167th Meeting of the Acoustical Society of America (Providence, Rhode Island; May 5-9, 2014) Speech Communication **1pSC27**

PraatR: An architecture for controlling the phonetics software 'Praat' with the R programming language

Aaron Albin (aaalbin@indiana.edu) Department of Linguistics, Indiana University - Bloomington

1. Motivation

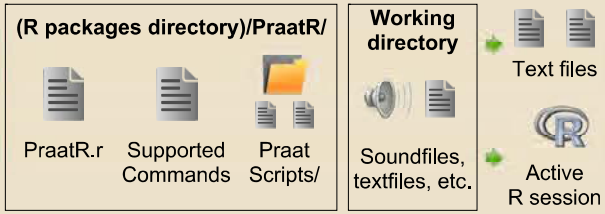
Common workflow:	Problems:
1. measure in acoustics program (e.g. Praat)	• if find outlier, inconvenient to go back, check, fix, and re-analyze
2. organize in spreadsheet program (e.g. Excel)	• measurement extraction isolated from quantitative modeling
3. analyze in statistics program (e.g. SPSS)	• slower workflow (more clicks, multiple scripting languages)

Goal = to conduct entire analysis without switching software

More and more linguists using R (since powerful and free)

Thus, proposed solution: **bring functionality of Praat into R by having R functions execute Praat scripts**

2. How it works



PraatR.r contains definition of core R function: praat()

- Pass Praat command as character: praat("To Formant", ...)

Inside praat(), check against SupportedCommands.txt:

- Determine command type: Create, Modify, Query, Play, etc.

Use shell() / system() to instruct operating system (OS) with:

- path to Praat, path to script, command name, arguments

OS invokes Praat with script, executes requested command:

- If Create or Modify, read input file, execute, save output file
- If Query, capture text written to info window and bring into R

Support for ~2000 pairings of ~100 objects + ~1000 commands

- Command names same as in Praat, reducing learning curve

3. Usage

At top of script, load in PraatR.r with one line of code:

```
source(paste(.libPaths(),"PraatR","PraatR.r",sep="/"))
```

Specify 'input' and 'output' file paths:

```
praat( "Down to PitchTier", input="C:/sound.Pitch",
       output="C:/sound.PitchTier" )
```

Defaults to not overwriting files; Can easily override if desired:

```
praat( "To SpectrumTier (peaks)",input="C:/sound.Ltas",
       output="C:/sound.SpectrumTier", overwrite=TRUE)
```

Specify arguments by passing as a list():

```
praat ("To Intensity...", input=___, output=___,
       arguments=list(100, 0, TRUE) )
```

Can save output as text file, short text file, or binary file:

```
praat( "Down to PointProcess", input=___, output=___,
       filetype="binary")
```

Can also modify existing file (replacing original by default):

```
praat( "Insert boundary...", input="C:/sound.TextGrid",
       arguments=list(1, 0.5) )
```

Query information from object directly into R:

```
x = praat( "Get mean...", input="sound.Formant",
           arguments=list(1,0,0,"Hertz") )
# [1] "492.38648475717173 Hertz"
```

Trim off all but core numeric information with 'simplify=TRUE':

```
x = praat("Get mean...", input="sound.Formant",
          arguments=list(1,0,0,"Hertz"),simplify=TRUE )
# [1] "492.38648475717173"
```

Can also use Praat to play audo:

```
praat( "Play", input="C:/sound.wav" )
```

Where to obtain

- Download from http://www.aaronalbin.com/praatr/
- Installation instructions; documentation of available commands
- Released under GNU General Public License
- Currently supported on Windows and Mac; Linux forthcoming

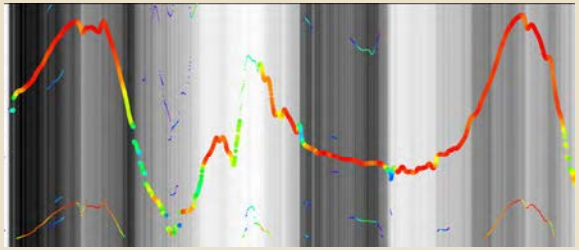
4. Visualization and interactivity

Visualization

Use R's powerful graphics to highlight relevant aspects of signal

- Can help to 'see' the quality of a given measurement

Example: rich visualization of F0+intensity ("Anna married Lenny")

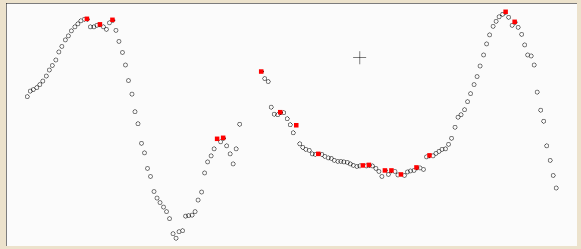


Interactivity

Use R's locator() and identify() functions or the 'tcltk' package

- Help speed annotation process (Human work minimized)

Example: selecting local F0 maxima (Same utterance as above)



5. Conclusion

User empowered to conduct entire analysis in single environment

- No need to manually shuttle data between programs
- More efficient, speeds up analysis

Relatively easy to innovate novel visualization / interactivity

- Building custom programs becomes accessible to many linguists
- Not contrained by one program; possibilities become infinite!