Workshop Speech Prosody. Part III: Trees for Smoothing Splines, or GAM trees

Marjolein Fokkema

Context and challenges

It likely goes without saying that smoothing or penalized splines are a powerful tool. Many would argue that with parametric (or unpenalized) spline models, "choosing the optimal number and positions of knots is a complex task" (e.g., Eilers and Marx, 1996). Others argue that "simple unpenalized regression splines work as well as penalized splines that use more parameters" (Chen & Harrell, 2019).

In Part II, it was shown how *parametric* splines can be fitted with GLMM trees. Yet, correct specification of the spline bases for parametric estimation may be challenging, while it may also be critical for accurate recovery of the tree structure. Much of the popularity of smoothing splines may be due to their relatively low dependence on (correct) user specification of the spline bases, which may also be beneficial for subgroup detection. A current challenge is that smoothing splines have a heavy computational burden, which worsens if we want to perform subgroup detection.

The methods discussed this far employ model scores, which are partial derivatives of parameter estimates. For parametric models, these are often easy to compute. Yet, for semi-parametric models, and even mixed-effects models, these are difficult to compute. Often, these scores or partial derivatives arise as a by-product of model estimation and can simply be extracted from a fitting model and used to test for parameter stability. Yet, packages such as **`mgcv`** and **`lme4`** use a penalized iteratively re-weighted least squares (PIRLS) algorithm. PIRLS indirectly maximizes the marginal likelihood, that conditions out the random effects (in **`lme4`**) and/or smoothing parameter (in **`lme4`**). This provides accurate and fast computation. But to obtain the scores, they have to be computed afterwards, which brings a very heavy computation load.

To make it possible to partition based on smoothing spline models, package **`gamtree`** critically uses packages **`gamm4`** for the spline fitting, **`merDeriv`** (Wang et al., 2018, 2022) for obtaining the scores and **`partykit`** for the partitioning. It is a slow but feasible procedure, and our initial results indicate it is the only currently possible way obtain valid subgroup detection from smoothing-spline models.

Fitting a smoothing-spline or GAM tree

Where `splinetree` fits an unpenalized or parametric spline model, `gamtree` fits a smoothing spline model. Functionality is currently limited; only 1 smoothing spline term can currently be specified, and no random effects or global part of the model can be specified (this will be supported in future versions). In order to account for the nested data structure, the `cluster` argument can be employed to account for the correlated nature of the data. For those familiar with multilevel models, this resembles a GEE-type approach, as compared to a mixed-effects modeling approach.

We use the same dataset from Wieling (2018) as in the previous example:

```
load("full2.rda")
summary(full)
```

```
##        Speaker       Lang           Word       Sound         Loc
##  VENI_EN_10:  4302   EN:71152   thighs : 7199   T :62428   Final:62052
##  VENI_EN_19:  3867   NL:55025   tongs  : 7127   TH:63749   Init :64125
##  VENI_EN_17:  3860              fort   : 7000
##  VENI_EN_21:  3739              ties   : 6840
##  VENI_EN_2 :  3717              forth  : 6727
##  VENI_EN_11:  3714              thongs : 6640
##  (Other)   :102978              (Other):84644
##      Trial            Time              Pos               Pos01
##  355    :   922   Min.   :0.0000   Min.   :-6.7536   Min.   :-0.9791
##  317    :   887   1st Qu.:0.2466   1st Qu.:-0.3441   1st Qu.: 0.4235
##  300    :   883   Median :0.5000   Median : 0.3214   Median : 0.5582
##  305    :   864   Mean   :0.5000   Mean   : 0.3029   Mean   : 0.5555
##  301    :   843   3rd Qu.:0.7534   3rd Qu.: 1.0126   3rd Qu.: 0.6983
##  137    :   816   Max.   :1.0000   Max.   : 5.5685   Max.   : 1.6425
##  (Other):120962
##        xs                  xt
##  Min.   :-2.656455   Min.   :-2.99309
##  1st Qu.:-0.564698   1st Qu.:-0.72922
##  Median :-0.094659   Median :-0.02276
##  Mean   : 0.009832   Mean   :-0.03114
##  3rd Qu.: 0.704837   3rd Qu.: 0.65035
##  Max.   : 2.286645   Max.   : 2.96587
##
```

```
words <- c("tent","tenth")
dat <- droplevels(full[full$Word %in% words,])
```

```
library("gamtree")
```

We apply function `gamtree` to a subsample of the datatto a sample of data to reduce computation:

```
set.seed(42)
train <- sample(nrow(dat), size = nrow(dat)/2)
test <- (1:nrow(dat))[-train]
gt <- gamtree(Pos ~ s(Time) | Word + Lang + xt + xs, data = dat[train, ],
              tree_ctrl = list(minsize = (78/2)*10, verbose = TRUE), cluster = Trial)
```

Note that for function `gamtree`, unlike functions `(g)lmertree` and `splinetree`, there are 2 instead of 3 vertical bars in the model formula. There is no 'global' part of the model, there is only the node-specific smoothing splines.

Note that for function `gamtree`, unlike functions `(g)lmertree` and `splinetree`, arguments passed to the tree fitting method (such as `minsize`) must be supplied indirectly, through the `tree_ctrl` argument. See `?gamtree` for more information.

The fraction of data used can be increased by decreasing the value of 2. This will yield a smaller training sample, faster computation, but also lower power to detect splits.

I specified `verbose = TRUE`, so model-fitting progress is printed to the command line, which is helpful with computationally intensive procedures:

```
-- Node 1 ---------------------------------
Number of observations: 6419

Parameter instability tests:
                  Word        Lang        xt        xs
statistic 3.664566e+01 18.239390896 9.0737169 8.0189338
p.value   2.187296e-07  0.001570277 0.8075415 0.9132161

Best splitting variable: Word
Perform split? yes
Selected split: tent | tenth

-- Node 2 ---------------------------------
Number of observations: 3171

Parameter instability tests:
          Word     Lang       xt        xs
statistic    0 7.429725 5.716734 7.4429399
p.value     NA 0.167803 0.982616 0.8755934

Best splitting variable: Lang
Perform split? no

-- Node 3 ---------------------------------
Number of observations: 3248

Parameter instability tests:
          Word        Lang        xt        xs
statistic    0 11.53103154 5.7293237 6.6883481
p.value     NA  0.02727358 0.9830592 0.9395335

Best splitting variable: Lang
Perform split? yes
Selected split: EN | NL

-- Node 4 ---------------------------------
Number of observations: 1895

Parameter instability tests:
          Word Lang        xt        xs
statistic    0    0 6.6126884 4.0047402
p.value     NA   NA 0.7569283 0.9822791

Best splitting variable: xt
Perform split? no

-- Node 5 ---------------------------------
Number of observations: 1353

Parameter instability tests:
          Word Lang        xt        xs
statistic    0    0 2.2878959 5.7623263
p.value     NA   NA 0.9992052 0.7693788
```

```
Best splitting variable: xs
Perform split? no
```

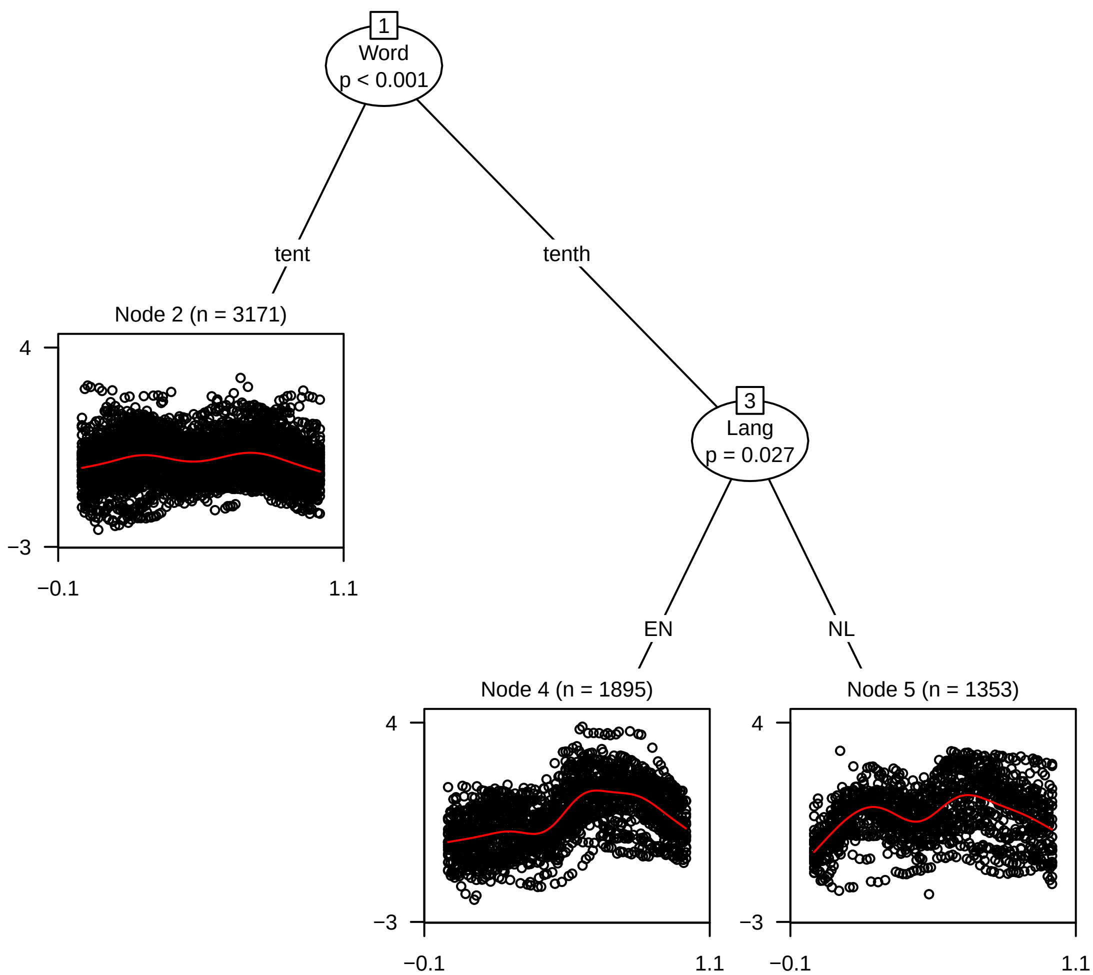
We see that the parameter stability tests for the noise variables are very well behaved. Significant p-values are only observed for the true predictors `Word` and `Lang`.

If you could not repeat the analyses, you can download the fitted tree from the GitHub repository and load it into **R** as follows:

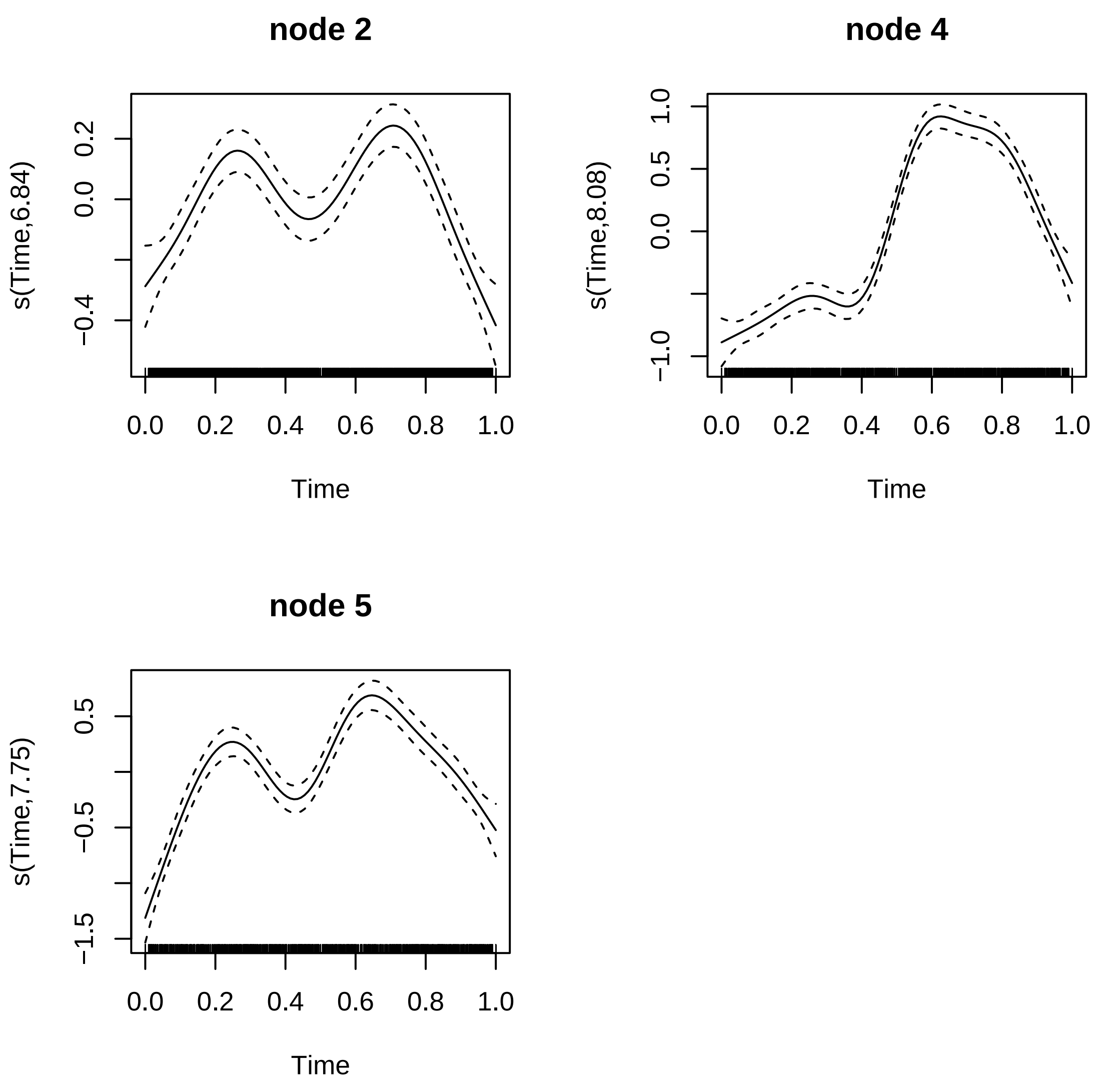
```
load(file = "gamtree.rda")
```

We can plot the resulting tree, as well as the node-specific models. But beware that the confidence bands are overly optimistic, because they do *not* account for the searching of the tree structure!

```
plot(gt, which = "tree", treeplot_ctrl = list(gp = gpar(cex = .7)))
```



```r
par(mfrow = c(2, 2))
plot(gt, which = "terms")
```



From exploration to confirmation: Sample splitting can be beneficial!

Although reducing the number of training observations reduces the power to detect differences and subgroups, it brings a striking advantage: The part of the data (test data) that has not been used for fitting the tree can be used to refit a GAM with the subgroup structure found by the tree. This (re)fitted GAM will provide valid (a.k.a. 'honest') model estimates and hypothesis tests (e.g., Athey & Imbens, 2015). The idea is to estimate the unknown subgroup structure (i.e., exploration) on one part of the data, and to test the significance of effects (i.e., confirmation) on another part of the data.

We can extract node memberships for new data using the `predict` method (by specifying `type = "response"`, we could obtain predicted values instead of predicted nodes):

```r
test_dat <- dat[test, ]
test_dat$nodes <- factor(predict(gt, newdata = dat[test, ], type = "node"))
```

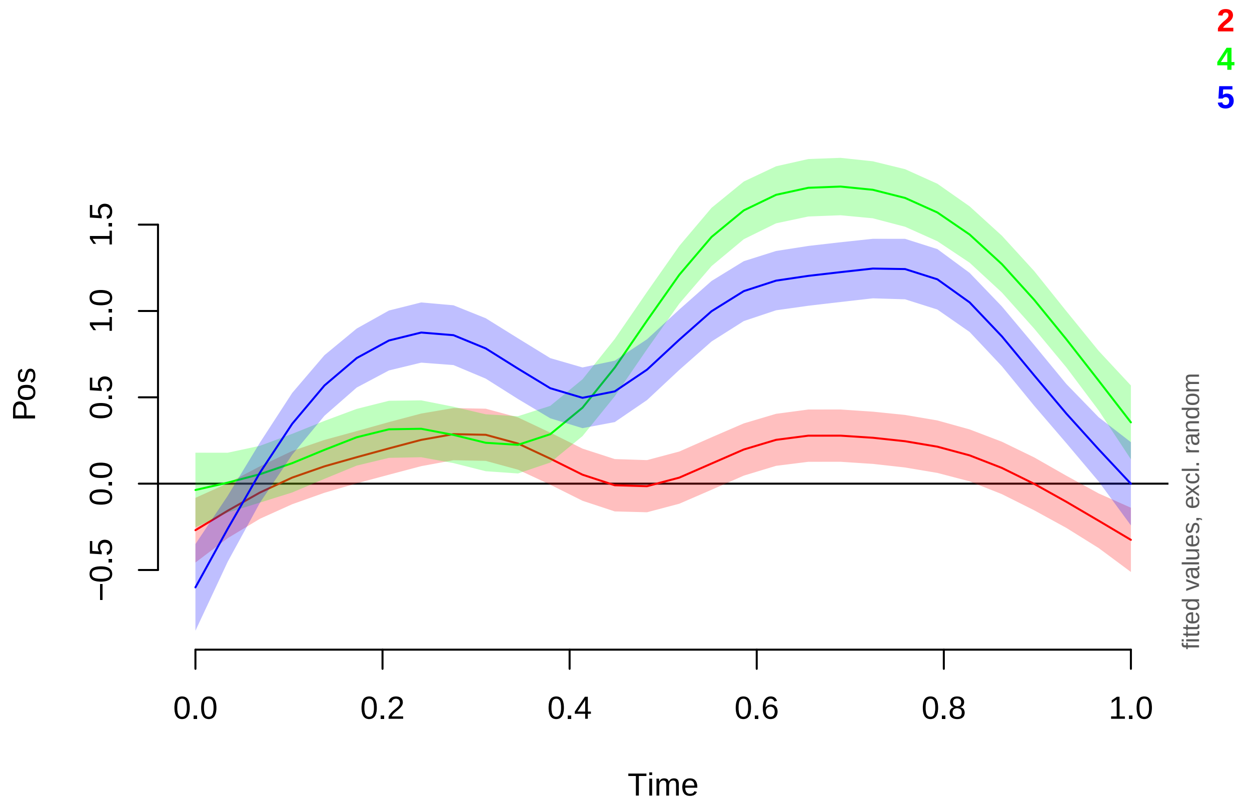
For the latter part, familiar techniques such as those described by Wieling (2018) and plotting methods such as implemented in package **itsadug** can be used. We could use the original indicators `Word` and `Lang`, but we can also incorporate the newly derived subgroups in the `by` argument of function `s`:

```r
library("mgcv")
library("itsadug") ## for plot_smooth function
test_m <- bam(Pos ~ nodes + s(Time, by=nodes) + s(Speaker, bs="re"),
              data = test_dat)
summary(test_m)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Pos ~ nodes + s(Time, by = nodes) + s(Speaker, bs = "re")
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.10223    0.06969   1.467    0.142
## nodes4       0.72349    0.02444  29.603   <2e-16 ***
## nodes5       0.60786    0.02807  21.654   <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):nodes2  7.481  8.429  18.63  <2e-16 ***
## s(Time):nodes4  7.843  8.653 151.71  <2e-16 ***
## s(Time):nodes5  7.868  8.667  53.77  <2e-16 ***
## s(Speaker)     40.197 41.000  44.97  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.445   Deviance explained = 45.1%
## fREML = 7254.4  Scale est. = 0.53813   n = 6420
```

```r
plot_smooth(test_m, view = "Time", plot_all = "nodes", rug = FALSE,
            rm.ranef = TRUE)
```

```
## Summary:
##  * nodes : factor; set to the value(s): 2, 4, 5.
##  * Time : numeric predictor; with 30 values ranging from 0.000000 to 1.000000.
##  * Speaker : factor; set to the value(s): VENI_NL_5. (Might be canceled as random effect, check belo
##  * NOTE : The following random effects columns are canceled: s(Speaker)
##
```



Although we used a different set of observations, and we 'did not know' the truly relevant variables in advance, we obtained very similar results and identical conclusions as Wieling (2018): Native English speakers pronounce "Tent" and "Tenth" with clear differences, while Native Dutch speakers do not show such a clear difference. We can easily distinguish Native English speakers from Native Dutch speakers by their pronounciation of "Tenth", but not by their pronounciation of "Tent".

References

Athey, S., & Imbens, G. W. (2015). Machine learning methods for estimating heterogeneous causal effects. *Proceedings of the National Academy of Sciences, 1050*(5), 1-26. https://doi.org/10.1073/pnas.1510489113

Chen, Q., & Harrell Jr, F. E. (2019). Comment: Penalized spline of propensity methods for treatment comparison. *Journal of the American Statistical Association, 114*(525), 28-30. https://doi.org/10.1080/01621459.2018.1537915

Eilers, P. H., & Marx, B. D. (1996). Flexible smoothing with B-splines and penalties. *Statistical Science, 11*(2), 89-121. https://doi.org/10.1214/ss/1038425655

Wang, T., & Merkle, E. C. (2018). **merDeriv**: derivative computations for linear mixed effects models with application to robust standard errors. *Journal of Statistical Software, 87*, 1-16. https://doi.org/10.18637/jss.v087.c01

Wang, T., Graves, B., Rosseel, Y., & Merkle, E. C. (2022). Computation and application of generalized linear mixed model derivatives using **lme4**. *Psychometrika, 87*(3), 1173-1193.