Analyzing dynamic phonetic data using generalized additive mixed modeling

Wieling, Martijn

Published in:
Journal of Phonetics

DOI:
10.1016/j.wocn.2018.03.002

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2018

Link to publication in University of Groningen/UMCG research database

Citation for published version (APA):

Copyright
Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

Take-down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): http://www.rug.nl/research/portal. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.
Special Issue: Emerging Data Analysis in Phonetic Sciences, eds. Roettger, Winter & Baayen

Analyzing dynamic phonetic data using generalized additive mixed modeling: A tutorial focusing on articulatory differences between L1 and L2 speakers of English

Martijn Wieling
University of Groningen, Haskins Laboratories, The Netherlands

ARTICLE INFO
Article history:
Received 18 August 2017
Received in revised form 28 February 2018
Accepted 8 March 2018
Available online 21 June 2018

Keywords:
Generalized additive modeling
Electromagnetic articulography
Dynamic data
Tutorial

ABSTRACT
In phonetics, many datasets are encountered which deal with dynamic data collected over time. Examples include diphthongal formant trajectories and articulator trajectories observed using electromagnetic articulography. Traditional approaches for analyzing this type of data generally aggregate data over a certain timespan, or only include measurements at a fixed time point (e.g., formant measurements at the midpoint of a vowel). This paper discusses generalized additive modeling, a non-linear regression method which does not require aggregation or the pre-selection of a fixed time point. Instead, the method is able to identify general patterns over dynamically varying data, while simultaneously accounting for subject and item-related variability. An advantage of this approach is that patterns may be discovered which are hidden when data is aggregated or when a single time point is selected. A corresponding disadvantage is that these analyses are generally more time consuming and complex. This tutorial aims to overcome this disadvantage by providing a hands-on introduction to generalized additive modeling using articulatory trajectories from L1 and L2 speakers of English within the freely available R environment. All data and R code is made available to reproduce the analysis presented in this paper.

© 2018 Elsevier Ltd. All rights reserved.

1. Introduction
In phonetics, many types of data are collected, and frequently these types of data involve some kind of dynamic data collected over time. For example, in Volume 65 of Journal of Phonetics, seven out of nine papers focused on dynamic data. Most papers investigated vowel formant measurements in speech production (Hay, Podlubny, Drager, & McAuliffe, 2017; Hualde, Luchkina, & Eager, 2017; Hübscher, Borras-Comes, & Prieto, 2017; Ots, 2017; Rao, Sanghvi, Mixdorff, & Sabu, 2017; Yang & Fox, 2017). The authors of these papers either analyzed formant measurements at pre-selected time points (Hualde et al., 2017; Yang & Fox, 2017), average formant measurements (Hay et al., 2017; Hübscher et al., 2017), or simplified descriptions of formant contours (Ots, 2017; Rao et al., 2017). Another type of dynamic data, articulatory measurements (analyzed at the vowel midpoint), was analyzed by Pastätter and Pouplier (2017).

As the aforementioned studies illustrate, dynamic data is frequently simplified in one way or another before being analyzed. The advantage of simplification is clear. It not only reduces the data to a more manageable size, but it also allows the researcher to use well-known and well-established statistical approaches for analyzing the data, such as analysis of variance or linear mixed-effects regression modeling. But there is also a disadvantage associated with simplification: potentially interesting patterns in the dynamic data may be left undiscovered. For example, Van der Harst, Van de Velde, and Van Hout (2014) showed that analyzing dynamic formant trajectories revealed relevant (sociolinguistic) information, which was not apparent when analyzing a single time point.

When the full range of dynamic data is the subject of analysis, more sophisticated statistical techniques need to be employed, particularly those which are able to identify non-linear patterns. For example, one can use growth curve analysis (Mirman, 2014; Mirman, Dixon, & Magnuson, 2008; see Winter & Wieling, 2016 for a tutorial introduction) which requires the researcher to provide the specification of the non-linear pattern a priori. Another popular approach is to use (a variant of) functional data analysis (e.g., Gubian, Torreira, & Boves, 2015; Ramsay & Silverman, 2005) or sparse functional linear mixed modeling (Cederbaum, Pouplier, Hoole,
where functional principal components analysis can be used to characterize different types of non-linear patterns. In this paper, however, we will focus on generalized additive models (GAMs; Hastie & Tibshirani, 1986; Wood, 2006, 2017). In generalized additive modeling, the non-linear relationship between one or more predictors and the dependent variable is determined automatically as a function of the algorithm. While this type of analysis is not new, analyzing dynamic data in linguistics (potentially involving millions of data points) has been – until recently – computationally prohibitive. Nevertheless, various studies have recently been conducted which illustrate the potential of generalized additive modeling in linguistics and phonetics.

Meulman, Wieling, Sprenger, Stowe, and Schmid (2015) showed how to analyze EEG trajectories over time while simultaneously assessing the continuous influence of (second language learners’) age of acquisition in a dataset of over 1.6 million observations. Importantly, they compared their analysis using GAMs to a more traditional analysis of variance, and showed that the latter analysis was less sensitive and would have missed important results. Another example is provided by Nixon, van Rij, Mok, Baayan, and Chen (2016), who illustrated how visual world (i.e. eye tracking) data could suitably be analyzed with GAMs in a study on Cantonese tone perception. Finally, Wieling et al. (2016) used GAMs to compare articulatory trajectories between two groups of Dutch dialect speakers.

While the second edition of the book Generalized Additive Models: an introduction with R (Wood, 2017) provides an excellent discussion and introduction to GAMs, it assumes a reasonably high level of technical sophistication. The main aim of the present study is to illustrate and explain the use of generalized additive modeling in a more accessible way, such that it may be used by linguists to analyze their own (dynamic) data. In this tutorial, we will analyze a dataset of articulatory trajectories comparing native speakers of English to Dutch speakers of English as a second language (L2). We will systematically increase the sophistication of our analysis by starting from a simple generalized additive model and extending it step-by-step. While this procedure is not an approach someone would normally use (i.e. one would normally start with the model reflecting the hypothesis), we use this approach here to incrementally explain all necessary concepts with respect to generalized additive modeling.

There are already a few existing tutorials on GAMs. Sóskuthy (2017) provides an excellent tutorial introduction to GAMs, where he shows how to analyze formant trajectories over time using real-world data from Stuart-Smith et al. (2015). In addition, Winter and Wieling (2016) take a hands-on approach to discuss various statistical approaches, including mixed-effects regression, growth curve analysis and generalized additive modeling, to model linguistic change. The present paper differs from Winter and Wieling (2016) by not providing a comparison between different analysis approaches, but instead providing a more comprehensive overview of generalized additive modeling (e.g., including non-linear interactions, model criticism, etc.). Compared to Sóskuthy (2017), the present paper provides less detail about GAM theory, but places more emphasis on evaluating whether model assumptions are satisfied. In addition, Sóskuthy provides an analysis of an acoustic dataset of about 5000 observations, whereas the present paper shows how to apply GAMs to a much larger (articulatory) dataset containing over 100,000 observations. Finally, this tutorial also illustrates how to fit a non-Gaussian GAM, which neither of the two other tutorials do.

In the following two sections, we will discuss the research question and the data collection procedure. In Sections 4 and 5, we will illustrate and explain the details of the model specification (in the statistical software package R; R Core Team, 2017), and also explain important concepts necessary to understand the analysis. Finally, Sections 6 and 7 provide a discussion of the advantages and disadvantages of generalized additive modeling and a conclusion.

2. Research project description and research question

In this research project, our goal was to compare the pronunciation of native English speakers to non-native (Dutch) speakers of English. Speech learning models, such as Flege’s Speech Learning Model (SLM; Flege, 1995) or Best’s Perceptual Assimilation Model (PAM; Best, 1995), explain L2 pronunciation difficulties by considering the phonetic similarity of the speaker’s L1 and L2. Sound segments in the L2 that are very similar to those in the L1 (and map to the same category) are predicted to be harder to learn than those which are not (as these map to a new sound category). In this tutorial we focus on data collected for Dutch L2 speakers of English when they pronounce the sound /l/ (which does not occur in the native Dutch consonant inventory, but is very similar to the Dutch sounds /l/ or /l/), and compare their pronunciations to those of native Standard Southern British English speakers. Based on earlier acoustic analyses of different data (Hanulíková & Weber, 2012; Westers, Gilbers, & Lowie, 2007), Dutch speakers were shown to frequently substitute /l/ with /l/. This finding is in line with predictions of the SLM and PAM, and is used to guide our hypothesis.

Instead of focusing on perceptual or acoustic differences, here we will focus on the underlying articulatory trajectories. There are only a small number of studies which have investigated L2 differences in pronunciation from an articulatory perspective. One of the few studies was conducted by Nissen, Dromey, and Wheeler (2007) who investigated differences between the L2 English pronunciation of native Korean and...
native Spanish speakers. However, in contrast to our study, they did not include a native speaker group.

In the present study, we will investigate the movement of the tongue tip during the pronunciation of words (minimal pairs) containing either /t/ or /θ/. Consequently, the research question of our study is as follows:

Do Dutch non-native speakers of English differ from native English speakers contrasting the dental fricative /t/ from the alveolar plosive /θ/ in articulation?

Our associated null hypothesis is that the two groups will show the same contrast between /t/ and /θ/, and the alternative hypothesis – on the basis of the SLM and PAM – is that the Dutch speakers will show a smaller contrast between the two sounds, as they will more often merge the two sounds.

3. Data collection procedure

The Dutch L2 data was collected at the University of Groningen (20 university students), and the English L1 data was collected at the University College London (22 university students). Before conducting the experiment, ethical approval was obtained at the respective universities. Before the experiment, participants were informed about the nature and goal of the experiment and signed an informed consent form. Participants were reimbursed either via course credit (Groningen) or payment (London) for their participation, which generally took about 90 min.

We collected data for 10 minimal pairs of English words for all speakers (i.e., ‘tent’-‘tenths’, ‘fate’-‘faith’, ‘fort’-‘forth’, ‘kit’-‘kith’, ‘mitt’-‘myth’, ‘tank’-‘thank’, ‘team’-‘theme’, ‘tick’-‘thick’, ‘ties’-‘thighs’, and ‘tongs’-‘thongs’). Each word was pronounced individually, but preceded and succeeded by the pronunciation of /a/ in order to ensure a neutral articulatory context. In order to achieve this, the participants were shown stimuli consisting of a single word surrounded by two schwas (e.g., “a thank a”). The order of the words was randomized and every word was pronounced twice during the course of the experiment. While the speakers were pronouncing these words, we tracked the movement of sensors placed on their tongue and lips using a 16-channel Wave electromagnetic articulography (EMA) device (Northern Digital Inc.) at a sampling rate of 100 Hz. Sensors were glued to the tongue and lips with PeriAcryl 90HV dental glue. Concurrently recorded acoustic data (collected using an Audio-Technica AT875R microphone) was automatically synchronized with the articulatory data. In post-processing, articulatory data were corrected for head movement using four reference sensors (left and right mastoid processes, forehead, upper incisor), and aligned to each speaker’s occlusal plane based on a biteplane trial (see Wieling et al., 2016).

In this tutorial, we only focus on the anterior-posterior position of the T1 sensor (positioned about 0.5-1 cm behind the tongue tip), as articulatory differences between /t/ and /θ/ should be most clearly apparent on this trajectory and dimension. The individual words were subsequently segmented on the basis of the articulatory gestures (i.e. from the gestural onset of the initial sound to the gestural offset of the final sound; using mvow; Tiede, 2005) and time-normalized between 0 (gestural start of the word) to 1 (gestural end of the word). Furthermore, the T1 sensor positions were normalized for each speaker by z-transforming the positions per speaker (i.e. subtracting the mean and dividing by the standard deviation; the mean and standard deviation per speaker were obtained on the basis of about 250 utterances elicited in the context of the broader experiment in which the present data was collected). Higher values signify more anterior positions, whereas lower values indicate more posterior positions. As generalized additive modeling essentially smooths the data, filtering is not necessary. In fact, it is even beneficial to analyze raw instead of filtered data, as this will result in less autocorrelation in the residuals (i.e. the difference between the fitted values and the actual values; see Section 4.8 for an explanation). Consequently, we analyze the raw, unfiltered data in this paper.

Note that due to the fixed sampling rate (of 100 Hz) the number of sampling points per word is dependent on the word’s length. Our present dataset consists of 126,177 measurement points collected across 1618 trials (62 trials were missing due to sensor failure or synchronization issues). The average duration of each word (from the articulatory start to the articulatory end) is therefore about 0.78 seconds, yielding on average 78 measurement points per word production.


A generalized additive model can be seen as a regression model which is able to model non-linear patterns. Rather than explaining the basic concepts underlying generalized additive modeling at the start, in this tutorial we will explain the concepts when we first need them in the analysis. Importantly, this tutorial will not focus on the underlying mathematics, but rather take a more hands-on approach. For a more mathematical background, we refer the reader to the excellent, recently revised book of Simon Wood on generalized additive modeling (Wood, 2017).

To create a generalized additive model, we will use the mgcv package in R (version 1.8–23; Wood, 2011, 2017). Furthermore, for convenient plotting functions, we will use the itsadug R package (version 2.3.0; van Rij, Wieling, Baayen, & van Rijn, 2017). Both can be loaded via the library command (e.g., library(mgcv)). (Note that R commands as well as the output will be explicitly marked by using a monospace font.)

Instead of starting immediately with a suitable model for our data, we will start with a simple model and make it gradually more complex, eventually arriving at the model appropriate for our data. Particularly, we will first discuss models which do not include any random effects, even though this is clearly inappropriate (given that speakers pronounce multiple words). Consequently, please keep in mind that the p-values and confidence bands will be overconfident for these first few models (e.g., Judd, Westfall, & Kenny, 2012).

Of course, over time the function calls or function parameters may become outdated, while this tutorial text, once published, will remain fixed. Therefore, we will endeavor to keep the associated paper package up-to-date. The paper package is available at the author’s personal website, http://www.martijnwieling.nl, and includes all data, code, and output (direct link: http://www.let.rug.nl/wieling/Tutorial).
4.1. The dataset

Our dataset, `dat`, has the following structure (only the first six out of 126,117 lines are shown using the command `head(dat)`):

```
  Speaker Lang Word  Loc  Trial  Time  Pos
1  VENI_RN_1 EE tick T Init 0.0000  -0.392
2  VENI_RN_1 EE tick T Init 0.0161  -0.440
3  VENI_RN_1 EE tick T Init 0.0523  -0.440
4  VENI_RN_1 EE tick T Init 0.0484  -0.503
5  VENI_RN_1 EE tick T Init 0.0645  -0.513
6  VENI_RN_1 EE tick T Init 0.0806  -0.677
```

The first column (i.e. variable), `Speaker`, shows the speaker ID, whereas the second column, `Lang`, shows the native language of the speaker (EN for native English speakers, or NL for native Dutch speakers). The third column, `Word`, shows the item label. Column four, `Loc`, contains either T or TH for minimal pairs involving the /t/ or the /θ/, respectively. Column five, `Trial`, contains either the value Init or the value Final, indicating where in the word the sound /t/ or /θ/ occurs (e.g., for the words ‘tent’ and ‘tenth’ this is word-final). The sixth column, `Time`, contains the trial number during which the word was pronounced by the speaker. The final two columns, `Pos` and `Time`, contain the normalized time point (between 0 and 1) and the associated (standardized) anterior position of the T1 sensor.

4.2. A first (linear) model

For simplicity, we will illustrate the generalized additive modeling approach by focusing only on the minimal pair ‘tent’-‘tenth’. We will use this example to illustrate all necessary concepts, but we will later extend our analysis to all words in Section 5.

The first model we construct is:

```
ml <- bam(Pos ~ Word, data=dat, method="fREML")
```

This model simply estimates the average (constant) anterior position difference (of the T1 sensor) between the two words (‘tent’ and ‘tenth’), and is shown to illustrate the general model specification. We use the function `bam` to fit a generalized additive model. (The alternative function `gam` becomes prohibitively slow for complex models which are fit to datasets exceeding 10,000 data points.) The first parameter of the function is the formula reflecting the model specification, in this case: `Pos ~ Word`. The first variable of the formula, `Pos`, is the dependent variable (the anterior position of the T1 sensor). The dependent variable is followed by the tilde (~), after which one or more independent variables are added. In this case, the inclusion of a single predictor, `Word`, allows the model to estimate a constant difference between its two levels (‘tenth’ versus ‘tent’; the latter word has been set as the reference level of the predictor). The parameter `data` is set to the name of the data frame variable in which the values of the dependent and independent variables are stored (in this case: `dat`). The third parameter (method) specifies the smoothing parameter estimation method, which is currently set to the default of "fREML", fast restricted maximum likelihood estimation. This is one of the fastest fitting methods, but it is important to keep in mind that models fit with (f)REML cannot be compared when the models differ in their fixed effects (i.e. the predictors in which we are generally interested; see Section 4.7 for more details). In that case, method should be set to "ML" (maximum likelihood estimation), which is much slower. To obtain a summary of the model we can use the following command in R:

```
(smry1 <- summary(ml))
```

Note that it is generally good practice to store the summary in a variable, since the summary of a complex model might take a while to compute. The summary (which is printed since the full command is put between parentheses) shows the following:

```
Family: gaussian
Link function: identity

Formula:
  Pos ~ Word

Parametric coefficients:
  Estimate  Std. t value  Pr(>|t|)
(Intercept)   0.0654   0.0117   5.57  2.5e-08 ***
Wordtenth    0.6642   0.0164  40.41  < 2e-16 ***
```

R-sq.(adj) = 0.113  Deviance explained = 11.3%
-REML = 17307  Scale est. = 0.86694  n = 12839

The top lines show that we use a Gaussian model with an identity link function (i.e. we use the original, non-transformed, dependent variable), together with the model formula. The next block shows the parametric coefficients. As usual in regression, the intercept is the value of the dependent variable when all numerical predictors are equal to 0 and nominal variables are at their reference level. Since the reference level for the nominal variable `Word` is ‘tent’, this means the average anterior position of the T1 sensor for the word ‘tent’ for all speakers is about 0.07. The line associated with `Wordtenth` (the non-reference level, i.e. tenth, is appended to the variable name) indicates that the anterior position of the T1 sensor for the word ‘tenth’ is about 0.66 higher (more anterior) than for the word ‘tent’, and that this difference is significant with a very small p-value (at least, according to this analysis, which does not yet take the random-effects structure into account).

The final two lines of the summary show the goodness-of-fit statistics. The adjusted $R^2$ represents the amount of variance explained by the regression (corrected to use unbiased estimators; see Wood, 2006, p. 29). The deviance explained is a generalization of $R^2$ and will be very similar to the actual $R^2$ value for Gaussian models (Wood, 2006, p. 84). The REML (restricted maximum likelihood) value by itself is not informative. The value is only meaningful when two models are compared which are fit to the same data, but only differ in their random effects. In that case lower values are associated with a model which is a better fit to the data. The minus sign (–REML) is added as the REML value is mostly negative. (Note that for later models, i.e. those including non-linear patterns, the –REML label is replaced by fREML.) The scale (parameter) estimate represents the variance of the residuals. Finally, the number of data points which are included in the model are shown (in this case: 12,839).
4.3. Modeling non-linear patterns

Of course, we are not only interested in a constant T1 anterior position difference between the two words, but also in the anterior position of the T1 sensor over time. A generalized additive model allows us to assess if there are non-linear patterns in our data by using so-called smooths. These smooths model non-linear patterns by combining a pre-specified number of basis functions. For example, a cubic regression spline smooth constructs a non-linear pattern by joining several cubic polynomials (see also Sóska andy, 2017). The default type of smooth, which we will use in this tutorial, is the thin plate regression spline. The thin plate regression spline is a computationally efficient approximation of the optimal thin plate spline (Wood, 2003). The thin plate regression spline models a non-linear pattern by combining increasingly complex non-linear basis functions (see Fig. 1). Each basis function is first multiplied by a coefficient (i.e. the magnitude of the contribution of that basis function) and then all resulting patterns are summed to yield the final (potentially) non-linear pattern. Note that the first basis function is not incorporated in the actual smooth, but is included in the model's intercept. While modeling non-linear patterns may seem to be an approach which is bound to lead to overfitting, GAMs apply a penalization to non-linearity (i.e. ‘wiggliness’) to prevent this. Rather than minimizing the error only (i.e. the difference between the fitted values and the actual values), GAMs minimize a combination of the error and a non-linearity penalty, thereby preventing overfitting and minimizing prediction error. Consequently, a generalized additive model will only identify a non-linear effect if there is substantial support for such a pattern in the data, but will instead detect a linear effect if there is only support for a linear pattern. With respect to the thin plate regression spline basis functions visualized in Fig. 1, especially the more complex non-linear patterns will generally be more heavily penalized (i.e. have coefficients closer to zero).

To extend m1 by including a non-linear pattern over time for both groups separately, the following generalized additive model can be specified (we exclude the method parameter as it is set to the default value of "fREML"):  
m2 <- bam(Pos ~ Word + s(Time, by=Word, bs="tp", k=10), data=dat)

The text in boldface shows the additional term compared to model m1. The function s sets up a smooth over the first parameter (Time), separately for each level of the nominal variable indicated by the by-parameter (i.e. Word). The bs-parameter specifies the type of smooth, and in this case is set to "tp", the default thin plate regression spline (a cubic regression spline can be fit instead by setting bs to the value "cr"). The k-parameter, finally, sets the size of the basis dimension. In the example above, by setting k to 10 (the default value), there are at most 9 \((k-1)\) basis functions used in each smooth (see Fig. 1). Since the smooth type and the basis dimension are both set to their default, a simpler specification of the smooth is \(s(Time, by=Word)\). If the by-parameter were left out, the model would fit only a single non-linear pattern, and not a separate pattern per word.

The summary of model m2 shows the following (starting from the parametric coefficients):

```
Parametric coefficients:
  Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.0655  0.0107  6.14 8.3e-10 ***
Wordtenth  0.6624  0.0149 44.34 <2e-16 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Approximate significance of smooth terms:
 edf   Ref.df  F p-value
s(Time):Wordtenth  9.00  8.55  1.00 0.49
s(Time):Wordtent  9.00  7.52  1.00 0.47
s(Time):Wordtenth  9.00  8.55  1.00 0.49
```

In addition to the parametric coefficients, now an additional block is added consisting of the approximate significance of smooth terms. Here two lines are visible, \(s(Time):Wordtenth\), representing the smooth for the Word ‘tenth’ and \(s(Time):Wordtenth\), reflecting the smooth for the Word ‘tenth’. The \(p\)-value associated with each smooth indicates if the smooth is significantly different from 0 (which both are in this, still sub-optimal, analysis). The Ref.df value is the reference number of degrees of freedom used for hypothesis testing (on the basis of the associated \(F\)-value). The edf value reflects the number of effective degrees of freedom, which can be seen as an estimate of how many parameters are needed to represent the smooth. (Due to penalization, both edf and Ref.df are almost always non-integer.) The edf value is indicative of the amount of non-linearity of the smooth. If the edf value for a certain smooth is (close to) 1, this means that the pattern is (close to) linear (i.e. cf. the second basis function in Fig. 1). A value greater than 1 indicates that the pattern is more complex (i.e. non-linear). The edf value is limited by k minus one (as the intercept is part of the parametric coefficients). Due to penalization, the edf value will generally be lower than its maximum value. If the edf value is close to its maximum (which is the case for m2, particularly for the ‘tenth’ smooth), then this suggests that a higher basis dimension might be necessary to prevent oversmoothing (i.e. oversimplifying the non-linear pattern). To more formally assess this, we can use the function gam.check with as input model m2: gam.check(m2). The output of this call is:

```
Method: fREML Optimizer: perf newton full convergence after 9 iterations.
Gradient range [-4.6e-07,3.86e-07] (score 16112 & scale 0.716).
Hessian positive definite, eigenvalue range [2.95,6418].
Model rank = 20 / 20
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):Wordtenth  9.00  7.52  1  0.47
s(Time):Wordtenth  9.00  8.55  1  0.49
```
The first lines show that the model converged on a solution. The bottom lines are associated with the smooths. It shows the edf values together with k’ (i.e. k − 1). If the value of k-index is lower than 1 and the associated p-value is low, this suggests that the basis dimension has been restricted too much. In that case, it is good practice to refit the model with the value of k doubled. In this case, there is no reason to do so, as the value of k-index is not smaller than 1 and the p-value is relatively high.

In principle, the k-parameter can be set as high as the number of unique values in the data, as penalization will result in the appropriate shape. However, allowing for more complexity negatively impacts computation time.

4.4. Visualizing GAMs

While it is possible to summarize a linear pattern in only a single line, this is obviously not possible for a non-linear pattern. Correspondingly, visualization is essential to interpret the non-linear patterns. The command: plot(m2) yields the visualizations shown in Fig. 2 (abline(h=0) was used to add the horizontal line for the x-axis in both visualizations).

It is important to realize that this plotting function only visualizes the two non-linear patterns without taking into account anything else in the model. This means that only the partial effects are visualized. It is also good to keep in mind that the smooths themselves are centered (i.e. move around the x-axis, y = 0). Visualizing the smooths in this way, i.e. as a partial effect, is insightful to identify the non-linear patterns, but it does not give any information about the relative height of the smooths. For this we need to take into account the full model (i.e. the fitted values). Particularly, the intercept and the constant difference between the two smooths shown in the parametric part of the model need to be taken into account. For this type of visualization, we use the function plot_smooth from the itsadug package as follows:

plot_smooth(m2, view="Time", plot_all="Word", rug=FALSE)

The first parameter is the name of the stored model. The parameter view is set to the name of the variable visualized on the x-axis. The parameter plot_all should be set to the name of the nominal variable if smooths need to be displayed for all levels of this variable. This is generally equal to the name of the variable set using the by-parameter in the smooth specification. If the parameter is excluded, it only shows a graph for a single level (a notification will report which level is shown in case there are multiple levels). The final parameter rug is used to show or suppress small vertical lines on the x-axis for all individual data points. Since there are many unique values, we suppress these vertical lines here by setting the value of the parameter to FALSE. Fig. 3 shows the result of this call and visualizes both patterns in a single graph. It is clear that the smooths are not centered (i.e. they represent full effects, rather than partial effects), and that the ‘tenth’-curve lies above the ‘tent’-curve, reflecting that the /l/ is pronounced with a more anterior T1 position than the /n/. The shapes of the curves are, as would be expected, identical to the partial effects shown in Fig. 2.

To visualize the difference, we can use the itsadug function plot_diff as follows:

plot_diff(m2, view="Time", comp=list(Word=c("tenth","tent")))

The parameters are similar to those of the plot_smooth function, with the addition of the comp parameter which requires a list of one or more variables together with two levels which should be compared. In this case, the first word (i.e. ‘tenth’) is contrasted with the second word (i.e. ‘tent’) in the plot. Fig. 4 shows this difference.

4.5. Is the additional complexity necessary?

While it may be obvious from Figs. 3 and 4 that the two patterns need to be distinguished, it is necessary to assess this formally (i.e. using statistics). There are three approaches for this, each with its own merits.
4.5.1. Model comparison

The first approach is to fit two models, one model without the distinction and one with the distinction, and compare the two models, for example using the Akaike Information Criterion (AIC; Akaike, 1974) measuring the goodness of fit of the two models while taking into account the complexity of the models. In this paper we use a minimum reduction threshold of 2 AIC units to select a more complex model (cf. Wieling, Montemagni, Nerbonne, & Baayen, 2014). The itsadug function compareML can be used to compare (the AIC of) two models. As mentioned before, models differing in their fixed effects can only be compared when fit with the maximum likelihood (ML) estimation method. Consequently, we refit m2 using ML (naming this model m2b.ml) and we fit a simpler model (m2a.ml) which includes the constant difference between the two words, but only a single smooth. As such, model m2a.ml assumes that the pattern over time is the same for both words. Both models include Word as a predictor, as it was found to be highly significant in m1.

\[
m2a.ml \leftarrow \text{bam}(\text{Pos} \sim \text{Word} + s(\text{Time}), \text{data}=\text{dat}, \text{method}="\text{ML}"
\]
\[
m2b.ml \leftarrow \text{bam}(\text{Pos} \sim \text{Word} + s(\text{Time}, \text{by}=\text{Word}), \text{data}=\text{dat}, \text{method}="\text{ML}"
\]

**Fig. 2.** Visualization of the non-linear smooths (partial effects) for the word ‘tent’ (left) and the word ‘tenth’ (right) of model m2. The pointwise 95%-confidence intervals are shown by the dashed lines. Note that the range of the y-axis, showing the anterior position of the T1 sensor, has been set to [-1,2] to be comparable with the other plots in this paper.

**Fig. 3.** Non-linear smooths (fitted values) for the word ‘tent’ (blue, dark) and the word ‘tenth’ (red, light) of model m2. The pointwise 95%-confidence intervals are shown by shaded bands. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

**Fig. 4.** Difference between the two (non-linear) smooths comparing the word ‘tenth’ to the word ‘tent’ of model m2. The pointwise 95%-confidence interval is shown by a shaded band. When the shaded confidence band does not overlap with the x-axis (i.e. the value is significantly different from zero), this is indicated by a red line on the x-axis (and vertical dotted lines). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)
Note that the k-parameter and the bs-parameter were not explicitly specified, as these parameters were set to their default values. We can now compare the two models using:

```r
compareML(m2a.ml,m2b.ml)
```

This results in the following output:

```r
m2a.ml: Pos ~ Word + s(Time)
m2b.ml: Pos ~ Word + s(Time, by = Word)
```

Chi-square test of ML scores

<table>
<thead>
<tr>
<th></th>
<th>Score</th>
<th>edf</th>
<th>Difference</th>
<th>Df</th>
<th>p.value</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>m2a.ml</td>
<td>16505</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>m2b.ml</td>
<td>16103</td>
<td>6</td>
<td>2e-16 ***</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AIC difference: 823.83, model m2b.ml has lower AIC.

These results show that model m2b.ml is preferred as both its AIC score is much lower and the ML score is significantly lower when taking the number of parameters into account. Note that in the model comparison procedure, each smooth counts as two degrees of freedom (a random and a fixed part), and not the difference in number of effective degrees of freedom shown in the model summary.

While the model comparison approach is straightforward, it has one clear drawback. To compare models differing in their fixed effects, the models need to be fit with maximum likelihood estimation. This method is substantially slower than edf, as these parameters were set to their default values. We can now compare the two models using:

```r
compareML(m2a.ml,m2b.ml)
```

This results in the following output:

```r
m2a.ml: Pos ~ Word + s(Time)
m2b.ml: Pos ~ Word + s(Time, by = Word)
```

Chi-square test of ML scores

<table>
<thead>
<tr>
<th></th>
<th>Score</th>
<th>edf</th>
<th>Difference</th>
<th>Df</th>
<th>p.value</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>m2a.ml</td>
<td>16505</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>m2b.ml</td>
<td>16103</td>
<td>6</td>
<td>2e-16 ***</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AIC difference: 823.83, model m2b.ml has lower AIC.

The summary of this model shows the following:

| Parametric coefficients: | Estimate | Std. Error | t value | Pr(>|t|) |
|--------------------------|----------|------------|---------|---------|
| (Intercept)              | 0.0654   | 0.0107     | 6.14    | 8.8e-10 *** |
|                          |          |            |         |         |
| Signif. codes:           |          |            |         |         |
| 0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’’ 1 |

Approximate significance of smooth terms:

<table>
<thead>
<tr>
<th>edf</th>
<th>Ref.df</th>
<th>F</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m2b.ml</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16103</td>
<td>7.69</td>
<td>28.8</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>9.01</td>
<td>293.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

R-sq.(adj) = 0.267 Deviance explained = 26.8%

fREML = 16111 Scale est. = 0.71584 n = 12839

The model specification is now quite different. The first part, s(Time), indicates the pattern over time which is included irrespective of the value of IsTenth (i.e. irrespective of the word). The second part s(Time, by=IsTenth) has a special interpretation due to IsTenth being a binary variable. In this case, the smooth is equal to 0 whenever the binary variable equals 0. If the binary by-variable equals 1, it models a (potentially) non-linear pattern without a centering constraint. In contrast to a normal centered smooth (e.g., see Fig. 2), these so-called binary smooths also model the constant difference between the two levels. This is also the reason that the predictor IsTenth (or Word) should not be included as a fixed-effect factor.

The interpretation of this model is now as follows. When IsTenth equals 0 (i.e. for the word ‘tent’), the position of the sensor is modeled by s(Time) + 0. This means that the first s(Time) represents the smooth for the word ‘tent’ (the reference level). When IsTenth equals 1 (i.e. for the word ‘tenth’), the position of the sensor is modeled by s(Time) + s(Time, by=IsTenth). Given that s(Time) models the pattern for the word ‘tent’, and both smooths together model the pattern for the word ‘tenth’, it logically follows that s(Time, by=IsTenth) models the difference between the non-linear patterns of ‘tenth’ and ‘tent’.

That is indeed the case, but can be seen by visualizing the binary difference smooth (i.e. the partial effect) directly via plot(m2.bin, select=2, shade=TRUE). Note that the parameter select determines which smooth to visualize (in this case, the second smooth in the model summary, s(Time):IsTenth), whereas the parameter shade is used to denote whether the confidence interval needs to be shaded (i.e. when set to TRUE), or whether dashed lines should be used (i.e. when set to FALSE, the default). The graphical result of this command is shown in Fig. 5, and this graph nicely matches Fig. 4. It is also clear that the partial effect includes
the intercept difference, given that the smooth is not centered. Importantly, the model summary shows that the non-linear pattern for the difference between the two words is highly significant, thereby alleviating the need for model comparison. (But note that we still have ignored the required random-effects structure here.)

Of course, the disadvantage of this approach is that the difference smooth simultaneously includes the non-linear as well as the intercept difference between the two levels, and it may be desirable to separate these. Particularly, we might be interested in assessing if the difference between the two words is significant due to a constant difference, a non-linear difference, or a combination of the two. It is also important to keep in mind that each distinct binary predictor (e.g., IsTenth) may only occur exactly once in the model specification. Otherwise, the model is not able to determine which of the binary difference smooths will include the constant difference between the two words. For more details, see Section 5.4.2.1 in the supplementary material.

4.5.3. Refitting the model with an ordered factor difference smooth

Fortunately, separating the intercept difference and the non-linear difference is possible as well. In that case, one can use an ordered factor predictor instead of the binary (dummy) predictor. The ordered factor can be created as follows (the ‘O’ is appended here to the original variable name to indicate mnemonically that it is an ordered factor):

```r
dat$WordO <- as.ordered(dat$Word)
contrasts(dat$WordO) <- "contr.treatment"
```

It is essential to set the contrasts of the ordered factor to contrast treatment. This ensures that the contrasts of the ordered factor are identical to using a binary predictor (i.e. contrasting other levels to a reference level, whose value is set to 0). The model can now be fit as follows:

```r
m2.ord <- bam(Pos ~ WordO + s(Time) + s(Time, by=WordO), data=dat)
```

This model is essentially identical to model m2.bin (i.e. the iREML score and the predictions of the two models are the same). Comparing the two summaries, it is clear that model m2.ord has an additional parametric coefficient (similar to the constant difference shown in model m2) which models the constant difference between the word ‘tenth’ and ‘tent’. Comparing the effective degrees of freedom of the final (difference) smooth in both models shows that they almost exactly differ by 1 (m2.ord: 8.02, m2.bin: 9.01). This reflects the intercept difference, which is included in the final non-centered smooth in the binary smooth model, but by a separate parametric coefficient in the ordered factor difference smooth model. Visualizing the difference smooth of model m2.ord in Fig. 6 indeed reveals that the pattern is identical to the pattern shown in Fig. 5. The only exception is that it is centered in Fig. 6. In principle, the width of the confidence bands will also differ, as the binary smooth incorporates the uncertainty about the intercept difference. In this case, however, the intercept difference has a very low standard error (see the estimate of WordOtenth in the summary of m2.ord), and this difference is therefore visually unistinguishable.

The advantage of the ordered factor approach over the binary approach is that the constant difference (shown in the parametric coefficients part of the model) and the non-linear difference can be distinguished when using an ordered factor. For both a $p$-value is shown which can be used to assess if the difference between two patterns is caused by a non-linear difference over time, a constant difference, or both. In this case both are highly significant, but there are situations in which there might be much certainty about the non-linear difference, but less certainty about the intercept difference. In that case, the use of a binary difference smooth would show a non-linear pattern with a very wide confidence interval, which might lead one to incorrectly conclude that there is insufficient support for a non-linear pattern.
4.6. Model criticism

We have already shown part of the output of the function `gam.check` in Section 4.3. Besides checking if the basis dimension for the smooths is sufficient, this function also provides important diagnostic information about the model. In particular, the function also results in a series of four graphs, shown in Fig. 7.

The top-left graph shows a normal quantile plot of the (deviance) residuals of the model. If the residuals are approximately normally distributed, they should approximately follow the straight line. Correspondingly, the histogram of the residuals is shown in the bottom-left graph. For model `m2` the residuals are approximately normally distributed, thereby satisfying one of the (Gaussian) model assumptions. The underlying idea of requiring a normal distribution of the residuals, is that the part which is left unexplained by the model (i.e. the residuals) are assumed to represent random noise and therefore should follow a normal distribution. The remaining two plots can be used to assess heteroscedasticity (i.e. unequal variance depending on the values of the predictors in the top-right graph, or the fitted values in the bottom-right graph). Substantial differences in the variability over the range of the values of the predictors and fitted values point to problems in the model fitting (as homogeneity of variances is one of the leading assumptions of the model), and affect the standard errors of the model. In this case, there seems to be only minor heteroscedasticity present, which is unlikely to be a problem. An example of clear heteroscedasticity would be revealed by a distinct pattern in the residuals, such as a ‘V’-like shape where increasing variability is associated with increasing values of the predictor. If there is much heteroscedasticity, including additional predictors or transforming the dependent variable may help (see also Baayen, 2008: Section 7.9). In addition, the function `gam` (but, presently, not `bam`) includes the family `"gauss"`, which is able to model unequal variance in the context of a Gaussian model (see also Wood, 2017: Section 7.9). Note that both scatter plots also nicely illustrate the dependencies within trajectories (i.e. the spaghetti-like patterns), especially at the top and bottom of the graphs. These dependencies will also need to be taken into account (see Section 4.8).

One essential point, which we have been ignoring up until now, is that in our present model every individual data point is treated as being independent. This is, of course, completely incorrect, given that each participant provides multiple productions. In addition, as we are dealing with time series data, sequential points in time will also not be independent. When incorrectly treating all data points as being independent, the net effect is that $p$-values will be too low and confidence bands will be too thin (e.g., Judd et al., 2012). For an appropriate analysis, we need to take these dependencies into account.

4.7. Mixed-effects regression within the GAM framework

By using mixed-effects regression we are able to take the structural variability in our data into account, and thereby obtain reliable and generalizable results (i.e. results not specific to our sample). In mixed-effects regression a distinction is made between fixed-effect factors and random-effect factors. Fixed-effect factors are nominal (i.e. factor) variables with a small number of levels, out of which all (or most) levels are included in the data. For example, both native and non-native speakers are present in our data. In addition, numerical predictors are always part of the fixed-effects specification of the model. In a regular linear (non-mixed-effects) regression model, the fixed effects are all predictors which are included in the model. Random-effect factors are those factors which introduce systematic variation, generally have a large number of levels, and which the researcher would like to generalize over. In many studies in linguistics, the random-effect factors include participant and word, as the levels of these factors are sampled from a much larger population (i.e. other participants and other words could have been included). Note that for the present small dataset the predictor `Word` is a fixed-effect factor, given that we are currently only interested in the difference between the two words 'tenth' and 'ten'.

With respect to random-effect factors, it is important to distinguish random intercepts and random slopes. Some speakers (or words) will on average have a more anterior tongue position than others, and this structural variability is captured by a by-speaker (or by-word) random intercept. Failing to take this variability into account generally results in overconfident (i.e. too low) $p$-values (Baayen, Davidson, & Bates, 2008; Judd et al., 2012). Random slopes allow the influence of a predictor to vary for each level of the random-effect factor. For example, the exact difference between the word ‘tenth’ and ‘ten’ may vary per speaker. It is essential to assess which random intercepts and slopes need to be included, as failing to include a necessary random slope may yield $p$-values which are overconfident (Gurka, Edwards, & Muller, 2011). For example, suppose that ninety percent of the speakers shows a negligible difference between ‘tenth’ and ‘ten’, and the remaining ten percent shows a substantial difference, the average difference might be just above the threshold for significance. However, it is clear that in the above situation this difference should not reach significance (given that the majority of speakers do not show the effect). Including a by-speaker random slope for the word contrast would account for this individual variability and result in a more appropriate (higher) $p$-value.
Of course, if there is almost no individual variability, model comparison will reveal that the random slope is unnecessary. For more information about the merits about mixed-effects regression, we refer the interested reader to Baayen et al. (2008), Baayen (2008), Winter (2013), and Winter and Wieling (2016).

We would like to remark that even though the paper of Barr, Levy, Scheepers, and Tilly (2013) was important in that it made researchers aware that a random-effects structure only consisting of random intercepts is often problematic, we are not in favor of an approach in which the maximally possible random-effects structure is used (Barr et al., 2013). Instead, we are proponents of using model selection (e.g., used by Wieling, Nerbonne, & Baayen, 2011; Wieling et al., 2014) to determine the optimal random-effects structure appropriate for the data. The advantage of such an approach is that it does not result in a lack of power (as the maximal approach does; Matuschek, Kliegl, Vasishth, Baayen, & Bates, 2017) and is more suitable to be used in conjunction with generalized additive modeling (Baayen, Vasishth, Kliegl, & Bates, 2017).

Within the generalized additive modeling framework, random intercepts, random slopes and non-linear random effects can be included. In the following, we will see how to construct these generalized additive (mixed) models.

4.7.1. Including a random intercept

To add a random intercept per speaker to a GAM, the following model specification can be used (the difference, with respect to m2, i.e. the random intercept, is again marked in boldface):

\[ m3 <- \text{bam}(\text{Pos} \sim \text{Word} + s(\text{Time}, by=\text{Word}) + s(\text{Speaker}, bs=^{\text{re}}), \text{data=dat}) \]
As random effects and smooths are linked (see Wood, 2017), random intercepts and slopes may be modeled by smooths. For these random-effect smooths the basis needs to be set to the value "re". The first parameter of the random-effect smooth is the random-effect factor. If there is a second parameter (besides the obligatory bs="re" part), this is interpreted as a random slope for the random-effect factor. If there is only a single parameter (as in m3, above), it is interpreted to be a random intercept. As readers are likely more familiar with the lme4 (Bates, Maechler, Bolker, & Walker, 2014) function lmer to specify random effects, the analogue of s(Speaker,bs="re") would be (1|Speaker) in lmer. The summary of m3 shows the following:

```
Parametric coefficients:
   Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.0919    0.0680   1.35    0.18
Wordtenth    0.6799    0.0134  50.91  < 2e-16 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
   edf Ref.df F p-value
s(Time):  Wordtenth:  7.77    8.61  36.3 < 2e-16 ***
          Wordtenth:  8.64    8.96  352.7 < 2e-16 ***
          s(Speaker): 40.58   41.00  86.9 < 2e-16 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) = 0.427  Deviance explained = 42.9%
---

m4 <- bam(Pos + s(Time) + s(Speaker) + s(Word,bs="re"), data=dat)
```

One additional line, s(Speaker), has been added to the list of smooth terms. The Ref.df value shows the number of speakers minus one. Due to the penalization (i.e. effectively representing shrinkage in the case of mixed-effects regression; see Baayen et al., 2008) the estimated degrees of freedom will generally be somewhat lower than the value of Ref.df. Importantly, however, the p-value associated with the random-effect smooth conveniently indicates if the random intercept is necessary or not (in this case it is necessary), alleviating the need for model comparison to assess the inclusion of random effects. Note that a clear consequence of including the random intercept for speaker is that the estimate of the intercept becomes much lower than the value of \( \text{Ref.df} \) (due to the increased uncertainty about the intercept). Comparing the right graph of Fig. 8 to Fig. 4, however, does not reveal such a difference. Given that the model does not include individual variability in the difference between 'tenth' versus 'tent', this is not surprising.

4.7.2. Including a random slope

In similar fashion, we may include a by-speaker linear random slope for the two-word-contrast (Word) as follows:

```
m4 <- bam(Pos ~ Word + s(Time, by=Word) + s(Speaker, bs="re") + s(Word,Word,bs="re"), data=dat)
```

In the lmer specification this random slope would be represented by \((0 + \text{Word} | \text{Speaker})\). Unfortunately, in the GAM specification, it is not possible to model a correlation between random intercepts and random slopes (i.e. an lmer specification such as \((1 + \text{Word} | \text{Speaker})\) is not possible). At present this is a drawback compared to linear mixed-effects regression, at least when linear random slopes are used (but see 4.7.3, below). The summary of model m4 is as follows.

```
Parametric coefficients:
   Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.1091    0.0828   1.32    0.19
Wordtenth    0.6195    0.1032  6.00    2e-09 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
   edf Ref.df F p-value
s(Time):  Wordtenth:  7.95    8.71  44.6 < 2e-16 ***
          Wordtenth:  8.70    8.97  433.0 < 2e-16 ***
          s(Speaker): 15.48   41.00 1080.1  0.12
          s(Word,Word): 64.59  81.00  960.4   2.9e-05 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) = 0.534  Deviance explained = 53.7%
---

The summary shows an additional line, s(Word,Word), which is clearly significant, thereby supporting the inclusion of the random slope. The random intercept has become non-significant, indicating that most of the subject-variability is now captured by the random slope (i.e. distinguishing the two words). As before, adding a more appropriate random-effects structure affects the fixed effects (the supplementary material shows the output of compareML(m3,m4): m4 is a significant
improvement over m3, \( p < 0.001 \). Specifically, the intercept (i.e. the average anterior position of the T1 sensor for the word ‘tent’) does not differ significantly from 0 anymore due to the larger uncertainty, and also the constant difference between the word ‘tenth’ and ‘tent’ is associated with more uncertainty (i.e. much larger standard errors).

To visualize the effect of the additional random slope on the non-linear patterns, Fig. 9 shows both smooths (left) as well as their difference (right). (As before, the parameter \texttt{rm.ranef} has been set to \texttt{TRUE} in the plotting functions.) Comparing the left graph of Fig. 9 to the left graph of Fig. 8, the confidence bands are slightly wider, reflecting the increased standard errors in the model summary. The greatest change can be observed with respect to confidence bands of the difference, which have become much wider comparing the right graph of Fig. 9 (m4) to the right graph of Fig. 8 (m3). This, of course, is in line with allowing (necessary) variability in the difference between the two words ‘tenth’ and ‘tent’, and it mirrors the pattern visible in the model summary of m4.

4.7.3. Including non-linear random effects

While we are now able to model random intercepts and random slopes, our present model does not yet take the individual (non-linear) variability in the anterior position of the T1 sensor over time into account. Consequently, there is a need for a
non-linear random effect. Fortunately, this is possible within the generalized additive modeling framework. The following model specification illustrates how this can be achieved:

\[
m5 <- bam(Pos ~ Word + s(Time, by=Word) + s(Speaker, Word, bs="re") + s(Time, Speaker, bs="fs", m=1), data=dat)
\]

In this model the random intercept part has been replaced by the smooth specification \( s(\text{Time}, \text{Speaker}, bs="fs", m=1) \). This is a so-called factor smooth (hence the "fs" basis) which models a (potentially) non-linear difference over time (the first parameter) with respect to the general time pattern for each of the speakers (the second parameter: the random-effect factor). (Note the different ordering compared to the random intercepts and slopes.) The final parameter, \( m \), indicates the order of the non-linearity penalty. In this case it is set to 1, which means that the first derivative of the smooth (i.e. the speed) is penalized, rather than the default, second derivative of the smooth (i.e. the acceleration). Effectively, this results in factor smooths which are penalized more strongly than regular smooths. This, in turn, means that the estimated non-linear differences for the levels of the random-effect factor are assumed to be somewhat less ‘wiggly’ than their actual patterns. This reduced non-linearity therefore lines up nicely with the idea of shrinkage of the random effects (see footnote 3). Importantly, the factor smooths are not centered (i.e. they contain an intercept shift), and therefore the by-speaker random intercept term was dropped from the model specification. The summary of model m5 is shown below:

```
Parametric coefficients:
 Estimate Std.  t Pr(>|t|)
 (Intercept) 0.0768 0.0967 0.79 0.43
 Wordtenth 0.6196 0.1032 6.00 2e-09 ***
 ---
 Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
 edf Ref.df  F p-value
 s(Time): 7.47 8.03 9.61 2.6e-13 ***
 s(Time): 8.59 8.81 44.66 < 2e-16 ***
 s(Speaker, Word) 62.42 81.00 52.09 < 2e-16 ***
 s(Time, Speaker) 297.13 377.00 1168.20 < 2e-16 ***
 ---
 Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Chi-square test of fREML scores

<table>
<thead>
<tr>
<th>Model</th>
<th>Score</th>
<th>Edf</th>
<th>Chisq</th>
<th>Df</th>
<th>p.value</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>m4</td>
<td>11732.645</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m5</td>
<td>9755.733</td>
<td>9</td>
<td>1976.912</td>
<td>1</td>
<td>&lt; 2e-16</td>
<td>***</td>
</tr>
</tbody>
</table>

AIC difference: 4453.17, model m5 has lower AIC.

Fig. 10 shows the impact of this more complex random-effects structure on the resulting smooths (left), as well as their difference (right). Comparing the left graph of Fig. 11 to the left graph of Fig. 9, the confidence bands again are slightly wider, and the patterns also become slightly different. This is a logical consequence of allowing variability in the specific tongue trajectories for each individual speaker. By contrast, the confidence bands around the difference smooth have not changed. However, this is unsurprising given that m5 only models a single non-linear pattern over time, and the model does not yet allow for individual variability over time in distinguishing ‘tenth’ from ‘tent’.

To also include this type of (essential) random-effect variability, we fit the following model:

\[
m6 <- bam(Pos ~ Word + s(Time, by=Word) + s(Time, Speaker, by=Word, bs="fs", m=1), data=dat)
\]

The new model specification contains two changes. The first change consists of adding by=Word to the factor smooth specification. The second change is dropping the by-speaker random slope for Word. The reason for dropping the speaker-based variability in the constant difference between ‘tenth’ versus ‘tent’, is that this constant difference is already incorporated by the non-centered
factor smooth (i.e. by including two non-centered smooths per speaker).

The summary of the model shows the following:

| Estimate | Std. Error | t value | Pr(>|t|) |
|----------|------------|---------|----------|
| Intercept| 0.0844     | 0.0968  | 0.87     | 0.38     |
| Wordtenth| 0.5902     | 0.1427  | 4.14     | 3.6e-05 *** |

---

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘’ 1

Approximate significance of smooth terms:

<table>
<thead>
<tr>
<th>edf</th>
<th>Ref.df</th>
<th>F</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>s(Time): Wordtent</td>
<td>7.59</td>
<td>8.00</td>
<td>9.41</td>
</tr>
<tr>
<td>s(Time): Wordtenth</td>
<td>8.42</td>
<td>8.58</td>
<td>23.44</td>
</tr>
<tr>
<td>s(Time, Speaker): Wordtent</td>
<td>315.66</td>
<td>377.00</td>
<td>38.05</td>
</tr>
<tr>
<td>s(Time, Speaker): Wordtenth</td>
<td>327.18</td>
<td>368.00</td>
<td>43.13</td>
</tr>
</tbody>
</table>

---

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘’ 1

R-sq.(adj) = 0.782 Deviance explained = 79.3%
fREML = 9397.5 Scale est. = 0.21325 n = 12839

It is clear from the summary that both factor smooths (one for each word) are necessary. Furthermore, model comparison (see supplementary material) also revealed that the additional complexity of model m6 over model m5 was warranted. Fig. 12 visualizes the associated non-linear patterns and mainly shows that the confidence bands for the non-linear difference distinguishing 'tenth' from 'tent' have become much wider compared to Fig. 11 (i.e. m5). Of course, this is expected given that m6 allows for individual variability in the articulatory trajectories over time for the two words.

4.8. Taking into account autocorrelation in the residuals

In the previous section, we have accounted for the speaker-specific variability in the data, by using a (non-linear) mixed-effects regression approach. However, as we are analyzing time-series data, there is another type of dependency involved. Specifically, the residuals (i.e. the difference between the fitted values and the actual values) of subsequent time points in the time series will be correlated. How severe this so-called autocorrelation is, can be seen in Fig. 13. This graph was obtained by using the itsadug function acf_resid:

m6acf<-acf_resid(m6)

The first vertical line in this autocorrelation graph is always at height 1 (i.e. each point has a correlation of 1 with itself). The second line shows the amount of autocorrelation present at a lag of 1 (i.e. comparing measurements at time C0 and time t). In Fig. 13, this value is about 0.91, which means that each additional time point only yields relatively little additional information. (There is also autocorrelation present at higher lags, but this may (partly) be caused by the autocorrelation at lag 1.) If this dependency is not brought into the model, it is likely that the strength of the effects is severely overestimated. Fortunately, the function bam is able to incorporate an AR(1) error model for the residuals. While an AR(1) model is a very simple model of autocorrelation and may not be adequate to alleviate the autocorrelation problem, in most cases this simple approach seems to be sufficient.

Note that autocorrelation can only be assessed adequately if the dataset is ordered (otherwise the autocorrelation graph is useless as a diagnostic tool). This means that for each speaker and each word pronunciation (and sensor, and axis, if applicable), the rows have to be ordered by (increasing) time. Consequently, in the dataset each separate time series will have to be positioned one after another. To make sure the data is ordered, it is useful to use the itsadug function start_event:

dat <- start_event(dat, event=c("Speaker","Trial"))

Fig. 11. Non-linear smooths and difference comparing 'tenth' to 'tent' for model m5. See details in Fig. 8 caption.
The function `start_event` assumes there is a column `Time` in dataset `dat`, including the time points associated with each data point. It subsequently orders the data by `Time` for each individual time series as determined by the `event` parameter (in this case, there is a single articulatory trajectory of the T1 sensor in the anterior-posterior dimension for every combination of `Speaker` and `Trial`). In addition, this function adds a column `start.event` to the dataset which is equal to `TRUE` whenever the row is associated with the first data point of every time series and equal to `FALSE` otherwise. This column is useful to identify which subsequent points are expected to show autocorrelation in the residuals. Whenever the value of the column `start.event` equals `FALSE`, the residual at that point is assumed to correlate with the residual at the previous point, whereas if the column equals `TRUE` this is not expected to be the case (i.e. the residual of the first point in a new trial is not assumed to be correlated with the residual of the last point of the previous trial, as the words were not pronounced immediately after one another).

As indicated, the function `bam` is able to incorporate an AR (1) error model for the residuals in a Gaussian model. There are two additional parameters which need to be set for this. The first parameter is `rho`. This is an estimate of the amount of autocorrelation. Using the height of the second line in the autocorrelation graph (i.e. `m6acf[2]`) is generally a good estimate. The second parameter is `AR.start` which should be set to a variable containing `TRUE` at the start of a new time series and `FALSE` otherwise. This parameter should be set to the column `start.event` of the data frame (in our case, `dat`) if the function `start_event` was used. The revised `bam` function call now becomes:
m7 <- bam(Pos ~ Word + s(Time, by=Word) + 
s(Time,Speaker, by=Word, bs="fs", m=1), data=dat, 
rho=m6acf[2], AR.start=dat$start.event)

Inspecting the new autocorrelation graph in Fig. 14, shows that the autocorrelation has been removed almost completely. As the autocorrelation at lag 1 is slightly negative, a lower rho value might seem a better option. However, the supplementary material (model m7.alt) shows that this resulted in an increase of the autocorrelation at higher lags. We therefore used a rho value of 0.912 (i.e. equal to m6acf[2]) in all subsequent models in Section 4. In our experience, setting the rho value to the autocorrelation at lag 1 as determined via the acf function is the best approach to correct for autocorrelation, and this is the approach we use throughout the manuscript.

The summary of model m7 shows the following:

Parametric coefficients:

| Estimate  | Std. Error | t value | Pr(>|t|) |
|-----------|------------|---------|----------|
| (Intercept) | 0.0791 | 0.0875 | 0.9 | 0.37 |
| Wordtenth | 0.5814 | 0.1292 | 4.5 | 6.8e-06 *** |

---

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:

<table>
<thead>
<tr>
<th>Edf</th>
<th>Ref.df</th>
<th>F value</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>s(Time): Wordtenth</td>
<td>7.47</td>
<td>8.12</td>
<td>9.00</td>
</tr>
<tr>
<td>s(Time): Wordtenth</td>
<td>8.32</td>
<td>8.60</td>
<td>22.05</td>
</tr>
<tr>
<td>s(Time, Speaker): Wordtenth</td>
<td>229.34</td>
<td>377.00</td>
<td>2.87</td>
</tr>
<tr>
<td>s(Time, Speaker): Wordtenth</td>
<td>267.85</td>
<td>368.00</td>
<td>3.84</td>
</tr>
</tbody>
</table>

---

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) = 0.763 Deviance explained = 77.3%
fREML = -3286.8 Scale est. = 0.18714 n = 12839

The visualization in Fig. 15 shows that including the autocorrelation in the residuals has had only a negligible influence on the standard errors (the F-values associated with the smooths in the model summaries are only slightly lower). Note that the explained deviance has dropped slightly. This is due to the model taking into account the autocorrelation, and therefore predicting the actual values slightly less well than before.

4.9. Including a two-dimensional interaction

Frequently, it is very insightful to look at interactions which involve two numerical predictors. To illustrate how two-dimensional non-linear interactions can be included, we will extend the above model by investigating if there are trial effects present in our data. Trial effects are frequently included in the analysis, in order to take into account effects of repetition (Winter, 2015), fatigue, attention, or learning (Baayen et al., 2017). While this interaction is not particularly interesting for our data, given that we only focus on a few trials (in this example, only four trials), we nevertheless include it here to illustrate the concepts necessary to understand two-dimensional non-linear interactions.

A thin plate regression spline can also be used to model non-linear interactions. However, it is essential that the predictors involved in a thin plate regression spline interaction are isotropic, i.e. they need to be measured on the same scale (such as longitude and latitude; see Wieling et al., 2011 for an example). In a thin plate regression spline the amount of non-linearity associated with a unit change in the value of each incorporated predictor is assumed to be identical, and this assumption is only valid for isotropic predictors.

To model predictors which are not on the same scale (such as Time and Trial in our case), a tensor product smooth interaction (in short, tensor product) can be used. A tensor product essentially models a non-linear interaction by allowing the coefficients underlying the smooth for one variable to vary non-linearly depending on the value of the other variable (see Wood, 2017, pp. 224–232). In mgcv, a tensor product can be included in the model specification by using the te function. By default, the te-constructor uses two (default) 5-dimensional cubic regression splines (bs="te"). Consequently, the k-parameter for each variable is limited to 5: k=5. Extending model m7 to include a two dimensional interaction between Time and Trial thus results in the following function call:

m8 <- bam(Pos ~ Word + te(Time, Trial, by=Word) + 
s(Time,Speaker, by=Word, bs="fs", m=1), data=dat, 
rho=0.912, AR.start=dat$start.event)

The summary shows the following:

Parametric coefficients:

| Estimate  | Std. Error | t value | Pr(>|t|) |
|-----------|------------|---------|----------|
| (Intercept) | 0.0479 | 0.0895 | 0.54 | 0.59 |
| Wordtenth | 0.6084 | 0.1328 | 4.58 | 6.6e-06 *** |

---

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:

<table>
<thead>
<tr>
<th>edf</th>
<th>Ref.df</th>
<th>F value</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>te(Time,Trial): Wordtenth</td>
<td>9.16</td>
<td>10.03</td>
<td>8.08</td>
</tr>
<tr>
<td>te(Time,Trial): Wordtenth</td>
<td>8.57</td>
<td>8.78</td>
<td>16.20</td>
</tr>
<tr>
<td>s(Time,Speaker): Wordtenth</td>
<td>231.45</td>
<td>377.00</td>
<td>2.89</td>
</tr>
<tr>
<td>s(Time,Speaker): Wordtenth</td>
<td>278.97</td>
<td>368.00</td>
<td>4.11</td>
</tr>
</tbody>
</table>

---

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) = 0.777 Deviance explained = 78.6%
fREML = -3282.8 Scale est. = 0.18683 n = 12839
Fig. 15. Non-linear smooths and difference comparing ‘tenth’ to ‘tent’ for model m7 (using a rho value of 0.912). See details in Fig. 8 caption.

Fig. 16. Contour plots visualizing the non-linear interactions of model m8 between time and trial for the word ‘tent’ (top-left), ‘tenth’ (top-right) and their difference (bottom-left).
It is clear that the first two s-terms have been replaced by te-terms in the summary. In both cases, however, the effective degrees of freedom of the tensor products have not changed much. Of course, visualization is essential to see what is going on. As we need to visualize two-dimensional patterns, we have to use other visualization functions than before. In particular, we will use the itsadug functions fvisgam and plot_diff2 which both yield contour plots. (Note that the function fvisgam differs from the mgcv function vis.gam in that it allows random effects to be excluded from the visualization.)

The commands to visualize the contour plots for ‘tenth’ and for ‘tent’, as well as their difference are as follows:

```r
fvisgam(m8, view=c("Time","Trial"),
cond=list(Word=c("tent")), main="m8: tent",
rm.ranef=TRUE, zlim=(-0.9,1.6), color="gray"

fvisgam(m8, view=c("Time","Trial"),
cond=list(Word=c("tenth")), main="m8: tenth",
rm.ranef=TRUE, zlim=(-0.9,1.6), color="gray"

plot_diff2(m8, view=c("Time","Trial"),
comp=list(Word=c("tenth","tent")), rm.ranef=TRUE,
main="Difference tenth - tent",
color="gray"
```

For both functions, the first parameter is the model name. The second parameter, view, should contain two variable names included in the tensor product of the model. The first variable is plotted at the x-axis, whereas the second variable is plotted at the y-axis. Other common parameters include main, which sets the title of the plot, rm.ranef, which (if set to TRUE) excludes the influence of the random effects when creating the visualization, color, which sets the color scheme (in this case, grayscale), and zlim, which sets the lower and upper limit of the color range.

Furthermore, the function fvisgam has an additional cond parameter, which is a named list containing the value of the predictors in the model which should be set to fixed values (i.e. in this case only the specific word). The function plot_diff2 has a comp parameter to determine which two levels should be compared (see explanation for plot_diff above). The resulting three contour plots are shown in Fig. 16. Lighter shades of gray indicate higher values (i.e. a more anterior T1 position), whereas darker shades of gray indicate lower values. Black contour lines connect points with identical values. For example, the contour plot associated with ‘tent’, shows two peaks over time (around 0.2 and 0.7), which are reduced in size for later trials. By contrast, the contour plot associated with ‘tenth’ shows a single, higher peak over time (around 0.7) which gets lower (and somewhat delayed) for later trials. To further help interpretation, Fig. 17 shows a visualization of the difference contour plot together with the associated one-dimensional differences smooths for three trials (trial 500, 300, and 100). The one-dimensional graphs have been generated using the function plot_diff with the parameter cond set to (e.g.) list(Trial=100). In this case, all three one-dimensional graphs show a very similar pattern, with only slightly higher and earlier peaks for earlier trials. (The black dotted lines have been added to each graph to make these differences more apparent.)

The two-dimensional tensor product of time and trial implicitly incorporates three parts: an effect over time, an effect over trial, and the pure interaction between time and trial. Inspecting Figs. 16 and 17, it does not appear there is a very strong influence of trial. Consequently, it makes sense to see whether an effect of trial would need to be included at all. For this reason, it is useful to decompose the tensor product into its separate parts. While we already have seen how to model one-dimensional smooths, we need to introduce a new constructor, ti, to model a pure interaction term. This constructor, with identical syntax as the te- constructor, models the pure interaction between the variables. The specification of the model (m8.dc) of the decomposed tensor product is as follows:

```r
m8.dc <- bam(Pos~Word* s(Time, by=Word) + s(Trial, by=Word) + ti(Time, Trial, by=Word) ~ s(Time,Speaker,by=Word,bc="bs","m1"), data=dat,
rho=0.912, AR.start=dat$start.event)
```

The summary of the model is as follows:

**Parametric coefficients:**

| Estimate | Std. t Pr(>|t|) |
|----------|------------------|
| (Intercept) | 0.0550 0.0894 0.62 0.54 |
| Wordtenth | 0.6561 0.1314 4.84 1.3e-06 *** |

**Approximate significance of smooth terms:**

<table>
<thead>
<tr>
<th>Rank</th>
<th>edf</th>
<th>Ref.df</th>
<th>F.p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ti(Time,Trial): Wordtent</td>
<td>1.00</td>
<td>1.00</td>
<td>2.35 0.05865</td>
</tr>
<tr>
<td>ti(Time,Trial): Wordtent</td>
<td>1.67</td>
<td>2.03</td>
<td>2.31 0.09316</td>
</tr>
<tr>
<td>s(Time,Speaker): Wordtent</td>
<td>229.36 377.00 2.89 &lt;2e-16 ***</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s(Time,Speaker): Wordtenth</td>
<td>268.42 368.00 3.89 &lt;2e-16 ***</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Clearly, the main effects of time and trial for both words are still significant. While the smooths over time are non-linear (high edf), the smooths over trial are (almost) linear (edf close to 1). Furthermore, the pure interaction between time and trial is not significant. The relative size of the effects (excluding the non-significant interactions) can be visualized using the mgcv plot function and is shown in Fig. 18. It is clear that the influence of the trial number on the anterior position of the T1 sensor is relatively modest and (almost) linear. While the significant trial effects may seem interesting, we hasten to add that we only investigate four trials per speaker in this example. Therefore, any trial effects we observe here will necessarily...
Fig. 17. Difference contour plot showing the interaction between time and trial (left) of model m8. The plots on the right show the corresponding non-linear pattern over time for three distinct trials: 500 (top row), 300 (middle row) and 100 (bottom row). The dotted black lines at (1.035, 0.745) facilitate comparison between the three graphs.
be linked to where in the experiment each speaker encountered the two words, and will almost certainly not be representative of real trial effects. Consequently, and also to keep the models relatively simple, we will exclude trial effects in the remaining part of this tutorial.

4.10. Including the language difference

We only considered the anterior-posterior T1 articulation difference between the two words in the models above, but a more relevant question is how this difference varies depending on the language of the speaker. As the sound /t/ does not occur in the Dutch language, we are particularly interested in assessing if Dutch speakers show a different (i.e. smaller) contrast between a minimal pair involving /t/ versus /t/ than the English speakers. Whereas a naive approach to achieve this might be to fit two separate models (one for each language) and visually compare the patterns, this approach is not adequate. By fitting two separate models it is not possible to evaluate whether the additional complexity (i.e. the addition of the language factor) is warranted. For example, while a visual comparison (even when fitting a single model for all data) may show that the patterns are relatively similar, there may be enough evidence to conclude that the small difference between them is real. Alternatively, the two patterns may seem quite different, but if the confidence bands are very wide, the difference between the two patterns may never be significantly different from zero. Note that a difference in significance of the patterns is also not informative. For example, even though the confidence bands for the non-linear difference between might be completely overlapping with the x-axis for one group, but not for the other group, they may still be statistically indistinguishable. For example, the patterns may be identical, with simply more variability (i.e. wider confidence bands) for one group than the other. In sum, a visual inspection does not provide enough information to decide if the additional complexity is necessary. Instead, we follow the approach put forward in Section 4.5 and more formally evaluate whether the additional complexity is warranted.

To distinguish the two language groups, we first create a new variable, WordLang, which is the interaction between Word and Lang (i.e. having four levels, the words ‘tent’ and ‘tenth’ for both English and Dutch speakers):

```
dat$WordLang <- interaction(dat$Word, dat$Lang)
```

We now use the new variable WordLang in our model instead of Word:

```
m9 <- bam(Res~WordLang + s(Time, by=WordLang) + s(Time,Speaker,by=Word,bs="fs",m=1), data=dat, rho=0.912, AR.start=dat$start.event)
```

Comparing model m9 to model m7 (both now fitted with method="ML" and named m7.ml and m9.ml) shows that it is necessary to include a distinction between languages:
we create two reference levels via \( \text{Pos} \) and \( \text{Pos}' \) one for each level of the new nominal variable. With respect to the intercept (in this case the reference level is the (top-left) as well as the difference (top-right) which reveals a clear and significant pattern. The bottom row shows the individual smooths for the English speakers and the native language is English and "FALSE" otherwise, whereas the other factor (NLTenthO) is set to "TRUE" whenever the word equals ‘tenth’ and the native language is Dutch and "FALSE" otherwise. The complete model specification, including the creation of the two ordered factors is as follows:

\[
\text{m9.ord} <- \text{bam}(\text{Pos} \sim \text{Word}, k=20) + \text{s}(\text{Time, by=ENTenthO, k=20}) + \text{s}(\text{Time, by=NLTenthO, k=20}) + \text{s}(\text{Time, by=Word, bs="fs", m=1}), \text{data} = \text{dat}, \text{rho}=0.912, \text{AR.start} = \text{dat}\_\text{start.event})
\]

The summary of model m9 now shows three contrasts with respect to the intercept (in this case the reference level is the word ‘tenth’ for the native English speakers) and four smooths, one for each level of the new nominal variable.

### Parametric coefficients:

| Estimate  | Std. | t | Pr(>|t|) |
|-----------|------|---|---------|
| (Intercept) | -0.098 | 0.119 | -0.82 | 0.410 |
| WordLangtenth.EN | 0.732 | 0.174 | 4.20 | 2.7e-05 *** |
| WordLangtenth.NL | 0.363 | 0.173 | 2.09 | 0.037 * |
| WordLangtenth.NL | 0.790 | 0.182 | 4.34 | 1.4e-05 *** |

**Signif. codes: 0 *** 0.001 ** 0.01 * 0.05 . 0.1 ' ' 1**

Approximate significance of smooth terms:

<table>
<thead>
<tr>
<th>s(Time):</th>
<th>edf</th>
<th>Ref.df</th>
<th>F</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WordLangtenth.EN</td>
<td>3.84</td>
<td>4.53</td>
<td>2.27</td>
<td>0.045 *</td>
</tr>
<tr>
<td>WordLangtenth.EN</td>
<td>7.95</td>
<td>8.37</td>
<td>15.77</td>
<td>&lt;2e-16 ***</td>
</tr>
<tr>
<td>WordLangtenth.NL</td>
<td>7.48</td>
<td>8.15</td>
<td>10.67</td>
<td>1.2e-15 ***</td>
</tr>
<tr>
<td>WordLangtenth.NL</td>
<td>7.73</td>
<td>8.21</td>
<td>11.72</td>
<td>&lt;2e-16 ***</td>
</tr>
<tr>
<td>Wordtenth</td>
<td>218.56</td>
<td>376.00</td>
<td>2.67</td>
<td>&lt;2e-16 ***</td>
</tr>
<tr>
<td>Wordtenth</td>
<td>255.53</td>
<td>367.00</td>
<td>3.49</td>
<td>&lt;2e-16 ***</td>
</tr>
</tbody>
</table>

**Signif. codes: 0 *** 0.001 ** 0.01 * 0.05 . 0.1 ' ' 1**

The results of this model are shown in Fig. 19. The top row shows the individual smooths for the English speakers (top-left) as well as the difference (top-right) which reveals a clear and significant pattern. The bottom row shows the same graphs for the Dutch speakers, with a much smaller difference between the two words. Whether or not the latter (small) difference is significant, should be assessed formally, however.

For this reason, we re-specify the model using ordered factors. As we want to evaluate the difference between ‘tenth’ and ‘tent’ for both the English and Dutch speakers, separately, we create two reference levels via \( s(\text{Time, by=Lang}) + \text{Lang}, \) one for each group. We then create two separate ordered factors. One factor (ENTenthO) is set to "TRUE" whenever the word equals ‘tenth’ and the language is English and "FALSE" otherwise, whereas the other factor (NLTenthO) is set to "TRUE" whenever the word equals ‘tenth’ and the native language is Dutch and "FALSE" otherwise. The complete model specification, including the creation of the two ordered factors is as follows:

\[
\text{dat}\_\text{NLTenthO} \leftarrow \text{as.ordered(\text{dat}\_\text{Lang} == "NL" & \text{dat}\_\text{Word} == "tenth")}
\]

\[
\text{dat}\_\text{ENTenthO} \leftarrow \text{as.ordered(\text{dat}\_\text{Lang} == "EN" & \text{dat}\_\text{Word} == "tenth")}
\]

The position of the T1 sensor in the anterior-posterior direction of the English speakers for the word ‘tenth’ can be found at the intercept of the parametric coefficients. It is clear from the line.
for LangNL that Dutch speakers differ significantly from the English speakers for the reference-level word 'tent'. When focusing on the English speakers, the line starting with ENTenthOTRUE indicates that the English speakers show a more frontal position for the word 'tenth' than for the word 'tent' during the pronunciation of the whole word, and that this difference is significant. Similarly, the line starting with NLTenthOTRUE shows that there is a significant constant difference between the word 'tenth' and 'tent' for the Dutch speakers. (Since NLTenthO is never "TRUE" for the English speakers, it functions only as a contrast for the Dutch speakers.) It is useful to compare the estimates of model m9.ord to those of model m9. In m9, the estimate for WordLangtent.NL is about 0.36 (higher than the reference level), whereas it is 0.79 (higher than the same reference level) for WordLangtenth.NL. Clearly the difference between 'tenth' and 'tent' for the Dutch speakers is therefore about 0.43. And this value is indeed close to the value of 0.45 shown in the line associated with NLTenthOTRUE in model m9.ord. Note that the computation does not exactly hold, as the models are not completely identical (i.e. in one model separate smooths for each level are included, whereas the other model includes explicit difference smooths).

Similarly to the parametric coefficients, there are now two difference smooths, one for the English speakers (s(Time):ENTenthOTRUE) which is highly significant, and one for the Dutch speakers (s(Time):NLTenthOTRUE) which is not. When dropping this non-significant smooth and refitting the model, the constant difference between ‘tenth’ and ‘tent’ also does not reach significance anymore (p = 0.084; see supplementary material: model m9.ord2). We therefore conclude that there is not enough support for a statistically significant (non-linear) difference between the word ‘tent’ and the word ‘tenth’ for the Dutch speakers, at least not when taking the complete word pronunciation into account. To provide further support for this conclusion, we may also investigate this difference using a binary difference smooth (combining the intercept and non-linear difference). The specification for this model, including the creation of the two binary variables is as follows:

![Fig. 19. Non-linear smooths and difference comparing 'tenth' to 'tent' for model m9, for both English (top row) and Dutch speakers (bottom row). See details in Fig. 8 caption.](image-url)
The summary of this model shows the following:

```
Parametric coefficients:          Estimate  Std.  t Pr(>|t|)
                                (Intercept)   -0.0939 0.1189 -0.79  0.430
LangNL                          0.3635  0.1733  2.10  0.036 *
---                                Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 1
Approximate significance of smooth terms:  edf Ref.df F  p-value
s(Time):LangEN                   4.32    3.89  5.05  2.38  0.037 *
```

In this model, s(Time, by=IsNLTenth) represents the difference between the ‘tenth’-‘tent’ contrast for the Dutch speakers versus that of the English speakers, while s(Time, by=IsTenth) represents the difference between the ‘tenth’-‘tent’ contrast for the English speakers (i.e. comparable to s(Time, by=IsENTenth) in model m9.bin). To see why this is the case, it is useful to see which smooths are combined to model the four conditions. It is helpful to first recall that s(Time, by=IsTenth) equals 0 for the word ‘tent’ and represents a smooth without a centering constraint for the word ‘tenth’. Similarly, s(Time, by=IsNLTenth) equals 0 for the word ‘tent’ pronounced by both groups and also when the word ‘tenth’ is pronounced by the English speaker group. When the word ‘tenth’ is pronounced by the Dutch speaker group, s(Time, by=IsNLTenth) represents a smooth without a centering constraint. The smooths which have to be summed for each condition can therefore be listed as follows:

- **English ‘tent’**: s(Time):LangEN
- **English ‘tenth’**: s(Time):LangEN + s(Time, by=IsTenth)
- **Dutch ‘tent’**: s(Time):LangNL
- **Dutch ‘tenth’**: s(Time):LangNL + s(Time, by=IsTenth)

Following the same reasoning as in Section 4.5.2, s(Time, by=IsTenth) represents the difference (i.e. the contrast) between ‘tenth’ and ‘tent’ for the English speakers. The contrast between ‘tenth’ and ‘tent’ for the Dutch speakers consists of both s(Time, by=IsTenth) and s(Time, by=IsNLTenth). Consequently, the difference between the Dutch and the English ‘tenth’-‘tent’ contrast must be represented by s(Time, by=IsNLTenth).
The summary of m9b.bin shows that this difference does not reach significance:

```
Parametric coefficients:
  Estimate Std. t value Pr(>|t|)
(Intercept) -0.0869  0.1187  -0.73   0.464
LangNL  0.3472  0.1724   2.01  0.044 *
---
Signif. codes: 0 '***' 0.001 '*' 0.01 '.' 0.05 ' ' 1
```

Approximate significance of smooth terms:
```
  edf Ref.df F value p-value
s(Time):LangEN 4.80  5.58  2.98  0.0089 **
s(Time):LangNL 7.74  8.28 12.33 < 2e-16 ***
s(Time):IsTenth 8.93  9.30  8.58  2.4e-13 ***
s(Time):IsNLTenth 4.29  4.77  1.65  0.1630
s(Time,Speaker): Wordtent 217.43 376.00  2.64  0.0089 **
s(Time,Speaker): Wordtenth 256.57 376.00  3.49  < 2e-16 ***
---
Signif. codes: 0 '***' 0.001 '*' 0.01 '.' 0.05 ' ' 1
```

For completeness, Fig. 21 shows the English difference (i.e. contrast) between 'tenth' and 'tent' and the non-significant difference comparing the Dutch 'tenth'- 'tent' contrast to that of the English speakers. Corresponding with the binary difference model, both an ordered factor model and model comparison indicate that this difference is indeed not significant (see supplementary material: m9b.ord, and m9a.bin.ml vs. m9b.bin.ml). Consequently, the reason that model m9 (including the language difference) was preferred over model m7 (without the language difference) is due to the difference in how both groups of speakers pronounce the reference word 'tent' (i.e. more anterior for the Dutch speakers, see the dark, blue lines in Fig. 19). This finding also emphasizes the need for modeling these differences directly, rather than inadequately comparing significance values.

Of course, given that the minimal pair 'tenth'- 'tent' only differs at the end of the word, we might be reducing the power of our analysis by focusing on the entire time course. Since we don’t observe any differences (as would be expected, considering that the minimal pair only differs at the end) in the first half of the word (see Fig. 21, right), we also conducted the same analysis using only the second half of the word pronunciations (i.e. from normalized time 0.5 to 1.0). The supplementary material (Section 5.12) indeed shows that the difference between the Dutch and English speakers in how they contrast 'tenth' from 'tent' in the second half of the word significantly differs (p = 0.01). Dutch speakers exhibit a smaller distinction between 'tenth' and 'tent' than the English speakers, in line with our expectations. Note that while in this case there is a clear argument for limiting the analysis to a certain time window, we caution against limiting the time window (subjectively) in order to identify significant differences when there is not an a priori reason to do so.

4.11. Speeding up computation

For our present small dataset, which only includes 2 words, the most complex models take about 30 seconds to fit on a single core of a 36-core 2.3 GHz Intel Xeon E5-2699 v3 using fast restricted maximum likelihood estimation (fitting with maximum likelihood takes about seven times as long). However, this dataset only contains about 10,000 rows. Especially if we use larger datasets (the full dataset contains more than 100,000 rows, while Wieling et al., 2016 analyzed a dataset with more than a million rows), computational time will become rather substantial. While bam is already much faster than gam, it can be made even faster by taking advantage of the fact that numerical predictors often only have a modest number of unique (rounded) values. Consequently, at the cost of some precision, substantial reductions in computation time can be achieved. To use this discretization approach, the bam parameter discrete has to be set to TRUE (the default is FALSE). Together with the discrete parameter, it is also possible to set the nthreads parameter which controls the number of cores used in parallel to obtain the model fit (the default value is 1). For example, model m9b.bin took 17.4 seconds to fit with discrete set to FALSE. When set to TRUE and using single core, computation time was reduced to 5.3 seconds. Using two processors instead of one, the computation time was further reduced to 5.1 seconds. However, note that the speed-up using multiple processors is much more substantial when the models take several hours to fit rather than several seconds. The only restriction for using discrete, is that the model has to be fit with fast restricted maximum likelihood estimation and thus model comparison of models differing in the fixed effects is not possible (but, of course, binary smooths and ordered factors can still be used).

To see that the model fit with discrete set to TRUE is indeed highly similar to the model fit with discrete set to FALSE, the summary of m9b.bin.discrete is shown below and the visualization is shown in Fig. 22 (for direct comparison with Fig. 21).

```
Parametric coefficients:
  Estimate Std. t value Pr(>|t|)
(Intercept) -0.0855  0.1190  -0.71   0.476
LangNL  0.3583  0.1734   2.07  0.039 *
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Approximate significance of smooth terms:
```
  edf Ref.df F value p-value
s(Time):LangEN 4.75  5.52  2.96  0.0084 **
s(Time):LangNL 7.71  8.25 11.96 < 2e-16 ***
s(Time):IsTenth 8.93  9.30  9.00  1e-13 ***
s(Time):IsNLTenth 4.32  4.81  1.68  0.1548
s(Time,Speaker): Wordtenth 217.08 376.00  2.64  0.0089 **
s(Time,Speaker): Wordtent 256.57 376.00  3.49  < 2e-16 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

R-sq.(adj) = 0.762 Deviance explained = 77.2%
\[ \text{fREML} = -3303.5 \; \text{Scale est.} = 0.18717 \; n = 12839 \]
Fig. 20. Visualization of the partial effects of model m9.ord (top row) and m9.bin (bottom row) representing the difference between 'tenth' and 'tent' for both English (left) and Dutch (right) speakers. In all graphs, the pointwise 95%-confidence intervals are visualized by shaded bands.

Fig. 21. Visualization of the partial effects of model m9.b.bin representing the difference between 'tenth' and 'tent' for the English speakers (left) and how this difference needs to change to obtain the difference between 'tenth' and 'tent' for the Dutch speakers (right). In both graphs, the pointwise 95%-confidence intervals are visualized by shaded bands. The pattern on the right is not significant.
5. Generalized additive modeling: including all words

In Section 4, we have illustrated and explained all separate parts of an appropriate generalized additive modeling analysis. Now we are ready to create an appropriate model for all data (dataset full), including the appropriate random-effects structure and a correction for autocorrelation. An important distinction with respect to the previous models is that we now seek to generalize over all words. Consequently, Word now becomes a random-effect factor (i.e. a factor smooth over time), whereas the nominal variable Sound allows us to distinguish between /h/-words ("TH") and /t/-words ("T"). We further need to take into account the location of the contrast (Loc: "Init" vs. "Final"). However, to keep the models discussed in this section relatively simple, we will restrict our analysis to words with a word-final contrast and only analyze the pattern in the second half of the word (cf. Section 4.10, final paragraph; dataset: fullfinal). The supplementary material (Sections 7 and 8) contains the analysis for both sets of words (i.e. those with a word-final contrast and those with a word-initial contrast) in a single model. Importantly, the conclusion on the basis of the full model is similar to that of the simpler model discussed below.

As we are interested in assessing if Dutch speakers contrast /h/-words from /t/-words less strongly than English speakers, we will create a binary smooth model similar to m9b.bin and therefore fit the following model (the optimal value for rho was determined to be 0.952; see supplementary material):

```
m9b.bin.discrete <- bam(Pos~Lang + s(Time, by=Lang) + s(Time, by=IsTH) + s(Time, by=IsNLTH) + s(Time, Speaker, by=Sound, bs="fs",m=1) + s(Time, Word, by=Lang, bs="fs",m=1), data=fullfinal, discrete=TRUE, rho=0.952, AR.start=fullfinal$start.event)
```

In this specification, IsTH is equal to 1 for /h/-words and 0 otherwise. Similarly, IsNLTH is equal to 1 for /h/-words pronounced by the Dutch speakers and 0 otherwise. It is easy to see that this model specification is very similar to that of m9b.bin. The only differences are that we (1) used IsTH and IsNLTH instead of IsTenth and IsNLTenth, (2) replaced by=Word with by=Sound in the by-speaker factor smooth specification, and (3) included an additional factor smooth for the (now) random-effect factor Word, to take into account the structural variability in tongue movement per word. As words may be pronounced differently depending on the language group the speaker belongs to, two smooths are modeled for each word via the by=Lang part of the by-word factor smooth specification.

Fitting this model took about 15 seconds with discrete set to TRUE. The remaining autocorrelation in this model was comparable to that shown in Fig. 14 (see supplementary material). The model summary shows the following:

```
Parametric coefficients:
 Estimate Std. Error t value Pr(>|t|)
 (Intercept) -0.162 0.269 -0.60 0.55
 LangNL 0.137 0.427 0.32 0.75

Approximate significance of smooth terms:
edf Ref.df F p-value
 s(Time):LangEN 3.08 3.49 2.92 0.02745 *
 s(Time):LangNL 5.26 5.70 2.53 0.01831 *
 s(Time):IsTH 5.45 5.90 4.46 0.00018 ***
 s(Time):IsNLTH 2.06 2.09 0.44 0.63448

---
```

Fig. 22. Visualization of the partial effects of model m9b.bin.discrete representing the difference between 'tenth' and 'tent' for the English speakers (left) and how this difference needs to change to obtain the difference between 'tenth' and 'tent' for the Dutch speakers (right). In both graphs, the pointwise 95%-confidence intervals are visualized by shaded bands. The pattern on the right is not significant. Note the negligible difference from the full precision results shown in Fig. 21.
Importantly, the line shown in boldface reveals that the difference between the Dutch and English /h/-/t/ (word-final) contrast is not significant. In other words, the analysis does not allow us to reject the null hypothesis outlined in Section 2. Dutch non-native speakers therefore do not significantly differ from native English speakers in contrasting /h/ and /t/ in articulation. Fig. 23 visualizes the associated binary difference smooths, corroborating the model summary.

After having fitted the final model, the only remaining issue is to conduct model criticism. Fig. 24 shows the result of gam.check(ffmc1). As these diagnostic graphs are based on uncorrected residuals (i.e. ignoring the autocorrelation parameter rho), the scatter plots still show the spaghetti-like patterns indicative of dependencies within the trajectories (which have, in fact, been corrected). Unfortunately, the left graphs of Fig. 24 reveal that the residuals also show a problematic non-normal distribution, which almost certainly will affect the estimates and p-values of the model. This will need to be addressed, as we therefore cannot trust the results of model ffmc1.

Given that the pattern of the residuals resembles that of a normal distribution with heavier tails, a sensible approach is to fit the model using the scaled-t family for heavy tailed data. To do this, only a single parameter needs to be added to the model specification of ffmc1: family="scat". While this change is very simple, the time needed to fit this type of model has increased from 15 seconds to almost 7 minutes. Using multiple processors is beneficial here: using 32 processors reduces the time needed to less than a minute (see the supplementary material for a more substantial speedup when using the full dataset: doubling the number of processors divides the running time on average by a factor of about 1.7). Fortunately, the resulting model summary for model ffmc1s, shown below, is reasonably similar to the Gaussian model (as are the associated patterns; see supplementary material) and the conclusion on the basis of model ffmc1 still appears to hold (see line in boldface).

Model criticism of the scaled-t model (shown in Fig. 25) shows that the distribution of the residuals now nicely matches the assumed distribution.

### 6. Discussion

In this tutorial, we have explained the use of generalized additive (mixed-effects) modeling by analyzing an articulatory dataset contrasting the pronunciation of L1 and L2 speakers of English. With respect to our research question, we have shown that while native English speakers seem to more clearly distinguish /h/ from /t/, there is insufficient evidence (at least when analyzing a single sensor in a single dimension) to conclude that the distinction made by non-native Dutch (highly

---

**Signif. codes:** 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj)= 0.633 Deviance explained = 64%

fREML = -2831.1 Scale est. = 0.6407 n = 31225

**Parametric coefficients:**

| Estimate | Std. Error | t value | Pr(>|t|) |
|----------|------------|---------|----------|
| (Intercept) | -0.197 | 0.267 | -0.74 | 0.46 |
| LangNL | 0.234 | 0.412 | 0.57 | 0.57 |

**Approximate significance of smooth terms:**

<table>
<thead>
<tr>
<th>edf</th>
<th>Ref.df</th>
<th>F value</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>s(Time):LangEN</td>
<td>2.36</td>
<td>2.67</td>
<td>1.53</td>
</tr>
<tr>
<td>s(Time):LangNL</td>
<td>4.94</td>
<td>5.30</td>
<td>2.87</td>
</tr>
<tr>
<td>s(Time):IsTH</td>
<td>6.37</td>
<td>6.75</td>
<td>4.81</td>
</tr>
<tr>
<td>s(Time):IsNLTH</td>
<td>3.84</td>
<td>4.17</td>
<td>0.88</td>
</tr>
</tbody>
</table>

---

**Signif. codes:** 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj)= 0.645 Deviance explained = 53.9%

fREML = 19236 Scale est. = 1 n = 31225

---

6. Discussion

In this tutorial, we have explained the use of generalized additive (mixed-effects) modeling by analyzing an articulatory dataset contrasting the pronunciation of L1 and L2 speakers of English. With respect to our research question, we have shown that while native English speakers seem to more clearly distinguish /h/ from /t/, there is insufficient evidence (at least when analyzing a single sensor in a single dimension) to conclude that the distinction made by non-native Dutch (highly

---

**Fig. 23.** Difference smooths of model ffmc1 for the words that have a word-final contrast. The left graph shows the difference between the /h/-words and /t/-words for the English speakers. The right graph shows how this difference needs to change to obtain the difference between the /h/-words and /t/-words for the Dutch speakers. In all graphs, the pointwise 95%-confidence intervals are visualized by shaded bands. The patterns in the right graph is (clearly) not significant.
educated) speakers is different from that of native English speakers. By using generalized additive modeling, we were able to analyze all dynamic data, and did not have to average over time or select a specific time point. The analysis allowed us to assess how the specific non-linear tongue movement patterns varied depending on the speaker group, while simultaneously taking all dependencies in our data into account. Wieling, Veenstra, Adank, and Tiede (2017) evaluated the acoustic recordings underlying this dataset and showed that while for Dutch speakers (as opposed to English speakers) /t/-words (e.g., ‘tenth’) were significantly more often recognized (by a native Dutch listener) as /f/-words (e.g., ‘tent’), almost 70% was still correctly recognized (compared to 88% for the native English speakers). An automatic (British English) speech recognition system confirmed this pattern of results. Consequently, with respect to prominent speech learning models (Best, 1995; Flege, 1995), Dutch speakers, at least if highly educated, do not appear to have completely merged the two sounds.

With respect to the actual modeling, we have used the R package mgcv (Wood 2011, 2017) for model fitting, and the R package itsadug (van Rij et al., 2017) for visualizing most of the resulting patterns. We have shown how potentially non-linear patterns may be modeled by smooths, and that the pre-specified basis dimension limits the maximum complexity of the smooths. We have further discussed how an informed choice can be made about how to select the best model given the data. Three approaches were illustrated: model comparison, using ordered factor difference smooths, and using binary difference smooths. Model comparison involves fitting two models and requires extensive computation due to the necessity of fitting using maximum likelihood estimation. By contrast, the latter two approaches are more efficient, as they evaluate whether the additional complexity (i.e. the distinction between two groups or categories) is necessary by directly modeling a difference smooth. The binary difference smooth model evaluates whether the combined constant and non-linear difference between the two categories is necessary, whereas the ordered factor difference smooth model separately assesses the necessity of including the constant and non-linear difference.

We would also like to emphasize that comparing the significance of two smooths does not allow any conclusion about
these patterns being significantly different or not. Furthermore, while it is essential to visualize the (differences between) smooths in order to interpret the results, deciding if a more complex model is warranted should also consist of a more formal assessment (i.e. using one of the three approaches listed above). For example, while the visualization of the binary difference smooth in Fig. 21 (right) might suggest a significant difference, this was not supported by any of the more formal approaches.

We have also observed how dependencies associated with subjects and items may be modeled by including random intercepts, random (linear) slopes, and, most importantly, factor smooths which are able to model non-linear random effects. In addition, we discussed how another type of dependency, autocorrelation, may be alleviated via the \( \rho \) parameter of the \texttt{mgcv} function \texttt{bam}. Besides modeling one-dimensional patterns, we have modeled two-dimensional patterns using a tensor product (see Wieling et al., 2014 for a tensor product involving more than two numerical variables), and we have decomposed the tensor product into separate smooths for each variable, as well as a separate tensor product interaction.

Finally, we have discussed aspects of model criticism and illustrated an example of fitting a non-Gaussian scaled-\( t \) model. Especially here, discretization and parallelization were important in reducing computation time to a manageable duration.

While the generalized additive modeling approach is certainly powerful and flexible, it is not perfect. At present, no correlation structure can be incorporated in the linear random effects structure, at least not when using the function \texttt{gam} or \texttt{bam}. Consequently, if the patterns in the data are linear, or can be adequately represented by simple polynomials, it might be preferable to use growth curve analysis (Mirman, 2014; Mirman et al., 2008) or linear mixed-effects regression modeling via the \texttt{R lme4} package (see also Winter & Wieling, 2016 for a discussion of both techniques). Furthermore, if heteroscedasticity and dependencies in the data (e.g., autocorrelation) cannot be adequately coped with, it may be useful to investigate whether sparse functional linear mixed modeling (Cederbaum et al., 2016; Pouplier et al., 2017) is a more suitable analysis approach. Unfortunately, sparse functional linear mixed modeling does not allow for the inclusion of random slopes, which are almost always necessary.
7. Conclusion

By providing a hands-on approach, together with the original data and all R commands, readers should be able to replicate the analyses and gain more understanding about the material at hand. Importantly, other studies employing generalized additive modeling by Wieling and others have also made their data and code available (e.g., Meulman et al., 2015; Sóskuthy, 2017; Wieling et al., 2011, 2014, 2016, 2017; Winter & Wieling, 2016), thereby helping other researchers become familiar with this powerful analysis tool.

Acknowledgments

This work is part of the research program "Improving speech learning models and English pronunciation with articulography", which was financed by the Netherlands Organisation for Scientific Research (NWO) via a Veni grant awarded to Martijn Wieling (grant number 275-70-039). I am grateful to Mark Tiede and John Nerbonne, as well as two anonymous reviewers, Martón Sóskuthy and the guest editor Timo B. Roettger for providing comments which have helped to substantially improve the manuscript. Any mistakes in the manuscript remain my own.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.wocn.2018.03.002.

References