Chapter 3. Analysis of formants and formant transitions

The focus of this Chapter is on some of the different techniques that are used to analyse formant frequencies and the way that formants change in time. The discussion is centred predominantly around vowels and the type of acoustic information that is available for distinguishing between them. Sections 3.1 – 3.3 are for the most part concerned with representing vowels in terms of their first two formant frequencies extracted at the vowel targets. A technique known as **kmeans clustering** for assessing the influence of context is briefly reviewed as well as some methods for locating vowel targets automatically from vowel formant data. Outliers that can arise as a result of formant tracking errors are discussed as well as methods for removing them.

As is well known, the formants of the same phonetic vowel vary not only because of context, but also due to speaker differences and in 3.4 some techniques of vowel normalisation are applied to some vowel data in order to determine how far they reduce the different formant characteristics of male and female vowels.

The final sections of this Chapter deal with vowel reduction, undershoot and coaticulatory influences. In 3.5, some metrics for measuring the Euclidean distance are introduced and applied to determining the expansion of the vowel space relative to its centre: this method is especially relevant for modelling the relationship between vowel positions and vowel hyperarticulation (e.g., Moon & Lindblom, 1994; Wright, 2003). But Euclidean distance measurements can also be used to assess how close one vowel space is to another and this is found to have an application in quantifying sound change that is relevant for sociolinguistic investigations.

Whereas all the techniques in 3.1-3.5 are static, in the sense that they rely on applying analyses to formants extracted at a single point in time, in sections 3.6 and 3.7 the focus in on the shape of the entire formant movement **as a function of time**. In 3.6, a parabola is fitted to a changing formant and it is shown how the coefficients of the parabola can be used to quantify vowel undershoot. As discussed in 3.6, a useful by-product of this technique is in smoothing formant frequencies. Finally, 3.7 focuses on the second formant frequency transition as a cue to place of the articulation of consonants and the way that so-called **locus equations** can be used to quantify the coarticulatory influence of a vowel on a preceding or following consonant.

The main functions in the Emu-R library that are presented in this Chapter are:

`eplot()`	plot an ellipse
`dplot()`	plot and synchronise tracks as a function of time
`locus()`	plot and calculate locus equations
`makelab()`	export labels
`plafit()`	calculate the coefficients of a parabola
`dct()`	calculate the coefficients of the discrete cosine transformation
`trapply()`	apply functions to trackdata objects

3.1 Vowel ellipses in the F2 x F1 plane

There is extensive evidence going back to the 19th and early part of the 20th Century (see e.g., Ladefoged, 1967 and Traunmüller & Lacerda, 1987) that vowel quality distinctions depend on the first two, or first three, resonances of the vocal tract. Since the first formant frequency is negatively correlated with phonetic vowel height, and since F2 is correlated with vowel backness, then a shape resembling the vowel quadrilateral emerges by plotting vowels in the (decreasing) F1 x F2 plane. Essner (1947) and Joos (1948) were amongst the first to

demonstrate this relationship, and since then, many different kinds of experimental studies have shown that this space is important for making judgements of vowel quality (see e.g., Harrington & Cassidy, 1999, p. 60-78).

We will begin with an examination of the data from the male speaker and to do this, we will identify that speaker's data using a logical vector and (for convenience of analysis) store that speaker's data as a set of objects that are parallel to each other. The objects to be analysed are from the `vowlax` dataset.

```
vowlax          # A segment list of four German lax vowels
vowlax.fdat     # A trackdata object of F1-F4
vowlax.l        # A vector of parallel vowel labels
vowlax.left     # A vector of labels of the segments preceding the vowels
vowlax.right    # A vector of labels of the segments following the vowels
vowlax.spkr     # A vector of speaker labels
vowlax.word     # A vector of word labels for the vowel
```

The dataset includes two speakers, one male (speaker "67") and one female (speaker "68") and the vowels (IPA: [ɪ, ɛ, ɔ, a]; MRPA: `I`, `E`, `O`, `a`) were extracted from the same 100 sentences read by each of these speakers. In the following, a logical vector is used to extract the data from the above for the male speaker:

```
temp = vowlax.spkr == "67"          # Logical vector - True for speaker "67"
m.fdat = vowlax.fdat[temp,]         # Formant data for speaker "67"
m.s = vowlax[temp,]                 # Segment list for speaker 67
m.l = vowlax.l[temp]                # Vowel labels for speaker "67"
m.left = vowlax.left[temp]          # Left context for speaker "67"
m.right = vowlax.left[temp]         # Right context for speaker "67"
m.word = vowlax.word[temp]          # Word labels for the vowel for speaker "67"
```

In plotting vowels in the F1 x F2 plane, a decision has to be made about the time point from which the data are to be extracted. Usually, the extraction point should be at or near the **vowel target**, which can be considered to be the point in the vowel at which the formants are least influenced by context and/or where the formants change minimally in time. Some issues to do with the vowel target are discussed in 3.3 below. For the present, the target is taken to be at the temporal midpoint of the vowel, on the assumption that this is usually the time point nearest to which the target occurs:

```
m.fdat.5 = dcut(m.fdat, .5, prop=T)
# Fig. 3.1
eplot(m.fdat.5[,1:2], m.l, centroid=T, form=T, xlab="F2 (Hz)", ylab="F1
(Hz)")
```

Note that in using `eplot()`, the number of rows of data must be the same as the number of elements in the parallel label vector. This can be checked as follows:

```
nrow(m.fdat.5[,1:2]) == length(m.l)
[1] TRUE
```

The `centroid=T` argument displays the means of the distributions using the corresponding character label; and as discussed in Chapter 2, the `form=T` argument rotates the space so that

the x-axis has decreasing F2 and the y-axis decreasing F1 as a result of which the vowels are positioned analogously to the phonetic backness and height axes of the vowel quadrilateral.

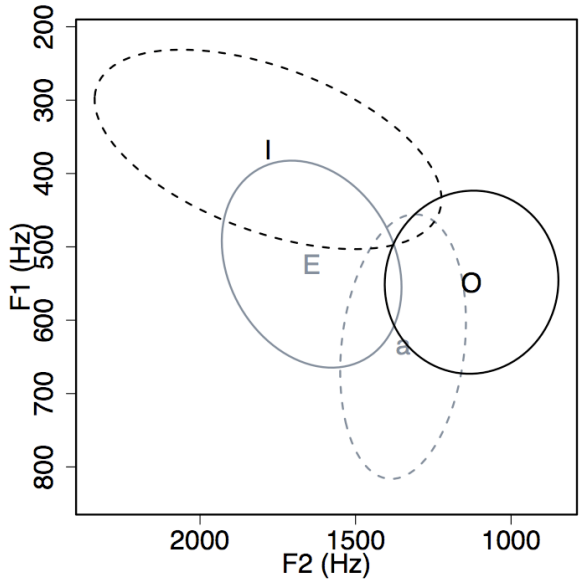

Fig. 3.1: 95% ellipse contours for F1 x F2 data extracted from the temporal midpoint of four German lax vowels produced by one speaker.

As discussed in more detail in connection with probabilistic classification in Chapter 6, an ellipse is a contour of equal probability. In the default implementation of the `eplot()` function, each ellipse includes about 95% of the data points corresponding to 2.447747 ellipse standard deviations (see in particular section 6.5).

Those researchers who tend not to look at very much at data from continuous speech often find the extent of overlap quite alarming because in laboratory speech of isolated word data, the ellipses of one speaker are usually quite well separated. The overlap arises in part because vowel targets in continuous speech are affected by different contexts and prosodic factors. Let's look at the distribution of the [ɪ] vowel in further detail according to the left context:

```
temp = m.l=="I"; par(mfrow=c(1,2))
eplot(m.fdat.5[temp,1:2], m.l[temp],m.left[temp], dopoints=T, form=T,
xlab="F2 (Hz)", ylab="F1 (Hz)")
```

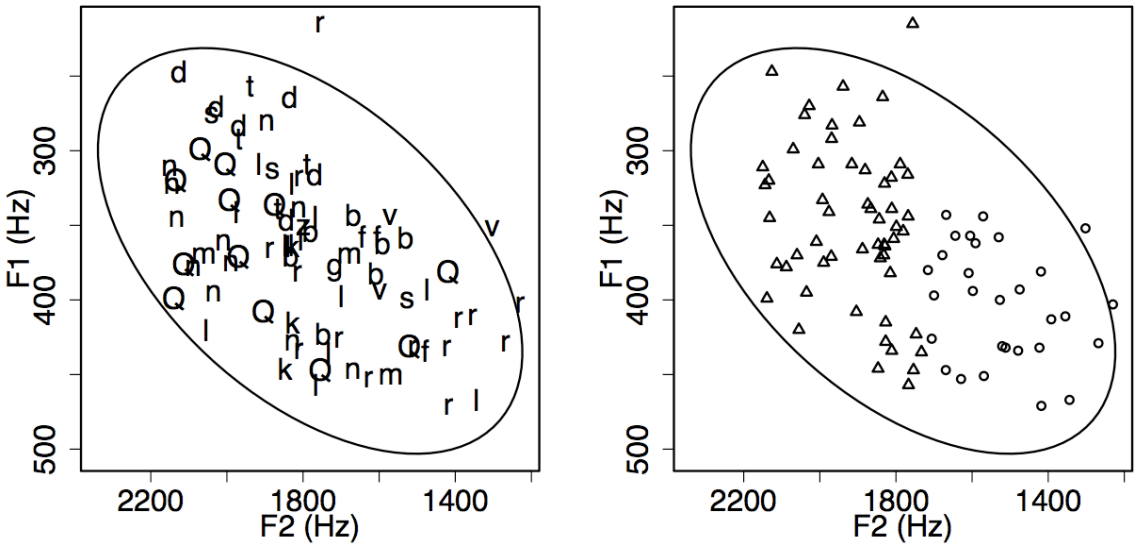

Fig. 3.2. *Left*. The [ɪ] ellipse from Fig. 3.1 with the left-context labels superimposed on the data points. *Right*: the same data partitioned into two clusters using kmeans-clustering.

There is not an immediately obvious pattern to the data in Fig. 3.2 and nor can one be reasonably expected, given that it does not take account of some other variables, especially of the right context. Nevertheless, when the preceding context is alveolar it does seem that [ɪ] is mostly positioned in the top left of the display with low F1 and high F2. There are also a number of "Q" labels with a high F2: these denote vowels and that are preceded by a glottal stop i.e., syllable or word-initial [ʔɪ] (vowels in a domain-initial position in German are usually glottalised).

The technique of **kmeans clustering** can be applied to give an indication of whether the variability is affected by different types of context. This technique partitions the data into *k* different clusters in such a way that the distance from the data points to the derived clusters of which they are members is minimised – in other words, the data tends to be partitioned into *k* maximally different clusters. Here, the `kmeans()` algorithm is used to divide the [ɪ] data into two clusters:

```
temp = m.l=="I"
k = kmeans(m.fdat.5[temp,1:2], 2)
# Fig. 3.2 (right)
eplot(m.fdat.5[temp,1:2], m.l[temp], k$cluster, dopoints=T, form=T,
xlab="F2 (Hz)", ylab="F1 (Hz)")
```

As is apparent from the right panel of Fig. 3.2, the algorithm has split up the data according to whether F2 is less, or greater, than roughly 1800 Hz. We can see whether this also partitions the data along the lines of the left context as follows:

```
temp = m.l == "I"
# Left context preceding [ɪ]
m.left.I = m.left[temp]
# Left context preceding [ɪ] in cluster 1 (the circles in Fig. 3.2, right panel).
temp = k$cluster==1
table(m.left.I[temp])
b  d  g  k  l  m  n  Q  r  s  t  z
3  6  1  3  9  1 10 10  6  2  4  1
```

```
# Left context preceding [ɪ] in cluster 2
table(m.left.I[!temp])
b f l m n Q r s v
4 3 4 2 1 2 9 1 3
```

So, as the above results show, cluster 1 (the circles in Fig. 3.2) includes all of [d, t], 10/11 of [n] and 10/12 [ʔ] ("Q"). Cluster 2 tends to include more contexts like [ʁ] ("r") and labials (there are more [b, f, m, v] in cluster 2 that for reasons to do with their low F2-locus, are likely to have a lowering effect on F2).

Thus left context obviously has an effect on F2 at the formant midpoint of [ɪ]. Just how much of an effect can be see by plotting the entire F2 trajectory between the vowel onset and vowel midpoint for two left contexts that fall predominantly in cluster 1 and cluster 2 respectively. Here is such a plot of [ɪ] following [ʔ] and the labiodentals [f, v] together:

```
# Logical vector that is true when the left context is "Q", "f", and "v"
temp = m.l == "I" & m.left %in% c("Q", "f", "v")
# The next two lines relabel "f" and "v" to a single category "LAB"
lab = m.left[temp]
lab[lab %in% c("f", "v")] = "LAB"
dplot(m.fdat[temp,2], lab, ylab="F2 (Hz)", xlab="Duration (ms)")
```

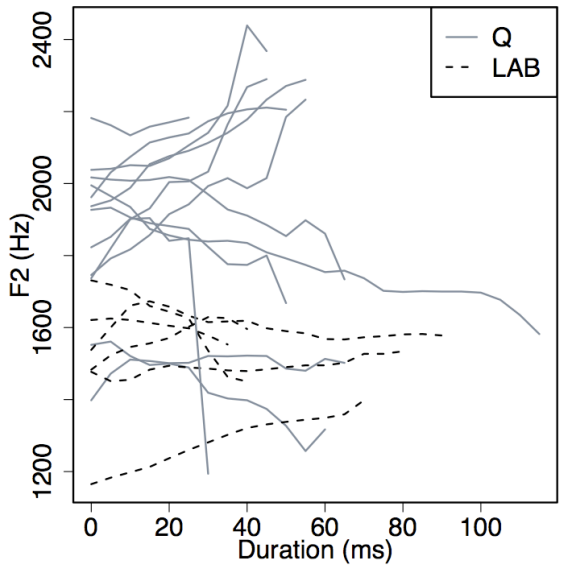

Fig. 3.3: F2 trajectories for [ʔɪ] (gray) and for [fɪ] and [vɪ] together (black, dashed) synchronised at the temporal midpoint.

Apart from two [ʔɪ] trajectories, there is a separation in F2 throughout the vowel depending on whether the left context is [ʔ] or a labiodental fricative. But before these very clear F2 differences are blamed just on left context, it would be a good idea to check the word label (and hence the right context). For example, for [ʔ]:

```
# Word labels for [ʔɪ]
table(vowlax.word[m.left=="Q" & m.l=="I"])
 ich  Ich   In Inge  Iss isst  ist
```

```
  4     4     2     4     2     2     6
```

So the words that begin with [ʔɪ] almost all have a right context which is likely to contribute to the high F2 i.e., [ɪ] in [ɪç] (*I*), [ɪs] (*eat*), and [ɪst] (*eats*, *is*): that is, the high F2 in [ʔɪ] is unlikely to be due just to [ɪ] alone.

3.2 Outliers

When a formant tracker is run over speech data in the manner described in Chapter 1.4.1, there will inevitably be errors due to formant tracking. Errors are especially common at the boundaries between voiceless and voiced segments and whenever two formants are close together in frequency, such as F1 and F2 for back vowels. If such errors occur, then it likely that they will show up as outliers in ellipse plots of the kind examined so far. If the outliers are far from the ellipse's centre, then they can have quite a dramatic effect on the ellipse orientation.

Fig. 3.5 shows and one, and possibly two, outliers from the [ɪ] vowels of the male speaker for the data extracted at the onset of the vowel:

```
# Speaker 67's ɪ vowels
temp = vowlax.spkr=="67" & vowlax.l=="I"
# Segment list thereof
m.seg = vowlax[temp,]
# F1 and F2 at the segment onset
m.Ion = dcut(vowlax.fdat[temp,1:2], 0, prop=T)
# Set x- and y-ranges to compare two plots to the same scale
xlim = c(150,500);  ylim =c(0,2500); par(mfrow=c(1,2))
# Ellipse plot with outliers
eplot(m.Ion, label(m.seg), dopoints=T, xlim=xlim, ylim=ylim, xlab="F2
(Hz)", ylab="F1 (Hz)")
```

As the left panel of Fig. 3.5 shows, there are two outliers: one of these is where F2 = 0 Hz at the bottom of the plot and is almost certainly due to a formant tracking error. The other outlier has a very low F1 but it is not possible to conclude without looking at the spectrogram whether this is a formant error or the result of a context effect.

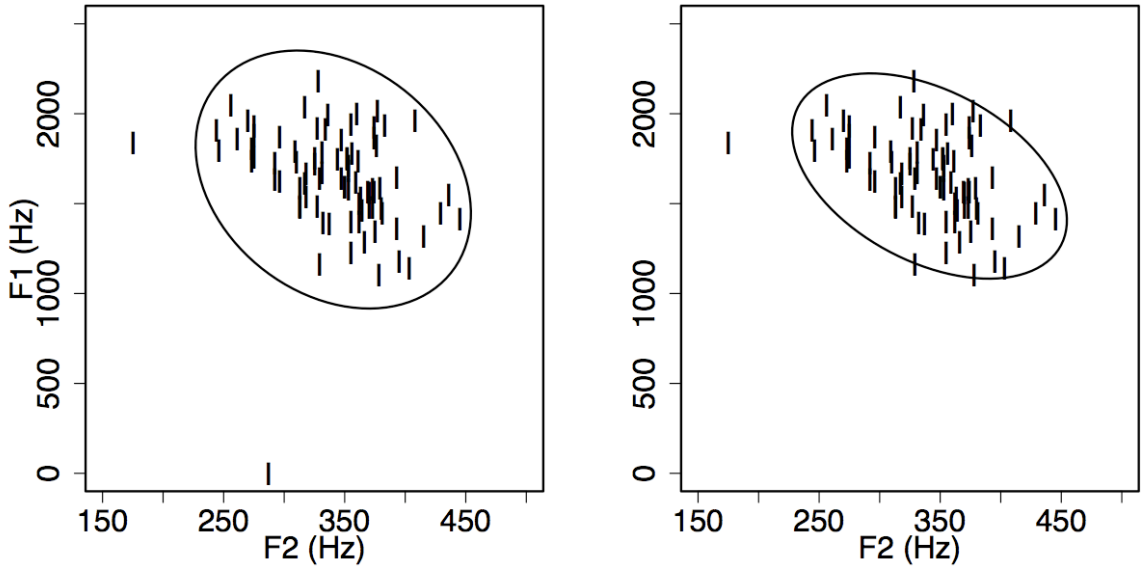

Fig. 3.4: 95% confidence intervals ellipses for [ɪ] vowels of speaker "67" at segment onset in the F1 x F2 plane before (left) and after (right) removing the outlier at F2 = 0 Hz.

The first of these outliers can, and should, be removed by identifying all values that have F2 less than a certain value, say 50 Hz. This can be done with a logical vector that is also passed to `eplot()` to produce the plot without the outlier on the right.

```
# Identify F2 values less than 50 Hz
temp = m.Ion[,2] < 50
eplot(m.Ion[!temp,], label(m.seg[!temp,]),dopoints=T, xlim=xlim, ylim=ylim,
xlab="F2 (Hz)")
```

Because the outlier was a long way from the centre of the distribution, its removal has shrunk the ellipse size and also changed its orientation slightly.

It is a nuisance to have to constantly remove outliers with logical vectors, so a better solution that the one in Fig. 3.5 is to locate its utterance identifier and redraw the formant track by hand in the database from which these data were extracted (following the procedure in 1.4.1). The utterance in which the outlier occurs as well as its time stamp can be found by combining the logical vector with the segment list:

```
temp = m.Ion[,2] < 50
m.seg[temp,]
segment  list from database:  kielread
query was:  Kanonic=a | E | I | O
    labels   start     end     utts
194      I 911.563 964.625 K67MR096
```

That is, the outlier occurs somewhere between 911 ms and 964 ms in the utterance `K67MR096` of the *kielread* database. The corresponding spectrogram shows that the outlier is very clearly a formant tracking error that can be manually corrected as in Fig. 3.6.

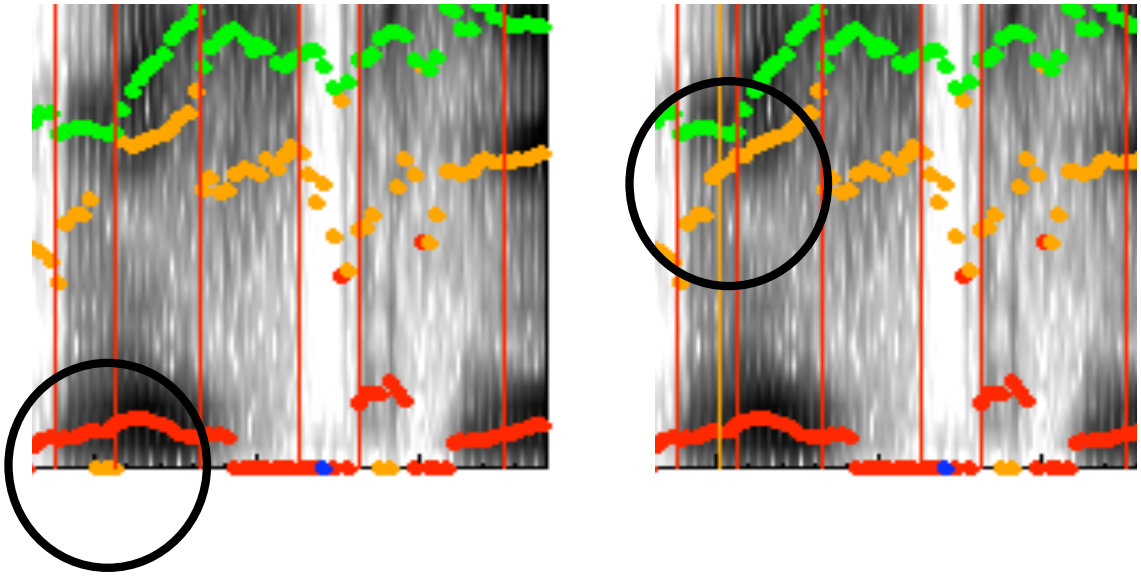

Fig. 3.5: Spectrogram (0 – 3000 Hz) in of part of the utterance K67MR096 showing an [ɪ] vowel with superimposed F1-F3 tracks; the miscalculated F2 values are redrawn on the right. The redrawn values can be saved – which has the effect of overwriting the signal file data containing the formant track for that utterance permanently.

Manually correcting outliers when they are obviously due to formant tracking errors as in Fig. 3.6 is necessary. But this method should definitely be used sparingly: the more manual intervention that is used, the greater the risk that the researcher might unwittingly bias the experimental data.

3.3 Vowel targets

As discussed earlier, the vowel target can be considered to be the section of the vowel that is least influenced by consonantal context and most similar to a citation-form production of the same vowel. It is also sometimes defined as the most **steady-state** part of the vowel: that is, the section of the vowel during which the formants (and hence the phonetic quality) change minimally (e.g., Broad & Wakita, 1977; Schouton & Pols, 1979). It is however not always the case that the vowel target need occur at the temporal midpoint. For example, in most accents of Australian English, in particular the broad variety, the targets of the long high vowels in *heed* and *who'd* occur late in the vowel and are preceded by a long onglide (e.g., Cox & Palethorpe, 2007). Apart from factors such as these, the vowel target could shift proportionally because of coarticulation. For example, Harrington, Fletcher and Roberts (1995) present articulatory data showing that in prosodically unaccented vowels (that is those produced without sentence stress), the final consonant is timed to occur earlier in the vowel than in prosodically accented vowels (with sentence stress). If the difference between the production of accented and unaccented vowels has an influence on the final transition in the vowel, then the effect would be that the vowel target occurs proportionally somewhat later in unaccented than in accented vowels. For reasons such as these, the vowel target cannot always be assumed to be at the temporal midpoint. At the same time, some studies have found that different strategies for locating the vowel target have not made a great deal of

difference to the analysis of vowel formant data (see e.g., van Son & Pols, 1990 who compared three different methods for vowel target identification in Dutch).

One method that is sometimes used for vowel target identification is to find **the time point at which the F1 is at a maximum**. This is based on the idea that vowels reach their targets when the oral tract is maximally open which often coincides with an F1-maximum, at least in non-high vowels (Lindblom & Sundberg, 1971). The time of the F1-maximum can be obtained in two steps: by writing a function to do this for a single segment of trackdata; and then using the `trapply()` function which applies a function to all of the segments of trackdata.

Consider the formant data from the first segment of the male speaker's data for which ellipses were drawn in Fig. 3.1. The formants are stored in `m.fdat` and the data in the `$data` component of this list (recall from the preceding Chapter that trackdata objects are lists):

```
# Logical vector - True for speaker "67"
temp = vowlax.spkr == "67"
# Formant data for speaker "67"
m.fdat = vowlax.fdat[temp,]
# Corresponding segment list and labels
m.s = vowlax[temp,]; m.l = vowlax.l[temp]

# Formant data from the first segment of this trackdata object
m.fdat[1,1]$data
      T1
857.5 372
862.5 412
867.5 467
872.5 506
877.5 532
882.5 547
887.5 556
892.5 561
897.5 562
902.5 556
907.5 550
912.5 538
917.5 525
922.5 521
927.5 517
932.5 504
937.5 494
942.5 492
```

This is the row of data at which F1 is at a maximum value. We want to get the time at which this occurs which is given by 897.5 in the row dimension name.

Whatever function we write needs to come up with the answer 897.5 because this is the time (in milliseconds) at which the F1-maximum occurs. Here is a function to get this data:

```
targtime <- function(dat, fun=max)
{
# Find the time at which dat first reaches
# a maximum if fun=max or a minimum if fun=min
# dat must be a vector or a one-columned matrix
temp = dat == fun(dat)
times = tracktimes(dat)
times[temp][1]
}
```

The function has been written with a default `fun=max`. This means that the default is to apply the unction `max()` for finding the maximum; so this leaves open the possibility of using the same function to find the time of minimum (e.g., for locating target times in back vowels based on the time of the F2-minimum). The point of `[1]` in the last line is to return the time at which the maximum *first* occurs (there might be several maxima, if F1 reaches a plateau). When the function is used on the data matrix of this first segment, the expected answer is produced:

```
targtime(m.fdat[1,1]$data)
[1] 897.5
```

Before applying this function to all segments using `trapply()`, it might be an idea to constrain the times within which the F1-maximum time is to be found, perhaps by excluding the first and last 25% of each vowel from consideration, given that these intervals are substantially influenced by the left and right contexts. This can be done, as discussed in 2.3, using `dcut()`:

```
# Create a new trackdata object  between the vowels' 25% and 75% time points
m.fdat.int = dcut(m.fdat, .25, .75, prop=T)
# Get the times at which the F1 maximum first occurs in this interval
m.maxf1.t = trapply(m.fdat.int[,1], targtime, simplify=T)
```

The calculated target times could be checked by plotting the trackdata synchronised at these calculated target times (Fig. 3.6, left panel):

```
# Logical vector to identify all a vowels
temp = m.l == "a"
# F1 and F2 of a synchronised at the F1-maximum time
dplot(m.fdat[temp,1:2], offset=m.maxf1.t[temp] , prop=F, ylab="F1 and F2
(Hz)", xlab="Duration (ms)")
```

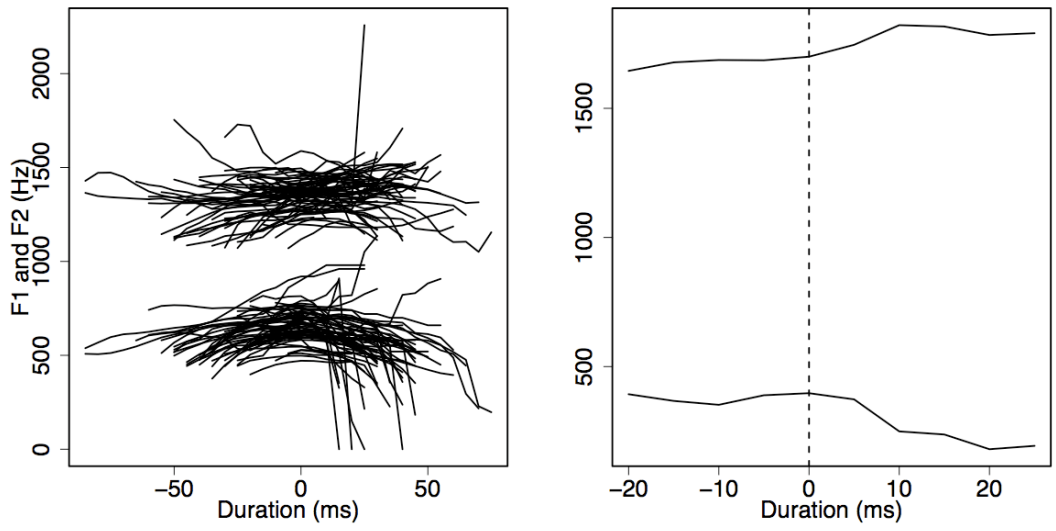

Fig. 3.6. *Left*: F1 and F2 of [a] for a male speaker of standard German synchronised at time $t = 0$, the time of the F1 maximum. *Right*: F1 and F2 from segment onset to offset for an [ɪ]. The vertical line mark the time within the middle 50% of the vowel at which F1 first reaches a maximum.

The results of this alignment can also be inspected segment by segment using a for-loop. The results of the last iteration is shown in the right panel of Fig. 3.6 and was created as follows (use the left mouse button to advance through the segments):

```
# For the first five segments separately…
for(j in 1:5){
# plot F1 and F2 as a function of time with the vowel label as the main title
dplot(m.fdat[j,1:2], main=m.l[j], offset= m.maxf1.t[j], prop=F, ylab="F1
and F2 (Hz)")
# Draw a vertical line at the F1-maximum
abline(v = 0, col=2)
# Left button to advance
locator(1)
}
```

It will almost certainly be necessary to change some of these times manually but this should be done not in R (which is not very good for this kind of thing) but in Praat or Emu. To do this, the vector of times needs to be exported so that the times are stored separately in different label files. The `makelab()` function can be used for this. The following writes out label files so that they can be loaded into Emu. You have to supply the name of the directory where you want all these label files to be stored as the third argument to `makelab()`.

```
path = "enter your preferred directory path here in quotes"
makelab(m.maxf1.t, utt(m.s), path, labels="T")
```

This will create a number of files, one per utterance, in the specified directory. For example, a file `K67MR001.xlab` will be created that looks like this:

```
signal K67MR001
nfields 1
#
0.8975 125 T
1.1225 125 T
1.4775 125 T
1.6825 125 T
2.1175 125 T
```

The template file for the database *kielread* must now be edited in the manner described in 1.2 and 1.3 so that the database can find these new label files. In Fig. 3.7, this is done by specifying a new level called Target that is associated with the Phonetic tier:

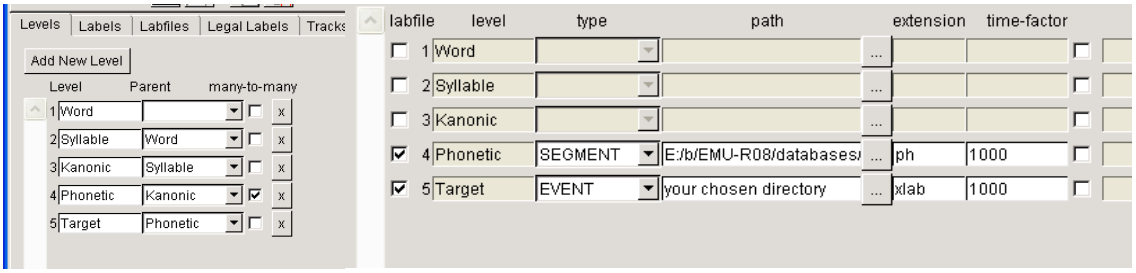

Fig. 3.7: The revised template file for the *kielread* database to include a level Target.

As a result of modifying the template, the target times are visible and can be manipulated (in Emulabel or Praat) as shown in Fig. 3.8.

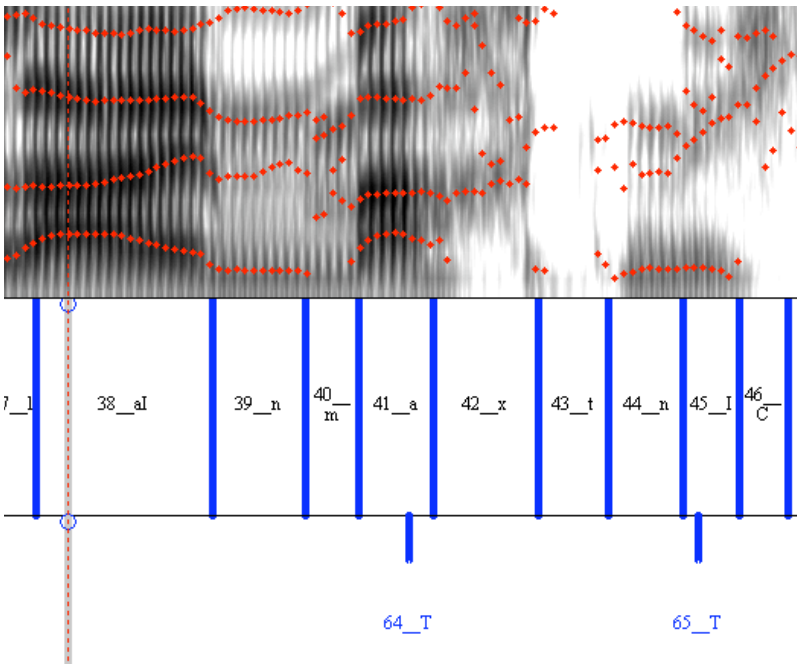

Fig. 3.8: The target times based on the F1-maximum loaded into Praat for utterance `K67MR001`.

The function has been used, as have seen, to find automatically the target based on the F1-maximum time. But recall that `targtime()` was written with an optional default to find a minimum instead of a maximum:

```
# Find targets based on the F2 minimum
m.minf2.t = trapply(m.fdat.int[,2], targtime, min, simplify=T)
```

What if want to find the vowel target in a vowel-specific way – based on the F1 maximum for the open and half-open vowels [a, ɛ], on the F2-maximum for the mid-high vowel [ɪ], and on F2-minimum for the back rounded vowel [ɔ]? In order to collect up the results so that the vector of target times is parallel to all the other objects, a logical vector could be created per vowel category and filled up a vector, vowel by vowel. This is done below: the resulting target times according to these criteria are stored in the vector `times`:

```
# A vector of zeros the same length as the label vector (and trackdata object)
times = rep(0, length(m.l))
# Target times based on F2-max for I
temp = m.l=="I"
times[temp] = trapply(m.fdat.int[temp,2], targtime, simplify=T)
# Target time based on F1-max for E and a
temp = m.l %in% c("E", "a")
times[temp] = trapply(m.fdat.int[temp,1], targtime, simplify=T)
# Target time based on F2-min for O
temp = m.l == "O"
times[temp] = trapply(m.fdat.int[temp,2], targtime, min, simplify=T)
```

3.4 Vowel normalisation

Much acoustic variation comes about because speakers have different sized and shaped vocal organs. This type of variation was demonstrated spectrographically in Peterson & Barney's (1952) classic study of vowels produced by men, women, and children and the extensive overlap between the formants of children and men was further investigated in Peterson (1961) and also in Ladefoged (1967) and Pols, Tromp & Plomp (1973). Some recent contributions to the field are given in Johnson & Mullenix (1997) and Adank et al. (2004).

We will begin with a comparison of ellipse plots of the same lax monophthongs that we have been examining so far, but this time for both the male and female speaker. These are shown together with the data points in Fig. 3.11 and they were created with the following commands:

```
# Set the ranges for the x- and y-axes to plot two panels in one row and two columns
xlim = c(800,2800);  ylim = c(250, 1050); par(mfrow=c(1,2))
# Logical vector for identifying the male speaker; !temp is the female speaker
temp = vowlax.spkr=="67"
eplot(vowlax.fdat.5[temp,1:2], vowlax.l[temp], dopoints=T, form=T,
xlim=xlim, ylim=ylim, xlab="F2 (Hz)", ylab="F1 (Hz)")
eplot(vowlax.fdat.5[!temp,1:2], vowlax.l[!temp], dopoints=T, form=T,
xlim=xlim, ylim=ylim, xlab="F2 (Hz)", ylab="")
```

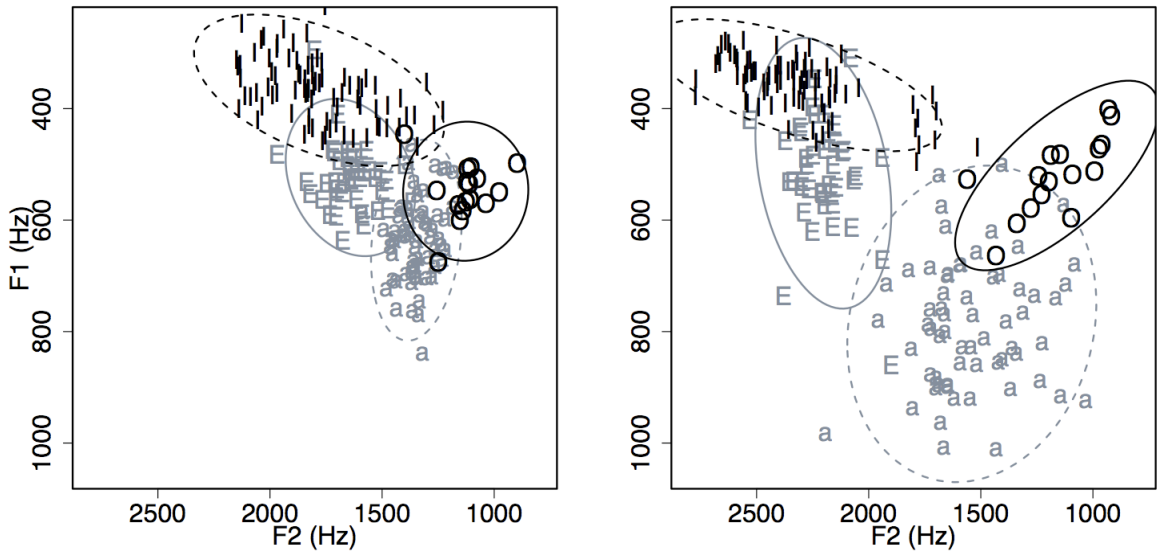

Fig. 3.9: German lax monophthongs produced by a male (left) and a female (right) speaker in the F2 x F1 plane. Data extracted at the temporal midpoint of the vowel.

The differences are quite substantial, especially considering that these are speakers of the same accent producing the same read sentences! The figure shows, in general, how the formants of the female speaker are higher than those of the male which is to be expected because female vocal tracts are on average shorter than those of males. But as argued in Fant (1966), because the ratio of the mouth cavity to the pharyngeal cavity lengths are different in male and female speakers, the changes to the vowels due to gender is *non-uniform* which means that the male-female differences are greater in some vowels than others. Fig. 3.9 shows that, whereas the differences between the speakers in [ɪ] are not that substantial, those between [a] on F1 and F2 are quite considerable. Also the F2 differences are much

more marked than those in F1 as a comparison between the two speakers in the relative positions of [ɪ, ɛ] shows. Finally, the differences need not be the result entirely of anatomical and physiological differences between the speakers. Some differences may be as a result of speaking-style: indeed, for the female speaker there is a greater separation between [ɪ, ɛ] on the one hand and [ɔ, a] on the other than for the male speaker and this may be because there is greater vowel hyperarticulation for this speaker – this issue will be taken up in more detail in the analysis of Euclidean distances in 3.5.

An overview of the main male-female vowel differences can be obtained by plotting a polygon that connects the means (centroids) of each vowel category for the separate speakers on the same axes. The first step is to get the speaker means. The function `tapply()` applies a function to a *vector* per category (see Appendix A). So the F1 category means for speaker `67` are:

```
temp = vowlax.spkr=="67"
tapply(vowlax.fdat.5[temp,1], vowlax.l[temp], mean)
       a        E        I        O
635.9524 523.5610 367.1647 548.1875
```

But in order to get these category means for a *matrix* of F1 and F2, `tapply()` could be used inside the `apply()` function[1]. The basic syntax for applying a function, to the columns of a matrix is `apply(matrix, 2, fun, arg1, arg2…argn)` where `arg1`, `arg2`, …`argn` are the arguments of the function `fun()` that is to be applied to the matrix. So the F1 and F2 category means for speaker `67` are:

```
temp = vowlax.spkr=="67"
apply(vowlax.fdat.5[temp,1:2], 2, tapply, vowlax.l[temp], mean)
         T1       T2
a 635.9524 1347.254
E 523.5610 1641.073
I 367.1647 1781.329
O 548.1875 1127.000
```

The desired polygon could be plotted first by calling `eplot()` with `doellipse=F` (don't plot the ellipses) and then joining up these means using the `polygon()` function. The *x*- and *y*-ranges need to be set in the call to `eplot()`, in order to superimpose the corresponding polygon from the female speaker on the same axes:

```
# For the male speaker
xlim = c(1000, 2500); ylim = c(300, 900)
eplot(vowlax.fdat.5[temp,1:2], vowlax.l[temp],  form=T, xlim=xlim,
ylim=ylim, doellipse=F, col=c(1,1,1,1), xlab="F2 (Hz)", ylab="F1 (Hz)" )
m = apply(vowlax.fdat.5[temp,1:2], 2, tapply, vowlax.l[temp], mean)
# Negate the mean values because this is a plot in the –F2 x –F1 plane
polygon(-m[,2], -m[,1])
```

Then, since the logical vector, `temp`, is True for the male speaker and False for the female speaker, the above instructions can be repeated with `!temp` for the corresponding plot for the female speaker. The line `par(new=T)` is for superimposing the second plot on the same axes and `lty=2` in the call to `polygon()` produces dashed lines:

```
par(new=T)
```

[1] The `trapply()` function can be used for the same effect:

```
trapply(vowlax.fdat.5[temp,1:2], vowlax.l[temp], mean, simplify=T)
```

```
eplot(vowlax.fdat.5[!temp,1:2], vowlax.l[!temp],  form=T, xlim=xlim,
ylim=ylim, doellipse=F, col=c(1,1,1,1), xlab="", ylab="")
m = apply(vowlax.fdat.5[!temp,1:2], 2, tapply, vowlax.l[!temp], mean)
polygon(-m[,2], -m[,1], lty=2)
```

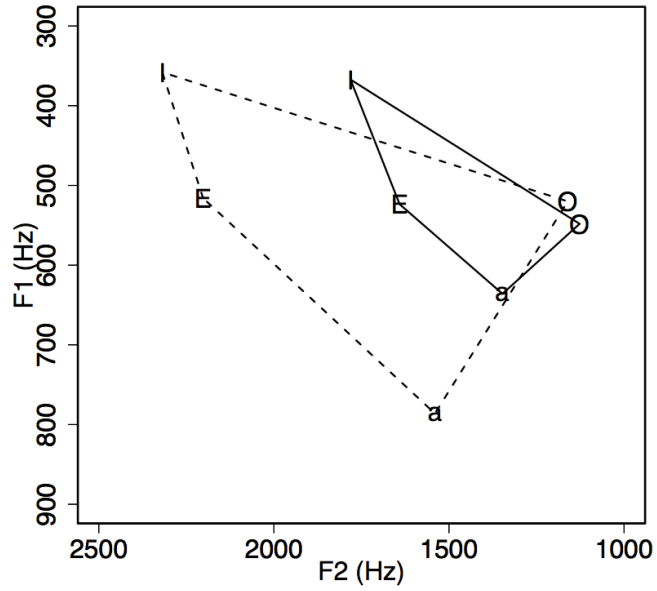

Fig. 3.10: Mean F2 x F1 values for the male (solid) and female (dashed) data from Fig. 3.9.

Strategies for vowel normalisation are designed to reduce the extent of these divergences due to the speaker and they fall into two categories: **speaker-dependent** and **speaker-independent**. In the first of these, normalisation can only be carried out using statistical data from the speaker **beyond the vowel that is to be normalised** (for this reason, speaker-dependent strategies are also called **extrinsic**, because information for normalisation is extrinsic to the vowel that is to be normalised). In a speaker-independent strategy by contrast, all the information needed for normalising a vowel is **within the vowel itself**, i.e., intrinsic to the vowel.

The idea that normalisation might be extrinsic can be traced back to Joos (1948) who suggested that listeners judge the phonetic quality of a vowel in relation to a speaker's point vowels [i, a, u]. Some evidence in favour of extrinsic normalisation is provided in Ladefoged & Broadbent (1957) who found that the listeners' perceptions of the vowel in the same test word shifted when the formant frequencies in a preceding carrier phrase were manipulated. On the other hand, there were also various perception experiments in the 1970s and 1980s showing that listeners' identifications of a speaker's vowels were not substantially improved if they were initially exposed to the same speaker's point vowels (e.g., Assmann et al, 1982; Verbrugge et al., 1976).

Whatever the arguments from studies of speech perception for or against extrinsic normalisation, there is evidence to show that when extrinsic normalisation is applied to acoustic vowel data, the differences due to speakers can often be quite substantially reduced (see e.g. Disner, 1980 for an evaluation of some extrinsic vowel normalisation procedures). A very basic and effective extrinsic normalisation technique is to **transform the data to z-**

scores by subtracting out the speaker's mean and dividing by the speaker's standard-deviation for each parameter separately. The technique was first used by Lobanov (1971) for vowels and so is sometimes called **Lobanov-normalisation**. The transformation has the effect of centering each speaker's vowel space at coordinates of zero (the mean); the axes are then the **number of standard deviations away from the speaker's mean**. So for a vector of values:

```
vec = c(-4, -9, -4,  7,  5, -7,  0,  3,  2, -3)
```

their Lobanov-normalised equivalents are:

```
(vec - mean(vec)) / sd(vec)
-0.5715006 -1.5240015 -0.5715006  1.5240015  1.1430011
-1.1430011  0.1905002  0.7620008  0.5715006 -0.3810004
```

A function that will carry out Lobanov-normalisation when applied to a vector is as follows:

```
lob <- function(x)
{
# transform x to z-scores (Lobanov normalisation); x is a vector
(x - mean(x))/sd(x)
}
```

Thus `lob(vec)` gives the same results as above. But since there is more than one parameter (F1, F2), the function needs to be applied to a matrix. As discussed earlier, `apply(matrix, 2, fun)` has the effect of applying a function, `fun()`, separately to the columns of a matrix. The function is modified below so that it can be applied to a matrix; notice that the simpler function above is called from inside this modified function:

```
lobnorm <- function(x)
{
# transform x to z-scores (Lobanov normalisation); x is a matrix
lob <- function(x)
{
(x - mean(x))/sd(x)
}
apply(x, 2, lob)
}
```

The Lobanov-normalised F1 and F2 for the male speaker are now:

```
temp = vowlax.spkr == "67"
norm67 = lobnorm(vowlax.fdat.5[temp,])
```

and those for the female speakers can be obtained with the inverse of the logical vector, i.e., `vowlax.fdat.5[!temp,]`. However, it is always a good idea to keep objects that belong together (segment list, trackdata, label files, matrices derived from trackdata, normalised data derived from such data, etc.) **parallel to each other** in order to retain the power of being able to manipulate the objects relative to each other. Here is one way to do this:

```
# Set up a matrix of zeros with the same dimensions as the matrix to be Lobanov-normalised
vow.norm.lob = matrix(0, nrow(vowlax.fdat.5), ncol(vowlax.fdat.5))
temp = vowlax.spkr == "67"
vow.norm.lob[temp,] = lobnorm(vowlax.fdat.5[temp,])
vow.norm.lob[!temp,] = lobnorm(vowlax.fdat.5[!temp,])
```

The same effect can be achieved with a for-loop and indeed, this is the preferred approach if there are several speakers whose data is to be normalised (but it works just the same when there are only two):

```
vow.norm.lob = matrix(0, nrow(vowlax.fdat.5), ncol(vowlax.fdat.5))
for(j in unique(vowlax.spkr)){
temp = vowlax.spkr==j
vow.norm.lob[temp,] = lobnorm(vowlax.fdat.5[temp,])
}
```

Since `vow.norm.lob` is parallel to the vector of labels `vowlax.l`, the `eplot()` function can be used in the same way as for the non-normalised data to plot ellipses.

```
# Fig. 3.11
xlim = ylim = c(-2.5, 2.5);  par(mfrow=c(1,2))
temp = vowlax.spkr=="67"
eplot(vow.norm.lob[temp,1:2], vowlax.l[temp], dopoints=T, form=T,
xlim=xlim, ylim=ylim, xlab="F2 (normalised)", ylab="F1 (normalised)")
eplot(vow.norm.lob[!temp,1:2], vowlax.l[!temp], dopoints=T, form=T,
xlim=xlim, ylim=ylim, xlab="F2 (normalised)", ylab="")
```

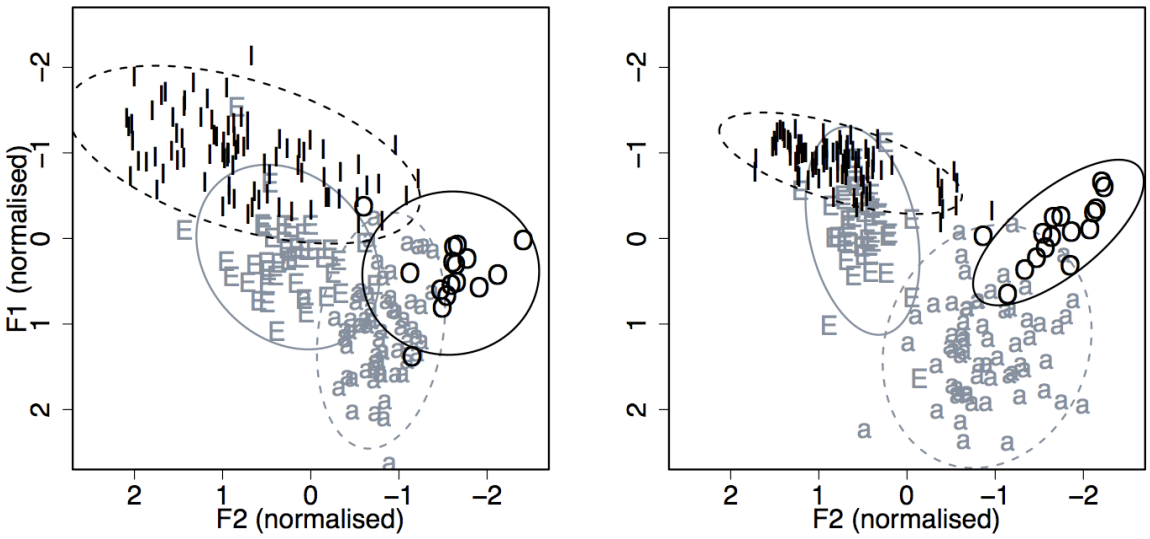

Fig. 3.11: Lobanov-normalised F1 and F2 of the data in Fig. 3.9. The axes are numbers of standard deviations from the mean.

The point [0,0] is the mean or **centroid** across all the data points per speaker and the axes are the numbers of standard deviations away from [0,0]. Compared with the raw data in Fig. 3.9, it is clear enough that there is a much closer alignment between the vowel categories of the male and female speakers in these normalised data. For larger studies, the mean and standard deviation should be based not just on those of a handful of lax vowels, but a much wider selection of vowel categories.

Another extrinsic normalisation technique is due to Nearey (e.g., Assmann et al., 1982; Nearey, 1989). The version demonstrated here is the one in which normalisation is accomplished by subtracting a **speaker-dependent constant** from the logarithm of the

formants. This speaker-dependent constant is obtained by working out (a) the mean of the logarithm of F1 (across all tokens for a given speaker) and (b) the mean of the logarithm of F2 (across the same tokens) and then averaging (a) and (b). An expression for the speaker-dependent constant in R is therefore `mean( apply(log(mat), 2, mean))`, where `mat` is a two-columned matrix of F1 and F2 values. So for the male speaker, the speaker-dependent constant is:

```
temp = vowlax.spkr == "67"
mean(apply(log(vowlax.fdat.5[temp,1:2]), 2, mean))
6.755133
```

This value must now be subtracted from the logarithm of the raw formant values separately for each speaker. This can be done with a single-line function:

```
nearey <- function(x)
{
# Function for extrinsic normalisation according to Nearey
# x is  a two-columned matrix
log(x) -  mean(apply(log(x), 2, mean))
}
```

Thus `nearey(vowlax.fdat.5[temp,1:2])` gives the Nearey-normalised formant (F1 and F2) data for the male speaker. The same methodology as for Lobanov-normalisation above can be used to obtain Nearey-normalised data that is parallel to all the other objects – thereby allowing ellipse and polygon plots to be drawn in the F2 x F1 plane in the manner described earlier.

Lobanov- and Nearey-normalisation are, then, two examples of speaker-dependent strategies that require data beyond the vowel that is to be normalised. In speaker-*independent* strategies all the information for normalisation is supposed to be in the vowel itself. Earlier speaker-independent strategies made use of formant ratios (e.g., Peterson, 1952, 1961; Potter & Steinberg, 1950; see also Millar, 1989) and they often copy some aspects of the auditory transformations to acoustic data that are known to take place in the ear (e.g., Bladon et al, 1984). These types of speaker-independent auditory transformations are based on the idea that two equivalent vowels, even if they are produced by different speakers, result in a similar pattern of motion along the basilar membrane, even if the actual position of the pattern varies (Potter & Steinberg, 1950; see also Chiba & Kajiyama, 1941). Since there is a direct correspondence between basilar membrane motion and a sound's frequency on a scale known as the **Bark scale**, a transformation to an auditory scale like the Bark scale (or ERB scale – see e.g., Glasberg & Moore, 1990) is usually the starting point for speaker-independent normalisation. Independently of these normalisation issues, many researchers transform formant values from Hertz to Bark before applying any further analysis, on the grounds that an analogous translation is presumed to be carried out in the ear.

There is a function `bark(x)` in the Emu-R library to carry out such a transformation, where `x` is a vector, matrix, or trackdata object of Hertz values. (The same function with the `inv=T` argument converts Hertz back into Bark). The formulae for these transformations are given in Traunmüller (1990) and they are based on analyses by Zwicker (1961).

A graph of the relationship between the two scales (up to 10 kHz) with horizontal lines at intervals of 1 Bark superimposed on the Hz axis can be drawn as follows:

```
# Fig. 3.12
```

```
plot(0:10000, bark(0:10000), type="l", xlab="Frequency (Hz)",
ylab="Frequency (Bark)")
abline(v=bark(1:22, inv=T), lty=2)
```

This plot shows that the interval corresponding to 1 Bark becomes progressively wider for values higher up the Hertz scale (the Bark scale, like the Mel scale (Fant, 1968), is roughly linear up to 1 kHz and then quasi-logarithmic thereafter).

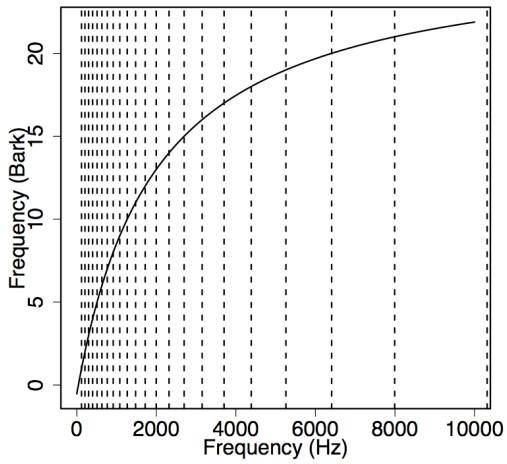

Fig. 3.12: Relationship between the Hertz and Bark scales. The vertical lines mark interval widths of 1 Bark.

Ellipse plots analogous to those in Fig. 3.11 can be created by converting the matrix into Bark values (thus the first argument to `eplot()` for the male speaker is `bark(vowlax.fdat.5[temp,1:2])`) or else by leaving the values in Hz and adding the argument `scaling="bark"` in the `eplot()` function). A detailed exploration of Bark-scaled, vowel formant data is given in Syrdal & Gopal (1986).

3.5 Euclidean distances

3.5.1 Vowel space expansion

A number of studies in the last fifty years have been concerned with phonetic vowel reduction that is with the changes in vowel quality brought about by segmental, prosodic, and situational contexts. In Lindblom's (1990) hyper- and hypoarticulation theory, speech production varies along a continuum from clear to less clear speech. According to this theory, speakers make as much effort to speak clearly as is required by the listener for understanding what is being said. Thus, the first time that a person's name is mentioned, the production is likely to be clear because this is largely unpredictable information for the listener; but subsequent productions of the same name in an ongoing dialogue are likely to be less clear, because the listener can more easily predict its occurrence from context (Fowler & Housum, 1987).

A vowel that is spoken les clearly tends to be reduced which means that there is a deviation from its position in an acoustic space relative to a clear or citation-form production. The deviation is often manifested as centralisation, that is the vowel is produced nearer the centre of the speaker's vowel space than in clear speech. Another way of putting the same idea is to say that in clear speech there is an **expansion of the vowel space**. This conveys the idea that vowels move away from the centre in a space such as the F2 x F1 plane that has been considered so far. There is also articulatory evidence for this type of vowel space expansion when vowels occur in prosodically accented words, probably because these tend to

be points of information focus (i.e., intervals of an utterance that are especially important for understanding what is being said; e.g., de Jong, 1995; Harrington, Fletcher & Beckman, 2000).

One of the ways of quantifying vowel space expansion is to measure the Euclidean or straight line distance between a vowel and the centre of the vowel space. Wright (2003) used just such a measure to compare so called *easy* and *hard* words on their distances to the centre of the vowel space. Easy words were those that have high lexical frequency (i.e., occur often) and low neighborhood density (there are few words that are phonemically similar). Since such words tend to be relatively easy for the listener to understand, then, applying Lindblom's (1990) model, the vowels should be relatively centralised compared with hard words which were both infrequent and high in neighborhood density.

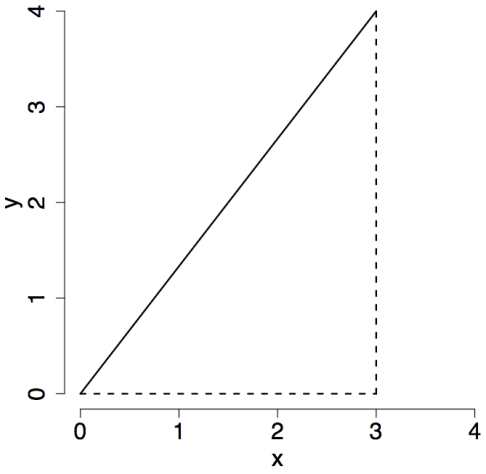

Fig. 3.13. A 3-4-5 triangle. The length of the solid line is the Euclidean distance between the points (0, 0) and (3, 4). The dotted lines show the horizontal and vertical distances that are used for the Euclidean distance calculation.

One of the ways of quantifying the extent of centralisation, which was also used by Wright (2003), is to calculate the Euclidean distance of each vowel token to the centre of the vowel space. In a two-dimensional space, the Euclidean distance is calculated by summing the square of the horizontal and vertical distances between the points and taking the square root. For example, the expressions in R for horizontal and vertical distances between the two points (0, 0) and (3,4) in Fig. 3.13 are `(0 - 3)^2` and `(0 - 4)^2` respectively. Thus the Euclidean distance between them is:

```
sqrt( (0 - 3)^2 + (0 - 4)^2 )
5
```

Because of the nice way that vectors work in R, the same result is given by:

```
a = c(0, 0)
b = c(3, 4)
sqrt( sum( (a - b)^2 ))
5
```

So a function to calculate the Euclidean distance between any two points `a` and `b` is:

```
euclid <- function(a, b)
{
# Function to calculate Euclidean distance between a and b;
# a and b are vectors of the same length
sqrt(sum((a - b)^2))
```

```
}
```

In fact, this function works not just in a two-dimensional space, but in an *n*-dimensional space. So if there are two vowels, `a` and `b`, in a three-dimensional F1, F2, F3 space with coordinates for vowel `a` F1 = 500 Hz, F2 = 1500 Hz, F3 = 2500 Hz and for vowel `b` F1 = 220 Hz, F2 = 2400 Hz, F3 = 3000 Hz, then the straight line, Euclidean distance between `a` and `b` is just over 1066 Hz as follows:

```
a = c(500, 1500, 2500)
b = c(220, 2400, 3000)
euclid(a, b)
1066.958
```

Exactly the same principle and hence the same function works in 4, 5, …*n* dimensional spaces even though any space higher than three dimensions cannot be seen or drawn. The only obligation on the function is that the vectors should be of the same length. The function can be made to break giving an error message, if the user should try to do otherwise:

```
euclid <- function(a, b)
{
# Function to calculate Euclidean distance between a and b;
# a and b are vectors of the same length
if(length(a) != length(b))
stop("a and b must be of the same length")
sqrt(sum((a - b)^2))
}

a = c(3, 4)
b = c(10, 1, 2)
euclid(a, b)
Error in euclid(a, b) : a and b must be of the same length
```

For the present task of assessing vowel space expansion, the distance of all the vowel tokens to the centre of the space will have to be measured. For illustrative purposes, a comparison will be made between the male and female speakers on the lax vowel data considered so far, although in practice, this technique is more likely to be used to compare vowels in easy and hard words or in accented and unaccented words as described earlier. The question to which we seek an answer is as follows: is there any evidence that the lax vowels of the female speaker are more expanded, i.e. more distant from the centre of the vowel space than those of the male speaker in Figs. 3.9 and 3.10? A glance at Fig. 3.10 in particular must surely suggest that the answer to this question is 'yes'. But let us now quantify what is obvious visually. First of all, a single point that is at the centre of the speaker's vowel space, known as the **centroid**, has to be defined. This could be taken across a much larger sample of the speaker's vowels than are available in these data sets: for the present, it will be taken to be the average across all of the speaker's lax vowels. For the male and female speaker these are:

```
temp = vowlax.spkr == "67"
m.av = apply(vowlax.fdat.5[temp,1:2], 2, mean)
m.av
       T1        T2
 495.1756 1568.8098

f.av = apply(vowlax.fdat.5[!temp,1:2], 2, mean)
```

```
f.av
        T1        T2
 533.8439 1965.8293
```

But there are good grounds for objecting to these means: in particular, the distribution of vowel tokens across the categories is not equal, as the following shows for the male speaker (the distribution is the same for the female speaker):

```
table(vowlax.l[temp])

 a  E  I  O
63 41 85 16
```

In view of the relatively few back vowels, the centroids are likely to be biased towards the front of the vowel space. As an alternative, the centroids could be defined as the mean of the vowel means, which is the point that is at the centre of the polygons in Fig. 3.12. Recall that for the female speaker the mean positions of the vowels was given by:

```
temp = vowlax.spkr == "67"
f = apply(vowlax.fdat.5[!temp,1:2], 2, tapply, vowlax.l[!temp], mean)
f
          T1       T2
a 786.1429 1540.159
E 515.9268 2202.268
I 358.0941 2318.812
O 520.0000 1160.813
```

So the mean of these means is:

```
f.av = apply(f, 2, mean)
f.av
      T1       T2
 545.041 1805.51
```

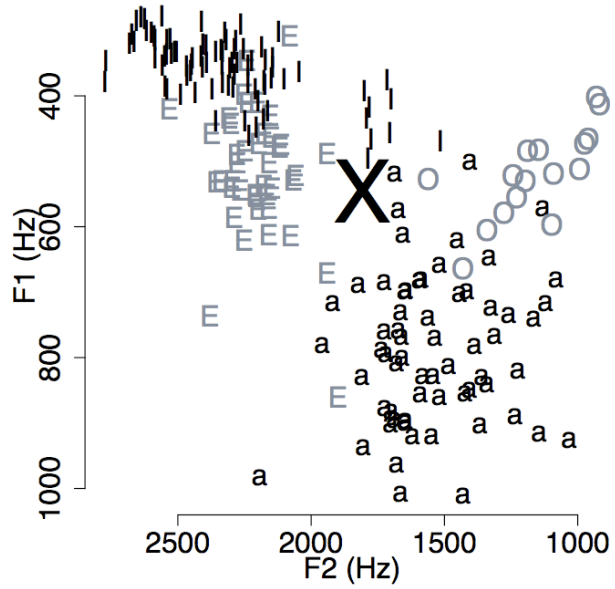

Fig. 3.16: Lax monophthongs in German for female speaker `68` in the F2 x F1 plane for data extracted at the vowels' temporal midpoint. X is the centroid defined as the mean position of the same speaker's mean across all tokens of the four vowel categories.

The centroid is shown in Fig. 3.14 which was produced as follows:

```
temp = vowlax.spkr=="68"
eplot(vowlax.fdat.5[temp,1:2], vowlax.l[temp], dopoints=T, form=T,
xlab="F2 (Hz)", ylab="F1 (Hz)", doellipse=F)
text(-f.av[2], -f.av[1], "X", cex=3)
```

The Euclidean distances of each data point in Fig. 3.14 to X can be obtained by applying `euclid()` to the rows of the matrix using `apply()` with a second argument of `1` (meaning apply to rows – see Appendix A).

```
temp = vowlax.spkr=="68"
e.f = apply(vowlax.fdat.5[temp,1:2], 1, euclid, f.av)
```

The same technique as in 3.4 could be used to keep all the various objects that have something to do with lax vowels parallel to each other, as follows:

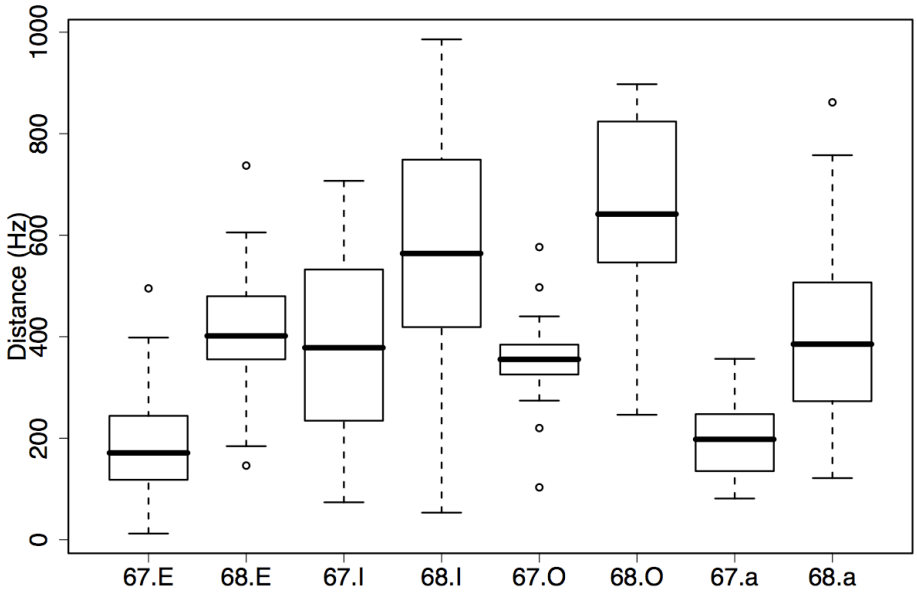

Fig. 3.15: Boxplots of Euclidean distances to the centroid (Hz) for speaker `67` (male) and speaker `68` (female) for four lax vowel categories in German.

```
# Vector of zeros to store the results
edistances = rep(0, nrow(vowlax.fdat.5))
# Logical vector to identify speaker 67
temp = vowlax.spkr == "67"
# The next two lines give the male speaker's centroid analogous to f.av
m = apply(vowlax.fdat.5[temp,1:2], 2, tapply, vowlax.l[temp], mean)
m.av = apply(m, 2, mean)
# Distances to the centroid for the male speaker
edistances[temp] = apply(vowlax.fdat.5[temp,1:2], 1, euclid, m.av)
# Distances to the centroid for the female speaker
edistances[!temp] = apply(vowlax.fdat.5[!temp,1:2], 1, euclid, f.av)
```

Since all the objects are parallel to each other, it only takes one line to produce a boxplot of the results comparing the Euclidean distances for the male and female speakers separately by vowel category:

```
boxplot(edistances ~ factor(vowlax.spkr) * factor(vowlax.l), ylab=
"Distance (Hz)")
```

The plot confirms what was suspected: the Euclidean distances are greater on every vowel category for the female speaker.

3.5.2 Relative distance between vowel categories

In dialectology and studies of sound change, there is often a need to compare the relative position of two vowel categories in a formant space. The sound change can sometimes be linked to age and social class, as the various pioneering studies by Labov (1994, 2001) have shown. It might be hypothesised that a vowel is in the process of fronting or raising: for example, the vowel in *who'd* in the standard accent of English has fronted in the last fifty years (Harrington et al, 2008; Hawkins & Midgley, 2005), there has been a substantial rearrangement of the front lax vowels in New Zealand English (Bauer, 1986; Maclagen & Hay, 2004; Gordon et al., 2004), and there is extensive evidence of numerous changes to North American vowels that is documented in Labov (1994, 2001).

Vowels are often compared across two different age groups so that if there is a vowel change in progress, the position of the vowel in the older and younger groups might be different (this type of study is known as an **apparent time study** – see e.g,, Bailey et al, 1991). Of course independently of sound change, studies comparing different dialects might seek to provide quantitative evidence for the relative differences in vowel positions: whether, for example, the vowel in Australian English *head* is higher and/or fronter than that of Standard Southern British English.

There are a number of ways of providing quantitative data of this kind. The one to be illustrated here is concerned with determining whether the position of a vowel in relation to other vowels is different in one set of data compared with another. I used just this technique (Harrington, 2006) to assess whether the long, final lax vowel in words like *cit**y***, *plent**y***, *read**y***, was relatively closer to heed than to hid in the more recent Christmas messages broadcast by Queen Elizabeth II over a fifty year period.

For illustrative purposes, we will again make use of the lax vowel data. Fig. 3.11 suggest that [ɛ] is closer to [ɪ] than it is to [a] in the female than in the male speaker. Perhaps this is a sound change in progress, perhaps the female subject does not speak exactly the same variety as the male speaker, or perhaps it has something to do with differences between the speakers along the hyper- and hypoarticulation continuum. Whatever the reasons, it is just this sort of problem that can arise in sociophonetics in dealing with gradual and incremental sound change.

The way of addressing this issue in Harrington (2006) was to work out two Euclidean distances: d_1, the distance of all of the [ɛ] tokens to the centroid of [ɪ]; and d_2, the distance of all of the same [ɛ] tokens to the centroid of [a]. The ratio of these two distances, d_1/d_2 is indicative of how close (in terms of Euclidean distances) the [ɛ] tokens are to [ɪ] in relation to [a]. The logarithm of this, which will be termed E_{RATIO}, gives the same information but in a more convenient form. More specifically, since

$$E_{RATIO} = \log(d_1/ d_2)$$
$$= \log(d_1) – \log(d_2)$$

The following three relationships hold for any single token of [ɛ]:

(a) if an [ɛ] token is exactly equidistant between the [ɪ] and [a] centroids, $\log(d_1) = \log(d_2)$, i.e. E_{RATIO} is zero.

(b) if an [ɛ] token is closer to the centroid of [ɪ], then $\log(d_1) < \log(d_2)$ and so E_{RATIO} is negative.

(c) if an [ɛ] token is closer to [a] than to [ɪ], $\log(d_1) > \log(d_2)$ and so E_{RATIO} is positive.

The hypothesis to be tested is that the female speaker's [ɛ] vowels are closer to her [ɪ] than to her [a] compared with those for the male speaker. If so, then the female speaker's E_{RATIO} should be smaller than for the male speaker. The Euclidean distance calculations will be carried out as before in the F2 x F1 vowel space using the `euclid()` function written in 3.5.1. Here are the commands for the female speaker:

```
# Next two lines calculate the centroid of female [ɪ]
temp = vowlax.spkr == "68" & vowlax.l=="I"
mean.I = apply(vowlax.fdat.5[temp,1:2], 2, mean)

# Next two lines calculate the centroid of female [a]
temp = vowlax.spkr == "68" & vowlax.l=="a"
mean.a = apply(vowlax.fdat.5[temp,1:2], 2, mean)

# Logical vector to identify all the female speaker's [ɛ] vowels
temp = vowlax.spkr == "68" & vowlax.l=="E"

# This is d1 above i.e., the distance of [ɛ] tokens to [ɪ] centroid
etoI = apply(vowlax.fdat.5[temp,1:2], 1, euclid, mean.I)

# This is d2 above i.e., the distance of [ɛ] tokens to [a] centroid
etoa = apply(vowlax.fdat.5[temp,1:2], 1, euclid, mean.a)

# E_RATIO for the female speaker
ratio.log.f = log(etoI/etoa)
```

Exactly the same instructions can be carried out for the male speaker except that `68` should be replaced with `67` throughout in the above instructions. For the final line for the male speaker, `ratio.log.m` is used to store the male speaker's E_{RATIO} values. A histogram of the E_{RATIO} distributions for these two speakers can then be created as follows (Fig. 3.16):

```
par(mfrow=c(1,2)); xlim = c(-3, 2)
col = "steelblue"; xlab=expression(E[RATIO])
hist(ratio.log.f, xlim=xlim, col=col, xlab=xlab,  main="Speaker 67")
hist(ratio.log.m, xlim=xlim, col=col, xlab=xlab, main="Speaker 68")
```

It is clear enough that the E_{RATIO} values are smaller than those for the male speaker as a statistical test would confirm (e.g, assuming the data are normally distributed, `t.test(ratio.log.f, ratio.log.m)`). So compared with the male speaker, the female speaker's [ɛ] is relatively closer to [ɪ] than it is to [a].

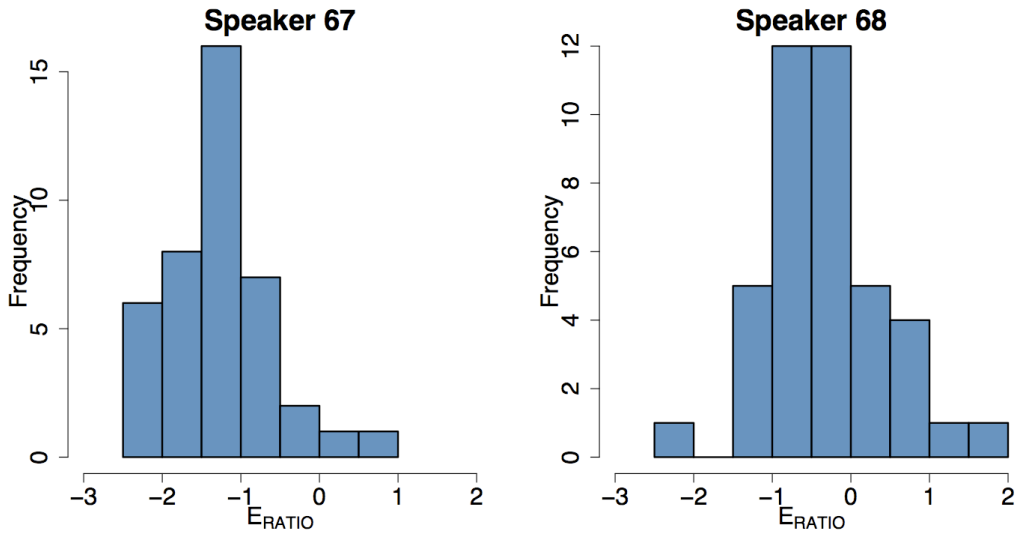

Fig. 3.16: Histogram plots of log. Euclidean distance ratios obtained from measuring the relative distance of [ɛ] tokens to the centroids of [ɪ] and [a] in the F1 x F2 space separately for a female (left) and a male (right) speaker.

3.6 Vowel undershoot and formant smoothing

The calculation of the Euclidean distance to the centre of the vowel space discussed in 3.5.1 is one of the possible methods for measuring vowel undershoot, a term first used by Lindblom (1963) to refer to the way in which vowels failed to reach their targets due to contextual influences such as the flanking consonants and stress. But in such calculations, the extent of vowel undershoot (or expansion) is being measured only at a **single time point**. The technique to be discussed in this section is based on a parameterisation of the **entire formant trajectory**. These parameterisations involve reducing an entire formant trajectory to a set of **coefficients** – this can be thought of as the **analysis** mode. A by-product of this reduction is that if the formant trajectories are reconstructed from the coefficients that were obtained in the analysis mode, then a smoothed formant contour can be derived – this is the **synthesis** mode and it is discussed more fully at the end of this section.

The type of coefficients to be considered are due to van Bergem (1993) and involve fitting a parabola, that is an equation of the form $F = c_0 + c_1t + c_2t^2$, where F is a formant from the start to the end of a vowel that changes as a function of time t. As the equation shows, three are three coefficients c_0, c_1, and c_2 that have to be calculated for each vowel separately from the formant's trajectory. The shape of a parabola is necessarily curved in an arc, either U-shaped if c_2 is positive, or ∩-shaped if c_2 is negative. The principle that lies behind fitting such an equation is as follows. The shape of a formant trajectory, and in particular that of F2, over the extent of a vowel is influenced mostly both by the vowel and by the immediately preceding and following sounds i.e., by the **left and right contexts**. At the vowel target, which for most monophthongs is nearest the vowel's temporal midpoint, the shape is predominantly determined by the phonetic quality of the vowel, but it is reasonable to assume that the influence from the context increases progressively nearer the vowel onset and offset (e.g. Broad & Fertig, 1970). We could imagine a situation on the one hand in which the vowel has no target at all. Just this hypothesis has been suggested for schwa vowels by Browman & Goldstein (1992) in an articulatory analysis and by van Bergem (1994) using acoustic data. In such a situation, a formant trajectory might more or less follow a straight line between its values at the vowel onset and vowel offset: this would happen if

the vowel target has no influence so that the trajectory's shape is entirely determined by the left and right contexts. On the other hand, if a vowel has a prominent target – as if often the case if it is emphasised or prosodically accented (e.g., Pierrehumbert & Talkin, 1990) – then, it is likely to deviate considerably from a straight line joining its onset and offset. Since a formant trajectory often follows reasonably well a parabolic trajectory (Lindblom, 1963), one way to measure the extent of deviation from the straight line and hence to estimate how much it is undershot is to fit a parabola to the formant and then measure the parabola's **curvature**. If the formant is heavily undershot and follows more or less a straight line path between its endpoints, then the curvature will be almost zero while the more prominent the target, the greater deviation from the straight line, and the greater the magnitude of the curvature, in either a positive or a negative direction.

The way that the parabola is fitted in van Bergem (1993) is essentially to rescale the time axis of a trajectory linearly between t = -1 and t = 1. This rescaled time-axis can be obtained using the `seq()` function, if the length of the trajectory in data points is known. As an example, the length of the F2-trajectory for the first segment in the lax vowel data is given by:

```
N = length(vowlax.fdat[1,2]$data)
N
18
```

So the linearly rescaled time axis between t = ± 1 is given by:

```
times = seq(-1, 1, length=N)
```

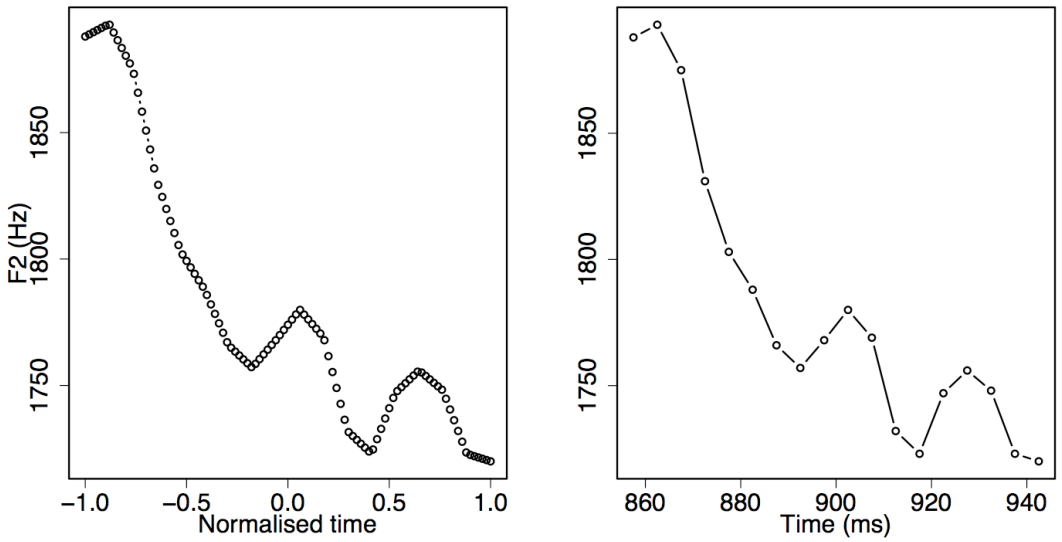

Fig. 3.17: An F2 trajectory (right) and its linearly time normalised equivalent using 101 data points between t = ± 1.

Since a precise estimate of the formant will need to be made at time t = 0, the number of data points that supports the trajectory could be increased using **linear interpolation** with the `approx()` function. The shape of the trajectory stays exactly the same, but the interval along the time axis becomes more fine grained (this procedure is sometimes known as **linear time normalisation**). For example, the F2-trajectory for the first segment in the lax vowel dataset could be given 101 rather than 17 points as follows (an odd number of points is chosen here, because this makes sure that there will be a value at t = 0):

```
# Fig. 3.17
N = 101
F2int = approx(vowlax.fdat[1,2]$data, n=N)
times = seq(-1, 1, length=N)
par(mfrow=c(1,2));
plot(times, F2int$y, type="b", xlab="Normalised time", ylab="F2 (Hz)")
# The original F2 for this segment
plot(vowlax.fdat[1,2], type="b", xlab="Time (ms)", ylab="")
```

There are three unknown coefficients to be found in the parabola $F = c_0 + c_1t + c_2t^2$ that is to be fitted to the data of the left panel in Fig. 3.17 and this requires inserting three sets of data points into this equation. It can be shown (van Bergem, 1993) that when the data is extended on the time axis between $t = \pm 1$, the coefficients have the following values:

```
# c0 is the value at t = 0.
c0 = F2int$y[times==0]
# c1 is half of the difference between the first and last data points.
c1 <- 0.5 * (F2int$y[N] - F2int$y[1])
# c2 is half of the sum of the first and last data points minus c0
c2 <- 0.5 * (F2int$y[N] + F2int$y[1]) - c0
```

If you follow through the example in R, you will get values of 1774, -84, and 30 for c_0, c_1, and c_2 respectively. Since these are the coefficients, the parabola over the entire trajectory can be calculated by inserting these coefficients values into the equation $c_0 + c_1t + c_2t$. So for this segment, the values of the parabola are:

```
c0 + c1 * times + c2 * (times^2)
```

These values could be plotted as a function of time to obtain the fitted curve. However, there is a function in the Emu-R library `plafit()` that does all these steps. So for the present data, the coefficients are:

```
plafit(vowlax.fdat[1,2]$data)
    c0     c1     c2
  1774    -84     30
```

while `plafit(vowlax.fdat[1,2]$data, fit=T)` gives the formant values of the fitted parabola, linearly time-normalised back to the same length as the original data to which the `plafit()` function was applied. So a superimposed plot of the raw and parabolically-smoothed F2-track for this first segment is:

```
# Calculate the values of the parabola
F2par = plafit(vowlax.fdat[1,2]$data, fit=T)
ylim = range(c(F2par, vowlax.fdat[1,2]$data))
xlab="Time (ms)"; ylab="F2 (Hz)"
# Plot the raw values
plot(vowlax.fdat[1,2], type="b", ylim=ylim, xlab=xlab, ylab=ylab)
# Superimpose the smoothed values
par(new=T)
plot(as.numeric(names(F2par)), F2par, type="l", ylim=ylim, xlab=xlab,
ylab=ylab, lwd=2)
```

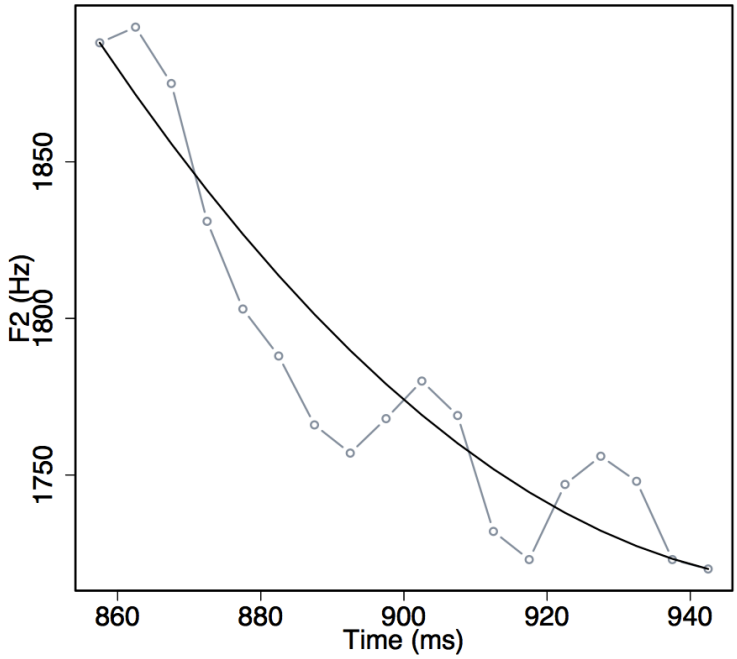

Fig. 3.18: A raw F2 trajectory (gray) and a fitted parabola.

The fitted parabola (Fig. 3.18) always passes through the first and last points of the trajectory and through whichever point is closest to the temporal midpoint. The coefficient c_0 is the y-axis value at the temporal midpoint. The coefficient c_1, being the average of the first and last values, is negative for falling trajectories and positive for rising trajectories. As already mentioned, c_2 measures the trajectory's curvature: positive values on c_2 mean that the parabola has a U-shape, as in Fig. 3.21, negative values that it is ∩-shaped. Notice that these coefficients encode the trajectory's shape **independently of time**. So the above trajectory extends over a duration of about 80 ms, but even if the duration were $1/10^{th}$ or 100 times as great, the coefficients would all be the same, if the trajectory's shape were unchanged. So it would be wrong to say that very much can be inferred about the rate of change of the formant (in Hz/s) from c_1 (or to do so, c_1 would have to be divided by the formant's duration).

We will now consider a worked example of measuring formant curvatures in a larger sample of speech. The analysis of the vowel spaces in the F2 x F1 plane, as well as the Euclidean distance measurements in 3.5 have suggested that the female speaker produces more distinctive vowel targets, or in terms of Lindblom's (1990) H&H theory, her vowels show greater evidence of hyperarticulation and less of a tendency to be undershot. Is this also reflected in a difference in the extent of formant curvature?

In order to address this question, the two speakers' [ɛ] vowels will be compared. Before applying an algorithm for quantifying the data, it is always helpful to look at a few plots first, if this is possible (if only because a gross inconsistency between what is seen and what is obtained numerically often indicates that there is a mistake in the calculation!). A plot of all of the F2 trajectories lined up at the temporal midpoint and shown separately for the two speakers does not seem to be especially revealing (Fig. 3.19, left panel), perhaps in part because the trajectories have different durations and, as was mentioned earlier, in order to compare whether one trajectory is more curved than another, they need to be time normalised to the same length. The option `norm=T` in the `dplot()` function does just this by linearly

stretching and compressing the trajectories so that they extend in time between 0 and 1. There is some suggestion from the time- normalised data (Fig. 3.19 centre) that the female's F2 trajectories are more curved than for the male speaker. This emerges especially clearly when the linearly time-normalised, male and female F2-trajectories are separately averaged (Fig. 3.19, right). However, the average is just that: at best a trend, and one that we will now seek to quantify by calculating the c_2 coefficients of the fitted parabola. Before doing this, here are the instructions for producing Fig. 3.19:

```
# Fig. 3.19
temp = vowlax.l == "E"
par(mfrow=c(1,3)); ylim = c(1500, 2500)
# F2 of E separately for M and F synchronised at the midpoint
dplot(vowlax.fdat[temp,2], vowlax.spkr[temp], offset=.5, ylab="F2 (Hz)",
ylim=ylim, xlab="Time (ms)",legend=F)
# As above with linear time normalisation
dplot(vowlax.fdat[temp,2], vowlax.spkr[temp], norm=T, xlab="Normalised
time", ylim=ylim, legend=F)
# As above and averaged
dplot(vowlax.fdat[temp,2], vowlax.spkr[temp], norm=T, average=T, ylim=ylim,
xlab="Normalised time")
```

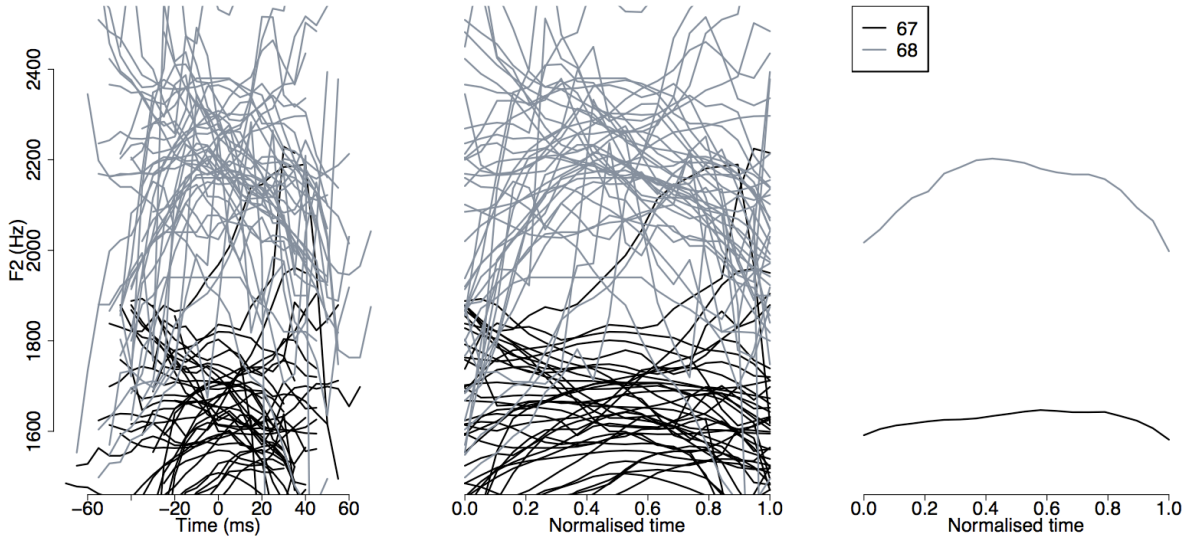

Fig. 3.19: F2 of [ɛ] for the male (black) and female (gray) speakers synchronised at the temporal midpoint (left), linearly time normalised (centre), and linearly time normalised and averaged (right).

The `plafit()` function works on any vector of values, just like the `euclid()` function created earlier. For example, this instruction finds the three coefficients of a parabola that is fitted to 10 random numbers.

```
r = runif(10)
plafit(r)
```

Such a function can be applied to each segment of trackdata separately using the `trapply()` function. By default, `trapply()` returns **a list** because applying a function can often produce results of different lengths for the separate segments. However, since in this case it is known that `plafit()` always returns three elements no matter what the length of the vector to which it is applied, `simplify=T` can be set in the `trapply()` function which will produce a **matrix** whose number of rows is equal to the number of segments (see Appendix A for further details). Thus:

```
# Logical vector to identify E vowels
temp = vowlax.l == "E"
# Matrix of coefficients, one row per segment.
coeffs = trapply(vowlax.fdat[temp,2], plafit, simplify=T)
```

`coeffs` has 3 columns (one per coefficient) and the same number of rows as there are [ɛ] segments (this can be verified with `nrow(coeffs) == sum(temp)`). Fig. 3.20 compares the F2-curvatures of the male and female speakers using a boxplot. There are two outliers (both for the female speaker) with values less than -500 (as `sum(coeffs[,3] < -500)` shows) and these are excluded from the plot by setting the *y*-axis limits:

```
ylim = c(-550, 150)
boxplot(coeffs[,3] ~ factor(vowlax.spkr[temp]), ylab="Amplitude",
ylim=ylim)
```

Fig. 3.20 shows greater negative values on c_2 for the female speaker which is consistent with the view that there is indeed greater curvature in the female speaker's F2 of [ɛ] than for the male speaker.

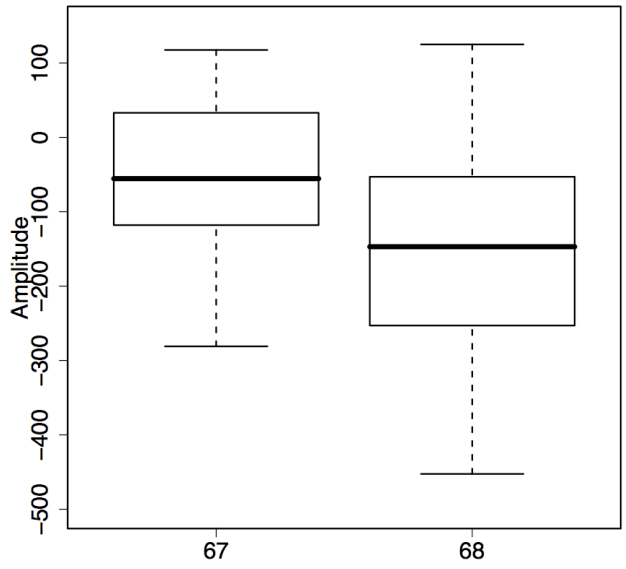

Fig. 3.20: Boxplots of the c_2 coefficient of a parabola fitted to F2 of [ɛ] for the male (left) and female (right) speaker.

As foreshadowed at various stages in this section, the `fit=T` argument applies the function in synthesis mode: it works out the corresponding fitted formant parabola as a function of time. In order to smooth an entire trackdata object, the `trapply()` function can again be used. This time, the output of `trapply()` needs to be a list (thus, `simplify=T` should *not* be set) because the length of the data that is returned is different for each segment

(because segments have different lengths). The `buildtrack` argument can then be used to build the list that is output from `trapply()` into a trackdata object:

```
# Calculate the fitted F2 parabolas for all the vowel data
vow.sm2 = trapply(vowlax.fdat[,2], plafit, T, buildtrack=T)
```

The smoothed and raw F2 data can be superimposed on each other for any segment as in Fig. 3.21 and in the manner described below.

Although fitting a parabola is an effective method of data reduction that is especially useful for measuring formant curvature and hence undershoot, there are two disadvantages as far as obtaining a smoothed contour are concerned:

- not every formant trajectory has a parabolic shape
- parabolic fitting forces a fit at the segment onset, offset, and midpoint as is evident from Figs. 3.21.

One way around both of these problems is to use the **discrete cosine transformation** (see e.g., Watson & Harrington; 1999; Harrington, 2006; Harrington et al., 2008) which will be discussed more fully in Chapter 5. This transformation decomposes a trajectory into a set of coefficients (this is the *analysis* mode) that are the amplitudes of cosine waves of increasing frequency. The number of coefficients derived from the discrete cosine transformation (DCT) is the same as the length of the trajectory. If, in *synthesis* mode, all of these cosine waves are summed, then the original, raw trajectory is exactly reconstructed. However, if only the first few lowest frequency cosine waves are summed, then a smoothed trajectory is derived. Moreover, the fewer the cosine waves that are summed, the greater the degree of smoothing. Therefore, an advantage of this type of smoothing over that of fitting parabolas is that it is possible to control the degree of smoothing. Another advantage is that the DCT does not necessarily force the smoothed trajectory to pass through the values at the onset, offset, and midpoint and so is not as prone to produce a wildly inaccurate contour, if formant onsets and offsets were inaccurately tracked (which is often the case especially if there is a preceding or following voiceless segment).

There is a function in the Emu-R library for computing the DCT coefficients, `dct()` which, just like `plafit()` and `euclid()` takes a vector of values as its main argument. The function can be used in an exactly analogous way to the `plafit()` function in synthesis mode for obtaining smoothed trajectories from the coefficients that are calculated in analysis mode. In the example in Fig. 3.21, a smoothed F2 trajectory is calculated from the first three DCT coefficients, which essentially also models the trajectory as a parabola. This can be done as follows.

```
# Calculate a smoothed-trajectory based on the lowest 4 DCT coefficients
vow.dct2 = trapply(vowlax.fdat[,2], dct, fit=T, 4, buildtrack=T)
```

Fig. 3.21 containing the raw and two types of smoothed trajectories for the 8th segment in the segment list was produced with the following commands:

```
j = 8
# A label vector to identify the trajectories
lab = c("raw", "parabola", "DCT")
# Row-bind the three trajectories into one trackdata object
dat = rbind(vowlax.fdat[j,2], vow.sm2[j,], vow.dct2[j,])
dplot(dat, lab, ylab="F2 (Hz)", xlab="Time (ms)")
```

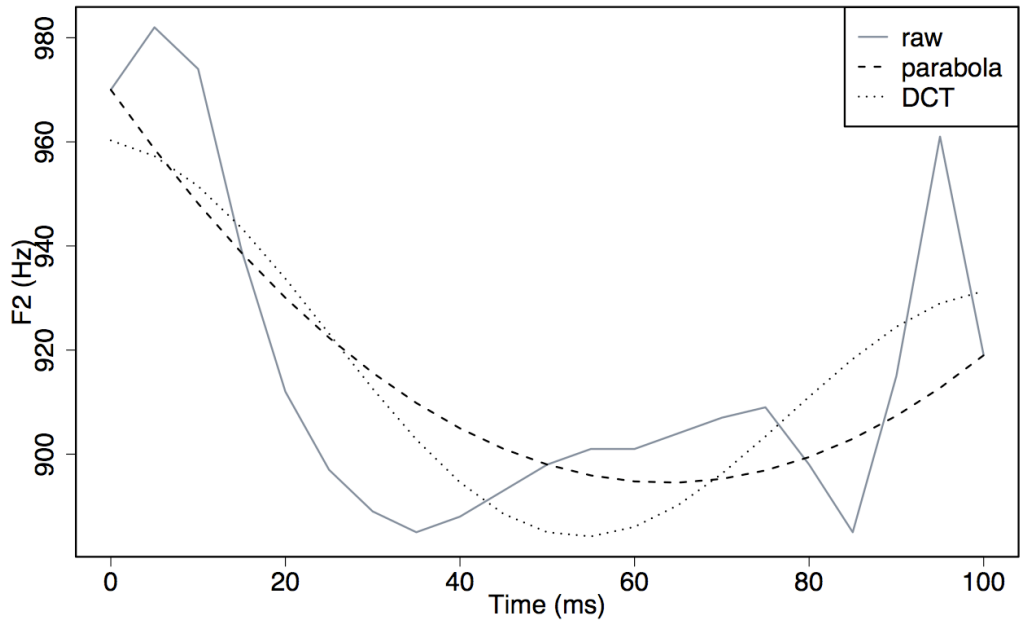

Fig. 3.21: A raw F2 formant trajectory of a back vowel [ɔ] (solid, gray) produced by a male speaker of Standard German and two smoothed contours of the raw signal based on fitting a parabola following van Bergem (1993) (dashed) and the first three coefficients of the discrete cosine transformation (dotted).

3.7 F2 locus, place of articulation and variability

In the production of an oral stop, the vocal tract is initially sealed during which time air-pressure builds up that is then released. At the point of release, the shape of the vocal tract has a marked influence on the acoustic signal and this influence extends at least to the beginning of the formant transitions, if the following segment is a vowel. Since the shape of the vocal tract is quite different for labial, alveolar, and velar places of articulation, the way in which the acoustic signal is influenced following the release differs correspondingly. If the influence extends to the onset of periodicity for the following vowel, then the formant onset frequencies of the same phonetic vowel should be different when it occurs after consonants at different places of articulation.

As a study by Potter, Kopp and Green (1947) had shown, different places of articulation have their greatest influence on the onset of the second formant frequency. But it was the famous perception experiments using hand-painted spectrograms at the Haskins Laboratories that gave rise to the concept of an **F2-locus**. In these experiments, Liberman and colleagues (Delattre, Liberman & Cooper, 1955; Liberman, Delattre, Cooper & Gerstman, 1954; Liberman, et al, 1958) showed that the origin of the second formant frequency influenced the perception of the place of articulation of the following consonant. More specifically, if the F2-onset was low and at around 700 Hz, listeners would predominantly hear /b/; if it was roughly in the 1800 Hz region, they would hear /d/, while if the F2-onset began near 3000 Hz, listeners would perceive /ɡ/ before front vowels. The locus frequency was not at the acoustic vowel onset itself, but at some point prior to it during the consonantal closure. More specifically, in synthesising a CV transition, formants would be painted from the F2-locus somewhere in the consonant closure to the F2-vowel target and then the first part of the transition up the where the voicing for the vowel began would be erased. With

these experiments, Liberman and colleagues also demonstrated that the perception of place of articulation was categorical and this finding was interpreted in favour of the famous motor theory of speech perception (see e.g. Hawkins, 1998 for a thorough review of these issues).

In the years following the Haskins Laboratories experiments, various large-scale acoustic studies were concerned with finding evidence for an F2-locus (including e.g., Fant, 1973, Kewley-Port, 1982. Lehiste & Peterson, 1961b, Öhman, 1966). These studies showed that the most stable F2-locus was for alveolars (i.e., alveolars exhibit the least F2-onset variation of the three places of articulation) whereas velars, which exhibit a great deal of variability depending on the backness of the following vowel, showed no real evidence from acoustic data of a single F2-locus.

From the early 1990s, Sussman and Colleagues applied the concept of a **locus equation** to suggest that F2 transitions might in some form provide invariant cues to the place of articulation (see e.g., Sussman, McCaffrey, & Matthews, 1991; Sussman, Fruchter & Cable, 1995), a position that has also not been without its critics (e.g., Brancazio & Fowler, 1998; Löfqvist, 1999).

Locus equations were first investigated systematically by Krull (1988, 1989) and they provide a numerical index of the extent to which vowels exert a coarticulatory influence on a consonant's place of articulation. Thus, whereas in studies of vowel undershoot, the concern is with the way in which context influences vowel targets, here it is the other way round: that is, the aim is to find out the extent to which vowel targets exert a coarticulatory influence on the place of articulation of flanking consonants.

The basic idea behind a locus equation is illustrated with some made-up data in Fig. 3.24 showing two hypothetical F2-trajectories for [bɛb] and [bob]. In the trajectories on the left, the extent of influence of the vowel on the consonant is nil. This is because, in spite of very different F2 vowel targets, the F2 frequencies of [ɛ] and of [o] both converge to exactly the same F2-locus at 700 Hz for the bilabial – that is, the F2-onset frequency is determined entirely by the consonant with no influence from the vowel. The other (equally unlikely) extreme is shown on the right. In this case, the influence of the vowel on the consonant's place of articulation is maximal so that there is absolutely no convergence to any locus and therefore no transition.

A locus equation is computed by transforming F2 frequencies (top row, Fig. 3.21) as a function of time into the plane of F2-target x F2-onset (bottom row, Fig. 3.24) in order to estimate the extent of coarticulatory influence of the vowel on the consonant. Firstly, when there is no vowel-on-consonant coarticulation (left), the locus equation, which is the line that connects these two [bɛb] and [bob] is horizontal: this is because they both converge to the same locus frequency and so F2-onset is 700 Hz in both cases. Secondly, when vowel-on-coarticulation is at a maximum (right), then the locus equation is a diagonal in this plane because the locus frequency is equal to the target frequency.

A fundamental difference between these two extreme cases of coarticulation is in the **slope** of the locus equation. On the left, the slope in this plane is zero (because the locus equation is horizontal) and on the right it is one (because $F2_{Target} = F2_{Onset}$). Therefore, for real speech data, the slope of the locus equation can be used as a measure of the extent of vowel-on-consonant coarticulation: the closer the slope is to one, the more the consonant's place of articulation is influenced by the vowel (and the slope must always lie between 0 and 1 since these are the two extreme cases of zero and maximal coarticulation). Finally, it can be shown (with some algebraic manipulation – see Harrington & Cassidy 1999, p. 128-130) that the locus frequency itself – that is the frequency towards which transitions tend to converge – can be estimated by establishing where the locus equation and the line $F2_{Target} = F2_{Onset}$ bisect.

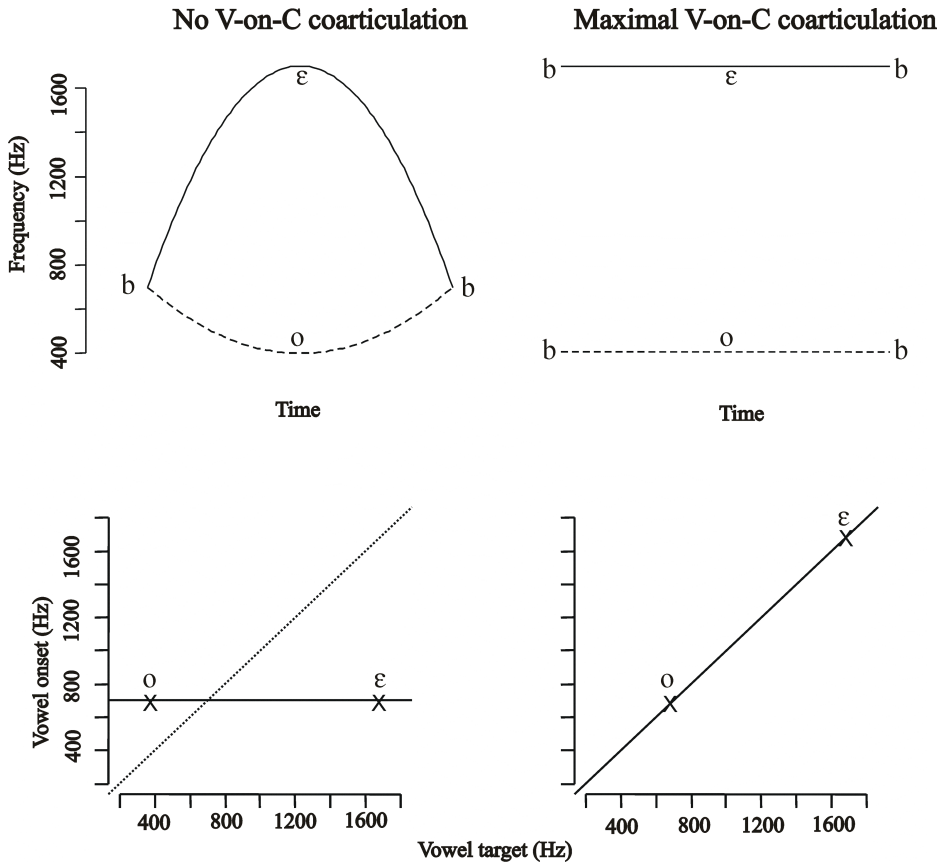

Fig. 3.22: Hypothetical F2-trajectories of [bɛb] (solid) and [bob] (dashed) when there is no V-on-C coarticulation (left) and when V-on-C coarticulation is maximal (right). First row: the trajectories as a function of time. Second row: A plot of the F2 values in the plane of the vowel target x vowel onset for the data in the first row. The dotted line bottom left is the line $F2_{Target} = F2_{Onset}$ that can be used to estimate the locus frequency.

This is shown for the case of zero vowel-on-consonant coarticulation on left in Fig. 3.22: this line bisects the locus equation at the frequency at which the transitions in the top left panel of Fig. 3.22 converge, i.e. at 700 Hz which is the locus frequency. On the right, the line $F2_{Target} = F2_{Onset}$ cannot bisect the locus equation, because it is the same as the locus equation. Because these lines do not bisect, there is no locus frequency, as is of course obvious from the 'transitions' in Fig. 3.22 top right that never converge.

We will now apply this theory to some simple isolated word data produced by a well-known speaker of Australian English, the first author of Clark, Yallop and Fletcher (2007). In 1991, JEC produced a number of isolated /dVd/ words where the V is one of the 13 possible monophthongs of Australian English. The relevant dataset included in the Emu-R library (and taken from the database **isolated** – see Appendix B) is as follows:

`isol`	a segment list of the vowel in /dVd/
`isol.l`	the corresponding vector of vowel labels
`isol.fdat`	parallel trackdata of F1-F4 from the vowel onset to the offset

The task is to investigate whether the coarticulatory influence of the vowel is greater on the final, than on the initial, consonant. There are various reasons for expecting this to be so. Foremost are the arguments presented in Ohala (1990) and Ohala & Kawasaki (1984) that initial CV (consonant-vowel) transitions tend to be a good deal more salient than VC transitions: compatibly, there are many more sound changes in which the vowel and syllable-final consonant merge resulting in consonant loss (such as the nasalization of vowels and associated final nasal consonant deletion in French) than is the case for initial CV syllables. This would suggest that synchronically a consonant and vowel are more sharply delineated from each other in CV than in VC syllables and again there are numerous aerodynamic and acoustic experiments to support this view. Secondly, there are various investigations (e.g., Butcher, 1989, Hoole et al, 1990) which show that in V_1CV_2 sequences where C is an alveolar, the perseverative influence of V_1 on V_2 is greater than the anticipatory influence of V_2 on V_1. This suggests that the alveolar resists coarticulatory influences of the following V_2 (so the alveolar has a blocking effect on the coarticulatory influences of a *following* vowel) but is more transparent to the coarticulatory influences of the preceding V_2 (so the alveolar does not block the coarticulatory influences of a *preceding* vowel to the same extent). Analogously, the vowel-on-consonant coarticulatory influences can be expected to be weaker when the vowel follows the alveolar in /dV/ than when it precedes it in /Vd/.

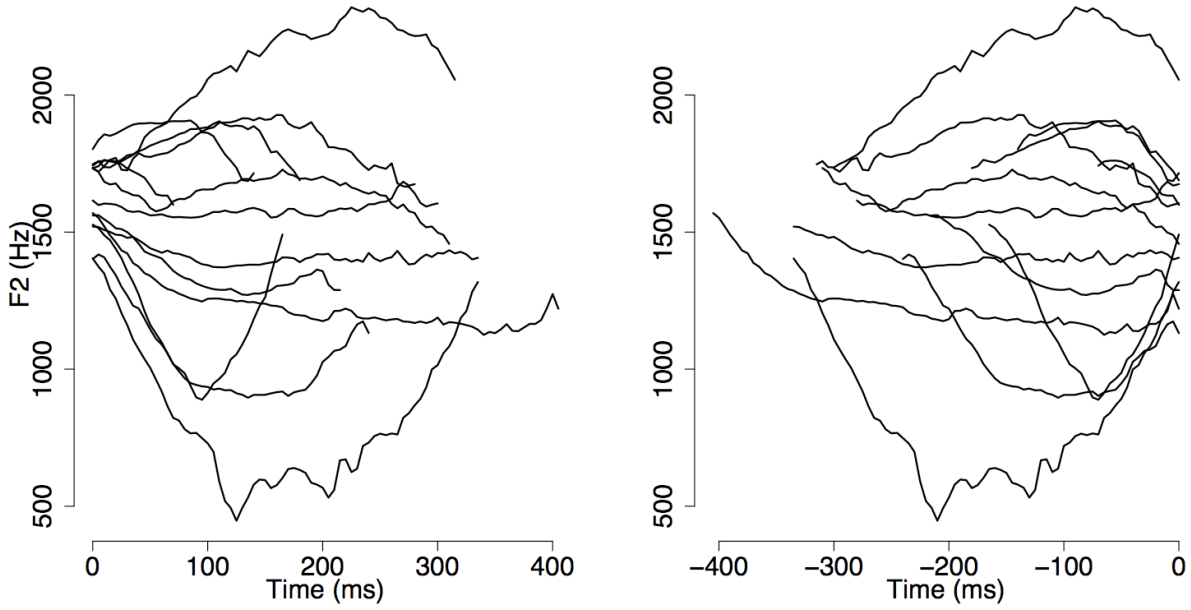

Fig. 3.22: F2 trajectories in isolated /dVd/ syllables produced by a male speaker of Australian English for a number of different vowel categories synchronised at the vowel onset (left) and at the vowel offset (right).

Before proceeding to the details of quantifying the data, we should, as always, produce some plots. In Fig. 3.22, the `dplot()` function is used to plot the second formant frequency trajectories synchronised at the vowel onset on the left and at the vowel offset on the right. The trajectories are of F2 of the separate vowel categories and there is only one token per vowel category, as `table(isol.l)` will show[2]. The plots can be obtained by

[2] The relationship between the machine readable ad phonetic alphabet for Australian vowels is given at the beginning of the book.

setting the `offset` argument in this function argument to `0` (for alignment at the segment onset) and to `1` (for alignment at the segment offset).

```
par(mfrow=c(1,2))
dplot(isol.fdat[,2], offset=0, ylab = "F2 (Hz)", xlab="Time (ms)")
dplot(isol.fdat[,2], offset=1, xlab="Time (ms)")
```

It seems clear enough from Fig. 3.22 that there is greater convergence of the F2-transitions at the alignment point on the left (segment onset) than on the right (segment offset). So far, the hypothesis seems to be supported.

We will now switch to the plane of F2-target x F2-onset (F2-target x F2-offset) and calculate locus equations in these two planes. The relevant data can be extracted using the `dcut()` function. For the vowel target, the instruction below uses the F2-data at the temporal midpoint of the formant:

```
# F2-onset
f2onset = dcut(isol.fdat[,2], 0, prop=T)
# F2-"target"
f2targ = dcut(isol.fdat[,2], .5, prop=T)
# F2-offset
f2offset= dcut(isol.fdat[,2], 1, prop=T)
```

The procedure is now to plot the data and then to calculate a straight line of best fit known as a **regression line** through the scatter of points: that is, there will not be two points that lie on the same line as in the theoretical example in Fig. 3.22, but several points that lie close to a **line of best fit**. The technique of linear regression, which for speech analysis in R is described in detail in Johnson (in press), is to calculate such a line, which is defined as the line to which the distances of the points are minimised. The function `lm()` is used to do this. Thus for the vowel onset data:

```
plot(f2targ, f2onset)
# calculate the line of best fit by regressing F2-onset on F2-target
regr = lm(f2onset ~ f2targ)
# plot the regression line
abline(regr)
```

The information about the slope is stored in `$coeff` which also gives the intercept (the value at which the regression line cuts the y-axis, i.e. the F2-onset axis). The slope of the line is just over 0.27:

```
regr$coeff
 (Intercept)        f2targ
1217.8046955    0.2720668
```

Recall that the best estimate of the locus is where the line $F2_{Target} = F2_{Onset}$ cuts this regression line. Such a line can be superimposed on the plot as follows:

```
abline(0, 1, lty=2)
```

There is a function in the Emu-R library, `locus()` that carries out all of these operations. The first two arguments are the *x*- and *y*-axis data (in this case the F2-target and the F2-onset respectively) and there is a third optional argument for superimposing a parallel set of labels.

In this example, the vowel labels are superimposed on the scatter and the *x*- and *y*-ranges are set to be the same. The F2-target x F2-onset data is plotted in the left panel, the F2-target x F2-offset data in the right panel:

```
xlim = c(500, 2500); ylim = xlim; par(mfrow=c(1,2))
xlab = "F2-target (Hz)"; ylab = "F2-onset (Hz)"
stats.on = locus(f2targ, f2onset, isol.l, xlim=xlim, ylim=ylim, xlab=xlab,
ylab=ylab)
stats.off = locus(f2targ, f2offset, isol.l, xlim=xlim, ylim=ylim,
xlab=xlab)
```

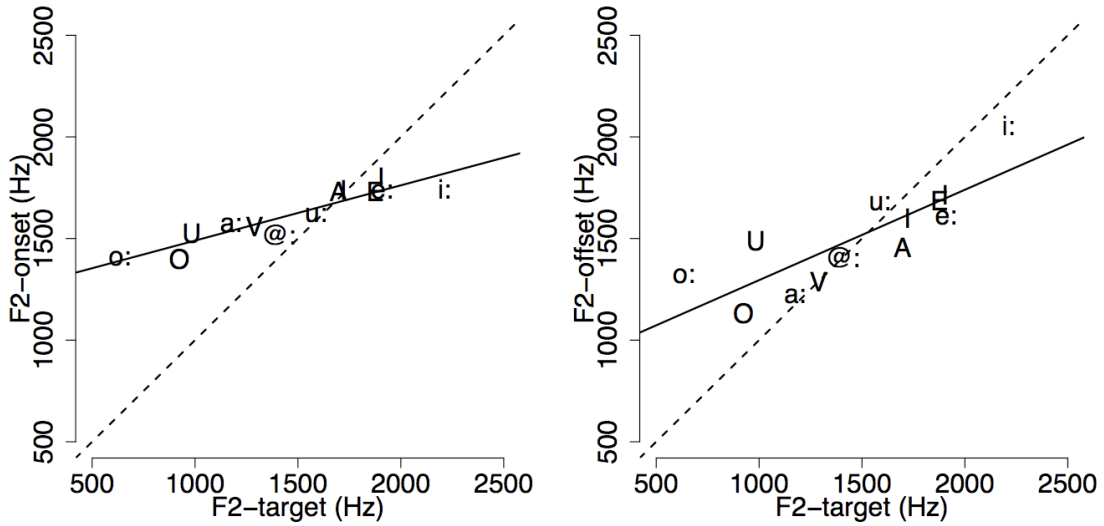

Fig. 3.23. Locus equations (solid lines) for /dVd/ words produced by a male speaker of Australian English for F2-onset (*left*) and F2-offset (*right*) as a function of F2-target. The dotted line is the line $y = x$ and is used to estimate the locus frequency at the point of intersection with the locus equation.

The Figure shows that the regression line (locus equation) is not as steep for the F2-onset compared with the F2-offset data. The actual values of the slopes (and intercepts) are given by:

#Slope in the F2-target x F2-onset plane

```
stats.on$coeff
 (Intercept)      target
1217.8046955   0.2720668
```

Slope in F2-target x F2-offset plane

```
stats.off$coeff
(Intercept)      target
850.4935279   0.4447689
```

The `locus()` function also calculates the point at which the regression line and the lines $F2_{Target} = F2_{Onset}$ (Fig. 3.23, left) and $F2_{Target} = F2_{Offset}$ (Fig. 3.23, right) bisect, thus giving a best estimate of the locus frequency. These estimates are in `$locus` and the calculated values for the data in the left and right panels of Fig. 3.23 respectively are 1673

Hz and 1532 Hz. The reason why there is this 141 Hz difference in the calculation of the locus frequencies for initial as opposed to final /d/ is not immediately clear; but nevertheless as Fig. 3.23 shows, these are reasonable estimates of the point at which the F2 transitions tend, on average, to converge. Finally, statistical diagnostics on the extent to which the regression line could be fitted to the points is given by the summary function:

```
summary(stats.on)
Call:
lm(formula = onset ~ target)

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.218e+03  5.134e+01  23.721 8.51e-11 ***
target      2.721e-01  3.308e-02   8.224 5.02e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 53.12 on 11 degrees of freedom
Multiple R-Squared: 0.8601,     Adjusted R-squared: 0.8474
F-statistic: 67.64 on 1 and 11 DF,  p-value: 5.018e-06
```

The probability that the regression line slope accounts for the data is given by the results of *t*-test in the line beginning `target`. The null hypothesis is that there is no relationship at all between the F2-onset and F2-target (which would mean that the slope of the regression line would be zero): the *t*-test computes the likelihood that this could be the case. This probability is `5.02e-06` or 0.00000502 i.e., very small. So the null hypothesis is rejected. The other important diagnostic is given under `adjusted R-squared` which is a measure of how much of the variance is explained by the regression line. It shows that for these data, just over 86% of the variance is explained by the regression line and the probability that this value is different from zero is given by the results of an F-test in the next line beginning 'F-statistic'.

The analysis of these (very limited) data lends some support to the view that the coarticulatory influence of the vowel on an alveolar stop is greater when the consonant is in final than when it is in initial position. However, data from continuous speech and above all for other consonant categories will not always give such clear results so the locus equation calculations must be used with care. In particular, an initial plot of the data to check the formant transitions for any outliers due to formant tracking errors is essential. If the slope of the line is negative or greater 1, then it means either that the data could not be reliably modelled by any kind of regression line and/or that no sensible locus frequency could be computed (i.e., the formant transitions do not converge, and the resulting locus frequency calculation is likely to be absurd).

3.8 Exercises

These questions make use of the objects in the Emu-R library (see also Appendix B).

A. Plotting vowel ellipses and removing outliers (sections 3.1 and 3.2)

A1. Extract the formant data at the temporal midpoint from the trackdata object `vowlax.fdat` and convert the Hz values to Bark.

A2. F3–F2 in Bark was suggested by Syrdal & Gopal (1986) as an alternative to F2 in Hz for representing vowel backness. Make a plot like the one in Fig. 3.27 of the female speaker's (speaker `68`) `E` vowels in the plane of –F1 in Bark (*x*-axis) and F3–F2 in Bark (*y*-axis) showing the data points.

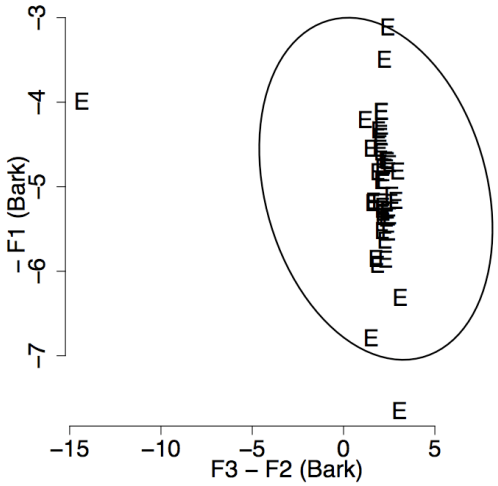

Fig. 3.24: The female speaker's `E` vowels in the plane of F3-F2 Bark x –F1 Bark.

A3. Using a logical vector, create a matrix that is the same as the one in A.1 (of Bark values at the temporal midpoint) but which excludes the very obvious outlier shown in the upper left of Fig. 3.24.

A4. Use the same logical vector as in A3, make a vector of speaker labels from `vowlax.spkr` and a vector of vowel labels from `vowlax.l` that are each parallel to the matrix that you created in A3.

A5. Make two plots as shown in Fig. 3.25 of the male and female speakers' data in the plane of F3–F2 (Bark) x –F1 (Bark) scaled to the same axes.

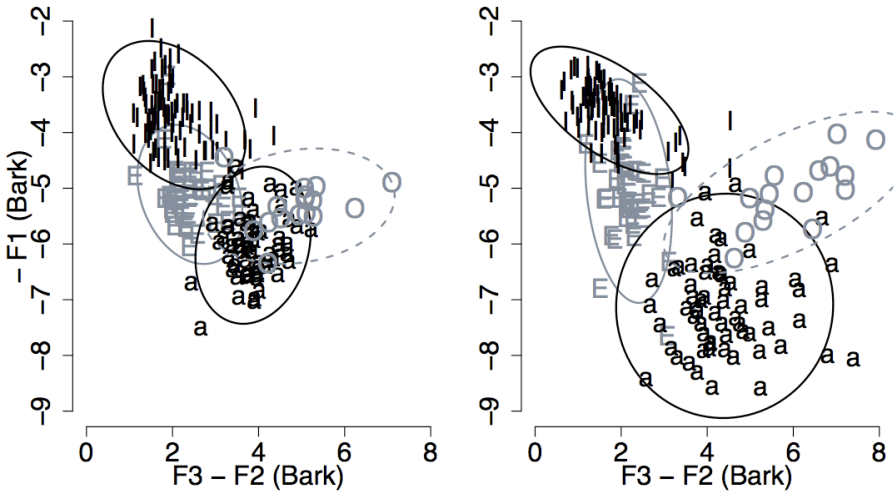

Fig. 3.25: German lax monophthongs produced by a male (left) and a female (right) speaker in the plane of F3-F2 Bark x –F1 Bark. Data extracted at the temporal midpoint of the vowel.

B. Finding targets (section 3.3)

B1. The trackdata object `vowlax.rms` contains dB-RMS data that is parallel to the segment list `vowlax` and the other objects associated with it. Use the `targtime()` function from 3.3 or otherwise to find for each segment the time within the vowel at which the dB-RMS reaches a maximum value.

B2. Extract the F1 and F2 data from the segment list `vowlax` for the male speaker `67` at the target time defined by B1 and make a plot of the vowels in the F2 x F1 plane.

B3. The following R objects are available for three diphthongs in German [aɪ, aʊ, ɔʏ] for the same two speakers as for the lax vowel data in A above. The diphthongs are `aI`, `aU`, and `OY` in MRPA respectively (see the beginning of the book for a Table of IPA-MRPA correspondences and examples).

`dip`	a segment list
`dip.fdat`	trackdata of F1-F4
`dip.l`	a vector of parallel diphthong labels
`dip.spkr`	a vector of speaker labels

Make a plot of F1 as a function of time for the male speaker's `aU` diphthong thereby verifying that F1 seems to reach a target/plateau in the first half of the vowel.

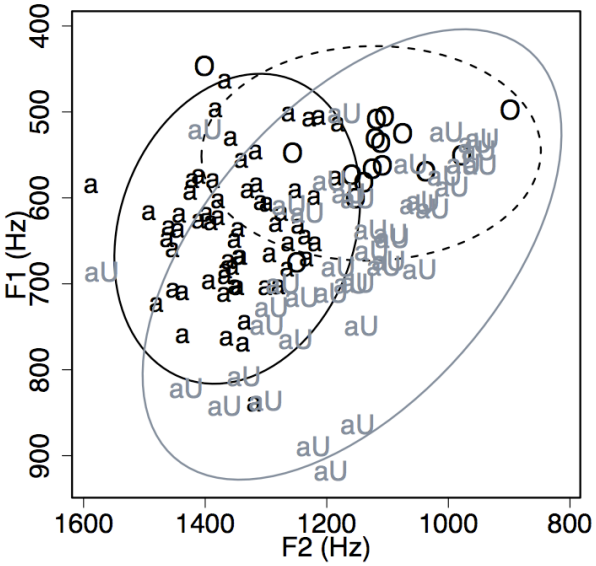

Fig. 3.26: 95% confidence ellipses in the plane of F2 x F1 at the temporal midpoint of [a, ɔ] (`a`, `O`) and at the time at which F1 reaches a maximum in [aʊ] (`aU`) for a male speaker of Standard German.

B4. Make a matrix of the F1 and F2 values at the time at which F1 reaches a maximum in the first half of `aU`. Create a vector of `aU` labels that is parallel to this matrix (i.e., a vector of the same length as there are rows in the matrix and consisting entirely of `aU`, `aU`, `aU`…).

B5. The task is to check how close the quality of the diphthong at the alignment point in B4 is to the same speaker's (`67`) lax [a, ɔ] (`a`, `O`) vowels extracted from F1-F2 data at the temporal midpoint. Using the data from B4 above in combination with the R-objects of these

lax monophthongs in A1, make a plot of the kind shown in Fig. 3.26 for this purpose. What can you conclude about the quality of the first target of this diphthong? What phonetic factor might cause it to have lower F2 values than the lax monophthong `a`?

C. Formant curvature and undershoot (section 3.6)
(This question makes use of the related objects `dip`, `dip.fdat`, `dip.l`, `dip.spkr` – see `data(package="emu")`).

C1 Which diphthong, [a`I`] or [`OY`] would you expect to have greater curvature in F1 and why?

C2. Produce a linearly time-normalised plot of F1 of the female speaker's `aI` and `OY` diphthongs. Does this match your predictions in C1?

C3. Fit parabolas to F1 of the diphthongs for both speakers (i.e., create a matrix of coefficients that is parallel to the R objects for the diphthong segments). Based on the evidence of the hyperarticulation differences, which speaker is expected to show less formant F1-curvature in the diphthongs?

C4. Produce on one page four boxplots (analogous to Fig. 3.20) of the curvature-parameter for the label combinations `aI.67` (the male speaker's `aI` diphthongs), `OY.67`, `aI.68` (the female speaker's `aI` diphthongs), and `OY.68`.

D. F2-loci (section 3.7)
D1.
(a) How would you expect the slope of the locus equation to be different in continuous speech compared with the type of isolated, citation-form production examined in 3.7 above?

(b) If the female speaker `68` hyperarticulates compared with the male speaker `67`, i.e., produces more distinctive consonant and vowel targets, how might their locus equation slopes differ for the same place of articulation?

(c) Check your predictions by calculating and plotting locus equations in the plane of F2-Onset x F2-Target for /d/ preceding the lax vowels in the segment list `vowlax` separately for both speakers (the left context labels are given in `vowlax.left`; take F2 at the temporal midpoint for F2-target values). Set the ranges for the *x*- and *y*-axes to be between 1000 Hz and 2500 Hz in both cases.

D2. How would you expect (a) the estimated F2-locus frequency and (b) the slope of the locus equation for initial [v] to compare with those of initial [d]? Check your hypotheses by making locus equations following initial [v] (MRPA `v`) analogous to the ones above for initial [d] and compare the F2-loci and slopes between these places of articulation for both speakers.

D3. Make a time-aligned plot colour-coded for [v] or [d] of F2-trajectories for the female speaker following these two consonants (a plot like the one in Fig. 3.19 but where the trajectories are of F2 for the vowels following [d] or [v])). Is the plot consistent with your results from E2 (i.e, locus differences and slope differences?).

3.9 Answers

A1.

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

A2.

```
temp = vowlax.spkr=="68" & vowlax.l=="E"
eplot(cbind(mid[temp,3]-mid[temp,2], -mid[temp,1]), vowlax.l[temp],
dopoints=T, xlab="- F1 (Bark)", ylab="F3 - F2 (Bark)")
```

A3.

```
temp = vowlax.spkr=="68" & vowlax.l=="E" & mid[,3]-mid[,2] < -12
mid = mid[!temp,]
```

A4.

```
mid.sp = vowlax.spkr[!temp]
mid.l = vowlax.l[!temp]
```

A5.

```
temp = mid.sp=="67"
par(mfrow=c(1,2)); xlim = c(0, 8);  ylim = c(-9, -2)
ylab="- F1 (Bark)"; xlab="F3 - F2 (Bark)"
eplot(cbind(mid[temp,3]-mid[temp,2], -mid[temp,1]), mid.l[temp],
dopoints=T, xlim=xlim, ylim=ylim, xlab=xlab, ylab=ylab)
eplot(cbind(mid[!temp,3]-mid[!temp,2], -mid[!temp,1]), mid.l[!temp],
dopoints=T, xlim=xlim, ylim=ylim, xlab=xlab)
```

B1.

```
targtime <- function(dat, fun=max)
{
temp = dat == fun(dat)
times = tracktimes(dat)
times[temp][1]
}

mtime = trapply(vowlax.rms, targtime, simplify=T)
```

B2.

```
form = dcut(vowlax.fdat[,1:2], mtime)
temp = vowlax.spkr == "68"
eplot(form[temp,], vowlax.l[temp], centroid=T, form=T)
```

B3.

```
temp =  dip.l == "aU"
dplot(dip.fdat[temp,1], ylab="F1 (Hz)")
```

B4.

```
maxf1 = trapply(dcut(dip.fdat[,1], 0, 0.5, prop=T), targtime, simplify=T)
temp =  dip.l == "aU"
formau = dcut(dip.fdat[temp,1:2], maxf1[temp])
labsau = dip.l[temp]
```

B5. [ʊ] has a backing effect on [a] in the diphthong [aʊ], thus lowering F2.

```
temp = vowlax.l %in% c("a", "O") & vowlax.spkr == "67"
mono.f.5 = dcut(vowlax.fdat[temp,1:2], .5, prop=T)
both = rbind(mono.f.5, formau)
both.l = c(vowlax.l[temp], labsau)
eplot(both, both.l, form=T, dopoints=T, xlab="F2 (Hz)", ylab="F1 (Hz)")
```

C1. [aɪ] because the change in openness of the vocal tract is greater than for [ɔɪ]

C2.

```
temp = dip.l  %in% c("aI", "OY") & dip.spkr == "68"
dplot(dip.fdat[temp,1], dip.l[temp], norm=T)
```

Yes.

C3. Speaker 67.

```
coeff = trapply(dip.fdat[,1], plafit, simplify=T)
```

C4.

```
temp = dip.l %in% c("aI", "OY")
boxplot(coeff[temp,3] ~ dip.l[temp] * dip.spkr[temp])
```

D1.
(a) The slope in continuous speech should be higher because of the greater coarticulatory effects.
(b) The slopes for speaker 68 should be lower.
(c)

```
temp = vowlax.left=="d" & vowlax.spkr == "67"
on.m = dcut(vowlax.fdat[temp,2], 0, prop=T)
mid.m = dcut(vowlax.fdat[temp,2], .5, prop=T)

temp = vowlax.left=="d" & vowlax.spkr == "68"
on.f = dcut(vowlax.fdat[temp,2], 0, prop=T)
mid.f = dcut(vowlax.fdat[temp,2], .5, prop=T)


par(mfrow=c(1,2)); xlim=ylim=c(1000, 2500)
l.m.d = locus(mid.m, on.m, xlim=xlim, ylim=ylim)
l.f.d = locus(mid.f, on.f, xlim=xlim, ylim=ylim)
```

D2.
(a) The locus frequency for [v] should be lower because labials according to the locus theory have a lower F2-locus than alveolars.

(b) The slope of the locus equation for [v] should be higher. This is because alveolars are supposed to have the most stable locus of labials, alveolars, and velars, i.e., the place of articulation for alveolars, and hence the F2-locus, shifts the least due to the effects of vowel context.

```
temp = vowlax.left=="v" & vowlax.spkr == "67"
on.m = dcut(vowlax.fdat[temp,2], 0, prop=T)
mid.m = dcut(vowlax.fdat[temp,2], .5, prop=T)
```

```
temp = vowlax.left=="v" & vowlax.spkr == "68"
on.f = dcut(vowlax.fdat[temp,2], 0, prop=T)
mid.f = dcut(vowlax.fdat[temp,2], .5, prop=T)

par(mfrow=c(1,2)); xlim=ylim=c(1000, 2500)
l.m.v = locus(mid.m, on.m, xlim=xlim, ylim=ylim)
l.f.v = locus(mid.f, on.f, xlim=xlim, ylim=ylim)
```

The F2 locus for [v] (given by `l.m.v$locus` and `l.f.v$locus`) for the male and female speakers respectively are, rounded to the nearest Hz, 749 Hz and 995 Hz, i.e. markedly lower than for alveolars. The slopes of the locus equations for [v] (given by entering by `l.m.v` and `l.f.v`) are 0.794 and 0.792 for the male and female speakers respectively and these are both higher than for the alveolar.

D3.

```
temp = vowlax.left %in% c("v", "d") & vowlax.spkr=="67"
dplot(vowlax.fdat[temp,2], vowlax.left[temp], ylab="F2 (Hz)")
```

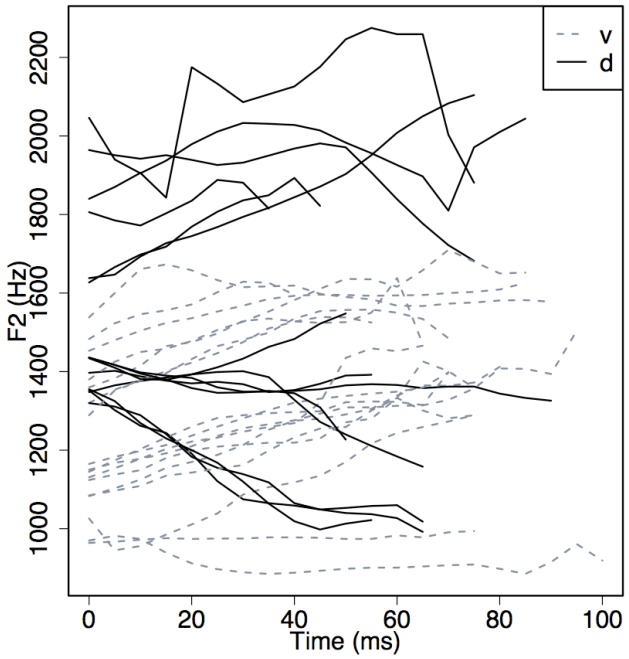

Fig. 3.27: F2 transitions for a female speaker of Standard German following [d] (solid) and [d] (dashed, gray).

Yes. The [v] and [d] transitions point to different locis as Fig. 3.27 shows. Fig. 3.27 also shows that the locus equation slope for [d] is likely to be lower because, even though the F2 range at the vowel target is quite large (over 1000 Hz) at time point 50 ms, the F2-transitions for [d] nevertheless converge to a range of about 700 Hz at F2-onset (t = 0 ms). There is much less evidence of any convergence for [v] and this is why the locus equation slope for [v] is higher than for [d].