Non-metric multidimensional scaling (NMDS)

PAZ

8 août 2017

Introduction

Non-metric multidimensional scaling (NMDS) is an indirect gradient analysis approach which produces an ordination (plot) based on a dissimilarity matrix. Points represent objects. Objects that are more similar to one another are ordinated closer together. The axes are arbitrary as is the orientation of the plot. Stress values should always accompany an NMDS ordination.

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 (like an outlier), despite the overall solution being acceptable.

ImportsFiles

- **Data/WaterSoils_R.csv** (Book 10.1 Clustering)

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 session.
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")

library("zoo")

# Melting data sets & changin axes
library("reshape2")
```

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

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

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

sw = read.csv2("Data/WaterSoils_R.csv")

sw$Date.ti <- as.character(sw$Date.ti)
sw$Date.ti <- as.POSIXct(strptime(sw$Date.ti, "%Y-%m-%d %H:%M", tz="EST"))
```

Standardize Event-reltaed (waterX) variables for NMDS analysis

See: https://sites.google.com/site/mb3gustame/reference/transformations

Given that my original motivation was to transform data to aid comparability of variables (columns) with different magnitudes, scales and different quantities (hrs, m3/h, m3), I've settled for transformations employing both translation (substraction by a scalar quantity) and expansion (ividing (or multiplying) by a scalar

quantity) available via the **scale()** function which results in a mean of 0 and a standard deviation of 1 for each column. This is called:

- **Z-scoring** (The mean of each variable is subtracted from the original values and the difference divided by the variable's standard deviation). Standardised values differ from one another in units of standard deviation, a key difference to ranging.

```
waterX <- sw[c("Date.ti", "Events",
           # "dryHrs",
           "maxQ", "AveDischarge.m3.h", # "chExtreme",
           "Duration.Hrs",
           "Volume.m3" #, "DIa",
           #"ExpMES.Kg" #, "Sequence"
           #"DD13C.diss",  "DD.diss.norm", "SM.g.nrm.prc", "MEL.g.nrm.prc"
           )]

y <- sw[c("Events",
          # Response variables
          # "Conc.mug.L", "OXA_mean", "ESA_mean",
          "DD.diss.nrm", "SM.g.nrm", "TP.g.nrm")]

names(waterX)
```

```
## [1] "Date.ti"           "Events"            "maxQ"
## [4] "AveDischarge.m3.h" "Duration.Hrs"      "Volume.m3"
```

```
if ( class(waterX[, 2])== "factor") {
  # Hellinger
  waterX.hell <- decostand(waterX[, 3:ncol(waterX)], "hellinger", na.rm=T, MARGIN = 2) # Margin 2 = col
  # Normalize to 1
  # make margin sum of squares equal to one (default MARGIN = 1)
  # waterX.norm <- decostand(waterX[, 2:ncol(waterX)], "norm", na.rm=T, MARGIN = 2) # Margin 2 = column

  waterX.z <- scale(waterX[, 3:ncol(waterX)])
}

# Test:
colMeans(waterX.z) # mean = 0
```

```
##              maxQ AveDischarge.m3.h      Duration.Hrs         Volume.m3
##     -1.360567e-19      2.217725e-17     -3.285770e-17     -7.455910e-17
```

```
apply(waterX.z, 2, sd) # SD = 1
```

```
##              maxQ AveDischarge.m3.h      Duration.Hrs         Volume.m3
##                 1                 1                 1                 1
```

Dissimilarity matrix

See: https://sites.google.com/site/mb3gustame/reference/dissimilarity

*Note: The (dis)similarity matrix is computed directly in the NDMS function below (**metaMDS**), so not necessary here. However, it is done before as well to find otpimal clusters for plotting afterwards.*

Steps following the Gustame dissimilarity wizard (https://sites.google.com/site/mb3gustame/wizards/-dis-similarity-wizard):

1. I'd like to know how (dis)similar my objects (rows) are
2. The variables (columns) describing my objects (rows) are more like: physicochemical data
3. Association is to be measured between: individual objects (not group of objects)
4. For my variables: partial (dis)similarities can be calculated between them (as opposed from being mutually exclusive or prescence/abscence)
5. Variables are: Quantitative and dimensionally homogeneous (i.e. have the same units) -> Yes, as they have been standardized above.

Results suggest an association for objects described by quantitative and homogeneous variables with four valid alternatives (chosen in **bold**):

I. **Euclidean distance (D1)** II. Average distance (D2) III. Manhattan metric (D7) IV. Mean character difference (D8)

Note: In **D1** double zeros result in decreased distances. This property makes the Euclidean distance unsuitable for many ecological data sets and ecologically-motivated transformations should be considered.

Package: **daisy()** (can compute a Gower coefficient for both quantitative and categorical variables) Package: **dist()** (needed for the ordiplot)

```
# Compute dissimilarity and distance matrices (Q mode)
# waterY.dh = vegdist(waterY.hell, "euclidean") # Hellinger
# vegdist doesn't allow for NAs
waterX.z.daisy =  daisy(waterX.z, "euclidean")
```

Hierarchical clustering and Ward agglomeration (minimum variance)

Find optimal clustering in the X-variables so as to identify groups to inspect. Not necessary for NDMS, but handy to graph afterwards.

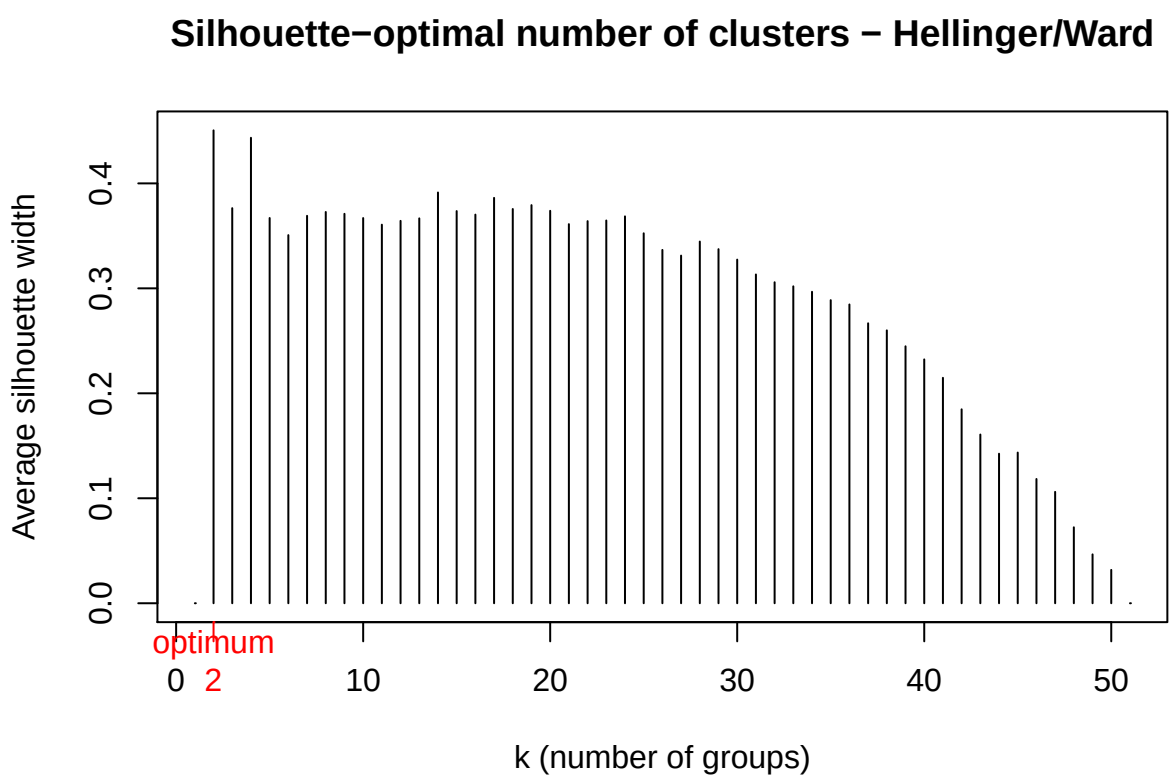
```
findOptimal <- function(x, x.hw, x.dh) {
  # 1st arg: dataframe
  # 2nd arg: clustered object
  # 3rd arg: transformed (e.g. normalized) data frame
  Si = numeric(nrow(x))

  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)) # Plot two rows, one column

# Clusterings based on the species/rows distances
waterX.z.clust = hclust(waterX.z.daisy,  "ward.D")

k <- findOptimal(waterX, waterX.z.clust, waterX.z.daisy)[1] # Optimal number of groups
```




```
gr = cutree(waterX.z.clust, k = k)

waterX <- cbind(gr, waterX)
names(waterX)[names(waterX) == "gr"] <- "Cluster"
```

Now onto NDMS

See: https://sites.google.com/site/mb3gustame/dissimilarity-based-methods

Look like we do not need to prepare a distance matrix before the NDMS computations (**metaMDS**), only take the standardized matrix.

```
# Does not need
# waterX.z.dist =  dist(waterX.z, "euclidean")
# waterX.hell.dist = dist(waterX.hell,  "euclidean")

# Replace the distance matrix to plot
# the chosen standardization
standardized.dist <- waterX.z

# Create the NMDS object
# Function metaMDS uses isoMDS to perform Nonmetric Multidimensional Scaling (NMDS),
# but tries to find a stable solution using several random starts (function initMDS).
# Improved method by Jari Oksanen (with projection of species)
waterX.nmds <- metaMDS(standardized.dist, distance="euc")
```

```
## Run 0 stress 0.04790458
## Run 1 stress 0.06352918
## Run 2 stress 0.1325742
## Run 3 stress 0.06352854
## Run 4 stress 0.04790452
## ... New best solution
## ... Procrustes: rmse 0.000234695  max resid 0.001466281
## ... Similar to previous best
## Run 5 stress 0.1437559
## Run 6 stress 0.1298171
## Run 7 stress 0.1431208
## Run 8 stress 0.06352914
## Run 9 stress 0.04790431
## ... New best solution
## ... Procrustes: rmse 0.0001475858  max resid 0.0009232754
## ... Similar to previous best
## Run 10 stress 0.04790488
## ... Procrustes: rmse 0.0002117785  max resid 0.001325004
## ... Similar to previous best
## Run 11 stress 0.1241214
## Run 12 stress 0.04790446
## ... Procrustes: rmse 0.0001308473  max resid 0.0008184782
## ... Similar to previous best
## Run 13 stress 0.06353011
## Run 14 stress 0.04790437
## ... Procrustes: rmse 0.0001032018  max resid 0.0006457761
## ... Similar to previous best
## Run 15 stress 0.04790479
## ... Procrustes: rmse 0.0001985557  max resid 0.001240424
## ... Similar to previous best
## Run 16 stress 0.1241171
## Run 17 stress 0.06352962
## Run 18 stress 0.1433275
## Run 19 stress 0.1241192
## Run 20 stress 0.04790456
## ... Procrustes: rmse 0.0001540109  max resid 0.0009613564
## ... Similar to previous best
## *** Solution reached
```

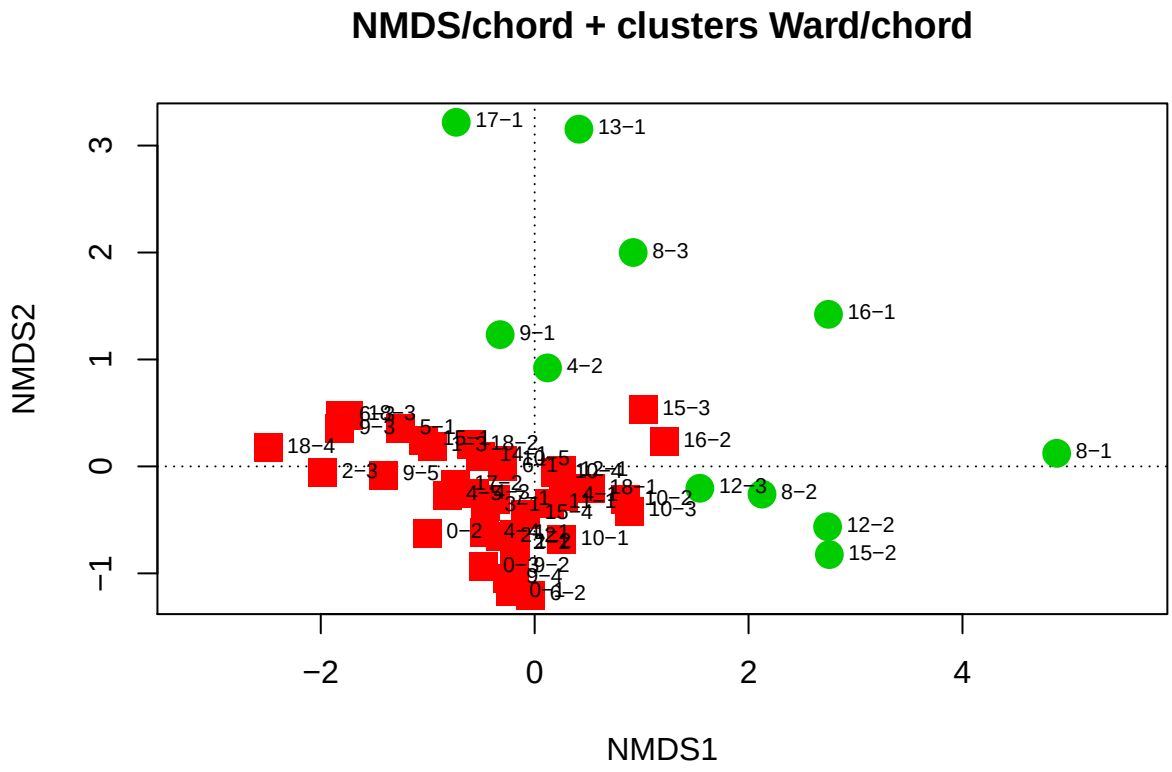
```
waterX.nmds
```

```
##
## Call:
## metaMDS(comm = standardized.dist, distance = "euc")
##
## global Multidimensional Scaling using monoMDS
##
## Data:     standardized.dist
## Distance: euclidean
##
## Dimensions: 2
## Stress:     0.04790431
## Stress type 1, weak ties
## Two convergent solutions found after 20 tries
## Scaling: centring, PC rotation
```

```
## Species: scores missing
```

```
#ordiplot(waterX.nmds) # Without row number, if row number: type = "t"
#text(waterX.mds, labels = waterX$Events) #, display="wa", choices=c(ax1, ax2), cex=cex, pos=3, scaling=
```

```
# Plot the sites (rows) with cluster symbols
    # windows(30,30)
    k = length(levels(factor(gr)))
    sit.sc = scores(waterX.nmds)
    pl = ordiplot(waterX.nmds, type="n", display="sites", main="NMDS/chord + clusters Ward/chord")
    abline(h=0, lty=3)
    abline(v=0, lty=3)
# Plot the points with different symbols and colors
    points(sit.sc, cex=2, col=1+c(1:k)[gr], pch=14+c(1:k)[gr])
    text(sit.sc, labels = waterX$Events, pos=4, cex=.7)
```




```
# Add a legend for groups
#   legend(locator(1), paste("Group",c(1:k)), pch=14+c(1:k), col=1+c(1:k), pt.cex=2)
```

What does it mean?

Points represent objects. Objects that are more similar to one another are ordinated closer together. The axes are arbitrary as is the orientation of the plot. Stress values should always accompany an NMDS ordination.

```
# Stress
waterX.nmds$stress
```

```
## [1] 0.04790431
```

```
# Sheperd plot
stressplot(waterX.nmds, standardized.dist)
```