

Working notes

ggplot2 implementation of the graphical functions of the ade4 package (in working!)

P.BADY

March 31, 2021

License: GPL version 2 or newer
Copyright (C) 2000-2021 Pierre Bady
This program/document is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
This program/document is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

Contents

1	Motivations	2
2	Gestion of the limits	2
3	Representations of the variables	2
4	Representations of samples	5
5	Representations of samples by group	6
5.1	separated Representations of samples by group	6
5.2	Ellipses	9
5.3	Ellipses and stars	10
5.4	Chull representation	12
6	2D density	15

7	Adding a picture	17
8	Complex figure with the package cowplot	18
9	Conclusion	18
10	Appendix	18

1 Motivations

The objectives of this document are to propose elements and alternatives for `ggplot2` implementation of the graphical function from R package ADE-4 ([1, 2, 3]). why `ggplot2` ?

2 Gestion of the limits

A function `getLimits` is written to extract the limits of the axes x and y as done in the R package `ade4`.

```
getLimits <- function (dfxy, xax=1, yax=2,include.origin=TRUE,origin=c(0,0)){
  df <- data.frame(dfxy)
  if (!is.data.frame(df))
    stop("Non convenient selection for df")
  if ((xax < 1) || (xax > ncol(df)))
    stop("Non convenient selection for xax")
  if ((yax < 1) || (yax > ncol(df)))
    stop("Non convenient selection for yax")
  x <- df[, xax]
  y <- df[, yax]
  x1 <- x
  if (include.origin)
    x1 <- c(x1, origin[1])
  x1 <- c(x1 - diff(range(x1))/10), x1 + diff(range(x1))/10)
  xlim <- range(x1)

  y1 <- y
  if (include.origin)
    y1 <- c(y1, origin[2])
  y1 <- c(y1 - diff(range(y1))/10), y1 + diff(range(y1))/10)
  ylim <- range(y1)
  return(list(xlim=xlim, ylim=ylim))
}
```

Example for a futur prototype of the function `ggade` (or `ggscatter`?)

```
require(ggplot2)
ggade <- function(dfxy,xax=1,yax=2,...,include.origin=TRUE,origin=c(0,0)){
  xlim <- getLimits(dfxy,xax=xax,yax=yax,include.origin=include.origin,origin=origin)
  ggplot(...) + coord_cartesian(xlim=xlim$xlim,ylim=ylim$ylim) + coord_fixed(ratio=1)
}
```

3 Representations of the variables

```
data(deug)
deug0 <- dudi.pca(deug$tab, center = deug$cent, scale = FALSE, scan = FALSE)
deug1 <- dudi.pca(deug$tab, center = TRUE, scale = TRUE, scan = FALSE)
```

```
require(ggplot2)
gg <- ggplot(data=data.frame(eig=deug1$eig,nf=1:length(deug1$eig)), aes(x=nf, y=eig))
gg <- gg + geom_bar(stat="identity") + ggtitle("Eigenvalues from PCA")
gg <- gg + ylab("Eigenvalues") + xlab("axis") + theme_light()
gg
```

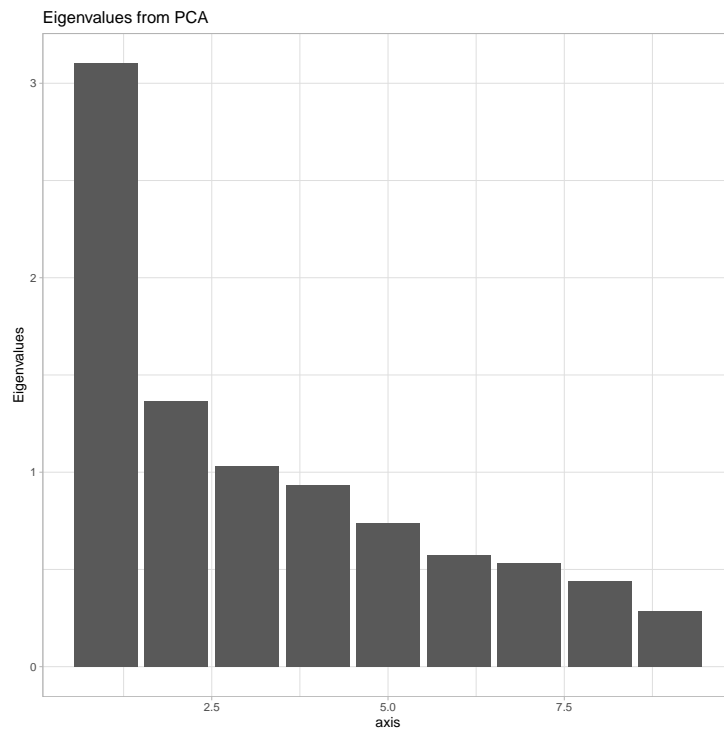


Figure 1: Representation of the Eigenvalues from PCA on correlation matrix

```

require(cowplot)
require(ggplot2)
require(ggrepel)
auxi <- deug0$co
auxi$label <- rownames(auxi)
ggx <- ggplot(data=auxi,aes(Comp1,Comp2,label=label) )
ggx <- ggx + geom_hline(yintercept = 0)+geom_vline(xintercept = 0)
ggx <- ggx + xlab("axis 1") + ylab("axis 2")
ggx <- ggx + geom_segment(aes(x=0,xend =Comp1, y=0,yend = Comp2),arrow = arrow(length = unit(0.2,"cm")))
ggx <- ggx + geom_label_repel(size = 3.5,segment.alpha=0.7,segment.color = "darkgrey",
                             fontface = 'bold',col="black")
ggx <- ggx + theme_bw() + coord_fixed(ratio=1)
ggx

```

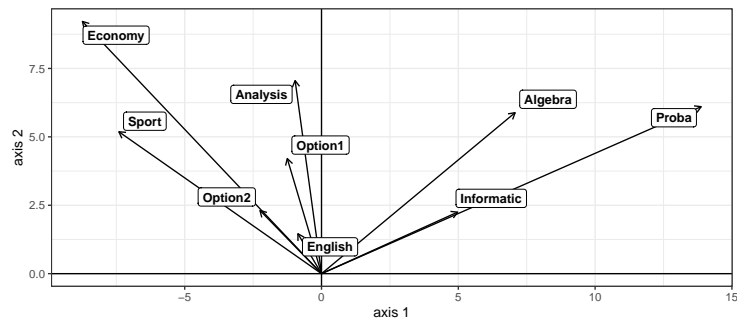


Figure 2: Representation of the variables with correlation circle for PCA on covariance

```

require(cowplot)
require(ggplot2)
require(ggrepel)
require(ggforce)
auxi <- deug1$co
auxi$label <- rownames(auxi)
ggx <- ggplot(data=auxi,aes(Comp1,Comp2,label=label) )
ggx <- ggx + geom_hline(yintercept = 0)+geom_vline(xintercept = 0)
ggx <- ggx + xlab("axis 1") + ylab("axis 2")
ggx <- ggx + geom_circle(data=data.frame(x0=0,y0=0),aes(x0=x0, y0=y0, r=1),inherit.aes = FALSE)
ggx <- ggx + geom_segment(aes(x=0,xend =Comp1, y=0,yend = Comp2),arrow = arrow(length = unit(0.2,"cm")))
ggx <- ggx + geom_label_repel(size = 3.5,segment.alpha=0.7,segment.color = "darkgrey",fontface = 'bold',col="")
ggx <- ggx + theme_bw() + coord_fixed(ratio=1)
ggx

```

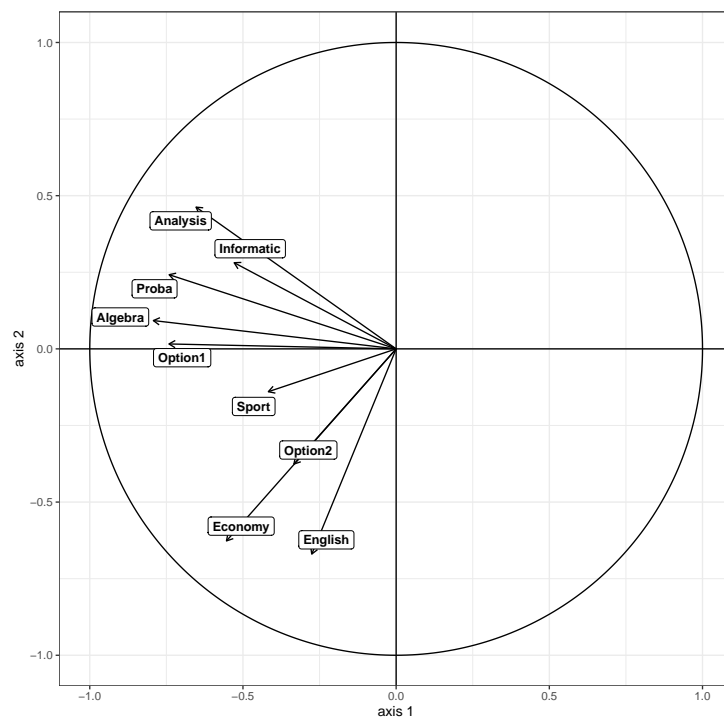


Figure 3: Representation of the variables with correlation circle for PCA on correlation

4 Representations of samples

representation of the samples on the first factorial plan

```

require(ggplot2)
require(ggrepel)
auxi <- deugi$li
auxi$label <- rownames(auxi)
gg <- ggplot(auxi,aes(Axis1,Axis2,label=label))
gg <- gg + geom_hline(yintercept = 0) + geom_vline(xintercept = 0)
gg <- gg + geom_point(shape=19,size=3)
gg <- gg + geom_label_repel(size = 3,segment.alpha=0.7,segment.color = "darkgrey")
gg <- gg + theme_bw()+ coord_fixed(ratio=1)
gg

```

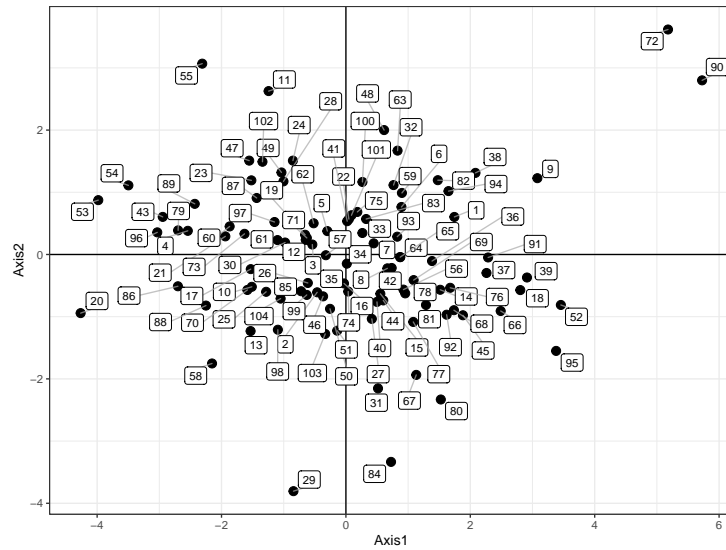


Figure 4: Representation of the observation of the first vectorial plan

5 Representations of samples by group

5.1 separated Representations of samples by group

Representation of the samples on the first factorial plan

```

require(ggplot2)
require(ggrepel)
auxi <- deugi$li
auxi$label <- rownames(auxi)
auxi$group <- substring(as.character(deug$result),1,1)
gg <- ggplot(auxi,aes(Axis1,Axis2,label=label))
gg <- gg + geom_hline(yintercept = 0) + geom_vline(xintercept = 0)
gg <- gg + geom_point(shape=19,size=3)
gg <- gg + geom_label_repel(size = 3,segment.alpha=0.7,segment.color = "darkgrey")
gg <- gg + theme_bw() + coord_fixed(ratio=1) + facet_wrap(~ group)
gg

```

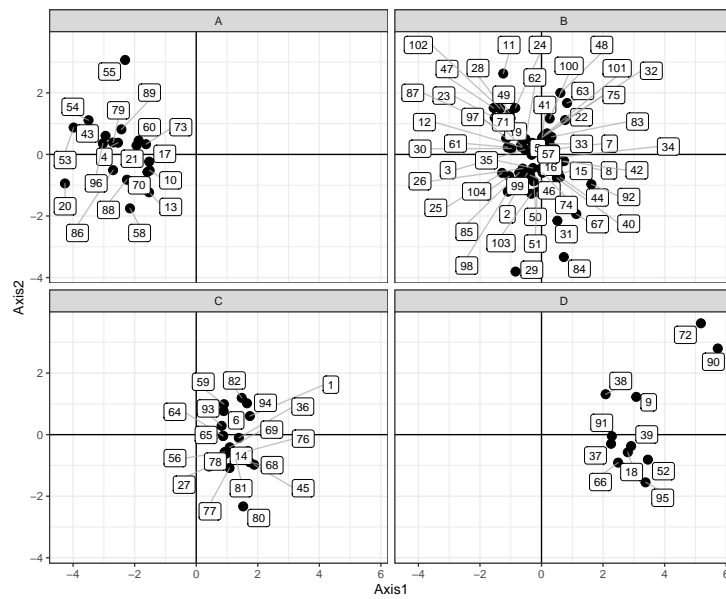


Figure 5: Representation of the observation of the first vectorial plan stratified by group.

```

require(ggplot2)
require(ggrepel)
auxi <- deugi$li
auxi$label <- rownames(auxi)
auxi$group <- substring(as.character(deug$result),1,1)
gg <- ggplot(auxi,aes(Axis1,Axis2,label=label,color=group))
gg <- gg + geom_hline(yintercept = 0) + geom_vline(xintercept = 0)
gg <- gg + geom_point(shape=19,size=3)
gg <- gg + geom_label_repel(size = 3,segment.alpha=0.7,segment.color = "darkgrey")
gg <- gg + theme_bw() + coord_fixed(ratio=1) + facet_wrap(~ group)
gg

```

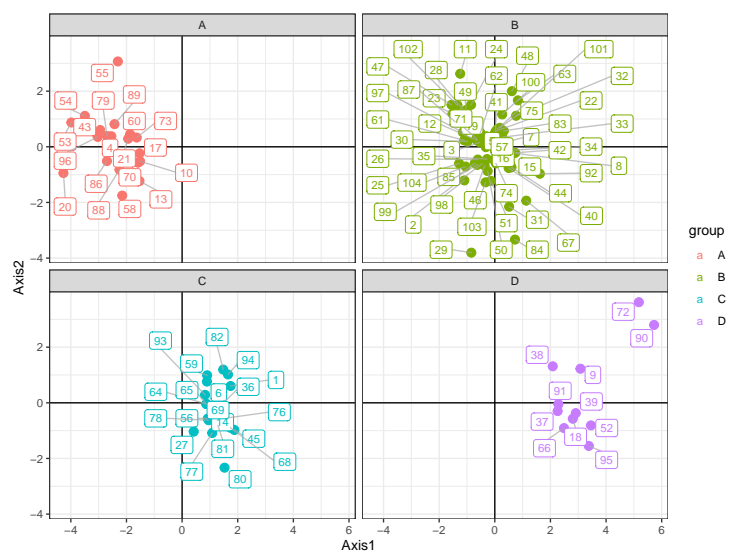


Figure 6: Representation of the observation of the first vectorial plan stratified by group (with color).

5.2 Ellipses

```
require(ggplot2)
require(ggrepel)
auxi <- deug1$li
auxi$label <- rownames(auxi)
auxi$group <- substring(as.character(deug$result),1,1)
gg <- ggplot(auxi,aes(Axis1,Axis2,label=label,color=group))
gg <- gg + geom_hline(yintercept = 0) + geom_vline(xintercept = 0)
gg <- gg + geom_point(shape=19,size=3)
gg <- gg + stat_ellipse(type = "norm",level=0.66)
gg <- gg + geom_label_repel(size = 3,segment.alpha=0.7,segment.color = "darkgrey")
gg <- gg + theme_bw() + coord_fixed(ratio=1)
gg
```

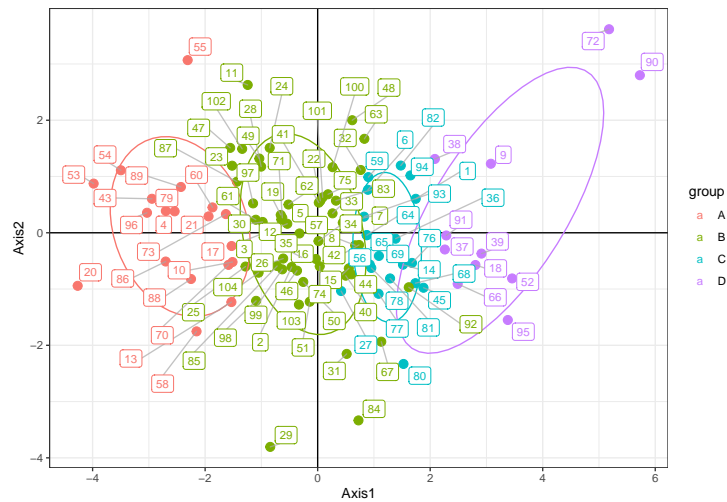


Figure 7: Representation of the observation of the first vectorial plan with ellipse of Inertia for each group

```

require(ggplot2)
require(ggrepel)
auxi <- deugi$li
auxi$label <- rownames(auxi)
auxi$group <- substring(as.character(deug$result),1,1)
gg <- ggplot(auxi,aes(Axis1,Axis2,label=label,color=group))
gg <- gg + geom_hline(yintercept = 0) + geom_vline(xintercept = 0)
gg <- gg + geom_point(shape=19,size=3)
gg <- gg + stat_ellipse(type = "norm",level=0.66)
gg <- gg + geom_label_repel(size = 3,segment.alpha=0.7,segment.color = "darkgrey")
gg <- gg + theme_bw() + coord_fixed(ratio=1) + facet_wrap(~group)
gg

```

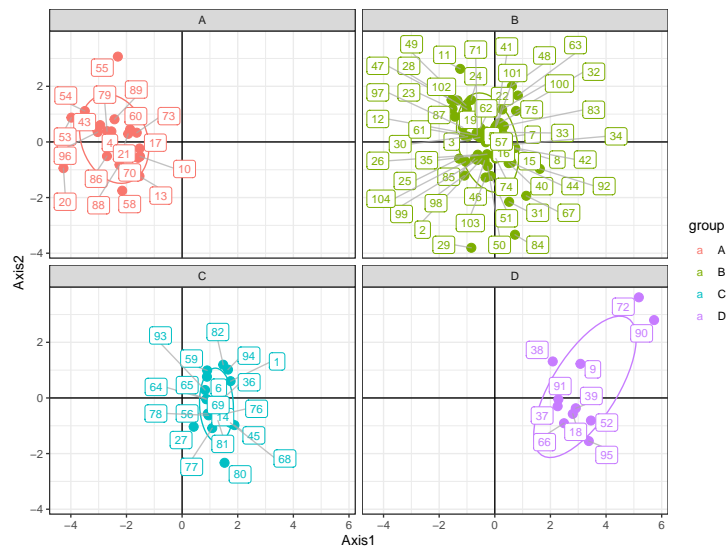


Figure 8: Representation of the observation of the first vectorial plan with ellipse of Inertia for each group

5.3 Ellipses and stars

For this representation, we need to compute the position of the gravity center of the each ellipse. This operation was performed by the R function `prep.tab.class`. The implementation of this function is given below:

```

prep.tab.class <- function(x,fac,varnames=c("Axis1","Axis2"),rm.X=TRUE,...){
  x <- as.data.frame(x)
  x$fac <- factor(x[,fac])
  w <- tab.class(x[,varnames],fac=x$fac)
  colnames(w) <- c("Axis1","Axis2")
  if(rm.X){
    w$label <- gsub("X","",rownames(w)) #levels(x$fac)
  }else{
    w$label <- rownames(w)
  }
  rownames(w) <- as.character(w$label)
}

```

```

x$cAxis1 <- w$Axis1[match(as.character(x$fac),w$label)]
x$cAxis2 <- w$Axis2[match(as.character(x$fac),w$label)]
return(list(data=x,center=w))
}

require(ggplot2)
require(ggrepel)
auxi <- deug1$li
auxi$label <- rownames(auxi)
auxi$group <- substring(as.character(deug$result),1,1)
w <- prep.tab.class(auxi,varnames=c("Axis1","Axis2"),fac="group",rm.X=FALSE)
datax <- w$data
centerx <- w$center
gg <- ggplot(datax,aes(x=Axis1,y=Axis2,col=group,group=group))
gg <- gg + geom_hline(yintercept = 0) + geom_vline(xintercept = 0)
gg <- gg + stat_ellipse(type = "norm",level=0.66,lwd=1,show.legend = FALSE)
gg <- gg + geom_segment(data=datax,aes(x=Axis1,y=Axis2, xend = cAxis1, yend = cAxis2))
gg <- gg + geom_point(data=centerx,aes(x=Axis1,y=Axis2,col=label),inherit.aes = FALSE,size=0)
gg <- gg + geom_point(shape=19,size=2,show.legend = FALSE)
# gg <- gg + geom_label_repel(data=centerx,aes(x=Axis1,y=Axis2,col=label,label=label),size = 3,
#                             segment.alpha=0.7,segment.color = "darkgrey",fontface = 'bold',inherit.aes = FALSE)
gg <- gg + geom_label(data=centerx,aes(x=Axis1,y=Axis2,col=label,label=label),size = 3,
                     fontface = 'bold',inherit.aes = FALSE)
gg <- gg + theme_light() +theme(legend.position = "none") + coord_fixed(ratio=1)
gg

```

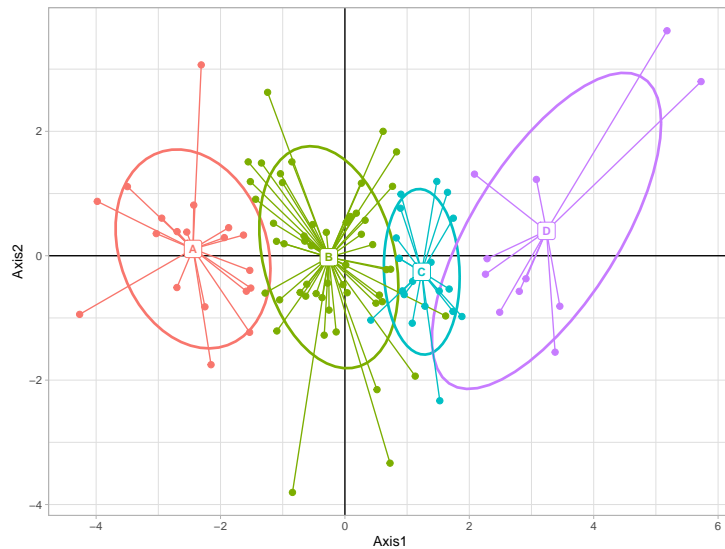


Figure 9: Representation of the observation of the first vectorial plan with ellipse of Inertia for each group

5.4 Chull representation

The following code is based on the vignette related to the extending ggplot2 functions (<https://cran.r-project.org/web/packages/ggplot2/vignettes/extending-ggplot2.htm>). The additional functions for the computation necessary to the Convex hull representation are given below:

```
StatChull <- ggproto("StatChull", Stat,
  compute_group = function(data, scales) {
    data[chull(data$x, data$y), , drop = FALSE]
  },
  required_aes = c("x", "y")
)
stat_chull <- function(mapping = NULL, data = NULL, geom = "polygon",
  position = "identity", na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE, ...) {
  layer(
    stat = StatChull, data = data, mapping = mapping, geom = geom,
    position = position, show.legend = show.legend, inherit.aes = inherit.aes,
    params = list(na.rm = na.rm, ...)
  )
}
```

The representation for one group is given below:

```

require(ggplot2)
# data
auxi <- deug1$li
auxi$label <- rownames(auxi)
auxi$group <- substring(as.character(deug$result),1,1)
# Find the convex hull of the points being plotted
# Define the scatterplot
gg <- ggplot(auxi,aes(Axis1,Axis2,col=group))
gg <- gg + geom_hline(yintercept = 0) + geom_vline(xintercept = 0)
gg <- gg + geom_point(shape=19,size=3)
gg <- gg + stat_chull(fill=NA)
gg <- gg + theme_bw() + coord_fixed(ratio=1)
gg

```

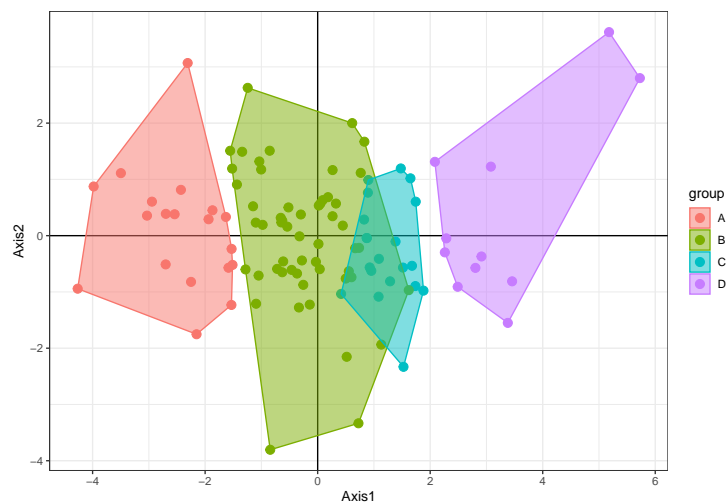


Figure 10: Chull representation of the observation of the first vectorial plan.

```

require(ggplot2)
# data
auxi <- deug1$li
auxi$label <- rownames(auxi)
auxi$group <- substring(as.character(deug$result),1,1)
# Find the convex hull of the points being plotted
# Define the scatterplot
gg <- ggplot(auxi,aes(Axis1,Axis2,col=group,fill=group))
gg <- gg + geom_hline(yintercept = 0) + geom_vline(xintercept = 0)
gg <- gg + geom_point(shape=19,size=3)
gg <- gg + stat_chull(alpha=0.5)
gg <- gg + theme_bw() + coord_fixed(ratio=1)
gg

```

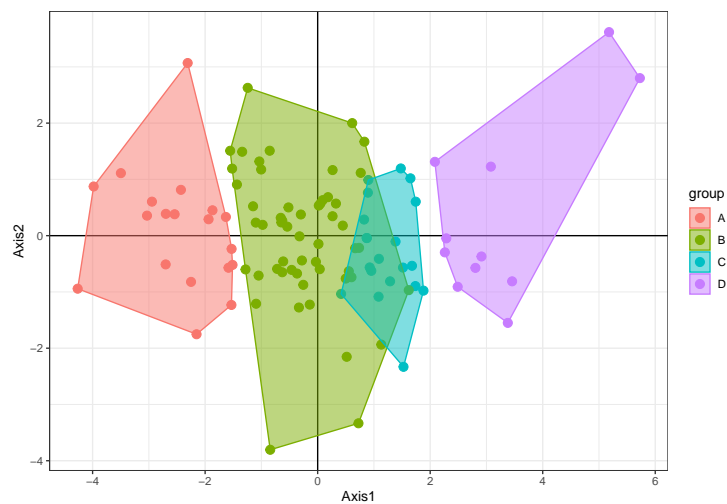


Figure 11: Chull representation of the observation of the first vectorial plan.

6 2D density

```
require(ggplot2)
require(ggrepel)
auxi <- deugi$li
auxi$label <- rownames(auxi)
auxi$group <- deug$result
gg <- ggplot(auxi,aes(Axis1,Axis2,label=label))
gg <- gg + geom_hline(yintercept = 0) + geom_vline(xintercept = 0)
gg <- gg + geom_point(shape=19,size=3)
gg <- gg + stat_density2d(col="blue")
gg <- gg + theme_bw() + coord_fixed(ratio=1)
gg
```

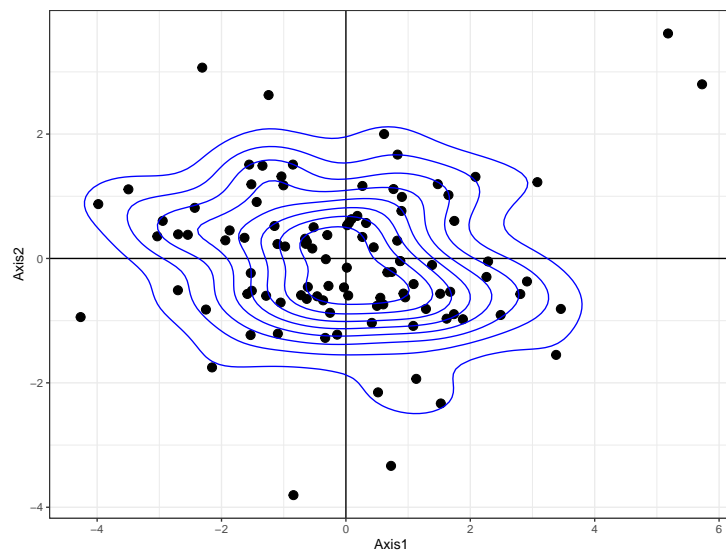


Figure 12: Representation of the observation with 2-D kernel density

```

require(ggplot2)
require(ggrepel)
auxi <- deug1$li
auxi$label <- rownames(auxi)
auxi$group <- substring(as.character(deug$result),1,1)
gg <- ggplot(auxi,aes(Axis1,Axis2,label=label,col=group))
gg <- gg + geom_hline(yintercept = 0) + geom_vline(xintercept = 0)
gg <- gg + geom_point(shape=19,size=3)
gg <- gg + stat_density2d(col="blue")
gg <- gg + theme_bw() + coord_fixed(ratio=1)
gg

```

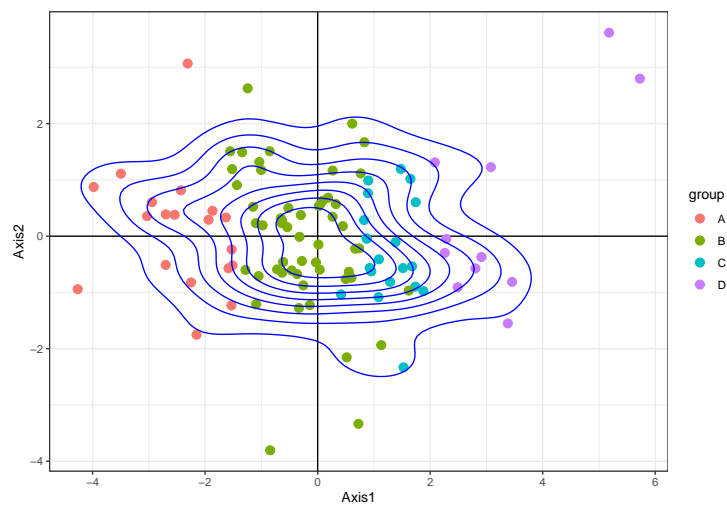


Figure 13: Representation of the observation with 2-D kernel density


```

require(ggplot2)
require(ggrepel)
auxi <- deugi$li
auxi$label <- rownames(auxi)
auxi$group <- substring(as.character(deug$result),1,1)
gg <- ggplot(auxi,aes(Axis1,Axis2,label=label,col=group))
gg <- gg + geom_hline(yintercept = 0) + geom_vline(xintercept = 0)
gg <- gg + geom_point(shape=19,size=3)
gg <- gg + stat_density2d(col="blue")
gg <- gg + theme_bw() + coord_fixed(ratio=1) + facet_wrap(~group)
gg

```

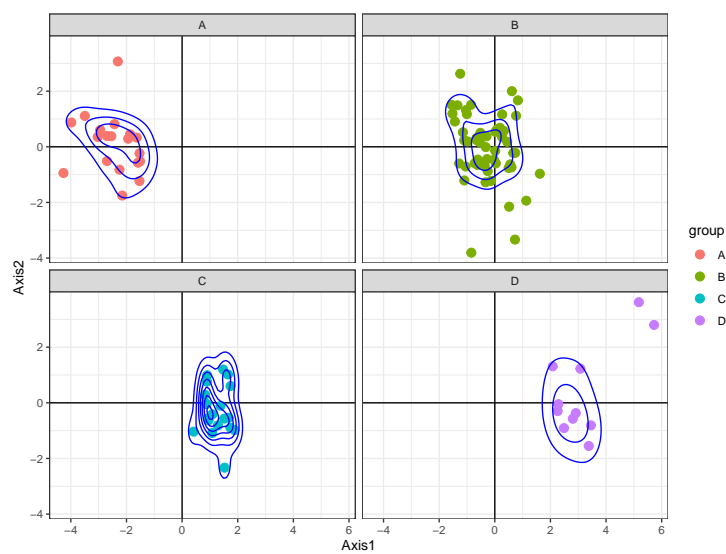


Figure 14: Representation of the observation with 2-D kernel density

7 Adding a picture

<https://cran.r-project.org/web/packages/imager/vignettes/gettingstarted.html>

```

library(ggplot2)
library(dplyr)
library(imager)
pdf("figs/image1.pdf",width=8, height=8)
parrots <- load.image(file.path(.libPaths()[1], 'imager/extdata/parrots.png'))
df <- as.data.frame(parrots,width="c") %>% mutate(rgb.val=rgb(c.1,c.2,c.3))
df$y <- rev(df$y)
p <- ggplot(df,aes(x,y)) + geom_raster(aes(fill=rgb.val)) + scale_fill_identity() + coord_fixed(ratio=1)
p
dev.off()

```



Figure 15: Representation of "parrots" picture from R package `imager`.

8 Complex figure with the package `cowplot`

9 Conclusion

References

- [1] D. Chessel, A.B. Dufour, and J. Thioulouse. The ade4 package-I- One-table methods. *R News*, 4:5–10, 2004.
- [2] S. Dray and A.B. Dufour. The ade4 package: implementing the duality diagram for ecologists. *Journal of Statistical Software*, 22(4):1–20, 2007.
- [3] S. Dray, A.B. Dufour, and D. Chessel. The ade4 package-II: Two-table and K-table methods. *R News*, 7(2):47–52, 2007.

10 Appendix

```
print(sessionInfo(), locale=FALSE)
R version 3.6.3 (2020-02-29)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Linux Mint 18.3

Matrix products: default
```

```

BLAS: /usr/lib/openblas-base/libblas.so.3
LAPACK: /usr/lib/libopenblas-p-r0.2.18.so

Random number generation:
RNG: Mersenne-Twister
Normal: Inversion
Sample: Rounding

attached base packages:
[1] datasets parallel stats graphics utils stats4 tools grDevices
[9] methods base

other attached packages:
[1] ggforce_0.2.2 ggrepel_0.8.1 cowplot_0.9.4
[4] ggplot2_3.3.2 knitr_1.23 pixmap_0.4-11
[7] ade4_1.7-15 RColorBrewer_1.1-2 rtracklayer_1.44.0
[10] GenomicRanges_1.36.0 GenomeInfoDb_1.20.0 IRanges_2.18.0
[13] S4Vectors_0.22.1 BiocGenerics_0.30.0

loaded via a namespace (and not attached):
[1] SummarizedExperiment_1.14.0 tidyselect_1.1.0
[3] xfun_0.10 purrr_0.3.4
[5] lattice_0.20-38 colorspace_1.4-1
[7] vctrs_0.3.5 generics_0.0.2
[9] XML_3.98-1.20 rlang_0.4.9
[11] isoband_0.2.3 pillar_1.4.7
[13] glue_1.4.2 withr_2.1.2
[15] tweenr_1.0.1 BiocParallel_1.18.1
[17] matrixStats_0.55.0 GenomeInfoDbData_1.2.1
[19] lifecycle_0.2.0 zlibbioc_1.30.0
[21] Biostrings_2.52.0 munsell_0.5.0
[23] gtable_0.3.0 labeling_0.3
[25] Biobase_2.44.0 Rcpp_1.0.5
[27] scales_1.0.0 DelayedArray_0.10.0
[29] XVector_0.24.0 farver_1.1.0
[31] Rsamtools_2.0.0 digest_0.6.22
[33] dplyr_1.0.2 polyclip_1.10-0
[35] grid_3.6.3 bitops_1.0-6
[37] magrittr_1.5 RCurl_1.95-4.12
[39] tibble_3.0.4 crayon_1.3.4
[41] pkgconfig_2.0.3 MASS_7.3-51.4
[43] ellipsis_0.3.0 Matrix_1.2-17
[45] rstudioapi_0.10 R6_2.4.0
[47] GenomicAlignments_1.20.0 compiler_3.6.3

options(encoding="latin1",prompt="> ", continue=" ", width = 85)
> save.image()

```