Novel methods for estimating homogeneous clusters and underlying processes: combining independent component analysis and cluster analysis with applications to neuroimaging data

Durieux, J.

Chapter 5

Multi-CHull for multiverse analysis

Studying model sensitivity by applying CHull with multiple fit or complexity values

Submitted as: Durieux, J., Vervloet, M., Ceulemans, E., & Wilderjans, T. F. (2024). Multi-CHull for multiverse analysis: Studying model sensitivity by applying CHull with multiple fit or complexity values.

5.1 Abstract

The CHull method, which was proposed by Ceulemans and Kiers (2006), is a generic procedure for model selection that identifies the model that optimally balances model fit and complexity; MATLAB software for CHull was provided by Wilderjans, Ceulemans, and Meers (2013). Several choices that typically have to be made before and/or during the analysis (e.g., pre-processing strategy, choice of model complexity definition) will have an influence on which model is optimal. It is good practice for users to critically evaluate their choices made in this regard and to investigate the

effect of these choices on the selected optimal model. To this end, users should run a Multi-CHull analysis in which CHull is applied several times (e.g., determining the optimal model for each pre-processing strategy). This approach of performing multiple analyses, herewith using different settings each time such that effects or outcomes of these choices can be evaluated, aligns with the concept of multiverse analysis (Steegen, Tuerlinckx, Gelman, & Vanpaemel, 2016). To aid the user in this, we translated the CHull software into an R package and extended it with functions for a Multi-CHull analysis. In this paper, we, therefore, present the open source **multichull** R package which can be installed from CRAN and an associated RShiny App `https://multichullapp.shinyapps.io/multichull/` to assist the user in applying the Multi-CHull procedure. The functionality of the package is demonstrated with three empirical data analysis applications.

5.2 Introduction

In the behavioral sciences, choosing the appropriate model to suit the available data is a common difficulty that arises throughout the modeling process. In psychology, for instance, a researcher may give a questionnaire to a sample of participants in order to study the dimensionality of a latent construct (such as anxiety). Then, exploratory factor analysis (EFA) or principal component analysis (PCA) can be used to analyze the dimensionality of this construct. For these kinds of techniques, one must solve a model selection problem by determining the optimal number of factors or components to retain.

Numerous model selection techniques have been proposed over the years with some of these techniques being very specific for a particular model selection problem at hand and others being more general in the sense that they can be used for various types of models. Regarding the former, parallel analysis (Horn, 1965) is a method for model selection that aims at determining the optimal number of factors (for EFA) or components (for PCA) underlying an empirical data set. Other frequently used examples of specific model selection methods that can be used to determine the number of clusters underlying a data set at hand include the silhouette statistic (Kaufman & Rousseeuw, 1990), the CH statistic

(Caliński & Harabasz, 1974) and the gap statistic (Tibshirani et al., 2001). Regarding more general model selection criteria, the Akaike Information Criterion (AIC; Akaike, 1974), the Bayesian Information Criterion (BIC; Schwarz, 1978) and the Minimum Description Length criteria (MDL; Rissanen, 1978) are a few well-known examples of information theory-based model selection approaches that are often used for divers kinds of regression models (i.e., linear, logistic, survival, multilevel extensions) but also for, for example, SEM models. A drawback of these latter criteria is that they can only be applied to stochastic models, which requires some kind of distributional assumption(s) regarding the data and/or the noise therein.

As an alternative to these (stochastic) criteria, CHull was proposed by Ceulemans and Kiers (2006) as a general and broadly applicable procedure for model selection. CHull can be used, for example, for selecting the best subset of predictors in a regression analysis, determining the number of components/clusters underlying a data set and choosing an appropriate level of regularization in penalized regression (for several applications, see Wilderjans, Ceulemans, & Meers, 2013). In the context of mixtures-of-factor-analyzers and additive profile clustering (ADPROCLUS), simulation experiments demonstrated that CHull performs favorably compared to other commonly adopted model selection criteria like the AIC (Bulteel, Wilderjans, Tuerlinckx, & Ceulemans, 2013; Rossbroich et al., 2022).

The main idea behind CHull is to automate the visual inspection-based elbow searching method of the scree test proposed by Cattell (1966). In this method, a scree plot (i.e., a plot with model complexity on the x-axis and fit or misfit on the y-axis) is visually searched for a model on (or near) an elbow. CHull automates this subjective visual inspection method by making it numerically (see Section 5.1). Therefore, CHull can also be used as a model selection method in scenarios where numerous analyses are carried out, such as in simulation studies.

An attractive feature of CHull is that it can be used in many contexts as it only requires that for each model under consideration a sensible complexity and (mis)fit measure is defined. This is an interesting feature of CHull as it is not always immediately clear which model fit and/or model complexity measure should be chosen to determine the optimal model among a set of candidate models. For example, the model (mis)fit for a regression analysis can be quantified as the sum of squared residuals,

the R-squared or the k-fold cross-validation error, with a different choice in this regard possibly leading to a different selected model that optimally balances model fit and complexity.

Also, for some types of analyses, like for example regularized (e.g., ridge) regression, it is not entirely clear what the complexity of a model is. When no regularization is performed, the number of parameters (i.e., complexity) equals the number of variables (as in ordinary least squares). However, when some regularization is applied in a ridge regression context, all regression coefficients are shrunken towards zero but never will become exactly zero. In this case, the number of non-zero coefficients, which here also equals the number of variables, does not seem to be a valid measure for model complexity.

Another example is multivariate adaptive regression splines (MARS), a regression method that models nonlinearities and interactions between variables and that is well suited for high-dimensional data (Friedman, 1991). In this model, the *effective* number of parameters is based on the number of hinge knots. Some authors, however, suggest that searching for these knots, which requires three degrees of freedom (Owen, 1991; Ye, 1998), should be taken into account when computing the complexity of the fitted model. Also here, the complexity of the model is not well-defined. As a final example, take a clustering method like Clusterwise Independen Component Analysis (C-ICA; Durieux et al., 2022), which can be used for clustering subjects based on fMRI data. In C-ICA, besides the optimal number of clusters, a user also needs to decide on the optimal number of ICA components. Also here it is not obvious how the number of clusters and components should be combined into an optimal quantification of the complexity of a C-ICA model.

When choosing a different measure for model fit and/or model complexity, it can be expected that a different model will be selected by CHull as the optimal one. To investigate the effect of the selected fit/complexity measure on the retained solution, a Multi-CHull analysis is advised in which CHull is applied to each model fit/complexity measure separately. Such a Multi-CHull analysis is also recommended to study the effect of different analysis options on the final selected model. Until now, limited attention has been given to the fact that, when analyzing data, several choices have to be made before and/or during the analysis that may im-

pact the final model retained. Before the actual data analysis, already the preprocessing of the raw data can give rise to a multiverse of reasonable options (e.g., exclusion of data, transformation of variables, centering and/or scaling the variables), and making one -rather arbitrary- choice in this regard can have major consequences for the final conclusions (Steegen et al., 2016). Also, during the analysis, often a variety of options exists that influence the obtained results. For example, when applying factor analysis or structural equation modeling (SEM), multiple estimation procedures to estimate the parameters are available (e.g., maximum likelihood, generalized least squares), with the same being true for Independent Component Analysis (ICA; e.g., optimizing kurtosis, negentropy, cumulants).

Running a Multi-CHull analysis aids the user in studying the effect of the abovementioned choices on the final retained model that optimally balances model fit and complexity and as such is a valuable tool for performing a multiverse analysis (Steegen et al., 2016). No software, however, is publicly available to perform such a Multi-CHull analysis. In this paper, we therefore present a new R package, called **multichull** (Vervloet et al., 2024)[1], which does not only contain the original CHull function (Wilderjans, Ceulemans, & Meers, 2013), but also the Multi-CHull function. The latter function allows users to inspect multiple fit or complexity values per model, resulting from the different analysis options that were discussed earlier. As such, users are encouraged to critically evaluate their choices and to investigate the effect of these choices on the selected optimal model.

The remainder of this paper is structured as follows: First, we will recapitulate the CHull model selection procedure, and explain how it can be applied when several (mis)fit and/or complexity values per model are available. Next, in Section 5.4, we will describe the usage of the **multichull** package and the associated R Shiny web application. Three real-data applications to show the benefits of a Multi-CHull analysis are presented in Section 5.5. Lastly, in Section 5.6, we provide some concluding remarks.

[1] available from CRAN at `http://CRAN.R-project.org/package=multichull`

5.3 CHull method with multiple fit or complexity values per model

5.3.1 Original CHull

Performing CHull model selection (Ceulemans & Kiers, 2006) requires the calculation of the complexity (c) and fit (f) for each model under consideration. The fit measure (f) can either indicate goodness-of-fit (e.g., the variance in the data that is accounted for by the model) or misfit (e.g., the sum of squared residuals). The complexity measure (c) could, for example, refer to the number of parameters, but also to the number of factors or components in a dimension reduction method or a penalty value. Most of all, it is important that the fit and complexity measure that is used is calculated in the same way for each of the candidate models.

In particular, the CHull method consists of the following seven steps (see Ceulemans & Kiers, 2006; Wilderjans, Ceulemans, & Meers, 2013):

1. For each level of complexity (c), retain the model with the largest fit (or lowest misfit, depending on the type of fit measure f). If two or more models with the same complexity have the same (mis)fit, one of these models is retained at random.

2. Order the models m_i (i=1,...,n), based on their complexity value c_i, from most simple (lowest c_i) too most complex (largest c_i).

3. For all pairs of adjacent models, exclude the more complex model (larger c_i) when its goodness-of-fit is equal or smaller than the fit of a more simple model (smaller c_i). In the case of misfit, a more complex model is excluded when the misfit is equal to or larger than the misfit of a less complex model. This step is repeated until no more models can be excluded and will result in a set of models where the fit values are monotonically increasing (or decreasing in the case of misfit).

4. Identify models that lie on the convex hull. In particular, for all triplets of adjacent models (m_i,m_j,m_k), exclude model m_j if this

model is located under (in the case of goodness-of-fit) or above (in the case of misfit) the line that connects the other two models m_i and m_k.

5. As an optional step, to avoid the selection of overly complex models, exclude models with a (better) fit that is almost equal to the fit of a model that is less complex (e.g., exclude models for which the improvement in fit compared to a less complex model is less than 1%).

6. For each retained model $\tilde{m}_i$ (i=2, ..., $\widetilde{n-1}$) on the convex hull, compute a scree test value $st_i = \frac{f_i - f_{i-1}}{c_i - c_{i-1}} / \frac{f_{i+1} - f_i}{f_{i+1} - f_i}$. Note that no scree value can be computed for the most simple $\tilde{m}_1$ and most complex model $\tilde{m}_{\tilde{n}}$.

   Select the model with the largest st value. If multiple models have a maximal st value, select the simplest model. Moreover, a user should also check whether for the retained model duplicate models with an equal fit and complexity exist (that were discarded in step 1).

In essence, the steps 1-5 from above identify a set of models that lie on either the upper (in the case of goodness-of-fit) or lower (in the case of misfit) boundary of the convex hull in a scree plot. Note that for some methods, such as PCA, in which each subsequent component explains less variance in the data than the previous extracted components, all fitted models already lie on that boundary. Next, in step 6, a numerical procedure is used that aims at identifying the optimal model with the largest st value. Note that large st values are desirable because this implies that a model with complexity c_i fits the data considerably better than a less complex model (with complexity c_{i-1}), whereas a more complex model (c_{i+1}) hardly improves the (mis)fit of the model (Ceulemans & Kiers, 2006).

As a final note, due to the formula in step 6, CHull cannot select the most simple and the most complex model that lie on the boundary of the convex hull since the st values cannot be computed for these models. It is therefore important to take care of the selection of the set of models that one wants to choose from. In practice this means that a wide enough range of different models (i.e., covering those complexity values one is interested

in) should be fitted and supplied to the CHull procedure. In particular, it is advised to include overly simple but also overly complex models in the analysis.

5.3.2 Multi-CHull

Unless no scree test values (see step 6 in Section 5.3.1) can be computed (i.e., when the boundary of the hull contains less than three models) and unless multiple models have the same complexity and fit value, only a single model is selected by CHull. It is possible that choices made before and/or during the analysis (see Section 5.2) are the determining factor to prefer one model over the other. Therefore, adopting a multiverse analysis, it is informative to run a Multi-CHull analysis, which consists of performing several CHull analyses with varying modeling options (e.g., different preprocessing strategies), and to check how consistent the results of CHull are across these different modeling choices. It is recommended to focus on models that are selected (or that are located on the boundary of the hull) most often across the different CHull analyses.

Note that the results of the CHull procedure are influenced by which fit and complexity measure is used. A good fit and complexity measure should reflect the research question at hand and should be decided upon theoretical grounds. In case of multiple reasonable fit or complexity measures, also here Multi-CHull can be executed by applying CHull to each fit or complexity measure separately. Again, it can be investigated whether the results are consistent across these different measures, and it is recommended to inspect all models that are (often) indicated as optimal.

5.4 Using the multichull package

In this section, we discuss how CHull and Multi-CHull can be applied in R by means of the **multichull** package. First, after installing the package (with the `install.packages()`-command), one loads the **multichull**:

```
library(multichull)
```

To illustrate how to use the **multichull** package, we will make use of a dataset that is also illustrated in Wilderjans, Ceulemans, and Meers (2013). The dataset contains model complexity and (mis)fit values from multiple reduced K-means (RKM; De Soete & Carroll, 1994; Timmerman et al., 2010) analyses applied on the well-known iris data set (Anderson, 1935). To load the dataset chulldata, which contains the complexity and fit values (see Figure 5.1) and which is included in the package, one types

```
data("chulldata", package = "multichull")
```

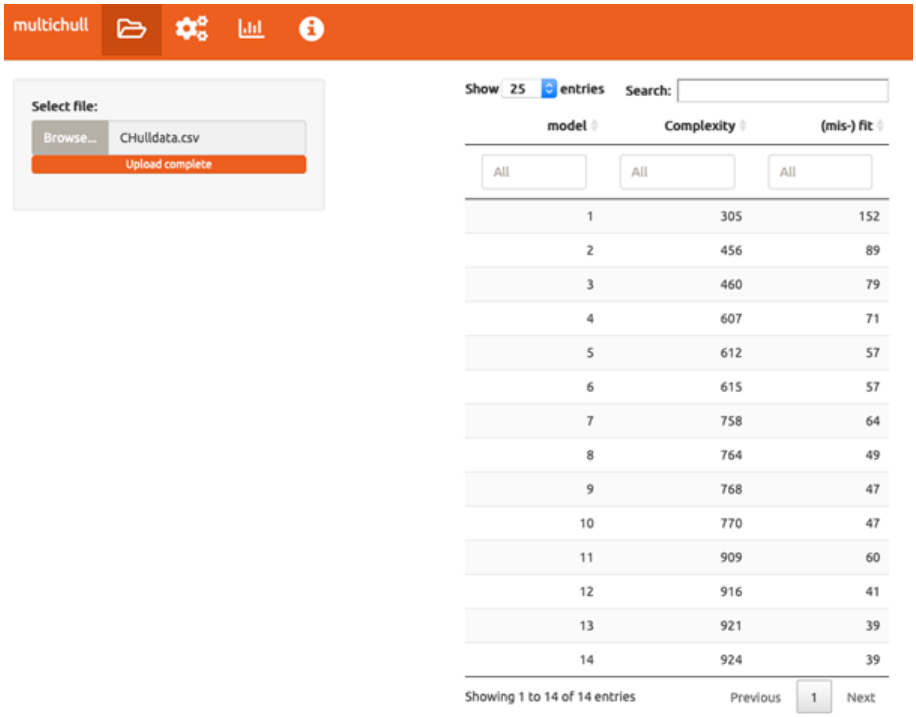

Figure 5.1: Screenshot of the R shiny Multi-CHull web application (also see Section 5.4.3) showing the chulldata dataset. After uploading a comma separated file (i.e., a .csv file), the input data is displayed. In the first column, a number to index the different candidate models, which are ranked in terms of increasing complexity, is given. In the second column, the model complexity values of a Reduced K-means analysis selecting for different number of clusters K and components Q is displayed. In the third column, the associated misfit values (residual sum of squares) for these RKM models applied to the iris data are presented. Note that the .csv-file should only contain the last two columns.

In RKM, the variables are reduced to a (smaller) set of components and objects are simultaneously clustered based on the scores of the objects on these components. The model selection problem therefore pertains to both a selection about the number of components to retain and the number of clusters to select. RKM is fitted to the mean-centered iris data set, which contains morphologic information (i.e., length and width of the sepals and petals) measured on 4 variables for three species of iris flowers with 50 objects per species. Here, RKM models were fitted with two up to six clusters (K) and the number of components (Q) varying for each K, from one to K–1 components. Note that an RKM solution with K or more components (i.e., the same or more components than clusters) fits the data as good as the solution with K–1 components -due to centering (also see, Wilderjans, Ceulemans, & Meers, 2013)- and therefore will never be preferred over the solution with K–1 components. As a misfit measure, the sum of squared residuals was adopted. For this illustration, we selected the complexity measure as the number of fitted parameters, defined here as the $150 \times K$ entries of the partition matrix, the $4 \times Q$ component loadings and the $K \times Q$ cluster centroids, minus the Q^2 parameters that account for the rotational freedom of the components. The data set is shown in Figure 5.1 along with a (mis)fit versus complexity plot in Figure 5.2.

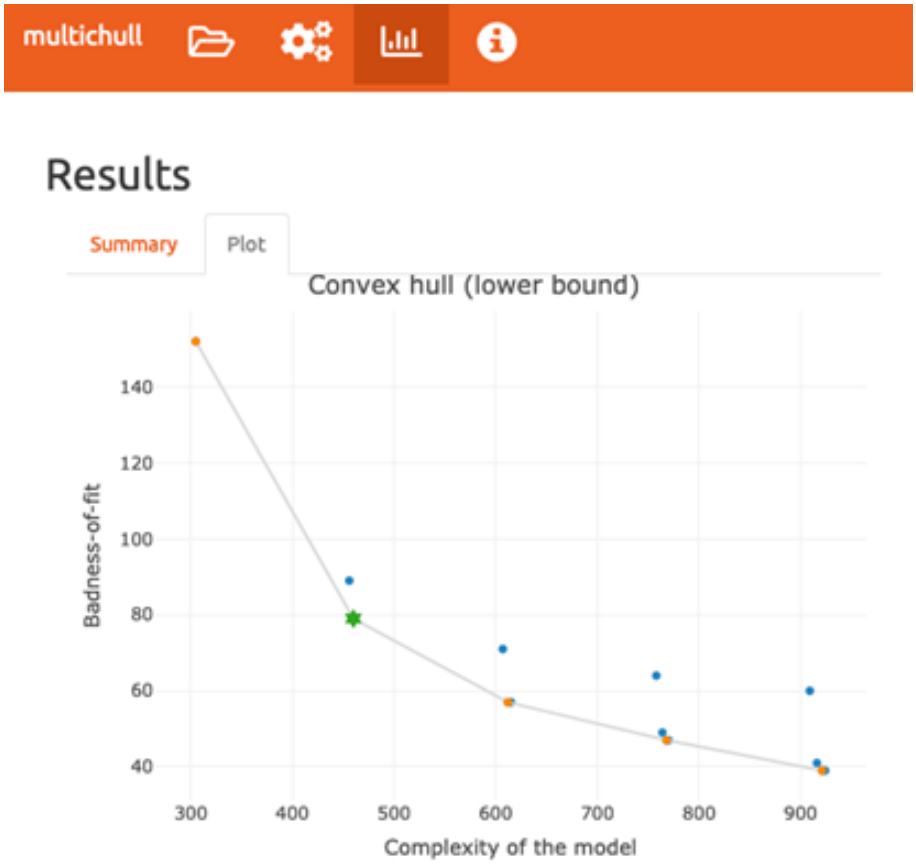

Figure 5.2: Screenshot of the R shiny multichull web application. By selecting the 'Plot' tab in the results menu, an interactive (mis-)fit versus complexity plot is shown. The user can hover over the data points (which represent the candidate models) and the associated model complexity value and (mis-)fit is shown. In this figure, the results of a CHull analysis on the complexity values and misfit values of several Reduced K-means analyses (selecting for different combinations of number of clusters and components) on the iris data is shown. The selected model (on the boundary of the convex hull) is indicated automatically in this plot (here with a green star). Candidate models that are not selected but are lying on the boundary of the hull are indicated with orange dots, whereas the other models are displayed with blue dots.

5.4.1 Function `CHull()`

The `CHull()` function can be called by

```
results <- CHull(data=chulldata , bound = "lower",
  PercentageFit = .01)
```

The input parameters needed for performing a CHull analysis are `data`, `bound` and `PercentageFit`. Similar to the chulldata in the example above, the data should always be a matrix or data frame with two columns. The first column should hold the complexity values of each model under consideration (in the rows) and the second columns the fit values (also see Figure 5.1).

The parameter `bound` indicates whether the fit measure represents goodness-of-fit or badness-of-fit. In case of badness-of-fit, the selected model(s) should be on the lower boundary of the convex hull (`bound="lower"`), and in case of goodness-of-fit on the upper boundary (`bound="upper"`). The default setting is to search for optimal solutions on the lower boundary. A warning is generated if the correlation between complexity and fit values suggests that the parameter bound is specified incorrectly.

After inspecting which of the models lie on the boundary of the convex hull, models with less than (by default) 1% improvement in fit (`PercentageFit = .01`) compared to a less complex model are discarded. This percentage can be freely adjusted through the parameter `PercentageFit`. If `PercentageFit = 0`, only models are discarded for which there is no improvement in fit at all, compared to a less complex model.

To start the analysis with the default settings, one types

```
results <- CHull(chulldata)
```

which will save the output in a list variable, called results here, consisting of:

- `Solution`: a data frame containing the optimal model(s) and its (their) complexity, fit and scree test value(s).

- `Hull`: a data frame containing the models that are located on the upper or lower (depending on how `bound` was specified) boundary of the convex hull, and their complexity, fit and scree test values. Because scree test values cannot be calculated for the most simple and most complex model, the latter models receive an `NA` for the scree test value.

- The input values for the input parameters `OrigData` (corresponding with data), `bound` and `PercentageFit`.

Note that in case of models with identical fit and complexity values, only one of them is shown in `Hull`, because the other model(s) does not have an improvement in fit larger than `PercentageFit`. These models, however, will be shown in `Solution`.

Also, S3-methods `plot`, `summary` and `print` are defined in the package multichull for objects produced by CHull. To plot, summarize or print the output of `CHull`, respectively, one can type:

```
plot(results, plottype = "static")
summary(results)
print(results)
```

The plot function creates a static graph by default (i.e., plottype = "static") or an interactive graph (i.e., plottype = "interactive") with the model complexities on the x-axis and the fit values on the y-axis (like Figure 5.2). The models lying on the boundary (orange dots) are connected by a dashed line; the optimal model (here, a model with $K = 3$ clusters and $Q = 2$ components) is clearly indicated (with a green star). Note that this graph is interactive and that information about the model number is shown to the user by hovering a cursor over the data points.

Both the print and summary function display the selected model(s) -along with its associated complexity, fit and scree test value(s)- and the models that lie on the boundary of the convex hull to the console. In addition to that, the summary function prints out the complexity and fit measure of all original models that were included in the analysis.

5.4.2 Function `MultiCHull()`

`MultiCHull()` applies the `CHull()` procedure on either multiple fit values or multiple complexity values. The function can be executed by typing:

```
results <- MultiCHull(data=chulldataboot, bound=
  "lower", PercentageFit = .01, type = "multifit")
```

When `type = "multifit"`, the input parameter data should consist of a data frame (or matrix) with complexity values in the first column and multiple (mis)fit values in the next columns. Note that when a `"multifit"` type of analysis is performed, the first column of the data should be named `"comp"`. The different columns of fit values can for example refer to the fit values obtained by applying different pre-processing strategies (see Section 5.5.2), estimating the model on different bootstrap samples generated from the same data sample (see Section 5.5.1), seeding the algorithm with different start configurations (e.g., random vs. rational vs. semi-rational starts; see, for example, Ceulemans et al., 2007) or to the values computed from different fit measures (e.g., sum of squares, proportion explained). However, when multiple complexity measures (see Section 5.5.3) are being analyzed (`type = "multicom"`), the input parameter `data` should be a data frame (or matrix) where the first columns refer to different complexity measures/definitions and the last column to the associated model (mis)fit. For this type of analysis (i.e., `"multicom"`), the last column of the data should be named either `"fit"` or `"misfit"`[2]. The argument type specifies whether multiple fit values (`"multifit"`) or multiple complexity measures (`"multicom"`) are analyzed. Note that it is not possible to run a Multi-CHull analysis with multiple fit and complexity values at the same time. An example dataset is included in the multichull package and can be loaded by:

```
data(chulldataboot, package="multichull").
```

[2] In order to change the name of the first column, the following R command can be used: `names(data)[1] <- "comp"`. The following command can be used to change the last column name: `names(data)[ncol(data)] <- "misfit"`.

This dataset contains multiple fit values for RKM analyses (for multiple values of the number of components Q and clusters K) performed on bootstrapped iris data sets, using 100 bootstrap samples. The sum of squared residuals for each bootstrap sample is used as fit measure. Also, as discussed in Section 5.4.1, the input parameters `bound` and `PercentageFit` can be altered (the default options equal the default options for `CHull()`).

The output when `"multifit"` is provided as an argument, stored as results in our example, consists of:

- `st`: a data frame, holding the scree test values per specified fit value (i.e., column in the input data) of the solutions (e.g., different bootstrap samples) that were found on the upper or lower boundary of the hull (depending on how `bound` was specified). For each specified fit value, the least and most complex model on the boundary is indicated by an `NA` value as no scree test value for these models can be computed and these models thus cannot be selected by CHull. Also for solutions not located on the boundary of the hull the scree value is `NA`.
- `tab`: a data frame which indicates, for each specified fit value, the top three optimal models in terms of the scree test value (indicated by 1, 2 and 3, with 1 being the optimal model). The other models get an `NA` value.
- `frq`: the frequency for each model of being selected as the optimal model.
- The input value for the input parameters `OrigData` (corresponding with `data`), `bound` and `PercentageFit`.

The output of MultiCHull when `type = "multicom"` is specified, simply consists of a list where each element is a `CHull` output (see Section 5.4.1), with the number of elements of the list equaling the number of different complexity measures (i.e., columns from the input data set) considered. Consequently, all `CHull` associated S3 methods can be performed on each element of that list separately. Note, however, that a useful S3 plot method

can be performed on the `"multicom"` output directly. Here a shiny app is started where different CHull plots, with each plot being associated with a different complexity definition, can be viewed interactively (see Section 5.5.3 for an example).

Applying the method `plot` on output of `MultiCHull` using `type = "multifit"` yields, as can be seen in Figure 5.3, an interactive scatter plot in which per specified fit value the top three models are shown, with the models on the x-axis -ordered by increasing complexity- and the scree test values on the y-axis. The top three models per specified fit value are shown in different colors and the most frequently selected model is indicated by a vertical grey line. Note that when the number of models is smaller than or equal to 20, the model tick labels are printed on the x-axis. When it is larger than 20, no tick labels are printed to avoid clutter on the x-axis. The model number of a data point can be viewed, however, by hovering a cursor over a particular data point. Similarly as for the `CHull` function, the output of `MultiCHull` also has a `summary` and `print` method. Here, information about the selected model(s) is printed to the console. For example, for each selected model a percentage is printed that indicates how many times that model is selected out of the total number of specified fit values.

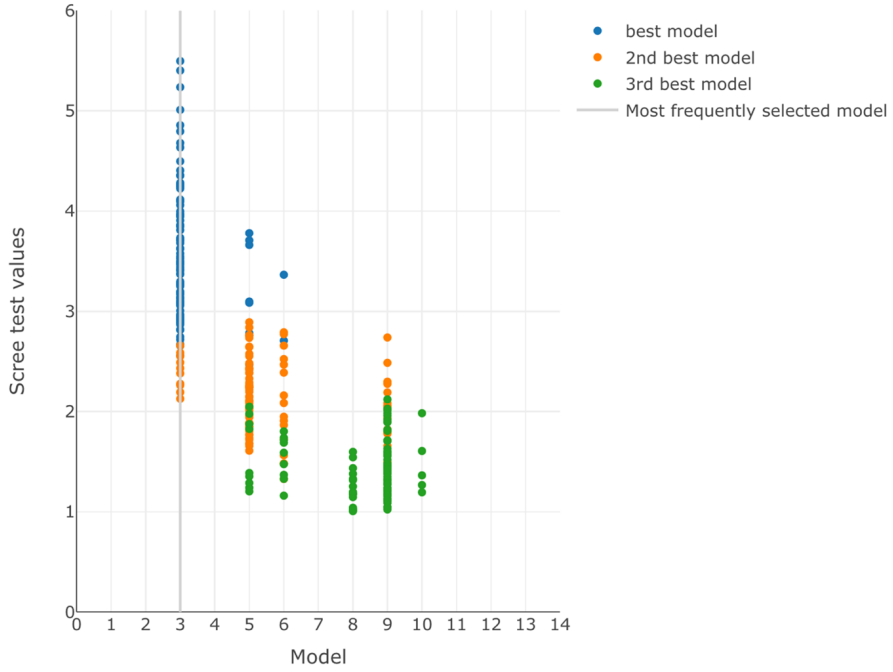

Figure 5.3: Interactive scatter plot of the results of applying Multi-CHull to the complexity and multiple fit measures of the Reduced K-means analyses on the bootstrapped iris data. Here, per specified fit measure (i.e., bootstrap sample), the top three models are shown, with the models on the x-axis (ordered by increasing complexity) and the scree test values on the y-axis. The top three models per specified fit measure are shown in different colors and the most frequently selected model is indicated by a vertical grey line.

5.4.3 R shiny app

In order to facilitate the use of `CHull` and `MultiCHull`, a user-friendly R Shiny web application is provided that is able to apply all main functions of the package using an interactive graphical user interface (see `https://multichullapp.shinyapps.io/multichull/`). The application consists of three main parts, which are indicated by three different icons (see Figure 5.1 and 5.2): (1) data uploading indicated by an icon of a folder, (2) analysis options indicated by the gears icon, and (3) analysis output indicated by a graph icon.

Data uploading. The application requests a comma separated file (.csv)[3], which can simply be uploaded by clicking the 'Browse...' button in the 'Select file' box. After clicking the button, a window will show up that opens Finder (on mac operating systems) or File Explorer (on Windows operating systems). After uploading the file, a data table is displayed, and the user can visually inspect whether the correct data is uploaded. Note that the first column refers to the model identification number; this column is automatically added by the web application and should not be included in the CSV file. The CSV file, however, should contain a header line with column names. Models can easily be looked up by using the search toolbar in the upper right corner.

Analysis options. In Figure 5.4, the analysis options of `CHull` and `MultiCHull` of the web application are shown. By selecting the appropriate tab panel (`CHull` or `MultiCHull`), the analysis options of the corresponding function are displayed. Here the aforementioned function arguments (see Section 5.4.1 and 5.4.2) can be selected and the analysis is started after clicking the grey button 'Start Multi-Chull'.

[3] Two example datasets in .csv format are available via: `https://osf.io/4n7tr/`

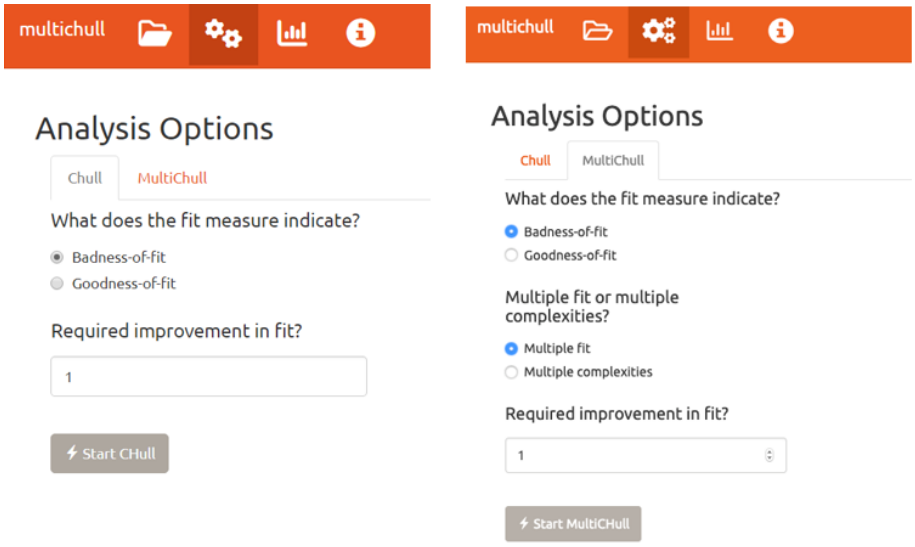

Figure 5.4: CHull (left panel) and Multi-CHull (right panel) analysis option menu in the R shiny app.

Analysis output. An example of the analysis output screen of the application is shown in Figure 5.2 (for `CHull`) and Figure 5.3 (for `MultiCHull` with different specified fit measures). The output consists of a tab panel, either summary or plot. By selecting the summary tab panel, the output of the S3 summary methods of `CHull` or `MultiCHull` (depending on the requested analysis method) is displayed. Similarly, for the plot tab panel, the interactive plot of the `CHull` or `MultiCHull` output is displayed. Here a user can inspect the selected model given by `CHull` (Figure 5.2) or, when `MultiCHull` is conducted, a scatter plot where the top three selected models are shown (Figure 5.3).

5.5 Applications

The goal of this section is to illustrate by means of three empirical applications, the usefulness and working of a Multi-CHull analysis. First, we demonstrate the `MultiCHull` function in the context of a Tucker3 analysis

(Tucker, 1966) that is performed on the ciders data which are available[4] in the R package **ClustVarLV** (Vigneau, Chen, & Qannari, 2015). The `ciders` data, which is a three-way dataset, contains scores on ten different ciders for ten sensory attributes (e.g., fruitiness, odor strength, bitterness) rated by seven different judges (Ledauphin, Hanafi, & Qannari, 2006). The goal of a Tucker3 analysis, which can be considered as an extension of PCA to three-way data, is to decompose the data into a relatively small number of components for each of the three modes (i.e., ciders, sensory attributes, and judges) and a core array that defines how components from each mode are linked to each other. Note that, contrary to PCA, Tucker3 allows the number of components to differ across the modes. An important step in Tucker3 analysis -and many other decomposition methods- is to determine the optimal number of components for each mode to use and to this end a model selection procedure is often employed.

In order to showcase the usefulness of `MultiCHull`, we study how the Tucker3 model selection results are influenced by two data analytic choices that a researcher needs to consider when conducting a Tucker3 analysis. The first choice is whether or not bootstrapping the data (i.e., a post-processing step) in order to check the stability of the results and/or to compute confidence intervals for the component loadings of the three modes (Kiers, 2004). The second choice refers to centering and/or scaling the data, which is a pre-processing step (Bro & Smilde, 2003). These two data analytic choices were studied because they differ in terms of the expected effect of these choices on the obtained results for model selection (i.e., determining the optimal number of components for each of the three modes). It can be expected that the Multi-CHull (model selection) procedure applied to fit values obtained from Tucker3 analyses on different bootstrap samples will most often select the same model as being optimal (i.e., bootstrapping the data should have only a little effect on which model is optimal). As such, the bootstrap in combination with Multi-CHull yields insight into the stability of the Tucker3 model selection procedure. However, applying the Multi-CHull procedure to fit values produced by Tucker3 on differently preprocessed data is predicted to result in a different selected model depending on how the data are preprocessed. Indeed, centering or not centering across one of the modes of

[4] the data can be loaded using `data("ciders", package = "ClustVarLV")`.

the data may result in a different selected (optimal) model with a different number of components for the three modes.

As a third empirical study, we demonstrate the application of the Multi-CHull function to the results of a Clusterwise Independent Component Analysis (Durieux et al., 2022). The goal of C-ICA is to simultaneously cluster subjects based on similarities and differences in the (cluster-specific) independent components underlying the data. This method is developed in the context of analyzing resting-state functional magnetic resonance imaging (rs-fMRI) data. By applying this method, researchers can identify homogenous subgroups of patients and can study differences in brain connectivity networks (i.e., spatially independent components) -also called resting state networks (RSNs)- across patient clusters in an exploratory fashion. The model selection problem for C-ICA is that one has to decide on both the number of clusters and components to retrieve. However, quantifying the complexity of a C-ICA model can be done in several ways and by applying the Multi-CHull procedure on different definitions of the C-ICA complexity users can easily inspect the influence of using a different model complexity definition on the model selection procedure outcome. For this application, we will use rs-fMRI data of 20 patients suffering from (different types of) dementia. Note that the data that were used for the three empirical Multi-CHull analyses are available via `https://osf.io/4n7tr/`.

5.5.1 Bootstrap resampling of a Tucker3 analyis (multiple fit measures)

For the first example, we took 1000 bootstrap samples from the ciders dataset (Kiers, 2004) by resampling with replacement cases from the first mode (i.e., the ciders). For each of the 1000 bootstrapped three-way arrays, a Tucker3 analysis was conducted selecting for all possible combinations of the number of components (i.e., selecting for 1 up to 10, 10 and 7 components for the cider, sensory attribute and judge mode, respec-

tively), resulting in a total of 465 different possible models[5]. The Tucker3 analyses were conducted using the `T3func()` from the **ThreeWay** package (Giordani, Kiers, & Del Ferraro, 2014).

As a complexity measure, we used the number of free parameters of a Tucker3 model (Weesie & Van Houwelingen, 1983; Ceulemans & Kiers, 2006):

$$IP + JQ + KR + PQR - P^2 - Q^2 - R^2 \tag{5.1}$$

where I is the number of ciders (10), J the number of sensory attributes (10), K the number of judges (7) and P, Q and R refer to the number of components for each mode respectively. As fit measure, we used the goodness-of-fit percentages obtained from the Tucker3 analyses.

Multi-CHull results for the bootstrap application.

The `MultiCHull` function was applied to the fit values of the Tucker3 analyses on the bootstrap samples. As each bootstrap sample yields different fit values (but the same complexity values), `type` was set to `"multifit"`. Since goodness-of-fit values were used, the bound argument was set to `"upper"`; percentage fit was set to 1%.

```
output <- MultiCHull(data = bootstrapdata, bound =
  "upper", PercentageFit = .01, type = "multifit")
```

In Figure 5.5, the results of the Multi-CHull analysis are shown. As can be seen by the blue data points in the plot, for a relatively large number of bootstrap samples (about 80%) model 2 with a relatively low complexity (with P=2, Q=2, R=1) is selected. The advantage of using an interactive

[5] Note that we only considered models where the number of components in one mode is smaller than or equal to the product of the number of components in the other two modes. That is, we applied the minimum product rule, selecting only models with $R \leq PQ$, $P \leq QR$ and $Q \leq RP$ (Kroonenberg, 2008a). The rationale for this is that a model with $R > PQ$ components will not fit the data better than a model with $R = PQ$ components. As the former model is more complex than the latter, it will never lie on the boundary of the convex hull and thus never will be selected by a (Multi-)CHull analysis. A similar rationale holds for models with $P \leq QR$ and $Q \leq RP$.

plot is that the user can zoom in to a specific part of the graph. By zooming in to the area to the left one can see that the most frequently selected model is model 2. Another option is to move the mouse cursor over the data points on the grey line (i.e., most frequently selected model) and the model name will be showed.

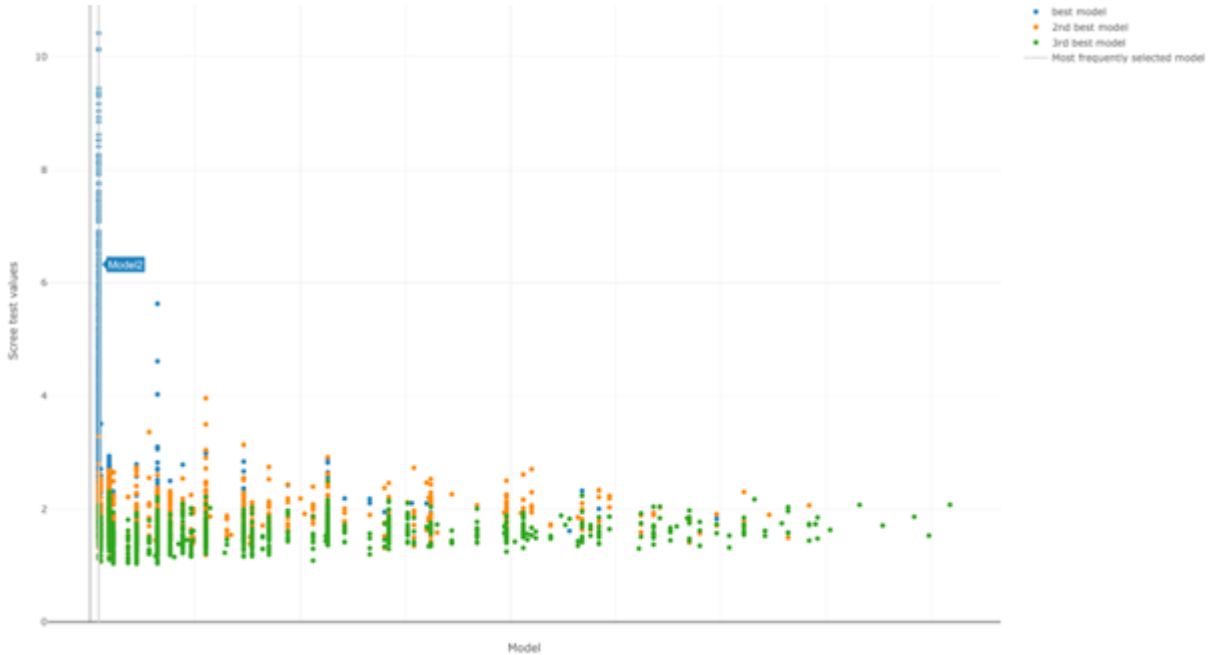

Figure 5.5: Result of applying the S3 plot method on the output of the Multi-CHull function for the bootstrap application (with multiple specified fit measures). On the x-axis all considered models (ordered in terms of complexity) are presented and on the y-axis the corresponding scree test values. The top three models per specified fit measure are shown in different colors and the most frequently selected model is indicated by a vertical grey line. By hovering the cursor over a data point, the associated model number is highlighted.

In order to view the top 5 most frequently selected models, the following R code can be used:

```
top5 <- order(output$frq, decreasing = TRUE)[1:5]
```

As can be seen from Table 5.1, across the 1000 bootstrap samples, the top 5 most frequently retained models are selected 80.9% (with P=2, Q=2, R=1), 11.6%, 1.9%, 1.0% and .8% of the times. In total, one of these 5 models is retained as the best model in about 96% of the bootstrap samples.

Table 5.1: Top 5 selected models from the Multi-CHull analysis for the bootstrap application (across 1000 bootstrap samples)

Rank	Percentage	Complexity	P	Q	R	GOF range
1	80.9%	42	2	2	1	83.80 - 91.38
2	11.6%	57	3	3	1	86.35 - 91.66
3	1.9%	79	3	4	2	88.93 - 94.52
4	1%	70	3	3	2	87.71 - 93.45
5	0.8%	50	2	2	2	84.35 - 91.92

Note. P, Q and R indicate the number of components for the cider, attribute, and judge mode, respectively. GOF range indicate the range of Goodness-of-Fit values.

As can be expected for this bootstrap example, the same model was selected in a large majority (about 80%) of the bootstrap samples. This was a model with a (low) complexity value of 42, containing 2 components for the cider and attribute mode and 1 component for the judge mode. This indicates the strong stability of the selected model across bootstrap samples.

5.5.2 Preprocessing data before a Tucker3 analysis (multiple fit measures)

An important step in many data analysis applications is the pre-processing of the data. If not taken care of properly, unintended results can occur.

If, for example, a dataset is not properly mean-centered before applying PCA, the first principal component will most likely reflect the (variance in) variable means as differences in these means capture a lot of variance in the data. However, in most cases, the differences in variable means are of no substantive interest to a researcher as a researcher is often only interested in the correlations between variables. Similarly, by not scaling the data, arbitrary differences –due to a mere choice of measurement scale– in the variances of variables within a dataset can have a profound influence on the obtained results. Here, variables with a larger variance may –unintentionally by a researcher– be given more weight in a fitting procedure.

The Multi-CHull procedure can easily aid a researcher in model selection and at the same time provide information on how preprocessing affects model selection. For the current application, we again used the `ciders` dataset and we used different combinations of preprocessing in which the variables were centered and normalized in various ways. We centered the data across each of the modes separately (resulting in three different centering options) and additionally also considered double centering for each possible combination of two of the three modes (resulting in three other centering options). Next, in combination with each of the six centering options, we also applied a normalization within each mode, resulting, as can be seen in Table 5.3, in a total of 18 different preprocessed datasets (i.e., 6 centering times 3 normalization options).

Multi-CHull results for the preprocessing application.

Similar as in the bootstrap application, for each preprocessed dataset, a Tucker3 analysis selecting for each possible combination of number of components (465 different combinations in total) was conducted and the goodness-of-fit and complexity values (computed as before) were stored. This resulted in a `data.frame` object with 465 rows (i.e., different models) and 19 columns (i.e., model complexity as the first column and goodness-of-fit values for each of the 18 datasets in the other columns), which was used as input for the `MultiCHull()` function:

```
1 results <- MultiCHull(data = preprocessingdata,
```

```
  bound = "upper", PercentageFit = .04, type =
  "multifit")
```

Since goodness-of-fit values were used, the `bound` parameter was set to `"upper"`. As multiple fit measures were provided, `type` was set to `"multifit"`. Moreover, for this application the `PercentageFit` parameter was set at 4%, meaning that models that lie on the boundary of the convex hull with less than 4% improvement in fit (compared to a less complex model) are discarded. A choice for 4% was made based on an inspection of the obtained fit values for each preprocessed dataset. Moreover, we were mainly interested in less complex models and setting a too low percentage fit value may cause too complex models to be included in the convex hull (see step 5 in Section 5.3.1)[6].

An overview of the results can easily be given to the user by using the plot function on the output of a `MultiCHull` call:

```
plot(results)
```

[6] Performing the analysis using a `PercentageFit` parameter of 1% resulted in the same first ranked model (i.e, the model with complexity value 42, see Table 5.2). However, an overly complex model (P=9, Q=9, R=6) was the second ranked model, selected in 17% of the analyses.

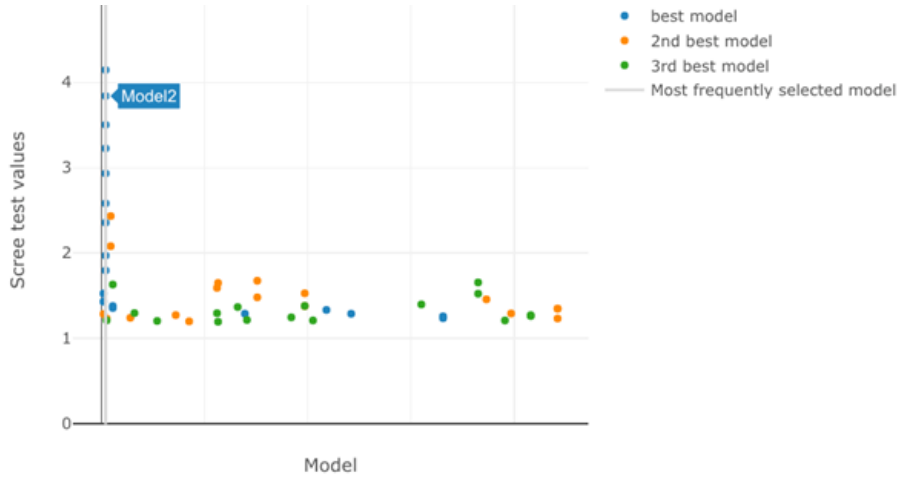

Figure 5.6: Result of applying the S3 plot method on the output of the Multi-CHull function for the preprocessing application (with multiple specified fit measures). On the x-axis all considered models (ordered in terms of complexity) are displayed, with the corresponding scree test values on the y-axis. The top three models per specified fit measure are shown in different colors and the most frequently selected model is indicated by a vertical grey line. By hovering the cursor over a data point, the associated model number is highlighted.

In Figure 5.6, the results of the Multi-CHull analysis for the preprocessed datasets are shown and in Table 5.2, the top 5 selected models are presented. It appears that the model with P=2, Q=2 and R=1 components (with a model complexity value of 42) is selected 50% of the times. Note that this is the same model that is selected in 80% of the bootstrap analysis (in which data were not preprocessed; see Section 5.5.1). For the other 50% of selected solutions, other models with a different complexity are selected as the optimal model. From Table 5.2, one can see that there are four other models that were selected of which three different models were selected in 11.1% of the analyses and one model in 5.6% of the analyses. In Table 5.3 an overview of the selected model for each of the different preprocessed datasets (i.e., combination of -double- centering and normalization) is given. Here, model 2 (with a model complexity of 42), which

is a very simple model, is most often selected when the data are centered across the first (ciders) and/or the second (attributes) mode. When centering or double centering the data across the third (raters) mode, the model selection procedure often selected a model with a relatively large complexity. For example, for a dataset normalized within mode 1 and centered across mode 3 (i.e., the third model presented in Table 5.3), a model with 8, 5 and 5 components for mode 1, 2, and 3 respectively, was selected.

Table 5.2: Top 5 selected models from the Multi-CHull analysis for the preprocessing application

Rank	Percentage	Complexity	P	Q	R	GOF range
1	50%	42	2	2	1	15.28 – 59.88
2	11.1%	39	3	1	2	15.20 – 45.49
2	11.1%	59	2	3	2	22.72 – 62.58
2	11.1%	251	8	5	5	61.40 – 85.42
5	5.6%	169	7	4	4	49.18 – 78.54

Note. P, Q and R indicate the number of components for the cider, attribute, and judge mode, respectively. GOF range indicate the range of Goodness-of-Fit values.

Table 5.3: Selected model per preprocessed dataset

Normalized mode	1						2						3					
Centered Mode	1	2	3	1,2	1,3	2,3	1	2	3	1,2	1,3	2,3	1	2	3	1,2	1,3	2,3
Selected Model #	2	2	256	2	166	256	2	2	16	2	182	16	2	2	11	2	234	11
Component mode 1	2	2	8	2	7	8	2	2	2	2	5	2	2	2	2	2	5	2
Component mode 2	2	2	5	2	4	5	2	2	3	2	6	3	2	2	1	2	3	1
Component mode 3	1	1	5	1	4	5	1	1	2	1	4	2	1	1	2	1	5	2
Model complexity	42	42	251	42	169	251	42	42	59	42	181	59	42	42	39	42	131	39

Note. Centered Mode indicates across which mode the data were centered. When two mode indices are indicated, doubling centering was applied to the dataset.

A possible explanation for these differences in -complexity of the- models selected is the amount of variation that is still present in the data after the pre-processing. When large differences, like, for example, a strong main effect for a certain mode (i.e., large differences between the means of the mode's elements), are preserved in the data, a few strong components only can capture a lot of variance in the data. This will result in a simple model being selected. When the pre-processing, however, removes large differences from the data, also weaker components, together with the strong ones, are needed to explain a large enough portion of the variance in the data. As such, model selection will result in a rather complex model. A three-way analysis of variance applied to the Ciders data shows, as presented in Table 5.4, that there are large differences between the attributes and to a smaller extent between the ciders, whereas almost no variation is contributed to the raters (i.e., trained raters are expected to give stable ratings with a low variation).

Table 5.4: Three-way analysis of variance of the Ciders data after subtraction of the grand mean, with ciders, attributes and raters as fixed factors

Effect	SS	%
Ciders	24.18	1.88
Attributes	394.44	30.66
Raters	6.43	0.5
Ciders × attributes	448.67	34.87
Ciders × raters	29.93	2.33
Attributes × raters	64.90	5.04
Ciders × attributes × raters	318.09	24.72

Note. SS = sum of squares.

Centering across the first (cider) and/or second (attribute) mode keeps the large variation across ciders and/or attributes in the data, and, therefore, a simple model with only a few strong components is selected. Normalization here does not affect the results. Centering across the third (rater) mode, however, results in data with limited variation, necessitating selecting a complex model that also contains weak components. Here, a complex

model is especially needed when normalizing within the cider mode. A simpler model is obtained when normalizing within the rater mode, and, to a lesser extent, within the attribute mode. The same results are observed when centering across the raters and the attribute mode. Finally, centering across raters and ciders, which removes the strong main effect of the attributes from the data, always results in a complex model being identified as optimal, irrespective of the normalization used.

5.5.3 Multi-CHull for multiple complexity values in the context of C-ICA

The previous two applications concerned Multi-CHull analyses on multiple fit values, resulting either from a bootstrap procedure or different data pre-processing options. However, it is also possible to apply multiple CHull analyses using different definitions of model complexity. A researcher can then investigate how different definitions of model complexity affect model selection. In order to showcase this model selection strategy, we analyzed resting state fMRI data concerning patients with Alzheimer's Disease ($n = 11$) and behavior frontotemporal dementia ($n = 9$) using C-ICA (Durieux et al., 2022). Note that these types of data have a three-way structure, that is brain activity (by measuring the release of oxygen in blood) for many voxels (three dimensional pixels or spatial locations in the brain) is measured for a few minutes for multiple subjects, resulting in the data being structured as a subject $\times$ voxels $\times$ time points array. More information about the data is described in Hafkemeijer et al. (2015) and Durieux and Wilderjans (2019).

The goal of C-ICA is to cluster subjects (into R clusters) based on the resting-state patterns present in their data (with Q patterns per cluster). The model formulation of C-ICA is shown in Equation 5.2, with p_{ir} referring to the binary entries of the partitioning matrix $\mathbf{P}$ ($I \times R$), $\mathbf{S}^r$ ($Q \times V$) to a matrix holding the cluster-specific independent components, $\mathbf{A}_i$ ($T \times Q$) to a matrix containing the subject-specific time courses associated to the independent components and $\mathbf{E}_i$ ($T \times V$) to a subject-specific noise matrix. Here, I indicates the number of subjects, R the number of clusters, Q the number of components, V the number of voxels and T the number of time points. In C-ICA, the data $\mathbf{X}_i$ ($T \times V$) are decomposed as:

$$\mathbf{X}_i = \sum_{r=1}^{R} p_{ir} \mathbf{A}_i \mathbf{S}^r + \mathbf{E}_i \tag{5.2}$$

The goal of C-ICA is to cluster subjects into homogeneous groups based on similarities and differences in the estimated resting-state connectivity networks (i.e., spatially independent components stored in the rows of $\mathbf{S}^r$) and associated time courses (i.e., columns of $\mathbf{A}_i$). Hence, a researcher both needs to decide on the number of clusters R and components Q. For the C-ICA model, there is no consensus about the effective number of parameters the model contains given a certain number of components Q and clusters R. One could, for example, argue that $\mathbf{P}$ has $I \times R$ (effective) parameters. However, since $\mathbf{P}$ is restricted to be binary (and each row should sum to one), one could also argue that $\mathbf{P}$ only has I effective parameters (i.e., the number of subjects that have to be clustered).

For this application, C-ICA analyses were performed on the data with R ranging from 1 to 7 and with Q components selecting either for 2, 5, 10, 15, 20, 25 (i.e., $7 \times 6 = 42$ different models in total). C-ICA uses an Alternating Least Squares (ALS) type of algorithm in order to estimate the model parameters. As the C-ICA sums of squares loss function suffers from local minima, we employed a multi-start procedure using 20 random starting partitions (except when $R = 1$). As misfit value, we selected the loss function value associated with the best solution found across the 20 random starts.

For this application, we selected seven different model complexity definitions (see Table 5.5). First, we considered QR as this equals the total number of patterns/components extracted by C-ICA (i.e., Q patterns/-components are extracted for each of the R clusters). Second, as an alternative, we also computed $Q+R$ as complexity value. To extract a single pattern, the loading of each of the V voxels on this pattern should be estimated. As such, QV loadings should be calculated for each of the R clusters, implying QVR loadings in total, which we take as a third complexity measure. We also wanted to include the number of parameters for the clustering of the subjects: I or IR (see earlier). As such, we took $QVR+I$ as fourth and $QVR+IR$ as fifth complexity measure. Finally, we also wanted to consider the subject-specific time courses; these can be obtained by estimating for each of the T time points, the loadings of

the time point on each of the Q components and this for each of the I subjects, resulting in QTI parameters in total. Therefore, we considered $QVR{+}QTI{+}I$ and $QVR{+}QTI{+}IR$ as a sixth and seventh complexity measure, respectively. Note that these last two measures capture the total number of C-ICA parameters estimated.

Multi-CHull results

In order to apply CHull on each of the seven complexity values and associated misfit value the `MultiCHull` function can be used by specifying the `type` argument to `"multicom"`:

```
results <- MultiCHull(data=chulldatacomplexity,
  bound = "lower", PercentageFit = .01, type =
  "multicom")
```

The argument `type = "multicom"` indicates that several `CHull` analyses will be performed on multiple complexity values. As output a list object of class `'multichullcom'` will be returned. Each element of that list contains a CHull analysis and the corresponding S3 methods (plot, summary and print) can be performed on *each element* of that list. However, a S3 summary method is also available for the object `'multichullcom'` (i.e., the list containing results of the `'multicom'` version of `MultiCHull`. Here, the summary method prints out the `CHull` solution (i.e., model name and associated complexity, fit values and *st*-ratio) for each considered complexity definition to the console. In order to facilitate a graphical inspection of each CHull solution, a S3 plot method can also be used directly on the object `'multichullcom'`. By setting the `browser` argument to `TRUE`, a shiny app will be opened in the default web browser and each CHull plot can easily be inspected using a dropdown menu. If the `browser` argument equals `FALSE` a small shiny app is started in the plot viewer pane of Rstudio (see Figure 5.7).

```
plot(results, browser = FALSE)
```

In Table 5.5 results of the `MultiCHull` `"multicom"` version are shown. The

results indicate that when a relatively simple model complexity was used that only considers the multiplication between the number of components and clusters (i.e., Complexity ID 1), a model with 25 components and a single cluster was selected (also see Figure 5.7). Although the scree plot yields a nice 'elbow', a one cluster solution is unfortunate since the data are known to consist of a sample of two different diagnostic class labels (i.e., patients with Alzheimer's disease versus frontotemporal dementia).

Results further indicate that for three of the seven (i.e., Complexity ID 3, 4 and 5) considered measures, again a -relative complex- model with 25 components and one cluster is selected (equal to the selected model from Figure 5.7). Note that the complexity value for ID 4 and 5 is equal to each other, since the selected model has only a single cluster.

Next, analyses with the two complexity measures capturing the total number of C-ICA parameters (Complexity ID 6 and 7) resulted in the selection of a model with two components and two clusters. Note that these two complexity measures take the number of subjects, clusters, components, time points and voxels into account. The difference between these two definitions lies in the aforementioned number of parameters that are estimated for the partition matrix $\mathbf{P}$ (i.e., subject clustering).

Finally, for the complexity definition that focuses on the summation of the number of components and voxels (Complexity ID 2), a model selecting for 20 components and seven clusters -the maximum number of clusters- is selected, which is clearly a too complex model for this data.

All in all, different definitions of model complexity led to a clearly different selected model. Hence, researchers should carefully consider a model complexity definition for C-ICA (and also other models for which the underlying model complexity cannot be determined in a straightforward way). To this end, one can use the Multi-CHull function in order to inspect the effect of the choice of model complexity measure on the retained C-ICA solution with easy to use software. Based on our results, we would recommend to use one of the two complexity measures that take the total number of C-ICA parameters into account (Complexity ID 6 and 7) as these definitions for complexity resulted in the retention of a sensible model with two clusters and two brain patterns per cluster.

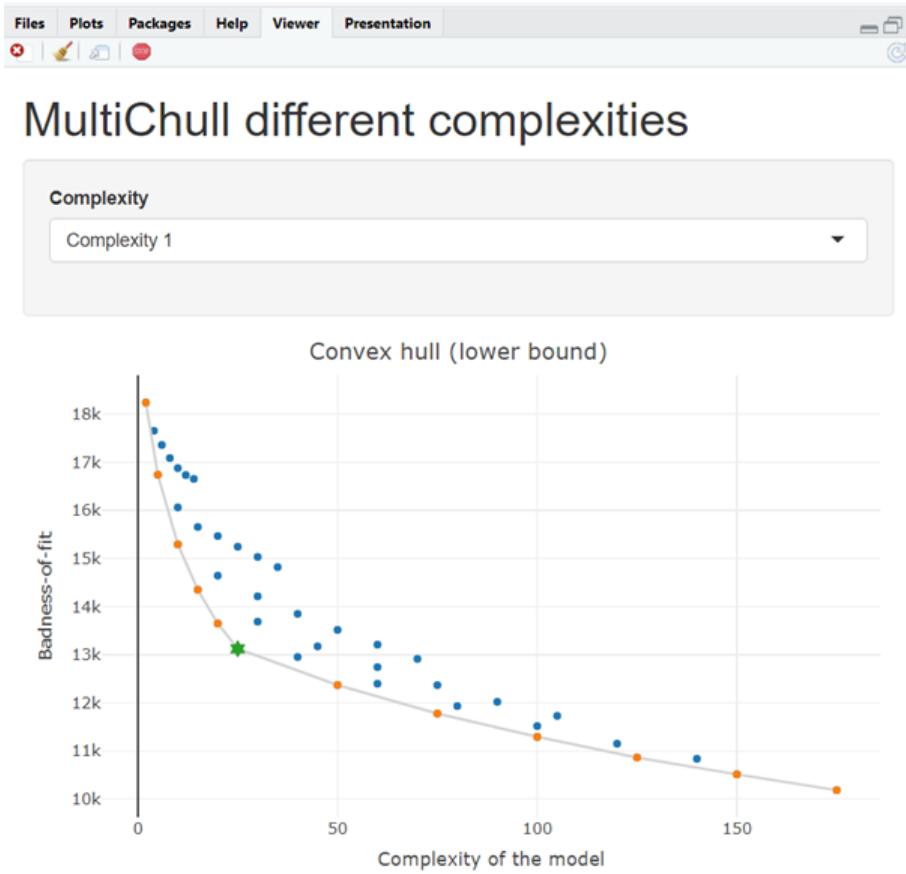

Figure 5.7: Result of applying the S3 plot method to the object of class `"multichullcom"` which contains the C-ICA results (for multiple complexity measures). Each CHull analysis result (per specified complexity measure) can be viewed easily by selecting the required complexity measure from a dropdown menu. In this figure, Complexity 1 refers to a CHull analysis where the complexity of a Clusterwise Independent Component Analysis equals Q (components) $\times$ R (clusters). On the x-axis all considered models (ordered in terms of complexity) are displayed, with the specified fit measure on the y-axis. The selected model (on the boundary of the convex hull) is indicated automatically in this plot (here with a green star). Candidate models that are not selected but are lying on the boundary of the hull are indicated with orange dots, whereas the other models are displayed with blue dots. By hovering the cursor over a data point, the associated model number is highlighted.

Table 5.5: Results of the Multi-CHull analysis using multiple model complexity definitions of a Clusterwise Independent Component Analysis (C-ICA)

Complexity definition ID	Complexity definition	Model ID	Number of components	Number of clusters	Complexity	Misfit
1	QR	6	25	1	25	13123.82
2	$Q + R$	41	20	7	27	10837.42
3	QVR	6	25	1	63825	13123.82
4	$QVR + I$	6	25	1	63845	13123.82
5	$QVR + IR$	6	25	1	63845	13123.82
6	$QVR + QTI$	7	2	2	18212	17651.80
7	$QVR + QTI + IR$	7	2	2	18252	17651.80

Note. I = number of subjects, R = number of clusters, Q = number of components, T = number of time points, V = number of voxels.

5.6 Conclusion

In this paper, the Multi-CHull procedure and associated R function and package is presented that allows for the inspection of multiple CHull analyses on multiple fit values or multiple complexity definitions, which allows for performing a multiverse analysis (Steegen et al., 2016). The Multi-CHull procedure was illustrated on two data analytic choices in the context of a Tucker3 analysis and one analytic choice in the context of a C-ICA analysis. As expected, when applying Multi-CHull on fit indices obtained from Tucker3 analyses on bootstrapped datasets, a stable model with a rather low complexity was retained for the majority of the bootstrap samples. As such, the user gains confidence in the stability of the retained solution. However, when applying Multi-CHull to the same dataset but preprocessed in a different way, other models were retained depending on whether or not the data were (double) centered across the rater mode.

Besides using `MultiCHull` for analyzing multiple fit measures, the utility of Multi-CHull was also demonstrated by analyzing multiple complexity measures of a C-ICA analysis. By applying CHull on multiple complexity values, a user can easily inspect different solutions and obtain information on how different model complexity definitions affect the model selection procedure. The results suggest that, for the current example, a model complexity definition that takes the total number of C-ICA parameters into account resulted in a sensible model with relatively few clusters and components. This is arguably a desirable model from an interpretational point of view.

Next to the three examples illustrated in this paper, `MultiCHull` can also easily be used to aid the user in several other data analytic choices that need to be decided upon before, during or after the analysis. Again, a distinction can be drawn between choices that are not expected to drastically change the model selection outcome (e.g., bootstrap samples) and choices that may profoundly influence the complexity of the retained model (e.g., pre-processing and using different definitions for model complexity). Regarding the former, Multi-CHull can also be used to determine the stability/consistency of the (complexity of the) retained solution across several imputed data sets in the context of a missing value analysis, or across data folds when studying the generalizability of the results with cross-

validation, jackknife or an odd-even sample split. Other applications of Multi-CHull in this regard can be, in the context of variable selection, the study of the consistency of the retained solution across various data sets including different subsets of variables or across analysis results obtained with different types of starting configurations for the parameters of the algorithm. In all these examples, it is expected that mostly the same solution -with a certain complexity- will be selected. There, however, also exists situations in which Multi-CHull can be used to study how data analytic choices influence model selection. In particular, Multi-CHull can disclose how model selection is affected when different preprocessing options for fMRI data, which is a notorious problem for these types of data (for an example, Aurich, Alves Filho, Marques da Silva, & Franco, 2015), are adopted. Other applications in this regard are studying the effect of using, in the context of factor analysis for categorical data, Pearson's correlations versus polychoric correlations, or investigating the influence of different fitting procedures and algorithms in the context of SEM, nonnegative matrix factorization or clustering.

It can be concluded that a Multi-CHull analysis, which can be performed by the multichull package introduced in this paper, enables the user to study the effect of several data analytic choices on which model is retained by CHull as optimal. As such, the presented multichull package facilitates a multiverse analysis which allows users to investigate the variability in obtained results across several reasonable analytic choice options (Steegen et al., 2016). Performing such a multiverse analysis enlarges the awareness of researchers for the (sometimes seemingly arbitrary) choices that are most consequential in the fragility of the study results.