Clustering Techniques

PAZ

10 avril 2017

Reference

Modified from: D. Borcard & F. Gillet Multivariate Analysis in Community Ecology: Constrained ordination and other analysis

Adapted from: Gwenaël Imfeld, LyGeS,2009

Import packages

```
# Preparation of the workspace
# Remove all R objects in the workspace
rm(list = ls())

# ipak function: install and load multiple R packages.
# check to see if packages are installed. Install them if they are not, then load them into the R sessi
ipak <- function(pkg){
  new.pkg <- pkg[!(pkg %in% installed.packages()[, "Package"])]
  if (length(new.pkg))
    install.packages(new.pkg, dependencies = TRUE)
  sapply(pkg, require, character.only = TRUE)
}

# usage
packages <- c("vegan", "cluster", "gclus", "MASS")
# ipak(packages)

# Load required libraries
require("vegan")
require("cluster")
require("gclus")

library("ggplot2")
library("ggrepel")
library("MASS")
```

Import data sets

```
# Check working directory
getwd()
```

```
## [1] "D:/Documents/these_pablo/Alteckendorf2016/HydrologicalMonitoring"
```

```
# setwd("D:/Documents/these_pablo/Rscripts/Clustering")

waters = read.csv2("Data/WeeklyHydroContam_R.csv")
```

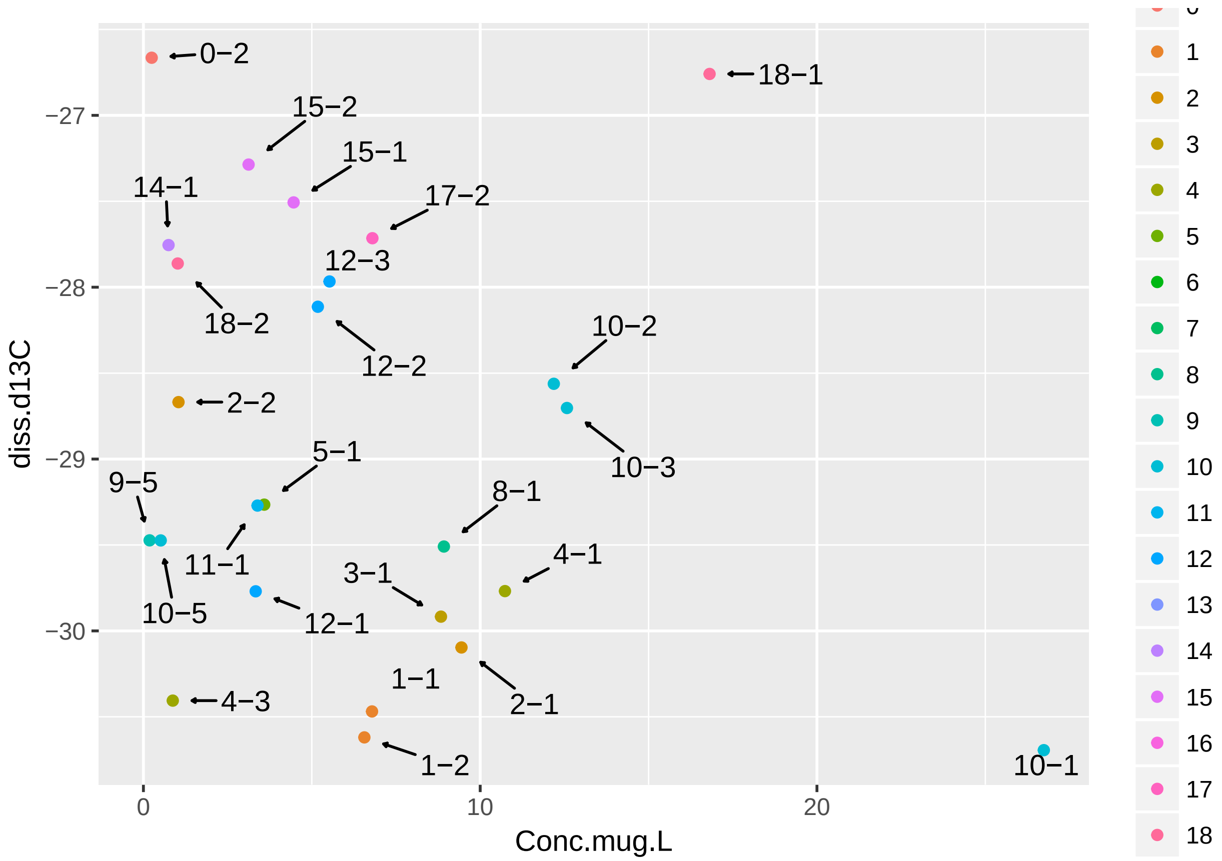
```r
waters$ti <- as.POSIXct(strptime(waters$ti, "%Y-%m-%d %H:%M", tz="EST"))
colnames(waters)[colnames(waters) == "ti"] <- "Date.ti"
waters$Events <- factor(waters$Events, levels = unique(waters$Events))
waters$Event <- factor(waters$Event, levels = unique(waters$Event))

dropWater <- c("N.x", "N.y",
               "Markers" , "TimeDiff",
               "se.d13C", "MES.mg.L", "MES.sd", "MO.mg.L", "filt.se.d13C", "f.diss", "f.filt",
               "Appl.Mass.g",
               "DissSmeto.mg", "DissSmeto.mg.SD",
               "DissOXA.mg", "DissOXA.mg.SD",
               "DissESA.mg", "DissESA.mg.SD",
               "FiltSmeto.mg", "DissSmeto.mg.SD",
               "TotSMout.mg", "TotSMout.mg.SD",
               "FracDiss", "FracFilt")
waters <- waters[ , !(names(waters) %in% dropWater)]

# Date conversion correct:
sum(is.na(waters$Date.ti)) == 0
```

```
## [1] TRUE
```

```r
str(waters)
```

```
## 'data.frame':    51 obs. of  64 variables:
##  $ Date.ti               : POSIXct, format: "2016-03-25 00:04:00" "2016-03-25 12:04:00" ...
##  $ WeekSubWeek           : Factor w/ 51 levels "W0-0x","W0-1",..: 1 2 3 4 5 6 26 27 28 29 ...
##  $ tf                    : Factor w/ 51 levels "2016-03-25 12:02:00",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ iflux                 : num  1.25 1.12 1.31 1.46 16.33 ...
##  $ fflux                 : num  1.13 1.31 1.46 16.45 15.18 ...
##  $ changeflux            : num  -0.119 0.189 0.148 14.989 -1.15 ...
##  $ maxQ                  : num  1.25 1.38 1.64 38.4 18.67 ...
##  $ minQ                  : num  1.118 1.082 0.929 1.449 13.201 ...
##  $ Duration.Hrs          : num  12 82.5 37.6 27.3 23.1 ...
##  $ chExtreme             : num  -0.13 0.256 0.33 36.944 -3.133 ...
##  $ Peak                  : int  NA NA NA 1 NA NA 2 NA NA 3 ...
##  $ AveDischarge.m3.h     : num  1.2 1.21 1.28 14.32 15.53 ...
##  $ Volume.m3             : num  14.4 100.2 48.3 390.4 359.2 ...
##  $ Sampled.Hrs           : num  12 82.5 37.6 27.3 23.1 ...
##  $ Sampled               : Factor w/ 2 levels "Not Sampled",..: 1 2 1 2 2 1 2 2 1 2 ...
##  $ Conc.mug.L            : num  0.246 0.246 3.517 6.788 6.561 ...
##  $ Conc.SD               : num  0.0193 0.0193 0.1544 0.2894 0.1906 ...
##  $ OXA_mean              : num  4.82 4.82 17.68 30.53 32.49 ...
##  $ OXA_SD                : num  1.141 1.141 5.663 10.185 0.243 ...
##  $ ESA_mean              : num  18.1 18.1 32 46 41.3 ...
##  $ ESA_SD                : num  3.497 3.497 3.267 3.037 0.853 ...
##  $ diss.d13C             : num  NA -26.7 NA -30.5 -30.6 ...
##  $ SD.d13C               : num  NA 0.936 NA 0.106 0.151 ...
##  $ Conc.Solids.mug.gMES  : num  0.645 0.645 0.385 0.126 0.436 ...
##  $ Conc.Solids.ug.gMES.SD: num  0.0232 0.0232 0.0252 0.0271 0.1232 ...
##  $ filt.d13C             : num  NA NA NA NA NA ...
##  $ filt.SD.d13C          : num  NA NA NA NA NA ...
##  $ DD13C.diss            : num  NA 4.545 NA 0.741 0.59 ...
##  $ DD13C.filt            : num  NA NA NA NA NA ...
##  $ B.diss                : num  NA 93.1 NA 35.4 29.4 ...
##  $ B.filt                : num  NA NA NA NA NA ...
##  $ NH4.mM                : num  NA NA NA 0.05 NA NA NA NA NA NA ...
##  $ TIC.ppm.filt          : num  NA NA NA 51.8 44.8 NA 66.7 52.1 NA 69.4 ...
##  $ Cl.mM                 : num  NA NA NA 1.48 1574 ...
##  $ NO3...mM              : num  NA NA NA 616 778 ...
##  $ PO4..mM               : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ NPOC.ppm              : num  NA NA NA 4 4.4 NA 5.8 3.4 NA 9.1 ...
##  $ TIC.ppm.unfilt        : num  NA NA NA 44.8 26.4 NA 39 32.3 NA 54.8 ...
##  $ TOC.ppm.unfilt        : num  NA NA NA 4.7 5.4 NA 2.7 3.8 NA 3.9 ...
##  $ ExpMES.Kg             : num  5.35 5.35 14.88 24.4 8.08 ...
##  $ CumAppMass.g          : num  6369 6369 6369 6369 6369 ...
##  $ DissSmeto.g           : num  0.00354 0.0246 0.17004 2.64991 2.357 ...
##  $ DissSmeto.g.SD        : num  0.000278 0.001934 0.007463 0.11298 0.068486 ...
##  $ DissOXA.g             : num  0.0695 0.4832 0.8547 11.9184 11.6727 ...
##  $ DissOXA.g.SD          : num  0.0165 0.1143 0.2738 3.976 0.0873 ...
##  $ DissESA.g             : num  0.26 1.81 1.55 17.95 14.83 ...
##  $ DissESA.g.SD          : num  0.0504 0.3503 0.158 1.1855 0.3066 ...
##  $ FiltSmeto.mg.SD       : num  0.124 0.124 0.374 0.66 0.996 ...
##  $ FiltSmeto.g           : num  0.00345 0.00345 0.00573 0.00307 0.00352 ...
##  $ FiltSmeto.g.SD        : num  0.000124 0.000124 0.000374 0.00066 0.000996 ...
##  $ TotSMout.g            : num  0.00699 0.02806 0.17577 2.65298 2.36052 ...
##  $ TotSMout.g.SD         : num  0.000216 0.00137 0.005284 0.07989 0.048432 ...
##  $ MELsm.g               : num  0.302 2.078 2.379 30.241 27.008 ...
##  $ MELsm.g.SD            : num  0.0269 0.1868 0.1789 2.4062 0.1634 ...
##  $ CumOutDiss.g          : num  0.00354 0.02815 0.19818 2.84809 5.2051 ...
##  $ CumOutFilt.g          : num  0.00345 0.0069 0.01263 0.01571 0.01923 ...
##  $ CumOutSmeto.g         : num  0.00699 0.03505 0.21082 2.8638 5.22432 ...
##  $ CumOutMELsm.g         : num  0.302 2.38 4.76 35.001 62.009 ...
##  $ BalMassDisch.g        : num  6369 6367 6365 6334 6307 ...
##  $ prctMassOut           : num  4.98e-05 2.00e-04 1.25e-03 1.89e-02 1.68e-02 ...
##  $ FracDeltaOut          : num  0 -0.00533 0 -0.57576 -0.51483 ...
##  $ Events                : Factor w/ 51 levels "0-1","0-2","0-3",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ Weeks                 : Factor w/ 16 levels "W0","W1","W10",..: 1 1 1 2 2 2 9 9 9 10 ...
##  $ Event                 : Factor w/ 19 levels "0","1","2","3",..: 1 1 1 2 2 2 3 3 3 4 ...
```

```
# Conc.mug.L
# TotSMout.g
# MELsm.g
ggplot(waters, aes(x=Conc.mug.L, y=diss.d13C))+
  geom_point(aes(group = Event, colour = Event))+
  geom_text_repel(aes(label=Events),
                 arrow = arrow(length = unit(0.005, 'npc'), type = "closed"),
                 force = 1,
                 point.padding = unit(1.0, 'lines'),
                 max.iter = 2e3,
                 nudge_x = .2)
```



Variable generation

We would like to determine whether there are different clusters in the data.

Via response variables:

- Concentrations ($\mu g/L$)
- MEL-sm (g)
- Loads (SM g)
- Transformation products (OXA and ESA in $\mu g/L$ and in loads g)

Via hydrological characteristics:

- Event index:

$$\frac{I_{max} \cdot R_{tot}}{D}$$

- Event duration ($t_f - t_i$)
- Volume discharged ($\sum_{i=1}^{N} Q_i \cdot dt_i$, N: no. of measurements within the event)
- Average discharge ($\sum_{i=1}^{N} Q_i/N$)

Imax = max rainfall intensity mm/h ; Rtot = rainfall amount (mm); D = duration (min)

"A high EVI represents a short but intense rainfall event, whereas a low EVI indicates an event with a low intensity but long duration. The catchment response time is defined as the time between the gravity centre of the rain event and the peak outflow. (Baartman et al., 2013; in Lefrancq etal2017)"

The EVI has been adapted to reflect discharge index such that:

- Discharge index A [m3/h x m3/h]

$$DIa = \frac{Q_{max} \cdot V_{tot}}{D}$$

- Discharge index B [-]

$$DIb = \frac{V_{tot}}{D \cdot Q_{max}}$$

```
waters$DIa <- waters$maxQ*waters$Volume.m3/waters$Duration.Hrs
waters$DIb <- waters$Volume.m3/waters$Duration.Hrs * 1/waters$maxQ
waters$TPs.g <- waters$MELsm.g-waters$TotSMout.g
```

Normalization choice

```
# Option 1.
# Divide by estimated mass in catchment available # [-]
waters$SM.g.nrm <- waters$TotSMout.g/waters$BalMassDisch.g # [-]
waters$MEL.g.nrm <- waters$MELsm.g/waters$BalMassDisch.g # [-]

# Option 2
# Divide by estimated prct. mass in catchment available # [g]
waters$CumPrctMassOut <- cumsum(waters$prctMassOut)
waters$SM.g.nrm.prc <- waters$TotSMout.g/waters$CumPrctMassOut # [-]
waters$MEL.g.nrm.prc <- waters$MELsm.g/waters$CumPrctMassOut # [-]
```

Variable reduction

```
# Main data frame -> "waterSmall"
includeWater <- c(
  "Events",
  # Response variables
  "Conc.mug.L", "OXA_mean", "ESA_mean",
  "SM.g.nrm", "MEL.g.nrm", "SM.g.nrm.prc", "MEL.g.nrm.prc",
  "DD13C.diss", # "diss.d13C",
  "TotSMout.g", "MELsm.g",
  "TPs.g", "DissOXA.g", "DissESA.g",
  # "Cl.mM", "NO3..mM",
  "ExpMES.Kg",
  # Independent/event variables
  "DIa", "DIb", "maxQ",
  "Duration.Hrs", "Volume.m3", "AveDischarge.m3.h")

waterSmall <- waters[ , (names(waters) %in% includeWater)]

# Lets reduce the data set even more to only response variables.
includeResponse <- c(
  # Response variables
  #"Conc.mug.L", "OXA_mean", "ESA_mean",
  # "SM.g.nrm", "MEL.g.nrm",
  "SM.g.nrm.prc", "MEL.g.nrm.prc",
  #"diss.d13C",
```

```r
  "DD13C.diss"
  #"TotSMout.g", "MELsm.g",
  #"TPs.g", "DissOXA.g", "DissESA.g",
  # "Cl.mM", "NO3..mM",
  #"ExpMES.Kg"
  )

respHydro <- c(
  "Events",
  # Response variables
  #"Conc.mug.L", "OXA_mean", "ESA_mean",
  # "SM.g.nrm", "MEL.g.nrm",
  "SM.g.nrm.prc", "MEL.g.nrm.prc",
  #"diss.d13C",
  "DD13C.diss",
  #"TotSMout.g", "MELsm.g",
  #"TPs.g", "DissOXA.g", "DissESA.g",
  # "Cl.mM", "NO3..mM",
  #"ExpMES.Kg"
  # Independent/event variables
  "DIa", "DIb", "maxQ",
  "Duration.Hrs", "Volume.m3", "AveDischarge.m3.h")

responseWater <- waterSmall[ , (names(waterSmall) %in% includeResponse)]
waterXY <-  waterSmall[ , (names(waterSmall) %in% respHydro)]
waterXY.nona <- waterXY[complete.cases(waterXY), ]
```

Transformations

```r
responseWater.hell <- decostand(responseWater, "hellinger", na.rm=T)
# responseWater.norm <- decostand(responseWater, "norm", na.rm=T)

# Re-arrange columns to have "Events" as Index
if (ncol(waterXY == 7)){
  waterXY <- waterXY[, c(6,1:5,7:ncol(waterXY))]
  waterXY.hell <- decostand(waterXY[2:ncol(waterXY)], "hellinger", na.rm=T, MARGIN = 2)
} else {
  waterXY.hell <- decostand(waterXY, "hellinger", na.rm=T, MARGIN = 2)
}
waterXY.hell.nona <- waterXY.hell[complete.cases(waterXY.hell),]

# Normalized response variables
str(responseWater.hell)
```

```
## 'data.frame':    51 obs. of  3 variables:
##  $ DD13C.diss   : num  NA 0.0232 NA 0.0214 0.0273 ...
##  $ SM.g.nrm.prc : num  0.15 0.115 0.262 0.284 0.283 ...
##  $ MEL.g.nrm.prc: num  0.989 0.993 0.965 0.959 0.959 ...
##  - attr(*, "decostand")= chr "hellinger"
```

```r
str(waterXY.hell)
```

```
## 'data.frame':    51 obs. of  9 variables:
```

```
##  $ maxQ             : num  0.0138 0.0145 0.0158 0.0765 0.0533 ...
##  $ Duration.Hrs     : num  0.0675 0.1773 0.1197 0.1019 0.0939 ...
##  $ AveDischarge.m3.h: num  0.0234 0.0235 0.0242 0.0807 0.084 ...
##  $ Volume.m3        : num  0.019 0.05 0.0347 0.0987 0.0947 ...
##  $ DD13C.diss       : num  NA 0.28 NA 0.113 0.101 ...
##  $ DIa              : num  0.00143 0.00151 0.00169 0.02733 0.01985 ...
##  $ DIb              : num  0.192 0.183 0.173 0.119 0.178 ...
##  $ SM.g.nrm.prc     : num  0.382 0.342 0.349 0.368 0.257 ...
##  $ MEL.g.nrm.prc    : num  0.478 0.56 0.244 0.236 0.165 ...
##  - attr(*, "decostand")= chr "hellinger"
```

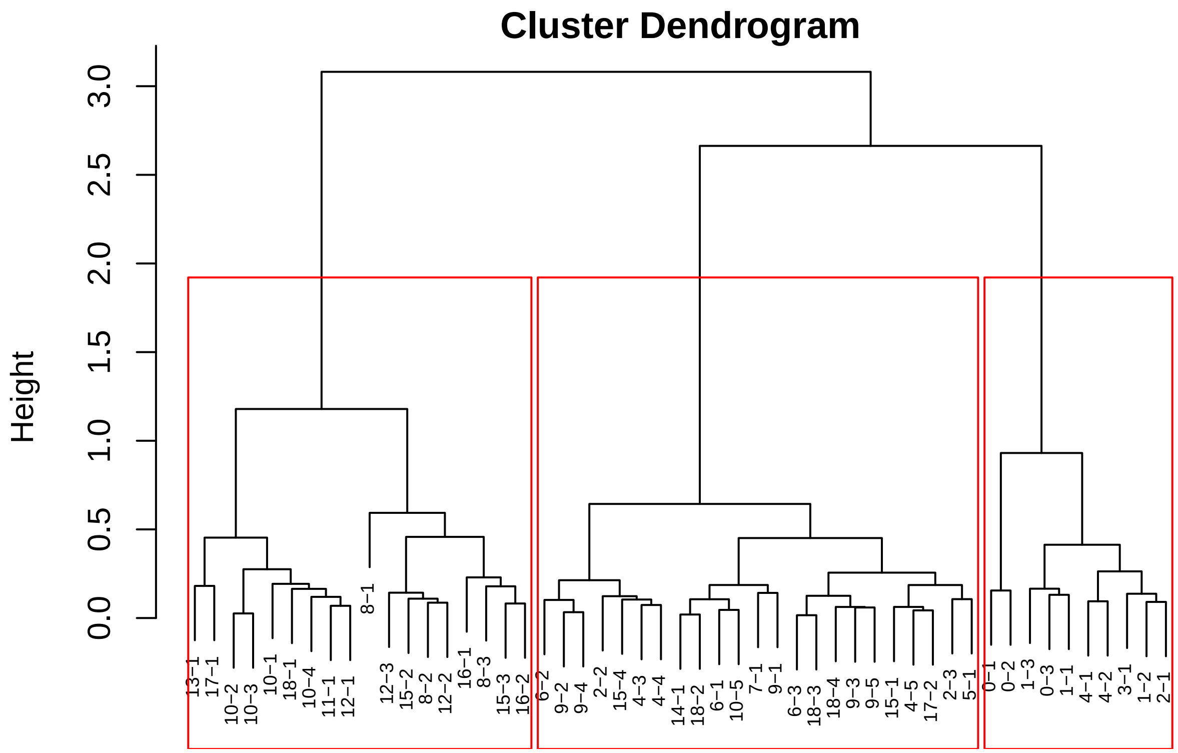
Clustering

First compute the dissimilarity matrix of the normalized response variables, which are normalized loads and $\Delta\delta^{13}C$ values.

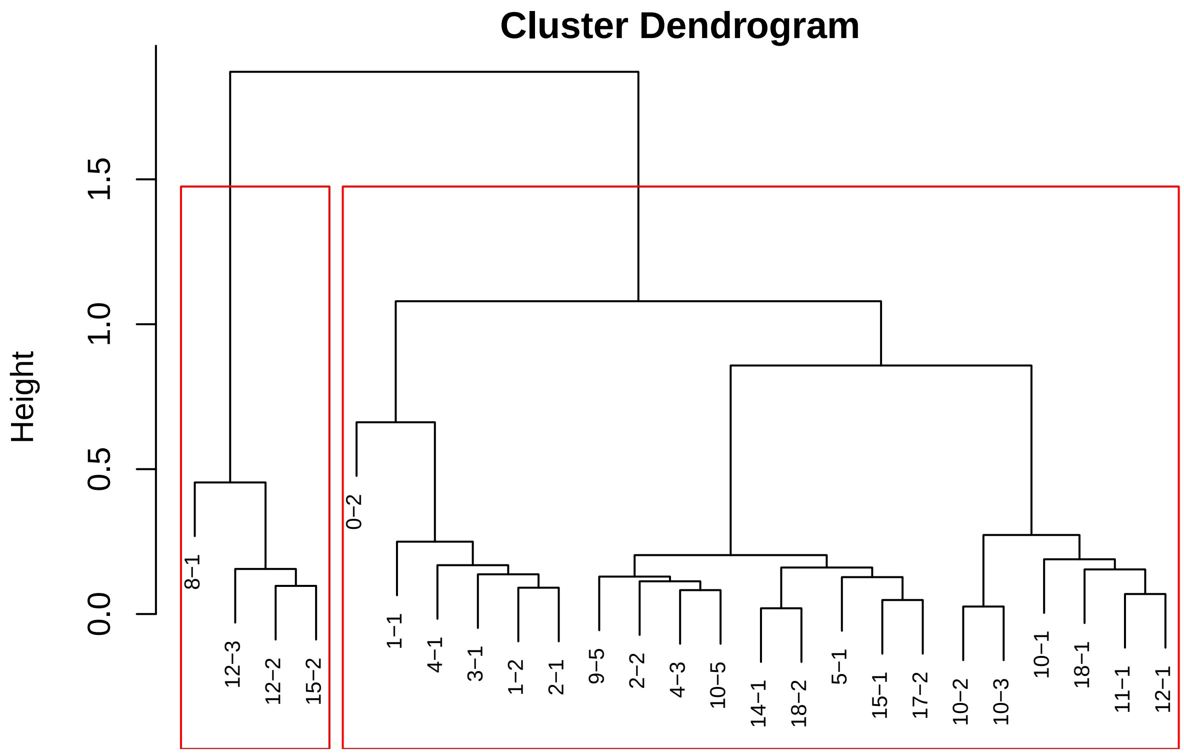
Hierarchical clustering and Ward agglomeration (minimum variance)

We'll be using "eucledian" distances, which are the root sum-of-squares of differences and the defining clusters via hierarchical clustering (hclust) using the "Ward" agglomeration method. Ward's method (Ward, 1963) determines which clusters to merge by evaluating the 'cost' of such a merge against an objective function. Merges with the minimum cost are performed at each stage of the algorithm. Typically, this is implemented by evaluating the sum of squared deviations from cluster centroids. Every possible merge is evaluated at each stage of the algorithm and that which yields the smallest increase in the sum of squared deviations is selected.

```r
# Compute dissimilarity and distance matrices (Q mode)
# resWater.dh = vegdist(responseWater.hell, "euclidean") # Hellinger
# vegdist doesn't allow for NAs
resWater.dh = daisy(responseWater.hell, "euclidean") # Hellinger
waterXY.dh = daisy(waterXY.hell, "euclidean")
waterXY.dh.nona = dist(waterXY.hell.nona, method = "euclidean")

# Clusterings based on the species/rows dataset
resWater.hw = hclust(resWater.dh, "ward.D") # Minimum variance clustering
waterXY.hw = hclust(waterXY.dh, "ward.D")
waterXY.hw.nona = hclust(waterXY.dh.nona, "ward.D")

# Plot dendrograms of Hellinger distance based clusterings
#windows(5,10)
# par(mfrow=c(1,1))

# The mar command defines plot margins in order bottom, left, up, right using
# row height (text height) as a unit.
par(mar=c(3,4,1,1)+.1)

#plot(resWater.hw, method ="ward.D", xlab="", sub="")
plot(waterXY.hw, labels = waterXY$Events, cex=0.6,
     method ="ward.D", xlab="With NAs", sub="")

rect.hclust(waterXY.hw, 3)
```

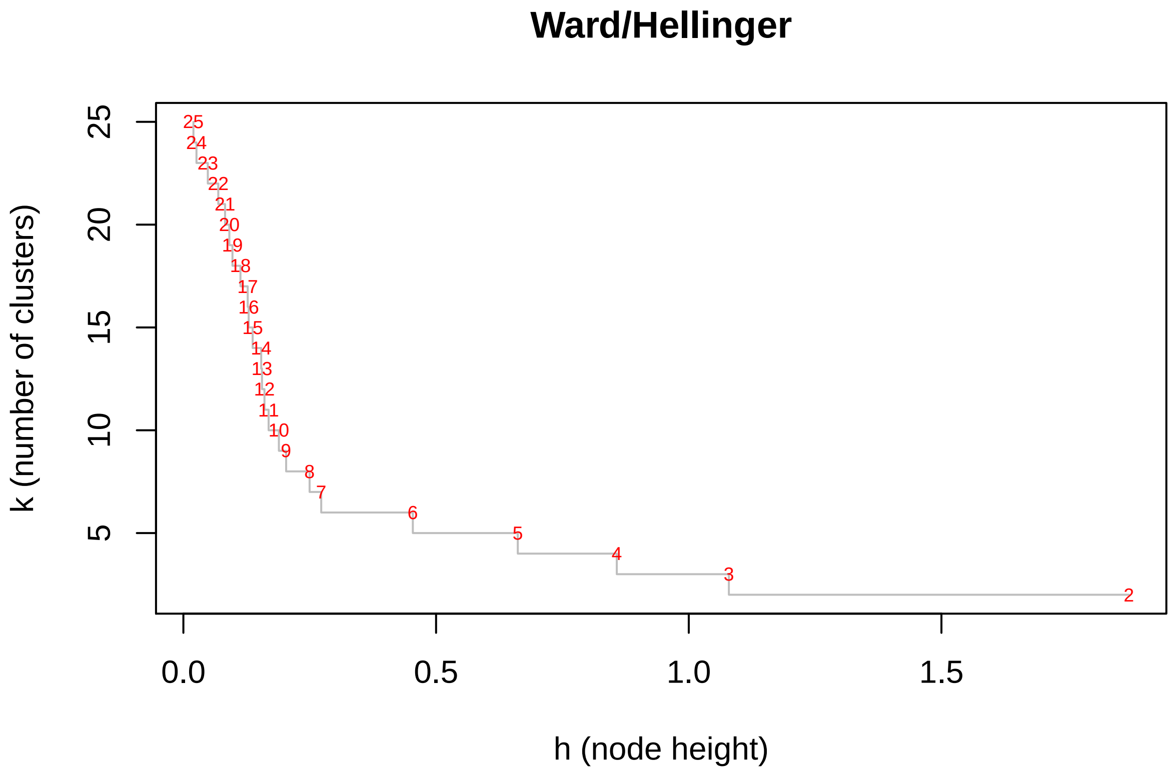


```
plot(waterXY.hw.nona, labels = waterXY.nona$Events, cex=0.7,
     method ="ward.D", xlab="Without NAs", sub="")
rect.hclust(waterXY.hw.nona, 2)
```



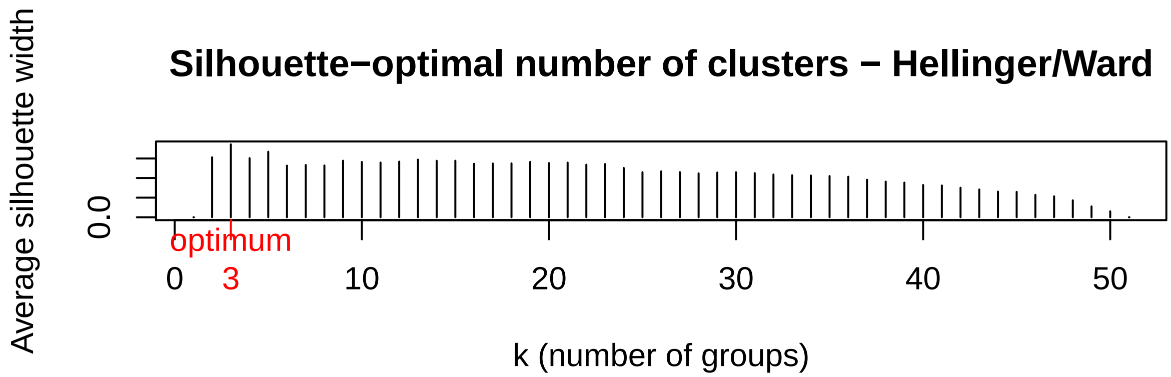
Plotting K-nodes vs. node height

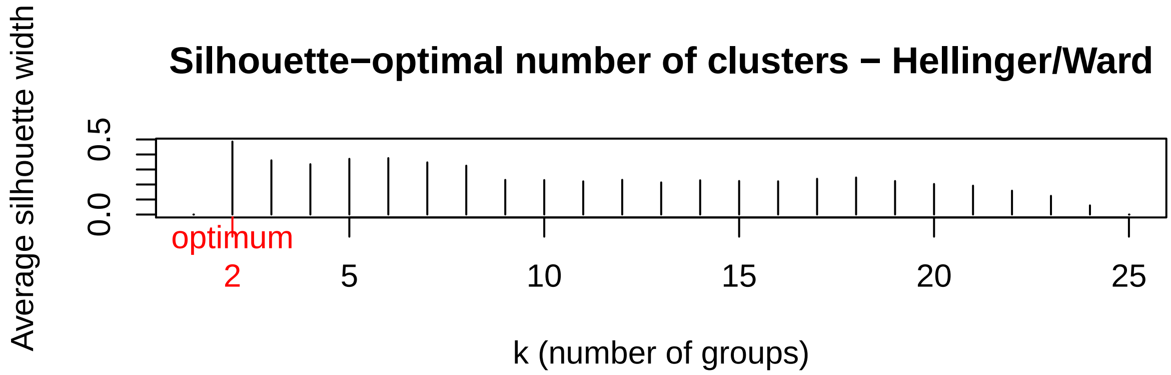
```
# Hellinger distance based clustering
# windows(8,8)
## par(mfrow=c(2,2))

whichDistPlot <- function(x, x.hw){
  plot(x.hw$height, nrow(x):2, type="S", main="Ward/Hellinger",
       ylab="k (number of clusters)", xlab="h (node height)", col="grey")
  text(x.hw$height, nrow(x):2, nrow(x):2, col="red", cex=0.6)
}

# whichDistPlot(responseWater, resWater.hw)
# whichDistPlot(waterXY, waterXY.hw)
whichDistPlot(waterXY.nona, waterXY.hw.nona)
```



Optimal number of clusters

```
# Average silhouette width (Rousseeuw internal quality index)
# Hellinger/Ward
# windows(16,8)
# par(mfrow=c(1,2))

findOptimal <- function(x, x.hw, x.dh) {
  Si = numeric(nrow(x))

  for (k in 2:(nrow(x)-1)) {
    sil = silhouette(cutree(x.hw, k=k), x.dh)
    Si[k] = summary(sil)$avg.width
  }

  k.best = which.max(Si)
  plot(1:nrow(x), Si, type="h", main="Silhouette-optimal number of clusters - Hellinger/Ward",
       xlab="k (number of groups)", ylab="Average silhouette width")
  axis(1, k.best, paste("optimum",k.best,sep="\n"), col="red", col.axis="red")
}
par(mfrow=c(2,1))
findOptimal(waterXY, waterXY.hw, waterXY.dh)
findOptimal(waterXY.nona, waterXY.hw.nona, waterXY.dh.nona)
```





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

PCA Eigenvalues

Note that currently, we have not conducted any data inputation, so we are working with less observations (due to missing isotope observations).

Plot the eigen values for each component and a line depicting the mean eigen value, below which components will not be considered.

```
# The original data frame
waterXY.nona <- waterXY[complete.cases(waterXY), ]

# The transformed data frame
waterXY.hell.nona = waterXY.hell[complete.cases(waterXY.hell),]

# PCA on a covariance matrix (default scale=FALSE)
waterXY.pca = rda(waterXY.hell.nona)


# Automatic scaling (no need for previous transform)
waterXY.pca2 <- rda(waterXY.nona[2:ncol(waterXY.nona)], scale = T)

waterXY.pca2
```

```
## Call: rda(X = waterXY.nona[2:ncol(waterXY.nona)], scale = T)
```

```
##
##                Inertia Rank
## Total                9
## Unconstrained        9    9
## Inertia is correlations
##
## Eigenvalues for unconstrained axes:
##   PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8   PC9
## 3.376 2.243 1.281 0.845 0.649 0.365 0.156 0.075 0.009
```

```
# plot (waterXY.pca2)
summary(waterXY.pca2, scaling = 1)
```

```
##
## Call:
## rda(X = waterXY.nona[2:ncol(waterXY.nona)], scale = T)
##
## Partitioning of correlations:
##                Inertia Proportion
## Total                9          1
## Unconstrained        9          1
##
## Eigenvalues, and their contribution to the correlations
##
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Eigenvalue            3.3764 2.2433 1.2815 0.84503 0.64936 0.36502 0.15579
## Proportion Explained  0.3751 0.2492 0.1424 0.09389 0.07215 0.04056 0.01731
## Cumulative Proportion 0.3751 0.6244 0.7668 0.86068 0.93283 0.97339 0.99070
##                           PC8      PC9
## Eigenvalue            0.07491 0.008783
## Proportion Explained  0.00832 0.000980
## Cumulative Proportion 0.99902 1.000000
##
## Scaling 1 for species and site scores
## * Sites are scaled proportional to eigenvalues
## * Species are unscaled: weighted dispersion equal on all dimensions
## * General scaling constant of scores:  3.833659
##
##
## Species scores
##
##                         PC1     PC2     PC3     PC4     PC5     PC6
## maxQ                2.00300 -0.3461  0.1771 -0.6669 -0.5342  0.4742
## Duration.Hrs       -1.38503  0.6522  1.3803 -0.6931 -2.0226  2.1123
## AveDischarge.m3.h  1.92679 -0.5333  0.5556  0.5132 -0.2964 -0.4412
## Volume.m3          -0.06418  2.0433  0.7113 -1.0975 -1.2672 -2.6688
## DD13C.diss          0.14796 -0.3054  3.0292 -0.4615  1.8572 -0.2236
## DIa                 1.86576 -0.5002  0.3925 -0.4267 -1.6467  0.4906
## DIb                -0.71225 -1.2680  0.9696  2.7319 -1.4592 -1.0974
## SM.g.nrm.prc       -0.65731 -1.9311 -1.0159 -1.5808 -0.5431 -1.0116
## MEL.g.nrm.prc      -0.77954 -2.0029  0.8023 -1.3951 -0.2046 -0.4177
##
##
## Site scores (weighted sums of species scores)
```

```
##
##          PC1       PC2       PC3      PC4        PC5       PC6
## 2  -0.76719 -1.342197  0.526294 -0.36607 -1.565e-02  0.012604
## 4  -0.30778 -0.504935 -0.603570 -0.37082 -5.064e-02 -0.108577
## 5  -0.29658 -0.376541 -0.398224  0.29198 -1.611e-01 -0.108276
## 7  -0.23141 -0.224013 -0.265488  0.33715 -3.532e-02 -0.008421
## 8  -0.20283 -0.082977  0.034273  0.37107  1.171e-01  0.074913
## 10 -0.19265  0.113734 -0.277998 -0.11937  2.416e-02  0.142971
## 11 -0.09332 -0.078442 -0.183361  0.05854 -5.987e-02 -0.275745
## 13 -0.23425  0.214689 -0.209073  0.14635 -1.496e-01  0.102545
## 16 -0.32052  0.521954  0.088871 -0.31995 -2.413e-01  0.163802
## 21  1.58153 -0.207625  0.005146 -0.32222 -5.303e-01  0.062146
## 28 -0.44798  0.115873  0.149029  0.29983 -3.377e-01  0.328267
## 29  0.07734  0.063095 -0.565150 -0.16378  2.228e-01  0.298384
## 30  0.20006 -0.014034 -0.018636  0.07620  1.758e-01 -0.107530
## 31  0.21338 -0.048772 -0.052903  0.13247  1.904e-01 -0.046624
## 33 -0.18407  0.359054  0.011494  0.06043 -1.584e-01 -0.230860
## 34  0.05947  0.235530 -0.211759 -0.16047  2.231e-01  0.069012
## 35  0.04114  0.333484 -0.177820 -0.06527 -2.779e-07 -0.210235
## 36  0.67100 -0.193724  0.231716  0.30568 -3.491e-02 -0.159400
## 37  0.45150  0.007028  0.136810  0.05614  1.605e-01 -0.114898
## 39 -0.17734  0.403721  0.276349 -0.16073  5.941e-02 -0.093347
## 40 -0.33686  0.355977  0.442774 -0.05670 -1.200e-01  0.006128
## 41  0.80046 -0.373962  0.243507  0.24603  2.473e-01  0.228833
## 47 -0.30356  0.102881  0.333777  0.21139 -1.409e-02  0.047245
## 48  0.19991  0.165963  0.193954 -0.28198  4.887e-01  0.039792
## 49 -0.19945  0.454239  0.289990 -0.20590 -1.750e-04 -0.112727
```

```
# Eigen values
(ev <- waterXY.pca2$CA$eig)
```

```
##         PC1         PC2         PC3         PC4         PC5         PC6
## 3.376375205 2.243251147 1.281480531 0.845033545 0.649364704 0.365019118
##         PC7         PC8         PC9
## 0.155786937 0.074905499 0.008783313
```

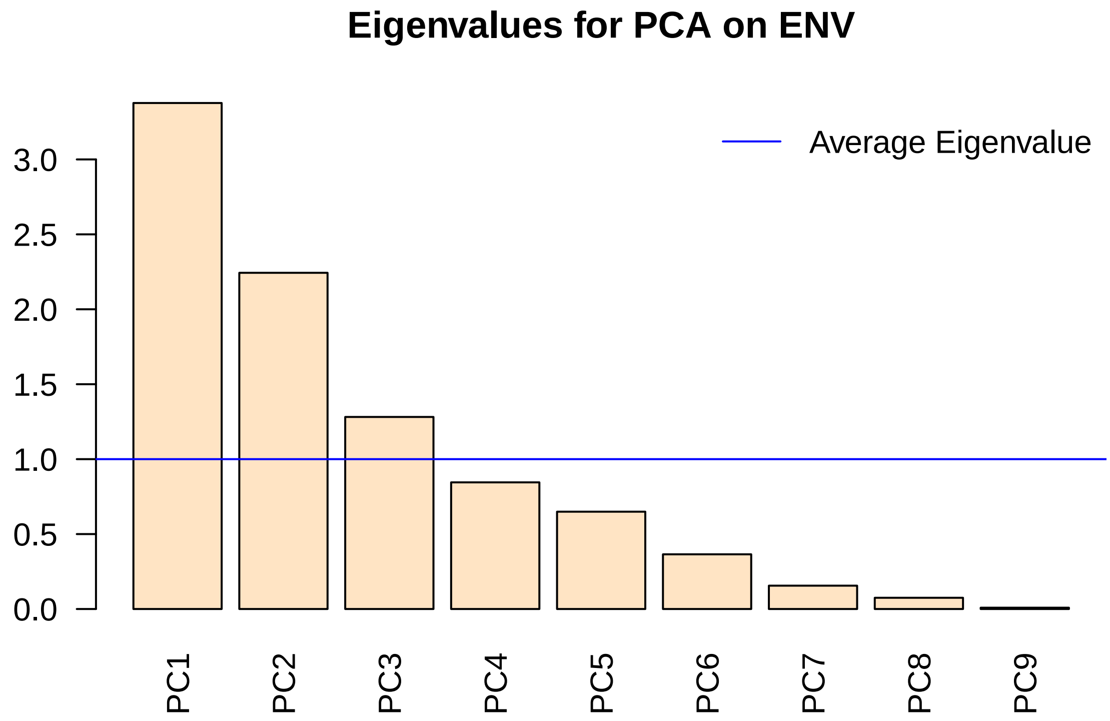
```
# Percentage of variance for each axis
100*ev/sum(ev)
```

```
##         PC1         PC2         PC3         PC4         PC5         PC6
## 37.51528006 24.92501275 14.23867256  9.38926162  7.21516337  4.05576798
##         PC7         PC8         PC9
##  1.73096597  0.83228333  0.09759237
```

```
# Apply Kaiser's rule to select axes
ev[ev > mean(ev)]
```

```
##      PC1      PC2      PC3
## 3.376375 2.243251 1.281481
```

```
# Plot eigen values and % variance for each axis
barplot(ev, main = "Eigenvalues for PCA on ENV", col = "bisque", las=2)
abline(h=mean(ev), col = "blue")
legend("topright", "Average Eigenvalue", lwd = 1, col = "blue", bty = "n")
```



PCA Biplots with clustering

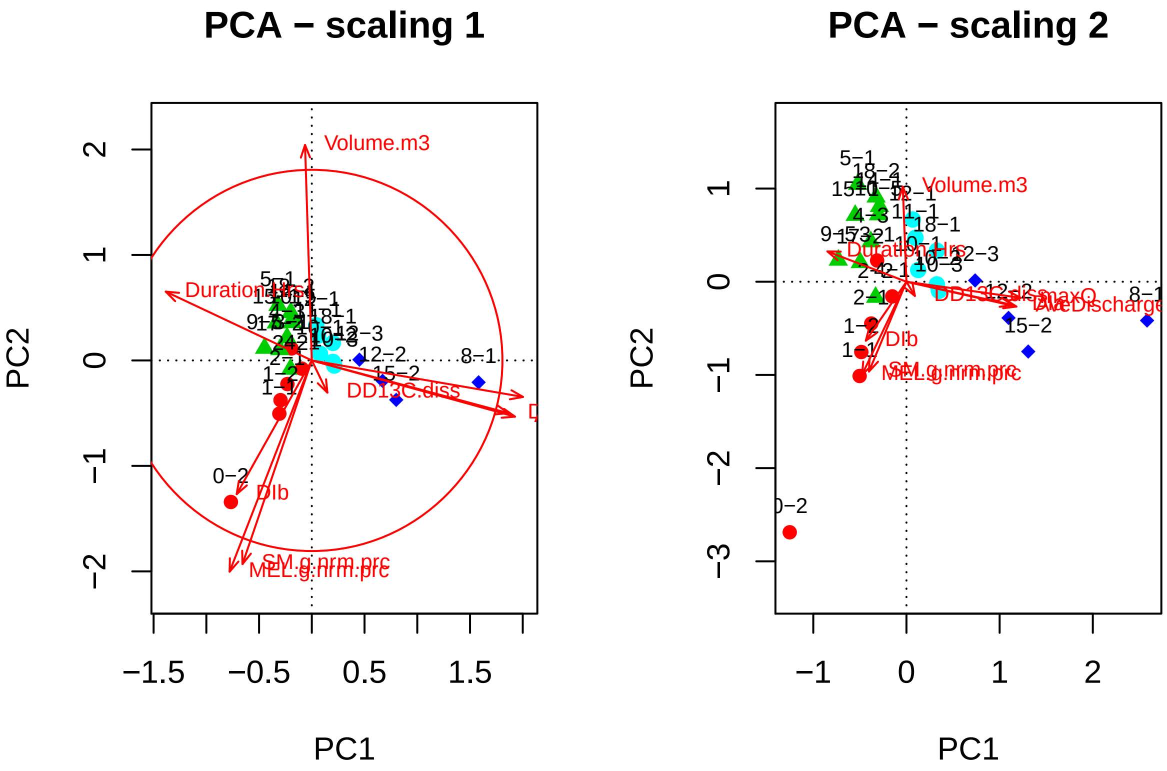
Scaling 1

- Distances between object points approximate the Euclidean distances between objects. Thus, objects ordinated closer together can be expected to have similar variable values.
- The length of a variable vector in the ordination plot reflects its contribution to the ordination
- Angles between variable vectors are meaningless

Scaling 2

- The angles between all vectors approximate their (linear) covariance/correlation.
- Distances between object points may be non-Euclidean and should not be interpreted with great confidence.

```r
# ,echo=FALSE, out.width='.49\\linewidth', fig.width=3, fig.height=3,fig.show='hold',fig.align='center'
source("cleanplotPCA.R")
# source ('http://www.davidzeleny.net/anadat-r/doku.php/en:numecolr:cleanplot.pca?do=export_code&codebl
# http://www.davidzeleny.net/anadat-r/doku.php/en:numecolr:cleanplot.pca
cleanplot.pca(waterXY.pca2, point = T,
              labs = waterXY.nona$Events, k = 4, dfcut = waterXY.hw.nona,
              cluster = TRUE)
```



```
# One plot
#k = 2
#gr = cutree(waterXY.hw.nona, k)
# Plot the sites with cluster symbols
#k = length(levels(factor(gr)))
#sit.sc = scores(waterXY.pca2, choices = c(1,2), display="wa", scaling=1)
#pl = plot(waterXY.pca2, display="sites", type="n", scaling=1,
#           main="PCA cov/Hell + clusters Ward/Hellinger")
# Plot the points with different symbols and colors
#points(sit.sc, cex=2, col=1+c(1:k)[gr], pch=15+c(1:k)[gr])
#text(sit.sc, rownames(waterXY.nona), pos=4, cex=.7)
# text(sit.sc, labels = waterXY.nona$Events, pos=4, cex=.7)
# legend(locator(1), paste("Group",c(1:k)), pch=14+c(1:k), col=1+c(1:k), pt.cex=2)
```

Based on Scaling 1, above suggest that…

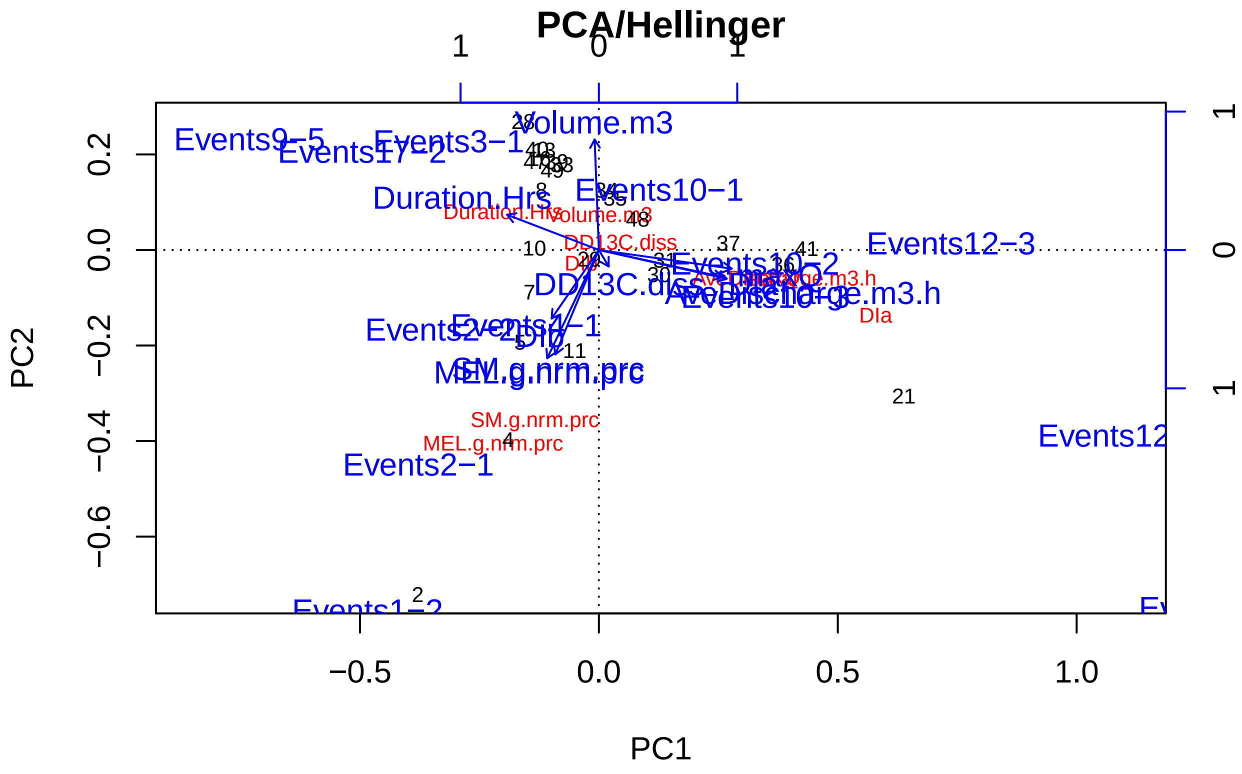
Environmental interpretation

```
# A posteriori interpretation of the species by significative environmental variables
## Selection of the significant variables
# windows(8,8)
# par(mfrow=c(1,1))
fit = envfit(waterXY.pca2, waterXY.nona, perm=1000)
fit
```

```
##
```

```
## ***VECTORS
## 
##                         PC1      PC2     r2   Pr(>r)    
## maxQ                0.99023 -0.13948 0.9400 0.000999 ***
## Duration.Hrs       -0.93360  0.35833 0.5056 0.000999 ***
## AveDischarge.m3.h   0.97548 -0.22009 0.8963 0.000999 ***
## Volume.m3          -0.03851  0.99926 0.6382 0.000999 ***
## DD13C.diss          0.51095 -0.85961 0.0193 0.828172    
## DIa                 0.97695 -0.21349 0.8379 0.000999 ***
## DIb                -0.56745 -0.82341 0.3619 0.006993 ** 
## SM.g.nrm.prc       -0.38534 -0.92278 0.6685 0.001998 ** 
## MEL.g.nrm.prc      -0.43088 -0.90241 0.7519 0.000999 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Permutation: free
## Number of permutations: 1000
## 
## ***FACTORS:
## 
## Centroids:
##                 PC1     PC2
## Events0-2  -1.2526 -2.6884
## Events1-1  -0.5025 -1.0114
## Events1-2  -0.4842 -0.7542
## Events2-1  -0.3778 -0.4487
## Events2-2  -0.3312 -0.1662
## Events3-1  -0.3145  0.2278
## Events4-1  -0.1524 -0.1571
## Events4-3  -0.3825  0.4300
## Events5-1  -0.5233  1.0455
## Events8-1   2.5821 -0.4159
## Events9-5  -0.7314  0.2321
## Events10-1  0.1263  0.1264
## Events10-2  0.3266 -0.0281
## Events10-3  0.3484 -0.0977
## Events10-5 -0.3005  0.7192
## Events11-1  0.0971  0.4718
## Events12-1  0.0672  0.6680
## Events12-2  1.0955 -0.3880
## Events12-3  0.7372  0.0141
## Events14-1 -0.2895  0.8087
## Events15-1 -0.5500  0.7130
## Events15-2  1.3069 -0.7490
## Events17-2 -0.4956  0.2061
## Events18-1  0.3264  0.3324
## Events18-2 -0.3256  0.9098
## 
## Goodness of fit:
##        r2 Pr(>r)
## Events  1      1
## Permutation: free
## Number of permutations: 1000
```

```
plot(waterXY.pca, type="t", main=paste("PCA/Hellinger"))
plot(fit, axis=T)
```



```
# waterXY.pca2
```

Non-metric Multidimensional Scaling

Non-metric multidimensional scaling (NMDS) is an indirect gradient analysis approach which produces an ordination based on a dissimilarity matrix. In other words, NMDS maps community dissimilarities into ordination space.

Unlike methods which attempt to maximise the variance between objects in an ordination, NMDS attempts to represent the pairwise dissimilarity between objects in a low-dimensional space. Any dissimilarity coefficient or distance measure may be used to build the distance matrix used as input (so you may need to normalize anyway?).

NMDS is a rank-based approach. This means that the original distance data is substituted with ranks. While information about the magnitude of distances is lost, rank-based methods are generally more robust to data which do not have an identifiable distribution.

NMDS can: - tolerate missing pairwise distances - be applied to a (dis)similarity matrix built with any (dis)similarity measure and - use quantitative, semi-quantitative, qualitative, or mixed variables

Jargon

- Stress (S): is a goodness of fit statistic

How many dimensions?

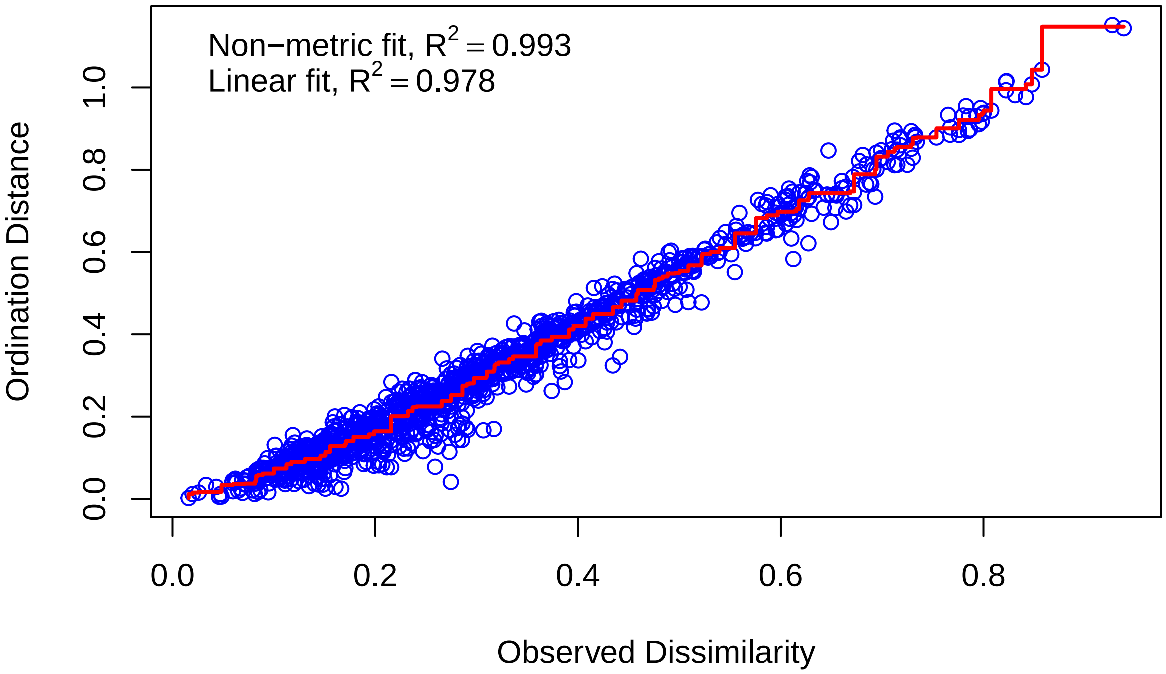
As a rule of thumb, an NMDS ordination with a stress value around or above 0.2 is deemed suspect and a stress value approaching 0.3 indicates that the ordination is arbitrary. Stress values equal to or below 0.1 are considered fair, while values equal to or below 0.05 indicate good fit. Allowing the algorithm to ordinate in more dimensions can reduce the stress; however, allowing more than 3 dimensions quickly makes interpretation more challenging.

Shepard stress plot shows the relationship between the actual dissimilarities between objects (from the original dissimilarity matrix) and the ordination distances (i.e. the distances on the final plot). If these are well correlated, the ordination stress will be low and the visualisation trustworthy. If there is a large amount of scatter (i.e. a poor linear relationship), then the ordination is not representative of the original distances. Occasionally, specific objects may be ordinated poorly (blue arrow), despite the overall solution being acceptable.

```r
# Normalized data with NAs (waterXY.hell)
# Dissimilarity matrix (waterXY.dh)
waterXY.mds0 <- isoMDS(waterXY.dh)
```

```
## initial  value 11.435552
## iter   5 value 8.848462
## iter  10 value 8.586669
## iter  15 value 8.516880
## iter  15 value 8.508646
## final  value 8.389745
## converged
```

```r
# Sheperd plot
stressplot(waterXY.mds0, waterXY.dh)
```


Non−metric fit, $R^2 = 0.993$
Linear fit, $R^2 = 0.978$
Ordination Distance
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
Observed Dissimilarity