Tools rPraat and mPraat

Interfacing phonetic analyses with signal processing

Tomáš Bořil, Radek Skarnitzl
Introduction

Someone would think that everything in phonetics could be solved in software Praat

Which is obviously not true...
What makes phonetic analysis unique?

- Precise processing – manual segmentation
  - Time-consuming, exacting, elaborate
  - Consistent, well-defined rules
  - Valuable

- Thorough understanding of perceived sounds ↔ physiological and linguistic processes
Phonetics

Languages – dialects – sound change – social variation – individual differences
Phonetics

- Praat is a primary tool for manual segmentation
- And for that, we have qualified students 😊
Praat

- Praat has many built-in functions for analyses
- But how to apply something special?
- New methods, modifications, detailed settings?
- Then, one has to switch to a more elaborate tool and create it on its own
- Matlab, R, etc.
- Signal processing, statistics, graphics…
F₀ contours

- Another important fundamental in speech science
- Praat is reputable in F₀ contours detection

- Anyhow, subsequent interventions are necessary
  - Manual correction of errors
    - Octave jumps, creaky phonation, etc.
  - Stylization of contours
    - Perception modelling
The valuable work

- Manual segmentation and text annotations
  - TextGrids
- Manually corrected $F_0$ contours
  - PitchTiers

- Missing tools
  - Native support of these files in Matlab, R
What did we create?

- Package / toolbox rPraat & mPraat
  - Opensource – github: everyone can contribute
  - Implement new functions? Modifications? pythonPraat?
rPraat & mPraat

- Import / export of TextGrids and PitchTiers
- Complete set of functions for their processing
  - Identical to Praat
- And also extras: what we missed in Praat
- New algorithms may be created and the results can be passed back into Praat
- Matlab and R languages are more flexible and well-arranged, more comfortable even for tasks soluble in Praat
# rPraat & mPraat

**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()`.

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tg.read</code> / <code>tg.write</code> / <code>tg.plot</code> / <code>tg.createNewTextGrid</code> / <code>tg.repairContinuity</code></td>
</tr>
<tr>
<td><code>pt.read</code> / <code>pt.write</code> (<strong>Pitch Tiers containing F0 tracks</strong>)</td>
</tr>
<tr>
<td><code>tg.getStartTime</code> / <code>tg.getEndTime</code> / <code>tg.getTotalDuration</code></td>
</tr>
<tr>
<td><code>tg.getNumberOfWeeks</code></td>
</tr>
<tr>
<td><code>tg.getTierName</code> / <code>tg.setTierName</code> / <code>tg.isIntervalTier</code> / <code>tg.isPointTier</code></td>
</tr>
<tr>
<td><code>tg.insertNewIntervalTier</code> / <code>tg.insertNewPointTier</code> / <code>tg.duplicateTier</code> / <code>tg.removeTier</code></td>
</tr>
<tr>
<td><code>tg.getNumberOfWeeks</code> / <code>tg.getIntervalStartTime</code> / <code>tg.getIntervalEndTime</code></td>
</tr>
<tr>
<td><code>tg.getIntervalDuration</code> / <code>tg.getIntervalIndexAtTime</code></td>
</tr>
<tr>
<td><code>tg.insertBoundary</code> / <code>tg.insertInterval</code> / <code>tg.removeIntervalLeftBoundary</code></td>
</tr>
<tr>
<td><code>tg.removeIntervalRightBoundary</code> / <code>tg.removeIntervalBothBoundaries</code></td>
</tr>
<tr>
<td><code>tg.getNumberOfWeeks</code> / <code>tg.getPointTime</code> / <code>tg.insertPoint</code> / <code>tg.removePoint</code></td>
</tr>
<tr>
<td><code>tg.getPointIndexLowerThanTime</code> / <code>tg.getPointIndexHigherThanTime</code> / <code>tg.getPointIndexNearestTime</code></td>
</tr>
<tr>
<td><code>tg.getLabel</code> / <code>tg.setLabel</code> / <code>tg.countLabels</code></td>
</tr>
</tbody>
</table>

**Help and samples are included**
Example: mean energy [e, e:]

tgID = Read from file... R01_3_PILA.TextGrid
sndID = Read from file... R01_3_PILA.wav

numInt = Get number of intervals: 2

for i from 1 to numInt
    select tgID
    lab$ = Get label of interval: 2, i
    if lab$ == "e" or lab$ == "e:"
        t1 = Get start point: 2, i
        t2 = Get end point: 2, i
        select sndID
        en = Get energy: t1, t2
        sum = sum + en
        count = count + 1
    endif
endfor

select tgID
Remove
select sndID
Remove

tg <- tg.read("R01_3_PILA.TextGrid")
snd <- readWave("R01_3_PILA.wav")
fs <- snd@samp.rate; signal <- snd@left / (2^(snd@bit - 1))

condition <- tg$phone$label %in% c("e", "e:"

t2 <- tg$phone$t2[condition]
t1 <- tg$phone$t1[condition]

for (S in seqM(1, length(t1))) {
    segment <- signal[(trunc(t1[S]*fs+1)) : (trunc(t2[S]*fs+1))]
    SUM <- SUM + sum(segment^2)
    COUNT <- COUNT + 1
}
Overview

“Low-level”

Traditional “Praat-style”

\begin{verbatim}
TG[[2]]$t1[7]  length(TG[[2]]$t1)  

tg.getIntervalStartTime(TG, 2, 7)  # 2nd tier, 7th interval

 tg.getNumberofIntervals(TG, 2)
\end{verbatim}

Advantages

\begin{verbatim}
# access by name
TG$phone$t1[7]  tg.getIntervalStartTime(TG, "phone", 7)

# pipeline operator
tg.read("H.TextGrid")  %>%  tg.removeTier("word")  %>%  tg.write("out.TextGrid")

# vectorized operations
unique(TG$phone$label)  # unique labels
hist(TG$phone$t2 - TG$phone$t1)  # interval duration

# cut PitchTiers according to TextGrids
- no similar interface with Praat is available
\end{verbatim}
Smart code completion

\[ \text{tg.s|} \]

- phone
- word
- phrase

\[ \text{tg.word|} \]

- name
- type
- t1
- t2
- label

```
tg.|
- tg.getTotDuration
- tg.checkTierInd
- tg.insertBoundary
- tg.insertInterval
- tg.insertNewIntervalTier
- tg.insertNewPointTier
- tg.insertPoint
- tg.isIntervalTier
- tg.isPointTier
```

`tg.insertInterval(tg, tierInd, tStart, tEnd, label = "")`

Inserts new interval into an empty space in interval tier: a) Into an already existing interval with empty label (most common situation because, e.g., a new interval tier has one empty interval from beginning to the end. b) Outside of existing intervals (left or right), this may create another empty interval between.

Press F1 for additional help
Performance analysis

- **Data:** 1000 files, mean of 10 repetitions
  - **TextGrids:** Full, short, binary format:
    - slightly different structure, identical content

- **Experiments**
  - **Ex1:** Mean average duration of all [e/e:] vowels (TG)
  - **Ex2:** Mean energy of all [e/e:] vowels (TG + Wav)
  - **Ex3:** List all labels of the phone tier from all TextGrids into one single file, each label on a separate line (TG)
Performance analysis

- **Ex1**: Mean average duration of all [e/e:] vowels (TG)

- **Ex2**: Mean energy of all [e/e:] vowels (TG + Wav)

- **Ex3**: List all labels of the phone tier from all TextGrids into one single file, each label on a separate line (TG)
Real vs synthetic speech

Is there any kind of declination (effect of position in sentence) in postalveolar voiceless fricative [ʃ]?

- Duration – Intensity – Centre of gravity (COG)

Real speech vs synthetic (dynamic unit selection)
- ARTIC system, University of West Bohemia
Real vs synthetic speech

- **Type 1 sentence (9×)**
  - 1\textsuperscript{st} [ʃ] within sentence, 2\textsuperscript{nd} [ʃ] (near-)final
  
  E.g., “V naší vile občas straší.”
  
  *Our villa is sometimes haunted.*

- **Type 2 sentence (22×)**
  - 1\textsuperscript{st} and 2\textsuperscript{nd} [ʃ] both non-final

  E.g., “Budeš jist, že dojdeš jistě k cíli.”

  *You’ll know for sure that you’ll reach the goal.*

- **Real speech:** 8 females, synthetic: 3 female voices
Real vs synthetic speech

![Graphs comparing delta Duration (ms), delta Intensity (dB), and delta COG (kHz) between real speech and synthetic speech.]
Conclusions

- Declination of acoustic properties of the fricative $[ʃ]$ is measurable

- There is a distinct difference between real and synthetic speech
  - Helpful parameter for detection of spoofing (speaker verification), manipulations to the speech signal

- rPraat and mPraat tools are open-source, available at http://fu.ff.cuni.cz/praat/
Thank you for your attention.

http://fu.ff.cuni.cz/praat/