1. Open Terminal and navigate to the folder containing the MetICA Code

```
cd directory/with/code
```

2. Create a directory named 'Results' lying in the same parent directory as the code. This folder will contain files produced by the code

```
cd ..
mkdir Results
```

3. Open R Console and execute the following steps to load all libraries and scripts and to parse and preprocess the data.
   Note: I assume here that the data are saved in a csv-file where columns represent samples and rows represent peaks. The first column contains peak-IDs. This is a typical data structure used at the UFZ. The developers of MetICA, however, require data in a form where columns represent peaks and rows represent samples. If your data is structured that way, please leave out the transpose operator at the end of this code example.

```
source('MetICA_load_all.R')

example_data = read.table('example.csv', sep=',',dec='.',
    header=TRUE, check.names=FALSE)
peaks = data.matrix(example_data[,2:ncol(example_data)])

#Log-scalling of peaks
peaks = log(peaks + min(peaks) + 1)

#Taking only peaks appearing in more than one sample...
peaks = peaks[apply(peaks>0, MARGIN=1, sum) > 1, ]

#and having different values for at least 2 samples
ranges = apply(peaks, MARGIN=1, range)
peaks = peaks[ranges[1,] != ranges[2,], ]

#Finally, the matrix needs to be transposed
#in order to match the input format of the source generator
peaks = t(peaks)
```

4. Start generating independent components. (The terms 'independent components' and 'sources' are used interchangably here.) This will output a file named `source_list_<nbcomp>.rda`, where `<nbcomp>` is replaced by the number of independent components.
   Note: The second parameter of the function specifies the total variance kept while the fourth parameter specifies the number of iterations. Both are important for the expected time of computation. In this example, only 30% of variance is kept to make things run faster. The total variance will also directly determine the number of principal components which is equal to the number of sources.

```
M1 = MetICA_source_generator(peaks, 0.3, 'gaussian', 800)
```

5. Cluster the generated sources: The cluster generator evaluates the optimal number of clusters and computes centrotypes for this number. It produces the following output files in the 'Results' folder:

- `nbclust_eval_<nbcomp>_<nbclust>.csv`, where `<nbcomp>` is the number of components and `<nbclust>` is the determinded optimal number of clusters. The file contains average silhouette scores for different numbers of clusters
- `center_<nbclust>.rda`, which contains the resulting centrotypes

This will also create several auxiliary files in the 'Dist' folder, storing dissimilarity matrices. Before running the code again, all files from this folder should be deleted.

```
M2 = MetICA_cluster_generator(M1$S, 'spearman', M1$nbcomp)
```

6. (Optional) An evaluation of statsitical reliability of the resulting centrotypes can be done on bootstrapped datasets. This will produce the following outputs in 'Results'

- `IC_notes_summary_<nbclust>.csv`, storing scores for each bootstrapped dataset (rows) and centrotype (columns)
- `IC_ranking_<nbclust>.png`, showing the corresponding ranking of centrotypes in a boxplot

Warning: This might take a long time even for small numbers of components.

```
M3 = MetICA_bootstrap(peaks, 5, M2$centers, M1$nbcomp,
                              'gaussian', 100, 50)
```