multivariate stats (1)

Jérôme Mathieu

9/28/2021

Data

```
X <- read.csv("TD_R_data.csv")
idx_sp <- grep("sp[A-Z]", names(X))
```

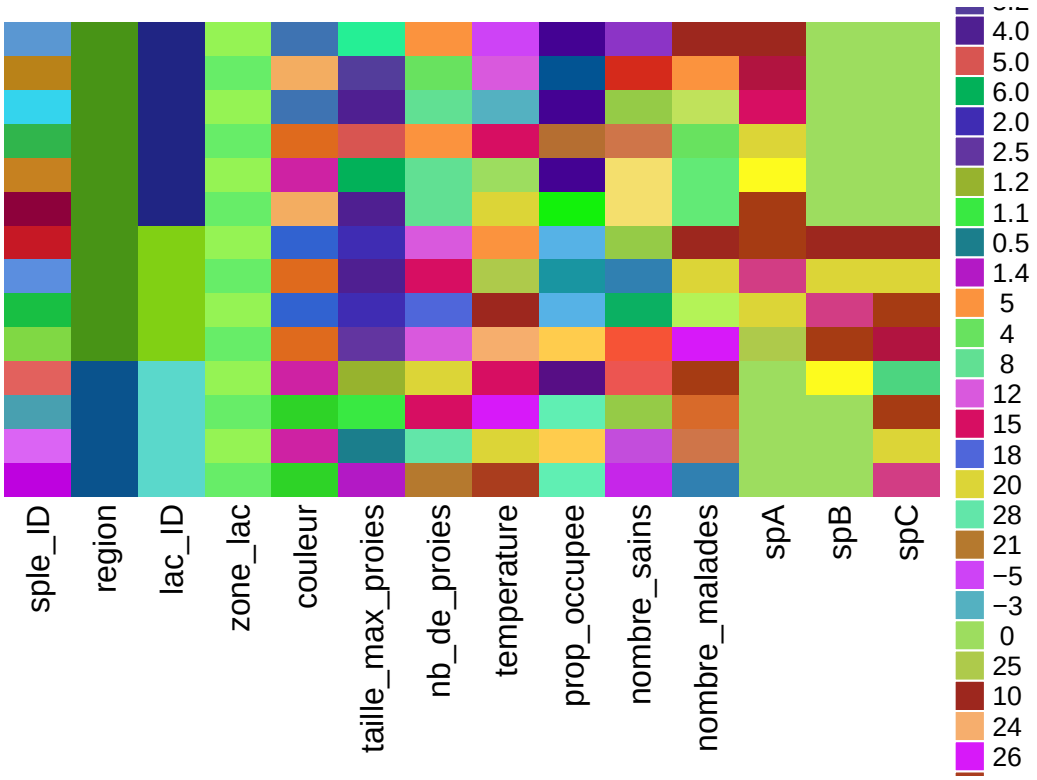
HEATMAPS

link

```
library(ComplexHeatmap)
```

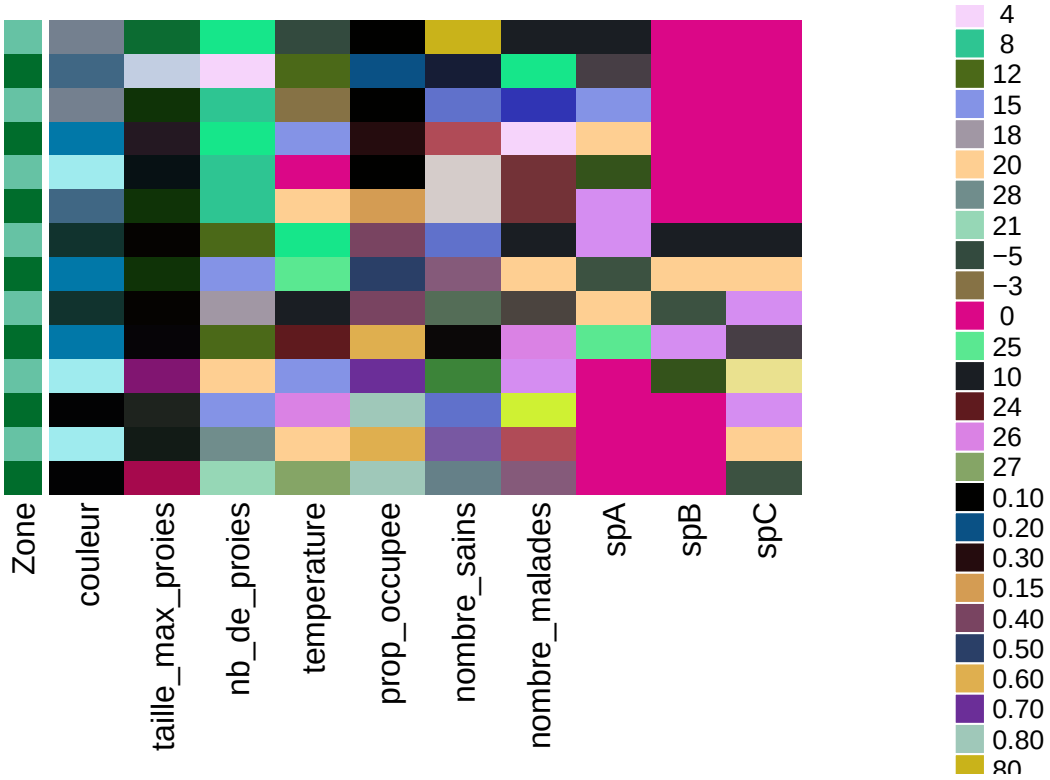
Heatmap of all data together

```
Heatmap(X)
```

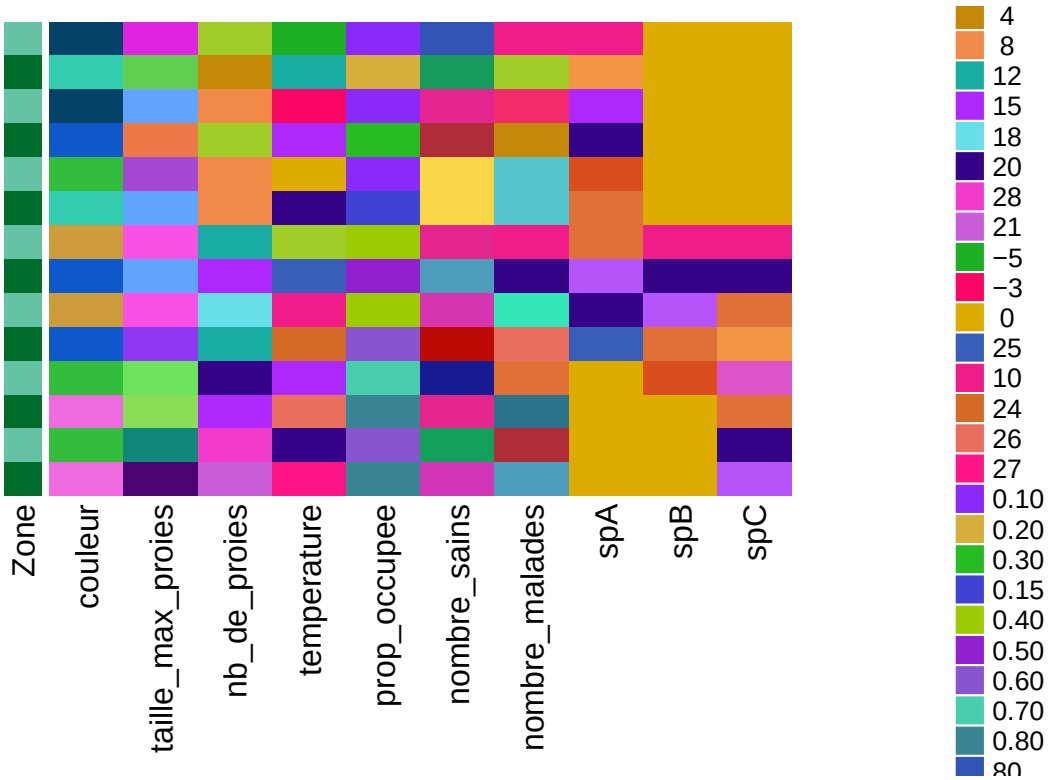


Add labels to observations

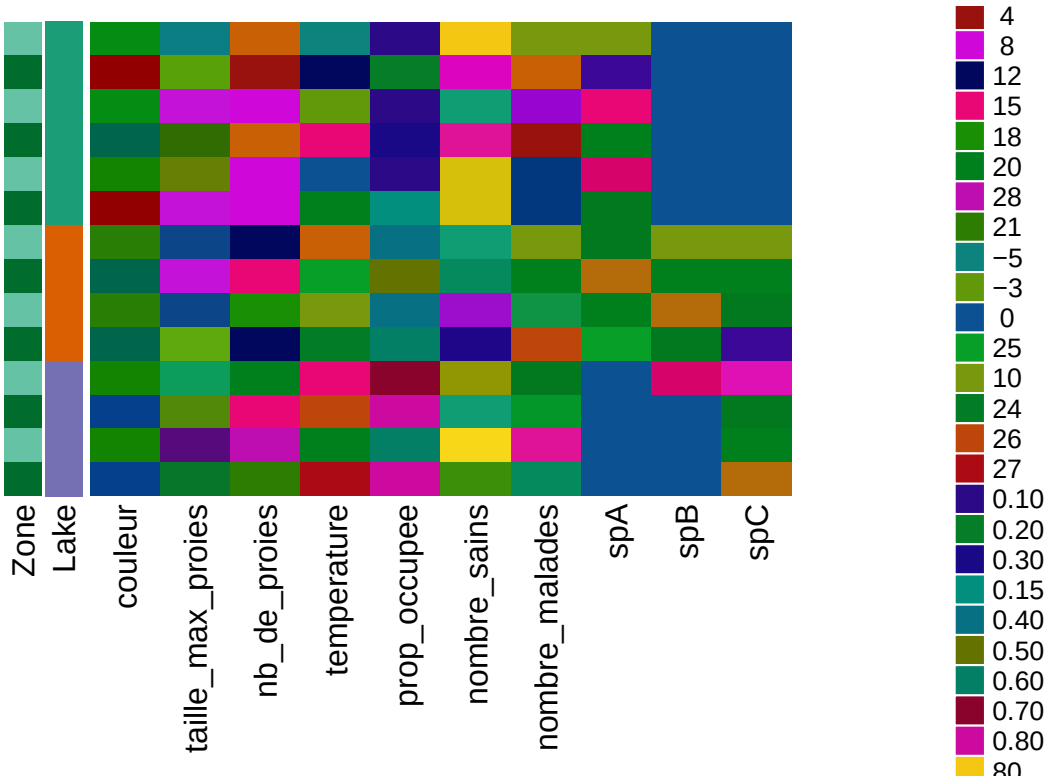
```
    # add a categorical label to observations

        row_ha <- rowAnnotation( Zone = X[,"zone_lac"],
                                 col = list (Zone = c("milieu" = '#66c2a4',"bord" = '#006d2c')))

        Heatmap(X[,-c(1:4)], left_annotation = row_ha)
```




```
        # add a categorical label to observations with specific labels

row_ha_engl <- rowAnnotation( Zone = X[,"zone_lac"],
                              col = list (Zone = c("milieu" = '#66c2a4', "bord" = '#006d2c')),
                              annotation_legend_param = list (title = "Zone",
                                                              at = c("bord","milieu"),
                                                              labels = c("Border","Center"))
                                                    )

            Heatmap(X[,-c(1:4)], left_annotation = row_ha_engl)
```

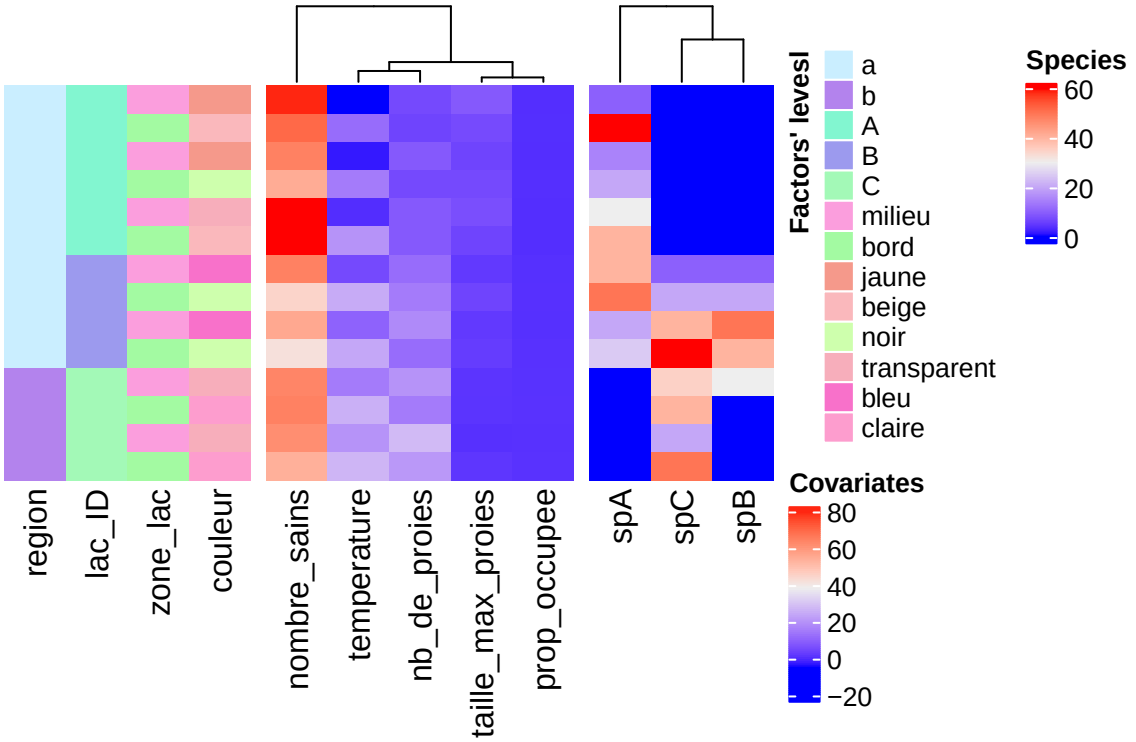



```
# Add two labels to observations with customized colors and custom labels

row_ha_2labs <- rowAnnotation( Zone = X[,"zone_lac"],
                               Lake = X[,"lac_ID"],
                                col = list (
                                   Zone = c("milieu" = '#66c2a4', "bord" = '#006d2c'),
                                         Lake = c("A" = '#1b9e77', "B" = '#d95f02', "C" = '#7570b3')
                                             ),
                                  annotation_legend_param = list (
                                       Zone = list(title = "Zone",
                                                   at = c("bord","milieu"),
                                                   labels = c("Border","Center")),
                                             Lake = list(title = "Lake",
                                                         at = c("A","B","C"),
                                                         labels = c("Creteil","Avon","Foljuif"))
                                                        )


                                         )

Heatmap(X[,-c(1:4)], left_annotation = row_ha_2labs)
```



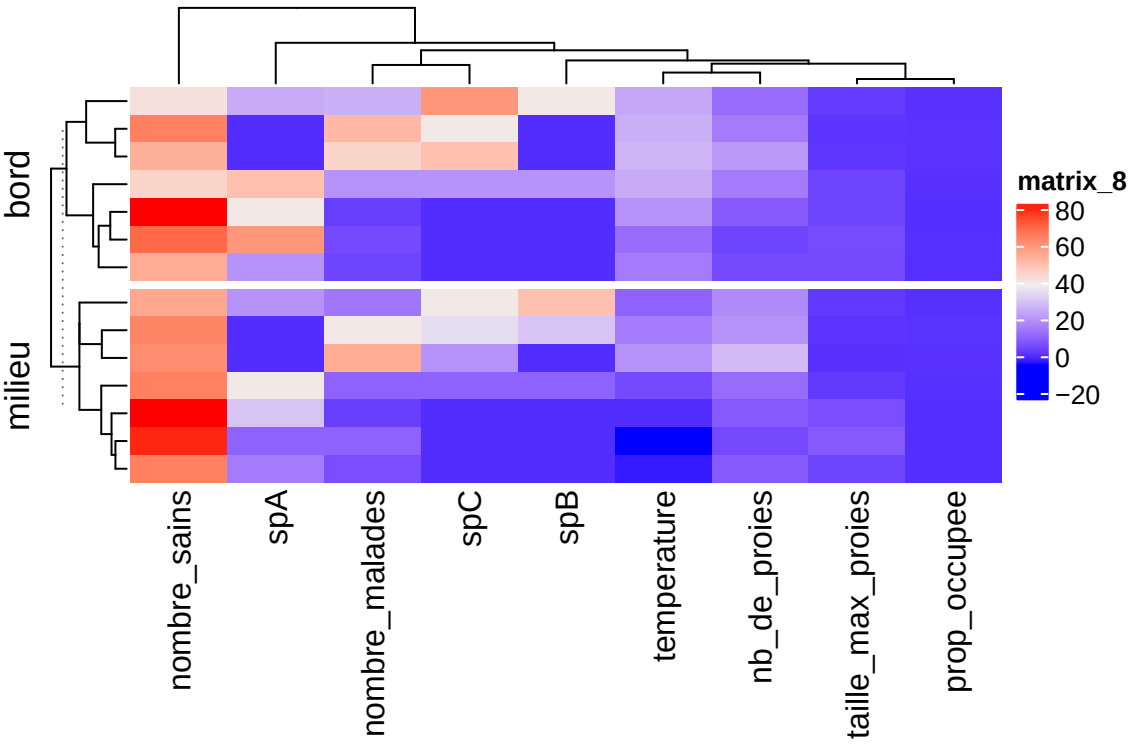
List of heatmaps by blocks of data

```
heat_cat <- Heatmap(X[,c(2:5)],
                    heatmap_legend_param = list(title = "Factors' levesl",
                                                title_position = "lefttop-rot"))
heat_cont <- Heatmap(X[,c(6:10)],
                     heatmap_legend_param = list(title = "Covariates"))
heat_sp <- Heatmap(X[,grep("sp[A-Z]", names(X))],
                   heatmap_legend_param = list(title = "Species"))

heat_cat + heat_cont + heat_sp
```

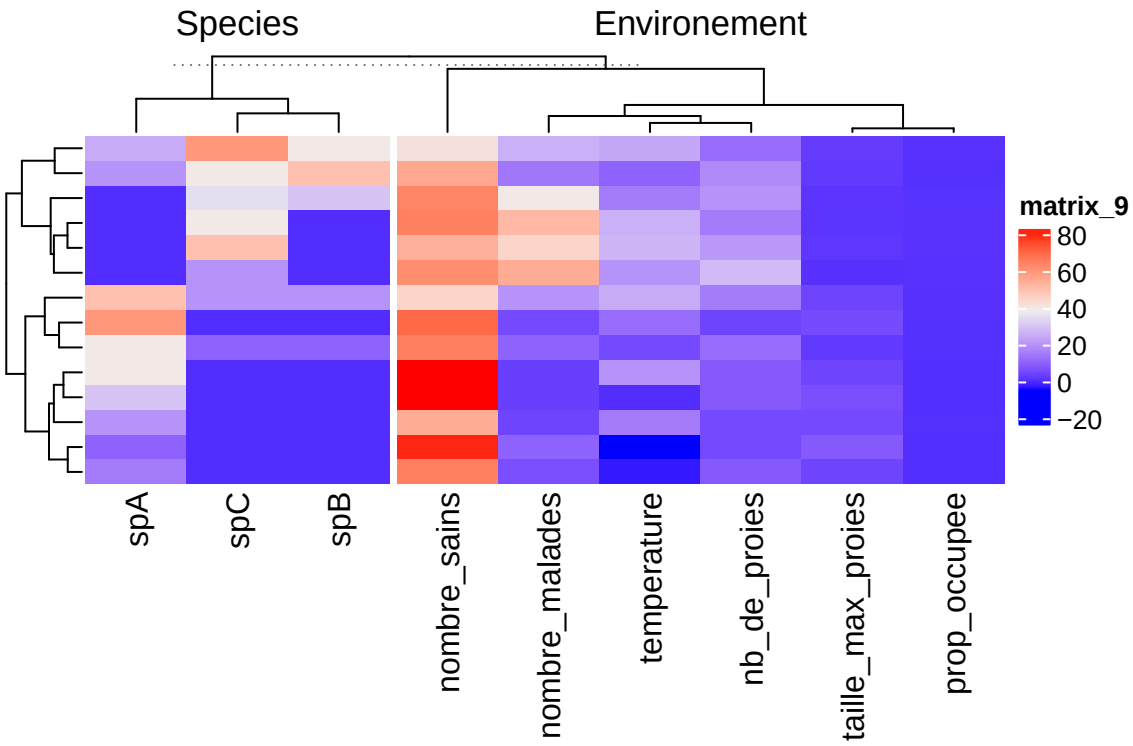


Splitting a heatmap

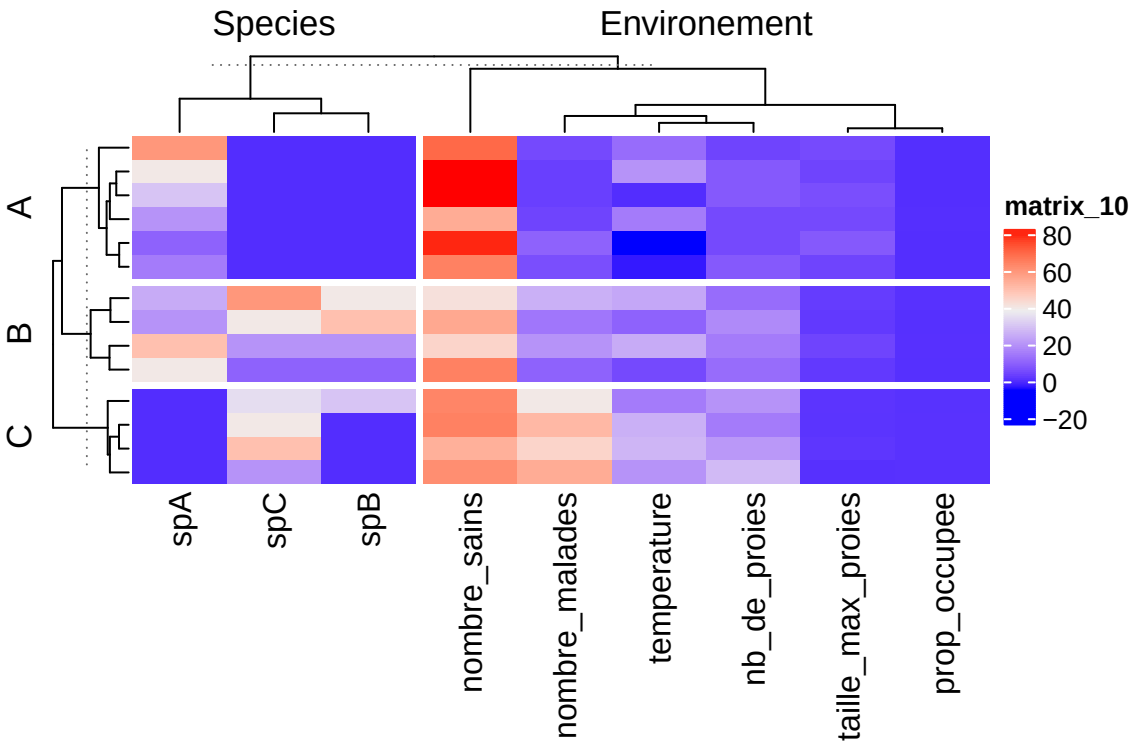
```
# heatmap | rows splited by a factor
    Heatmap( X[,-c(1:5)], row_split = factor( X[,"zone_lac"] ) )
```



```
# heatmap | columns splited by a factor
    Heatmap( X[,-c(1:5)], column_split =c(rep( "Environement",6), rep("Species",3) ) )
```




```
# heatmap | columns and rows splited by a factor
    Heatmap( X[,-c(1:5)], column_split =c(rep("Environement",6), rep("Species",3)),
                          row_split = factor(X[,"lac_ID"]) )
```



DISTANCE MATRICES

```
    library(vegan)
```

Compute distance between samples - toy example

```
        # euclidian distance between samples
            X[ c(1,12), idx_sp ]
```

```
##    spA spB spC
## 1   10   0   0
## 12   0   0  40
```

```
            X[ 1, idx_sp ] - X[ 12, idx_sp ]
```

```
##   spA spB spC
## 1  10   0 -40
```

```
            sqrt(sum((X[ 1, idx_sp ] - X[ 12, idx_sp ])^2))
```

```
## [1] 41.23106
```

```
            data.matrix( dist(X[ c(1,12), idx_sp ]) )
```

```
##           1       12
## 1   0.00000 41.23106
## 12 41.23106  0.00000
```

Euclidian paradox

```
  # smaller dataset to work on
            ( Xsmall <- X[ c(1,2,7,12), idx_sp ] )
```

```
##    spA spB spC
## 1   10   0   0
## 2   60   0   0
## 7   40  10  10
## 12   0   0  40
```

```
        data.matrix( dist(Xsmall) )
```

```
##           1        2        7       12
## 1   0.00000 50.00000 33.16625 41.23106
## 2  50.00000  0.00000 24.49490 72.11103
## 7  33.16625 24.49490  0.00000 50.99020
## 12 41.23106 72.11103 50.99020  0.00000
```

```
        # 1-2 more different than 1-12 !
        # 2-7 more different than 1-2 !

        # Hellinger transformation can fix the issue
                Xsmall_h <- decostand(Xsmall, "hellinger")
                data.matrix( dist(Xsmall_h ))   # looks ok
```

```
##           1         2         7       12
## 1  0.0000000 0.0000000 0.6058109 1.414214
## 2  0.0000000 0.0000000 0.6058109 1.414214
```

```
## 7  0.6058109 0.6058109 0.0000000 1.087889
## 12 1.4142136 1.4142136 1.0878894 0.000000
```

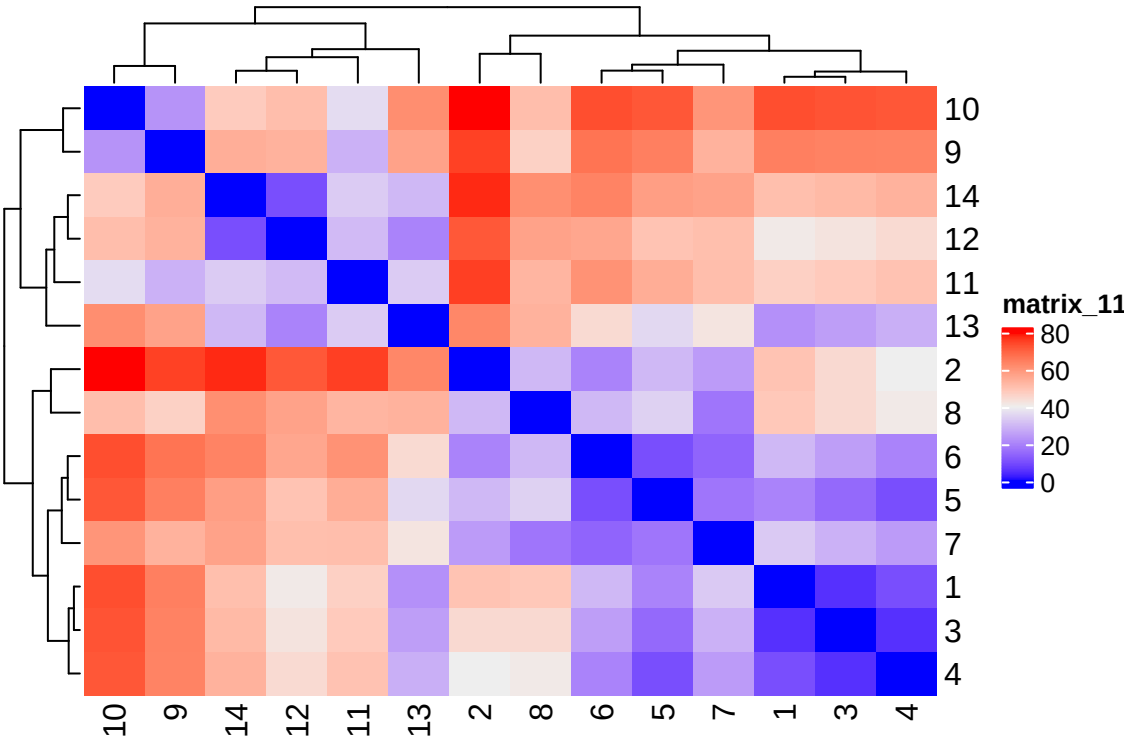
Distance matrices Vizualisation

Compute distance matrices

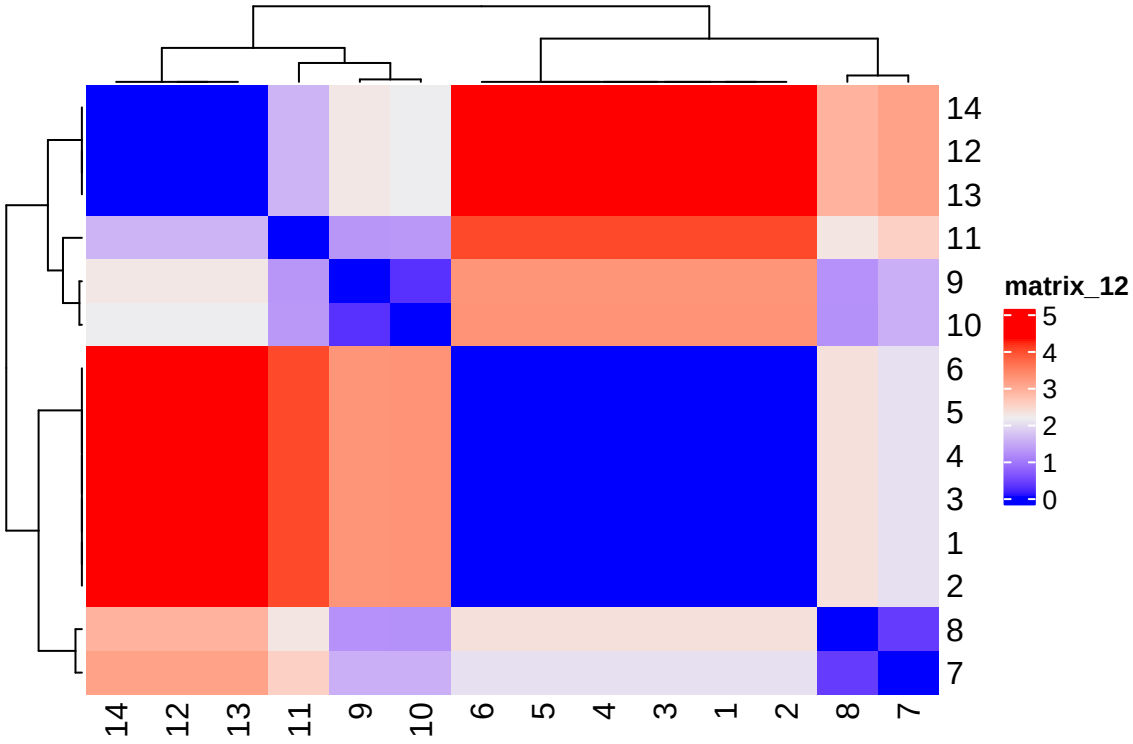
```
  # euclidian matrix
        dist_euclid <- data.matrix(dist( X[, idx_sp] ))

# hellinger distance (euclidian distance on hellinger transformed data)
            X_h <- decostand(X[,idx_sp], "hellinger")
            dist_hell <- data.matrix(dist(data.matrix(dist( X_h ) )))
```

Plots of distance matrix

```
# euclidian distance
Heatmap(dist_euclid)
```

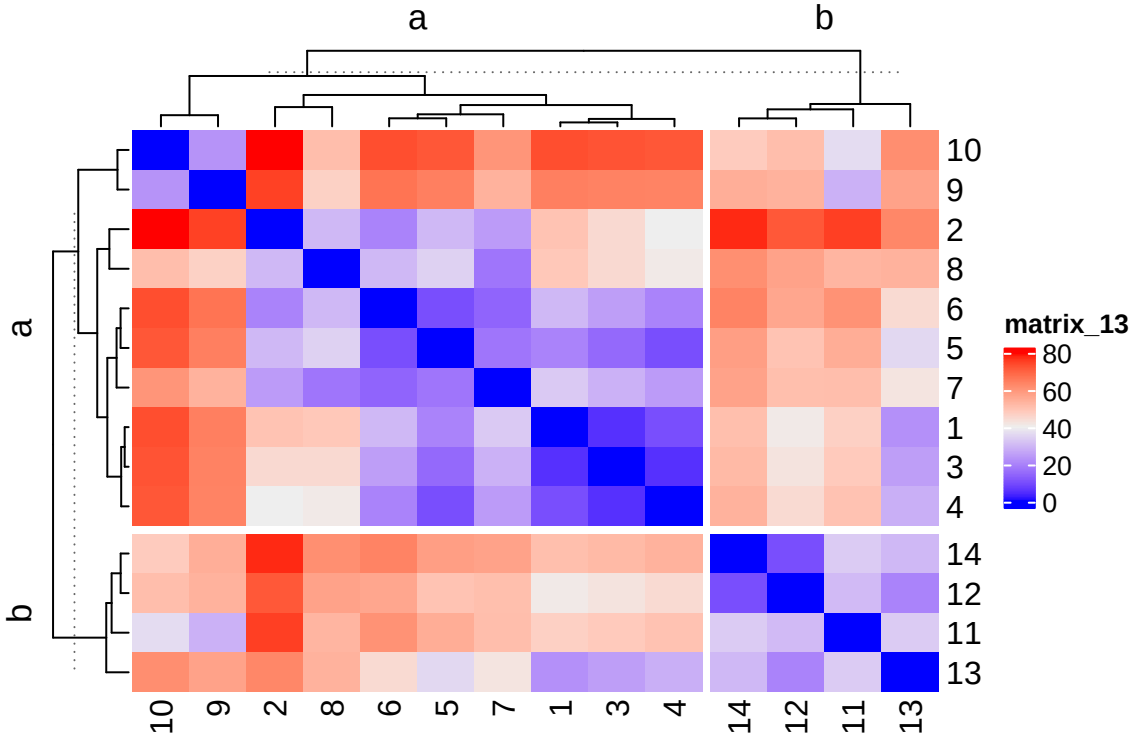



```
# hellinger distance
Heatmap(dist_hell)
```

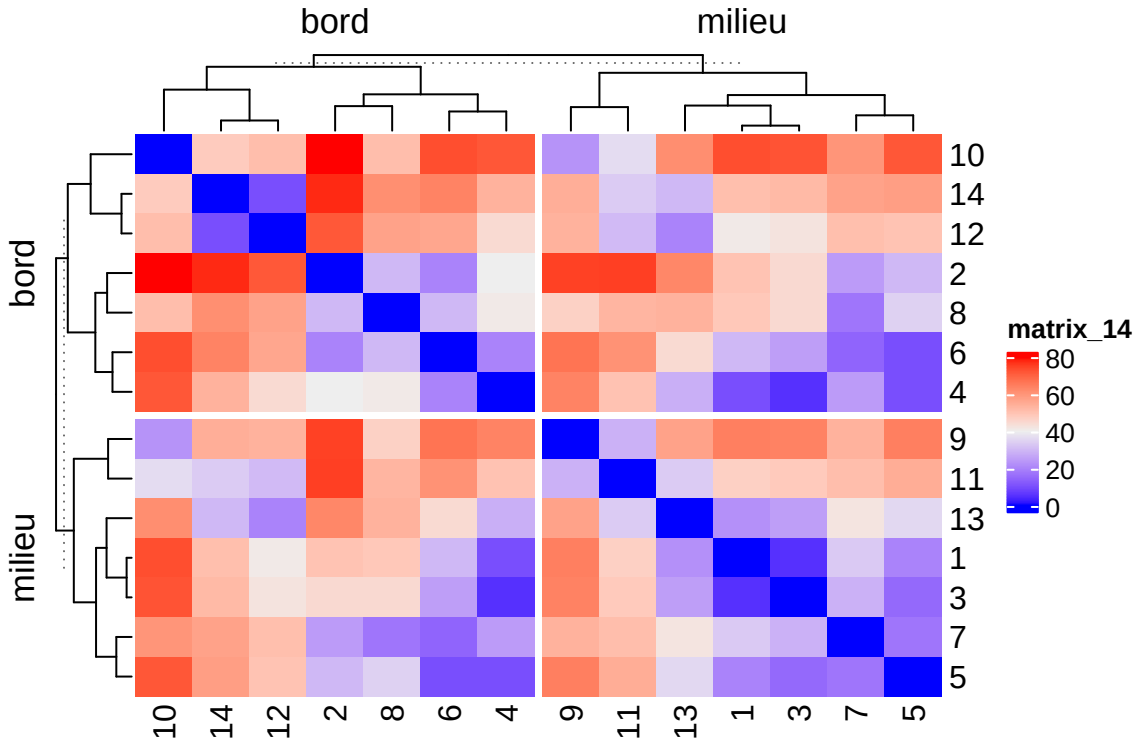


Split distance matrices

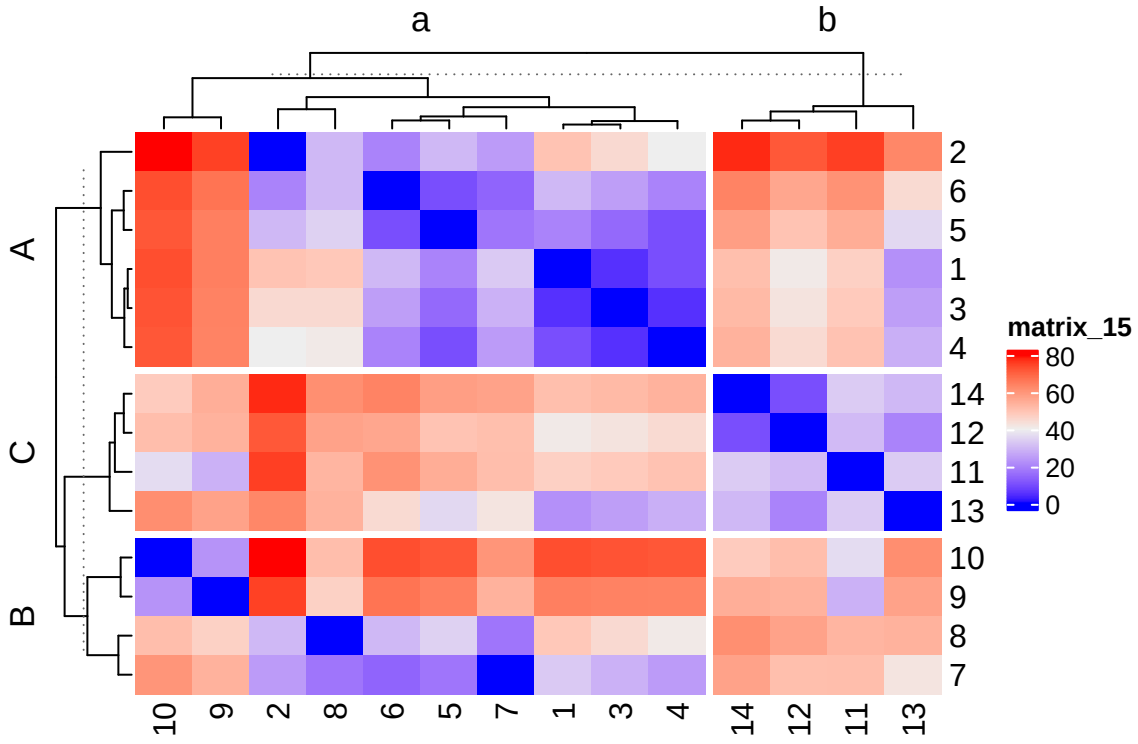
```
Heatmap(dist_euclid, row_split = factor(X[,"region"]), column_split = factor(X[,"region"]) )
```



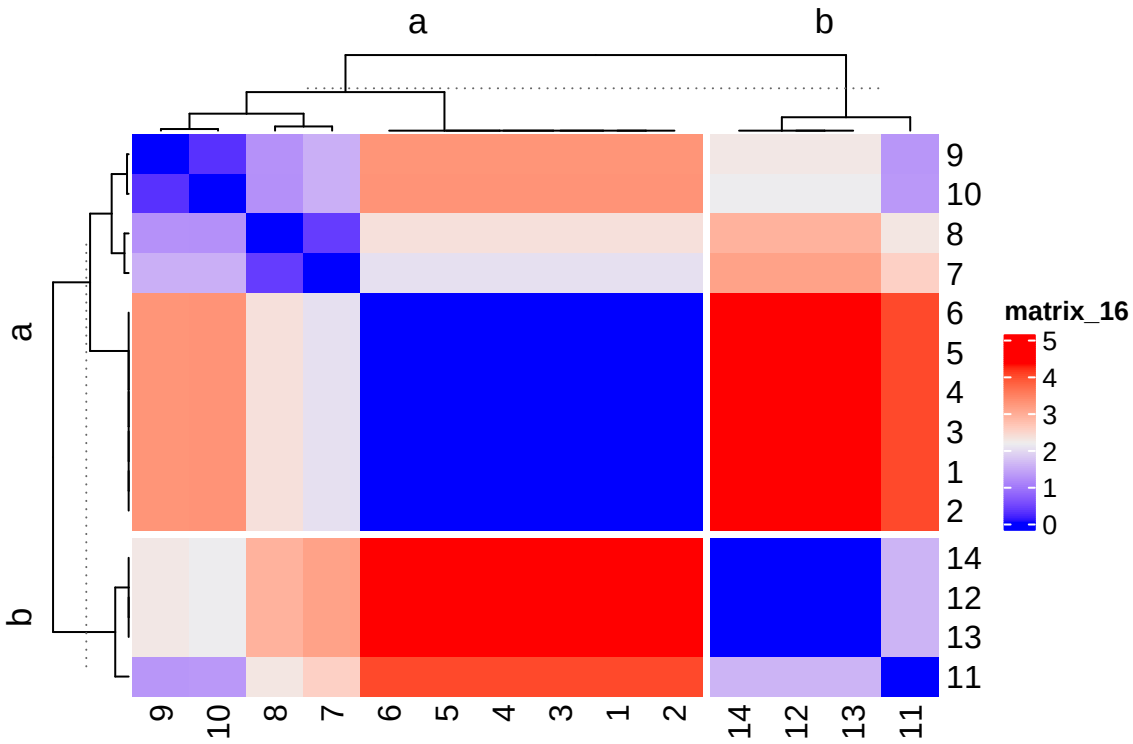

```
Heatmap(dist_euclid, row_split = factor(X[,"zone_lac"]), column_split = factor(X[,"zone_lac"]) )
```




```
# split by column and rows
Heatmap(dist_euclid, row_split = factor(X[,"lac_ID"]), column_split = factor(X[,"region"]) )
```




```
Heatmap(dist_hell, row_split = factor(X[,"region"]), column_split = factor(X[,"region"]) )
```



Ordination

```
library(ade4)
library(factoextra)
```

```
## Loading required package: ggplot2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

Ordination of quantitative data

PCA

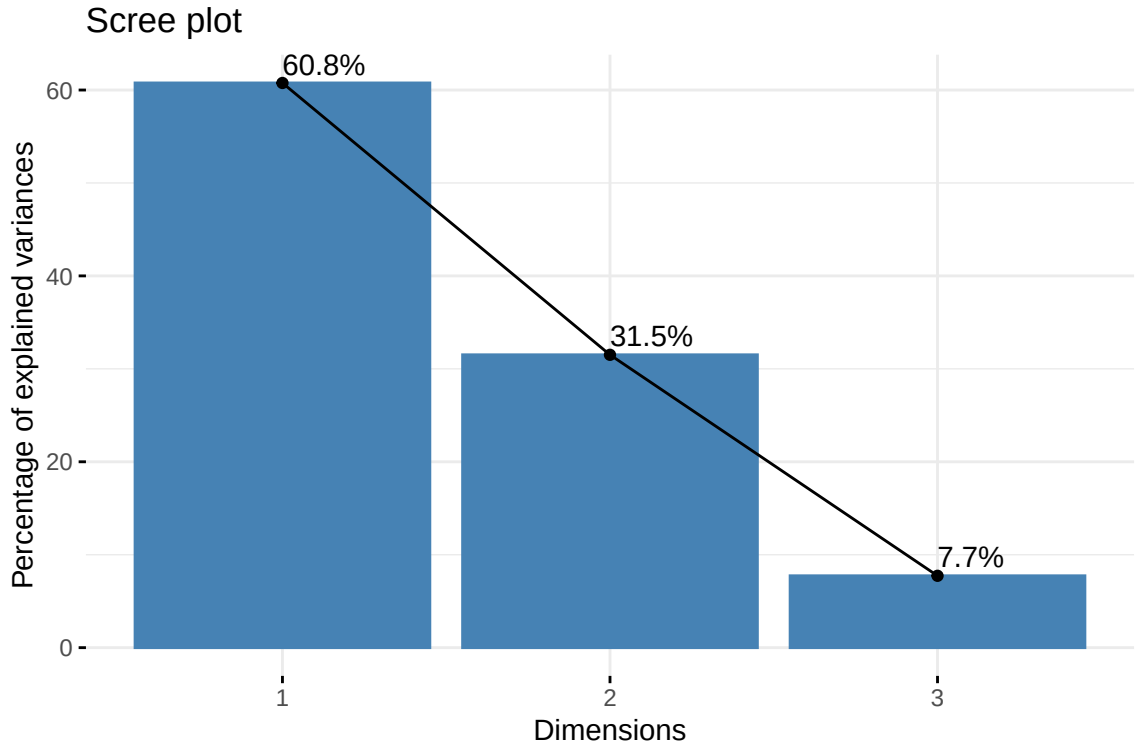
link

```
        my_pca <- dudi.pca( X[,idx_sp], scannf = FALSE, nf = 2, scale = F)
    my_pca
```

```
## Duality diagramm
## class: pca dudi
## $call: dudi.pca(df = X[, idx_sp], scale = F, scannf = FALSE, nf = 2)
##
## $nf: 2 axis-components saved
## $rank: 3
## eigen values: 651.1 337.6 82.86
##   vector length mode    content
## 1 $cw    3      numeric column weights
## 2 $lw    14     numeric row weights
## 3 $eig   3      numeric eigen values
##
##   data.frame nrow ncol content
```

```
## 1 $tab       14   3    modified array
## 2 $li        14   2    row coordinates
## 3 $l1        14   2    row normed scores
## 4 $co        3    2    column coordinates
## 5 $c1        3    2    column normed scores
## other elements: cent norm
```

```
        my_pca_hell <- dudi.pca( X_h, scannf = FALSE, nf = 2, scale = F)

        # eigenvalues

            fviz_screeplot(my_pca, addlabels = TRUE)
```




```
        # analysis of variables

                # Extract the results for variables
                  vars <- get_pca_var(my_pca)
                  vars
```

```
## Principal Component Analysis Results for variables
##  ===================================================
##   Name       Description
## 1 "$coord"   "Coordinates for the variables"
## 2 "$cor"     "Correlations between variables and dimensions"
## 3 "$cos2"    "Cos2 for the variables"
## 4 "$contrib" "contributions of the variables"
```

```
                  vars$contrib
```

```
##        Dim.1     Dim.2
## spA 20.40765 64.546995
## spB 19.33277 33.501388
```

```
## spC 60.25959  1.951617
```

```
                  # variable's coordinates
                     vars$coord
```

```
##         Dim.1     Dim.2
## spA -11.52680 14.762518
## spB  11.21913 10.635400
## spC  19.80731  2.566963
```

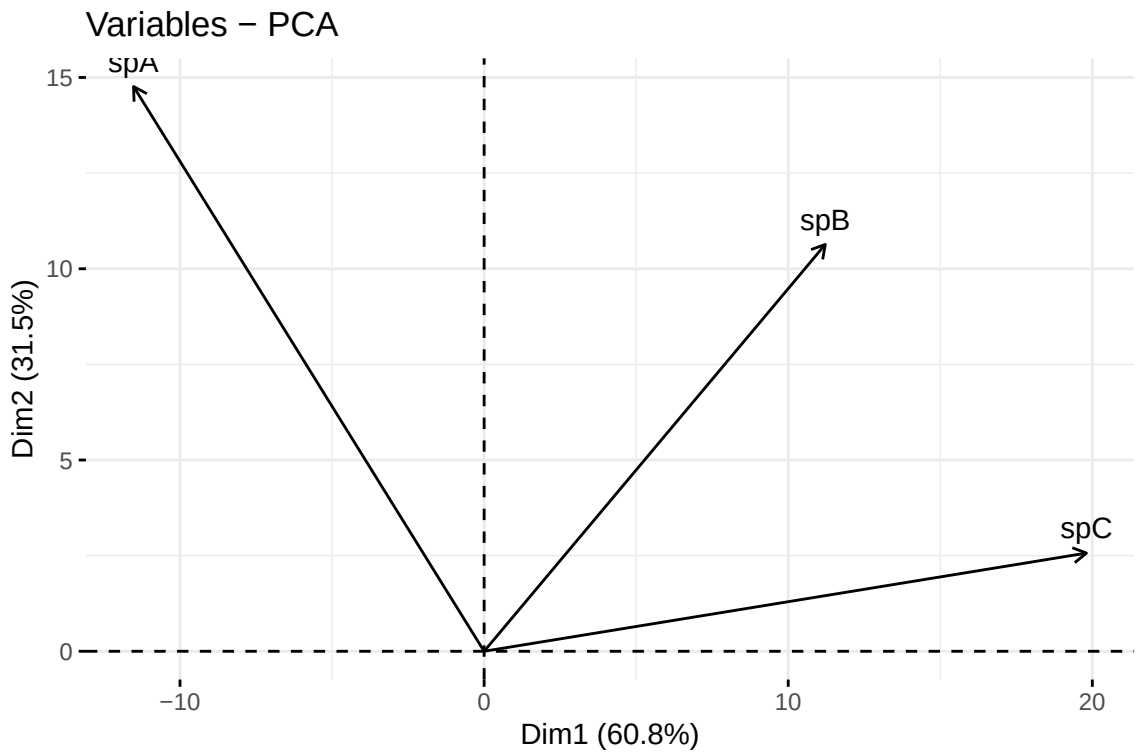
```
                     my_pca$co
```

```
##         Comp1     Comp2
## spA -11.52680 14.762518
## spB  11.21913 10.635400
## spC  19.80731  2.566963
```

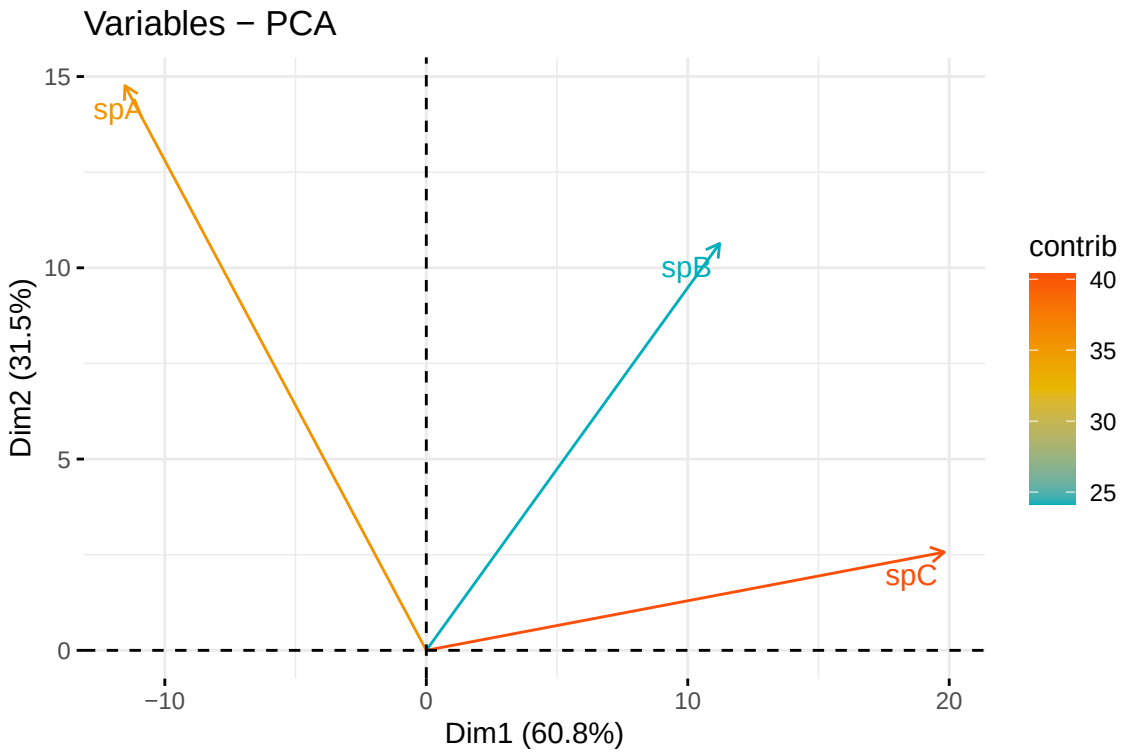
```
                  apply(vars$contrib,2,sum) # check : contribution of variables sum up to 100%
```

```
## Dim.1 Dim.2
##   100   100
```

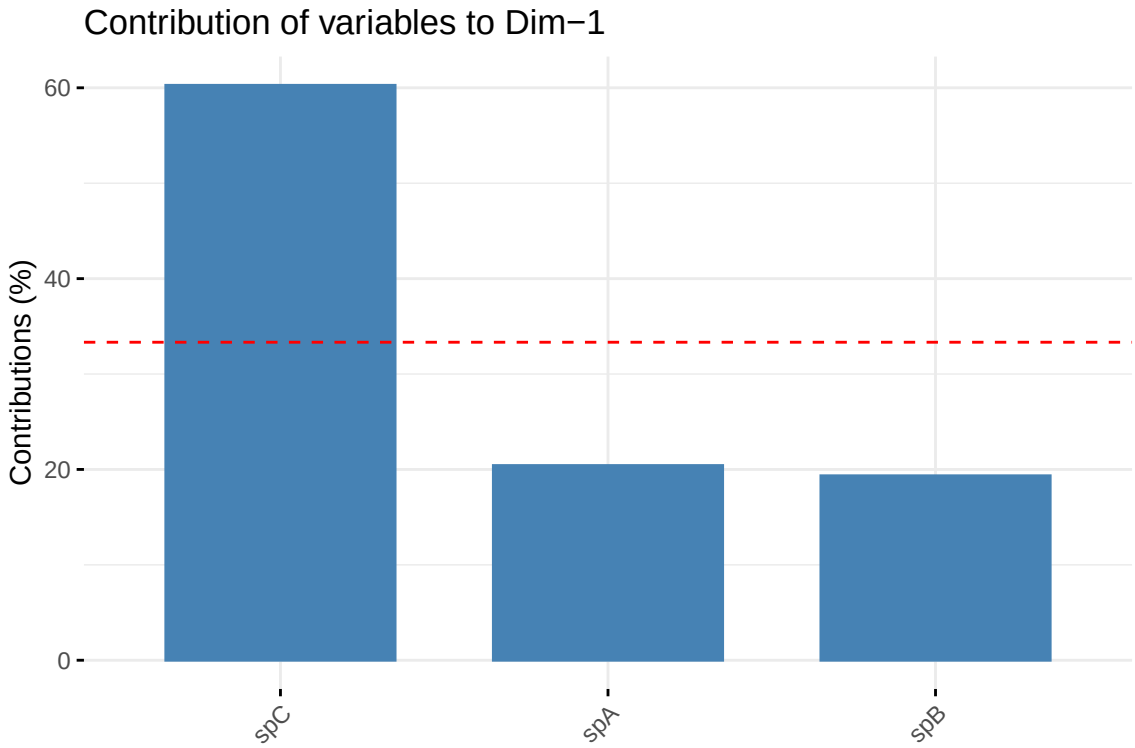
```
                # correlation circle
                   fviz_pca_var(my_pca)
```




```
                  fviz_pca_var(my_pca, col.var="contrib",
                   gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
                   repel = TRUE)
```




```
# Plot contributions of variables to PC1
    fviz_contrib(my_pca, choice = "var", axes = 1, top = 10)
```




```
# analysis of samples

    # Extract the results for individuals
    inds <- get_pca_ind(my_pca)
```

```
                  inds
```

```
## Principal Component Analysis Results for individuals
##  ===================================================
##   Name       Description
## 1 "$coord"   "Coordinates for the individuals"
## 2 "$cos2"    "Cos2 for the individuals"
## 3 "$contrib" "contributions of the individuals"
```

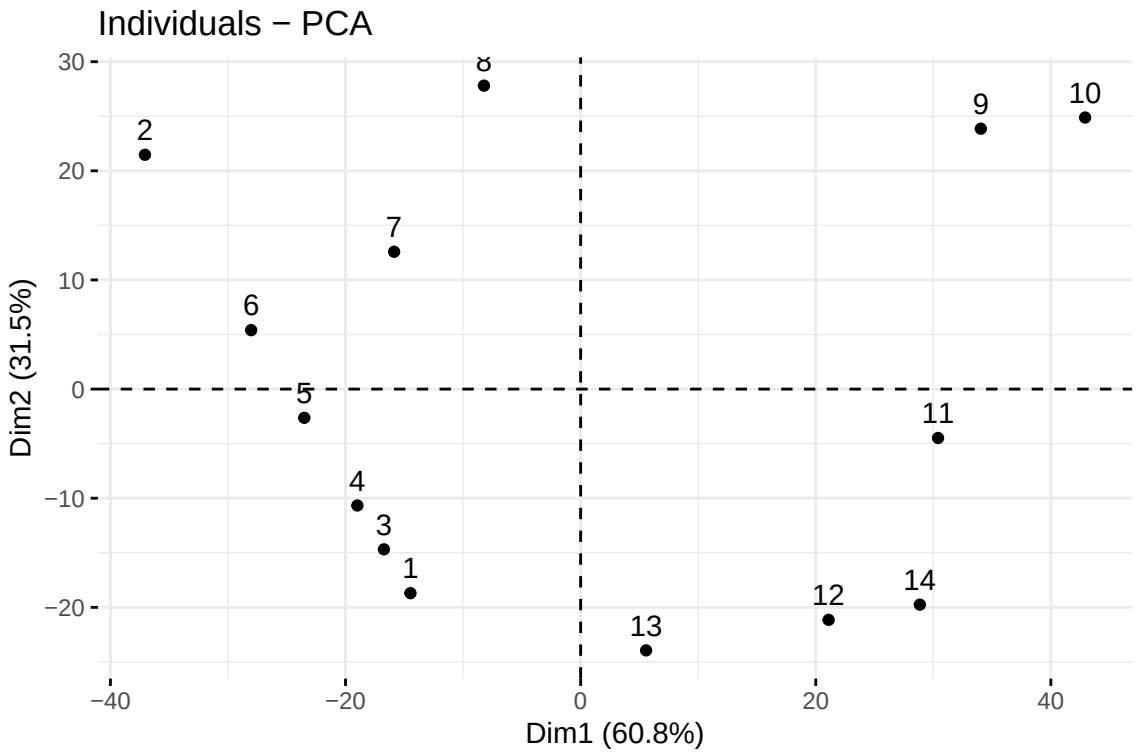
```
                # samples' coordinates
                  inds$coord
```

```
##         Dim.1      Dim.2
## 1  -14.473625 -18.701293
## 2  -37.061036  21.469280
## 3  -16.732366 -14.684235
## 4  -18.991107 -10.667178
## 5  -23.508589  -2.633064
## 6  -28.026072   5.401051
## 7  -15.866463  12.586092
## 8   -8.224336  27.805247
## 9   34.044233  23.861025
## 10  42.913998  24.884049
## 11  30.404037  -4.481782
## 12  21.094677 -21.147396
## 13   5.569267 -23.941402
## 14  28.857382 -19.750393
```

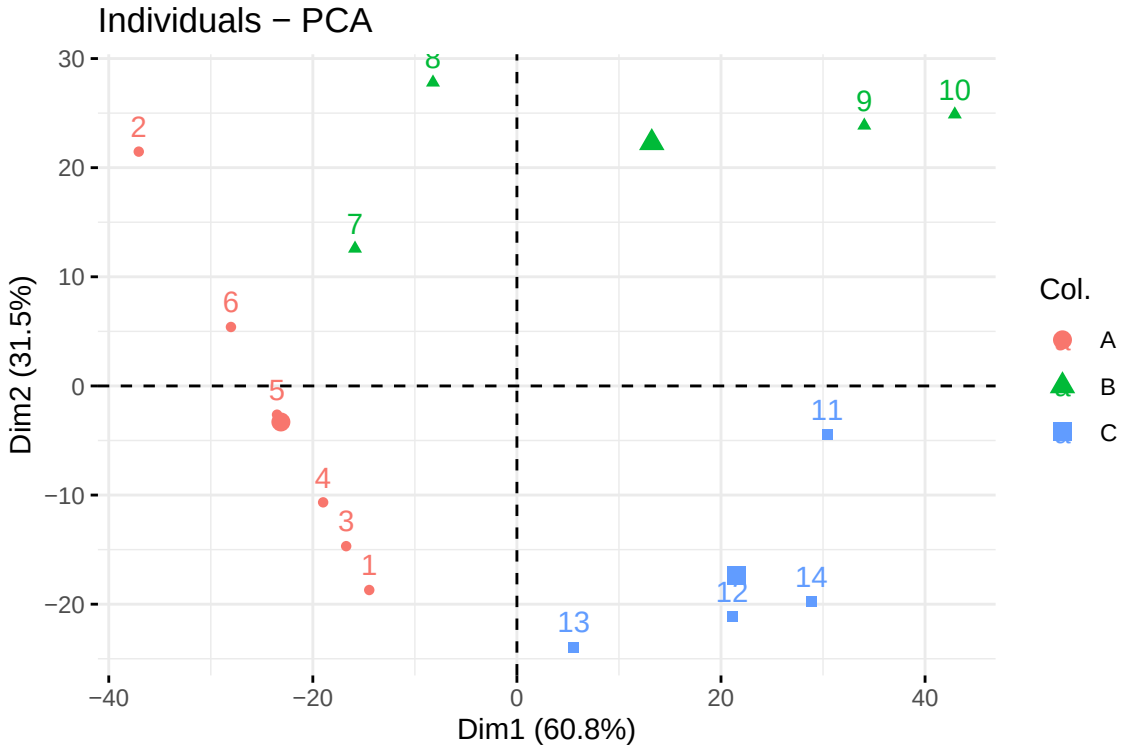
```
                  my_pca$li
```

```
##         Axis1      Axis2
## 1  -14.473625 -18.701293
## 2  -37.061036  21.469280
## 3  -16.732366 -14.684235
## 4  -18.991107 -10.667178
## 5  -23.508589  -2.633064
## 6  -28.026072   5.401051
## 7  -15.866463  12.586092
## 8   -8.224336  27.805247
## 9   34.044233  23.861025
## 10  42.913998  24.884049
## 11  30.404037  -4.481782
## 12  21.094677 -21.147396
## 13   5.569267 -23.941402
## 14  28.857382 -19.750393
```

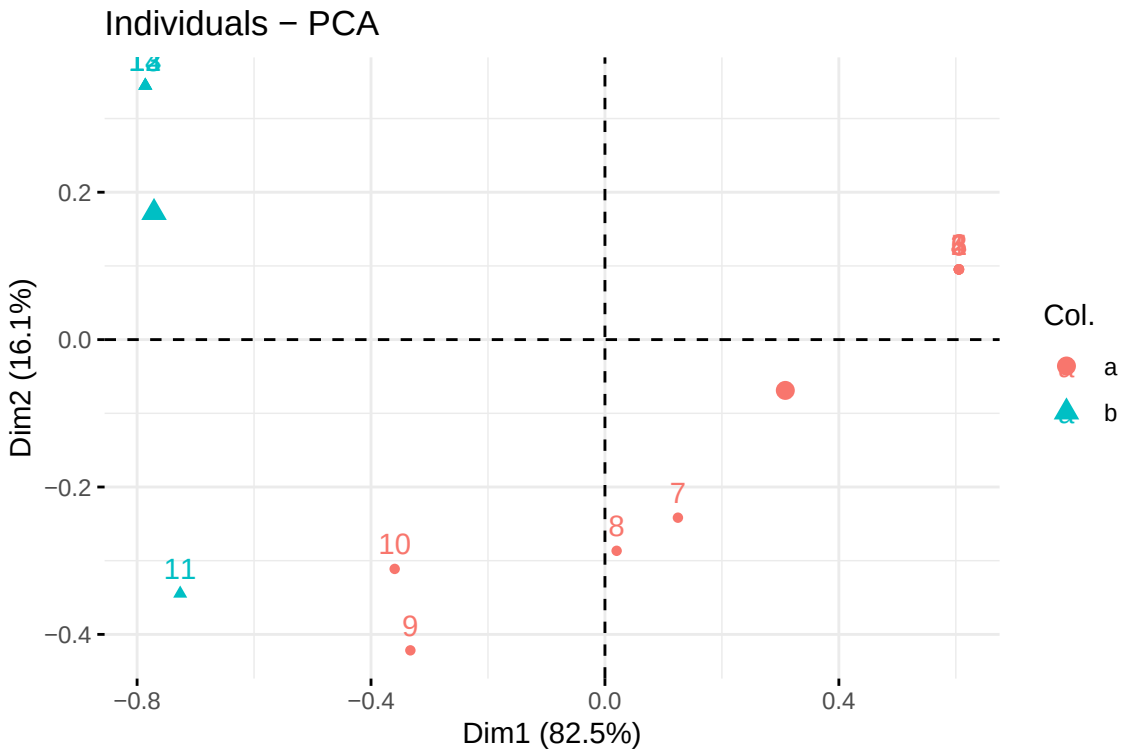
```
                # plot samples
                    # regular pca
                        fviz_pca_ind(my_pca)
```



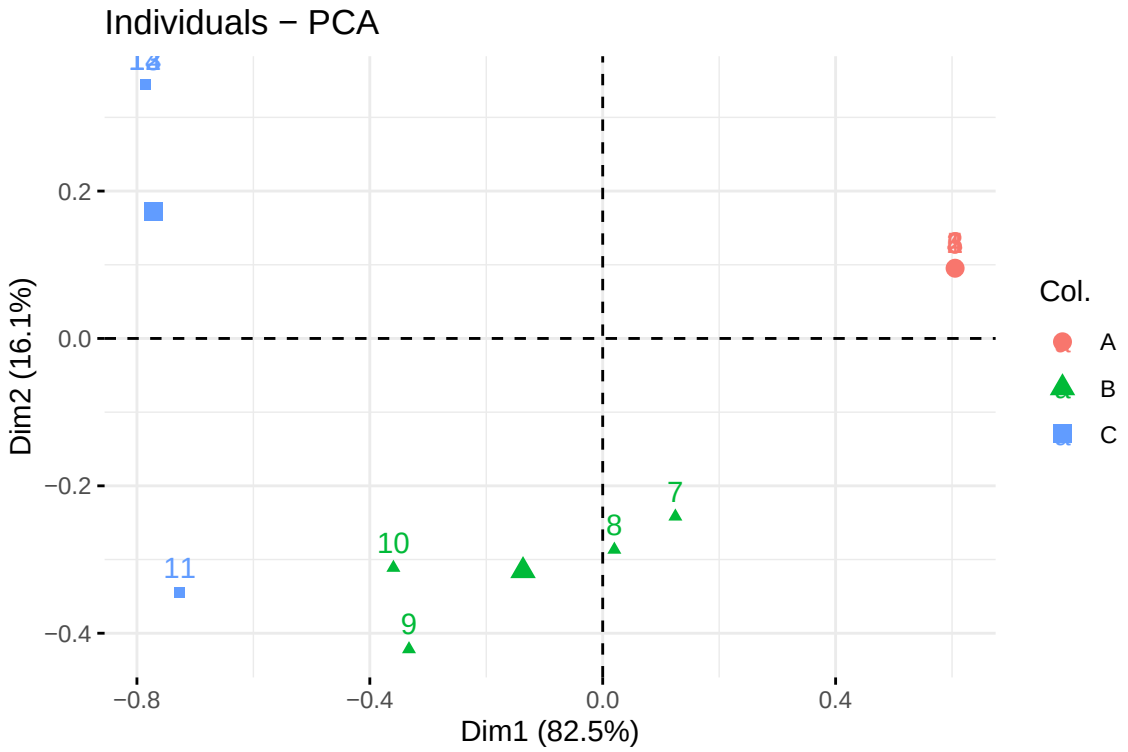

```
fviz_pca_ind(my_pca, col.ind = as.factor(X$lac_ID))
```




```
# hellinger pca
fviz_pca_ind(my_pca_hell, col.ind = as.factor(X$region))
```

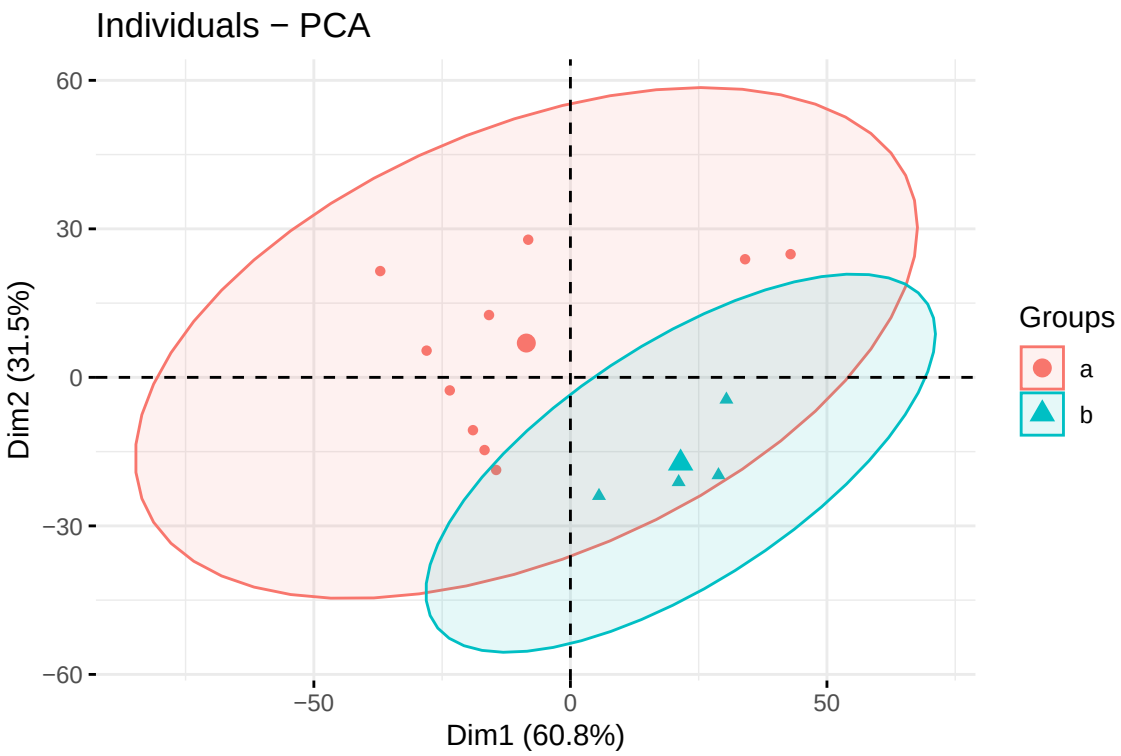



```
fviz_pca_ind(my_pca_hell, col.ind = as.factor(X$lac_ID))
```

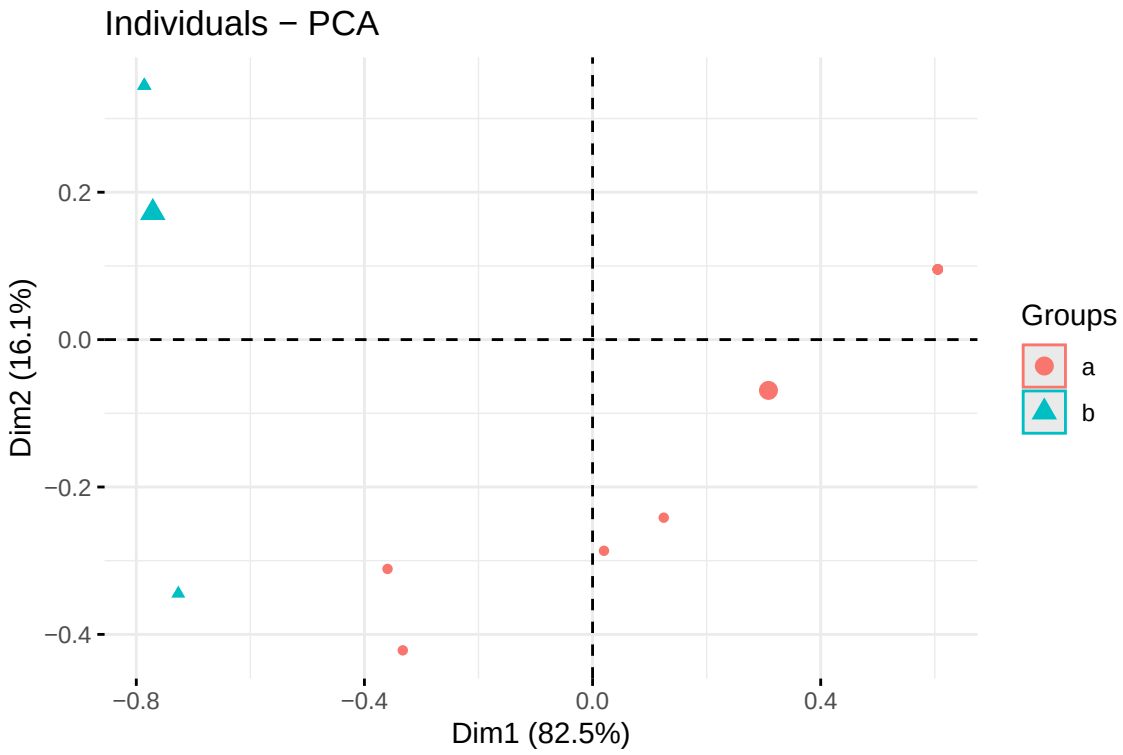



```
# ellipses by class
    # regular pca
        fviz_pca_ind(my_pca,
```

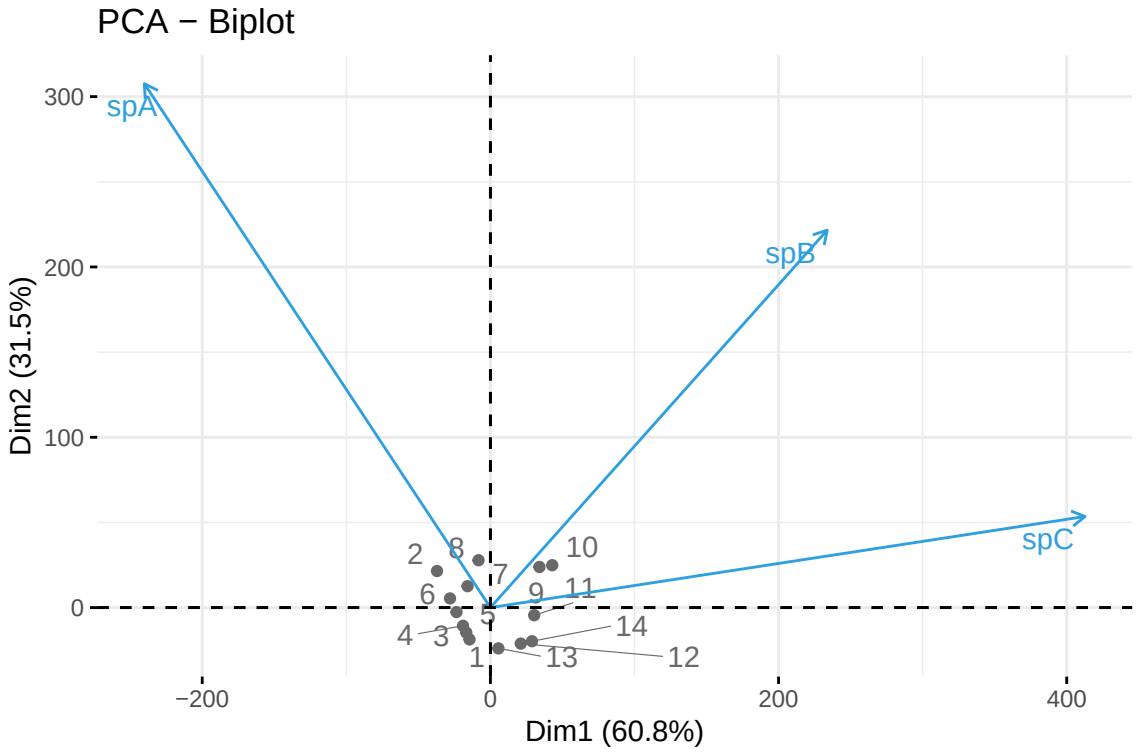
```
                label = "none", # hide individual labels
                habillage = X$region, # color by groups
                addEllipses = TRUE # Concentration ellipses
                )
```



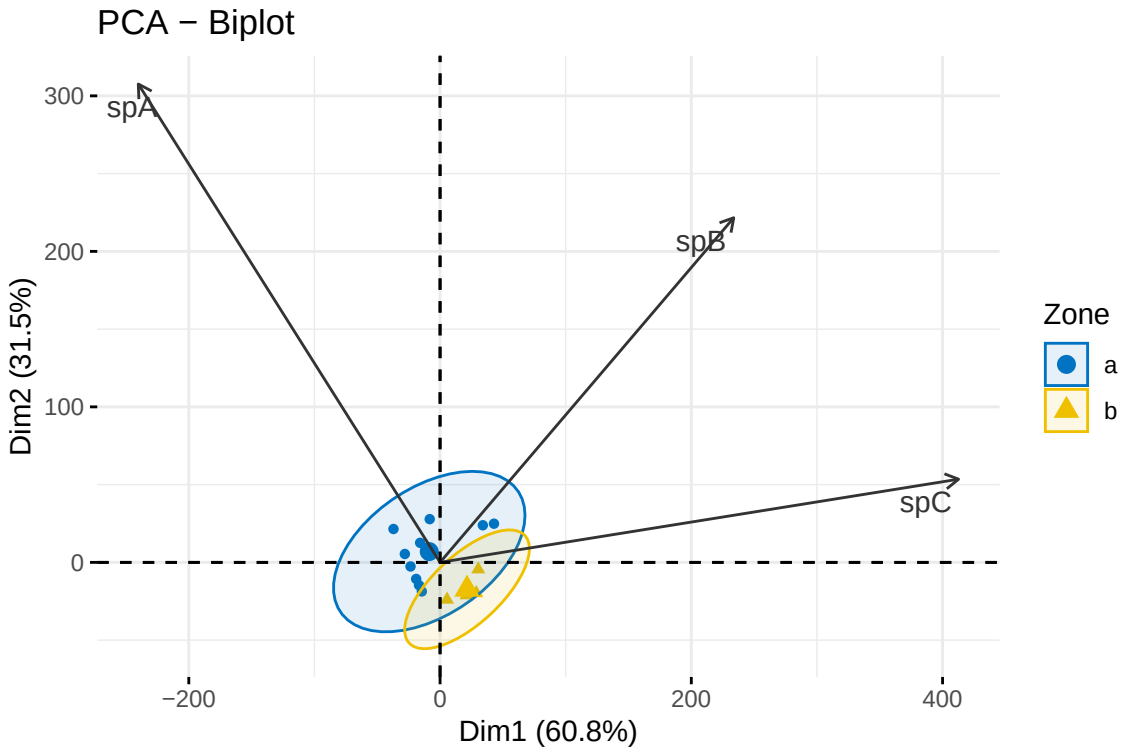

```
# hellinger pca
    fviz_pca_ind(my_pca_hell,
                 label = "none", # hide individual labels
                 habillage = X$region, # color by groups
                 addEllipses = TRUE # Concentration ellipses
                 )
```




```
# biplot : samples and variables in the same plan

    fviz_pca_biplot(my_pca, repel = TRUE,
                    col.var = "#2E9FDF", # Couleur des variables
                    col.ind = "#696969"  # Couleur des individues
                    )
```

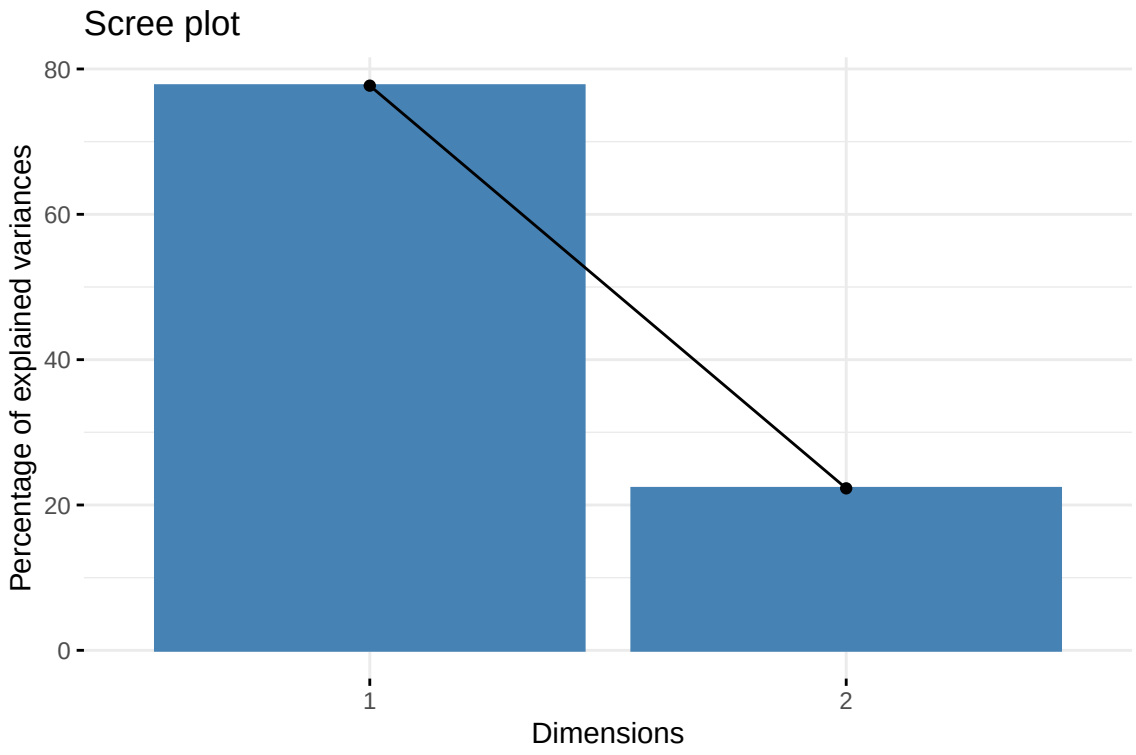


```
# with grouping
    fviz_pca_biplot (my_pca,
                     col.ind = X$region, palette = "jco",
                     addEllipses = TRUE, label = "var",
                     col.var = "grey20", repel = TRUE,
                     legend.title = "Zone")
```



CA

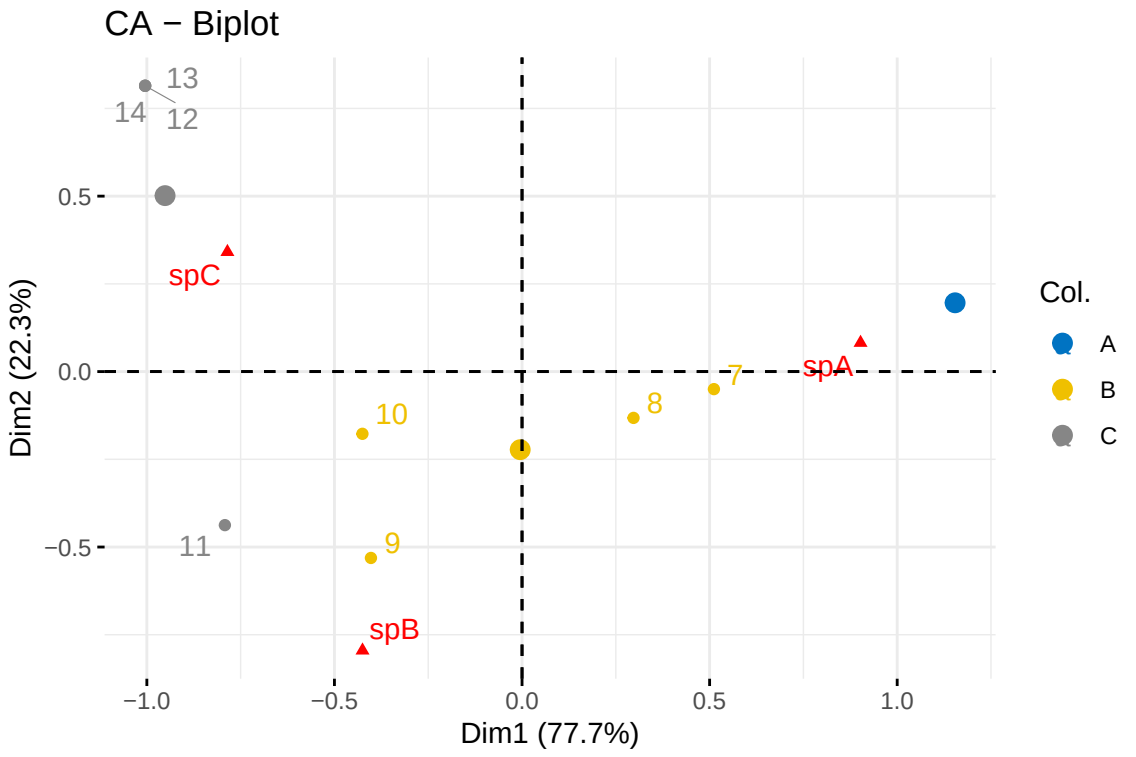
```
my_ca <- dudi.coa( X[,idx_sp], scannf = FALSE, nf = 2)
fviz_eig(my_ca)
```




```
fviz_ca_biplot(my_ca, col.row = X$lac_ID, palette = "jco", repel = TRUE)
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```



NMDS

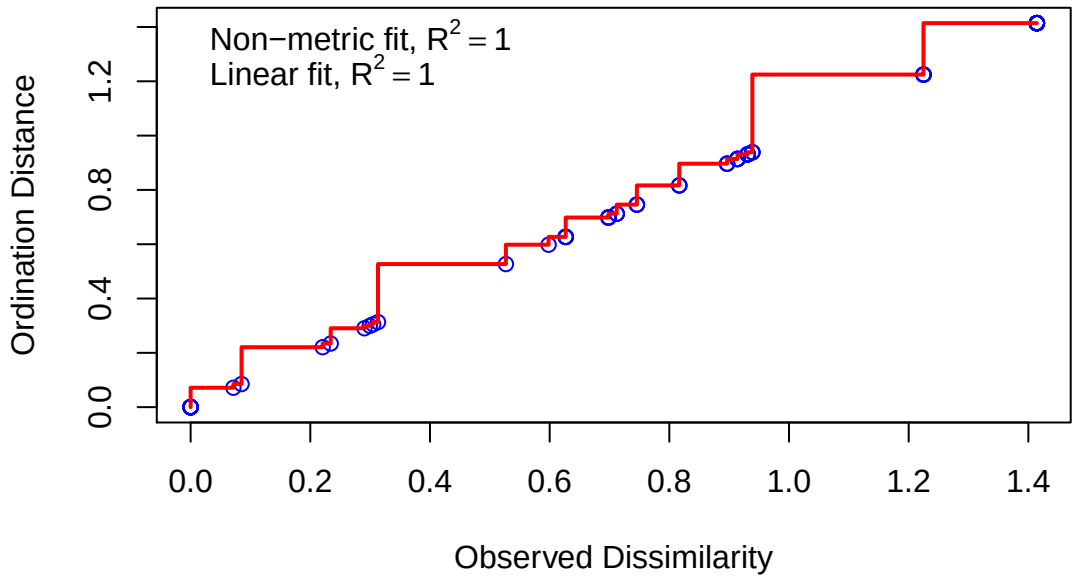
Projection based on ranks

```
        library(vegan)

            set.seed(156) # for reproducibility
            my_NMDS <- metaMDS(X[,idx_sp],k=2, distance = 'euclidian', try  = 40)
```

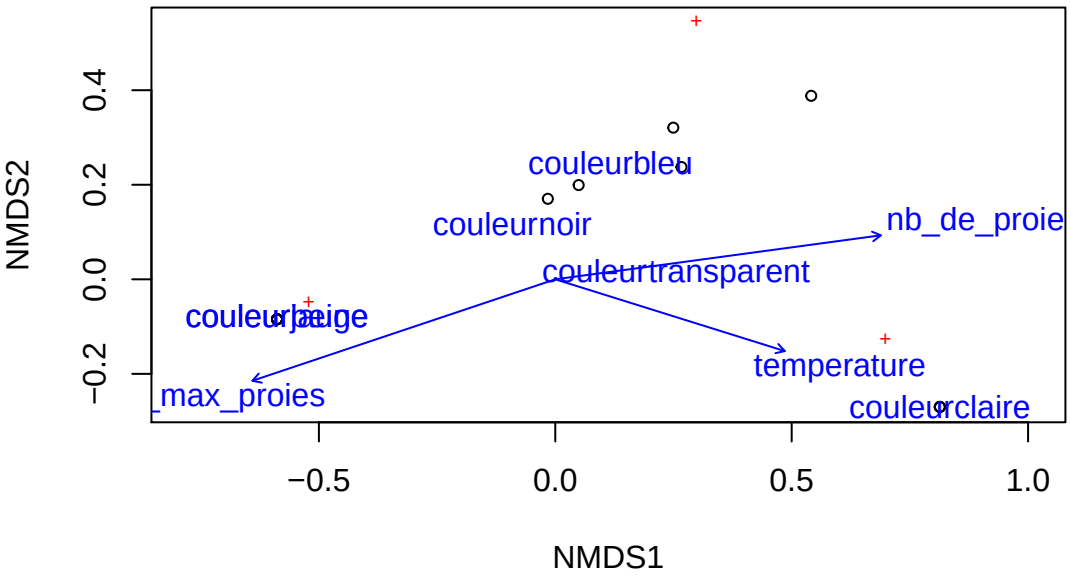
```
## Square root transformation
## Wisconsin double standardization
## Run 0 stress 1.098836e-16
## Run 1 stress 0.0001792619
## ... Procrustes: rmse 0.1004401  max resid 0.1762309
## Run 2 stress 0.001412527
## Run 3 stress 9.786874e-05
## ... Procrustes: rmse 0.08953175  max resid 0.1469207
## Run 4 stress 0.0001893011
## ... Procrustes: rmse 0.08531323  max resid 0.25042
## Run 5 stress 9.854552e-05
## ... Procrustes: rmse 0.09782977  max resid 0.1724606
## Run 6 stress 0.0001504108
## ... Procrustes: rmse 0.06658247  max resid 0.1338225
## Run 7 stress 9.735297e-05
## ... Procrustes: rmse 0.06173418  max resid 0.1161699
## Run 8 stress 9.854143e-05
## ... Procrustes: rmse 0.08450149  max resid 0.254246
## Run 9 stress 9.552966e-05
## ... Procrustes: rmse 0.1029779  max resid 0.259549
## Run 10 stress 9.721278e-05
## ... Procrustes: rmse 0.0415004  max resid 0.0823927
## Run 11 stress 9.292348e-05
## ... Procrustes: rmse 0.07740056  max resid 0.1416923
## Run 12 stress 9.475959e-05
## ... Procrustes: rmse 0.02877601  max resid 0.05136328
## Run 13 stress 0.2629862
## Run 14 stress 9.956576e-05
## ... Procrustes: rmse 0.1048407  max resid 0.2393921
## Run 15 stress 8.92502e-05
## ... Procrustes: rmse 0.04335137  max resid 0.08800629
## Run 16 stress 9.926463e-05
## ... Procrustes: rmse 0.09381226  max resid 0.1610475
## Run 17 stress 9.419578e-05
## ... Procrustes: rmse 0.04162864  max resid 0.08151304
## Run 18 stress 9.713507e-05
## ... Procrustes: rmse 0.0933731  max resid 0.277694
## Run 19 stress 6.743785e-05
## ... Procrustes: rmse 0.03509975  max resid 0.06475752
## Run 20 stress 9.736286e-05
## ... Procrustes: rmse 0.0468005  max resid 0.08732262
## Run 21 stress 9.558222e-05
## ... Procrustes: rmse 0.08472925  max resid 0.1337588
## Run 22 stress 9.997065e-05
## ... Procrustes: rmse 0.07336918  max resid 0.1335628
## Run 23 stress 9.865702e-05
```

```
## ... Procrustes: rmse 0.09582977  max resid 0.2734865
## Run 24 stress 9.800318e-05
## ... Procrustes: rmse 0.0271833  max resid 0.0596122
## Run 25 stress 9.819355e-05
## ... Procrustes: rmse 0.04559812  max resid 0.07840705
## Run 26 stress 9.905424e-05
## ... Procrustes: rmse 0.07842365  max resid 0.1373529
## Run 27 stress 9.823161e-05
## ... Procrustes: rmse 0.02264789  max resid 0.06057948
## Run 28 stress 9.478853e-05
## ... Procrustes: rmse 0.0904561  max resid 0.161567
## Run 29 stress 9.996787e-05
## ... Procrustes: rmse 0.01902668  max resid 0.0425469
## Run 30 stress 0.0002457178
## ... Procrustes: rmse 0.0838454  max resid 0.1364648
## Run 31 stress 9.352473e-05
## ... Procrustes: rmse 0.1030368  max resid 0.2830842
## Run 32 stress 9.626785e-05
## ... Procrustes: rmse 0.05529618  max resid 0.1041941
## Run 33 stress 9.927033e-05
## ... Procrustes: rmse 0.05182149  max resid 0.09264297
## Run 34 stress 9.956597e-05
## ... Procrustes: rmse 0.05144992  max resid 0.09920007
## Run 35 stress 9.668104e-05
## ... Procrustes: rmse 0.09707998  max resid 0.2768351
## Run 36 stress 8.306372e-05
## ... Procrustes: rmse 0.03212628  max resid 0.07887744
## Run 37 stress 8.088222e-05
## ... Procrustes: rmse 0.05412741  max resid 0.1047771
## Run 38 stress 9.946154e-05
## ... Procrustes: rmse 0.09291569  max resid 0.1601825
## Run 39 stress 9.053205e-05
## ... Procrustes: rmse 0.04383347  max resid 0.08917077
## Run 40 stress 9.78441e-05
## ... Procrustes: rmse 0.08844009  max resid 0.1575658
## *** No convergence -- monoMDS stopping criteria:
##      5: no. of iterations >= maxit
##     34: stress < smin
##      1: stress ratio > sratmax
```

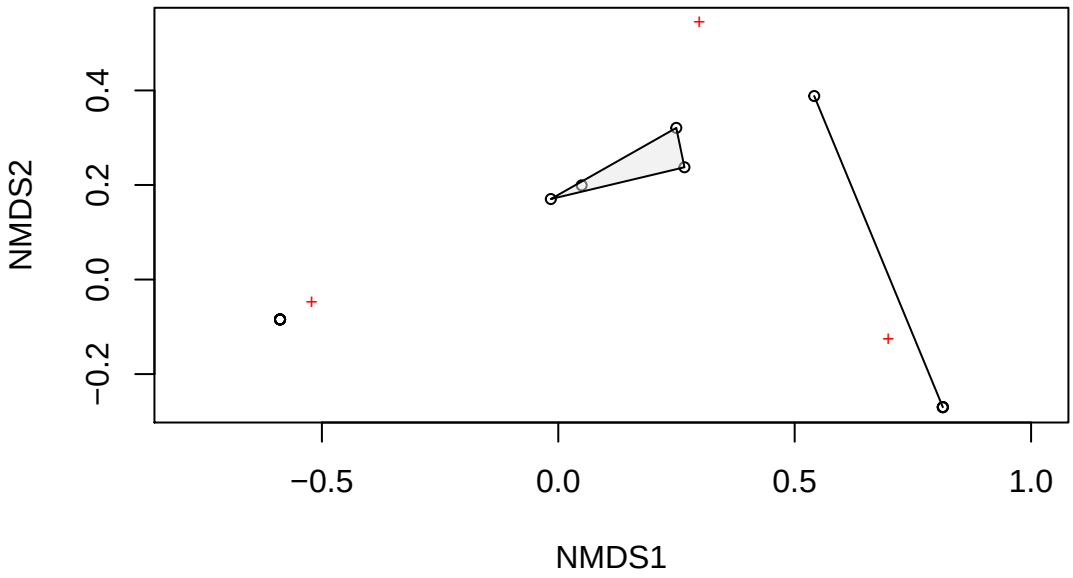
```r
# quality of projection (stress)
    stressplot(my_NMDS)
```



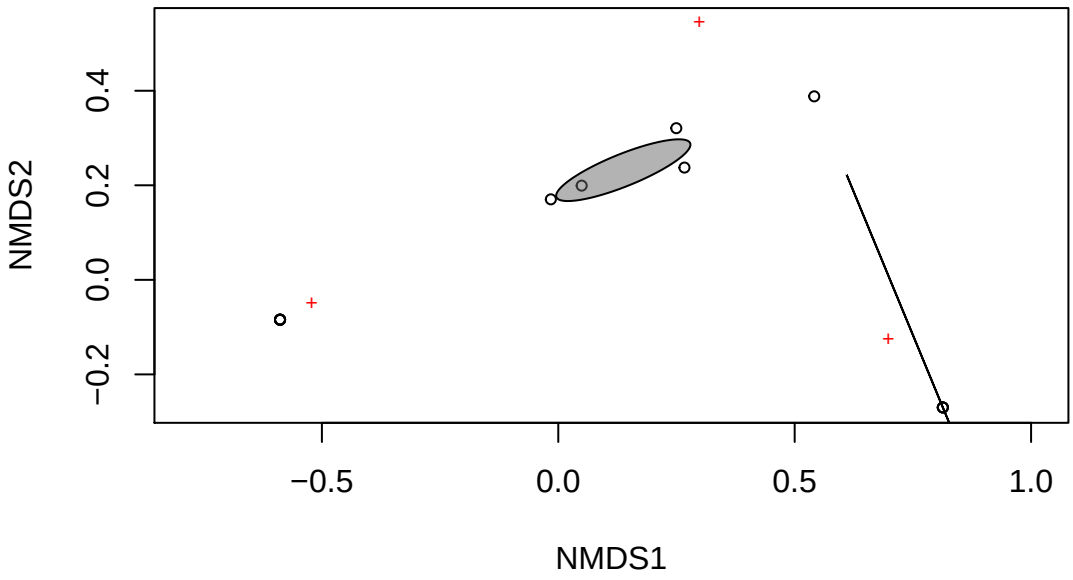

```
# 2D plan
    ordiplot(my_NMDS)

# add a continous covariates
    covariates <- envfit (my_NMDS, X[, 5:8])
    ordiplot(my_NMDS)
    plot(covariates)
```




```
# add convex hulls
    ordiplot(my_NMDS)
    ordihull(my_NMDS, groups = X$lac_ID, draw="polygon",
             col="grey90", label = F)
```




```
# add ellipses
    ordiplot(my_NMDS)
    ordiellipse(my_NMDS, groups = X$lac_ID, draw="polygon",
                col=c("grey10","grey40","grey90"), label = F)
```




```
    #library(ggvegan)
    #autoplot(my_NMDS)
```

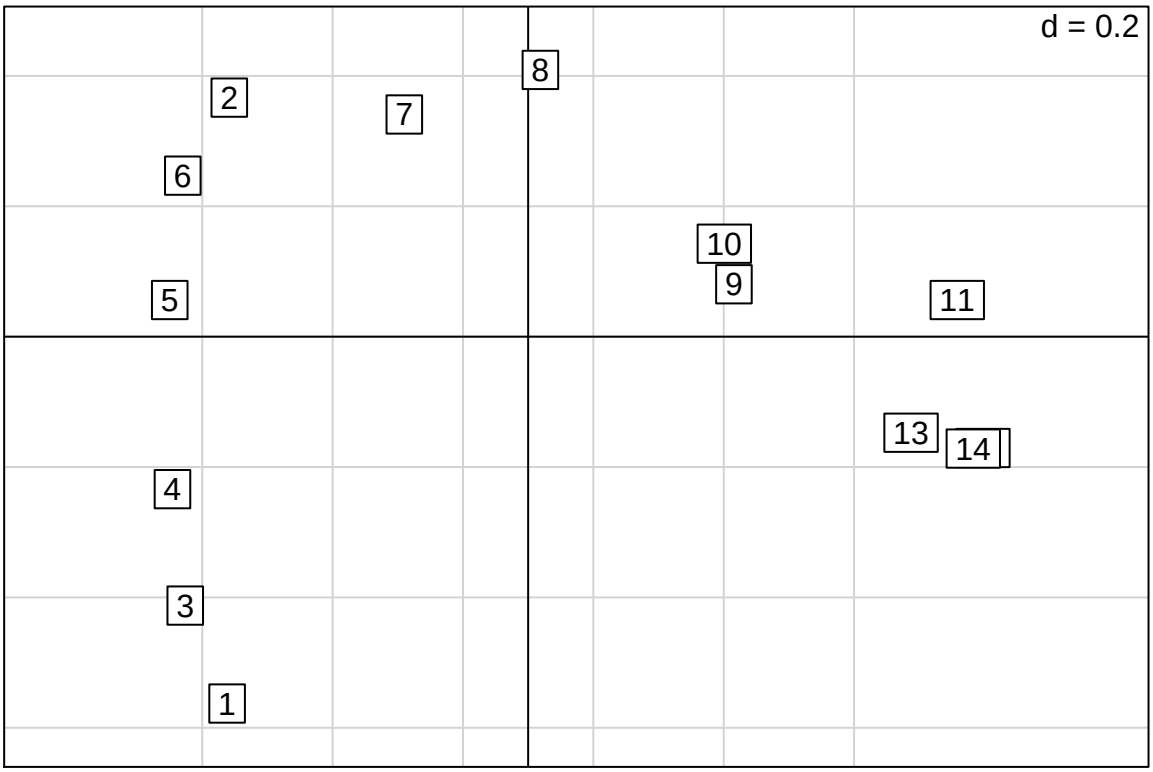
PCoa (= MDS)

Any metric distance

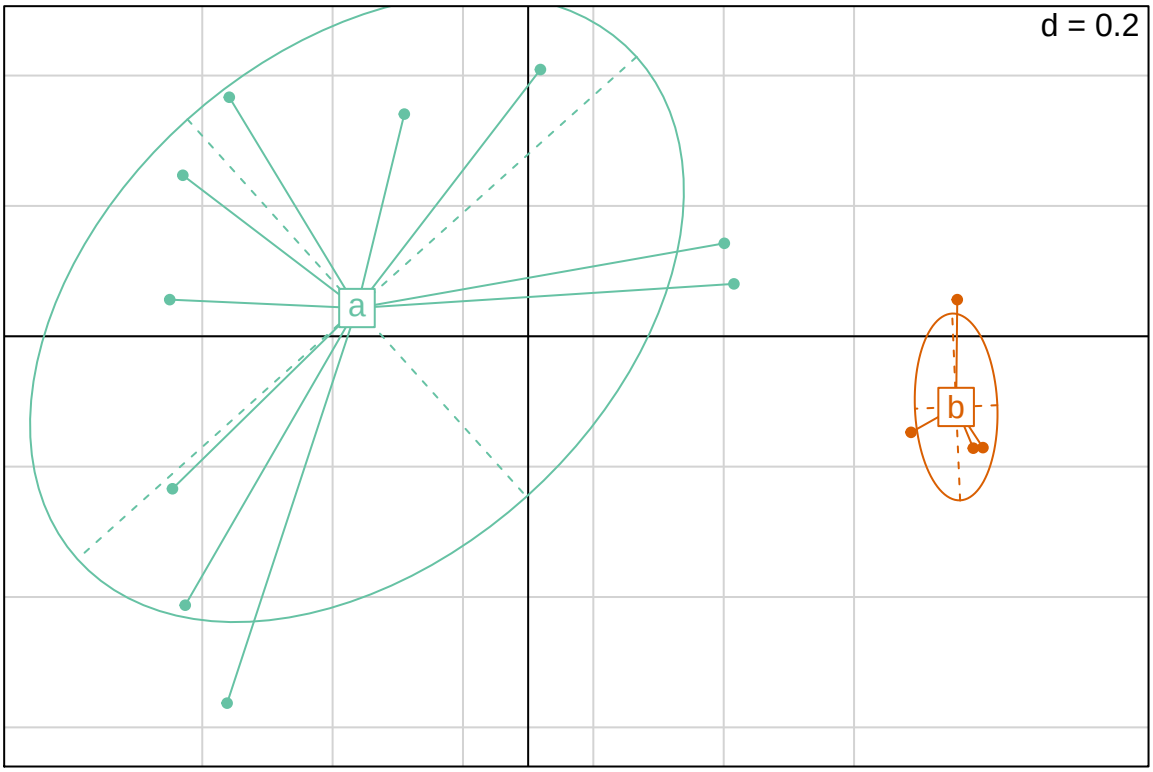
```
    dist_bray <- vegdist(X[,idx_sp], "bray")
# check that dist matrix is euclidian
    is.euclid(dist_bray)
```

```
## [1] FALSE
```

```r
# transform to euclidian matrix
    dist_bray_euclid <- cailliez(dist_bray)
# pCoa
    my_pcoa <- dudi.pco(dist_bray_euclid, scannf = FALSE)

s.label(my_pcoa$li)
```




```r
s.class(my_pcoa$li, factor(X$region), col = c( '#66c2a4', '#d95f02'))
```



Kernel PCA

```
            library(kernlab)
```

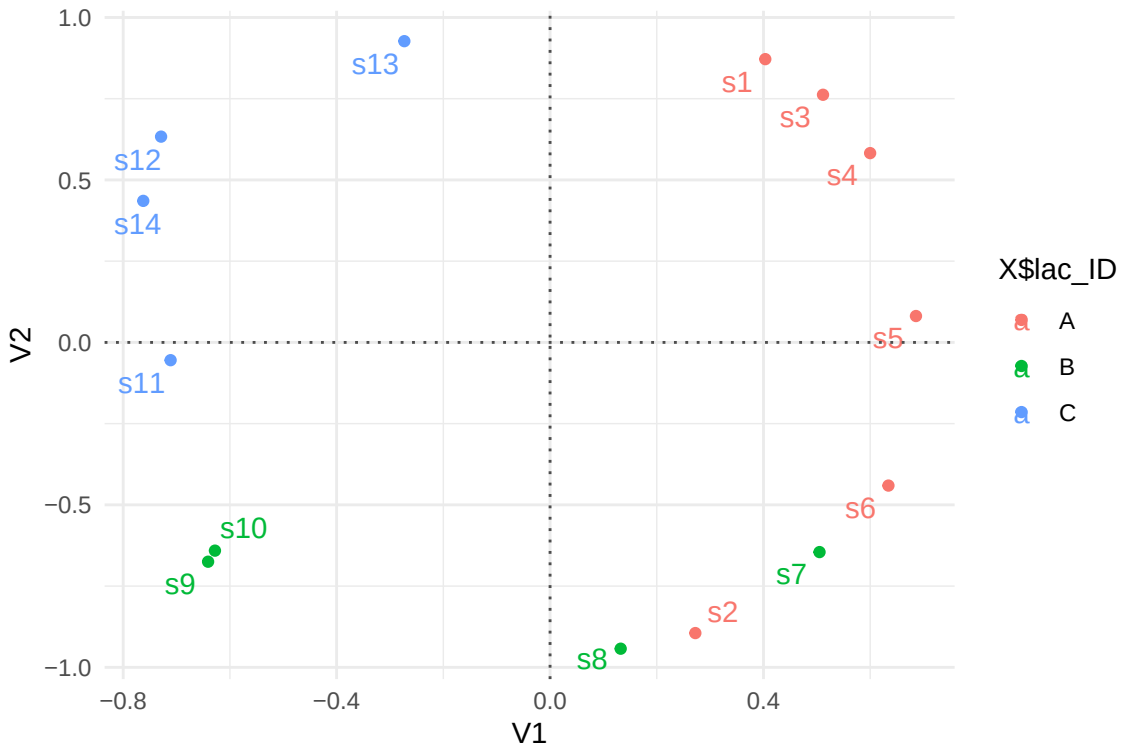
```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
## The following object is masked from 'package:permute':
##
##     how
```

```
            k_pca <- kpca(~., data = X[,idx_sp], kernel = "rbfdot",
                          kpar = list(sigma=0.001), features = 2)

            # get principal component vectors
                k_pca_v <- pcv(k_pca)
                kpca_v_df <- as.data.frame(k_pca_v)

            # plot
                library(ggplot2)
                library(ggrepel)

                ggplot(kpca_v_df, aes(x = V1, y = V2, col = X$lac_ID)) + geom_point() +
                          geom_text_repel(label = X$sple_ID) +
                          geom_hline(yintercept = 0, linetype = "dotted", color = "grey30") +
                          geom_vline(xintercept = 0, linetype = "dotted", color = "grey30") +
```

```
theme_minimal()
```



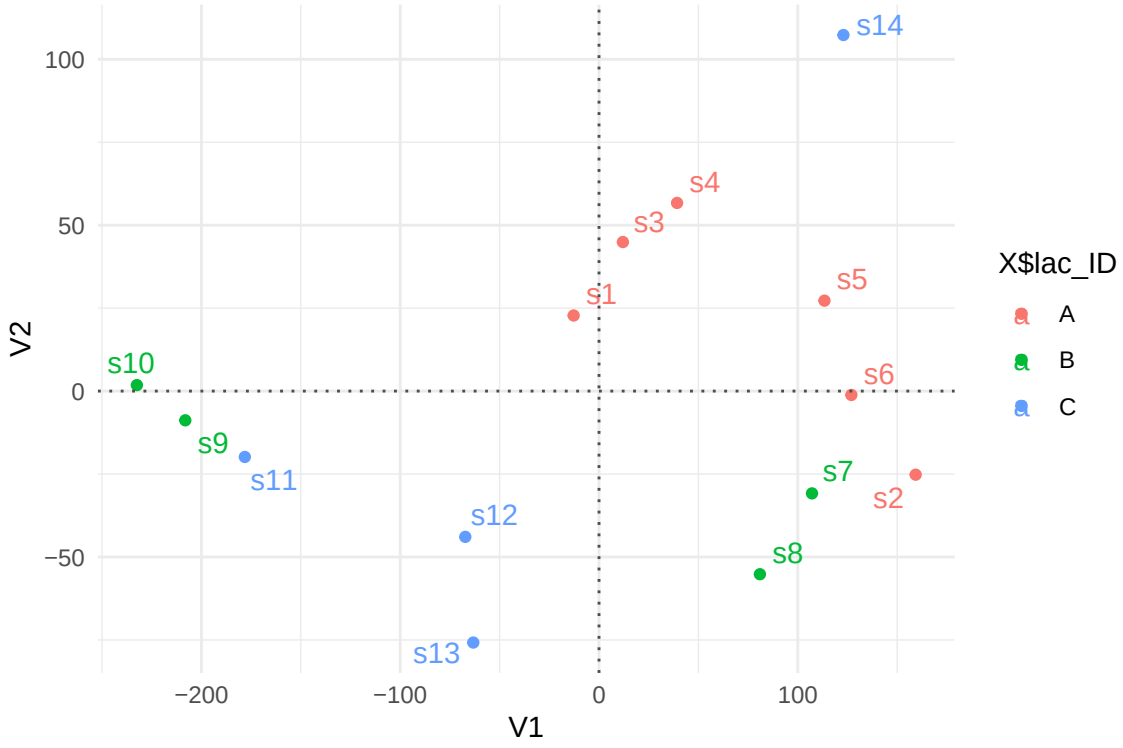
t sne

```
library(Rtsne)

my_tsne <- Rtsne(X[,idx_sp], dims = 2, perplexity = 2, verbose = TRUE, max_iter = 500)
```

```
## Performing PCA
## Read the 14 x 3 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 2.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.00 seconds (sparsity = 0.530612)!
## Learning embedding...
## Iteration 50: error is 66.676221 (50 iterations in 0.00 seconds)
## Iteration 100: error is 52.610902 (50 iterations in 0.00 seconds)
## Iteration 150: error is 61.418478 (50 iterations in 0.01 seconds)
## Iteration 200: error is 52.822730 (50 iterations in 0.01 seconds)
## Iteration 250: error is 61.402287 (50 iterations in 0.00 seconds)
## Iteration 300: error is 1.574219 (50 iterations in 0.02 seconds)
## Iteration 350: error is 1.121256 (50 iterations in 0.00 seconds)
## Iteration 400: error is 1.048211 (50 iterations in 0.01 seconds)
## Iteration 450: error is 0.943382 (50 iterations in 0.01 seconds)
## Iteration 500: error is 0.645730 (50 iterations in 0.00 seconds)
## Fitting performed in 0.07 seconds.
```

```
my_tsne_rows <- as.data.frame(my_tsne$Y)
```

```
                ggplot(my_tsne_rows, aes(x = V1, y = V2, col = X$lac_ID)) + geom_point() +
                           geom_text_repel(label = X$sple_ID) +
                           geom_hline(yintercept = 0, linetype = "dotted", color = "grey30") +
                           geom_vline(xintercept = 0, linetype = "dotted", color = "grey30") +
                           theme_minimal()
```



SOM

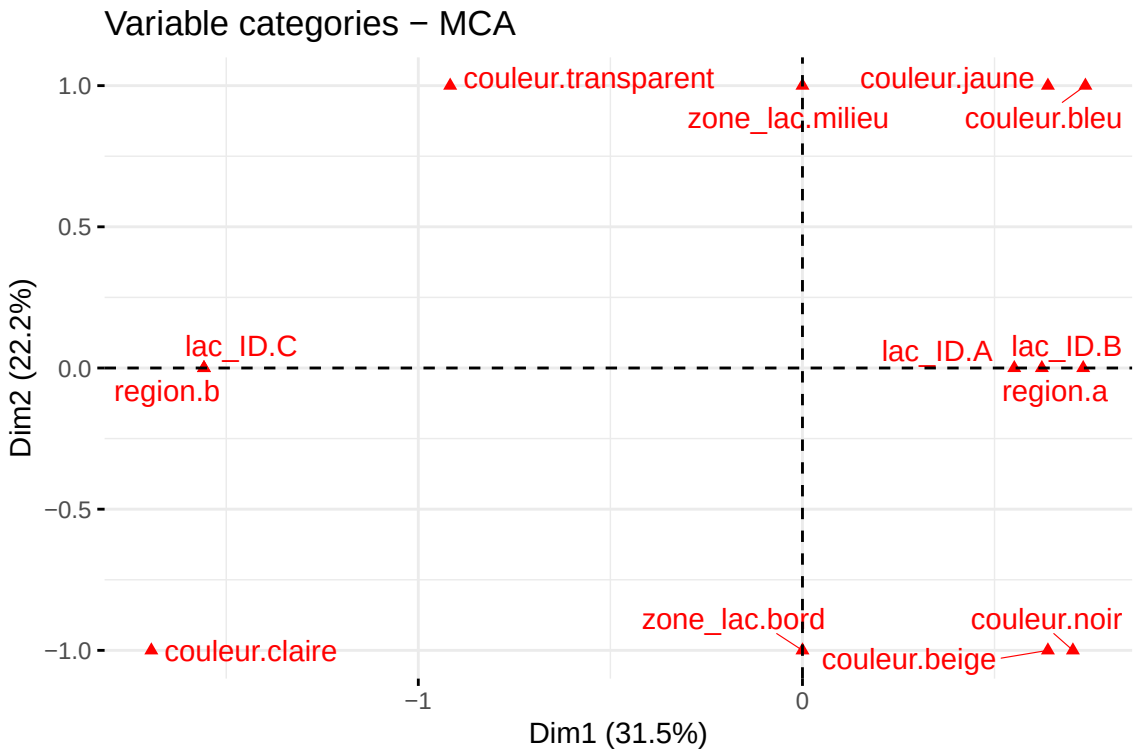
todo

Ordination of categorical data

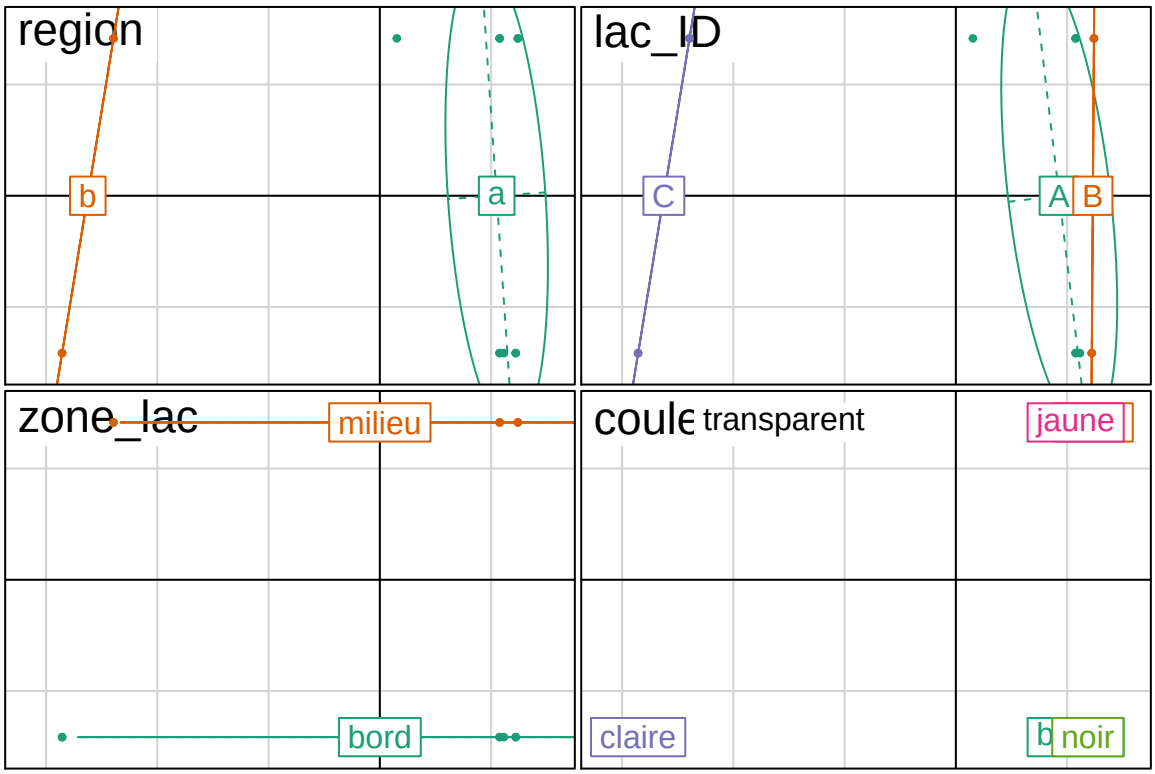
MCA

link

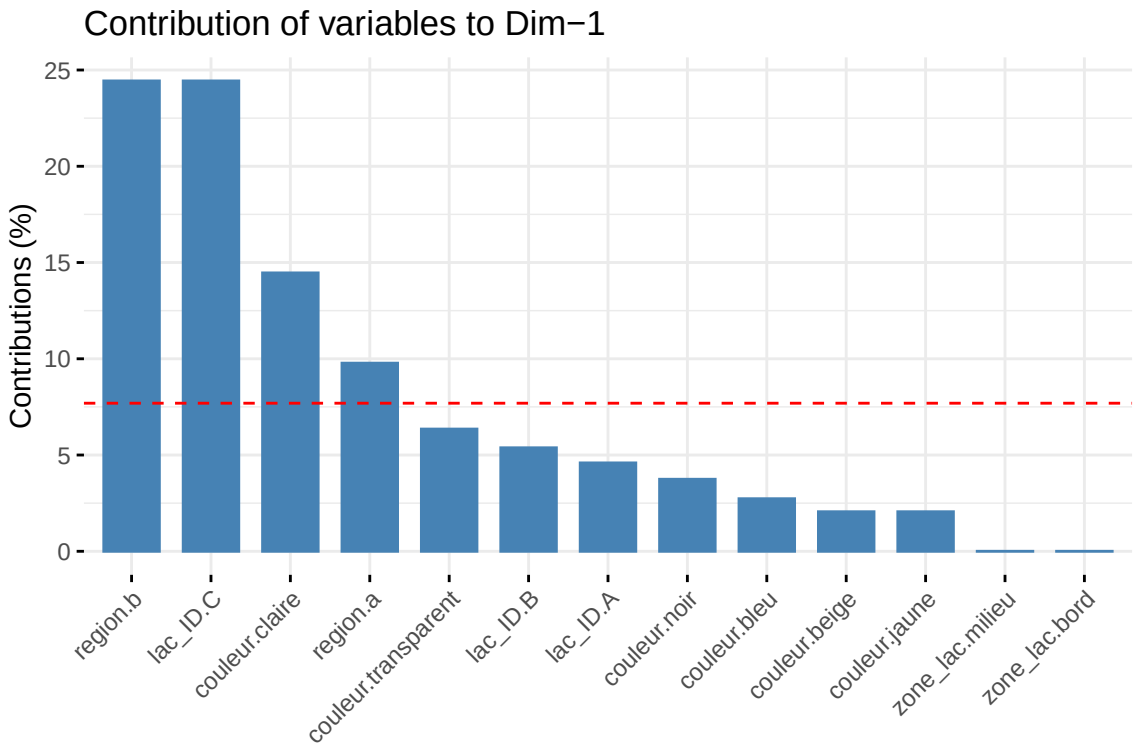
```
        X$region <- factor(X$region)
        X$lac_ID <- factor(X$lac_ID)
        X$zone_lac <- factor(X$zone_lac)
        X$couleur <- factor(X$couleur)
        my_mca <- dudi.acm(X[, 2:5], scannf = FALSE, nf = 3)

        # plot of modalities
            fviz_mca_var(my_mca, repel = TRUE)
```



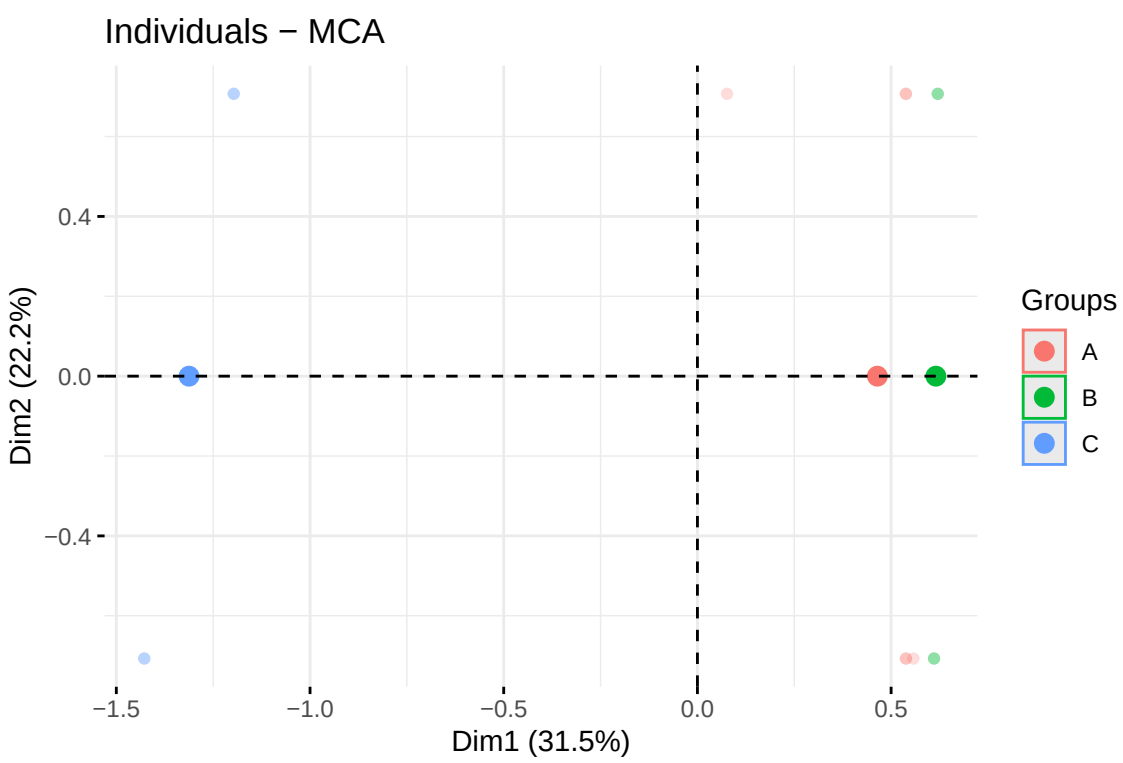

```
library(RColorBrewer)
scatter(my_mca,col = brewer.pal(5, "Dark2"))
```




```
# contribution of modalites to axis 1
    fviz_contrib(my_mca, choice = "var", axes = 1)
```




```
# plot of samples
    fviz_mca_ind(my_mca, geom = "point", alpha.ind = .25, habillage = X$lac_ID, addEllipses = T
```



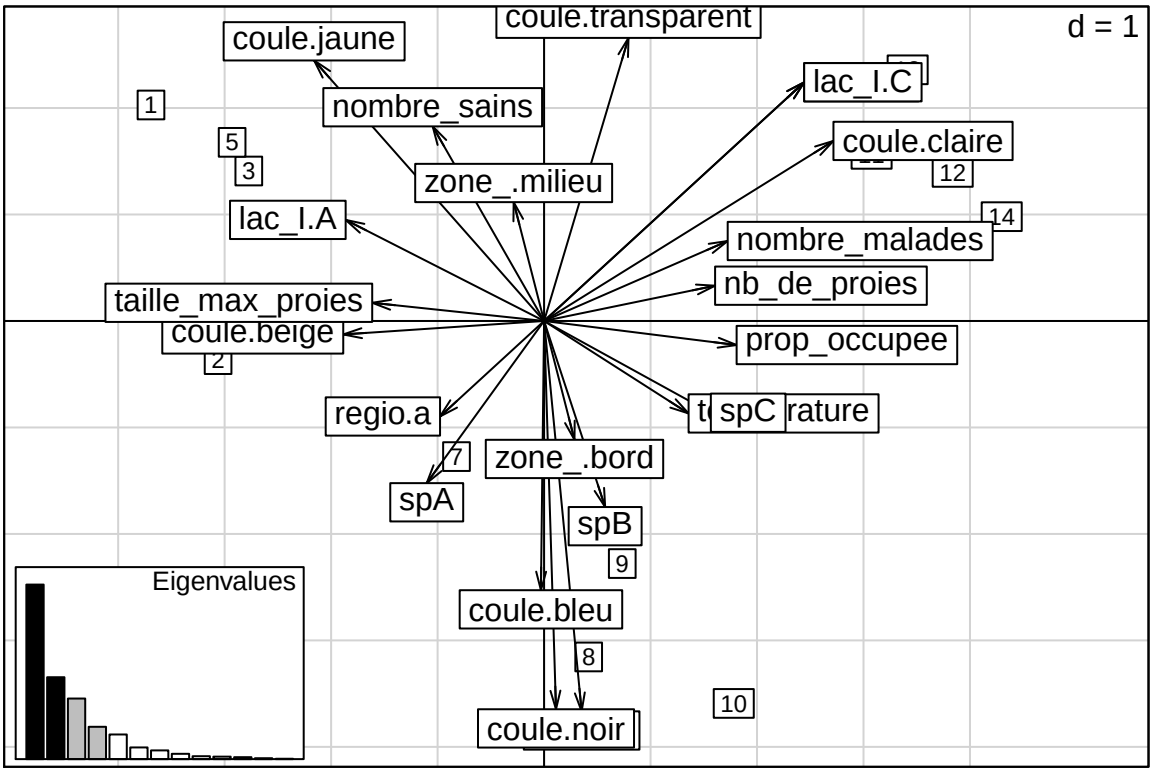
Ordination of mixed data

The differences are that dudi.hillsmith allow to use various row weights, while dudi.mix deals with ordered variables.

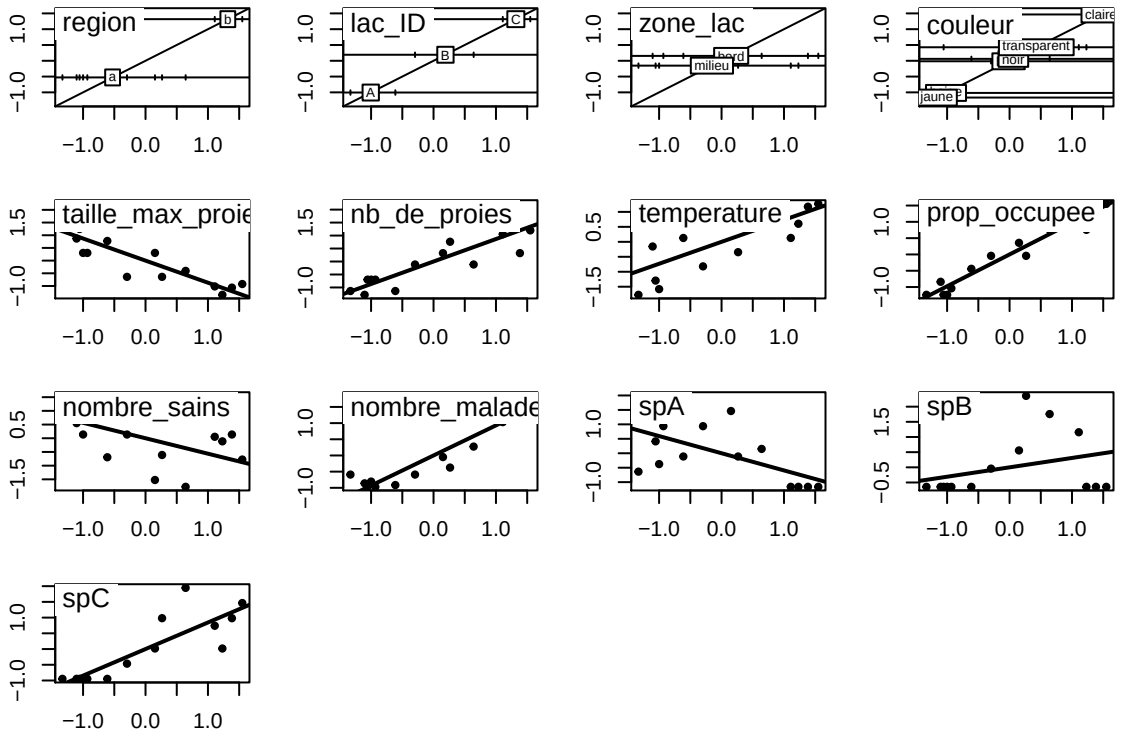
Hill-Smith (=FAMD)

link

```
my_HS <- dudi.hillsmith(X[,-1], scannf = FALSE, nf = 4)
scatter(my_HS, posieig = "bottomleft")
```

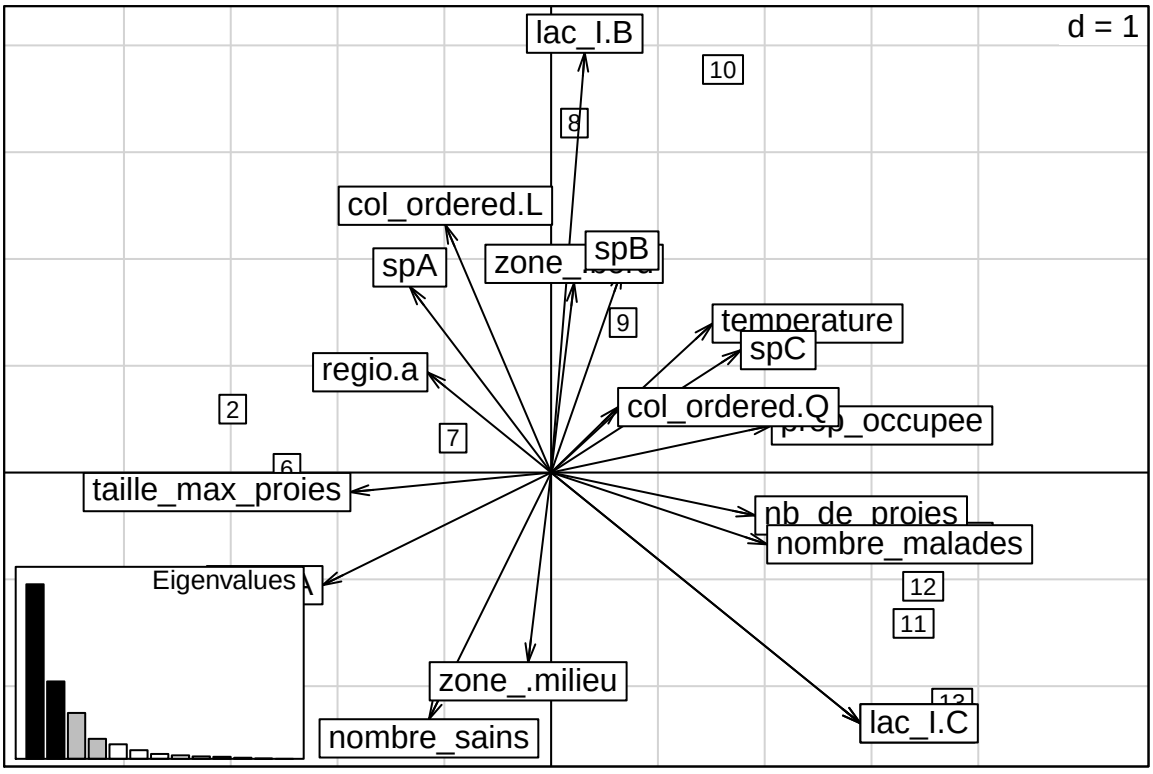



```
score(my_HS, col = TRUE)
```



Mixed type analysis

```
X$col_ordered <- factor(X$couleur, ordered = TRUE,
                        levels = c("transparent","claire", "bleu", "jaune", "beige", "noir" ))

my_dmix <- dudi.mix(X[,-grep("sple_ID|couleur$", names(X))], scannf = FALSE, nf = 4)
scatter(my_dmix, posieig = "bottomleft")
```




```
score(my_dmix, col = TRUE)
```