flowCHIC - Analyze flow cytometric data of complex microbial communities based on histogram images

Schumann, J., Koch, C., Fetzer, I., Müller, S.
Helmholtz Centre for Environmental Research - UFZ
Department of Environmental Microbiology

October 17, 2016

Abstract

CHIC means cytometric histogram image comparison and is a tool for evaluation of cytometric datasets nearly automatically and very fast. CHIC neither requires gate setting procedures nor relies on quantification and binning of event distributions within histograms. The evaluation of histograms using CHIC is based on the main idea that histograms are converted into images thus working person independent and with low computational demand. Although cell specific information is given up in this way it allows, on the other hand, huge datasets to be evaluated within very short time frames. It is, therefore, a tool to visualize cell population or community dynamics and to suggest causes of dynamic variations. The sensitivity of this technique was found to discriminate down to 0.5% internal variation. On the first hand it was developed for evaluation of cytometric microbial community data but it can also be used for any other cytometric dataset.

1 Introduction

The background of the CHIC tool is published in [**?**]. flowCHIC contains four parts. The first part provides information on cell distributions and their respective abundances in histograms, the second part transforms vector based histograms into PNG images, the third part calculates the values needed for comparing these images and the forth part evaluates the differences based on the calculated values. The first part of flowCHIC directly uses cytometric raw data (usually FCS files) that can be obtained from any cytometric analysis. As the later evaluation occurs on grey value comparison in a starting step arbitrary color keys of cytometric histograms are converted into equally spaced grey graduations. The second part creates images with instrumental noise, bead signals, or axis denominations eliminated by defining a rectangular gate. The third part calculates the overlapping area as well as the sum of all pixel intensities based on XOR operation for each combination of every image. The forth part performs nonmetric multidimensional scaling (NMDS) giving information on how similar/dissimilar certain samples are or how the structure of populations or communities is changing over time. Examples are discussed in [**?**, **?**] and also given below. To find out the driving forces of population variation the rationale of abiotic parameters such as pH values, temperature, or concentrations of chemicals can be evaluated. By using gate information obtained from flowCyBar [**?**] influences coming from certain subpopulations or subcommunities can also be detected. It needs to be stated that flowCHIC requires datasets that are highly calibrated by using e.g. fuorescent beads. Every shift in the adjustment of a cytometer will be discovered using flowCHIC and will have an influence on the outcome. It is also important that for every sample the same number of cells is measured. flowCHIC can be used for any cytometric dataset. The strength of the tool lies in its easy application, especially if big datasets needs to be evaluated and its nearly automatic, person-independent application. The tool provides trends in community or population dynamics or informs on stability of cell systems.

2 Overview

Creation of images from flow cytometric raw data The packages flowCore and ggplot2 are used for creating histogram images of flow cytometric raw data.

Subsets of the histogram images A subset of each histogram is created to define the area containing only relevant information and exclude the noise.

Image analysis with flowViz and EBImage The packages flowViz and EBImage are used for analysing the subset histogram images.

Similarity calculation The similarities of the images are calculated using nonmetric multidimensional scaling (NMDS).

3 Details

3.1 Creation of images from flow cytometric raw data

flowCore and ggplot2 are used for creating images of flow cytometric raw data. After the folder containing the FCS files is set as working directory a list with all filenames is saved. The channels of interest are selected and the histogram images are created and saved as PNG image files in a new subfolder called "chic_images". Note that only two channels are used for this approach. For the next step you need to look at the created images.

3.2 Subsets of the histogram images

Typically, cytometric measurements of microbial communities contain instrumental noise for instrumental adjustment. For the comparism of the samples the noise is not relevant. Ensuing from the created histogram images a rectangular region is defined to cut out the noise from each image. It is very important to set the boundaries correct which means that the selected region should only include relevant information. The boundaries are only set once so you should check if they are suitable for every image. The resulting subset images are saved as PNG image files in a new subfolder called "chic_subset". To check the boundaries look at the images located in this subfolder.

3.3 Image analysis with flowViz and EBImage

At first, a list with all filenames of the subset histogram images is saved. The overlap and XOR images are calculated, the areas and intensities are returned and can be saved as text files. The overlap area describes the number of informative pixels of two images. The XOR operation highlights the differences between two images. This procedure is described more detailed in [**?**].

3.4 Similarity calculation

The similarities found in the processed images are calculated using nonmetric multidimensional scaling. Therefore, a dissimilarity matrix is calculated based on the intensities of the XOR image and the overlap area fo every image pair. The results are shown as an NMDS plot. In addition, a cluster analysis can be performed to reveal samples with high similarities. For additional information see [**?**].

4 Example data

For the next sections the FCS files that are included to the package and two datasets from the CHIC paper [**?**] will serve as examples. These datasets are published on the FlowRepository online database. They can be downloaded from https://flowrepository.org/id/FR-FCM-ZZ42 and from http://flowrepository.org/id/FR-FCM-ZZ43. See the CHIC paper for more information about these datasets. At the bottom left corner you can download the FCS files of the corresponding dataset. For using the datasets save all the FCS files of one dataset in one folder. The FCS files that are included to the package are selected from the first dataset.

5 Methods

This section gives an overview of the methods that are included in the package. For all parts the FCS files that are included to the package are used for demonstration. In the last part the results are also shown for the two downloadable datasets (see above).

5.1 Creating histogram images

First of all the package has to be loaded in R.

```
> # Load package
> library(flowCHIC)
```

For creating the histogram images the folder containing the FCS files should be set as working directory and a list with the filenames of all FCS files should be saved. In this example it is also shown how to create this list if you want to use the FCS files that are attached to this package. These files are also used for the next parts. Then the X parameter (channel 1) and Y parameter (channel 2) used for creating the histogram images have to be set. You can check the names of the parameters/channels of your FCS files in the following way:

```
> # Set the working directory to the path where your FCS files are located
> # Type the path into the quotes
> setwd("")
> # Or in Windows use
> setwd(choose.dir(getwd(), "Choose the folder containing the FCS files"))
> # Get a list of the filenames of the FCS files located in your working directory
> files <- list.files(getwd(),full=TRUE,pattern="*.fcs")
> # Get a list of the filenames of the FCS files included to the package
> files <- list.files(system.file("extdata",package="flowCHIC"),
+ full=TRUE,pattern="*.fcs")
> # Read the first FCS file as flowFrame
> frame <- read.FCS(files[1],alter.names=TRUE,transformation=FALSE)
> # Get a list of the parameter/channel names
> unname(frame@parameters@data$name)
```

Both parameters/channels are used for every FCS file. The default parameters/channels are ”FS.Log” and ”FL.4.Log”. Note that the FCS files included to this package only contain the channels ”FS.Log” and ”FL.4.Log”. Maybe there are some warning messages printed to the R console. They have no effect on the results.

```
> # Create the histogram images using default parameters
> fcs_to_img(files)
> # Create the histogram images using different channels
> # Not working for the FCS files attached to the package
> fcs_to_img(files,ch1="SS.Log",ch2="FL.1.Log")
```

A new subfolder called ”chic_images” is created in the working directory containing the histogram images of all FCS files.

5.2 Creating subsets of histogram images

For creating the subsets the list with the filenames of the FCS files from section 5.1 is used again. Now, a rectangular region is defined to cut out the noise from each image. Therefore, the parameters `"x_start"` (default=0), `"x_end"` (default=4095), `"y_start"` (default=0) and `"y_end"` (default=4095) are used to set the boundaries. The default values of the rectangular gate represent the whole histogram image if the values of the parameters/channels are stored on log scale. Again, the parameters/channels for creating the subset images are set. They should be equal to the parameters/channels used for creating the histogram images in section 5.1.

```
> # Create the histogram image subsets using default parameters
> img_sub(files,ch1="FS.Log",ch2="FL.4.Log")
```

A new folder called ”chic_subset” containing the subset histogram images of all FCS files is created in the working directory.

The boundaries are changed to create subset images that only contain relevant information (see section 3.2). The parameters `"xbin"` and `"maxv"` can be changed in this part, too. `"xbin"` describes the number of bins within the histogram and can be seen as the resolution of the data values. The default value (128) is taken over from [**?**]. The default value will probably work for most measurements recorded under similar settings (diverse microbial community, 200.000 to 250.000 recorded events). For very different applications see [**?**] for a possible way of optimizing the resolution. `"maxv"` describes the maximal value of the expressions within the histogram that is set to the highest color value (black). This parameter depends on the data and may be adapted. Usually, we are working with samples containing a very high number of events and very different distributions regarding to the abundance of a subcommunity. The default value (200) works well in this case. If the distributions are more similar this value should be decreased. In practice, values between 130 and 200 revealed the best results.

```
> # Create the histogram images using different boundaries for the x/y axis
> # and a different value for the maximal value set to black
> # These values work well for the first downloadable dataset and
> # the FCS files included to the package
> img_sub(files,x_start=200,x_end=3500,y_start=1000,y_end=3000,maxv=160)
```

5.3 Image analysis

At first, save the filenames of all subset image files as a list. The next method calculates the overlap and XOR images for each combination of every image from this list. To continue with the next part save the returned values as a list called ”results”.

```
> # Save the names of all subset images as a list
> subsets <- list.files(path=paste(getwd(),"chic_subset",sep="/"),full=TRUE,pattern="*.png")
> # Calculate overlap and XOR images and write values to two new files
> results<-calculate_overlaps_xor(subsets)
```

You can also save the calculated values to text files.

```
> # Save the names of all subset images as a list
> subsets <- list.files(path=paste(getwd(),"chic_subset",sep="/"),full=TRUE,pattern="*.png")
> # Calculate overlap and XOR images and write values to two new files
> calculate_overlaps_xor(subsets,verbose=TRUE)
```

Two files called `"Results_overlaps.txt"` and `"Results_xor.txt"` are created in the working directoy. The file `"Results_overlaps.txt"` contains the number of pixel within the overlapping area for each combination of every subset histogram image. The file `"Results_xor.txt"` contains the sum of all pixel values of the XOR images for each combination of every subset histogram image. These are the same values that are returned and saved to the list ”results” in the example before.

5.4 Calculating the similarities

The last part includes the calculation of the similarities found in the histogram images of cytometric data. NMDS (nonmetric multidimensional scaling) is a mathematical technique to visualize the distances between the samples. The distances within the plot are used to be in accordance with the dis-/similarities of the respective samples. For more details see [**?**]. After finishing the previous part you can use the values saved to the returned list called "results" or you can read the created text files in R.
Use the retuned values to show the similarities of these 3 samples. A NMDS plot is displayed using the names of the samples as data point texts. By default the plotting symbols are black circles. The position of the text is below the data points. The order and the number of the samples is printed to the R console.

```
> # Show a NMDS plot of the samples
> plot_nmds(results$overlap,results$xor)
```

If you saved the values as text files you first have to read them in R.

```
> # Type the filename of the overlap data (including extension) into the quotes
> Results_overlaps<-read.table("Results_overlaps.txt",header=TRUE,sep="\t")
> # Type the filename of the XOR data (including extension) into the quotes
> Results_xor<-read.table("Results_xor.txt",header=TRUE,sep="\t")
```

Then you can use them to to create the NMDS plot as well.

```
> # Show a NMDS plot of the samples
> plot_nmds(Results_overlaps,Results_xor)
```

The attached datasets `Results_overlaps` and `Results_xor` contain the same values and are used to show the result.

The content of Results_overlaps looks as follows:

Label	Area
overlap_BCK_30_Sep_11_21_&_BCK_30_Sep_11_30.png	51941
overlap_BCK_30_Sep_11_21_&_BCK_30_Sep_11_77.png	51625
overlap_BCK_30_Sep_11_30_&_BCK_30_Sep_11_77.png	50736

Table 1: Content of Results_overlaps

The content of Results_xor looks as follows:

Label	IntDen
overlap_BCK_30_Sep_11_21_&_BCK_30_Sep_11_30.png	390644010
overlap_BCK_30_Sep_11_21_&_BCK_30_Sep_11_77.png	347357340
overlap_BCK_30_Sep_11_30_&_BCK_30_Sep_11_77.png	313110630

Table 2: Content of Results_xor

```
> # Load datasets
> data(Results_overlaps)
> data(Results_xor)
> # Show a NMDS plot of the samples
> plot_nmds(Results_overlaps,Results_xor)
```

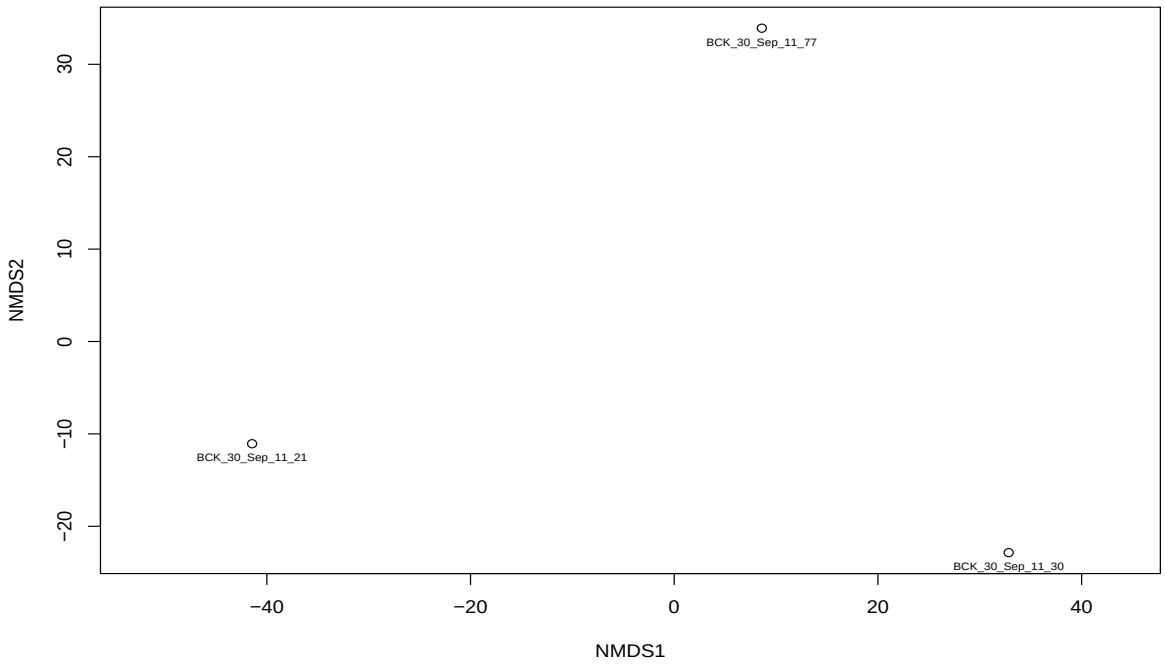

Figure 1: NMDS plot of the samples included to the package using default parameters

Instead of the default parameters the plotting symbols can be changed e.g. to red triangles. The text can be set above the data points and a title can be added to the plot.

```
> # Change default parameters
> plot_nmds(Results_overlaps,Results_xor,main="NMDS plot",pos=3,pch=2,col_nmds="red")
```

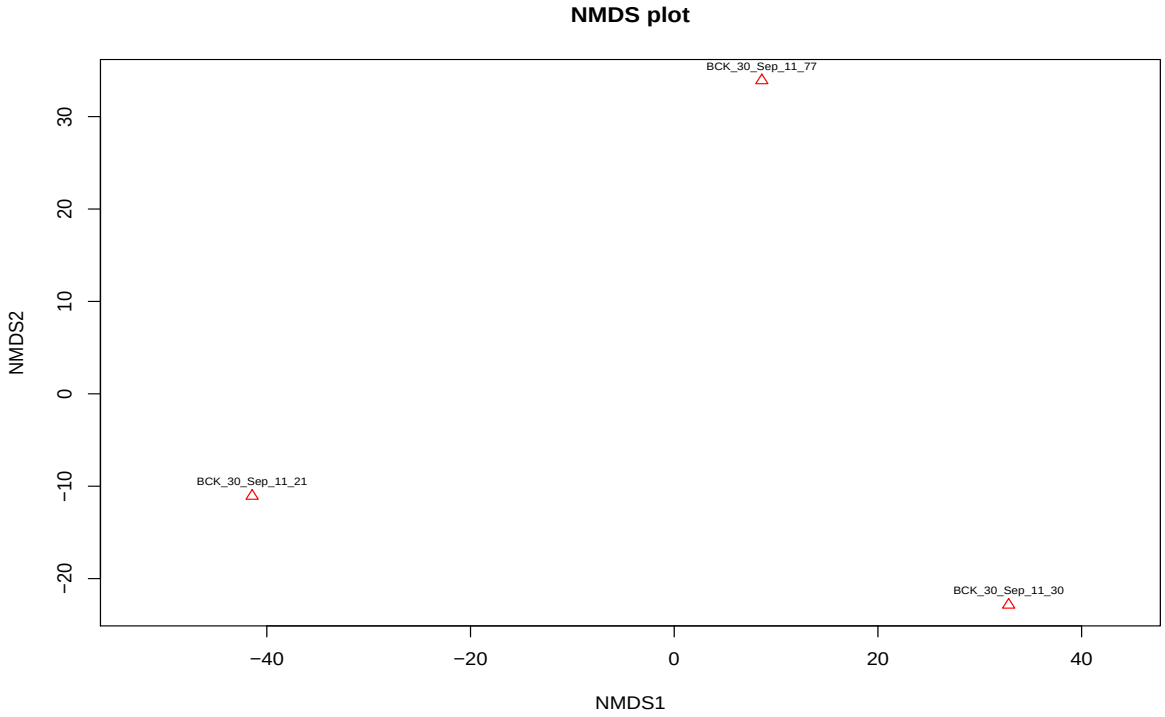

Figure 2: NMDS plot of the samples included to the package using different parameters

In the following the datasets `Results_overlaps_mix`, `Results_xor_mix`, `Results_overlaps_incol` and `Results_xor_incol` are used for demonstration and the influence of different parameter settings of the method on the resulting NMDS plot (e.g. groups, colors). The 4 datasets are attached to the package.

`Results_overlaps_mix` and `Results_xor_mix` contain the results after performing the last three steps on the first downloadable dataset. `Results_overlaps_incol` and `Results_xor_incol` contain the results after performing the last three steps on the second downloadable dataset. For the second dataset the boundaries for creating the subset histogram images were adapted. The contents of these datasets (excerpts) are shown before the plots.

```
> # Load dataset
> data(Results_overlaps_mix)
> # Show data (first five lines)
> Results_overlaps_mix[1:5,]

                                                Label  Area
1 overlap_BCK_30_Sep_11_100_&_BCK_30_Sep_11_101.png 46359
2 overlap_BCK_30_Sep_11_100_&_BCK_30_Sep_11_102.png 46526
3 overlap_BCK_30_Sep_11_100_&_BCK_30_Sep_11_103.png 46577
4 overlap_BCK_30_Sep_11_100_&_BCK_30_Sep_11_104.png 46629
5 overlap_BCK_30_Sep_11_100_&_BCK_30_Sep_11_105.png 46573

> # Load dataset
> data(Results_xor_mix)
> # Show data (first five lines)
> Results_xor_mix[1:5,]

                                               Label    IntDen
1 xor_BCK_30_Sep_11_100_&_BCK_30_Sep_11_101.png 142388910
2 xor_BCK_30_Sep_11_100_&_BCK_30_Sep_11_102.png 322747110
3 xor_BCK_30_Sep_11_100_&_BCK_30_Sep_11_103.png 325748070
4 xor_BCK_30_Sep_11_100_&_BCK_30_Sep_11_104.png 325175130
5 xor_BCK_30_Sep_11_100_&_BCK_30_Sep_11_105.png 275680440


> # Plot
> plot_nmds(Results_overlaps_mix,Results_xor_mix)
```

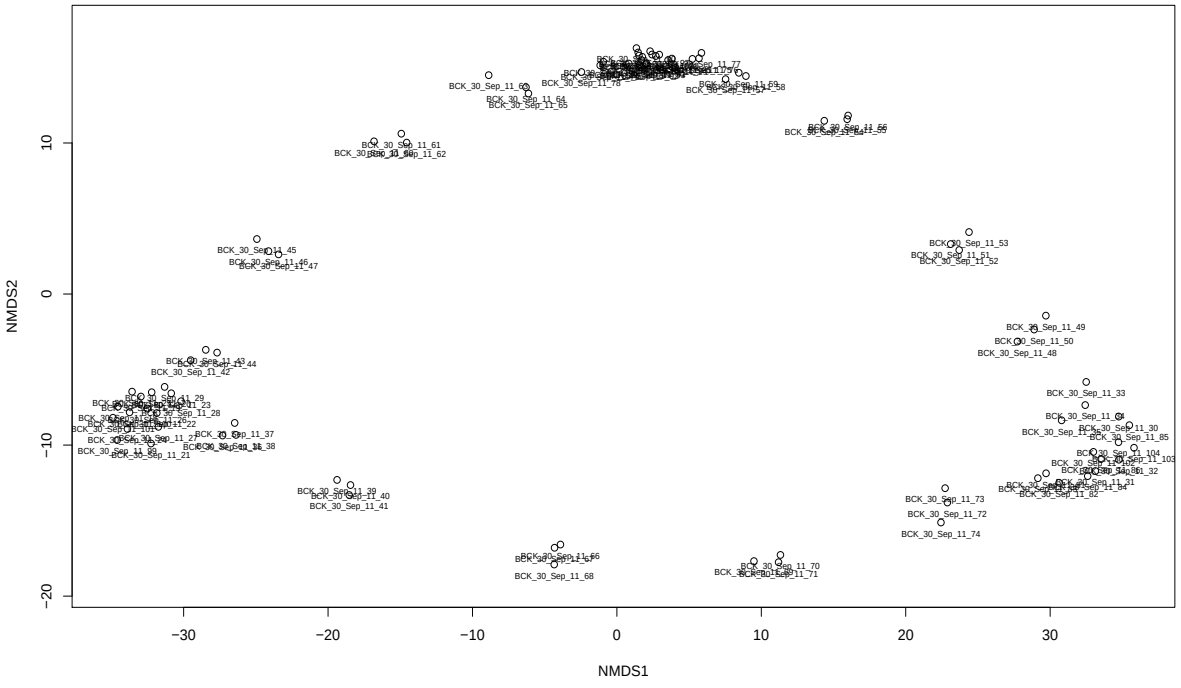

Figure 3: NMDS plot of the samples of the first dataset using default parameters

```
> # Load dataset
> data(Results_overlaps_incol)
> # Show data (first five lines)
> Results_overlaps_incol[1:5,]

                Label  Area
1 overlap_S1_&_S10.png 48168
2 overlap_S1_&_S11.png 48114
3 overlap_S1_&_S12.png 50115
4 overlap_S1_&_S13.png 52019
5 overlap_S1_&_S14.png 50615

> # Load dataset
> data(Results_xor_incol)
> # Show data (first five lines)
> Results_xor_incol[1:5,]

             Label    IntDen
1 xor_S1_&_S10.png 152087580
2 xor_S1_&_S11.png 153945000
3 xor_S1_&_S12.png 152040870
4 xor_S1_&_S13.png 135726840
5 xor_S1_&_S14.png 151484040


> # Plot
> plot_nmds(Results_overlaps_incol,Results_xor_incol)
```

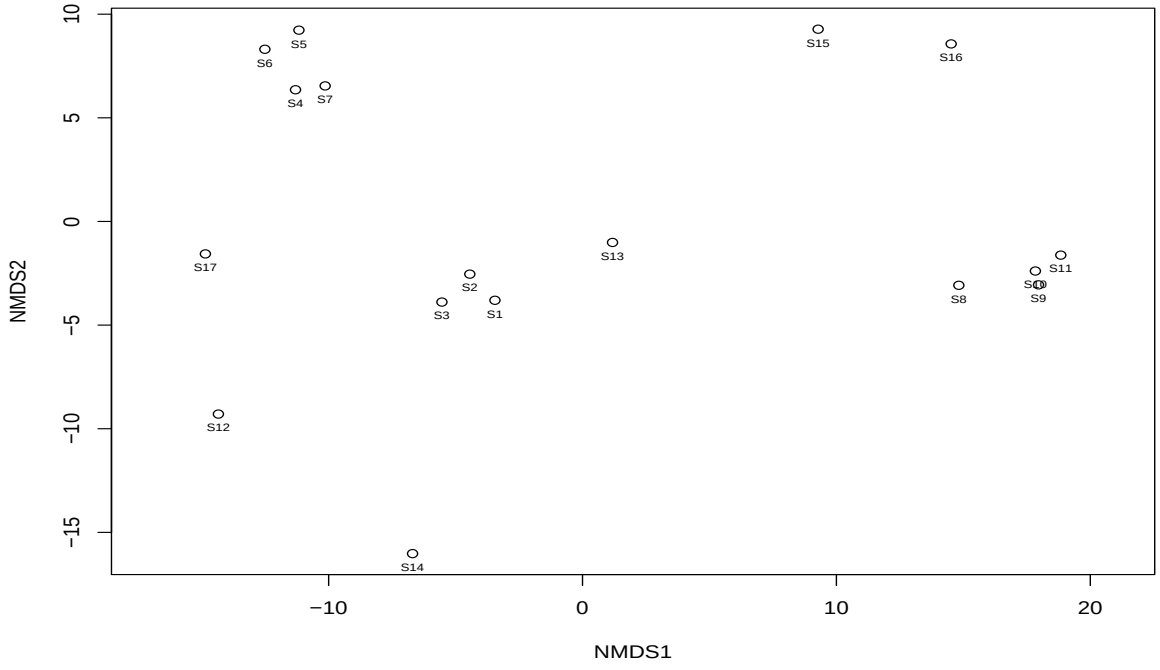

Figure 4: NMDS plot of the samples of the second dataset using default parameters

The samples can be classified into groups manually. Therefore, every sample has to be assigned to one group. In this example 17 samples are assigned to three different groups using the data frame `groups`. The data frame looks as follows:

```
> # Create data frame with group assignments
> groups<-data.frame("groups"=c(15,19,19,19,15,22,19,15,22,15,15,22,22,22,22,19,19))
```

	groups
1	15
2	19
3	19
4	19
5	15
6	22
7	19
8	15
9	22
10	15
11	15
12	22
13	22
14	22
15	22
16	19
17	19

Table 3: Group assignments

This data frame has seventeen entries and describes the assignment of every sample to one group (three groups overall in this example). Every sample is assigned to one value (1-25) so the same value means the same group. Up to 25 groups are possible (as only 25 plotting symbols are available).

It is also possible to read a text file containing as many lines as samples whereas every line contains one value that assigns the sample to the respective group. The samples can be grouped by e.g. treatment, day of sampling or personal preferences. Always make sure to check the order of the samples. The order and the number of the samples is printed to the R console after calling the `plot_nmds()` method.

```
> # Plot
> plot_nmds(Results_overlaps_incol,Results_xor_incol,group=groups)
```

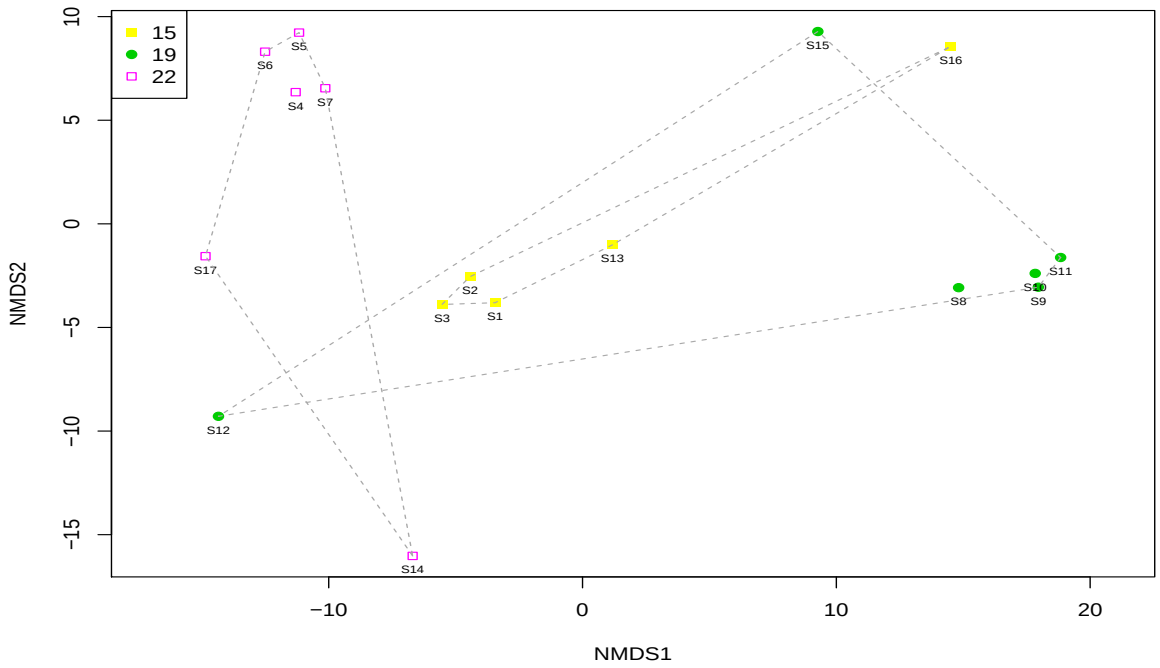

Figure 5: NMDS plot of the samples of the second dataset with group assignment

In addition, abiotic parameters/gate information can be added to the NMDS plot suggesting the differences found within the plot. In this example the attached dataset `abiotic_incol` is used for demonstration. This dataset contains experimental data for every sample. The content looks as follows (excerpt):

```
> # Load dataset
> data(abiotic_incol)
> # Show data (first five lines)
> abiotic_incol[1:5,]

  sample acetate methanol  pH inoculumA inoculumB
1     S1       1        0   7         1         0
2    S10       0        1   8         1         0
3    S11       0        1 8,5         1         0
4    S12       0        1 7,5         1         0
5    S13       1        0 7,5         1         0
```

It is important that the order and the number of the lines are identical to the order and the number of the samples printed in R. Otherwise, the output will be wrong since association of abiotic data with their samples will fail. The significance level p is set to 0.05 and the color used for plotting is set to `"magenta"` by default. If p is changed to 1 all abiotic parameters will be shown.

```
> # Plot
> plot_nmds(Results_overlaps_incol,Results_xor_incol,
+           abiotic=abiotic_incol[,-1])
```

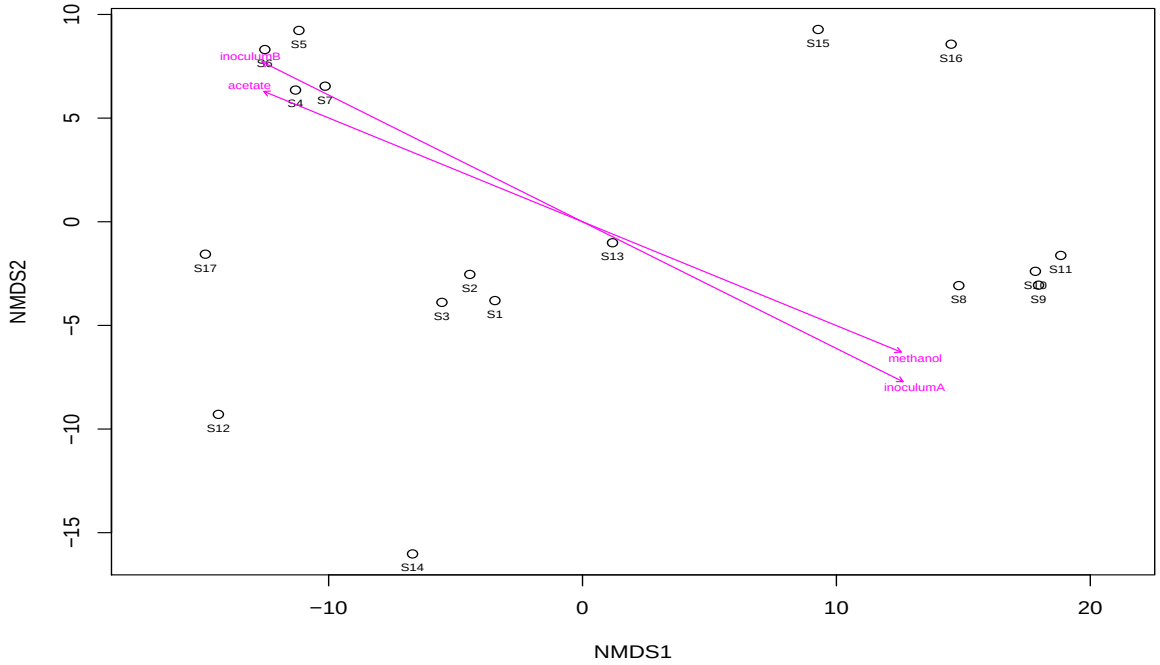

Figure 6: NMDS plot of the samples of the second dataset with abiotic parameters

Grouping and plotting abiotic parameters/gate information can be combined in one plot.

```
> # Plot
> plot_nmds(Results_overlaps_incol,Results_xor_incol,group=groups,abiotic=abiotic_incol[,-1])
```

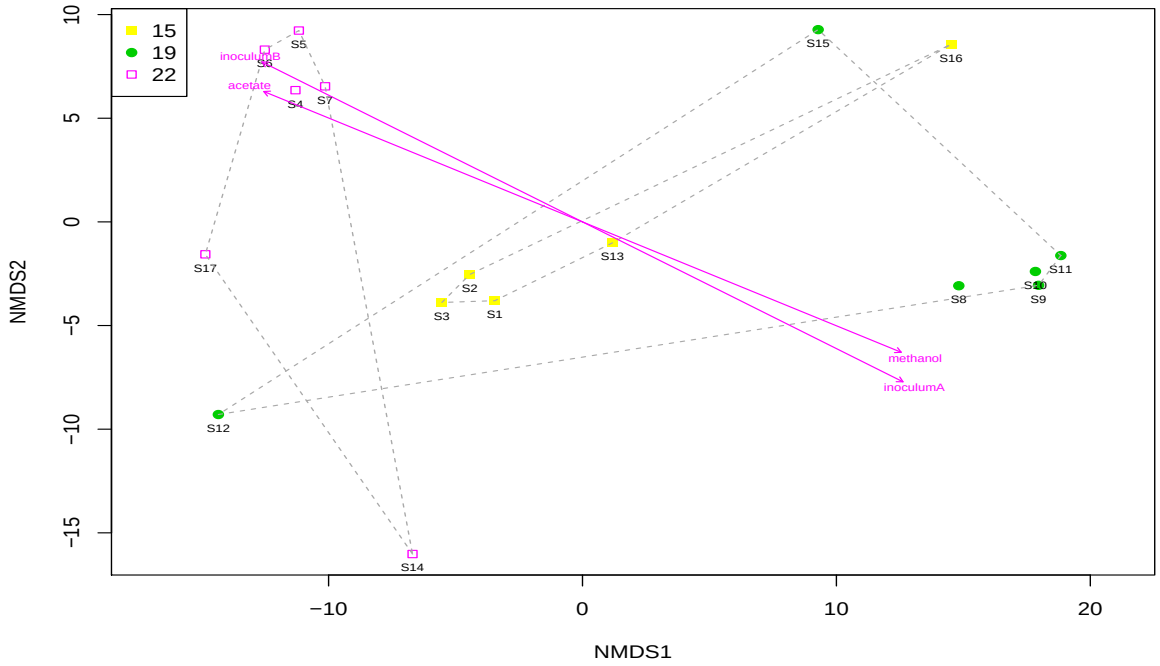

Figure 7: NMDS plot of the samples of the second dataset with group assignment and abiotic parameters

Now, the legend is placed to the top right corner of the plot, the color of the arrows is changed to `"darkred"` and the maximum p-value is set to 0.01. All abiotic parameters less or equal the maximum p-value are added to the plot. Small variations are possible because the results of the NMDS plot vary slightly everytime the method is called.

```
> # Plot
> plot_nmds(Results_overlaps_incol,Results_xor_incol,group=groups,
+           legend_pos="topright",abiotic=abiotic_incol[,-1],
+           p.max=0.01,col_abiotic="darkred")
```

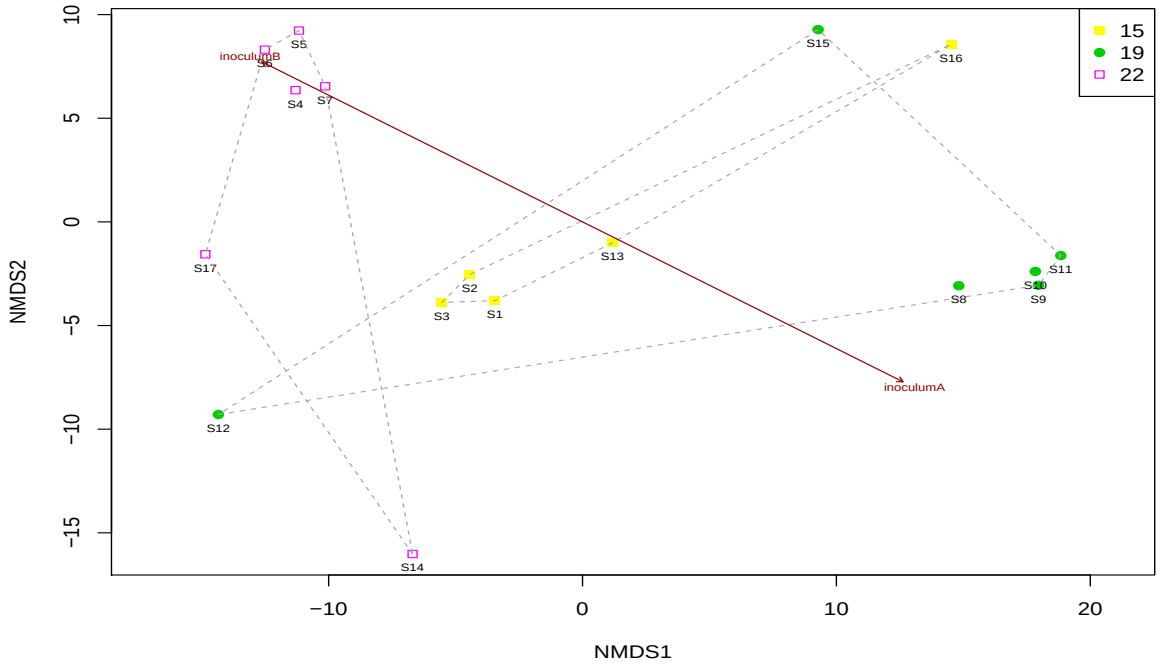

Figure 8: NMDS plot of the samples of the second dataset with group assignment, abiotic parameters and changed plotting parameters

In addition, dendrograms can be shown to reveal clusters (samples with high similarity) within the data.

```
> # Plot
> plot_nmds(Results_overlaps_incol,Results_xor_incol,show_cluster=TRUE,group=groups,
+           legend_pos="topright",abiotic=abiotic_incol[,-1],
+           p.max=0.01,col_abiotic="darkred")
```

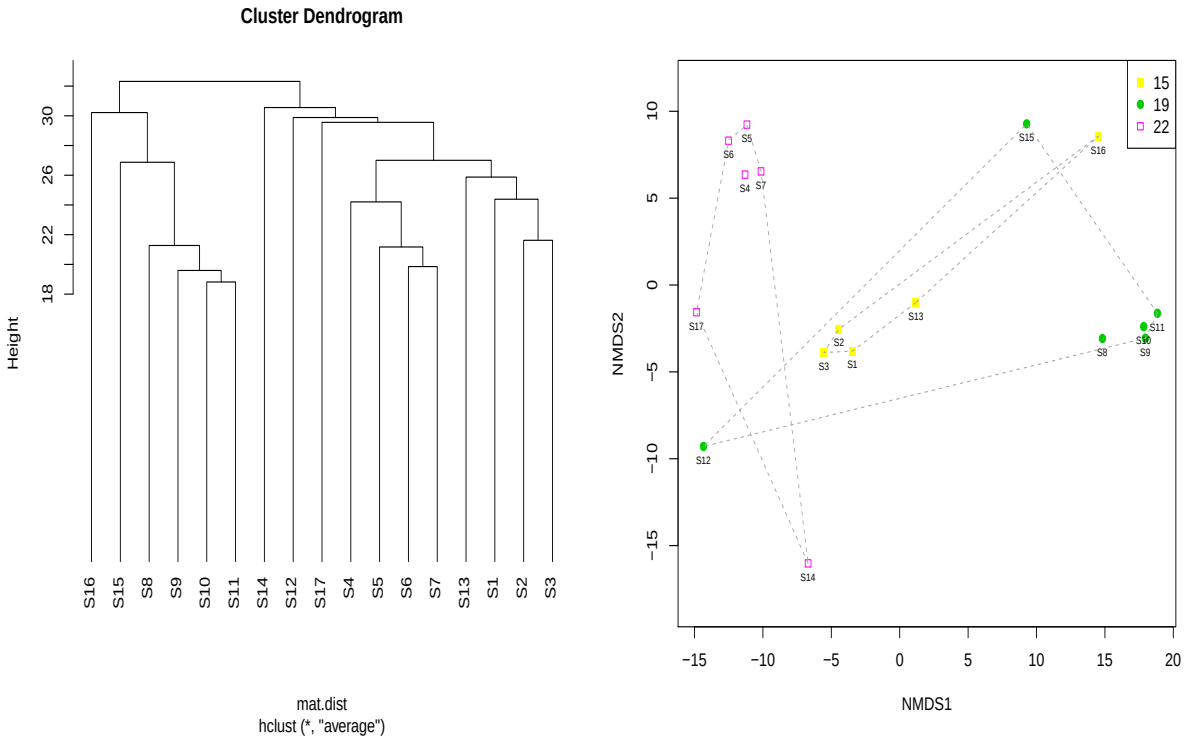

Figure 9: NMDS plot of the samples of the second dataset with group assignment, abiotic parameters, changed plotting parameters and cluster dendrogram

The results of the metaMDS method and the p-values of the abiotic parameters can also be printed to the R console.

```
> # Plot and print results
> plot_nmds(Results_overlaps_incol,Results_xor_incol,show_cluster=TRUE,group=groups,
+           legend_pos="topright",abiotic=abiotic_incol[,-1],
+           p.max=0.01,col_abiotic="darkred",verbose=TRUE)
```