Tools rPraat and mPraat: Interfacing phonetic analyses with signal processing

Tomáš Bořil and Radek Skarnitzl

Institute of Phonetics, Faculty of Arts, Charles University in Prague,
nám. Jana Palacha 2, Praha 1, Czech Republic
{tomas.boril,radek.skarnitzl}@ff.cuni.cz
http://fu.ff.cuni.cz/

Abstract. The paper presents the rPraat package for R / mPraat toolbox for Matlab which constitutes an interface between the most popular software for phonetic analyses, Praat, and the two more general programmes. The package adds on to the functionality of Praat, it is shown to be superior in terms of processing speed to other tools, while maintaining the interconnection with the data structure of R and Matlab, which provides a wide range of subsequent processing possibilities. The use of the proposed tool is demonstrated on a comparison of real speech data with synthetic speech generated by means of dynamic unit selection.

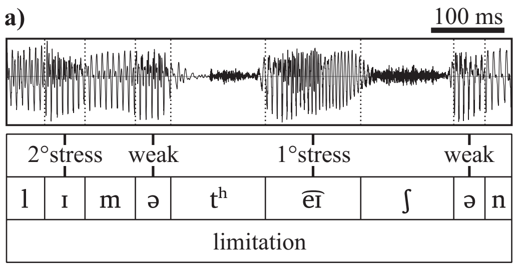
Key words: Matlab, R, Praat, phonetics, speech synthesis.

1 Introduction

Speech sciences have always been an interdisciplinary field of study. With this cross-disciplinary focus, it is natural that scientists from diverse backgrounds – linguists, speech engineers, psychologists, sociologists, neurologists and others – have to be able to communicate their research to each other, and to cooperate on mutual research goals. Although arguments for increased collaboration within the speech sciences have been put forward [1], this is no easy task. One of the reasons may include differing terminology, divergent mindsets, and also tools used to process speech data.

This paper addresses the last of these issues. Linguistically trained phoneticians and also psycholinguists typically use software tools such as Praat [2] to perform phonetic analyses. On the other hand, speech engineers typically rely on more general programmes like Matlab [3] or R [4]. The goal of this paper is to at least partially bridge the gap and introduce a tool which constitutes an interface between Praat on the one hand and Matlab and R on the other; the tool is called *mPraat* in the Matlab environment and *rPraat* in the R environment.

Praat works with various types of objects, most typical of which is the Sound and a corresponding text object (TextGrid) which includes the desired labels on interval tiers (e.g., word and phone boundaries) and on point tiers (e.g., prosodic break locations or midpoints of syllable nuclei); see Figure 1a for an example.



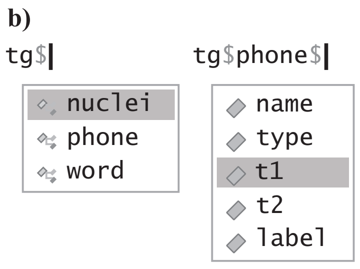


Fig. 1. a) An illustration of a TextGrid with one point (*nuclei*) and two interval (*phone*, *word*) tiers. b) Access to tier names and variables using RStudio [5] code completion.

The tool introduced here will be of assistance to phoneticians who want to make their analyses more accessible to technically oriented scientists, as well as to speech engineers who can use Praat TextGrids for segmenting sound files (freely available forced alignment tools often provide TextGrids as their output [6], [7]). The general applicability of our tool allows for subsequent advanced DSP processing (e.g., computing power spectral density, spectral slope [8], or adaptive threshold algorithms working with the spectral envelope [9]), statistical analyses, or generating high-quality graphics in Matlab or R.

Other tools have already been introduced which make phonetic analyses more compatible with general programmes. PhonTools [10] is an R package for phonetic analyses: users can display spectra or spectrograms, create vowel formant plots and F0 tracks, or filter sound files. PraatR [11] is a package for R which operates on the principle of external calling of Praat from R. Our rPraat differs from PraatR in two main respects. First, and importantly, rPraat does not run Praat in the background but performs all operations directly on the TextGrid, which makes the analyses faster (see section 3 for details). Also, it is possible to exploit the comfort of R's internal data structure, like the advantages of the effective vectorized access or a universal access to TextGrid components via tier names (see Figure 1b) etc. All these benefits are lost in PraatR. Second, unlike in PraatR, the range of rPraat commands extends beyond those offered in Praat, making some measurements more accessible and intuitive. The package thus does not merely provide an interface between Praat and R, but builds upon Praat in the R environment. This includes more effective work with PitchTiers (objects which contain the modifiable fundamental frequency (F0) track); our tool can be used to cut PitchTiers according to information in TextGrids. Additionally, the same functionality is provided for Matlab, for which no similar interface with Praat is, to our best knowledge, available.

Our rPraat (or mPraat) will be introduced in more detail in the following section. Section 3 presents an analysis of the tools performance. In section 4, we will illustrate its use in a comparison of real speech data from actual conversations with synthetic speech; we will focus on several acoustic parameters of fricative sounds and point to possible problems of concatenative speech synthesis.

2 Introducing rPraat and mPraat

The current version of the tool is available on `http://fu.ff.cuni.cz/praat/`, along with detailed instructions and a demonstration. Tab. 1 provides an overview of rPraat / mPraat functions. These include reading and writing, querying and modifying TextGrids. As was mentioned in the Introduction, some of the functions, such as `tg.insertInterval`, are new (i.e., not featured in Praat). These functions make work with TextGrids considerably easier.

Table 1. Summary of rPraat functions. mPraat features similar names, but without fullstops and with the following letter capitalized, e.g., tg.getLabel() *vs.* tgGetLabel().

tg.read / tg.write / tg.plot / tg.createNewTextGrid / tg.repairContinuity pt.read / pt.write *(PitchTiers containing F0 tracks)*
tg.getStartTime / tg.getEndTime / tg.getTotalDuration
tg.getNumberOfTiers tg.getTierName / tg.setTierName / tg.isIntervalTier / tg.isPointTier tg.insertNewIntervalTier / tg.insertNewPointTier / tg.duplicateTier / tg.removeTier
tg.getNumberOfIntervals / tg.getIntervalStartTime / tg.getIntervalEndTime tg.getIntervalDuration / tg.getIntervalIndexAtTime tg.insertBoundary / tg.insertInterval / tg.removeIntervalLeftBoundary tg.removeIntervalRightBoundary / tg.removeIntervalBothBoundaries
tg.getNumberOfPoints / tg.getPointTime / tg.insertPoint / tg.removePoint tg.getPointIndexLowerThanTime / tg.getPointIndexHigherThanTime / tg.getPointIndexNearestTime
tg.getLabel / tg.setLabel / tg.countLabels

Many operations may be performed at a high level, using the `tg` functions which correspond to their equivalents in Praat with thorough checks and warnings, or directly using low-level commands. For instance, let us assume a TextGrid read in the TG variable. We may find out the number of descriptive tiers using `tg.getNumberOfTiers(TG)` or merely `length(TG)`. To find out the start time of the 7th interval on the phone tier, the command is
`tg.getIntervalStartTime(TG, "phone", 7)` or `TG$phone$t1[7]`.

It is possible to use the pipeline operator `%>%` for sending output from one function to the following function, instead of its first parameter, for example:

```
tg.read("H.TextGrid") %>% tg.removeTier("word") %>%
    tg.write("out.TextGrid")
```

The tremendous advantage of vectorized operations can be illustrated on the calculation of the mean duration of all [e/eː] vowels on the phone tier:

```
condition <- TG$phone$label %in% c("e", "e:")
meanDur <- mean(TG$phone$t2[condition] - TG$phone$t1[condition])
```

A single change – `hist` for `mean` – would produce the histogram of the duration values. As opposed to this simple procedure, we would have to create a for-loop in PraatR and repeat the following code for every index of an interval:

```
lab <- praat("Get label of interval...", list(2, index),
                 input = "file.TextGrid")
if (lab == "e" | lab == "e:") {
    t1 <- as.numeric(praat("Get start point...", list(2, index),
                 input = "file.TextGrid", simplify = TRUE))
    t2 <- as.numeric(praat("Get end point...", list(2, index),
                 input = "file.TextGrid", simplify = TRUE))
    dur <- t2 - t1; totalDur <- totalDur + dur; cnt <- cnt + 1
}
```

Praat would also require the use of a for-loop, but the notation is considerably more concise. Moreover, it is not necessary to read each file again when making a new query, for example: `t1 = Get start point: 2, index`.

3 Performance analysis

The objective of the three experiments reported in this section was to compare the performance of our Matlab and R package with that of Praat and the PraatR package. For this purpose, we used recordings of 200 short dialogues, each composed of five speaking turns. The dialogues were recorded in a sound-treated studio of the Institute of Phonetics in Prague, as mono recordings with a sampling frequency of 32 kHz and 16-bit depth. The recordings were segmented manually; each recording contains, on average, 29 phones (mean recording duration 2.065 s) and 3.4 [e/eː] vowels (mean duration 216 ms). The duration of processing a given task was measured on all 1,000 files. Each task was repeated 12 times, and the results are based on the last 10 cycles: the first two cycles were ignored so as to eliminate the initial reading of the files from the hard drive into memory. In PraatR, only 10 files were tested with 5 repetitions (and the first two ignored), due to the extremely slow processing speed. The TextGrids contain seven tiers (see section 1). The mean duration values, as well as the 95% confidence intervals using bootstrap are reported below.

All TextGrid formats were used for the experiments: the Full format (full text with each line led by its name, such as `'xmin = '`); the Short format (an abbreviated form of the Full format which includes only the values); and the Binary format (a separate native binary format for each software programme).

The experiments were run on a Samsung 530U laptop with 64-bit Windows 7 Home Premium, Intel(R) Core(TM) i5-2467M CPU @ 1.60GHz processor, 4 GB DDR3 RAM, and a hybrid (500 GB HDD, 5400 rpm + 16 GB SSD) drive. The following software versions were used: Praat 6.0.14 64bit, Matlab R2013b 64bit, and R version 3.2.4 64bit with RStudio 0.99.891, and the attached packages dplyr_0.4.3, dygraphs_0.8, PraatR_2.3, stringr_1.0.0, tidyr_0.4.1.

Experiment 1 consisted in calculating the mean average duration of all [e/eː] vowels. It was calculated as the difference between the End and Start time of those intervals which correspond to the label criterion (i.e., [e] or [eː]).

In *Experiment 2*, we calculated the mean energy of all [e/eː] vowels. The task thus involved the reading of the wav files, along with the TextGrids. The computations were slightly different in R and Matlab, as we always chose that procedure which yielded a faster result. In R, energy was computed as the sum of the squared amplitudes of the individual samples, `sum(segment^2)`. Matrix multiplication `segment'*segment` was used in Matlab and the `Get energy` function in Praat and PraatR.

Experiment 3 consisted in listing all the labels of the second (phone) tier from all TextGrids into one single file each label written on a separate line, yielding a table with one column. This can be useful if we are interested in calculating the occurrences of individual phones in all the files. This task is the simplest of the three, with labels being merely listed in one file.

At this point, we would like to mention one way of listing results in Praat, the Info window; less experienced Praat users often write results into the Praat Info window rather than into a file, but they should be warned that this is an extremely slow process. We carried out Experiment 3 with the binary format TextGrids: it took 17.11 seconds to list the results of 100 files, 80.76 sec for 200 files, and 370.65 sec for 500 files. In other words, listing into the Praat Window becomes slower with every new line.

The results of all the three performance experiments are shown in Figure 2. It is clear that PraatR is by a magnitude slower: the processing of one file took, in all the experiments, more than 9 seconds. This is caused by the fact that for every single query, Praat has to be run in the background and the file has to be read into memory again.

The results of Experiment 1 (Fig. 2a) show that mPraat and rPraat are slower when reading the text-format TextGrids because the operations are performed in a scripting language, while Praat is programmed in C and compiled into machine code. Still, the processing is quite fast in mPraat and rPraat; when a repeated access to TextGrids is required, it is beneficial to convert them into the binary format, where the reading is ca. twice as fast as in Praat.

In Experiment 2 (Fig. 2b), the analyses which include access to sounds are fastest in Praat, due to its low-level implementation, but the performance of mPraat and rPraat is quite acceptable. The reading of the text formats of the TextGrid is, again, what takes longest: especially rPraat approaches Praat when binary files are used.

In Experiment 3 (Fig. 2c), where individual labels are written into a text file, the processing speed of mPraat and rPraat is similar as in Experiment 1, but Praat is considerably slower. The performance is best in rPraat with binary files, where processing time per file equals 0.4 seconds.

4 Comparison of real and synthetic speech

The aim of this experiment was to compare selected acoustic properties of speech sounds from real and synthetic speech. We focused on the postalveolar voiceless fricative [ʃ]. This sound appeared twice in each of 31 target sentences. In type 1 sentences, one [ʃ] appears within the sentence and the other is (near-)final (e.g., "V naší vile občas straší" (*Our villa is sometimes haunted*), while in type 2

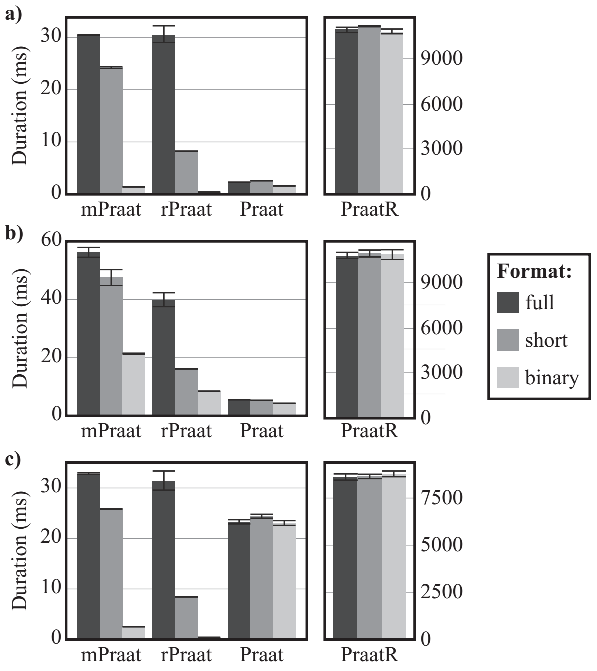


Fig. 2. Results of a) experiment 1, b) experiment 2, c) experiment 3.

sentences both [ʃ] sounds are non-final (e.g., "Bude**š** jist, že dojde**š** jistě k cíli." (*You'll know for sure that you'll reach the goal*). The target [ʃ] sounds always appeared in the same segmental context. That allowed us to compare the acoustic make-up of the target fricatives and to examine the effect of position within a sentence. In other words, we were interested whether real and synthetic speech would yield different results in terms of the declination of acoustic parameters.

Eight female students of the Faculty of Arts were asked to read the 31 sentences. They were recorded under the same conditions as those mentioned in section 3. The same sentences were synthesized using dynamic unit selection in three female voices featured in the ARTIC system [12]. The target phones were identified and their boundaries set manually in Praat TextGrids, so that only their voiceless portions (and no carryover voicing) were included.

We analyzed the difference in the values between the first and second [ʃ] in three parameters: duration, power (intensity) and the centre of gravity (COG). The results of the analyses performed in R are shown in Figure 3. When the boxplots are centred around 0, there is no significant trend which would point to differences between the first and second [ʃ]. An interesting trend found in real speech data, with no such trend in synthetic speech, points to possible deficiencies of speech synthesis.

As for duration, we can see in both real and synthetic speech higher values in the second [ʃ] in type 1 sentences (those where the second [ʃ] is located towards

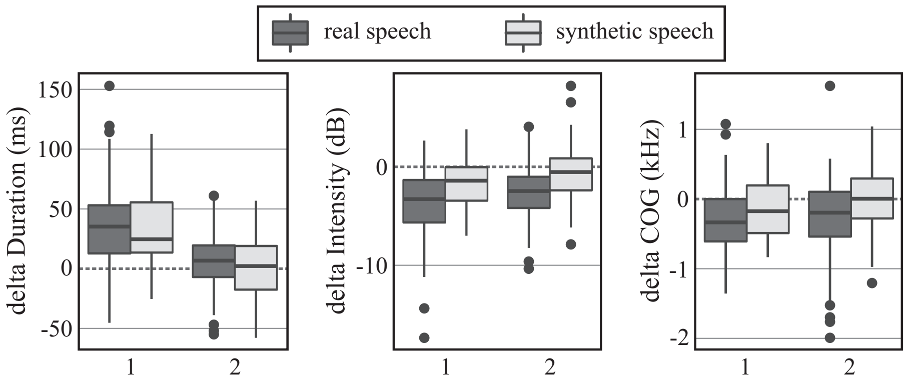


Fig. 3. Comparison of real and synthetic [ʃ] in type 1 and type 2 sentences (see text), showing the difference in duration (left), intensity (centre) and centre of gravity (right) between the second and first [ʃ]. A positive value thus means, e.g., longer duration of the second [ʃ].

the sentence end). No such trend is apparent in type 2 sentences (the boxplots are more or less centred around zero). The agreement between real and synthetic speech is not surprising, since diphones are selected from the corpus according to the context.

Similarly, the data document the declination of intensity in type 1 sentences in both real and synthetic speech, while in type 2 sentences some degree of declination is apparent only in real speech; this may point to natural, gradual (but also inconspicuous) declination. Since non-final diphones are selected from various portions of the original sentences, on average there is no change between the first and second [ʃ] in synthetic speech.

Centre of gravity show a declination tendency in both sentence types in real speech: as the sentence end approaches and breath becomes weaker, the less "sharp" the sound of the fricative is and the COG drops. Such lower resonances are easily audible (*cf.* You don't wa**sh** where I wa**sh**). Some degree of COG drop is visible also in type 1 synthetic sentences but not in type 2, due to the random selection of segments from the original sentences.

5 Conclusions

The objective of this paper was to introduce a new tool which provides an interface of Praat with R and Matlab, to evaluate its performance with respect to other tools, and to demonstrate some applications on real data. The rPraat R package / mPraat Matlab toolbox proved to be computationally very effective especially when we use the binary format (see section 3). The package offers more options for processing TextGrids and also PitchTiers which are absent in Praat. Most importantly, however, the tool's interconnection with the data structure of R and Matlab provides a wide range of subsequent processing possibilities.

The experiment reported in section 4 demonstrated the use of our tool on a real-life research task: the comparison of real and synthetic speech. We wanted to see whether the acoustic properties of the fricative [ʃ] change in a similar way in real speech (see [13] for the concept of segmental intonation) and speech created by means of concatenative synthesis. Informal perceptual observations suggest that the differences sometimes are audible. Comparing such acoustic properties might also be a useful method for detecting spoofing for the purposes of identifying manipulations to the speech signal or speaker verification.

Acknowledgement

This research was supported by the Czech Science Foundation project No. 16-04420S.

References

1. Barry, W.J., Van Dommelen, W.A. (eds.): The Integration of Phonetic Knowledge in Speech Technology. Springer, Dordrecht (2005)
2. Boersma, P., Weenink, D: Praat: doing phonetics by computer (version 6.0.14), `http://www.praat.org/` (2016)
3. MathWorks: MATLAB Release 2013b. MathWorks, Inc., Natick, Massachusetts (2013)
4. R Core Team: R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria, `http://www.R-project.org/` (2016)
5. RStudio Team: RStudio: Integrated Development Environment for R. RStudio, Inc., Boston, MA, `http://www.rstudio.com/` (2015)
6. Yuan, J., Liberman, M.: Speaker identification on the SCOTUS corpus. In: Proceedings of Acoustics 2008, pp. 5687–5690 (2008)
7. Pollák, P., Volín, J., Skarnitzl, R.: HMM-Based Phonetic Segmentation in Praat Environment. In: Proceedings of SPECOM 2007, pp. 537–541 (2007)
8. Volín, J., Zimmermann, J.: Spectral slope parameters and detection of word stress. In: Proceedings of Technical Computing Prague, pp. 125–129 (2011)
9. Šturm, P., Volín, J.: P-centres in natural disyllabic Czech words in a large-scale speech-metronome synchronization experiment. Journal of Phonetics 55, 38–52 (2016)
10. Barreda, S.: phonTools, Functions for phonetics in R. R package (version 0.2-2.1), `http://www.santiagobarreda.com/` (2015)
11. Albin, A.: PraatR: An architecture for controlling the phonetics software "Praat" with the R programming language. J. Acoust. Soc. Am. 135(4), 2198–2199 (2014)
12. Matoušek, J., Tihelka, D., Romportl, J.: Current state of Czech text-to-speech system ARTIC. In: Text, Speech and Dialogue, LNCS 4188, pp. 439–446. Springer-Verlag, Berlin (2006)
13. Niebuhr, O.: At the edge of intonation: the interplay of utterance-final F0 movements and voiceless fricative sounds. Phonetica 69(1-2), 7–27 (2012)