

GAM exercise

Cesko Voeten

Leiden University Centre for Linguistics
Leiden Institute for Brain and Cognition
Fryske Akademy

1 The problem

We're going to reproduce a small part of chapter 2 from my dissertation (Voeten 2020), which you can download from <https://surfdrive.surf.nl/files/index.php/s/1smuplppss501ga>. The goal of this chapter was to investigate a Dutch allophonic rule that looks something like the below:

$$(1) \quad \varepsilon i \rightarrow \varepsilon: / _ l]_\sigma$$

What makes this hard is that Dutch coda /l/ is a dark [ɫ] (exactly like English) *and* is currently undergoing a process of vocalization, which makes it impossible to segment it off of the vowel of interest. But with GAMs, we don't have to!

2 The data

The data come from a word-list reading task, where 160 participants total from 8 dialect regions (four in the Netherlands and four in Flanders) were asked to read a small number of words. Two example words are *meid* [meit] and *geil* [yɛ:ɫ]. Words like these have been processed through the following steps:

1. The start of the vowel was identified in Praat, so a boundary was placed between the [m] and the [ɛi] or between the [y] and the [ɛ:];
2. For the non-/l/ words, including *meid*, the end of the vowel was identified and a boundary was placed there, so between e.g. the [ɛi] and the [t];
3. For the /l/ words, this wasn't possible so the boundary was placed after the [ɫ];
4. Starting *at* the initial boundary, formants were measured every 10 ms;

- These samples were scaled to range from 0 (0% realization, i.e. the first measure taken at the initial boundary) to 100 (100% realization rounded down from the final boundary).

You can get the data from GitHub. Read them in like so:

```
data <- read.csv('data.csv', stringsAsFactors=TRUE) |>
  within({
    contrasts(gender) <- contr.sum
    contrasts(following) <- contr.SAS
    region <- factor(region, levels=c('NM', 'NN', 'NS', 'FB',
                                         'FL', 'FE', 'FW', 'NR'))
    contrasts(region) <- contr.sum
  })
```

If your R doesn't understand the `|>` operator, update it, or load library `magrittr` and use its `%>%` pipe instead.

This is the structure of the data:

```
str(data)

## 'data.frame': 301689 obs. of 10 variables:
## $ participant: Factor w/ 160 levels "NMmj1","NMmj2",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ word       : Factor w/ 22 levels "beuk","beul",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ region     : Factor w/ 8 levels "NM","NN","NS",...: 1 1 1 1 1 1 1 1 ...
## ...- attr(*, "contrasts")= num [1:8, 1:7] 1 0 0 0 0 0 -1 0 1 ...
## ...- attr(*, "dimnames")=List of 2
## ... .$. : chr [1:8] "NM" "NN" "NS" "FB" ...
## ... .$. : NULL
## $ gender     : Factor w/ 2 levels "f","m": 2 2 2 2 2 2 2 2 2 2 ...
## ...- attr(*, "contrasts")= num [1:2, 1] 1 -1
## ...- attr(*, "dimnames")=List of 2
## ... .$. : chr [1:2] "f" "m"
## ... .$. : NULL
## $ age        : Factor w/ 2 levels "o","y": 2 2 2 2 2 2 2 2 2 2 ...
## $ vowel      : Factor w/ 6 levels "2:","9y","Au",...: 1 1 1 1 1 1 ...
## $ time       : num 0 5.56 11.11 16.67 22.22 ...
## $ following   : Factor w/ 2 levels "l","P": 2 2 2 2 2 2 2 2 2 2 ...
## ...- attr(*, "contrasts")= num [1:2, 1] 1 0
## ...- attr(*, "dimnames")=List of 2
## ... .$. : chr [1:2] "l" "P"
## ... .$. : chr "1"
## $ value      : num 270 304 341 362 381 ...
## $ formant    : Factor w/ 3 levels "f1","f2","f3": 1 1 1 1 1 1 1 1 1 1 ...
```

This is what's in the columns:

participant : the speaker ID code;
word : the word that they read;
region : the dialect region, where regions starting with *N* are in The Netherlands and regions starting with *F* are in Flanders;
gender : the speaker's sex;
age : the speaker's age (coded as young or old);
vowel : the vowel they're saying;
time : the measurement point (from 0 to 100);
following : either 1 if the following vowel was a coda /l/ or P if it wasn't;
value : the formant measurement;
formant : the name of the formant (f1, f2, f3).

We will only focus on the first formant (this makes the critical difference between [ɛɪ] and [ɛɔ]) and only for the (ɛɪ) vowel. Please execute the following:

```
data <- data[data$formant == 'f1' & data$vowel == 'Ei',]
```

Explore the data as you like, e.g. by plotting things.

3 Running our first GAM

We're interested in a model that fits the first formant (which is now all in **value**) as a smooth function of time. We additionally want to incorporate regional differences. Finally, your exploratory playing around might have revealed a minor gender effect, in that there is a main effect of gender. To make sure we're on the same page, a very simple translation of these requirements into a GAM call would get you:

```
library(mgcv)

## Loading required package: nlme
## This is mgcv 1.8-36. For overview type 'help("mgcv-package")'.

model <- bam(value ~ gender + region + s(time,by=region),data=data,
              discrete=TRUE,control=list(trace=TRUE))

## Setup complete. Calling fit
```

```

## Deviance = 0 Iterations - 1
## Deviance = 477586508.492094 Iterations - 2
## Deviance = 472378523.759899 Iterations - 3
## Deviance = 469362358.400384 Iterations - 4
## Deviance = 462733569.341472 Iterations - 5
## Deviance = 459169311.608237 Iterations - 6
## Deviance = 455550335.382758 Iterations - 7
## Deviance = 453368579.004614 Iterations - 8
## Deviance = 453082137.686567 Iterations - 9
## Deviance = 453061822.373329 Iterations - 10
## Deviance = 453061587.518168 Iterations - 11
## Deviance = 453061587.445153 Iterations - 12

## Fit complete. Finishing gam object.
##           user.self sys.self elapsed
## initial      0.04    0.00    0.04
## gam.setup     0.14    0.00    0.14
## pre-fit       0.00    0.00    0.00
## fit          0.03    0.02    0.04
## finalise     0.00    0.00    0.00

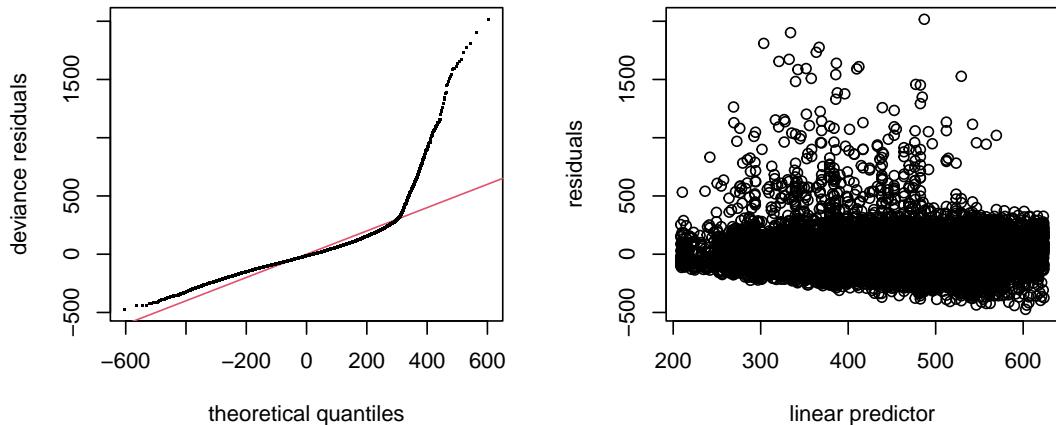
```

We're using `bam` because `gam` is too slow (with the default optimizer). We're using `discrete=TRUE` to make it even faster and because we'll need it later. This means REML only and no model comparisons. I'm also passing `control=list(trace=TRUE)` because I like to be able to guess if convergence is going to take one more minute or one more day. You might also want to fiddle with arguments like `nthreads` for even more speed—see the help pages.

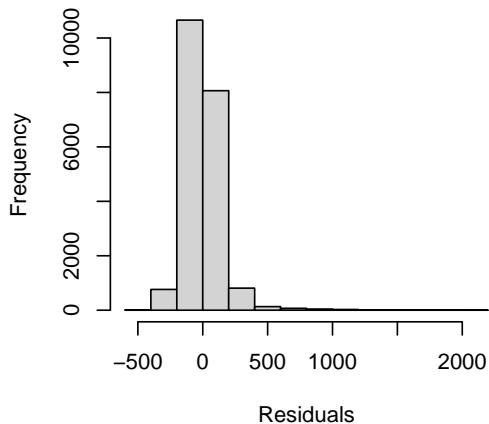
Here is the output of `gam.check`. What do you think? Also, do you think we missed something in our formula, conceptually? Here starts a journey that's all yours! But below you will find some pointers.

```
gam.check(model)
```

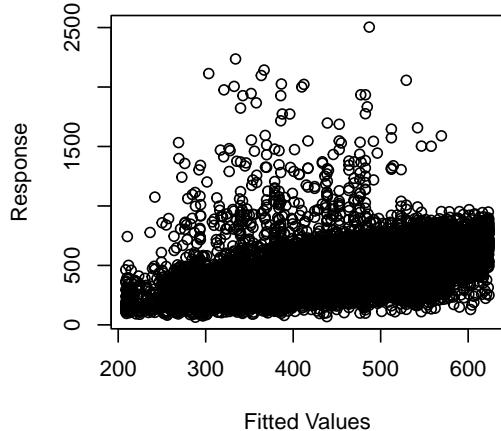
Resids vs. linear pred.



Histogram of residuals



Response vs. Fitted Values



```
##  
## Method: fREML   Optimizer: perf chol  
## $grad  
## [1] 1.381117e-13 -4.467537e-13  1.194600e-13  3.237410e-13 -4.884981e-15  
## [6] 5.417888e-14 -1.887379e-13 -3.330669e-13 -9.822543e-11  
##  
## $hess  
## [,1]           [,2]           [,3]           [,4]           [,5]  
## 2.733731e+00 -2.553735e-07 -3.313483e-07 -4.146672e-07 -1.509659e-08  
## -2.553735e-07  2.916525e+00 -7.869724e-08 -9.848600e-08 -3.585533e-09  
## -3.313483e-07 -7.869724e-08  2.881069e+00 -1.277860e-07 -4.652245e-09  
## -4.146672e-07 -9.848600e-08 -1.277860e-07  3.006109e+00 -5.822071e-09
```

```

## -1.509659e-08 -3.585533e-09 -4.652245e-09 -5.822071e-09 1.867927e+00
## -7.667683e-08 -1.821122e-08 -2.362913e-08 -2.957078e-08 -1.076569e-09
## -4.006264e-07 -9.515121e-08 -1.234592e-07 -1.545034e-07 -5.624935e-09
## 2.367090e-07 5.621984e-08 7.294549e-08 9.128794e-08 3.323477e-09
## d -3.661317e+00 -3.622263e+00 -3.424560e+00 -3.571212e+00 -2.663806e+00
## [,6] [,7] [,8] [,9]
## -7.667683e-08 -4.006264e-07 2.367090e-07 -3.661317
## -1.821122e-08 -9.515121e-08 5.621984e-08 -3.622263
## -2.362913e-08 -1.234592e-07 7.294549e-08 -3.424560
## -2.957078e-08 -1.545034e-07 9.128794e-08 -3.571212
## -1.076569e-09 -5.624935e-09 3.323477e-09 -2.663806
## 2.583967e+00 -2.856951e-08 1.688021e-08 -2.643622
## -2.856951e-08 2.325637e+00 8.819686e-08 -3.228235
## 1.688021e-08 8.819686e-08 3.009272e+00 -3.493817
## d -2.643622e+00 -3.228235e+00 -3.493817e+00 10294.000000
##
## Model rank = 81 / 81
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##          k'   edf k-index p-value
## s(time):regionNM 9.00 8.32    0.99    0.17
## s(time):regionNN 9.00 8.24    0.99    0.25
## s(time):regionNS 9.00 7.85    0.99    0.20
## s(time):regionFB 9.00 8.14    0.99    0.20
## s(time):regionFL 9.00 6.33    0.99    0.21
## s(time):regionFE 9.00 6.29    0.99    0.20
## s(time):regionFW 9.00 7.46    0.99    0.22
## s(time):regionNR 9.00 7.99    0.99    0.18

```

4 Moving on from here

Here are things you might want to try, roughly in the order you might want to try them:

1. Remove obvious outliers incurred by the automatic formant measurements. Very rough guideline: if it's < 100 Hz or > 1000 Hz, it definitely ain't a remotely valid F1. You can do better by doing some plotting, but this is what I did as a very rough first step in my chapter.
2. Check if you missed any predictors, such as random effects! Don't bother with random effects by words, though, because you need at least > 5 levels of a ran-

dom effect for it to be stably estimable (according to the friendly statisticians at [r-sig-mixed](#));

- Free hint: note that every subject is from exactly one dialect region.
3. Check if you missed any structure your smooths by checking their k -values;
 4. Check if you missed any structure in your smooths by checking their autocorrelation;
 5. Fit to scaled- t errors (this is gonna be slow, though).

For autocorrelation, read `bam`'s help page to figure out how to use the `AR.start` argument (hint: `time == 0`). `rho` can be guesstimated by fitting a model without AR and computing the average autocorrelation. Here is some free code to get you started—it computes the AR in the model residuals for every trial separately, and then gives you its average:

```
res <- resid(model)
bins <- findInterval(1:length(res), which(data$AR.start))
res.binned <- split(res, bins)
rhos <- sapply(res.binned, \(\mathbf{x}\) acf(x, plot=FALSE)$acf[2])
(rho <- mean(rhos))
```

5 Our end goal

What we ultimately want to know about is:

1. Differences between the regions
2. Differences between non-/l/ and /l/ vowels
3. Differences between non-/l/ and /l/ vowels between the regions

For number 1, you should be all set after today's bit on inference. There's only one extra step you need to take into account: you can only `predict()` from completely-specified datasets, which means that if you've included random effects, it'll also ask you for a specific random-effect level to make predictions for. The solution is to simply give it a random level (e.g. the first in the data), and then to *zero out the columns in the linear-predictor matrix for the random effects*. This means you'll do something like:

```
newdata <- expand.grid(time=0:100, gender='m', region='NR', subject='NRmj1') |>
  as.data.frame()
lp <- predict(model, newdata=newdata, type='lpmatrix')
lp[, grep('subject', colnames(lp))] <- 0
```

You can suffice with specifically looking at regional differences between NR and the others: NR means ‘Netherlands-Randstad’, which is the area around Amsterdam. The regional differences we’re interested in are the result of ongoing sound changes, and they’re the most advanced in this region, which means you can use it as a good basis for comparison.

Should you finish all this and still have workshop time left over, feel free to think of ideas on how we could test the difference between /l/ and non-/l/ vowels!

References

- Voeten, Cesko Cis. 2020. *The adoption of sound change: synchronic and diachronic processing of regional variation in Dutch*. Leiden: Leiden University dissertation.
<https://doi.org/https://hdl.handle.net/1887/137723>.