Table of Contents

Section: Ordination analysis

Variable selection (constrained ordination)

Examples

Example 1: tb-RDA with forward selection: Which chemical variables measured in fen water are important for vegetation of Carpathian wetlands?

We will use data about species composition of vascular plants in Carpathian wetlands, and ask which chemical variables, measured in fen water running through these wetlands, are the most important to explain the compositional changes.

First, prepare the data:

```
# Carpathian wetlands - import data
vasc <- read.delim
('https://raw.githubusercontent.com/zdealveindy/anadat-r/master/data/vasc_pl
ants.txt', row.names = 1)
chem <- read.delim
('https://raw.githubusercontent.com/zdealveindy/anadat-r/master/data/chemist
ry.txt', row.names = 1)

# transform data using Hellinger transformation to prepare them for tb-RDA:
library (vegan)
vasc.hell <- decostand (vasc, 'hell')

# the last variable in the 'chem' dataset is 'slope', which is not a
chemical variable - remove it:
chem1 <- chem[,-15]
```

Then, test the global model (including all explanatory variables from which you plan to do selection) to see whether it is significant; if it is not, the forward selection should not be done.

```
tb_rda.all <- rda (vasc.hell ~ ., chem1)
anova (tb_rda.all)  # 0.001 *** - it is significant
adjR2.tbrda <- RsquareAdj (tb_rda.all)$adj.r.squared # 0.2011048 - adjusted
R2 explained by all 14 variables is 20.1%
```

Option a) Use of forward.sel function

The function `forward.sel` from the package `adespatial` is elaborated forward selection approach based on linear constrained ordination (i.e. based on RDA - if you want to calculate CCA, you cannot use this function and need to resolve to use `ordiR2step` from `vegan` instead). Along to the testing significance of selected variable, this function includes also other stopping rules, published by Blanchet et al. (2008). The method based on significance is prone to be inflated, which may result into models with too many variables; Blanchet et al. (2008) came with another stopping rules along to the

significance one. First, adjusted R^2 of the model including all variables (from which the selection is made) is calculated. While selecting variables during the selection process, the selection stops if (1) the next selected variables is not significant (does not significantly improve the explained variation), and (2) if the R^2_{adj} of the model including this variable exceeds the R^2_{adj} of the global model.

```
# install.packages ('adespatial')  # run this code if you haven't install
the adespatial package before
library (adespatial)
sel.fs <- forward.sel (Y = vasc.hell, X = chem1, adjR2thresh = adjR2.tbrda)
# NOTE: forward.sel calculates only RDA (or tb-RDA in case of Hell.
transformed data)
```

```
Testing variable 1
Testing variable 2
Testing variable 3
Testing variable 4
Testing variable 5
Testing variable 6
Testing variable 7
Testing variable 8
Procedure stopped (alpha criteria): pvalue for variable 8 is 0.118000 (>
0.050000)
```

See the table with selected variables saved in `sel.fs` object:

```
sel.fs
```

```
  variables order         R2     R2Cum  AdjR2Cum         F  pval
1        Ca     1 0.13898260 0.1389826 0.1263206 10.976337 0.001
2   conduct    14 0.03248342 0.1714660 0.1467337  2.626796 0.001
3        Si     6 0.02722659 0.1986926 0.1622696  2.242529 0.001
4       NH3    10 0.02398868 0.2226813 0.1748463  2.005953 0.002
5       NO3     9 0.02112004 0.2438013 0.1847233  1.787470 0.001
6        Mg     2 0.01914995 0.2629513 0.1927562  1.636862 0.003
7        pH    13 0.01744989 0.2804012 0.1991562  1.503467 0.024
```

Or, if you want to see only the names of selected variables, subset it from the `sel.fs` object:

```
sel.fs$variables
```

```
[1] "Ca"      "conduct" "Si"      "NH3"     "NO3"     "Mg"      "pH"
```

Since there is (a potentially high) number of tests of significance during the forward selection procedure, it is better to apply a correction for multiple testing issue. The most common correction (but rather conservative) is Bonferroni, which multiplies each calculated *P*-value by the overall number of tests. In the forward selection, the overall number of tests (which can be potentially done) equals to the number of variables from which the selection is made (i.e. not the number of selected variables), and this should be considered while applying the adjustment.

The function available for adjusting the *P*-values is `p.adjust`. It can be applied on a single *P*-value or

a vector of *P*-values, and optionally accepts also the argument n indicating the number of tests.

The *P*-values from the forward selection above are in the object `sel.fs$pval`, and the adjustement can be done in the following way (Holm's correction):

```
n.tests <- ncol (chem1)  # number of tests equals to number of all variables
from which is being selected
pval.adj <- p.adjust (sel.fs$pval, method = 'holm', n = n.tests)
sel.fs$pval.adj <- pval.adj
sel.fs
```

```
  variables order         R2     R2Cum  AdjR2Cum         F  pval pval.adj
1        Ca     1 0.13898260 0.1389826 0.1263206 10.976337 0.001    0.014
2   conduct    14 0.03248342 0.1714660 0.1467337  2.626796 0.001    0.014
3        Si     6 0.02722659 0.1986926 0.1622696  2.242529 0.001    0.014
4       NH3    10 0.02398868 0.2226813 0.1748463  2.005953 0.003    0.033
5       NO3     9 0.02112004 0.2438013 0.1847233  1.787470 0.005    0.050
6        Mg     2 0.01914995 0.2629513 0.1927562  1.636862 0.014    0.126
7        pH    13 0.01744989 0.2804012 0.1991562  1.503467 0.020    0.160
```

After the correction, the last two variables (Mg and pH) are already not significant.

It is possible to help the intrepretation by adding the significance stars (e.g. three stars for P < 0.001). This can be done by many ways; perhaps the simplest is to use the function `stars.pval` from the package `gtools` (you need to install it if you haven't used it before):

```
# install.packages ('gtools')
sel.fs$pval.adj.stars <- gtools::stars.pval (pval.adj)
sel.fs
```

```
  variables order         R2     R2Cum  AdjR2Cum         F  pval pval.adj
pval.adj.stars
1        Ca     1 0.13898260 0.1389826 0.1263206 10.976337 0.001    0.014
*
2   conduct    14 0.03248342 0.1714660 0.1467337  2.626796 0.001    0.014
*
3        Si     6 0.02722659 0.1986926 0.1622696  2.242529 0.001    0.014
*
4       NH3    10 0.02398868 0.2226813 0.1748463  2.005953 0.003    0.033
*
5       NO3     9 0.02112004 0.2438013 0.1847233  1.787470 0.005    0.050
*
6        Mg     2 0.01914995 0.2629513 0.1927562  1.636862 0.014    0.126
7        pH    13 0.01744989 0.2804012 0.1991562  1.503467 0.020    0.160
```

Here, even the first selected variable has the significance *P* < 0.05 (one star). The reason is the number of permutations used in the `forward.sel` function, which by default equals to 999 (argument `nperm`). The lowest *P*-value which can be obtained with 999 permutations is 1/(999+1) = 0.001, and the correction will considerably increase this value (to 0.014 in case of 14 variables). The workaround is to increase the number of permutations to be able to get lower *P*-values, which even after correction could still fall below 0.001 threshold (here using 49,999 permutations to achieve P-

value 1/50000 = 0.00002).

```
sel.fs <- forward.sel (vasc.hell, chem1, nperm = 49999)
pval.adj <- p.adjust (sel.fs$pval, method = 'holm', n = ncol (chem1))
sel.fs$pval.adj <- pval.adj
sel.fs$pval.adj.stars <- gtools::stars.pval (pval.adj)
sel.fs
```

```
  variables order         R2     R2Cum  AdjR2Cum         F    pval pval.adj
pval.adj.stars
1        Ca     1 0.13898260 0.1389826 0.1263206 10.976337 0.00002  0.00028
***
2   conduct    14 0.03248342 0.1714660 0.1467337  2.626796 0.00004  0.00052
***
3        Si     6 0.02722659 0.1986926 0.1622696  2.242529 0.00016  0.00192
**
4       NH3    10 0.02398868 0.2226813 0.1748463  2.005953 0.00080  0.00880
**
5       NO3     9 0.02112004 0.2438013 0.1847233  1.787470 0.00304  0.03040
*
6        Mg     2 0.01914995 0.2629513 0.1927562  1.636862 0.00824  0.07416
.
7        pH    13 0.01744989 0.2804012 0.1991562  1.503467 0.02094  0.16752
```

Since the `forward.sel` function is not optimised, the calculation with 49,999 permutation take some time, but the reward is the *P*-values (after correction) below 0.001 in case of the first two variables.

You can see that `forward.sel` function selected (after the correction) five environmental variables, which together explain 18.5% of the variance (adjusted R2, the column `AdjR2Cum` in the table above showing cumulative adjusted R2 of all variables). This is just slightly less than 20.1% which is explained by global model with 14 variables (see the result of `RsquareAdj (rda.all)$adj.r.squared` above).

In this case, the results of `forward.sel` function are the same as `ordiR2step` and `ordistep` functions below (option b and c), with seven included environmental variables (if calculated without correction for multiple testing).

Option b) Use of ordiR2step function

The vegan's function `ordiR2step` does similar job as `forward.sel` from `adespatial`, but it's use is a bit more complex. On the other side, the method is more general - it allows to use also CCA or db-RDA methods, while `forward.sel` is based purely on linear constrained ordination (RDA, and tb-RDA in the case that the species composition data are pre-transformed e.g. by Hellinger transformation). In the default setting the criteria for including the variable is based on both significance of the newly selected variables, and the comparison of adjusted variation (R^2_{adj}) explained by the selected variables to R^2_{adj} explained by the global model (with all variables); if the new variable is not significant *or* the R^2_{adj} of the model including this new variable would exceed the R^2_{adj} of the global model, the selection will be stopped.

To use the `ordiR2step` function, you need to specify two models - the “empty” model containing

only intercept, and the "full" model including all variables (aka global model). Also, the argument `direction` allows for selecting the selection direction (`forward`, `backward`, `both`, with the default to `both`):

```
tb_rda.vasc.0 <- rda (vasc.hell ~ 1, data = chem1) # model containing only species matrix and intercept
tb_rda.vasc.all <- rda (vasc.hell ~ ., data = chem1) # model including all variables from matrix chem1 (the dot after tilda (~) means "include all from data")
```

The function `ordiR2step` then uses the empty model and the scope (variables) from the full model to proceed the selection. Set `direction = 'forward'` to ensure that forward instead of stepwise selection is done, and increse the number of permutations to 999 (instead of default 499) to make results comparable to `forward.sel` (which by default is based on 999 permutations):

```
sel.osR2 <- ordiR2step (tb_rda.vasc.0, scope = formula (tb_rda.vasc.all), R2scope = adjR2.tbrda, direction = 'forward', permutations = 999)
```

```
Step: R2.adj= 0
Call: vasc.hell ~ 1

                 R2.adjusted
<All variables>  0.201104784
+ Ca             0.126320581
+ conduct        0.109418731
+ Mg             0.106926067
+ Corg           0.076260217
+ pH             0.074169805
+ Na             0.043030978
+ NH3            0.031875777
+ Si             0.022863900
+ SO4            0.018929801
+ K              0.015918981
+ NO3            0.008112568
+ Fe             0.004863012
+ PO4            0.002933349
<none>           0.000000000
+ Cl            -0.002347612

     Df     AIC      F Pr(>F)
+ Ca  1 -47.149 10.976  0.001 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step: R2.adj= 0.1263206
Call: vasc.hell ~ Ca

                R2.adjusted
<All variables>   0.2011048
+ conduct         0.1467337
```

```
+ NH3             0.1417066
+ Na              0.1403837
+ Si              0.1403827
+ pH              0.1403289
+ Mg              0.1386000
+ Corg            0.1375824
+ NO3             0.1340217
+ K               0.1302515
+ Fe              0.1299960
+ SO4             0.1266017
<none>            0.1263206
+ Cl              0.1254155
+ PO4             0.1235944

          Df     AIC      F Pr(>F)
+ conduct  1 -47.841 2.6268  0.001 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step: R2.adj= 0.1467337
Call: vasc.hell ~ Ca + conduct

                R2.adjusted
<All variables>   0.2011048
+ Si              0.1622696
+ NH3             0.1592776
+ NO3             0.1572584
+ Mg              0.1565375
+ Corg            0.1538422
+ Na              0.1533884
+ Fe              0.1517167
+ pH              0.1513672
+ K               0.1511453
+ SO4             0.1495485
+ Cl              0.1470011
<none>            0.1467337
+ PO4             0.1444723

     Df    AIC      F Pr(>F)
+ Si  1 -48.18 2.2425  0.002 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step: R2.adj= 0.1622696
Call: vasc.hell ~ Ca + conduct + Si

                R2.adjusted
<All variables>   0.2011048
+ NH3             0.1748463
+ NO3             0.1739710
+ Mg              0.1708815
```

```
+ Corg            0.1689184
+ pH              0.1667252
+ SO4             0.1666206
+ Na              0.1661906
+ Fe              0.1655162
+ K               0.1653789
<none>            0.1622696
+ Cl              0.1620351
+ PO4             0.1599773

      Df     AIC     F Pr(>F)
+ NH3  1 -48.308 2.006  0.001 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step: R2.adj= 0.1748463
Call: vasc.hell ~ Ca + conduct + Si + NH3

                R2.adjusted
<All variables>   0.2011048
+ NO3             0.1847233
+ Mg              0.1835775
+ SO4             0.1799079
+ pH              0.1798906
+ Corg            0.1793391
+ Fe              0.1786200
+ K               0.1766585
+ Na              0.1756240
<none>            0.1748463
+ Cl              0.1737180
+ PO4             0.1725072

      Df     AIC      F Pr(>F)
+ NO3  1 -48.236 1.7875  0.005 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step: R2.adj= 0.1847233
Call: vasc.hell ~ Ca + conduct + Si + NH3 + NO3

                R2.adjusted
<All variables>   0.2011048
+ Mg              0.1927562
+ pH              0.1907475
+ Corg            0.1890699
+ SO4             0.1876281
+ Fe              0.1869468
+ K               0.1868746
+ Na              0.1848251
<none>            0.1847233
+ Cl              0.1834910
```

```
+ PO4             0.1826904

     Df     AIC      F Pr(>F)
+ Mg  1 -48.032 1.6369  0.011 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step: R2.adj= 0.1927562
Call: vasc.hell ~ Ca + conduct + Si + NH3 + NO3 + Mg

                R2.adjusted
<All variables>   0.2011048
+ pH              0.1991562
+ Corg            0.1963215
+ SO4             0.1958876
+ K               0.1956246
+ Fe              0.1953612
<none>            0.1927562
+ Na              0.1922729
+ Cl              0.1915194
+ PO4             0.1907648

     Df     AIC      F Pr(>F)
+ pH  1 -47.709 1.5035  0.028 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step: R2.adj= 0.1991562
Call: vasc.hell ~ Ca + conduct + Si + NH3 + NO3 + Mg + pH

                R2.adjusted
+ Corg            0.2023822
+ Fe              0.2015249
+ K               0.2014550
+ SO4             0.2012822
<All variables>   0.2011048
<none>            0.1991562
+ Na              0.1988753
+ PO4             0.1974424
+ Cl              0.1974018
```

```
sel.osR2
```

```
Call: rda(formula = vasc.hell ~ Ca + conduct + Si + NH3 + NO3 + Mg + pH,
data = chem1)

              Inertia Proportion Rank
Total          0.5674     1.0000
Constrained    0.1591     0.2804    7
Unconstrained  0.4083     0.7196   62
Inertia is variance
```

```
Eigenvalues for constrained axes:
   RDA1    RDA2    RDA3    RDA4    RDA5    RDA6    RDA7
0.09590 0.01904 0.01205 0.01027 0.00871 0.00718 0.00595

Eigenvalues for unconstrained axes:
    PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8
0.03601 0.03279 0.02383 0.01908 0.01687 0.01593 0.01504 0.01358
(Showed only 8 of all 62 unconstrained eigenvalues)
```

The summary table is part of the `ordiR2step` function output, stored in the component `$anova`:

```
sel.osR2$anova
```

```
                 R2.adj Df     AIC       F Pr(>F)
+ Ca            0.12632  1 -47.149 10.9763  0.001 ***
+ conduct       0.14673  1 -47.841  2.6268  0.001 ***
+ Si            0.16227  1 -48.180  2.2425  0.002 **
+ NH3           0.17485  1 -48.308  2.0060  0.001 ***
+ NO3           0.18472  1 -48.236  1.7875  0.005 **
+ Mg            0.19276  1 -48.032  1.6369  0.011 *
+ pH            0.19916  1 -47.709  1.5035  0.028 *
<All variables> 0.20111
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The result is exactly the same as results of `forward.sel` function and also as results of `ordistep` method based on P-values and AIC criteria (below). P-values can be adjusted for multiple testing (e.g. by Holm's correction) as follows:

```
sel.osR2_adj <- sel.osR2
sel.osR2_adj$anova$`Pr(>F)` <- p.adjust (sel.osR2$anova$`Pr(>F)`, method =
'holm', n = ncol (chem1))
sel.osR2_adj$anova
```

```
                 R2.adj Df     AIC       F Pr(>F)
+ Ca            0.12632  1 -47.149 10.9763  0.014 *
+ conduct       0.14673  1 -47.841  2.6268  0.014 *
+ Si            0.16227  1 -48.180  2.2425  0.022 *
+ NH3           0.17485  1 -48.308  2.0060  0.014 *
+ NO3           0.18472  1 -48.236  1.7875  0.050 *
+ Mg            0.19276  1 -48.032  1.6369  0.099 .
+ pH            0.19916  1 -47.709  1.5035  0.224
<All variables> 0.20111
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

You may consider to increase the number of permutations (argument `permutations` in `ordiR2step` to reach lower P-values; in the `forward.sel` example above we increased the permutations to 49,999 to make sure that even after adjustement of P-values the most significant one could be smaller than 0.001.

Option c) Use of ordistep function

Function `ordistep` performs step-wise selection of environmental variables based two criteria: if their inclusion into the model leads to significant increase of explained variance (the same as in `forward.sel` and `ordiR2step`), and if the AIC of the new model is lower than AIC of the more simple model. In contrary to previous functions, the function does not consider as criteria whether the adjusted R^2 of the model exceeds the adjusted R^2 of the global model. The use of `ordistep` function has the same logic as `ordiR2step`.

```
rda.vasc.0 <- rda (vasc.hell ~ 1, data = chem1) # model containing only
species matrix and intercept
rda.vasc.all <- rda (vasc.hell ~ ., data = chem1) # model including all
variables from matrix chem1 (the dot after tilda (~) means ALL!)
sel.os <- ordistep (rda.vasc.0, scope = formula (rda.vasc.all), direction =
'forward')
```

```
Start: vasc.hell ~ 1

          Df     AIC       F Pr(>F)
+ Ca       1 -47.149 10.9763  0.005 **
+ conduct  1 -45.808  9.4775  0.005 **
+ Mg       1 -45.612  9.2612  0.005 **
+ Corg     1 -43.249  6.6964  0.005 **
+ pH       1 -43.091  6.5277  0.005 **
+ Na       1 -40.775  4.1026  0.005 **
+ NH3      1 -39.964  3.2718  0.005 **
+ Si       1 -39.316  2.6145  0.005 **
+ SO4      1 -39.034  2.3314  0.005 **
+ K        1 -38.820  2.1162  0.010 **
+ NO3      1 -38.267  1.5643  0.060 .
+ Fe       1 -38.038  1.3372  0.105
+ PO4      1 -37.902  1.2030  0.150
+ Cl       1 -37.532  0.8384  0.700
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step: vasc.hell ~ Ca

          Df     AIC      F Pr(>F)
+ conduct  1 -47.841 2.6268  0.005 **
+ NH3      1 -47.430 2.2190  0.005 **
+ pH       1 -47.318 2.1081  0.005 **
+ Na       1 -47.322 2.1125  0.010 **
+ Si       1 -47.322 2.1124  0.010 **
+ Mg       1 -47.177 1.9694  0.010 **
+ Corg     1 -47.095 1.8880  0.010 **
+ NO3      1 -46.806 1.6047  0.035 *
+ K        1 -46.502 1.3073  0.110
+ Fe       1 -46.482 1.2873  0.115
+ SO4      1 -46.209 1.0219  0.435
```

```
+ Cl       1 -46.114 0.9296  0.590
+ PO4      1 -45.968 0.7885  0.890
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step: vasc.hell ~ Ca + conduct

       Df     AIC      F Pr(>F)
+ Si    1 -48.180 2.2425  0.005 **
+ NH3   1 -47.931 1.9997  0.005 **
+ NO3   1 -47.763 1.8367  0.005 **
+ Mg    1 -47.703 1.7788  0.005 **
+ Corg  1 -47.480 1.5629  0.020 *
+ Na    1 -47.442 1.5266  0.035 *
+ K     1 -47.257 1.3482  0.040 *
+ pH    1 -47.275 1.3658  0.050 *
+ SO4   1 -47.125 1.2218  0.085 .
+ Fe    1 -47.304 1.3936  0.095 .
+ Cl    1 -46.916 1.0210  0.420
+ PO4   1 -46.709 0.8229  0.810
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step: vasc.hell ~ Ca + conduct + Si

       Df     AIC      F Pr(>F)
+ NH3   1 -48.308 2.0060  0.005 **
+ NO3   1 -48.234 1.9350  0.005 **
+ Mg    1 -47.972 1.6855  0.020 *
+ Corg  1 -47.807 1.5280  0.025 *
+ SO4   1 -47.614 1.3446  0.055 .
+ pH    1 -47.622 1.3529  0.075 .
+ Na    1 -47.577 1.3104  0.095 .
+ Fe    1 -47.521 1.2568  0.105
+ K     1 -47.509 1.2459  0.130
+ Cl    1 -47.229 0.9815  0.470
+ PO4   1 -47.058 0.8199  0.805
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step: vasc.hell ~ Ca + conduct + Si + NH3

       Df     AIC      F Pr(>F)
+ NO3   1 -48.236 1.7875  0.005 **
+ Mg    1 -48.138 1.6951  0.005 **
+ SO4   1 -47.824 1.4012  0.035 *
+ pH    1 -47.822 1.3998  0.045 *
+ Corg  1 -47.775 1.3558  0.045 *
+ Fe    1 -47.714 1.2986  0.125
+ K     1 -47.547 1.1431  0.235
+ Na    1 -47.459 1.0613  0.370
```

```
+ Cl    1 -47.298 0.9112  0.665
+ PO4   1 -47.195 0.8163  0.805
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step: vasc.hell ~ Ca + conduct + Si + NH3 + NO3

       Df     AIC      F Pr(>F)
+ Mg    1 -48.032 1.6369  0.010 **
+ pH    1 -47.858 1.4764  0.040 *
+ Corg  1 -47.713 1.3430  0.040 *
+ SO4   1 -47.588 1.2288  0.135
+ Fe    1 -47.530 1.1750  0.170
+ K     1 -47.523 1.1693  0.210
+ Na    1 -47.347 1.0080  0.480
+ Cl    1 -47.233 0.9034  0.600
+ PO4   1 -47.164 0.8408  0.785
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step: vasc.hell ~ Ca + conduct + Si + NH3 + NO3 + Mg

       Df     AIC      F Pr(>F)
+ pH    1 -47.709 1.5035  0.040 *
+ SO4   1 -47.424 1.2453  0.095 .
+ Corg  1 -47.462 1.2795  0.100 .
+ Fe    1 -47.378 1.2040  0.115
+ K     1 -47.401 1.2247  0.135
+ Na    1 -47.110 0.9623  0.580
+ Cl    1 -47.044 0.9036  0.650
+ PO4   1 -46.979 0.8450  0.775
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Step: vasc.hell ~ Ca + conduct + Si + NH3 + NO3 + Mg + pH

       Df     AIC      F Pr(>F)
+ Corg  1 -47.130 1.2508  0.140
+ Fe    1 -47.054 1.1839  0.150
+ K     1 -47.048 1.1785  0.175
+ SO4   1 -47.033 1.1650  0.185
+ Na    1 -46.823 0.9783  0.595
+ PO4   1 -46.697 0.8676  0.715
+ Cl    1 -46.694 0.8645  0.735
```

Results are stored in `sel.os` object:

```
sel.os
```

```
Call: rda(formula = vasc.hell ~ Ca + conduct + Si + NH3 + NO3 + Mg + pH,
data = chem1)
```

```
              Inertia Proportion Rank
Total          0.5674     1.0000
Constrained    0.1591     0.2804    7
Unconstrained  0.4083     0.7196   62
Inertia is variance

Eigenvalues for constrained axes:
   RDA1    RDA2    RDA3    RDA4    RDA5    RDA6    RDA7
0.09590 0.01904 0.01205 0.01027 0.00871 0.00718 0.00595

Eigenvalues for unconstrained axes:
    PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8
0.03601 0.03279 0.02383 0.01908 0.01687 0.01593 0.01504 0.01358
(Showed only 8 of all 62 unconstrained eigenvalues)
```

Selected variables are Ca, conduct, Si, NH3, NO3, Mg and pH.

Example 2: CCA with forward selection on data from Carpathian wetlands

This example is using the same data as the Example 1 above, namely composition of vascular plants in Carpathian wetlands and chemical variables measured in fen water running through them. While in Example 1 we analysed the data using tb-RDA (RDA applied on data after Hellinger transformation), here we do this using CCA (note: no Hellinger transformation is done). Since the function `forward.sel` from `adespatial` package is using only RDA algorithm, we cannot use it here; instead, we will do the calculation using `ordiR2step` from `vegan`, which also implements variable selection with double stopping criteria: the selection is finished if the new variable to be selected is not significant at certain alpha level, or if the adjusted R^2 explained by the model with this variable exceeds the adjusted R^2 of the global model.

```
# Carpathian wetlands - import data
vasc <- read.delim
('https://raw.githubusercontent.com/zdealveindy/anadat-r/master/data/vasc_pl
ants.txt', row.names = 1)
chem <- read.delim
('https://raw.githubusercontent.com/zdealveindy/anadat-r/master/data/chemist
ry.txt', row.names = 1)

# the last variable in the 'chem' dataset is 'slope', which is not a
chemical variable - remove it:
chem1 <- chem[,-15]
```

We do similar steps as in the option b) of Example 1 above: create global model with CCA (including all variables) and test it's significance, and if it is significant, we will create also an empty model with only intercept and use the `ordiR2step` function with appropriate arguments:

```
cca.all <- cca (vasc ~ ., data = chem1)
anova (cca.all) # P < 0.001 - yes, significant!
adjRsq.cca <- RsquareAdj (cca.all)$adj.r.square  # adjR2 as stopping
criteria used in ordiR2step
```

```
cca.0 <- cca (vasc ~ 1, data = chem1) # empty model only with intercept
```

The function `ordiR2step` needs to know the variables which are being selected (this is defined by the empty model and the scope taken from the global model), the `direction` of the selection (`forward`, `backward`, `both` - default is `both`, which means combined forward and backward), the threshold of adjusted R^2 which should be used as stopping criteria (`R2scope`), number of `permutations` for the Monte Carlo permutation test applied on each variable (default is 499). Additionally, I set also argument `trace = FALSE` (default is TRUE), which limits the verbal output of the function (as you can see above, the function is rather talkative).

```
sel.cca <- ordiR2step (cca.0, scope = formula (cca.all), R2scope =
adjRsq.cca, direction = 'forward', permutations = 999, trace = FALSE)
sel.cca
```

```
Call: cca(formula = vasc ~ Ca + conduct + Si + NH3 + NO3 + SO4 + Corg, data
= chem1)

              Inertia Proportion Rank
Total          3.0238     1.0000
Constrained    0.6772     0.2240    7
Unconstrained  2.3466     0.7760   62
Inertia is scaled Chi-square

Eigenvalues for constrained axes:
  CCA1   CCA2   CCA3   CCA4   CCA5   CCA6   CCA7
0.3388 0.0942 0.0618 0.0569 0.0485 0.0418 0.0352

Eigenvalues for unconstrained axes:
    CA1     CA2     CA3     CA4     CA5     CA6     CA7     CA8
0.17567 0.16047 0.12242 0.10303 0.09427 0.08952 0.08455 0.07697
(Showing 8 of 62 unconstrained eigenvalues)
```

```
sel.cca$anova
```

```
                  R2.adj Df    AIC      F Pr(>F)
+ Ca            0.082091  1 385.90 7.1738  0.001 ***
+ conduct       0.095737  1 385.81 2.0268  0.001 ***
+ Si            0.108800  1 385.76 1.9704  0.001 ***
+ NH3           0.117746  1 385.96 1.6923  0.006 **
+ NO3           0.125431  1 386.27 1.5578  0.019 *
+ Corg          0.130893  1 386.75 1.3870  0.022 *
+ SO4           0.136586  1 387.17 1.4114  0.041 *
+ Mg            0.141018  1 387.65 1.3382  0.043 *
<All variables> 0.141602
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The first five variables (Ca, conductivity, Si, NH3 and NO3) are the same as in tb-RDA analysis above, the last two differ. If we increase the number of permutations to 49,999 (to reduce the lowest P-value we can obtain - but this takes quite some time!) and adjust the P-values with Holm's correction for

multiple testing issue, only the first four variables will be selected (compared to the first five in the case of tb-RDA):

```
sel.cca.49999 <- ordiR2step (cca.0, scope = formula (cca.all), direction =
'forward', R2scope = adjRsq.cca, permutations = 49999, trace = F)
sel.cca.49999_adj <- sel.cca.49999
sel.cca.49999_adj$anova$`Pr(>F)` <- p.adjust (sel.cca.49999$anova$`Pr(>F)`,
method = 'holm', n = ncol (chem1))
sel.cca.49999_adj$anova
```

```
                  R2.adj Df    AIC      F  Pr(>F)
+ Ca            0.082081  1 385.90 7.1738 0.00028 ***
+ conduct       0.095628  1 385.81 2.0268 0.00312 **
+ Si            0.108801  1 385.76 1.9704 0.00312 **
+ NO3           0.117863  1 385.99 1.6612 0.04378 *
+ NH3           0.125336  1 386.27 1.5884 0.11820
+ SO4           0.131177  1 386.73 1.4028 0.30888
+ Mg            0.136262  1 387.19 1.3789 0.30888
+ Corg          0.141158  1 387.65 1.3550 0.30888
<All variables> 0.142456
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

You may have noticed that the use of `ordiR2step` function with CCA takes considerably longer than when applied on RDA (or tb-RDA as above). This is because in the case of CCA, adjusted R2 needs to be calculated using the permutation method introduced by Peres-Neto et al. (2006), while in the case of RDA the adjusted R^2 can be calculated analytically by Ezekiel's formula. This also means that in CCA, the resulting adjusted R^2 values will slightly change between calculations, because the adjusted R^2 value calculation is using the mean of R^2 explained by randomized environmental variables (by default 1000 randomizations is used; this number can be increased by including the argument `R2permutations` in the `ordiR2step` function with a higher number). These small changes between calculation can eventually lead to selecting slightly different sets of variables or selecting them in a different order (as in the case of this example - when you see the results of the forward selection in CCA, the first one selected variables in the order Ca > conduct > Si > NH3 > NO3 > Corg > SO4 > Mg, while the latter (with 49,999 permutations) selected them in the order Ca > conduct > Si > NO3 > NH3 > SO4 > Mg > Corg (the order changed after the third selected variable). Increasing `R2permutations` should decrease such fluctuation (but will also increase the calculation time).