Constrained Ordination: Tutorial with R and vegan

Jari Oksanen

January 26, 2012

Abstract

Constrained ordination methods include constrained (or canonical) correspondence analysis (CCA), redundancy analysis (RDA) and distance-based redundancy analysis (db-RDA). Constraining means that ordination only shows the community variation that can be explained by external environmental variables or constraints. This document describes the following basic concepts (1) usage of constrained ordination, (2) construction of model formulae, (3) updating of the model and choice of the constraining variables, (4) statistical significance of the overall model, single model terms or axes with permutation tests, and (5) partial ordination where the effects of some variables are removed before main ordination. The document also describes basic R concepts of inspecting data, and indexing, selecting and subsetting data.

Contents

1 Introduction

This tutorial describes the use of constrained ordination techniques in R packages **vegan**. This is a sequel to a similar tutorial describing unconstrained ordination.

The following constrained ordination methods are available in **vegan**:

1. `rda` for redundancy analysis (RDA), based on principal components analysis (PCA)
2. `cca` for constrained correspondence analysis (CCA), a.k.a. canonical correspondence analysis, and based on correspondence analysis
3. `capscale` for distance-based redundancy analysis (db-RDA), based on metric multidimensional scaling, a.k.a. principal coordinates analysis (PCoA).

These three functions work similarly, and have a similar user interface. You can freely select your favourite, although this tutorial focuses on CCA with some sidetracks to RDA.

The following data sets in **vegan** have both community data and environmental data, and can be used in constrained analysis, although only the Dutch dune meadow data and East Fennoscandian reindeer pastures are used in this tutorial:

- Reindeer pastures with 24 sites and 44 species (`varespec`), and environmental data with 14 continuous soil variables (`varechem`).
- Dutch dune meadows with 20 sites and 30 species (`dune`), and environmental data with four factors (some of these ordered) and one continuous variable (`dune.env`).
- Oribatid mites with 70 soil cores and 35 species sampled in a $2.5 \times 10\,\mathrm{m}$ plot (`mite`). The environmental data contains two continuous and three factor variables (`mite.env`). In addition, there are data sets of spatial coordinates of cores (`mite.xy`) and principal coordinates of neighbourhood matrix (PCNM) derived from these (`mite.pcnm`).
- Zooplankton in 12 mesocosms sampled 11 times after Pyrifos treatment. The community data has 132 observations and 178 species (`pyrifos`). The environmental data on experimental design can be generated using command `example(pyrifos)`.

2 Preliminaries: Inspecting Data

We study first the lichen pasture data with only continuous constraints. The data sets are taken into use with:[1]

[1] If you do not have the library, you must first install the required library from a repository using command `install.packages("vegan")`, or do the same from the menu entry.

```
R> library(vegan)
R> data(varespec)
R> data(varechem)
```

The environmental data can be inspected using commands `str` which shows the structure of any object in a compact form, and asking for a `summary` of a data frame:

```
R> str(varechem)
```

```
'data.frame':        24 obs. of  14 variables:
 $ N       : num  19.8 13.4 20.2 20.6 23.8 22.8 26.6 24.2 29.8 28.1 ...
 $ P       : num  42.1 39.1 67.7 60.8 54.5 40.9 36.7 31 73.5 40.5 ...
 $ K       : num  140 167 207 234 181 ...
 $ Ca      : num  519 357 973 834 777 ...
 $ Mg      : num  90 70.7 209.1 127.2 125.8 ...
 $ S       : num  32.3 35.2 58.1 40.7 39.5 40.8 33.8 27.1 42.5 60.2 ...
 $ Al      : num  39 88.1 138 15.4 24.2 ...
 $ Fe      : num  40.9 39 35.4 4.4 3 ...
 $ Mn      : num  58.1 52.4 32.1 132 50.1 ...
 $ Zn      : num  4.5 5.4 16.8 10.7 6.6 9.1 7.4 5.2 9.3 9.1 ...
 $ Mo      : num  0.3 0.3 0.8 0.2 0.3 0.4 0.3 0.3 0.3 0.5 ...
 $ Baresoil: num  43.9 23.6 21.2 18.7 46 40.5 23 29.8 17.6 29.9 ...
 $ Humdepth: num  2.2 2.2 2 2.9 3 3.8 2.8 2 3 2.2 ...
 $ pH      : num  2.7 2.8 3 2.8 2.7 2.7 2.8 2.8 2.8 2.8 ...
```

```
R> summary(varechem)
```

```
       N               P               K                Ca
 Min.   :13.40   Min.   :22.70   Min.   : 43.6   Min.   : 188.5
 1st Qu.:18.82   1st Qu.:32.60   1st Qu.:127.2   1st Qu.: 425.7
 Median :22.05   Median :41.50   Median :166.6   Median : 518.5
 Mean   :22.38   Mean   :45.08   Mean   :162.9   Mean   : 569.7
 3rd Qu.:26.30   3rd Qu.:57.05   3rd Qu.:205.8   3rd Qu.: 739.2
 Max.   :33.10   Max.   :73.50   Max.   :313.8   Max.   :1169.7
       Mg              S               Al              Fe
 Min.   : 25.70   Min.   :14.90   Min.   : 12.1   Min.   :  2.300
 1st Qu.: 60.90   1st Qu.:29.43   1st Qu.: 38.2   1st Qu.:  5.575
 Median : 75.05   Median :36.25   Median :106.8   Median : 27.800
 Mean   : 87.46   Mean   :37.19   Mean   :142.5   Mean   : 49.612
 3rd Qu.:108.70   3rd Qu.:43.60   3rd Qu.:233.7   3rd Qu.: 85.150
 Max.   :209.10   Max.   :60.20   Max.   :435.1   Max.   :204.400
       Mn              Zn              Mo            Baresoil
 Min.   : 10.10   Min.   : 2.600   Min.   :0.0500   Min.   : 0.010
 1st Qu.: 26.70   1st Qu.: 5.375   1st Qu.:0.2750   1st Qu.: 7.975
 Median : 36.55   Median : 8.100   Median :0.3000   Median :21.250
 Mean   : 49.33   Mean   : 7.596   Mean   :0.3958   Mean   :22.949
```

```
 3rd Qu.: 59.00   3rd Qu.: 9.100   3rd Qu.:0.5000   3rd Qu.:30.750
 Max.   :132.00   Max.   :16.800   Max.   :1.1000   Max.   :56.900
    Humdepth           pH
 Min.   :1.000   Min.   :2.700
 1st Qu.:1.800   1st Qu.:2.800
 Median :2.200   Median :2.900
 Mean   :2.200   Mean   :2.933
 3rd Qu.:2.625   3rd Qu.:3.000
 Max.   :3.800   Max.   :3.600
```

There is a difference between a data frame and a matrix in R: data frame is actually a list of variables which can be of different types (continuous, factors, ordered factors), whereas matrix only contains numbers of the same type.

The dependencies among variables can be inspected visually using `plot` command for data frames; under the hood this calls `pairs` function

```
R> plot(varechem, gap=0, panel=panel.smooth)
```

It is always useful to have a look at the data before rushing into analysis. It is necessary to have a look at the data if you read in your own data: you really must check that the data were imported correctly.

You can refer to rows and columns in the data frame in various ways. One way is to use numeric indices within square brackets "`[]`". The first item refers to a row, the second item to a column:

```
R> varechem[3,7]

[1] 138
```

Several items can be referred to by combining indices within `c()` or by using a regular sequence with "`:`" between first and last item:

```
R> varechem[2, c(3,1, 7)]

       K    N   Al
15 167.3 13.4 88.1

R> varechem[3:5, 7]

[1] 138.0  15.4  24.2
```

If you omit one index, whole row or whole column will be given to you:

```
R> varechem[2,]

      N    P     K    Ca   Mg    S   Al Fe   Mn  Zn  Mo Baresoil
15 13.4 39.1 167.3 356.7 70.7 35.2 88.1 39 52.4 5.4 0.3     23.6
   Humdepth  pH
15      2.2 2.8
```

```
R> varechem[,7]

 [1]  39.0  88.1 138.0  15.4  24.2 104.8  20.7  74.2  17.9 329.7  92.3
[12] 124.3  12.1 294.9  39.0 155.1 304.6 435.1 316.5 227.1 108.8 168.2
[23] 253.6  35.8
```

Finally, you can also refer to rows and columns by their names instead of numeric indices, and in data frames you can also use the dollar sign for variables:

```
R> varechem[, "pH"]

 [1] 2.7 2.8 3.0 2.8 2.7 2.7 2.8 2.8 2.8 2.8 2.7 2.9 2.9 3.1 3.1 3.0 3.3
[18] 2.9 2.9 3.0 2.9 3.2 3.6 3.0

R> varechem$pH

 [1] 2.7 2.8 3.0 2.8 2.7 2.7 2.8 2.8 2.8 2.8 2.7 2.9 2.9 3.1 3.1 3.0 3.3
[18] 2.9 2.9 3.0 2.9 3.2 3.6 3.0
```

3 Simple Case of Constrained Ordination

If you only have continuous variables, you can constrain the ordination by adding a second argument to the call of `cca` or `rda`:

```
R> m <- cca(varespec)
R> mm <- cca(varespec, varechem)
R> m

Call: cca(X = varespec)

              Inertia Rank
Total           2.083
Unconstrained   2.083   23
Inertia is mean squared contingency coefficient

Eigenvalues for unconstrained axes:
    CA1     CA2     CA3     CA4     CA5     CA6     CA7     CA8
0.52493 0.35680 0.23444 0.19546 0.17762 0.12156 0.11549 0.08894
(Showed only 8 of all 23 unconstrained eigenvalues)

R> mm

Call: cca(X = varespec, Y = varechem)

              Inertia Proportion Rank
Total          2.0832     1.0000
Constrained    1.4415     0.6920   14
Unconstrained  0.6417     0.3080    9
```

```
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
    CCA1     CCA2     CCA3     CCA4     CCA5     CCA6     CCA7     CCA8
0.438870 0.291775 0.162847 0.142130 0.117952 0.089029 0.070295 0.058359
    CCA9    CCA10    CCA11    CCA12    CCA13    CCA14
0.031141 0.013294 0.008364 0.006538 0.006156 0.004733

Eigenvalues for unconstrained axes:
     CA1      CA2      CA3      CA4      CA5      CA6      CA7      CA8
0.197765 0.141926 0.101174 0.070787 0.053303 0.033299 0.018868 0.015104
     CA9
0.009488
```

Compare the output of these two ordinations. In particular, see how the inertia and and rank (number of axes) are decomposed in constrained ordination.

The only way to select environmental variables from the full data set is to use a subset of the matrix:

```
R> cca(varespec, varechem[, c("Al", "P", "K")])

Call: cca(X = varespec, Y = varechem[, c("Al", "P", "K")])

              Inertia Proportion Rank
Total          2.0832     1.0000
Constrained    0.6441     0.3092    3
Unconstrained  1.4391     0.6908   20
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
  CCA1   CCA2   CCA3
0.3616 0.1700 0.1126

Eigenvalues for unconstrained axes:
    CA1     CA2     CA3     CA4     CA5     CA6     CA7     CA8
0.35004 0.22008 0.18507 0.15512 0.13511 0.10027 0.07730 0.05369
(Showed only 8 of all 20 unconstrained eigenvalues)
```

For a full numerical survey, you can use `summary`. The `summary` lists scores for every species and every site – and for sites two different kind of scores. The output is long, but if you only want to get an idea of the style of the results, you can delimit the output to the `head` (or `tail`) of each type of scores:

```
R> head(summary(mm))

Call:
cca(X = varespec, Y = varechem)
```

```
Partitioning of mean squared contingency coefficient:
              Inertia Proportion
Total          2.0832      1.000
Constrained    1.4415      0.692
Unconstrained  0.6417      0.308

Eigenvalues, and their contribution to the mean squared contingency coefficient

Importance of components:
                         CCA1   CCA2    CCA3    CCA4    CCA5    CCA6
Eigenvalue            0.4389 0.2918 0.16285 0.14213 0.11795 0.08903
Proportion Explained  0.2107 0.1401 0.07817 0.06823 0.05662 0.04274
Cumulative Proportion 0.2107 0.3507 0.42890 0.49713 0.55375 0.59649
                         CCA7    CCA8    CCA9   CCA10    CCA11    CCA12
Eigenvalue            0.07029 0.05836 0.03114 0.01329 0.008364 0.006538
Proportion Explained  0.03374 0.02801 0.01495 0.00638 0.004020 0.003140
Cumulative Proportion 0.63023 0.65825 0.67319 0.67958 0.683590 0.686730
                         CCA13    CCA14     CA1     CA2     CA3     CA4
Eigenvalue            0.006156 0.004733 0.19776 0.14193 0.10117 0.07079
Proportion Explained  0.002960 0.002270 0.09493 0.06813 0.04857 0.03398
Cumulative Proportion 0.689690 0.691960 0.78689 0.85502 0.90359 0.93757
                          CA5     CA6     CA7     CA8      CA9
Eigenvalue            0.05330 0.03330 0.01887 0.01510 0.009488
Proportion Explained  0.02559 0.01598 0.00906 0.00725 0.004550
Cumulative Proportion 0.96315 0.97914 0.98820 0.99545 1.000000

Accumulated constrained eigenvalues
Importance of components:
                         CCA1   CCA2   CCA3   CCA4    CCA5    CCA6
Eigenvalue            0.4389 0.2918 0.1628 0.1421 0.11795 0.08903
Proportion Explained  0.3045 0.2024 0.1130 0.0986 0.08183 0.06176
Cumulative Proportion 0.3045 0.5069 0.6198 0.7184 0.80027 0.86203
                         CCA7    CCA8    CCA9   CCA10    CCA11    CCA12
Eigenvalue            0.07029 0.05836 0.03114 0.01329 0.008364 0.006538
Proportion Explained  0.04877 0.04049 0.02160 0.00922 0.005800 0.004540
Cumulative Proportion 0.91080 0.95128 0.97288 0.98211 0.987910 0.992450
                         CCA13    CCA14
Eigenvalue            0.006156 0.004733
Proportion Explained  0.004270 0.003280
Cumulative Proportion 0.996720 1.000000

Scaling 2 for species and site scores
* Species are scaled proportional to eigenvalues
* Sites are unscaled: weighted dispersion equal on all dimensions
```

Species scores

```
            CCA1     CCA2     CCA3     CCA4     CCA5      CCA6
Cal.vul  0.07535 -0.93581  1.67774  0.69551  1.07752 -0.345001
Emp.nig -0.18134  0.07610  0.03646 -0.42773 -0.13815  0.010517
Led.pal -1.05355 -0.06026  0.07743 -0.93890 -0.21394 -0.518031
Vac.myr -1.27743  0.30759  0.30370 -0.09209 -0.56882 -0.613023
Vac.vit -0.15256  0.12054 -0.05303 -0.36228  0.08394  0.008938
Pin.syl  0.24296  0.26432  0.22326 -0.27381  0.29210 -0.063335
....
```

Site scores (weighted averages of species scores)

```
        CCA1    CCA2     CCA3     CCA4    CCA5     CCA6
18    0.1785 -1.0599 -0.40884 -0.60721 -0.5649  0.24175
15   -0.9702 -0.1971  0.42105  0.30324  0.1517  0.80394
24   -1.2798  0.4764 -2.94686  0.39292  3.9543  0.76592
27   -1.5009  0.6522  0.08584  0.76207 -1.2325 -0.09756
23   -0.5981 -0.1840 -0.13561 -1.16425 -0.3025  0.07033
19   -0.1103  0.7143  0.01659 -0.07773 -0.5521 -0.08258
....
```

Site constraints (linear combinations of constraining variables)

```
        CCA1    CCA2     CCA3    CCA4     CCA5    CCA6
18   -0.4231 -1.3247 -0.49215 -0.9449 -0.04846  0.9398
15   -0.1903  0.4969  0.45454 -0.5295 -0.07660 -0.7899
24   -0.8633  0.2521 -2.76035  0.5699  3.29271  0.2629
27   -1.6981  0.4867 -0.56351  1.0736 -0.61415  0.4988
23   -0.7956  0.1072  0.25751 -0.9042 -0.28756  0.4387
19   -0.6770  1.0013  0.03344 -1.0035 -0.14128 -0.9383
....
```

Biplot scores for constraining variables

```
            CCA1     CCA2      CCA3     CCA4      CCA5      CCA6
N        -0.22263 -0.52873  0.006846  0.17783 -0.253589  0.102585
P        -0.31879  0.57898 -0.162025  0.47953  0.184179 -0.121978
K        -0.36622  0.30801  0.359825  0.47953  0.325514 -0.196760
Ca       -0.44781  0.42181 -0.037791  0.09821  0.308082  0.043457
Mg       -0.43504  0.34068 -0.142159  0.10798  0.497884 -0.005699
S        -0.02402  0.41589  0.148402  0.44463  0.597121 -0.166312
```

```
Al         0.76978 -0.04774  0.037545  0.39088  0.161110 -0.337018
Fe         0.64898 -0.08855 -0.042180  0.26270 -0.069551 -0.111877
Mn        -0.72248  0.22467  0.113055  0.29160 -0.138701  0.180553
Zn        -0.35807  0.33518 -0.277890  0.34595  0.619201 -0.001033
Mo         0.20471 -0.10281 -0.156893  0.32497  0.516250 -0.313054
Baresoil -0.53655 -0.25382  0.137513 -0.52018  0.165924 -0.351435
Humdepth -0.69660  0.20229  0.271840 -0.13530 -0.003628 -0.050743
pH         0.49689  0.07444 -0.326659  0.02028 -0.145165 -0.059964
```

The `summary` also shows how the total inertia is divided between individual axes, and the lines for accounted inertia shows the accumulated variation explained by all axes, as well as the accounted constrained inertia. Note also that scaling of the site and species scores happens only when you look at them in `summary`, and you can give `scaling` as an argument to the summary. Note also that there are two different kind of site scores in the output. The lectures explain their difference.

The two kind of site scores are often called WA scores and LC scores. The WA scores are derived from species scores, and the LC scores are derived from constraints as their linear combinations. The analysis tries to keep these two sets as similar as possible. Their similarity can be inspected with species–environment correlation which simply is the correlation between LC and WA score for an axis.

```
R> spenvcor(mm)
```

```
     CCA1      CCA2      CCA3      CCA4      CCA5      CCA6      CCA7
0.9266005 0.9288995 0.9293730 0.9274700 0.8148392 0.7753491 0.9004049
     CCA8      CCA9     CCA10     CCA11     CCA12     CCA13     CCA14
0.7939871 0.6386331 0.7550495 0.4014950 0.5700990 0.5923515 0.5878917
```

The default plot displays species scores, WA scores and environmental variables.

```
R> plot(mm)
```

You can change this by giving the displayed elements in argument `display`. The following shows only LC scores and environmental variables:

```
R> plot(mm, display = c("lc","bp"))
```

The items are called `"sp", "wa", "lc", "bp", "cn"`, which are self explanatory, except the last which refers to the centroids of the environmental variables. You can visually inspect the difference between WA and LC scores (or the species – environment relationship) with command `ordispider` which (among other alternatives) joins WA and LC scores by a line:

```
R> plot(mm, dis=c("wa","lc"))
R> ordispider(mm)
```

It is often a bad idea to use constrained ordination if you have a very large number of constraints: probably you will not constrain at all, but your analysis may be very similar to ordinary unconstrained analysis that could have been used just as well. You can inspect this with Procrustes rotation:

```
R> plot(procrustes(m, mm))
```

You can also see how similar the ordinations are by fitting environmental variables to unconstrained ordination.

```
R> plot(m)
R> plot(envfit(m, varechem))
```

4 Formula Interface

You can use formula interface to select the variables used as constraints:

```
R> cca(varespec ~  Al + P + K, data=varechem)

Call: cca(formula = varespec ~ Al + P + K, data = varechem)

              Inertia Proportion Rank
Total          2.0832     1.0000
Constrained    0.6441     0.3092    3
Unconstrained  1.4391     0.6908   20
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
  CCA1   CCA2   CCA3
0.3616 0.1700 0.1126

Eigenvalues for unconstrained axes:
    CA1     CA2     CA3     CA4     CA5     CA6     CA7     CA8
0.35004 0.22008 0.18507 0.15512 0.13511 0.10027 0.07730 0.05369
(Showed only 8 of all 20 unconstrained eigenvalues)
```

This indeed is the recommended method of defining the model, because it allows full control of the model and some further analyses are possible only when the model was defined through a formula.

With formula interface you can also use factor constraints:

```
R> data(dune)
R> data(dune.env)
R> str(dune.env)

'data.frame':        20 obs. of  5 variables:
 $ A1        : num  3.5 6 4.2 5.7 4.3 2.8 4.2 6.3 4 11.5 ...
 $ Moisture  : Ord.factor w/ 4 levels "1"<"2"<"4"<"5": 1 4 2 4 1 1 4 1 2 4 ...
```

```
 $ Management: Factor w/ 4 levels "BF","HF","NM",..: 1 4 4 4 2 4 2 2 3 3 ...
 $ Use       : Ord.factor w/ 3 levels "Hayfield"<"Haypastu"<..: 2 2 2 3 2 2 3 1 1 2 ...
 $ Manure    : Ord.factor w/ 5 levels "0"<"1"<"2"<"3"<..: 3 4 5 4 3 5 4 3 1 1 ...
```

```
R> 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
```

```
R> mdun <- cca(dune ~  Management + A1, dune.env)
R> mdun
```

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

              Inertia Proportion Rank
Total          2.1153     1.0000
Constrained    0.7798     0.3686    4
Unconstrained  1.3355     0.6314   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
```

```
R> plot(mdun)
```

Note here how the use of factors increases the rank of the constrained solution. If you have p levels of a factor, you will get $p - 1$ axes – provided your factor levels really are independent.

4.1 Model Choice

With formula we have a full control of the model, but we face with the problem of model choice. Models must be built carefully, and preferably used to test specific hypotheses. Sometimes we may want to use automatic model building, but this must be done carefully. There are some shortcuts and tricks for this in **vegan**, but these should be used with utmost care.

In automatic model building we usually need two extreme models: the smallest and the largest model considered. The following shortcuts build a model with all environmental variables and a model with no environmental variables, but both with a formula so that terms can be added or removed from the model:

```
R> m1 <- cca(varespec ~ ., varechem)
R> m0 <- cca(varespec ~ 1, varechem)
R> m1

Call: cca(formula = varespec ~ N + P + K + Ca + Mg + S + Al +
Fe + Mn + Zn + Mo + Baresoil + Humdepth + pH, data = varechem)

              Inertia Proportion Rank
Total          2.0832     1.0000
Constrained    1.4415     0.6920   14
Unconstrained  0.6417     0.3080    9
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
    CCA1     CCA2     CCA3     CCA4     CCA5     CCA6     CCA7     CCA8
0.438870 0.291775 0.162847 0.142130 0.117952 0.089029 0.070295 0.058359
    CCA9    CCA10    CCA11    CCA12    CCA13    CCA14
0.031141 0.013294 0.008364 0.006538 0.006156 0.004733

Eigenvalues for unconstrained axes:
     CA1      CA2      CA3      CA4      CA5      CA6      CA7      CA8
0.197765 0.141926 0.101174 0.070787 0.053303 0.033299 0.018868 0.015104
     CA9
0.009488

R> m0

Call: cca(formula = varespec ~ 1, data = varechem)

              Inertia Rank
Total           2.083
Unconstrained   2.083   23
Inertia is mean squared contingency coefficient

Eigenvalues for unconstrained axes:
    CA1     CA2     CA3     CA4     CA5     CA6     CA7     CA8
0.52493 0.35680 0.23444 0.19546 0.17762 0.12156 0.11549 0.08894
(Showed only 8 of all 23 unconstrained eigenvalues)
```

Then we can use `step` function to select the "best" model. The `step` function uses Akaike's Information Criterion (AIC) in model choice. The AIC is based on the goodness of fit (high constrained inertia), but it is penalized by the number

of estimated parameters (constrained rank). The alternative models are ordered by AIC. In each case, "+" indicated the effect of adding a term, and "-" the effect of removing a term, while the current model is marked as "<none>". The model building proceeds by steps until the current model ("<none>") is the best.

Special care is needed, because there really is no AIC for constrained ordination (although it is calculated!), and we always should inspect the validity of model choice. One way is to ask for approximate significance tests. If the model choice was valid, all included variables ("-" before their name) should be significant, and all excluded variables ("+" before their name) should be insignificant.[2]

```
R> m <- step(m0, scope=formula(m1), test="perm")

Start:  AIC=130.31
varespec ~ 1

           Df    AIC      F N.Perm Pr(>F)
+ Al        1 128.61 3.6749    199  0.005 **
+ Mn        1 128.95 3.3115    199  0.010 **
+ Humdepth  1 129.24 3.0072    199  0.005 **
+ Baresoil  1 129.77 2.4574    199  0.015 *
+ Fe        1 129.79 2.4360    199  0.010 **
+ P         1 130.03 2.1926    199  0.020 *
+ Zn        1 130.30 1.9278    199  0.040 *
<none>        130.31
+ Mg        1 130.35 1.8749    199  0.055 .
+ K         1 130.37 1.8609    199  0.065 .
+ Ca        1 130.43 1.7959    199  0.080 .
+ pH        1 130.57 1.6560     99  0.130
+ S         1 130.72 1.5114     99  0.190
+ N         1 130.77 1.4644     99  0.160
+ Mo        1 131.19 1.0561     99  0.510
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Step:  AIC=128.61
varespec ~ Al

           Df    AIC      F N.Perm Pr(>F)
+ P         1 127.91 2.5001    199  0.010 **
+ K         1 128.09 2.3240    199  0.010 **
+ S         1 128.26 2.1596    199  0.015 *
+ Zn        1 128.44 1.9851    199  0.050 *
+ Mn        1 128.53 1.8945    199  0.055 .
```

[2] In Windows you should set off "buffered output" from the menu so that you can see the results as soon as they are calculated; with buffered output the all results are printed only after the whole analysis ends.

```
<none>        128.61
+ Mg         1 128.70 1.7379    199  0.075 .
+ N          1 128.85 1.5900    199  0.075 .
+ Baresoil   1 128.88 1.5670    199  0.075 .
+ Ca         1 129.04 1.4180     99  0.160
+ Humdepth   1 129.08 1.3814     99  0.150
+ Mo         1 129.50 0.9884     99  0.530
+ pH         1 129.63 0.8753     99  0.560
+ Fe         1 130.02 0.5222     99  0.930
- Al         1 130.31 3.6749    199  0.005 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step:  AIC=127.91
varespec ~ Al + P

           Df    AIC      F N.Perm Pr(>F)
+ K         1 127.44 2.1688    199  0.035 *
<none>        127.91
+ Baresoil  1 127.99 1.6606     99  0.130
+ N         1 128.11 1.5543    199  0.090 .
+ S         1 128.36 1.3351     99  0.180
+ Mn        1 128.44 1.2641     99  0.250
+ Zn        1 128.51 1.2002     99  0.270
+ Humdepth  1 128.56 1.1536     99  0.300
- P         1 128.61 2.5001    199  0.010 **
+ Mo        1 128.75 0.9837     99  0.480
+ Mg        1 128.79 0.9555     99  0.530
+ pH        1 128.82 0.9247     99  0.460
+ Fe        1 129.28 0.5253     99  0.870
+ Ca        1 129.36 0.4648     99  0.930
- Al        1 130.03 3.9401    199  0.010 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step:  AIC=127.44
varespec ~ Al + P + K

           Df    AIC      F N.Perm Pr(>F)
<none>        127.44
+ N         1 127.59 1.5148    199  0.110
+ Baresoil  1 127.67 1.4544     99  0.160
+ Zn        1 127.84 1.3067     99  0.190
+ S         1 127.89 1.2604     99  0.190
- K         1 127.91 2.1688    199  0.035 *
+ Mo        1 127.92 1.2350     99  0.220
```

```
- P         1 128.09 2.3362    199  0.020 *
+ Mg        1 128.17 1.0300     99  0.350
+ Mn        1 128.34 0.8879     99  0.570
+ Humdepth  1 128.44 0.8056     99  0.720
+ Fe        1 128.79 0.5215     99  0.890
+ pH        1 128.81 0.5067     99  0.910
+ Ca        1 128.89 0.4358     99  0.920
- Al        1 130.14 4.3340    199  0.005 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

With continuous variables, the model building often works rather well, but it should not be trusted blindly (if at all). You can see this if you use `rda` instead of `cca`, or `dune` and `dune.env` data sets instead of lichen pastures (you may try). Moreover, you may end up with different models if you change the modelling strategy. The following simplifies the maximum model:

```
R> mback <- step(m1, test="perm")
```

```
Start:  AIC=130.05
varespec ~ N + P + K + Ca + Mg + S + Al + Fe + Mn + Zn + Mo +
    Baresoil + Humdepth + pH

           Df    AIC      F N.Perm Pr(>F)
- Fe        1 129.77 0.6683     99  0.700
<none>        130.05
- Ca        1 130.26 0.8652     99  0.500
- N         1 130.30 0.8838     99  0.590
- Zn        1 130.35 0.9043     99  0.470
- pH        1 130.47 0.9529     99  0.390
- Al        1 130.50 0.9644     99  0.550
- Mo        1 130.79 1.0885     99  0.420
- Mn        1 130.91 1.1375     99  0.300
- Mg        1 131.15 1.2384     99  0.300
- Baresoil  1 131.19 1.2574     99  0.250
- P         1 131.63 1.4484     99  0.160
- K         1 131.65 1.4565     99  0.150
- S         1 131.66 1.4569     99  0.170
- Humdepth  1 132.65 1.9001    199  0.095 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step:  AIC=129.77
varespec ~ N + P + K + Ca + Mg + S + Al + Mn + Zn + Mo + Baresoil +
    Humdepth + pH

           Df    AIC      F N.Perm Pr(>F)
```

```
- Al         1 129.25 0.6358     99   0.76
- pH         1 129.37 0.6885     99   0.70
- N          1 129.54 0.7630     99   0.71
- Zn         1 129.58 0.7801     99   0.66
<none>         129.77
- Ca         1 129.86 0.9074     99   0.48
- Mn         1 130.02 0.9811     99   0.44
- Mg         1 130.62 1.2589     99   0.30
- Mo         1 130.70 1.2965     99   0.27
- Baresoil   1 130.84 1.3639     99   0.23
- S          1 130.98 1.4283     99   0.18
- K          1 131.32 1.5912     99   0.16
- P          1 131.33 1.5956    199   0.12
- Humdepth   1 131.58 1.7177    199   0.12

Step:  AIC=129.25
varespec ~ N + P + K + Ca + Mg + S + Mn + Zn + Mo + Baresoil +
    Humdepth + pH

           Df    AIC      F N.Perm Pr(>F)
- N          1 128.79 0.7294     99  0.680
<none>         129.25
- Ca         1 129.26 0.9585     99  0.520
- Mn         1 129.34 1.0018     99  0.450
- Zn         1 129.46 1.0612     99  0.470
- pH         1 129.47 1.0637     99  0.430
- Mg         1 129.97 1.3178     99  0.300
- Baresoil   1 130.14 1.4053     99  0.250
- K          1 130.48 1.5850     99  0.210
- P          1 130.56 1.6249    199  0.085 .
- Mo         1 130.69 1.6954     99  0.150
- Humdepth   1 130.94 1.8241    199  0.100 .
- S          1 131.27 2.0063    199  0.060 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step:  AIC=128.79
varespec ~ P + K + Ca + Mg + S + Mn + Zn + Mo + Baresoil + Humdepth +
    pH

           Df    AIC      F N.Perm Pr(>F)
- pH         1 128.51 0.8916     99  0.600
- Ca         1 128.78 1.0342     99  0.460
<none>         128.79
- Zn         1 128.83 1.0606     99  0.350
- Mn         1 128.90 1.0998     99  0.380
```

```
- Baresoil  1 129.48 1.4210     99  0.280
- Mg        1 129.69 1.5385     99  0.160
- P         1 129.90 1.6599    199  0.065 .
- Humdepth  1 130.05 1.7448    199  0.130
- Mo        1 130.24 1.8537    199  0.100 .
- K         1 130.69 2.1165    199  0.035 *
- S         1 131.64 2.6833    199  0.015 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step:  AIC=128.51
varespec ~ P + K + Ca + Mg + S + Mn + Zn + Mo + Baresoil + Humdepth

           Df    AIC      F N.Perm Pr(>F)
- Ca        1 128.13 0.9042     99  0.460
- Zn        1 128.45 1.0937     99  0.270
- Mn        1 128.46 1.0993     99  0.340
<none>        128.51
- Mg        1 129.16 1.5142    199  0.120
- Baresoil  1 129.33 1.6169     99  0.130
- P         1 129.44 1.6850    199  0.125
- Mo        1 129.47 1.7030    199  0.120
- Humdepth  1 129.72 1.8581    199  0.075 .
- K         1 130.27 2.2028    199  0.040 *
- S         1 131.01 2.6799    199  0.010 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step:  AIC=128.13
varespec ~ P + K + Mg + S + Mn + Zn + Mo + Baresoil + Humdepth

           Df    AIC      F N.Perm Pr(>F)
- Zn        1 127.89 1.0653     99  0.320
- Mg        1 128.06 1.1760     99  0.270
- Mn        1 128.10 1.1995     99  0.210
<none>        128.13
- Baresoil  1 128.74 1.6108     99  0.150
- P         1 129.03 1.8030    199  0.040 *
- Humdepth  1 129.09 1.8393    199  0.075 .
- Mo        1 129.38 2.0341    199  0.035 *
- K         1 129.41 2.0551    199  0.060 .
- S         1 129.71 2.2571    199  0.035 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step:  AIC=127.89
```

```
varespec ~ P + K + Mg + S + Mn + Mo + Baresoil + Humdepth

           Df    AIC      F N.Perm Pr(>F)
<none>        127.89
- Mn        1 128.10 1.4525    199  0.150
- Baresoil  1 128.31 1.5953    199  0.090 .
- P         1 128.65 1.8291    199  0.055 .
- Mg        1 128.78 1.9210    199  0.075 .
- Humdepth  1 128.85 1.9700    199  0.035 *
- K         1 128.91 2.0155    199  0.035 *
- S         1 129.07 2.1254    199  0.055 .
- Mo        1 129.32 2.3076    199  0.020 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Compare the order in which variables were added in the first model, and removed in this model.

The constrained ordination methods really do not have AIC, although it is calculated in `step`. Function `ordistep` uses significance test based on permutation instead of the non-existing AIC (see the next chapter).[3] The results of random permutations really are random, and therefore the model choice can have a random component. The usage is almost identical to the AIC based `step`:

```
R> m <- ordistep(m0, scope = formula(m1))

Start: varespec ~ 1

           Df    AIC      F N.Perm  Pr(>F)
+ Al        1 128.61 3.6749    199 0.00500 **
+ Mn        1 128.95 3.3115    199 0.00500 **
+ Humdepth  1 129.24 3.0072    199 0.00500 **
+ Baresoil  1 129.77 2.4574    199 0.00500 **
+ Fe        1 129.79 2.4360    399 0.01750 *
+ P         1 130.03 2.1926    599 0.02667 *
+ Zn        1 130.30 1.9278    999 0.03800 *
+ Mg        1 130.35 1.8749    999 0.06900 .
+ Ca        1 130.43 1.7959    999 0.07000 .
+ K         1 130.37 1.8609    499 0.08600 .
+ pH        1 130.57 1.6560    199 0.10500
+ N         1 130.77 1.4644     99 0.15000
+ S         1 130.72 1.5114     99 0.17000
+ Mo        1 131.19 1.0561     99 0.45000
---
```

[3] `ordistep` is available from **vegan** version 1.17-0. If you have older **vegan** you can skip the following command (or install new **vegan**).

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step: varespec ~ Al

     Df    AIC      F N.Perm Pr(>F)
- Al  1 130.31 3.6749     99   0.01 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

           Df    AIC      F N.Perm  Pr(>F)
+ P         1 127.91 2.5001    199 0.00500 **
+ S         1 128.26 2.1596    299 0.01667 *
+ K         1 128.09 2.3240    299 0.02000 *
+ Mn        1 128.53 1.8945    399 0.02250 *
+ Zn        1 128.44 1.9851    999 0.03300 *
+ N         1 128.85 1.5900    999 0.06300 .
+ Baresoil  1 128.88 1.5670    599 0.08000 .
+ Humdepth  1 129.08 1.3814     99 0.13000
+ Mg        1 128.70 1.7379     99 0.16000
+ Ca        1 129.04 1.4180     99 0.18000
+ Mo        1 129.50 0.9884     99 0.47000
+ pH        1 129.63 0.8753     99 0.56000
+ Fe        1 130.02 0.5222     99 0.93000
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step: varespec ~ Al + P

     Df    AIC      F N.Perm Pr(>F)
- P   1 128.61 2.5001     99   0.01 **
- Al  1 130.03 3.9401     99   0.01 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

           Df    AIC      F N.Perm  Pr(>F)
+ K         1 127.44 2.1688    399 0.02000 *
+ Baresoil  1 127.99 1.6606    299 0.09333 .
+ N         1 128.11 1.5543     99 0.13000
+ S         1 128.36 1.3351     99 0.21000
+ Mn        1 128.44 1.2641     99 0.25000
+ Zn        1 128.51 1.2002     99 0.25000
+ Humdepth  1 128.56 1.1536     99 0.36000
+ Mg        1 128.79 0.9555     99 0.41000
+ Mo        1 128.75 0.9837     99 0.50000
+ pH        1 128.82 0.9247     99 0.51000
+ Fe        1 129.28 0.5253     99 0.92000
```

```
+ Ca        1 129.36 0.4648     99 0.93000
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step: varespec ~ Al + P + K

     Df    AIC      F N.Perm Pr(>F)
- K   1 127.91 2.1688    199  0.045 *
- P   1 128.09 2.3362     99  0.030 *
- Al  1 130.14 4.3340     99  0.020 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

           Df    AIC      F N.Perm Pr(>F)
+ N         1 127.59 1.5148     99   0.13
+ Baresoil  1 127.67 1.4544     99   0.15
+ Mo        1 127.92 1.2350     99   0.23
+ Zn        1 127.84 1.3067     99   0.24
+ S         1 127.89 1.2604     99   0.26
+ Mg        1 128.17 1.0300     99   0.46
+ Mn        1 128.34 0.8879     99   0.48
+ Humdepth  1 128.44 0.8056     99   0.60
+ Ca        1 128.89 0.4358     99   0.88
+ Fe        1 128.79 0.5215     99   0.90
+ pH        1 128.81 0.5067     99   0.93
```

```
R> m$anova
```

```
     Df    AIC      F N.Perm Pr(>F)
+ Al  1 128.61 3.6749    199  0.005 **
+ P   1 127.91 2.5001    199  0.005 **
+ K   1 127.44 2.1688    399  0.020 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

One problem with model building is that constraining variables are not independent, but they are correlated. Any correlated variable can be explained with other variables. Such variables are redundant (“expendable”) when they are with other variables, but they may be the best variables alone and prevent other variables from entering the model. A statistic describing this is called variance inflation factor (VIF) which is 1 for completely independent variables, and values above 10 or 20 (depending on your taste) are regarded as highly multicollinear (dependent on others). The VIF of a variable will depend on the set it sits with:

```
R> vif.cca(m1)
```

```
       N         P         K        Ca        Mg         S        Al
1.981742  6.028515 12.009357  9.925801  9.810609 18.378794 21.192739
      Fe        Mn        Zn        Mo  Baresoil  Humdepth        pH
9.127762  5.380432  7.739664  4.320346  2.253683  6.012537  7.389267
```

```
R> vif.cca(m)
```

```
      Al        P        K
1.011808 2.365413 2.378806
```

```
R> vif.cca(mback)
```

```
       P        K       Mg        S       Mn       Mo Baresoil Humdepth
3.929461 7.627516 2.860558 9.250139 3.467043 2.447455 1.681052 2.841139
```

4.2 Updating a Model

You can manually build models if the automatic procedure fails. For instance, it does not work for Dutch dunes, where the building stops too early:

```
R> m0 <- cca(dune ~  1, dune.env)
R> m1 <- cca(dune ~ ., dune.env)
R> m <- step(m0, scope=formula(m1), test="p")
```

```
Start:  AIC=87.66
dune ~ 1

             Df    AIC      F N.Perm Pr(>F)
+ Moisture    3 86.608 2.2536    199  0.005 **
+ Management  3 86.935 2.1307    199  0.005 **
+ A1          1 87.411 2.1400    199  0.015 *
<none>          87.657
+ Manure      4 88.832 1.5251    199  0.035 *
+ Use         2 89.134 1.1431     99  0.330
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step:  AIC=86.61
dune ~ Moisture

             Df    AIC      F N.Perm Pr(>F)
<none>          86.608
+ Management  3 86.813 1.4565    199  0.035 *
+ A1          1 86.992 1.2624     99  0.160
+ Use         2 87.259 1.2760    199  0.095 .
+ Manure      4 87.342 1.3143    199  0.090 .
- Moisture    3 87.657 2.2536    199  0.005 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
R> m

Call: cca(formula = dune ~ Moisture, data = dune.env)

              Inertia Proportion Rank
Total          2.1153     1.0000
Constrained    0.6283     0.2970    3
Unconstrained  1.4870     0.7030   16
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
   CCA1    CCA2    CCA3
0.41868 0.13304 0.07659

Eigenvalues for unconstrained axes:
     CA1      CA2      CA3      CA4      CA5      CA6      CA7      CA8
0.409782 0.225913 0.176062 0.123389 0.108171 0.090751 0.085878 0.060894
     CA9     CA10     CA11     CA12     CA13     CA14     CA15     CA16
0.056606 0.046688 0.041926 0.020103 0.014335 0.009917 0.008505 0.008033
```

You can use `update` command where you change the formula. The retained parts of the formula are shown by a dot (`.`) and terms are added with `+` or removed with `-`. The following keeps the dependent variable (left hand side) and all previously entered constraints (hence "`. ~ .`"), and adds `Management` to the previous model:

```
R> m <- update(m, . ~ . + Management)
```

You can see if any other terms should be added to this model, or if removing an included term could improve the model – if constraints are correlated, the significance can change with adding or removing variables from the model:

```
R> add1(m, scope=formula(m1), test="perm")

       Df    AIC      F N.Perm Pr(>F)
<none>    86.813
A1      1 86.190 1.6817    199   0.11
Use     2 88.245 0.7534     99   0.69
Manure  3 88.430 0.8167     99   0.60

R> drop1(m, test="perm")

           Df    AIC      F N.Perm Pr(>F)
<none>        86.813
Moisture    3 86.935 1.5518    199   0.03 *
Management  3 86.608 1.4565    199   0.03 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If needed, you can manually continue model building.

5 Significance tests

We already saw significance tests with model building. These tests were based on permutation: there is no known distribution of inertia that could be used for strict statistical testing. We simply permute the rows of community data, repeat the analysis, and get a random result. If our observed result is better than most of the random models (say, better than 95 % of them), we say that our results are significant.

Package **vegan** has several alternative types of significance tests. They all can be performed with a function called `anova`. The name is somewhat misleading: the test are based on permutations although the layout of the results is similar as in the standard ANOVA table. The default is an overall test of all variables together:

```
R> anova(m)

Permutation test for cca under reduced model

Model: cca(formula = dune ~ Moisture + Management, data = dune.env)
         Df  Chisq      F N.Perm Pr(>F)
Model     6 1.0024 1.9515    199  0.005 **
Residual 13 1.1129
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

We actually used function `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. This kind of functions that are automatically adapted to the result type are called *method functions*. We have already seen many of them: for instance `plot` and `summary` work differently for different types of results.

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 ($P = 0.05$ 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:

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

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

Model: cca(formula = dune ~ Moisture + Management, data = dune.env)
           Df  Chisq      F N.Perm Pr(>F)
Moisture    3 0.6283 2.4465    199  0.005 **
Management  3 0.3741 1.4565    199  0.030 *
Residual   13 1.1129
```

```
---
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”):

```
R> anova(m, by="mar")

Permutation test for cca under reduced model
Marginal effects of terms

Model: cca(formula = dune ~ Moisture + Management, data = dune.env)
           Df  Chisq      F N.Perm  Pr(>F)
Moisture    3 0.3985 1.5518    199 0.01500 *
Management  3 0.3741 1.4565    299 0.01333 *
Residual   13 1.1129
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

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

```
R> anova(m, by="axis", perm=500)

Model: cca(formula = dune ~ Moisture + Management, data = dune.env)
         Df  Chisq      F N.Perm Pr(>F)
CCA1      1 0.4458 5.2079    199  0.005 **
CCA2      1 0.2887 3.3723    199  0.005 **
CCA3      1 0.1124 1.3128     99  0.180
CCA4      1 0.0717 0.8370     99  0.430
CCA5      1 0.0494 0.5767     99  0.860
CCA6      1 0.0344 0.4023     99  0.950
Residual 13 1.1129
---
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).

6 Conditioned or partial ordination

All constrained ordination methods can have terms that are partialled out from the analysis before constraints:

```
R> m <- cca(dune ~ Management + Condition(Moisture + A1), data=dune.env)
R> m

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

              Inertia Proportion Rank
Total          2.1153     1.0000
Conditional    0.7437     0.3516    4
Constrained    0.3954     0.1869    3
Unconstrained  0.9761     0.4615   12
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
   CCA1    CCA2    CCA3
0.23854 0.08653 0.07036

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` and `A1` before analysing the effects of `Management`. This also influences the significances of the terms:

```
R> anova(m, by="term", perm=500)

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

Model: cca(formula = dune ~ Management + Condition(Moisture + A1), data = dune.env)
           Df  Chisq      F N.Perm Pr(>F)
Management  3 0.3954 1.6205     99   0.03 *
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`:

```
R> with(dune.env, anova(m, 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 ~ Management + Condition(Moisture + A1), data = dune.env)
```

```
           Df  Chisq      F N.Perm Pr(>F)
Management  3 0.3954 1.6205     99   0.02 *
Residual   12 0.9761
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Here `with()` is a special function that makes variables in `dune.env` visible to the following command. If you only type `Moisture` in an R prompt, you will get an error of missing variables. Functions with formulae have a `data` argument giving the name of the data frame from which the variables are found, but other functions usually do not have such an argument. Instead of `with(dune.env, command())`, you can first `attach(dune.env)` and after that all variables in the data frame are visible in the session. This may be dangerous if you have similar names in your session and several `attach`ed data frames: it is difficult to know which of these was used.

6.1 Partition of Variation

The basic partial ordination gives decomposition of the inertia into conditional (partialled out) and (residual) constrained component. The decomposition is sequential: conditions are partialled out before analysing constraints. Reversing the order of terms changes the components. Often we want to have a neutral decomposition of variation into unique and shared components of several sources. For this we can use function `varpart` of **vegan** which can handle upto four sources of variation.

Function `varpart` partitions adjusted R^2. Unlike ordinary R^2 or variance, adjusted R^2 is unbiased and its expected value is $R^2 = 0$ for random data (more in lectures). However, it can be easily calculated only for RDA and db-RDA. In principle, CCA can also be used, but calculations are lengthy and complicated, and no software is implemented for them.

The example above can be analysed using command:

```
R> mod <- varpart(dune, ~ Management, ~ A1 + Moisture, data = dune.env)
R> mod

Partition of variation in RDA

Call: varpart(Y = dune, X = ~Management, ~A1 + Moisture, data =
dune.env)

Explanatory tables:
X1:  ~Management
X2:  ~A1 + Moisture

No. of explanatory tables: 2
Total variation (SS): 1598.4
            Variance: 84.124
```

```
No. of observations: 20

Partition table:
                     Df R.squared Adj.R.squared Testable
[a+b] = X1            3   0.34747       0.22512     TRUE
[b+c] = X2            4   0.35382       0.18150     TRUE
[a+b+c] = X1+X2       7   0.58105       0.33666     TRUE
Individual fractions
[a] = X1|X2           3                 0.15515     TRUE
[b]                   0                 0.06997    FALSE
[c] = X2|X1           4                 0.11153     TRUE
[d] = Residuals                         0.66334    FALSE
---
Use function 'rda' to test significance of fractions of interest
```

The first argument (`dune`) gives the dependent data, and the next two to four arguments give the sources of variation. These can be single variables, matrices or one-sided model formulae like above.

The output can be tricky to read: names `X1`, `X2` etc refer to the original sources. These are divided into unique and shared components. In our example, the source `X1` (`Management`) has two components: unique component `a` and a component `b` that is shared with the source `X2`. Source `X2` consists of its unique component `c` and the same shared component `b`. Consquently, `X1` = `a` + `b` and `X2` = `b` + `c`. The unique component is the variation that can be explained only by the source and not by other source, and it is shown with operator `|` in the output: `a` = `X1 | X2`. The lower case letters can be deciphered using:

```
R> showvarparts(2)
R> showvarparts(3)
```

and the decomposition can be displayed in the same diagrams with

```
R> plot(mod)
```

The decomposition can be done by hand:

```
R> ab <- rda(dune ~ Management, dune.env)
R> bc <- rda(dune ~ A1 + Moisture, dune.env)
R> abc <- rda(dune ~ Management + A1 + Moisture, dune.env)
R> a <- rda(dune ~ Management + Condition(A1 + Moisture), dune.env)
R> c <- rda(dune ~ A1 + Moisture + Condition(Management), dune.env)
```

However, the shared component `b` cannot be found with a direct model. All the models above define testable models, but component `c` is non-testable and can be only found indirectly as a difference of the models. The components of variation can be negative for various reasons. Simplest reasons is that they are negative because of random variation in adjusted R^2, but there can be other, more complicated reasons (like multivariate non-linear dependence between sources of variation).