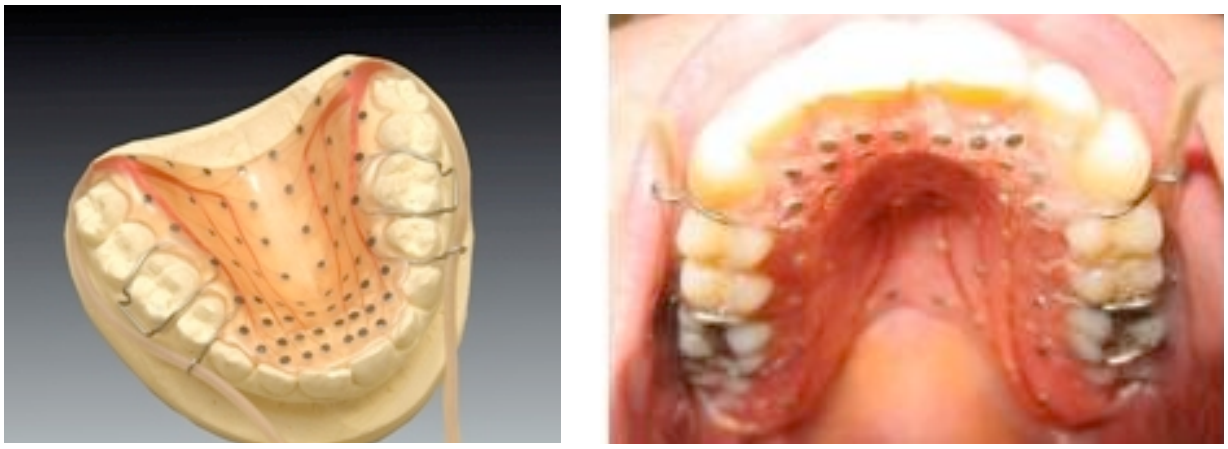
Chapter 7. Electropalatography

7.1. Palatography and electropalatography

Palatography is the general term given to the experimental technique for obtaining records of where the tongue makes a contact with the roof of the mouth. The earliest types of palatographic techniques were static allowing recordings to be made of a single consonant typically produced between vowels. In static palatography, which is still very useful especially in fieldwork (e.g., Ladefoged, 2003), the roof of the mouth is coated in a mixture of olive oil and powdered charcoal and the subject produces a consonant. Details of the consonant's place of articulation and stricture are obtained from a photograph taken of the roof of the mouth showing where the powder was wiped-off and sometimes also of the tongue (which is coated in the powder at the point where tongue-palate contact was made). Dynamic electropalatography (Hardcastle, 1972; Hardcastle et al, 1991) is an extension of this technique in which tongue-palate contacts are recorded as a function of time. In dynamic palatography, an acrylic palate is custom-made for each subject and fixed to the roof of the mouth using clasps placed over the teeth. The palate is very thin and contains a number of electrodes that are exposed to the surface of the tongue (Fig. 7.1).



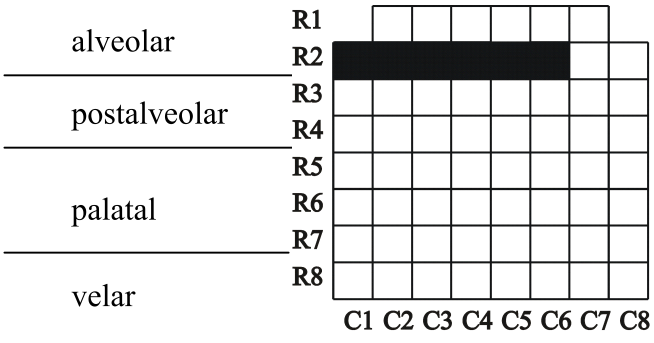


Fig. 7.1: The palate of the EPG3 system in a plaster cast impression of the subject's upper teeth and roof of the mouth (left) and fixed in the mouth (right). Pictures from the Speech Science Research centre, Queen Margaret University College, Edinburgh, http://www.qmuc.ac.uk/ssrc/DownSyndrome/EPG.htm. Bottom left is a figure of the palatographic array as it appears in R showing 6 contacts in the second row. The relationship to phonetic zones and to the row (R1-R8) and column (C1-C8) numbers are also shown.

Each electrode is connected to a wire and all the wires from the electrodes are passed out of the corner of the subject's mouth in two bundles. The wires are fed into a processing unit whose job it is to detect whether or not there is electrical activity in any of the electrodes. The choice is binary in all cases: either there is activity or there is not. Electrical activity is registered whenever the tongue surface touches an electrode because this closes an electrical circuit that is created by means of a small electrical current passed through the subject's body via a hand-held electrode.

Three EPG systems that have been commercially available include the Reading EPG3 system developed at the University of Reading and now sold by Articulate Instruments; a Japanese system produced by the Rion corporation and an American system that has been sold by Kay Elemetrics Corporation (see Gibbon & Nicolaidis, 1999 for a comparison of the three systems).

The palate of the Reading EPG3 system, which is the system that is compatible with Emu-R, contains 62 electrodes as shown in Fig. 7.1 that are arranged in eight rows. The first row, at the front of the palate and just behind the upper front teeth contains six electrodes, and the remaining rows each have 8 electrodes. There is a greater density of electrodes in the dental-alveolar than in the dorsal region to ensure that the fine detail of lingual activity that is possible in the dental, alveolar, and post-alveolar zones can be recorded. The last row is generally positioned at the junction between the subject's hard and soft-palate.

Fig 7.1 also shows the type of display produced by the EPG-system; the cells are either black (1) when the corresponding electrode is touched by the tongue surface or white (0) when it is not. This type of display is known as a **palatogram** and the EPG3 system typically produces palatograms at a sampling frequency of 100 Hz, i.e., one palatogram every 10 ms. As Fig. 7.1 shows, the palate is designed to register contacts extending from the alveolar to velar articulations with divisions broadly into alveolar (rows 1-2), post-alveolar (rows 3-4), palatal (rows 5-7) and velar (row 8).

Electropalatography is an excellent tool for studying consonant cluster overlap and timing. It also has an important application in the diagnosis and the treatment of speech disorders. Another major advantage of EPG is that there is often a reasonably transparent relationship between phonetic quality and EPG output: a [t] really does show up as contacts in the alveolar zone, the different groove widths between [s] and [ʃ] are usually very clearly manifested in EPG displays, and coarticulatory and assimilatory influences can often be seen and quantified. (See Gibbon, 2005, for a bibliography of electropalatographic studies since 1957).

At the same time, it is important to be clear about some of the limitations of this technique:

- A separate palate (involving a visit to the dentist for a plaster-cast impression of the roof of the mouth) has to be made for each subject which can be both time-consuming and expensive.
- As with any articulatory technique, subject-to-subject variation can be considerable. One subject's production of the stop in *key* can show up as palatal and lateral contact, for another there may be only limited lateral contact and fewer rows may be contacted. This variation can come about not only because subjects may invoke different articulatory strategies for producing the same phonetic segment, but also because the rows of electrodes are not always aligned with exactly the same articulatory landmarks across subjects.
- EPG can obviously give no direct information about labial consonants (apart from coarticulatory effects induced by other segments) and there is usually only limited information for places of articulation beyond a post-palatal or pre-velar articulation:

that is, /k/ in English shows up clearly in *key*, but for many subjects there may be scarcely any recorded activity for the retracted /k/ in *call*.

- EPG can only give limited information about vowels. It does register the lateral contact in non-low front vowels, but provides little information about tongue position and velocity.
- Many EPG3 systems have a fixed EPG sampling rate of 100 Hz and the synchronised acoustic signal is fixed at a sampling frequency of 10000 Hz (although in more recent models this can be changed). A 100 Hz palatogram rate is often too slow to record the details especially in apical articulations; a 10000 Hz sampling frequency with the associated 5000 Hz cut-off is often too low for carrying out articulatory-acoustic modelling of fricatives.

7.2. An overview of electropalatography in Emu-R

The databases listed at the beginning of this book whose names begin with *epg* include electropalatographic data and they can all be downloaded following the procedure discussed in 2.1. When an utterance is opened from any of these databases, a palatographic frame appears at the time point of the cursor (Fig. 7.2). The electropalatographic data that is compatible with Emu is derived from the 62-electrode EPG system manufactured by Articulate Instruments (2008). If you already have your own EPG data from this system, then, in order to read it into Emu, it first needs to be converted into an SSFF (simple signal file format): this can be done from `Arrange Tools` in the Emu-DB tool and then `EPG2SSFF`.

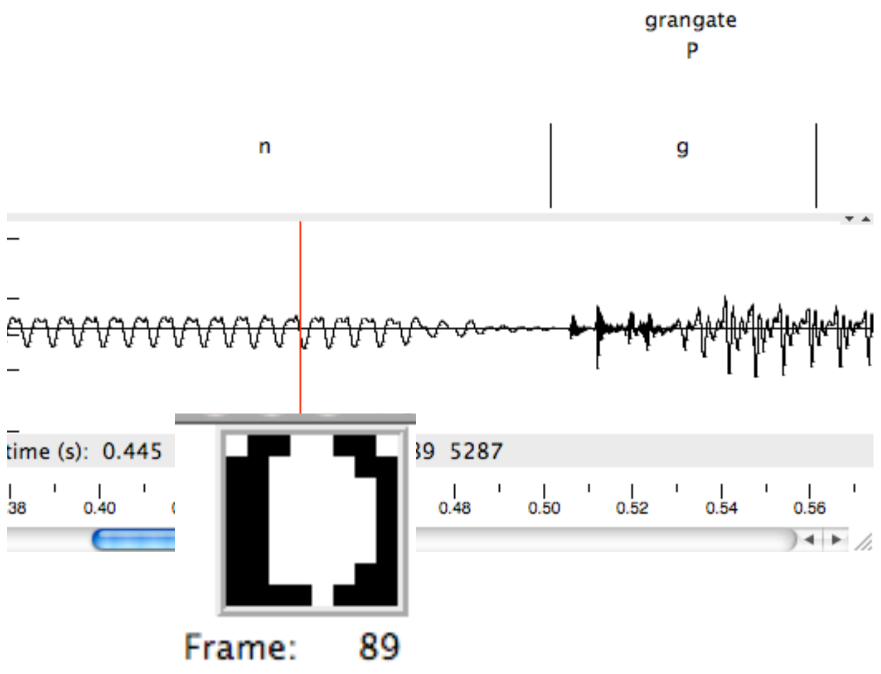


Fig. 7.2. Palatogram in the /n/ of *Grangate* in the utterance of the same name from the **epgassim** database. The palatogram is at the time point shown by the vertical line in the waveform.

Once an EPG-database is available in Emu, then the EPG signal files of the database are accessible to Emu-R in all of the ways that have been described in the preceding Chapters. In addition, there are some functions that are specific to an EPG analysis in Emu-R and these and the relationship between them are summarised in Fig. 7.3.

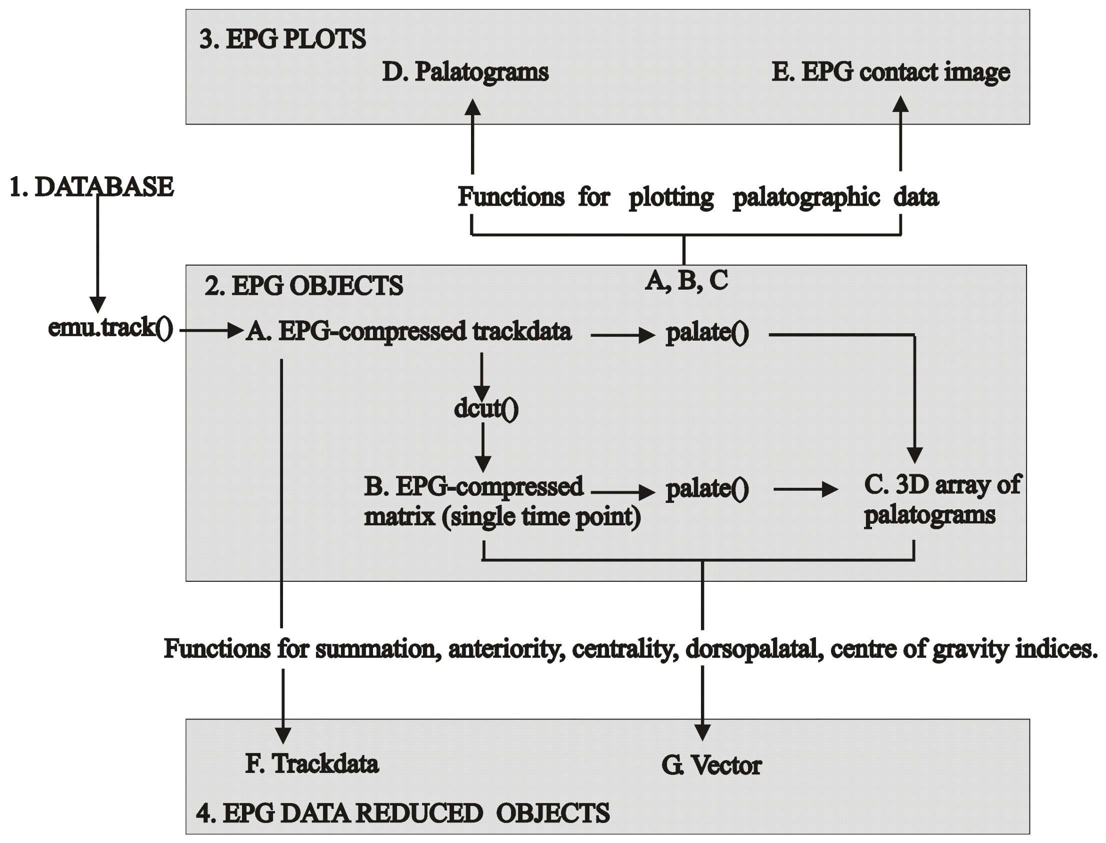

3. EPG PLOTS
D. Palatograms
E. EPG contact image
1. DATABASE
Functions for plotting palatographic data
2. EPG OBJECTS
A, B, C
emu.track()
A. EPG-compressed trackdata
palate()
dcut()
B. EPG-compressed matrix (single time point)
palate()
C. 3D array of palatograms
Functions for summation, anteriority, centrality, dorsopalatal, centre of gravity indices.
F. Trackdata
G. Vector
4. EPG DATA REDUCED OBJECTS

Fig. 7.3: Schematic outline of the relationship between electropalatographic objects and functions in R.

As Fig. 7.3 shows, there are four main components to the EPG anlaysis in Emu-R.

1. **Accessing the database**. The EPG-data is accessed from the database in the usual way from a segment list via the emu.track() function.
2. **EPG Objects.** The EPG-data that is read into R with emu.track() is an **EPG-compressed trackdata** object (Fig. 7.3, box 2, A) which compresses the 62 zero and one values of each palatogram into a vector of just 8 values. Since this is a trackdata object, then it is amenable to dcut() for obtaining an **EPG-compressed matrix at a single time point** (Fig. 7.3, box 2, B). Both of these EPG-compressed objects can be uncompressed in R (using the palate() function) to produce a **3D palatographic array** (Fig. 7.3, box 2, C): that is, an array of palatograms containing 0s and 1s in an 8 x 8 matrix.

Any of the objects listed under 2. are then amenable to two kinds of analysis: plotting or further paramaterisation, as follows:

3. **EPG Plots**. Two kinds of plots are possible: either the palatograms showing their time-stamps, or a three-dimensional grey-scale plot that represents the frequency of contact over two or more palatograms.

4. **EPG data-reduced objects.** In this case, the 62 palatographic values from each palatogram are reduced to a single value. As will be shown later in this Chapter, these data-reduced objects can be very useful for quantifying consonantal overlap and coarticulation.

It will be helpful to begin by looking in some further detail at the types of R objects in box 2 (EPG Objects) of Fig. 7.3, because they are central to all the other forms of EPG analysis, as the figure shows. All of the EPG-databases that are pre-stored and accessible within the Emu-R library and used as examples in this Chapter are initially in the form of EPG-compressed-trackdata objects (A. in Fig. 7.3) and this is also always the way that you would first encounter EPG data in R if you are using your own EPG database using the EPG system from Articulate Instruments via Emu. One of the available EPG-database fragments is **coutts** and it includes the following R objects:

coutts	word segment list (of the sentence: 'just relax said Coutts'; one segment per word)
coutts.sam	sampled speech trackdata object of coutts
coutts.epg	EPG-compressed-trackdata object of coutts (frame rate 5 ms)

The segment list, coutts, consists of four words of a sentence produced by a female speaker of Australian English and the sentence forms part of a passage that was constructed by Hewlett & Shockey (1992) for investigating (acoustically) coarticulation in /k/ and /t/. Here is the segment list:

```
coutts
segment  list from database:  epgcoutts
query was:  [Word!=x ^ Utterance=u1]
  labels   start     end          utts
1   just 16018.8 16348.8 spstoryfast01
2  relax 16348.8 16685.7 spstoryfast01
3   said 16685.7 16840.1 spstoryfast01
4 Coutts 16840.1 17413.7 spstoryfast01
```

The EPG-compressed trackdata object coutts.epg therefore also necessarily consists of four segments, as can be verified with nrow(coutts.epg). Thus the speech frames of EPG data for the first word in the segment list, *just*, is given by frames(coutts.epg[1,]) and as dim(frames(coutts.epg[1,])) shows, this is a 66 x 8 matrix: 66 rows because there are 66 palatograms between the start and end time of just and 8 columns which provide the information about palatographic contacts in columns 8-1 respectively. As for all trackdata objects, the times at which these EPG-frames of data occur are stored as row names (accessible with tracktimes(coutts.epg)) and for this example they show that palatographic frames occur at intervals of 5 ms (i.e. at times 16020 ms, 16025 ms, etc.).

Each of the EPG-frames can be unpacked into a series of zeros and ones corresponding to the absence and presence of contact in the palatogram. The unpacking is done by converting these values into binary numbers after adding 1 (one). More specifically, consider e.g. the 23rd EPG-frame of the 1st segment:

```
frames(coutts.epg[1,])[23,]
T1  T2  T3  T4  T5  T6  T7  T8
195 195 131 131 129   1   0   0
```

The first value, corresponding to row 8 is 195. In order to derive the corresponding palatographic contacts for this row, 195 + 1 = 196 is converted into binary numbers. 196 in binary form is 11000011 and so this is the contact pattern for the last (8th row) of the palate at time 16020 ms (i.e., there is lateral contact and no contact at the centre of the palate). Since the next entry is also 195, then row 7 evidently has the same contact pattern.

This job of converting EPG-frames into binary values and hence palatographic contacts is done by the palate() function. So the palatogram for all 66 rows of data in coutts.epg[1,] i.e., of the word *just* extending in time from 16020 ms to 16340 ms is obtained as follows:

```
p <- palate(coutts.epg[1,])
```

p is a three-dimensional array of palatograms, as shown by the following:

```
dim(p)
8  8 66
```

The first element that is returned by dim(p) refers to the **number of palatographic rows** and the second to the **number of palatographic columns**: these are therefore always both 8 because each palatogram contains contacts defined over an 8 x 8 grid. The third entry is **the number of palatograms**. The result here is 66 because, as has just been shown, this is the number of palatograms between the start and end times of *just*.

A three-dimensional palatographic array is indexed in R with [r, c, n] where r and c are the row and column number of the palatogram and n is the frame number (from 1-66 in the present example). In order to get at the entire palatogram, omit the r and c arguments. So the first palatogram at the onset of the word *just* (at time 16020 ms corresponding to the first row in frames(coutts.epg[1,] is:

```
p[,,1]
   C1 C2 C3 C4 C5 C6 C7 C8
R1  0  1  1  1  1  1  0  0
R2  1  1  1  1  1  1  1  1
R3  1  1  1  0  0  1  1  1
R4  1  1  1  0  0  0  1  1
R5  1  1  0  0  0  0  0  1
R6  1  1  0  0  0  0  1  1
R7  1  1  0  0  0  0  1  1
R8  1  1  0  0  0  0  1  1
```

In this type of array, the row and column numbers are given as the respective dimension names. Since the first row of the EPG3 palate has 6 contacts (i.e., it is missing the two most lateral contacts), the values in both row 1 column 1 and in row 1 column 8 are always zero.

The indexing on the palatograms works as for matrices, but since this is a 3D-array, *two* preceding commas have to be included to get at the palatogram number: so p[,,1:3] refers to the first three palatograms, p[,,c(2, 4)], to palatograms 2 and 4, p[,,-1] to all palatograms except the first one, and so on. It is worthwhile getting used to manipulating these kinds of palatographic arrays because this is often the primary data that you will have to work with, if you ever need to write your own functions for analysing EPG data (all of the functions for EPG plotting and EPG data reduction in boxes 3 and 4 of Fig. 7.3 are operations on these kinds of arrays). A useful way in which to become familiar with them is to make up some palatographic data. For example:

```
# Create 4 empty palatograms
fake = array(0, c(8, 8, 4))
# Give fake appropriate row and dimension names for a palatogram
rownames(fake) = paste("R", 1:8, sep="")
colnames(fake) = paste("C", 1:8, sep="")
# Fill up row 2 of the 3rd palatogram with contacts
fake[2,,3] = 1
# Fill up row 1, columns 3-6, of the 3rd palatogram only with contacts
fake[1,3:6,3] = 1
# Look at the 3rd palatogram
fake[,,3]
   C1 C2 C3 C4 C5 C6 C7 C8
R1  0  0  1  1  1  1  0  0
R2  1  1  1  1  1  1  1  1
R3  0  0  0  0  0  0  0  0
R4  0  0  0  0  0  0  0  0
R5  0  0  0  0  0  0  0  0
R6  0  0  0  0  0  0  0  0
R7  0  0  0  0  0  0  0  0
R8  0  0  0  0  0  0  0  0

# Give contacts to rows 7-8, columns 1, 2, 7, 8 of palatograms 1, 2, 4
fake[7:8, c(1, 2, 7, 8), c(1, 2, 4)] = 1
# Look at rows 5 and 7, columns 6 and 8, of the palatograms 2 and 4:
fake[c(5,7), c(6, 8), c(2,4)]
, , 1
   C6 C8
R5  0  0
R7  0  1

, , 2
   C6 C8
R5  0  0
R7  0  1
```

The times at which palatograms occur are stored as the names of the third dimension and they can be set as follows:

```
# Assume that these four palatograms occur at times 0, 5, 10, 15 ms
times <- seq(0, by=5, length=4)
# Store these times as dimension names of fake
dimnames(fake)[[3]] = times
```

This causes the time values to appear instead of the index number. So the same instruction as the previous one now looks like this[1]:

```
, , 5
   C6 C8
R5  0  0
R7  0  1

, , 15
   C6 C8
R5  0  0
R7  0  1
```

These two (fragments of) palatograms occur at 5 ms and 15 ms.

Functions can be applied to the separate components of arrays in R using the apply() function. For 3D-arrays, 1 and 2 in the second argument to apply() refer to the rows and columns (as they do for matrices) and 3 to the 3rd dimension of the array, for example:

```
# Sum the number of contacts in the 4 palatograms
apply(fake, 3, sum)
8  8 12  8
# Sum the number of contacts in the columns
apply(fake, c(2,3), sum)
   0 5 10 15
C1 2 2  1  2
C2 2 2  1  2
C3 0 0  2  0
C4 0 0  2  0
C5 0 0  2  0
C6 0 0  2  0
C7 2 2  1  2
C8 2 2  1  2
```

Notice that the above command returns a matrix whose columns refer to palatograms 1-4 respectively (at times 0, 5, 10, 15 ms) and whose rows show the summed values per palatographic column. So the entries in row 1 means: the number of contacts in column 1 of the palatograms occurring at 0, 5, 10, 15 ms are 2, 2, 1, 2 respectively. If you want to sum (or to apply any meaningful function) by row or column **across all palatograms together**, then the second argument has to be 1 (for rows) of 2 (for columns) on its own. Thus:

```
apply(fake, 1, sum)
R1 R2 R3 R4 R5 R6 R7 R8
 4  8  0  0  0  0 12 12
```

The first returned entry under R1 means that the sum of the contacts in row 1 of all four palatograms together is 4 (which is also given by sum(fake[1,,])).

As already mentioned, arrays can be combined with logical vectors in the usual way – but take great care where to place the comma! For example, suppose that these are four palatograms corresponding to the labels `k`, `k`, `t`, `k` respectively. Then the palatograms for `k` can be given by:

```
lab = c("k", "k", "t", "k")
```

[1] But the times do not appear as dimension names if you look at only a single palatogram – because in this special case, an array is turned into a matrix (which has only 2 dimensions, hence nowhere to put the 3rd dimension name).

```
temp = lab=="k"
fake[,,temp]
```

and rows 1-4 of the palatograms for t are:

```
fake[1:4,,!temp]
```

and so on. Finally, in order to apply the functions in boxes 3 and 4 of Fig. 7.3 to made-up data of this kind, the data must be declared to be of class "EPG" (this tells the functions that these are EPG-objects). This is done straightforwardly as:

```
class(fake) = "EPG"
```

Having established some basic attributes of EPG objects in R, the two functions for plotting palatograms can now be considered. As Fig. 7.4 shows, palatograms can be plotted directly from EPG-compressed trackdata objects or from time slices extracted from these using dcut(), or else from the 3D palatographic arrays of the kind discussed above. We will begin by looking at EPG data from the third and fourth segments *said Coutts*. This is given by epgplot(coutts.epg[3:4,]) (or by epgplot(palate(coutts.epg[3:4,]))) and the corresponding waveform, from which the palatograms are derived, by plot(coutts.sam[3:4,], type="l").

Some of the main characteristics of the resulting palatograms shown in Fig. 7.4 are:

- The alveolar constriction for the fricative [s] of *said* is in evidence in the first 7 palatograms between 16690 ms and 16720 ms.
- The alveolar constriction for [d] of *said* begins to form at 16800 ms and there is a complete alveolar closure for 8 palatograms, i.e., for 40 ms.
- There is clear evidence of a doubly-articulated [d͡k] in *said Coutts* (i.e., a stop produced with simultaneous alveolar and velar closures) between 16825 ms and 16835 ms.
- [k] of *Coutts* is released at 16920 ms.
- The aspiration of *Coutts* and the following [ʉ] vowel extend through to about 17105 ms.

- The closure for the final alveolar [t] of *Coutts* is first completed at 17120 ms. The release of this stop into the final [s] is at 17205 ms.

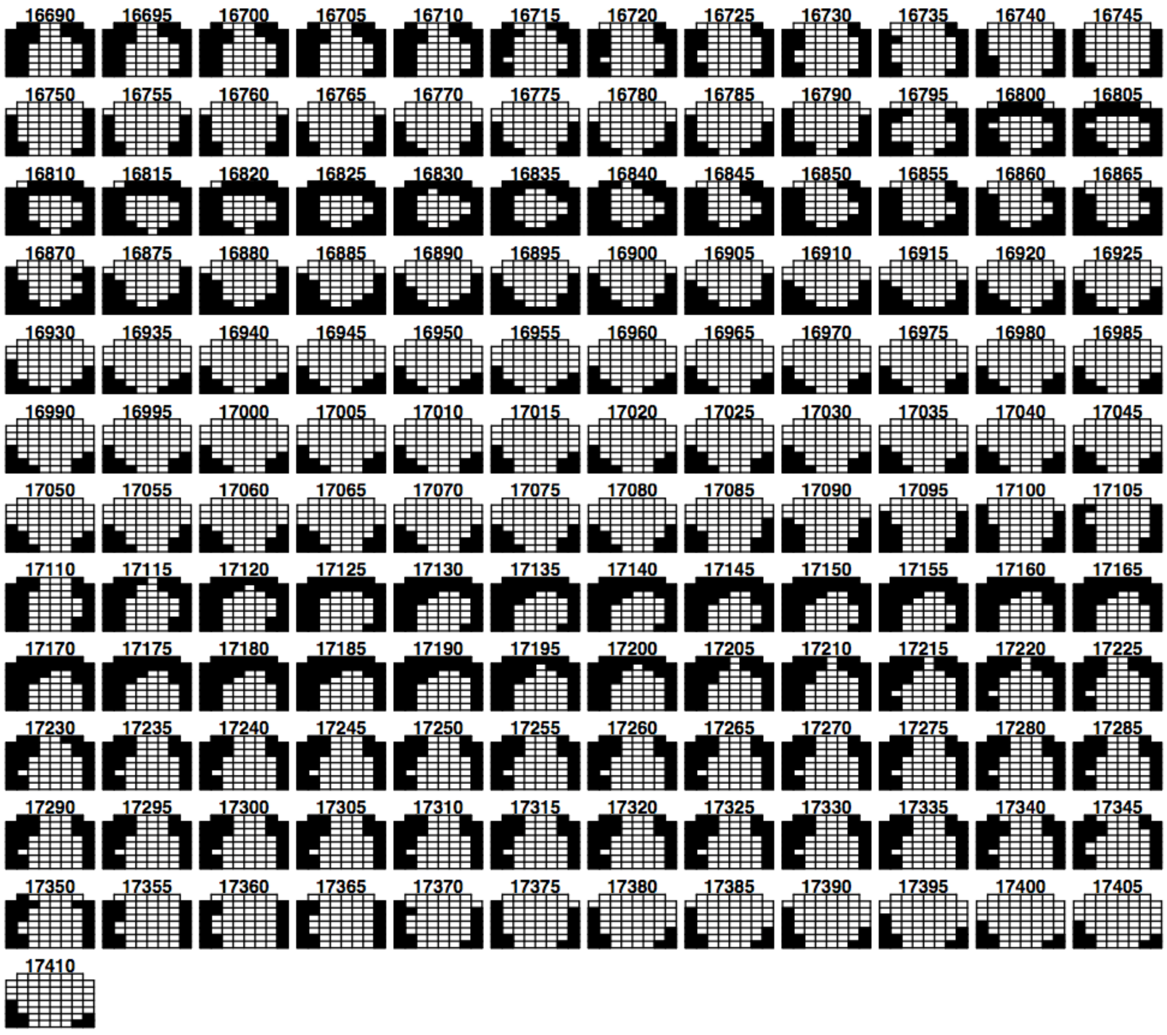


Fig. 7.4: Palatograms of *said Coutts* showing the times (ms) at which they occurred.

The interval including at least the doubly-articulated [d͡k] has been marked by vertical lines on the waveform in Fig. 7.5. This was done with the locator() function that allows any number of points on a plot to be selected and the values in either *x*- or *y*-dimension to be stored (these commands must be entered after those used to plot Fig. 7.5):

```
# Select two time points at store the x-coordinates
times <- locator(2)$x
# The vertical boundaries in Fig. 7.5 are at these times
times
16828.48 16932.20
```

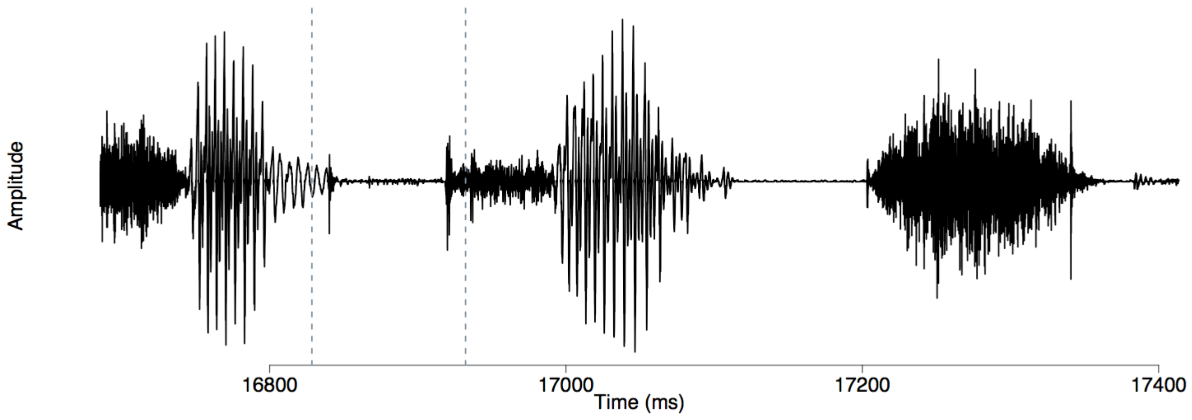


Fig. 7.5: Waveform over the same time interval as the palatograms in Fig. 7.3. The vertical lines dotted mark the interval that is selected in Fig. 7.6.

The xlim argument can be used to plot the palatograms over this time interval and optionally the mfrow argument to set the number of rows and columns (you will also often need to sweep out the graphics window in R to get an approximately square shape for the palatograms):

```
# Palatograms plotted between the interval defined by times and displayed in 2 x 11
epgplot(coutts.epg, xlim=times, mfrow=c(2,11))
```

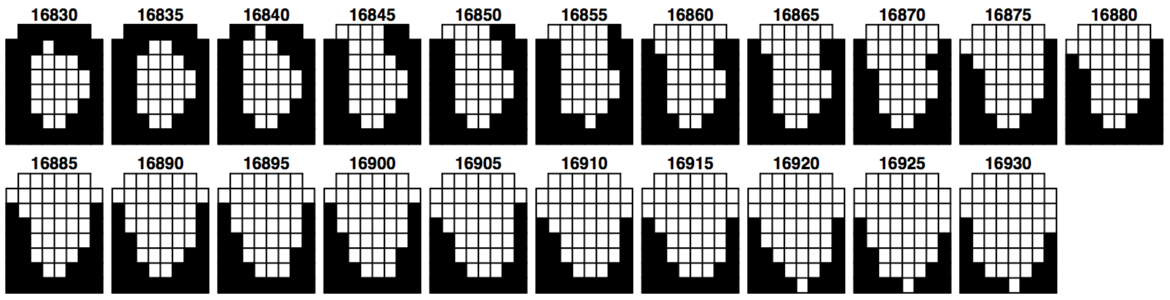


Fig. 7.6: Palatograms over the interval marked by the vertical lines in Fig. 7.5.

The next example of manipulating and plotting electropalatographic data is taken from a fragment of a database of Polish fricatives that was collected in Guzik & Harrington (2007). This database was used to investigate the relative stability of fricatives in word-final and word-initial position. Four fricatives were investigated: the alveolar [s], a post-alveolar [ʃ], an alveolo-palatal [ɕ], and a velar [x]. They were produced in word-pairs in all possible combinations with each other across word boundaries. So there are sequences like [s#ʃ] (in *wlos szary*), [ʃ#ɕ] (in *pytasz siostre*), [x#s] (in *dach sali*) and so on for all possible 4 x 4 cross-word boundary combinations, including the homorganic sequences [s#s], [ʃ#ʃ], [ɕ#ɕ], [x#x]. The database fragment **polhom** is of the homorganic sequences produced by one native, adult male speaker of Polish. The palatographic data was sampled at 100 Hz:

polhom	Segment list of Polish homorganic fricatives
polhom.l	A parallel vector of labels (`s, S, c, x`, for [s#s], [ʃ#ʃ], [ɕ#ɕ], [x#x])
polhom.epg	Parallel EPG trackdata

As table(polhom.l) shows, there are 10 homorganic fricatives in each category. If you have accessed the corresponding database **epgpolish** from the Arrange tools -> DB Installer in Emu, then you will see that the segment boundaries in the segment list polhom extend approximately from the acoustic onset to the acoustic offset of each of these homorganic fricatives.

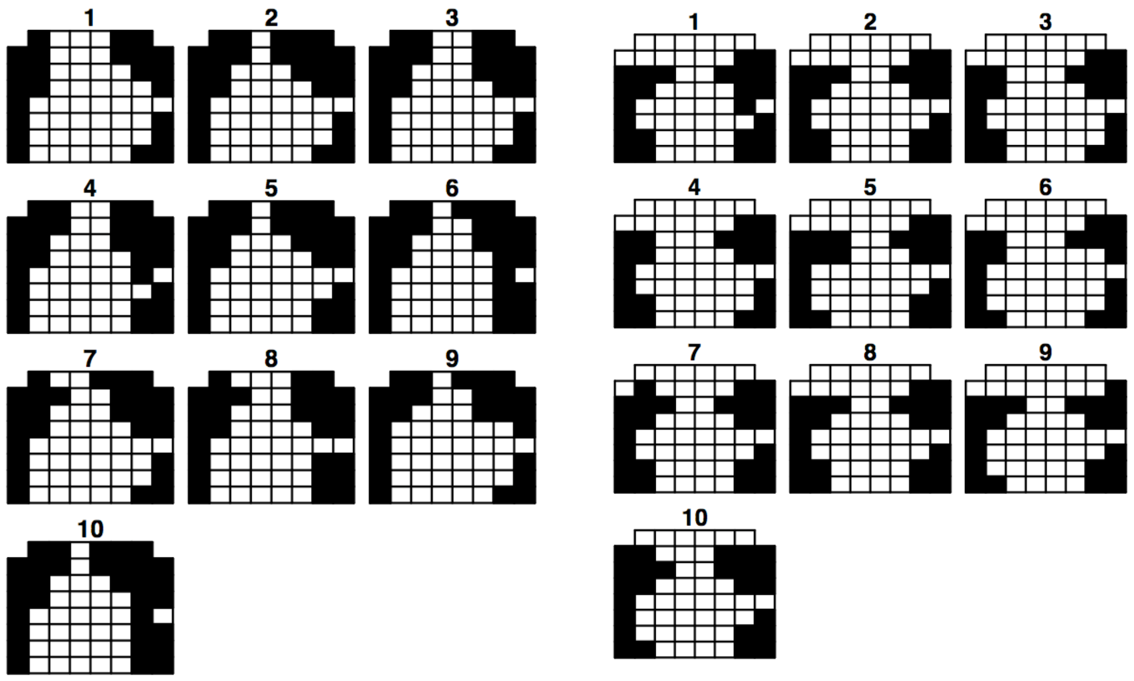


Fig. 7.7: Palatograms for 10 [s] (left) and 10 [ʃ] (right) Polish fricatives extracted at the temporal midpoint from homorganic [s#s] and [ʃ#ʃ] sequences produced by an adult male speaker of Polish. (The electrode in column 8 row 5 malfunctioned and was off throughout all productions).

The first task will be to compare [s] with [ʃ] as far as differences and similarities in palatographic contact patterns are concerned and this will be done by extracting the palatographic frames closest to the temporal midpoint of the fricatives. The data for [s] and [ʃ] are accessed with a logical vector, and dcut() is used for extracting the frames at the midpoint:

```
# Logical vector to identify s and S
temp = polhom.l %in% c("s", "S")
# EPG-compressed trackdata for s and S
cor.epg = polhom.epg[temp,]
# Matrix of EPG-compressed data  for s and S at the temporal midpoint
```

```
cor.epg.5 = dcut(cor.epg, 0.5, prop=T)
# Labels for the above
cor.l = polhom.l[temp]
```

sum(temp) shows that there are 20 fricatives and table(cor.l) confirms that there are 10 fricatives per category. The following produces a plot of the palatograms at the temporal midpoint, firstly for [s], then for [ʃ]. Rather than displaying the times at which they occur, the palatograms are numbered with the num=T argument:

```
# Logical vector:  T when cor.l is s, F when cor.l is S
temp = cor.l =="s"
# palatograms for [s]
epgplot(cor.epg.5[temp,], num=T)
# palatograms for [ʃ]
epgplot(cor.epg.5[!temp,], num=T)
```

As expected, the primary stricture for [s] is further forward than for [ʃ] as shown by the presence of contacts for [s] but not for [ʃ] in row 1. A three-dimensional, gray-scale image can be a useful way of summarising the differences between two different types of segments and in R: the function for doing this is epggs():

```
par(mfrow=c(1,2))
epggs(cor.epg.5[temp,], main="s")
epggs(cor.epg.5[!temp,], main="S")
```

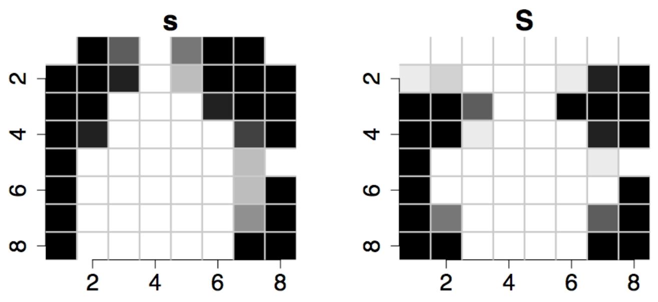


Fig. 7.8: Gray-scale images of the data in Fig. 7.7 for [s] (left) and [ʃ] (right). The darkness of a cell is proportional to the number of times that the cell was contacted.

At the core of the epggs() function is a calculation of **the proportional number of times** a **cell was contacted**. When a cell is black, then it means that it was contacted in all the palatograms over which the function was calculated, and when a cell is white, then there were no contacts. Thus for [s] in Fig. 7.8, the entire first column is black in this three-dimensional display because, as Fig. 7.7 shows, all ten palatograms for [s] have their contacts on in column 1; and column 5 of rows 1 and 2 for [s] are dark-gray, because, while most [s] palatograms had a contact for this cell (numbers 2, 5, 6, 7, 9, 10 in Fig. 7.7), others did not.

7.3. EPG data reduced objects

As discussed earlier, various functions can be applied to EPG-data that reduce each palatogram to a single value (Fig. 7.3, box 4). The most basic of these is a function for producing a contact profile in which the contacts per palate are summed (7.3.1). The other data reduction functions which are discussed in 7.3.2 are essentially further operations on contact profiles. In the final part of the Chapter (section 4) some of these data reduction functions are put to use for measuring the extent of overlap in consonant clusters and vowel-induced consonantal coarticulation.

All data reduction functions work on the same kinds of EPG-objects as those for plotting electropalatographic data in 7.3. Thus, they can be applied to EPG-compressed trackdata objects, a matrix of EPG-compressed data extracted at a single time slice, or to a 3D-palatographic array. In all cases, the output is a single value per palatogram: if the data reduction functions are applied to an EPG-compressed trackdata object, these values are structured into a trackdata object. These points are elaborated further in the next section.

7.3.1 Contact profiles

A contact profile is a data reduction involving a summation of palatographic data by row(s) and/or by column(s). Contact profiles have a number of applications in phonetics: they can be used to distinguish between stops and fricatives at the same place of articulation (by summing the number of contacts in certain rows) or between different places of articulation (by summing contacts in different rows).

The function for calculating a contact profile is epgsum() and its default is to **sum all the contacts per palate**. Thus for the 3D-array fake created earlier, epgsum(fake) gives the same result as the operation applied in 7.2 for summing contacts in the four palatograms, apply(fake, 3, sum)[2]. But epgsum() can also be used to **sum selectively by row and column**. So epgsum(fake, rows=1:4) sums the contacts in rows 1-4, epgsum(fake, rows=1:4, columns=c(1, 2, 7, 8)) sums contacts in rows 1-4 of columns 1, 2, 7 and 8. The additional argument inactive=T can be used to **sum the inactive electrodes** (also by row and by column), i.e., the 0s of the palatograms. The default is to sum the entire palatogram (in selected rows and or columns) but it is also possible to show the summations for the separate rows or columns using a second argument of 1 (for rows) or 2 (for columns). For example, in the previous section it was shown how apply(fake, c(2,3), sum) gives the sum of the contacts in the columns: an equivalent way of doing this is epgsum(fake, 2). See help(epgsum) for further examples.

In Fig. 7.4, the separate palatograms at 5 ms intervals were shown for the words *said Coutts*. By making a display of the summed contacts in rows 1-3, the articulations in the front part of the palate should become very clearly visible, while a summation in the back two rows over columns 3-6 should produce a display which is associated with the degree of tongue-dorsum contact in /k/ of *Coutts*. Here are these two contact profiles:

```
# Sum rows 1-3 of the EPG-trackdata object over said Coutts
fsum <- epgsum(coutts.epg[3:4,], rows=1:3)
# Sum rows 7-8, columns 3-6 of the EPG-trackdata object over said Coutts
bsum <- epgsum(coutts.epg[3:4,], rows=7:8, columns=3:6)
```

A plot of the contact profiles superimposed on each other together with the waveform is shown in Fig. 7.9 and can be produced as follows:

```
# Column-bind the trackdata objects
both = cbind(fsum, bsum)
par(mfrow=c(2,1)); par(mar=c(1,4,1,1))
```

[2] As described earlier, fake must an object of class 'EPG' for this to work. So if class(fake) returns 'array', then enter class(fake) = "EPG"

```
xlim = c(start(coutts[3,]), end(coutts[4,]))
plot(both, type="l", ylab="Summed contacts", xlab="", axes=F, xlim=xlim)
axis(side=2); axis(side=1)
mtext("Time (ms)", side=1, at=17300)
# Superimpose some symbols
text( c(16816, 16846,  17158,  17291), c(19.6,  8.7,  17.1, 15.0), c("d", "k", "t", "s"))
# Plot the synchronised acoustic waveform
plot(coutts.sam[3:4,], type="l", axes=F, xlab="Time (ms)", ylab="", xlim=xlim)
# Restore the margin defaults
par(mar=c(5.1, 7.1, 7.1, 2.1))
```

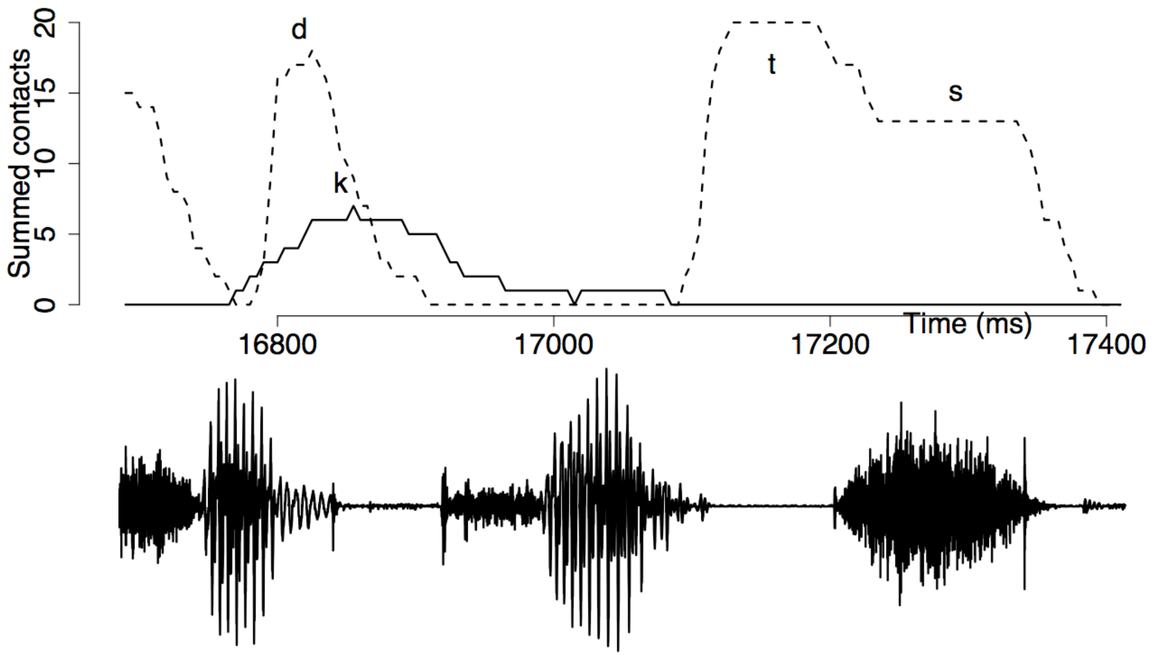


Fig. 7.9: Sum of the contacts in rows 1-3 (dashed) and in and rows 6-8 (solid) showing some phonetic landmarks synchronised with an acoustic waveform in *said Coutts* produced by an adult female speaker of Australian English.

The synchronised contact profiles in Fig. 7.9 provide a great deal of information about the overlap and lenition of the alveolar and velar articulations, for example:

- The tongue dorsum for [k] already begins to rise during [ɛ] of *said*.
- The maximum overlap between [d] and [k] is at the point of the final stop release in *said*.
- The [t] of *Coutts* is less lenited compared with [d] of *said*, as shown by the greater number of contacts for the former extending over a greater duration.

Contact profiles could be used to distinguish between the Polish [s,ʃ] fricatives discussed earlier according to the **central groove width** which could be defined as the smallest number of inactive electrodes in any row over the central columns 3-6. For example, in the first five palatograms of [s] in Fig. 7.7, this central groove width is 3, 1, 2, 2, 1 respectively; for the first 5 [ʃ] palatograms in Fig. 7.7., the central groove width is usually at least one inactive contact greater: 3, 3, 2, 3, 2.

In order to obtain groove widths for the data in Fig. 7.7, the first step is to count the number of *inactive* electrodes (i.e., those with a value of zero) over a particular row and column range: we will restrict this to the first four rows and to columns 3-6, since, as Fig. 7.7 shows, this is the region of the palate within which the point of maximum narrowing occurs:

```
# Commands repeated from before
temp = polhom.l %in% c("s", "S")
cor.epg = polhom.epg[temp,]
cor.epg.5 = dcut(cor.epg, 0.5, prop=T)
cor.l = polhom.l[temp]
# Count the number of inactive electrodes in rows 1-4, columns 3-6
# and display the result separately by row
in.sum = epgsum(cor.epg.5, 1, rows=1:4, columns=3:6, inactive=T)
# Show the first two rows of in.sum
in.sum[1:2,]
     R1 R2 R3 R4
2120  3  3  4  4
1170  1  1  3  4
```

So that it is completely clear what is being counted, the first two palatograms of the array are listed below. The count on the right is of the zeros in bold:

```
p = palate(cor.epg.5)
p [,,1:2]
```

Number of inactive cells, rows 1-4, columns 3-6.

```
, , 2120

   C1 C2 C3 C4 C5 C6 C7 C8
R1  0  1  0  0  0  1  1  0                3
R2  1  1  0  0  0  1  1  1                3
R3  1  1  0  0  0  0  1  1                4
R4  1  1  0  0  0  0  0  1                4
R5  1  0  0  0  0  0  0  0
R6  1  0  0  0  0  0  0  1
R7  1  0  0  0  0  0  0  1
R8  1  0  0  0  0  0  1  1


, , 1170

   C1 C2 C3 C4 C5 C6 C7 C8
R1  0  1  1  0  1  1  1  0                1
R2  1  1  1  0  1  1  1  1                1
R3  1  1  0  0  0  1  1  1                3
R4  1  1  0  0  0  0  1  1                4
R5  1  0  0  0  0  0  0  0
R6  1  0  0  0  0  0  0  1
R7  1  0  0  0  0  0  0  1
R8  1  0  0  0  0  0  1  1
```

A function is needed to get the *minimum* groove width – that is, the function should return 3 and 1 respectively for the above two palatograms. Since in.sum is a matrix, this can be done with the apply() function:

```
# Find the row with the fewest 0s and return the number of 0s for that row
min.groove = apply(in.sum, 1, min)
# Minima for the first two palatograms above: this is correct (see the palatograms above)
min.groove[1:2]
2120 1170
   3    1
```

The histogram in Fig. 7.10 of the minimum groove width provides some limited evidence that it is less for [s] than for [ʃ]. The histogram was created with the following commands:

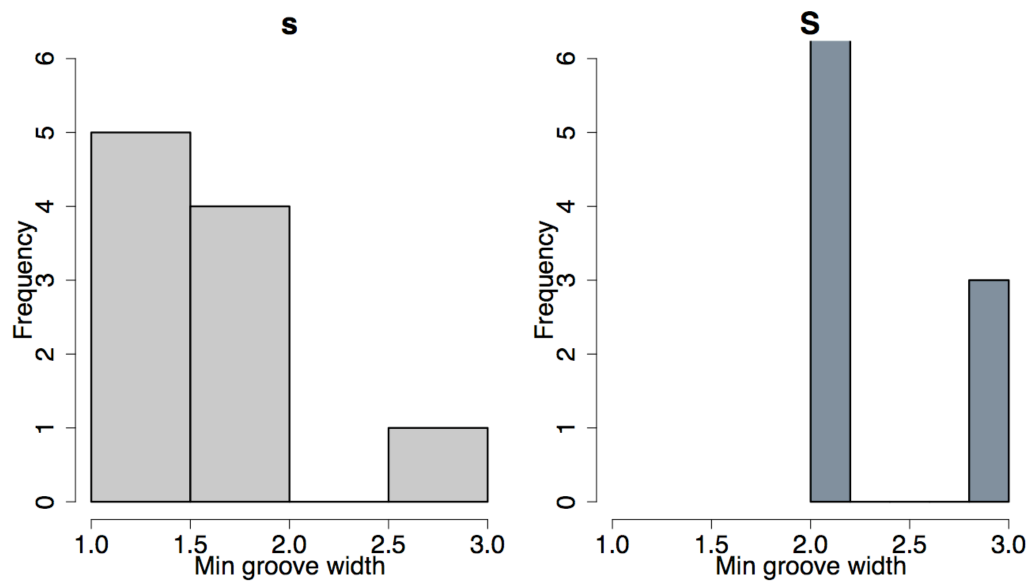


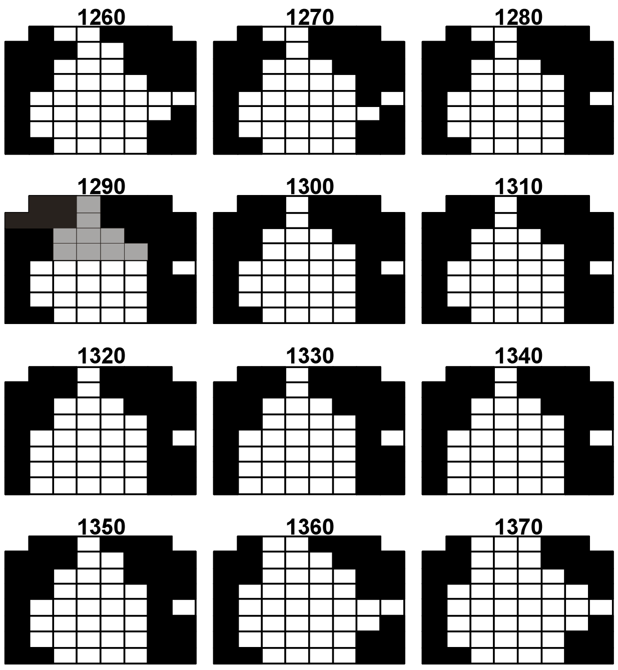
Fig. 7.10: A histogram of the distribution of the minimum groove width shown separately for palatograms of Polish [s,ʃ]. The minimum groove width is obtained by finding whichever row over rows 1-5, columns 3-6 has the fewest number of inactive electrodes and then summing them.

```
xlim = c(1,3); ylim = c(0, 6); par(mfrow=c(1,2))
xlab = "Min groove width"
# Logical vector that is True for [s]
temp = cor.l=="s"
hist(min.groove[temp], col="gray", main="s", xlim=xlim, ylim=ylim, xlab=xlab)
hist(min.groove[!temp], col="slategray", main="S", xlim=xlim, ylim=ylim,  xlab=xlab)
```

The above analysis was for one single palatogram per segment extracted at the temporal midpoint. The same kind of analysis could be carried out for *every* palatogram between the temporal onset and offset of these fricatives. This would allow us to see not only if there is a difference in minimum groove width between [s,ʃ], but also whether groove width decreases from the fricative margins towards the fricatives' temporal midpoint (this is to be expected given that the homorganic fricatives were flanked by vowels and given that the extent of stricture in fricative production tends to increase from the margins towards the temporal midpoint).

The first step is to count the number of inactive electrodes in rows 1-4 and columns 3-6 as before, but this time for all the palatograms contained in the entire EPG-compressed trackdata object. This is done in the following command by summing the number of inactive electrodes from the onset to the offset for all segments in the EPG-trackdata object polhom.epg and storing the count separately by row:

```
in.sum.all = epgsum(polhom.epg, 1, rows=1:4, columns=3:6, inactive=T)
```



```
frames(in.sum.all[10,])
R1  R2  R3  R4
1260  2  2  3  4
1270  2  1  3  4
1280  2  1  3  4
1290  1  1  3  4
1300  1  1  3  4
1310  1  1  3  4
1320  1  1  3  4
1330  1  1  3  4
1340  1  1  3  4
1350  1  2  3  4
1360  2  3  4  4
1370  3  3  4  4
```

Fig. 7.11: Palatograms for the first 12 frames between the acoustic onset and offset of a Polish [s]. On the right is the number of inactive electrodes for each palatogram in rows 1-7. The count of inactive electrodes for the palatogram 1290 ms is highlighted.

The object in.sum.all is four-dimensional (as shown by summary(in.sum.all)) and consists of the sum of inactive electrodes in rows 1-4 of columns 3-6 for every palatogram between the onset and offset of each fricative. So that it is clear what has just been calculated, Fig. 7.11 shows the EPG data for the 10th segment (given by epgplot(polhom.epg[10,])), together with the corresponding minimum groove widths (given by frames(in.sum.all[10,])). Thus, the values of the rows at 1290 ms in the matrix on the right of Fig. 7.11 are 1, 1, 3, 4 because this is the count of inactive electrodes in rows 1-4, columns 3-6 of the palatogram shown on the left at that time. A function is now needed similar to the one before to find the minimum value per row in the EPG frames

```
minfun <- function(contacts)
{
# Find the minimum per row
apply(contacts, 1, min)
}
```

When this function is applied to the data of the 10th segment, the minimum groove widths of the palatograms at intervals of 10 ms between the start and end time of the 10th segment are returned:

```
minfun(frames(in.sum.all[10,]))
1260 1270 1280 1290 1300 1310 1320 1330 1340 1350 1360 1370 1380
   2    1    1    1    1    1    1    1    1    1    2    3    4
```

This function must now be applied to every segment which can be done using the trapply() function with returntrack=T to build a corresponding trackdata object (see 5.5.2):

```
groove.min = trapply(in.sum.all, minfun, returntrack=T)
```

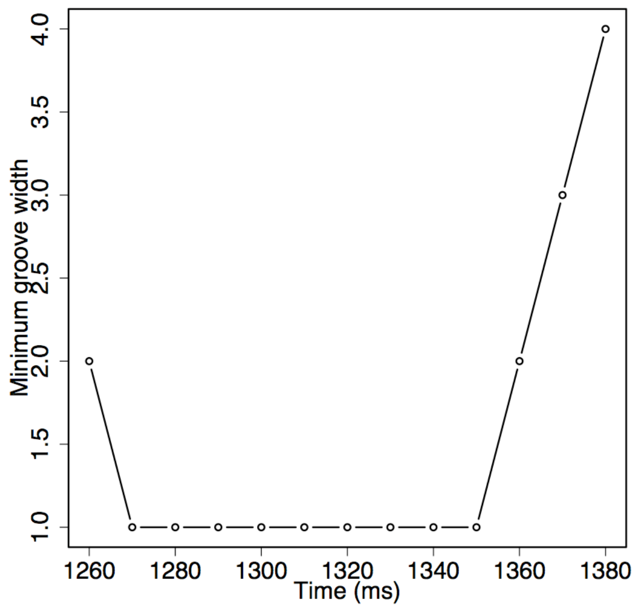


Fig. 7.12: Minimum groove width (number of off electrodes in the midline of the palate) between the acoustic onset and offset of a Polish [s].

A plot of the 10th segment of this trackdata object should give the same values that were returned by minfun(frames(in.sum.all[10,]), which is indeed the case (Fig. 7.12).

```
plot(groove.min[10,], type="b", ylab="Minimum groove width", xlab="Time (ms)")
```

Finally, a plot from segment onset to segment offset should show both the differences on this parameter between [s] and [ʃ] and also a progressively decreasing minimum groove width towards the temporal midpoint of the segments, as the fricative's stricture is increased. Such a plot can be produced with dplot() and in this example, the 10 fricatives per category are averaged after linear time normalisation (Fig. 7.13):

```
temp = polhom.l %in% c("s", "S")
dplot(groove.min[temp,], polhom.l[temp], norm=T, average=T, ylab="Minimum groove width", xlab="Normalised time", leg="topleft")
```

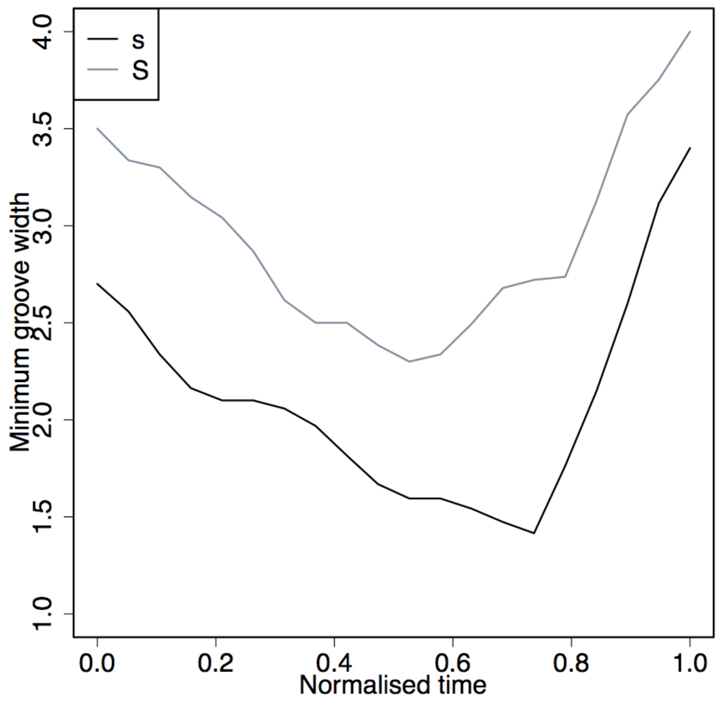


Fig. 7.13: Minimum groove width between the acoustic onset and offset of Polish [s] (black) and [ʃ] (gray) averaged after linear time normalisation.

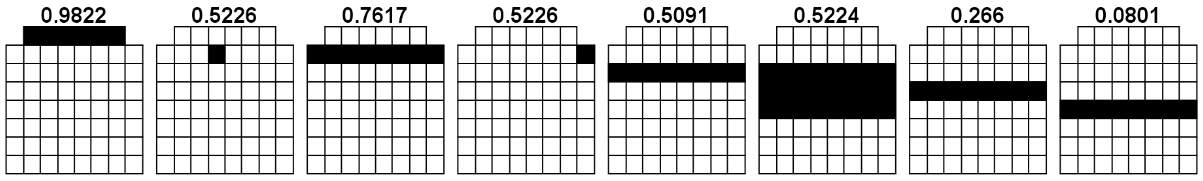
Evidently, the groove width decreases on average towards the temporal midpoint for [ʃ] and somewhat after the temporal midpoint for [s]. Fig. 7.13 also shows that the groove width for [s] is well below that of [ʃ] at equal proportional time points from segment onset to segment offset.

7.3.2 Contact distribution indices

As discussed in Gibbon & Nicolaidis (1999), various EPG parameters have been devised for quantifying both the distribution and the extent of tongue palate contacts. Almost all of these are based on some form of summation of the palates (see e.g., Recasens et al., 1993; Hardcastle, Gibbon and Nicolaidis, 1991 for details). These are the **anteriority index** (AI), the **centrality index** (CI), the **dorsopalatal index** (DI) and the **centre of gravity** (COG). The first three of these all vary between 0 and 1 and COG varies between 0.5 and 7.6. The R functions in the Emu-R library for calculating them are epgai(), epgci(), epgdi(), and epgcog() respectively.

The anteriority index quantifies how far forward the contacts are on the palate in rows 1-5. Rows 6-8 are not taken into account in this calculation. AI is especially useful for quantifying the place of articulation back as far as the post-alveolar zone (row 5) and can also be used to quantify the degree of stricture for two consonants at the same place of articulation. The data in Fig. 7.14 shows AI for various made-up palatograms. (Details of how to produce these are given at the end of this Chapter in the exercises).

Four general principles are involved in calculating AI (Fig. 7.14):



(a) (b) (c) (d) (e) (f) (g) (h)

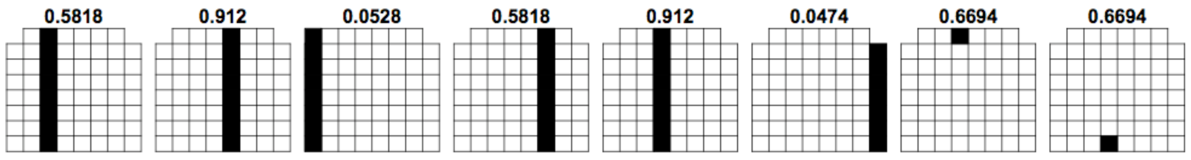
Fig. 7.14: Palatograms with corresponding values on the anteriority index shown above.

1. The further forward the contacts in any row, the higher AI. Thus, the palatogram with the filled row of contacts in row 1 in (a) has a higher AI value than (c) for which the contacts are filled in row 2. AI decreases from 0.9822 (filled row of contacts in row 1) to 0.08 (filled row of contacts in row 5). Any palatogram with contacts exclusively in rows 6-8 has an AI of 0.

2. Any single contact in row *i* always has a higher AI than any number of contacts in row *j*, where *i* < *j*. So the AIs for palatograms (b) and (d) that each have a single contact in row 2 are greater than the AI of palatogram (e) in which all contacts are filled in a lower row number, row 3.

3. The same number of contacts in any row has the same AI irrespective of their lateral distribution (distribution by column). So the fact that the lateral distribution of the single contact is different in palatograms (b) and (d) makes no difference as far as AI is concerned, since both palatograms have a single contact in row 2.

4. The greater the number of contacts, the higher AI – but only up to the limit specified by 2. above. So palatogram (f) which has rows 3-5 completely filled has a higher AI than palatogram (e), in which only row 3 is filled; but since palatogram (f) has no contacts forward of row 3, its AI is lower than those of (b) or (c) that have a single contact in row 2.



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 7.15: Palatograms with corresponding values on the centrality index shown above.

The **centrality index** (CI), as its name suggests, measures the extent of contact at the centre of the palate and varies between 0 and 1. In general, the more the contacts are laterally distributed, the lower the value of CI. This parameter could be used to distinguish between consonants that have a narrow vs. wide central groove, as in the [s,ʃ] fricatives discussed earlier. The actual calculation of CI can be explained in terms of a set of principles that are very similar to those of AI, except that they are based on columns and the relative lateralisation of contacts:

1. In the case of a single filled column of contacts, CI is higher nearer the centre of the palate: thus higher for filled columns 4 or 5 (palatograms (b), (e) in Fig. 7.15) than for the more laterally filled columns 3 or 6 (palatograms (a), (d)).

2. Any single contact in a given column has a higher CI than a palatogram filled with any number of contacts in more lateral columns. So the CIs for palatograms (g) and (h) which have a single contact in columns 4 and 5 are higher than those of palatograms (a) and (d) in which all contacts are filled in the more lateral columns 3 and 6.

3. The same number of contacts in any column has the same CI irrespective of the distribution by row: thus, palatograms (g) and (h) have the same CI.

4. The greater the number of contacts, the higher CI – but only up to the limit specified by 2. above.

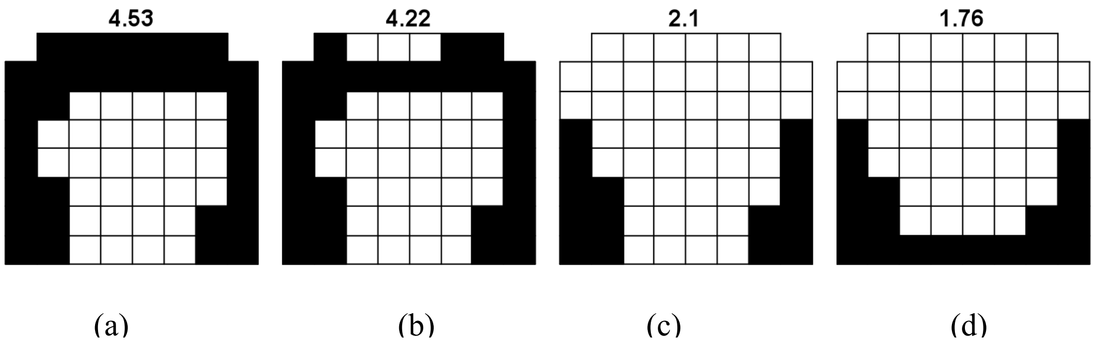


Fig. 7.16: Palatograms with corresponding centre of gravity values shown above.

The **dorsopalatal index** (DI) also varies between 0 and 1 and is a measure of the extent of contact in the last three rows, i.e. in the palatal and post-palatal region. It is a simple proportional measure: when all 24 electrodes are contacted in rows 6-8, then DI has a value of 1; if 12 are contacted, then DI is 0.5, etc.

Finally, the **centre of gravity index** (COG) is a measures the distribution of the place of articulation between the front and back of the palate: further advanced/retracted places of articulation are associated with higher/lower COG values. COG varies between 7.6 (when row 1 alone is filled) to 0.5 (when row 8 alone is filled). COG is calculated from a weighted average of the sum of contacts in the rows, where the weights on rows 1-8 are 7.6, 6.5…0.5. For example, for palatogram (c) in Fig. 7.16, COG is calculated as follows:

```
# Sum of the contacts in rows 1-8 for (c) in Fig. 7.16
contacts = c(0, 0, 0, 2, 2, 3, 4, 4)
# Weights on rows 1-8
```

```
weights = seq(7.6, 0.5, by = -1)
# COG for (c)
sum(contacts * weights)/sum(contacts)
2.1
```

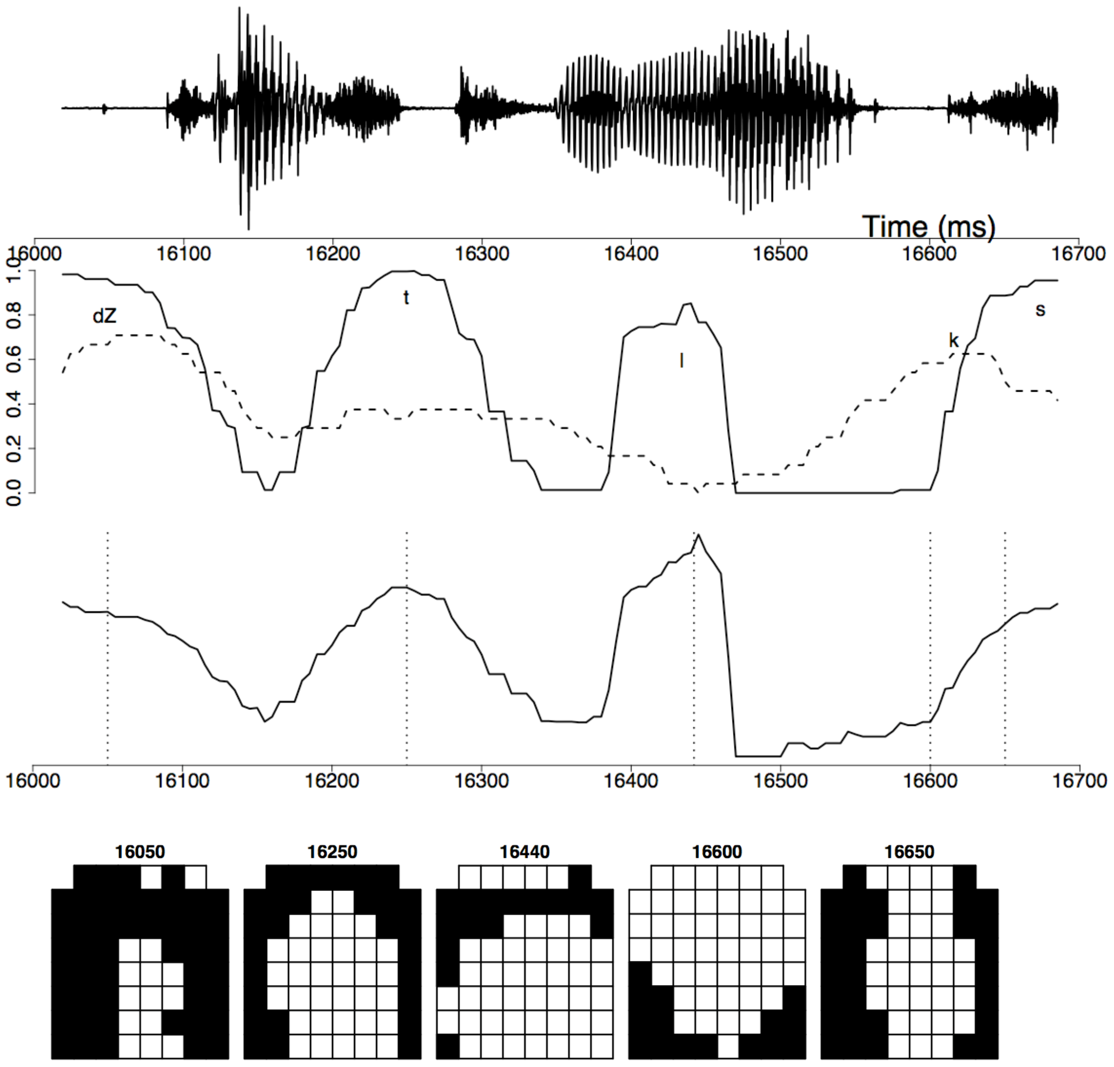


Fig. 7.17: Synchronised waveform (top) anteriority index (middle panel, solid), dorsopalatal index (middle panel, dashed), centre of gravity (lower panel) for *just relax*. Some palatograms that occur closest to the time points marked by the vertical dotted lines in the figure of the third row (in [ʤ] and [t] of *just* and [l], [k], [s] of *relax* respectively) are in the bottom row.

n Fig 7.16, (a) and (b) have the same contact patterns, except that in (b) some contacts in the first row are missing. Thus, the overall distribution of contacts is further towards the front in (a) than in (b) and so COG is higher. (c) and (d) have no contacts in the first three rows and so have lower COG values than those of either (a) or (b). Finally (c) and (d) have the same pattern of contacts except that in (d) the last row is filled: consequently the overall distribution of contacts in (d) is furthest towards the back of all the palatograms and so it has the lowest COG of all.

An example of how AI, DI and COG vary is shown for the first two words *just relax* from the **coutts** database considered earlier in Fig. 7.16. AI, DI, and COG for the first two segments are obtained as follows:

```
ai = epgai(coutts.epg[1:2,])
di = epgdi(coutts.epg[1:2,])
cog = epgcog(coutts.epg[1:2,])
```

The plot in Fig. 7.17 is then given by:

```
# This save the default parameters for the plotting device so that
# the defaults are restored after plotting (see last command below)
oldpar = par(no.readonly=TRUE)
# Set the display for a 3 x 1 plot and define the margins
par(mfrow=c(3,1)); par(mar=c(1, 2, 1, 1))
# Waveform
plot(coutts.sam[1:2,], type="l", axes=F, ylab="Amplitude")
axis(side=1)
mtext("Time (ms)", side=1, at=16600, line=-1)
# AI and DI
plot(cbind(ai, di), type="l", axes=F)
axis(side=2, line=-1)
# Some superimposed labels
text(c(16048, 16250, 16434, 16616, 16674), c(0.80, 0.88, 0.60, 0.69, 0.82), c("dZ", "t", "l",
"k", "s"))
# COG
plot(cog, type="l", axes=F, ylab="COG")
axis(side=1)
# Mark in some time values
times = c(16050, 16250, 16442, 16600, 16650)
abline(v=times)
# Restore the plotting device defaults
par(oldpar)

# Plot palatograms at these time values
epgplot(coutts.epg[1:2,], times, mfrow=c(1,5))
```

The contact profiles in Fig. 7.17 lead to the following conclusions:

- AI is somewhat lower for the lateral of *relax* than either [ʤ] (`dZ`) or [st] of *just* because, in contrast to these segments, [l] has only one contact in the first row, as the palatogram at 16440 ms shows.

- DI has a high value during [ʤ] and this is because, as the palatogram at 16050 ms shows, there is quite a lot of contact in the back three rows.

- COG often tends to track AI quite closely and this is also evident for the data in figure 7.17. However unlike AI, COG takes account of the overall distribution of the contacts from front to back; and unlike AI, COG is not biased towards giving a higher ranking if there is a single contact in a low row number. Therefore, because the two leftmost palatograms in Fig 7.17 have contacts in the first row, they have high AI values and higher than those of the 3rd palatogram from the left at 16440 ms during [l]. But of these three, the leftmost palatogram at 16050 ms has the lowest COG because of the large number of contacts in the back rows.

Finally, some of these data-reduction parameters with be applied to the Polish fricatives considered earlier. For this analysis, the data from the alveolo-palatal [ɕ] is included as well as from [s,ʃ]. Here again is a grayscale palatographic display, this time averaged over the middle third of each fricative:

```
par(mfrow=c(1,3))
for(j in c("s", "S", "c")){
temp = polhom.l == j
epggs(dcut(polhom.epg[temp,], .33, .67, prop=T), main=j)
}
```

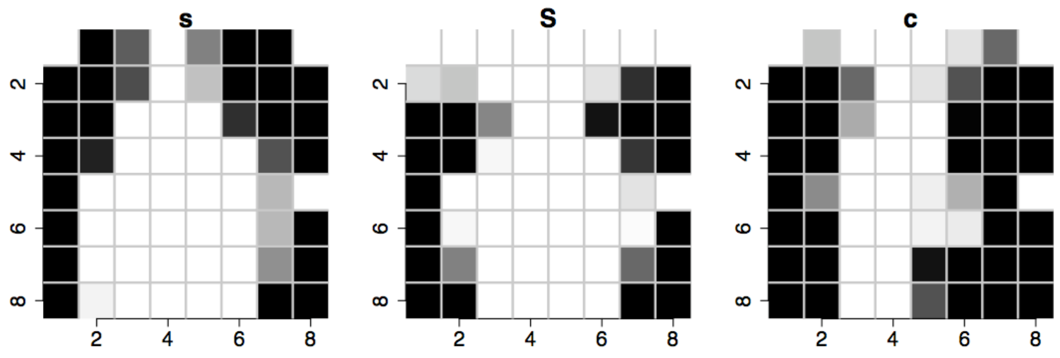


Fig. 7.18: Greyscale images for 10 tokens each of the Polish fricatives [s,ʃ,ɕ].

These greyscale palatographic displays in Fig. 7.18 can be used to make various predictions about how these three places of articulation might be separated on some of the EPG data reduction parameters:

- AI: highest for [s] (greatest number of contacts in rows 1 and 2) and possibly higher for [ɕ] than for [ʃ] (more contacts in rows 1-2).
- DI: highest for [ɕ] (greatest number of contacts in rows 5-8)
- CI: lowest for [ʃ] (least number of contacts medially in columns 4-5)
- COG: highest for [s], possibly with little distinction between [ʃ] and [ɕ] since the distribution of contacts from front to back is about the same for these fricatives.

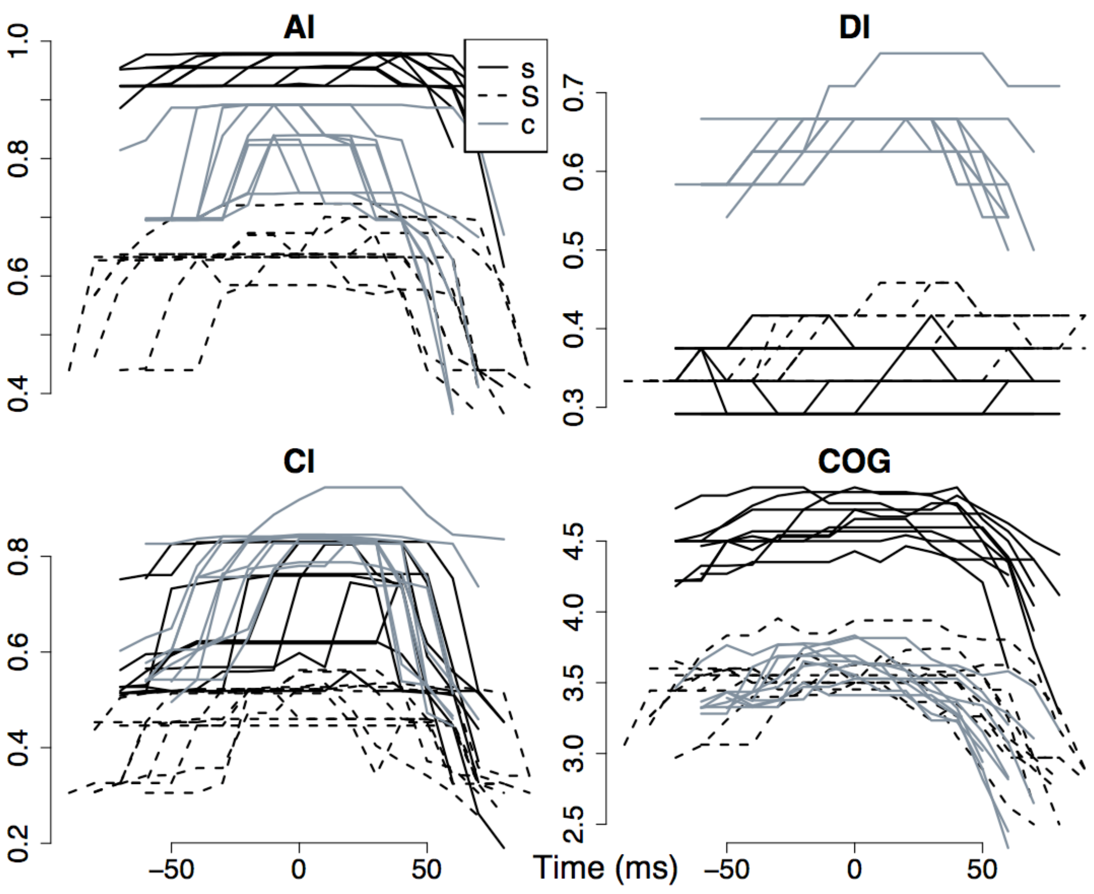


Fig. 7.19: Anteriority (AI), dorsopalatal (DI), centrality (CI), and centre of gravity (COG) indices for 10 tokens of the Polish fricatives [s,ʃ,ɕ] (solid, dashed, gray) synchronised at their temporal midpoints.

In the example in Fig. 7.19, these parameters were calculated across the entire temporal extent of the homorganic fricatives. Since the fricatives were flanked by vowels, then the parameters might be expected to rise towards the temporal midpoint in most cases. The commands for creating Fig. 7.19 are as follows.

```
# AI, DI, CI, COG
ai = epgai(polhom.epg);  di = epgdi(polhom.epg)
ci = epgci(polhom.epg);   cog = epgcog(polhom.epg)
# Logical vector to identify the three fricatives
temp = polhom.l %in% c("s", "S", "c")

par(mfrow=c(2,2)); par(mar=c(1,2,1.3,1))
dplot(ai[temp,], polhom.l[temp], offset=.5, axes=F, main="AI", bty="n")
axis(side=2)
dplot(di[temp,], polhom.l[temp], offset=.5, axes=F, legend=F, main="DI", bty="n")
axis(side=2)
dplot(ci[temp,], polhom.l[temp], offset=.5, legend=F, axes=F, main="CI", bty="n")
axis(side=1, line=-1); axis(side=2)
```

```
mtext("Time (ms)", side=1, at=120, line=0)

dplot(cog[temp,], polhom.l[temp], offset=.5, legend=F, axes=F, main="COG", bty="n")
axis(side=1, line=-1); axis(side=2)
```

Three of the parameters distinguish one fricative category from the other two: thus DI separates [ɕ] from [s,ʃ], CI separates [ʃ] from [s,ʃ,ɕ], COG separates [s] from [ɕ,ʃ] while AI produces a clear distinction between all three categories.

7.4. Analysis of EPG data

The mechanisms are now in place to carry out many different kinds of analysis using electropalatographic data. Two common kinds of investigation to which an EPG-analysis is particularly suited are presented in this section: an investigation into the extent of consonant overlap in alveolar-velar consonant clusters (7.4.1) ; and vowel-induced place of articulation variation in dorsal fricatives and stops (7.4.2).

7.4.1 Consonant overlap

The database fragment in this section is part of a larger database that was collected and analysed by Lisa Stephenson (Stephenson, 2003, 2004, 2005; Stephenson & Harrington, 2002) in studying consonant overlap in the production of blends in English and Japanese. In her experiments, subjects saw two hypothetical town names on a screen and had to produce a blend from the two words as quickly as possible after seeing them. They might see for example *Randon* and *Pressgate* and the task was to produce a blend by combining the first syllable of the first word with the second syllable of the second word, thus *Rangate*.

Stephenson's database included a number of blends formed with combinations of /n/ and a following consonant and in the analysis in this section, two of these types will be compared: blends formed with /nk, nɡ/ and blends formed with /sk, sɡ/ clusters. No differentiation will be made between the voicing status of the final consonant: so the comparison is between /nK/ vs. /sK/ where /K/ stands for either /k/ or /ɡ/. The question that is addressed is the following: is the extent of alveolar-velar overlap the same in /nK/ and /sK/?

As an initial hypothesis, it is reasonable to expect more overlap in /nK/ for at least two reasons. Firstly because of the well-known tendency for /n/ to assimilate in this context (see e.g., Hardcastle, 1994) whereas /s/ does not audibly retract its place of articulation in e.g., *mascot* or *must get* and is often resistant to coarticulatory influences (e.g., Recasens, 2004). Secondly, whereas it is quite possible to sustain an alveolar [n] production when there is tongue-dorsum contact at the velum for [k] or for [ɡ], this type of overlapping or double-articulation is likely to be more difficult in [sk] or [sɡ]: this is because if there is substantial velar closure during the production of the alveolar, then the airflow through the oral cavity will be inhibited as a result of which it will be difficult to sustain the high aerodynamic power required for the production of the sibilant fricative [s].

The available database fragment is **engassim** and there are the usual sets of parallel R objects associated with this:

engassim	Segment list from the acoustic onset to the acoustic offset of the entire [nk,ng,sk,sg] sequences
engassim.l	Label vector of the above. `nK` for [nk, nɡ] vs. `sK` for [sk, sɡ].
engassim.w	Label vector of the words from which the sequences were derived.
engassim.epg	Parallel EPG trackdata at a frame rate of 5 ms.

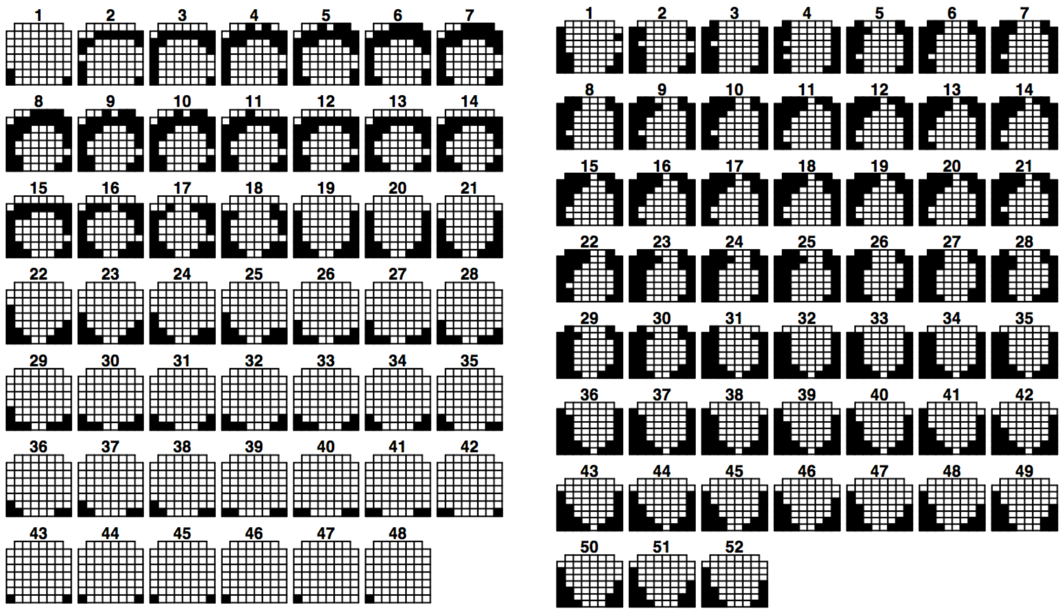


Fig. 7.20: Palatograms from the acoustic onset to the acoustic offset of /nk/ (left) in the blend *duncourt* and /sk/ (right) in the blend *bescan* produced by an adult female speaker of Australian English.

In 7.3 it was shown how the anteriority and dorsopalatal indices tend to provide positive evidence for productions at alveolar and velar places of articulation respectively. The data will therefore be analysed for these parameters, but as with any more complicated parametric analysis, it is always a good idea to look at some samples of the data first. A plot of all of the `nK` data separately per segment and from the onset of the [n] to the offset of the velar can be produced as follows. (Use the left-mouse button to advance through each plot; you will have to do this 17 times, since there are 17 `nK` segments. Use the same commands to get the corresponding `sK` data, but replace temp with its logical inverse !temp). The EPG-frames from the first run through the loop (for the first `nK` and `sK` segments) are shown in Fig. 7.20:

```
temp = engassim.l == "nK"
for(j in 1:sum(temp)){
# show palate numbers rather than times
epgplot(engassim.epg[temp,][j,], numbering=T)
# left mouse button to advance
locator(1)
}
```

Palatograms from two segments are shown in Fig. 7.20: on the left is `nK` from *duncourt* and on the right, `sK` from *bescan*. For `nK` in *bescan*, the alveolar stricture increases from palatogram 2. It is complete by palatogram 6 and the release of the alveolar occurs 50 ms after that by frame 16. The same display shows how the velar closure begins to form during this interval such that the maximum visible extent of velar closure takes place by frame 16. Evidently then, although the alveolar and velar articulations are not simultaneous (i.e. are not completely doubly articulated), they overlap a good deal. Consider now the `sK` data on the right of Fig. 7.20. The alveolar constriction for the [s] extends approximately over 115 ms between roughly palatograms 10 and 23, but the greatest degree of narrowing for the velar stop /k/ does not take place until well after this at frame 31.

The aim now is to see whether there is any evidence for a greater extent of alveolar-velar overlap in `nK` in all of the data, using anteriority and dorsopalatal indices to parameterise the extent of contact at the front and at the back of the palate respectively:

```
# Anteriority and dorsopalatal indices for all of the data from
# segment onset to segment offset
ai = epgai(engassim.epg); di = epgdi(engassim.epg)

par(mfrow=c(1,2))
temp = engassim.l == "nK"
# data for nK
dplot(ai[temp,], ylim=c(0,1), main="/nK/", lwd=2, leg=F)
par(new=T)
dplot(di[temp,], ylim=c(0,1), col="slategray", lwd=2, leg=F)
# data for sK
dplot(ai[!temp,], ylim=c(0,1), main="/sK/", lwd=2, leg=F)
par(new=T)
dplot(di[!temp,], ylim=c(0,1), col="slategray", lwd=2, leg=F)
```

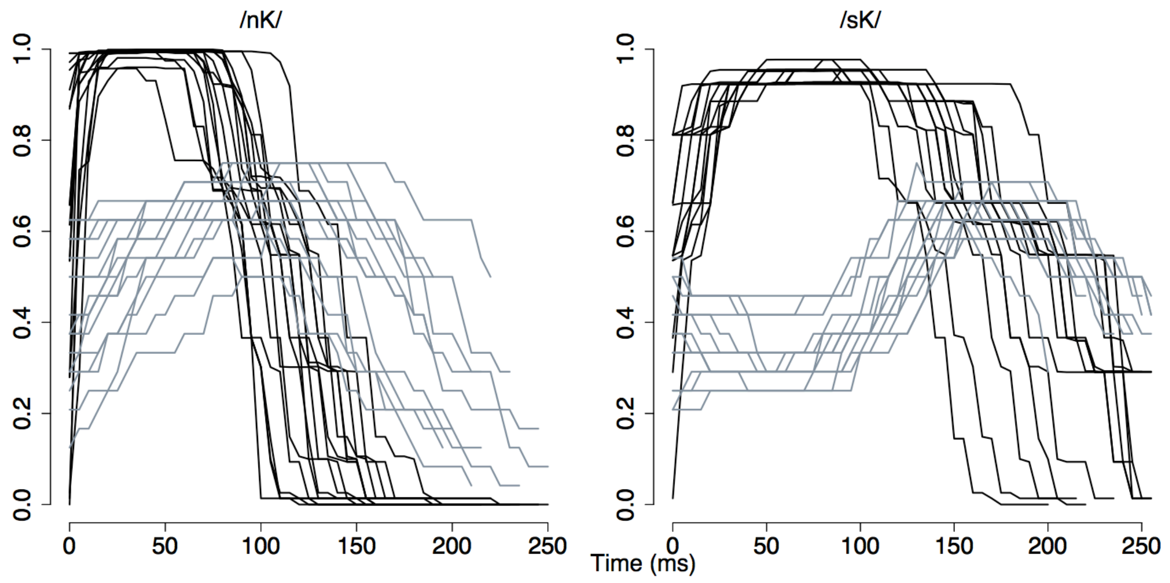


Fig. 7.21: Anteriority (black) and dorsopalatal (gray) indices for 17 /nK/ (left) and 15 /sK/ (right) sequences (K= /k,g/) produced by an adult female speaker of Australian English.

It is apparent from Fig. 7.21 that the tongue-dorsum activity for [k] is timed to occur a good deal earlier relative to the preceding consonant in the clusters with [n] compared with those of [s]. In particular, the left panel of Fig. 7.20 shows how the dorsopalatal index rises throughout the AI-plateau for [n]; by contrast, there is a dorsopalatal trough for most of the AI-plateau for [s] between roughly 40 ms and 100 ms on the right.

The differences in the extent of alveolar-velar overlap could be further highlighted by producing grayscale EPG-images at about 50 ms after the acoustic onset of the consonant cluster – which, as Fig. 7.20 shows, is roughly the time at which the AI maxima are first attained in /nK/ and /sK/.

```
par(mfrow=c(1,2))
temp = engassim.l == "nK"
epggs(dcut(engassim.epg[temp,], start(engassim[temp,])+50), main="/nK/")
epggs(dcut(engassim.epg[!temp,], start(engassim[!temp,])+50), main="/sK/")
```

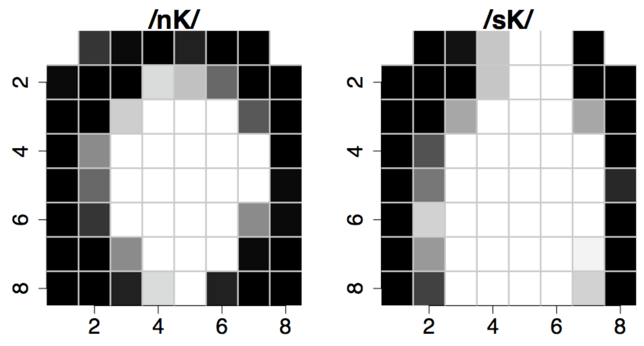


Fig. 7.22: Grayscale EPG images for the /nK/ (left) and the /sK/ (right) for the data in Fig. 7.21 extracted 50 ms after the acoustic segment onset of the cluster.

The grayscale images in Fig. 7.22 show greater evidence of alveolar-velar overlap for /nK/ which, in contrast to /sK/ has more filled cells in the last two rows.

7.4.2 VC coarticulation in German dorsal fricatives

The analysis in this section is concerned with dorsal fricative assimilation in German and more specifically with whether the influence of a vowel on the following consonant is greater when the consonant is a dorsal fricative, which, for compatibility with the MRPA, will be denoted phonemically as /x/, compared with an oral stop, /k/. This analysis was carried out in a seminar at the IPDS, University of Kiel, and then further developed in a paper by Ambrazaitis & John (2004).

In German, a post-vocalic dorsal fricative varies in place of articulation depending largely on the backness of a preceding tautomorphemic vowel. After front vowels, /x/ is produced in Standard German and many German dialects as a palatal fricative (e.g., [ri:ç], [lɪçt], [pɛç]; *riech*/*smell*, *Licht*/*light*; *Pech*/*bad luck* respectively), as a velar fricative after high back vowels (e.g. [bu:x], *Buch*/*book*) and quite possibly as a uvular fricative after central or back non-high vowels (e.g., [maχ], *make*; [lɔχ], *Loch*/*hole*). In his extensive analysis of German phonology, Wiese (1996) raises the interesting point that, while this type of vowel-dependent place of articulation in the *fricative* is both audible and well-documented, the same cannot be said for analogous contexts with /k/. Thus, there are tautomorphemic sequences of /i:k, ɪk, ɛk/ (*flieg*/*fly*; *Blick*/*view*; *Fleck*/*stain*), and of /u:k, ɔk/ (*Pflug*/*plough*; *Stock*/*stick*) and of /ak/ (*Lack*/*paint*), but it not so clear either auditorily nor from any experimental analysis whether there is the same extent of allophonic variation between palatal and uvular places of articulation.

We can consider two hypotheses as far as these possible differences in coarticulatory influences on /x/ and /k/ are concerned. Firstly, if the size of coarticulatory effects is entirely determined by the phonetic quality of the preceding vowel, then there is indeed no reason to expect there to be any differences in the variation of place of articulation between the fricative and the stop: that is, the extent of vowel-on-consonant coarticulation is the same for both, but perhaps the coarticulatory variation is simply much less audible in the case of /k/ because the release of the stop, which together with the burst contains most of the acoustic cues to place of articulation, is so much shorter than in the fricative. The alternative hypothesis is that there is a categorical distinction between the allophones of the fricative, but not of /k/. Under this hypothesis, we might expect not only a sharper distinction in speech production between the front and back allophones of the fricative but also that the variation *within* the front or back allophones might be less for the fricative than the stop.

It is certainly difficult to answer this question completely with the available fragment of the database of a single speaker, but it is nevertheless possible to develop a methodology that could be applied to many speakers in subsequent experiments. The database fragment here is **dorsal** that forms part of a corpus recorded by phonetics students at the IPDS Kiel in 2003 and also part of the study by Ambrazaitis & John (2004). The EPG data was recorded at a frame rate of 10 ms at the Zentrum für allgemeine Sprachwissenschaft in Berlin. For the recording, the subject had to create and produce a street name by forming a blend of a hypothetical town name and a suffix that were shown simultaneously on a screen. For example, the subject was shown RIEKEN and –UNTERWEG at the same time, and had to produce RIEKUNTERWEG as quickly as possible. In this example, the underlined part of this blend includes /i:kʊ/. Blends were formed in an analogous way to create /V_1CV_2/ sequences where V_1 included vowels varying in backness and height, C = /k, x/, and V_2 = /ʊ, ɪ/. In all cases, primary stress is necessarily on V_2. To take another example, the subject produced RECHINSTERWEG in response to RECHEN and –INSTERWEG resulting in a blend containing /ɛxɪ/ over the underlined segments.

For the database fragment to be examined here, there are seven different V_1 vowels whose qualities are close to IPA [i:, ɪ, ɛ, a, ɔ, ʊ] (MRPA `i, I, E, a, O, U` respectively in this database) and that vary phonetically in backness more or less in the order shown. So assuming that the following V_2 = /ʊ, ɪ/ has much less influence on the dorsal fricative than the preceding vowel, which was indeed shown to be the case in Ambrazaitis & John (2004), we can expect a relatively front allophone of the fricative or stop after the front vowels [i:, ɪ, ɛ] but a back allophone after [ɔ, ʊ].

The following parallel objects are available for investigating this issue. With he exception of `dorsal.bound` which marks the time of V_1C acoustic boundary, their boundary times extend from the acoustic onset V_1 to the acoustic offset of C (the acoustic offset of the dorsal:

dorsal	Segment list of V_1C (C = /k, x/)
dorsal.epg	EPG-compressed trackdata of dorsal
dorsal.sam	sampled waveform trackdata of dorsal
dorsal.fm	Formant trackdata of dorsal
dorsal.vlab	Label vector of V_1 ("i", "I", "E", "a", "O", "U")
dorsal.clab	Label vector of C (k or x)
dorsal.bound	Event times of the acoustic V_1C boundary

There were 2 tokens per /V_1CV_2/ category (2 tokens each of /i:kɪ/, /i:kʊ/, /i:xɪ/, /i:xʊ/… etc.) giving 4 tokens for each separate V_1 in /V_1k/ and 4 tokens per /V_1x/ (although since V_1 was not always realised in the way that was intended - e.g., /ɪ/ was sometimes produced instead of /i:/ - there is some deviation from this number, as table(label(dorsal)) shows). In order to be clear about how the above R objects are related, Fig. 7.23 shows the sampled waveform and electropalatographic data over the third segment in the database which (as label(dorsal[3,]) shows) was `ak`[3]:

```
plot(dorsal.sam[3,], type="l",main="ak", xlab="Time (ms)", ylab="",
axes=F, bty="n")
epgplot(dorsal.epg[3,], mfrow=c(2,8))
```

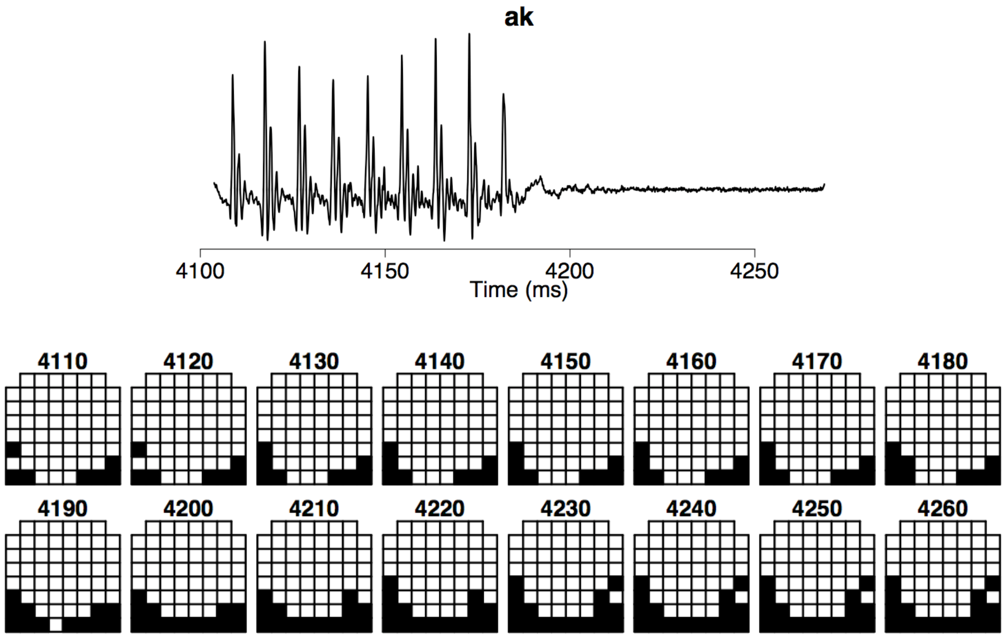


Fig. 7.23: Acoustic waveform (top) of /ak/ produced by an adult male speaker of standard German and the palatograms over the same time interval.

For the investigation of the variation in place of articulation in dorsal consonants, the anteriority index is not appropriate because this only registers contact in rows 1-5. The dorsopalatal index might shed more light on place of articulation variation – however, given that it is based on summing the number of contacts in the back three rows, it is likely to distinguish between the lesser stricture of the fricatives than the stops. But this is not what is needed. Instead, we need a parameter that is affected mostly by shifting the tongue from front to back along the palate and which does so in more or less the same way for the fricative and the stop categories.

The parameter that is most likely to be useful here is the EPG centre of gravity (COG) which should show decreasing values as the primary dorsal stricture moves back along the palate. COG should also show a predictable relationship by vowel category. It should be highest for a high front vowel like [i:] that tends to have a good deal of contact laterally in the palatal region and decrease for [ɪ,ɛ] which have a weaker palatal contact. It should have the lowest values for [ʊ,ɔ] in which any contact is expected at the back of the palate.

The EPG-COG parameter should show some relationship to the vowel's second formant frequency, since F2 of [i:] is higher than F2 of [ɪ,ɛ] and since of course F2 of front

[3] The waveform and EPG-data have to be created as separate plots in the current implementation of Emu-R.

vowels is greater than F2 of low, central and back vowels. These relationships between COG, vowel category and F2 can be examined during the interval for which sensible formant data is available, i.e., during the voiced part of the vowel. Given that the interest in this analysis is in the influence of the vowel on the following consonant, we will consider data extracted at the vowel-consonant boundary close the vowel's last glottal pulse, i.e. close to the time point at which the voiced vowel gives way to the (voiceless) fricative or stop. Two different types of COG will be presented. In one, COG is calculated as in section 7.3 over the entire palate: in the other, which will be called the **posterior centre of gravity** (P-COG), the COG calculations are restricted to rows 5-8. P-COG is relevant for the present investigation because the study is concerned exclusively with sounds made in the dorsal region, i.e., with vowels followed by dorsal consonants. It should be mentioned at this point that this version of P-COG is not quite the same as the one in Gibbon & Nicolaidis (1999) who restrict the calculations not only to rows 5-8 but also to columns 3-6 (see the picture on the jacket cover of Hardcastle & Hewlett, 1999), i.e. to a central region of the palate. However, this parameter is likely to exclude much of the information that is relevant in the present investigation, given that the distinction between high front and back vowels often shows up as differences in *lateral* tongue-palate contact (present for high front vowels, absent for back vowels), i.e. at the palatographic margins.

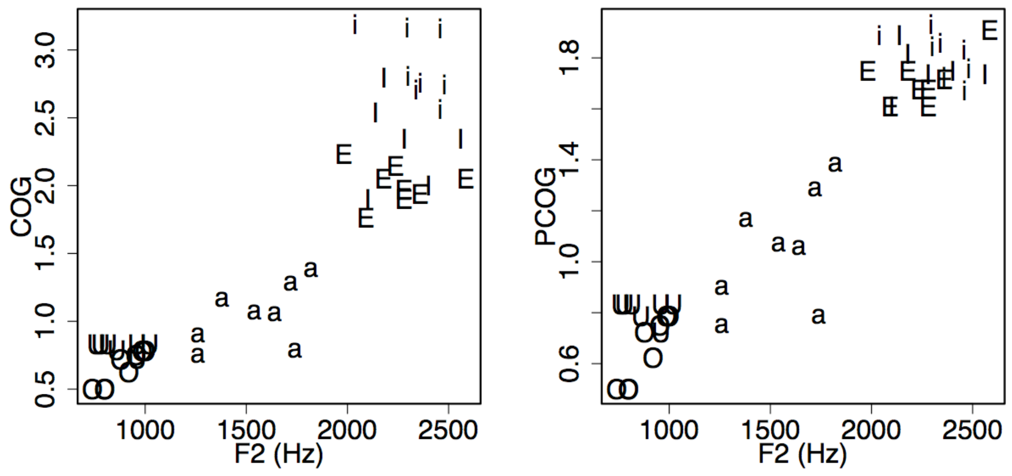


Fig. 7.24: The EPG centre of gravity (left) and EPG posterior centre of gravity (right) plotted as a function of F2 for data taken at the acoustic vowel offset for data pooled across /x/ and /k/.

The relationship between the centre of gravity parameters and F2 at the acoustic vowel offset is shown in Fig. 7.24 which was created with the following commands:

```
# COG and PCOG, from the onset to the offset of VC
cog = epgcog(dorsal.epg); pcog = epgcog(dorsal.epg, rows=5:8)
# COG and PCOG at the VC boundary
cog.voffset = dcut(cog, dorsal.bound)
pcog.voffset = dcut(pcog, dorsal.bound)
# F2 at the VC boundary
f2.voffset = dcut(dorsal.fm[,2], dorsal.bound)
par(mfrow=c(1,2))
plot(f2.voffset, cog.voffset, pch=dorsal.vlab, xlab="F2 (Hz)", ylab="COG")
plot(f2.voffset, pcog.voffset, pch=dorsal.vlab, xlab="F2 (Hz)", ylab="PCOG")
```

As Fig. 7.24 shows, both COG and PCOG show a fairly linear relationship to the second formant frequency at the vowel offset, as well as a clear separation between vowel categories, with the low back vowels appearing at the bottom left of the display and the high and mid-high front vowel in the top right. For this particular speaker, these relationships between acoustic data, articulatory data and vowel category emerge especially clearly. It must be emphasised that this will not always be so for all speakers! PCOG shows a slightly better correlation with the F2-data than COG (as `cor.test(f2.voffset, pcog.voffset)` and `cor.test(f2.voffset, cog.voffset)` show) but then COG shows a clearer distinction within the front vowel categories [i:,ɪ,ɛ] – and this could be important in determining whether the coarticulatory influences of the vowel on the consonant are more categorical for /x/ than for /k/ (in which case, we would expect less variation in /x/ following these different front vowels, if /x/ is realised as basically the same front allophone in all three cases). The subsequent analysis are all based on COG – some further analysis with PCOG is given in the exercises.

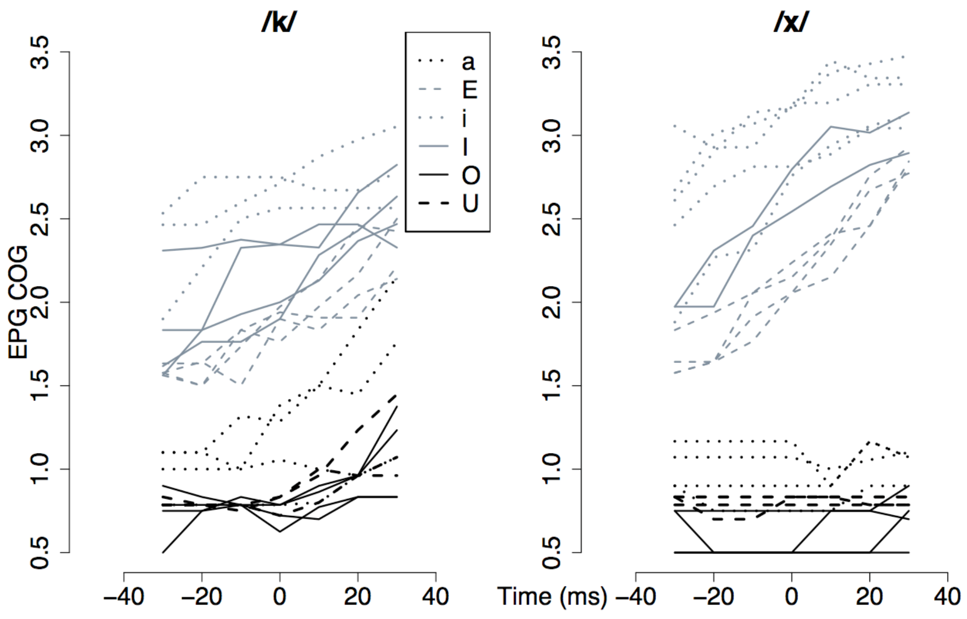


Fig. 7.25: The EPG centre of gravity calculated 30 ms on either side of the acoustic V_1C boundary for /k/ (left) and /x/ (right) shown separately as a function of time by V_1 category.

In order to get some insight into how /k,x/ vary with the preceding vowel context, a plot of COG will be made 30 ms on either side of the vowel boundary. This is shown in Fig. 7.25 and was produced as follows:

```
# Cut the EPG-data to ±30 ms either side of V1C boundary
epg30 = dcut(dorsal.epg, dorsal.bound-30, dorsal.bound+30)
# Calculate COG
cog30 = epgcog(epg30)
```

```
# Logical vector that is True when the consonant is /k/ as opposed to /x/
temp = dorsal.clab=="k"
ylim = c(0.5, 3.5); xlim=c(-50, 50)
par(mfrow=c(1,2))
dplot(cog30[temp,], dorsal.vlab[temp], offset=.5, xlim=xlim, ylim=ylim, leg="topright",
ylab="EPG COG", main="/k/", bty="n")
mtext("Time (ms)", side=1, line=1, at=70)
dplot(cog30[!temp,], dorsal.vlab[!temp], offset=.5, xlim=xlim, ylim=ylim, leg=F, main="/x/",
bty="n")
```

As Fig. 7.25 shows, there is a clearer separation (for this speaker at least) on this parameter between the front vowels [i, ɪ, ɛ] on the one hand and the non-front /a, ɔ, ʊ/ in the context of /x/ but much less so in the context of /k/. A histogram of the EPG-COG values at 30 ms after the acoustic VC boundary brings out the greater categorical separation between these allophone groups preceding /x/ quite clearly.

```
# COG values at 30 ms after the VC boundary. Either:
cog30end = dcut(cog30, 1, prop=T)
# Or
cog30end = dcut(cog30, dorsal.bound+30)

# Logical vector, T when clab is /k/, F when clab is /x/
temp = dorsal.clab=="k"
par(mfrow=c(1,2))
# Histogram of EPG-COG 30 ms after the VC boundary for /k/
hist(cog30end[temp], main="/k/", xlab="EPG-COG at t = 30 ms", col="blue")
# as above but for /x/
hist(cog30end[!temp], main="/x/", xlab="EPG-COG at t = 30 ms", col="blue")
```

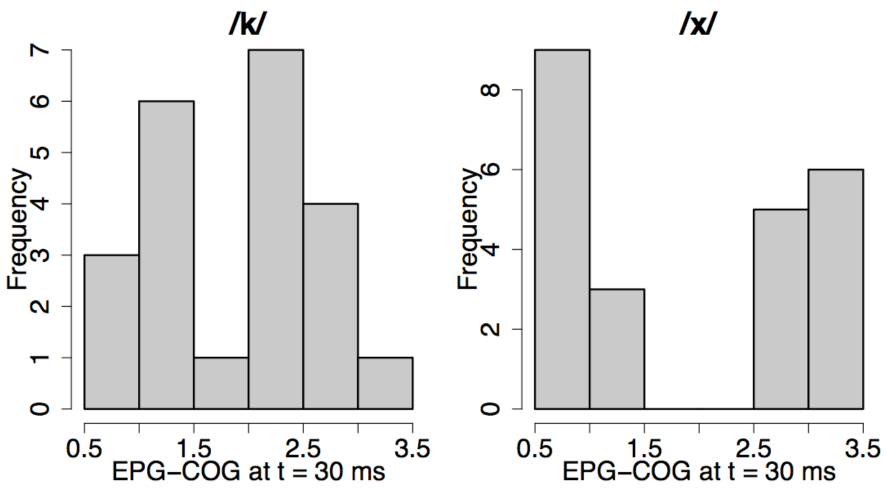


Fig. 7.26: EPG-centre of gravity for /k/ (left) and /x/ (right) at the V_1C boundary.

There is evidently a bimodal distribution of EPG-COG 30 ms after the VC boundary for both /x/ and /k/, but this is somewhat more pronounced for /x/: such a finding is consistent with the view that there may be a more marked separation into front and non-front allophones for /x/

than for /k/. In order to test this hypothesis further, the EPG-COG data are plotted over the extent of the consonant (over the fricative or the stop closure) in Fig. 7.26:

```
# Centre of gravity from acoustic onset to offset of the consonant
cogcons = epgcog(dcut(dorsal.epg, dorsal.bound, end(dorsal.epg)))
# Logical vector that is True when dorsal.clab is k
temp = dorsal.clab=="k"
par(mfrow=c(1,2)); ylim = c(0.5, 3.5); xlim=c(-60, 60)
col = c(1, "slategray", "slategray", 1, 1, "slategray")
linet=c(1,1,5,5,1,1) ; lwd=c(2,2,1,1,1,1)
dplot(cogcons[temp,], dorsal.vlab[temp], offset=.5, leg="topleft", ylab="COG", ylim=ylim,
xlim=xlim, main="/k/", col=col, linet=linet, lwd=lwd)
dplot(cogcons[!temp,], dorsal.vlab[!temp], offset=.5, ylim=ylim, xlim=xlim, leg=F,
main="/x/", col=col, linet=linet, lwd=lwd)
```

There is once again a clearer separation of EPG-COG in /x/ depending on whether the preceding vowel is front or back. Notice in particular how EPG COG seems to climb to a target for /ɛx/ and reach a position that is not very different from that for /ix/ or /ɪx/.

For this single speaker, the data does indeed suggest a greater categorical allophonic distinction for /x/ than for /k/.

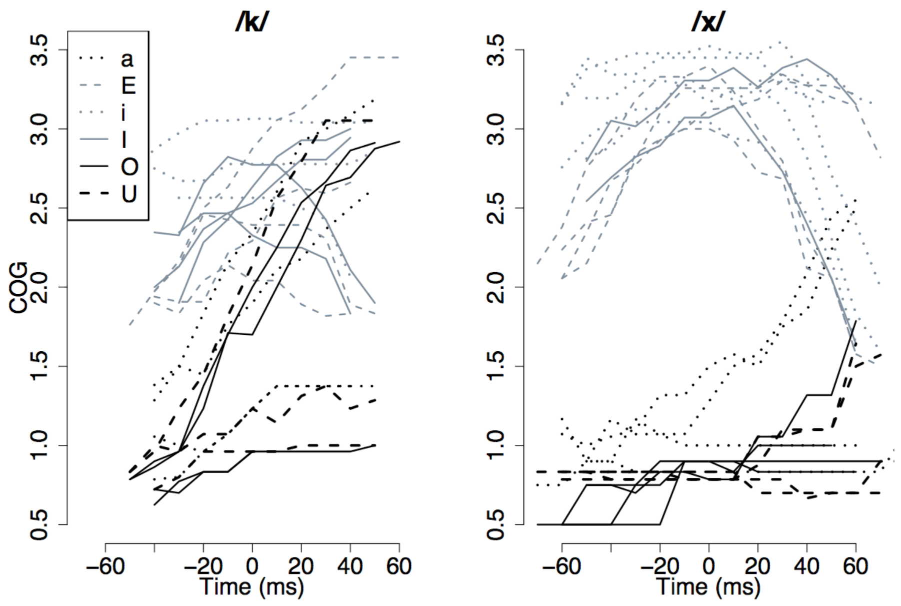


Fig. 7.27: EPG-COG data over the extent of /k/ closure (left) and /x/ frication (right) shown by vowel category and synchronised at the consonants' acoustic temporal midpoints.

7.5. Summary

One of the central concerns in experimental phonetics is with how segments overlap and are coordinated with each other. The acoustic speech signal provides a rich source of information allowing these types of processes in speech production to be inferred indirectly. However, it is clear that acoustics is of little use for the kind of study presented in the latter part of this Chapter in analysing how the tongue moves due to the influence of context during an acoustic stop closure. Also, it is very difficult and probably impossible to quantify reliably from speech acoustics the way in which the tongue is repositioned from an alveolar to a velar position in the kinds of /nk/ sequences that were examined earlier, largely because this kind of subtle change is very difficult to detect in the acoustics of nasal consonants. Moreover, an acoustic analysis could not reveal the differences in segmental coordination between /sk/ compared with /nk/ that were in evidence in analysing these productions electropalatographically.

As discussed earlier, electropalatography is much more limited compared with a technique like electromagnetic articulometry (EMA) presented in Chapter 5, because it cannot provide as much information about the dynamics of tongue movement; and EPG in comparion with EMA has little to offer in analysing vowels or consonants produced beyond the hard/soft palate junction. On the other hand, EPG tracks can often be more transparently related to phonetic landmarks than the data from EMA, although critics of EPG also argue (not unjustifiably) that the EPG parameters like AI, DI, and COG are too simplistic for inferring the complexities of speech motor control.

Although there are necessarily specialised functions for displaying palatograms and although handling palatograms required an extension to arrays that is not necessary for most kinds of acoustic analysis, a central aim in this Chapter has been to show how many of the procedures for handling acoustic data in R can be applied to data-reduced versions of the EPG signal. Thus the tools for plotting and quantifying EPG data are, for the most part, the same as those that were used in the analysis of movement and formant data in the preceding two Chapters and for spectral data to the discussed in the next Chapter. As a result, the mechanisms are in place for carrying out various kinds of articulatory-acoustic relationships, of which one example was provided earlier (Fig. 7.24). In addition, the extensive resources for quantifying data that are available from the numerous R libraries can also be applied to further analyses of palatographic data.

7.6 Questions

1. Make a 3D palatographic array for creating the figure in Fig. 7.14, then plot Fig. 7.14 and use the made-up array to verify the values for the anteriority index.

2. Write R commands to display the 1st, 4th, and 7th palatograms at the acoustic temporal midpoint of [ɕ] (MRPA c) in the polhom database fragment.

3. The database fragment **coutts2** contains the same utterance produced by the same speaker as **coutts** but at a slower rate. The R-objects for **coutts2** are:

coutts2	segment list of words
coutts2.l	vector of word labels
coutts2.epg	EPG-compressed trackdata object
coutts2.sam	Trackdata of the acoustic waveform

Produce palatographic plots over a comparable extent as in Fig. 7.5 from the /d/ of *said* up to the release of /k/ in *said Coutts*. Comment on the main ways the timing of /d/ and /k/ differ in the normal and slow database fragments.

7. For the **polhom** data set of Polish homorganic fricatives (segment list, vector of labels, and trackdata polhom, polhom.l, polhom.epg respectively), write R-expressions for the following:
7.1 For each segment, the sum of the contacts in rows 1-3.

7.2 For each segment, the sum of all palatographic contacts at 20 ms after the segment onset.

7.3 For each segment, the sum of the contacts in rows 1-3 and columns 1-2 and 7-8 at the segment midpoint.

7.5 For each `s` segment, the anteriority index at the segment offset.

7.6 For each `s` and `S` segment, the dorsopalatal index 20 ms after the segment midpoint.

7.7 An ensemble plot as a function of time of the sum of the contacts in rows 2 and 4 for all segments, colour-coded for segment type (i.e., a different colour or line-type for each of `s`, `S`, `c`, `x`) and synchronised at the temporal midpoint of the segment.

7.8 An ensemble plot as a function of time of the sum of the inactive electrodes in columns 1, 2, 7, and 8 and rows 2-8 for all `S` and `c` segments for a duration of 40 ms after the segment onset and synchronised 20 ms after segment onset.

7.9 An averaged, and linearly time-normalized ensemble plot for `c` and `x` as a function of time of the posterior centre of gravity (PCOG).

7.10 For each segment, the median of the anteriority index between segment onset and offset.

7.11 A boxplot of the centre of gravity index averaged across a 50 ms window, 25 ms on either side of the segment's temporal midpoint, for `s` and `S` segments.

5. For the **engassim** dataset, the AI and DI indices were calculated as follows:

```
ai = epgai(engassim.epg); di = epgdi(engassim.epg)
```

Calculate over these data (a) the time at which AI first reaches a maximum value (b) the time at which DI first reaches a maximum value. Make a boxplot of the difference between these times, (b) – (a), to show that the duration between these two maxima is greater for sK than for nK.

7.7 Answers

1.

```
palai = array(0, c(8, 8, 8))
palai[1,2:7,1] = 1
palai[2,4,2] = 1
palai[2,,3] = 1
palai[2,8,4] = 1
palai[3,,5] = 1
palai[3:5,,6] = 1
palai[4,,7] = 1
palai[5,,8] = 1
class(palai) = "EPG"
aivals = round(epgai(palai), 4)
```

```
epgplot(palai, mfrow=c(1,8), num=as.character(aivals))
```

2.

```
# EPG data at the midpoint
polhom.epg.5 = dcut(polhom.epg, 0.5, prop=T)
# EPG data at the midpoint of c
temp = polhom.l == "c"
polhom.epg.c.5 = polhom.epg.5[temp,]
# plot of the 1st, 11th, 15th c segments at the midpoint
epgplot(polhom.epg.c.5[c(1,4,7),], mfrow=c(1,3))
```

3.

```
epgplot(coutts2.epg, xlim=c(14840, 15010))
```

The main difference is that in the slow rate, /d/ is released (at 14910 ms) well before the maximum extent of dorsal closure is formed (at 14935 ms), i.e., the stops are not doubly articulated.

7.1

```
epgsum(dcut(polhom.epg, 0, prop=T), r=1:3)
```

7.2

```
times = start(polhom)+20
epgsum(dcut(polhom.epg, times))
```

7.3

```
epgsum(dcut(polhom.epg, 0.5, prop=T), r=1:3, c=c(1, 2, 7, 8))
```

7.5

```
epgai(dcut(polhom.epg[polhom.l=="s",], 1, prop=T))
```

7.6

```
temp = polhom.l %in% c("s", "S")
times = (start(polhom[temp,])+end(polhom[temp,]))/2
epgdi(dcut(polhom.epg[temp,], times[temp]))
```

7.7

```
dplot(epgsum(polhom.epg, r=c(2,4)), polhom.l, offset=0.5)
```

7.8

```
# EPG-trackdata from the onset for 40 ms
trackto40 = dcut(polhom.epg, start(polhom.epg), start(polhom.epg)+40)
# Trackdata of the above but with rows and columns summed
esum = epgsum(trackto40, r=2:8, c=c(1, 2, 7, 8), inactive=T)
# Logical vector that is True for S or c
temp = polhom.l %in% c("S", "c")
# The EPG-SUM plot synchronised 20 ms after segment onset
dplot(esum[temp,], polhom.l[temp], offset=start(polhom.epg[temp,])+20, prop=F)
```

7.9

```
temp = polhom.l %in% c("c", "x")
dplot(epgcog(polhom.epg[temp,], rows=5:8), polhom.l[temp], norm=T, av=T)
```

7.10

```
trapply(epgai(polhom.epg), median, simplify=T)
```

7.11

```
# EPG-trackdata from the temporally medial 50 ms
midtime = (start(polhom.epg) + end(polhom.epg))/2
trackmid = dcut(polhom.epg, midtime-25, midtime+25)

# COG index of the above
cogvals = epgcog(trackmid)

# The mean COG value per segment over this interval
mcog = trapply(cogvals, mean, simplify=T)

# A boxplot of the mean COG for s and S
temp = polhom.l %in% c("S", "s")
boxplot(mcog[temp] ~ polhom.l[temp], ylab="Average COG")
```

5.

```
# Function for calculating the time at which the maximum first occurs
peakfun <- function(fr, fun=max)
{
temp = fr == fun(fr)
times = tracktimes(fr)
times[temp][1]
}

ai = epgai(engassim.epg); di = epgdi(engassim.epg)
# Get the times at which the AI- and DI-maxima first occur
aimax = trapply(ai, peakfun, simplify=T); dimax = trapply(di, peakfun, simplify=T)

boxplot(dimax - aimax ~ engassim.l, ylab="Duration (ms)")
```