DESIGN DECISIONS AND IMPLEMENTATION DETAILS IN VEGAN

JARI OKSANEN

ABSTRACT. This document describes design decisions, and discusses implementation and algorithmic details in some vegan functions. The proper FAQ is another document.

CONTENTS

1. NESTEDNESS AND NULL MODELS

Some indicators of nestedness and null models of communities are only described in general terms, and they could be implemented in various ways. Here I discuss the implementation in `vegan`.

1.1. **Matrix temperature.** The matrix temperature is intuitively simple (Fig. 1), but the the exact calculations were not explained in the original publication [1]. The function can be implemented in many ways following the general principles. Rodríguez-Gironés and Santamaria [6] have seen the original code and reveal more details of calculations, and their explanation is the basis of the implementation in `vegan`. However, there are still some open issues, and probably `vegan` function `nestedtemp` will never exactly reproduce results from other programs, although it is based on the same general principles. I try to give main computation details in this documents — all details can be seen in the source code of `nestedtemp`.

- Species and sites are put into unit square [6]. The coordinates for n item will be $(k-0.5)/n$ for $k = 1 \dots n$, so that there are no points in the corners or the margins of the unit square, and a diagonal line can be drawn through any point. I do not know how the rows and columns are converted to the unit square in other software, and this may be a considerable source of differences among implementations.

Date: Id: decision-vegan.Rnw 1442 2011-01-08 07:54:53Z jarioksa processed with vegan 1.17-9 in R version 2.13.0 beta (2011-03-30 r55204) on March 31, 2011.

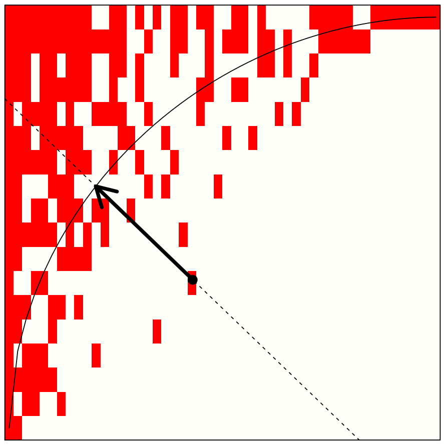

Figure 1. Matrix temperature for *Falco subbuteo* on Sibbo Svartholmen (dot). The curve is the fill line, and in a cold matrix, all presences (red squares) should be in the upper left corner behind the fill line. Dashed diagonal line of length D goes through the point, and an arrow of length d connects the point to the fill line. The "surprise" for this point is $u = (d/D)^2$ and the matrix temperature is based on the sum of surprises: presences outside the fill line or absences within the fill line.

- Species and sites are ordered alternately using indices [6]:

$$\begin{aligned} s_j &= \sum_{i|x_{ij}=1} i^2 \\ t_j &= \sum_{i|x_{ij}=0} (n-i+1)^2 \end{aligned} \tag{1}$$

  Here x is the data matrix, where 1 is presence, and 0 is absence, i and j are row and column indices, and n is the number of rows. The equations give the indices for columns, but the indices can be reversed for corresponding row indexing. Ordering by s packs presences to the top left corner, and ordering by t pack zeros away from the top left corner. The final sorting should be "a compromise" [6] between these scores, and `vegan` uses $s + t$. The result should be cool, but the packing does not try to minimize the temperature [6]. I do not know how the "compromise" is defined, and this can cause some differences to other implementations.
- The following function is used to define the fill line:

$$y = (1 - (1 - x)^p)^{1/p} \tag{2}$$

  This is similar to the equation suggested by [6, eq. 4], but omits all terms dependent on the numbers of species or sites, because I could not understand why they were needed. The differences are visible only in small data sets. The y and x are the coordinates in the unit square, and the parameter p is selected so that the curve covers the same area as is the proportion of presences (Fig. 1). The parameter p is found numerically using R functions `integrate` and `uniroot`. The fill line used in the original matrix temperature software [1] is supposed to be similar [6]. Small details in the fill line combined with differences in scores used in the unit square (especially in the corners) can cause large differences in the results.
- A line with slope -1 is drawn through the point and the x coordinate of the intersection of this line and the fill line is found using function `uniroot`. The difference of this intersection and the row coordinate gives the argument d of matrix temperature (Fig. 1).
- In other software, "duplicated" species occurring on every site are removed, as well as empty sites and species after reordering [6]. This is not done in `vegan`.

1.2. **Backtracking.** Gotelli's and Entsminger's seminal paper [2] on filling algorithms is somewhat confusing: it explicitly deals with "knight's tour" which is quite a different problem than the one we face with null models. The chess piece "knight"[1] has a history: a piece in a certain position could only have entered from some candidate squares. The filling of incidence matrix has not such a history: if we know that the item last added was in certain row and column, we have no information to guess which of the filled items was entered previously. A consequence of dealing with a different problem is that [2] does not give many hints on implementing a fill algorithm as a community null model.

The backtracking is implemented in two stages in **vegan**: filling and backtracking.

(1) The matrix is filled in the order given by the marginal probabilities. In this way the matrix will look similar to the final matrix at all stages of filling. Equal filling probabilities are not used since that is ineffective and produces strange fill patterns: the rows and columns with one or a couple of presences are filled first, and the process is cornered to columns and rows with many presences. As a consequence, the the process tries harder to fill that corner, and the result is a more tightly packed quadratic fill pattern than with other methods.

(2) The filling stage stops when no new points can be added without exceeding row or column totals. "Backtracking" means removing random points and seeing if this allows adding new points to the plot. No record of history is kept (and there is no reason to keep a record of history), but random points are removed and filled again. The number of removed points increases from one to four points. New configuration is kept if it is at least as good as the previous one, and the number of removed points is reduced back to one if the new configuration is better than the old one. Because there is no record of history, this does not sound like a backtracking, but it still fits the general definition of backtracking: "try something, and if it fails, try something else" [7].

2. Scaling in redundancy analysis

This chapter discusses the scaling of scores (results) in redundancy analysis and principal component analysis performed by function `rda` in the `vegan` library.

Principal component analysis, and hence redundancy analysis, is a case of singular value decomposition (SVD). Functions `rda` and `prcomp` even use SVD internally in their algorithm.

In SVD a centred data matrix is decomposed into orthogonal components so that $x_{ij} = \sum_k \sigma_k u_{ik} v_{jk}$, where u_{ik} and v_{jk} are orthonormal coefficient matrices and σ_k are singular values. Orthonormality means that sums of squared columns is one and their cross-product is zero, or $\sum_i u_{ik}^2 = \sum_j v_{jk}^2 = 1$, and $\sum_i u_{ik} u_{il} = \sum_j v_{jk} v_{jl} = 0$ for $k \neq l$. This is a decomposition, and the original matrix is found exactly from the singular vectors and corresponding singular values, and first two singular components give the rank = 2 least squares estimate of the original matrix.

Principal component analysis is often presented (and performed in legacy software) as an eigenanalysis of covariance matrices. Instead of a data matrix, we analyse a matrix of covariances and variances $\mathbf{S}$. The result are orthonormal coefficient matrix $\mathbf{U}$ and eigenvalues $\mathbf{\Lambda}$. The coefficients u_{ik} ares identical to SVD (except for possible sign changes), and eigenvalues λ_k are related to the corresponding singular values by $\lambda_k = \sigma_k^2/(n-1)$. With classical definitions, the sum of

[1]"Knight" is "Springer" in German which is very appropriate as Springer was the publisher of the paper on "knight's tour"

TABLE 1. Alternative scalings for RDA used in the functions `prcomp` and `princomp`, and the one used in the **vegan** function `rda` and the proprietary software `Canoco` scores in terms of orthonormal species (u_{ik}) and site scores (v_{jk}), eigenvalues (λ_k), number of sites (n) and species standard deviations (s_j). In `rda`, const $= \sqrt[4]{(n-1)\sum \lambda_k}$. Corresponding negative scaling in `vegan` is derived dividing each species by its standard deviation s_j (possibly with some additional constant multiplier).

	Site scores u^*_{ik}	**Species scores** v^*_{jk}
`prcomp, princomp`	$u_{ik}\sqrt{n-1}\sqrt{\lambda_k}$	v_{jk}
`rda, scaling=1`	$u_{ik}\sqrt{\lambda_k/\sum\lambda_k} \times \text{const}$	$v_{jk} \times \text{const}$
`rda, scaling=2`	$u_{ik} \times \text{const}$	$v_{jk}\sqrt{\lambda_k/\sum\lambda_k} \times \text{const}$
`rda, scaling=3`	$u_{ik}\sqrt[4]{\lambda_k/\sum\lambda_k} \times \text{const}$	$v_{jk}\sqrt[4]{\lambda_k/\sum\lambda_k} \times \text{const}$
`rda, scaling < 0`	u^*_{ik}	$\sqrt{\sum\lambda_k/(n-1)}s_j^{-1}v^*_{jk}$

all eigenvalues equals the sum of variances of species, or $\sum_k \lambda_k = \sum_j s_j^2$, and it is often said that first axes explain a certain proportion of total variance in the data. The orthonormal matrix $\mathbf{V}$ of SVD can be found indirectly as well, so that we have the same components in both methods.

The coefficients u_{ik} and v_{jk} are scaled similarly for all axes k. Singular values σ_k or eigenvalues λ_k give the information of the importance of axes, or the 'axis lengths.' Instead of the orthonormal coefficients, or equal length axes, it is customary to scale species (column) or site (row) scores or both by eigenvalues to display the importance of axes and to describe the true configuration of points. Table 1 shows some alternative scalings. These alternatives apply to principal components analysis in all cases, and in redundancy analysis, they apply to species scores and constraints or linear combination scores; weighted averaging scores have somewhat wider dispersion.

In community ecology, it is common to plot both species and sites in the same graph. If this graph is a graphical display of SVD, or a graphical, low-dimensional approximation of the data, the graph is called a biplot. The graph is a biplot if the transformed scores satisfy $x_{ij} = c\sum_k u^*_{ij} v^*_{jk}$ where c is a scaling constant. In functions `princomp`, `prcomp` and `rda`, $c = 1$ and the plotted scores are a biplot so that the singular values (or eigenvalues) are expressed for sites, and species are left unscaled.

There is no natural way of scaling species and site scores to each other. The eigenvalues in redundancy and principal components analysis are scale-dependent and change when the the data are multiplied by a constant. If we have percent cover data, the eigenvalues are typically very high, and the scores scaled by eigenvalues will have much wider dispersion than the orthonormal set. If we express the percentages as proportions, and divide the matrix by 100, the eigenvalues will be reduced by factor 100^2, and the scores scaled by eigenvalues will have a narrower dispersion. For graphical biplots we should be able to fix the relations of row and column scores to be invariant against scaling of data. The solution in R standard function `biplot` is to scale site and species scores independently, and typically very differently, but plot each independently to fill the graph area. The solution in `Canoco` and and `rda` is to use proportional eigenvalues $\lambda_k/\sum\lambda_k$ instead of original eigenvalues. These proportions are invariant with scale changes, and typically they have a nice range for plotting two data sets in the same graph.

The **vegan** package uses a scaling constant $c = \sqrt[4]{(n-1)\sum\lambda_k}$ in order to be able to use scaling by proportional eigenvalues (like in `Canoco`) and still be able to have a biplot scaling. Because of this, the scaling of `rda` scores is non-standard.

TABLE 2. Values of the `const` argument in **vegan** to get the scores that are equal to those from other functions and software. Number of sites (rows) is n, the number of species (columns) is m, and the sum of all eigenvalues is $\sum_k \lambda_k$ (this is saved as the item `tot.chi` in the `rda` result)

	Scaling	Species constant	Site constant
`vegan`	any	$\sqrt[4]{(n-1)\sum \lambda_k}$	$\sqrt[4]{(n-1)\sum \lambda_k}$
`prcomp`, `princomp`	`1`	1	$\sqrt{(n-1)\sum_k \lambda_k}$
`Canoco 3`	`-1, -2, -3`	$\sqrt{n-1}$	$\sqrt{n}$
`Canoco 4`	`-1, -2, -3`	$\sqrt{m}$	$\sqrt{n}$

However, the `scores` function lets you to set the scaling constant to any desired values. It is also possible to have two separate scaling constants: the first for the species, and the second for sites and friends, and this allows getting scores of other software or R functions (Table 2).

In this chapter, I used always centred data matrices. In principle SVD could be done with original, non-centred data, but there is no option for this in `rda`, because I think that non-centred analysis is dubious and I do not want to encourage its use (if you think you need it, you are certainly so good in programming that you can change that one line in `rda.default`). I do think that the arguments for non-centred analysis are often twisted, and the method is not very good for its intended purpose, but there are better methods for finding fuzzy classes. Normal, centred analysis moves the origin to the average of all species, and the dimensions describe differences from this average. Non-centred analysis leaves the origin in the empty site with no species, and the first axis usually runs from the empty site to the average site. Second and third non-centred components are often very similar to first and second (etc.) centred components, and the best way to use non-centred analysis is to discard the first component and use only the rest. This is better done with directly centred analysis.

3. Why to use weighted averages scores instead of linear combinations in constrained ordination

Constrained ordination methods such as Constrained Correspondence Analysis (CCA) and Redundancy Analysis (RDA) produce two kind of site scores [5, 8]:

- LC or Linear Combination Scores which are linear combinations of constraining variables.
- WA or Weighted Averages Scores which are such weighted averages of species scores that are as similar to LC scores as possible.

Many computer programs for constrained ordinations give only or primarily LC scores, following Mike Palmer's recommendation [5]. However, functions `cca` and `rda` in the `vegan` package use primarily WA scores. This chapter explains the reasons for this choice.

Briefly, the main reasons are that

- LC scores *are* linear combinations, so they give us only the (scaled) environmental variables. This means that they are independent of vegetation and cannot be found from the species composition. Moreover, identical combinations of environmental variables give identical LC scores irrespective of vegetation.

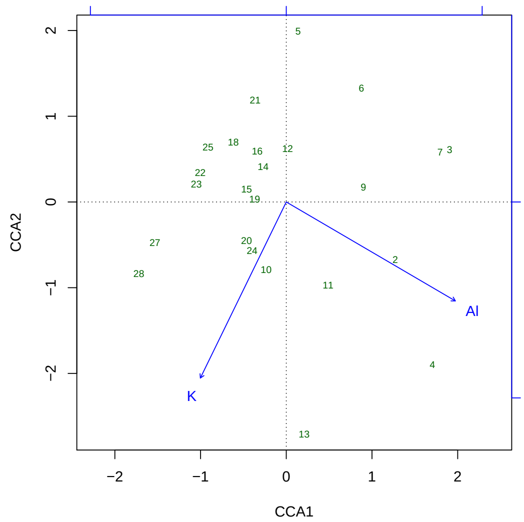

Figure 2. LC scores in CCA of the original data.

- Bruce McCune has demonstrated that noisy environmental variables result in deteriorated LC scores whereas WA scores tolerate some errors in environmental variables [4]. All environmental measurements contain some errors, and therefore it is safer to use WA scores.

This article studies mainly the first point. The users of `vegan` have a choice of either LC or WA (default) scores, but after reading this article, I believe that most of them do not want to use LC scores, because they are not what they were looking for in ordination.

3.1. **LC Scores are Linear Combinations.** Let us perform a simple CCA analysis using only two environmental variables so that we can see the constrained solution completely in two dimensions:

```
> library(vegan)
> data(varespec)
> data(varechem)
> orig <- cca(varespec ~ Al + K, varechem)
```

Function `cca` in `vegan` uses WA scores as default. So we must specifically ask for LC scores (Fig. 2).

```
> plot(orig, dis = c("lc", "bp"))
```

What would happen to linear combinations of LC scores if we shuffle the ordering of sites in species data? Function `sample()` below shuffles the indices.

```
> i <- sample(nrow(varespec))
> shuff <- cca(varespec[i, ] ~ Al + K, varechem)
```

It seems that site scores are fairly similar, but oriented differently (Fig. 3). We can use Procrustes rotation to see how similar the site scores indeed are (Fig. 4).

```
> plot(procrustes(scores(orig, dis = "lc"), scores(shuff, dis = "lc")))
```

There is a small difference, but this will disappear if we use Redundancy Analysis (RDA) instead of CCA (Fig. 5). Here we use a new shuffling as well.

```
> tmp1 <- rda(varespec ~ Al + K, varechem)
> i <- sample(nrow(varespec))
> tmp2 <- rda(varespec[i, ] ~ Al + K, varechem)
```

LC scores indeed are linear combinations of constraints (environmental variables) and *independent of species data*: You can shuffle your species data, or change the

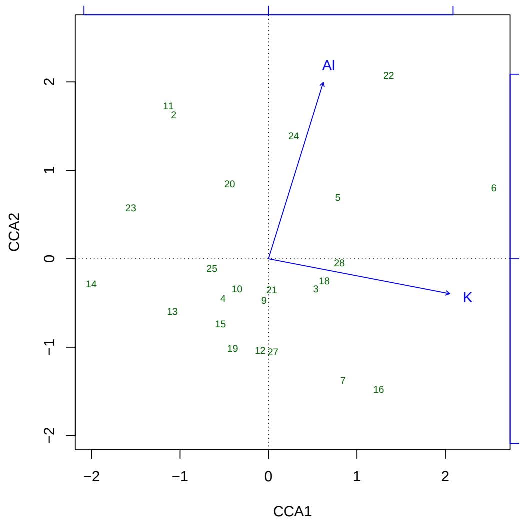

FIGURE 3. LC scores of shuffled species data.

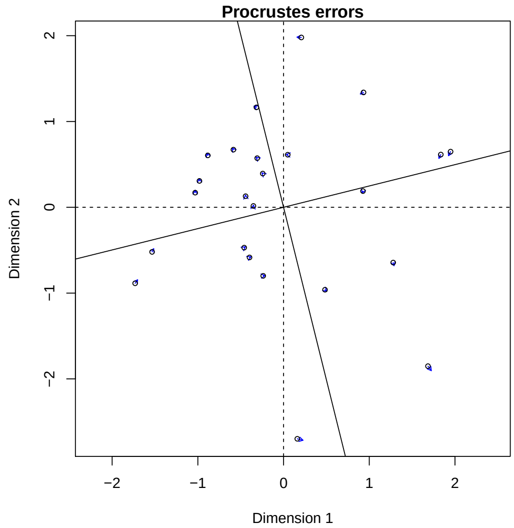

FIGURE 4. Procrustes rotation of LC scores from CCA of original and shuffled data.

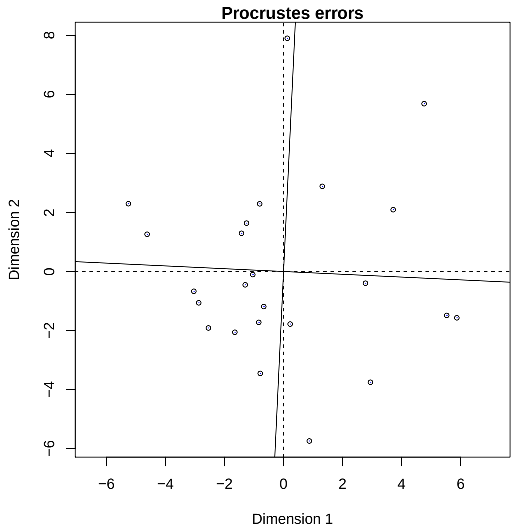

FIGURE 5. Procrustes rotation of LC scores in RDA of the original and shuffled data.

data completely, but the LC scores will be unchanged in RDA. In CCA the LC scores are *weighted* linear combinations with site totals of species data as weights. Shuffling species data in CCA changes the weights, and this can cause changes in LC scores. The magnitude of changes depends on the variability of site totals.

The original data and shuffled data differ in their goodness of fit[2].

```
> orig
Call: cca(formula = varespec ~ Al + K, data = varechem)

              Inertia Proportion Rank
Total          2.0832     1.0000
Constrained    0.4760     0.2285    2
Unconstrained  1.6072     0.7715   21
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
  CCA1   CCA2
0.3608 0.1152

Eigenvalues for unconstrained axes:
    CA1     CA2     CA3     CA4     CA5     CA6     CA7     CA8
0.37476 0.24036 0.19696 0.17818 0.15209 0.11840 0.08364 0.07567
(Showed only 8 of all 21 unconstrained eigenvalues)
> shuff
Call: cca(formula = varespec[i, ] ~ Al + K, data = varechem)

              Inertia Proportion Rank
Total         2.08320    1.00000
Constrained   0.10461    0.05022    2
Unconstrained 1.97859    0.94978   21
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
   CCA1    CCA2
0.07888 0.02573

Eigenvalues for unconstrained axes:
    CA1     CA2     CA3     CA4     CA5     CA6     CA7     CA8
0.51971 0.34596 0.21486 0.19188 0.17284 0.11517 0.10951 0.08847
(Showed only 8 of all 21 unconstrained eigenvalues)
```

Similarly their WA scores will be (probably) very different (Fig. 6).

The example used only two environmental variables so that we can easily plot all constrained axes. With a larger number of environmental variables the full configuration remains similarly unchanged, but its orientation may change, so that two-dimensional projections look different. In the full space, the differences should remain within numerical accuracy:

```
> tmp1 <- rda(varespec ~ ., varechem)
> tmp2 <- rda(varespec[i, ] ~ ., varechem)
> proc <- procrustes(scores(tmp1, dis = "lc", choi = 1:14), scores(tmp2,
```

[2]Or probably differ: The randomization is done while generating this article, and different versions may have different randomizations.

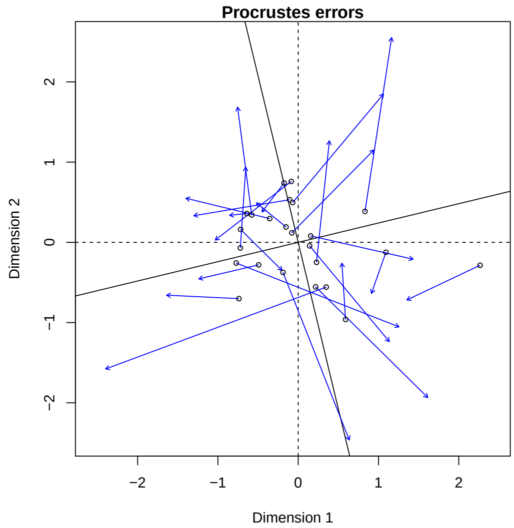

FIGURE 6. Procrustes rotation of WA scores of CCA with the original and shuffled data.

```
+      dis = "lc", choi = 1:14))
> max(residuals(proc))
[1] 3.068336e-14
```

In `cca` the difference would be somewhat larger than now observed 3.0683e-14 because site weights used for environmental variables are shuffled with the species data.

3.2. **Factor constraints.** It seems that users often get confused when they perform constrained analysis using only one factor (class variable) as constraint. The following example uses the classical dune meadow data [3]:

```
> data(dune)
> data(dune.env)
> summary(dune.env)
       A1          Moisture Management       Use      Manure
 Min.   : 2.800   1:7      BF:3       Hayfield:7   0:6
 1st Qu.: 3.500   2:4      HF:5       Haypastu:8   1:3
 Median : 4.200   4:2      NM:6       Pasture :5   2:4
 Mean   : 4.850   5:7      SF:6                    3:4
 3rd Qu.: 5.725                                    4:3
 Max.   :11.500
> orig <- cca(dune ~ Moisture, dune.env)
```

When the results are plotted using LC scores, sample plots fall only in four alternative positions (Fig. 7). In the previous chapter we saw that this happens because LC scores *are* the environmental variables, and they can be distinct only if the environmental variables are distinct. However, normally the user would like to see how well the environmental variables separate the vegetation, or inversely, how we could use the vegetation to discriminate the environmental conditions. For this purpose we should plot WA scores, or LC scores and WA scores together: The LC scores show where the site *should* be, the WA scores shows where the site *is*.

Function `ordispider` adds line segments to connect each WA score with the corresponding LC (Fig. 8).

```
> plot(orig, display = "wa", type = "points")
> ordispider(orig, col = "red")
> text(orig, dis = "cn", col = "blue")
```

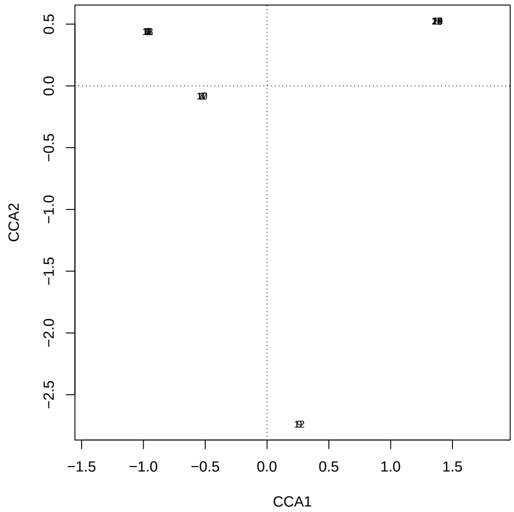

FIGURE 7. LC scores of the dune meadow data using only one factor as a constraint.

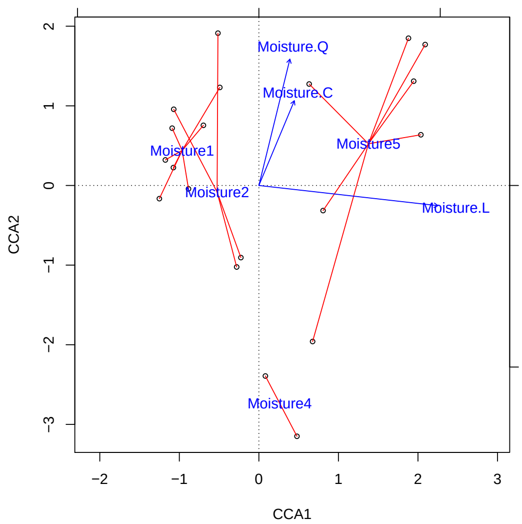

FIGURE 8. A "spider plot" connecting WA scores to corresponding LC scores. The shorter the web segments, the better the ordination.

This is the standard way of displaying results of discriminant analysis, too. Moisture classes `1` and `2` seem to be overlapping, and cannot be completely separated by their vegetation. Other classes are more distinct, but there seems to be a clear arc effect or a "horseshoe" despite using CCA.

3.3. **Conclusion.** LC scores are only the (weighted and scaled) constraints and independent of vegetation. If you plot them, you plot only your environmental variables. WA scores are based on vegetation data but are constrained to be as similar to the LC scores as only possible. Therefore `vegan` calls LC scores as `constraints` and WA scores as `site scores`, and uses primarily WA scores in plotting. However, the user makes the ultimate choice, since both scores are available.

REFERENCES

[1] W. Atmar and B. D. Patterson. The measure of order and disorder in the distribution of species in fragmented habitat. *Oecologia*, 96:373–382, 1993.
[2] N. J. Gotelli and G. L. Entsminger. Swap and fill algorithms in null model analysis: rethinking the knight's tour. *Oecologia*, 129:281–291, 2001.

[3] R. H. Jongman, C. J. F. ter Braak, and O. F. R. van Tongeren. *Data analysis in community and landscape ecology*. Pudoc, Wageningen, 1987.

[4] B. McCune. Influence of noisy environmental data on canonical correspondence analysis. *Ecology*, 78:2617–2623, 1997.

[5] M. W. Palmer. Putting things in even better order: The advantages of canonical correspondence analysis. *Ecology*, 74:2215–2230, 1993.

[6] M. A. Rodríguez-Gironés and L. Santamaria. A new algorithm to calculate the nestedness temperature of presence–absence matrices. *Journal of Biogeography*, 33:921–935, 2006.

[7] R. Sedgewidk. *Algorithms in C*. Addison Wesley, 1990.

[8] C. J. F. ter Braak. Canonical correspondence analysis: a new eigenvector technique for multivariate direct gradient analysis. *Ecology*, 67:1167–1179, 1986.