Calculating diversity faultlines with the asw.cluster package in R

Bertolt Meyer[1] & Andreas Glenz[2]

[1] TU Chemnitz, Germany
[2] University of Zurich, Switzerland

Version 2.01
last updated: March 03, 2017

if you require help with using the software and/or would like to connect with other researchers who are interested in subgroups and faultlines, please join or mailing list by sending an empty email with the subject

subscribe faultlines your_first_name your_last_name

(e.g., subscribe faultlines Bertolt Meyer) to sympa@psychologie.lists.uzh.ch. After you receive the confirmation mail, you can send mail to faultlines@psychologie.lists.uzh.ch.

Before posting a question, please check if somebody else has already posted it. If you ask for help or report a bug, please make sure to include a code snippet so that others can reproduce the problem.

Introduction

This manual describes how to calculate various diversity faultline measures with the open source statistical environment R (R Development Core Team, 2011), using the `asw.cluster` package. R packages are extensions (like plug-ins) that extend the functionality of R by adding new commands to the basic set of commands available in R. The `asw.cluster` package adds the function `faultlines()` to R. This function can calculate the faultline measures proposed by K. Bezrukova, Jehn, Zanutto, and Thatcher (2009), by Gibson and Vermeulen (2003), by Meyer and Glenz (2013), by Shaw (2004), by S. Thatcher, Jehn, and Zanutto (2003), by Trezzini (2008), and by van Knippenberg, Dawson, West, and Homan (2011).

This manual is intended for both novice and experienced users of R. Novice users who have never used R before can read it as a step-by-step guide for calculating diversity faultline measures for a data set that already exists in a file such as an SPSS data file or a comma-separated-value text file. Experienced R users will find information on organizing their team data prior to calculations (see Section *Importing team data*) and examples (see Section *Calculating diversity faultlines with the faultlines() function*).

Experienced R users can install the package with the following two commands:

```
install.packages(c("flexmix", "nnet", "nFactors", "QuantPsyc", "psych"))
install.packages("asw.cluster",
                 repos="http://www.group-faultlines.org/packages",
                 type="source")
```

and can call `?faultlines()` afterwards.

For license reasons, we are currently unable to make the package available from the CRAN repository.

Installing R

The first step is the installation of R. To install R, download the most current version for the required operating system from http://www.r-project.org. Access the website, click on the link "CRAN" (Comprehensive R Archive Network) in the navigation on the left and choose the download server closest to your location. Next, select your operating system, download the installation file, and execute it. After the installation is complete, launch R by double-clicking on its program icon.[1]

First steps in R

When you launch R, you see one window titled "Console". This is where R outputs the results from the user's commands (like the output viewer in SPSS). Commands can also be entered in a one-by-one fashion into the console by typing them in at the > prompt.

Opening a new empty syntax file

Typing commands into the console is cumbersome. It is better practice to open a syntax file in a separate window and to type your syntax into it. This syntax can be executed line by line, or you can execute several highlighted lines of code at once. To open a new empty syntax file on a Mac, select "New File" from the "Files" menu. On a Windows installation of R, select "New Script" from "File".

[exec] In a syntax file, highlighted code is executed by simultaneously hitting the keys Ctrl and the letter R on Windows machines. On Mac installations, highlighted code is executed by pressing the cmd key and the enter key simultaneously. If no code his highlighted, doing this will result in the execution of the code in the line where the cursor is positioned.

Setting the working directory

One important concept when working with R is the so-called working directory. In a given R session, R treats one folder on the hard drive as a standard directory where it saves files and where it looks for the files that the user wishes to open. Thus, as a first step in

[1] R is a powerful and flexible statistical environment for analyses and visualizations. If you do not already use R, the authors strongly recommend to familiarize yourself with its the core principles. An excellent introduction to the R environment is available as a PDF from this address: <http://cran.r-project.org/doc/manuals/R-intro.pdf>

your R session, you should specify a directory as a working directory. This directory should contain the data set containing the teams for which faultlines are to be calculated.

On Windows, the working directory is set by selecting "Change directory" from the "Files" menu. On a Mac, the working directory is set by selecting "Change working directory" from the "Extras" menu. Navigate to the directory that contains the data and the package files, and click "choose".

To see the path of the current working directory, you can execute the command `getwd()` from your syntax file. It will print the location of the current working directory to the console.

You are now ready to install the package that enables R to calculate faultlines.

Installing the asw.cluster package

The `asw.cluster` package requires some other packages to work properly. These so-called dependencies must be installed before the package can be activated. To install the dependent packages, make sure that your Internet connection is working and copy the following command into your syntax file and execute it (see Section [exec] for how to execute syntax):

```
install.packages(c("flexmix", "nnet", "nFactors", "QuantPsyc", "psych"))
```

Subsequently, execute the following command to download the `asw.cluster` package (make sure to highlight all lines of code if this command is wrapped in several lines before you execute it):

```
install.packages("asw.cluster",
                 repos="http://www.group-faultlines.org/packages",
                 type="source")
```

As a next step, the package must be activated. To do so, execute the following command:

```
library(asw.cluster)
```

The console should echo this command, possibly informing you that other necessary packages are also installed, followed by the input prompt `>` indicating that no errors occurred during the activation of the package. You are now ready to import your data set into R and to calculate diversity faultlines.

Note that the package needs to be installed only once on your computer, but that you must activate it again if you quit and re-launch R.

In other words, `install.packages("asw.cluster")` must be executed only once, but `library(asw.cluster)` must be called after every relaunch of R.

Importing team data

To import your data into R, the data file needs to be read into R's working memory, which is called the workspace. More specifically, you need to create a data frame object that

contains your data. The term data frame is R's denomination for data set. The commands required for creating a data frame object with your data in it depend on the type of file that holds your data. The following two subsections explain how to read SPSS data and comma-separated-value plain text data.

Importing SPSS data sets

To read an SPSS data file with the ending .sav with the following syntax, it must reside in the current working directory (see Section [wd] for how to set the working directory). The command required to read SPSS files, `read.spss()`, is included in the package `foreign`, which is included in the standard installation of R but must be activated prior to use. Activate it by executing the following command:

```
library(foreign)
```

Before importing your SPSS data file, here are some guidelines on how the file should be organized:

- The file must contain the demographic information for the team members of all teams for which faultlines are to be calculated. Each row should represent a team member, and the diversity attributes of team members such as age, gender, or personality, should be in the columns. The file can also contain further variables that are not of interest for the faultline calculation, such as an experimental treatment, date of data acquisition, and so forth.

- If the data file contains more than one team, it must include a numeric variable indicating team membership to a given numbered team for each team member. For example, if the first three rows of the data set represent three members of team 1, the first three entries of this variable must read 1, 1, 1. Teams need not be numbered consecutively, but a consecutive numbering from 1 to n where n denotes the total number of teams can make things easier in a later stage of the analyses.

- In R, SPSS variable labels cannot be used to address specific variables (they can however be displayed). Only the actual variable names are used in R so make sure that your variables have short and meaningful names such as "age", "gender", or "ethnicity".

- Make sure that in the SPSS data file, nominal variables such as gender or ethnicity are either coded as numeric values with value labels or as character strings (e.g., "female" or "male" as values in the gender variable) and that their scale type is set to "nominal".

- Make sure that numeric variables with metric or interval scale such as questionnaire items denoting agreement or disagreement on a scale from 1 to 5 **do not** have value labels such as "I strongly agree". The reason for this is that when importing the SPSS data into R, R will overwrite the numeric values with the value labels, which makes numeric operations on these variables impossible.

- Make sure that missing values are defined as system missing ($SYSMIS) in SPSS, i.e., that they show up as an empty cell with a dot in them in the SPSS data view. Only system missing variables will be recognized as missing values by R.

If these considerations are met, you can import your SPSS data file into R using the following syntax, which assumes your data file is called teamdata.sav and resides in your current working directory (make sure to highlight all of the lines and to call `library(foreign)` before):

```
teamdata <- read.spss(file = "teamdata.sav",
                      to.data.frame = TRUE,
                      reencode = TRUE)
```

This command might return a warning with regard to an unknown data type, but those can be ignored as long as R does not echo an error. These warnings simply indicate that R encountered a special character that it had trouble converting, but the data set has been read completely.

By executing this command, you have learned the basic principle of R syntax: A command or function has a specific name followed by round brackets, `read.spss()` in this case. Inside the brackets, you specify the parameters that the function needs. Each parameter has a name and a value, e.g., `file = "teamdata.sav"`. The assignment operator, the left arrow `<-`, is used to store the result of the command, a complete data frame in this case, into an object of an arbitrary name, `teamdata` in this case. This object resides in the workspace and can be used for further operations until it is deleted or R is quit. To learn which parameters a function needs, open its help page by typing `?` followed by the function name, e.g.,

```
?read.spss()
```

Importing data from comma separated values files

A very common file format for data sets is the comma separated values (.csv) format. Files in the csv format are plain text files that contain a row for each observation in the data set, and the values of different variables are separated by commas. All statistical software packages and spreadsheet applications such as Microsoft Excel can read and write csv files.

To read a csv data file with the ending .csv with the following syntax, it must reside in the current working directory (see Section [wd] for how to set the working directory). Before importing your csv data file, here are some guidelines on how the file should be organized:

- The file must contain the demographic information for the team members of all teams for which faultlines are to be calculated. Each row should represent a team member, and the diversity attributes of team members such as age, gender, or personality, should be in the columns. The file can also contain further variables that are not of interest for the faultline calculation, such as an experimental treatment, date of data acquisition, and so forth.

- If the data file contains more than one team, it must include a numeric variable indicating team membership to a given numbered team for each team member. For example, if the first three rows of the data set represent three members of team 1, the first three entries of this variable must read 1, 1, 1. Teams need not be numbered consecutively, but a consecutive numbering from 1 to n where n denotes the total number of teams can make things easier in later stages of the analyses.
- Make sure that missing values are simply missing values in the data frame.

If these considerations are met, you can import your csv data file into R using the following syntax, which assumes your data file is called teamdata.csv and resides in your current working directory:

```
teamdata <- read.csv(file = "teamdata.csv")
```

In countries employing the comma as the decimal separator (e.g., Germany), csv files use the semicolon for separating the values in rows. In these countries, use the `read.csv2` function:

```
teamdata <- read.csv2(file = "teamdata.csv")
```

By executing this command, you have learned the basic principle of R syntax: A command or function has a specific name followed by round brackets, `read.csv()` in this case. Inside the brackets, you specify the parameters that the function needs. Each parameter has a name and a value, e.g., `file = "teamdata.csv"`. The assignment operator, the left arrow `<-`, is used to store the result of the command, a complete data frame in this case, into an object of an arbitrary name, `teamdata` in this case. This object resides in the workspace and can be used for further operations until it is deleted or R is quit. To learn which parameters a function needs, open its help page by typing `?` followed by the function name, e.g.,

```
?read.csv()
```

Preparing the data set for calculating faultline measures

After the successful import of your data, the R workspace now contains a data frame object `teamdata` that contains the entire data set, i.e., all variables and all cases. You can display the names of the variables included in the data set by calling

```
names(teamdata)
```

The first few cases of the data set are displayed by calling

```
head(teamdata)
```

The entire data frame is printed into the console by calling its name:

```
teamdata
```

An SPSS-like table view of the data frame is also available:

```
fix(teamdata)
```

This table view of the data must be closed before R accepts further commands.

If your data set contains more variables than those required for the calculation of faultlines, you need to create a subset of the data set that only holds the diversity attributes that you want to use for the faultline, plus a team number variable. To create a new data frame in the R workspace called `teamdata_sub` that only contains the variables `teamid, age, gender,` and `ethnicity`, execute the following command:

```
teamdata_sub <- teamdata[,c("teamid", "age", "gender", "ethnicity")]
```

Faultlines cannot be calculated for missing data. Thus, prior to calculating faultlines, you need to remove all team members with missing values in the diversity attributes from the dataset. This can be achieved with the following command that assumes that your data frame is called `teamdata_sub`:

```
teamdata_sub <- teamdata_sub[complete.cases(teamdata_sub),]
```

Now that the data frame only contains the diversity attributes that are intended for the calculation of the faultline, a variable denoting team membership (`teamid` in this example), and no missing values, you are ready to calculate diversity faultlines.

Calculating diversity faultlines with the faultlines() function

This section assumes that one has successfully installed and activated the `asw.cluster` package (see Section [install]) and that you one has a data frame `teamdata_sub` in one's R workspace that has been prepared according to the guidelines provided in Section [import].

As a first step, read the help page for the `faultlines()` function by calling `?faultlines()`. The "Details" section of the help page explains the principles of the function:

The `faultlines()` function is the main function of the package and is intended for calculating the faultlines; most other functions in the package are workhorse and sub functions that are called by `faultlines()`.

The function is run over a data set (a data frame passed to the function with the parameter data) containing the members of one or more teams as rows and their diversity attributes that are used for calculating a given faultline measure as columns. Note that all columns will be used for calculating the faultline. Thus, this will most likely be a subset of the "full" data frame that was gathered in a given research. If the data set contains more than one team, it must contain a column that specifies a team number for each team member, thus indicating team membership. The name of this column must be passed to the argument `group.par`.

For each diversity attribute contained in the data frame, the user must specify its scale (either numeric or nominal) in the same order of the variables in the data frame as a character vector and pass it to the function with the parameter `attr.type`. For example, if the data set contains the variables age (numeric in years), ethnicity (character factor), and gender (character factor), this parameter must be specified as
`attr.type = c("numeric", "nominal", "nominal")`. Note that the faultline measures proposed by Shaw (2004) and Trezzini (2008) require that all attributes are nominal. Thus, prior to calculating diversity faultline strengths with these two methods, you must recode numeric attributes such as age to factors with levels such as 'young', 'middle-aged', and 'old' and specify the attribute type of this variable as nominal.

If the user wishes to calculate one of the diversity faultline measures `"thatcher"`, `"bezrukova"`, or `"asw"` that are capable of dealing with numeric attributes such as age or tenure, you must specify a weight for each diversity attribute with the `attr.weight` parameter. These weights indicate how strong a difference of 1 (in case of numeric attributes) or a different category (in case of nominal attributes) is fractured into the faultline. In the example case of age (in years), gender, and ethnicity, specifying this parameter as `attr.weight = c(0.1, 1, 1)` means that an age difference of ten years is equally weighted as a difference in gender, which is equally weighted as a difference in ethnicity. Note that these are the default values for Thatcher's et al. (2003) Fau that are probably used in most papers, but these appear to be arbitrary. More research is required with regard to the choice of these weights in a given context. Note that `rescale` combines with the `attr.weight`-parameter, which means that numeric attributes are first rescaled according to the rescale-parameter. In a second step, all attributes (dummy-coded values for nominal attributes) are multiplied by their appropriate weight given by the `attr.weight`-parameter.

The `metric` parameter lets the user specify whether Euclidean or Mahalanobis distances should be employed in determining how different team members are from each other. This metric is only employed by the methods `"thatcher"`, `"bezrukova"`, and `"asw"`. Note that the former two methods (Bezrukova et al., 2009; Thatcher et al., 2003) were introduced based on Euclidean distances, which assume that diversity attributes are uncorrelated. Meyer and Glenz (2013) showed that correlations between diversity attributes (e.g., between age and tenure) can have a significant influence on diversity faultline measures. They thus suggested to employ Mahalanobis distances to control for such correlations. They explicitly included this option in the calculation of the ASW measure, but invoking it for Thatcher's et al. Fau or for Bezrukova's Faultline Distance measure is purely experimental. Employing Mahalanobis distances for the latter two measures will thus deliver a measure that has not been described in the literature. Furthermore, calculating Mahalanobis distances requires an inversion of the variance-covariance-martix of attributes. Using Mahalanobis metrics is therefore restricted to data sets with invertible variance-/covariance matrices, i.e., to numeric attributes only.

ASW cluster faultlines for multiple subgroups (Meyer & Glenz, 2013)

As illustrated by (Meyer & Glenz, 2013), the ASW measure is the only diversity faultline measure that is suitable for the cases where more than two homogeneous subgroups are possible. In the following, we illustrate how to calculate ASW for an example data set `teamdata_sub`. It consists of two teams with six members each. For each team member,

age, gender, and ethnicity have been collected. The data set can be created by executing the following syntax:

```
teamdata_sub <- data.frame(teamid = c(rep(1,6),rep(2,6)),
                  age = c(44,18,40,33,33,50,22,23,39,42,57,51),
                  gender = c("f","m","f","f","m","f","f","f","m","m","m","m"),
                  ethnicity = c("A","B","A","D","C","B","A","A","B","B","C","C"))
```

Executing the name of the data frame prints its content to the console:

```
teamdata_sub
```

```
   teamid age gender ethnicity
1       1  44      f         A
2       1  18      m         B
3       1  40      f         A
4       1  33      f         D
5       1  33      m         C
6       1  50      f         B
7       2  22      f         A
8       2  23      f         A
9       2  39      m         B
10      2  42      m         B
11      2  57      m         C
12      2  51      m         C
```

The first team appears to be rather heterogeneous and cross-cut, but the second team appears to consist of three rather homogeneous subgroups. To calculate the ASW faultline measure for both teams, we need to specify the scales of the diversity attributes age, gender, and ethnicity as being numeric, nominal, and nominal. Instead of specifying the scale types in the call to the `faultlines()` function, they can also be stored in a variable that can be passed to the function:

```
my_attr <- c("numeric", "nominal", "nominal")
```

You may consider particular attributes as more important for group dynamics in a given context than others. If this is the case, the ASW faultline algorithm needs to know how to weigh the attributes, i.e., how much age difference is seen as equivalent to a difference in gender or ethnicity. Following the example in the introduction of this section, these can be stored in a variable as well:

```
my_weights <- c(0.1, 1, 1)
```

After these considerations have been made, the faultlines can be calculated with the results being stored in a data frame that we call `my_ASW`. Note how in the call to the

`faultline()` function, the name of the data frame containing the demographic information and the name of the variable in that data frame specifying team membership are also passed as parameters:

```
    my_ASW <- faultlines(data = teamdata_sub,
                         group.par = "teamid",
                         attr.type = my_attr,
                         attr.weight = my_weights,
                         method = "asw")
```

Calling the `my_ASW` object reveals its content:

```
    my_ASW
```

```
  team          fl.value mbr_to_subgroups number_of_subgroups
1    1 0.331799912553719      1 2 1 2 2 1                   2
2    2 0.805489497404053      1 1 2 2 3 3                   3
  subgroup_sizes
1            3 3
2          2 2 2
```

In the `my_ASW` object, each line represents a team. The first column denotes the team number and the second, `fl.value`, its faultline strength (the ASW value). The column `mbr_to_subgroups` shows to which subgroup each member belongs. Members are listed left-to right with reference to the top-to-bottom order of the data frame containing the raw data. The column `number.of.subgroups` indicates how many subgroups the algorithm detected in the given team, and the last column lists the subgroup sizes of the subgroups.

The result can be converted into a format where each row represents a team member, see Section [convert]. Regardless of its format, it can be exported for use in other applications such as SPSS, see Section [export].

A longer and more detailed report of the results can be obtained by calling the function `summary()` with the result object as parameter:

```
    summary(my_ASW)
```

```
Number of Teams: 2

Calculation features:
    Method:      ASW
    Level:   team
    Metric:      euclid


Team 1 (1):
===========
```

```
Faultline Strength:
[1] 0.3317999

Individual Faultline Strengths (silhouette widths):
[1]  0.6233902  0.3507280  0.4973520 -0.1024386  0.1353660  0.4864019

Member to Subgroup Association:
[1] 1 2 1 2 2 1

Number of Subgroups:
[1] 2

Distances:
         X1       X2        X3        X4        X5        X6
1  0.000000 26.03843  4.000000 11.045361 11.090537  6.082763
2 26.038433  0.00000 22.045408 15.066519 15.033296 32.015621
3  4.000000 22.04541  0.000000  7.071068  7.141428 10.049876
4 11.045361 15.06652  7.071068  0.000000  1.414214 17.029386
5 11.090537 15.03330  7.141428  1.414214  0.000000 17.058722
6  6.082763 32.01562 10.049876 17.029386 17.058722  0.000000


Team 2 (2):
===========
Faultline Strength:
[1] 0.8054895

Individual Faultline Strengths (silhouette widths):
[1] 0.9570891 0.9555946 0.8343080 0.8094112 0.6892723 0.5872618

Member to Subgroup Association:
[1] 1 1 2 2 3 3

Number of Subgroups:
[1] 3

Distances:
        X1       X2       X3        X4       X5        X6
1  0.00000  1.00000 17.05872 20.049938 35.02856 29.034462
2  1.00000  0.00000 16.06238 19.052559 34.02940 28.035692
3 17.05872 16.06238  0.00000  3.000000 18.02776 12.041595
4 20.04994 19.05256  3.00000  0.000000 15.03330  9.055385
5 35.02856 34.02940 18.02776 15.033296  0.00000  6.000000
6 29.03446 28.03569 12.04159  9.055385  6.00000  0.000000
```

Scaling attributes. To circumvent the issue of assigning arbitrary weights (e.g., a difference in ten years of age equals a difference in gender) to diversity attributes, it has been recommended to scale numeric attributes by their standard deviation, and to scale nominal attributes by $1/\sqrt{2}$ (E. Zanutto, Bezrukova, & Jehn, 2010). The latter results in an Euclidean distance of one between nominal attributes. This scaling is used by default in all papers employing the $Fau \times Dist$ faultline measure, and we also recommend to employ this scaling when calculating ASW faultlines. The application of this scaling is illustrated in the following example:

```
    my_ASW_sc <- faultlines(data = teamdata_sub,
                                    group.par = "teamid",
                                    attr.type = my_attr,
                                    rescale = "sd",
                                    method = "asw")
```

Note how the use of scaling makes the specification of weights unnecessary. The scaling of attributes leads to different results, as can be seen by calling the resulting object:

```
    my_ASW_sc
```

```
  team          fl.value mbr_to_subgroups number_of_subgroups
1    1 0.337558681138197      1 2 1 1 2 1                   2
2    2 0.821859114925443      1 1 2 2 3 3                   3
  subgroup_sizes
1            4 2
2          2 2 2
```

Ego-Faultlines

Version 2.0 of the *asw.cluster* package introduces a new option to determinate faultlines on the level of individual group members. As the classic conceptualization of faultlines takes a birds-eye perspective on latent subgroups in a team, by maximizing the silhouette width measures of its members *on average*, the member to subgroup classification may not correspons to every single memeber's individual perspective on the group. Determining Ego-Faultlines with the *faultlines()*-function means to take the individual in- and outgroup perspective of every single member by maximizing his/her partiular silhouette width. This is done for every group member, which means that the result consist of as many faultlines (i.e. subgroup partitions and faultline strength values) as there are members in the group. These faultlines can be congruent, but typically different individual perspectives lead to different faultlines. The *faultline()*-function calculates the individual-level faultline strength and the individual member to subgroup association, as well as an aggregated measure of individual faultline strengths on the team-level.

All results are recalled by the *summary()*-function.

```
    my_ego_ASW <- faultlines(data = teamdata_sub,
                             group.par = "teamid",
                             i.level = TRUE,
                             attr.type = my_attr,
                             method = "asw")
```

```
   summary(my_ego_ASW)
```

```
Number of Teams: 2

Calculation features:
    Method:      ASW
    Level:   individual
    Metric:      euclid


Team 1 (1):
===========
Faultline Strength:
[1] 0.5530582

Individual Faultline Strengths (silhouette widths):
[1] 0.7051077 0.3681231 0.6794982 0.3933564 0.4917633 0.6805004

Member to Subgroup Association:
  X1 X2 X3 X4 X5 X6
1  1  2  1  2  2  2
2  2  1  2  2  1  2
3  1  2  1  1  1  1
4  1  2  1  1  1  1
5  1  1  1  1  1  2
6  1  2  2  2  2  1

Number of Subgroups:
[1] 2 2 2 2 2 2

Subgroup Network:
         X1        X2        X3        X4        X5        X6
1 1.0000000 0.1666667 0.8333333 0.6666667 0.5000000 0.6666667
2 0.1666667 1.0000000 0.3333333 0.5000000 0.6666667 0.1666667
3 0.8333333 0.3333333 1.0000000 0.8333333 0.6666667 0.5000000
4 0.6666667 0.5000000 0.8333333 1.0000000 0.8333333 0.6666667
5 0.5000000 0.6666667 0.6666667 0.8333333 1.0000000 0.5000000
6 0.6666667 0.1666667 0.5000000 0.6666667 0.5000000 1.0000000
```

```
Distances:
         X1       X2        X3        X4        X5        X6
1  0.000000 26.03843  4.000000 11.045361 11.090537  6.082763
2 26.038433  0.00000 22.045408 15.066519 15.033296 32.015621
3  4.000000 22.04541  0.000000  7.071068  7.141428 10.049876
4 11.045361 15.06652  7.071068  0.000000  1.414214 17.029386
5 11.090537 15.03330  7.141428  1.414214  0.000000 17.058722
6  6.082763 32.01562 10.049876 17.029386 17.058722  0.000000


Team 2 (2):
===========
Faultline Strength:
[1] 0.8329077

Individual Faultline Strengths (silhouette widths):
[1] 0.9604632 0.9588393 0.8100979 0.8101001 0.7649801 0.6929656

Member to Subgroup Association:
  X1 X2 X3 X4 X5 X6
1  1  1  2  2  2  2
2  1  1  2  2  2  2
3  2  2  1  1  2  2
4  2  2  1  1  2  2
5  2  2  2  2  1  1
6  2  2  2  2  1  1

Number of Subgroups:
[1] 2 2 2 2 2 2

Subgroup Network:
         X1        X2        X3        X4        X5        X6
1 1.0000000 1.0000000 0.3333333 0.3333333 0.3333333 0.3333333
2 1.0000000 1.0000000 0.3333333 0.3333333 0.3333333 0.3333333
3 0.3333333 0.3333333 1.0000000 1.0000000 0.3333333 0.3333333
4 0.3333333 0.3333333 1.0000000 1.0000000 0.3333333 0.3333333
5 0.3333333 0.3333333 0.3333333 0.3333333 1.0000000 1.0000000
6 0.3333333 0.3333333 0.3333333 0.3333333 1.0000000 1.0000000

Distances:
        X1       X2       X3        X4       X5        X6
1  0.00000  1.00000 17.05872 20.049938 35.02856 29.034462
2  1.00000  0.00000 16.06238 19.052559 34.02940 28.035692
3 17.05872 16.06238  0.00000  3.000000 18.02776 12.041595
4 20.04994 19.05256  3.00000  0.000000 15.03330  9.055385
```

```
5 35.02856 34.02940 18.02776 15.033296  0.00000  6.000000
6 29.03446 28.03569 12.04159  9.055385  6.00000  0.000000
```

Change log since Version 1.1

Version 1.101:. Some minor bugfixes Columns in the $long format are now convertet to “numeric” values instead of factors. Van Knippenbergs measure can now be calculated for datasets with only two attributes

Version 1.102:. Changed asw_cluster.agglomerative function to return a fl-value of zero instead of “NA” for completely homogeneous teams

Version 1.103i:. Individual-Level Faultlines

Version 1.104ip:. Parallel processing option. Specify “cores=”. Applies to datasets that include multiple groups, separated by the variable specified by the “group.par”-parameter. Output is omitted in parallel mode. Note that R may be “freezed” during parallel calculations.

Version 1.105ip:. FL-level (team/individual) as new option in silhouette width function

Version 1.2:. New silhouette.with algorithm which takes responds to subgroup heterogenity New function “s.outliers” to detect outlier-members New function ingroups.ind which detects individual member’s ingroups

Version 1.21:. Extended output of summary()$long for individual Faultlines

Version 1.23:. Modified asw.cluster routine for increased efficiency

Version 1.3:. Added option (usesghomo) to calculate silhouette widths that are sensitive to subgroup homogeneity

Version 1.4:. Added option (by.attr) to calculate silhouette widths separately for each attribute and apply weighting factors afterwards. This enables non-variance attributes to reduce faultline-strengths. Added “gower”-Metric, which uses Manhattan Distances instead of Euclidean Distances

Version 1.5:. Modified silhouette.width calculation for by.attr option to correctly use attribute-relevance (weights) Added by.attr calculation for individual mode

Version 1.51:. Bug fixes in individual mode with by.attr=T

Version 1.52:. when determining individual faultlines, procedure does not stop at local maximum anymore, but goes through entire team

Version 1.53:. “Demo”-Mode for particular methods (ASW,. . .) based on Version 1.52 to demonstrate calculation process

Version 1.54:. by.attr-mode implemented for individual-level faultlines

Version 1.55:. individual faultline calculation: if “by.attr” option is set to TRUE, attributes are standardized by their range and Euclidean distances between individuals are replaced by the weighted average of singe-attribute distances.#

Version 1.56:. Small change in ingroup.ind procedure that causes group members to be included in the focal member’s ingroup during ingroup growing phase, even if they don’t cause an immediate increase of the focal member’s individual silhouette width by themselves.

Version 1.57:. outliers calculated based on percentile of distances between group members (in this version only - discontinued)

Version 1.58:. “All-zero”-dummyvariables removed from faultline calculation

Version 1.581:. Minor bug fix in faultlines.default

Version 2.0:. Published Version

Version 2.01:. Bug Fixes in fuctions fau.dist.mahal, faultlines.calc and fau.numerator.mahal Individual faultline strength now included in summary()-output

References

Bezrukova, K., Jehn, K. A., Zanutto, E. L., & Thatcher, S. M. B. (2009). Do workgroup faultlines help or hurt? A moderated model of faultlines, team identification, and group performance. *Organization Science*, *20*, 35–50. doi:10.1287/orsc.1080.0379

Gibson, C., & Vermeulen, F. (2003). A healthy divide: Subgroups as a stimulus for team learning behavior. *Administrative Science Quarterly*, *48*, 202–239. doi:10.2307/3556657

Meyer, B., & Glenz, A. (2013). Team faultline measures: A computational comparison and a new approach to multiple subgroups. *Organizational Research Methods*, *16*(3), 393–424.

R Development Core Team. (2011). R: A language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing; Computer software. Retrieved from http://www.R-project.org

Shaw, J. (2004). The development and analysis of a measure of group faultlines. *Organizational Research Methods*, *7*, 66–100. doi:10.1177/1094428103259562

Thatcher, S., Jehn, K., & Zanutto, E. (2003). Cracks in diversity research: The effects of diversity faultlines on conflict and performance. *Group Decision and Negotiation*, *12*, 217–241. doi:10.1023/A:1023325406946

Trezzini, B. (2008). Probing the group faultline concept: An evaluation of measures of patterned multi-dimensional group diversity. *Quality and Quantity*, *42*, 339–368. doi:10.1007/s11135-006-9049-z

van Knippenberg, D., Dawson, J., West, M., & Homan, A. (2011). Diversity faultlines, shared objectives, and top management team performance. *Human Relations*, *64*, 307–336. doi:10.1177/0018726710378384

Zanutto, E., Bezrukova, K., & Jehn, K. (2010). Revisiting faultline conceptualization: Measuring faultline strength and distance. *Quality and Quantity*, 1–14. doi:10.1007/s11135-009-9299-7