**Appendix C: Commands for creating the Emu-R datasets.**

These commands can be used to recreate the R datasets referred to in this book assuming that the relevant databases have been downloaded (use Arrange tools -> DB Installer from within Emu). You will need to specify a path to which segment lists are to be written and from which trackdata objects are to be read as follows (for example "something" might be "c:/mydata" on Windows):

```
path = "something"
```

where the path goes between the double quotes (for example "something" might be "c:/mydata" on Windows).

## C.1 Database: andosl, dataset: keng

Segment list of the aspiration dominated by syllable-initial /k/ that precedes three front vowels

```
front=emu.query("andosl", "*",
"[[Phoneme = k & Start(Syllable, Phoneme)=1 ^ Phonetic=H] -> Phoneme=i: | I | E]")
```

Segment list of the aspiration dominated by syllable-initial /k/ that precedes two back vowels

```
back =  emu.query("andosl", "*",
"[[Phoneme = k & Start(Syllable, Phoneme)=1 ^ Phonetic=H] -> Phoneme=o: | U]")
```

Bind the two segment lists into one

```
keng = rbind(front,back)
```

Make a parallel vector of labels

```
front.l = rep("front", nrow(front))
back.l = rep("back", nrow(back))
keng.l  = c(front.l, back.l)
```

Write out the segment list

```
write.emusegs(keng, paste(path, "keng.txt", sep="/"))
```

Read the segment list into EMU-tkassp and calculate spectral data using a 512 point DFT with defaults for the other parameters. Store the spectral data in the same directory to which the segment list has been written.

Then read the spectral data into R. The sampling frequency of the audio file needs to be specified and the object made into a spectral trackdata object.

```
keng.dft = read.trackdata(paste(path, "keng-dft.txt", sep="/"))
keng.dft = as.spectral(keng.dft, 20000)
```

Spectral matrix at the segment midpoint

```
keng.dft.5 = dcut(keng.dft, .5, prop=T)
```

## C.2 Databases: andosl and kielread, dataset: geraus

Function to check if a trackdata has any zero values

```
dfun <- function(tdat)
{
any(tdat==0)
}
```

Segment list of /i:/ in strong syllables, speaker JC
```
auseng.i = emu.query("andosl", "*jc*", "[Phoneme=i: ^ Syllable=S]")
```

Trackdata of the corresponding formants
```
auseng.fm = emu.track(auseng.i, "fm")
```

A vector of any segments of trackdata containing zero values
```
zeros = trapply(auseng.fm[,2], dfun, simplify=T)
```

Remove the above
```
auseng.fm = auseng.fm[!zeros,]
```

Vector of annotations
```
aus.l = rep("aus", nrow(auseng.fm))
```

The next commands are the same as above except using the `kielread` database and for speaker 67
```
ger.i = emu.query("kielread", "*67*", "Kanonic=i:")
ger.fm = emu.track(ger.i, "fm")
zeros = trapply(ger.fm[,2], dfun, simplify=T)
ger.fm = ger.fm[!zeros,]
ger.l = rep("ger", nrow(ger.fm))
```

Make a single trackdata object of F2 for the Australian English and German data and a parallel vector of labels
```
f2geraus = rbind(auseng.fm[,2], ger.fm[,2])
f2geraus.l = c(aus.l, ger.l)
```

**C.3 Database: epgassim, dataset: engassim**
A segment list of a sequence of segments /nk/, /ng/, /sk/, /sg/
```
engassim  = emu.query("epgassim", "*", "[Segment=n|s -> Segment=k|g]")
```

As above but consisting only of the first segment /n/ or /s/
```
s.n = emu.query("epgassim", "*", "[Segment=n|s -> Segment=k|g]")
```

As the penultimate command, but consisting only of the second segment /k/ or /g/
```
k.n = emu.query("epgassim", "*", "[Segment=n|s -> Segment=k|g]")
```

Make a parallel vector of labels /sK/ or /nK/ (where /K/ is collapsed across  /k/ or /g/)
```
engassim.l = paste(label(s.n), rep("K", nrow(k.n)), sep="")
```

A vector of the corresponding word labels
```
engassim.w = emu.requery(s.n, "Segment", "Word", j=T)
```

EPG trackdata
```
engassim.epg = emu.track(engassim, "epg")
```

**C.4 Database: epgcoutts, datasets: coutts, coutts2**
Segment list of all words in utterance `u1` and from fast speech

```
coutts = emu.query("epgcoutts", "spstoryfast01", "[Word!=x ^ Utterance=u1]")

EPG-trackdata
coutts.epg = emu.track(coutts, "epg")

Trackdata of audio waveform
coutts.sam = emu.track(coutts, "samples")

Vector of word labels
coutts.l = label(coutts)

The next commands are the same as above but for slow speech
coutts2 = emu.query("epgcoutts", "spstoryslow01", "[Word!=x ^ Utterance=u1]")
coutts2.epg = emu.track(coutts2, "epg")
coutts2.sam = emu.track(coutts2, "samples")
coutts2.l = label(coutts2)
```

**C.5 Database: epgdorsal, dataset: dorsal**

```
Segment list (vowels), speaker JD of any segment preceding /k/
dorsala = emu.query("epgdorsal", "JD*", "[Phonetic != xx -> Phonetic= k]")

Vector of labels
dorsala.l = label(dorsala)

The same segment listed sorted on the labels
sa = sort.list(dorsala.l)
dorsala = dorsala[sa,]

The same as the above commands but for all segments preceding /x/
dorsalb = emu.query("epgdorsal", "JD*", "[Phonetic != xx -> Phonetic= x]")
dorsalb.l = label(dorsalb)
sb = sort.list(dorsalb.l)
dorsalb = dorsalb[sb,]

The two vowel segment lists bound together in a single segment list
dorsal = rbind(dorsala, dorsalb)

A segment list of the following /k/ or /x/ segments
dorsalk = emu.requery(dorsal, "Phonetic", "Phonetic", sequence=1)

Vector of start times of /k/ or /x/
dorsal.bound = start(dorsalk)

Word-labels corresponding to /k/ or /x/
dorsal.w = emu.requery(dorsalk, "Phonetic", "Word", justlabels=T)
```

Reset the end times of dorsal to those of dorsalk – i.e. the new segment list dorsal extends from the onset of the vowel to the offset of the following /k/ or /x/

```
dorsal[,3] = dorsalk[,3]
```

Make new labels for the segment list dorsal by appending /k/ or /x/
```
dorsal[,1] = paste(label(dorsal), label(dorsalk), sep="")
dorsal.l = label(dorsal)
```

A vector of vowel labels
```
dorsal.vlab = substring(dorsal.l, 1, 1)
```

A vector of /k/ or /x/ labels
```
dorsal.clab = substring(dorsal.l, 2, 2)
```

EPG trackdata
```
dorsal.epg = emu.track(dorsal, "epg")
```

Trackdata of formants
```
dorsal.fm = emu.track(dorsal, "fm")
```

Trackdata of audio waveform
```
dorsal.sam = emu.track(dorsal, "samples")
```

## C.6 Database: epgpolish, dataset polhom

Four segment lists, one for each of the four homorganic fricatives
```
polhom.s  = emu.query("epgpolish", "*", "[Segment=s -> Segment=s]")
polhom.S  = emu.query("epgpolish", "*", "[Segment=S -> Segment=S]")
polhom.c  = emu.query("epgpolish", "*", "[Segment=c -> Segment=c]")
polhom.x  = emu.query("epgpolish", "*", "[Segment=x -> Segment=x]")
```

The four segment lists bound into a single segment list
```
polhom = rbind(polhom.s, polhom.S, polhom.c, polhom.x)
```

A vector of annotations
```
polhom.l = substring(label(polhom), 1, 1)
```

EPG trackdata
```
polhom.epg = emu.track(polhom, "epg")
```

## C.7 Database: gerplosives, datasets plos and stops10

Segment list of word-initial /b/, /d/
```
plos = emu.query("gerplosives", "*", "[Phoneme=b|d & Start(Word, Phoneme)=1]")
```

A vector of the corresponding word labels
```
plos.w = emu.requery(plos, "Phoneme", "Word", justlabels=T)
```

A vector of /b/ or /d/ segment labels
```
plos.l = label(plos)
```

A vector of the corresponding following segment (vowel) labels
```
plos.lv = emu.requery(plos, "Phoneme", "Phoneme", sequence=1,  justlabels=T)
```

A segment list of the corresponding burst

```
plos.asp = start(emu.query("gerplosives", "*", "[Phonetic=H ^ Phoneme=b|d & Start(Word, Phoneme)=1]"))
```

Spectral trackdata of word-initial /b/, /d/

```
plos.dft = emu.track(plos, "dft")
```

Trackdata of the audio waveform

```
plos.sam = emu.track(plos, "samples")
```

Segment list of word-initial /g/

```
plosg = emu.query("gerplosives", "*", "[Phoneme=g & Start(Word, Phoneme)=1]")
```

A vector of the corresponding word annotations

```
plosg.w = emu.requery(plosg, "Phoneme", "Word", justlabels=T)
```

A vector of the corresponding phoneme annotations

```
plosg.l = label(plosg)
```

A vector of annotations of the following vowel

```
plosg.lv = emu.requery(plosg, "Phoneme", "Phoneme", sequence=1,  justlabels=T)
```

Vector of start times of the word-initial /g/-burst

```
plosg.asp = start(emu.query("gerplosives", "*", "[Phonetic=H ^ Phoneme=g & Start(Word, Phoneme)=1]"))
```

Write out the segment list of /g/

```
write.emusegs(plosg, paste(path, "plosg.txt", sep="/"))
```

Read the segment list into EMU-tkassp and calculate spectral data using a 256 point DFT with defaults for the other parameters. Store the spectral data in the same directory to which the segment list has been written.

Then read the spectral data into R. The sampling frequency of the audio file needs to be specified and the object made into a spectral trackdata object.

```
plosg.dft = read.trackdata(paste(path, "plosg-dft.txt", sep="/"))
plosg.dft = as.spectral(plosg.dft, 16000)
```

Get the spectral data 10 ms after the onset of the /g/-burst

```
afterg = dcut(plosg.dft, plosg.asp+10)
```

Get the spectral data 10 ms after the onset of the /b/ and /d/ bursts

```
after = dcut(plos.dft, plos.asp+10)
```

Make a matrix of these /b, d, g/ spectral data

```
stops10 = rbind(after, afterg)
stops10 = as.spectral(stops10, 16000)
```

Make a corresponding vector of annotations

```
stops10.lab = c(plos.l, rep("g", nrow(afterg)))
```

## C.8 Database: kielread, datasets dip, dorfric, fric, sib, vowlax

Segment list of three diphthongs

```
dip  = emu.query("kielread", "*", "[Kanonic=aI | aU | OY]")
```

Vector of speaker annotations

```
dip.spkr = substring(utt(dip), 2, 3)
```

Vector of phonetic annotations

```
dip.l = label(dip)
```

Trackdata of formants

```
dip.fdat = emu.track(dip, "fm")
```

Segment list of /ç, x/ following various vowels

```
dorfric = emu.query("kielread",
"K68*", "[ [ Kanonic = a | a: | E | I | O | o: | u:] -> Kanonic = C | x ]")
```

Corresponding vector of annotations

```
dorfric.l = label(dorfric)
```

Corresponding vector of following (vowel) annotations

```
dorfric.lv = emu.requery(dorfric, "Kanonic", "Kanonic", seq=-1, j=T)
```

Corresponding vector of word labels

```
dorfric.w = emu.requery(dorfric, "Kanonic", "Word", justlabels=T)
```

Reset all /a/ vowel labels to /a:/

```
dorfric.lv[dorfric.lv=="a"] = "a:"
```

Write out the segment list

```
write.emusegs(dorfric, paste(path, "dorfric.txt", sep="/"))
```

Read the segment list into EMU-tkassp and calculate spectral data using a 256 point DFT with defaults for the other parameters. Store the spectral data in the same directory to which the segment list has been written.

Then read the spectral data into R. The sampling frequency of the audio file needs to be specified and the object made into a spectral trackdata object.

```
dorfric.dft = read.trackdata(paste(path, "dorfric-dft.txt", sep="/"))
dorfric.dft = as.spectral(dorfric.dft, 16000)
```

Segment list of intervocalic and word-medial /s, z/

```
fric = emu.query("kielread", "K67*",
"[ [ Kanonic = vowel -> Kanonic = s | z & Medial ( Word,Kanonic ) = 1 ] -> Kanonic = vowel
]")
```

Corresponding vector of word labels

```
fric.w = emu.requery(fric, "Kanonic", "Word", justlabels=T)
```

Corresponding vector of segment labels

```
fric.l = label(fric)
```

Write out the segment list
```
write.emusegs(fric, paste(path, "fric.txt", sep="/"))
```

Read the segment list into EMU-tkassp and calculate spectral data using a 256 point DFT with defaults for the other parameters. Store the spectral data in the same directory to which the segment list has been written.

Then read the spectral data into R. The sampling frequency of the audio file needs to be specified and the object made into a spectral trackdata object.
```
fric.dft = read.trackdata(paste(path, "fric-dft.txt", sep="/"))
fric.dft = as.spectral(fric.dft, 16000)
```

Segment list of syllable-initial /z/ preceding two back rounded vowels
```
back = emu.query("kielread", "*",
"[ Kanonic =  z  & Start ( Syllable,Kanonic ) = 1 -> Kanonic = u:|U ]")
```

Segment list of syllable-initial /z/ preceding two front unrounded vowels
```
front  = emu.query("kielread", "K67*",
"[ Kanonic =  z  & Start ( Syllable,Kanonic ) = 1 -> Kanonic = i:|I ]")
```

A single segment list of the above lists bound together
```
sib = rbind(front, back)
```

A vector of labels indicating front or back
```
sib.l = c(rep("f", nrow(front)), rep("b", nrow(back)))
```

Write out the segment list
```
write.emusegs(sib, paste(path, "sib.txt", sep="/"))
```

Read the segment list into EMU-tkassp and calculate spectral data using a 256 point DFT with defaults for the other parameters. Store the spectral data in the same directory to which the segment list has been written.

Then read the spectral data into R. The sampling frequency of the audio file needs to be specified and the object made into a spectral trackdata object.
```
sib.dft = read.trackdata(paste(path, "sib-dft.txt", sep="/"))
sib.dft = as.spectral(sib.dft, 16000)
```

Segment list of four lax vowels
```
vowlax = emu.query("kielread", "*", "Kanonic=a | E | I | O")
```

Speaker labels
```
vowlax.spkr = substring(utt(vowlax), 2, 3)
```

Phonetic labels
```
vowlax.l = label(vowlax)
```

Word labels
```
vowlax.word = emu.requery(vowlax, "Kanonic", "Word", justlabels=T)
```

Trackdata, formants

```
vowlax.fdat = emu.track(vowlax, "fm")
```

F1-F4 at the segment midpoint

```
vowlax.fdat.5 = dcut(vowlax.fdat, .5, prop=T)
```

Write out the segment list

```
write.emusegs(vowlax, paste(path, "vowlax.txt", sep="/"))
```

Read the segment list into EMU-tkassp and calculate spectral data using a 256 point DFT, f0 and dB-RMS with defaults for all other parameters. Store the spectral data in the same directory to which the segment list has been written.

Then read the spectral data into R. The sampling frequency of the audio file needs to be specified and the object made into a spectral trackdata object.

```
vowlax.fund = read.trackdata(paste(path, "vowlax-f0.txt", sep="/"))
vowlax.rms = read.trackdata(paste(path, "vowlax-rms.txt", sep="/"))
vowlax.dft = read.trackdata(paste(path, "vowlax-dft.txt", sep="/"))
vowlax.dft = as.spectral(vowlax.dft, 16000)
```

f0, dB-RMS, and spectral data at the temporal midpoint

```
vowlax.fund.5 = dcut(vowlax.fund, .5, prop=T)
vowlax.rms.5 = dcut(vowlax.rms, .5, prop=T)
vowlax.dft.5 = dcut(vowlax.dft, 0.5, prop=T)
```

Data frame of F1-F3, f0, dB-RMS all at the temporal midpoint, segment duration, phonetic, word, and speaker labels

```
vowlax.df = data.frame(F1=vowlax.fdat.5[,1], F2=vowlax.fdat.5[,2], F3=vowlax.fdat.5[,3],
f0=vowlax.fund.5, rms=vowlax.rms.5, dur=end(vowlax)-start(vowlax), phonetic=vowlax.l,
word=vowlax.word, speaker=vowlax.spkr)
```

## C.9 Database: isolated, dataset: isol

Segment list, word-initial /d/

```
stop.d = emu.query("isolated", "*jc*", "[Phoneme=d & Start(Word, Phoneme)=1]")
```

Segment list of the following vowel

```
isol = emu.requery(stop.d, "Phoneme", "Phoneme", sequence=1)
```

Formant trackdata of the vowels

```
isol.fdat = emu.track(isol, "fm")
```

Vector of vowel labels

```
isol.l = label(isol)
```

Select on various diphthongs and create the corresponding trackdata object and label vector just for these

```
m = c("@u", "au", "oi", "i@", "ei", "ai")
temp = isol.l %in% m
isol = isol[!temp,]
isol.l = isol.l[!temp]
isol.fdat = isol.fdat[!temp,]
```

## C.10 Database: timetable, dataset timevow

Segment list of three vowels

```
timevow = emu.query("timetable", "*", "Phonetic=I | U | a")
```

Vector of their labels

```
timevow.l = label(timevow)
```

Write out the segment list

```
write.emusegs(timevow, paste(path, "timevow.txt", sep="/"))
```

Read the segment list into EMU-tkassp and calculate spectral data using a 256 point DFT and formants with defaults for the other parameters. Store the spectral data and formants in the same directory to which the segment list has been written.

Then read the data into R. The sampling frequency of the audio file needs to be specified and the object made into a spectral trackdata object.

```
timevow.fm = read.trackdata(paste(path, "timevow-fms.txt", sep="/"))
timevow.dft = read.trackdata(paste(path, "timevow-dft.txt", sep="/"))
timevow.dft = as.spectral(timevow.dft, 16000)
```

F1-F4 at the temporal midpoint

```
timevow.fm5 = dcut(timevow.fm, .5, prop=T)
```

## C.11 Database: stops, dataset stops

Word-initial stops

```
stops = emu.query("stops", "*", "[Phoneme= b | d | g & Start(Word, Phoneme) = 1 ^ Phonetic= H]")
```

Their labels

```
stops.l = label(stops)
```

The burst of the stops

```
asp = emu.query("stops", "*", "[Phonetic = H ^ Phoneme= b | d | g & Start(Word, Phoneme) = 1]")
```

The following vowel

```
stopsvow = emu.requery(asp, "Phonetic", "Phonetic", sequence=1)
```

Their formants

```
stopsvow.fm = emu.track(stopsvow, "fm")
```

An utterance identifier

```
stops.sp = substring(utt(stops), 1, 3)
```

The vowel labels

```
stopsvow.l = label(stopsvow)
```

Calculate spectra in tkassp for the entire database using the defaults. Readjust the template file so that the stops database can find the dft files.

```
stops.dft = emu.track(asp, "dft")
```

Bark spectra in the range 200-7800 Hz
```
stops.bark = bark(stops.dft[,200:7800])
```

DCT-0, -1, and -2 of the Bark spectra
```
stops.dct = fapply(stops.bark, dct, 2)
```