A Course in the EMU-SDMS

2017-10-08

Contents

1 Prerequisites **5**
1.1 Installing the EMU-SDMS 5
1.2 Installing additional R-packages that are used throughout this course 5

2 Introduction **7**

3 An overview of the EMU-SDMS **9**

4 Toolchain SpeechRecorder — MAUS — EMU-SDMS **11**
4.1 What do SpeechRecorder and MAUS do? 11
4.2 Different types of prompts 11
4.3 Combining the tools 12

5 The EMU-webApp **15**
5.1 Example one 15
5.2 Example two 15

6 Creating segment lists and trackdata objects **17**

7 Wrassp and the application of DSP **19**

8 Annotation types and the Emu Query-language **21**

Chapter 1

Prerequisites

(copy and pasted from EMU manual) A prerequisite that is presumed throughout this document is the reader's familiarity with basic terminology in the speech sciences (e.g., familiarity with the international phonetic alphabet and how speech is annotated at a coarse and fine grained level). Further, we assume the reader has a grasp of the basic concepts of the R language and environment for statistical computing and graphics. For readers new to R, there are multiple, freely available R tutorials online (e.g., Statistical Analysis: an Introduction using R/R_basics. R also has a set of very detailed manuals and tutorials that come preinstalled with R. To be able to access R's own "An Introduction to R" introduction, simply type `help.start()` into the R console and click on the link to the tutorial.

1.1 Installing the EMU-SDMS

(copy and pasted from EMU manual)

1.) R

- Download the R programming language from http://www.cran.r-project.org
- Install the R programming language by executing the downloaded file and following the on-screen instructions.

2.) emuR

- Start up R.
- Enter `install.packages("emuR")` after the `>` prompt to install the package. (You will only need to repeat this if package updates become available.)
- As the `wrassp` package is a dependency of the `emuR` package, it does not have to be installed separately.

3.) EMU-webApp (prerequisite)

- The only thing needed to use the EMU-webApp is a current HTML5 compatible browser (Chrome/Firefox/Safari/Opera/...). However, as most of the development and testing is done using Chrome we recommend using it, as it is by far the best tested browser.

1.2 Installing additional R-packages that are used throughout this course

- ggplot2 for visualization purposes: `install.packages("emuR")`
- dplyr for data manipulation: `install.packages("dplyr")`

Chapter 2

Introduction

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter 2. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter **??**.

Figures and tables with captions will be placed in `figure` and `table` environments, respectively.

```
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 2.1. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 2.1.

```
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

You can write citations, too. For example, we are using the **bookdown** package (Xie, 2017) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015).

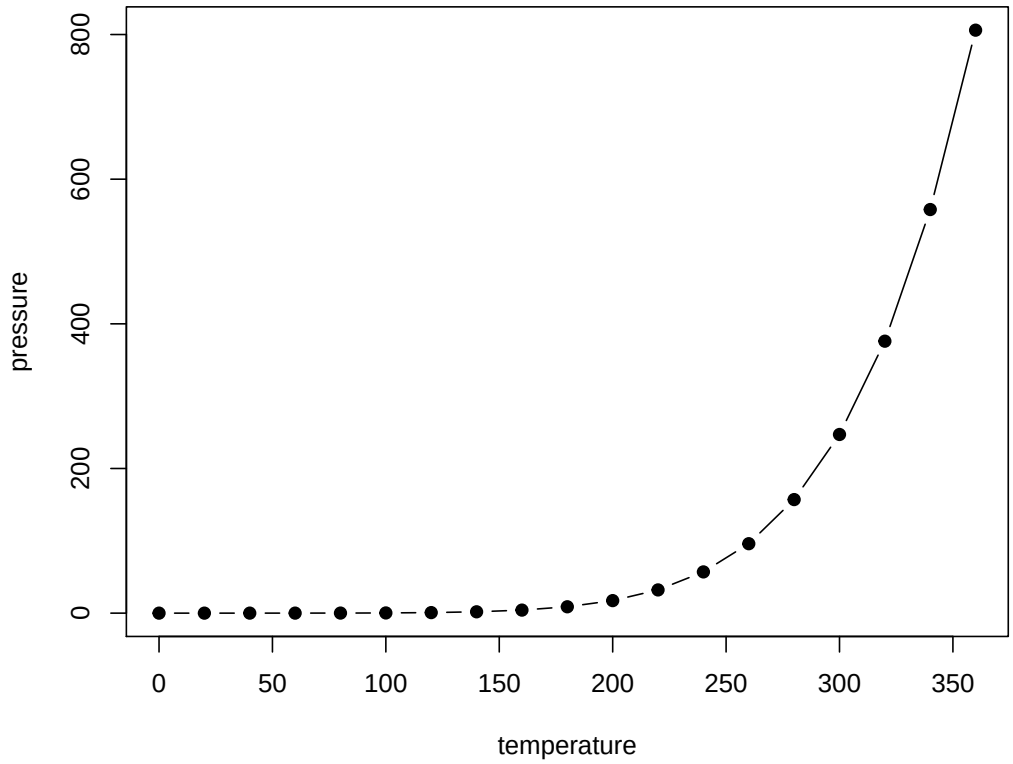

Figure 2.1: Here is a nice figure!

Table 2.1: Here is a nice table!

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa

Chapter 3

An overview of the EMU-SDMS

Here is a review of existing methods.

Chapter 4

Toolchain SpeechRecorder — MAUS — EMU-SDMS

Most phonetic research projects involve this workflow:

1. Record speech
2. Annotate speech using automatic tools
3. Check and correct the generated annotations by hand
4. Analyze speech (connecting the primary data with annotations and derived signals)

The EMU Speech Database Management System is focused on steps 3 and 4 of this workflow. For the first two steps, it can very usefully be complemented by two other tools: SpeechRecorder and MAUS (or, more broadly speaking — and more correctly, for that matter — the BAS Web Services). This chapter introduces how the tools can be combined — in a systematic way, with as little fuss as possible.

4.1 What do SpeechRecorder and MAUS do?

"SpeechRecorder is a platform independent audio recording software customized to the requirements of speech recordings" (http://www.speechrecorder.org/). To this end, SpeechRecorder lets you define prompts that participants will read (or otherwise react to) while you are recording them. At the end of a session, instead of one large recording of the whole session, you have a set of smaller audio recordings, each one representing a single prompt.

MAUS (Munich AUtomatic Segmentation) processes audio recordings with corresponding orthographic transcriptions, and outputs (1) a corresponding phonetic transcription and (2) a segmentation of the signal into individual speech sounds. As such, MAUS is a part of the BAS Web Services. [1]

4.2 Different types of prompts

Prompts are very often sentences that participants read out aloud (producing *read speech*). However, this need not be the case. Prompts may as well be, for example, images that participants have to name or describe, or written questions that participants answer.

[1] Strictly speaking, MAUS is used in conjunction with G2P and possibly Chunker to achieve this result. The whole package is often referred to as MAUS, however.

In terms of processing, *read speech* has the advantage that the researcher already has orthographic transcriptions of all recordings, because the prompts *are* the transcriptions (this neglects hesitations, misread utterances, etc.).

For all cases besides read speech, transcriptions have to be prepared. For read speech, when hesitations etc. need to be considered, the existing transcriptions need to be corrected (per token). For the time being, this is out of the scope of this book.[2]

4.3 Combining the tools

The order of the processing steps is unchangeable: The files have to be recorded first, then annotated and then analyzed.

However, there are different ways of passing data around between the tools. For example SpeechRecorder might *export* files into emuDB format or the EMU-SDMS might *import* files stored in SpeechRecorder's format. After all, they are separate tools and can be used individually (although the combination makes great sense).

Moreover, sometimes a database grows, and some new files are recorded while others have already been annotated or analyzed (which challenges the "unchangeable order of processing steps").

We can see now that we have to convert our data between different formats. We want to benefit from all tools but keep the conversion work down to a minimum. In this section, we explain the (currently) best way to do this. We will import SpeechRecorder's recordings and prompts into the EMU-SDMS and then use emuR to send the data to MAUS and other BAS Web Services.

4.3.1 Importing recordings and transcriptions into Emu

4.3.1.1 Using the import_speechRecorder() function

The `import_speechRecorder()` function is in the making, but unfortunately not finished yet. We therefore have to resort to the second-best way of importing SpeechRecorder's results into Emu:

4.3.1.2 Using intermediary text files

SpeechRecorder, per default, saves one .wav file for each prompt. With additional settings, it will save a .txt file containing the corresponding prompt along with each .wav file. This is great because it is exactly what MAUS needs as its input.

The following options must be configured *before recordings are made*. Note that parts of SpeechRecorder's user interface are in German:

- Under "Projekt / Einstellungen… / Annotation":
  - Under "Persist", tick the checkbox that says "Simple text loader writer for annotation template for MAUS processing"
  - Under "Auto annotation", tick the checkbox that says "Prompt template auto annotator" (note that the two checkboxes are very similar. Make sure to tick the right one.)
- In your script:
  - If you edit the XML file directly: Make sure each of your `<mediaitem>` elements has the attribute `annotationTemplate="true"`
  - If you use the integrated script editor: Make sure to tick the checkbox "Use as annotation template" for *every recording*.

[2] It will be covered, at a later time, in this or a separate chapter.

Now, after your recording session, you will have audio and text files. In emuR, this combination is called a *txt collection*. We will thus use the function `convert_txtCollection`, to import SpeechRecorder's files into Emu's format. In the following example, we will use the sample txt collection included with the emuR package.

```
# Load the emuR package
library(emuR)

# Create demo data in directory provided by tempdir()
create_emuRdemoData(dir = tempdir())

# Import the sample txt collection.
# When used with real data, sourceDir should point to the RECS directory of your
# SpeechRecorder project.
convert_txtCollection(dbName = "myEmuDatabase",
                      sourceDir = file.path(tempdir(), "emuR_demoData", "txt_collection"),
                      targetDir = tempdir())

dbHandle = load_emuDB(file.path(tempdir(), "myEmuDatabase_emuDB"))
```

Now, you have an Emu database. You can inspect it using

```
summary(dbHandle)
```

or

```
serve(dbHandle)
```

The database contains all our recordings, and exactly one type of annotation: orthographic transcription. No phonetic transcription, no segmentation. The next section will cover that.

4.3.2 Feeding the data into MAUS

The .wav and .txt files could have been uploaded on the BAS web site, but we will do it using emuR. This is generally less error-prone and requires less manual work. Moreover, since it is a scripted way of doing things, we can reproduce it reliably.

To process the data, we make use of several of emuR's functions called `runBASwebservice_...`. They will upload the data to WebMAUS and accompanying services and save the results directly inside your existing emuDB (this of course takes some time, depending on the size of your database and the speed of your internet connection).

```
runBASwebservice_g2pForTokenization(handle = dbHandle,
                                    language = "eng-US",
                                    transcriptionAttributeDefinitionName = "transcription",
                                    orthoAttributeDefinitionName = "Word")

runBASwebservice_g2pForPronunciation(handle = dbHandle,
                                     language = "eng-US",
                                     orthoAttributeDefinitionName = "Word",
                                     canoAttributeDefinitionName = "Canonical")

runBASwebservice_maus(handle = dbHandle,
                      language = "eng-US",
                      canoAttributeDefinitionName = "Canonical",
                      mausAttributeDefinitionName = "Phonetic")
```

When the services are finished, we can use

```
serve(db)
```

to inspect the database with the new annotations. This time, it includes segmentation into words and phonemes, canonical phonetic transcription and realized phonetic transcription.

This chapter described the current best practice of combining SpeechRecorder, MAUS and the EMU-SDMS to fit a typical phonetic project workflow.

Chapter 5

The EMU-webApp

Some *significant* applications are demonstrated in this chapter.

5.1 Example one

5.2 Example two

Chapter 6

Creating segment lists and trackdata objects

We have finished a nice book.

Chapter 7

Wrassp and the application of DSP

We have finished a nice book.

Chapter 8

Annotation types and the Emu Query-language

We have finished a nice book.

Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2017). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.5.