Fiche TD avec le logiciel R : tdr622

L'effet arc-en-ciel

P^r Jean R. Lobry

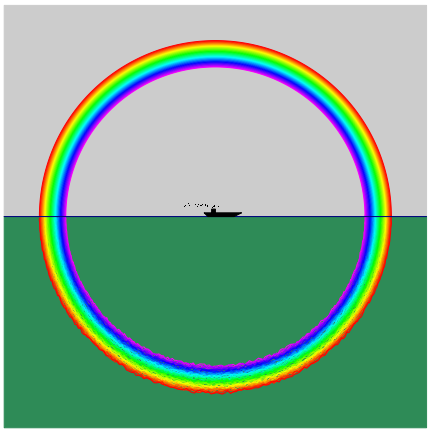
Exploration de l'effet Guttman (*Guttman effect* ou *Arch effect* ou *horseshoe effect*) dans l'analyse des correspondances au travers de simulations de données d'abondance d'espèces bactériennes le long d'un gradient thermique.)

Table des matières

1 Introduction

Dessinons un arc-en-ciel :

```
arc.en.ciel <- function(intx = 1.7, extx = 2, nl = 50, np = 200, sky = grey(0.8),
see = "seagreen")
{
  opar <- par(no.readonly = TRUE)
  par( mar = c(0.1, 0.1, 0.1, 0.1))
  angle <- seq(from = 0, to = pi, length = np)
  cols <- rev(rainbow(nl, end = 5/6))
  plot(x = 1.2*c(-extx, extx), y = c(0,0), type = "l", ylim = 1.2*c(-extx, extx),
  col = "black", bty = "n", xaxt = "n", yaxt = "n", ylab = "", xlab = "")
  rect(-1.2*extx, 0, 1.2*extx, 1.2*extx, col = sky, border = sky)
  rect(-1.2*extx, 0, 1.2*extx, -1.2*extx, col = see, border = see)
  icol <- 1
  for(r in seq(from = intx, to = extx, length = nl))
  {
    lines(x = r*cos(angle), y = r*sin(angle), type = "l", col = cols[icol])
    lines(x = jitter(r*cos(angle),factor = 10), y = -jitter(r*sin(angle), factor = 100),
    type = "l", col = cols[icol])
    icol <- icol + 1
  }
  lines(x = 1.2*c(-extx, extx), y = c(0,0), col = "navy")
  eps <- intx/20
  polygon(c(-eps, -1.5*eps, -eps/2, -eps/2, 0, 0, 3.5*eps, 2.5*eps),
          c(0,        eps/2,  eps/2, eps, eps, eps/2, eps/2, 0), col = "black")
  points(rnorm(20, -2*eps, eps), rnorm(20,1.5*eps,eps/10), pch = ".")
  par(opar)
}
arc.en.ciel()
```



Strictement aucun intérêt, mais c'est joli. On utilise ici, et uniquement ici, l'expression *effet arc-en-ciel* comme synonyme de :

— L'effet Guttman (Guttman effect)
— L'effet fer à cheval (horseshoe effect)
— L'effet arche (arch effect)

L'effet arc-en-ciel n'est pas un terme technique mais juste un *moyen mnémotechnique* qu'il ne faudrait pas employer en dehors du contexte de cette fiche.

On trouvera des exemples d'effet arc-en-ciel sur des données réelles dans la fiche tdr62 (pratique de l'analyse des correspondances). L'objectif ici est simplement de travailler avec des données entièrement simulées pour essayer de comprendre l'effet arc-en-ciel. On utilise :

```
library(ade4)
```

2 Simulation du jeu de données

Nous allons supposer pour simplifier qu'il n'y a que trois espèces observées, ceci nous permettra d'utiliser des représentations triangulaires très proches du jeu de données. Ces trois espèces différent pour leur thermotropisme : nous avons une espèce qui aime le froid (psychrophile) une espèce qui aime le tiède (mesophile) et une espèce qui aime le chaud (thermophile). Nous observons les abondances de ces espèces dans plusieurs stations réparties régulièrement le long d'un gradient de température. Mais nous allons faire comme si nous ne connaissions pas l'existence de ce gradient thermique.

L'influence de la température T, sur le taux de croissance en phase exponentielle des micro-organismes, μ, est donné en première approximation, par :

$$\mu(T) \begin{cases} = 0 & \text{si } T \notin [T_{min}, T_{max}] \\ = \frac{\mu_{opt}(T-T_{max})(T-T_{min})^2}{(T_{opt}-T_{min})[(T_{opt}-T_{min})(T-T_{opt})-(T_{opt}-T_{max})(T_{opt}+T_{min}-2T)]} & \text{si } T \in [T_{min}, T_{max}] \end{cases}$$

où T_{min} représente la température en deçà de laquelle il n'y a plus de croissance, T_{max} la température au delà de laquelle il n'y a plus de croissance, T_{opt} la température pour laquelle le taux de croissance atteint son maximum μ_{opt}. C'est le modèle dit des températures cardinales [2].

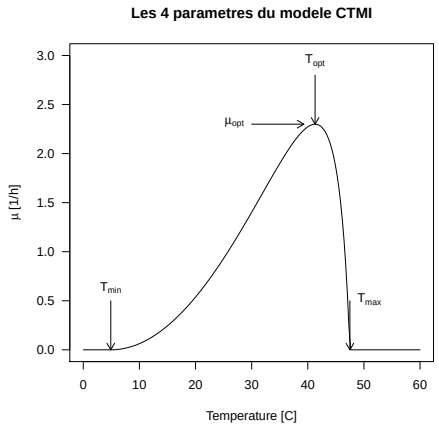
Commençons par définir une fonction donnant la valeur prédite du taux de croissance pour une température et un jeu de valeurs des paramètres donnés :

```
CTMI <- function(T, param)
{
  Tmin <- param[1]
  Topt <- param[2]
  Tmax <- param[3]
  Muopt <- param[4]
  if( T <= Tmin || T >= Tmax )
  {
    return(0)
  }
  else
  {
    Num <- (T-Tmax)*(T-Tmin)^2
    Den <- (Topt-Tmin)*((Topt-Tmin)*(T-Topt)-(Topt-Tmax)*(Topt+Tmin-2*T))
    return(Muopt*Num/Den)
  }
}
```

Représentons graphiquement la signification des paramètres de ce modèle, avec par exemple les valeurs pour *Escherichia coli* :

```
Tmin <- 4.9
Topt <- 41.3
Tmax <- 47.5
muopt <- 2.3
Taxis <- seq(from = 0, to = 60, length = 200)
predict <- sapply(Taxis, CTMI, param = c(Tmin, Topt, Tmax, muopt))
plot(Taxis, predict, type = "l", las = 1, ylim = c(0,3),
 xlab = "Temperature [C]", ylab = expression(paste(mu, " [1/h]")),
 main = "Les 4 parametres du modele CTMI")
arrows(Tmin, 0.5, Tmin, 0, length = 0.1)
text(x = Tmin, y = 0.5, labels = expression(T[min]), pos = 3)
arrows(Topt, muopt+0.5, Topt, muopt, length = 0.1)
text(x = Topt, y = muopt+0.5, labels = expression(T[opt]), pos = 3)
```

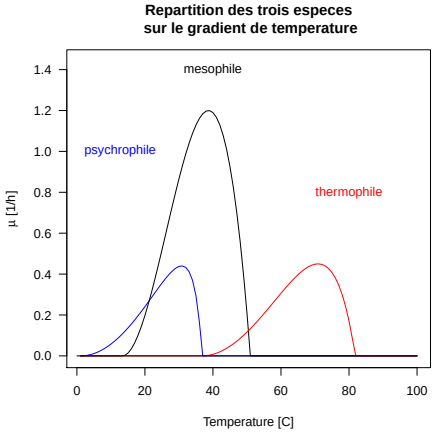
```
arrows(Tmax, 0.5, Tmax, 0, length = 0.1)
text(x = Tmax, y = 0.5, labels = expression(T[max]), pos = 4)
arrows(30, muopt, Topt-2, muopt, length = 0.1)
text(x = 30, y = muopt, labels = expression(mu[opt]), pos = 2)
```



Pour les trois espèces de notre simulation, nous allons utiliser les valeurs suivantes :

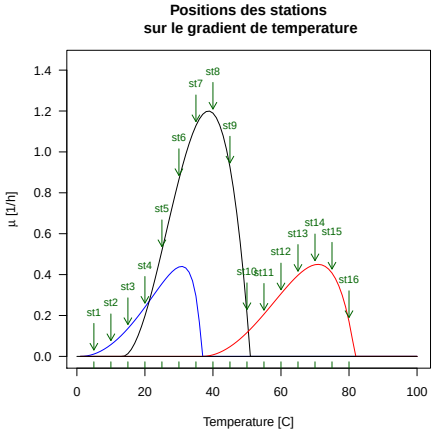
type	espèce	T_{min}	T_{opt}	T_{max}	μ [1/h]
psychrophile	*Xanthomonas pruni*	1.1	30.8	36.8	0.44
mesophile	*Bacillus subtilis*	13.4	38.7	51.0	1.2
thermophile	*Thermus aquaticus*	36.8	70.9	81.8	0.45

```
psychro <- c(1.1, 30.8, 36.8, 0.44)
meso <- c(13.4, 38.7, 51.0, 1.2)
thermo <- c(36.8, 70.9, 81.8, 0.45)
Taxis <- 1:100
ypsychro <- sapply(Taxis, CTMI, param = psychro)
ymeso <- sapply(Taxis, CTMI, param = meso)
ythermo <- sapply(Taxis, CTMI, param = thermo)
ymax <- max(ypsychro, ymeso, ythermo)
plot(Taxis, ypsychro, ylim = c(0, 1.2*ymax), type = "l",
  las = 1, col = "blue",
  xlab = "Temperature [C]", ylab = expression(paste(mu, " [1/h]")),
 main = "Repartition des trois especes\n sur le gradient de temperature")
lines(Taxis, ymeso)
lines(Taxis, ythermo, col = "red")
segments(0,0,100,0)
text(0, 1, "psychrophile", col = "blue", pos = 4)
text(40, 1.4, "mesophile")
text(80, 0.8, "thermophile", col = "red")
```



Nous allons maintenant supposer que nous avons des stations d'échantillonnage régulièrement réparties sur ce gradient :

```
plot(Taxis, ypsychro, ylim = c(0, 1.2*ymax), type = "l",
  las = 1, col = "blue",
  xlab = "Temperature [C]", ylab = expression(paste(mu, " [1/h]")),
 main = "Positions des stations\n sur le gradient de temperature")
lines(Taxis, ymeso)
lines(Taxis, ythermo, col = "red")
segments(0,0,100,0)
Tstations <- seq(from = 5, to = 80, by = 5)
for(s in Tstations)
{
  y <- max(ypsychro[s], ymeso[s], ythermo[s])
  arrows(s, 0.15+y, s, y+0.02, length = 0.1, col = "darkgreen")
  text(s, 0.15+y, paste("st",s/5, sep = ""), pos = 3, col = "darkgreen", cex = 0.8)
}
rug(Tstations, ticksize = 0.02, col = "darkgreen")
```



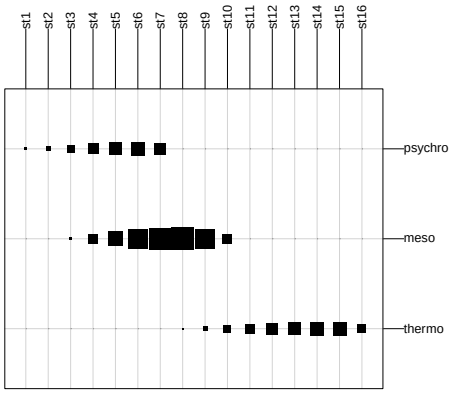
Nous simulons le jeu de données en supposant que les effectifs observés pour chaque espèce sont proportionnels à leur taux de croissance. On normalise pour avoir au maximum 1000 individus observés pour une espèce et on arrondi pour avoir un nombre entier d'individus :

```
data <- data.frame(rbind(ypsychro[Tstations], ymeso[Tstations],
 ythermo[Tstations]))
data <- round(1000*data/max(data),0)
names(data) <- paste("st",1:16, sep = "")
row.names(data) <- c("psychro", "meso", "thermo")
print(data)
```

```
        st1 st2 st3 st4 st5 st6 st7  st8 st9 st10 st11 st12 st13 st14 st15 st16
psychro   9  48 115 202 299 368 248    0   0    0    0    0    0    0    0    0
meso      0   0  11 166 435 727 948 1000 779  176    0    0    0    0    0    0
thermo    0   0   0   0   0   0   0    6  40   98  174  257  334  377  343  143
```

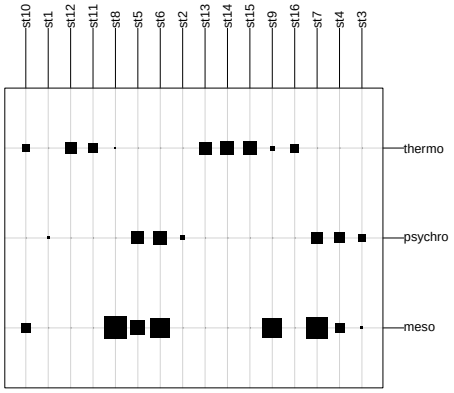
Représentons directement nos données :

```
table.value(data, clegend = 0)
```



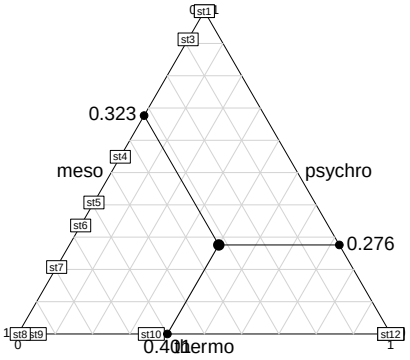
Mais comme nous ne sommes pas censés connaître le gradient de température, nous mélangeons nos données :

```
set.seed(01071966)
data <- data[sample(1:nrow(data)), sample(1:ncol(data))]
table.value(data, clegend = 0)
```



3 Représentation triangulaire

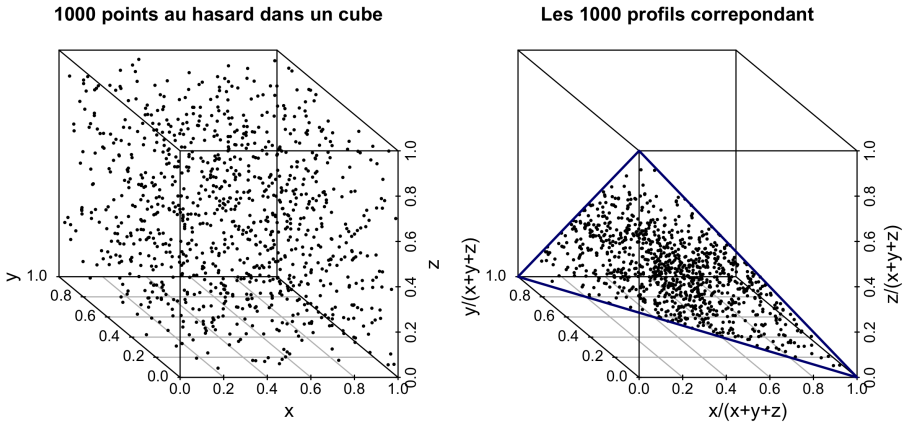
```
triangle.plot(as.data.frame(t(data)), label = names(data), clabel = 0.8,
addmean = TRUE, show.position = FALSE)
```



Cette représentation nous permet de voir exactement le jeu de données. Nous commençons par les stations 1 et 2 où il n'y a que des psychrophiles, puis nous passons aux stations 3 à 8 où la fréquence des mésophiles augmente jusqu'à ce qu'il n'y ait plus que des mésophiles, puis, pour les stations suivantes ce sont les mésophiles qui laissent la place aux thermophiles (les dernières stations sont superposées car elles ne contiennent que des thermophiles). Le gradient est sévère dans le sens où il n'y a aucune station qui contienne à la fois des psychrophiles, des mésophiles et des thermophiles (on parle de corrélation pointue). Nous avons triché ici pour l'interprétation puisque les stations sont déjà ordonnées dans l'ordre du gradient thermique, mais même sans cette connaissance, la figure resterait la même.

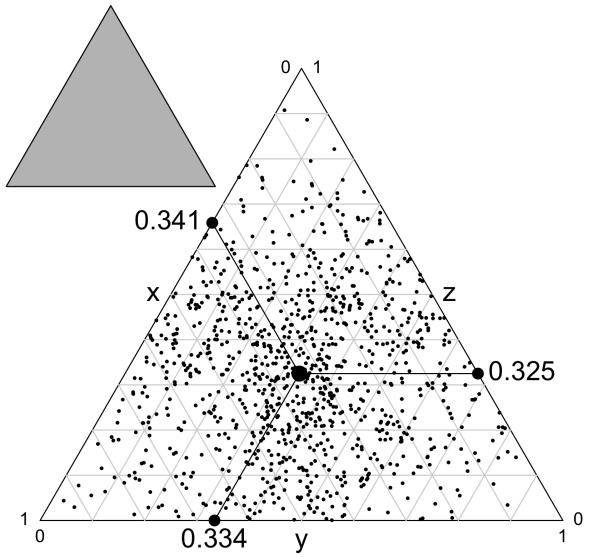
Ce dont il faut bien se convaincre pour la suite est que cette représentation est une représentation directe (sans projection, sans distorsion, sans trucage) de la réalité brute des profils (*i.e.* des fréquences relatives) des espèces par stations.

```
library("scatterplot3d")
n <- 1000
opar <- par(no.readonly = TRUE)
par(mfrow = c(1, 2), cex = 0.5, cex.main = 2)
x <- runif(n)
y <- runif(n)
z <- runif(n)
scatterplot3d(x,y,z, angle = 140 , pch = 19,
main = paste(n, "points au hasard dans un cube"))
tot <- x + y + z
x2 <- x/tot
y2 <- y/tot
z2 <- z/tot
sd3 <- scatterplot3d(x2,y2,z2, angle = 140,  pch = 19,
main = paste("Les", n, "profils correpondant"),
xlab = "x/(x+y+z)", ylab = "y/(x+y+z)", zlab = "z/(x+y+z)")
lines(sd3$xyz.convert(x = c(0, 0, 1, 0), y = c(0, 1, 0, 0), z = c(1, 0, 0, 1)), col = "navy", lwd = 2)
par(opar)
```



En calculant les profils nous imposons aux points une contrainte supplémentaire, ils se retrouvent sur un plan, et la représentation triangulaire reflète fidèlement ceci :

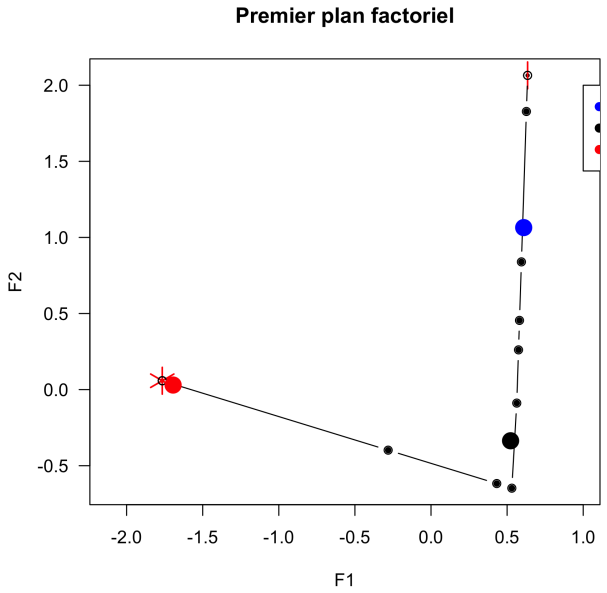
```
triangle.plot(cbind.data.frame(x = x, y = y, z = z), cpoint = 0.5,
addmean = TRUE)
```



4 Analyse factorielle des correspondances

```
afc <- dudi.coa(data, scann = FALSE, nf = 2)
x <- afc$co[order(afc$co[, 1]),1]
y <- afc$co[order(afc$co[, 1]),2]
sunflowerplot(x, y, las = 1, main = "Premier plan factoriel",
xlab = "F1", ylab = "F2", asp = 1)
lines(x, y, type = "b")
```

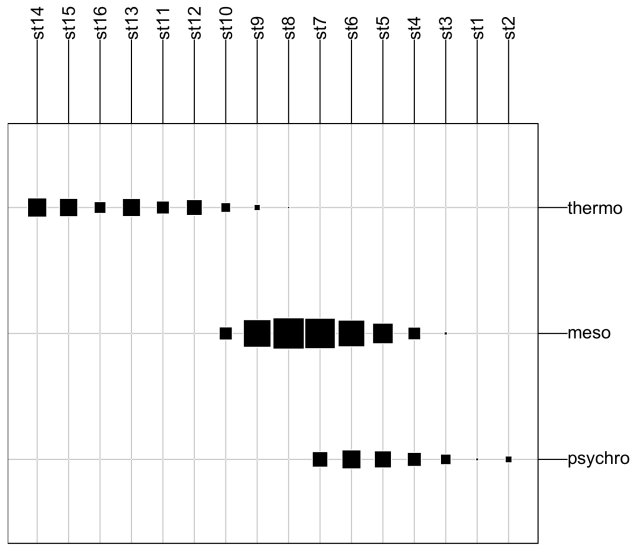
```
points(afc$li["psychro", 1], afc$li["psychro", 2], pch = 19, cex = 2, col = "blue")
points(afc$li["meso", 1], afc$li["meso", 2], pch = 19, cex = 2, col = "black")
points(afc$li["thermo", 1], afc$li["thermo", 2], pch = 19, cex = 2, col = "red")
legend(x = 1, y = 2, legend = c("psychrophile", "mesophile", "thermophile"),
  col = c("blue", "black", "red"), pch = 19)
```



Le premier facteur oppose les extrêmes : les psychrophiles contre les thermophiles. Le deuxième facteur nous oppose les espèces moyennes, les mesophiles, aux espèces extrêmes. L'effet arc-en-ciel est très brutal ici parce que nous sommes en présence de corrélations pointues avec plusieurs stations où ne sont présentes qu'une seule espèces (en particulier pour les thermophiles ici).

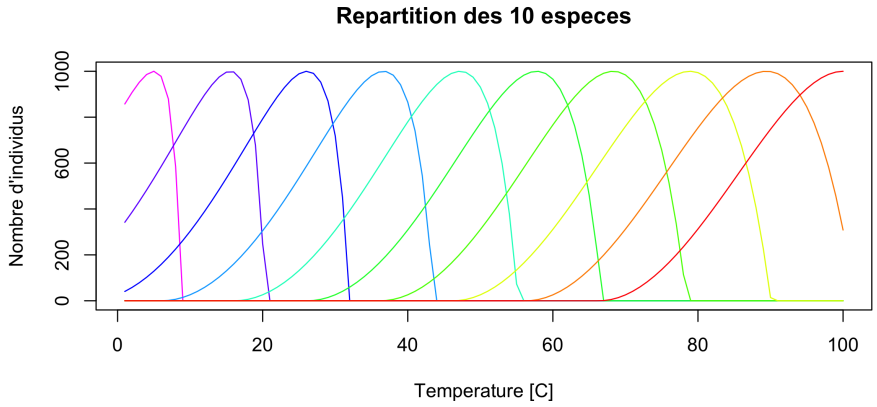
Le premier facteur de l'afc nous permet de retrouver le gradient de température :

```
table.value(data[order(afc$li[,1]),order(afc$co[,1])], clegend = 0)
```



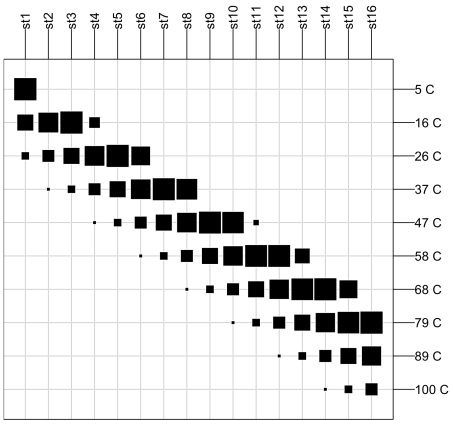
Pour avoir un plus bel effet arc-en-ciel il faut un dégradé un peu moins brutal, par exemple :

```
nsp <- 10
topt <- seq(from = 5, to = 100, length = nsp)
tmin <- 0.953*topt - 28.913
tmax <- 1.101*topt + 3.203
data <- data.frame(matrix(nrow = nsp, ncol = length(Tstations)))
names(data) <- paste("st",1:16, sep = "")
row.names(data) <- paste(round(topt,0), "C")
for( i in 1:nsp )
  for( j in 1:length(Tstations))
    data[i, j] <- round(CTMI(T = Tstations[j], param = c(tmin[i], topt[i], tmax[i], 1000)), 0)
Taxis <- 1:100
cols <- rev(rainbow(nsp, end = 5/6))
plot(x = Taxis, y = sapply(Taxis, CTMI, param = c(tmin[1], topt[1], tmax[1], 1000)),
type = "l", col = cols[1], main = paste("Repartition des", nsp, "especes"),
xlab = "Temperature [C]", ylab = "Nombre d'individus")
for( i in 2:nsp )
lines(x = Taxis, y = sapply(Taxis, CTMI, param = c(tmin[i], topt[i], tmax[i], 1000)),
col = cols[i])
```



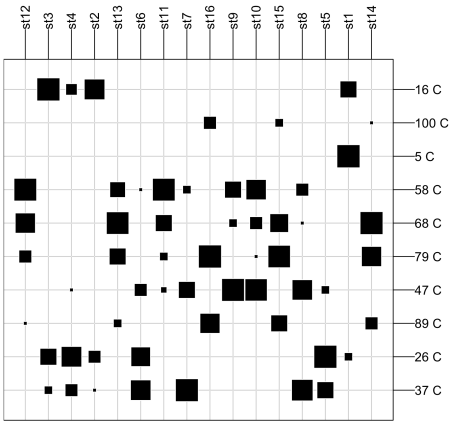
Regardons les données brutes :

```
table.value(data, clegend = 0)
```



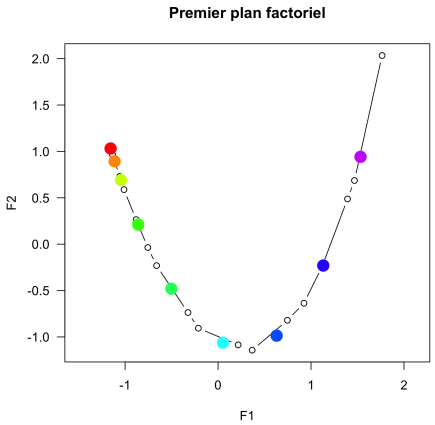
Mélangeons le tout :

```
data <- data[sample(1:nrow(data)), sample(1:ncol(data))]
table.value(data, clegend = 0)
```



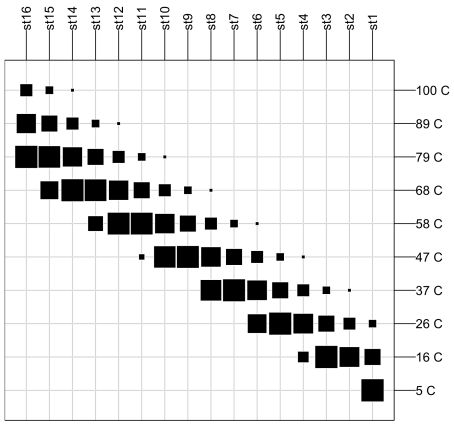
Voyons le premier plan factoriel :

```
afc <- dudi.coa(data, scann = FALSE, nf = 2)
x <- afc$co[order(afc$co[, 1]),1]
y <- afc$co[order(afc$co[, 1]),2]
plot(x, y, las = 1, main = "Premier plan factoriel",
xlab = "F1", ylab = "F2", asp = 1)
lines(x, y, type = "b")
points(afc$li[order(afc$li[, 1]), 1], afc$li[order(afc$li[, 1]), 2], pch = 19, cex = 2, col = rainbow(nsp))
```



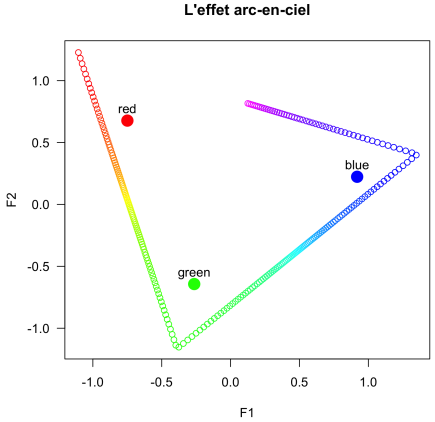
Le premier facteur de l'afc nous permet encore ici de retrouver le gradient de température :

```
table.value(data[order(afc$li[,1]),order(afc$co[,1])], clegend = 0)
```



Voici donc l'effet arc-en-ciel : quand il y a un dégradé dans les données, comme dans les couleurs d'un arc-en-ciel, le gradient tend à être replié sur le premier plan factoriel, comme dans la forme d'un arc-en-ciel :

```
n <- 255
cols <- rainbow(n = n, end = 5/6)
data <- as.data.frame(t(col2rgb(cols)))
afc <- dudi.coa(data, nf = 2, scann = FALSE)
plot(afc$li[,1], afc$li[,2], col = cols, las = 1,
main = "L'effet arc-en-ciel", xlab = "F1", ylab = "F2")
points(x = afc$co[,1], y = afc$co[,2], col = names(data), pch = 19, cex = 2)
text(x = afc$co[,1], y = afc$co[,2], labels = names(data), pos = 3)
```

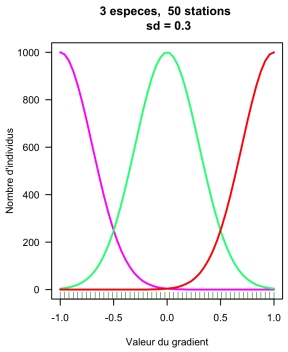


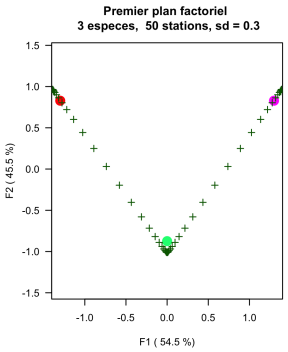
On peut s'amuser à définir une petite fonction pour explorer un peu l'effet arc-en-ciel :

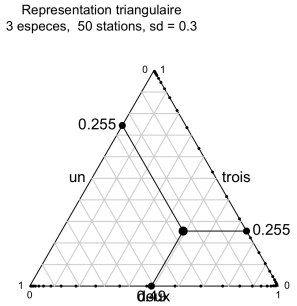
```
simu <- function(nsp = 10, nx = 100, sig = 0.15, nmax = 1000)
{
  opar <- par(no.readonly = TRUE)
  if( nsp == 3 )
  {
    par(mfrow = c(1,3))
  }
  else
  {
    par(mfrow = c(1,2))
  }

  means <- seq(-1, 1, length = nsp)
  xvals <- seq(-1, 1, length = nx)
  ymax <- dnorm(0, sd = sig)
  cols <- rev(rainbow(nsp, end = 5/6))
  plot(0,0, ylim = c(0, nmax), type = "n", las = 1,
    xlab = "Valeur du gradient", ylab = "Nombre d'individus",
    main = paste(nsp, "especes, ", nx, "stations \n sd =", sig))
  rug(xvals, col = "darkgreen")
  data <- as.data.frame(matrix(nrow = nsp, ncol = nx))
  for( i in 1:nsp)
  {
    data[i, ] <- round(nmax*dnorm(xvals, mean = means[i], sd = sig)/ymax, 0)
    #data[i, is.na(data[i,])] <- 0
    #print(i)
    lines(xvals, data[i,], col = cols[i], lwd = 2)
  }
  afc <- dudi.coa(data, nf = 2, scann = FALSE)
  plot(afc$li[,1], afc$li[,2], asp = 1, las = 1,
    xlab = paste("F1 (", round(100*afc$eig[1]/sum(afc$eig), 1), "%)"),
    ylab = paste("F2 (", round(100*afc$eig[2]/sum(afc$eig), 1), "%)"),
    main = paste("Premier plan factoriel \n", nsp, "especes, ", nx, "stations, sd =", sig), col = cols, pch =
  points(afc$co[,1], afc$co[,2], pch = 3, col = "darkgreen")

  if( nsp == 3 )
  {
    tmp <- t(data)
    names(tmp) <- c("un","deux","trois")
    triangle.plot(tmp, sub = paste("Representation triangulaire \n", nsp, "especes, ", nx, "stations, sd =", s
    csub = 2, possub = "topleft", addmean = TRUE, clabel = 0, labeltriangle = TRUE,
      show.position = FALSE)
  }
  par(opar)
}
```

Avec trois espèces, le premier plan factoriel donne une représentation exacte des données. Regardons comment l'on passe progressivement d'une corrélation pointue à un effet arc-en-ciel :

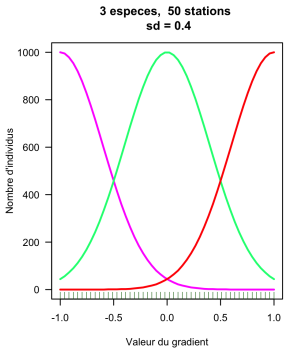
```
simu(sig = 0.3, nsp = 3, nx = 50)
```

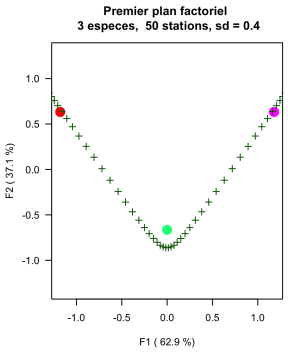


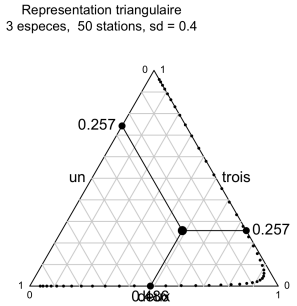




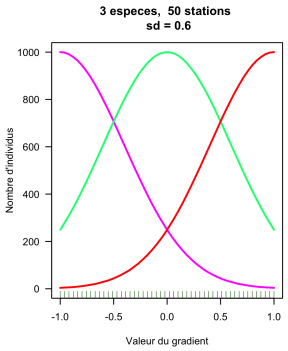

```
simu(sig = 0.4, nsp = 3, nx = 50)
```

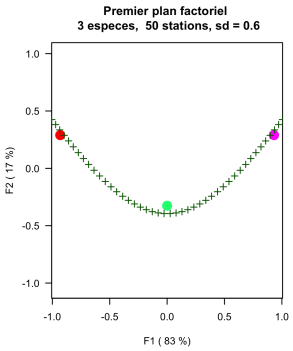


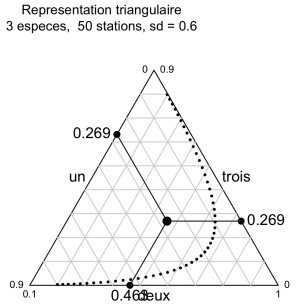




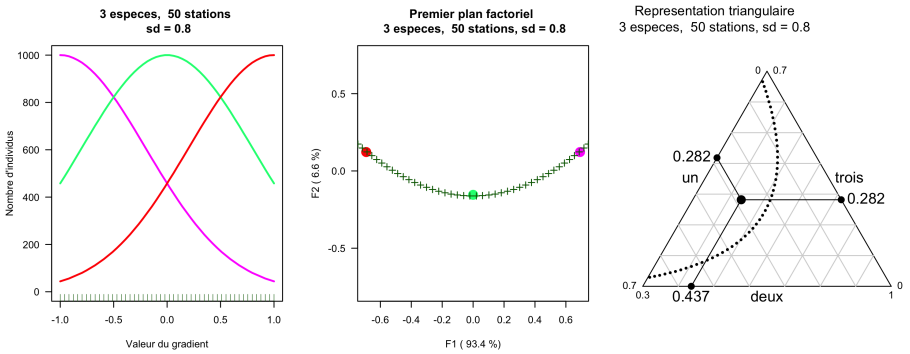

```
simu(sig = 0.6, nsp = 3, nx = 50)
```





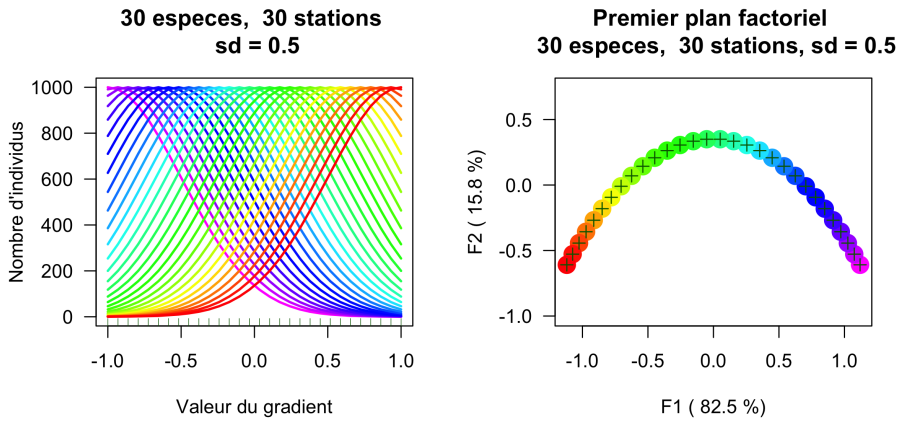



```
simu(sig = 0.8, nsp = 3, nx = 50)
```

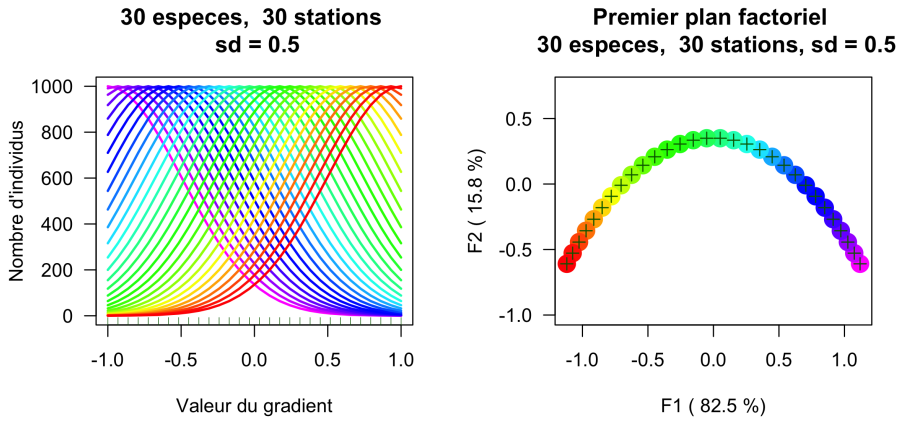


On peut aussi augmenter le nombre d'espèces pour avoir un plus bel effet arc-en-ciel (mais on ne peux plus alors utiliser la représentation triangulaire) :

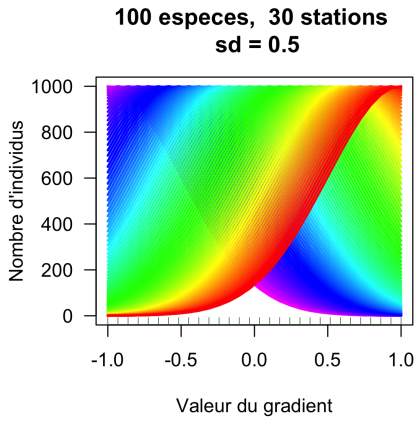
```
simu(sig = 0.5, nsp = 10, nx = 30)
```

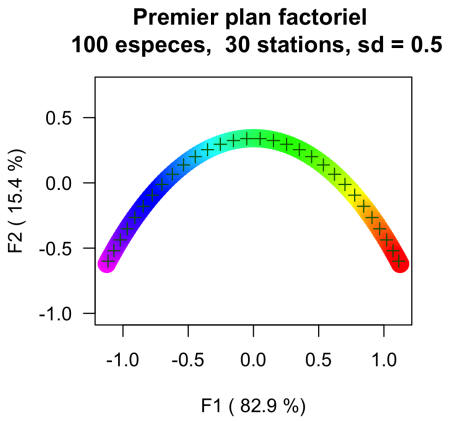



```
simu(sig = 0.5, nsp = 30, nx = 30)
```

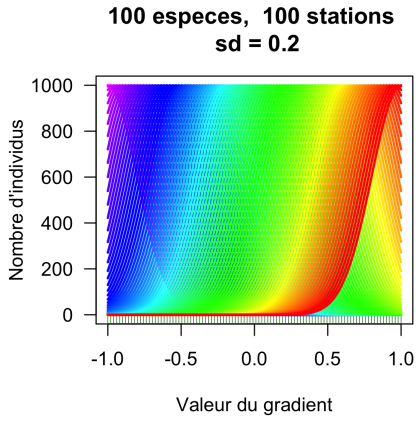


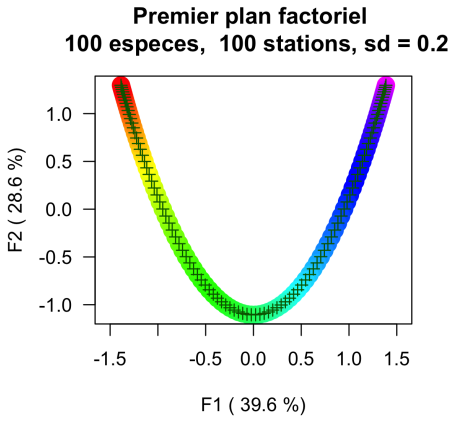

```
simu(sig = 0.5, nsp = 100, nx = 30)
```






```
simu(sig = 0.2, nsp = 100, nx = 100)
```





Ceci illustre bien l'effet arc-en-ciel : quand il y a un dégradé dans les données, comme dans les couleurs d'un arc-en-ciel, le gradient tend à être replié sur le premier plan factoriel, comme dans la forme d'un arc-en-ciel.

Annexe : Analyse factorielle des correspondances détendancée

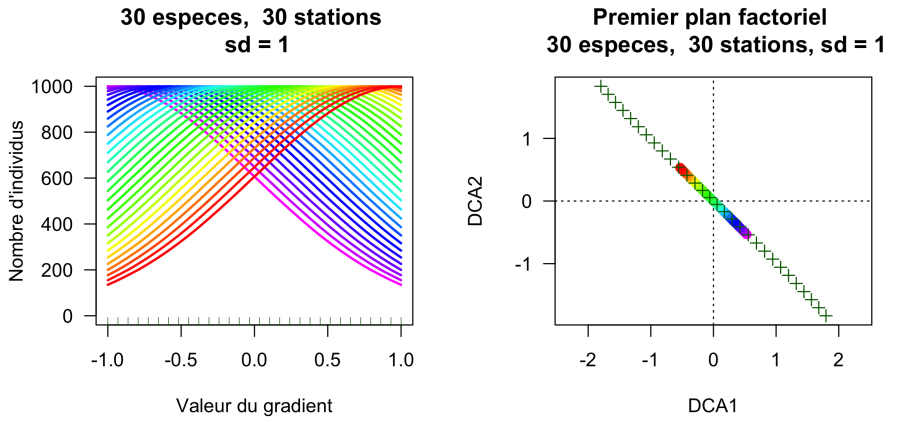
Bien que la représentation triangulaire nous montre bien dans le cas de trois espèces que le nuage de points est *intrinséquement* en forme d'arc-en-ciel quand il y a un dégradé dans les données, c'est une propriété qui a été jugée indésirable par Hill et Gauch [1] parce que :

— L'effet arc-en-ciel est un simple artéfact mathématique ne correspondant à aucune structure réelle dans les données (? ? ?)

— Il conduit à un écrasement des espèces extrêmes et un éloignement des espèces moyennes sur les plans factoriels. On peut voir dans les simulations précédentes que l'effet est d'autant plus marqué que l'on se rapproche des situations de corrélation pointues.

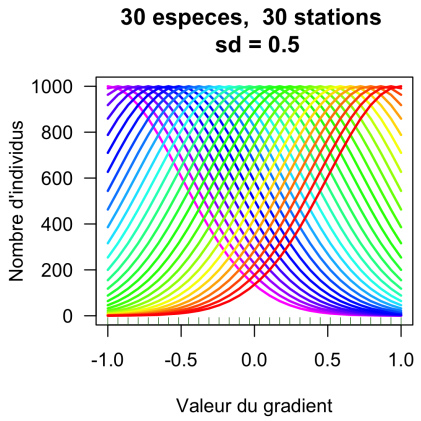
Pour pallier ces inconvénients ils proposent une méthode *ad hoc* consistant essentiellement à découper le premier facteur en tranches et à recentrer les valeurs sur le deuxième facteur pour qu'elles soient de moyenne nulle localement. La procédure est décrite exactement par la fonction `decorana()` de la bibliothèque `vegan`. Voyons ce que cela donne :

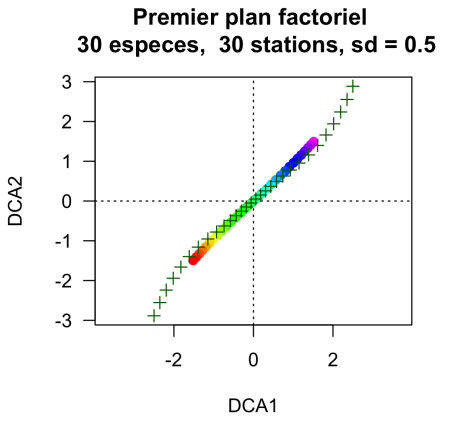
```
library(vegan)
simudca <- function(nsp = 10, nx = 100, sig = 0.15, nmax = 1000)
{
  opar <- par(no.readonly = TRUE)
  par(mfrow = c(1,2))

  means <- seq(-1, 1, length = nsp)
  xvals <- seq(-1, 1, length = nx)
  ymax <- dnorm(0, sd = sig)
  cols <- rev(rainbow(nsp, end = 5/6))
  plot(0,0, ylim = c(0, nmax), type = "n", las = 1,
    xlab = "Valeur du gradient", ylab = "Nombre d'individus",
    main = paste(nsp, "especes, ", nx, "stations \n sd =", sig))
  rug(xvals, col = "darkgreen")
  data <- as.data.frame(matrix(nrow = nsp, ncol = nx))
  for( i in 1:nsp)
  {
    data[i, ] <- round(nmax*dnorm(xvals, mean = means[i], sd = sig)/ymax, 0)
    lines(xvals, data[i,], col = cols[i], lwd = 2)
  }
  dafc <- decorana(data)
  plot(dafc, las = 1,
    main = paste("Premier plan factoriel \n", nsp, "especes, ", nx, "stations, sd =", sig),
    cex = 0.8, type = "points", display = "none")
  especes <- scores(dafc, display = "sites", choice = 1:2)
  points(x = especes[, 1], y = especes[,2], col = cols, pch = 19)
  stations <- scores(dafc, display = "species", choice = 1:2)
  points(x = stations[, 1], y = stations[,2], col = "darkgreen", pch = 3)
  par(opar)
}
simudca(sig =1, nsp=30, nx = 30)
```



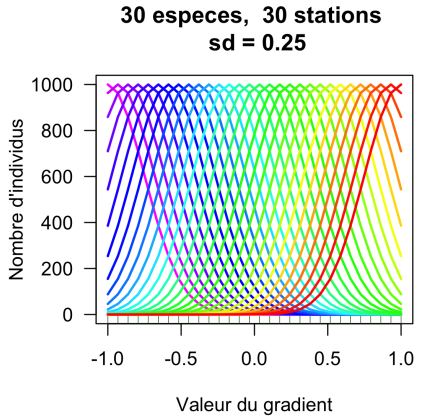
On a bien détordu l'arc-en-ciel, mais du coup le premier et le deuxième facteur sont identiques.

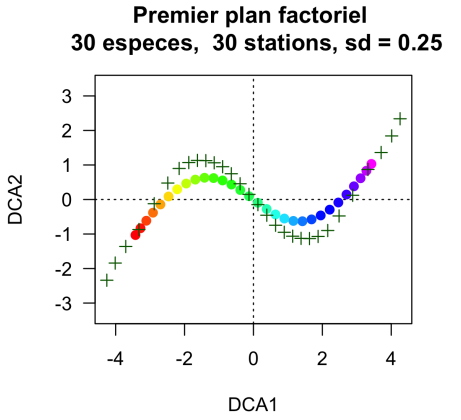
```
simudca(sig =0.5, nsp=30, nx = 30)
```



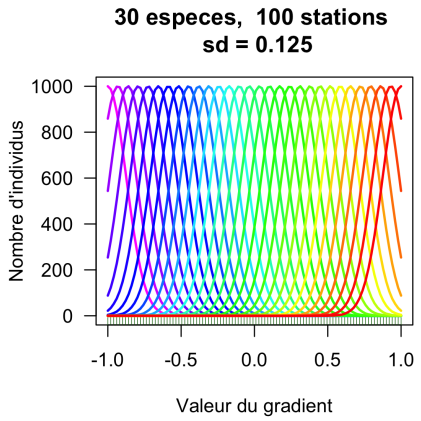


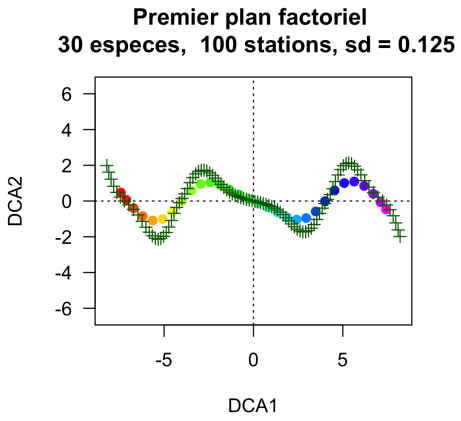

```
simudca(sig =0.25, nsp=30, nx = 30)
```



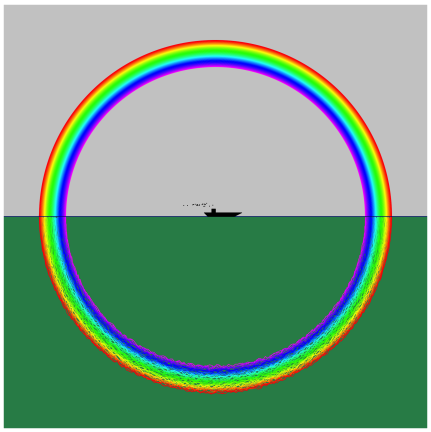



```
simudca(sig =0.125, nsp=30, nx = 100)
```





On neutralise bien l'effet arc-en-ciel entre le premier et le deuxième facteur, mais du coup on fait apparaître les polynômes de degré supérieur entre le premier facteur et ceux suivant le deuxième facteur. C'est un peu du bricolage tout ça.



Références

[1] M.O. Hill and H.G. Gauch. Detrended correspondence analysis : an improved ordination technique. *Vegetatio*, 42 :47–58, 1980.

[2] L. Rosso, J.R. Lobry, and J.-P. Flandrois. An unexpected correlation between cardinal temperatures of microbial growth highlighted by a new model. *Journal of Theoretical Biology*, 162(4) :447–463, 1993.