MVA Course

First we need to import the data we will be using. This is included in the "vegan" package.

```
library(vegan)# Package for multivariate analyses of ecological data
```

```
## Loading required package: permute
```

```
## Loading required package: lattice
```

```
## This is vegan 2.5-5
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

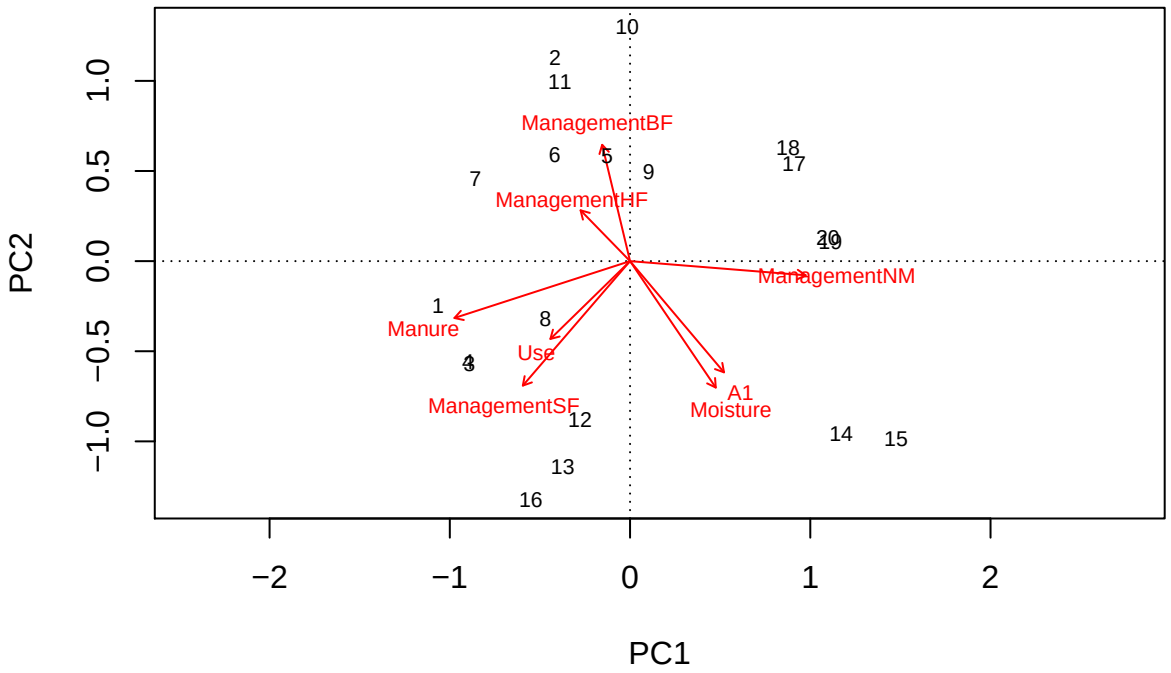
```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
data("dune")
data("dune.env")
#Management factor is character but we need it to be numeric for some analyses. Convert to "dummy" vari
#If you are converting a factor that is numbers already you need as.numeric(as.character(x))

#give each management method its own column
dummy_management <- as.data.frame(model.matrix( ~ Management - 1, data=dune.env ))
#add these to the dataset
dune.env.original <- dune.env
dune.env <- dune.env %>% select(A1, Moisture, Manure, Use) %>% cbind(.,dummy_management)
dune.env$Moisture <- as.numeric(as.character(dune.env$Moisture)) #make numeric
dune.env$Manure <- as.numeric(as.character(dune.env$Manure))
dune.env$Use <- as.numeric(dune.env$Use)
```

Question 2: Do a PCA on the environmental data related to the Dune meadow dataset.

What are the results telling you? In what way do objects/samples to the left differ from objects to the right, and at the bottom from those at the top? Which are the most important gradients in the dataset? Which descriptor variables are related, and which are unrelated?

```
env.pca <- rda(dune.env, scale = TRUE) #vegan uses the same function for PCA and RDA
biplot(env.pca)#plot the results
```




```
env.pca #summarise results
```

```
## Call: rda(X = dune.env, scale = TRUE)
##
##               Inertia Rank
## Total                8
## Unconstrained        8    7
## Inertia is correlations
##
## Eigenvalues for unconstrained axes:
##    PC1    PC2    PC3    PC4    PC5    PC6    PC7
## 2.7348 1.9195 1.3056 1.0477 0.5106 0.4141 0.0677
```

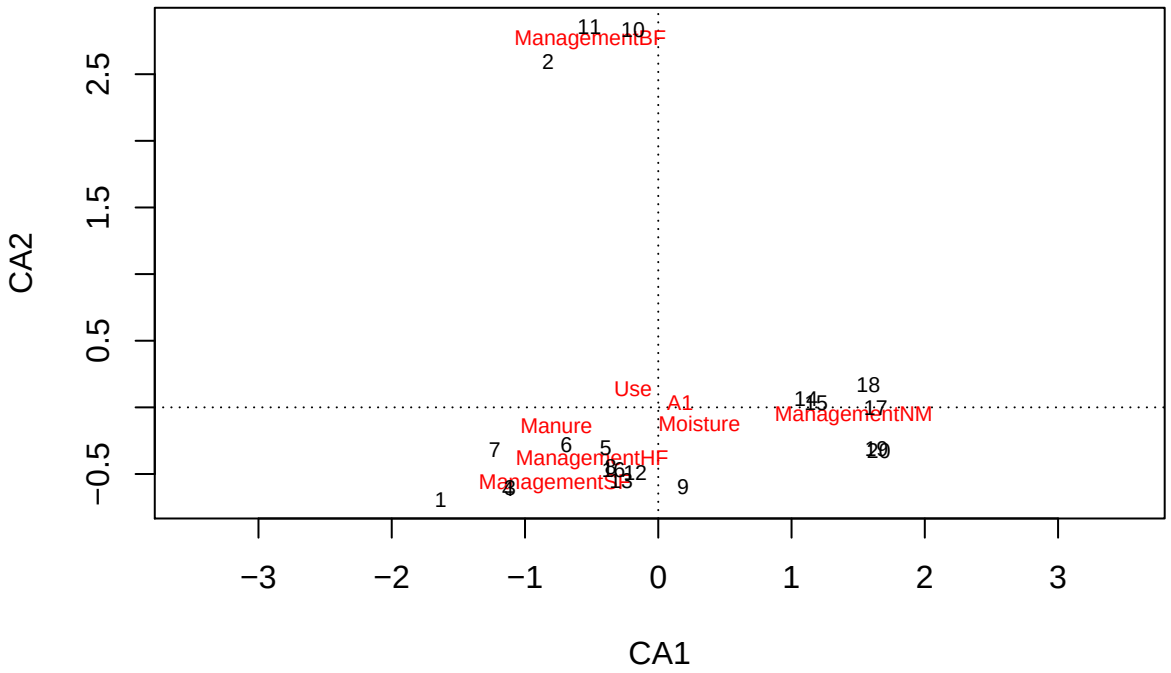
```
summary(eigenvals(env.pca)) #see variance explained
```

```
## Importance of components:
##                            PC1    PC2    PC3   PC4     PC5     PC6      PC7
## Eigenvalue              2.7348 1.9195 1.3056 1.048 0.51059 0.41413 0.067668
## Proportion Explained    0.3418 0.2399 0.1632 0.131 0.06382 0.05177 0.008458
## Cumulative Proportion 0.3418 0.5818 0.7450 0.876 0.93977 0.99154 1.000000
```

Question 3: For comparison, do also a CA on the Dune Meadow Environmental variables and compare the result with the PCA on the same data!

Why do the results in exercise 2 and 3 differ?

```
#unconstrained ordination on environmental data (CA)
env.ca <- cca(dune.env) #vegan uses the same function for CA and CCA
plot(env.ca)
```




```
env.ca
```

```
## Call: cca(X = dune.env)
##
##               Inertia Rank
## Total          0.4709
## Unconstrained  0.4709    7
## Inertia is scaled Chi-square
##
## Eigenvalues for unconstrained axes:
##     CA1     CA2     CA3     CA4     CA5     CA6     CA7
## 0.19532 0.11305 0.08393 0.04417 0.02144 0.01081 0.00215
```

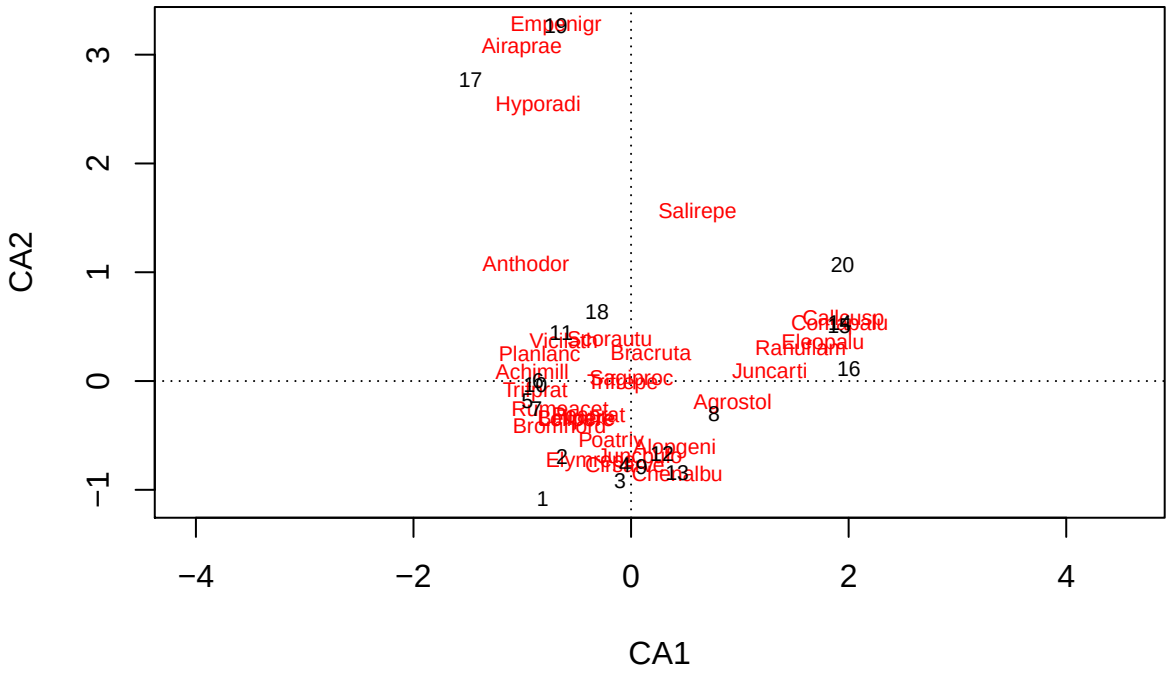
```
summary(eigenvals(env.ca)) #proportion variance explained
```

```
## Importance of components:
##                            CA1    CA2     CA3     CA4     CA5     CA6
## Eigenvalue            0.1953 0.1130 0.08393 0.04417 0.02144 0.01081
## Proportion Explained  0.4148 0.2401 0.17825 0.09380 0.04554 0.02295
## Cumulative Proportion 0.4148 0.6549 0.83315 0.92695 0.97249 0.99544
##                            CA7
## Eigenvalue            0.002146
## Proportion Explained  0.004557
## Cumulative Proportion 1.000000
```

Question 4: Do a CA-ordination on the Dune Meadow species dataset.What are the results telling you?

Give a conceptual description on why objects/samples and descriptors/species to the left differ from objects and descriptors to the right, and those at the bottom from those at the top! (Plant ecologists may give a more detailed description, using their knowledge about the species in the dataset).

```
dune.ca <- cca(dune)
plot(dune.ca)
```




```
dune.ca
```

```
## Call: cca(X = dune)
##
##               Inertia Rank
## Total           2.115
## Unconstrained   2.115   19
## Inertia is scaled Chi-square
##
## Eigenvalues for unconstrained axes:
##    CA1    CA2    CA3    CA4    CA5    CA6    CA7    CA8
## 0.5360 0.4001 0.2598 0.1760 0.1448 0.1079 0.0925 0.0809
## (Showing 8 of 19 unconstrained eigenvalues)
```

```
summary(eigenvals(dune.ca)) #proportion variance explained
```

```
## Importance of components:
##                            CA1    CA2    CA3     CA4     CA5     CA6     CA7
## Eigenvalue            0.5360 0.4001 0.2598 0.17598 0.14476 0.10791 0.09247
## Proportion Explained  0.2534 0.1892 0.1228 0.08319 0.06844 0.05102 0.04372
## Cumulative Proportion 0.2534 0.4426 0.5654 0.64858 0.71702 0.76804 0.81175
##                            CA8     CA9    CA10    CA11    CA12    CA13
## Eigenvalue            0.08091 0.07332 0.05630 0.04826 0.04125 0.03523
## Proportion Explained  0.03825 0.03466 0.02661 0.02282 0.01950 0.01665
## Cumulative Proportion 0.85000 0.88467 0.91128 0.93410 0.95360 0.97025
##                           CA14     CA15     CA16     CA17     CA18
## Eigenvalue            0.020529 0.014911 0.009074 0.007938 0.007002
## Proportion Explained  0.009705 0.007049 0.004290 0.003753 0.003310
## Cumulative Proportion 0.979955 0.987004 0.991293 0.995046 0.998356
##                           CA19
## Eigenvalue            0.003477
```

```
## Proportion Explained  0.001644
## Cumulative Proportion 1.000000
```

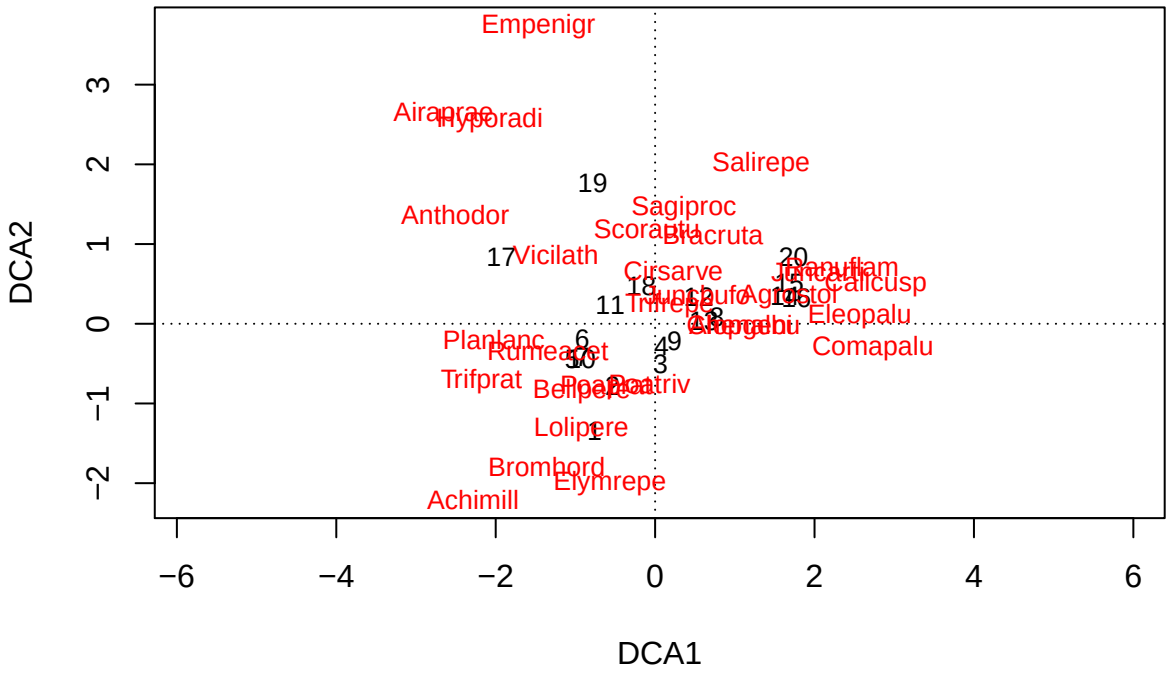
Question 5: Repeat exercise 4, but with DCA ordination instead.

Look at the eigenvalues, the length of gradient, the total variation and the ordination diagram. Explain the differences between results from CA and DCA.

```
dune.dca <- decorana(dune)
dune.dca
```

```
##
## Call:
## decorana(veg = dune)
##
## Detrended correspondence analysis with 26 segments.
## Rescaling of axes with 4 iterations.
##
##                   DCA1   DCA2    DCA3    DCA4
## Eigenvalues     0.5117 0.3036 0.12125 0.14267
## Decorana values 0.5360 0.2869 0.08136 0.04814
## Axis lengths    3.7004 3.1166 1.30055 1.47888
```

```
#Detrended correspondence analysis (function decorana).
#The total amount of variation is undefined in detrended
#correspondence analysis and therefore proportions from total
#are unknown and undefined.
#DCA is not a method for decomposition of variation, and therefore
#these proportions would not make sense either.

plot(dune.dca)
```



Question 21: Interpreting ordination results using species traits

Experienced plant ecologists may already have looked at the species in the Dune Meadow ordination graphs and concluded that species with similar traits occur together. This is possible if you have good knowledge about plant species ecological preferences, and if there are relatively few species in your dataset. In this exercise we will use tabulated data on species ecological preferences (the Ellenberg indicator values) to interpret the results of the ordinations. The Ellenberg indicator values are described on the first page in this booklet. Which Ellenberg values are most important for the distribution of the species? What do the different axes represent in terms of environmental gradients?

```
#Ulf's version of Dune data with ellenbergs
dune.u <- read.table("Data/Data_for_R/dune.txt", sep=";",header=T,row.names=1)
dune.ell.u <- read_csv("Data/exports_from_canoco/Ellenberg.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1], 'X2' [2]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_character(),
##   L = col_double(),
##   T = col_double(),
##   K = col_double(),
##   F = col_double(),
##   R = col_double(),
##   N = col_double(),
##   S = col_double()
## )
```

```
dune.mean.ell <- read_csv("Data/exports_from_canoco/meanell.csv", col_types = cols(Plot = col_skip()))
dune.env.u <- read.table("Data/Data_for_R/env.txt", sep="\t",header=T,row.names=1)

## Quick code to replace missing values with the column mean, forward selection fails with NA
## Need to tweak dune data really

dune.mean.ell.impute <- data.frame(
    sapply(
        dune.mean.ell,
        function(x) ifelse(is.na(x),
            mean(x, na.rm = TRUE),
            x)))


#CCA analysis
#create global model with CCA (including all variables) and test it's significance, and if it is signif
#we use the ordistep function with appropriate arguments to do forward selection of variables.
cca1 <- cca(dune.u ~ ., data = dune.mean.ell.impute) # full model (with all explanatory variables)
anova(cca1) #overall model is significant
```

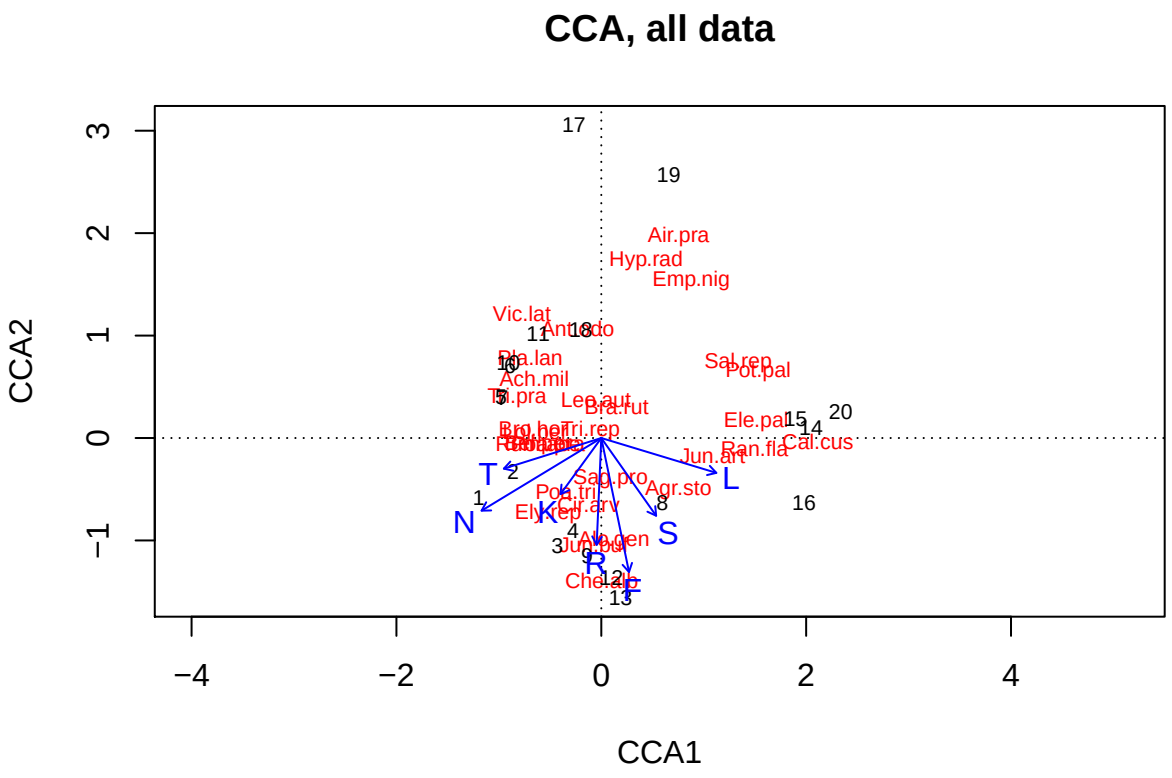
```
## Permutation test for cca under reduced model
## Permutation: free
## Number of permutations: 999
##
## Model: cca(formula = dune.u ~ L + T + K + F + R + N + S, data = dune.mean.ell.impute)
##          Df ChiSquare      F Pr(>F)
## Model     7   1.33129 2.9111  0.001 ***
```

```
## Residual 12   0.78397
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
adjR2.cca <- RsquareAdj (cca1)$adj.r.squared

# Start by make ordinations
cca0 <- cca (dune.u~ 1, data = dune.mean.ell.impute) # empty model only with intercept
cca1 <- cca(dune.u ~ ., data = dune.mean.ell.impute, na.action = na.omit) # full model (with all explan
cca1
```

```
## Call: cca(formula = dune.u ~ L + T + K + F + R + N + S, data =
## dune.mean.ell.impute, na.action = na.omit)
##
##               Inertia Proportion Rank
## Total          2.1153     1.0000
## Constrained    1.3313     0.6294    7
## Unconstrained  0.7840     0.3706   12
## Inertia is scaled Chi-square
##
## Eigenvalues for constrained axes:
##   CCA1   CCA2   CCA3   CCA4   CCA5   CCA6   CCA7
## 0.4646 0.3298 0.1774 0.1490 0.1091 0.0597 0.0417
##
## Eigenvalues for unconstrained axes:
##     CA1     CA2     CA3     CA4     CA5     CA6     CA7     CA8     CA9
## 0.24494 0.14476 0.08921 0.07975 0.06167 0.04826 0.03878 0.02701 0.01952
##    CA10    CA11    CA12
## 0.01271 0.01006 0.00729
```

```
plot(cca1, main="CCA, all data")
```



```r
#Stepwise approach, using "ordistep"
step2<-ordistep(cca0, scope = formula(cca1), perm.max=9999)
```

```
##
## Start: dune.u ~ 1
##
##     Df    AIC      F Pr(>F)
## + N  1 85.551 4.1029  0.005 **
## + L  1 86.138 3.4630  0.005 **
## + T  1 86.624 2.9480  0.005 **
## + F  1 86.635 2.9367  0.010 **
## + S  1 87.111 2.4440  0.010 **
## + R  1 87.551 1.9994  0.035 *
## + K  1 87.850 1.7028  0.060 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Step: dune.u ~ N
##
##     Df    AIC      F Pr(>F)
## - N  1 87.657 4.1029  0.005 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##     Df    AIC      F Pr(>F)
## + F  1 83.748 3.5595  0.005 **
## + L  1 84.501 2.8000  0.005 **
## + S  1 84.490 2.8115  0.010 **
## + R  1 84.753 2.5526  0.015 *
## + T  1 85.234 2.0875  0.035 *
## + K  1 85.992 1.3778  0.195
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Step: dune.u ~ N + F
##
##     Df    AIC      F Pr(>F)
## - F  1 85.551 3.5595  0.005 **
## - N  1 86.635 4.7046  0.005 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##     Df    AIC      F Pr(>F)
## + T  1 82.828 2.5154  0.015 *
## + L  1 83.543 1.8649  0.025 *
## + S  1 83.716 1.7118  0.035 *
## + R  1 83.824 1.6159  0.095 .
## + K  1 83.881 1.5660  0.105
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Step: dune.u ~ N + F + T
##
##     Df    AIC      F Pr(>F)
```

```
## - T  1 83.748 2.5154  0.015 *
## - N  1 84.964 3.6753  0.005 **
## - F  1 85.234 3.9434  0.005 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##     Df    AIC      F Pr(>F)
## + L  1 82.381 1.9524  0.015 *
## + R  1 82.582 1.7830  0.050 *
## + S  1 82.728 1.6611  0.055 .
## + K  1 83.052 1.3932  0.190
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Step: dune.u ~ N + F + T + L
##
##     Df    AIC      F Pr(>F)
## - L  1 82.828 1.9524  0.030 *
## - T  1 83.543 2.5697  0.020 *
## - F  1 83.875 2.8631  0.015 *
## - N  1 83.882 2.8693  0.005 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##     Df    AIC      F Pr(>F)
## + S  1 81.835 1.9002  0.055 .
## + R  1 82.299 1.5356  0.095 .
## + K  1 82.328 1.5135  0.140
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
step2$anova # N F T L S sig
```

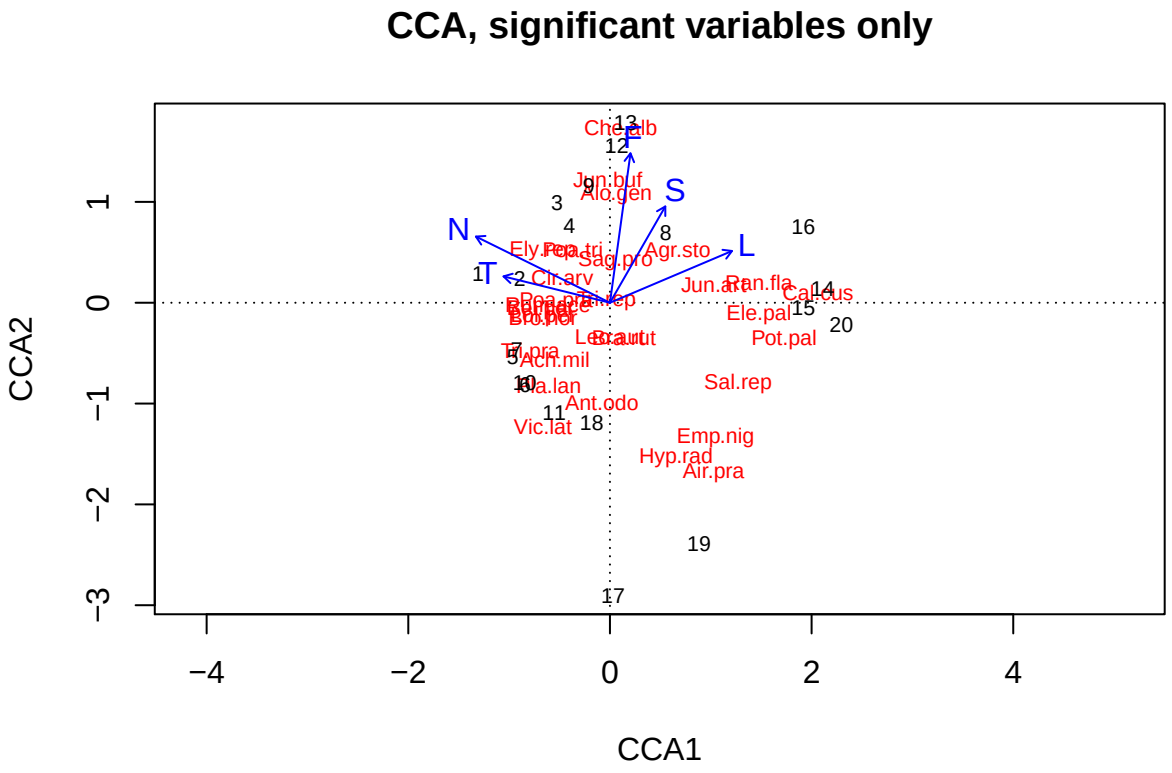
```
##     Df    AIC      F Pr(>F)
## + N  1 85.551 4.1029  0.005 **
## + F  1 83.748 3.5595  0.005 **
## + T  1 82.828 2.5154  0.015 *
## + L  1 82.381 1.9524  0.015 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#alternatively
#sel.osR2 <- ordiR2step (cca0, scope = formula (cca1), R2scope = adjR2.cca, direction = 'forward', perm
#sel.osR2$anova # N F T L S sig

#or model selection using permutations
# By terms
#anova(cca1, by="term", perm.max = 200)
# By margin
#anova(cca1, by="margin", perm.max = 200)
# By axis
#anova(cca1, by="axis", perm.max = 200)

# New model with only significant explanatory variables?
cca3 <- cca(dune.u ~ N + F + T + L + S, data = dune.mean.ell.impute)
```

```
cca3
```

```
## Call: cca(formula = dune.u ~ N + F + T + L + S, data =
## dune.mean.ell.impute)
##
##               Inertia Proportion Rank
## Total          2.1153     1.0000
## Constrained    1.1563     0.5467    5
## Unconstrained  0.9590     0.4533   14
## Inertia is scaled Chi-square
##
## Eigenvalues for constrained axes:
##   CCA1   CCA2   CCA3   CCA4   CCA5
## 0.4550 0.3168 0.1695 0.1461 0.0689
##
## Eigenvalues for unconstrained axes:
##     CA1     CA2     CA3     CA4     CA5     CA6     CA7     CA8     CA9
## 0.27289 0.16766 0.09949 0.09653 0.08494 0.06400 0.04940 0.03751 0.02673
##    CA10    CA11    CA12    CA13    CA14
## 0.01981 0.01224 0.01158 0.00940 0.00677
```

```
ordiplot(cca3, main = "CCA, significant variables only", type = "text")
```



Question 22: Decomposition of variance

In usual analysis of variance experiments, the variance is decomposed into components. The same can be done in multivariate analyses. In this exercise you will decompose the variance in the Dune meadow data set into different variance components. You will use two groups of variance components: Group 1 is Management and Group 2 is the soil variables (A1 and Moisture). In ordinations, the variance is expressed by the sum of the eigenvalues.

The result gives you the fraction of the total variation that is explained by: a) uniquely by Management (effect of soil removed), b) uniquely by soil (effect of Management removed), c) jointly by Soil and Management (the interaction).

The total explained variation is the sum of a), b) and c).

The total variation in the dataset is the sum of all unconstrained eigenvalues.

Perform the analyses and interpret the results! How much of the total variation (% of All) is explained by: - uniquely by Management (effect of Soil removed), - uniquely by Soil (effect of Management removed), - jointly by Soil and Management (the interaction) - Not by Soil or Management?

```
## variance partitioning

# because management is a factor we need to convert it into
# a dummy matrix using the function dummies::dummy
library(dummies)
```

```
## dummies-1.5.6 provided by Decision Patterns
```

```
management <-  dummy(dune.env.original$Management)
```

```
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts =
## FALSE): non-list contrasts argument ignored
```

```
#management <- dune.env[,c("ManagementBF" + "ManagementHF" + "ManagementNM" + "ManagementSF")]
soil <- dune.env.original[,c("A1", "Moisture")]
# examine the explanatory variable of each class of variables.
varp <- varpart(dune, management, soil)
```

```
## Warning: collinearity detected in X1: mm = 4, m = 3
```

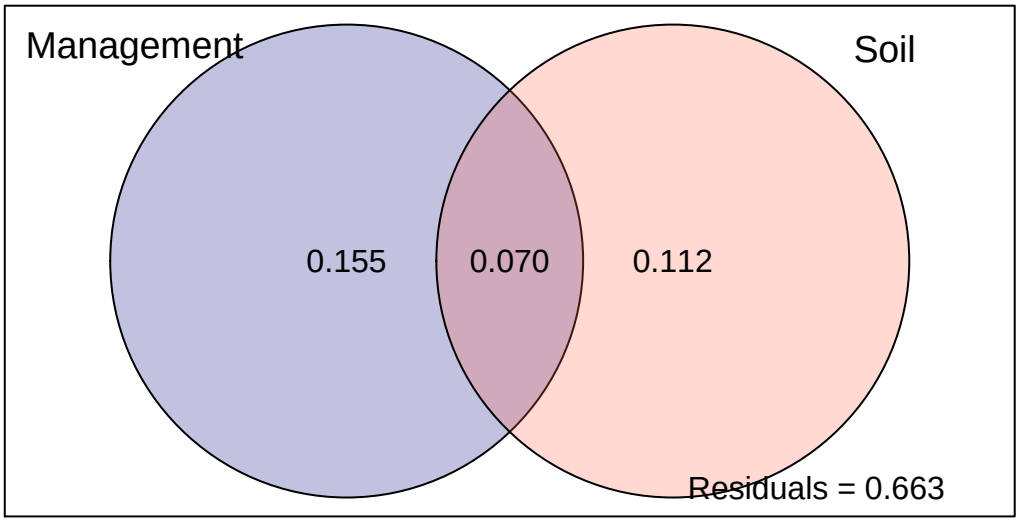
```
## Warning: collinearity detected in cbind(X1,X2): mm = 8, m = 7
```

```
#varp2 <- varpart (dune, ~ ManagementBF + ManagementHF + ManagementNM + ManagementSF,
# ~ A1 + Moisture, data = dune.env)
varp
```

```
##
## Partition of variance in RDA
##
## Call: varpart(Y = dune, X = management, soil)
##
## Explanatory tables:
## X1:  management
## X2:  soil
##
## 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
```

```
plot (varp, digits = 2, Xnames = c('Management', 'Soil'), bg = c('navy', 'tomato'))
```



Question 23: Diagnostic species for a priori defined groups

In this exercise we will investigate if any particular species are indicative for the four different management types. The CANOCO version of this analysis is a simplified version of these types of tests. The full versions also give significance testing and more comprehensive result presentations. See for instance IndVal by Dufresne & Legendre. 1. Go to menu Data /Add new table(s) / Indicator values. 2. Mark the Species data table, and Management as the factor defining groups. 3. Opt for Indicator Value (quantitative). 4. Export the table to Excel to be able to sort the data to find the highest values = best indicators for a group. Use the resulting data table to find the two top indicator species for each of the four Management types!

```
library(labdsv)
```

```
## Loading required package: mgcv
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
## This is mgcv 1.8-28. For overview type 'help("mgcv-package")'.
```

```
## This is labdsv 2.0-1
## convert existing ordinations with as.dsvord()
```

```
##
## Attaching package: 'labdsv'
```

```
## The following object is masked from 'package:stats':
##
##     density
```

```r
iva <- indval(dune, dune.env.original$Management)

iva.df <- as.data.frame(iva$indval)
#arrange in order to find best indicator species, change BF to other management types as needed
arrange(rownames_to_column(iva.df), BF)
```

```
##     rowname         BF         HF          NM          SF
## 1  Agrostol 0.00000000 0.06801619 0.131578947 0.472334683
## 2  Airaprae 0.00000000 0.00000000 0.333333333 0.000000000
## 3  Chenalbu 0.00000000 0.00000000 0.000000000 0.166666667
## 4  Cirsarve 0.00000000 0.00000000 0.000000000 0.166666667
## 5  Comapalu 0.00000000 0.00000000 0.333333333 0.000000000
## 6  Eleopalu 0.00000000 0.03720930 0.251937984 0.051679587
## 7  Empenigr 0.00000000 0.00000000 0.166666667 0.000000000
## 8  Juncarti 0.00000000 0.19591837 0.119047619 0.025510204
## 9  Juncbufo 0.00000000 0.20281690 0.000000000 0.164319249
## 10 Ranuflam 0.00000000 0.03333333 0.277777778 0.092592593
## 11 Rumeacet 0.00000000 0.72452830 0.000000000 0.015723270
## 12 Salirepe 0.00000000 0.00000000 0.500000000 0.000000000
## 13 Trifprat 0.00000000 0.60000000 0.000000000 0.000000000
## 14 Callcusp 0.00000000 0.00000000 0.233333333 0.050000000
## 15 Alopgeni 0.03367003 0.09696970 0.000000000 0.547138047
## 16 Sagiproc 0.05847953 0.08421053 0.021929825 0.241228070
## 17 Elymrepe 0.08333333 0.15000000 0.000000000 0.187500000
## 18 Anthodor 0.09950249 0.24179104 0.099502488 0.000000000
## 19 Hyporadi 0.12121212 0.00000000 0.212121212 0.000000000
## 20 Bracruta 0.13840830 0.29065744 0.196078431 0.138408304
## 21  Poatriv 0.18612521 0.36548223 0.000000000 0.355329949
## 22 Planlanc 0.22857143 0.30857143 0.047619048 0.000000000
## 23 Bellpere 0.36231884 0.02608696 0.018115942 0.072463768
## 24 Scorautu 0.37249284 0.24068768 0.272206304 0.076408787
## 25  Poaprat 0.37854890 0.32176656 0.021030494 0.157728707
## 26 Achimill 0.38567493 0.17851240 0.013774105 0.006887052
## 27 Trifrepe 0.43887147 0.26332288 0.114942529 0.083594566
## 28 Bromhord 0.44817927 0.08067227 0.000000000 0.021008403
## 29 Lolipere 0.45000000 0.30000000 0.004166667 0.112500000
## 30 Vicilath 0.57142857 0.00000000 0.023809524 0.000000000
```

```r
#or create dataframe of best results for each management
#gr <- iva$maxcls[iva$pval<0.1]
#iv <- iva$indcls[iva$pval<0.1]
#pv <- iva$pval[iva$pval<0.1]

#indvalsummary <- data.frame(group=gr, indval=iv, pvalue=pv)
#indvalsummary <- indvalsummary[order(indvalsummary$group, -indvalsummary$indval),]
#indvalsummary
```

Question 24: Analysing a time series with vegetation data

Quite often vegetation ecologists have data from repeated inventories in permanent plots. The overall question to answer is if there has been a significant and directional change in the species composition. In this exercise we have rearranged the Dune Meadow data so that it consists of only 10 plots, each analysed 2 times (same species as in all other exercises, just grouping and rearrangement of plots). We want to analyse if there has been a consistent change in species composition between the two inventories. We are thus not interested in

differences between the ten plots.

```
#import data
EnvTS <- read_csv("~/Documents/Courses/MVA R version/exports from canoco/EnvTS.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   `A1 hor` = col_double(),
##   Moisture = col_double(),
##   Use = col_double(),
##   Manure = col_double(),
##   Managmnt = col_character(),
##   Plot = col_double(),
##   Time = col_double(),
##   Treat = col_character()
## )
```

```
SpeTS <- read_csv("~/Documents/Courses/MVA R version/exports from canoco/SpeTS.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   X1 = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
rownames(EnvTS) <- EnvTS$X1
```

```
## Warning: Setting row names on a tibble is deprecated.
```

```
rownames(SpeTS) <- SpeTS$X1
```

```
## Warning: Setting row names on a tibble is deprecated.
```

```
EnvTS$X1 <- NULL
SpeTS$X1 <- NULL
```

```
#This partials out the effect of Plot before analysing the effects of Time
time.cca <- cca(SpeTS ~ Time + Condition(Plot), data=EnvTS)
time.cca
```

```
## Call: cca(formula = SpeTS ~ Time + Condition(Plot), data = EnvTS)
##
##               Inertia Proportion Rank
## Total         2.11526    1.00000
## Conditional   0.20908    0.09885    1
## Constrained   0.15941    0.07536    1
## Unconstrained 1.74677    0.82579   17
## Inertia is scaled Chi-square
##
## Eigenvalues for constrained axes:
##    CCA1
## 0.15941
##
```

```
## Eigenvalues for unconstrained axes:
##    CA1    CA2    CA3    CA4    CA5    CA6    CA7    CA8
## 0.4215 0.2761 0.2461 0.1722 0.1358 0.0999 0.0913 0.0783
## (Showing 8 of 17 unconstrained eigenvalues)
```

```
#permutation test
anova(time.cca, by="term", perm=999)
```

```
## Permutation test for cca under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## Model: cca(formula = SpeTS ~ Time + Condition(Plot), data = EnvTS)
##          Df ChiSquare      F Pr(>F)
## Time      1   0.15941 1.5515  0.097 .
## Residual 17   1.74677
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#with(EnvTS, anova(time.cca, by="term", perm=500, strata=Plot))
```

```
#Compare with results for time as only explanatory variable and no cofactors
time2.cca <- cca(SpeTS ~ Time, data=EnvTS)
time2.cca
```

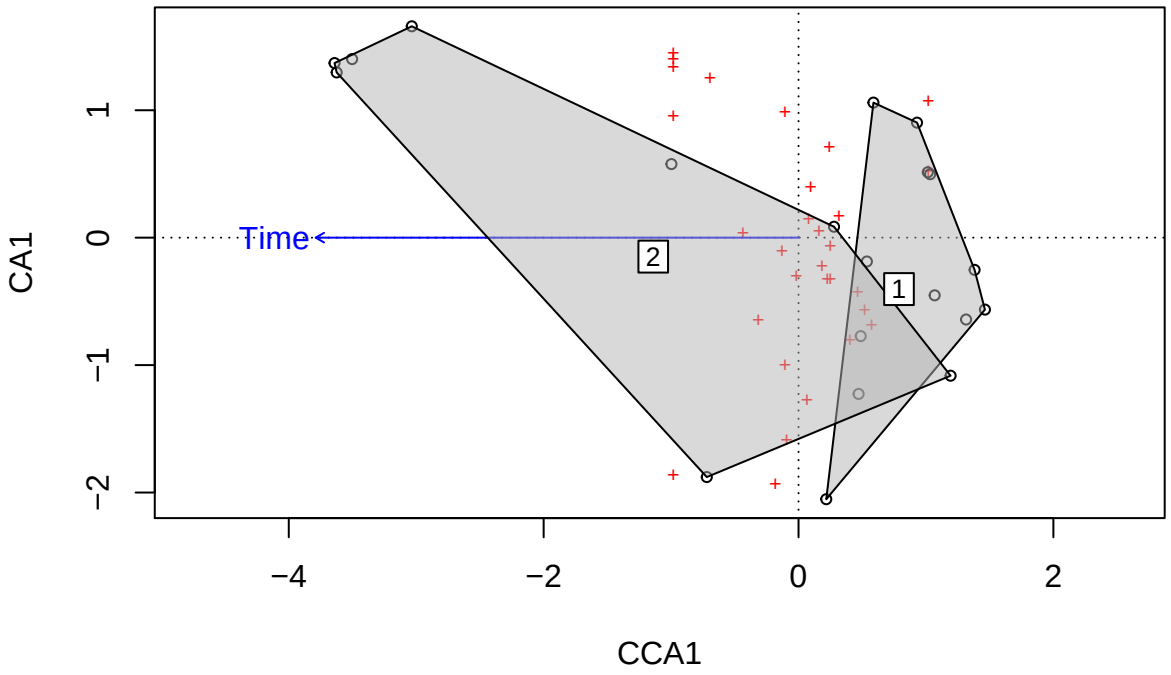
```
## Call: cca(formula = SpeTS ~ Time, data = EnvTS)
##
##               Inertia Proportion Rank
## Total         2.11526    1.00000
## Constrained   0.14240    0.06732    1
## Unconstrained 1.97287    0.93268   18
## Inertia is scaled Chi-square
##
## Eigenvalues for constrained axes:
##   CCA1
## 0.1424
##
## Eigenvalues for unconstrained axes:
##    CA1    CA2    CA3    CA4    CA5    CA6    CA7    CA8
## 0.4796 0.3721 0.2530 0.1736 0.1365 0.1063 0.0921 0.0783
## (Showing 8 of 18 unconstrained eigenvalues)
```

```
#permutation test
anova(time2.cca, by="term", perm=999)
```

```
## Permutation test for cca under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## Model: cca(formula = SpeTS ~ Time, data = EnvTS)
##          Df ChiSquare      F Pr(>F)
## Time      1    0.1424 1.2992  0.189
## Residual 18    1.9729
```

```
#ordiplot(time.cca)
#ordihull(time.cca, groups=treat, draw="polygon",col="grey70", label=T)
```

```
## plot ellipsoid hulls

treat <- EnvTS$Time
ordiplot(time2.cca)
ordihull(time2.cca,groups=treat,draw="polygon",col="grey70",label=T)
```



Question 25: A multivariate Before-After-Control- Impact (BACI) study

In this exercise we will continue to use the rearranged Dune Meadow data from exercise 24. A difference is that the plots are now divided into four groups: 1. Control plots before a treatment (i.e. not treated) 2. Control plots after a treatment (i.e. not treated) 3. Impact plots before treatment (i.e. not treated) 4. Impact plots after treatment (this is where the treatment is made)

The four groups are indicated in variable Treat in the environmental data set (EnvTS). The question you want to answer is if the treatment caused a change in species composition that is significantly different from the change in the control plots.

```
baci <- cca(SpeTS ~ Treat + Condition(Plot + Time), data=EnvTS)
baci
```

```
## Call: cca(formula = SpeTS ~ Treat + Condition(Plot + Time), data =
## EnvTS)
##
##                Inertia Proportion Rank
## Total           2.1153     1.0000
## Conditional     0.3685     0.1742    2
## Constrained     0.2443     0.1155    1
## Unconstrained   1.5024     0.7103   16
## Inertia is scaled Chi-square
##
```

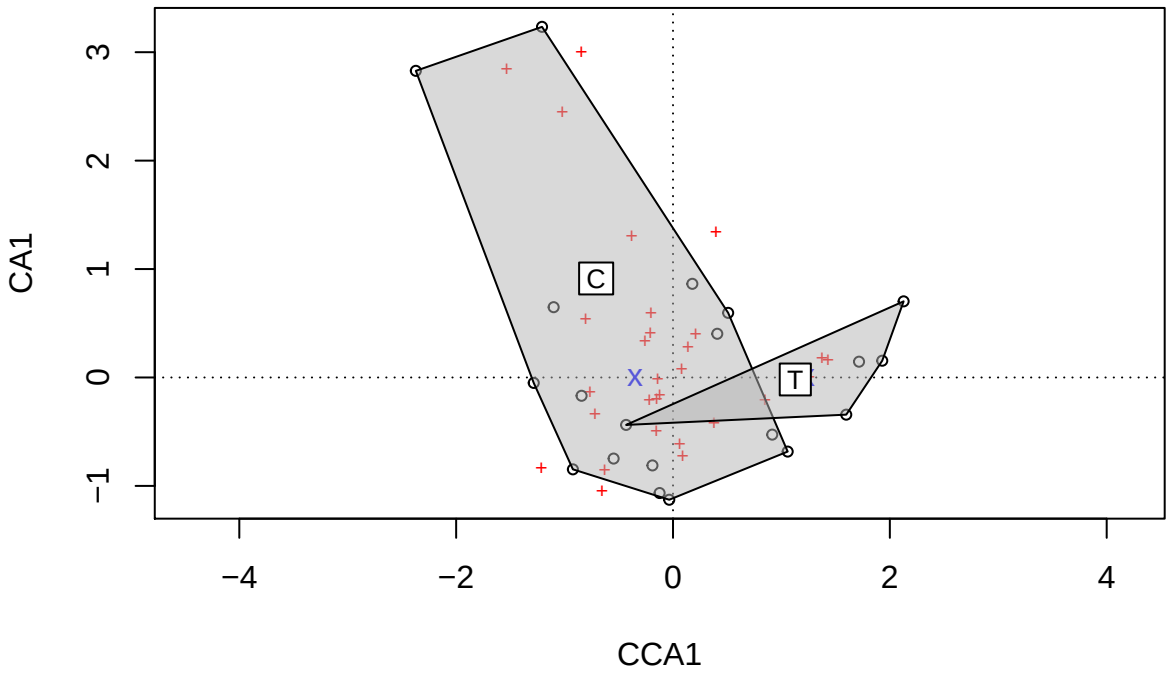
```
## Eigenvalues for constrained axes:
##    CCA1
## 0.24432
##
## Eigenvalues for unconstrained axes:
##    CA1    CA2    CA3    CA4    CA5    CA6    CA7    CA8    CA9   CA10
## 0.3925 0.2472 0.1740 0.1453 0.1182 0.0933 0.0829 0.0570 0.0532 0.0441
##   CA11   CA12   CA13   CA14   CA15   CA16
## 0.0324 0.0218 0.0141 0.0115 0.0082 0.0069
```

```
with(EnvTS, anova(baci, by="term", perm=500, strata=Treat))
```

```
## Permutation test for cca under reduced model
## Terms added sequentially (first to last)
## Blocks:  strata
## Permutation: free
## Number of permutations: 999
##
## Model: cca(formula = SpeTS ~ Treat + Condition(Plot + Time), data = EnvTS)
##          Df ChiSquare      F Pr(>F)
## Treat     1   0.24432 2.6018  0.007 **
## Residual 16   1.50245
## ---
## 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
```

```
## plot ellipsoid hulls

treat <- EnvTS$Treat
ordiplot(baci,type="points")
ordihull(baci,groups=treat,draw="polygon",col="grey70",label=T)
```



Question 26: SIMCA PLS STUFF

```
library(readxl)
Trend_Lakes_2015_PLS <- read_excel("Data/Data_for_PLS/Trend Lakes 2015_PLS.xls")
data.pls <- Trend_Lakes_2015_PLS
#View(data.pls)

install.packages("pls")
```

```
## Installing package into '/home/james/R/x86_64-pc-linux-gnu-library/3.6'
## (as 'lib' is unspecified)
```

```
library(pls)
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:vegan':
##
##     scores
```

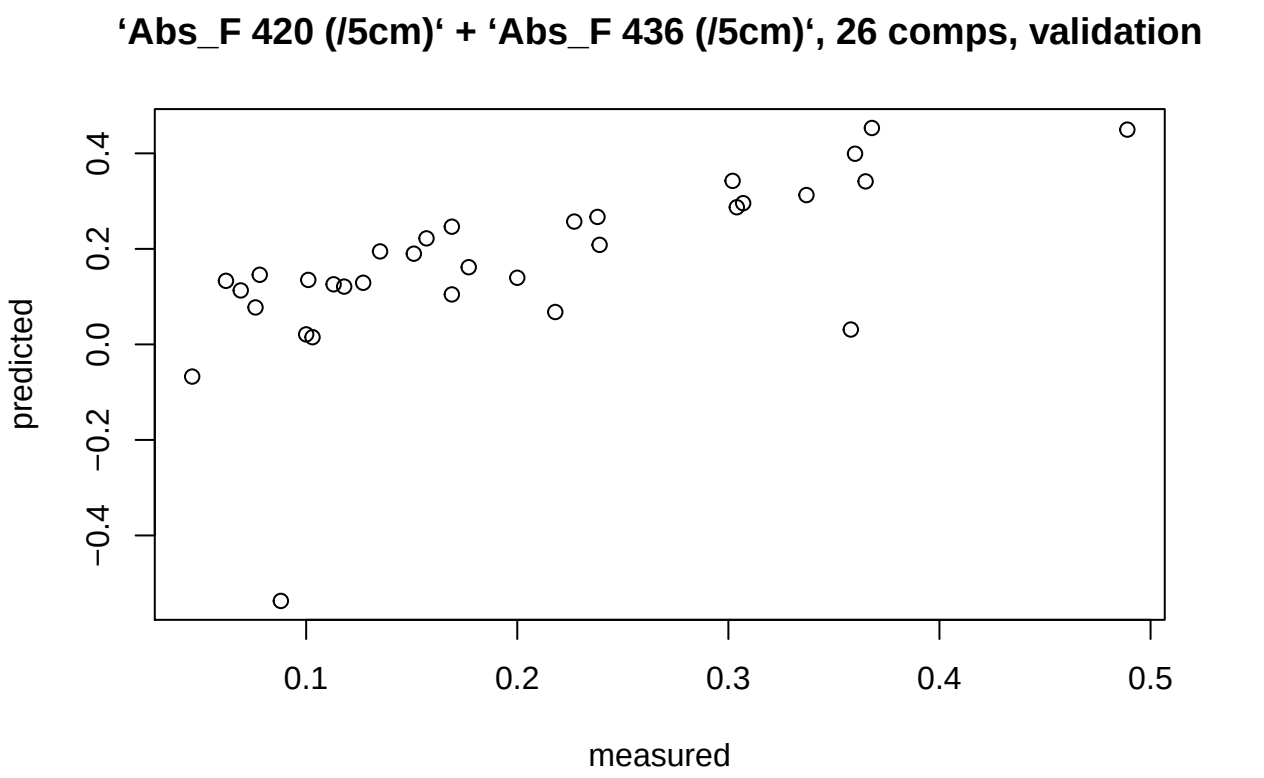
```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
pls.fit <- plsr(`Abs_F 420 (/5cm)` + `Abs_F 436 (/5cm)` ~ ., data = data.pls[c(8:35)], scale = TRUE, va
pcr.fit <- pcr(`Abs_F 436 (/5cm)` ~ ., data = data.pls[c(9:35)],scale = TRUE, validation = "CV")

summary(pls.fit)
```

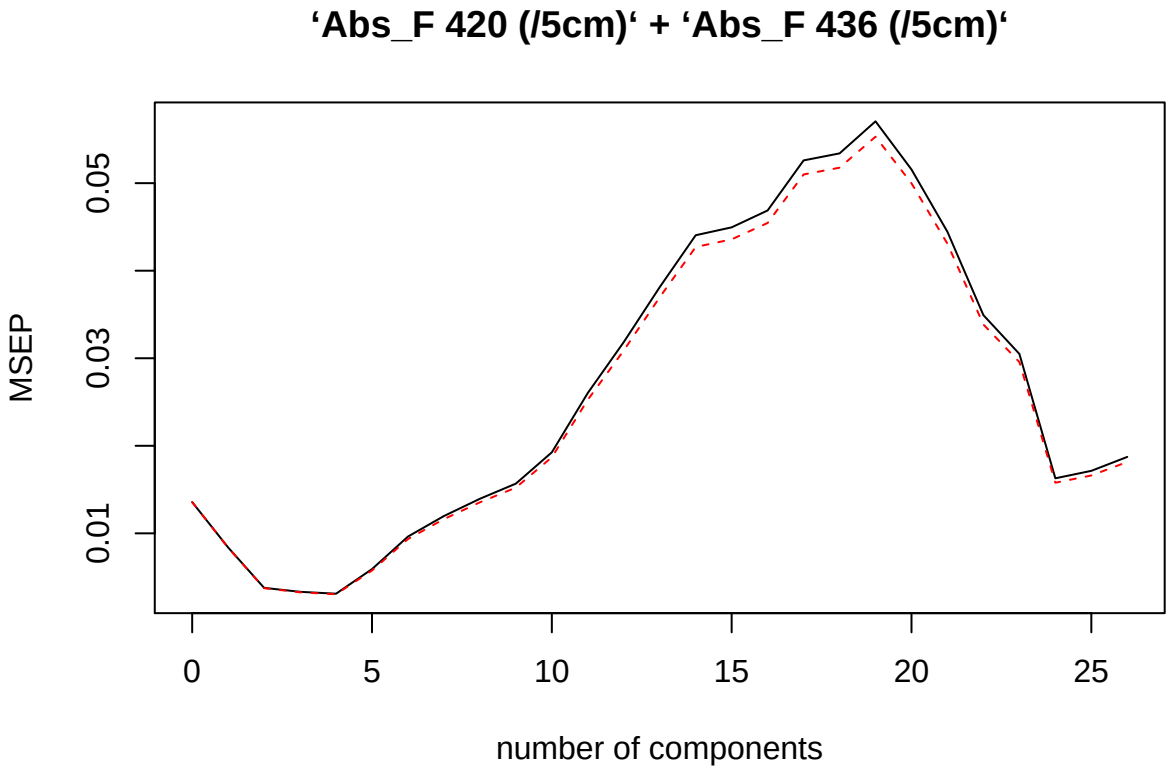
```
## Data:    X dimension: 32 26
##  Y dimension: 32 1
## Fit method: kernelpls
## Number of components considered: 26
##
## VALIDATION: RMSEP
## Cross-validated using 32 leave-one-out segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.1165  0.09158  0.06141  0.05761  0.05568  0.07691  0.09813
## adjCV       0.1165  0.09140  0.06112  0.05712  0.05516  0.07587  0.09668
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.1094   0.1181   0.1252    0.1388    0.1613    0.1785    0.1952
## adjCV   0.1078   0.1164   0.1234    0.1367    0.1589    0.1758    0.1923
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV       0.2099    0.2120    0.2165    0.2293    0.2311    0.2389
## adjCV    0.2067    0.2088    0.2132    0.2258    0.2275    0.2352
##        20 comps  21 comps  22 comps  23 comps  24 comps  25 comps
## CV       0.2271    0.2108    0.1869    0.1746    0.1276    0.1309
## adjCV    0.2236    0.2075    0.1840    0.1719    0.1256    0.1289
##        26 comps
## CV       0.1369
## adjCV    0.1347
##
## TRAINING: % variance explained
##                                          1 comps  2 comps  3 comps
## X                                          37.33    64.02    70.37
## `Abs_F 420 (/5cm)` + `Abs_F 436 (/5cm)`    54.31    84.34    92.19
```

```
##                                         4 comps  5 comps  6 comps
## X                                         76.49    78.33    80.82
## `Abs_F 420 (/5cm)` + `Abs_F 436 (/5cm)`   93.96    95.32    96.14
##                                         7 comps  8 comps  9 comps
## X                                         84.98    89.21    92.38
## `Abs_F 420 (/5cm)` + `Abs_F 436 (/5cm)`   96.56    96.89    97.06
##                                        10 comps  11 comps  12 comps
## X                                         94.84     96.56     98.13
## `Abs_F 420 (/5cm)` + `Abs_F 436 (/5cm)`   97.18     97.29     97.35
##                                        13 comps  14 comps  15 comps
## X                                         98.79     99.04     99.51
## `Abs_F 420 (/5cm)` + `Abs_F 436 (/5cm)`   97.49     97.76     97.82
##                                        16 comps  17 comps  18 comps
## X                                         99.70     99.77     99.84
## `Abs_F 420 (/5cm)` + `Abs_F 436 (/5cm)`   97.89     98.05     98.25
##                                        19 comps  20 comps  21 comps
## X                                         99.88     99.96     99.97
## `Abs_F 420 (/5cm)` + `Abs_F 436 (/5cm)`   98.52     98.56     98.64
##                                        22 comps  23 comps  24 comps
## X                                         99.99    100.00    100.00
## `Abs_F 420 (/5cm)` + `Abs_F 436 (/5cm)`   98.84     99.11     99.52
##                                        25 comps  26 comps
## X                                        100.00    100.00
## `Abs_F 420 (/5cm)` + `Abs_F 436 (/5cm)`   99.59     99.61
```

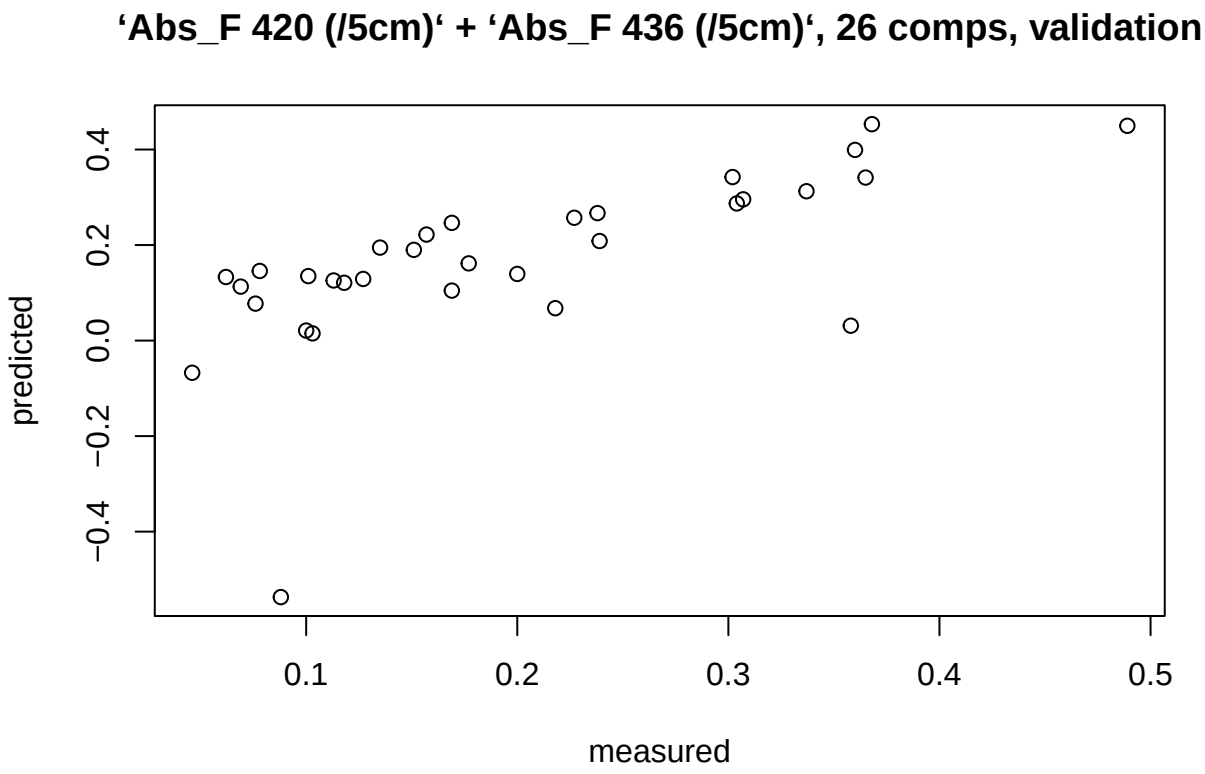
```
plot(pls.fit)
```



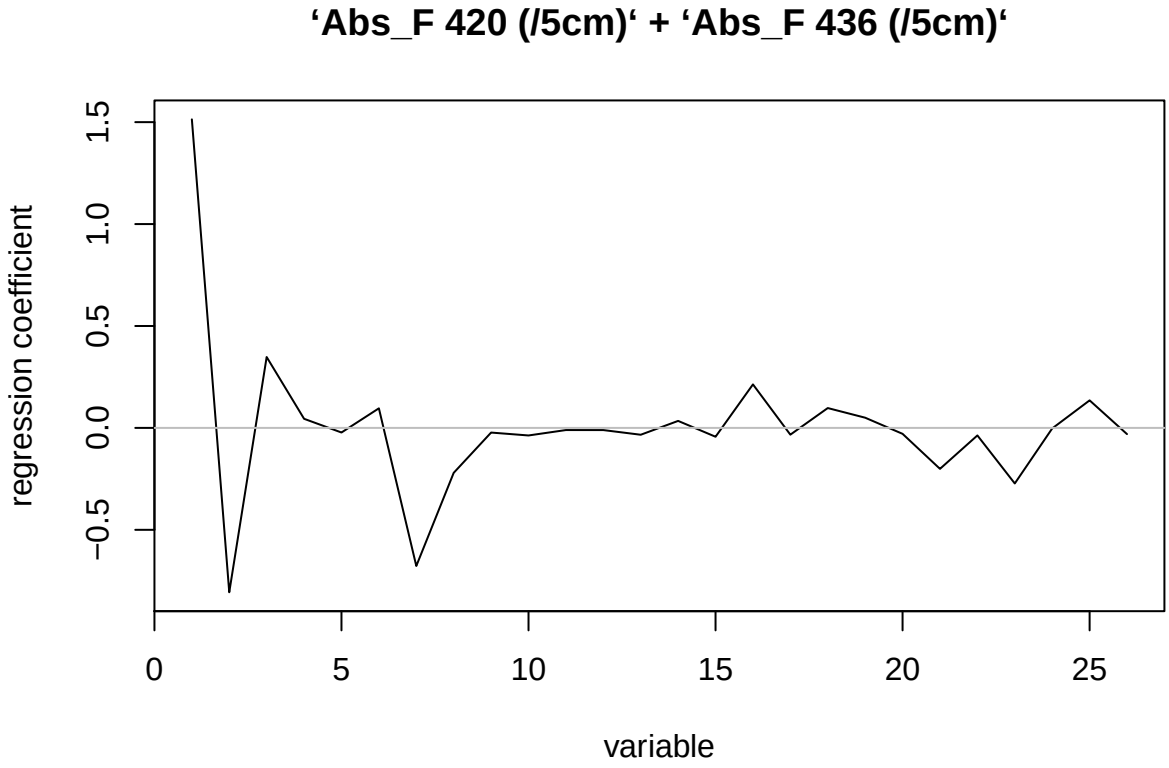

```
validationplot(pls.fit, val.type = "MSEP")
```




```
predplot(pls.fit)
```




```
coefplot(pls.fit)
```

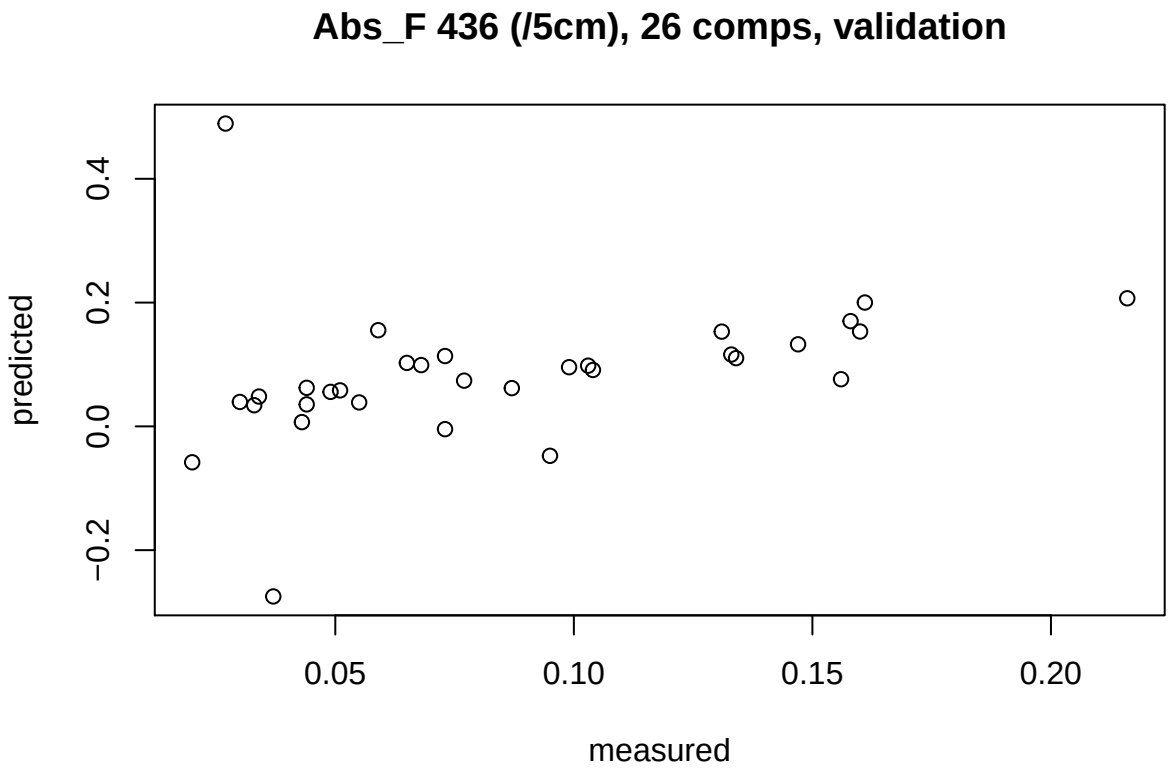



```
summary(pcr.fit)
```

```
## Data:    X dimension: 32 26
##  Y dimension: 32 1
## Fit method: svdpc
## Number of components considered: 26
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV         0.05127  0.05049  0.03288  0.02970  0.03370  0.03565  0.03111
## adjCV      0.05127  0.05037  0.03271  0.02858  0.03347  0.03542  0.03077
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV     0.02627  0.02446  0.02366   0.02398   0.02503   0.02506   0.04065
## adjCV  0.02542  0.02369  0.02310   0.02341   0.02437   0.02447   0.03915
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV      0.04186   0.05386   0.04593   0.05492   0.09122   0.10267
## adjCV   0.04009   0.05140   0.04392   0.05249   0.08698   0.09783
##        20 comps  21 comps  22 comps  23 comps  24 comps  25 comps
## CV      0.10225   0.08778   0.10441   0.09302   0.08768    0.1079
## adjCV   0.09742   0.08369   0.09952   0.08863   0.08356    0.1028
##        26 comps
## CV       0.1075
## adjCV    0.1023
##
## TRAINING: % variance explained
##                   1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## X                  51.658    64.82    73.16    80.57    85.06    88.78
## Abs_F 436 (/5cm)    7.039    63.21    75.35    75.35    75.46    84.13
##                   7 comps  8 comps  9 comps  10 comps  11 comps  12 comps
```

```
## X                   92.01    94.63    96.44     97.59     98.34     98.92
## Abs_F 436 (/5cm)    91.50    92.05    92.27     92.93     93.23     93.35
##                   13 comps  14 comps  15 comps  16 comps  17 comps
## X                    99.33     99.53     99.68     99.80     99.88
## Abs_F 436 (/5cm)     94.02     96.15     96.70     96.75     96.75
##                   18 comps  19 comps  20 comps  21 comps  22 comps
## X                    99.93     99.96     99.98     99.99    100.00
## Abs_F 436 (/5cm)     96.85     97.54     97.55     97.58     97.62
##                   23 comps  24 comps  25 comps  26 comps
## X                   100.00    100.00    100.00     100.0
## Abs_F 436 (/5cm)     98.31     98.33     98.38      99.6
```

```
plot(pcr.fit)
```



Question 27 : Classification

Start by testing the non-hierarchical K-means clustering, using Multivar / K-Means. Note that it is recommended that K-means need at least 100 observations (500 according to some sources) to be reliable! Let us ignore the sample size issue for the moment, and ask for 4 clusters (for later comparison with the four management types). Test different hierarchical agglomeration algorithms and similarity indices. Use at least the Euclidian and the Bray-Curtis similarity measures for the hierarchical clustering technique. Try also the Ward's method! Repeat the analysis above, but copy the Management types to a new grouping column.

```
# install.packages("dendextend")
library(dendextend)
```

```
## Registered S3 method overwritten by 'dendextend':
##   method     from
##   rev.hclust vegan
```

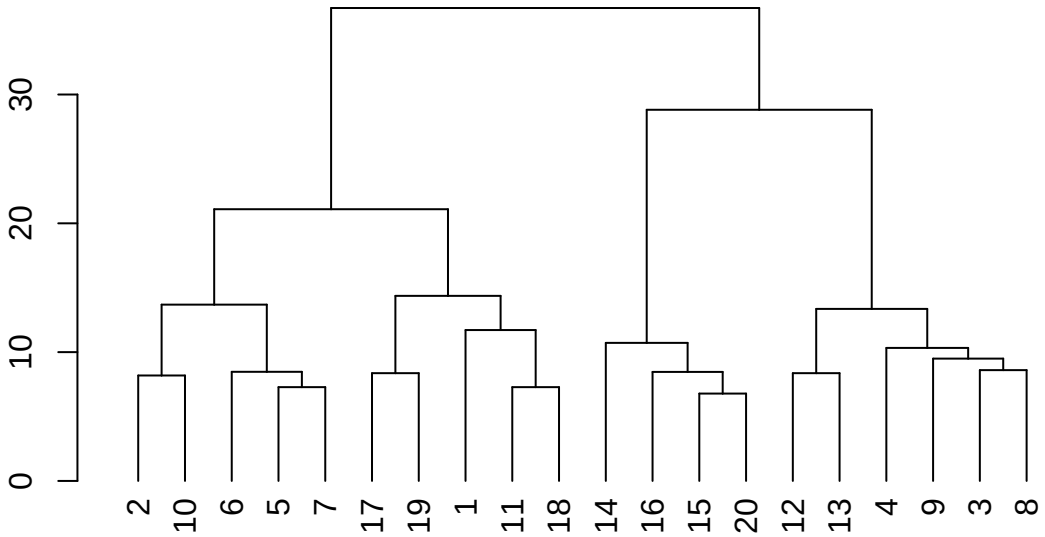
```
##
```

```
## ---------------------
## Welcome to dendextend version 1.12.0
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------

##
## Attaching package: 'dendextend'

## The following object is masked from 'package:permute':
##
##     shuffle

## The following object is masked from 'package:stats':
##
##     cutree
```

```
dune.dist <- vegdist(dune, method = "bray")

# K-Means Cluster Analysis
fit <- kmeans(dune.dist, 4, nstart = 25) # 4 cluster solution
#As the final result of k-means clustering result is sensitive to the random starting assignments, we s
table(fit$cluster, dune.env.original$Management)
```

```
##
##     BF HF NM SF
##   1  0  0  2  0
##   2  3  3  1  1
##   3  0  0  3  1
##   4  0  2  0  4
```

```
# append cluster assignment
dune.clusters <- data.frame(dune, fit$cluster)
dune.clusters <- data.frame(dune.clusters, dune.env.original$Management)

#try some approaches to hierarchical clustering
dend <- dune %>% # data
        dist(method = "euclidean") %>% # calculate a distance matrix,
        hclust(method = "ward.D") %>% # Hierarchical clustering
        as.dendrogram # Turn the object into a dendrogram.
plot(dend)
```



Question 28: ANOSIM

ANOSIM (ANalysis Of Similarities) is a non-parametric test of significant difference between two or more groups, based on any distance measure. In this case, use the clusters from the K-means exercise. Change to Bray-Curtis similarity index.

What does the result tell you? What other types of predefined clusters could you use?

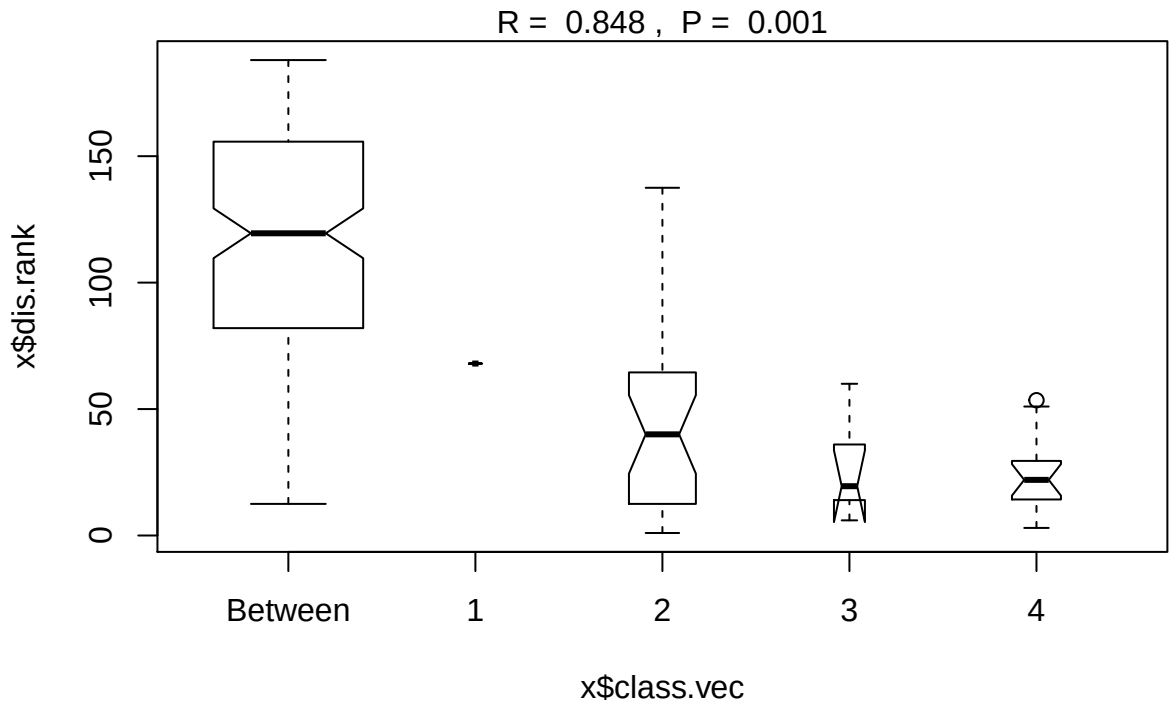
```
dune.dist <- vegdist(dune, method = "bray")
dune.ano <- anosim(dune.dist, fit$cluster)
summary(dune.ano)
```

```
##
## Call:
## anosim(x = dune.dist, grouping = fit$cluster)
## Dissimilarity: bray
##
## ANOSIM statistic R: 0.8483
##       Significance: 0.001
##
## Permutation: free
## Number of permutations: 999
##
## Upper quantiles of permutations (null model):
##   90%   95% 97.5%   99%
## 0.137 0.183 0.214 0.267
##
## Dissimilarity ranks between and within classes:
##           0%   25%   50%     75%  100%   N
## Between 12.5 82.50 119.5 155.375 188.0 140
## 1       68.0 68.00  68.0  68.000  68.0   1
## 2        1.0 13.75  40.0  63.750 137.5  28
## 3        6.0 15.25  19.5  32.000  60.0   6
## 4        3.0 14.25  22.0  29.500  53.5  15
```

```
plot(dune.ano)
```

```
## Warning in bxp(list(stats = structure(c(12.5, 82, 119.5, 155.75, 188, 68, :
```

```
## some notches went outside hinges ('box'): maybe set notch=FALSE
```

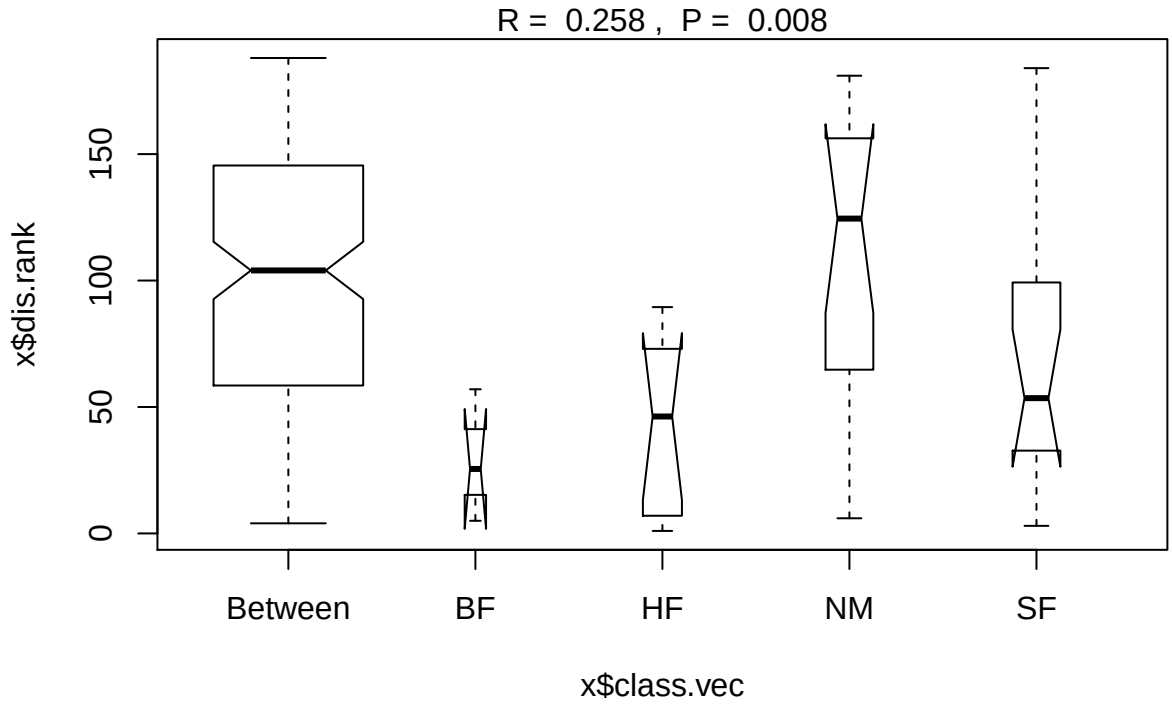



```
dune.ano2 <- anosim(dune.dist, dune.env.original$Management)
summary(dune.ano2)
```

```
##
## Call:
## anosim(x = dune.dist, grouping = dune.env.original$Management)
## Dissimilarity: bray
##
## ANOSIM statistic R: 0.2579
##       Significance: 0.008
##
## Permutation: free
## Number of permutations: 999
##
## Upper quantiles of permutations (null model):
##   90%   95% 97.5%   99%
## 0.122 0.163 0.206 0.250
##
## Dissimilarity ranks between and within classes:
##         0%   25%    50%     75%  100%   N
## Between  4 58.50 104.00 145.500 188.0 147
## BF       5 15.25  25.50  41.250  57.0   3
## HF       1  7.25  46.25  68.125  89.5  10
## NM       6 64.75 124.50 156.250 181.0  15
## SF       3 32.75  53.50  99.250 184.0  15
```

```
plot(dune.ano2)
```

```
## Warning in bxp(list(stats = structure(c(4, 58.5, 104, 145.5, 188, 5,
## 15.25, : some notches went outside hinges ('box'): maybe set notch=FALSE
```



Question 28: SIMPER

After a significant ANOSIM, you may want to know which species are primarily responsible for the observed difference between clusters. SIMPER (Similarity Percentage) will do this for you. The test does not come with significance testing. In the output table, taxa are sorted in descending order of contribution to group difference. The last columns show the mean abundance in the groups.

```
sim <- simper(dune, dune.env.original$Management)
summary(sim)
```

```
##
## Contrast: SF_BF
##
##           average       sd  ratio    ava    avb  cumsum
## Agrostol 0.061374 0.034193 1.7949 4.6667 0.0000 0.09824
## Alopgeni 0.052667 0.036476 1.4439 4.3333 0.6667 0.18255
## Lolipere 0.048116 0.039445 1.2198 3.0000 6.0000 0.25957
## Trifrepe 0.046297 0.025525 1.8138 1.3333 4.6667 0.33368
## Poatriv  0.046020 0.033801 1.3615 4.6667 3.6667 0.40734
## Scorautu 0.043697 0.024922 1.7534 1.3333 4.3333 0.47729
## Bromhord 0.033677 0.025860 1.3023 0.5000 2.6667 0.53120
## Achimill 0.030152 0.020821 1.4482 0.1667 2.3333 0.57947
## Planlanc 0.028585 0.021549 1.3265 0.0000 2.0000 0.62522
## Elymrepe 0.028074 0.029778 0.9428 2.0000 1.3333 0.67016
## Bracruta 0.025501 0.023902 1.0669 2.0000 2.0000 0.71098
## Poaprat  0.025129 0.023967 1.0485 2.5000 4.0000 0.75121
## Sagiproc 0.024326 0.022149 1.0983 1.8333 0.6667 0.79014
## Bellpere 0.019859 0.017088 1.1622 0.6667 1.6667 0.82193
## Eleopalu 0.018611 0.042958 0.4333 1.3333 0.0000 0.85172
## Anthodor 0.017543 0.025804 0.6798 0.0000 1.3333 0.87981
## Juncbufo 0.016031 0.023708 0.6762 1.1667 0.0000 0.90547
## Vicilath 0.014671 0.013306 1.1026 0.0000 1.0000 0.92895
```

```
## Hyporadi 0.010286 0.015198 0.6768 0.0000 0.6667 0.94542
## Ranuflam 0.009306 0.013595 0.6845 0.6667 0.0000 0.96031
## Juncarti 0.006979 0.016109 0.4333 0.5000 0.0000 0.97148
## Callcusp 0.006979 0.016109 0.4333 0.5000 0.0000 0.98266
## Rumeacet 0.004526 0.010444 0.4333 0.3333 0.0000 0.98990
## Cirsarve 0.003983 0.009185 0.4336 0.3333 0.0000 0.99628
## Chenalbu 0.002326 0.005370 0.4333 0.1667 0.0000 1.00000
## Airaprae 0.000000 0.000000    NaN 0.0000 0.0000 1.00000
## Comapalu 0.000000 0.000000    NaN 0.0000 0.0000 1.00000
## Empenigr 0.000000 0.000000    NaN 0.0000 0.0000 1.00000
## Salirepe 0.000000 0.000000    NaN 0.0000 0.0000 1.00000
## Trifprat 0.000000 0.000000    NaN 0.0000 0.0000 1.00000
##
## Contrast: SF_HF
##
##           average       sd  ratio    ava avb  cumsum
## Agrostol 0.047380 0.031273 1.5151 4.6667 1.4 0.08351
## Alopgeni 0.046433 0.032897 1.4115 4.3333 1.6 0.16535
## Lolipere 0.041986 0.027007 1.5546 3.0000 4.0 0.23935
## Planlanc 0.039198 0.033208 1.1804 0.0000 3.0 0.30844
## Rumeacet 0.038992 0.027369 1.4247 0.3333 3.2 0.37716
## Elymrepe 0.031877 0.029550 1.0787 2.0000 2.0 0.43334
## Poatriv  0.028466 0.021522 1.3227 4.6667 4.8 0.48352
## Bracruta 0.025261 0.021044 1.2004 2.0000 2.8 0.52804
## Eleopalu 0.024974 0.038877 0.6424 1.3333 0.8 0.57206
## Poaprat  0.023932 0.019180 1.2478 2.5000 3.4 0.61424
## Anthodor 0.023409 0.021430 1.0923 0.0000 1.8 0.65550
## Sagiproc 0.023144 0.020479 1.1301 1.8333 0.8 0.69629
## Trifprat 0.023080 0.023432 0.9850 0.0000 1.8 0.73697
## Juncarti 0.022850 0.025677 0.8899 0.5000 1.6 0.77724
## Trifrepe 0.022383 0.019487 1.1486 1.3333 2.8 0.81669
## Juncbufo 0.021643 0.022237 0.9733 1.1667 1.2 0.85484
## Scorautu 0.020509 0.016422 1.2489 1.3333 2.8 0.89099
## Achimill 0.015183 0.011393 1.3326 0.1667 1.2 0.91775
## Bromhord 0.013375 0.014504 0.9222 0.5000 0.8 0.94132
## Ranuflam 0.010661 0.013387 0.7964 0.6667 0.4 0.96011
## Bellpere 0.009991 0.012571 0.7948 0.6667 0.4 0.97772
## Callcusp 0.006623 0.015076 0.4393 0.5000 0.0 0.98939
## Cirsarve 0.003809 0.008669 0.4394 0.3333 0.0 0.99611
## Chenalbu 0.002208 0.005025 0.4393 0.1667 0.0 1.00000
## Airaprae 0.000000 0.000000    NaN 0.0000 0.0 1.00000
## Comapalu 0.000000 0.000000    NaN 0.0000 0.0 1.00000
## Empenigr 0.000000 0.000000    NaN 0.0000 0.0 1.00000
## Hyporadi 0.000000 0.000000    NaN 0.0000 0.0 1.00000
## Salirepe 0.000000 0.000000    NaN 0.0000 0.0 1.00000
## Vicilath 0.000000 0.000000    NaN 0.0000 0.0 1.00000
##
## Contrast: SF_NM
##
##           average       sd  ratio    ava    avb cumsum
## Poatriv  0.078284 0.040947 1.9118 4.6667 0.0000 0.1014
## Alopgeni 0.071219 0.046958 1.5167 4.3333 0.0000 0.1936
## Agrostol 0.056508 0.044176 1.2792 4.6667 2.1667 0.2667
## Lolipere 0.054851 0.059914 0.9155 3.0000 0.3333 0.3378
```

```
## Eleopalu 0.048027 0.047168 1.0182 1.3333 2.1667 0.3999
## Poaprat  0.040724 0.031790 1.2810 2.5000 0.6667 0.4527
## Bracruta 0.040008 0.034398 1.1631 2.0000 2.8333 0.5045
## Elymrepe 0.035598 0.038515 0.9243 2.0000 0.0000 0.5506
## Scorautu 0.032475 0.034813 0.9328 1.3333 3.1667 0.5926
## Trifrepe 0.030430 0.031634 0.9619 1.3333 1.8333 0.6320
## Sagiproc 0.030304 0.030477 0.9943 1.8333 0.5000 0.6712
## Salirepe 0.029275 0.032014 0.9144 0.0000 1.8333 0.7092
## Anthodor 0.024541 0.036694 0.6688 0.0000 1.3333 0.7409
## Callcusp 0.022763 0.029443 0.7731 0.5000 1.1667 0.7704
## Ranuflam 0.022566 0.022819 0.9889 0.6667 1.3333 0.7996
## Juncarti 0.022543 0.028598 0.7883 0.5000 1.1667 0.8288
## Hyporadi 0.020108 0.031291 0.6426 0.0000 1.1667 0.8548
## Juncbufo 0.019860 0.029034 0.6840 1.1667 0.0000 0.8806
## Planlanc 0.015420 0.022769 0.6772 0.0000 0.8333 0.9005
## Airaprae 0.014883 0.021881 0.6802 0.0000 0.8333 0.9198
## Bellpere 0.012317 0.015921 0.7737 0.6667 0.3333 0.9357
## Comapalu 0.011883 0.017407 0.6826 0.0000 0.6667 0.9511
## Achimill 0.009294 0.014931 0.6224 0.1667 0.3333 0.9632
## Bromhord 0.007172 0.016333 0.4391 0.5000 0.0000 0.9724
## Rumeacet 0.005590 0.012751 0.4384 0.3333 0.0000 0.9797
## Empenigr 0.005225 0.012001 0.4354 0.0000 0.3333 0.9864
## Cirsarve 0.004782 0.010889 0.4391 0.3333 0.0000 0.9926
## Chenalbu 0.002893 0.006602 0.4382 0.1667 0.0000 0.9964
## Vicilath 0.002792 0.006425 0.4345 0.0000 0.1667 1.0000
## Trifprat 0.000000 0.000000    NaN 0.0000 0.0000 1.0000
##
## Contrast: BF_HF
##
##           average      sd  ratio    ava avb  cumsum
## Rumeacet 0.038666 0.02606 1.4838 0.0000 3.2 0.08163
## Poatriv  0.033301 0.02579 1.2911 3.6667 4.8 0.15194
## Planlanc 0.031852 0.01830 1.7401 2.0000 3.0 0.21918
## Bromhord 0.028651 0.01799 1.5926 2.6667 0.8 0.27967
## Lolipere 0.028431 0.02215 1.2834 6.0000 4.0 0.33970
## Elymrepe 0.027822 0.02959 0.9404 1.3333 2.0 0.39843
## Trifrepe 0.025838 0.01656 1.5603 4.6667 2.8 0.45298
## Anthodor 0.023582 0.02042 1.1547 1.3333 1.8 0.50277
## Achimill 0.023426 0.01474 1.5893 2.3333 1.2 0.55223
## Bracruta 0.022733 0.01802 1.2617 2.0000 2.8 0.60022
## Alopgeni 0.021610 0.02308 0.9363 0.6667 1.6 0.64584
## Trifprat 0.021514 0.02207 0.9747 0.0000 1.8 0.69126
## Juncarti 0.020084 0.02555 0.7860 0.0000 1.6 0.73367
## Scorautu 0.019318 0.01356 1.4241 4.3333 2.8 0.77445
## Bellpere 0.018290 0.01486 1.2305 1.6667 0.4 0.81306
## Agrostol 0.017605 0.02284 0.7708 0.0000 1.4 0.85023
## Juncbufo 0.015000 0.02066 0.7260 0.0000 1.2 0.88190
## Vicilath 0.012848 0.01140 1.1274 1.0000 0.0 0.90903
## Sagiproc 0.011685 0.01297 0.9008 0.6667 0.8 0.93369
## Eleopalu 0.010169 0.02111 0.4817 0.0000 0.8 0.95516
## Hyporadi 0.008950 0.01312 0.6824 0.6667 0.0 0.97406
## Poaprat  0.007203 0.01010 0.7133 4.0000 3.4 0.98927
## Ranuflam 0.005084 0.01055 0.4817 0.0000 0.4 1.00000
## Airaprae 0.000000 0.00000    NaN 0.0000 0.0 1.00000
```

```
## Chenalbu 0.000000 0.00000    NaN 0.0000 0.0 1.00000
## Cirsarve 0.000000 0.00000    NaN 0.0000 0.0 1.00000
## Comapalu 0.000000 0.00000    NaN 0.0000 0.0 1.00000
## Empenigr 0.000000 0.00000    NaN 0.0000 0.0 1.00000
## Salirepe 0.000000 0.00000    NaN 0.0000 0.0 1.00000
## Callcusp 0.000000 0.00000    NaN 0.0000 0.0 1.00000
##
## Contrast: BF_NM
##
##           average      sd  ratio    ava    avb cumsum
## Lolipere 0.090681 0.02644 3.4292 6.0000 0.3333 0.1243
## Poatriv  0.054684 0.04465 1.2248 3.6667 0.0000 0.1992
## Poaprat  0.052511 0.01813 2.8966 4.0000 0.6667 0.2712
## Trifrepe 0.051287 0.02756 1.8611 4.6667 1.8333 0.3415
## Bromhord 0.039689 0.02920 1.3590 2.6667 0.0000 0.3959
## Bracruta 0.035723 0.02869 1.2452 2.0000 2.8333 0.4448
## Eleopalu 0.033759 0.03573 0.9449 0.0000 2.1667 0.4911
## Agrostol 0.033446 0.03473 0.9630 0.0000 2.1667 0.5369
## Achimill 0.033190 0.02338 1.4198 2.3333 0.3333 0.5824
## Scorautu 0.031356 0.02026 1.5480 4.3333 3.1667 0.6254
## Anthodor 0.028060 0.03295 0.8517 1.3333 1.3333 0.6638
## Planlanc 0.027319 0.02193 1.2458 2.0000 0.8333 0.7013
## Salirepe 0.026770 0.02927 0.9145 0.0000 1.8333 0.7379
## Bellpere 0.023529 0.01909 1.2322 1.6667 0.3333 0.7702
## Hyporadi 0.021721 0.02450 0.8864 0.6667 1.1667 0.8000
## Ranuflam 0.020314 0.02275 0.8928 0.0000 1.3333 0.8278
## Elymrepe 0.019993 0.02926 0.6833 1.3333 0.0000 0.8552
## Callcusp 0.017833 0.02681 0.6653 0.0000 1.1667 0.8796
## Juncarti 0.017694 0.02600 0.6806 0.0000 1.1667 0.9039
## Vicilath 0.015773 0.01447 1.0902 1.0000 0.1667 0.9255
## Sagiproc 0.015432 0.01857 0.8310 0.6667 0.5000 0.9466
## Airaprae 0.013410 0.01969 0.6809 0.0000 0.8333 0.9650
## Comapalu 0.010739 0.01571 0.6835 0.0000 0.6667 0.9797
## Alopgeni 0.009997 0.01463 0.6833 0.6667 0.0000 0.9934
## Empenigr 0.004787 0.01105 0.4332 0.0000 0.3333 1.0000
## Chenalbu 0.000000 0.00000    NaN 0.0000 0.0000 1.0000
## Cirsarve 0.000000 0.00000    NaN 0.0000 0.0000 1.0000
## Juncbufo 0.000000 0.00000    NaN 0.0000 0.0000 1.0000
## Rumeacet 0.000000 0.00000    NaN 0.0000 0.0000 1.0000
## Trifprat 0.000000 0.00000    NaN 0.0000 0.0000 1.0000
##
## Contrast: HF_NM
##
##           average       sd  ratio ava    avb  cumsum
## Poatriv  0.071553 0.013681 5.2302 4.8 0.0000 0.09913
## Lolipere 0.054533 0.029625 1.8408 4.0 0.3333 0.17468
## Rumeacet 0.046546 0.030806 1.5109 3.2 0.0000 0.23917
## Poaprat  0.041750 0.018852 2.2146 3.4 0.6667 0.29701
## Planlanc 0.041633 0.029560 1.4084 3.0 0.8333 0.35469
## Bracruta 0.035340 0.020104 1.7579 2.8 2.8333 0.40365
## Eleopalu 0.032043 0.032315 0.9916 0.8 2.1667 0.44805
## Agrostol 0.031915 0.028887 1.1048 1.4 2.1667 0.49227
## Trifrepe 0.030372 0.022871 1.3280 2.8 1.8333 0.53434
## Elymrepe 0.029811 0.038676 0.7708 2.0 0.0000 0.57565
```

```
## Anthodor 0.028717 0.024799 1.1580 1.8 1.3333 0.61543
## Juncarti 0.027414 0.028537 0.9607 1.6 1.1667 0.65341
## Trifprat 0.025843 0.025972 0.9951 1.8 0.0000 0.68922
## Salirepe 0.025340 0.027291 0.9285 0.0 1.8333 0.72432
## Alopgeni 0.024459 0.032399 0.7549 1.6 0.0000 0.75821
## Scorautu 0.020703 0.014125 1.4658 2.8 3.1667 0.78689
## Ranuflam 0.019285 0.019939 0.9672 0.4 1.3333 0.81361
## Juncbufo 0.018181 0.024648 0.7376 1.2 0.0000 0.83880
## Hyporadi 0.017141 0.026548 0.6457 0.0 1.1667 0.86255
## Callcusp 0.016833 0.024901 0.6760 0.0 1.1667 0.88587
## Achimill 0.016555 0.014900 1.1111 1.2 0.3333 0.90881
## Sagiproc 0.015282 0.016535 0.9243 0.8 0.5000 0.92998
## Airaprae 0.012605 0.018243 0.6910 0.0 0.8333 0.94744
## Bromhord 0.012094 0.015169 0.7973 0.8 0.0000 0.96420
## Comapalu 0.010105 0.014556 0.6942 0.0 0.6667 0.97820
## Bellpere 0.008801 0.013732 0.6409 0.4 0.3333 0.99039
## Empenigr 0.004536 0.010325 0.4393 0.0 0.3333 0.99668
## Vicilath 0.002399 0.005461 0.4393 0.0 0.1667 1.00000
## Chenalbu 0.000000 0.000000    NaN 0.0 0.0000 1.00000
## Cirsarve 0.000000 0.000000    NaN 0.0 0.0000 1.00000
## Permutation: free
## Number of permutations: 0
```

```
#alternate
#(sim <- with(dune.env.original, simper(dune, Management)))
#summary(sim)
```