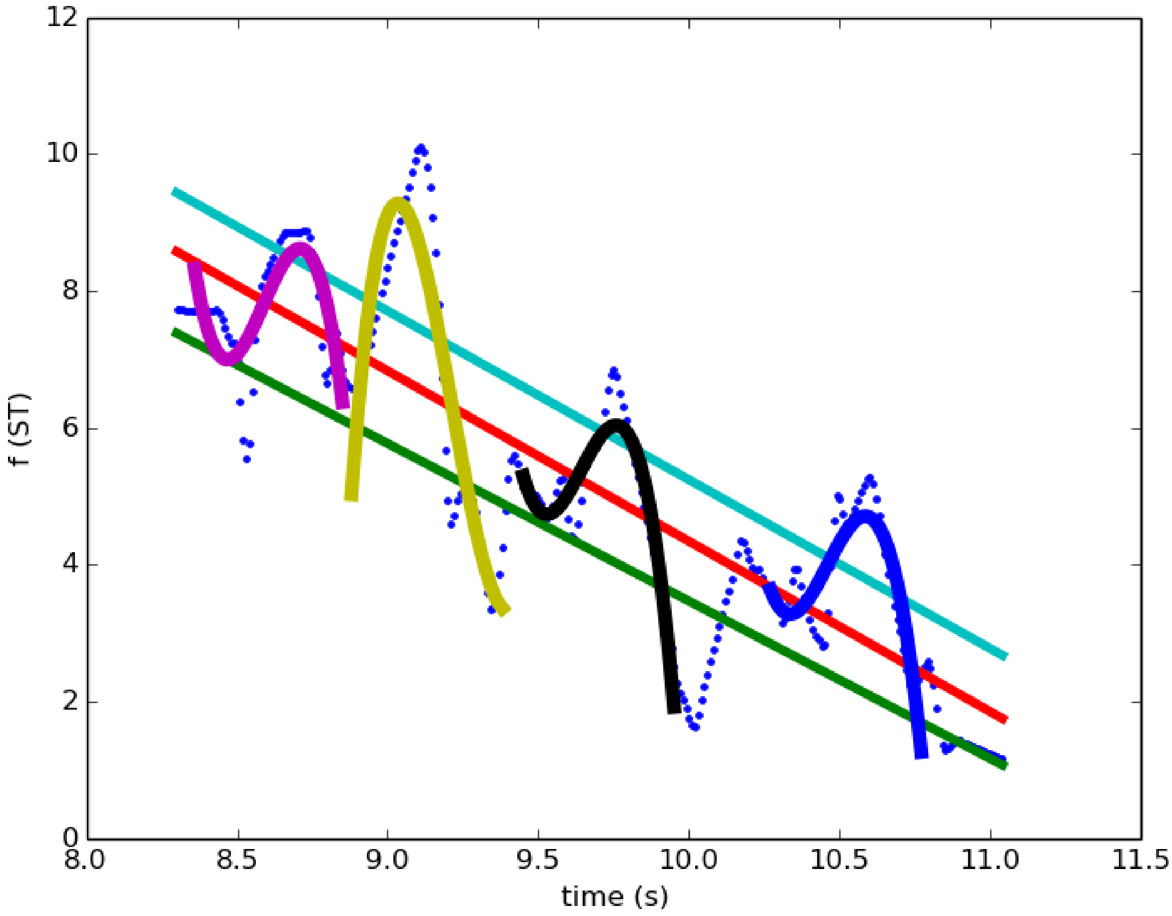
CoPaSul Manual
Contour-based, parametric, and superpositional intonation stylization

Uwe D. Reichel
Research Institute for Linguistics
Hungarian Academy of Sciences
uwe.reichel@nytud.mta.hu

Version 1.0.0, March 20th, 2021



Contents

1 Introduction **4**

2 Download and installation **4**

3 Call **5**

4 Input **6**
4.1 Audio files . . . 6
4.2 F0 files . . . 6
4.3 Pulse files . . . 6
4.4 Annotation files . . . 6
4.5 Annotation tiers . . . 7

5 F0 extraction **8**

6 Pulse extraction **8**

7 Automatic annotation **9**
7.1 Chunking . . . 9
7.2 Syllable nucleus and boundary extraction . . . 9
7.3 Prosodic phrase boundary location . . . 10
7.3.1 Percentile split . . . 10
7.3.2 Bootstrapping seed centroids for kMeans . . . 10
7.3.3 Bootstrapping seed centroids for percentile split . . . 11
7.3.4 Practical considerations . . . 11
7.3.5 Feature selection and weighting . . . 11
7.3.6 Output . . . 12
7.4 Pitch accent detection . . . 12
7.4.1 Percentile split . . . 12
7.4.2 Bootstrapping seed centroids for kMeans . . . 12
7.4.3 Bootstrapping seed centroids for percentile split . . . 13
7.4.4 Practical considerations . . . 13
7.4.5 Feature selection and weighting . . . 13
7.4.6 Output . . . 13

8 Stylization **13**
8.1 F0 preprocessing . . . 14
8.2 Energy calculation . . . 14
8.3 Analysis and normalization windows . . . 14
8.4 Superposition . . . 16
8.5 Global segments . . . 16
8.5.1 Annotation . . . 16
8.5.2 Register . . . 16
8.5.3 Contour classes . . . 17
8.6 F0 residual . . . 17
8.7 Local segments . . . 18
8.7.1 Annotation . . . 18
8.7.2 Contour stylization . . . 18
8.7.3 Contour classes . . . 19
8.7.4 Standard features . . . 19
8.7.5 Register features . . . 19
8.7.6 Gestalt features . . . 20
8.8 Standard features . . . 20
8.8.1 For f0 contours . . . 20
8.8.2 For energy contours . . . 20
8.9 Boundaries . . . 21
8.9.1 Of adjacent segments . . . 22
8.9.2 For fixed-length windows . . . 23
8.9.3 For global trends . . . 23
8.10 Rhythm . . . 23
8.10.1 Rhythmic aspects of the f0 contour . . . 24
8.10.2 Rhythmic aspects of the energy contour . . . 24

9 Voice quality **24**

10 Feature sets **25**

11 Configurations **28**
11.1 Sample rate . . . 29
11.2 Navigation . . . 29
11.2.1 Augmentation . . . 29
11.2.2 Feature extraction . . . 29
11.3 Directories, tiers, grouping . . . 32
11.4 F0 preprocessing, windowing . . . 38
11.5 Augmentation: Chunking . . . 39
11.6 Augmentation: Syllable nucleus detection . . . 41
11.7 Augmentation: Prosodic boundary detection . . . 41
11.8 Augmentation: Pitch accent detection . . . 43
11.9 Stylization: Global contours . . . 44
11.10Stylization: Local contours . . . 45
11.11Stylization: Register representation . . . 45
11.12Stylization: Boundaries . . . 45
11.13Stylization: General (energy) features . . . 46
11.14Stylization: F0 rhythm features . . . 47
11.15Stylization: Energy rhythm features . . . 48
11.16Stylization: Voice quality features . . . 49
11.17Clustering: Global contours . . . 49
11.18Clustering: Local contours . . . 50
11.19Plotting: Browsing . . . 51
11.20Plotting: Grouping . . . 53

12 Output **53**
12.1 Table files . . . 53
12.2 Summary table files . . . 54
12.3 Nested Python dictionary . . . 54
12.3.1 Configuration sub-dictionary . . . 54
12.3.2 Stylization feature table sub-dictionary . . . 55
12.3.3 Stylization feature nested sub-dictionary . . . 56
12.3.4 Clustering result sub-dictionary . . . 91
12.3.5 Validation sub-dictionary . . . 91
12.4 F0 files . . . 92
12.5 Log file . . . 92

13 Plotting **92**

14 Known bugs **93**

References **94**

1 Introduction

The purposes of the CoPaSul toolkit are (1) automatic prosodic annotation and (2) prosodic feature extraction from syllable to utterance level.

CoPaSul stands for *contour-based, parametric, superpositional* intonation stylization. The core model is introduced amongst others in [14]. In this framework intonation is represented as a superposition of global and local contours that are described parametrically in terms of polynomial coefficients. On the global level (usually associated but not necessarily restricted to intonation phrases) the stylization serves to represent register in terms of time-varying f0 level and range. On the local level (e.g. accent groups), local contour shapes are described. From this parameterization several features related to prosodic boundaries and prominence can be derived. Furthermore, by coefficient clustering prosodic contour classes can be derived in a bottom-up way. Next to the stylization-based feature extraction also standard f0 and energy measures (e.g. mean and variance) as well as rhythmic aspects can be calculated.

At the current state **automatic annotation** comprises:

- segmentation into interpausal chunks
- syllable nucleus extraction
- unsupervised localization of prosodic phrase boundaries and prominent syllables

F0 and partly also energy **feature sets** can be extracted for:

- standard measurements (as median and IQR)
- register in terms of f0 level and range
- prosodic boundaries
- local contour shapes
- bottom-up derived contour classes
- Gestalt of accent groups in terms of their deviation from higher level prosodic units
- rhythmic aspects quantifying the relation between f0 and energy contours and prosodic event rates

Please see section 10 for a list of application examples.

2 Download and installation

The CoPaSul command-line toolkit can be cloned or downloaded from this location:

`https://github.com/reichelu/copasul`

The code is written in Python 3 and depends on the following Python packages (with the specified version or higher):

- matplotlib >= 1.3.1
- numpy >= 1.8.2
- pandas >= 0.13.1
- scipy >= 0.15.1
- scikit_learn >= 0.17.1

So far the software is tested only for Linux!

The installation steps are:

1. unzip the copasul.zip in your target folder DIR
2. change to DIR
3. setup a virtual environment "venv_copasul", activate it, and install the requirements. For Linux this works as e.g. follows:

```
$ virtualenv --python="/usr/bin/python3" --no-site-packages venv_copasul
$ source venv_copasul/bin/activate
(venv_copasul) $ pip install -r requirements.txt
```

The target directory now contains a subfolder *src/* with several Python and Praat scripts [4] as well as a *minex/* and *config/* subfolder:

src/: Software Python	
copasul.py	the main script
copasul_root.py	specifies the copasul tool path
copasul_init.py	initializations
copasul_preproc.py	data preprocessing
copasul_augment.py	automatic prosodic annotation
copasul_styl.py	stylization
copasul_clst.py	contour clustering
copasul_resyn.py	f0 generation
sigFunc.py	signal processing
mylib.py	general function collection
src/: Software Praat	
extract_f0.praat	f0 extraction for mono files
extract_f0_stereo.praat	f0 extraction for stereo files
extract_pulse.praat	pulse extraction for mono files
extract_pulse_stereo.praat	pulse extraction for stereo files
doc/: Documentation	
LEGAL	license, disclaimer
copasul_commented_config.json.txt	configuration explanations
copasul_manual_latest.pdf	latest version of this document
history.txt	update log file
config/: example configurations	
minex/: minimal example	

3 Call

The main script *copasul.py* can be used from the terminal or within the Python3 environment. After having changed to the copasul directory, from the shell for the minimal example it is called as follows:

```
(venv_copasul) $ cd src/
(venv_copasul) $ python copasul.py -c ../config/minex.json
```

Input and output of this minimal example can be found in the subfolder *minex/*. The content of the configuration file is explained in section 11. Within the Python environment the tool is used this way (replace *my/path/to/Copasul* and *my/path/to/* accordingly):

```
>>> import sys
>>> import json
>>> sys.path.append('my/path/to/Copasul/src')
>>> import copasul
>>> with open('my/path/to/minex.json', 'r') as h:
...      opt = json.load(h)
>>> fex = copasul.Copasul()
>>> copa = fex.process(config=opt)
```

The input argument for *fex.process()* is a dictionary which contains the configurations (see section 11). *fex.process()* returns the output dictionary *copa* with the extracted features (see section 12.3). The feature tabels are stored as Pandas Dataframes with alphanumerically sorted columns and can be accessed as follows:

```
>>> copa['export']['loc']
```

loc refers to local contour parameters. Other feature sets are *glob* (global contour parameters), *gnl_f0* (standard f0 features), *gnl_en* (standard energy features), *rhy_f0* (f0 rhythm features) *rhy_en* (energy rhythm features) *bnd* (boundary features), and *voi* voice quality features. All standard and rhythm features are additionally calculated on the file level (*gnl_f0_file* ... *rhy_en_file*).

In case feature extraction should not start from scratch, but an already existing output should be corrected or expanded, it can be passed to *fex.process()* with the argument *copa* as follows.

```
>>> copa = fex.process(config=opt, copa=copa)
```

For shell calls as well as for calls within the Python environment the stylization output is written to a binary Pickle file and to CSV table files as specified in the configurations. See section 12.

4 Input

For automatic annotation CoPaSul needs audio and f0 table files. For feature extraction it additionally needs annotation files. For the *voice* feature set furthermore pulse table files are needed. Corresponding files do not necessarily need to have the same name stem, but it is assumed that all audio, f0, and annotation files are sorted the same. An example can be found in the *input* subdirectory.

Additionally a configuration file in *JSON* format is needed as further specified in section 11.

4.1 Audio files

Currently only wav files are supported. The files can be mono or stereo. For conversion to wav, e.g. *Praat, Audacity*, or *Sox* software can be used.

4.2 F0 files

Plain text files. Tables with whitespace column separator. The first column contains time information. All further columns contain the f0 of the respective channel. For mono files f0 tables thus consist of 2 columns, for stereo files of 3, etc. All columns need to have the same lengths. Undefined f0 values are to be replaced by 0. Only 100 Hz sample rate is supported, and resampling is carried out from other rates. The Praat scripts *extract_f0.praat* and *extract_f0_stereo.praat* which are contained in this package provide the required input format.

4.3 Pulse files

Plain text files. Only needed for the *voice* feature extraction. Tables with whitespace column separator. Each column contains the pulse time stamps for one channel in seconds. All columns must contain the same number of rows so that for files with more than one channel -1 has to be padded to the shorter columns. The Praat scripts *extract_pulse.praat* and *extract_pulse_stereo.praat* which are contained in this package provide the required input format.

4.4 Annotation files

The Praat TextGrid format (long and short) and an XML format of the following form are supported.

```
<?xml version="1.0" encoding="UTF-8"?>
<annotation>
  ...
  <tiers>
    <tier>
      <name>mySegmentTier</name>
      <items>
        <item>
          <label>x</label>
          <t_start>0.3</t_start>
          <t_end>0.9</t_end>
        </item>
        ...
      </items>
    </tier>
    <tier>
      <name>myEventTier</name>
      <items>
        <item>
          <label>y</label>
          <t>0.7</t>
        </item>
        ...
      </items>
    </tier>
    ...
  </tiers>
  ...
</annotation>
```

The tiers need to be stored in the *tiers* subtree right below the root element.

Each tier must have a name assigned by the element *name*. The items of each tier are collected in the *items* subtree, in which each item is stored in an *item* subtree.

Segment tiers (see next section) must contain the elements *label, t_start, t_end*. Event tiers must contain the elements *label, t*.

The XML annotation file can be extended by the user as long as it fulfills the specified requirements in the *tiers* subtree.

4.5 Annotation tiers

In the following the notation `a:b:...` refers to branches through the configuration dictionary which is introduced in section 11. The annotation files can contain tiers of the following types:

Segment tiers contain items defined by a label, a start point and an endpoint. They correspond to Praat IntervalTiers.

Event tiers contain items without a temporal extension. They are defined by a label and a time stamp and correspond to Praat TextTiers.

Both segment tiers and event tiers are supported for most of the analyses. Wherever needed, an event is converted to a segment by centering a window of length `preproc:point_win` on the event as is explained in more detail in section 8.3. Pause information can only be extracted for segment tiers. In TextGrids pauses are considered to be items with empty labels or labeled as `fsys:label:pau`. Both event and segment tiers can serve as:

Analysis tiers In the context of automatic annotation these tiers contain or limit the candidate locations for prosodic events. Can be segment or event tiers.

```
fsys:augment:glob:tier
fsys:augment:loc:tier_acc
fsys:augment:loc:tier_ag
```

For feature extraction these segment or event tiers define the units of analysis.

```
fsys:chunk:tier
fsys:glob:tier
fsys:loc:tier_acc
fsys:loc:tier_ag
fsys:bnd:tier
fsys:gnl_f0:tier
fsys:gnl_en:tier
fsys:rhy_f0:tier
fsys:rhy_en:tier
```

Parent tiers Parent tiers (1) limit the analysis and normalization windows by their segment boundaries. As an example, normalization across chunk boundaries can be suppressed. (2) They limit the domain of global trends against which local deviation is measured. It's strongly recommended to use segment tiers for this purpose. If not specified, the whole file is treated as a single parenting segment. For automatic annotation parent tiers are to be defined by:

```
fsys:augment:syl:tier_parent
fsys:augment:glob:tier_parent
fsys:augment:loc:tier_parent
```

For *glob, bnd, gnl_en, gnl_f0, rhy_en, rhy_f0* feature extraction (see section 10) only speech chunks can serve as parent domains:

```
fsys:chunk:tier
```

Fallback is again the entire file. For *loc* feature extraction only the segments of the *glob* analysis tier can form the parent domain due to the:

Superpositional framework Within the CoPaSul approach (see section 8.4) the intonation contour is considered as a superposition of a global and local components. Their domains are defined by the *glob* and *loc* option branches, respectively:

```
fsys:glob:tier
fsys:loc:tier_acc
fsys:loc:tier_ag
```

This has two implications on the annotation tier definitions:

- for each channel only one tier is supported each for the global and the local local domain
- the global domain tier is treated as the parent tier for the local domain tier

Output tiers For automatic annotation these tiers are defined by a stem which is always expanded by the recording channel index.

```
fsys:augment:chunk:tier_out_stm
fsys:augment:syl:tier_out_stm
fsys:augment:glob:tier_out_stm
fsys:augment:loc:tier_out_stm
```

As an example, given a stereo file and the chunk output tier name CHUNK, the tiers CHUNK_1 and CHUNK_2 will be added to the annotation file. For the sake of an uniform treatment, also for mono files the channel index will be added.

Tier specification For all tiers, that were not automatically generated, the user needs to specify the recording channel index it refers to (also for mono files!), e.g.:

```
fsys:channel:'tierA'=1
fsys:channel:'tierB'=2
```

tierA thus refers to channel 1, and *tierB* to channel 2. Tier names can be specified as strings, or as list of strings.

```
fsys:bnd:tier='tierA'
```

means, that the *bnd* feature extraction is to be carried out for units defined by the content of *tierA*.

```
fsys:bnd:tier=['tierA','tierB']
```

triggers a *bnd* feature extraction for the content of two tiers. The channels the specified tiers refer to are looked up in `fsys:channel:*`.

The name stem of a tier resulting from automatic annotation (e.g. CHUNK) will be expanded automatically, thus for a chunked stereo file these two specifications are equivalent:

```
fsys:bnd:tier='CHUNK'
fsys:bnd:tier=['CHUNK_1', 'CHUNK_2']
```

For the feature sets *bnd, gnl_en, gnl_f0, rhy_en, rhy_f0* (see section 10) an arbitrary number of tiers can be specified for each channel. For *chunk, glob, loc* only one tier per channel is supported.

5 F0 extraction

For f0 extraction in mono or stereo wav files the two Praat scripts contained in this package can be used.

They can be called this way:

```
> praat extract_f0.praat myStepsize myMinFreq myMaxFreq \
                        myAudioInputDir myF0OutputDir myAudioExt myF0Ext
```

The usage of extract_f0_stereo.praat is the same. Note that subsequent stylization **in any case initiates a resampling to 100 Hz**, so that myStepsize here can be directly set to 0.01. *myMinFreq* and *myMaxFreq* refer to the minimum and maximum of allowed f0 values in Hz. Values below or above are considered as measurement errors and are set to 0. The f0 range choice depends on the recorded speakers. As a rule of thumb the parameters can be set to 50 and 400 Hz, respectively. In my *myAudioInputDir* the sound files with the extension *myAudioExt* are collected, and corresponding f0 plain text table files with the audio file's name stem and the extension *myF0Ext* are outputted to the directory *myF0OutputDir*.

6 Pulse extraction

Pulse extraction is needed for the *voice* feature set only. For its extraction in mono or stereo wav files the two Praat scripts contained in this package can be used.

They can be called this way:

```
> praat extract_pulse.praat myMinFreq myMaxFreq \
                        myAudioInputDir myPulseOutputDir myAudioExt myPulseExt
```

The usage of extract_pulse_stereo.praat is the same. The scripts make use of Praat's *To PointProcess (cc)* routine operating on sound and pitch objects. For pitch object creation the minimum and maximum of allowed f0 values *myMinFreq* and *myMaxFreq* need to be specified in Hz. In my *myAudioInputDir* the sound files with the extension *myAudioExt* are collected, and corresponding pulse plain text table files with the audio file's name stem and the extension *myPulseExt* are outputted to the directory *myPulseOutputDir*.

7 Automatic annotation

Automatic unsupervised prosodic annotation comprises chunking, syllable nucleus and boundary extraction, prosodic phrase extraction, and pitch accent localization. Details of the algorithms will be given in [16]. At the beginning of each introductory paragraph it is specified:

navigation: *which navigation option to set to True in the configuration file (see section 11)*
feature sets: which feature sets result from the annotation (see section 10)
option sub-dictionary: *which configuration sub-dictionaries serve to customize the respective processing (see section 11)*
output sub-dictionary: *which subdirectory of the resulting python nested dictionary contains the extracted feature set (see section 12.3).*

Paths through the configuration dictionary are referred to by `my:path:to:option`.

navigation: *do_augment_**
feature sets: –
option sub-dictionary: *fsys:augment:*:*; augment:*:**
output sub-dictionary: *(augmented annotation file)*

7.1 Chunking

navigation: *do_augment_chunk*
feature sets: –
option sub-dictionary: *fsys:augment:chunk:*; augment:chunk:**
output sub-dictionary: *(augmented annotation file)*

Chunking serves to segment the utterance into interpausal units. It is based on a pause detector, that works the following way: an analysis window w_a with length `augment:chunk:l` is moved over the lowpass-filtered signal together with a longer reference window w_r of length `augment:chunk:l_ref` with the same midpoint. A pause is set where the mean energy in w_a is below a threshold defined relative to the energy in w_r, i.e. if $e(w_a) < e(w_r) \cdot$ *augment:chunk:e_rel*. Chunks are then trivially assigned to interpausal intervals. Silence margins can be set at chunk starts and ends by `augment:chunk:margin`.

If w_r itself is identified as a pause by $e(w_r) < e(s) \cdot$ *augment:chunk:e_rel* it is replaced by s; where s consists of selected parts of the acoustic signal in the analysed channel with absolute amplitude values above the median. By this lower threshold the robustness against a high occurrence of speech pauses is increased.

The filtering of the signal can be customized by the sub-dictionary `augment:chunk:flt`. In there `btype` gives the Butterworth filter type (*high, low, band, or none*), `f` the cutoff frequencie(s), and `ord` the order. For pauses as well as for inter-pause intervals minimum lengths can be defined by `augment:min_pau_l` and `min_chunk_l`, respectively. Pauses are then merged across too short chunks, and chunks are merged across too short pauses. The segment tier output will be added to the annotation file. The tier name is specified by `fsys:augment:chunk:tier_out_stm` concatenated with the respective channel index. Standard labels 'x' are assigned to chunk segments, and `fsys:label:pau` to the pauses inbetween.

7.2 Syllable nucleus and boundary extraction

navigation: *do_augment_syl*
feature sets: –
option sub-dictionary: *fsys:augment:syl:*; augment:syl:**
output sub-dictionary: *(augmented annotation file)*

For syllable nucleus detection the method proposed by [13] is adopted. Again an analysis w_a with length `augment:syl:l` and a longer reference window w_r of length with length `augment:syl:l_ref` with the same midpoint are moved along the signal, which this time is band-pass filtered to focus on the frequency band related to vocalic nuclei. The filter specification in `augment:syl:flt` works as described for chunking. From this energy contour the local maxima are extracted. If for a local maximum the mean energy in w_a supersedes the mean energy in w_r by a defined factor, i.e. if $e(w_a) > e(w_r) \cdot$ *augment:syl:e_rel*, and if $e(w_a)$ is not below a defined fraction of the energy in the current chunk w_c (fallback: whole file), i.e. $e(w_a) \geq e(w_c) \cdot$ *augment:syl:e_min*, a syllable nucleus is set. From which tier to get the current chunk is to be defined by `augment:syl:tier_parent`. E.g. it can be the output tier of a preceding chunking step. A further constraint `augment:syl:d_min` specifies the minimum distance between subsequent syllable nuclei. If two nuclei are too close, they are merged to a single syllable and the point of energy maximum in this interval is assigned to be the nucleus.

Subsequently syllable boundaries are assigned to the energy minimum between adjacent syllable nuclei. They just serve as fallback prosodic boundary candidates.

The output consists of two event tiers for syllable nuclei and boundaries and will be added to the annotation file. The tier name is specified by `fsys:augment:syl:tier_out_stm`. For the nuclei it is concatenated with the respective channel index. For the boundaries it is concatenated with a 'bnd' infix and the channel index. Standard labels 'x' are assigned for both tiers.

7.3 Prosodic phrase boundary location

navigation: *do_augment_glob*
feature sets: –
option sub-dictionary: *fsys:augment:glob:*; augment:glob:**
output sub-dictionary: *(augmented annotation file)*

Prosodic phrase boundary decisions are based on nearest centroid classification. The user needs to specify the tier that contains boundary candidates in `fsys:augment:glob:tier`. For segment tiers these candidates are the segment boundaries, for event tiers, the candidates are the time stamps. If no tier is specified, syllable boundaries derived by step 7.2 will be selected as candidates. At each boundary candidate a feature set is extracted that had been proven to be related to prosodic boundaries in former studies [21, 22]. This feature set is introduced in section 8.9. The user needs to specify which of these features should be selected by

`augment:glob:wgt:myBndFeatset+:myRegister+:myFeat+`.

In case a phone segment tier is available and if centroids are derived from the entire data set and not separately for each file (see below), in addition z-scored vowel length can be used as a feature. The length of the vowel associated with the prosodic event candidate is divided by its mean length derived from the entire dataset. The associated vowel is the last vowel segment with an onset before the boundary candidate time stamp. The length feature can be added by:

`augment:glob:wgt:pho=1`

The phonetic segment tiers (one for each channel) are to be specified in

`fsys:pho:tier`

Vowels are identified in these tiers by a regular expression stored in

`fsys:pho:vow`

This feature will be beneficial for languages in which phrase boundaries and/or accents are marked by phone segment lengthening.

Furthermore the user can select whether the current feature values at time i, v_i, or the delta values (i.e. the differences to the preceding values $v_i - v_{i-1}$) or both should be taken:

`augment:glob:measure`

Some features require units from a parent tier which is to be specified by `augment:glob:tier_parent`, e.g. to measure local f0 trend discontinuities within a superordinate unit and to limit analysis and normalization windows. Such units are e.g. chunks derived from preceding chunking. Fallback is the entire file.

From the features for each of the two classes *boundary B* and *no boundary NB* a centroid can be bootstrapped in several ways given the specification in `augment:glob:cntr_mtd` as described in the following sections. Centroids can be calculated separately for each file or over the entire data set by setting the value of

`augment:glob:unit`

to *file* or *batch*, respectively. The latter is strongly recommended for corpora containing lots of short recordings.

7.3.1 Percentile split

```
augment:glob:cntr_mtd=split
augment:glob:prct=mySplitPoint
```

Since for all extracted pause length and pitch discontinuity boundary features are positive correlation has been found to perceived boundary strength [21, 22] B and NB centroids can be straight-forwardly derived from high and low feature values, respectively. Centroids are thus derived by splitting each column in the feature matrix at the percentile `augment:glob:prct`. The B centroid is defined by the median of the values above the splitpoint, the NB centroid by the median of the values below. All feature vectors are then assigned to the nearest centroid in a single pass. Boundaries are subsequently inserted at all candidate time points classified as B. This method works for both segment and event tier input.

7.3.2 Bootstrapping seed centroids for kMeans

```
augment:glob:cntr_mtd=seed_kmeans
augment:glob:min_l=myMinPhraseLength
```

This procedure works for segment tier input only since it makes use of pauses between adjacent segments. As visualized in Figure 1 B and NB centroids are bootstrapped based on two assumptions: (1) each pause indicates a prosodic boundary, and (2) prosodic phrases have a minimum length, thus in the vicinity of pauses there are no further boundaries. KMeans clustering is then initialized by these two centroids and subdivides all candidates into the B and NB cluster. Boundaries are inserted at all candidate time points belonging to the B cluster.

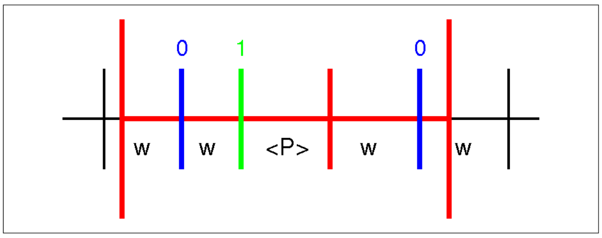


Figure 1: Bootstrapping seed centroids for the classes 1 (boundary) and 0 (no boundary). Word boundaries are indicated by the short vertical lines. Assumptions: each pause indicates a prosodic boundary (green), and prosodic phrases have a minimum length (red window), thus in the vicinity of pauses there are no further boundaries (blue).

7.3.3 Bootstrapping seed centroids for percentile split

```
augment:glob:cntr_mtd=seed_prct
augment:glob:prct=mySplitPoint
augment:glob:min_l=myMinPhraseLength
```

The seed centroid bootstrapping works as for the preceding method. Instead of kMeans, for the remaining feature vectors the Euclidean distance to the NB seed centroid is calculated. Vectors with a distance above the *mySplitPoint*-th percentile of all measured distances are assigned to the B class, the others to the NB class.

7.3.4 Practical considerations

The percentile split method works for both segment and event tiers, whereas the two centroid bootstrapping methods need segment tier input to infer pause locations. For the two percentile split approaches, the parameter `augment:glob:prct` serves to control for the number of inserted boundaries. The higher, the smaller the B class, thus the fewer boundaries will be assigned.

If a text transcription is at hand the user can ensure that prosodic boundaries only occur at word boundaries by preceding signal-text alignment, e.g. by WebMAUS [26, 9].

Heuristics

```
augment:glob:heuristics=ORT
```

If set by the user, this heuristics assumes a word segment tier as input and rejects boundaries after too short and thus probably function words ($< 0.2s$)

7.3.5 Feature selection and weighting

```
augment:glob:wgt:myBndFeatset+:myRegister+:myFeat+=myWeight
augment:glob:wgt_mtd=myWeightingMethod
```

By the `augment:glob:wgt:myBndFeatset+:myRegister+:myFeat+` branches the user at the same time selects and weights features. As an example

```
augment:glob:wgt:win:ml:rms=1
```

selects the feature `rms` derived from the register representation `ml` within the boundary feature set `win` (see sections 8.9 and 10 for explanations). If the weighting method in `augment:glob:wgt_mtd` is set to 'user', the weight of this feature becomes 1. If no weighting is intended, to all selected features should be assigned the same weight. As an alternative to the definition by the user, weights can also be extracted by correlation to the median or by the cluster silhouette measure.

Correlation Each feature is correlated with the medians of the feature vectors. Since as mentioned all boundary features are expected to be positively correlated to boundary strength, and since the median is expected to be more robustly related to boundary strength than single features, the correlation between a feature and the medians to some extend reflects the goodness of this feature to predict boundary strength. Features with a negative correlation to the median will be removed from the pool. All remaining correlations are transformed to weights summing up to 1 by dividing them by the sum of correlations.

Silhouette The mean silhouette over all clustered data points measures how well clusters can be separated. Here it is measured separately for each feature within the clearly assignable feature vectors from which the B and NB seed centroids were derived. It is minmax-normalized to the range [0 1].

7.3.6 Output

The output consists of a segment tier for each channel with the name `fsys:glob:tier_out_stm` + channelIndex. Each segment spans the interval between two subsequent B events. If `fsys:glob:tier` is a segment tier, then pauses are taken over from this tier. Standard labels 'x' are assigned to the prosodic phrase segments.

7.4 Pitch accent detection

Pitch accents are derived in an analogous bootstrap fashion as prosodic boundaries. The user needs to specify an event tier (default: syllable nuclei) for localization of the pitch accent candidates. Furthermore the user can specify a segment tier (e.g. words) to restrict the maximum number of detected pitch accents within each segment to 1.

```
fsys:augment:loc:tier_acc
fsys:augment:loc:tier_ag
```

Given a segment tier, the user can furthermore specify (1) whether each segment should get an accent or only the prominent ones

```
augment:loc:ag_select
```

and (2) where within a segment an accent should be placed: left- or rightmost, e.g. for prosodically left- or right-headed languages, or on the most prominent candidate.

```
augment:loc:acc_select
```

Prominence can be parameterized by several feature sets measuring standard f0 and energy features, contour shapes within local segments and their deviation from a global declination trend.

The user can select whether the current feature values at time i, v_i, or the delta values (i.e. the differences to the preceding values $v_i - v_{i-1}$) or both should be taken:

```
augment:loc:measure
```

Some features require units from a parent tier which is to be specified by `augment:loc:tier_parent`, e.g. to measure local f0 deviations relative to some superordinate unit and to limit analysis and normalization windows. Such units are e.g. prosodic phrases derived from preceding phrase extraction. Fallback is the entire file.

From these features for each of the two classes *accented A* and *not accented NA* a centroid can be bootstrapped in several ways analogously to the prosodic boundary extraction, this time given the specification in `augment:loc:cntr_mtd`.

Centroids can be calculated separately for each file or over the entire data set by setting the value of

```
augment:loc:unit
```

to *file* or *batch*, respectively. The latter is strongly recommended for corpora containing lots of short recordings.

7.4.1 Percentile split

```
augment:loc:cntr_mtd=split
augment:loc:prct=mySplitPoint
```

Given a user-defined feature set where for each feature high values indicate prominence A and NA centroids can be straight-forwardly derived from high and low feature values, respectively. Centroids are thus derived by splitting each column in the feature matrix at the percentile `augment:loc:prct`. The A centroid is defined by the median of the values above the splitpoint, the NA centroid by the median of the values below. All feature vectors are then assigned to the nearest centroid in a single pass. Boundaries are then inserted at all candidate time points classified as B. This method works for both segment and event tier input.

7.4.2 Bootstrapping seed centroids for kMeans

```
augment:loc:cntr_mtd=seed_kmeans
augment:loc:max_l_na=myMaxLengthNA
augment:loc:min_l_a=myMinLengthA
augment:loc:min_l=myMinLengthAG
```

This procedure works only if a segment tier is provided next to the event tier, and if this segment tier contains word-like units. As for the phrase boundary detection described above there are 2 (this time even more) simplifying assumptions to derive seed centroids for cluster initialization (cf. Figure 2): (1) each word longer than `augment:loc:min_l_a` contains an accent, due to its expected high information content. (2) each word shorter than `augment:loc:max_l_na` does not contain an accent due to its expected low information content. Depending on `augment:loc:acc_select` the A centroid is then calculated from all leftmost, rightmost, or most prominent `tier_acc` candidates in the `tier_ag` segments fulfilling criterion (1). The NA centroid is calculated from all `tier_acc` candidates in in the `tier_ag` segments fulfilling criterion (2). KMeans clustering is then initialized by these two centroids and subdivides all candidates into the A and NA cluster. Multiple A cases within the same segment are reduced by `augment:loc:acc_select`. Furthermore, among A cases closer than `augment:loc:min_l` only the more prominent ones are kept.

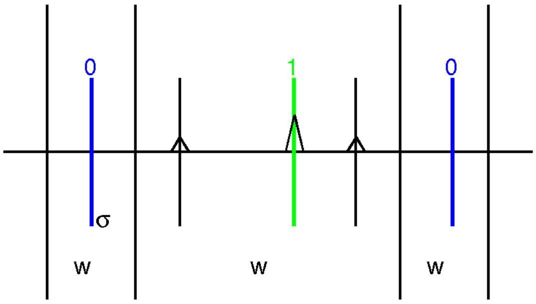


Figure 2: Bootstrapping seed centroids for the classes 1 (accent) and 0 (no accent). Word boundaries are indicated by long vertical lines, and syllable nuclei by short vertical lines. Prominence is encoded by the size of the triangles. Assumptions: each word longer than some threshold contains an accent (green); each word shorter than some threshold does not contain an accent (blue). Within the accented word the accent is placed on the most prominent syllable (as in this example), or on the left- or rightmost syllable.

7.4.3 Bootstrapping seed centroids for percentile split

```
augment:loc:cntr_mtd=seed_prct
augment:loc:prct=mySplitPoint
augment:loc:max_l_na=myMaxLengthNA
augment:loc:min_l_a=myMinLengthA
```

The seed centroid bootstrapping works as for the preceding method. Instead of kMeans, for the remaining feature vectors the Euclidean distance to the NA seed centroid is calculated. Vectors with a distance above the *mySplitPoint*-th percentile of all measured distances are assigned to the A class, the others to the NA class.

7.4.4 Practical considerations

The percentile split method works with and without segment tiers, whereas the two centroid bootstrapping methods need segment tier input next to the event tier to infer word length. As with boundary detection, the parameter `augment:loc:prct` serves to control for the number of assigned accents. The higher, the smaller the A class, thus the fewer accents will be assigned.

As mentioned for prosodic boundary detection, a supporting word segmentation can be derived by preceding signal-text alignment, e.g. by WebMAUS [26, 9].

7.4.5 Feature selection and weighting

```
augment:loc:wgt:myFeatset+:...
```

The same selection and weighting mechanisms apply as described in section 7.3.5.

The following feature sets can be used: *acc, gst, gnl_f0, gnl_en* (see section 10). In section 11 examples are given how to expand the corresponding configuration branches.

As for boundary detection also for pitch accent detection z-scored vowel length can be added to the feature set. The vowel interval associated to a pitch accent candidate includes the candidate's time stamp. See section 7.3 for further details. The length feature can be added by:

```
augment:loc:wgt:pho=1
```

7.4.6 Output

The output consists of an event tier for each channel with the name `fsys:loc:tier_out_stm` + channelIndex. Standard labels 'x' are assigned to each accent.

8 Stylization

In the following the f0 preprocessing and the f0 and energy stylization steps are introduced. For each stylization step it is specified:

navigation: *which navigation option to set to True in the configuration file (see section 11)*
feature sets: which feature sets result from the stylization (see section 10)
option sub-dictionary: *which configuration parts serve to customize the respective processing (see section 11)*
output sub-dictionary: *which part of the resulting Python nested dictionary variable contains the extracted feature set (see section 12.3).*

Branches through the configuration as well as trough the result dictionary are referred to by `my:branch:to:value`.

8.1 F0 preprocessing

F0 preprocessing comprises resampling to 100 Hz, outlier detection, interpolation over outliers and voiceless utterance parts, smoothing, and semitone conversion including speaker normalization.

Outliers Outliers are identified separetely for each channel in a file. They are defined in terms of deviation from a mean value or from the 1st and 3rd quartile. The deviation factor is controlled by `preproc:out:f`, and the reference point by `preproc:out:m`. For `m=mean` outliers lie outside the interval $[m - f \cdot \text{sd}, m + f \cdot \text{sd}]$. For `m=median` outliers lie outside of $[m - f \cdot \text{iqr}, m + f \cdot \text{iqr}]$. For `m=fence` outliers lie outside of $[Q_1 - f \cdot \text{iqr}, Q_3 + f \cdot \text{iqr}]$ (sd: standard deviation; iqr: interquartile range; Q1, Q3: 1st and 3rd quartile).

Interpolation Only linear interpolation is supported. Horizontal extrapolation is carried out at file boundaries.

Smoothing The smoothing method is chosen by `preproc:smooth:mtd`. Median and Savitzky-Golay filtering are supported. Median filtering yields smoother contours, while Savitzky-Golay better preserves local f0 maxima and minima. The higher the window length `preproc:smooth:win`, the more smooth the contours. For the Savitzky-Golay filtering the polynomial order needs to be specified by `preproc:smooth:ord`. The lower, the more the result gets smoothed away from the input data.

navigation: *do_preproc*
feature sets: –
option sub-dictionary: *preproc:**
output sub-dictionary: *data:myFileIdx:myChannelIdx:f0:**

Semitone conversion If `preproc:st=1`, Hertz (Hz) values are transformed to semitones (st) as follows: $F0_{st} = 12 \cdot log_2(\frac{F0_{Hz}}{b})$. b is a base value which is calculated separately for each channel in each f0 file. It is defined as the median of the values below the percentile `preproc:base_prct` and can be used for f0 normalization by file and channel. Alternatively, a grouping variable can be specified, so that for each of its levels a separate f0 base value is calculated. This is done by `preproc:base_prct_grp`. There it can be specified which grouping variable is to be assigned to each channel. The grouping variable must be encoded in the filename and must be extractable from `fsys:grp:lab`. An example: you have stereo f0 files with the name pattern *speakerChannel1_speakerChannel2*. And you want to calculate separately for each speaker an f0 base value which is the median of the values below the 5th percentile over all this speaker's utterances in the corpus. This is to be configured as follows:

```
fsys:grp:src='f0'
fsys:grp:sep='_'
fsys:grp:lab=['speakerChannel1','speakerChannel2']
preproc:base_prct=5
preproc:base_prct_grp:'1'='speakerChannel1'
preproc:base_prct_grp:'2'='speakerChannel2'
```

This assigns to each channel the grouping variable to be read from the f0 file names. Note, that (1) channel indices need to be written in quotation marks, and (2) a shared semantics across the grouping variables is assumed. E.g. just one base value will be calculated for speaker x, regardless whether she was recorded in channel 1 or 2.

Base value subtraction If `preproc:st` is 0, the base value introduced in the preceding paragraph will be subtracted from the f0 contour without semitone conversion. If you don't want to use any base value, neither for subtraction nor as conversion reference, set `preproc:base_prct=0`.

8.2 Energy calculation

The energy contour is simply represented in terms of the root mean squared deviation (RMSD) within the windowed signal. The relevant parameters can be found below `styl:gnl_en` `styl:rhy_en:sig`. `win` defines the window length and `sts` the stepsize. The energy value sample rate is thus $1/sts$. *wintyp* and `winparam` give the window type and an additional parameters passed on the *get_window()* function of the *scipy.signal* module. For customizing energy extraction with other than default values, please consult the *scipy.signal* documentation for *get_window(). wintyp* and *winparam* can contain any value specified in this documentation.

8.3 Analysis and normalization windows

Windows serve (1) to transform time stamps from an event tier to segments, and (2) to locally normalize feature values.

Time stamps to segments Most feature sets are calculated for segments, not for time stamps. Thus event tier input is converted to segments by centering a symmetric analysis window with the length `preproc:point_win` on each time stamp as shown in Figure 3. Features are then extracted within this window. The window can also be separately specified for each feature set by `preproc:myFeatureSet:point_win`. For local contour stylization a segment and an event tier can be processed in parallel as explained in section 8.7.

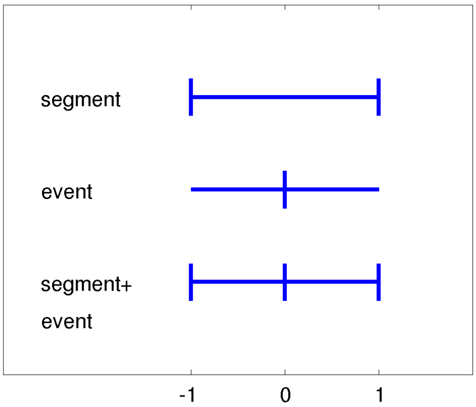


Figure 3: Segment and event tier input. A symmetric analysis window is centered on events. For local contour stylization, segment and event tiers can be integrated for time normalization: the event is set to 0, the pre-event part of the segment to $[-1\ 0[$, and the post-event part of the segment to $]0\ 1]$.

Normalization For the feature sets *loc, gnl_f0* and *gnl_en* several feature values are additionally locally normalized to capture their relative amount compared to the local environment. This environment length is defined by `preproc:nrm_win`. For event tier input the normalization window is centered on each time stamp. For segment tier input, it is centered on the midpoint of each segment. For parallel segment and event tier input which can be provided for *loc* feature extraction, the window is centered on the event's time stamp within the segment. The window can also be separately specified for each feature set by `preproc:myFeatureSet:nrm_win`.

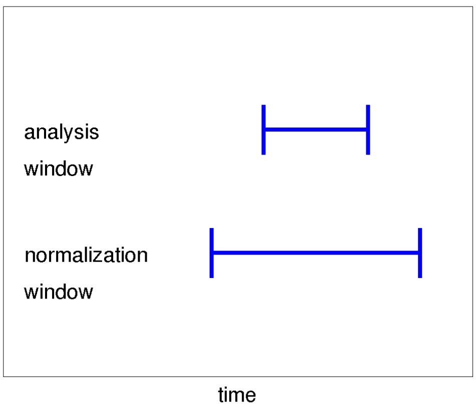


Figure 4: Analysis and longer normalization window. The values derived in the analysis window are divided by the corresponding values in the normalization window.

Window constraints Analysis and normalization window are limited to the corresponding segment in the parent tier domain. For *loc* features this domain is given by the global segment tier. For the other features it is given by the speech chunk tier if this tier is defined in *fsys:chunk:tier*. This means that analysis and normalization is not carried out across global segments or chunks, respectively. An exception can be made for the *bnd* feature set, that might be meaningful for chunk boundaries, too. If so, `styl:bnd:cross_chunk` is to be set to 1. For segment tier input the minimum length of the normalization window is set to the length of the respective segment. This implies that for segments longer than the defined normalization window, normalized feature values are the same as the not normalized ones.

navigation: *do_preproc*
feature sets: –
option sub-dictionary: *preproc:**
output sub-dictionary: *data:myFileIdx:myChannelIdx:...:*{*t*|*to*|*tn*}

8.4 Superposition

The core concept of CoPaSul is to represent an f0 contour as a superposition of linear global component and polynomial local components as shown in Figure 5.

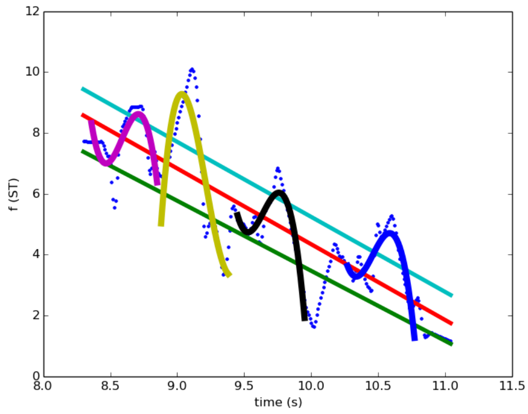


Figure 5: Superposition of one global and four local contours.

Stylization is carried out as follows: Within each global segment of the tier `fsys:glob:tier` (e.g. an intonation phrase) a linear register level and range representation is fitted. After subtraction of this global component, within each local segment an n-th order polynomial is fitted to the f0 residual. As an alternative to register level subtraction, the f0 residual can also be derived by normalization of the contour to the register range.

8.5 Global segments

8.5.1 Annotation

In the annotation files global segments can be defined in 2 ways:

1. by start and end point (segment tier input specified in `fsys:glob:tier`)
2. by the segments' right end points (event tier input specified in `fsys:glob:tier` that contains e.g. break index labels)

In the second case the events are expanded to segments between the annotated boundary time stamps. Pauses marked by an empty label or a pause label (`fsys:label:pau`) are skipped and the onset of the subsequent segment is set to the end of the pause. Therefore, in point tiers pauses should be marked at their right end. Furthermore, if chunks are provided by `fsys:tier:chunk`, then the expanded segments do not cross chunk boundaries but end and start with the boundaries of the respective chunk they are part of.

8.5.2 Register

Global segments are represented in terms of a time-varying f0 register. Register aspects are level (midline) and range (topline − baseline).

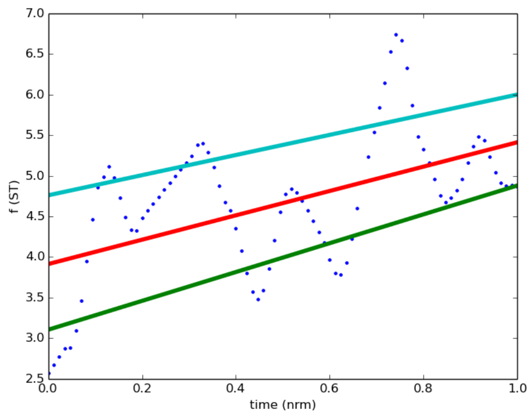


Figure 6: Register (level and range) stylization in global contour segments.

navigation: *do_styl_glob*
feature sets: glob
option sub-dictionary: *styl:glob:**
output sub-dictionary: *data:myFileIdx:myChannelIdx:glob:**

The register fitting procedure consists of the following steps:

- A window of length `styl:glob:decl_win` is shifted along the f0 contour with a step size of 10 ms.
- Within each window the f0 median is calculated
  - of the values below the `styl:glob:prct:bl` percentile for the baseline,
  - of the values above the `styl:glob:prct:tl` percentile for the topline, and
  - of all values for the midline.

  This gives 3 sequences of medians, one for the base-, the mid-, and the topline, respectively.
- To each of the three median sequences a linear regression line is fitted. To be able to compare contours across global segments of different lengths, time is normalized as specified by `styl:glob:nrm:mtd` to the range `styl:glob:nrm:rng`.

The motivation for using f0 medians relative to respective percentiles instead of local peaks and valleys is twofold. First, the stylization is less affected by prominent pitch accents and boundary tones. Second, errors resulting from incorrect local peak detection are circumvented. Both enhances stylization robustness as is shown in [22].

The following configuration parameters serve to customize how closely the base- and topline should follow local minima and maxima:

```
styl:glob:prct:bl
styl:glob:prct:tl
styl:glob:decl_win
```

A closer fit to local peaks and valleys is achieved by lowering `styl:glob:prct:bl` and `styl:glob:decl_win`, and by raising `styl:glob:prct:tl`. Note however, that a closer fit will result in a higher percentage of base- and topline crossings. In the resulting Python dictionary such error cases are marked as described here: 12.3.3.

From this stylization, regression line slope and intercept features are collected for the base-, mid-, and topline, as well as for the range. For the latter these features are simply derived by fitting a linear regression line through the point-wise distances between the base- and the topline. A negative slope means that base- and topline converge, whereas a positive slope signals line divergence.

8.5.3 Contour classes

Global contour classes for analyses on the categorical level are derived by slope clustering. The cluster method can be chosen by `clst:glob:mtd`. If the user expects a certain number of classes, this number can be specified by `clst:glob:kMeans:n_cluster`. Otherwise, meanShift clustering should be chosen, either as the cluster method, or in combination with kmeans for the sake of centroid initialization. For customizing the clustering settings by non-default values several parameters are provided whose values are passed on to the respective Python *sklearn* functions. These parameters are named as in *sklearn*. If needed, please consult the descriptions of the sklearn functions *KMeans*, *MeanShift*, and *estimate_bandwidth*. Figure 7 gives an example for global and local contour classes.

navigation: *do_clst_glob*
feature sets: glob
option sub-dictionary: *clst:glob:**
output sub-dictionary: *data:myFileIdx:myChannelIdx:glob:class*

8.6 F0 residual

Dependent on `styl:register` the influence of the global component is removed from the f0 contour in order to derive the f0 residual for subsequent local contour stylization. If `styl:register` is set to *bl, ml*, or *tl*, then the base, mid, or topline is subtracted. If the parameter is set to *rng*, each f0 point is normalized to the local f0 range: the corresponding points on the base- and topline are set to 0 and 1, respectively. Thus f0 values between base- and topline are within the range [0 1], f0 values below the baseline are < 0, and values above the topline are > 1. For `styl:register=none` no global component influence is removed.

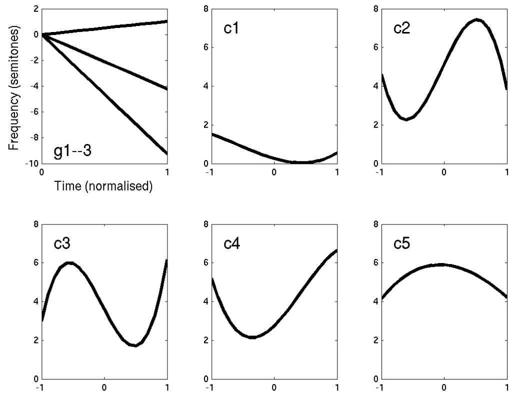


Figure 7: Global and local intonation contour classes derived by clustering.

8.7 Local segments

8.7.1 Annotation

In the annotation files local segments can be defined in 3 ways:

1. by start and end (segment tier input specified in `fsys:loc:tier_ag`)
2. by a center (event tier input specified in `fsys:loc:tier_acc`)
3. by both (segment + event tier input)

For case (2) time stamps are transformed to segments by placing a symmetric window of length `preproc:point_win` on each time stamp. In order to be able to compare contours across different segment lengths, for (1) and (2) time is normalized as specified in `styl:loc:nrm`. `styl:loc:nrm:mtd=minmax` yields a minmax time normalization to the range `styl:loc:nrm:rng`.

For (3) the time stamp within the segment is treated as the zero-center, that is, time is $[-1\ 0[$ normalized from the segment start to the center, and $[0\ 1]$ normalized from the center to the segment end.

For (3) only those segments in tier `fsys:loc:tier_ag` are considered for feature extraction to which at least one center is assigned in tier `fsys:loc:tier_acc`. `preproc:loc_align` serves for a robus treatment of multiple center assignments. Setting this option to *skip* segments with more than one center are skipped. By *left* the first center is kept, by *right* the last one.

8.7.2 Contour stylization

The f0 residual contour (see section 8.6) in each local segment is stylized by n-th order polynomials. The order is given by `styl:loc:ord`.

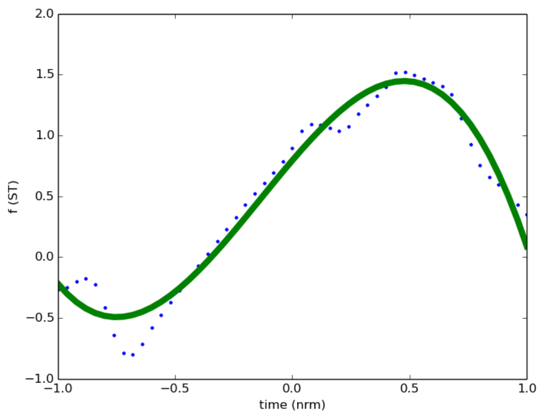


Figure 8: Local contour stylization by means of a 3rd order polynomial. Time is normalized to the range [0 1].

As can be seen in Figure 9 the polynomial coefficients are related to several aspects of local f0 shapes. Given the polynomial $\sum_{i=0}^{3} s_i \cdot t^i$, s_0 is related to the local f0 level relative to the register level. s_1 and s_3 are related to the local f0 trend (rising or falling) and – for annotation cases (2) and (3) – to peak alignment, that is negative values indicate early, and positive values late peaks. s_2 determines the peak shape (convex or concave) and its acuity: positive s_2 values indicate convex (falling-rising) shapes, negative values concave (rising-falling) shapes, and high values indicate stronger acuity.

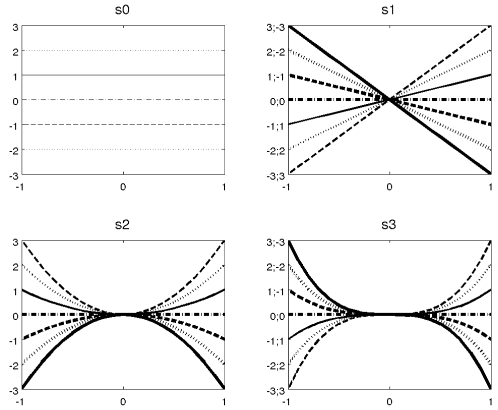


Figure 9: Influence of each coefficient of the third order polynomial $\sum_{i=0}^{3} s_i \cdot t^i$ on the local contour shape. All other coefficients set to 0. For compactness purpose on the y-axis both function and coefficient values are shown if they differ.

navigation: *do_styl_loc*
feature sets: loc
option sub-dictionary: *styl:loc:*; styl:register*
output sub-dictionary: *data:myFileIdx:myChannelIdx:loc:acc:**

8.7.3 Contour classes

Local contour classes for analyses on the categorical level are derived by polynomial coefficient clustering. The cluster method can be chosen by `clst:loc:mtd`. If the user expects a certain number of classes, this number can be specified by *clst:loc:kMeans:n_cluster*. Otherwise, meanShift clustering should be chosen, either as cluster method, or in combination with kmeans for the sake of centroid initialization. For customizing the clustering settings by non-default values several parameters are provided whose values are passed on to the respective Python *sklearn* functions. These parameters are named as in *sklearn*. If needed, please consult the descriptions of the *sklearn* functions *KMeans, MeanShift*, and *estimate_bandwidth*. Figure 7 gives an example for global and local contour classes.

navigation: *do_clst_loc*
feature sets: loc
option sub-dictionary: *clst:loc:**
output sub-dictionary: *data:myFileIdx:myChannelIdx:loc:class*

8.7.4 Standard features

Standard f0 and energy features are e.g. mean, standard deviation, median, interquartile range, and maximum. They will be calculated for the f0 contours for local contour segments. Additionally, the feature values are locally normalized within a window of length `preproc:nrm_win`. See section 8.3 for window length specifications in dependence of the annotation tier type.

navigation: *do_styl_loc_ext*
feature sets: loc
option sub-dictionary: *styl:gnl_*:**
output sub-dictionary: *data:myFileIdx:myChannelIdx:loc:gnl:**

8.7.5 Register features

As with global segments, register features can also be extracted for local features exactly the same way as introduced in section 8.5.2.

navigation: *do_styl_loc_ext*
feature sets: loc
option sub-dictionary: *styl:glob:**
output sub-dictionary: *data:myFileIdx:myChannelIdx:loc:decl:**

8.7.6 Gestalt features

Gestalt features quantify the deviation of the local contour register from the global contour register as shown in Figure 10. For this purpose the register properties of the local segment are compared with the properties of the dominating global segment in terms of root mean squared deviations and slope differences. For each register representation (base-, mid-, topline, and range regression line), the RMSD between the local and global declination line is calculated. The higher these values, the more the local contour sticks out from the global contour, which is of relevance for studies on prominence, accent group patterns [2], and prosodic headedness [23, 24].

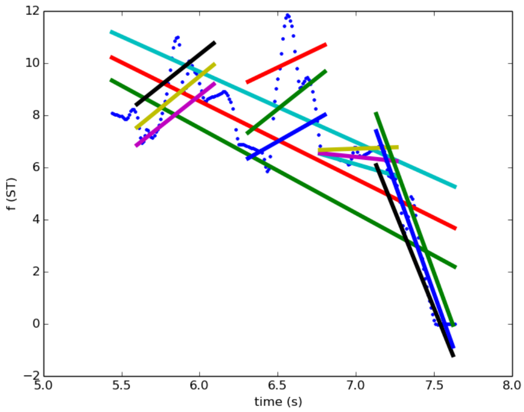


Figure 10: Gestalt stylization: Deviation of the local contour register aspects (base, mid, topline, range) from the global contour register.

The inherent Gestalt properties of the local contours are represented again in terms of polynomial coefficients. For this purpose polynomials of n-th order specified by `styl:loc:ord` are fitted to all supported kinds of f0 residuals: subtraction of base-, mid-, and topline, and range normalization. This yields 4 coefficient vectors, one for each residual.

navigation: *do_styl_loc_ext*
feature sets: loc
option sub-dictionary: *styl:loc:**
output sub-dictionary: *data:myFileIdx:myChannelIdx:loc:gst:**

8.8 Standard features

Standard features are e.g. mean, standard deviation, median, interquartile range, and maximum. They will be calculated for f0 and energy contours over the entire file and for segments in an arbitrary number of annotation tiers specified in `fsys:gnl_f0:tier` and `fsys:gnl_en:tier`, respectively. For event tiers, the segments are given by centering an analysis window of length `preproc:point_win` on the time stamps. Additionally, the feature values are locally normalized within a window of length `preproc:nrm_win`. See section 8.3 for window length specifications in dependence of the annotation tier type. Furthermore, f0 and energy quotients are calculated between the mean values derived in contour initial and final windows and in the respective remainder part of the contour. The length of this window is specified by `styl:gnl:win`. Finally, a second order polynomial is fitted through the f0 or energy contour, for which time is normalized to the range [0 1].

8.8.1 For f0 contours

navigation: *do_styl_gnl_f0*
feature sets: gnl_f0, gnl_f0_file
option sub-dictionary: *styl:gnl_f0:**
output sub-dictionary: *data:myFileIdx:myChannelIdx:gnl_f0:*, data:myFileIdx:myChannelIdx:gnl_f0_file:**

8.8.2 For energy contours

An additional standard feature for energy only is spectral balance. It is realized as the SPLH–SPL measure, i.e. the signal's sound pressure level subtracted from the level after pre-emphasis. Pre-emphasis can be carried out in the time of frequency domain `styl:gnl_en:sb:domain`. The latter is implemented as proposed by [5]. In the time domain pre-emphasis is calculated as follows: $s'[i] = s[i] - \alpha \cdot s[i-1]$. α is set by `styl:gnl_en:sb:alpha` and determines the lower frequency boundary for pre-emphasis by 6dB per octave. 0.95 roughly corresponds to 150 Hz; the smaller the value for α, the higher the lower boundary. Alternatively, α can be set directly to the lower frequency boundary F and will be internally transformed to $\alpha = e^{-2\cdot\pi\cdot F\cdot\Delta t}$. Note that pre-emphasis in the time domain usually leads to an overall lower energy so that SPLH–SPL will be negative.

In the frequency domain pre-emphasis is carried out according to [5] by adding $10 \cdot \log_{10}((1 + \frac{f^2}{200^2})/(1 + \frac{f^2}{5000^2}))$ to the logarithmic spectrum.

The spectral balance calculation can be restricted to a specified time and/or frequency window. The time window length is specified by `styl:gnl_en:sb:win` to cut out the center of that length of the segment to be analysed. It serves to reduce the influence of coarticualtion on the results. High-, low- or band-pass cutoff frequencies (`styl:gnl_en:sb:f`; filter type: `styl:gnl_en:sb:btype`) might be used to limit the analysis to a specified frequency-band (e.g. an upper cutoff frequency 5000 Hz for vowels).

navigation: *do_styl_gnl_en*
feature sets: gnl_en, gnl_en_file
option sub-dictionary: *styl:gnl_en:**
output sub-dictionary: *data:myFileIdx:myChannelIdx:gnl_en:*, data:myFileIdx:myChannelIdx:gnl_en_file:**

8.9 Boundaries

Boundaries are parameterized in terms of discontinuity features of several register representations. Details and an application for perceived prosodic boundary strength prediction can be found in [22].

Boundary features can be extracted for any number of segment or event tiers specified by `fsys:bnd:tier`. Features can be extracted for:

1. `navigate:do_styl_bnd`: each adjacent segment pair. For event tiers, segments are defined as the intervals between two time stamps. Note that this implies, that pause length is only available for segment tier input, where it is defined as the gap between the second segment's starting point and the first segment's endpoint.

2. `navigate:do_styl_win`: fixed time windows. For segment tiers, the pre- and post-boundary units are not given by the adjacent segments, but by windows of fixed length each of half of the value of `styl:bnd:win`. For event tiers the window halfs of `preproc:point_win` centered on a time stamp are considered as pre- and post-boundary units.

3. `navigate:do_styl_trend`: pre- and post-boundary units, that range from the current chunk start to the boundary, and from the boundary to the chunk end. If no chunking available, the file start and endpoint are taken.

For cases (2) and (3) holds: If `styl:bnd:cross_chunk` is set to 0, and if a chunk tier is given by `fsys:tier:chunk`, the analyses windows are limited by the start and endpoint of the current chunk.

A boundary is parameterized in terms of pause length (for segment tier input only) and pitch discontinuities. For the latter, register features (as described in section 8.5.2) are extracted three times: for the pre-boundary segment, for the post-boundary segment, and for the concatenation of both segments. Figure 11 illustrates the threefold register stylization for the pre- and post-boundary as well as for the concatenated segment. Figure 12 shows, how discontinuity for each of the register lines is expressed. Let seg_1, seg_2 be the pre- and post-boundary segments, and seg_{12} their concatenation. Then discontinuity is given by:

- the RMSD between the four register representations of seg_1 and the corresponding part of seg_{12}. The register representations are base-, mid-, topline, and range regression line.
- the RMSD between the register representations of seg_2 and the corresponding part of seg_{12}
- the RMSD between the register representations of seg_1 and seg_2 opposed to seg_{12}
- the reset d_{1_2}, i.e. the difference between the initial value of the regression line in seg_2 and the final value of the regression line in seg_1
- the onset difference of the regression lines d_o, i.e. the initial value of the seg_2 regression line subtracted from the initial value of the seg_1 line
- the difference of the regression line mean values d_m, the seg_2 mean being subtracted from the seg_1 mean. Both d_o and d_m could be used to measure downstep.
- the pairwise slope differences s_* between the 3 regression lines: for s_{1_2} the seg_2 is subtracted from the seg_1 slope. For s_{12_1} and s_{12_2} the slopes of seg_1 and seg_2 are subtracted from the seg_{12} slope.
- the correlation-based distances between the fitted lines calculated for the same combinations as the RMSD values above. Pearson r correlations are turned into distance d values ranging from 0 to 1 by $d = \frac{1-r}{2}$.
- the quotient of RMS errors between stylization input (the respective sequence of medians) and output (the fitted lines). The error of the joint stylization is divided by the error from the single pre- and post boundary fits. The quotient is reported separately for the entire, the pre-boundary, and the post-boundary segment.

- the increase of the Akaike information criterion (AIC) resulting from one joint vs two separate fits. The AIC does not only account for the fitting error but also for the number of model parameters. The lower its value, the better the model. For least squares fit comparisons the AIC can be calculated as: $2 \cdot k + n \cdot \ln \mathrm{RSS}$. k denotes the number of model parameters, n the number of stylization input values, and RSS the residual sum of squares. To each fitted line 3 parameters are assigned: intercept, slope, and Gaussian noise variation. The AIC increase is measured by subtracting the single line fit AIC from the joint fit AIC. It is reported separately for the entire, the pre-boundary, and the post-boundary segment.

All features are calculated 4 times, for the base-, mid- and toplines, as well as for the range regression lines.

All but the reset and the slope difference variables are positively related to discontinuity. The user might want to replace the reset and slope differences by their absolute values.

In the `styl:bnd` option sub-dictionary *nrm, decl_win*, and *prct* have the same purpose right as in the `styl:glob` context, see section 8.5.2. `styl:bnd:win` specifies the window length of seg_{12} for window case (2).

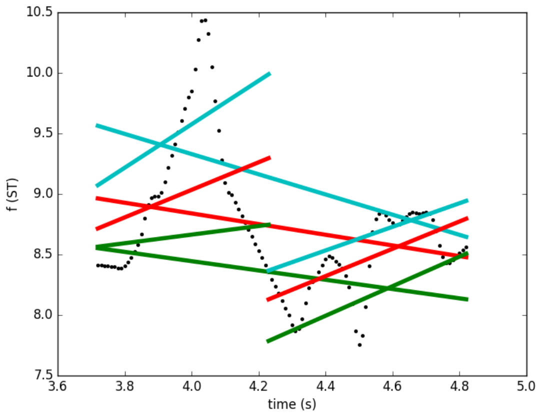


Figure 11: Prosodic boundaries: threefold base-, mid-, and topline register stylization for the pre-boundary, post-boundary, and the concatenated segment.

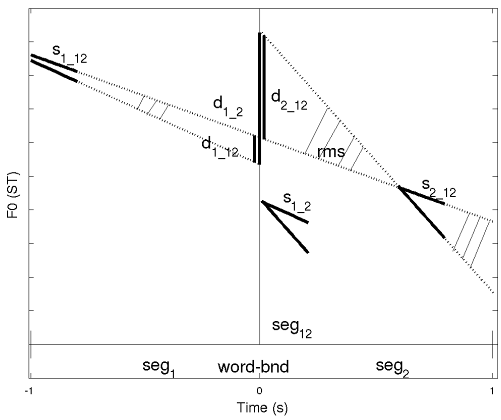


Figure 12: Boundary features describing reset and deviation from a common trend. In this case features are extracted at a word boundary *wrd-bnd*. The 3 regression lines can refer to f0 baselines, midlines, toplines, and to range. The same features are outputted for these 4 register aspects.

The boundary feature extraction can be carried out on the (preprocessed) f0 contour or on the f0 residual by setting `styl:bnd:residual` to 0 or 1, respectively. The former should be used if boundaries between global segments as intonation phrases are examined. The residual might be used if the user is interested in boundaries between e.g. accent groups within the same global segment. Note that for residuals the boundary examination across global segments might not be meaningful, since at these boundaries the residuals are derived from different register regression lines. These cases can be identified in the output by means of the *is_fin* column (see section 12.1). The residual calculation is described in section 8.6. Running boundary stylization on residuals requires a previous global contour stylization, i.e. `styl:navigate:do_styl_glob` needs to be set to 1.

The subsequent paragraphs name the configuration branches associated to the stylization cases (1)–(3), respectively.

8.9.1 Of adjacent segments

navigation: *do_styl_bnd*

feature sets: bnd
option sub-dictionary: *styl:bnd:**
output sub-dictionary: *data:myFileIdx:myChannelIdx:bnd:std:**

8.9.2 For fixed-length windows

navigation: *do_styl_bnd_win*
feature sets: bnd
option sub-dictionary: *styl:bnd:**
output sub-dictionary: *data:myFileIdx:myChannelIdx:bnd:win:**

8.9.3 For global trends

navigation: *do_styl_trend*
feature sets: bnd
option sub-dictionary: *styl:bnd:**
output sub-dictionary: *data:myFileIdx:myChannelIdx:bnd:trend:**

8.10 Rhythm

Rhythm features can be extracted for any number of segment or event tiers specified by `fsys:rhy_*:tier`, * representing *f0* and *en* for the f0 and the energy contour, respectively. Time stamps of event tiers are transformed to segments as introduced in section 8.3.

Rhythm measures consist of:

- spectral moments of a DCT analysis of the contour
- the number of peaks in the absolute-value DCT spectrum
- the frequency associated with the highest peak
- event rates within the analyzed segment
- the influence of these events on the f0 or energy contour within the analyzed segment

To extract the relative weight of the low- and high-frequency components of a contour, a discrete cosine transform (DCT) is applied on the contour as in [7]. For the absolute DCT coefficient values the first n `rhy_*:rhy:nsm` spectral moments are calculated that (up to the forth moment) give the mean, variance, skew, and kurtosis of the DCT coefficient weight distribution, repsectively.

Before applying the DCT the contour is weighted by the two parameters `rhy_*:rhy:wintyp` and `rhy_*:rhy:winparam` as introduced in section 8.2.

The events (time stamps or segments) for which rate and influence is to be calculated are read from one or more tier names in `fsys:rhy_*:tier_rate`. Thereby within each recording channel each *analysis tier* in `fsys:rhy_*:tier` is combined with each *rate tier* in `fsys:rhy_*:tier_rate`. Rate is simply measured by counting the events, that fall within the segment of analysis, and dividing it by the length of the analyzed segment. For segment tiers in `fsys:rhy_*:tier_rate` only proportions included in the segment of analysis are added to the count.

The influence s of events on the f0 or energy contour is quantified as the relative weight of the DCT coefficients around the event rate r (+/− `rhy_*:rhy:wgt:rb` Hz) within all coefficients between `rhy_*:rhy:lb` and `rhy_*:rhy:ub` Hz as follows:

$$s = \frac{\sum_{c:r-1 \leq f(c) \leq r+1 \mathrm{Hz}} |c|}{\sum_{c:lb \leq f(c) \leq ub \mathrm{Hz}} |c|}$$

The higher s the higher thus the influence of the event rate on the f0 or energy contour. Figure 13 compares a low event rate with a high impact on the energy contour with a high event rate with low impact (high vs low absolute coefficient values).

The relative weight is outputted to the feature table's columns *myRateTier_prop* (see sections 10 and 12.1). *myRateTier* refers to each entry in `fsys:rhy_*:tier_rate`. The respective analysis tiers from `fsys:rhy_*:tier` are displayed in the *tier* column. The proportion is outputted for each segment in the analysis tiers.

Additionally, the rate of rate tier events in each analysis tier segment is provided by *myRateTier_rate*. Finally, *myRateTier_mae* gives the mean absolute error between the original contour and the inverse cosine transform output that is based on the coefficients with frequencies around the event rates. The following paragraphs name the configuration branches responsible for the rhythmic analyses of the f0 and energy contour, respectively.

*myRateTier_** parameters are not calculated for analysis/rate tier combinations across recording channels. That is: Given are analysis tier *TA1* and rate tier *RT2* refering to channels 1 and 2, respectively. Then cells in the *RT2_** columns are set to *NA* in all *TA1* rows, which are identified by the *tier* column.

The number of peaks *n_peak* in the DCT spectrum is derived by counting the local amplitude maxima in this spectrum among the values greater or equal than the amplitude related to the center of gravity.

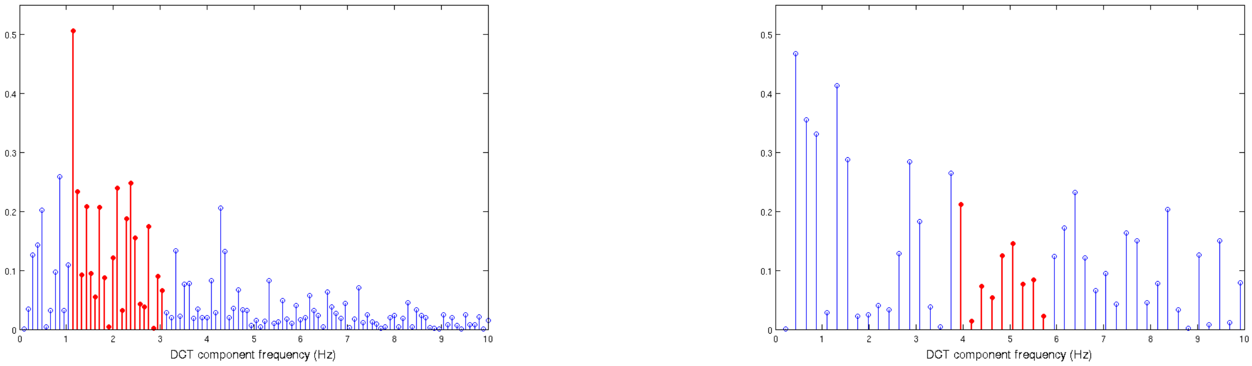


Figure 13: Influence of events on a contour in terms of the relative weight of the DCT coefficients around the event frequency.

8.10.1 Rhythmic aspects of the f0 contour

navigation: *do_styl_rhy_f0*
feature sets: rhy_f0, rhy_f0_file
option sub-dictionary: *styl:rhy_f0:**
output sub-dictionary: *data:myFileIdx:myChannelIdx:rhy_f0:*; data:myFileIdx:myChannelIdx:rhy_f0_file:**

8.10.2 Rhythmic aspects of the energy contour

The energy contour extraction in the analyzed segment is controlled by the `styl:rhy_en:sig:*` sub-dictionary the same way as explained in section 8.2.

navigation: *do_styl_rhy_en*
feature sets: rhy_en, rhy_en_file
option sub-dictionary: *styl:rhy_en:**
output sub-dictionary: *data:myFileIdx:myChannelIdx:en_f0:*; data:myFileIdx:myChannelIdx:rhy_en_file:**

9 Voice quality

Voice quality features can be extracted for any number of segment or event tiers specified by `fsys:voice:tier`. Time stamps of event tiers are transformed to segments as introduced in section 8.3. At the current state voice measures consist of:

- jitter,
- shimmer,
- 3rd order polynomial coefficients describing the changes of jitter over time
- 3rd order polynomial coefficients describing the changes of shimmer over time

Note that these values are meaningful for certain domains only, e.g. for vowel segments!

Jitter is calculated the same way as Praat's *relative local jitter*[1] as the mean absolute difference between adjacent periods divided by the overall mean period. As for Praat the following parameters can be specified in `styl:voice:jit`. *t_min* and *t_max* refer to the minimum and maximum allowed period durations, and *fac_max* to the maximally allowed quotient of adjacent periods. Periods not fulfilling these constraints are discarded from calculation.

Shimmer again is calculated the same way as Praat does for the *Shimmer (local)* parameter,[2] i.e. it is the mean absolute difference between the amplitudes of adjacent periods, divided by the average amplitude.

For both jitter and shimmer a 3rd order polynomial is fitted through the obtained sequence of distance values of adjacent periods each distance divided by the average period, resp. amplitude. Time is normalized to the interval -1 to 1. The purpose of these polynomials is to represent the changes of jitter and shimmer over time. As an example a negative 1st order coefficient for the jitter sequence indicates a decrease in jitter over time (see Figure 9 for the interpretation of the coefficients).

The configuration branches related to the *voice* feature set are:

navigation: *do_styl_voice*
feature sets: voice, voice_file
option sub-dictionary: *styl:voice:**
output sub-dictionary: *data:myFileIdx:myChannelIdx:voice:*; data:myFileIdx:myChannelIdx:voice_file:**

[1] `http://www.fon.hum.uva.nl/praat/manual/PointProcess__Get_jitter__local____.html`
[2] `http://www.fon.hum.uva.nl/praat/manual/Voice_3__Shimmer.html`

10 Feature sets

All features are subdivided into the following sets which can be extracted independently of each other. In the subsequent listing `*_file` indicates that there is an additional feature extraction on the entire file level with minor deviations from the extraction on smaller domains (e.g. missing normalization).

- **gnl_f0, gnl_f0_file:** general standard f0 features as mean, median, standard deviation, interquartile range; for any number of tiers
- **gnl_en, gnl_en_file:** general standard energy features as mean, median, standard deviation, interquartile range; for any number of tiers
- **glob:** register (level and range) features in larger domains (e.g. intonation phrases); for one tier per channel
- **loc:** shape features in smaller domains (e.g. accent groups) of f0 residuals (after removal of global f0 aspects). Gestalt features, i.e. deviation of accent groups from intonation phrases. This feature set requires the precedent extraction of the *glob* set; for one tier per channel
- **bnd, bnd_win, bnd_trend:** boundary features between adjacent segments in the same domain. For *bnd* the features are derived from the stylization of adjacent segments. In *bnd_win* the stylization is carried out in uniform time windows centered on the segment boundaries irrespective of the segment lengths. In *bnd_trend* the stylization is carried out from the beginning of a speech chunk to the boundary in question, and from this boundary to the end of the chunk; for any number of tiers
- **rhy_f0, rhy_f0_file:** DCT-based rhythm features; rates of prosodic events (e.g. syllable nuclei, pitch accents) and their influence on the f0 contour; for any number of tiers
- **rhy_en, rhy_en_file:** DCT-based rhythm features; rates of prosodic events and their influence on the energy contour; for any number of tiers
- **voice, voice_file:** voice quality features as jitter and shimmer: mean values and polynomial stylization of their changing over time

Application examples for these feature sets are

application	feature sets
pitch accent prototypes for information status and discourse segmentation [14]	glob, loc
prosodic boundary strength prediction [22]	bnd
prosodic typology [23, 24]	loc
empirical evidence for prosodic constituents (accentual phrases) [2, 24]	loc
interplay of phrasing and prominence [25]	loc, bnd, gnl_en, gnl_f0
dialog act prediction [12]	glob, loc, gnl_f0
personality trait prediction [15]	glob, loc, gnl_f0
infant-directed speech [11]	glob, loc, gnl_f0, gnl_en
entrainment [18, 17]	glob, loc
cooperative vs competitive dialogs [20]	glob, loc, rhy_en
offtalk detection [10]	glob, loc, gnl_en, gnl_f0
speech disfluencies [1]	loc
pitch accent inventory for low-resource languages [8]	loc
Lombard speech characteristics [3]	bnd
Social media analyses [19]	bnd
Hand-stroke–speech coordination [6]	rhy_en, rhy_f0

The following tables list all currently available features in alphabetical order, give short descriptions and link them to the respective feature set. In these tables *loc* and *glob* within the superpositional setting refer to local (e.g. accent groups) and global segments (e.g. intonation phrases), respectively. For boundary parameterization *pre, post, joint* refer to the pre- and post-boundary segments, and to their concatenation, respectively. For boundary features *std, win*, and *trend* refer to the underlying windowing of neighboring segments, cf. section 8.9. The number of coefficient and spectral moment variables *c** and *sm** depend on the polynomial order and spectral moment number specified by the user. For the *rhy_** feature sets *myAnalysisTier* stands for the analysis tier, and *myRateTier* for the rate tier, i.e. the rate and influence of events in *myRateTier* within segments of *myAnalysisTier* is measured, and all possible combinations of analysis and rate tiers are outputted.

name	description	feature set
bl_c0	baseline intercept	glob, loc
bl_c1	baseline slope	glob, loc
bl_d	mean baseline deviation loc-glob	loc
bl_d_fin	final baseline value diff loc-glob	loc
bl_d_init	initial baseline value diff loc-glob	loc
bl_drop	baseline f0 drop (duration · rate)	glob
bl_m	baseline mean value	glob, loc
bl_r	baseline reset	glob
bl_rate	baseline declination rate	glob, loc
bl_rms	baseline RMSD loc-glob	loc
bl_sd	baseline slope diff loc-glob	loc
bv	file-domain f0 base value (Hz)	glob, gnl_f0_file
c*	polynomial loc contour coef *	loc, gnl_f0/en(_file)
ci	channel index (starting with 0)	*(all sets)*
class	contour class	glob, loc
dur	segment duration	glob, loc, gnl_f0/en(_file), rhy_f0/en(_file)
dur_nrm	normalized duration	loc, gnl_f0/en
r_en_f0	correlation between energy and f0 contour	gnl_en
f_max	freq of coef with max ampl. in DCT spectrum	rhy_f0/en(_file)
fi	file index (starting with 0)	*(all sets)*
gi	si value of corresponding row in glob	loc
iqr	f0 interquartile range	glob, loc, gnl_f0/en(_file)
iqr_nrm	nrm'd f0 interquartile range	loc, gnl_f0/en
is_fin	item in global segment's final position?	*(all sets w/o *_file)*
is_fin_chunk	item in chunk final position?	*(all sets w/o *_file)*
is_init	item in global segment's initial position?	*(all sets w/o *_file)*
is_init_chunk	item in chunk initial position?	*(all sets w/o *_file)*
jit	jitter	voice(_file)
jit_c*	polynomial coefs for jitter time course	voice(_file)
jit_m	mean pulse period	voice(_file)
jit_m_nrm	normalized mean pulse period	voice
jit_nrm	normalized jitter	voice
jit_sd	pulse period std	voice(_file)
jit_sd_nrm	normalized pulse period std	voice(_file)
shim	shimmer	voice(_file)
shim_c*	polynomial coefs for shimmer time course	voice(_file)
shim_m	mean pulse amplitude	voice(_file)
shim_m_nrm	normalized mean pulse amplitude	voice
shim_nrm	normalized shimmer	voice
shim_sd	pulse amplitude std	voice(_file)
shim_sd_nrm	normalized pulse amplitude std	voice(_file)
lab	label	glob, bnd, gnl_f0/en, rhy_f0/en
lab_acc	ACC tier label	loc
lab_ag	AG tier label	loc
lab_next	next segment's label	bnd
m	f0, energy arit. mean	glob, loc, gnl_f0/en(_file)
m_nrm	f0, energy arit. nrm'd mean	loc, gnl_f0/en
max	f0, energy max	glob, loc, gnl_f0/en(_file)
max_nrm	f0, energy nrm'd max	loc, gnl_f0/en
maxpos	relative position of maximum	glob, loc, gnl_f0/en
med	f0, energy median	glob, loc, gnl_f0/en(_file)
med_nrm	f0, energy nrm'd median	loc, gnl_f0/en
ml_bl_cross_f0	f0 of crossing point of mid- and baseline	glob
ml_bl_cross_t	time of crossing point of mid- and baseline	glob
ml_c0	midline intercept	glob, loc
ml_c1	midline slope	glob, loc
ml_d	mean midline deviation loc-glob	loc
ml_d_fin	final midline value diff loc-glob	loc
ml_d_init	initial midline value diff loc-glob	loc
ml_drop	midline f0 drop (duration · rate)	glob
ml_m	midline mean value	glob, loc
ml_r	midline reset	glob
ml_rate	midline declination rate	glob, loc
ml_rms	midlines RMSD loc-glob	loc
ml_sd	midline slope diff loc-glob	loc
n_peak	number of peaks in absoulte DCT spectrum	rhy_f0/en(_file)

p	pause length (sec)	bnd
qb	quotient of means of init and fin part	gnl_f0/en(_file)
qf	quotient of means of final and non-fin part	gnl_f0/en(_file)
qi	quotient of means of initial and non-init	gnl_f0/en(_file)
qm	quotient of means max(init, fin) part and remainder	gnl_f0/en(_file)
res_bl_c*	baseline residual poly coef *	loc
res_ml_c*	midline residual poly coef *	loc
res_rng_c*	range line residual poly coef *	loc
res_tl_c*	topline residual poly coef *	loc
rms	overall RMSD	gnl_en
rms_nrm	nrm'd overall RMSD	gnl_en
rmsd	RMSD under stylized contour	loc
rng_c0	range line intercept	glob, loc
rng_c1	range line slope	glob, loc
rng_d	mean range line deviation loc-glob	loc
rng_d_fin	final range line value diff loc-glob	loc
rng_d_init	initial range line value diff loc-glob	loc
rng_drop	range line f0 drop (duration · rate)	glob
rng_m	range mean value	glob, loc
rng_r	range line reset	glob
rng_rate	range declination rate	glob, loc
rng_rms	range lines RMSD loc-glob	loc
rng_sd	range line slope diff loc-glob	loc
sb	spectral balance	gnl_en
sd	f0, energy standard deviation	glob, loc, gnl_f0/en(_file)
sd_nrm	nrm'd f0, energy standard deviation	loc, gnl_f0, gnl_en
si	segment index (starting with 0)	glob, loc, gnl_f0/en, rhy_f0/en
sm*	*th spectral moment of DCT	rhy_f0/en(_file)
std|trend|win_bl_aicI	baseline fitting AIC increase joint vs pre+post	bnd
std|trend|win_bl_aicI_post	baseline fitting AIC increase joint vs post	bnd
std|trend|win_bl_aicI_pre	baseline fitting AIC increase joint vs pre	bnd
std|trend|win_bl_corrD	pre/post-joint baseline corr-based distance	bnd
std|trend|win_bl_corrD_post	post-joint baseline corr-based distance	bnd
std|trend|win_bl_corrD_pre	pre-joint baseline corr-based distance	bnd
std|trend|win_bl_d_m	difference of baseline means pre–post	bnd
std|trend|win_bl_d_o	difference of baseline onsets pre–post	bnd
std|trend|win_bl_r	pre-post baseline reset	bnd
std|trend|win_bl_rms	pre/post-joint baseline RMSD	bnd
std|trend|win_bl_rms_post	post-joint baseline RMSD	bnd
std|trend|win_bl_rms_pre	pre-joint baseline RMSD	bnd
std|trend|win_bl_rmsR	baseline fitting error ratio joint vs pre+post	bnd
std|trend|win_bl_rmsR_post	baseline fitting error ratio joint vs post	bnd
std|trend|win_bl_rmsR_pre	baseline fitting error ratio joint vs pre	bnd
std|trend|win_bl_sd_post	baseline slope diff post–joint	bnd
std|trend|win_bl_sd_pre	baseline slope diff pre–joint	bnd
std|trend|win_bl_sd_prepost	baseline slope diff pre–post	bnd
std|trend|win_ml_aicI	midline fitting AIC increase joint vs pre+post	bnd
std|trend|win_ml_aicI_post	midline fitting AIC increase joint vs post	bnd
std|trend|win_ml_aicI_pre	midline fitting AIC increase joint vs pre	bnd
std|trend|win_ml_corrD	pre/post-joint midline corr-based distance	bnd
std|trend|win_ml_corrD_post	post-joint midline corr-based distance	bnd
std|trend|win_ml_corrD_pre	pre-joint midline corr-based distance	bnd
std|trend|win_ml_d_m	difference of midline means pre–post	bnd
std|trend|win_ml_d_o	difference of midline onsets pre–post	bnd
std|trend|win_ml_r	pre–post midline reset	bnd
std|trend|win_ml_rms	pre/post–joint midline RMSD	bnd
std|trend|win_ml_rms_post	post-joint midline RMSD	bnd
std|trend|win_ml_rms_pre	pre-joint midline RMSD	bnd
std|trend|win_ml_rmsR	midline fitting error ratio joint vs pre+post	bnd
std|trend|win_ml_rmsR_post	midline fitting error ratio joint vs post	bnd
std|trend|win_ml_rmsR_pre	midline fitting error ratio joint vs pre	bnd
std|trend|win_ml_sd_post	midline slope diff post–joint	bnd
std|trend|win_ml_sd_pre	midline slope diff pre–joint	bnd
std|trend|win_ml_sd_prepost	midline slope diff pre-post	bnd
std|trend|win_rng_aicI	range fitting AIC increase joint vs pre+post	bnd
std|trend|win_rng_aicI_post	range fitting AIC increase joint vs post	bnd
std|trend|win_rng_aicI_pre	range fitting AIC increase joint vs pre	bnd
std|trend|win_rng_corrD	pre/post-joint range line corr-based distance	bnd
std|trend|win_rng_corrD_post	post-joint range line corr-based distance	bnd
std|trend|win_rng_corrD_pre	pre-joint range line corr-based distance	bnd

std|trend|win_rng_d_m	difference of range line means pre–post	bnd
std|trend|win_rng_d_o	difference of range line onsets pre–post	bnd
std|trend|win_rng_r	pre-post range line reset	bnd
std|trend|win_rng_rms	std pre/post-joint range line RMSD	bnd
std|trend|win_rng_rms_post	post-joint range line RMSD	bnd
std|trend|win_rng_rms_pre	pre-joint range line RMSD	bnd
std|trend|win_rng_rmsR	range fitting error ratio joint vs pre+post	bnd
std|trend|win_rng_rmsR_post	range fitting error ratio joint vs post	bnd
std|trend|win_rng_rmsR_pre	range fitting error ratio joint vs pre	bnd
std|trend|win_rng_sd_post	range line slope diff post-joint	bnd
std|trend|win_rng_sd_pre	range line slope diff pre-joint	bnd
std|trend|win_rng_sd_prepost	range line slope diff pre-post	bnd
std|trend|win_tl_aicI	topline fitting AIC increase joint vs pre+post	bnd
std|trend|win_tl_aicI_post	topline fitting AIC increase joint vs post	bnd
std|trend|win_tl_aicI_pre	topline fitting AIC increase joint vs pre	bnd
std|trend|win_tl_corrD	pre/post-joint topline corr-based distance	bnd
std|trend|win_tl_corrD_post	post-joint topline corr-based distance	bnd
std|trend|win_tl_corrD_pre	pre-joint topline corr-based distance	bnd
std|trend|win_tl_d_m	difference of topline means pre–post	bnd
std|trend|win_tl_d_o	difference of topline onsets pre–post	bnd
std|trend|win_tl_r	std pre-post topline reset	bnd
std|trend|win_tl_rms	pre/post-joint topline RMSD	bnd
std|trend|win_tl_rms_post	post-joint topline RMSD	bnd
std|trend|win_tl_rms_pre	pre-joint topline RMSD	bnd
std|trend|win_tl_rmsR	topline fitting error ratio joint vs pre+post	bnd
std|trend|win_tl_rmsR_post	topline fitting error ratio joint vs post	bnd
std|trend|win_tl_rmsR_pre	topline fitting error ratio joint vs pre	bnd
std|trend|win_tl_sd_post	topline slope diff post-joint	bnd
std|trend|win_tl_sd_pre	topline slope diff pre-joint	bnd
std|trend|win_tl_sd_prepost	topline slope diff pre-post	bnd
stm	f0 file name stem	glob, loc
t_off	time offset (sec; bnd: of pre-boundary segment)	glob, loc, gnl_f0/en, rhy_f0/en, bnd
t_on	time onset (sec; bnd: of post-boundary segment)	glob, loc, gnl_f0/en, rhy_f0/en, bnd
tier	tier name	bnd, glob, gnl_f0/en, rhy_f0/en
tier_acc	accent point tier name	loc
tier_ag	accent group segment tier name	loc
tl_bl_cross_f0	f0 of crossing point of top- and baseline	glob
tl_bl_cross_t	time of crossing point of top- and baseline	glob
tl_ml_cross_f0	f0 of crossing point of top- and midline	glob
tl_ml_cross_t	time of crossing point of top- and midline	glob
tl_c0	topline intercept	glob
tl_c1	topline slope	glob
tl_d	mean topline deviation loc-glob	loc
tl_d_fin	final topline value diff loc-glob	loc
tl_d_init	initial topline value diff loc-glob	loc
tl_drop	topline f0 drop (duration · rate)	glob
tl_m	topline mean value	glob, loc
tl_r	initial topline reset	glob
tl_rate	topline declination rate	glob, loc
tl_rms	topline RMSD loc-glob	loc
tl_sd	topline slope diff loc-glob	loc
myRateTier_dgm	difference between rate and frequency of max amplitude coef	rhy_f0/en(_file)
myRateTier_dlm	difference between rate and frequency of nearest peak coef	rhy_f0/en(_file)
myRateTier_mae	meanAbsErr(IDCT(myRateTier),contourOfFile)	rhy_f0/en(_file)
myRateTier_prop	influence of myRateTier on DCT coefs	rhy_f0/en(_file)
myRateTier_rate	event rate of myRateTier	rhy_f0/en(_file)

11 Configurations

The configuration file format is *JSON*. Examples can be found in the *config* subfolder of the code distribution. *copasul_default_config.json* contains all default values. In the *doc* subfolder you find the file *copasul_commented_config.json.txt* where all options are commented for a quick overview. In the following detailed introduction of all configuration parameters, the levels of the JSON dictionary are separated by a colon.

For numeric and boolean parameters the *"values, default"* field contains the default value. For string parameters, the default value is indicated in bold face. If a configuration field is named as *my** the name is user defined. + indicates "one or more" configuration branches of this kind. Example: `fsys:channel:myTiername+` indicates, that the user needs to specify for all tiers in the annotation files, to which audio channel they belong. Let's assume there are two tiers *spk1* and *spk2*, the first belongs to channel 1, the second to channel, two, then `fsys:channel:spk1=1` and `fsys:channel:spk2=2`.

11.1 Sample rate

fs

description: f0 sample frequency
type: integer
values, default: 100
remarks: currently only fs=100 supported. All f0 input will be resampled to this sample rate

---

11.2 Navigation

11.2.1 Augmentation

Automatic annotation steps can be carried out independently of each other as long they don't depend on the output of preceding annotation steps, e.g. if fallback events as syllable boundaries and nuclei are required for phrase boundary and accent detection, or if parent segments are defined to be the result of preceding automatic clustering or prosodic phrasing. Figure 14 displays the possible augmentation pipelines.

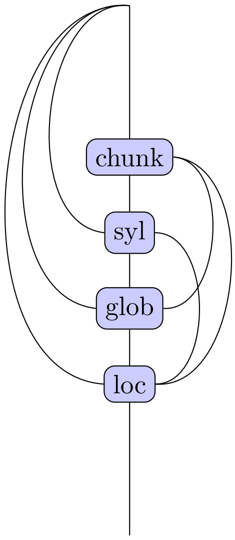


Figure 14: Automatic annotation *do_augment_** workflow

11.2.2 Feature extraction

Processing pipelines Pipelines are defined in the `navigate` configurations. Processing step dependencies are shown in Figure 15.

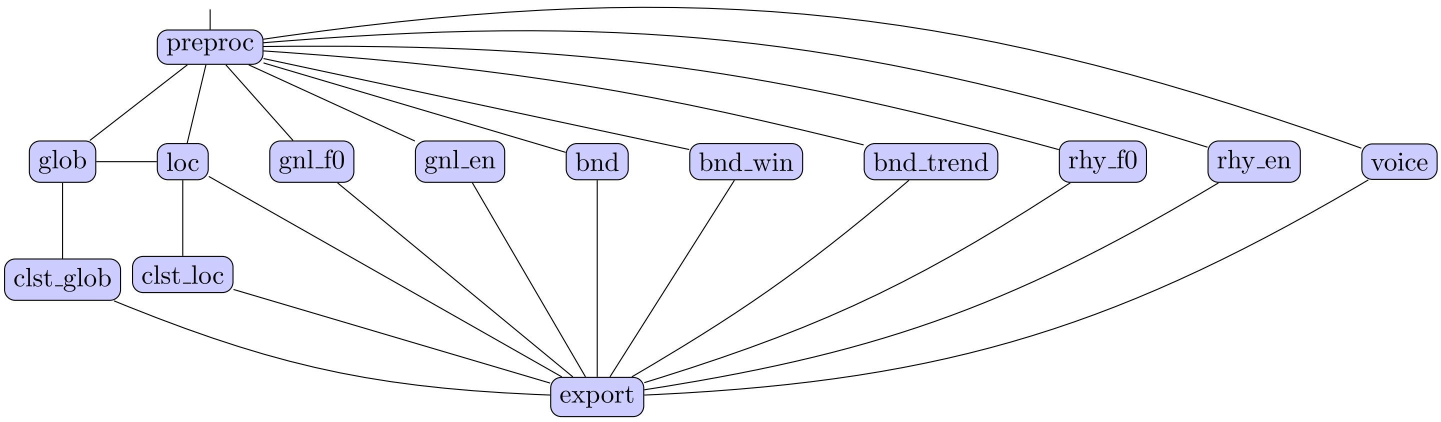


Figure 15: Stylization *do_styl* and clustering *do_clst* workflow

Processing does not always need to start from scratch. Intermediate feature extraction results are stored in Python pickle format and can be reloaded for further processing in a later session. The name of the pickle file to be loaded is given in

`fsys:export:dir` + `fsys:export:stm`

In order to continue an analysis of a previous session, the user thus needs to make sure that output directory and file name stem do not change across sessions. The content of the file can be deleted by setting `navigate:from_scratch` to 1. This and all other `navigate` configuration elements are introduced in the following:

navigate:do_augment_chunk

description: apply automatic chunking into interpausal units
type: boolean
values, default: 0
remarks: If 1, a chunk segment tier is generated for each channel and added to the annotation files.

---

navigate:do_augment_glob

description: apply unsupervised prosodic phrase extraction
type: boolean
values, default: 0
remarks: If 1, for each channel a segment tier with automatically extracted prosodic phrases is generated and added to the annotation files. If no input tier for prosodic boundary candidates is specified, this step requires preceding syllable extraction, since syllable boundaries will then be taken as candidates.

---

navigate:do_augment_loc

description: apply unsupervised pitch accent detection
type: boolean
values, default: 0
remarks: If 1, for each channel an event tier with automatically extracted pitch accent locations is generated and added to the annotation file. If no user-defined pitch accent candidates can be provided, this step requires preceding syllable nucleus extraction, which will then be taken as candidates.

---

navigate:do_augment_syl

description: apply automatic syllable nucleus and boundary detection
type: boolean
values, default: 0
remarks: If 1, for each channel two event tiers – a syllable nucleus and boundary tier – are generated and added to the annotation files.

---

navigate:do_clst_glob

description: apply local contour clustering
type: boolean
values, default: 0
remarks: cluster local contour polynomial coefficients to derive local intonation contour classes.

---

navigate:do_clst_loc

description: apply global contour clustering
type: boolean
values, default: 0
remarks: cluster global contour line slope coefficients to derive global intonation contour classes.

---

navigate:do_export

description: export the results
type: boolean
values, default: 0
remarks: generate csv feature table files, and f0 table files

---

navigate:do_plot

description: plot
type: boolean
values, default: 0
remarks: online or post-analysis plotting of stylization results. Online plotting serves to check the parameter settings before processing large data.

---

navigate:do_preproc

description: apply preprocessing
type: boolean
values, default: 0

remarks: F0 preprocessing and analysis and normalization windowing. If set to 1 at non-initial application to a data set, all information previously gathered from subsequent stylization steps is deleted.

---

navigate:do_styl_bnd_trend

description: extract boundary features
type: boolean
values, default: 0
remarks: Extract f0 discontinuity features at each segment boundary or time stamp. This time the pre- and post-boundary units range from file start to the boundary, and from the boundary to the file end. If `styl:bnd:cross_chunk` is set to 0, and if a chunk tier is given in `fsys:chunk:tier`, the analyses windows are limited by the start and endpoint of the current chunk.

---

navigate:do_styl_bnd_win

description: extract boundary features in fixed time windows
type: boolean
values, default: 0
remarks: Extract f0 discontinuity features. For segment tiers, the pre- and post-boundary units are not given by the adjacent segments as for `navigate:do_styl_bnd`, but by windows of fixed length. For event tiers the window halfs of `preproc:point_win` centered on a time stamp are considered as pre- and post-boundary units. If `styl:bnd:cross_chunk` is set to 0, and if a chunk tier is given in `fsys:chunk:tier`, the analyses windows are limited by the start and endpoint of the current chunk.

---

navigate:do_styl_bnd

description: extract boundary features
type: boolean
values, default: 0
remarks: Extract f0 discontinuity features across segments (segment tier input) or at time stamps (event tier input). Only for the former the extracted pause length is meaningful. Discontinuity is amongst others expressed in the deviation of the pre- and post-boundary part from a common declination trend. For segment tiers, this common trend is calculated over both segments. For event tiers, the inter-time stamp intervals are considered as segments.

---

navigate:do_styl_glob

description: apply global contour stylization
type: boolean
values, default: 0
remarks: Apply f0 register (level and range) stylizations within global segments as e.g. IPs.

---

navigate:do_styl_gnl_en

description: extract standard energy features
type: boolean
values, default: 0
remarks: Extract energy mean, variance and the like.

---

navigate:do_styl_gnl_f0

description: extract standard f0 features
type: boolean
values, default: 0
remarks: Extract f0 mean, variance and the like.

---

navigate:do_styl_loc_ext

description: extract extended feature set for local f0 contours
type: boolean
values, default: 0
remarks: Extract local register and Gestalt features, i.e. deviation of the local contour from the global register trend.

---

navigate:do_styl_loc

description: apply local contour stylization
type: boolean
values, default: 0
remarks: Apply polynomial f0 contour stylization in local segments as e.g. AGs.

---

navigate:do_styl_rhy_en

description: extract energy rhythm features
type: boolean
values, default: 0
remarks: apply DCT analyses on energy contour within user-defined segments and calculate the influence of events on the contour, in terms of the relative weight of DCT coefficients

---

navigate:do_styl_rhy_f0

description: extract f0 rhythm features
type: boolean
values, default: 0
remarks: apply DCT analyses on f0 contour within user-defined segments and calculate the influence of events on the contour, in terms of the relative weight of DCT coefficients

---

navigate:do_styl_voice

description: extract voice quality features
type: boolean
values, default: 0
remarks: extract jitter and shimmer

---

navigate:from_scratch

description: start from scratch
type: boolean
values, default: 0
remarks: If 1, all configurations and analyses results in the pickle file are overwritten.

---

navigate:overwrite_config

description: overwrite stored configurations
type: boolean
values, default: 0
remarks: If 1, the configuration stored in the pickle file is overwritten by the current user-defined setting. Useful, if e.g. selected analysis steps should be repeated by different preprocessing settings.

There are the following dependencies among the processing steps:

- all `do_styl*` steps require preceding `do_preproc`
- `do_styl_loc` requires preceding `do_styl_glob`
- `do_styl_bnd` requires preceding `do_styl_glob` if the boundary features are to be extracted from the f0 residuals.
- all `do_clst*` steps require a preceding `do_styl*` step of the same type (`loc` or `glob`)

If the preprocessing step `navigate:do_preproc` is repeated, all already extracted features are deleted since the updated preprocessing configuration might lead to different stylization results. Thus by repeating this step the user needs to redo all subsequent stylizations.

11.3 Directories, tiers, grouping

fsys:annot:dir

description: annotation file directory
type: string
values, default:
remarks: Can be nested. Depending on the task, audio, f0, and annotation files are obligatory or not. All obligatory directories must contain the same number of files in the same order. Optimally, same order is guaranteed using the same file name stem for corresponding audio, f0, and annotation files. However, this is not required.

---

fsys:annot:ext

description: annotation file extension
type: string
values, default: TextGrid, xml
remarks: no default

---

fsys:annot:typ

description: annotation file type
type: string
values, default: TextGrid, xml
remarks: Currently, only TextGrid and xml (see section 4.4) are supported. No default.

---

fsys:aud:dir

description: audio file directory
type: string
values, default:
remarks: Can be nested. Depending on the task, audio, f0, and annotation files are obligatory or not. All obligatory directories must contain the same number of files in the same order. Optimally, same order is guaranteed using the same file name stem for corresponding audio, f0, and annotation files. However, this is not required.

---

fsys:aud:ext

description: audio file extension
type: string
values, default:
remarks: Only files with this extension are collected from the directory.

---

fsys:aud:typ

description: audio file mimetype
type: string
values, default: wav
remarks: currently only *wav* supported

---

fsys:augment:chunk:tier_out_stm

description: tier name stem of chunking output
type: string
values, default: chunk
remarks: To the name stem the channel index will be added (**also for mono files!**). E.g. given a stereo file and `fsys:augment:chunk:tier_out_stm=CHUNK`, the two segment tiers *CHUNK_1* and *CHUNK_2* will be generated for channel 1 and 2, respectively.

---

fsys:augment:glob:tier_out_stm

description: phrasing output tier
type: string
values, default: glob
remarks: tier name stem of phrasing output. To the name stem the channel index will be added (also for mono files!). E.g. given a stereo file and `fsys:augment:glob:tier_out_stm=''IP''`, the two segment tiers *IP_1* and *IP_2* will be generated for channel 1 and 2, respectively.

---

fsys:augment:glob:tier_parent

description: parent tier for prosodic phrase extraction
type: string or list of strings
values, default: fsys:augment:chunk:tier_out_stm
remarks: Segment tiers defining the superordinate domain for overall trend measurement from which the pre- and post-candidate-boundary segment deviate. This field can contain a single string (a single tier for mono files or any `fsys:augment:*:tier_out_st` value which will be expanded by the channel index). The user can also explicitly specify multiple tier names in a list, if several channels are to be processed and the tier names cannot be derived from any `fsys:augment:*:tier_out_stm`. For segment tiers only.

---

fsys:augment:glob:tier

description: The tier in which to look for the prosodic boundary candidates.
type: fsys:augment:syl:tier_out_stm + '_bnd'
values, default: string or list of strings
remarks: This field can contain a single string (a single tier for mono files or any `fsys:augment:*:tier_out_stm` value which will be expanded by the channel index and the syllable boundary infix). The user can also explicitly specify multiple tier names in a list, if several channels are to be processed and the tier names cannot be derived from any `fsys:augment:*:tier_out_stm`. Tiers can be of segment or event type. Default is the the *_bnd*-output of `fsys:augment:syl:tier_out_stm`. Note that treating all syllable boundaries as phrase boundary candidates may result in prosodic boundaries within words. Thus a word segmentation tier is strongly recommended.

---

fsys:augment:loc:tier_acc

description: Pitch accent extraction event tier
type: string or list of strings
values, default: [ ]
remarks: Pitch accent candidate time stamps, e.g. syllable nucleus midpoints. This field can contain a single string (a single tier for mono files or `fsys:augment:syl:tier_out_stm` which will be expanded by the channel index). The user can also explicitly specify multiple tier names in a list, if several channels are to be processed and the tier names cannot be derived from `fsys:augment:syl:tier_out_stm`. For event tiers only. Field can be empty, but at least one of `fsys:augment:loc:tier_ag` and `fsys:augment:loc:tier_acc` needs to be specified. If only `fsys:augment:loc:tier_ag`: analysis within segment; if only `fsys:augment:loc:tier_acc`: analysis within symmetric window of length `preproc:point_win` centered on the time stamp; if both: analysis within *ag* segment, time normalization so that 0 position is at *acc* time stamp within *ag*.

---

fsys:augment:loc:tier_ag

description: pitch accent extraction segment tier
type: string or list of strings
values, default: [ ]
remarks: Tier with segments that are potential accent groups segment domain. This field can contain a single string for mono files or a list of strings for more channels. Tiers can be of segment type only. Field can be empty, but at least one of `fsys:augment:loc:tier_ag` and `fsys:augment:loc:tier_acc` needs to be specified. If only `fsys:augment:loc:tier_ag`: analysis within segment; if only `fsys:augment:loc:tier_acc`: analysis within symmetric window of length `preproc:point_win` centered on the time stamp; if both: analysis within *ag* segment, time normalization so that 0 position is at *acc* time stamp within *ag*.

fsys:augment:loc:tier_out_stm

description: accent output tier name stem
type: string
values, default: acc
remarks: To the name stem the channel index will be added (also for mono files!). E.g. given a stereo file and `fsys:augment:loc:tier_out_stm=''ACC''`, the two event tiers *ACC_1* and *ACC_2* will be generated for channel 1 and 2, respectively.

---

fsys:augment:loc:tier_parent

description: name of parent tier for pitch accent candidates
type: string or list of strings
values, default: [ ]
remarks: This parent tier contains segments of a superordinate domain with respect to which the deviation of the accent candidate segments or time stamps is calculated. This might be global segments or chunks. Fallback is file-level. Must be segment tiers. This field can contain a single string (a single tier for mono files or any `fsys:augment:*:tier_out_stm` value which will be expanded by the channel index). The user can also explicitly specify multiple tier names in a list, if several channels are to be processed and the tier names cannot be derived from any `fsys:augment:*:tier_out_stm`. Tiers can be segment tiers only.

---

fsys:augment:syl:tier_out_stm

description: tier name stem of syllable nucleus and boundary output
type: string
values, default: syl
remarks: To the name stem the channel index will be added (also for mono files!). Syllable boundary tiers are further marked by the infix *bnd*. E.g. given a stereo file and `fsys:augment:syl:tier_out_stm=''SYL''`, the four event tiers *SYL_1, SYL_bnd_1* and *SYL_2, SYL_bnd_2* will be generated for syllable nuclei and boundaries and for channel 1 and 2, respectively.

---

fsys:augment:syl:tier_parent

description: parent tier for syllable nucleus extraction
type: string or list of strings
values, default: chunk
remarks: The parent tier defines the boundaries over which the reference window for relative energy calculation must not cross. Fallback is file level. This field can contain a single string (a single tier for mono files or any `fsys:augment:*:tier_out_stm` value which will be expanded by the channel index). The user can also explicitly specify multiple tier names in a list, if several channels are to be processed and the tier names cannot be derived from any `fsys:augment:*:tier_out_stm`.

---

fsys:bnd:tier

description: boundary tier names
type: string or list of strings
values, default: [ ]
remarks: each channel can contain several tiers to be analyzed. Segment or event tiers. For segment tiers the boundary between adjacent segments is parameterized, and for point tiers, the boundary at time stamps.

---

fsys:channel:myTiername+

description: channel index for each relevant tier name in the annotation file
type: int
values, default: myChannelIdx
remarks: For augmentation output tiers this configuration branch is generated automatically.

---

fsys:chunk:tier

description: chunk tier names
type: string or list of strings
values, default: [ ]
remarks: one item for each channel. In case of multiple channels and single string, this string (e.g. "chunk") is expanded to "chunk_1", "chunk_2" ... for each available channel index. If chunk tiers specified, their segments' boundaries are not crossed by analysis and normalization windows for most feature sets. For the *bnd_trend* feature set pre- and post-boundary segments are limited by the start and endpoint of the superordinate chunk if `styl:bnd:cross_chunk` set to 1.

---

fsys:export:csv

description: output csv tables
type: boolean
values, default: 1
remarks: If 1, for each extracted feature set a csv file is outputted together with a code template file to read the table in R. The file names are concatenated by `fsys:export:stm` and the name of the feature set.

---

fsys:export:dir

description: output directory
type: string
values, default:
remarks: Directory in which all csv tables, the log file, and the pickle file are stored.

---

fsys:export:f0_preproc

description: output preprocessed f0 contours
type: boolean
values, default: 0
remarks: If 1, preprocessed f0 values are outputted for each input f0 file. The output format is as specified in section 4.2. The output is stored in the subdirectory *f0_preproc* below the directory `fsys:export:dir`.

---

fsys:export:f0_residual

description: output residual f0 contours
type: boolean
values, default: 0
remarks: If 1, residual f0 contours after register removal are outputted for each input f0 file. The output format is as specified in section 4.2. The output is stored in the subdirectory *f0_residual* below the directory `fsys:export:dir`.

---

fsys:export:f0_resyn

description: output resynthesized f0 contours
type: boolean
values, default: 0
remarks: If 1, the resynthesized f0 contours as a superposition of global and local contour shapes are outputted for each input f0 file. The output format is as specified in section 4.2. The output is stored in the subdirectory *f0_resyn* below the directory `fsys:export:dir`.

---

fsys:export:fullpath

description: whether or not to write the full path to the csv tables into the R code template files
type: boolean
values, default: 0
remarks: If 1, the full path to the csv tables is written into the R code. 0 is recommended in case the data is shared and further processed at different locations.

---

fsys:export:sep

description: table column separator
type: string
values, default: ,
remarks: column separator for csv output tables.

---

fsys:export:stm

description: output file name stem
type: string
values, default: copasul
remarks: Same file name stem for all csv files, the log file, and the pickle file.

---

fsys:export:summary

description: output file/channel summary statistics
type: boolean
values, default: 0
remarks: If 1, mean and variance values are calculated for all continuous-valued features outputted in the feature-set related csv files per file and analysis tier. For categorical features unigram entropies are calculated. A `fsys:export:stm`.summary.csv file is outputted together with an R code template file to read the table in R.

---

fsys:f0:dir

description: f0 file directory
type: string
values, default:
remarks: Can be nested. Depending on the task, audio, f0, and annotation files are obligatory or not. All obligatory directories must contain the same number of files in the same order. Optimally, same order is guaranteed using the same file name stem for corresponding audio, f0, and annotation files. However, this is not required.

---

fsys:f0:ext

description: F0 file extension
type: string
values, default:
remarks: only files with this extension are collected from the directory

---

fsys:f0:typ

description:
type: string
values, default: tab
remarks: Currently only *tab* supported.

---

fsys:glob:tier

description: global segment tier names
type: string or list of strings
values, default: [ ]
remarks: analysis tiers for global segment, only one per each channel supported, so that global and local segments can be assigned to each other. If taken over from `fsys:augment:*:tier_out_stm`, the names must be extended by the corresponding channel index, e.g. IP_1 etc, see `fsys:augment:*:tier_out_stm`. Segment or event tier. Events are considered to be right boundaries of segments and are expanded accordingly to segments.

---

fsys:gnl_en:tier

description: Tiers for standard energy variable extraction.
type: string or list of strings
values, default: [ ]
remarks: More than one tier per channel supported. Segment or event tiers. Events are expanded to segments by `preproc:point_win`.

---

fsys:gnl_f0:tier

description: Tiers for standard f0 variable extraction
type: string or list of strings
values, default: [ ]
remarks: More than one tier per channel supported. Segment or event tiers. Events are expanded to segments by `preproc:point_win`.

---

fsys:grp:lab

description: grouping labels with values derived from file names
type: list of strings
values, default: [ ]
remarks: Labels of file-name derived grouping. Non-relevant file parts are indicated by empty strings ". E.g. given the f0 filename stem a_b_2. Let's say, "a" represents the speaker ID, "b" is not relevant for the current analysis, and "2" represents the stimulus ID. Then set `fsys:grp:src=f0`, `fsys:grp:sep=_`, and `fsys:grp:lab=['spk','','stim']`. The output csv tables then contain two additional grouping columns *grp_spk* and *grp_stim* with values derived from the file names (in this case "a" and "2"). Note that all grouping values are treated as strings.

---

fsys:grp:sep

description: file name split pattern
type: string
values, default:
remarks: How to split the file name to access the grouping values. The string is interpreted as a regular expression. Thus predefined characters as the dot need to be protected! Thus if file parts are separated by a dot set this option to "\\.". If fileparts are separated by more than one symbol, e.g. dot and underscore, use "(_|\.)".

---

fsys:grp:src

description: grouping source
type: string
values, default: **f0**, annot, aud
remarks: from which file type to derive the file name based grouping

---

fsys:label:chunk

description: chunk label
type: string
values, default: x
remarks: will be used by automatic chunking

---

fsys:label:pau

description: pause label
type: string
values, default: <P>
remarks: in annotation files, segments labeled by this symbol are treated as pauses and are not analyzed. For boundary feature extraction these segments define the pause length feature between the preceding and following segment. Note, that this symbol as a pause identifier must be uniform over all analyzed tiers. In Praat TextGrids also not labeled segments are considered as pauses.

---

fsys:label:syl

description: syllable label
type: string
values, default: x
remarks: will be used by automatic syllable extraction

---

fsys:loc:tier_acc

description: local event tier names
type: string or list of strings
values, default: [ ]
remarks: tier (one for each channel) defining pitch accent time stamps. Event tiers only. Field can be empty, but at least one of `fsys:loc:tier_ag` and `fsys:loc:tier_acc` needs to be specified. If only `fsys:loc:tier_ag`: analysis within segment; if only `fsys:loc:tier_acc`: analysis within symmetric window of length `preproc:point_win` centered on the time stamp; if both: analysis within *ag* segment, time normalization so that 0 position is at *acc* time stamp within *ag*.

---

fsys:loc:tier_ag

description: local segment tier names
type: string or list of strings
values, default: [ ]
remarks: tier (one for each channel) defining accent group-like units. Segment tiers only. Field can be empty, but at least one of `fsys:loc:tier_ag` and `fsys:loc:tier_acc` needs to be specified. If only `fsys:loc:tier_ag`: analysis within segment; if only `fsys:loc:tier_acc`: analysis within symmetric window of length `preproc:point_win` centered on the time stamp; if both: analysis within *ag* segment, time normalization so that 0 position is at *acc* time stamp within *ag*.

---

fsys:pho:tier

description: name of tier with phonetic segments
type: string or list of strings
values, default: []
remarks: one tier per channel. Used for feature extraction in prosodic boundary and accent localization.

---

fsys:pho:vow

description: vowel pattern
type: string
values, default: [AEIOUYaeiouy29{]
remarks: to identify vowel segments in `fsys:pho:tier`. Is interpreted as a regular expression.

---

fsys:pic:dir

description: directory for plotting output
type: string
values, default:
remarks: directory for the png files generated by plotting.

---

fsys:pic:stm

description: file name stem of the plot files
type: string
values, default: copasul
remarks:

---

fsys:pulse:dir

description: Pulse file directory
type: string
values, default:
remarks: Can be nested. Only for extracting voice quality features pulse files are obligatory. All obligatory directories must contain the same number of files in the same order. Optimally, same order is guaranteed using the same file name stem for corresponding audio, f0, pulse, and annotation files. However, this is not required.

---

fsys:pulse:ext

description: Pulse file extension
type: string
values, default:
remarks: only files with this extension are collected from the directory

---

fsys:pulse:typ

description:
type: string
values, default: tab
remarks: Currently only *tab* supported.

---

fsys:rhy_en:tier

description: Tiers for energy rhythm extraction
type: string or list of strings
values, default: [ ]
remarks: More than one tier per channel supported. Segment or event tiers. Events are expanded to segments by `preproc:point_win`.

---

fsys:rhy_en:tier_rate

description: Tiers containing units whose rate is to be calculated within each segment of the `fsys:rhy_f0:tier` tiers
type: string or list of strings
values, default: [ ]
remarks: More than one tier per channel supported. Segment or event tiers.

---

fsys:rhy_f0:tier

description: Tiers for f0 rhythm extraction
type: string or list of strings
values, default: [ ]
remarks: More than one tier per channel supported. Segment or event tiers. Events are expanded to segments by `preproc:point_win`.

---

fsys:rhy_f0:tier_rate

description: Tiers containing units whose rate is to be calculated within each segment of the `fsys:rhy_f0:tier` tiers
type: string or list of strings
values, default: [ ]
remarks: More than one tier per channel supported. Segment or event tiers.

---

fsys:voice:tier

description: Tiers for voice quality extraction
type: string or list of strings
values, default: [ ]
remarks: More than one tier per channel supported. Segment or event tiers. Events are expanded to segments by `preproc:point_win`.

11.4 F0 preprocessing, windowing

preproc:base_prct

description: Percentile below which base value for semitone transform is calculated
type: float]0 100[
values, default: 5
remarks: Base value for semitone transform is defined as median of the values below the specified percentile. If set to 0, the base value will be set to 1, i.e. the semitone transform is carried out without normalization.

---

preproc:base_prct_grp:myChannelIndex

description: Grouping variable for which for each of its levels a base value for f0 semitone transform is calculated
type: string
values, default: ' '
remarks: Indicates for each channel index, which grouping variable is relevant. The grouping variable must be extractable from the file name as specified in `fsys:grp`. E.g. `preproc:base_prct_grp:1`=*spkId* requires a *spkId* element in the list of `fsys:grp:lab`. Channel indices must be written in quotation marks as strings.

---

preproc:loc_align

description: Robust treatment of local segments to which more than one center is assigned in the annotation.
type: string
values, default: **skip**, left, right
remarks: *skip* – such local segments are skipped; *left* – the first center is kept; *right* – the last center is kept.

---

preproc:loc_sync

description: Extract *gnl_** and *rhy_** features only at locations where *loc* features can be obtained.
type: boolean
values, default: 0
remarks: Due to the strict hierarchy principle and to window length constraints it is not always possible to extract loc features at any location where gnl and rhy features can be obtained. If the user is interested only in locations where all these feature sets are available, so that the corresponding feature matrices can be concatenated, this option should be set to 1.

---

preproc:nrm_win

description: normalization window length (in sec)
type: float

values, default: 0.6
remarks: length of the normalization window. For feature sets *gnl_** all mean, max, std values derived in the analysis window are normalized within longer time window which length is defined by this parameter. If segments to be analyzed are longer than the normalization window, this window is set equal to the analyzed segment. `nrm_win` can also be individually set for each of the feature sets *loc, gnl_f0, gnl_en, rhy_f0, rhy_en* (see section 10) by specifying `preproc:myFeatureSet:nrm_win`.

---

preproc:out:f

description: outlier definition factor
type: float
values, default: 2
remarks: identifies non-zero f0 values as outliers, that deviate more than this factor times dispersion from the mean value. If `preproc:out:m=mean`, the mean value is given by the arithmetic mean and the dispersion by the standard deviation. If `preproc:out:m=median`, the mean value is given by the median and the dispersion by the inter quartile range. If `preproc:out:m=fence`, instead of the mean value the first and third quartiles are used as references and dispersion is given by the interquartile range (*Tukey's fences*).

---

preproc:out:m

description: reference value definition for outlier identification
type: string
values, default: **mean**, median, fence
remarks: Specifies definition of mean/fence and dispersion, see `preproc:out:f` for details.

---

preproc:point_win

description: window length to transform events to segments (in sec)
type: float
values, default: 0.3
remarks: The extraction of the feature sets *glb_*, rhy_*, glob, loc* is based on segments. For event tier input, segments are obtained by centering a window of this length on the time stamps. `point_win` can also be individually set for each of the feature sets *loc, gnl_f0, gnl_en, rhy_f0, rhy_en* (see section 10) by specifying `preproc:myFeatureSet:point_win`.

---

preproc:smooth:mtd

description: F0 smoothing method
type: string
values, default: **sgolay**, med
remarks: Savitzky-Golay or median filtering of f0 contour. Median yields stronger smoothing, Savitzky-Golay performs better in keeping local minima and maxima at their place.

---

preproc:smooth:ord

description: polynomial order of smoothing method
type: integer
values, default: 3
remarks: relevant for `preproc:smooth:mtd=sgolay` only.

---

preproc:smooth:win

description: smoothing window length (in f0 sample indices)
type: int
values, default: 7
remarks: The longer the smoothing window, the more smooth the f0 contours.

---

preproc:st

description: Hertz to semitone conversion
type: boolean
values, default: 0, **1**
remarks: If 1, transformed to semitones.

11.5 Augmentation: Chunking

augment:chunk:e_rel

description: proportion of reference energy below which a pause is assumed
type: float
values, default: 0.0767
remarks: a pause is indicated, if the energy in the analysis window is below this factor times the energy in the longer reference window.

---

augment:chunk:fbnd

description: assume pause at beginning and end of file
type: boolean

values, default: 1
remarks: If set to 1, forced pause detection at file start and end. These pauses are subtracted from `augment:chunk:n` if set.

---

augment:chunk:flt:btype

description: filter type
type: string
values, default: **low**, high, band
remarks: Butterworth filter type to filter the signal for pause detection. Recommended: *low*.

---

augment:chunk:flt:f

description: filter cutoff frequencies (in Hz)
type: float or list of floats
values, default: **8000**
remarks: For `augment:chunk:flt:btype=low, high` a single cut-off frequency is expected; for *band* a 2-element list of lower and upper cutoff frequency.

---

augment:chunk:flt:ord

description: filter order
type: int
values, default: 5
remarks: Butterworth filter order.

---

augment:chunk:l_ref

description: reference window length for pause detection (in sec)
type: int
values, default: 5
remarks: Energy in analysis window of length `augment:chunk:l` is compared against the energy within the reference window. Same midpoint as analysis window.

---

augment:chunk:l

description: length of the analysis window (in sec)
type: float
values, default: 0.1524
remarks: analysis window for which is to be decided, whether or not it is (part of) a pause.

---

augment:chunk:margin

description: silence margin at chunk start and end (in sec)
type: float
values, default: 0
remarks: chunks are extended by this amount on both sides.

---

augment:chunk:min_chunk_l

description: minimum chunk length (in sec)
type: boolean
values, default: 0.3
remarks: shorter chunks are merged

---

augment:chunk:min_pau_l

description: minimum pause length (in sec)
type: boolean
values, default: 0.3
remarks: shorter pauses are ignored.

---

augment:chunk:n

description: pre-specified number of pauses [sic!]
type: boolean
values, default: -1
remarks: In this implementation chunks are defined as interpausal units and thus depend on pause detection. If set to -1, no pre-specified pause number.

11.6 Augmentation: Syllable nucleus detection

augment:syl:d_min

description: minimum distance between subsequent syllable nuclei (in sec)
type: float
values, default: 0.05
remarks: If 2 detected nuclei are closer than this distance the weaker candidate is discarded.

---

augment:syl:e_min

description: minimum energy factor relative to entire file
type: boolean
values, default: 0.16
remarks: For a syllable nucleus the RMS energy in the analysis window must be above this factor times the energy in the entire file.

---

augment:syl:e_rel

description: minimum energy factor relative to reference window
type: boolean
values, default: 1.07
remarks: For a syllable nucleus the RMS energy in the analysis window must be above this factor times the energy in the reference window.

---

augment:syl:flt:btype

description: filter type
type: string
values, default: low, high, **band**
remarks: Butterworth filter type to filter the signal for syllable nucleus detection. Recommended: *band*.

---

augment:syl:flt:f

description: filter cutoff frequencies (in Hz)
type: float or list of floats
values, default: [200 4000]
remarks: For `augment:syl:flt:btype=low, high` a single cut-off frequency is expected; for *band* a 2-element list of lower and upper cutoff frequency.

---

augment:syl:flt:ord

description: filter order
type: int
values, default: 5
remarks: Butterworth filter order.

---

augment:syl:l_ref

description: reference window length for syllable detection (in sec)
type: boolean
values, default: 0.15
remarks: Energy in analysis window with same midpoint is compared against the energy within the reference window.

---

augment:syl:l

description: analysis window length (in sec)
type: boolean
values, default: 0.08
remarks: length of window within energy is calculated. Same midpoint as reference window.

11.7 Augmentation: Prosodic boundary detection

augment:glob:cntr_mtd

description: how to define cluster centroids
type: string
values, default: **seed_prct**, seed_kmeans, split
remarks: *seed_**: initialize clustering by bootstrapped seed centroids. *seed_prct:* single-pass clustering of the boundary candidates by their distance to these centroids. Distance values to the no-boundary seed above a specified percentile `augment:glob:prct` indicate boundaries. *seed_kmeans:* kmeans clustering initialized by the seed centroids (gives a more balanced amount of boundary/no boundary cases than *seed_prct*). *split*: centroids are derived by splitting each column in the feature matrix at the percentile `augment:glob:prct`; the boundary centroid is defined by the median of the values above the splitpoint, the no-boundary centroid by the median of the values below; items are then assigned to the nearest centroid in a single pass. Depending on `augment:glob:unit` clustering is either carried out separately within each file and each channel, or over the entire dataset. Fallback: if cluster centroids cannot be bootstrapped, this parameter's value is changed to *split*.

---

augment:glob:heuristics

description: heuristic macro settings
type: string
values, default: ORT
remarks: Only *ORT* supported. *ORT* assumes a word segmentation tier for prosodic boundary prediction and rejects boundaries after too short and thus probably function words ($< 0.1s$). Not necessarily meaningful for any language.

---

augment:glob:measure

description: feature values, or deltas
type: string
values, default: **abs**, delta, abs+delta
remarks: Which values v to put in the feature matrix (i=time index): *abs*: feature values $v[i]$; *delta*: feature deltas $v[i] - v[i-1]$; *abs+delta*: both

---

augment:glob:min_l

description: minimum inter-boundary distance (in sec)
type: float
values, default: 0.5
remarks: If 2 detected boundaries are closer than this value, only the stronger one will be kept. This distance is also used in bootstrapping boundary and no-boundary centroids as described in section 7.3.

---

augment:glob:prct

description: percentile of cluster splitpoint
type: float]0 100[
values, default: 95
remarks: Splitpoint definition for clustering in terms of a percentile value. The higher the fewer boundaries will be detected. For `augment:glob:cntr_mtd=split` the percentile refers to the feature values, for `augment:glob:cntr_mtd=seed_prct`, it refers to the distance to the no-boundary seed centroid.

---

augment:glob:unit

description: derive centroids separately for each file or over entire data set
type: string
values, default: **batch**, file
remarks: batch mode recommended for corpora containing lots of short recordings, within which centroids cannot reliably be extracted.

---

augment:glob:wgt:myBndFeatset+:myRegister+:myFeat+

description: user defined feature weights
type: float
values, default: 1
remarks: create one config branch for each selected boundary feature and assign a weight. Only boundary features supported. The weight becomes a dummy in case of `augment:glob:wgt_mtd` is not *user*. However, the branches must be specified in order to mark which features to be used for boundary prediction. *myBndFeatset* $\in$ { *std, win, trend*}, *myRegister* $\in$ {*bl, ml, tl, rng*}, *myFeat* $\in$ {*r, rms, rms_pre, ...*}. The branches must correspond to branches in the sub-dictionary `copa:data:myFileIdx:myChannelIdx:bnd:myTierNameIndex:myBoundaryIndex` (see section 12.3).
E.g. `copa:data:myFileIdx:myChannelIdx:bnd:myTierNameIndex:myBoundaryIndex:win:bl:r` is addressed by `augment:glob:wgt:win:bl:r`.

---

augment:glob:wgt:pho

description: use/weight normalized vowel length as feature
type: float
values, default: 1
remarks: only compliant with `augment:glob:unit=batch`.

---

augment:glob:wgt_mtd

description: feature weighting method
type: string
values, default: **silhouette**, correlation, user
remarks: For *silhouette* an initial clustering is carried out, and for each feature its weight is then defined by its cluster-separating power. For *correlation* weights are defined for each feature by its correlation to the feature vector medians. For *user*, the weights specified in the `augment:glob:wgt:myBndFeatset+:myRegister+:myFeat+` branches are taken.

11.8 Augmentation: Pitch accent detection

augment:loc:acc_select

description: which syllable within a segment to select
type: string
values, default: **max**, left, right
remarks: Choose the accent position among all time stamps in `augment:loc:tier_acc` that are in the same segment of `fsys:augment:loc:tier_ag`. *max:* the most prominent one; *left, right*: accent first/last syllable, which might be useful if `fsys:augment:loc:tier_ag` contains word segments, and word stress is fixed.

---

augment:loc:ag_select

description: which segments to select for accentuation
type: string
values, default: **max**, all
remarks: *all:* assign an accent to each segment in `fsys:augment:loc:tier_ag`; *max:* assign accents to the most prominent segments only.

---

augment:loc:cntr_mtd

description: how to define cluster centroids
type: string
values, default: **seed_prct**, seed_kmeans, split
remarks: *seed_**: initialize clustering by bootstrapped seed centroids. *seed_prct:* single-pass clustering of the accent candidates by their distance to these centroids. Distance values to the no-accent seed above a specified percentile `augment:loc:prct` indicate accents. *seed_kmeans:* kmeans clustering initialized by the seed centroids (gives a more balanced amount of accent/no-accent cases than *seed_prct*). *split*: centroids are derived by splitting each column in the feature matrix at the percentile `augment:glob:prct`; the accent centroid is defined by the median of the values above the splitpoint, the no-accent centroid by the median of the values below; items are then assigned to the nearest centroid in a single pass. Depending on `augment:loc:unit` clustering is carried out either separately within each file and each channel, or over the entire dataset. Fallback: if cluster centroids cannot be bootstrapped, this parameter's value is changed to *split*.

---

augment:loc:heuristics

description: heuristic macro settings
type: string
values, default: ORT
remarks: only *ORT* supported. *ORT* assumes a word segmentation tier for accent extraction. Short words (see `augment:loc:max_l_na`) will be treated as non-accent seeds, long words (see `augment:loc:min_l_a`) as accent seeds.

---

augment:loc:max_l_na

description: maximum length of definitely non-accented words (in sec)
type: float
values, default: 0.1
remarks: from words below that length the *non-accented* seed centroid is derived

---

augment:loc:measure

description: feature values, or deltas
type: string
values, default: **abs**, delta, abs+delta
remarks: Which values v to put in the feature matrix (i=time index): *abs*: feature values $v[i]$; *delta*: feature deltas $v[i] - v[i-1]$; *abs+delta*: both

---

augment:loc:min_l_a

description: minimum length of definitely accented words (in sec)
type: float
values, default: 0.6
remarks: from words above that length the *accented* seed centroid is derived

---

augment:loc:min_l

description: minimum inter-accent distance (in sec)
type: float
values, default: 0.2
remarks: If 2 detected accents are closer than this value, only the more prominent one will be kept.

---

augment:loc:prct

description: percentile of cluster splitpoint
type: float]0 100[
values, default: 90

remarks: Splitpoint definition for clustering in terms of a percentile value. The higher the fewer accents will be detected. For `augment:loc:cntr_mtd=split` the percentile refers to the feature values, for `augment:loc:cntr_mtd=seed_prct`, it refers to the distance to the no-accent seed centroid.

---

augment:loc:unit

description: derive centroids separately for each file or over entire data set
type: string
values, default: **batch**, file
remarks: batch mode recommended for corpora containing lots of short recordings, within which centroids cannot reliably be extracted.

---

augment:loc:wgt:myFeatset+:...

description: user defined feature weights
type: float
values, default: 1
remarks: create one config branch for each selected prominence feature and assign a weight. *myFeatset* $\in$ {*acc, gst, gnl_f0, gnl_en*}. The weight becomes a dummy in case of `augment:loc:wgt_mtd` is not *user*. However, the branches must be specified in order to mark which features to be used for accent prediction. The branches must correspond to branches in the sub-dictionary `copa:data:myFileIdx:myChannelIdx:loc` (see section 12.3). E.g. `copa:data:myFileIdx:myChannelIdx:loc:gst:bl:rms` is addressed by `augment:loc:wgt:gst:bl:rms`. If the value at this branch is a list (e.g. the polynomial coefficients in `...augment:loc:wgt:acc:c`) the weight can either be a scalar to weight all list elements equally or a list of same length as the value list, to individually weight each element. (Only) for polynomial coefficients absolute values are taken.

---

augment:loc:wgt:pho

description: use/weight normalized vowel length as feature
type: float
values, default: 1
remarks: only compliant with `augment:loc:unit=batch`.

---

augment:loc:wgt_mtd

description: feature weighting method
type: string
values, default: **silhouette**, correlation, user
remarks: For *silhouette* an initial clustering is carried out, and for each feature its weight is then defined by its cluster-separating power. For *correlation* weights are defined for each feature by its correlation to the feature vector medians. For *user*, the weights specified in the `augment:loc:wgt:...+` branches are taken.

11.9 Stylization: Global contours

styl:glob:decl_win

description: window length for median calculation (in sec)
type: float
values, default: 0.1
remarks: Within each window a median each for the base-, mid-, and topline is derived.

---

styl:glob:nrm:mtd

description: time normalization method
type: string
values, default: **minmax**
remarks: for time normalization in global segment. Currently only *minmax* supported.

---

styl:glob:nrm:rng

description: normalized time range
type: list of floats
values, default: $[0, 1]$
remarks: normalized time of segment start and endpoint

---

styl:glob:prct:bl

description: percentile below which the baseline input medians are calculated
type: float]0 100[
values, default: 10
remarks: A sequence of lower range medians is calculated along the f0 contour. The baseline is given by linear regression through this sequence.

---

styl:glob:prct:tl

description: percentile, above which the topline input medians are calculated
type: float]0 100[
values, default: 90
remarks: A sequence of upper range medians is calculated along the f0 contour. The topline is given by linear regression through this sequence.

11.10 Stylization: Local contours

styl:loc:nrm:mtd

description: time normalization method
type: string
values, default: **minmax**
remarks: for time normalization in the local segment. Currently only *minmax* supported.

---

styl:loc:nrm:rng

description: normalized time range
type: list of floats
values, default: $[-1, 1]$
remarks: normalized time of segment start and endpoint. $[-1, 1]$ is recommended to center the polynomial around 0.

---

styl:loc:ord

description: polynomial order
type: int
values, default: 3
remarks: Each coefficient will get its output column in the exported tables, thus the table size depends on this order.

11.11 Stylization: Register representation

styl:register

description: register definition for residual calculation
type: string
values, default: **ml**, bl, tl, rng, none
remarks: how to remove the global component from the f0 contour to get the residual the local contour is calculated on; *bl, ml, tl*: base-, mid- or topline subtraction; *rng* pointwise [0 1] normalization of the f0 contour with respect to the base- and topline. Recommended: *ml, rng.* rng normalizes for range declination (lower f0 amplitudes at the end of prosodic phrases).

11.12 Stylization: Boundaries

styl:bnd:cross_chunk

description: stylization windows across chunks
type: boolean
values, default: 1
remarks: if set to 1, the windows defined by `styl:bnd:win` can cross chunks, else they are limited by the current chunk's boundaries. If set to 1 for `do_bnd_trend`, lines are fitted from file start and till file end. Else, they are limited by the current chunk's boundaries.

---

styl:bnd:decl_win

description: window length for median calculation (in sec)
type: float
values, default: 0.1
remarks: Within each window a median each for the base-, mid- and topline is derived.

---

styl:bnd:nrm:mtd

description: time normalization method
type: string
values, default: **minmax**
remarks: Only minmax supported.

---

styl:bnd:nrm:rng

description: normalized time range
type: list of floats
values, default: $[0, 1]$
remarks: to allow for comparisons independent of segment length, time is normalized to this range.

---

styl:bnd:prct:bl

description: percentile below which the baseline input medians are calculated
type: float]0 100[
values, default: 10
remarks: A sequence of lower range medians is calculated along the f0 contour. The baseline is given by linear regression through this sequence.

---

styl:bnd:prct:tl

description: percentile, above which the topline input medians are calculated

type: float]0 100[
values, default: 90
remarks: A sequence of upper range medians is calculated along the f0 contour. The topline is given by linear regression through this sequence.

---

styl:bnd:residual

description: use f0 residual
type: boolean
values, default: 0
remarks: measure discontinuity on (preprocessed) f0 contour or on its residual after register subtraction.

---

styl:bnd:win

description: window length (in sec)
type: float
values, default: 1
remarks: stylization window length for `navigate:do_styl_bnd_win`

11.13 Stylization: General (energy) features

styl:gnl:sb:alpha

description: pre-emphasis factor or lower boundary frequency
type: float
values, default: 0.95
remarks: For pre-emphasis in the time domain for spectral balance calculation. $0 \leq \alpha \leq 1$: factor in $s'[i] = s[i] - \alpha \cdot s[i-1]$. $\alpha > 1$: lower boundary frequency from which pre-emphasis should start. Will be internally converted to the factor in the above formula.

---

styl:gnl:sb:btype

description: filter type to restrict frequency window
type: string
values, default: **none**, band, high, low
remarks: Restrict frequency window for spectral balance calculation

---

styl:gnl:sb:domain

description: domain for spectral balance calculation
type: string
values, default: **time**, freq
remarks: Specifies whether spectral balance should be calculated in the time ('time') or frequency ('freq') domain.

---

styl:gnl:sb:f

description: filter cutoff frequencies (in Hz) for spectral balance calculation
type: float or list of floats
values, default: -1
remarks: Specifies the upper cutoff frequency for a low-pass filter, the lower cutoff frequency for a high-pass filter, or both for a bandpass filter. See `styl:gnl:sb:btype`.

---

styl:gnl:sb:win

description: length (in sec) of central analysis window in analysed segment
type: float
values, default: -1
remarks: To be set if coarticulatory influence on spectral balance calculation should be removed. If -1 the entire segment is used.

---

styl:gnl:win

description: window length to determine initial and final part of contour
type: float
values, default: 0.3
remarks: Length of window (in sec) for initial and final part of f0 or energy contour to calculate mean f0 pr energy quotients of these parts and the entire contour.

---

styl:gnl_en:alpha

description: pre-emphasis factor
type: float
values, default: 0.95
remarks: Pre-emphasis is carried out in the time domain as follows: $s'[i] = s[i] - \alpha \cdot s[i-1]$. **DEPRECATED! NOW SPECIFIED BY styl:gnl_en:sb:alpha**

---

styl:gnl_en:sts

description: step size (in sec)
type: float
values, default: 0.01
remarks: Stepsize by which energy window is shifted.

---

styl:gnl_en:winparam

description: window parameter
type: string or int
values, default: –
remarks: Depends on `styl:gnl_en:wintyp`; as required by *scipy.signal.get_window()*.

---

styl:gnl_en:wintyp

description: window type
type: string
values, default: **hamming**, kaiser, ...
remarks: All window types that are supported by *scipy.signal.get_window()* can be used.

---

styl:gnl_en:win

description: window length (in sec)
type: float
values, default: 0.05
remarks: Energy is calculated in terms of RMSD within windows of this length.

11.14 Stylization: F0 rhythm features

styl:rhy_f0:rhy:lb

description: Lower frequency boundary of DCT coefficients (in Hz)
type: boolean
values, default: 0
remarks: Can be raised if low-frequency events should be ignored.

---

styl:rhy_f0:rhy:nsm

description: number of spectral moments
type: int
values, default: 3
remarks: How many spectral moments to calculate from DCT analysis of f0 contour.

---

styl:rhy_f0:rhy:rmo

description: remove DCT offset
type: boolean
values, default: 0
remarks: Remove first DCT coefficient.

---

styl:rhy_f0:rhy:ub

description: upper frequency boundary of DCT coefficients (in Hz)
type: float
values, default: 10
remarks: Upper boundary of analyzed DCT spectrum (higher-frequency events assumed not to be influential for prosody).

---

styl:rhy_f0:rhy:wgt:rb

description: rate band (in Hz)
type: float
values, default: 1
remarks: Frequency band around event frequency, within which the influence of the event in terms of absolute DCT coefficient values is integrated. E.g. for an event rate of 4 Hz and a rate band of 1 Hz the absolute values of the DCT coefficients between 3 and 5 Hz are summed up.

---

styl:rhy_f0:rhy:winparam

description: window parameter
type: string or int
values, default: 1
remarks: depends on `styl:gnl_en:wintyp`; as required by *scipy.signal.get_window()*.

---

styl:rhy_f0:rhy:wintyp

description: window type for DCT analysis
type: string
values, default: hamming, **kaiser**, ...
remarks: All window types that are supported by *scipy.signal.get_window()* can be used.

11.15 Stylization: Energy rhythm features

styl:rhy_en:rhy:lb

description: lower frequency boundary of DCT coefficients (in Hz)
type: float
values, default: 0
remarks: Can be raised if low-frequency events should be ignored.

---

styl:rhy_en:rhy:nsm

description: number of spectral moments
type: int
values, default: 3
remarks: How many spectral moments to be calculated from DCT analysis of energy contour.

---

styl:rhy_en:rhy:rmo

description: remove DCT offset
type: boolean
values, default: 0
remarks: Remove first DCT coefficient.

---

styl:rhy_en:rhy:ub

description: upper frequency boundary of DCT coefficients (in Hz)
type: float
values, default: 10
remarks: Upper boundary of analyzed DCT spectrum (higher-frequency events assumed not to be influential for prosody).

---

styl:rhy_en:rhy:wgt:rb

description: rate band (in Hz)
type: float
values, default: 1
remarks: Frequency band around event frequency, within which the influence of the event in terms of absolute DCT coefficient values is integrated. E.g. for an event rate of 4 Hz and a rate band of 1 Hz the absolute values of the DCT coefficients between 3 and 5 Hz are summed up.

---

styl:rhy_en:rhy:winparam

description: DCT window parameter
type: string or int
values, default: 1
remarks: Depends on `styl:rhy_en:wintyp`; as required by *scipy.signal.get_window()*.

---

styl:rhy_en:rhy:wintyp

description: window type for DCT
type: string
values, default: hamming, **kaiser**, ...
remarks: All window types that are supported by *scipy.signal.get_window()*.

---

styl:rhy_en:sig:scale

description: scale signal to maximum amplitude 1
type: boolean
values, default: 1
remarks: if set to 1, the signal is scaled to its maximum amplitude. This is suggested especially if signals of different recording conditions are to be compared.

---

styl:rhy_en:sig:sts

description: step size (in sec)
type: float
values, default: 0.01
remarks: Step size by which the energy window is shifted.

---

styl:rhy_en:sig:winparam

description: window parameter
type: string or int
values, default: –
remarks: Depends on `styl:rhy_en:wintyp`; as required by *scipy.signal.get_window()*.

---

styl:rhy_en:sig:wintyp

description: window type of energy calculation
type: string

values, default: **hamming**, kaiser, ...
remarks: all window types that are supported by *scipy.signal.get_window()*.

---

styl:rhy_en:sig:win

description: window length (in sec)
type: float
values, default: 0.05
remarks: Energy is calculated in terms of RMSD within windows of this length.

11.16 Stylization: Voice quality features

styl:voice:jit:fac_max

description: maximally allowed quotient of adjacent periods
type: float
values, default: 1.3
remarks: corresponds to Praat parameter Maximum period factor.

---

styl:voice:jit:t_max

description: maximum period length in sec
type: float
values, default: 0.02
remarks: corresponds to Praat parameter Period ceiling.

---

styl:voice:jit:t_min

description: minimum period length in sec
type: float
values, default: 0.0001
remarks: corresponds to Praat parameter Period floor.

11.17 Clustering: Global contours

clst:glob:estimate_bandwidth:n_samples

description: number of samples to estimate bandwidth
type: integer
values, default: 1000
remarks: Computationally expensive, high numbers will require long processing time.

---

clst:glob:estimate_bandwidth:quantile

description: estimate_bandwidth quantile parameter
type: float
values, default: 0.3
remarks: Lower values result in higher clusters numbers.

---

clst:glob:kMeans:init

description: initialization method of kmeans
type: string
values, default: meanShift
remarks: All methods that are supported by *kMeans()* can be used. For *meanShift* the number of clusters does not need to be specified.

---

clst:glob:kMeans:max_iter

description: kMeans: maximum number of iterations
type: int
values, default: 300
remarks: When to stop cluster re-adjustment, if not yet converged.

---

clst:glob:kMeans:n_cluster

description: kMeans: predefined number of contour classes
type: int
values, default: 3
remarks: Irrelevant, if kmeans centroids are initialized by `clst:glob:kMeans:init=meanShift`.

---

clst:glob:kMeans:n_init

description: number of initialization trials
type: int
values, default: 10
remarks: kMeans is repeated with different cluster initializations from which the best clustering result is kept.

clst:glob:meanShift:bandwidth

description: bandwidth parameter for meanShift cluster center initialization
type: float
values, default: 0
remarks: 0 indicates, that the optimal bandwidth is internally calculated.

clst:glob:meanShift:bin_seeding

description: bin seeding
type: boolean
values, default: 0
remarks: parameter for meanShift clustering

clst:glob:meanShift:min_bin_freq

description: minimum number of items in each bin
type: int
values, default: 1
remarks: Parameter for meanShift clustering.

clst:glob:mtd

description: clustering method
type: string
values, default: **meanShift**, kmeans
remarks: No initial cluster number specification needed for *meanShift*.

11.18 Clustering: Local contours

clst:loc:estimate_bandwidth:n_samples

description: number of samples to estimate bandwidth
type: int
values, default: 1000
remarks: Computationally expensive, high numbers will require long processing time.

clst:loc:estimate_bandwidth:quantile

description: estimate_bandwidth quantile parameter
type: float
values, default: 0.3
remarks: Lower values result in higher clusters numbers.

clst:loc:kMeans:init

description: initialization method of kmeans
type: string
values, default: meanShift
remarks: All methods that are supported by *kMeans()* can be used. For *meanShift* the number of clusters does not need to be specified.

clst:loc:kMeans:max_iter

description: kMeans: maximum number of iterations
type: int
values, default: 300
remarks: When to stop cluster re-adjustment, if not yet converged.

clst:loc:kMeans:n_cluster

description: kMeans: predefined number of contour classes
type: int
values, default: 5
remarks: Irrelevant, if kmeans centroids are initialized by `clst:glob:kMeans:init=meanShift`.

clst:loc:kMeans:n_init

description: number of initialization trials
type: int
values, default: 10
remarks: kMeans is repeated with different cluster initializations from which the best clustering result is kept.

clst:loc:meanShift:bandwidth

description: bandwidth parameter for meanShift cluster center initialization
type: boolean

values, default: 0
remarks:

---

clst:loc:meanShift:bin_seeding

description: bin seeding
type: boolean
values, default: 0
remarks: parameter for meanShift clustering

---

clst:loc:meanShift:min_bin_freq

description: minimum number of items in each bin
type: int
values, default: 1
remarks: Parameter for meanShift clustering.

---

clst:loc:mtd

description: clustering method
type: string
values, default: **meanShift**, kmeans
remarks: No initial cluster number specification needed for *meanShift*.

11.19 Plotting: Browsing

plot:browse:grp

description: plot for selected grouping values only
type: dict
values, default: empty
remarks: This dict contains zero or more *myGroupingKey-myGroupingValue* pairs. Each *myGroupingKey* should match one of the strings in `fsys:grp:lab`. By this the user can select to plot images with (a combination of) certain grouping values only.

---

plot:browse:save

description: save plots according to `fsys:pic`
type: boolean
values, default: 0
remarks: Store png files in `fsys:pic:dir` with file name stem `fsys:pic:stm`.

---

plot:browse:single_plot:active

description: switch on single plot mode
type: boolean
values, default: 0
remarks: switch on single plot mode if only one segment specified by file index, channel index, and segment index is to be plotted

---

plot:browse:single_plot:channel_i

description: channel index of selected segment
type: integer
values, default: 0
remarks: channel index of selected segment to be plotted

---

plot:browse:single_plot:file_i

description: file index of selected segment
type: integer
values, default: 0
remarks: file index of selected segment to be plotted

---

plot:browse:single_plot:segment_i

description: segment index of selected segment
type: integer
values, default: 0
remarks: segment index of selected segment to be plotted

---

plot:browse:time

description: when to do plotting
type: string
values, default: online, **final**

remarks: *online:* plot at stylization stage for immediate check of appropriateness of configurations. *final:* plot segment-wise from the finally stored results. Click on plot: next; press *return*: quit.

plot:browse:type:clst:contours

description: plot global and local intonation class centroids
type: boolean
values, default: 0
remarks:

plot:browse:type:complex:gestalt

description: plot local contour Gestalt stylization
type: boolean
values, default: 0
remarks:

plot:browse:type:complex:superpos

description: plot global and local contour superposition
type: boolean
values, default: 0
remarks:

plot:browse:type:glob:decl

description: plot global contour register stylization
type: boolean
values, default: 0
remarks:

plot:browse:type:loc:acc

description: plot local contour polynomial stylization
type: boolean
values, default: 0
remarks:

plot:browse:type:loc:decl

description: plot local contour register stylization
type: boolean
values, default: 0
remarks:

plot:browse:type:complex:bnd

description: plot boundary stylization
type: boolean
values, default: 0
remarks:

plot:browse:type:complex:bnd_win

description: plot boundary stylization (fixed window)
type: boolean
values, default: 0
remarks:

plot:browse:type:complex:bnd_trend

description: plot boundary stylization (trend)
type: boolean
values, default: 0
remarks:

plot:browse:type:rhy_en:rhy

description: plot influence of rate tier events on DCT of energy contour in analysis tier
type: boolean
values, default: 0
remarks:

plot:browse:type:rhy_f0:rhy

description: plot influence of rate tier events on DCT of f0 contour in analysis tier
type: boolean
values, default: 0
remarks:

plot:browse:verbose

description: display file, channel and segment index for each plot
type: boolean
values, default: 0
remarks: written to STDOUT

plot:color

description: plot in color (1) or black-white (0)
type: boolean
values, default: 1
remarks:

11.20 Plotting: Grouping

plot:grp:grouping

description: list of selected grouping variables from `fsys:grp:lab`
type: list of strings
values, default: [ ]
remarks: For each combination of grouping factor levels the stylization plot based on the respective parameter mean vector is stored as a png file in `fsys:pic:dir` with file name stem `fsys:pic:stm` and an infix expressing the respective factor level combination.

plot:grp:save

description: save plots according to `fsys:pic`
type: boolean
values, default: 0
remarks: Store png files in `fsys:pic:dir` with file name stem `fsys:pic:stm`. One file per group.

plot:grp:type:glob:decl

description: plot global contour declination centroid for each group
type: boolean
values, default: 0
remarks: Plots are not displayed but saved as png files to `fsys:pic`.

plot:grp:type:loc:acc

description: plot local contour polynomial shape centroid for each group
type: boolean
values, default: 0
remarks: Plots are not displayed but saved as png files to `fsys:pic`.

plot:grp:type:loc:decl

description: plot local contour declination centroid for each group
type: boolean
values, default: 0
remarks: Plots are not displayed but saved as png files to `fsys:pic`.

12 Output

12.1 Table files

If `fsys:export:csv` is set to 1, for each feature set selected by the `navigate:*` options a csv table file with alphanumerically sorted columns is generated in `config:fsys:export:dir`. The file name is the underscore-concatenation of `config:fsys:export:stm` and the feature set name. Extension is *csv*. Columns are separated by a comma. The column titles correspond to the feature names given in the tables in section 10, and each row corresponds to one segment or event for which the features were extracted. These feature vectors are additionally linked to the data origin by the following columns:

name	**description**
ci	channel index (starting with 0)
fi	file index (starting with 0)
ii	item (segment or event) index (starting with 0)
stm	annotation file name stem
t_on	time onset
t_off	time offset (same as t_on for events)
tier	tier name

Inter-tier relations are provided by the following columns

name	description
is_init	initial position in a global segment
is_fin	final position in a global segment
is_init_chunk	initial position in a chunk
is_fin_chunk	final position in a chunk

All columns contain the values *yes* and *no*. Medial position is simply indicated by *is_init=no* and *is_fin=no*. These columns can be used for data subsetting. As an example, let's assume that boundary features were extracted between accent groups, and the global segments correspond to intonation phrases. Then *is_fin* serves to hold apart IP-final and non-final boundaries. Equivalently, phrase-final and non-final accents can be held apart. *is_init_chunk* and *is_fin_chunk* work the same on the chunk level. If no chunk tier is specified, the entire channel is considered to be a single chunk. If no global segment tier is specified, all *is_init* and *is_fin* are set to *no*.

Finally, if specified by the user, an arbitrary number of grouping columns will be added to the tables that are derived from the filenames. Their names are prefixed by `grp`. See the grouping options `fsys:grp:*` in section 11.3 for details. Each table file comes along with an R code template file with the same name and the extension *.R* to read this table by the R software.

12.2 Summary table files

By setting `fsys:export:summary` to 1 the table output described in section 12.1 can be summarized per file and analysis tier. Summarization for continuous-valued features is done in terms of their mean, median, standard deviation, and inter-quartile range. For categorical features as intonation contour classes the unigram entropy is calculated. The resulting table is written to the directory `fsys:export:dir` with the file stem `fsys:export:stm` plus the suffix *summary* and the extension *csv*. Columns are separated by a comma. There is one row of statistic values per analysed tier in a file. Each continous-valued feature within each analysis tier is represented by four columns. For features of the sets *glob* and *loc* for which there is only one analysis tier the column names follow the pattern *featureSet_featureName_statisticMeasure*. The suffixes representing the statistic measurements are listed in the table right below. For features of all other sets with potentially more than one analysis tier the column names are built like this: *featureSet_analysisTierName_featureName_statisticMeasure*. Categorical features are represented by one column each with the same name building schema.

suffix	meaning	feature type
m	arit. mean	continuous
med	median	continuous
sd	standard deviation	continuous
iqr	inter-quartile range	continuous
h	unigram entropy	categorical

File level groupings, i.e. the *grp_** columns of the csv tables decribed in section 12.1, are copied to the summary table. File and channel index are given in the columns *fi* and *ci*, respectively, the file stem is written to column *stm*. Columns are sorted alphanumerically by their names.

Next to the csv file an R code template file is generated with the same name and the extension *.R* to read the summary table by the R software.

12.3 Nested Python dictionary

The pickle file which is outputted in `config:fsys:export:dir` contains a nested dictionary *copa* for the sake of further processing within other Python projects.

On the top level *copa* can be subdivided into the sub-dictionaries

- `config`: configurations underlying the current analysis
- `export`: Pandas Dataframes of extracted features. One dataframe per feature set
- `data`: extracted features in a nested dictionary described below
- `clst`: contour clustering results
- `val`: validation metrics for stylization and clustering

12.3.1 Configuration sub-dictionary

This sub-dictionary is accessed by `copa['config']` and simply contains a copy of the user-defined and default configurations which are introduced in section 11.

12.3.2 Stylization feature table sub-dictionary

Is stored in `copa['export']`. The Pandas Dataframe for each feature set can be accessed by the feature set's name. For example, the local contour stylization parameters can be found here:

```
copa['export']['loc']
```

Same for the other feature sets. All standard and rhythm features are additionally accessible on the file level. Thus `copa['export']` is structured as follows[3]:

export:bnd

description: boundary features
type: Pandas Dataframe

---

export:gnl_en

description: standard energy features
type: Pandas Dataframe

---

export:gnl_f0

description: standard f0 features
type: Pandas Dataframe

---

export:gnl_en_file

description: ... on file level
type: Pandas Dataframe

---

export:gnl_f0_file

description: ... on file level
type: Pandas Dataframe

---

export:loc

description: local f0 contour features
type: Pandas Dataframe

---

export:glob

description: global f0 contour features
type: Pandas Dataframe

---

export:rhy_en

description: energy rhythm features
type: Pandas Dataframe

---

export:rhy_f0

description: f0 rhythm features
type: Pandas Dataframe

---

export:rhy_en_file

description: ... on file level
type: Pandas Dataframe

---

export:rhy_f0_file

description: ... on file level
type: Pandas Dataframe

---

export:voi

description: voice quality features
type: Pandas Dataframe

---

[3] `export:x` is to be expanded as `copa['export'][x]`

12.3.3 Stylization feature nested sub-dictionary

In the subsequent paragraphs all branches through the copasul nested output dictionary are described. The following index key conventions will be used:

fi	file index
ci	channel index
ti	tier index
ii	item (segment or event) index

All indices start with 0, thus channel 1 is represented by index 0, etc. Levels in the dictionary are separated by colons. To give an example for the `data` sub-dictionary how to translate this notation into Python code:

`data:fi:ci:bnd:ti:ii:lab` – with index values:
`data:0:0:bnd:0:0:lab`

refers to: file 1 : channel 1 : boundary feature set : first tier, for which this set was extracted : segment 1 in this tier : label of this segment. In Python this label can be accessed by:

```
copa['data'][0][0]['bnd'][0][0]['lab']
```

Variables to be replaced by annotation-dependent tiernames etc. are marked by `my*`. As an example

`data:0:0:rhy_f0:0:0:rate:myTierName*`

is expanded to one branch for each tier in `fsys:rhy_f0:tier_rate` referring to channel 1 in `fsys:channel` (see section 11). Let `fsys:rhy_f0:tier_rate`=["syl_1", "syl_2"], of which only the former refers to channel 1, i.e. `fsys:channel:syl_1`=1. Then the corresponding rate value of items in tier `syl_1` within file 1, channel 1, segment 1, and analysis tier 1 (`fi=ci=ti=ii=0`) is addressed in Python by:

```
copa['data'][0][0]['rhy_f0'][0][0]['rate']['syl_1']
```

The stylization feature subdirectory accessed by `copa['data']` and can further be subdivided into dictionaries for file information, f0 preprocessing output, chunk segmentation, and feature sets. Time information is always given in seconds and can be accessed by the keys `t, tn, to, tt`. `to` always contains the original time values derived from the annotations, while `t, tn`, and `tt` values are rounded to the second decimal place to be in sync with f0 values that are sampled at 100 Hz. The semantics of `t, tn, to, tt` depends on the respective sub-dictionary. In the following all `copa['data']` branches will be described in alphabetical order. If a feature variable at the end of a branch is listed in one of the tables in section 10, here only the feature name is given, which can be looked up in these tables.

Boundary features Boundary features can be extracted within an arbitrary number of tiers. These tiers are indexed by the variable *ti*. For segment tiers the index *ii* refers to the segment **preceding** the boundary.

data:fi:ci:bnd:ti:ii:decl:bl:c

description: F0 baseline coefficients (descending order)
type: 2-element list of floats

---

data:fi:ci:bnd:ti:ii:decl:bl:x

description: baseline stylization input
type: list of floats

---

data:fi:ci:bnd:ti:ii:decl:bl:y

description: stylized baseline values
type: list of floats

---

data:fi:ci:bnd:ti:ii:decl:err

description: True if top- and baseline cross
type: boolean

---

data:fi:ci:bnd:ti:ii:decl:ml:c

description: F0 midline coefficients (descending order)
type: 2-element list of floats

---

data:fi:ci:bnd:ti:ii:decl:ml:x

description: midline stylization input
type: list of floats

---

data:fi:ci:bnd:ti:ii:decl:ml:y

description: stylized midline values
type: list of floats

---

data:fi:ci:bnd:ti:ii:decl:rng:c

description: F0 range coefficients (descending order)
type: 2-element list of floats

---

data:fi:ci:bnd:ti:ii:decl:rng:x

description: range stylization input
type: list of floats

---

data:fi:ci:bnd:ti:ii:decl:rng:y

description: stylized range values
type: list of floats

---

data:fi:ci:bnd:ti:ii:decl:tl:c

description: F0 topline coefficients (descending order)
type: 2-element list of floats

---

data:fi:ci:bnd:ti:ii:decl:tl:x

description: topline stylization input
type: list of floats

---

data:fi:ci:bnd:ti:ii:decl:tl:y

description: stylized topline values
type: list of floats

---

data:fi:ci:bnd:ti:ii:decl:tn

description: normalized time values (same length as *bl|ml|rng|tl:y*)
type: list of floats

---

data:fi:ci:bnd:ti:ii:lab

description: *lab*
type: string

---

data:fi:ci:bnd:ti:ii:std:bl:aicI

description: *std_bl_aicI*
type: float

---

data:fi:ci:bnd:ti:ii:std:bl:aicI_post

description: *std_bl_aicI_post*
type: float

---

data:fi:ci:bnd:ti:ii:std:bl:aicI_pre

description: *std_bl_aicI_pre*
type: float

---

data:fi:ci:bnd:ti:ii:std:bl:corrD

description: *std_bl_corrD*
type: float

---

data:fi:ci:bnd:ti:ii:std:bl:corrD_post

description: *std_bl_corrD_post*
type: float

---

data:fi:ci:bnd:ti:ii:std:bl:corrD_pre

description: *std_bl_corrD_pre*
type: float

---

data:fi:ci:bnd:ti:ii:std:bl:d_m

description: *std_bl_d_m*
type: float

---

data:fi:ci:bnd:ti:ii:std:bl:d_o

description: *std_bl_d_o*
type: float

data:fi:ci:bnd:ti:ii:std:bl:r

description: *std_bl_r*
type: float

data:fi:ci:bnd:ti:ii:std:bl:rms

description: *std_bl_rms*
type: float

data:fi:ci:bnd:ti:ii:std:bl:rms_post

description: *std_bl_rms_post*
type: float

data:fi:ci:bnd:ti:ii:std:bl:rms_pre

description: *std_bl_rms_pre*
type: float

data:fi:ci:bnd:ti:ii:std:bl:rmsR

description: *std_bl_rmsR*
type: float

data:fi:ci:bnd:ti:ii:std:bl:rmsR_post

description: *std_bl_rmsR_post*
type: float

data:fi:ci:bnd:ti:ii:std:bl:rmsR_pre

description: *std_bl_rmsR_pre*
type: float

data:fi:ci:bnd:ti:ii:std:bl:sd_post

description: *std_bl_sd_post*
type: float

data:fi:ci:bnd:ti:ii:std:bl:sd_pre

description: *std_bl_sd_pre*
type: float

data:fi:ci:bnd:ti:ii:std:bl:sd_prepost

description: *std_bl_sd_prepost*
type: float

data:fi:ci:bnd:ti:ii:std:ml:aicI

description: *std_ml_aicI*
type: float

data:fi:ci:bnd:ti:ii:std:ml:aicI_post

description: *std_ml_aicI_post*
type: float

data:fi:ci:bnd:ti:ii:std:ml:aicI_pre

description: *std_ml_aicI_pre*
type: float

data:fi:ci:bnd:ti:ii:std:ml:corrD

description: *std_ml_corrD*
type: float

data:fi:ci:bnd:ti:ii:std:ml:corrD_post

description: *std_ml_corrD_post*
type: float

data:fi:ci:bnd:ti:ii:std:ml:corrD_pre

description: *std_ml_corrD_pre*
type: float

data:fi:ci:bnd:ti:ii:std:ml:d_m

description: *std_ml_d_m*
type: float

---

data:fi:ci:bnd:ti:ii:std:ml:d_o

description: *std_ml_d_o*
type: float

---

data:fi:ci:bnd:ti:ii:std:ml:r

description: *std_ml_r*
type: float

---

data:fi:ci:bnd:ti:ii:std:ml:rms

description: *std_ml_rms*
type: float

---

data:fi:ci:bnd:ti:ii:std:ml:rms_post

description: *std_ml_rms_post*
type: float

---

data:fi:ci:bnd:ti:ii:std:ml:rms_pre

description: *std_ml_rms_pre*
type: float

---

data:fi:ci:bnd:ti:ii:std:ml:rmsR

description: *std_ml_rmsR*
type: float

---

data:fi:ci:bnd:ti:ii:std:ml:rmsR_post

description: *std_ml_rmsR_post*
type: float

---

data:fi:ci:bnd:ti:ii:std:ml:rmsR_pre

description: *std_ml_rmsR_pre*
type: float

---

data:fi:ci:bnd:ti:ii:std:ml:sd_post

description: *std_ml_sd_post*
type: float

---

data:fi:ci:bnd:ti:ii:std:ml:sd_pre

description: *std_ml_sd_pre*
type: float

---

data:fi:ci:bnd:ti:ii:std:ml:sd_prepost

description: *std_ml_sd_prepost*
type: float

---

data:fi:ci:bnd:ti:ii:std:p

description: *p*
type: float

---

data:fi:ci:bnd:ti:ii:std:rng:aicI

description: *std_rng_aicI*
type: float

---

data:fi:ci:bnd:ti:ii:std:rng:aicI_post

description: *std_rng_aicI_post*
type: float

---

data:fi:ci:bnd:ti:ii:std:rng:aicI_pre

description: *std_rng_aicI_pre*
type: float

---

data:fi:ci:bnd:ti:ii:std:rng:corrD

description: *std_rng_corrD*
type: float

data:fi:ci:bnd:ti:ii:std:rng:corrD_post

description: *std_rng_corrD_post*
type: float

data:fi:ci:bnd:ti:ii:std:rng:corrD_pre

description: *std_rng_corrD_pre*
type: float

data:fi:ci:bnd:ti:ii:std:rng:d_m

description: *std_rng_d_m*
type: float

data:fi:ci:bnd:ti:ii:std:rng:d_o

description: *std_rng_d_o*
type: float

data:fi:ci:bnd:ti:ii:std:rng:r

description: *std_rng_r*
type: float

data:fi:ci:bnd:ti:ii:std:rng:rms

description: *std_rng_rms*
type: float

data:fi:ci:bnd:ti:ii:std:rng:rms_post

description: *std_rng_rms_post*
type: float

data:fi:ci:bnd:ti:ii:std:rng:rms_pre

description: *std_rng_rms_pre*
type: float

data:fi:ci:bnd:ti:ii:std:rng:rmsR

description: *std_rng_rmsR*
type: float

data:fi:ci:bnd:ti:ii:std:rng:rmsR_post

description: *std_rng_rmsR_post*
type: float

data:fi:ci:bnd:ti:ii:std:rng:rmsR_pre

description: *std_rng_rmsR_pre*
type: float

data:fi:ci:bnd:ti:ii:std:rng:sd_post

description: *std_rng_sd_post*
type: float

data:fi:ci:bnd:ti:ii:std:rng:sd_pre

description: *std_rng_sd_pre*
type: float

data:fi:ci:bnd:ti:ii:std:rng:sd_prepost

description: *std_rng_sd_prepost*
type: float

data:fi:ci:bnd:ti:ii:std:tl:aicI

description: *std_tl_aicI*
type: float

data:fi:ci:bnd:ti:ii:std:tl:aicI_post

description: *std_tl_aicI_post*
type: float

data:fi:ci:bnd:ti:ii:std:tl:aicI_pre

description: *std_tl_aicI_pre*
type: float

---

data:fi:ci:bnd:ti:ii:std:tl:corrD

description: *std_bl_corrD*
type: float

---

data:fi:ci:bnd:ti:ii:std:tl:corrD_post

description: *std_tl_corrD_post*
type: float

---

data:fi:ci:bnd:ti:ii:std:tl:corrD_pre

description: *std_tl_corrD_pre*
type: float

---

data:fi:ci:bnd:ti:ii:std:tl:d_m

description: *std_tl_d_m*
type: float

---

data:fi:ci:bnd:ti:ii:std:tl:d_o

description: *std_tl_d_o*
type: float

---

data:fi:ci:bnd:ti:ii:std:tl:r

description: *std_tl_r*
type: float

---

data:fi:ci:bnd:ti:ii:std:tl:rms

description: *std_tl_rms*
type: float

---

data:fi:ci:bnd:ti:ii:std:tl:rms_post

description: *std_tl_rms_post*
type: float

---

data:fi:ci:bnd:ti:ii:std:tl:rms_pre

description: *std_tl_rms_pre*
type: float

---

data:fi:ci:bnd:ti:ii:std:tl:rmsR

description: *std_tl_rmsR*
type: float

---

data:fi:ci:bnd:ti:ii:std:tl:rmsR_post

description: *std_tl_rmsR_post*
type: float

---

data:fi:ci:bnd:ti:ii:std:tl:rmsR_pre

description: *std_tl_rmsR_pre*
type: float

---

data:fi:ci:bnd:ti:ii:std:tl:sd_post

description: *std_tl_sd_post*
type: float

---

data:fi:ci:bnd:ti:ii:std:tl:sd_pre

description: *std_tl_sd_pre*
type: float

---

data:fi:ci:bnd:ti:ii:std:tl:sd_prepost

description: *std_tl_sd_prepost*
type: float

---

data:fi:ci:bnd:ti:ii:t

description: segment tier: time start and end of current segment; event tier: interval from the preceding to the current time stamp (for `bnd:...:std` features)

type: 2 element list of floats

---

data:fi:ci:bnd:ti:ii:tier

description: tier name
type: string

---

data:fi:ci:bnd:ti:ii:tn

description: start and end of pre-boundary analysis window, start and end of post-boundary analysis window (for `bnd:...:win` features)
type: 4 element list of floats

---

data:fi:ci:bnd:ti:ii:to

description: t non-rounded
type: 2 element list of floats

---

data:fi:ci:bnd:ti:ii:trend:bl:aicI

description: *trend_bl_aicI*
type: float

---

data:fi:ci:bnd:ti:ii:trend:bl:aicI_post

description: *trend_bl_aicI_post*
type: float

---

data:fi:ci:bnd:ti:ii:trend:bl:aicI_pre

description: *trend_bl_aicI_pre*
type: float

---

data:fi:ci:bnd:ti:ii:trend:bl:corrD

description: *trend_bl_corrD*
type: float

---

data:fi:ci:bnd:ti:ii:trend:bl:corrD_post

description: *trend_bl_corrD_post*
type: float

---

data:fi:ci:bnd:ti:ii:trend:bl:corrD_pre

description: *trend_bl_corrD_pre*
type: float

---

data:fi:ci:bnd:ti:ii:trend:bl:d_m

description: *trend_bl_d_m*
type: float

---

data:fi:ci:bnd:ti:ii:trend:bl:d_o

description: *trend_bl_d_o*
type: float

---

data:fi:ci:bnd:ti:ii:trend:bl:r

description: *trend_bl_r*
type: float

---

data:fi:ci:bnd:ti:ii:trend:bl:rms

description: *trend_bl_rms*
type: float

---

data:fi:ci:bnd:ti:ii:trend:bl:rms_post

description: *trend_bl_rms_post*
type: float

---

data:fi:ci:bnd:ti:ii:trend:bl:rms_pre

description: *trend_bl_rms_pre*
type: float

---

data:fi:ci:bnd:ti:ii:trend:bl:rmsR

description: *trend_bl_rmsR*
type: float

data:fi:ci:bnd:ti:ii:trend:bl:rmsR_post

description: *trend_bl_rmsR_post*
type: float

data:fi:ci:bnd:ti:ii:trend:bl:rmsR_pre

description: *trend_bl_rmsR_pre*
type: float

data:fi:ci:bnd:ti:ii:trend:bl:sd_post

description: *trend_bl_sd_post*
type: float

data:fi:ci:bnd:ti:ii:trend:bl:sd_pre

description: *trend_bl_sd_pre*
type: float

data:fi:ci:bnd:ti:ii:trend:bl:sd_prepost

description: *trend_bl_sd_prepost*
type: float

data:fi:ci:bnd:ti:ii:trend:ml:aicI

description: *trend_ml_aicI*
type: float

data:fi:ci:bnd:ti:ii:trend:ml:aicI_post

description: *trend_ml_aicI_post*
type: float

data:fi:ci:bnd:ti:ii:trend:ml:aicI_pre

description: *trend_ml_aicI_pre*
type: float

data:fi:ci:bnd:ti:ii:trend:ml:corrD

description: *trend_ml_corrD*
type: float

data:fi:ci:bnd:ti:ii:trend:ml:corrD_post

description: *trend_bl_corrD_post*
type: float

data:fi:ci:bnd:ti:ii:trend:ml:corrD_pre

description: *trend_bl_corrD_pre*
type: float

data:fi:ci:bnd:ti:ii:trend:ml:d_m

description: *trend_ml_d_m*
type: float

data:fi:ci:bnd:ti:ii:trend:ml:d_o

description: *trend_ml_d_o*
type: float

data:fi:ci:bnd:ti:ii:trend:ml:r

description: *trend_ml_r*
type: float

data:fi:ci:bnd:ti:ii:trend:ml:rms

description: *trend_ml_rms*
type: float

data:fi:ci:bnd:ti:ii:trend:ml:rms_post

description: *trend_ml_rms_post*
type: float

data:fi:ci:bnd:ti:ii:trend:ml:rms_pre

description: *trend_ml_rms_pre*
type: float

---

data:fi:ci:bnd:ti:ii:trend:ml:rmsR

description: *trend_ml_rmsR*
type: float

---

data:fi:ci:bnd:ti:ii:trend:ml:rmsR_post

description: *trend_ml_rmsR_post*
type: float

---

data:fi:ci:bnd:ti:ii:trend:ml:rmsR_pre

description: *trend_ml_rmsR_pre*
type: float

---

data:fi:ci:bnd:ti:ii:trend:ml:sd_post

description: *trend_ml_sd_post*
type: float

---

data:fi:ci:bnd:ti:ii:trend:ml:sd_pre

description: *trend_ml_sd_pre*
type: float

---

data:fi:ci:bnd:ti:ii:trend:ml:sd_prepost

description: *trend_ml_sd_prepost*
type: float

---

data:fi:ci:bnd:ti:ii:trend:p

description: *trend_ml_rms_pre*
type: float

---

data:fi:ci:bnd:ti:ii:trend:rng:aicI

description: *trend_rng_aicI*
type: float

---

data:fi:ci:bnd:ti:ii:trend:rng:aicI_post

description: *trend_rng_aicI_post*
type: float

---

data:fi:ci:bnd:ti:ii:trend:rng:aicI_pre

description: *trend_rng_aicI_pre*
type: float

---

data:fi:ci:bnd:ti:ii:trend:rng:corrD

description: *trend_rng_corrD*
type: float

---

data:fi:ci:bnd:ti:ii:trend:rng:corrD_post

description: *trend_rng_corrD_post*
type: float

---

data:fi:ci:bnd:ti:ii:trend:rng:corrD_pre

description: *trend_rng_corrD_pre*
type: float

---

data:fi:ci:bnd:ti:ii:trend:rng:d_m

description: *trend_rng_d_m*
type: float

---

data:fi:ci:bnd:ti:ii:trend:rng:d_o

description: *trend_rng_d_o*
type: float

---

data:fi:ci:bnd:ti:ii:trend:rng:r

description: *trend_rng_r*
type: float

data:fi:ci:bnd:ti:ii:trend:rng:rms

description: *trend_rng_rms*
type: float

data:fi:ci:bnd:ti:ii:trend:rng:rms_post

description: *trend_rng_rms_post*
type: float

data:fi:ci:bnd:ti:ii:trend:rng:rms_pre

description: *trend_rng_rms_pre*
type: float

data:fi:ci:bnd:ti:ii:trend:rng:rmsR

description: *trend_rng_rmsR*
type: float

data:fi:ci:bnd:ti:ii:trend:rng:rmsR_post

description: *trend_rng_rmsR_post*
type: float

data:fi:ci:bnd:ti:ii:trend:rng:rmsR_pre

description: *trend_rng_rmsR_pre*
type: float

data:fi:ci:bnd:ti:ii:trend:rng:sd_post

description: *trend_rng_sd_post*
type: float

data:fi:ci:bnd:ti:ii:trend:rng:sd_pre

description: *trend_rng_sd_pre*
type: float

data:fi:ci:bnd:ti:ii:trend:rng:sd_prepost

description: *trend_rng_sd_prepost*
type: float

data:fi:ci:bnd:ti:ii:trend:tl:aicI

description: *trend_bl_aicI*
type: float

data:fi:ci:bnd:ti:ii:trend:tl:aicI_post

description: *trend_tl_aicI_post*
type: float

data:fi:ci:bnd:ti:ii:trend:tl:aicI_pre

description: *trend_tl_aicI_pre*
type: float

data:fi:ci:bnd:ti:ii:trend:tl:corrD

description: *trend_tl_corrD*
type: float

data:fi:ci:bnd:ti:ii:trend:tl:corrD_post

description: *trend_tl_corrD_post*
type: float

data:fi:ci:bnd:ti:ii:trend:tl:corrD_pre

description: *trend_tl_corrD_pre*
type: float

data:fi:ci:bnd:ti:ii:trend:tl:d_m

description: *trend_tl_d_m*
type: float

data:fi:ci:bnd:ti:ii:trend:tl:d_o

description: *trend_tl_d_o*
type: float

---

data:fi:ci:bnd:ti:ii:trend:tl:r

description: *trend_tl_r*
type: float

---

data:fi:ci:bnd:ti:ii:trend:tl:rms

description: *trend_tl_rms*
type: float

---

data:fi:ci:bnd:ti:ii:trend:tl:rms_post

description: *trend_tl_rms_post*
type: float

---

data:fi:ci:bnd:ti:ii:trend:tl:rms_pre

description: *trend_tl_rms_pre*
type: float

---

data:fi:ci:bnd:ti:ii:trend:tl:rmsR

description: *trend_tl_rmsR*
type: float

---

data:fi:ci:bnd:ti:ii:trend:tl:rmsR_post

description: *trend_tl_rmsR_post*
type: float

---

data:fi:ci:bnd:ti:ii:trend:tl:rmsR_pre

description: *trend_tl_rmsR_pre*
type: float

---

data:fi:ci:bnd:ti:ii:trend:tl:sd_post

description: *trend_tl_sd_post*
type: float

---

data:fi:ci:bnd:ti:ii:trend:tl:sd_pre

description: *trend_tl_sd_pre*
type: float

---

data:fi:ci:bnd:ti:ii:trend:tl:sd_prepost

description: *trend_tl_sd_prepost*
type: float

---

data:fi:ci:bnd:ti:ii:tt

description: start and endpoint for 2 trend windows: from file or chunk start (depending on the *styl:bnd:cross_chunk* value in the configurations, see section 11) till the end of the pre-boundary segment; from the start of the post-boundary segment till file/chunk end (for `bnd:...:trend` features).
type: 4 element list of floats

---

data:fi:ci:bnd:ti:ii:win:bl:aicI

description: *win_bl_aicI*
type: float

---

data:fi:ci:bnd:ti:ii:win:bl:aicI_post

description: *win_bl_aicI_post*
type: float

---

data:fi:ci:bnd:ti:ii:win:bl:aicI_pre

description: *win_bl_aicI_pre*
type: float

---

data:fi:ci:bnd:ti:ii:win:bl:corrD

description: *win_bl_corrD*
type: float

---

data:fi:ci:bnd:ti:ii:win:bl:corrD_post

description: *win_bl_corrD_post*
type: float

---

data:fi:ci:bnd:ti:ii:win:bl:corrD_pre

description: *win_bl_corrD_pre*
type: float

---

data:fi:ci:bnd:ti:ii:win:bl:d_m

description: *win_bl_d_m*
type: float

---

data:fi:ci:bnd:ti:ii:win:bl:d_o

description: *win_bl_d_o*
type: float

---

data:fi:ci:bnd:ti:ii:win:bl:r

description: *win_bl_r*
type: float

---

data:fi:ci:bnd:ti:ii:win:bl:rms

description: *win_bl_rms*
type: float

---

data:fi:ci:bnd:ti:ii:win:bl:rms_post

description: *win_bl_rms_post*
type: float

---

data:fi:ci:bnd:ti:ii:win:bl:rms_pre

description: *win_bl_rms_pre*
type: float

---

data:fi:ci:bnd:ti:ii:win:bl:rmsR

description: *win_bl_rmsR*
type: float

---

data:fi:ci:bnd:ti:ii:win:bl:rmsR_post

description: *win_bl_rmsR_post*
type: float

---

data:fi:ci:bnd:ti:ii:win:bl:rmsR_pre

description: *win_bl_rmsR_pre*
type: float

---

data:fi:ci:bnd:ti:ii:win:bl:sd_post

description: *win_bl_sd_post*
type: float

---

data:fi:ci:bnd:ti:ii:win:bl:sd_pre

description: *win_bl_sd_pre*
type: float

---

data:fi:ci:bnd:ti:ii:win:bl:sd_prepost

description: *win_bl_sd_prepost*
type: float

---

data:fi:ci:bnd:ti:ii:win:ml:aicI

description: *win_ml_aicI*
type: float

---

data:fi:ci:bnd:ti:ii:win:ml:aicI_post

description: *win_ml_aicI_post*
type: float

---

data:fi:ci:bnd:ti:ii:win:ml:aicI_pre

description: *win_ml_aicI_pre*
type: float

data:fi:ci:bnd:ti:ii:win:ml:corrD

description: *win_ml_corrD*
type: float

data:fi:ci:bnd:ti:ii:win:ml:corrD_post

description: *win_ml_corrD_post*
type: float

data:fi:ci:bnd:ti:ii:win:ml:corrD_pre

description: *win_ml_corrD_pre*
type: float

data:fi:ci:bnd:ti:ii:win:ml:d_m

description: *win_ml_d_m*
type: float

data:fi:ci:bnd:ti:ii:win:ml:d_o

description: *win_ml_d_o*
type: float

data:fi:ci:bnd:ti:ii:win:ml:r

description: *win_ml_r*
type: float

data:fi:ci:bnd:ti:ii:win:ml:rms

description: *win_ml_rms*
type: float

data:fi:ci:bnd:ti:ii:win:ml:rms_post

description: *win_ml_rms_post*
type: float

data:fi:ci:bnd:ti:ii:win:ml:rms_pre

description: *win_ml_rms_pre*
type: float

data:fi:ci:bnd:ti:ii:win:ml:rmsR

description: *win_ml_rmsR*
type: float

data:fi:ci:bnd:ti:ii:win:ml:rmsR_post

description: *win_ml_rmsR_post*
type: float

data:fi:ci:bnd:ti:ii:win:ml:rmsR_pre

description: *win_ml_rmsR_pre*
type: float

data:fi:ci:bnd:ti:ii:win:ml:sd_post

description: *win_ml_sd_post*
type: float

data:fi:ci:bnd:ti:ii:win:ml:sd_pre

description: *win_ml_sd_pre*
type: float

data:fi:ci:bnd:ti:ii:win:ml:sd_prepost

description: *win_ml_sd_prepost*
type: float

data:fi:ci:bnd:ti:ii:win:p

description: *p*
type: float

data:fi:ci:bnd:ti:ii:win:rng:aicI

description: *win_rng_aicI*
type: float

---

data:fi:ci:bnd:ti:ii:win:rng:aicI_post

description: *win_rng_aicI_post*
type: float

---

data:fi:ci:bnd:ti:ii:win:rng:aicI_pre

description: *win_rng_aicI_pre*
type: float

---

data:fi:ci:bnd:ti:ii:win:rng:corrD

description: *win_rng_corrD*
type: float

---

data:fi:ci:bnd:ti:ii:win:rng:corrD_post

description: *win_rng_corrD_post*
type: float

---

data:fi:ci:bnd:ti:ii:win:rng:corrD_pre

description: *win_rng_corrD_pre*
type: float

---

data:fi:ci:bnd:ti:ii:win:rng:d_m

description: *win_rng_d_m*
type: float

---

data:fi:ci:bnd:ti:ii:win:rng:d_o

description: *win_rng_d_o*
type: float

---

data:fi:ci:bnd:ti:ii:win:rng:r

description: *win_rng_r*
type: float

---

data:fi:ci:bnd:ti:ii:win:rng:rms

description: *win_rng_rms*
type: float

---

data:fi:ci:bnd:ti:ii:win:rng:rms_post

description: *win_rng_rms_post*
type: float

---

data:fi:ci:bnd:ti:ii:win:rng:rms_pre

description: *win_rng_rms_pre*
type: float

---

data:fi:ci:bnd:ti:ii:win:rng:rmsR

description: *win_rng_rmsR*
type: float

---

data:fi:ci:bnd:ti:ii:win:rng:rmsR_post

description: *win_rng_rmsR_post*
type: float

---

data:fi:ci:bnd:ti:ii:win:rng:rmsR_pre

description: *win_rng_rmsR_pre*
type: float

---

data:fi:ci:bnd:ti:ii:win:rng:sd_post

description: *win_rng_sd_post*
type: float

---

data:fi:ci:bnd:ti:ii:win:rng:sd_pre

description: *win_rng_sd_pre*
type: float

data:fi:ci:bnd:ti:ii:win:rng:sd_prepost

description: *win_rng_sd_prepost*
type: float

data:fi:ci:bnd:ti:ii:win:tl:aicI

description: *win_tl_aicI*
type: float

data:fi:ci:bnd:ti:ii:win:tl:aicI_post

description: *win_tl_aicI_post*
type: float

data:fi:ci:bnd:ti:ii:win:tl:aicI_pre

description: *win_tl_aicI_pre*
type: float

data:fi:ci:bnd:ti:ii:win:tl:corrD

description: *win_tl_corrD*
type: float

data:fi:ci:bnd:ti:ii:win:tl:corrD_post

description: *win_tl_corrD_post*
type: float

data:fi:ci:bnd:ti:ii:win:tl:corrD_pre

description: *win_tl_corrD_pre*
type: float

data:fi:ci:bnd:ti:ii:win:tl:d_m

description: *win_tl_d_m*
type: float

data:fi:ci:bnd:ti:ii:win:tl:d_o

description: *win_tl_d_o*
type: float

data:fi:ci:bnd:ti:ii:win:tl:r

description: *win_tl_r*
type: float

data:fi:ci:bnd:ti:ii:win:tl:rms

description: *win_tl_rms*
type: float

data:fi:ci:bnd:ti:ii:win:tl:rms_post

description: *win_tl_rms_post*
type: float

data:fi:ci:bnd:ti:ii:win:tl:rms_pre

description: *win_tl_rms_pre*
type: float

data:fi:ci:bnd:ti:ii:win:tl:rmsR

description: *win_tl_rmsR*
type: float

data:fi:ci:bnd:ti:ii:win:tl:rmsR_post

description: *win_tl_rmsR_post*
type: float

data:fi:ci:bnd:ti:ii:win:tl:rmsR_pre

description: *win_tl_rmsR_pre*
type: float

data:fi:ci:bnd:ti:ii:win:tl:sd_post

description: *win_tl_sd_post*
type: float

data:fi:ci:bnd:ti:ii:win:tl:sd_pre

description: *win_tl_sd_pre*
type: float

data:fi:ci:bnd:ti:ii:win:tl:sd_prepost

description: *win_tl_sd_prepost*
type: float

Chunks

data:fi:ci:chunk:ii:lab

description: label
type: string

---

data:fi:ci:chunk:ii:t

description: time start and end
type: 2 element list of floats

---

data:fi:ci:chunk:ii:to

description: t non-rounded
type: 2 element list of floats

F0

data:fi:ci:f0:bv

description: file/channel related f0 base value
type: float

---

data:fi:ci:f0:r

description: f0 residual after removal of the global f0 component
type: list of floats

---

data:fi:ci:f0:t

description: time stamps
type: list of floats

---

data:fi:ci:f0:y

description: f0 values after preprocessing
type: list of floats, same length as t

File information

data:fi:ci:fsys:annot:dir

description: directory of annotation file
type: string

---

data:fi:ci:fsys:annot:ext

description: extension of annotation file
type: string

---

data:fi:ci:fsys:annot:lab_chunk

description: general chunk label
type: string

---

data:fi:ci:fsys:annot:lab_pau

description: general pause label
type: string

---

data:fi:ci:fsys:annot:lab_syl

description: general syllable nucleus label
type: string

---

data:fi:ci:fsys:annot:stm

description: annotation file name stem
type: string

---

data:fi:ci:fsys:annot:typ

description: annotation file type (*xml* or *TextGrid*)
type: string

---

data:fi:ci:fsys:aud:dir

description: directory of audio file
type: string

---

data:fi:ci:fsys:aud:ext

description: extension of audio file
type: string

---

data:fi:ci:fsys:aud:stm

description: audio file name stem
type: string

---

data:fi:ci:fsys:aud:typ

description: audio file type
type: string

---

data:fi:ci:fsys:augment:channel:myTierName

description: channel number of each relevant tier *myTierName* in the annotation. Names of tiers derived by automatic chunking, phrasing, etc. will be added automatically.
type: int

---

data:fi:ci:fsys:augment:chunk:tier_out_stm

description: chunk tier output stem. In the augmented annotation file, the stem is concatenated with the respective channel number
type: string

---

data:fi:ci:fsys:augment:glob:tier

description: analysis tier name for prosodic boundaries, i.e. tier with prosodic boundary candidates. Max. 1 for each channel!
type: string

---

data:fi:ci:fsys:augment:glob:tier_out_stm

description: prosodic phrase tier output stem. In the augmented annotation file, the stem is concatenated with the respective channel number
type: string

---

data:fi:ci:fsys:augment:glob:tier_parent

description: name of the parent tier (e.g. chunks), whose boundaries limit the analysis and normalization window boundaries for prosodic phrase extraction
type: string

---

data:fi:ci:fsys:augment:lab_chunk

description: uniform chunk label
type: string

---

data:fi:ci:fsys:augment:lab_pau

description: uniform pause label
type: string

---

data:fi:ci:fsys:augment:lab_syl

description: uniform syllable nucleus label. Syllable boundaries are derived from this string by concatenating *_bnd*
type: string

---

data:fi:ci:fsys:augment:loc:tier_acc

description: analysis event tier name for pitch accent detection, i.e. the tier containing the time stamps of pitch accent candidates (e.g. syllable nuclei). Max 1 for each channel.
type: list of strings

---

data:fi:ci:fsys:augment:loc:tier_ag

description: analysis segment tier name for pitch accent detection, i.e. the tier containing segments within which maximally one pitch accent can be realized (e.g. words)
type: string

---

data:fi:ci:fsys:augment:loc:tier_out_stm

description: pitch accent tier output stem. In the augmented annotation file, the stem is concatenated with the respective channel number
type: string

---

data:fi:ci:fsys:augment:loc:tier_parent

description: name of the parent tier (e.g. prosodic phrases), relative to which the accent Gestalt is measured, and whose boundaries limit the analysis and normalization window boundaries for pitch accent extraction
type: string

---

data:fi:ci:fsys:augment:nc

description: number of channels
type: int

---

data:fi:ci:fsys:augment:stm

description: annotation file name stem
type: string

---

data:fi:ci:fsys:augment:syl:tier_out_stm

description: syllable nucleus and boundary tier output stem. In the augmented annotation file, for the syllable boundary tier *_bnd* is added to the stem, and for both nuclei and boundaries, the stem is concatenated with the respective channel number
type: string

---

data:fi:ci:fsys:augment:syl:tier_parent

description: name of the parent tier (e.g. chunks), within which reference values are calculated, and whose boundaries limit the analysis and normalization window boundaries for syllable nucleus detection)
type: string

---

data:fi:ci:fsys:bnd:tier

description: analysis tiers for boundary parameterization. Arbitrary number for each channel.
type: list of strings

---

data:fi:ci:fsys:chunk:tier

description: names of tiers that contain a chunk segmentation (only 1 tier for each channel). Names of automatically generated tiers are expanded by channel index `ci`.
type: list of strings

---

data:fi:ci:fsys:f0:dir

description: f0 file directory
type: string

---

data:fi:ci:fsys:f0:ext

description: f0 file extension
type: string

---

data:fi:ci:fsys:f0:stm

description: f0 file name stem
type: string

---

data:fi:ci:fsys:f0:typ

description: f0 file type
type: string

---

data:fi:ci:fsys:glob:tier

description: analysis tiers for global contour stylization. Max. 1 tier per channel.
type: list of strings

---

data:fi:ci:fsys:gnl_en:tier

description: names of analysis tiers for standard energy feature extraction. Any number of tiers per channel supported.
type: list of strings

---

data:fi:ci:fsys:gnl_f0:tier

description: names of analysis tiers for standard f0 feature extraction. Any number of tiers per channel supported.
type: list of strings

---

data:fi:ci:fsys:loc:tier_acc

description: time stamp analysis tiers for the 0-center of normalized time within a local contour segment. Max. 1 tier per channel.
type: list of strings

---

data:fi:ci:fsys:loc:tier_ag

description: segment tiers for local contours. Max. 1 tier per channel.
type: list of strings

data:fi:ci:fsys:rhy_en:tier

description: names of analysis tiers for energy rhythm feature extraction. Any number per channel.
type: list of strings

data:fi:ci:fsys:rhy_en:tier_rate

description: names of rate tiers for energy rhythm feature extraction. Any number per channel.
type: list of strings

data:fi:ci:fsys:rhy_f0:tier

description: names of analysis tiers for f0 rhythm feature extraction. Any number per channel.
type: list of strings

data:fi:ci:fsys:rhy_f0:tier_rate

description: names of rate tiers for f0 rhythm feature extraction. Any number per channel.
type: list of strings

Global segment features

data:fi:ci:glob:ii:class

description: *class*; global contour class index derived by clustering
type: int

data:fi:ci:glob:ii:decl:bl:c

description: *bl_c1, bl_c0*
type: 2 element list of floats

data:fi:ci:glob:ii:decl:bl:r

description: *bl_r*
type: float

data:fi:ci:glob:ii:decl:bl:rate

description: *bl_rate*
type: float

data:fi:ci:glob:ii:decl:bl:y

description: stylized f0 baseline values
type: list of floats

data:fi:ci:glob:ii:decl:err

description: True if base and topline crossing
type: boolean

data:fi:ci:glob:ii:decl:ml:c

description: *ml_c1, ml_c0*
type: 2 element list of floats

data:fi:ci:glob:ii:decl:ml:r

description: *ml_r*
type: float

data:fi:ci:glob:ii:decl:ml:rate

description: *ml_rate*
type: float

data:fi:ci:glob:ii:decl:ml:y

description: stylized f0 midline values
type: list of floats

data:fi:ci:glob:ii:decl:rng:c

description: *rng_c1, rng_c0*
type: 2 element list of floats

data:fi:ci:glob:ii:decl:rng:r

description: *rng_r*
type: float

---

data:fi:ci:glob:ii:decl:rng:rate

description: *rng_rate*
type: float

---

data:fi:ci:glob:ii:decl:rng:y

description: stylized f0 range values
type: list of floats

---

data:fi:ci:glob:ii:decl:tl:c

description: *tl_c1, tl_c0*
type: 2 element list of floats

---

data:fi:ci:glob:ii:decl:tl:r

description: *tl_r*
type: float

---

data:fi:ci:glob:ii:decl:tl:rate

description: *tl_rate*
type: float

---

data:fi:ci:glob:ii:decl:tl:y

description: stylized f0 topline values
type: list of floats

---

data:fi:ci:glob:ii:decl:tl_bl:y

description:
type: list of floats

---

data:fi:ci:glob:ii:decl:tn

description: normalized time values (same length as all *bl|ml|tl|rng:y*)
type: list of floats

---

data:fi:ci:glob:ii:eou:bl_drop

description: *bl_drop*
type: float

---

data:fi:ci:glob:ii:eou:ml_drop

description: *ml_drop*
type: float

---

data:fi:ci:glob:ii:eou:ml_bl_cross_f0

description: *ml_bl_cross_f0*
type: float

---

data:fi:ci:glob:ii:eou:ml_bl_cross_t

description: *ml_bl_cross_t*
type: float

---

data:fi:ci:glob:ii:eou:rng_drop

description: *rng_drop*
type: float

---

data:fi:ci:glob:ii:eou:tl_drop

description: *tl_drop*
type: float

---

data:fi:ci:glob:ii:eou:tl_bl_cross_f0

description: *tl_bl_cross_f0*
type: float

---

data:fi:ci:glob:ii:eou:tl_bl_cross_t

description: *tl_bl_cross_t*
type: float

data:fi:ci:glob:ii:eou:tl_ml_cross_f0

description: *tl_ml_cross_f0*
type: float

---

data:fi:ci:glob:ii:eou:tl_ml_cross_t

description: *tl_ml_cross_t*
type: float

---

data:fi:ci:glob:ii:gnl:dur

description: *dur*
type: float

---

data:fi:ci:glob:ii:gnl:iqr

description: *iqr*
type: float

---

data:fi:ci:glob:ii:gnl:m

description: *m*
type: float

---

data:fi:ci:glob:ii:gnl:max

description: *max*
type: float

---

data:fi:ci:glob:ii:gnl:med

description: *med*
type: float

---

data:fi:ci:glob:ii:gnl:min

description: *min*
type: float

---

data:fi:ci:glob:ii:gnl:sd

description: *sd*
type: float

---

data:fi:ci:glob:ii:lab

description: *lab*
type: string

---

data:fi:ci:glob:ii:ri

description: indices of local segments contained in global segment *ii*
type: list of int

---

data:fi:ci:glob:ii:t

description: global phrase time start and end
type: 2 element list of floats

---

data:fi:ci:glob:ii:tier

description: tier name
type: string

---

data:fi:ci:glob:ii:to

description: t non-rounded
type: 2 element list of floats

Standard energy features

data:fi:ci:gnl_en:ti:ii:lab

description: *lab*
type: string

---

data:fi:ci:gnl_en:ti:ii:std:dur

description: *dur*
type: float

---

data:fi:ci:gnl_en:ti:ii:std:dur_nrm

description: normalized duration
type: float

---

data:fi:ci:gnl_en:ti:ii:std:iqr

description: *iqr*
type: float

---

data:fi:ci:gnl_en:ti:ii:std:iqr_nrm

description: *iqr_nrm*
type: float

---

data:fi:ci:gnl_en:ti:ii:std:m

description: *m*
type: float

---

data:fi:ci:gnl_en:ti:ii:std:m_nrm

description: *m_nrm*
type: float

---

data:fi:ci:gnl_en:ti:ii:std:max

description: *max*
type: float

---

data:fi:ci:gnl_en:ti:ii:std:max_nrm

description: *max_nrm*
type: float

---

data:fi:ci:gnl_en:ti:ii:std:med

description: *med*
type: float

---

data:fi:ci:gnl_en:ti:ii:std:med_nrm

description: *med_nrm*
type: float

---

data:fi:ci:gnl_en:ti:ii:std:min

description: *min*
type: float

---

data:fi:ci:gnl_en:ti:ii:std:min_nrm

description: *min_nrm*
type: float

---

data:fi:ci:gnl_en:ti:ii:std:r_en_f0

description: *r_en_f0*
type: float

---

data:fi:ci:gnl_en:ti:ii:std:rms

description: *rms*
type: float

---

data:fi:ci:gnl_en:ti:ii:std:rms_nrm

description: *rms_nrm*
type: float

---

data:fi:ci:gnl_en:ti:ii:std:sb

description: *sb*
type: float

---

data:fi:ci:gnl_en:ti:ii:std:sd

description: *sd*
type: float

---

data:fi:ci:gnl_en:ti:ii:std:sd_nrm

description: *sd_nrm*
type: float

---

data:fi:ci:gnl_en:ti:ii:t

description: analysis window start and endpoint
type: 2 element list of floats

---

data:fi:ci:gnl_en:ti:ii:tier

description: tier name related to index *ti*
type: string

---

data:fi:ci:gnl_en:ti:ii:tn

description: normalization window start and endpoint
type: 2 element list of floats

---

data:fi:ci:gnl_en:ti:ii:to

description: t non-rounded
type: list of floats

---

data:fi:ci:gnl_en:ti:ii:tt

description: trend window (not used)
type: list of floats

---

data:fi:ci:gnl_en_file:dur

description: *dur*
type: float

---

data:fi:ci:gnl_en_file:iqr

description: *iqr*
type: float

---

data:fi:ci:gnl_en_file:m

description: *m*
type: float

---

data:fi:ci:gnl_en_file:max

description: *max*
type: float

---

data:fi:ci:gnl_en_file:med

description: *med*
type: float

---

data:fi:ci:gnl_en_file:min

description: *min*
type: float

---

data:fi:ci:gnl_en_file:r_en_f0

description: *r_en_f0*
type: float

---

data:fi:ci:gnl_en_file:sd

description: *sd*
type: float

Standard f0 features

data:fi:ci:gnl_f0:ti:ii:lab

description: *lab*
type: string

---

data:fi:ci:gnl_f0:ti:ii:std:dur

description: *dur*
type: float

---

data:fi:ci:gnl_f0:ti:ii:std:dur_nrm

description: normalized duration
type: float

---

data:fi:ci:gnl_f0:ti:ii:std:iqr

description: *iqr*
type: float

---

data:fi:ci:gnl_f0:ti:ii:std:iqr_nrm

description: *iqr_nrm*
type: float

---

data:fi:ci:gnl_f0:ti:ii:std:m

description: *m*
type: float

---

data:fi:ci:gnl_f0:ti:ii:std:m_nrm

description: *m_nrm*
type: float

---

data:fi:ci:gnl_f0:ti:ii:std:max

description: *max*
type: float

---

data:fi:ci:gnl_f0:ti:ii:std:max_nrm

description: *max_nrm*
type: float

---

data:fi:ci:gnl_f0:ti:ii:std:med

description: *med*
type: float

---

data:fi:ci:gnl_f0:ti:ii:std:med_nrm

description: *med_nrm*
type: float

---

data:fi:ci:gnl_f0:ti:ii:std:min

description: *min*
type: float

---

data:fi:ci:gnl_f0:ti:ii:std:min_nrm

description: *min_nrm*
type: float

---

data:fi:ci:gnl_f0:ti:ii:std:sd

description: *sd*
type: float

---

data:fi:ci:gnl_f0:ti:ii:std:sd_nrm

description: *sd_nrm*
type: float

---

data:fi:ci:gnl_f0:ti:ii:t

description: analysis window start and endpoint
type: 2 element list of floats

---

data:fi:ci:gnl_f0:ti:ii:tier

description: tier name related to index *ti*
type: string

---

data:fi:ci:gnl_f0:ti:ii:tn

description: normalization window start and endpoint
type: 2 element list of floats

---

data:fi:ci:gnl_f0:ti:ii:to

description: t non-rounded
type: list of floats

---

data:fi:ci:gnl_f0:ti:ii:tt

description: trend window, not used
type: list of floats

---

data:fi:ci:gnl_f0_file:dur

description: *dur*
type: float

---

data:fi:ci:gnl_f0_file:iqr

description: *iqr*
type: float

---

data:fi:ci:gnl_f0_file:m

description: *m*
type: float

---

data:fi:ci:gnl_f0_file:max

description: *max*
type: float

---

data:fi:ci:gnl_f0_file:med

description: *med*
type: float

---

data:fi:ci:gnl_f0_file:min

description: *min*
type: float

---

data:fi:ci:gnl_f0_file:sd

description: *sd*
type: float

Grouping

data:fi:ci:grp:myGroupVar*

description: myGroupvar refers to the grouping variable names specified in the configuration sub-dictionary `fsys:grp` for file name based grouping. The values are extracted from the file name and are always strings
type: string

Local segment features

data:fi:ci:loc:ii:acc:c

description: c^*; polynomial coefficients (highest order first)
type: list of floats

---

data:fi:ci:loc:ii:acc:tn

description: normalized time values
type: list of floats

---

data:fi:ci:loc:ii:acc:y

description: polynomial stylization values (same length as tn)
type: list of floats

---

data:fi:ci:loc:ii:class

description: *class*; local contour class index derived by clustering

type: int

---

data:fi:ci:loc:ii:decl:bl:c

description: *bl_c1, bl_c0*
type: 2 element list of floats

---

data:fi:ci:loc:ii:decl:bl:rate

description: *bl_rate*
type: float

---

data:fi:ci:loc:ii:decl:bl:y

description: stylized f0 baseline values
type: list of floats

---

data:fi:ci:loc:ii:decl:err

description: True if base- and topline cross
type: boolean

---

data:fi:ci:loc:ii:decl:ml:c

description: *ml_c1, ml_c0*
type: 2 element list of floats

---

data:fi:ci:loc:ii:decl:ml:rate

description: *ml_rate*
type: float

---

data:fi:ci:loc:ii:decl:ml:y

description: stylized f0 midline values
type: list of floats

---

data:fi:ci:loc:ii:decl:rng:c

description: *rng_c1, rng_c0*
type: 2 element list of floats

---

data:fi:ci:loc:ii:decl:rng:rate

description: *rng_rate*
type: float

---

data:fi:ci:loc:ii:decl:rng:y

description: stylized f0 range values
type: list of floats

---

data:fi:ci:loc:ii:decl:tl:c

description: *tl_c1, tl_c0*
type: 2 element list of floats

---

data:fi:ci:loc:ii:decl:tl:rate

description: *tl_rate*
type: float

---

data:fi:ci:loc:ii:decl:tl:y

description: stylized f0 topline values
type: list of floats

---

data:fi:ci:loc:ii:decl:tn

description: normalized time values
type: list of floats

---

data:fi:ci:loc:ii:gnl:dur

description: *dur*
type: float

---

data:fi:ci:loc:ii:gnl:dur_nrm

description: *dur_nrm*
type: float

---

data:fi:ci:loc:ii:gnl:iqr

description: *iqr*
type: float

---

data:fi:ci:loc:ii:gnl:iqr_nrm

description: *iqr_nrm*
type: float

---

data:fi:ci:loc:ii:gnl:m

description: *m*
type: float

---

data:fi:ci:loc:ii:gnl:m_nrm

description: *m_nrm*
type: float

---

data:fi:ci:loc:ii:gnl:max

description: *max*
type: float

---

data:fi:ci:loc:ii:gnl:max_nrm

description: *max_nrm*
type: float

---

data:fi:ci:loc:ii:gnl:med

description: *med*
type: float

---

data:fi:ci:loc:ii:gnl:med_nrm

description: *med_nrm*
type: float

---

data:fi:ci:loc:ii:gnl:min

description: *min*
type: float

---

data:fi:ci:loc:ii:gnl:min_nrm

description: *min_nrm*
type: float

---

data:fi:ci:loc:ii:gnl:sd

description: *sd*
type: float

---

data:fi:ci:loc:ii:gnl:sd_nrm

description: *sd_nrm*
type: float

---

data:fi:ci:loc:ii:gst:bl:d_fin

description: *bl_d_fin*
type: float

---

data:fi:ci:loc:ii:gst:bl:d_init

description: *bl_d_init*
type: float

---

data:fi:ci:loc:ii:gst:bl:rms

description: *bl_rms*
type: float

---

data:fi:ci:loc:ii:gst:bl:sd

description: *bl_sd*
type: float

---

data:fi:ci:loc:ii:gst:ml:d_fin

description: *ml_d_fin*
type: float

data:fi:ci:loc:ii:gst:ml:d_init

description: *ml_d_init*
type: float

data:fi:ci:loc:ii:gst:ml:rms

description: *ml_rms*
type: float

data:fi:ci:loc:ii:gst:ml:sd

description: *ml_sd*
type: float

data:fi:ci:loc:ii:gst:residual:bl:c

description: c^*; local contour coefs in descending order. Polynomial fitted on residual after baseline subtraction
type: list of floats

data:fi:ci:loc:ii:gst:residual:ml:c

description: c^*; local contour coefs in descending order. Polynomial fitted on residual after midline subtraction
type: list of floats

data:fi:ci:loc:ii:gst:residual:rng:c

description: c^*; local contour coefs in descending order. Polynomial fitted on residual after range normalization
type: list of floats

data:fi:ci:loc:ii:gst:residual:tl:c

description: c^*; local contour coefs in descending order. Polynomial fitted on residual after topline subtraction
type: list of floats

data:fi:ci:loc:ii:gst:rng:d_fin

description: *rng_d_fin*
type: float

data:fi:ci:loc:ii:gst:rng:d_init

description: *rng_d_init*
type: float

data:fi:ci:loc:ii:gst:rng:rms

description: *rng_rms*
type: float

data:fi:ci:loc:ii:gst:rng:sd

description: *rng_sd*
type: float

data:fi:ci:loc:ii:gst:tl:d_fin

description: *tl_d_fin*
type: float

data:fi:ci:loc:ii:gst:tl:d_init

description: *tl_d_init*
type: float

data:fi:ci:loc:ii:gst:tl:rms

description: *tl_rms*
type: float

data:fi:ci:loc:ii:gst:tl:sd

description: *tl_sd*
type: float

data:fi:ci:loc:ii:is_fin

description: *is_fin* (yes, no)
type: string

data:fi:ci:loc:ii:is_fin_chunk

description: *is_fin_chunk* (yes, no)
type: string

data:fi:ci:loc:ii:is_init

description: *is_init* (yes, no)
type: string

data:fi:ci:loc:ii:is_init_chunk

description: *is_init_chunk* (yes, no)
type: string

data:fi:ci:loc:ii:lab_acc

description: *lab_pnt*; label from local event tier
type: string

data:fi:ci:loc:ii:lab_ag

description: *lab*; label from local segment tier
type: string

data:fi:ci:loc:ii:ri

description: index of global parent segment
type: int

data:fi:ci:loc:ii:t

description: analysis window starting point, endpoint, and center. For only event tier input, these values are given by start and end of a symmetric window around each time stamp, and the time stamp itself. For only segment tier input, start- and endpoint are given by the segment's on and offset, and the center corresponds to the segment's midpoint. For both event and segment tier input, start- and endpoint are given by the segment's on and offset, and the center by the event's time stamp.
type: 3 element list of floats

data:fi:ci:glob:ii:tier_acc

description: point tier name for accent
type: string

data:fi:ci:glob:ii:tier_ag

description: segment tier name for accent group
type: string

data:fi:ci:loc:ii:tn

description: Normalization window start and endpoint to normalize f0 standard features.
type: 2 element list of floats

data:fi:ci:loc:ii:to

description: original input time values (1 for events, 2 for segments, 3 for events+segments)
type: 1, 2 or 3 element list of floats

Event rates

data:fi:ci:rate:myTierName*

description: for the events or segments of all tier names specified in the configuration by rhy_f0:tier_rate their overall rate is measured within the file.
type: float

Energy rhythm features

data:fi:ci:rhy_en:ti:ii:lab

description: *lab*
type: string

---

data:fi:ci:rhy_en:ti:ii:rate:myTierName*

description: *myRateTier_rate*; rate of items in *myTierName* within the current interval *ii* of tier with index *ti*
type: float

---

data:fi:ci:rhy_en:ti:ii:rhy:c

description: DCT coefficients in user defined frequency band
type: list of floats

---

data:fi:ci:rhy_en:ti:ii:rhy:c_orig

description: all DCT coefficients
type: list of floats

---

data:fi:ci:rhy_en:ti:ii:rhy:cbin

description: summed DCT coefs within frequency bins
type: list of floats

---

data:fi:ci:rhy_en:ti:ii:rhy:dur

description: *dur*
type: float

---

data:fi:ci:rhy_en:ti:ii:rhy:f

description: frequencies of DCT coefs in `c` (same length as `c`)
type: list of floats

---

data:fi:ci:rhy_en:ti:ii:rhy:f_orig

description: frequencies of DCT coef in `c_orig` (same length as `c_orig`)
type: list of floats

---

data:fi:ci:rhy_en:ti:ii:rhy:fbin

description: lower boundaries of frequency bins (same length as `cbin`)
type: list of floats

---

data:fi:ci:rhy_en:ti:ii:rhy:m

description: weighted coefficient mean
type: float

---

data:fi:ci:rhy_en:ti:ii:rhy:mae

description: mean absolute error between IDCT and original contour
type: float

---

data:fi:ci:rhy_en:ti:ii:rhy:sd

description: weighted coefficient standard deviation
type: float

---

data:fi:ci:rhy_en:ti:ii:rhy:sm

description: *sm**; spectral moments of DCT coefs
type: list of floats; length depends on config branch `styl:rhy_en:rhy:nsm`

---

data:fi:ci:rhy_en:ti:ii:rhy:wgt:myTierName*:mae

description: *myAnalysisTier_myRateTier_mae*; mean absolute error between original contour and IDCT of coefficients around the rate of the items in tier *myTierName*
type: float

---

data:fi:ci:rhy_en:ti:ii:rhy:wgt:myTierName*:prop

description: *myAnalysisTier_myAnalysisTier_prop*; proportion of the coefficient weights around the rate of the items in tier *myTierName* relative to coefficients' overall sum
type: float

---

data:fi:ci:rhy_en:ti:ii:rhy:wgt:myTierName*:rate

description: *myAnalysisTier_myAnalysisTier_rate*; rate of the items in tier *myTierName*

type: float

---

data:fi:ci:rhy_en:ti:ii:ri:myTierName*

description: item indices in tier *myTierName* that fall within the segment *ii* of tier with index *ti*
type: list of int

---

data:fi:ci:rhy_en:ti:ii:t

description: segment *ii* start and endpoint
type: 2 element list of floats

---

data:fi:ci:rhy_en:ti:ii:tier

description: tier name related to index *ti*
type: string

---

data:fi:ci:rhy_en:ti:ii:tn

description: as t, irrelevant
type: 2 element list of floats

---

data:fi:ci:rhy_en:ti:ii:to

description: t non-rounded
type: 2 element list of floats

---

data:fi:ci:rhy_en:ti:ii:tt

description: segment *ii* start, mid, and endpoint; irrelevant
type: 3 element list of floats

---

data:fi:ci:rhy_en_file:c

description: DCT coefficients in user defined frequency band
type: list of floats

---

data:fi:ci:rhy_en_file:c_orig

description: all DCT coefficients
type: list of floats

---

data:fi:ci:rhy_en_file:cbin

description: summed DCT coefs within frequency bins
type: list of floats

---

data:fi:ci:rhy_en_file:dur

description: *dur*
type: float

---

data:fi:ci:rhy_en_file:f

description: frequencies of DCT coefs in *c* (same length as *c*)
type: list of floats

---

data:fi:ci:rhy_en_file:f_orig

description: frequencies of DCT coef in *c_orig* (same length as *c_orig*)
type: list of floats

---

data:fi:ci:rhy_en_file:fbin

description: lower boundaries of frequency bins
type: list of floats

---

data:fi:ci:rhy_en_file:m

description: weighted coefficient mean
type: string

---

data:fi:ci:rhy_en_file:mae

description: mean absolute error between IDCT and original contour
type: float

---

data:fi:ci:rhy_en_file:sd

description: weighted coefficient standard deviation
type: float

---

data:fi:ci:rhy_en_file:sm

description: *sm**; spectral moments of DCT coefs
type: list of floats

---

data:fi:ci:rhy_en_file:wgt:myTierName*:mae

description: *myAnalysisTier_mae*; mean absolute error between original contour and IDCT of coefficients around the rate of the items in tier *myTierName*
type: float

---

data:fi:ci:rhy_en_file:wgt:myTierName*:prop

description: *myAnalysisTier_prop*; proportion of the coefficient weights around the rate of the items in tier *myTierName* relative to coefficients' overall sum
type: float

---

data:fi:ci:rhy_en_file:wgt:myTierName*:rate

description: *myAnalysisTier_rate*; rate of the items in tier *myTierName*
type: float

F0 rhythm features

data:fi:ci:rhy_f0:ti:ii:lab

description: *lab*
type: string

---

data:fi:ci:rhy_f0:ti:ii:rate:myTierName*

description: *myAnalysisTier_rate*; rate of items in *myTierName* within the current interval *ii* of tier with index *ti*
type: float

---

data:fi:ci:rhy_f0:ti:ii:rhy:c

description: DCT coefficients in user defined frequency band
type: list of floats

---

data:fi:ci:rhy_en:ti:ii:rhy:c_orig

description: all DCT coefficients
type: list of floats

---

data:fi:ci:rhy_f0:ti:ii:rhy:cbin

description: summed DCT coefs within frequency bins
type: list of floats

---

data:fi:ci:rhy_f0:ti:ii:rhy:dur

description: *dur*
type: float

---

data:fi:ci:rhy_f0:ti:ii:rhy:f

description: frequencies of DCT coefs in *c* (same length as *c*)
type: list of floats

---

data:fi:ci:rhy_f0:ti:ii:rhy:f_orig

description: frequencies of DCT coef in *c_orig* (same length as *c_orig*)
type: list of floats

---

data:fi:ci:rhy_f0:ti:ii:rhy:fbin

description: lower boundaries of frequency bins
type: list of floats

---

data:fi:ci:rhy_f0:ti:ii:rhy:m

description: weighted coefficient mean
type: string

---

data:fi:ci:rhy_f0:ti:ii:rhy:mae

description: mean absolute error between IDCT and original contour
type: float

---

data:fi:ci:rhy_f0:ti:ii:rhy:sd

description: weighted coefficient standard deviation
type: float

---

data:fi:ci:rhy_f0:ti:ii:rhy:sm

description: sm^*; spectral moments of DCT coefs
type: list of floats

---

data:fi:ci:rhy_f0:ti:ii:rhy:wgt:myTierName*:mae

description: *myAnalysisTier_myAnalysisTier_mae*; mean absolute error between original contour and IDCT of coefficients around the rate of the items in tier *myTierName*
type: float

---

data:fi:ci:rhy_f0:ti:ii:rhy:wgt:myTierName*:prop

description: *myAnalysisTier_myAnalysisTier_prop*; proportion of the coefficient weights around the rate of the items in tier *myTierName* relative to coefficients' overall sum
type: float

---

data:fi:ci:rhy_f0:ti:ii:rhy:wgt:myTierName*:rate

description: *myAnalysisTier_myAnalysisTier_rate*; rate of the items in tier *myTierName*
type: float

---

data:fi:ci:rhy_f0:ti:ii:ri:myTierName

description: item indices in tier *myTierName* that fall within the segment *ii* of tier with index *ti*
type: list of int

---

data:fi:ci:rhy_f0:ti:ii:t

description: segment *ii* start and endpoint
type: 2 element list of floats

---

data:fi:ci:rhy_f0:ti:ii:tier

description: tier name related to index *ti*
type: string

---

data:fi:ci:rhy_f0:ti:ii:tn

description: as t, irrelevant
type: 2-element list of floats

---

data:fi:ci:rhy_f0:ti:ii:to

description: t non-rounded
type: list of floats

---

data:fi:ci:rhy_f0:ti:ii:tt

description: segment *ii* start, mid, and endpoint; irrelevant
type: list of floats

---

data:fi:ci:rhy_f0_file:c

description: DCT coefficients in user defined frequency band
type: list of floats

---

data:fi:ci:rhy_f0_file:c_orig

description: all DCT coefficients
type: list of floats

---

data:fi:ci:rhy_f0_file:cbin

description: summed DCT coefs within frequency bins
type: list of floats

---

data:fi:ci:rhy_f0_file:dur

description: *dur*
type: float

---

data:fi:ci:rhy_f0_file:f

description: frequencies of DCT coefs in *c* (same length as *c*)
type: list of floats

---

data:fi:ci:rhy_f0_file:f_orig

description: frequencies of DCT coef in *c_orig* (same length as *c_orig*)
type: list of floats

---

data:fi:ci:rhy_f0_file:fbin

description: lower boundaries of frequency bins
type: list of floats

---

data:fi:ci:rhy_f0_file:m

description: weighted coefficient mean
type: string

---

data:fi:ci:rhy_f0_file:mae

description: mean absolute error between IDCT and original contour
type: float

---

data:fi:ci:rhy_f0_file:sd

description: weighted coefficient standard deviation
type: float

---

data:fi:ci:rhy_f0_file:sm

description: *sm**; spectral moments of DCT coefs
type: list of floats

---

data:fi:ci:rhy_f0_file:wgt:myTierName*:mae

description: *myAnalysisTier_mae*; mean absolute error between original contour and IDCT of coefficients around the rate of the items in tier *myTierName*
type: float

---

data:fi:ci:rhy_f0_file:wgt:myTierName*:prop

description: *myAnalysisTier_prop*; proportion of the coefficient weights around the rate of the items in tier *myTierName* relative to coefficients' over all sum
type: float

---

data:fi:ci:rhy_f0_file:wgt:myTierName*:rate

description: *myAnalysisTier_rate*; rate of the items in tier *myTierName*
type: float

Voice quality features

data:fi:ci:voice:ti:ii:jit:c

description: 3rd order polycoefs for jitter contour
type: list of floats

---

data:fi:ci:voice:ti:ii:jit:m

description: mean period length (in sec)
type: float

---

data:fi:ci:voice:ti:ii:jit:m_nrm

description: normalized mean period length
type: float

---

data:fi:ci:voice:ti:ii:jit:sd

description: period length sd
type: float

---

data:fi:ci:voice:ti:ii:jit:sd_nrm

description: normalized period length sd
type: float

---

data:fi:ci:voice:ti:ii:jit:v

description: jitter (relative)
type: float

---

data:fi:ci:voice:ti:ii:jit:v_abs

description: jitter (absolute)
type: float

data:fi:ci:voice:ti:ii:jit:v_nrm

description: normalized jitter (relative)
type: float

data:fi:ci:voice_file:jit:c

description: 3rd order polycoefs for jitter contour
type: list of floats

data:fi:ci:voice_file:jit:m

description: mean period length (in sec)
type: float

data:fi:ci:voice_file:jit:sd

description: period length sd
type: float

data:fi:ci:voice_file:jit:v

description: jitter (relative)
type: float

data:fi:ci:voice_file:jit:v_abs

description: jitter (absolute)
type: float

data:fi:ci:voice:ti:ii:shim:c

description: 3rd order polycoefs for shimmer contour
type: list of floats

data:fi:ci:voice:ti:ii:shim:m

description: mean amplitude
type: float

data:fi:ci:voice:ti:ii:shim:m_nrm

description: normalized mean amplitude
type: float

data:fi:ci:voice:ti:ii:shim:sd

description: amplitude sd
type: float

data:fi:ci:voice:ti:ii:shim:sd_nrm

description: normalized amplitude sd
type: float

data:fi:ci:voice:ti:ii:shim:v

description: shimmer
type: float

data:fi:ci:voice:ti:ii:shim:v_nrm

description: normalized shimmer
type: float

data:fi:ci:voice:ti:ii:shim_file:c

description: 3rd order polycoefs for shimmer contour
type: list of floats

data:fi:ci:voice:shim_file:m

description: mean amplitude
type: float

data:fi:ci:voice:shim_file:sd

description: amplitude sd
type: float

data:fi:ci:voice:shim_file:v

description: shimmer
type: float

12.3.4 Clustering result sub-dictionary

This sub-dictionary is accessed by `copa['clst']` and contains the outcome of the clustering of global and local contours (cf. sections 8.5.3 and 8.7.3).

Global contour classes

clst:glob:c

description: slope coef matrix to be clustered
type: 2d array of floats

---

clst:glob:cntr

description: centroid matrix, one row per contour class
type: 2d array of floats

---

clst:glob:ij

description: location of each feature vector in `copa:data:fi:ci:glob:ii`: each row contains 3 indices for the file *fi*, the channel *ci*, and the global segment *ii*, respectively)
type: 2d array of int

---

clst:glob:obj

description: clustering object which was used for global contour clustering
type: object

---

clst:glob:val

description: mean silhouette
type: float

Local contour classes

clst:loc:c

description: polynomial coef matrix to be clustered
type: list of floats

---

clst:loc:cntr

description: centroid matrix, one row per contour class
type: list of strings

---

clst:loc:ij

description: location of each feature vector in `copa:data:fi:ci:loc:ii`: each row contains 3 indices for the file, the channel, and the local segment, respectively
type: list of floats

---

clst:loc:obj

description: clustering object which was used for local contour clustering
type: string

---

clst:loc:val

description: mean silhouette
type: float

12.3.5 Validation sub-dictionary

This sub-dictionary is accessed by `copa['val']` and contains validation measures for stylization and clustering.

Clustering

val:clst:glob:sil_mean

description: mean silhouette of all data points for global contour clustering
type: float

---

val:clst:loc:sil_mean

description: mean silhouette of all data points for local contour clustering
type: float

Stylization

val:styl:glob:err_prop

description: proportion of base/topline crossings over all global segments
type: float

---

val:styl:loc:rms_mean

description: mean RMSD between original and stylized local contour
type: float

12.4 F0 files

Three types of f0 tables can be exported:

- preprocessed f0
- residual f0 (after removal of the global register component)
- resynthesized f0 (superposition of global and local stylized component)

As the f0 table input format in each output table the first column gives the time stamps, and the second till last columns contain the f0 values (in Hz) for the recording channels. The tables will be stored below `fsys:export:dir` in sub-directories named after the type of f0 output (*f0_preproc, f0_residual, f0_resyn*). For each input f0 file an output file with the same name is generated.

12.5 Log file

The log file in *fsys:export:dir + fsys:export:stm + log.txt* contains warnings, information about too short segments to be skipped, and some validations below the line '# validation':

styl.glob.err_prop	the percentage of global contour segments where base and topline are crossing
styl.loc.rms_mean	the mean RMSD between original and stylized contour over all local contour segments

The log file is not overwritten, but new logging information is appended. Each session starts with the current time string in ISO 8601 format.

13 Plotting

To activate plotting, set

```
navigate:do_plot=1
```

Browsing Browsing through stylizations can be carried out online (in order to check for appropriate stylization parameter settings) or after feature extraction, which is controlled by

```
plot:browse:time
```

To select the stylization to be plotted the corresponding branches in

```
plot:browse:typ:*:*
```

need to be set to 1. E.g. `plot:browse:typ:complex:superpos=1` produces plots as in Figure 5.
It is possible to plot stylizations of segments with certain grouping values only. This is achieved by specifying one or more

```
plot:browse:grp:myGroupingVariable:myGroupingValue
```

myGroupingVariable should match one of the strings in `fsys:grp:lab`. The string *myGroupingValue* should match one of the values assigned to *myGroupingVariable* by file name parsing (cf section 11.3).

Grouping One can also plot stylizations based on parameter centroids for a specified grouping. By

```
plot:grp:typ:*:*=1
```

the user selects the stylization to be plotted. The grouping is defined by

```
plot:grp:grouping
```

The entries in this list can be *lab* for item labels or the grouping factor names specified in `fsys:grp:lab`. Centroids will be plotted for each factor level combination.

Browsing and grouping plots can be saved as .png files by

```
plot:browse:save=1
plot:grp:save=1
```

The *browse* mode output file names are the concatenation of `fsys:pic:dir` + `fsys:pic:stm` + *final*|*online* + *typ* + *set* + *fileIndex* + *channelIndex* + *tierName* + *itemIndex*. *typ* and *set* refer to the *-keys in plot:browse:typ:*:* set to 1.

The *grouping* mode output file names are concatenated from `fsys:pic:dir` + `fsys:pic:stm` + *factorLevelCombination*. One file is generated for each factor level combination.

14 Known bugs

Not yet all missing or wrong configurations will be reported in the log file but might result in some Python error message. Same with unexpected annotations. If you cannot locate the error, you can send the configuration file, the log file, and the error message to: uwe.reichel@nytud.mta.hu

An error message that includes the line *return json.load(h)* results from a wrongly formatted JSON configuration file. The last line of the error message points you to the erroneous line in the JSON file.

Currently (September 24th, 2018), depending on your *scipy* and *numpy* versions, scipy.signal may cause a *FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated.* This warning can be ignored.

References

[1] Belz, M. and U. Reichel: *Pitch characteristics of filled pauses in spontaneous speech*. In *DiSS 2015*, Edinburgh, Scotland, 2015. https://www.phonetik.uni-muenchen.de/~reichelu/publications/BelzReichelDiss2015.pdf.

[2] Beňuš, Š., U. Reichel and K. Mády: *Modelling accentual phrase intonation in Slovak and Hungarian*. In *Complex Visibles Out There*, vol. 4, pp. 677–689. Palacký University, Olomouc, Czech Republic, 2014. http://www.phonetik.uni-muenchen.de/~reichelu/publications/BenusReichelMadyOlinco2014.pdf.

[3] Beňuš, Š., U. Reichel and J. Šimko: *F0 discontinuity as a marker of prosodic boundary strength in Lombard speech*. In *Proc. Interspeech*, p. paper 953, Dresden, Germany, 2015. https://www.phonetik.uni-muenchen.de/~reichelu/publications/BenusReichelSimkoIS2015.pdf.

[4] Boersma, P. and D. Weenink: *PRAAT, a system for doing phonetics by computer*. Techn. Rep., Institute of Phonetic Sciences of the University of Amsterdam, 1999. 132–182.

[5] Fant, G., A. Kruckenberg, J. Liljecrants and S. Hertegard: *Acoustic-phonetic studies of prominence in Swedish*. TMH-QPSR, 41(2–3):1–52, 2000.

[6] Fuchs, S. and U. Reichel: *On the relation between pointing gestures and speech production in German counting out rhymes: Evidence from motion capture data and speech acoustics*. In *Proc. P&P*, Munich, Germany, 2016. http://www.phonetik.uni-muenchen.de/~reichelu/publications/fuchsReichelPandP2016.pdf.

[7] Heinrich, C. and F. Schiel: *The influence of alcoholic intoxication on the short-time energy function of speech*. J. Acoust. Soc. Am., 135(5):2942–2951, 2014.

[8] Kalkhoff, A.: *Corpus data and tools for the analysis of spoken Haitian Creole prosody*. Poster, 2015. https://methodenromanistentag2015.files.wordpress.com/2015/10/kalkhoff.pdf.

[9] Kisler, T., U. Reichel, F. Schiel, C. Draxler, B. Jackl and N. Pörner: *BAS Speech Science Web Services - an update of current developments*. In *Proc. LREC*, pp. 3880–3885, Portorož, Slovenia, 2016. https://www.phonetik.uni-muenchen.de/~reichelu/publications/KRSDJP_LREC2016.pdf.

[10] Mády, K. and U. Reichel: *How to distinguish between self– and other-directed wh-questions?*. In *Proc. P&P*, Munich, Germany, 2016. http://www.phonetik.uni-muenchen.de/~reichelu/publications/madyReichelPuP_final.pdf.

[11] Mády, K., U. Reichel, A. Szalontai, A. Kohári and A. Deme: *Prosodic characteristics of infant-directed speech as a function of multiple pregnancy*. In *Proc. Speech Prosody*, pp. 294–298, Poznan, Poland, 2018. https://www.phonetik.uni-muenchen.de/~reichelu/publications/MRSzKD_SP18.pdf.

[12] Mittelhammer, K. and U. Reichel: *Characterization and prediction of dialogue acts using prosodic features*. In Jokisch, O. (ed.): *Elektronische Sprachverarbeitung 2016*, vol. 81 of *Studientexte zur Sprachkommunikation*, pp. 160–167. TUDpress, Dresden, Germany, 2016. https://www.phonetik.uni-muenchen.de/~reichelu/publications/MR_ESSV2016.pdf.

[13] Pfitzinger, H., S. Burger and S. Heid: *Syllable Detection in Read and Spontaneous Speech*. In *Proc. ICSLP*, vol. 2, pp. 1261–1264, Philadelphia, 1996.

[14] Reichel, U.: *Linking bottom-up intonation stylization to discourse structure*. Computer, Speech, and Language, 28:1340–1365, 2014. http://www.phonetik.uni-muenchen.de/~reichelu/publications/ReichelCSLAccepted.pdf.

[15] Reichel, U.: *Personality prediction based on intonation stylization*. In *Proc. ICPhS*, p. paper 616, Glasgow, Scotland, 2015. https://www.phonetik.uni-muenchen.de/~reichelu/publications/reichelIcphs2015.pdf.

[16] Reichel, U.: *Unsupervised extraction of prosodic structure*. In *Elektronische Sprachsignalverarbeitung 2017*, vol. 86 of *Studientexte zur Sprachkommunikation*, pp. 262–269. TUDPress, Dresden, Germany, 2017. http://www.phonetik.uni-muenchen.de/~reichelu/publications/reichelESSV2017.pdf.

[17] Reichel, U., Š. Beňuš and K. Mády: *Entrainment profiles: Comparison by gender, role, and feature set*. Speech Communication, 100:46–57, 2018. https://doi.org/10.1016/j.specom.2018.04.009.

[18] Reichel, U. and J. Cole: *Entrainment analysis of categorical intonation representations*. In *Proc. P&P*, Munich, Germany, 2016. http://www.phonetik.uni-muenchen.de/~reichelu/publications/reichelColePuP.pdf.

[19] Reichel, U. and P. Lendvai: *Veracity computing from lexical cues and perceived certainty trends*. In *Proc. 2nd Workshop on Noisy User-generated Text*, Osaka, Japan, 2016. http://www.phonetik.uni-muenchen.de/~reichelu/publications/RL_Wnut16.pdf.

[20] Reichel, U. and P. Lendvai: *Dodging the question in competitive spoken dialogs: Semantic and prosodic characteristics*. In Berton, A., U. Haiber and W. Minker (eds.): *Elektronische Sprachverarbeitung 2018*, vol. 90 of *Studientexte zur Sprachkommunikation*, pp. 263–270. TUDpress, Dresden, Germany, 2018. https://www.phonetik.uni-muenchen.de/~reichelu/publications/RL_essv2018.pdf.

[21] Reichel, U. and K. Mády: *Parameterization of F0 register and discontinuity to predict prosodic boundary strength in Hungarian spontaneous speech*. In Wagner, P. (ed.): *Elektronische Sprachsignalverarbeitung 2013*, vol. 65 of *Studientexte zur Sprachkommunikation*, pp. 223–230. TUDpress, Dresden, Germany, 2013. http://www.phonetik.uni-muenchen.de/~reichelu/publications/ReichelMadyESSV2013.pdf.

[22] Reichel, U. and K. Mády: *Comparing parameterizations of pitch register and its discontinuities at prosodic boundaries for Hungarian*. In *Proc. Interspeech 2014*, pp. 111–115, Singapore, 2014. http://www.phonetik.uni-muenchen.de/~reichelu/publications/ReichelMadyIS2014.pdf.

[23] Reichel, U., K. Mády and Š. Beňuš: *Parameterization of prosodic headedness*. In *Proc. Interspeech*, p. paper 929, Dresden, Germany, 2015. https://www.phonetik.uni-muenchen.de/~reichelu/publications/RMB_IS15.pdf.

[24] Reichel, U., K. Mády and Š. Beňuš: *Acoustic profiles for prosodic headedness and constituency*. In *Proc. Speech Prosody*, pp. 699–703, Poznan, Poland, 2018. https://www.phonetik.uni-muenchen.de/~reichelu/publications/RMB_SP18.pdf.

[25] Reichel, U., K. Mády and F. Kleber: *How prominence and prosodic phrasing interact*. In Jokisch, O. (ed.): *Elektronische Sprachverarbeitung 2016*, vol. 81 of *Studientexte zur Sprachkommunikation*, pp. 153–159. TUDpress, Dresden, Germany, 2016. https://www.phonetik.uni-muenchen.de/~reichelu/publications/RMK_ESSV2016.pdf.

[26] Schiel, F.: *Automatic Phonetic Transcription of Non-Prompted Speech*. In *Proc. ICPhS*, pp. 607–610, San Francisco, 1999.