VEGAN: AN INTRODUCTION TO ORDINATION

JARI OKSANEN

Contents

`Vegan` is a package for community ecologists. This documents explains how the commonly used ordination methods can be done in `vegan`. The document only is a very basic introduction. Another document (*vegan tutorial*) (`http://cc.oulu.fi/~jarioksa/opetus/method/vegantutor.pdf`) gives a longer and more detailed introduction to ordination. The current document only describes a small part of all `vegan` functions. For most functions, the canonical references are the `vegan` help pages, and some of the most important additional functions are listed at this document.

1. Ordination

The `vegan` package contains all common ordination methods: Principal component analysis (function `rda`, or `prcomp` in the base R), correspondence analysis (`cca`), detrended correspondence analysis (`decorana`) and a wrapper for non-metric multidimensional scaling (`metaMDS`). Functions `rda` and `cca` mainly are designed for constrained ordination, and will be discussed later. In this chapter I describe functions `decorana` and `metaMDS`.

1.1. Detrended correspondence analysis.

Detrended correspondence analysis (DCA) is done like this:

```
> library(vegan)
> data(dune)
> ord <- decorana(dune)
```

This saves ordination results in `ord`:

```
> ord
```

Date: Id: intro-vegan.Rnw 508 2008-09-26 11:47:00Z jarioksa processed with vegan 1.15-3 in R version 2.9.0 (2009-04-17) on June 17, 2009.

```
Call:
decorana(veg = dune)

Detrended correspondence analysis with 26 segments.
Rescaling of axes with 4 iterations.

                  DCA1   DCA2    DCA3    DCA4
Eigenvalues     0.5117 0.3036 0.12125 0.14266
Decorana values 0.5360 0.2869 0.08136 0.04814
Axis lengths    3.7004 3.1166 1.30057 1.47883
```

The display of results is very brief: only eigenvalues and used options are listed. Actual ordination results are not shown, but you can see them with command `summary(ord)`, or extract the scores with command `scores`. The `plot` function also automatically knows how to access the scores.

1.2. **Non-metric multidimensional scaling.** Function `metaMDS` is a bit special case. The actual ordination is performed by function `isoMDS` of the `MASS` package. Function `metaMDS` is a wrapper to perform non-metric multidimensional scaling (NMDS) like recommended in community ordination: it uses adequate dissimilarity measures (function `vegdist`), then it runs NMDS several times with random starting configurations, compares results (function `procrustes`), and stops after finding twice a similar minimum stress solution. Finally it scales and rotates the solution, and adds species scores to the configuration as weighted averages (function `wascores`):

```
> ord <- metaMDS(dune)

Run 0 stress 12.05894
Run 1 stress 12.04546
... New best solution
... procrustes: rmse 0.003131675  max resid 0.01073634
Run 2 stress 11.97274
... New best solution
... procrustes: rmse 0.01992551  max resid 0.0627578
Run 3 stress 11.97273
... New best solution
... procrustes: rmse 5.583258e-05  max resid 0.0001340292
*** Solution reached

> ord

Call:
metaMDS(comm = dune)

Nonmetric Multidimensional Scaling using isoMDS (MASS package)

Data:     dune
Distance: bray

Dimensions: 2
Stress:     11.97273
Two convergent solutions found after 3 tries
Scaling: centring, PC rotation, halfchange scaling
Species: expanded scores based on 'dune'
```

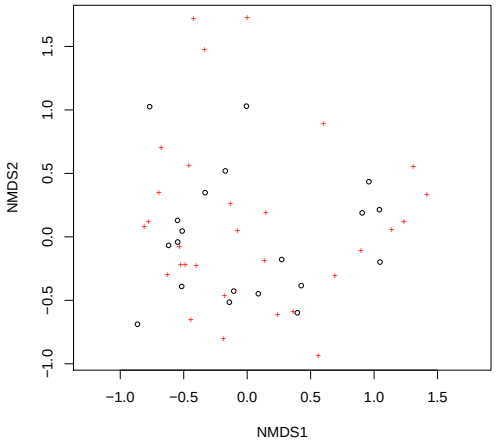

FIGURE 1. Default ordination plot.

2. Ordination graphics

Ordination is nothing but a way of drawing graphs, and it is best to inspect ordinations only graphically (which also implies that they should not be taken too seriously).

All ordination results of `vegan` can be displayed with a `plot` command (Fig. 1):

```
> plot(ord)
```

Default `plot` command uses either black circles for sites and red pluses for species, or black and red text for sites and species, resp. The choices depend on the number of items in the plot and ordination method. You can override the default choice by setting `type = "p"` for points, or `type = "t"` for text. For a better control of ordination graphics you can first draw an empty plot (`type = "n"`) and then add species and sites separately using `points` or `text` functions. In this way you can combine points and text, and you can select colours and character sizes freely (Fig. 2):

```
> plot(ord, type = "n")
> points(ord, display = "sites", cex = 0.8, pch = 21, col = "red",
+     bg = "yellow")
> text(ord, display = "spec", cex = 0.7, col = "blue")
```

All `vegan` ordination methods have a specific `plot` function. In addition, `vegan` has an alternative plotting function `ordiplot` that also knows many non-`vegan` ordination methods, such as `prcomp`, `cmdscale` and `isoMDS`. All `vegan` plot functions return invisibly an `ordiplot` object, so that you can use `ordiplot` support functions with the results (`points`, `text`, `identify`).

Function `ordirgl` (requires `rgl` package) provides dynamic three-dimensional graphics that can be spun around or zoomed into with your mouse. Function `ordiplot3d` (requires package `scatterplot3d`) displays simple three-dimensional scatterplots.

2.1. Cluttered plots.

Ordination plots are often congested: there is a large number of sites and species, and it may be impossible to display all clearly. In particular, two or more species may have identical scores and are plotted over each other. `Vegan` does not have (yet?) automatic tools for clean plotting in these cases, but here some methods you can try:

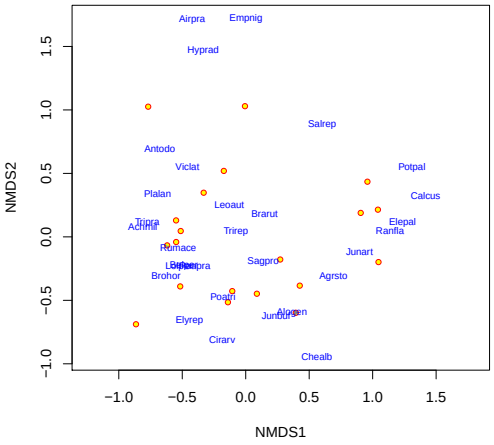

FIGURE 2. A more colourful ordination plot where sites are points, and species are text.

- Zoom into graph setting axis limits `xlim` and `ylim`. You must typically set both, because `vegan` will maintain equal aspect ratio of axes.
- Use points and label only some of these with `identify` command.
- Use `select` argument in ordination `text` and `points` functions to only show the specified items.
- Use `ordilabel` function that uses opaque background to the text: some text labels will be covered, but the uppermost are readable.
- Use automatic `orditorp` function that uses text only if this can be done without overwriting previous labels, but points in other cases.
- Use automatic `ordipointlabel` function that uses both points and text labels, and tries to optimize the location of the text to avoid overwriting.
- Use interactive `orditkplot` function that draws both points and labels for ordination scores, and allows you to drag labels to better positions. You can export the results of the edited graph to encapsulated postscript, pdf, png or jpeg files, or copy directly to encapsulated postscript, or return the edited positions to R for further processing.

2.2. **Adding items to ordination plots.** `Vegan` has a group of functions for adding information about classification or grouping of points onto ordination diagrams. Function `ordihull` adds convex hulls, `ordiellipse` (which needs package `ellipse`) adds ellipses of standard deviation, standard error or confidence areas, and `ordispider` combines items to their centroid (Fig. 3):

```
> data(dune.env)
> attach(dune.env)
> plot(ord, disp = "sites", type = "n")
> ordihull(ord, Management, col = "blue")
> ordiellipse(ord, Management, col = 3, lwd = 2)
> ordispider(ord, Management, col = "red")
> points(ord, disp = "sites", pch = 21, col = "red", bg = "yellow",
+     cex = 1.3)
```

In addition, you can overlay a cluster dendrogram from `hclust` using `ordicluster` or a minimum spanning tree from `spantree` with its `lines` function. Segmented arrows can be added with `ordiarrows`, lines with `ordisegments` and regular grids with `ordigrid`.

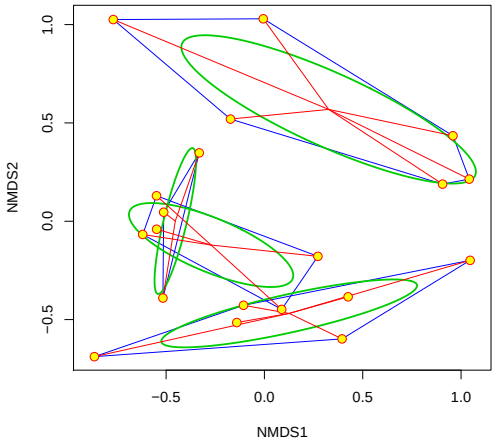

FIGURE 3. Convex hull, standard error ellipse and a spider web diagram for Management levels in ordination.

3. Fitting environmental variables

`Vegan` provides two functions for fitting environmental variables onto ordination:

- `envfit` fits vectors of continuous variables and centroids of levels of class variables (defined as `factor` in R). The direction of the vector shows the direction of the gradient, and the length of the arrow is proportional to the correlation between the variable and the ordination.
- `ordisurf` (which requires package `mgcv`) fits smooth surfaces for continuous variables onto ordination using thinplate splines with cross-validatory selection of smoothness.

Function `envfit` can be called with a `formula` interface, and it optionally can assess the "significance" of the variables using permutation tests:

```
> ord.fit <- envfit(ord ~ A1 + Management, data = dune.env,
+     perm = 1000)
> ord.fit
***VECTORS

     NMDS1   NMDS2     r2  Pr(>r)
A1 0.97951 0.20141 0.3689 0.02398 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
P values based on 1000 permutations.

***FACTORS:

Centroids:
                NMDS1   NMDS2
ManagementBF -0.4532  0.0012
ManagementHF -0.2712 -0.1209
ManagementNM  0.3266  0.5688
ManagementSF  0.1260 -0.4686

Goodness of fit:
               r2   Pr(>r)
```

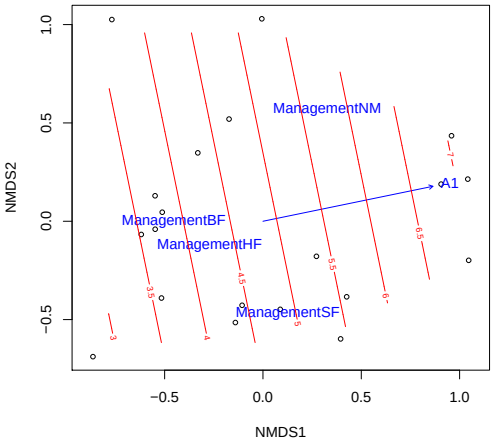

FIGURE 4. Fitted vector and smooth surface for the thickness of A1 horizon (`A1`, in cm), and centroids of Management levels.

```
Management 0.4206 0.002997 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
P values based on 1000 permutations.
```

The result can be drawn directly or added to an ordination diagram (Fig. 4):

```
> plot(ord, dis = "site")
> plot(ord.fit)
```

Function `ordisurf` directly adds a fitted surface onto ordination, but it returns the result of the fitted thinplate spline `gam` (Fig. 4):

```
> ordisurf(ord, A1, add = TRUE)
This is mgcv  1.5-2 . For overview type `help("mgcv-package")'.

Family: gaussian
Link function: identity

Formula:
y ~ s(x1, x2, k = knots)

Estimated degrees of freedom:
2  total = 3

GCV score: 3.940938
```

4. Constrained ordination

`Vegan` has three methods of constrained ordination: constrained or “canonical” correspondence analysis (function `cca`), redundancy analysis (function `rda`) and constrained analysis of proximities (function `capscale`). All these functions also can have a conditioning term that is “partialled out”. I only demonstrate `cca`, but all functions accept similar commands and can be used in the same way.

The preferred way is to use `formula` interface, where the left hand side gives the community data frame and the right hand side lists the constraining variables:

```
> ord <- cca(dune ~ A1 + Management, data = dune.env)
> ord
```

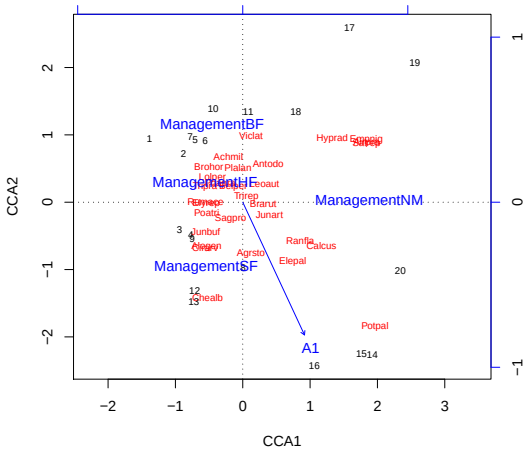

FIGURE 5. Default plot from constrained correspondence analysis.

```
Call: cca(formula = dune ~ A1 + Management, data = dune.env)

              Inertia Rank
Total          2.1153
Constrained    0.7798    4
Unconstrained  1.3355   15
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
   CCA1    CCA2    CCA3    CCA4
0.31875 0.23718 0.13217 0.09168

Eigenvalues for unconstrained axes:
     CA1      CA2      CA3      CA4      CA5      CA6      CA7      CA8
0.362024 0.202884 0.152661 0.134549 0.110957 0.079982 0.076698 0.055267
     CA9     CA10     CA11     CA12     CA13     CA14     CA15
0.044361 0.041528 0.031699 0.017786 0.011642 0.008736 0.004711
```

The results can be plotted with (Fig. 5):

```
> plot(ord)
```

There are three groups of items: sites, species and centroids (and biplot arrows) of environmental variables. All these can be added individually to an empty plot, and all previously explained tricks of controlling graphics still apply.

It is not recommended to perform constrained ordination with all environmental variables you happen to have: adding the number of constraints means slacker constraint, and you finally end up with solution similar to unconstrained ordination. In that case it is better to use unconstrained ordination with environmental fitting. However, if you really want to do so, it is possible with the following shortcut in `formula`:

```
> cca(dune ~ ., data = dune.env)
Call: cca(formula = dune ~ A1 + Moisture + Management + Use +
Manure, data = dune.env)

              Inertia Rank
Total          2.1153
```

```
Constrained    1.5032   12
Unconstrained  0.6121    7
Inertia is mean squared contingency coefficient
Some constraints were aliased because they were collinear (redundant)

Eigenvalues for constrained axes:
   CCA1    CCA2    CCA3    CCA4    CCA5    CCA6    CCA7    CCA8    CCA9
0.46713 0.34102 0.17606 0.15317 0.09528 0.07027 0.05887 0.04993 0.03183
  CCA10   CCA11   CCA12
0.02596 0.02282 0.01082

Eigenvalues for unconstrained axes:
    CA1     CA2     CA3     CA4     CA5     CA6     CA7
0.27237 0.10876 0.08975 0.06305 0.03489 0.02529 0.01798
```

4.1. **Significance tests.** `Vegan` provides permutation tests for the significance of constraints. The test mimics standard analysis of variance function (`anova`), and the default test analyses all constraints simultaneously:

```
> anova(ord)
```

```
Permutation test for cca under reduced model

Model: cca(formula = dune ~ A1 + Management, data = dune.env)
         Df  Chisq      F N.Perm Pr(>F)
Model     4 0.7798 2.1896    199  0.005 **
Residual 15 1.3355
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The function actually used was `anova.cca`, but you do not need to give its name in full, because R automatically chooses the correct `anova` variant for the result of constrained ordination.

The `anova.cca` function tries to be clever and lazy: it automatically stops if the observed permutation significance probably differs from the targeted critical value (0.05 as default), but it will continue long in uncertain cases. You must set `step` and `perm.max` to same values to override this behaviour.

It is also possible to analyse terms separately:

```
> anova(ord, by = "term", permu = 200)
```

```
Permutation test for cca under reduced model
Terms added sequentially (first to last)

Model: cca(formula = dune ~ A1 + Management, data = dune.env)
           Df  Chisq      F N.Perm Pr(>F)
A1          1 0.2248 2.5245    199  0.015 *
Management  3 0.5550 2.0780    199  0.010 **
Residual   15 1.3355
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

In this case, the function is unable to automatically select the number of iterations. This test is sequential: the terms are analysed in the order they happen to be in the model. You can also analyse significances of marginal effects (“Type III effects”):

```
> anova(ord, by = "mar")
```

```
Permutation test for cca under reduced model
Marginal effects of terms

Model: cca(formula = dune ~ A1 + Management, data = dune.env)
           Df  Chisq      F N.Perm Pr(>F)
A1          1 0.1759 1.9761    399  0.020 *
Management  3 0.5550 2.0780    199  0.005 **
Residual   15 1.3355
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Moreover, it is possible to analyse significance of each axis:

```
> anova(ord, by = "axis", perm = 500)
Permutation test for cca under reduced model

Model: cca(formula = dune ~ A1 + Management, data = dune.env)
         Df  Chisq      F N.Perm Pr(>F)
CCA1      1 0.3187 3.5801    299   0.02 *
CCA2      1 0.2372 2.6640    499   0.04 *
CCA3      1 0.1322 1.4845     99   0.32
CCA4      1 0.0917 1.0297     99   0.43
Residual 15 1.3355
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Now the automatic selection works, but typically some of your axes will be very close to the critical value, and it may be useful to set a lower `perm.max` than the default 10000 (typically you use higher limits than in these examples: we used lower limits to save time when this document is automatically generated with this package).

4.2. **Conditioned or partial ordination.** All constrained ordination methods can have terms that are partialled out from the analysis before constraints:

```
> ord <- cca(dune ~ A1 + Management + Condition(Moisture),
+     data = dune.env)
> ord
Call: cca(formula = dune ~ A1 + Management +
Condition(Moisture), data = dune.env)

              Inertia Rank
Total          2.1153
Conditional    0.6283    3
Constrained    0.5109    4
Unconstrained  0.9761   12
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
   CCA1    CCA2    CCA3    CCA4
0.24932 0.12090 0.08160 0.05904

Eigenvalues for unconstrained axes:
     CA1      CA2      CA3      CA4      CA5      CA6      CA7      CA8
0.306366 0.131911 0.115157 0.109469 0.077242 0.075754 0.048714 0.037582
```

```
     CA9     CA10     CA11     CA12
0.031058 0.021024 0.012542 0.009277
```

This partials out the effect of `Moisture` before analysing the effects of `A1` and `Management`. This also influences the signficances of the terms:

```
> anova(ord, by = "term", perm = 500)
Permutation test for cca under reduced model
Terms added sequentially (first to last)

Model: cca(formula = dune ~ A1 + Management + Condition(Moisture), data = dune.env)
           Df  Chisq      F N.Perm Pr(>F)
A1          1 0.1154 1.4190     99   0.14
Management  3 0.3954 1.6205     99   0.01 **
Residual   12 0.9761
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

If we had a designed experiment, we may wish to restrict the permutations so that the observations only are permuted within levels of `strata`:

```
> anova(ord, by = "term", perm = 500, strata = Moisture)
Permutation test for cca under reduced model
Terms added sequentially (first to last)
Permutations stratified within `Moisture'

Model: cca(formula = dune ~ A1 + Management + Condition(Moisture), data = dune.env)
           Df  Chisq      F N.Perm Pr(>F)
A1          1 0.1154 1.4190     99   0.34
Management  3 0.3954 1.6205     99   0.01 **
Residual   12 0.9761
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```