A tutorial on Generalised Additive Mixed Effects Models for bilingualism research

Motivation

...

Generalised additive (mixed) models

...

Pre-requisites

This tutorial assumes readers are already familiar with R and have at least some experience fitting linear models, including models with random effects (variably known as mixed, hierarchical, nested, multi-level models. The following packages need to be installed:

- tidyverse
- mgcv
- tidygam

You can find the tutorial code and data here: XXX

Case study 1: U-shaped learning

U-shaped learning

[Lit review]

The data

We have simulated data of learning scores from 200 subjects, taken at 10 time points. A proficiency score was also included for each subject at each time point. The data is intended to simulate a study in which participants perform a learning task and take a proficiency test at 10 consecutive time points. This is what the data looks like.

```
dat1
```

```
# A tibble: 2,200 x 4
   score proficiency subj   time
   <dbl>       <dbl> <fct> <int>
 1  26.7      -1.37  s1        0
 2  26.4      -1.29  s1        1
 3  26.0      -1.21  s1        2
 4  25.7      -1.14  s1        3
 5  25.5      -1.06  s1        4
 6  25.2      -0.983 s1        5
 7  25.0      -0.906 s1        6
 8  24.9      -0.830 s1        7
 9  24.6      -0.753 s1        8
10  24.5      -0.677 s1        9
# i 2,190 more rows
```

The `score` column contains the learning scores, while the `proficiency` column the proficiency scores. The subject ID is given in `subj`. time point 0 to 9 is in `time`. Figure 1 shows the relationship between proficiency and learning scores for individual subjects. From the figure, it is clear that such relationship has a U-shape, by which learning scores intially decrease, plateau, then increase again with higher proficiency.

In the following sections we will analyse this data using GAMMs. For pedagogical purposes, we will first focus on the effect of proficiency scores on learning scores at a single time point, time point 5. Subsequently, we will analyse the entire data, to illustrate how to conduct a time-series analysis.

Modelling a non-linear effect

Let's first focus on how to model a non-linear effect: here, we can look at the effect of the proficiency score on the learning score. As we have seen in Figure 1, the effect is U-shaped, i.e. it is not linear. To simplify things, let's look only at data from time point. We will extend the analysis to also include time point as a predictor later. We can model a non-linear effect of proficiency on the learning score with the following code.

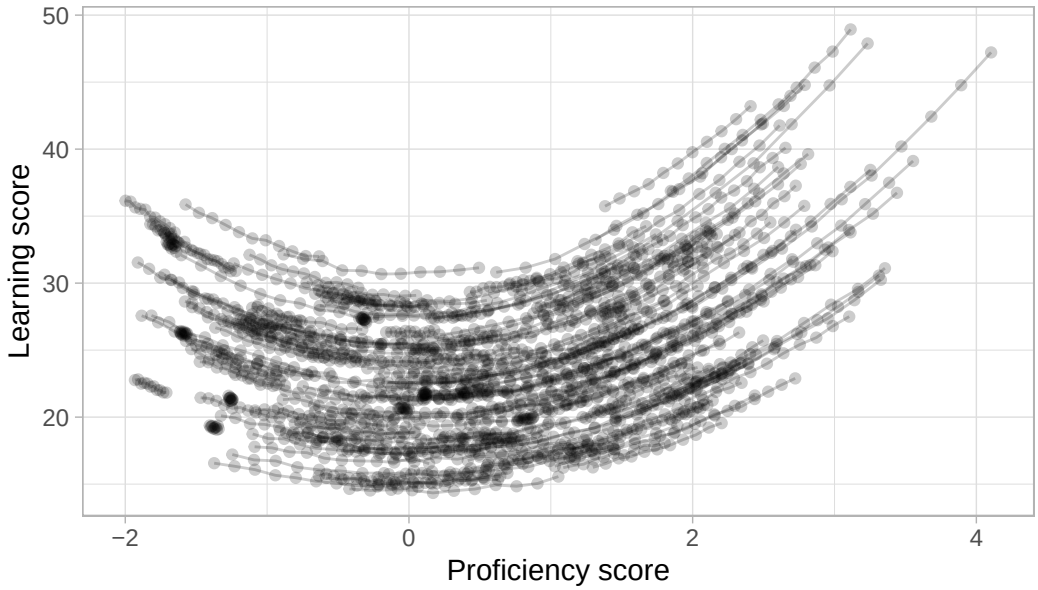


Figure 1: **?(caption)**

```
dat15 <- filter(dat1, time == 5)

# attach the mgcv package
library(mgcv)

# fit the model
gam_1 <- gam(
  score ~ s(proficiency),
  data = dat15
)
```

The formula states that `score`, the outcome variable (also known as the dependent variable) should be modelled as a function of `proficiency`, but we use the `s()` to indicate that we want to estimate a (potentially) non-linear effect. The name of the function, `s`, stands for "smooth term". Smooth terms (aka smoothers) are mathematical objects that allow GAMs to fit non-linear effects. A detailed treatment of smoothers is beyond the scope of this tutorial, so we refer the readers to XXX.

Before looking at the summary of the `gam_1` model, let's plot the predicted effect of proficiency. We will use the tidygam package, which provides users with utility functions that make extracting and plotting predictions from GAMs easier.

```r
# attach tidygam
library(tidygam)

# extract model predictions
gam_1_preds <- predict_gam(gam_1)

# plot predictions
plot(gam_1_preds, series = "proficiency")
```

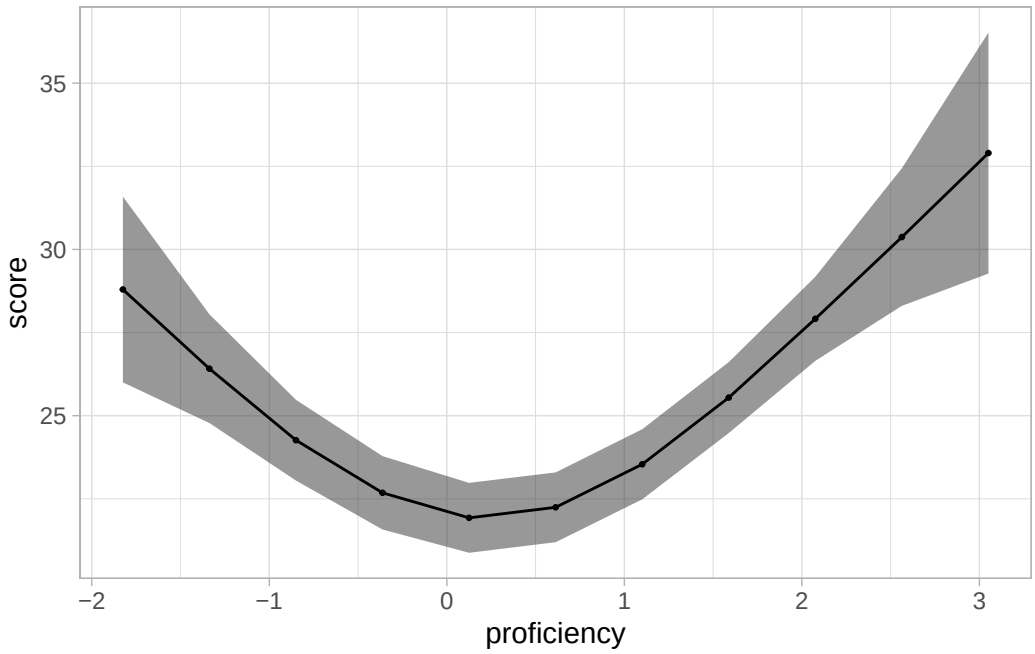


Figure 2: **?(caption)**

Figure 2 is a plot of the predicted effect of proficiency on learning score, based on the `gam_1` model from above. We will look into the details of `predict_gam()`. For now, just notice that the learning score initially decreases with increasing proficiency until about 0.5 and then starts increasing, thus producing the typical U-shaped curve. Let's inspect the model summary now.

```r
summary(gam_1)
```

```
Family: gaussian
Link function: identity
```

```
Formula:
score ~ s(proficiency)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   24.559      0.331   74.19   <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(proficiency) 2.922  3.675 16.46  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.242   Deviance explained = 25.3%
GCV = 22.352  Scale est. = 21.914    n = 200
```

The relevant parts of the summary for the time being are the `Parametric coefficients:` table and the `Approximate significance of smooth terms:` table. The first table contains the estimate of the intercept. This intercept is the same intercept you would get in a linear model: here, the estimate of the intercept is the predicted learning score when proficiency is 0. According to the summary, when proficiency is 0, the learning score is about 25. But what we are really interested in is the effect of proficiency on learning score, rather than the intercept per se.

Information on the effect of proficiency is found in the second table, which contains estimates of the smooth terms. Alas, these estimates don't say much about the effect per se. Rather, they just indicate if the effect is linear or not. More specifically, the `edf` estimate, which stands for Estimated Degrees of Freedom, would be 1 for perfectly linear effects and greater than 1 for non-linear effects. This number tells us nothing about the shape of the effect. The only way to assess this is to plot the model predictions, as we have done above.

In our `gam_1` model, the `edf` for the smooth term over proficiency is 2.9. This is significantly different from 1 (p < 2e-16), thus suggesting that the effect of proficiency is not linear (the `Ref.df`, reference degrees of freedom, and `F` values have the only function of being needed to get a p-value).

If we were to report this model and results, we could write something along the following lines: We fitted a Generalised Additive Model to learning score, with a smooth term over proficiency (to model non-linear effects). According to the model, the effect of proficiency is significantly non-linear ($F = 16.46$, $p < 0.0001$). Based on the prediction plot, we observe that

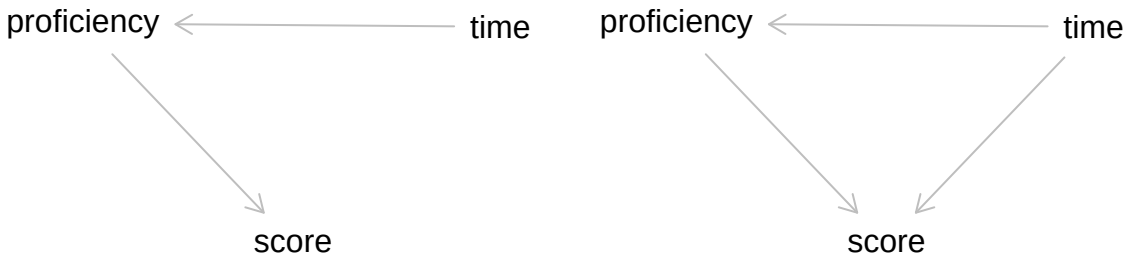
at lower proficiency, learning scores initially decrease and then start increasing again from about proficiency 0.5.

In the following section we will refit the data now including time point (note that this would be the model to fit in the first place and we have fitted a model only with proficiency for pedagogical purposes).

Multiple smooth terms

The data `dat1` contains learning and proficiency scores from 200 subjects, taken at 10 time points. The data was simulated so that proficiency increased with time (at different degrees for different speakers). An interesting question is whether learning scores improve with time independently of proficiency, or if proficiency alone is causing learning scores to improve. We can approach with question by applying a statistical method called causal inference, based on directed acyclic graphs (DAGs) theory (a full treatment of this is beyond the scope of the paper. Readers are referred to XXX).

The DAGs in Figure 3 represent the causal relationships between learning scores, proficiency scores and time point in the two scenarios. In (a), time point affects proficiency and proficiency affects learning scores. In other words, time point has a direct effect on proficiency but not on learning scores; learning scores can be predicted from proficiency alone. In (b), on the other hand, time point affects proficiency as in (a) but also learning scores (and proficiency affects learning scores as in (a)). This means that time point affects learning scores in two ways: through its effect on proficiency and directly through its effect on learning scores.



(a) No direct effect of time on learning scores.

(b) Direct effect of time on learning scores.

Figure 3: Dyrected Acyclic Graphs for time, proficiency and learning score.

DAGs allow us to make causal statements based on statistical results. In this case, when including both time point and proficiency as predictors in a GAM, time should not have an effect on learning score if scenario (a) is correct (while it should have an effect if scenario (b) is correct). Let's model the data.

```
gam_2 <- gam(
  score ~ s(proficiency) + s(time) +
```

```
    s(subj, bs = "re"),
  data = dat1
)
```

In `gam_2` we fit a GAM to learning scores `score` with two predictors: a smooth term over proficiency and a smooth term over time point. We also include random effects to account for data from multiple participants. The syntax for random effects in GAMs is different from the syntax in lme4. With `gam()`, we can specify random effects using smooth terms and the `re` (for Random Effects) basis function. Random intercept are added with the syntax `s(ranint, bs = "re")` and random slopes with the syntax `s(ranint, ranslope, bs = "re")`. Here we just add a random intercept by subject for illustration. Let's inspect the model summary.

```
summary(gam_2)
```

```
Family: gaussian
Link function: identity

Formula:
score ~ s(proficiency) + s(time) + s(subj, bs = "re")

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   24.738      0.441   56.09   <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(proficiency)   8.699   8.976 4638.38  < 2e-16 ***
s(time)          1.000   1.000   20.27 7.36e-06 ***
s(subj)        198.943 200.000 2637.63  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.997   Deviance explained = 99.7%
GCV = 0.10023  Scale est. = 0.090681  n = 2200
```

…

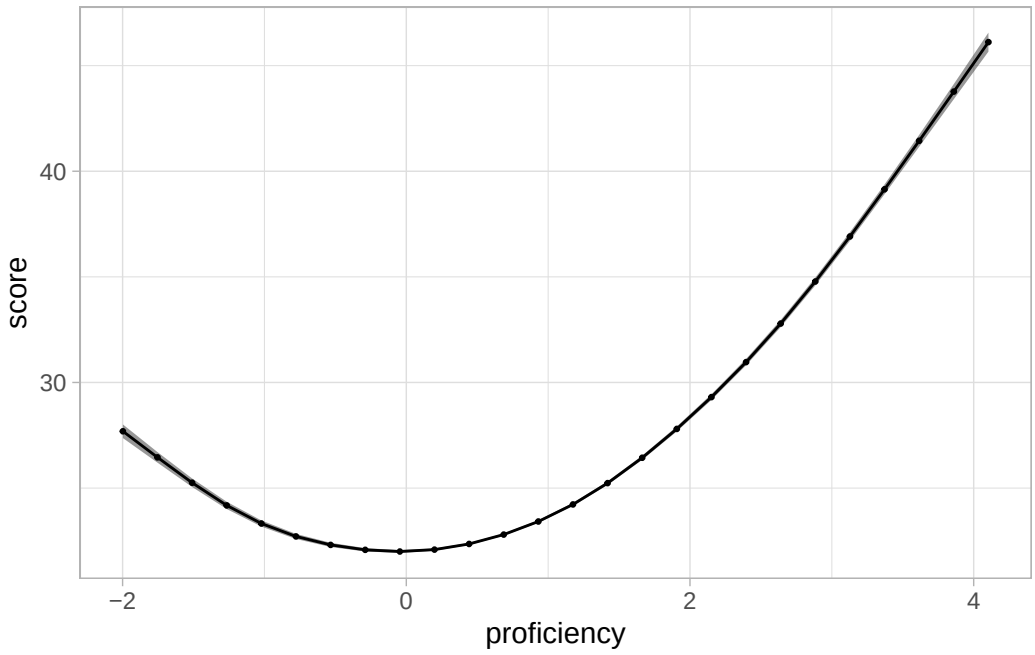
Let's plot the model predictions.

```
gam_2_preds <- predict_gam(
  gam_2, length_out = 25,
  series = "proficiency",
  exclude_terms = c("s(subj)", "s(subj,proficiency)")
)
```

```
Warning in mgcv::predict.gam(model, newdata = pred_grid, se.fit = TRUE, :
non-existent exclude terms requested - ignoring
```

```
Warning: There was 1 warning in `dplyr::mutate()`.
i In argument: `fit = rowSums(dplyr::across())`.
Caused by warning:
! Using `across()` without supplying `.cols` was deprecated in dplyr 1.1.0.
i Please supply `.cols` instead.
```

```
plot(gam_2_preds)
```

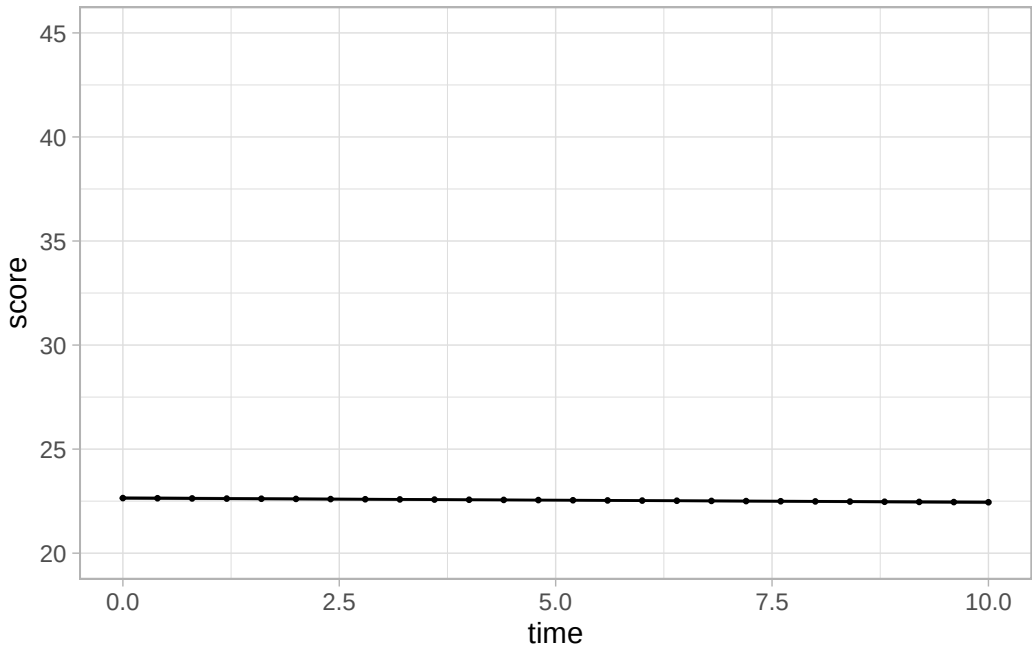



```
gam_2_preds_t <- predict_gam(
  gam_2, length_out = 25,
  series = "time",
```

```
    exclude_terms = c("s(subj)", "s(subj,proficiency)")
  )
```

```
Warning in mgcv::predict.gam(model, newdata = pred_grid, se.fit = TRUE, :
non-existent exclude terms requested - ignoring
```

```
plot(gam_2_preds_t) +
  ylim(20, 45)
```



Case study 2: simultaneous vs late bilinguals

[LIT REVIEW]

We have simulated phonetic data from three groups of bilinguals: simultaneous English/Spanish bilinguals, beginners late English/Spanish bilinguals and advanced late English/Spanish bilinguals. The data contains the Euclidean Distance (EuD) of three Spanish vowels /a, i, u/ from the vowel space centroid, taken from word-final non-stressed vowels, along 9 equidistant time points. The measure is a good proxy for vowel reduction phenomena, which is to be expected in late bilinguals.

```
dat2
```

```
# A tibble: 9,720 x 7
   subj  group vowel   rep   eud timep id
   <fct> <chr> <chr> <int> <dbl> <int> <chr>
 1 es_1  es    i         1  1.85     1 es.1.1
 2 es_1  es    i         1  1.86     2 es.1.1
 3 es_1  es    i         1  1.83     3 es.1.1
 4 es_1  es    i         1  1.87     4 es.1.1
 5 es_1  es    i         1  1.89     5 es.1.1
 6 es_1  es    i         1  1.81     6 es.1.1
 7 es_1  es    i         1  1.91     7 es.1.1
 8 es_1  es    i         1  1.91     8 es.1.1
 9 es_1  es    i         1  1.85     9 es.1.1
10 es_1  es    i         2  2.33     1 es.1.2
# i 9,710 more rows
```

This is what the data looks like. The top row reports the synthetic data of simultaneous English/Spanish bilinguals (`es`). It can be observed that the Euclidean distance is quite stable across the duration of the vowel for all three vowels and away from 0 (which corresponds to a mid-central vowel), indicating no reduction takes place (as it is to be expected for Spanish). On the other hand, in the second and third row, we can see some vowel reduction at play: for /a/, beginners late bilinguals (`es_0`) completely reduce the vowel to a mid-central vowel, while the advanced late bilinguals (`es_1`) reduce it to a lesser degree, but it is still not quite a full Spanish /a/; /i/ and /u/ are produced with a diphthongal quality, where the first part of the vowel is quite reduced. Finally, comparing beginners and advanced bilinguals shows that the latter have less reduction, as expected by increasing proficiency in Spanish.

We can use a GAM to model Euclidean distance along time point in the different vowels and different bilingual groups. Fitting smooth terms for different levels of categorical predictors can be achieved by specifying the categorical predictor as the value of the `by` argument inside the smooth term function `s()`: for example `s(timep, by = vowel)` would ensure a non-linear effect of time point is estimated for each vowel. In our data, we want to model the interaction of vowel (/a, i, u/) and group (`es, es_0, es_1`). Unfortunately, classical interactions cannot be model in GAMs (because of the "additive" aspect of the model; interactions require product operations).

To obviate this limitation, we can simply construct a new predictor with the combination of vowel and group and use that as the `by`-variable in the smooth terms. Due to how smooth terms are constructed when you include a `by`-variable, it is necessary to also include the variable as a parametric term (i.e. a classical linear term).

```r
dat2 <- dat2 %>%
  mutate(
    vowel = as.factor(vowel),
```

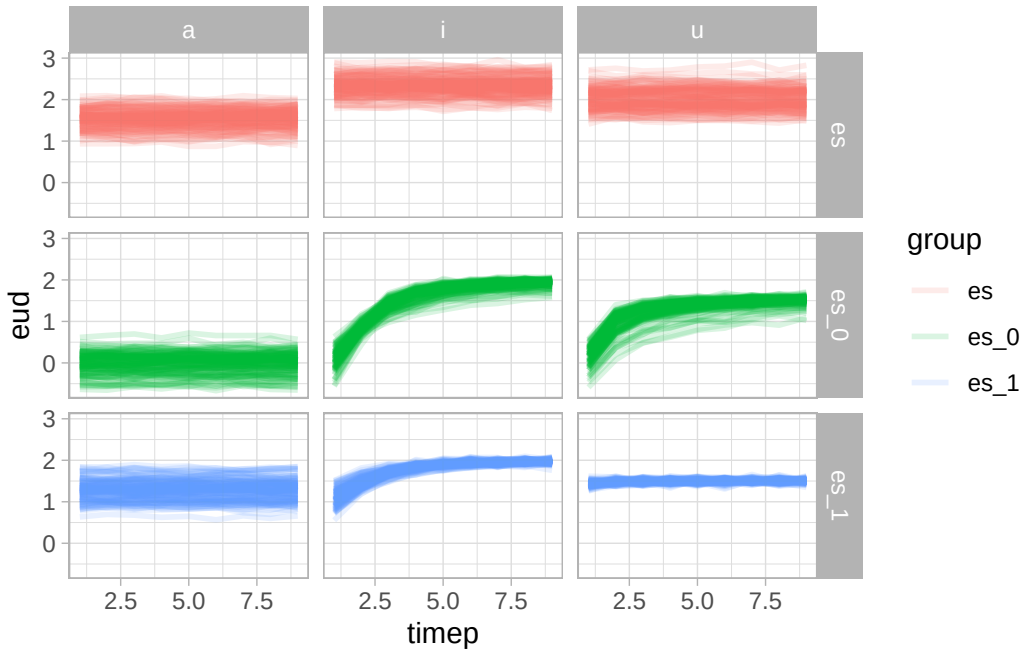


Figure 4: **?(caption)**

```
    group_vowel = interaction(group, vowel),
  )
```

```
gam_3 <- gam(
  eud ~
    # Parametric term for group_vowel
    group_vowel +
    # Smooth term with group_vowel by-variable
    s(timep, by = group_vowel, k = 9) +
    # Random factor smooths
    s(timep, subj, bs = "fs", m = 1),
  data = dat2
)
```

```
summary(gam_3)
```

```
Family: gaussian
Link function: identity
```

```
Formula:
eud ~ group_vowel + s(timep, by = group_vowel, k = 9) + s(timep,
    subj, bs = "fs", m = 1)

Parametric coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        1.533926   0.023247  65.983  < 2e-16 ***
group_voweles_0.a -1.539712   0.032877 -46.833  < 2e-16 ***
group_voweles_1.a -0.255481   0.032877  -7.771 8.59e-15 ***
group_voweles.i    0.791406   0.007385 107.157  < 2e-16 ***
group_voweles_0.i -0.052418   0.032877  -1.594    0.111
group_voweles_1.i  0.222520   0.032877   6.768 1.38e-11 ***
group_voweles.u    0.485860   0.007385  65.786  < 2e-16 ***
group_voweles_0.u -0.318526   0.032877  -9.689  < 2e-16 ***
group_voweles_1.u -0.044394   0.032877  -1.350    0.177
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                             edf  Ref.df        F p-value
s(timep):group_voweles.a   1.000   1.000    0.112 0.73770
s(timep):group_voweles_0.a 1.000   1.000    0.117 0.73226
s(timep):group_voweles_1.a 1.000   1.000    0.023 0.87934
s(timep):group_voweles.i   1.000   1.000    0.021 0.88372
s(timep):group_voweles_0.i 6.221   7.201 1770.941 < 2e-16 ***
s(timep):group_voweles_1.i 5.245   6.298  468.819 < 2e-16 ***
s(timep):group_voweles.u   1.000   1.000    0.009 0.92287
s(timep):group_voweles_0.u 7.568   7.945  777.603 < 2e-16 ***
s(timep):group_voweles_1.u 2.484   3.086    4.621 0.00269 **
s(timep,subj)             56.008 534.000    6.025 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.939   Deviance explained =   94%
GCV = 0.029735  Scale est. = 0.029455  n = 9720
```

Let's extract the model predictions and inspect them.

```
gam_3_preds <- predict_gam(gam_3, exclude_terms = "s(timep,subj)")
gam_3_preds
```

```
# A tibble: 99 x 6
```

```
   group_vowel timep      eud      se  lower_ci upper_ci
   <fct>       <dbl>    <dbl>   <dbl>     <dbl>    <dbl>
 1 es.i          1    2.32    0.0155   2.29       2.35
 2 es.u          1    2.02    0.0155   1.99       2.05
 3 es.a          1    1.53    0.00809  1.52       1.55
 4 es_0.i        1    0.0930  0.0473   0.000345   0.186
 5 es_0.u        1    0.154   0.0476   0.0611     0.248
 6 es_0.a        1   -0.00855 0.0410  -0.0888     0.0717
 7 es_1.i        1    1.10    0.0470   1.00       1.19
 8 es_1.u        1    1.45    0.0447   1.36       1.53
 9 es_1.a        1    1.28    0.0410   1.20       1.36
10 es.i          1.8  2.32    0.0139   2.30       2.35
# i 89 more rows
```

The function `predict_gam()` automatically samples 10 equidistant points from numeric variables (like `timep`, based on the default `length_out = 10` argument) and all levels in categorical variables (like `group_vowel`). For each combination of the sampled points and levels, the function returns the value of the outcome (`eud`) and the standard error (`se`). The lower (`lower_ci`) and upper (`upper_ci`) boundaries of the 95% Confidence Interval are also returned.

Furthermore, `predict_gam()` returns an object of class `tidygam` which can be plotted with `plot()` (similarly to how `ggpredict()` from the ggeffects package works, XXX). However, we might want to do some processing of the predictions before plotting: in particular, we might want to split the `group_vowel` column into the two original variables, `group` and `vowel`. We can achieve this straight from the function. And we might also want to extract more than 10 time points, to get a smoother curve.

```
gam_3_preds_2 <- predict_gam(
  gam_3,
  length_out = 25,
  exclude_terms = "s(timep,subj)",
  separate = list(group_vowel = c("group", "vowel"))
)
gam_3_preds_2
```

```
# A tibble: 234 x 7
   group vowel timep      eud      se  lower_ci upper_ci
   <chr> <chr> <dbl>    <dbl>   <dbl>     <dbl>    <dbl>
 1 es    i       1    2.32    0.0155   2.29       2.35
 2 es    u       1    2.02    0.0155   1.99       2.05
 3 es    a       1    1.53    0.00809  1.52       1.55
 4 es_0  i       1    0.0930  0.0473   0.000345   0.186
```

```
 5 es_0  u      1     0.154   0.0476   0.0611     0.248
 6 es_0  a      1    -0.00855 0.0410  -0.0888     0.0717
 7 es_1  i      1     1.10    0.0470   1.00       1.19
 8 es_1  u      1     1.45    0.0447   1.36       1.53
 9 es_1  a      1     1.28    0.0410   1.20       1.36
10 es    i      1.32  2.32    0.0148   2.30       2.35
# i 224 more rows
```

The syntax `list(group_vowel = c("group", "vowel"))` instructs the function to split the `group_vowel` column into two columns, `group` and `vowel`. Splitting is done based on the `sep_by` argument, which is `.` by default. Now we can plot the predictions.

```
gam_3_preds_2 %>%
  plot(series = "timep", comparison = "group") +
  facet_grid(cols = vars(vowel))
```

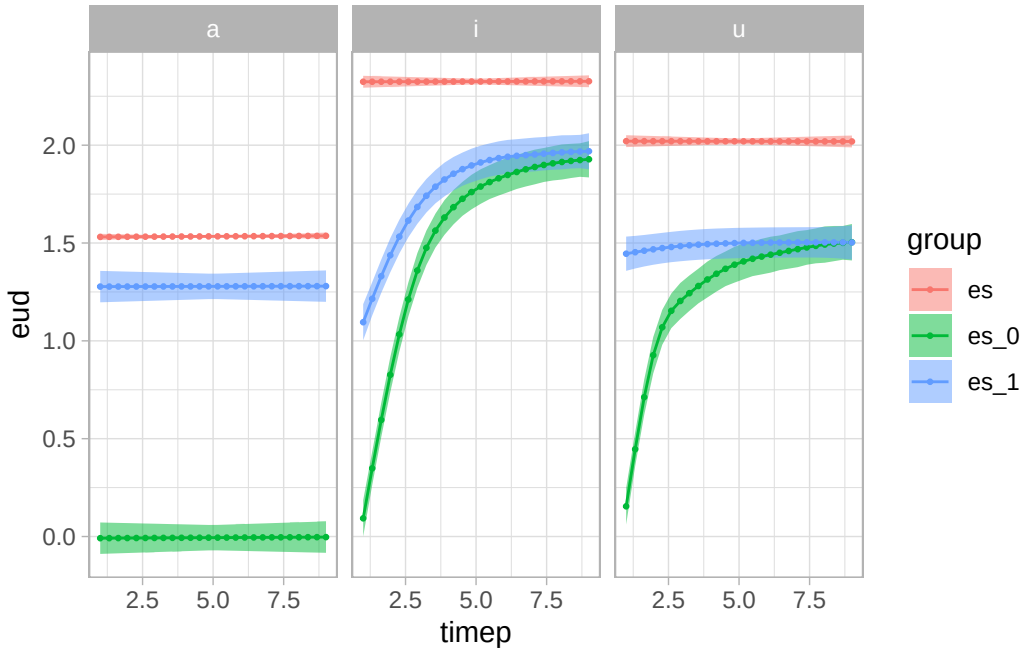


Figure 5: Predicted Euclidean distance along the duration of the vowel, for /a, i, u/ in simultaneous bilinguals (es), beginners late bilinguals (es_0) and advanced late bilinguals (es_1)