

# Implementing reproducibility in phonetic research: a computational workflow

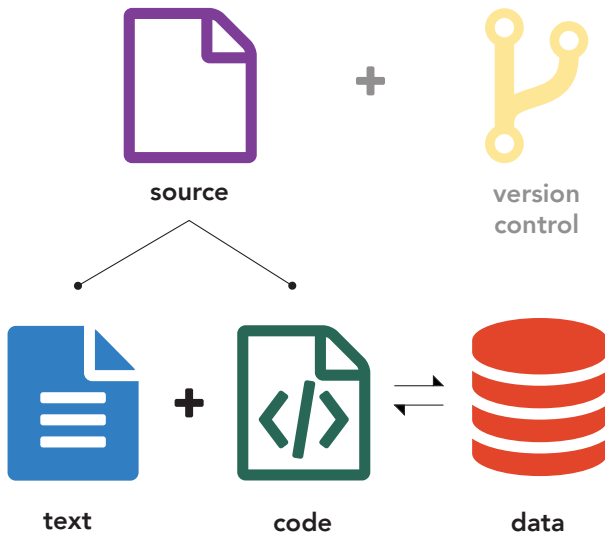
Stefano Coretta  
*University of Manchester*

**mFiL 2017**  
28 April 2017

# Reproducible research

A piece of research is **reproducible** when, along with its *results*, the *data* and the *computational environment* that produced those results are made available to other researchers (Fomel & Claerbout 2009).

# Reproducible research



# Why should we care?

The **problem** (Sandve et al. 2013):

- difficulty of reproduction
- difficulty of replication
- retracted papers (<http://retractionwatch.com>)

The “Yokuts vowels” case (Weigel 2002):

- about **75%** of the data is contrived (Weigel 2005:149)
- some of the generalisations are **wrong** (Blevins 2004)

The **solution**:

- **Reproducible Research** (RR)

# Reproducible Research in linguistics

- **linked data** (Bird & Simons 2003, Thieberger 2004)
- **computational grammar** (Maxwell & Amith 2005)
- RR in the Speech Sciences (Abari 2012)
  - lack of scientific culture
  - inefficiency of infrastructure

# The workflow of phonetic research

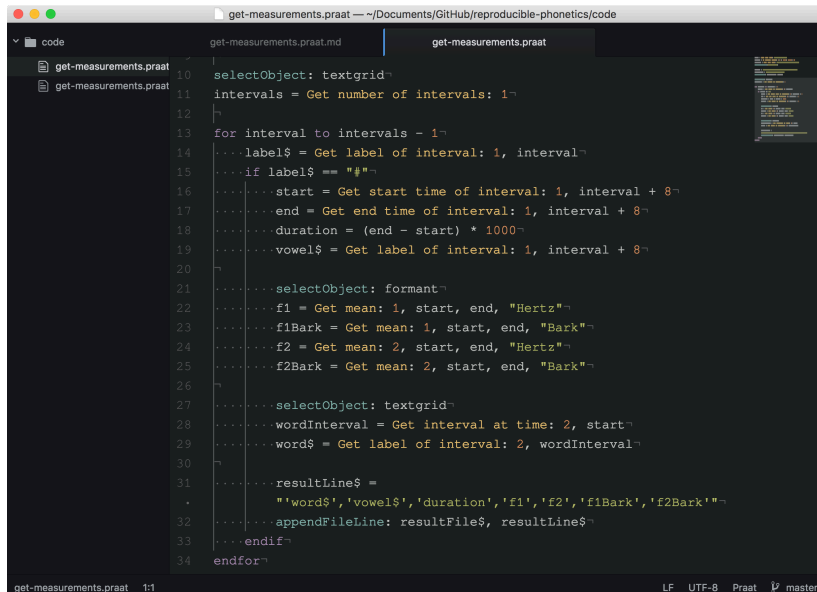
- **Phase A:** scripting (Praat, Boersma & Weenink (2016))
- **Phase B:** results and analysis
- **Phase C:** dissemination

# Phase A: source code and documentation

Praat scripting:

- Atom editor (<https://atom.io>)
  - syntax highlighting
  - autocompletion and snippets
- Literate Markdown
  - tangle: lmt (<https://github.com/driusan/lmt>)
  - weaving: pandoc (<http://pandoc.org>)

# Atom

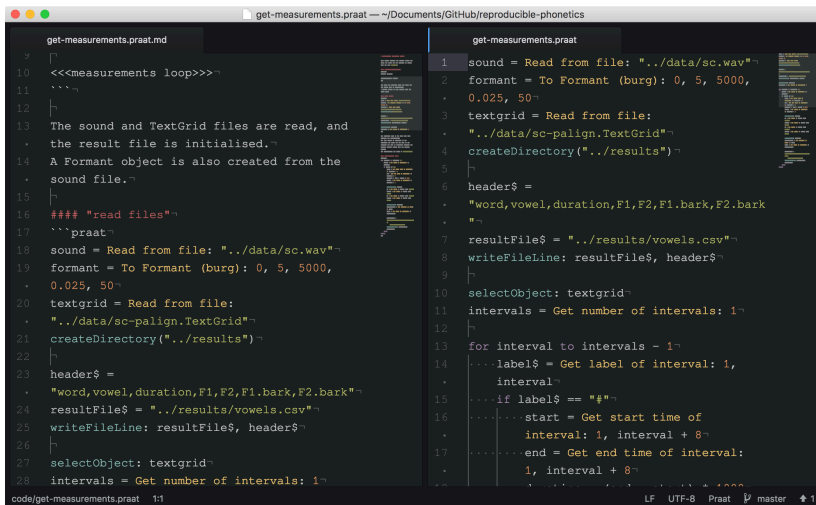


The screenshot shows the Atom text editor interface. The title bar indicates the file path: `get-measurements.praat — ~/Documents/GitHub/reproducible-phonetics/code`. The left sidebar shows a file explorer with a folder named `code` containing two files, both named `get-measurements.praat`. The main editor area displays the content of the selected file, which is a Praat script. The script is written in a dark theme with syntax highlighting. It defines a function `get-measurements.praat` that processes a textgrid and a formant file to generate a result line. The script includes comments in Chinese and uses Praat's `Get` and `selectObject` functions. The status bar at the bottom shows the file name `get-measurements.praat`, the line number `1:1`, and the encoding `LF UTF-8 Praat`. There is also a small icon for the master branch.

```
get-measurements.praat
10 selectObject: textgrid
11 intervals = Get number of intervals: 1
12
13 for interval to intervals - 1
14   ...label$ = Get label of interval: 1, interval
15   ...if label$ == "#"
16     ...start = Get start time of interval: 1, interval + 8
17     ...end = Get end time of interval: 1, interval + 8
18     ...duration = (end - start) * 1000
19     ...vowel$ = Get label of interval: 1, interval + 8
20   ...
21   ...selectObject: formant
22   ...f1 = Get mean: 1, start, end, "Hertz"
23   ...f1Bark = Get mean: 1, start, end, "Bark"
24   ...f2 = Get mean: 2, start, end, "Hertz"
25   ...f2Bark = Get mean: 2, start, end, "Bark"
26   ...
27   ...selectObject: textgrid
28   ...wordInterval = Get interval at time: 2, start
29   ...word$ = Get label of interval: 2, wordInterval
30   ...
31   ...resultLine$ =
32     ..."word$', 'vowel$', 'duration', 'f1', 'f2', 'f1Bark', 'f2Bark'"
33   ...appendFileLine: resultFile$, resultLine$
34   ...endif
35 endfor
```



# lmt (iterate markdown tangler)

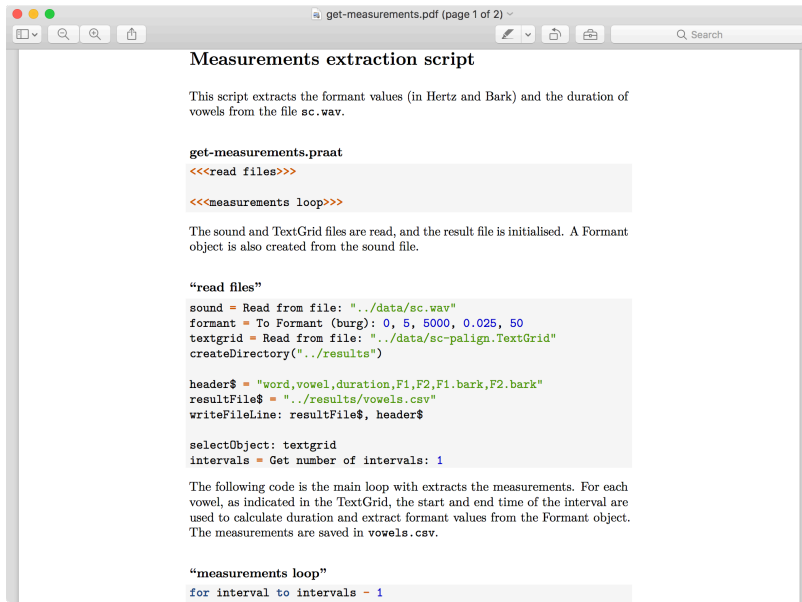


```
get-measurements.praat.md
10 <<<measurements loop>>>
11 ```
12 |
13 The sound and TextGrid files are read, and
14 the result file is initialised.
15 A Formant object is also created from the
16 sound file.
17 |
18 #### "read files"
19 ```praat
20 sound = Read from file: "../data/sc.wav"
21 formant = To Formant (burg): 0, 5, 5000,
22         0.025, 50
23 textgrid = Read from file:
24         "../data/sc-align.TextGrid"
25 createDirectory("../results")
26 |
27 header$ =
28         "word,vowel,duration,F1,F2,F1.bark,F2.bark"
29 resultFile$ = "../results/vowels.csv"
30 writeFileLine: resultFile$, header$
31 |
32 selectObject: textgrid
33 intervals = Get number of intervals: 1
```

```
get-measurements.praat
1 sound = Read from file: "../data/sc.wav"
2 formant = To Formant (burg): 0, 5, 5000,
3         0.025, 50
4 textgrid = Read from file:
5         "../data/sc-align.TextGrid"
6 createDirectory("../results")
7 |
8 header$ =
9         "word,vowel,duration,F1,F2,F1.bark,F2.bark"
10 resultFile$ = "../results/vowels.csv"
11 writeFileLine: resultFile$, header$
12 |
13 selectObject: textgrid
14 intervals = Get number of intervals: 1
15 for interval to intervals - 1
16     label$ = Get label of interval: 1,
17             interval
18     if label$ == "#"
19         start = Get start time of
20                 interval: 1, interval + 8
21         end = Get end time of interval:
22                1, interval + 8
23         ...
24     endfor
```

code/get-measurements.praat 1:1 LF UTF-8 Praat master ↗ 1

# pandoc (universal document converter)



The screenshot shows a PDF viewer window titled "get-measurements.pdf (page 1 of 2)". The document content is as follows:

## Measurements extraction script

This script extracts the formant values (in Hertz and Bark) and the duration of vowels from the file `sc.wav`.

**get-measurements.praat**

```
<<<read files>>>

<<<measurements loop>>>
```

The sound and TextGrid files are read, and the result file is initialised. A Formant object is also created from the sound file.

**“read files”**

```
sound = Read from file: "../data/sc.wav"
formant = To Formant (burg): 0, 5, 5000, 0.025, 50
textgrid = Read from file: "../data/sc-align.TextGrid"
createDirectory("../results")

header$ = "word,vowel,duration,F1,F2,F1.bark,F2.bark"
resultFile$ = "../results/vowels.csv"
writeFileLine: resultFile$, header$

selectObject: textgrid
intervals = Get number of intervals: 1
```

The following code is the main loop with extracts the measurements. For each vowel, as indicated in the TextGrid, the start and end time of the interval are used to calculate duration and extract formant values from the Formant object. The measurements are saved in `vowels.csv`.

**“measurements loop”**

```
for interval to intervals - 1
```

## Phase B: the speakr package

speakr is an R (R Core Team 2015) package to aid Praat users (under development):

- aim: tangle and run Praat scripts from within R
- two main functions
  - `lmt()`: tangle a Praat script
  - `praatRun()`: run a Praat script

## Phase B: the speakr package

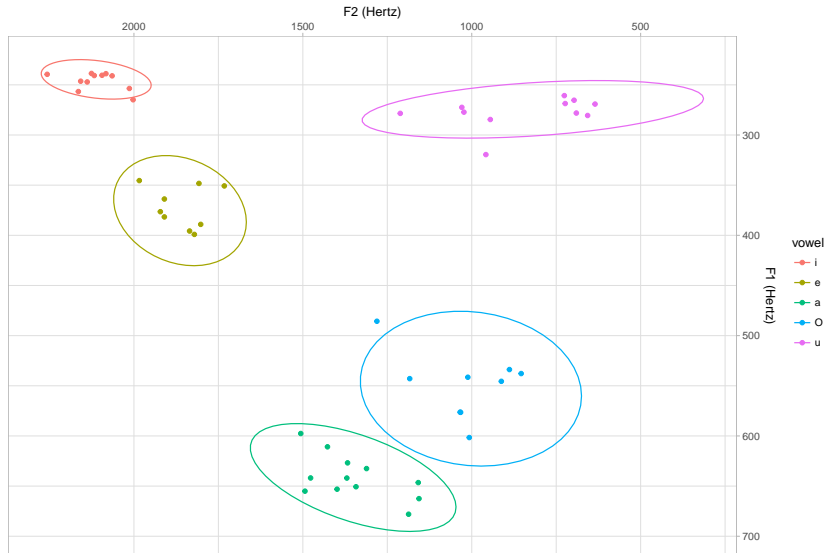
```
# Tangle a Praat script
lmt("code/get-measurements.praat.md")

# Run the script
praatRun("code/get-measurements.praat")

# Read the results of the script
vowels <- read_csv("results/vowels.csv") %>%
  mutate_if(is.character, as.factor) %>%
  mutate(vowel = factor(vowel, c("i", "e", "a",
                                "O", "u")))
```

# Phase B: the speakr package

Vowel plot of one speaker of Italian



# Phase C: dissemination

*There is no investigation without dissemination.*

Ricardo Bermúdez-Otero (p.c.)

- knitr (Xie 2014)
  - dynamic reports
  - reproducible documents
- GitHub (<https://github.com>)
  - versioning system (git)
  - online repository
- Open Science Framework (<https://osf.io>)
  - online repository (for data)

# Summary

- share data, source file(s), versioning
- increasing awareness of RR in linguistics
- Atom, lmt, pandoc, speakr, knitr
- this presentation (along with source code and data) is available at <https://github.com/stefanocoretta/reproducible-phonetics>

## Summary

THANK YOU!



# References I

- Abari, Kálmán. 2012. Reproducible research in speech sciences. *International Journal of Computer Science Issues* 9(6). 43–52.
- Bird, Steven & Gary Simons. 2003. Seven dimensions of portability for language documentation and description. *Language* 557–582.
- Blevins, Juliette. 2004. A reconsideration of Yokuts vowels. *International Journal of American Linguistics* 70(1). 33–51.
- Boersma, Paul & David Weenink. 2016. Praat: doing phonetics by computer [Computer program]. Version 6.0.23. <http://www.praat.org/>.
- Fomel, Sergey & Jon Claerbout. 2009. Guest editors' introduction: Reproducible research. *Computing in Science and Engineering* 11(1). 5–7.

## References II

- Maxwell, Michael & Jonathan D. Amith. 2005. Language documentation: the Nahuatl grammar. In A. Gelbukh (ed.), *Computational Linguistics and Intelligent Text Processing*, 474–485. Berlin Heidelberg: Springer-Verlag.
- R Core Team. 2015. R: A language and environment for statistical computing. <https://www.R-project.org>.
- Sandve, Geir Kjetil, Anton Nekrutenko, James Taylor & Eivind Hovig. 2013. Ten simple rules for reproducible computational research. *PLoS Computational Biology* 9(10). 1–4.
- Thieberger, Nicholas. 2004. Documentation in practice: Developing a linked media corpus of South Efate. In Peter K. Austin (ed.), *Language documenta and description*, vol. 2, Hans Rausing Endangered Languages Project, School of Oriental and African Studies, University of London.

## References III

- Weigel, William. 2005. *Yowlumne in the Twentieth century*: University of California, Berkley dissertation.
- Weigel, William F. 2002. The Yokuts canon: A case study in the interaction of theory and description. Paper presented at the annual meeting of the Linguistics Society of America, January 2002, San Francisco.
- Xie, Yihui. 2014. knitr: A comprehensive tool for reproducible research in R. In Victoria Stodden, Friedrich Leisch & Roger D. Peng (eds.), *Implementing reproducible computational research*, Chapman and Hall: CRC.