VokalJäger Enhanced Algorithmic Tool Pack – `VJ.EAT`

Documentation

Author: Carsten Keil
http://vokaljaeger.org/ - ckeil@vokaljaeger.org
Version: 0.15 as of 7/5/2021

A. Introduction 2
- A.1. Synopsis 2
- A.2. VokalJäger VJ.EAT.core and VJ.EAT.demo license (algorithms and data) 2
- A.3. VokalJäger VJ.EAT.demo sound file license 3
- A.4. Attribution 3

B. Install 4
- B.1. Introductionary notes 4
- B.2. Installation of the core algorithms (`VJ.EAT.core`) 4
- B.3. Setting Microsoft Windows `PATH` variable 4
- B.4. Install PRAAT 5
- B.5. Install R 5

C. Create a working environment and test it with `VJ.EAT.demo` 8
- C.1. Create a working environment 8
- C.2. Demonstration: end-to-end formant analysis 8

D. Methodology enhancements beyond the original VokalJäger 9
- D.1. Artificial intelligence driven sweep 9

E. Code and data documentation 10
- E.1. Directory structure of `VJ.EAT.core` 10
- E.2. Directory structure of `VJ.EAT.demo` 10
- E.3. PRAAT program `VJcalcFormants` 11
- E.4. R program VJsweepFormants 14
- E.5. R program VJnormalizeFormants 18
- E.6. R program VJanalyzeFormants 21
- E.7. Data dictionary 26

F. Change Log 29
- F.1. Version 0.12 29

G. References 29

A. Introduction

A.1. Synopsis

The `VJ.EAT` VokalJäger Enhanced Algorithmic Tool Box re-implements the PRAAT and R kernel algorithms *from Keil 2017: Der VokalJäger: Eine phonetisch-algorithmische Methode zur Vokaluntersuchung. Exemplarisch angewendet auf historische Tondokumente der Frankfurter Stadtmundart, Deutsche Dialektgeographie Vol. 122*. The core idea of `VJ.EAT` is to offer one-stop solution for a robust a fully automated formant measurement and phonetic classification, here packaged and tested for Microsoft Windows 10. The `VJ.EAT` re-implementation is by a factor of 10 more performant than the original version, more flexible and robust plus rearranged for modular use.

The 0.x series of the algorithms is a pre-release version. The final version will be published on CRAN.

In particular, `VJ.EAT` offers the following **algorithmic components**:

A.1.1. Using PRAAT, it calculates for a given sound file and a text grid, which defines the samples resp. time ranges to go for, intra-sample formant trajectories. That is done for a series of upper formant ceilings (PRAAT program `VJcalcFormants`, see E.3).

A.1.2. Using R, it performs the so called “sweep”. Here the formant trajectories of the preceding step are smoothed with a DTT and formant values are extracted from a specific point on the curve. Out of all different trajectory-bundles for each sample – one for each upper formant ceiling – the one is chosen, which is optimal under a certain heuristic (R program `VJsweepFormants`, see E.4).

A.1.3. Using R, it performs a formant normalization. Here the extracted formant values of the preceding step are normalized using the (optionally: robust) Lobanov or Gerstman procedures (R program `VJnormalizeFormants`, see E.5).

A.1.4. Using R, it classifies the phonems (R program `VJclassifyFormants`, see xx, not implemented yet).

A.1.5. Using R, it creates a series of statistical analysis and formant plots. (R program `VJanalyzeFormants`, see E.6).

A.2. VokalJäger VJ.EAT.core and VJ.EAT.demo license (algorithms and data)

The algorithms and data in `VJ.EAT.core` and `VJ.EAT.demo` are published under the license “Creative Commons Attribution 4.0 International CC BY 4.0” (with exception of the sound files: see A.3). You are free to:

Share — copy and redistribute the material in any medium or format

Adapt — remix, transform, and build upon the material

for any purpose, even commercially (This license is acceptable for Free Cultural Works. The licensor cannot revoke these freedoms as long as you follow the license terms). Under the following terms:

Attribution — You must give appropriate credit (see A.4), provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Notices: You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation. No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material.

Full CC 4.0 BY license text here: https://creativecommons.org/licenses/by/4.0/

A.3. VokalJäger VJ.EAT.demo sound file license

MP3 recordings by LibriVox (https://librivox.org/), hosted on Archive.Org (https://archive.org/details/sammlung_deutscher_gedichte_018_1506_librivox), published under "Public Domain Mark 1.0" license (https://creativecommons.org/publicdomain/mark/1.0/), means: "No Copyright. This work has been identified as being free of known restrictions under copyright law, including all related and neighboring rights. You can copy, modify, distribute and perform the work, even for commercial purposes, all without asking permission."

"Sammlung deutscher Gedichte 018" [published: 05/06/2015]

Karlson-Heine-Ritter.mp3 is:

https://archive.org/details/sammlung_deutscher_gedichte_018_1506_librivox/deutschegedichte018_20_ritter_ksn_128kb.mp3

Tabea-George-Vogelschau.mp3 is:

https://archive.org/details/sammlung_deutscher_gedichte_018_1506_librivox/deutschegedichte018_17_vogelschau_tab_128kb.mp3

Verlaine-Zwiegespraech.mp3 is:

https://archive.org/details/sammlung_deutscher_gedichte_018_1506_librivox/deutschegedichte018_19_zwiegespraech_hok_128kb.mp3

A.4. Attribution

If you use VJ.EAT you must give the following attribution (in your citation style):

Keil 2017: Der VokalJäger: Eine phonetisch-algorithmische Methode zur Vokaluntersuchung. Exemplarisch angewendet auf historische Tondokumente der Frankfurter Stadtmundart, Deutsche Dialektgeographie Vol. 122.

and

Keil 2020: VJ.EAT: VokalJäger - Enhanced Algorithmic Tool pack. Version [insert version number here]. http://vokaljaeger.org. Published under CC 4.0 BY (https://creativecommons.org/licenses/by/4.0).

In case you re-distribute the sound files you must give an attribution to the original source and license (see A.3).

B. Install

B.1. Introductory notes

B.1.1. This all works on only on Microsoft Windows.

B.1.2. Know basics of the windows command shell [1]. To open one, the default way is: Windows start > start typing “cmd” > Press <ENTER>.

B.1.3. You need a plain text editor (i.e. NOT Microsoft word). Windows built-in notepad application will do, but Notepad++ from https://notepad-plus-plus.org/downloads/ is much better.

B.1.4. You can edit the text files with Microsoft EXCEL as well, but make sure you save them as tab separated text file just you opened them, don’t save as XLSX.

B.1.5. All tags and preferably all other text elements (phoneme codes, annotations, file names etc.) should restrict to classical old school 7 bit ASCII letters - from 33 (’!’) to 126 (‘~’). Spaces (‘ ‘) should be avoided and replaced with underscores (‘_’). Umlaute, enhanced character sets like UTF-16 etc. should be avoided. That ensures proper processing, esp. where tags are automatically used as components of keys, filenames etc.

B.1.6. Recent key content changes in most recent version updates are highlighted.

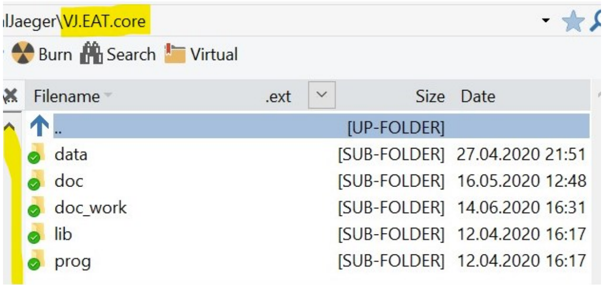
B.2. Installation of the core algorithms (`VJ.EAT.core`)

B.2.1. Download the latest `VJ.EAT.core` zip-packaged release (i.e. that with the highest version number) from here:

http://vokaljaeger.org/download.

Unpack it to where you like it (e.g. to your desktop).

B.2.2. You now have the core algorithms in a folder `VJ.EAT.core` with a structure described in E.1:



lJaeger\VJ.EAT.core

Filename	.ext	Size	Date
..		[UP-FOLDER]	
data		[SUB-FOLDER]	27.04.2020 21:51
doc		[SUB-FOLDER]	16.05.2020 12:48
doc_work		[SUB-FOLDER]	14.06.2020 16:31
lib		[SUB-FOLDER]	12.04.2020 16:17
prog		[SUB-FOLDER]	12.04.2020 16:17

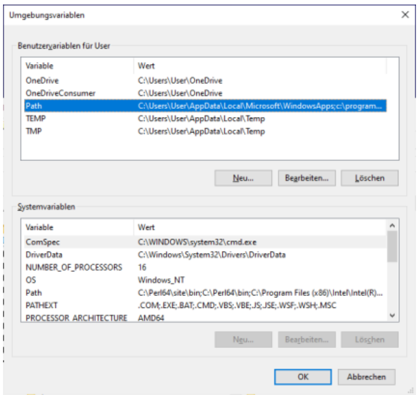
B.2.3. Next you should set up – an in case: test – a working environment to hold your sound files etc. (see C.1).

B.3. Setting Microsoft Windows `PATH` variable

B.3.1. You need to get your windows `PATH` variables right, which allow windows to find the programs of concern, most notably the executable `praaet.exe` and `rscript.exe`, on your computer. To that end you must tell Window to add the directory location of the executables, like `c:\Program Files\PRAAT` for PRAAT to the Windows `PATH` variable.

B.3.2. Go to Windows start, start typing “Umgebungsvariablen für dieses Konto bearbeiten”, then open the associated program.

B.3.3. A window like below listing the “Umgebungsvariablen” for the current user should pop up. It is important that this is actually the user which wants to run VJ, not the administrator.



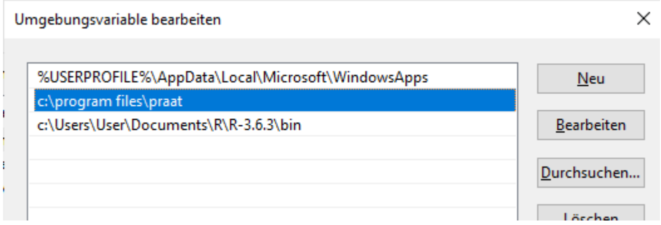
Benutzervariablen für User

Variable	Wert
OneDrive	C:\Users\User\OneDrive
OneDriveConsumer	C:\Users\User\OneDrive
Path	C:\Users\User\AppData\Local\Microsoft\WindowsApps;c:\program...
TEMP	C:\Users\User\AppData\Local\Temp
TMP	C:\Users\User\AppData\Local\Temp

Systemvariablen

Variable	Wert
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	16
OS	Windows_NT
Path	C:\Perl64\site\bin;C:\Perl64\bin;C:\Program Files (x86)\Intel\Intel(R)...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64

B.3.4. Select “Path”, edit (“bearbeiten”), and then create (“neu”) or change (“bearbeiten”) directory entries relating to the programs PRAAT and RScript. You obviously need to know where the executables are stored, see sections B.4 and B.5 for more program specific info. A proper set up could look like this:

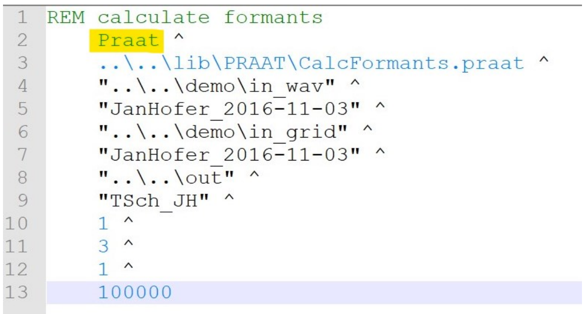


```
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
c:\program files\praat
c:\Users\User\Documents\R\R-3.6.3\bin
```

B.3.5. Then “ok” it from the 1st screen (“Umgebungsvariable bearbeiten”), then from root screen (“Umgebungsvariablen”).

B.3.6. Reboot your computer.

B.3.7. HACK: If assignment of the `PATH` variable fails you can always exchange the call to the executable of concern, say `PRAAT` with the full path call, say “`c:\Program Files\praat.exe`” instead of the simple call `praat` (note, you have to put the full-path call in parenthesis *and* you have to do that in *all* batch scripts you intend to use, which is a bit of a pain (and you have to do it again, resp. check, in case you update or move your install). So you would open you batch script with an editor and exchange the direct call (below: PRAAT) with the full path call.



```
REM calculate formants
    Praat ^
    ..\..\lib\PRAAT\CalcFormants.praat ^
    "..\..\demo\in_wav" ^
    "JanHofer_2016-11-03" ^
    "..\..\demo\in_grid" ^
    "JanHofer_2016-11-03" ^
    "..\..\out" ^
    "TSch_JH" ^
    1 ^
    3 ^
    1 ^
    100000
```

B.4. Install PRAAT

B.4.1. Install PRAAT from here: http://www.fon.hum.uva.nl/PRAAT/ (Tested to work with PRAAT version 6.1.09).

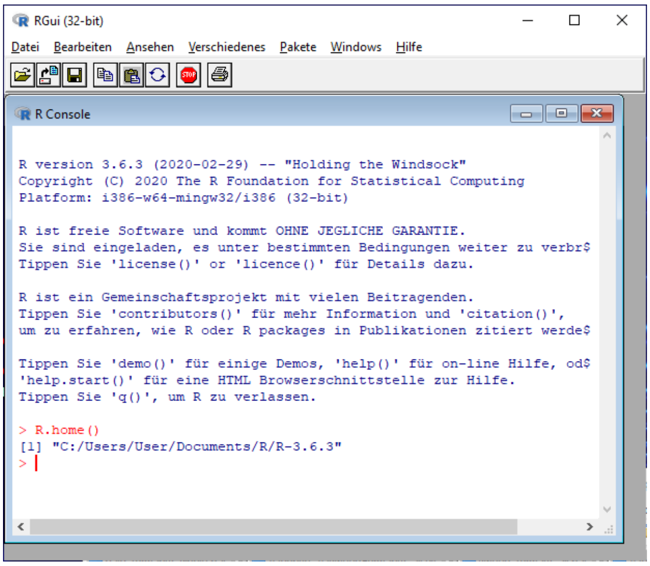
B.4.2. Add the PRAAT executable directory with `praat.exe` in to - usually something like `c:\Program Files\PRAAT` - to the Windows `PATH` variable (see above).

B.4.3. Test the PRAAT install: navigate to `\prog\batch` and start `VJtestPRAATinstall.bat`. This should display the PATH variable as PRAAT sees and reports – if all works fine – the PRAAT version number. If PRAAT doesn’t report, the PRAAAT executable path has not been added correctly (and should not show up on the screen on the PATH report). To fix, see sectn. B.3 above.

B.5. Install R

B.5.1. Install R from here: https://cran.r-project.org (tested to work with R version 3.6.3).

B.5.2. Add the R executable directory with `rscript.exe` in it is located to the Windows `PATH` variable (see well above). It may occur that R – especially when you try to "refresh" an older install – ends up as parallel install of the older version – say in programs: `c:\Program Files\R\R-3.5.2\bin` – and the newer version – say now in documents: `c:\Users\User\Documents\R\R-3.6.3\bin`. This is an utterly confusing situation, so you need to make sure that you get the correct directory added to you PATH variables. You can always find out where your "actual" R installation – the one you can access via the Windows menu – easily. Just start the R GUI from the windows start menu – in case of doubt start the version with the highest version number. Then type in `R.home()` <ENTER> and you get the associated path and version displayed (you need to add the trailing `\bin`, though):



```
R version 3.6.3 (2020-02-29) -- "Holding the Windsock"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/i386 (32-bit)

R ist freie Software und kommt OHNE JEGLICHE GARANTIE.
Sie sind eingeladen, es unter bestimmten Bedingungen weiter zu verbr$
Tippen Sie 'license()' or 'licence()' für Details dazu.

R ist ein Gemeinschaftsprojekt mit vielen Beitragenden.
Tippen Sie 'contributors()' für mehr Information und 'citation()',
um zu erfahren, wie R oder R packages in Publikationen zitiert werde$

Tippen Sie 'demo()' für einige Demos, 'help()' für on-line Hilfe, od$
'help.start()' für eine HTML Browserschnittstelle zur Hilfe.
Tippen Sie 'q()', um R zu verlassen.

> R.home()
[1] "C:/Users/User/Documents/R/R-3.6.3"
> 
```

B.5.3. Optionally: Install R-Studio desktop from here: https://rstudio.com/products/rstudio/download/ (tested to work with RStudio version 1.2.5033).

B.5.4. Install Rtools (for libraries, which require compile) from https://cran.r-project.org/bin/windows/Rtools/ (tested to work with Rtools version 3.5).

B.5.5. Install the following libraries (most conveniently done in RStudio > Tools > Install Packages… or via the R GUI, which should be found in the Windows start menu, via > packages > install packages). If asked, always install all dependencies and compile sources. Some care need to be taken, esp. when multiple newer and older R versions are installed in parallel: you need to ensure that the packages are installed against the latest R version and that is the one you call (either via `PATH` or the direct call). What always works is a simple clean refresh: download newest R, RTools, Rstudio, install in that order, then (re)install all packages – that should synchronize all (at one point in time you may have to get rid of all the older rubbish installs, though…).

- `data.table`
- `doBy`
- `dtt`
- `hqmisc`
- `ggplot2`
- `MASS`
- `ICSNP` [added in version 0.13]
- `doSNOW (only for multi-core processing; recommended)`
- `foreach (only for multi-core processing; recommended)`

B.5.6. Test the R install: navigate to `\prog\batch` and start `VJtestRinstall.bat`. This should display the `PATH` variable as R sees it and reports – if all works fine – the R version number, the R executable directory, the R library directory and all R libraries installed. If R doesn't report, the R executable path has not been added to `PATH` correctly (and should not show up on the screen on the `PATH` report). Check that all fits together: The R executable directory and the library directory should point to "the" actual R install. To fix, see B.3 above.

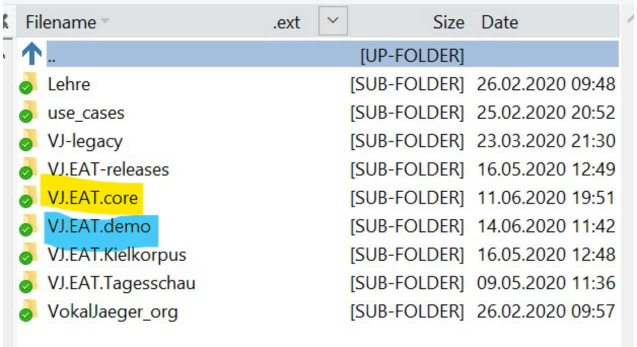
C. Create a working environment and test it with `VJ.EAT.demo`

C.1. Create a working environment

C.1.1. The working environment contains your sound files, text grids as input as well as the job batch files, the steering configuration and your output, the formants etc. The standard set up is documented in E.2, but you are entirely free to run your own structure (although that requires setting resp. adjusting the paths in all configuration and batch files).

C.1.2. A fully fledged test environment – VJ.EAT.demo – can be downloaded from: xxx.

C.1.3. Install it by copying the VJ.EAT.demo directory next to resp. in the same directory as VJ.EAT.core (You can copy it or any other working environment where you like, but then you need to adjust the paths in the configuration files). You should have now a directory structure like



Filename	.ext	Size	Date
..		[UP-FOLDER]	
Lehre		[SUB-FOLDER]	26.02.2020 09:48
use_cases		[SUB-FOLDER]	25.02.2020 20:52
VJ-legacy		[SUB-FOLDER]	23.03.2020 21:30
VJ.EAT-releases		[SUB-FOLDER]	16.05.2020 12:49
VJ.EAT.core		[SUB-FOLDER]	11.06.2020 19:51
VJ.EAT.demo		[SUB-FOLDER]	14.06.2020 11:42
VJ.EAT.Kielkorpus		[SUB-FOLDER]	16.05.2020 12:48
VJ.EAT.Tagesschau		[SUB-FOLDER]	09.05.2020 11:36
VokalJaeger_org		[SUB-FOLDER]	26.02.2020 09:57

C.2. Demonstration: end-to-end formant analysis

C.2.1. This one reads the sound files in `VJ.EAT.core` from the directory `\0 in_mp3` (the sound files, German lyrics have been choosen because of clear articulation and the license covering them), the TextGrid files from `\0 in_grid`, steered by the configuration in `\job_config` and produces for demonstration purposes in a series of outbound directories results. Navigate to `\do_jobs` and run the following batch scripts:

C.2.2. Run the batch script `1 calc_formants_all.bat`. This calculates formants using PRAAT and publishes its output into the directory `1 temp_data_formants_raw`: For documentation of the underlying algorithm, see E.3.

C.2.3. **Troubleshooting**: most common error is that PRAAT is not found, see B.3.

C.2.4. Run the batch script `2 sweep_formats_all.bat`. This selects formants using R and publishes its output into the directory `2 temp_data_formants_swept`: For documentation of the underlying algorithm, see E.4.

C.2.5. **Troubleshooting**: The most common errors are that either R is not found at all (R doesn't start) or it points to the wrong R version (i.e. not the one you installed the libraries against). You need to get the PATH variable right, see B.3. Other common failures are that the R libraries are either missing, outdated or not found (see B.5.5).

C.2.6. Run the batch script `3 normalize_formats_all.bat`. This normalizes formants using R and publishes its output into the directory `3 temp_data_formants_normalized`: For documentation of the underlying algorithm, see E.5.

C.2.7. Run the batch script `5 analyze_formants.bat`. This plots formants using R and publishes its output into the directory `5 temp_plots`: For documentation of the underlying algorithm, see E.6.

D. Methodology enhancements beyond the original VokalJäger

D.1. Artificial intelligence driven sweep

The original VJ basically selected the formant trajectory out of a large sample obtained by “sweeping” in PRAAT a series of upper frequency ceilings. From the selected trajectory the representative formants are taken. The selection process was mainly driven from physical considerations: which trajectory is the one with the best “fit” of a DTT, which trajectory is the “smoothest” [VJ, pp. 64-71]. But not necessarily the smoothest trajectory is the one yielding the most “appropriate” formant values.

The original VJ had already added “artificial intelligence” knowledge to “help” the algorithm to improve the selection process. The “Rückfaltung” penalized formant readings which are physically unlikely, given the area where the F1 and F2 values are usually expected, the “Extremdreieck” [VJ, pp. 94-98]. Other knowledge used was e.g. a test on unrealistic F3 values and B1/B2 ratios [VJ, p. 106, pt. 6].

VJ.EAT takes the artificial intelligence process to the next level. As usually – from the segmentation – the vowel quality as well as the speaker’s gender is already known on a qualitative base, we can optionally use that information as “guidance” to further refine the selection process. VJ.EAT now acts like a human person selecting the most “appropriate” curve / representative value.

A human agent especially would expect the upper frequency ceiling for female speaker to be about 5500 Hz or above and for a male speaker to be about 5000 Hz or below. The agent would further expect the formant values to be somehow in the ranges determined by statistics over a large sample of speakers [VJ, tab. 13, p. 135].

Hence it is now (optionally) possible to “switch on” an “artificial intelligence” agent to automatically help in the selection. Technically the physical criterion – the pass error [VJ, (7), p. 61] – is supplemented with probability measures, which firstly penalize unrealistic upper frequency ceilings for the gender at hand. Secondly unrealistic readings, not compatible with large speaker-sample statistics are penalized as well. All weights and parameters can and should be carefully calibrated / chosen, to avoid a forced “overfit” to the guidance. Technically the [sample] information is mapped against the reference statistics [VJ, tab. 13, p. 135] by the speaker’s gender.

Those probabilities are further optionally considered in the normalization task: too unrealistic readings, most likely caused by corrupt sound files or erroneous segmentation, can now – and should be – excluded from the reference used for normalization [added: V0.10].

Exact math to follow.

E. Code and data documentation

E.1. Directory structure of `VJ.EAT.core`

`\VJ.EAT.core:` Holds all files required for the VokalJäger algorithms to work.

`\doc`: Holds the documentation PDFs.

`\lib`: holds VokalJäger library files.

`\prog`: holds VokalJäger program files.

`\data`: holds reference data.

E.2. Directory structure of `VJ.EAT.demo`

`\VJ.EAT.demo`: Holds an exemplarily fully-operational workspace set-up, with all input sound and textgrid files plus and output formant files and plots. It comes with workspace specific batch scripts and config files. Explore this and clone – in case - for you specific tasks. Usually, unless you choose to change it for your project, the directory structure in `\VJ.EAT.demo` itself is as follows. This will work out of the boy only if you put the `\VJ.EAT.demo` directory and the `\VJ.EAT.CORE` directory in the same directory at the same level (see C.1).

`\do_jobs:` the project specific batch scripts, which call PRAAT and R.

`\job_config:` the project specific configuration files

`\0 in_grid:` the project specific PRAAT text grid files.

`\0 in_MP3:` the project specific sound files, here: MP3s.

`\1 temp_data_formants_raw:` the very original raw formant files as produced by `VJcalcFormants`, over all upper ceiling frequencies.

`\2 temp_data_formants_swept:` the formant files as produced by `VJcalcFormants`, i.e. by sample the one ceiling chosen and the one formant extracted from the trajectory.

`\3 temp_data_formants_normalized:` the formant files as produced by `VJnormlaizedFormants`, i.e. by sample normalized.

`\5 temp_plots:` analytical reports.

`\9 temp_data_reports:` various reports.

E.3. PRAAT program `VJcalcFormants`

Synopsis

E.3.1. Calculates with PRAAT for a given sound file and text grid formant, bandwidth and intensity trajectories for a series of upper formant ceilings, usually 4500-6000 Hz [VJ, sectn. 3.7.2, p. 104]. The trajectories are calculated inside the ”samples” (usually phonemes) which are identified via special text grid tier. The information is enriched with some supplementary data – usually the word / context from which the sample is taken - from an annotation text grid tier. See E.7 for exact documentation of output.

Location in `\VJ.EAT.core`:

```
\prog\PRAAT\ VJcalcFormants.praat
```

Batch script call:

```
start „VJ” praat [path]\VJcalcFormants.praat
```

Example call(s) in batch scripts in `\VJ.EAT.demo`:

```
\do_jobs\a calc_formants_all.bat
```

Configuration

E.3.2. A series of parameters need to be pushed to the PRAAT program in the exact order as documented below. Note that directory references handed to the PRAAT program are either relative (”..\..”) to the directory where the program is stored – i.e. relative to `\prog\PRAAT` – or are the full path (`”C:\...”`), best protected with parenthesis.

Troubleshooting and Errata

E.3.3. PRAAT occurs to be a bit sensitive in the naming convention of sound files (esp. when selecting them from within a script), so shorter non-whitespace names work.

E.3.4. In case you erroneously point PRAAT to the wrong directory to find the TextGrid files it will create empty ones in there.

E.3.5. Another common error is that you don’t properly quote the paths in the batch file – if you have chosen to change them. If you define a variable with a batch file with a path to a directory and the path contains spaces, you must quote it with “…”, always quoting is a got idea, e.g. `set _mypath=”C:\my path”`, not `set _mypath=C:\my path` without the quotes. When you use the variable you must *not* quote it, i.e. … `%_mypath%` and never `“%_mypath%”`.

`VJcalcFormants` call parameters	Example
`mySoundDirectory`: Input sound file directory.	`"..\..\demo\in_wav"`
`MySoundName`: Input sound file name, but excluding file extensions like `.wav`.	`"JanHofer_2016-11-03"`
`MySoundExtension`: Input sound file type/extension like `wav` (default) or `mp3`. See PRAAT manual for all supported file types. [added in Version 0.10].	`wav`
`myTextGridDirectory`: Input PRAAT text grid directory.	`"..\..\demo\in_grid"`
`myTextGridName`: Input PRAAT text grid file name, but excluding file extension `.textgrid`.	`"JanHofer_2016-11-03"`
`myOutputdirectory`: Output directory	`"..\..\temp_data_formants_raw"`

`tagOut`: The **tag** is the **fundamental key** to link different measurements (and evtl. recordings together). Usually it identifies one recording of one speaker in a specified context, mut as well be used to link different recordings together. Speaker normalization will be applied for all data assigned one tag. The tag is carried through all steps in the process and written into the file and filename. This tag must not contain spaces (use “_” in case instead).	`"TSch_JH"`
`tailTagOut`*:* The tail tag, as a numeric extension to the tag, is an optional information used to separate different recording files of the same speaker and/or different measurement files per recording (if the calculations are spread over tasks within a multi-thread environment). This is "just" a tailing tag for the filename, the subsequent R-kernel will merge all within one speaker/sample (=`tagOut`) together (default: `1`).	`1`
`theChannel`: Input WAV sound file channel to go for (default: `1`).	`1`
`shutterTier`: This PRAAT text grid tier is the sample tier and defines the time segments resp. the phonemes we calculate for (default: `1`). The tier acts as a “shutter” defining exact start end and point of the intra-sample formant trajectories to be evaluated. Only non-empty entries are processed and usually this sample tier should contain phonemes coded in generalized SAMPA letter codes [for vowels, VJ: 'S2' in tab. 2, p. 21], e.g. 'a:' (see as well filterSet, below).	`3`
`annotationTier`: This PRAAT text grid tier is the annotation tier, usually the full word from which the phoneme is taken, e.g. 'Tagesschau' (default: `2`). If not required, use same as `shutterTier`.	`1`
`Offset`: Offset for internal ID numbering (if not required, use: 0). This assigns different number ID-ranges to different samples / speakers. Note, that all samples must have different offset number ranges (or tags - see above), when later processed together in the R-kernel, BUT if you spread the same file over different sweeps you must make sure that the offsets are identical – else the R-kernel will treat exactly the same sample from this sweep (upper ceiling) different to exactly the same sample from that sweep (as the receive with different `offset` entries different `sampleIDs`)	`100000`
`lowerFrequency`: Lower formant frequency ceiling in PRAAT: start value of sweep in Hz (default: `4500`; [VJ: pp. 64-71].	`4500`
`upperFrequency`: higher formant frequency ceiling in PRAAT: end value of sweep in Hz (default: `6000`) [VJ: pp. 64-71].	`6000`
`frequencyStepWidth`: Formant frequency ceiling in PRAAT: increased in sweep by that amount in Hz (default: 100) [VJ: pp. 64-71].	`100`
`windowLength`: Window length in msec, default: 25 [VJ: p. 36].	`25`
`windowShift`: Window shift in msec, default: 2 [VJ: p. 36].	`2`
`maxCalcs`: Set to -1 to disable any limits (default), else calculation will terminate after here specified number of phonems calculated.	`-1`
`filterSet`: Restrict calculations to certain type of phonems pursuant on what is found in the phonem tier. `none`: No restriction are applied (default). `vowels`: Restrict to vowels - the phonem has to start with a generalized SAMPA vowel letter code [VJ: 'S2' in tab. 2, p. 21], what will catch diphthongs as well. `consonants`: Restrict to non-vowels (i.e. anything else).	`vowels`

`delayStart`*:* number of seconds the script waits before it actually starts, what is required to keep messages somehow tidy in multithreading environment (default: 0).	`0`

E.4. R program VJsweepFormants

Synopsis

E.4.1. Calculates for a series of formant files the sweep. Firstly all formant and bandwidth trajectories are smoothed with a DTT [VJ, pp. 56-64]. Then a set of "optimal" trajectories is selected from all PRAAT calibration settings, i.e. over all PRAAT upper frequency ceiling settings, as those with the lowest ruggedness, i.e. the lowest DTT pass error heuristics [VJ, pp. 64-69; 104-108]. Finally from those optimal trajectories formants are extracted at defined points and averaged, which then form the output of this program [VJ, pp. 69-71; 107]. As result the optimal PRAAT calibration was chosen and from the trajectories evaluated in that calibration, representative formant values are extracted. All calculations are performed on Bark scale, in- out outputs are reported in Hertz for convenience. See E.7 for exact documentation of output.

Location in `\VJ.EAT.core`:

```
\prog\R\ VJsweepFormants.R
```

Call:

```
rscript [path]\VJsweepFormants.R [CONFIG] [META]
```

Example call(s) in batch scripts and configuration files in `\VJ.EAT.demo`:

```
\do_jobs\2 sweep_formants_all.bat
\job_config\VJconfig_VJsweepFormants.txt
```

Sweep Configuration [CONFIG]

E.4.2. The program firstly takes a reference to the calculation configuration file, specific to `VJsweepFormants` which lists a series of more static option parameters how to perform and calibrate the calculation. Note that all directory references are either relative ("`..\..`") to the directory where the job was started or full path ("`C:\...`"). The config file is a text file which requires a *single* tabulator between the call parameter and its assigned actual value (always non-quoted), plus a carriage return in the last line of the file.

[CONFIG] calculation configuration file	Example
`inDirectory`*:* defines directory where to find the raw `VJformantsRaw` formant files, as produced by the PRAAT program `VJcalcFormants`. If entry is missing, this defaults to actual directory.	`..\temp_data_formants_raw`
`outDirectory`*:* Defines the directory where to write the resulting, selected formants (`VJformantSwept` files).	`..\temp_data_formants_swept`
`regexSample`: Allows to extract a subset from the original `[sample]` field as extracted from PRAAT (usually: the phoneme), e.g. to strip of trailing or leading information. What matches this Regex, is carried forward as (new, stripped) `[sample]` field, the original value is stored in a newly created `[sampleOrg]` field. In case all is to be used, apply as Regex `.+`, i.e. match-all (default). To e.g. extract everything before the first "`-`" in samples like `@-M-U-2`, use as Regex `[^\-]+`, match all except "-".	`.+`
`moduloCondition`*:* This optional feature allows a reduction of upper ceiling frequencies, esp. for debugging and testing purposes to reduce the processing load. Only upper ceiling frequencies `[tf]` modulo this number equal 0 will pass (e.g. if set to `500` the frequencies 4500, 5000, 5500 will be processed, but 4600, 4700 etc. will not. To disable, don't use or set to `-1` (default).	`-1`
`sweepDoParallel`*:* Defines whether (`yes`; default) or not (`no`) the sweep should be run In parallel, distributed over multiple cores at your PC.	`yes`

`sweepNumberOfCores`: For parallel processing, specify the number of cores you like to assign to the VokalJäger Sweep. Be careful: if you assign all cores / to many cores, you will not be able to do anything with your computer until the sweep is finished. If you assign a low number of cores (or disable parallel processing by setting it to `1`), you will have to endure a longer processing time. The performance does not scale linear with the number of cores assigned and all parallel processing makes only sense on big jobs. On a i9 with 8 cores going from 1 to 2 cores doubles the performance, going to 4 cores triples it, going to 6 and beyond does not scale above 3.5 – but all depends on your machine and number of cores (which may be as high a 64 on an AWS).	`5`
`runMode`: Specifies how the program executes and what reports it produces: `process` (default): Processes all files / all samples and writes results into the `outDirectory` and summary reports into the `reportDirectory`. `SelectedWithPictures` `SelectedWithPicturesAndData`: Processes only selected samples (see `reportSample`) and writes for those details charts and data files. They end up in to `reportDirectory`.	`process`
`sweepCushion`: defines which portion of the signal at beginning and end is ignored as security cushion, resp. which central portion is analyzed (default: 20% = `0.2`, resp. 60% center; [VJ, pp. 54, 104]).	`0.2`
`sweepOrderDTT`: defines the order of the DTT to be applied. Any deviation from the default value of `3` must be well contemplated [VJ, pp. 56-64, 105].	`3`
`sweepWeightBandwith`: defines the residual weight ($1-\alpha$) given to the bandwith pass-error when the pass error heuristic is evaluated (default: 0% = `0`; original VJ had 5%, resp. 95% to the formant pass error; [VJ, p. 105, (33)].	`0`
`sweepPickSingle`: defines the fixed point intra-sample location where the formant values are picked up, if fixed pick up is activated (default 30% = `0.3`; [VJ, p. 107, (34)]. Note, this excludes the cushion, i.e. is relative to the extracted center. This creates the `P`-marked results, e.g. `F1P`.	`0.3`
`sweepPickPair`: defines the fixed point intra-sample locations where an entry/left and exit/right formant pair is picked up (default 20% = `0.2` i.e. pick up at 20% and 80%). Note, this excludes the cushion, i.e. is relative to the extracted center. This creates the `L/R`-marked results, e.g. `F1L` and `F1R`.	`0.2`
`sweepApplyTriangleTest`: defines whether (default: `yes`) or not (`no`) the most-likely formant-triangle based “Rückfaltung” should be applied [VJ; pp. 93-98]. Note, that this works best with a “gender hint”, see `Gender` in the `META` configuration file below.	`yes`
`sweepTriangleCreep`: Defines by how many bark the original VJ [p. 120, tab. 9] reference triangle should be increase or decreased. A by `0.25` bark slightly widened reference formant-triangle is proposed, as it occurs the Kiel-Corpus speakers’ formant ranges are comparatively narrow. [added in Version 0.10].	`0.25`
`sweepAveragingBand`: defines which measurements, i.e. which upper frequency ceilings, should be considered when the final formant are	`0.02`

evaluated by averaging. This is done by defining a margin of error γ (default: 2% = `0.02`) to the total pass error [VJ, p. 107, pt. 8].	
`sweepGenderHintMalus:` if enabled (entry > `0`; disabled: `0`) and gender hits are provided in the `META` files, this artificially raises the pass error by a factor of `sweepGenderHintMalus` (defaut: `0.5`, i.e. 50%) for every 1000 Hz deviation in the actual upper ceiling applied in the sweep at hand from the PRAAT standard upper ceiling values of 5000 Hz (male) and 5500 (Hz) female (NB: ceiling below 5000 Hz and above 5500 Hz are not penalized for male resp. female speakers). This feature was not available in the original VJ and slightly “penalizes” unrealistic measurements respectively upper ceiling settings which are “too far off” from the text book values.	`0.2`
`sweepF3errorWeight:` when the pass error is constructed the F3 (B3) error component is downweighted by this factor (and the F1/F2 components are upweighted accordingly to achieve a 1 normalized weight sum). Default is `0.75` i.e. 75% (i.e. the “smoothness” of F1/F2 curves a slightly more important than that of the F3 curve). The original VJ had this feature disabled with a F3 error weight of `1`.	`0.75`
`sweepLocationProbabilityWeight:` Enables the AI-based sweep by assigning this weight to the probability component (default= `0.5`, i.e. 50%; switch off: `0`). Not available in the original VJ.	`0.5`
`sweepLocationProbabilityWidth:` width of equal-probability plateau: all samples within that range are dealt with a pseudo-probability of 1, measured a probability of the sample being in that range - after fitting a normal distribution (default: `0.5`, i.e. 50%; no plateau=0). The distribution / plateau is centered at the median of the empirical distributions as documented in [VJ, tab. 13, p. 135; factually tab. 59, p. 457], the standard deviations is approximated from there as 0.5 * IQR / 0.67. The plateau is required to give some flexibility over the expected values as documented in the VJ. Not available in the original VJ.	`0.5`
`reportSample`*:* Specify a list of sample IDs separated with a “`,`“ without spaces, for which details should be produced (see `reportMode`). The sample IDs usually are identified by spurious results by certain samples when reviewing the standard output in the `VJformantSwept` files.	`100004,100008`

Beta job configuration file [META]

E.4.3. Secondly it takes a reference to the meta job configuration file, specific to the project, which lists a series of the formant files names (referenced by their tag) plus supplementary information:

[META] job configuration file line entries (by column)	Example
`tag`*:* defines defines the tag to go for. Note that the data associated to this tag may be spread over several files by the `VJcalcFormants` program if the `tailTagOut` option has been used but the tail is *not* attached here.	`TSch_JH`
`Gender:` hint for the VokalJäger Rückfaltung (to select the reference triangle; see: `sweepApplyTriangleTest`) and gender bases malus (see: `sweepGenderHintMalus`) `male` [VJ, tab.9 "Männer", p. 120]. `female` [VJ, tab.9 "Frauen", p. 120]. `general` i.e. no hint [VJ, tab.9 "HG1", p. 120].	`male`
`doUse:` A flag to indicate whether or not this tag / formant file should actually be process (`Y`: default) or not (`N`). That allows selective calculations without deletion of lines.	`Y`
`doSweep:` A flag to indicate whether or not this tag / formant file should actually be swept (`Y`: default) or not (`N`). That allows selective calculations without deletion of lines.	`Y`
`_XXX:` Any other columns with a leading "_" would be just attached to the data in the `VJsweepFormants` program. This allows insertion of generic meta data to the process. Example: `_Name`	`Jan Hofer`

Standard Input

E.4.4. The program looks in the inbound directory for the raw

`VJformantsRaw … .TXT`

formant files, as produced by the PRAAT program `VJcalcFormants`.

Standard Output

E.4.5. The program produces in the output directory for each tag

`VJformantSwept … .TXT`

files containing the selected formants. It further produces into the report directory by tag a

`VJformantSwept_Stats … .TXT`

summary file, which holds descriptive statistics on the formant distributions by sample.

Detailed Output

E.4.6. In case the detailed analysis report were activated it produces in the report directory the picture files

`VJformantsSweep … .PNG`
`VJformantsSwept … .PNG`

The `Sweep` file show the formant trajectories of one sample for various ceiling frequencies plus colors those used for selection / averaging. The `Swept` file reports the minimum pass error trajectories (and annotates with the averaged values). The associated data is found in the

`VJformantsSwept … .TXT`

files, where the RsltSrs file hold all trajectories, the RsltWnnr only the trajectory with the minimum pass error and RsltFnl the one selected data set as it would be added to the standard output.

E.5. R program VJnormalizeFormants

Synopsis

E.5.1. Calculates robustly normalized formants and bandwidths [VJ, pp. 74-101]. See E.7 for exact documentation of output.

Location in `VJ.EAT.core`:

```
\prog\R\ VJnormalizeFormants.R
```

Call:

```
rscript [path]\VJnormalizeFormants.R [CONFIG] [META]
```

Example call(s) in batch scripts and configuration files in `VJ.EAT.demo`:

```
\do_jobs\3 normalize_formants_all.bat
\job_config\VJconfig_VJnormalizeFormants.txt
```

Normalization Configuration [CONFIG]

E.5.2. The program firstly takes a reference to the calculation configuration file, specific to `VJnormalizeFormants` which lists a series of more static option parameters how to perform and calibrate the calculation. Note that all directory references are either relative (”`..\..`”) to the directory where the job was started or full path (”`C:\...`”). The config file is a text file which requires a *single* tabulator between the call parameter and its assigned actual value (always non-quoted), plus a carriage return in the last line of the file.

[CONFIG] calculation configuration file	Example
`inDirectory`*:* defines directory where to find the `VJformantSwept` swept formant files, as produced by the R program `VJsweepFormants`. If entry is missing, this defaults to actual directory.	`..\temp_data_formants_swept`
`outDirectory`*:* Defines the directory where to write the resulting, normalized formants (`VJformantNormalized`) and statistics (`VJformantNormalizedStatistics`) files.	`..\temp_data_formants_normalized`
`normalizeRegexIncludeSample:` indicates whether `#all` (default) records or only a selected few should be *included* (unless explicitly excluded, see …`ExcludeSample` below) as reference for the normalization (i.e. those relative to which the order statistics resp. percentiles are evaluated [VJ, p. 86-88]. In case not `#all` should be selected, a standard Perl regular expression (Regex) [4] can be defined here, applied to the `samplecontext` field, which usually holds (VJ) SAMPA style phonetic coding [VJ, tab. 2, p. 21, col. S2], enclosed in preceding and following phonemes with “>” resp. “<”, which for the `sample` itself are equivalent to the standard start-`^` and end-`$` Regex symbols. Common VJ-Regex are: `>@<` … exactly single ‘`@`’ `>@<` … everything starting with ‘`@`’: `@`, `@n` … `@<` … everything ending with ‘`@`’: `@`, `o:@`, `E:@` … `>\w\:?<` …single vowel + optional ‘`:`’ `>\w\:<` …long vowel `>\w<` …short vowel `>\w\w<` …exact diphthong … >\w\:?<6 …long/short vowel followed by vocalized R	`#all`
`normalizeRegexExcludeSample:` indicates whether `#none` (default) records or only a selected few should be *excluded* from the reference for the normalization. See `IncludeSample` accordingly.	`^@$`

`normalizeMethodFormants` `normalizeMethodBandWidth:` defines the normalization methods applied to all formant (and bandwidth) columns. Implemented so far are: `robustLobanov` (default for formants): robust Lobanov Z-transformation [VJ, pp. 91-92, (29)]. `robustZ` (default for bandwidth): robust MAD Z-transformation [VJ, p. 91, (28)]. `robustGerstman:` robust Gerstman range transformation [VJ, pp. 90-91, (26)]. `Lobanov:` standard Lobanov Z-transformation [VJ, p. 82, (18)]. *Avoid, as not robust.* `Gerstman:` standard Gerstman range transformation [VJ, p. 81, (15)]. *Avoid, as not robust.*	`robustLobanov` `robustZ`
`normalizeBackprojectFormants`: Indicates whether (`yes`: default) or not (`no`) the normalized formants should be backprojected from Z to Hertz for convenience [VJ, p. 92; p. 128, tab. 11, entries under w+m]. As results the normalized formants (and/or bandwidth) are shown on the Hertz scale as those of a hypothetic androgyny / cross-gender speaker. So far, this only works for methods `robustLobanov` and `robustZ`.	`yes`
`normalizeBandStart`: Defines the maximum nominal band width α (default `0.05` = 5%) in the quantile statistics [VJ, pp. 86-88, (21-22)]. Required only for the robust Lobanov and Gerstman normalization measures.	`0.05`
`normalizeBandWidth`: Defines the actual band with ε after trimming for outliers (default `0.01` = 1%) in the quantile statistics [VJ, pp. 86-88, (21-22)]. Required only for the robust Lobanov and Gerstman normalization measures.	`0.01`
`normalizeF1hinting`: Enables (`yes`: default) or disables (`no`) the robustness increasing feature that for F2 quantile statistics only those F2 are considered, where the corresponding F1 values lie in the lower 1%-50% F1 percentile range. That helps to “find” the “upper” left and right F2-corners in the standard (plotted) F1/F2 formant triangle.	`yes`
`normalizeMaxError`: Only samples with a filter error below this threshold (as percentile of all filter errors) are considered for the normalization. That excludes obvious rubbish values – esp. those outside the norm triangle – from the calculation. Examples: set to `1.0` to include everything, `0.9` (default) to exclude worst 10% etc. Note that the percentile is evaluated excluding full failures with 0 dB (which are always excluded regardless of this setting). The percentile is evaluated on the selected set, see `IncludeSample` and `ExcludeSample` above. [added in Version 0.10].	`0.9`

Beta job configuration file [META]

E.5.3. Secondly it takes a reference to the meta job configuration file, specific to the project, which lists a series of the formant files names (referenced by their tag) plus supplementary information – *for general base entries, see: E.4.3.*

[META] job configuration file line entries (by column)	Example

`doNormalize:` A flag to indicate whether or not this tag / formant file should actually be normalized (`Y`: default) or not (`N`). That allows selective calculations without deletion of lines.	`Y`

Standard Input

E.5.4. The program looks in the inbound directory for the swept

`VJformantSwept… .TXT`

formant files, as produced by the R program `VJsweepFormants`.

Standard Output

E.5.5. The program produces in the output directory for each tag

`VJformantNormalized … .TXT`

files containing the normalized formants.

It further produces a statistics file

`VJformantNormalizedStatistics … .TXT`

summary file, which holds descriptive statistics on the formant distributions by sample.

E.6. R program VJanalyzeFormants

Synopsis

E.6.1. Calculates some statistics and plots a series of formant plots

Location in `\VJ.EAT.core`:

```
\prog\R\ VJanalyzeFormants.R
```

Call:

```
rscript [path]\VJanalyzeFormants.R [CONFIG] [META]
```

Example call(s) in batch scripts and configuration files in `\VJ.EAT.demo`:

```
\do_jobs\5 analyze_formants.bat
\job_config\VJconfig_VJanalyzeFormants_long_monophthongs
```

Normalization Configuration [CONFIG]

E.6.2. The program firstly takes a reference to the plot configuration file, specific to `VJanalyzeFormants` which lists a series of more static option parameters how to perform the plots. Note that all directory references are either relative (”`..\..`”) to the directory where the job was *started* or full path (”`C:\...`”). The config file is a text file which requires a *single* tabulator between the call parameter and its assigned actual value (always non-quoted), plus a carriage return in the last line of the file.

[CONFIG] calculation configuration file	Example
`inDirectory`*:* defines directory where to find the `VJnormalizeNormalized` normalized formant files, as produced by the R program `VJnormalizeFormants`. If entry is missing, this defaults to actual directory.	`..\temp_data_formants_normalized`
`outDirectory`*:* Defines the directory where to write the resulting plots.	`..\temp_plots`
`plotTitle`*:* a Text which will be inserted as leading text plot to title and plot file names.	`Long monophthongs`
`plotRegexIncludeSample:` indicates whether `#all` (default) records or only a selected few should be *included* in the plot, what is defined by a Regex analyzing the field `samplecontext` (see E.6 for more on Regex)	`\w\:$`
`normalizeRegexExcludeSample:` indicates whether `#none` (default) records or only a selected few should be *excluded* from the plot, , what is defined by a Regex analyzing the field `samplecontext` (see E.6 accordingly).	`#none`
`plotIncludeTag:` allows to select only one tag (usually: speaker) for analysis, default is inclusion of `#all` tags . [Added in version 0.11].	`#all`
`plotOriginal:` whether (`yes`: default) or not (`no`) to create a point by sample plot in the F1/F2 plane with original values before normalization.	`yes`
`plotNormalized:` …accordingly with normalized and back-projected values [VJ, p. 92-93].	`yes`
`plotSpectrum:` In case (`yes`; default: `no`), plots the spectrum respectively frequency response of the vocal tract as implied by the (original) first 3 formants and bandwidths. That’s done for all selected samples 1-file each, so you should ensure that not too much is selected. Technically it plots the amplitude spectrum (46) of the hypothized 3-pole	`no`

vocal tract transfer function (43), see VJ pp. 443-446 for details. Note, that this is idealized spectrum, “formed” from 3 formants, and not the (DFT) spectrum as measured in the sound file. [Added in version 0.11].	
`plotEllipseinclusionRate:` Defines, if ellipses are plotted, what ratio of data points they should include, under a 2-dimensional normal fit (default: `0.5`, i.e. the ellipses are supposed to cover 50% of the data points). The ellipse plots are based on a (robust) covariance estimator and are of lesser reliability the lower the assigned population counts are (starts plotting from 3 samples upwards).	`0.5`
`plotStatistics:` Indicates whether (`yes`) or not (`no`) a series of base F1/F2 statistics for both, the normalized and the original values, should be evaluated and plotted (NB: Usually you may want to include all samples here, see `plotRegexIncludeSample`). The statistics are evaluated in the bark domain for accuracy and reported in the hertz domain for readability (the number of significant digits follows standards in physics, see G.1). [Added in version 0.12]	`yes`

Beta job configuration file [META]

E.6.3. Secondly it takes a reference to the meta job configuration file, specific to the project, which lists a series of the formant files names (referenced by their tag) plus supplementary information – *for general base entries, see: E.4.3.*

[META] job configuration file line entries (by column)	Example
`doPlot:` A flag to indicate whether or not this tag / formant file should actually be plotted (`Y`: default) or not (`N`). That allows selective calculations without deletion of lines.	`Y`

Standard Input

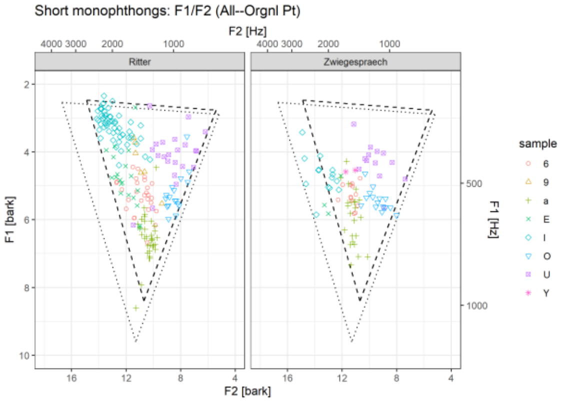
E.6.4. The program looks in the inbound directory for the normalized

`VJformantsNormalized… .TXT`

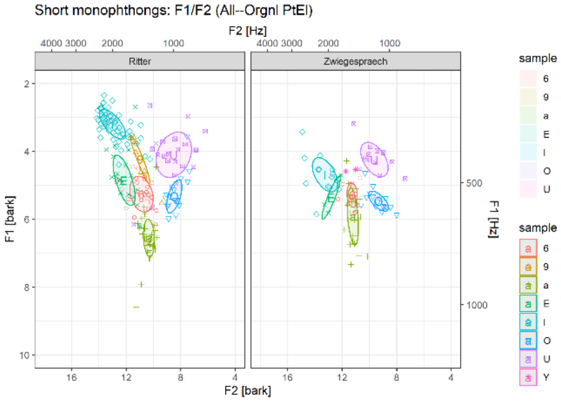
formant files, as produced by the R program `VJnormalizeFormants`.

Standard Output The program produces in the output directory

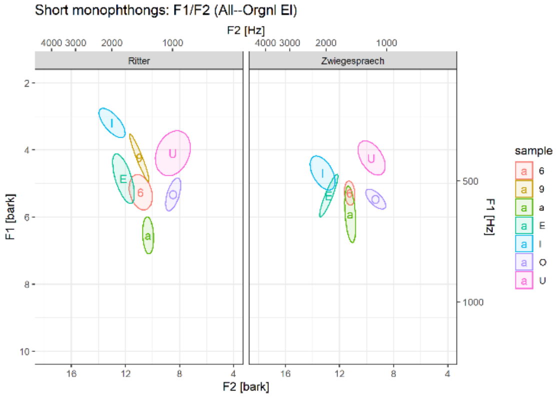
E.6.5. `VJplotF12_ … Orgnl (Pt)`: all data points on original scale before normalization. Plots as well the norm triangles for male (smaller) and female (larger) speakers. By tag, and across all tags (`All`). In case there are more than 15 distinct entries in [tag] it restricts to 15 and lumps all lower populated [tags] into one artificial "~other" category (applies to all VJ plots below as well).



E.6.6. `VJplotF12_ … Orgnl (PtEL)`: all data points on original scale before normalization, with fitted Ellipses. By tag, and across all tags (`All`);



E.6.7. `VJplotF12_ … Orgnl (EL)`: fitted Ellipses on original scale before normalization. By tag, and across all tags (`All`);

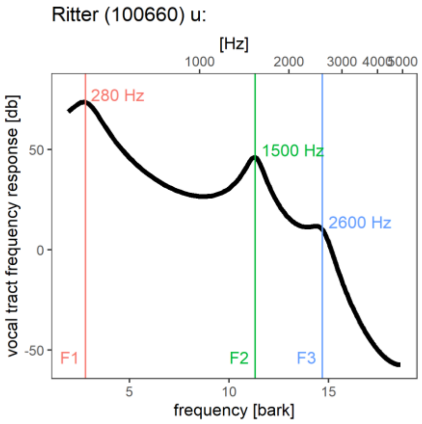


E.6.8. `VJplotF12_ … Nrmlzd (Pt)`: all data points after normalization and back-projection. By tag, and across all tags (`All`);

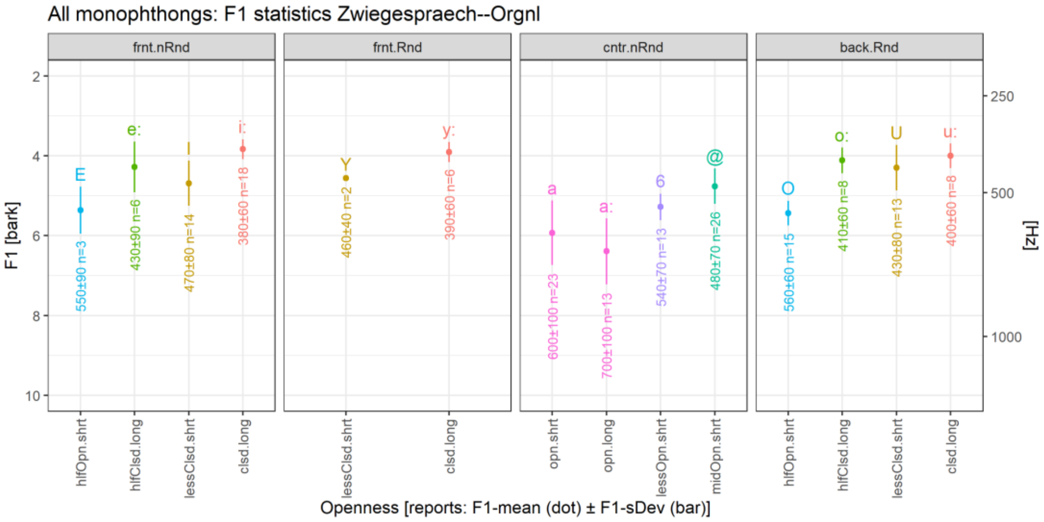
E.6.9. `VJplotF12_ … Nrmlzd (PtEl)`: all data points after normalization and back-projection, with fitted Ellipses. By tag, and across all tags (`All`);

E.6.10. `VJplotF12_ … Nrmlzd (EL)`: fitted Ellipses after normalization and back-projection. By tag, and across all tags (`All`);

E.6.11. `VJplotSpctrm_...`: Spectrum as implied by formants of sample, by sample (formant values are reported with 2 digits significance).

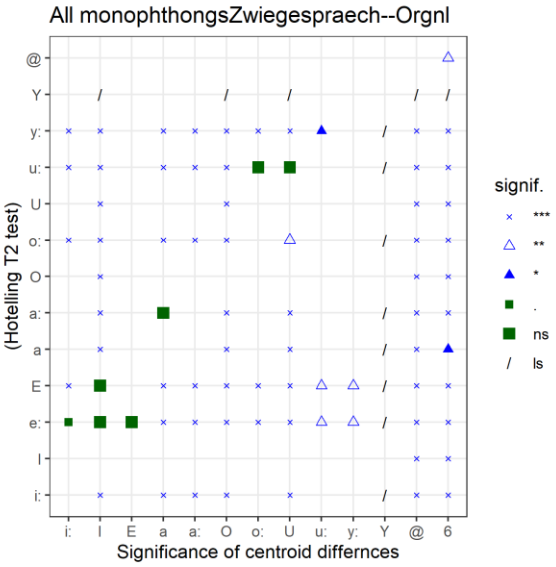


E.6.12. `VJstatsFx_...`: simple descriptive statistics on F1/F2, original and normalized (see G.1 for reporting of significant digits) [added in version 0.12].



E.6.13.

E.6.14. `VJsgnf_...`: Results of a two-dimensional t-test (Hotelling T2), which reports which intra-tag sample pairs are significantly different (whereby the euclidian distance of the sample centroids in the F1/F2 bark plane is the test measure; note there is no difference before/after normalization; sample groups with less than 3 samples are considered as too low sample counts to make an assessment and are excluded with a "ls" marker; see G.1 for significance coding). [added in version 0.13]. Note that this the plot is pruned, i.e. the pair X1-X2 is only shown once, hence missing under X2-X1. So if you miss X1-X2, find it as X2-X1.



E.7. Data dictionary

`[C]=config`: the data point was created based on a configuration.
`[R]=routed`: the data point was produced by a preceding program and is just routed through.
`[C]=calculated`: the data point was created by the program listed.

	File:	**`VJ formants Raw`**	**`VJ formants Swept RsltSrs`**	**`VJ formants Swept RsltWnnrs`**	**`VJ formants swept / Rslt Fnl`**	**`VJ formants normalized`**
	Produced by program:	`VJcalcFormants.praat`	`VJsweepFormants.R` (optional)	`VJsweepFormants.R` (optional)	`VJsweepFormants.R` (default)	`VJnormalizeFormants.R` (default)
Data structure (`tf`: PRAAT upper celling target frequency; `t`: intra sample trajectory time point)		For all samples: `tf * t`	For selected sample: `tf * t`	For selected sample: `tf`	For all samples / selected samples: tf^{win}	For all samples: tf^{win}
`tag`*:* core identifier.	`TSch_JH`	C	R	R	R	R
`tailtag`*:* number, in case the tag is spread over several files	`1`	C	R	R	–	–
`tf`: target ceiling frequency applied.	`4000`	**P**	R	R	–	–
`annotationID:` autonumbered entry as of annotation tier, starts with specified offset.	`100001`	**P**	R	R	R	R
`annotation`: the actual annotation entry (usually word from which the sample was taken) as of the annotation tier.	`Damen`	**P**	R	R	R	R
`sampleID:` autonumbered entry as of sample / shutter tier, starts with specified offset.	`100002`	**P**	R	R	R	R
`sample`: the actual sample entry as of the annotation tier (NB: from the original value only a subset which matches a Regex may be carried forward)	`a:`	**P**	**P***	R	R	R
`sampleOrg`: Keeps the very original [sample] field as created by PRAAT.	`a:`	**-**	**P**	R	R	R
`previoussample`: holds the `sample` of the preceding phoneme/sample under the condition that the preceding sample belongs to the same annotation/word (holds a ## in case the preceding	`d`	**P**	R	R	R	R

	File:	`VJ formants Raw`	`VJ formants Swept RsltSrs`	`VJ formants Swept RsltWnnrs`	`VJ formants swept / Rslt Fnl`	`VJ formants normalized`
sample belongs to another annotation; # in case there is no associated annotation).						
`nextsample`: holds the `sample` of the following phoneme/sample under the condition that the next sample belongs to the same annotation/word (holds a ## in case the preceding sample belongs to another annotation; # in case there is no associated annotation).	`M`	**P**	R	R	R	R
`samplecontext`: puts the sample/phoneme in the context of preceding (if any) and following (if any) phonemes. Constructed as: `[previoussample] + “>” [sample] + “<” [nextsample].` This allows convenient construction of Regex for analysis.	`d>a:<m`	**-**	**P**	R	R	R
`t`: the absolute time in the recording in [sec], 0 is beginning of recording.	`0.187`	**P**	R	R (@pickup)	R (@pickup)	R
`tRel`: the relative time in the sample in [sec], 0 is beginning of sample.	`0.01`	**P**	R	R (@pickup)	R (@pickup)	R
`tRelNrm`: the relative time in the sample in percent of sample length, 0 is beginning of sample, 100 is end of sample.	`12`	**P**	R	R (@pickup)	R (@pickup)	R (@pickup)
`fii`: measurement number, within sample (0 is beginning).	`1`	**P**	R	R (@pickup)	R (@pickup)	R (@pickup)
`F1-B3:` formant (F) /bandwidth (B) trajectory values, in Hertz over `t`	`553`	**P**	R	-	-	-
`F1b`**`Fltrd`**`-B3b`**`Fltrd:`** formant (F) / bandwidth (B) trajectory values in Bark, after DTT, over `t`	`553`	**-**	**P**	**-**	**-**	**-**
Filter error `FltErr` in (pseudo) dB per trajectory/`tf` [VJ, p. 106, (33)]		**-**	**P**	**R**	**R** (lowest error)	**R** (lowest error)
`FrmntSelect` indicates if the trajectory/`tf` is the one with the smallest filter error (`1`), if belonging to the set of trajectories/tf sufficiently near the one with the smallest error (`2`) [VJ, p. 107, pt.8], or else (`0`). The trajectories/`tf` marked with `1` or `2` are used to calculate the final formant values as averages [VJ, p. 107, (34)]. … holds the number of trajectories used in averaging		**-**	**P**	**R**	**-** **P**	**-** **R**

	File:	VJ formants Raw	VJ formants Swept RsltSrs	VJ formants Swept RsltWnnrs	VJ formants swept / Rslt Fnl	VJ formants normalized
F1**P**–F3**P**, B1**P**–B3**P**: formant/bandwidth extracted at central pick up point out of DDT-ed trajectory, in Hertz, spot values.	552	**–**	**P**	**R**	**–**	–
… averaged over minimal-error trajectories [VJ, p. 107, (34)].		**–**	**–**	**–**	**P**	R
F1**L**–F3**L**, F1**R**–F3**R**: formant extracted at left/intro (L) and right/extro (R) pick up point out of DDT-ed trajectory, in Hertz, spot values.	510	**–**	**P**	**R**	**–**	–
… averaged over minimal-error trajectories [VJ, p. 107, (34)].		**–**	**–**	**–**	**P**	R
F1P**N**–F3P**N**, B1P**N**–B3P**N**: normalized formant/bandwidth, in Z or back-projected in Hertz.	552	**–**	**–**	**–**	**–**	**P**
F1L**N**–F3L**N**, F1R**N**–F3R**N**: normalized formant extracted at left/intro (L) and right/extro (R) pick up point out of DDT-ed trajectory, in Z, , in Z or back-projected in Hertz.	510	**–**	**–**	**–**	**–**	**P**

F. Change Log

F.1. Version 0.15

F.1.1. Fixed bug that to small sample size can crash the hoteling T2 test

F.2. Version 0.14

F.2.1. Fixed bug that erroneously the F1/F2 probabilities got "normalized"

F.2.2. Added protection in plot in case of too many distinct entries in [sample]

F.3. Version 0.13

F.3.1. Adding Hotteling T2 test to `VJanalyzeFormants.R`.

F.4. Version 0.12

F.4.1. Adding base statistics to `VJanalyzeFormants.R`.

G. Appendix

G.1. Significance

G.1.1. Significant digits of a result depend on the variance resp. statistical error margin of the result. Following standards in physics the error is reported with one significant digit, and the result is reported with all digits up to the order of magnitude of the error, e.g. 9’123.41 is reported under an error of 0,678 as 9’124.4 ± 0.7 and under an error of 678 as 9’100 ± 700 [VJ, pp. 439-440]. This approach eliminates unscientific pseudo-accuracy which does not exist in the 1st place.

G.1.2. Significance of difference are evaluated with t-test resp. Hotellings T2-test or bootstrap equivalences. The p-values are coded following the standards in statistics as `***` p<0.001; `**` p<0.01; `*` p<0.05; `.` p<0.1; `ns` p>0.1 and `ls` in case there is test condition of (too) low sample counts [VJ, pp. 440-441].

H. References

[1] https://www.makeuseof.com/tag/a-beginners-guide-to-the-windows-command-line/
[2] https://helpdeskgeek.com/windows-10/add-windows-path-environment-variable/
[3] https://www.i8086.de/zeichensatz/ascii-7-bit.html
[4] http://kirste.userpage.fu-berlin.de/chemnet/use/suppl/perl-regex.html