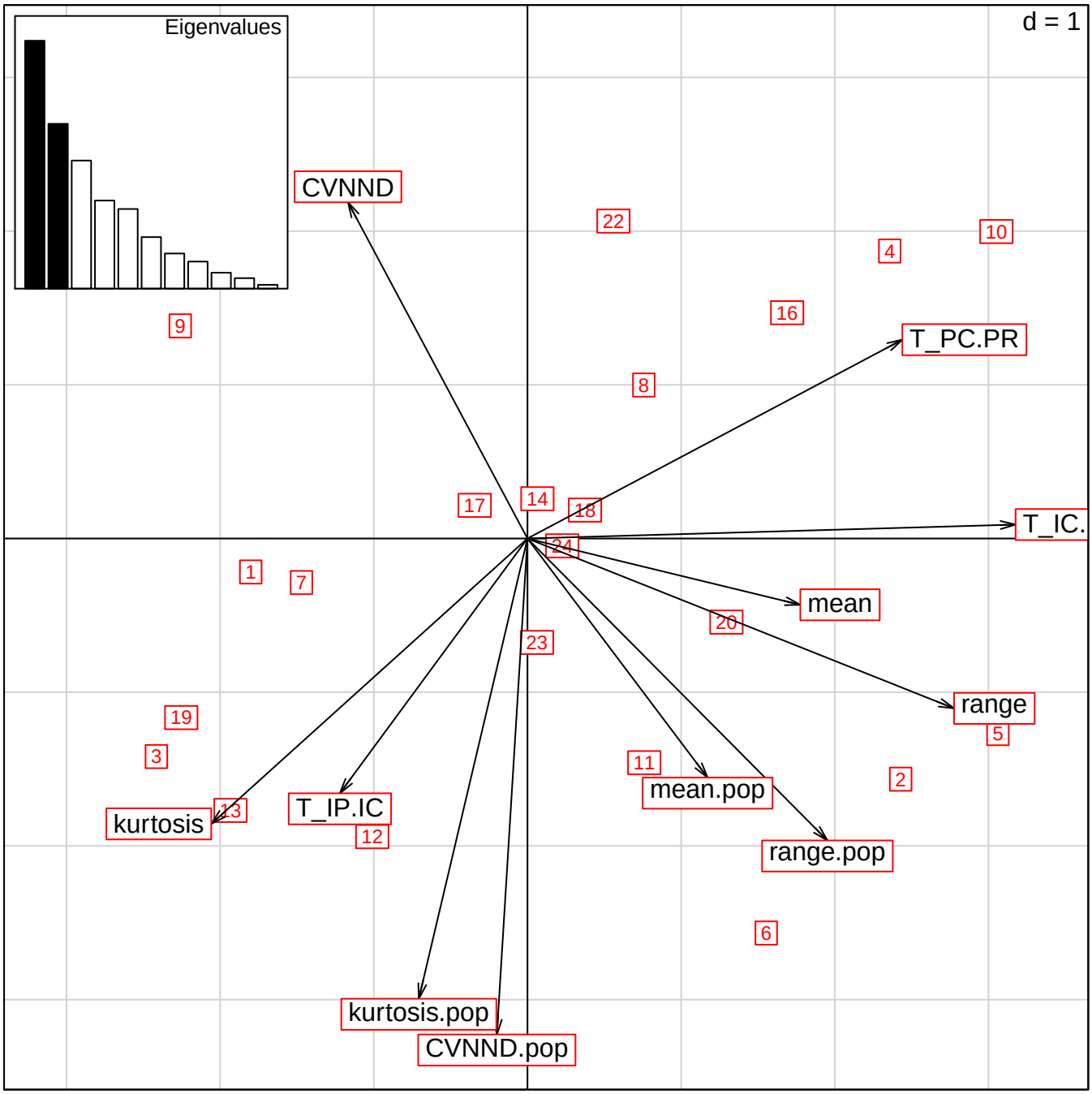

```
levelplot(t(as.matrix(ses.list.multi[[3]]$ses)), colorkey = my.ckey, par.settings = my.theme,
    border = "black")
```



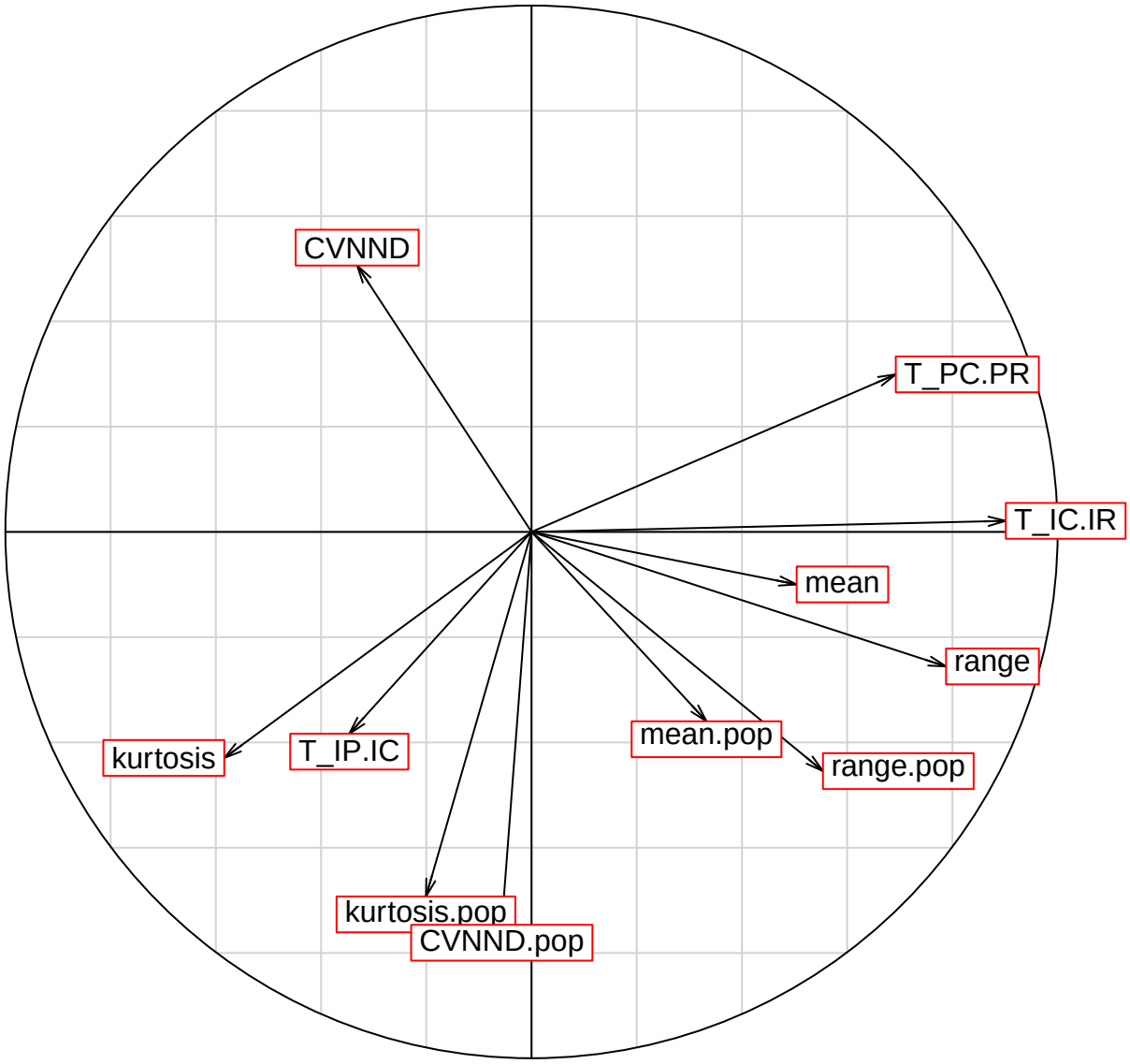
To finish, a multivariate analysis of

```
require(ade4)
require(adegenet)

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

## Error:  indice hors limites

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

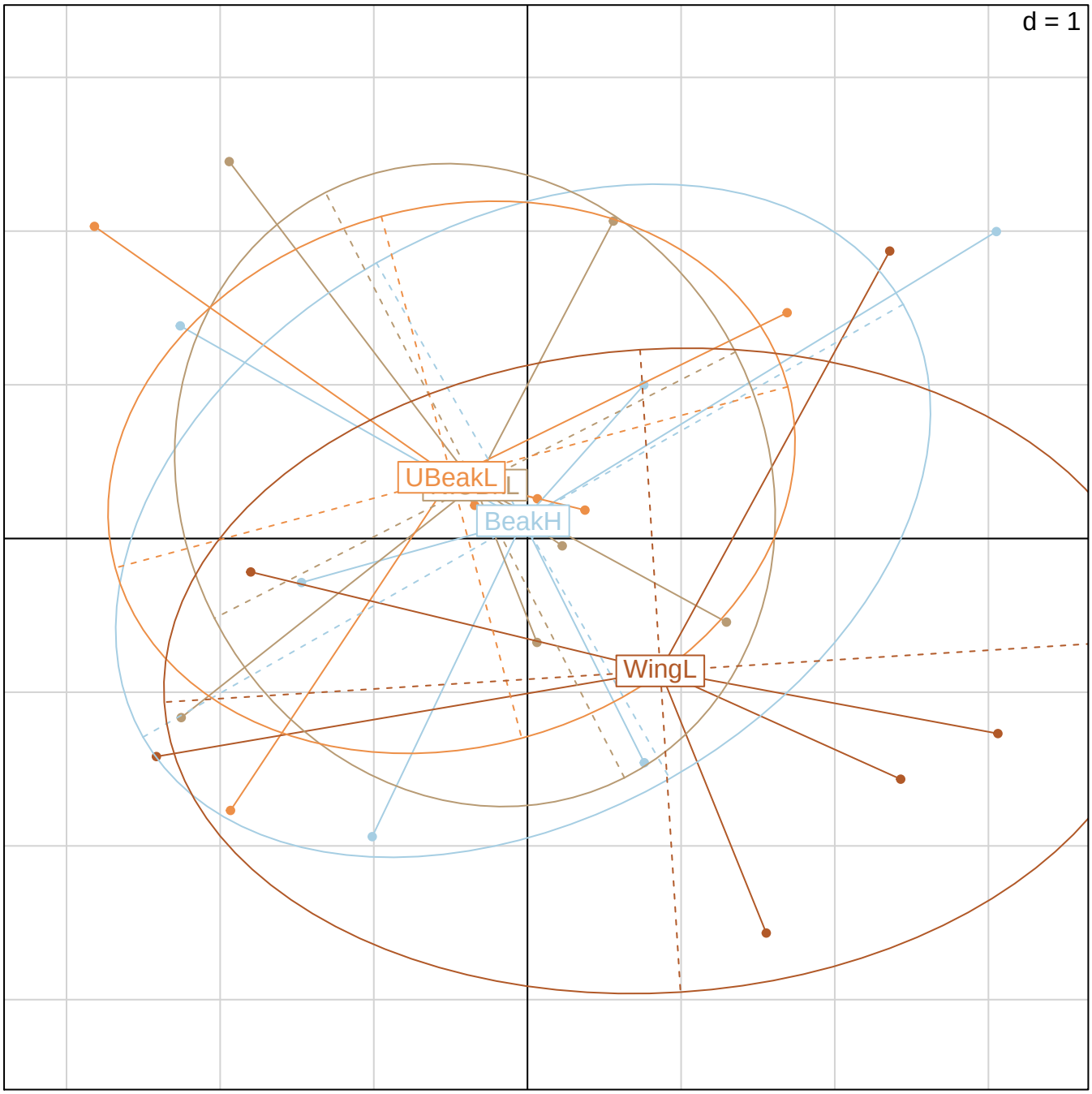
```
## Error:  objet 'matfordudi' introuvable

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

## Error:  objet 'matfordudi' introuvable

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

## Error:  objet 'matfordudi' introuvable

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

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

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

s.corcircle(res.dudi$co)

## Error:  objet 'res.dudi' introuvable

s.label(res.dudi$li, add.plot = t)

## Error:  objet 'res.dudi' introuvable

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



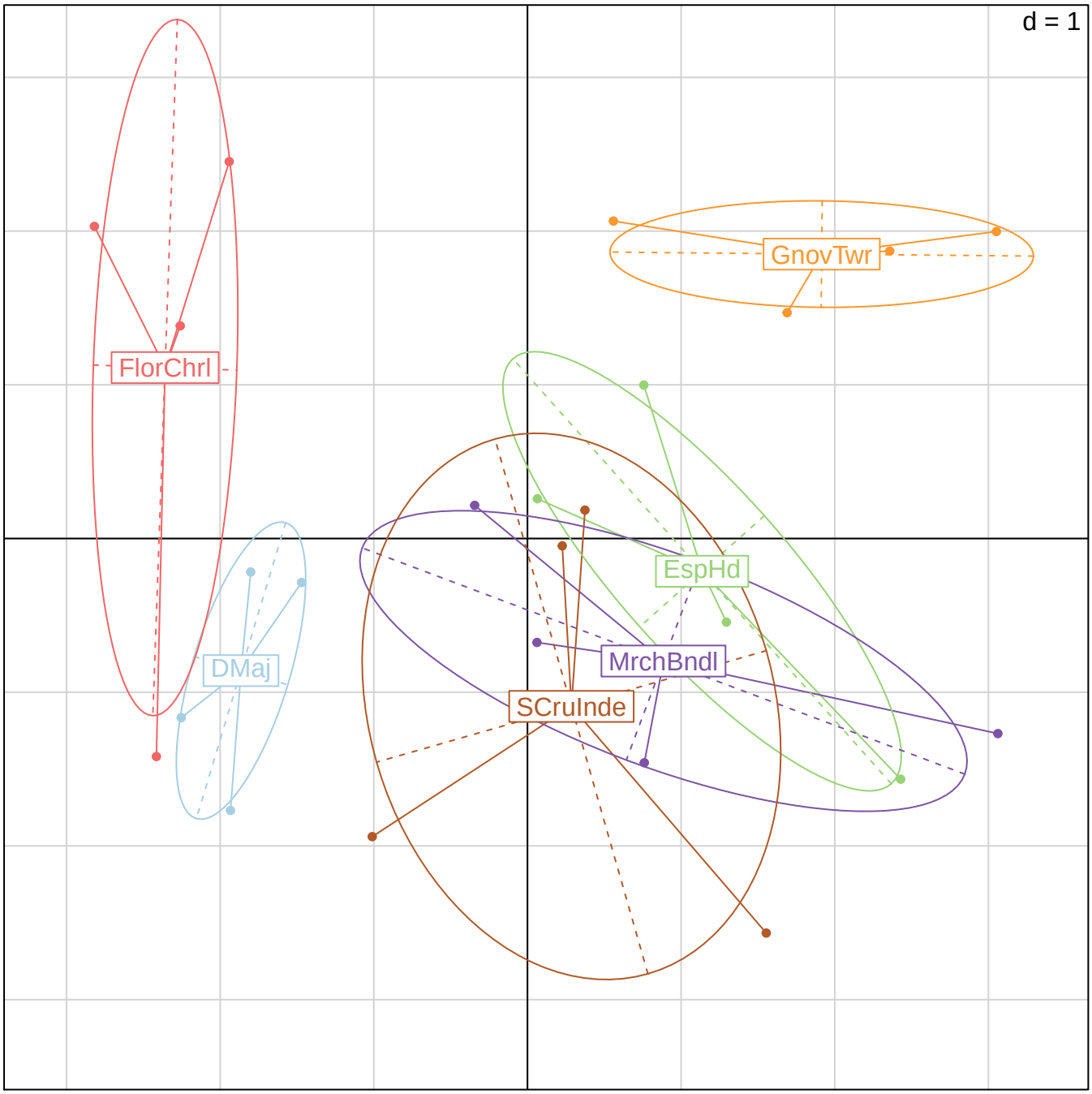

```
s.corcircle(res.dudi2$co)
```




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




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


d = 1
FlorChrl
GnovTwr
EspHd
DMaj
MrchBndl
SCruInde