

Contour Clustering

Manual

Constantijn Kaland
Institute of Linguistics - Phonetics
University of Cologne

This manual provides introductory usage instructions for the R shiny application Contour Clustering (Kaland, 2021), which offers a graphical user interface to perform cluster analysis on f0 and other acoustic cues. The scripts, a tutorial, and other supplementary materials can be downloaded on the website:

<https://constantijnkaland.github.io/contourclustering/>



March 2025

Contents

1	Tips to get started	3
2	Requirements	4
2.1	Unit of analysis	4
2.2	Dataset	4
2.3	Segmentation	5
3	Workflow	7
3.1	Run app	7
4	Data	9
4.1	Load file	9
4.2	Read folder	10
4.3	Acoustics	11
4.3.1	F0 floor and ceiling	11
4.3.2	F0 fit	11
4.3.3	F0 smoothing bandwidth	11
4.3.4	F0 time-step	12
4.3.5	Number of measurement points	12
4.3.6	Sampling	13
4.3.7	Intensity	13
4.3.8	Duration	14
4.4	Speakers	15
4.5	Clean	15
4.6	Representations	16
4.6.1	F0	16
4.6.2	Intensity	18
4.6.3	Duration	18

5	Clustering	19
5.1	Variables for clustering	19
5.2	Distance metric	20
5.3	Clustering method	21
5.3.1	Hierarchical agglomerative clustering - HAC	21
5.3.2	Partitioning around medoids - PAM	24
5.4	Number of clusters	25
5.5	Plot settings	25
5.6	Cluster outputs	26
5.6.1	Dendrogram (HAC only)	26
5.6.2	Plot	26
5.6.3	Table	27
5.7	Subsetting	29
5.8	Evaluation	30
6	Data conventions	33
6.1	df.u	33
6.2	df.g	33
6.3	df.l	33
6.3.1	Singleton vs. time-series	34
6.3.2	Decimals	35
6.4	df.e	35
7	Settings	36
7.1	Number of rows to display	36
7.2	File upload limit	36
7.3	Saving	36
7.4	Keeping objects	37
7.5	Garbage collection	37
7.6	Logfile	37
8	References	38
9	Index	42

1. Tips to get started

- download the latest version of the app
- use the index in this manual to jump to specific topics
- update R/-Studio and required packages (see versions in code)
- take a look at the workflow chart on page 8
- practice with a training dataset from the website
- hover your mouse on any location in the app to get tips
- don't forget to save your work in the app ('Save' or 'Save this' buttons)
- get inspired by previous work on various languages

2. Requirements

First, the three minimal requirements are discussed, which need to be fulfilled before contour clustering can be applied. A basic understanding of scripting R (R Core Team, 2022; R Studio Team, 2023) is helpful, but not required.

2.1 Unit of analysis

The researcher is required to decide which linguistic unit of analysis needs to be investigated. The unit of analysis depends on which level of prosody is subject to the interest of the researcher (e.g., word or phrase level). In principle, contour clustering can be applied to any interval extracted from speech. For example, for the investigation of lexical tone the unit of analysis would typically be the syllable or word, whereas for the investigation of phrase level intonation the unit of analysis would typically be the (intermediate) phrase. Note that some understanding of the prosodic relevance of these units in the language under investigation needs to be present. It is beyond the scope of this manual to provide fieldwork paradigms that help to identify the units that are relevant in the prosody of certain language (see e.g., Jun and Fletcher, 2014 for guidelines). Nespor and Vogel (2007) provide a phonological hierarchy that could be maintained. The decision for a unit has a couple of implications for later steps in the procedure. For this reason, it is advisable to keep the unit of analysis the same throughout one analysis, i.e. not comparing syllable-level f_0 movements with those found in phrases (Section 4.2).

2.2 Dataset

The researcher should have access to a collection of audio-recordings and corresponding annotations in Praat textgrids (Section 2.3 for annotation requirements), consisting of sufficient observations. Praat textgrids can be exported from several

programs, such as ELAN (Max Planck Institute for Psycholinguistics, 2022, see <https://www.mpi.nl/corpus/html/elan/ch01s04.html> and STx (Noll et al., 2019, see <https://projects.ari.oeaw.ac.at/stx>).

What a sufficient amount of observations is, depends on the quality of the recordings and a number of later steps in the analysis. For example, when only four clusters need to be identified, the analysis might reach a representative outcome with less observations than when eight clusters need to be identified (see Section 5.4 for instructions on the number of clusters). It is possible that some data needs to be discarded due to f0 measurement errors. It is therefore recommended to collect more units than strictly speaking needed. In the phrase contour example (Kaland, 2021, Section 3), 321 contours were used for initial analysis, on the basis of which at least one clearly defined follow-up hypothesis could be formulated concerning potential functional differences between in the contours. For the tonal analysis (Kaland, 2021, Section 4) the contours of 213 syllables were analysed and again provided insightful results for further testing. Another indication of whether the dataset is large enough for cluster analysis can be obtained from the mean standard errors within each cluster (provided in the output table, Section 5.6). Large standard errors indicate large variance within the cluster, which could be the result of (too) few observations. It is important to note that in general more observations will often lead to more representative results and that the ideal size of the dataset largely depends on decisions taken during the analysis (see Section 5.4 and 5.7). Unbalanced datasets are not necessarily problematic for the cluster analysis. For example, a dataset with 5% questions and 95% statements could still be accurately clustered when their differences in f0 contours are clear enough.

2.3 Segmentation

Once the unit of analysis and the dataset are decided upon, the researcher needs to segment all units of analysis on an interval tier in a Praat textgrid (Boersma and Weenink, 2022). This means that the left and right boundary of each unit (an interval) need to be indicated. It is highly recommended to label the units using glosses in the language under investigation or another common language (e.g., English), or ideally both. For purely exploratory purposes, contour clustering does not require text labels for the intervals on the textgrid tier. The more information is available in the textgrid for each unit, the more solid the interpretation of the results can be after analysis (see ‘contingency table’ in Section 5.6.3). This information can include other linguistic aspects that are not necessary included in the initial intonation analysis

(e.g. segments, syllables, words, phrases, discourse units, grammatical annotations). See <https://osf.io/cr4vn/> for Praat scripts that handle and count (overlapping) intervals on multiple tiers.

3. Workflow

Figure 3.1 provides a schematic overview of the contour clustering workflow when using the application. The workflow is divided into a ‘data’ stage (Chapter 4) and ‘clustering’ stage (Chapter 5). The names of the respective tab panels in the application are used as section titles in the following, in order to provide a transparent reference to the location in the workflow as well as the in the application.

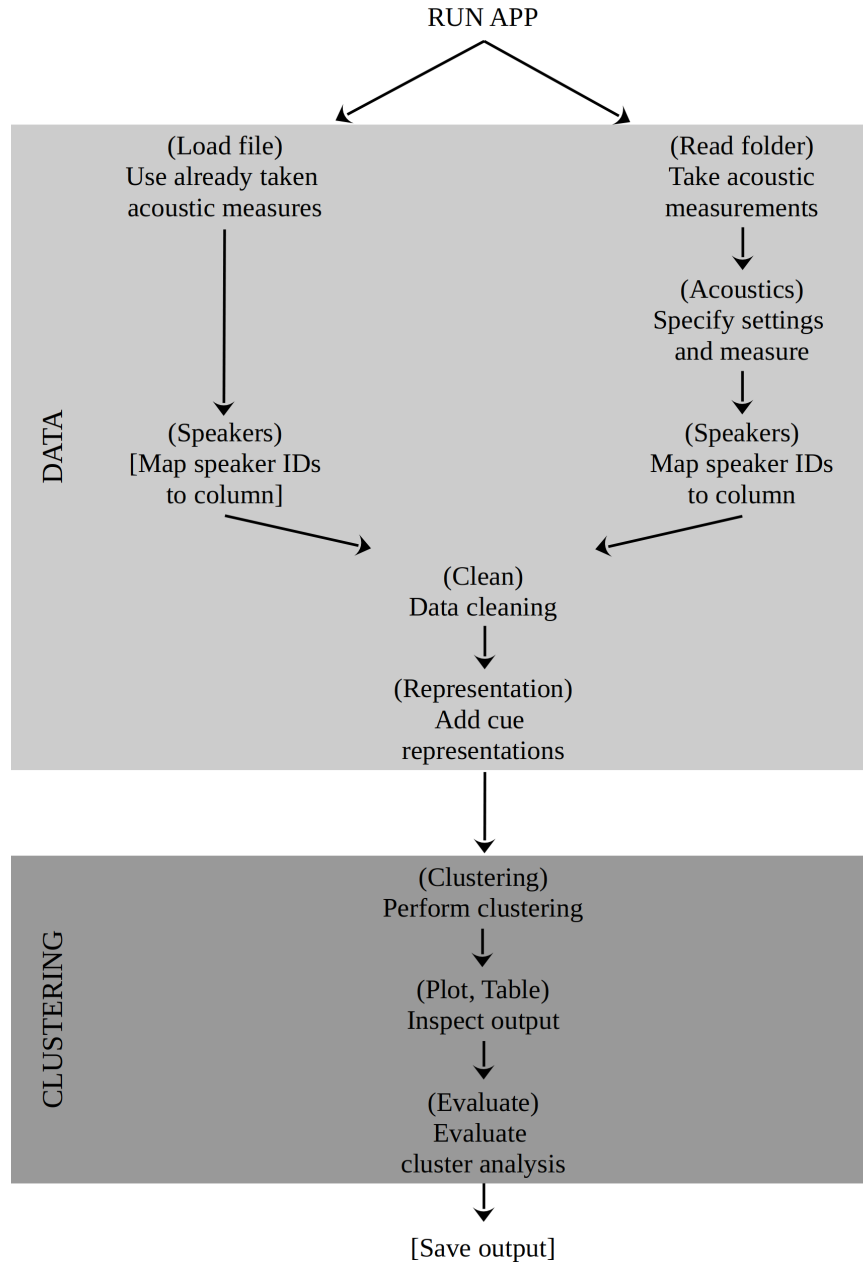
The data stage covers all operations that are required to prepare a dataset for clustering. Once that stage is completed, the clustering stage will be initiated and covers all operations related to the performance and processing of cluster analysis and its results.

In what follows, each step in the workflow is explained in a separate section, in the order of the workflow.

3.1 Run app

In order to run the application, R (R Core Team, 2022) and R Studio (R Studio Team, 2023) need to be installed. The application will install the latest versions of each required package automatically when the code is run. If any packages needed a (new) installation, the R session will automatically restart and the application need to be started again. If no ‘Run app’ button is displayed, the `shiny` package (Chang et al., 2012) needs to be installed first (run: `install.packages("shiny")`). An overview of the required packages is provided in the application. The application is best displayed in a maximized window. Any smaller app window might cause layout issues or controls being (partially) invisible. It is also possible to run the application in a browser window by choosing ‘Run external’ in R Studio or set the `launch.browser = T` option in the code (commented out in the code by default).

Figure 3.1: Schematic overview of the Contour Clustering workflow, divided in a ‘Data’ and a ‘Clustering’ stage referring to the respective main tabpanels in the application. For each step, the names of sub-tabpanels are given between brackets. Square brackets indicate optional steps in the workflow.



4. Data

Once the app has started, a choice needs to be made about the source of the to be clustered data (requirements see Section 2.2): either upload an existing dataset (Load file - Section 4.1) or read audio files and textgrids from a folder (Read folder - Section 4.2).

4.1 Load file

Use this option if acoustic measures are already taken. Files that can be uploaded may be obtained from the application in a previous session, from running the ‘time-series.f0.praat’ script in Praat (Boersma and Weenink, 2022), or from (manually) compiled data by another source and/or the researcher. Note that non-locally stored files (server/cloud) may not be readable.

Uploading a file requires a .csv file in ‘long’ format. This means that each row in the data represents a single measurement point (i.e., a single f0 value in Hertz). For each row it is furthermore required to have values in at least the following columns:

Table 4.1: Required columns when uploading a datafile.

Column name	Class	Content
<i>filename</i>	character	Refers to the filename (.wav/.mp3) from which the measurements were taken.
<i>start</i>	numeric	Start time of the interval (seconds)
<i>end</i>	numeric	End time of the interval (seconds)
<i>speaker</i>	character	Speaker IDs (can be identical to or extracted from another character column)
<i>interval.label</i>	character	Text referring to the interval (usually taken from textgrid intervals)
<i>stepnumber</i>	integer	The sequence number of the measurement point in the time-series
<i>f0</i>	numeric	Fundamental frequency measured in Hertz, ERB or semitone (app can convert)

Additional columns can be taken from the uploaded datafile. The application caters for columns that were used in older versions (e.g., ‘jumpkilleffect’, ‘steptime’),

as well as ones that represent other acoustic cues ('intensity' or 'duration') or additional columns that require manual specification of whether they contain character values, numeric values, or are referring to an (acoustic) variable that should be taken into account for clustering ('cc', see Section 6).

It is furthermore possible to subset the uploaded data for specific filenames and/or interval labels.

4.2 Read folder

The 'Read folder' tab panel should be selected if no acoustic measurement were taken yet. The path to a folder that contains both audio files and textgrids needs to be given, and the extension of the audio files (.wav or .mp3) specified. Note that non-locally stored files (server/cloud) may not be readable.

Reading the folder means that the application scans for audio files and textgrids that have identical names (only differ in extension, e.g., 'Recording_1.wav' and 'Recording_1.TextGrid'). The resulting list of names is returned and individual files can be (de)selected. The audio files are processed faster when they have only one channel (mono). The textgrid files should have the file encoding UTF-8 or ASCII and should not be saved as 'short text file' in Praat. Textgrids not conforming to those conditions cannot be read by the `readtextgrid` package (Mahr, 2020) and cause the application to halt and produce a warning message showing which file was processed when the error occurred. Note that the application does not automatically convert the file encoding as to avoid losing the use of specific (special) characters in the textgrids. It is left to the user to provide (a converted copy of) the textgrids in a readable encoding.

All interval tiers found in the textgrids can be selected for further processing. In case multiple tiers are selected, they should be of comparable units of analysis. That is, it is not advised to feed a tier with segmented syllables and a tier with segmented phrases into the same analysis. With multiple tiers selected, the name of each tier is prepended to the text in the 'interval_label' column. The data read from the textgrids (`readtextgrid::read_textgrid()`, Mahr, 2020) can be viewed in the 'Data (TextGrids)' tab.

It is furthermore possible to select specific interval labels from the selected tier(s).

4.3 Acoustics

The ‘Acoustics’ tab provides an environment to take acoustic measurements of f0, intensity and duration - only available after ‘Read folder’ (Figure 3.1). Each cue can be (de)selected depending whether it should become available for cluster analysis. Their respective settings are discussed in the following.

4.3.1 F0 floor and ceiling

The f0 tracking method is Modified Harmonic Sieve (Scheffers, 1983 as implemented in `wrassp::mhsF0()`, Winkelmann et al., 2023). This method requires an f0 window for the expected values, as defined by the f0 floor and f0 ceiling. These can be adjusted depending on the voice under analysis or the type of phonation (Gordon and Ladefoged, 2001).

4.3.2 F0 fit

The Modified Harmonic Sieve method aims at detecting f0 in noisy environments by a maximum likelihood estimation from its harmonics (Goldstein, 1973). The f0 fit threshold specifies the minimum probability for an f0 measurement to fit. Increasing this value makes the detection algorithm more strict, rejecting more f0 candidates and accepting only the better fitting ones (more accuracy, less candidates). Decreasing this value will accept more candidates at the expense of tracking accuracy.

4.3.3 F0 smoothing bandwidth

By default, the f0 measures are interpolated and extrapolated, which means that all missing f0 points are filled in by linear interpolation and missing f0 points at the edges are filled in by extrapolation using a constant based on the first/last available f0 point. This is done to obtain uninterrupted f0 contours, which is at least partially motivated by perception research (Mixdorff and Niebuhr, 2013).

The f0 contour is then smoothed using kernel density estimation (KDE). The applied method is described in Silverman (1986). The smoothing bandwidth positively correlates with the degree of smoothing (i.e. larger bandwidth = more smoothing). The bandwidth value is the input of the `stats::ksmooth()` function. The smoothing allow to reduce the influence of local (erroneous or irrelevant) f0 perturbations on the overall contour. It is recommended to try out multiple settings and inspect the results using the sampling method (Section 4.3.6).

4.3.4 F0 time-step

The time-step setting refers to the frame duration used to calculate f0 (`wrassp:mhsF0()` ‘windowShift = ’). The standard setting used in Praat depends on the pitch minimum (or pitch floor; see Praat Manual “Sound: To Pitch...”; Boersma and Weenink, 2022) and defaults to 10 ms with a minimum of 75 Hz. Manually specifying the time-step provides a way to configure the resolution of the f0 measures. This might be needed in addition to number of measures and smoothing.

The time-step setting determines how many measurement points are initially taken for a contour, which at the stage of f0 tracking is different from the number of measures setting (Section 4.3.5). For example, for a 200 ms interval one obtains 20 f0 measures with a 10 ms window and one obtains 40 f0 measures with a 5 ms window. This number of measurement points will be converted to number of points the user sets to estimate each contour (Section 4.3.5), i.e. after inter-, extrapolation and smoothing. If the final number of measurement points is more than the number of points obtained using the time-step setting, the likelihood of inaccurate f0 approximation increases. That is, a contour obtained from 20 measures (e.g. 200 ms interval, 10 ms time-step) will not become more accurate when setting the number of measurement points to 40. Approaching this issue the other way around is recommended, i.e., setting the time-step such that more f0 points are taken than the representation chosen in the app. Thus, the time-step setting offers a way to safeguard that the tracking resolution equals or exceeds the contour resolution. Some additional lines of text, dependent on the data and the settings, are provided in the application to guide this process.

4.3.5 Number of measurement points

The researcher needs to decide how many f0 measurements are needed to represent the smoothed contour as time-series. It is important to realise that the smoothed contour is already the product of interpolation, extrapolation and smoothing, potentially based on just a few tracked f0 points. Representing the smoothed contour by more points than actually tracked, could be misleading as this potentially generates the illusion of accurate representation (e.g. representing a smoothed f0 contour by 20 points when it is based on 5 tracked f0 points, see Section 4.3.4).

It is important that eventual turning points in the contour are well represented by both the smoothing and the number of measurement points. This depends largely on the unit of analysis and the desired accuracy of the measures. Taking too few measures could lead to missing essential f0 movements within the contour, whereas too many measures could give too much importance to insignificantly small f0 mea-

asures, although the latter should be largely taken care of by the smoothing process. Note that the smoothing process (Section 4.3.3) could also be responsible for the over-estimation of small f0 movements. It is recommended to change the number of measurement points only in accordance with the smoothing resolution, whilst considering the output of the cluster analysis (Section 5.6).

It is important to note that inaccurate f0 measurements are likely to occur in each dataset. The cluster analysis provides an automatic detection of erroneous and outlying contours, which can then be removed from the data (Section 5.7). Changing the number of measures to account for a small number of erroneous contours is not recommended, as this affects also the correct measures. Only if the same type of f0 errors occur repeatedly, changing the number of measures could be considered.

4.3.6 Sampling

Before taking f0 measures of an entire dataset, there is an option to sample a specified number of randomly chosen intervals to inspect the accuracy of the acoustic settings. In particular the f0 fit, number of measurement points and smoothing bandwidth should be changed to improve the quality of the measurements. It is likely that the standard settings produce inaccurately tracked f0 contours due to, e.g., octave jumps. After each sampling, plots are generated showing the tracked f0 points using the tracking method (black solid line) and the interpolated, extrapolated and smoothed f0 smoothing contour (red dashed line). Sampling provides therefore an essential quality control before taking acoustic measures for all observations in the data. It is possible that f0 cannot be tracked (accurately) in all intervals; this will be shown in the sampling plots and NAs will be written to the output file. In addition to the plot, the audio taken from the sample can be played. This is done by cutting the sampled interval(s) from the original file and storing them temporarily in the ‘www’ folder in the folder from which the application is run. The ‘www’ folder and its contents will be automatically removed when sampling has finished.

Each time ‘Visualize sample’ is clicked, the current f0 settings are used to generate the plots based on the sample that was already taken. Only when ‘Take sample’ is clicked, a new sample is taken. Changing the size of the sample is only effectuated when ‘Take sample’ is clicked.

4.3.7 Intensity

Intensity (short-term Root Mean Square amplitude in dB) can be measured as additional (or sole) acoustic cue. The intensity measures are taken using the `wrassp:rmsana()`

function (Winkelmann et al., 2023). The number of measurement points is identical for intensity and f0 time-series.

4.3.8 Duration

Duration measurements can be added for clustering and will be taken as singleton ones (not time-series) and provide a way to compensate for the temporal information that is lost when representing each time-series with the same number of measurement points (time-warping). The duration is calculated by subtracting the interval end time from the interval start time. This is done once per interval (hence ‘singleton’, see Section 6.3.1) and written to the first row in the data that represents this interval (i.e., for stepnumber 1).

4.4 Speakers

When taking acoustic measurements in the application or when the uploaded datafile does not have a column with speaker IDs, the tab ‘Speakers’ offers a way to create a speaker column. This can be a 1:1 copy of another character column that is already present in the data or an extracted string from the any character column in the data. The latter option is useful when, for example, the ‘interval_label’ column consists of multiple sources of information, e.g., task number, speaker ID and condition in the format `008_pp3_test`. The application offers a way to split the string according to any punctuation mark found in that column (regex class: `[[:punct:]]`). In the example, an underscore would be the separator to extract ‘pp3’ as speaker ID.

The extraction is done in three stages, from left to right in the ‘Speakers’ tab:

- 1) select the column containing the speaker IDs
- 2) select the separator character
- 3) select the column with the speaker IDs after separation.

The selection made in step 3 can be taken as the new column ‘speaker’. Note that the number of unique character strings found in the (new) ‘Speaker’ column equals the number of speakers.

4.5 Clean

The cleaning tab is presented once the dataset has all required columns and at least one column with acoustic measures for clustering (commonly f0 values). Upon display, the ‘Clean’ tab shows a summary of the number of filenames, speakers, measurement points and intervals (contours), and any problems currently detected in the data. Table 4.2 lists the most common problems that are checked.

Table 4.2: Problems checked by the cleaning process

Problem	Description
missing filenames	NA entries in ‘filename’ column
missing speaker IDs	NA entries in ‘speaker’ column
empty interval labels	accidental spaces ‘ ’ in ‘interval_label’ column
missing f0	NA entries in ‘f0’ column
negative f0	negative value in ‘f0’ column (e.g., result of previous standardization)
rate of f0 change	checks adjacent f0 values against rates in Xu and Sun (2002, Table X)

Any detected problems will be listed in the summary and can be solved by clicking ‘Clean and re-check’, after which a new summary is given. It is important to observe how many observations get removed in the cleaning process, to see whether there are any structural problems in the data.

Although it is possible to ignore the detected problems and continue without cleaning, this option is not recommended and only advised if the application has detected problems that can be ignored. Ignoring detected problems may lead to application crashes.

4.6 Representations

The ‘Representation’ tab offers a final step of bringing the data in the desired shape for clustering. Arriving at this tab means that all source data is present and cleaned. It is therefore possible to continue directly to clustering. However, it might be desired to apply conversions or (speaker) corrections to the (raw) acoustic measurements in order to obtain additional and more representative values for the acoustic cues. This can be done for f0, intensity and duration separately, when present in the data. The options for the respective cues are discussed below. Clicking ‘Add .. representation’ adds the chosen representation as new values in an additional column to the dataset (see Section 6).

4.6.1 F0

All operations to obtain f0 representations can be combined, i.e. it is possible to do a functional principal component analysis on the acceleration (d2) values of speaker-standardized semitone values.

F0 scale can be set to Hertz, ERB or semitone. ERB conversion according to the formula in Greenwood, 1961, p.1352, Eq. 2.

F0 values can be speaker corrected by means of standardization (z-scoring) or octave-median rescaling (De Looze and Hirst, 2014).

Derivatives (d1 - velocity, d2 - acceleration, d3 - jerk) can be taken from the (converted) (speaker-corrected) f0 values.

Functional principal component analysis (fPCA, see Gubian, Torreira, and Boves, 2015) can be run on the (converted) (speaker-corrected) (derived) f0 values. The analysis is automatically run as soon as the number of principal components (PCs) is not zero. The output is a graph showing the harmonic values for each PC and settings are given for smoothing (λ - lambda) and which PC should be selected for clustering. It is possible to run fPCA with four components whilst adding only PC1

and PC2 to the data for clustering. The values of each PC appear in a new column as singleton measure, i.e., one value per interval at measurement point 1 (Section 6.3.1).

Standardizing the f0 subtracts the mean f0 of a speaker from each f0 measure of that speaker and divides the outcome by the f0 standard deviation of that speaker. After standardizing, the mean f0 centers around zero and the standard deviation is one. The minimum and maximum f0 values still reflect the original distribution and outlying f0 values are therefore not accounted for. Note that if outliers are the result of f0 measurement errors, they can still effectively be handled by means of subsetting (Section 5.7). Standardizing is a recommended method to preserve functional f0 differences such as tone contrasts when comparing different speakers (e.g. Rose, 1987).

Another method to scale the f0 range is based on the octave scale, taking into account the speaker’s median f0. This makes the f0 measures more robust to outliers and is proposed as more representative method for speech melody compared to reference values based on the mean and expressed in semitones (De Looze and Hirst, 2014). Octave-median scaling appeared particularly accurate for estimating acoustic emphasis as a result of focus, topic change or turn-taking.

fPCA reduces the dimensionality of the f0 contours to a chosen number of principal components. The components represent deformation functions relative to the overall mean f0 contour and can be visualized using curves (contours). Each principal component comes with scores for each contour, which allow for a reconstruction of the original f0 contour from the mean f0 contour + the principal component curves + scores. The application offers a basic fPCA functionality: setting the number of components, a plot of the component curves, and an overview of the proportion of the variance explained by each component. The user can select which component(s) need to be included into the cluster analysis. fPCA is performed using the `fda` package (Ramsay, 2003) following the procedure outlined in <https://github.com/caojiguo/FDAcourse2019>. Additional materials can be found on <https://github.com/uasolo/FPCA-phonetics-workshop>.

The application of cluster analysis on the fPCA scores offers a promising way of handling the f0 variation, possibly less affected by noise in the data. Note that the fPCA is run after any of the selected cleaning and/or speaker correction methods are applied. The user needs to select the number of principal components (default: 4) and λ (default: 1), a smoothing factor used in the function `fdPar()` to define a functional parameter object. Higher lambda numbers indicate more smoothing, which could lead to larger deviations from the original f0 contours in the reconstructed fPCA curves.

4.6.2 Intensity

Intensity values can be standardized (z-scored) per filename or per speaker. Filename standardization is useful when recordings (files) had different microphone levels. Speaker standardization standardizes over the speaker IDs and is useful when speakers' voices differ in loudness.

4.6.3 Duration

Duration values can be standardized per speaker to account for individual differences in speech tempo (in the unit of analysis).

5. Clustering

The clustering stage starts when all required data is present, has been cleaned and the desired representations for the acoustic cue(s) have been added to the data. The ‘Clustering’ tab provides cluster settings (side panel) and cluster outputs such as plots, tables and evaluations (main panel).

5.1 Variables for clustering

From the acoustic representations present in the data, any number can be chosen as variable(s) to apply cluster analysis to. When more than one variable is chosen, the cluster analysis will be *multivariate clustering*. This is handled by the application in the following way. First, a distance matrix is computed for each variable (with the chosen distance metric, see Section 5.2). The distance values in each distance matrix are then scaled such that they lie between zero and one. The scaled distances are summed into a single distance matrix, which is then taken as the input for cluster analysis.

Multivariate time-series clustering can in theory also be done by concatenating the time-series. Note, however, that this does not allow that combinations of time-series values (e.g., f0) and singleton (e.g., duration) values are given equal weight. In addition, concatenation is not suitable when using, e.g., dynamic time warping as distance metric, because all measures of all variables are analysed as a single vector per observation. The application thus uses the scaling-summing approach for multivariate clustering, giving equal weight to the distances of the respective variables.

See Section 5.2 and Section 6.3.1 for restrictions on the use of distance metrics when choosing the variable(s) for clustering.

5.2 Distance metric

The available distance metrics are Euclidean (L2 Norm / RMSE), mean absolute scaled error (MASE), dynamic time warping, Pearson correlation and autocorrelation (see Table 5.1).

Table 5.1: Overview of available distance metrics and the respective R package used for computation. Only metrics marked with * are suitable for singleton measures.

Distance metric	R-package	pros	cons
*Euclidean (L2 Norm = RMSE)	stats	tested on f0 contours	insensitive to misalignment
mean absolute scaled error (MASE)	Metrics	outperforms RMSD	not widely used for f0
*dynamic time warping	proxy	sensitive to misalignment	overcorrection
pearson correlation	TSdist	overall similarity	contour cannot be flat
autocorrelation	TSdist	high potential for f0	computationally costly

Euclidean distance (L2 Norm) or root mean squared error (RMSE) is widely used to compute the distance between two vectors due to its low computational cost and frequent availability. It is however, insensitive to misalignments in time as it is a lock-step measure and does not handle outliers well (Esling and Agon, 2012). This means that clustering outputs using Euclidean distance as a distance metric become more reliable when the data is cleaned and/or converted beforehand. This metric has been tested on f0 contours (Hermes, 1998) reflecting intonation perception to some extent.

Mean absolute squared error (MASE) is a time-series metric used to calculate the accuracy of weather forecasts (Hyndman and Koehler, 2006). This metric is scale-invariant and penalizes negative and positive distances in equal ways, outperforming related metrics such as root mean squared deviation (RMSD). Although the (f0) contour comparison is different from quantifying prediction accuracy - and currently needs further testing, the method is a potential alternative and applicable to time-series (not suitable for singleton measures). For each distance calculation between two contours, one contour is taken as if it were the actual weather observation and the other contour is taken as the prediction. The output MASE value can be taken as a distance metric as it shared the assumption that at zero, there is no distance between the contours (no errors between observation and prediction). MASE values greater than one indicate that a naive forecast (assuming no change) would preform better than the given forecast values. It is likely that f0 contour distances expressed in MASE end up being higher than one, indicating that one contour could not be

predicted from the other (i.e. are highly dissimilar). In this way, the MASE threshold of 1 could be used as cut-off point to potentially improve cluster accuracy.

Dynamic time-warping provides a way to account for misalignments between two contours that are otherwise similar in shape. These misalignments could have been the results of the time-warping that is done by taking a fixed number of measurement points to represent the f0 contour. Dynamic time warping as a distance metric should be avoided if the length of the unit of analysis has already been controlled for. That is, there is a trade-off to the usefulness of dynamic time-warping if all intervals have (almost) the same length. In this case, time-warping might analyse misaligned portions of the f0 contours as been similar although they are not. The importance of time-warping thus depends on the research question and data at hand.

Pearson correlation distance is an effective lock-step measure to compute overall similarity between contours that are represented by time-series data (not suitable for singleton measures). It is important that Pearson correlation coefficients are computed in such a way that they range from -1 to 1 (maximum negative and maximum positive correlation respectively). If direction of the f0 contour is not taken into account (as by taking the absolute correlation coefficient), the distance metrics will regard simultaneously diverging contours as being similar. The computation as implemented in the application takes direction into account and is therefore suitable for application on acoustic variables.

Autocorrelation - being a feature-based measure - is effective to deal with f0 contours as it takes into account that adjacent measures in a time-series are correlated (not suitable for singleton measures). The autocorrelation distance is computationally costly. Note that for the computation of correlation coefficients, time-series values are expected to have a standard deviation that is higher than zero. This means that flat contours will obstruct the computation of correlation coefficients.

5.3 Clustering method

Two common clustering methods are available in the application: hierarchical agglomerative clustering (HAC) and partitioning around medoids (PAM). HAC and PAM differ fundamentally in their mathematical approach (further reading: Kaufman and Rousseeuw, 1990; Reynolds et al., 2006; Scitovski et al., 2021).

5.3.1 Hierarchical agglomerative clustering - HAC

For HAC there are multiple ways in which the clusters can be formed. Note that HAC in the current approach is bottom up, thus starting with each observation in

a separate cluster. Clusters are formed by merging existing clusters until there are only two clusters left. Theoretically, one extra step applies in that the final two clusters are merged such that all observations are in a single cluster. However, this final step is tantamount to no clustering and is not informative given that clustering aims at finding groups in a dataset. The dendrogram provides useful directions for the degree and type of subsetting needed, as further discussed in Section 5.6.1 and 5.7.

Which clusters get merged is determined by the *linkage criterion*. The linkage criterion specifies how distances (dissimilarities) between clusters are computed. An overview of the linkage criteria available in the application is given in Table 5.2. The default one recommended here is complete linkage, given the idea that maximal acoustic contrasts often underlie linguistically meaningful differences (e.g., the composition of vowel inventories, Lindblom, 1986). There is no a priori objection to select a different linkage criterion, depending on the type of f0 variation in the data.

Table 5.2: Overview of linkage criteria

Linkage criterion	Cluster dissimilarity definition
Complete	Computes all pairwise dissimilarities between the observation(s) in the clusters and takes the maximal dissimilarity. Clusters with the smallest maximal dissimilarity get merged, leading to maximal inter-cluster dissimilarity.
Single	Computes all pairwise dissimilarities between the observation(s) in the clusters and takes the minimal dissimilarity. Clusters with the smallest minimal dissimilarity get merged, leading to minimal inter-cluster dissimilarity.
Average (UPGMA)	Computes all pairwise dissimilarities between the observation(s) in the clusters and takes the mean dissimilarity. Clusters with the smallest mean dissimilarity get merged.
Ward	Computes the increase in sum of squares for all possible pairwise merges of clusters. Clusters with the smallest increase get merged. This method keeps the total within-cluster variance minimal. Ward.D and Ward.D2 differ in that the cluster differences are squared in the latter and therefore emphasized, leading to easier cluster differentiation.
McQuitty (WPGMA)	Similar method to average linkage, without considering the number of observations in a cluster and with taking into account the similarity between the most recently merged clusters.
Centroid (UPGMC)*	Computes the centroids (central point of all observations, i.e. a vector of means) of the clusters. Clusters with the centroids closest to each other get merged.
Median (WPGMC)*	Similar to centroid linkage, with taking into account the similarity between the most recently merged clusters.

* These linkage criteria have the risk of leading to inversion. For most link-

age criteria, merging happens on the basis of iteratively larger dissimilarities. With centroid-based linkage, centroid distances could get smaller in subsequent merging iterations. In such case, it is no longer possible to assume that with an increasing number of clusters variance among the observations in a cluster increases, whereas variance among the clusters decreases (see within and between cluster variance discussed in Section 5.8).

5.3.2 Partitioning around medoids - PAM

PAM (Kaufmann and Rousseeuw, 1987) makes use of medoids, which are actual observations that are taken as cluster ‘centers’ (BUILD stage). This is initially done by selecting k observations that have the smallest sum of distances to all other observations (with k being the number of clusters to be identified). Each (non-medoid) observation is then associated to the (according to the distance metric) nearest medoid. Then, PAM attempts to improve the clustering quality by reassigning medoids (SWAP stage). For each reassignment it is checked whether the average dissimilarity between the medoid and their associates decreases. If so, the reassignment is accepted. When the medoids do not change anymore, the partitioning of the data has arrived at a stable outcome, which is taken as the final cluster assignment.

To reduce computation cost, several PAM improvements have been proposed. The optimizers are therefore recommended for large datasets to optimize the processing speed. None of the optimizers changes the cluster assignment (unlike the linkage criterion in HAC). Table 5.3 lists the available optimizers in the application.

Table 5.3: Overview of PAM optimizers available in `cluster::pam()` (Maechler et al., 1999)

Optimizer	Description
K-medoids	Original PAM proposed in Kaufmann and Rousseeuw (1987)
PAM1 PAM2	Optimizations proposed in Reynolds et al. (2006)
FastPAM1 FastPAM2 FastPAM3	Optimizations proposed in Schubert and Rousseeuw (2019)
FasterPAM	Optimization proposed in Schubert and Rousseeuw (2021)

5.4 Number of clusters

Finding the ideal number of clusters is key in performing cluster analysis. Before deciding on the number of clusters, it is recommended for the researcher to have a theoretically motivated estimation of this number, i.e. before obtaining any result from the analysis. For example, if the aim is to find a basic set of different lexical tone contours from words, around four to six clusters could be sufficient to accurately capture the contrasts. However, if the aim is to find a broad set of all lexical tone contrasts in combination with phrase intonation patterns, e.g. 15 clusters could be the minimum number needed. One way of determining the number of clusters is to run several rounds of analysis, each time with an increasing number of clusters (Kaland, 2021, Section 2-4). This is particularly useful in an exploratory context, where only a rough estimation of the number of clusters can be made. It is also recommended to run the cluster analysis with a number of clusters that exceeds the hypothesized number. In this way, the researcher can reduce the risk of missing relevant contrast that were unexpected. The dendrogram (HAC only) provides some initial guidance in finding the ideal number of clusters. Obtaining the dendrogram for a given dataset does not require any prior decisions on the number of clusters. That is, the dendrogram remains as is when changing the number of clusters.

An indication of the accuracy of the number of clusters can also be derived from plotting the mean contours per cluster, as further explained below. Apart from the general guidelines outlined here, there are statistical methods to obtain an estimation of the ideal number of clusters in an analysis, which are left for the user to explore (e.g. Charrad et al., 2014).

5.5 Plot settings

The ‘Clustering’ tab offer basic settings for the way the cluster plots are displayed. The number of columns for the cluster plot can be set, which determines how many cluster panels are displayed horizontally (one panel per cluster). This number, depending on the number of clusters, determines width of the panels (height is set to 400 pixels per row/panel).

It is furthermore possible to set which time-series variable should be represented on the y-axis. When multiple time-series variables are available in the data, a second y-axis (on the right of the plot) can be chosen. In this way, multiple representations can be visually inspected for how they compare among each other, independent of whether they were actually used as a variable for clustering or not.

Note that the plot settings apply to displaying the plot in the application. Saving the plot takes the width and height as set in the application’s general settings (Section 7).

5.6 Cluster outputs

After each clustering, a plot and a table are generated. HAC additionally provides a dendrogram for initial inspection of the results. The output are discussed in the following.

5.6.1 Dendrogram (HAC only)

HAC provides the researcher with a dendrogram, a tree-structure showing the outcome of each merge. The dendrogram provides a first visualisation of the clustering process. On the basis of the dendrogram the researcher can decide on the amount of clusters. The dendrogram also provides insight in the scale of the f0 differences between the clusters. Since the largest numerical differences between clusters are found at the top of the dendrogram, an initial analysis might reveal only the differences involving a larger f0 range. In such an initial outcome, the dendrogram is likely to show asymmetry. This asymmetry is the result of late adjoining of major f0 excursions (e.g. large boundary tones) with a cluster containing all smaller scale (e.g. phrase-internal) f0 excursions. Asymmetry can be avoided to some extent by choosing a speaker correction method that accounts for outliers. Initial clustering outcomes thus provide insight into contour differences of the largest scale. Smaller scale differences, such as f0 peak height, are more likely to be successfully revealed when increasing the number of clusters. However, large numbers of clusters might result in clusters consisting of few contours, compromising the prototypicality of these contours. To accurately reveal small scale f0 differences, it is recommended to analyse a more homogeneous (controlled) dataset, or a subset of the data leaving out large-scale f0 excursions.

5.6.2 Plot

The plot shows the average values of the time-series variable(s) that were used for clustering (y) for each measurement point (x). The grey band around the values represents the standard deviations. A panel is displayed for each cluster, with the cluster number and number of observations in that cluster (n) given in the heading.

If duration is taken into account as a clustering variable, the panel headings also display the mean duration (d) per cluster.

It is plausible that some of the clusters in the plot show highly similar contours. This is to be expected with a (recommended) high number of clusters for initial analysis. It is recommended to not solely rely on the plot of mean contours to inspect the potential similarity between two clusters. It might not be immediately clear why two similar contours end up in separate clusters. Inspection of the actual acoustic differences between the contours is recommended in such a case. This can be done by reading the f0 values from the plot. These values should be compared to more obvious differences between other clusters obtained from the same analysis. For example, rising phrase final boundary tones could have a large f0 range and therefore be visually easy to detect. This type of “landmark” in the contour can provide an indication of the scale of the differences between clusters. Thus, visually similar contours in separate clusters might also be the result of small numerical differences which could be more accurately clustered after subsetting. Decreasing the number of assumed clusters could also be considered as a means to obtain visually more distinguished contours (an example of such a consideration is given in Kaland, 2021, Section 3). However, this method makes the cluster analysis more course-grained, introducing a higher risk of overlooking relevant contour differences.

5.6.3 Table

The summary table shows the number of observations per cluster, the percentage of the number of observations in the cluster relative to all observations, and for each clustering variable the mean standard error per cluster. The mean standard error per cluster is calculated by adding up the standard errors for each measurement point and divide the outcome by the number of measurement points. A tentative indication (flagging) is given of whether a cluster should be treated with caution. The flagging is given by a * based on the data in the table. That is, caution is advised when a cluster contains only one contour or when the mean standard error is more than two times the median of the mean standard errors of all clusters. The rationale behind the latter threshold is to obtain a single criterion that can be applied to f0 values regardless of the speaker correction method or number of clusters chosen. In particular, standard errors might be affected by some of these correction methods. By taking the median as center value, the distance from zero (no deviation) is known. Thus, mean standard error values that lie further away from median than zero does, are advised to be treated with caution. The criterion is a rough estimate of how deviant the mean contour in a particular cluster is.

Individual inspection is still advised to be able to quantify the standard errors on a f0 scale. Therefore, the inspection should be done before applying speaker correction. For example, a single cluster with a mean standard error of 10 Hz might already be enough to overlook potentially meaningful differences between the contours. That is, the f0 range for human speech can be taken as 75 to 500 Hz (standard setting in Praat; Boersma and Weenink, 2022). Thus, allowing for an average deviation of up to 10 Hz for the entire contour corresponds to a semitone difference that lies between 0.3 (480-500 Hz) and 2.2 (75-85 Hz). It has been shown that in Dutch, for example, 1.5 ST excursions (locally!) can be enough to perceive a linguistically meaningful prominence shift (Rietveld and Gussenhoven, 1985). Although the generalizability of this finding to other languages is open to further research, it is crucial to note that within cluster variance of 10 Hz might not fully avoid overlooking important f0 movements in the contour. The double-median threshold for the mean standard errors could therefore be too crude without consideration of the f0 scale. While smaller standard errors are generally preferred, increasing the number of assumed clusters can help to gain insight into this type of variability and is recommended prior to any subsetting procedures. The standard errors can thus be taken as an indication of how well the contours fit in the cluster. As a rule of thumb: if homogeneously sized clusters with a low mean standard error can be achieved with subsetting applied only to discard erroneous cases, the analysis is likely to provide an optimal outcome.

A contingency table can be generated after each clustering. The contingency table generates one row per cluster and columns for each level of a chosen variable. In this way, the contingency table may act as first interpretation of the clustering with regard to the variables that are relevant for the research question. The variable can be copied/extracted from an existing character column (i.e., ‘interval_label’). The extraction steps are similar to the ones explained for mapping the ‘speaker’ column (Section 4.4):

- 1) select the column containing the variable
- 2) select the separator character (if any)
- 3) select the column with the relevant levels of the variable after separation.

Note that each unique character string found in step 3 will become a column in the contingency table. The extracted column will be displayed after step 3 for further inspection.

5.7 Subsetting

A procedure for subsetting the data is provided in the application (tab ‘Clustering’, subtab ‘Table’). Clicking ‘Remove clusters’ provides an option to eliminate the observations in specific clusters. Note that this is irreversible within one session, i.e., data needs to be measured or uploaded again in order to have the complete dataset. The application automatically lists the flagged clusters as candidates for removal.

The removal procedure (henceforth ‘subsetting’) has two purposes. Its initial use can be applied to a high number of clusters (Kaland, 2021, Section 3 and 4 for examples) in order to remove erroneous or outlying contours. For this purpose, a high number of clusters (e.g., 25 or more) is recommended to obtain small sized clusters. If the data consists of erroneous or outlying contours, these will then be revealed in small sized clusters, potentially single-contour clusters. Thus, when removing small sized clusters, the risk of discarding error-free contours remains low. This way of “pruning” the data can be done using the automatic flagging of erroneous/outlying contours (Section 5.6.3).

A second purpose of the subsetting is to “zoom in” into a specific subset of the data, typically after initial round(s) of clustering. For example, initial clustering could reveal a small number of rising contours among an overall majority of falling contours. It could be useful to separate the rises and falls in subsets and perform subsequent clustering on either subset. This has the advantage of revealing smaller scale differences between the contours (e.g. different types of rises or falls), without their differences being affected by contours of the other category. Such an application of the subsetting procedure is particularly useful when there is clear indication or supportive evidence from distinguishing these two types as categories.

A general word of caution should be given here, as subsetting essentially ignores (potentially large) parts of the collected data. This procedure compromises the representativeness of the empirical investigation and re-introduces the risk of giving researchers’ intuitions a decisive role in the analysis-process. Although these disadvantages cannot be entirely avoided, it is crucial to keep the ultimate goal of the investigation in mind. The procedures outlined here are designed to reveal prototypical contours, for which some deviant instances can be naturally expected in spontaneous speech (see further discussion in Kaland, 2021, Section 3).

5.8 Evaluation

The application provides an additional interface to evaluate the ideal number of clusters (tab ‘Evaluation’). Two methods are implemented; one based on information cost (Shannon, 1948) and minimum description length (MDL, Rissanen, 1978) as described in Kaland and Ellison (2023), and one based on within and between cluster variance. In case multiple clustering variables were chosen, the evaluation is based on the first (preventing the treatment of different scales as identical).

The first evaluation method computes information cost of three aspects of the cluster analysis; 1) the cost of specifying the clusters, which is expected to increase with more clusters, 2) the probability of a contour being in a specific cluster, given what the prototype (mean) of that cluster is, which is likely to decrease with more clusters, and 3) the cost of specifying each contour within a certain cluster, this is likely to increase with more clusters, depending on how well an individual contour fits the cluster (see details and demonstration in Kaland and Ellison, 2023). The summed outcome is then taken as a single evaluative information cost measure for one round of cluster analysis (one ‘clustering’).

When applied over a range of clusterings, e.g. starting with two clusters up to ten clusters, the evaluation measures for each clustering round are likely to produce a U shaped curve when the number of clusters are plotted on the x-axis and the information cost measure on the y-axis. The user is required to specify the range of cluster rounds and a bending factor (measurement point dependency value). The former indicates the number of clusters that should be assumed for the first and the last round of cluster analysis that should be evaluated. The evaluation then runs those rounds and all rounds with intermediate numbers of clusters. The dependency value indicates how dependent adjacent measurement points are, with higher values corresponding to higher degrees of dependency. Note that this is a way to take into account the length of the unit of analysis, which could range from (portions of) syllables (high dependency) to entire utterances (low dependency). The bending factor value can be manually set (with decimals, if needed). This might be needed to obtain a clearer U shaped bend in the plotted curve, but should be informed by the unit of analysis and the absolute time-gap between measurement points. When analysing a dataset for which time-normalization has been done outside the provided Praat- or R-scripts, the bending factor needs to be adjusted accordingly.

The MDL evaluation procedure generates a plot with the information cost for each clustering round. The clustering round with the lowest information cost is the MDL and could be taken as the ideal number of clusters for the dataset under analysis. Note that all evaluation is based on the specific dataset under analysis

and does therefore not directly allow generalization for the contour inventory of that specific language.

The second evaluation method is based on within and between cluster variation. It is expected that within cluster variance of the f0 contours decreases with more clusters, because with more clusters the individual contour within a certain cluster are more alike than with less clusters. In addition, between cluster variance is expected to increase with more clusters as the mean f0 contour in a certain cluster will differ more from the mean f0 contours in other clusters (i.e. more varying mean contours with more clusters). Note that these principles do not necessarily apply when choosing centroid-based linkage criteria (inversion), see Table 5.2.

Within cluster variance is computed by taking the standard deviation of each measurement point iteratively for all measurement points in a cluster. That is, starting with cluster 1, the standard deviation is taken from all the first measurement points in that cluster, then the standard deviation is taken from all the second measurement points in that cluster (etc.), until the final measurement point in that cluster. Then, the mean of all the standard deviations is taken to represent the variance in that cluster, after which the same procedure is repeated for the remaining cluster(s). Thereafter, all the mean standard deviations (one per cluster) are averaged again to represent within cluster variance for one clustering.

Between cluster variance is computed by taking the mean f0 value of a measurement point across clusters. That is, starting with measurement point 1, their mean value in each cluster is taken iteratively for all clusters. Then, the absolute difference between the maximum and the minimum of these mean values is taken to represent the between cluster variance of that particular measurement point. This procedure is repeated for all measurement points and the single value representing the between cluster variance of a particular clustering is the mean of all these difference measures.

In order to make the within and between cluster variance comparable, the values are scaled between 0 and 1. The scaling is done for each evaluation round. For example, if an evaluation round is set to compare clusterings ranging from 2 to 10 clusters (9 clusterings), the resulting 9 values representing within cluster variance as well the 9 values representing the between cluster variance are scaled.

Theoretically, the optimum clustering is the one for which the lowest within cluster variance and the highest between cluster variance is observed. The optimum number of clusters will thus lie behind the cross-over point of both variances. Depending on the curvature of the variances a clear optimum might be observed, which is reached for the number of clusters before the distance between the two variances no longer or only minimally increases. Note that the cluster variance evaluation method is sensitive to the range of clusterings taken into account. That is, the scaled variance

might vary to some extent depending on the range of clusterings. It is recommended to observe the relative gain or loss in the curves representing within/between cluster variance. If a ceiling or floor effect can be observed, this could be taken as an indication that extending the maximum of the selected range is no longer needed. Rather, if the ceiling or floor is stretched over multiple clustering rounds at the high end of the range, the optimum lies well before the clustering round with the highest number of clusters. It is recommended to have an informed idea about the maximum number of clusters to test in the evaluation.

The two evaluation methods do not need to show corresponding results. It is important to note that the approaches have different backgrounds and use therefore different ways to express the optimal number of clusters. One method seeks the optimum in terms of informativeness of describing the dataset. The other method provides insight into the clustering process and how the contour variation is distributed within or between clusters. That is to say that they can both provide useful perspectives on what the number of clusters to choose *could* be.

6. Data conventions

The following sections describe the main data objects stored in R’s working memory by the names with which they are stored. This explanation provides (advanced) users with a basic understanding of how the application handles the data. All objects with a name starting with ‘df.’ refer to data tables (package `data.table`, Barrett et al., 2006). It is possible to directly access these objects from the R Global Environment (see Section 7.4).

6.1 df.u

This is the data as uploaded by the user and read by `data.table::fread()`. This data is displayed in the tab ‘Data (upload)’ immediately after uploading and before mapping the columns to the required ones. If uploaded data is not correctly displayed, either change the way the data is stored or specify additional options in the syntax of the `fread()` function.

6.2 df.g

This is the data as read from the textgrids by `readtextgrid::read_textgrid()` (Mahr, 2020). The data provides all information in the textgrids as a data table and can be viewed in the ‘Data (TextGrids)’ tab. This tab offers a way to check whether the textgrids are read correctly.

6.3 df.l

This is the main data with which the application works. It is generated on the basis of the uploaded (‘df.u’) or read (‘df.g’) data and has the long format. This means

that each row represents a single measurement point. Table 4.1 provides an overview of the columns that are minimally needed in this data.

All columns that refer to variables for clustering have a specific template for their naming. The names are concatenated character strings following the structure in Table 6.1.

Table 6.1: Clustering variable column naming conventions in ‘df.l’. Concatenation of strings from left to right, one string taken per column in the table, with optional strings in brackets. N.B. octave-median rescaling, derivation and fPCA are only available for f0. Standardization is available for all cues.

default	cue	scale	[spk.corr]	[derivation]	[fPCA]
		.Hz			
	.f0	.ERB		.d1	.PC1
.cc		.ST	.std	.d2	.PC2
	.int	.dB	.oMe	.d3	.PC{n}
	.dur	.s			...

Thus, `cc.f0.ERB.oMe.PC2` is the name of the column that contains the PC2 values taken from octave-median rescaled ERB values of fundamental frequency. And `cc.f0.Hz.d1` is the name of the column that contains the velocity values of the fundamental frequency values in Hertz. And `cc.dur.std` is the name of the column with standardized duration values.

6.3.1 Singleton vs. time-series

The long data has the format of each row per measurement point, which is mainly the case to cater for time-series variables such as f0 and intensity. This is because time-series measures represent the interval by the chosen number of measurement points. Duration values and principal component values are singleton measures in the sense that they represent the interval by a single value. Singleton measures are written to the data at stepnumber 1 (the other measurement points are `NA`), see Section 6.

Clustering with singleton measures restricts the use of distance metrics. That is, only distance metrics based on Euclidean distance are available. Other distance metrics are exclusively applicable to time-series measures.

6.3.2 Decimals

All numeric values in ‘df.l’ have a maximum of three decimals, in order to avoid bloated datafiles and reduce processing speed.

6.4 df.e

This data is obtained from the evaluation procedure and lists for each clustering round either an information cost value (MDL) or a combination of within and between cluster variance. This data is used to generate the evaluation plot and is displayed underneath it.

7. Settings

A number of general settings is provided that apply to one session of running the application. They are discussed one by one in the following.

7.1 Number of rows to display

This setting specifies how many rows should be displayed when viewing data (any of the sets described in Section 6. This number is by default limited to 50 as to prevent large datasets requiring long rendering times.

7.2 File upload limit

This sets the limit for uploading files to the application in megabytes. Depending on the computer, the way in which the application is run, and the size of the data, this setting provides a way to prevent the application from being slow due to the upload of large datasets. Note that with each addition of an acoustic cue representation (Section 4.6), additional values are written to the data and will make the data object larger in size after uploading. Adding a representation after a file has been uploaded is not affected by the upload limit. Therefore, it could be more important to start with a small dataset when uploading.

7.3 Saving

The application provides options to save the logfile, the data, the cluster plot, the summary table, the contingency table, and the evaluation plot. Individual ‘Save’ buttons are available for each. Alternatively, ‘Save this’ can be clicked to save all available output at once in a folder called ‘saved’ (in the directory from which the

application is run). Additional saving settings are provided in the application settings. These include setting the width and height in pixels for all plots that can be saved. It is furthermore possible to generate a compressed file (.zip) with for all outputs saved through the ‘Save this’ button. The compressed file is given a date and number of clusters in the filename for future reference.

Note that the data in long format is saved such that it can be re-uploaded to the application in a future session (see also Section 6).

7.4 Keeping objects

In order to keep the objects that were generated by the application in the Global Environment of R, it is possible to set this option. Keeping objects is useful for debugging purposes or for continuing to work with the data after the application was closed. See Section 6 for an overview of the relevant objects. Keeping objects in the Global Environment lasts as long as the R-session and still requires saving them manually if this is desired. By default, the objects generated by the application are removed upon closing.

7.5 Garbage collection

Runs R’s garbage collection function `gc()`. This function is only needed if the application is slow, in which case the effects might be minimal. Garbage collection happens automatically in R regardless of manually running this function.

7.6 Logfile

The application automatically writes a lines of text to a logfile for each relevant step in the workflow. The logfile provides all information needed to reproduce the specific steps in a session. It is highly recommended to save the logfile each time any of the cluster outputs are saved.

8. References

- Barrett, T., Dowle, M., Srinivasan, A., Gorecki, J., Chirico, M., Hocking, T., & Schwendinger, B. (2006). Data.table: Extension of ‘data.frame’ [Institution: Comprehensive R Archive Network Pages: 1.16.4]. <https://doi.org/10.32614/CRAN.package.data.table>
- Boersma, P., & Weenink, D. (2022). Praat: Doing Phonetics by Computer. <http://www.praat.org/>
- Chang, W., Cheng, J., Allaire, J., Sievert, C., Schloerke, B., Xie, Y., Allen, J., McPherson, J., Dipert, A., & Borges, B. (2012). Shiny: Web Application Framework for R [Institution: Comprehensive R Archive Network Pages: 1.10.0]. <https://doi.org/10.32614/CRAN.package.shiny>
- Charrad, M., Ghazzali, N., Boiteau, V., & Niknafs, A. (2014). NbClust: An R Package for Determining the Relevant Number of Clusters in a Data Set. *Journal of Statistical Software*, 61(6). <https://doi.org/10.18637/jss.v061.i06>
- De Looze, C., & Hirst, D. (2014). The OMe (Octave-Median) scale: A natural scale for speech melody. *7th International Conference on Speech Prosody 2014*, 910–914. <https://doi.org/10.21437/SpeechProsody.2014-170>
- Esling, P., & Agon, C. (2012). Time-series data mining. *ACM Computing Surveys*, 45(1), 1–34. <https://doi.org/10.1145/2379776.2379788>
- Goldstein, J. L. (1973). An optimum processor theory for the central formation of the pitch of complex tones. *The Journal of the Acoustical Society of America*, 54(6), 1496–1516. <https://doi.org/10.1121/1.1914448>
- Gordon, M., & Ladefoged, P. (2001). Phonation types: A cross-linguistic overview. *Journal of Phonetics*, 29(4), 383–406. <https://doi.org/10.1006/jpho.2001.0147>
- Greenwood, D. D. (1961). Critical Bandwidth and the Frequency Coordinates of the Basilar Membrane. *The Journal of the Acoustical Society of America*, 33(10), 1344–1356. <https://doi.org/10.1121/1.1908437>

- Gubian, M., Torreira, F., & Boves, L. (2015). Using Functional Data Analysis for investigating multidimensional dynamic phonetic contrasts. *Journal of Phonetics*, 49, 16–40. <https://doi.org/10.1016/j.wocn.2014.10.001>
- Hermes, D. J. (1998). Measuring the Perceptual Similarity of Pitch Contours. *Journal of Speech, Language, and Hearing Research*, 41(1), 73–82. <https://doi.org/10.1044/jslhr.4101.73>
- Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4), 679–688. <https://doi.org/10.1016/j.ijforecast.2006.03.001>
- Jun, S.-A., & Fletcher, J. (2014). Methodology of studying intonation: From data collection to data analysis. In S.-A. Jun (Ed.), *Prosodic Typology II* (pp. 493–519). Oxford University Press. <https://doi.org/10.1093/acprof:oso/9780199567300.003.0016>
- Kaland, C. (2021). Contour clustering: A field-data-driven approach for documenting and analysing prototypical f0 contours. *Journal of the International Phonetic Association*, 53(1), 159–188. <https://doi.org/10.1017/S0025100321000049>
- Kaland, C., & Ellison, T. M. (2023). Evaluating cluster analysis on f0 contours: An information theoretic approach on three languages. In R. Skarnitzl & J. Volín (Eds.), *Proceedings of the 20th International Congress of Phonetic Sciences* (pp. 3448–3452). Guarant International. https://sfb1252.uni-koeln.de/sites/sfb.1252/user_upload/Pdfs_Publikationen/Kaland_Ellison_2023_Evaluating_cluster_analysis.pdf
- Kaufman, L., & Rousseeuw, P. J. (Eds.). (1990). *Finding Groups in Data*. John Wiley & Sons, Inc. <https://doi.org/10.1002/9780470316801>
- Kaufmann, L., & Rousseeuw, P. (1987). Clustering by means of medoids. In Y. Dodge (Ed.), *Data Analysis based on the L1-Norm and Related Methods* (pp. 405–416). North Holland / Elsevier. https://www.researchgate.net/profile/Peter-Rousseeuw/publication/243777819_Clustering_by_Means_of_Medoids/links/00b7d531493fad342c000000/Clustering-by-Means-of-Medoids.pdf
- Lindblom, B. (1986). Phonetic universals in vowel systems. In J. Ohala & J. Jaeger (Eds.), *Experimental Phonology* (pp. 13–44). Academic Press.
- Maechler, M., Rousseeuw, P., Struyf, A., & Hubert, M. (1999). Cluster: "Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al. [Institution: Comprehensive R Archive Network Pages: 2.1.8]. <https://doi.org/10.32614/CRAN.package.cluster>
- Mahr, T. (2020). Readtextgrid: Read in a 'Praat' 'TextGrid' File [Institution: Comprehensive R Archive Network Pages: 0.1.2]. <https://doi.org/10.32614/CRAN.package.readtextgrid>

- Max Planck Institute for Psycholinguistics. (2022). ELAN [<https://archive.mpi.nl/tla/elan>]. Retrieved July 11, 2019, from <https://archive.mpi.nl/tla/elan>
- Mixdorff, H., & Niebuhr, O. (2013). The influence of F0 contour continuity on prominence perception. *Interspeech 2013*, 230–234. <https://doi.org/10.21437/Interspeech.2013-73>
- Nespor, M., & Vogel, I. (2007). *Prosodic Phonology: With a New Foreword* [Google-Books-ID: GAETNIP_H34C]. New York : Mouton de Gruyter.
- Noll, A., Stuefer, J., Klingler, N., Leykum, H., Lozo, C., Luttenberger, J., Pucher, M., & Schmid, C. (2019). Sound tools eXtended (stx) 5.0 — a powerful sound analysis tool optimized for speech [ISSN: 2958-1796]. *Interspeech 2019*, 2370–2371.
- R Core Team. (2022). R: The R project for statistical computing [Version 4.2.1]. <https://www.r-project.org/>
- R Studio Team. (2023). RStudio: Integrated Development for R. Retrieved July 11, 2019, from <https://www.rstudio.com/>
- Ramsay, J. (2003). fda: Functional Data Analysis [Institution: Comprehensive R Archive Network Pages: 6.1.8]. <https://doi.org/10.32614/CRAN.package.fda>
- Reynolds, A. P., Richards, G., De La Iglesia, B., & Rayward-Smith, V. J. (2006). Clustering Rules: A Comparison of Partitioning and Hierarchical Clustering Algorithms. *Journal of Mathematical Modelling and Algorithms*, 5(4), 475–504. <https://doi.org/10.1007/s10852-005-9022-1>
- Rietveld, A., & Gussenhoven, C. (1985). On the relation between pitch excursion size and prominence. *Journal of Phonetics*, 13(3), 299–308. [https://doi.org/10.1016/S0095-4470\(19\)30761-2](https://doi.org/10.1016/S0095-4470(19)30761-2)
- Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14(5), 465–471. [https://doi.org/10.1016/0005-1098\(78\)90005-5](https://doi.org/10.1016/0005-1098(78)90005-5)
- Rose, P. (1987). Considerations in the normalisation of the fundamental frequency of linguistic tone. *Speech Communication*, 6(4), 343–352. [https://doi.org/10.1016/0167-6393\(87\)90009-4](https://doi.org/10.1016/0167-6393(87)90009-4)
- Scheffers, M. T. M. (1983). Simulation of auditory analysis of pitch: An elaboration on the DWS pitch meter. *The Journal of the Acoustical Society of America*, 74(6), 1716–1725. <https://doi.org/10.1121/1.390280>
- Schubert, E., & Rousseeuw, P. J. (2019). Faster k-Medoids Clustering: Improving the PAM, CLARA, and CLARANS Algorithms [Series Title: Lecture Notes in Computer Science]. In G. Amato, C. Gennaro, V. Oria, & M. Radovanović (Eds.), *Similarity Search and Applications* (pp. 171–187). Springer International Publishing. https://doi.org/10.1007/978-3-030-32047-8_16

- Schubert, E., & Rousseeuw, P. J. (2021). Fast and eager k-medoids clustering: O (k) runtime improvement of the PAM, CLARA, and CLARANS algorithms. *Information Systems*, 101, 101804. <https://doi.org/10.1016/j.is.2021.101804>
- Scitovski, R., Sabo, K., Martínez-Álvarez, F., & Ungar, Š. (2021). *Cluster analysis and applications*. Springer Nature. <https://doi.org/10.1007/978-3-030-74552-3>
- Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3), 379–423. <https://doi.org/10.1002/j.1538-7305.1948.tb01338.x>
- Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. Chapman & Hall/CRC.
- Winkelmann, R., Bombien, L., Scheffers, M., & Jochim, M. (2023). *Wrassp: Interface to the 'ASSP' library* (manual). <https://CRAN.R-project.org/package=wrassp>
- Xu, Y., & Sun, X. (2002). Maximum speed of pitch change and how it may relate to speech. *The Journal of the Acoustical Society of America*, 111(3), 1399–1413. <https://doi.org/10.1121/1.1445789>

9. Index

- acceleration, 16
- acoustic measurements, 11
- alignment, 21
- application settings, 36
- ASCII, 10
- autocorrelation (distance metric), 20, 21

- bending factor, 30
- between cluster variance, 31
- browser window, 7

- cleaning, 15
- cluster analysis, 19
- clustering method, 21
- clustering variables, 19
- columns, 9, 34
- contingency table, 5, 28

- data (tab), 9
- data conventions, 33
- decimals, 35
- dendrogram, 22, 25, 26
- derivatives, 16
- display rows, 36
- distance metric, 20, 34
- duration, 14, 18
- dynamic time warping, 20, 21

- Euclidean distance, 20

- evaluation, 30, 35
- extrapolation, 11–13

- f0 ceiling, 11
- f0 correction, 16
- f0 fit, 11
- f0 floor, 11
- f0 scale, 16
- f0 smoothing, 11–13
- f0 time-step, 12
- file encoding, 10
- flagging, 27
- fPCA, 16, 17

- garbage collection, 37
- Global Environment (R), 37

- HAC, 21
- hierarchical agglomerative clustering (HAC), 21

- information cost, 30
- intensity, 13, 18
- interpolation, 11–13
- interval tier, 5
- inversion risk, 23

- jerk, 16

- kernel density estimation (KDE), 11

- lambda, 17

- linkage criterion, 22
- logfile, 37
- long format, 9, 33, 34
- MASE, 20, 21
- MDL, 30
- mean standard error, 5, 27
- medoid, 24
- minimum description length (MDL), 30
- Modified Harmonic Sieve, 11
- multivariate clustering, 19
- number of clusters, 5, 25–27, 29, 30
- number of measurement points, 12, 13, 34
- objects, 33, 37
- observations, 5
- octave jump, 13
- octave-median rescaling, 16, 17
- optimizers (PAM), 24
- packages, 7, 20
- PAM, 24
- partitioning around medoids (PAM), 24
- Pearson correlation, 20, 21
- phonation, 11
- plot output, 26
- plot settings, 25
- principal component analysis, 16
- problems (cleaning), 15
- prototypicality, 26, 29
- read folder, 10
- remove clusters, 29
- representations, 16, 19
- RMSE, 20
- run application, 7
- sampling, 13
- saved-folder, 36
- saving, 26, 36
- segmentation, 5
- singleton measure, 14, 17, 20, 34
- smoothing bandwidth, 11
- speakers, 15
- speed of processing, 10, 24, 35–37
- standardization, 16–18
- subsetting, 10, 17, 22, 27–29
- summary table, 27
- table output, 27, 28
- textgrid, 4, 5, 10, 33
- time-series measure, 12, 14, 34
- unit of analysis, 4, 5, 10, 12, 30
- upload file (tab), 9, 33, 36
- upload limit, 36
- UTF-8, 10
- velocity, 16
- window size, 7
- within cluster variance, 31
- workflow, 7
- www-folder, 13
- z-scoring, 16, 18
- zip file, 37