Effects of obstruent voicing on vowel fundamental frequency in Dutch: Supplementary Materials

Hugo Quené & Anne-France Pinget

8 March 2023 and 6 June 2023 and 9 Oct 2023

These supplementary materials provide further details and additional outputs of the GAMMs reported in the following publication:

> Pinget, A.-F. & Quené, H. (2023). Effects of obstruent voicing on vowel fundamental frequency in Dutch. *Journal of the Acoustical Society of America*, 154 (4), 2124–2136. [https://doi.org/10.1121/10.0021070].

Load and clean data

Load prepared data

We start the analysis by loading the pre-processed data set `carrier2B194v2`. This data set was prepared according to script `AFPHQprepare20221123a.Rmd`.

We also set `P50`, the median duration (in ms) of vowel tokens, for later use in figures.

```
load( file="carrier2B194v2.Rdata" ) # object name is carrier2B
P50 <- 97 # ms, lower weights after this point
```

> This dataset has 49342 observations, from 4492 vowel tokens, with 11 observations from each vowel token.

Ignore suspect f0 jumps

As dependent variable we use `measure.c4`, which is the f0 centered to the speaker's centroid, in semitones, and with that centroid based on the first 20 ms of the vowel token.

There are a few observations with very large f0 jumps, defined here as consecutive measurements within the same vowel token with an absolute difference of >6 semitones relative to the previous measurement in the same vowel token. These are likely to have been due to f0 measurement errors, and not to true f0 contours. Because these have resulted in unstable GAMMS (i.e. to overfitting of the smoothed f0 contours in the GAMM), we will ignore these few suspect measurements.

Column `start_event` is a boolean or logical variable, indicating whether this observation is the first f0 observation of a new vowel token (event).

```r
with( carrier2B194v2, diff(measure.c4) ) -> diff.measure.c4
diff.measure.c4 <- c(NA,diff.measure.c4)
which( abs(diff.measure.c4)>6 & !carrier2B194v2$start.event) -> jumps
# manual inspection here: suspect row numbers in carrier2B194v2
f0jumps <- c( 7235:7236,
              15081,
              18239:18240,
              40588:40598,
              41192,
              47868,
              48362:48363 )
length(f0jumps)
```

```
## [1] 20
```

```r
( unique(carrier2B194v2$file[f0jumps]) )
```

```
## [1] BR15_carrier1_pien  GR11_carrier1_pief  GR18_carrier1_bief
## [4] ZH03_carrier1_fieg  ZH04_carrier1_mieg  ZH18_carrier1_bief
## [7] ZH19_carrier1_bief
## 4500 Levels: BR01_carrier1_bief  BR01_carrier1_bieg  ... ZH20_carrier1_viet
```

```r
# rm(diff.measure.c4,jumps)
```

This results in ignoring 20 f0 observations from 7 vowel tokens.

Set color palette

For colour schemes used below, see https://personal.sron.nl/~pault/data/colourschemes.pdf. Here we use a *vibrant* color scheme (Fig.3 in the linked pdf), with warm colors for fricatives and cool colors for plosives. Lighter for voiced and darker for unvoiced consonants.
Colours are specified directly, so as to avoid installing additional packages containing palette definitions, etc.

```r
# make four vibrant colours
cp <- rgb( red  =c(000,000,238,204)/255,
           green=c(119,153,119,051)/255,
           blue =c(187,136,051,017)/255,
           names=c("blue","teal","orange","red") )
cp <- c( "black", cp[c(2,4,1,3)]) # as by consonant levels: m b f p v
```

Model 1: general GAMM over all consonants

```r
require(mgcv)
```

```
## Loading required package: mgcv
```

```
## Loading required package: nlme

## This is mgcv 1.8-42. For overview type 'help("mgcv-package")'.
```

```
require(itsadug)
```

```
## Loading required package: itsadug

## Loading required package: plotfunctions

## Loaded package itsadug 2.4 (see 'help("itsadug")' ).
```

Define GAMM

First we define the ideal full model. This model includes `cons` (the identity of the preceding consonant), smoothed effect of `vowtime`, and the interactions. All terms are allowed to vary over participants (random slopes).

Here we use a different smoothed curve for each consonant, using the `by` argument in the `s` call; see https://www.r-bloggers.com/2017/07/generalized-addictive-models-and-mixed-effects-in-agriculture/ and see Wieling 2018 (J. Phon.).

As suggested by pilot analyses, we use a `scat` distribution (not gaussian but scaled-t distribution) of residuals, with wider tails.

Run GAMM 1

Using suggestions and ideas from Wieling (2018, J Phon), in particular his models `m4` and m5‘ in sections 4.7.3 and 4.7.3.

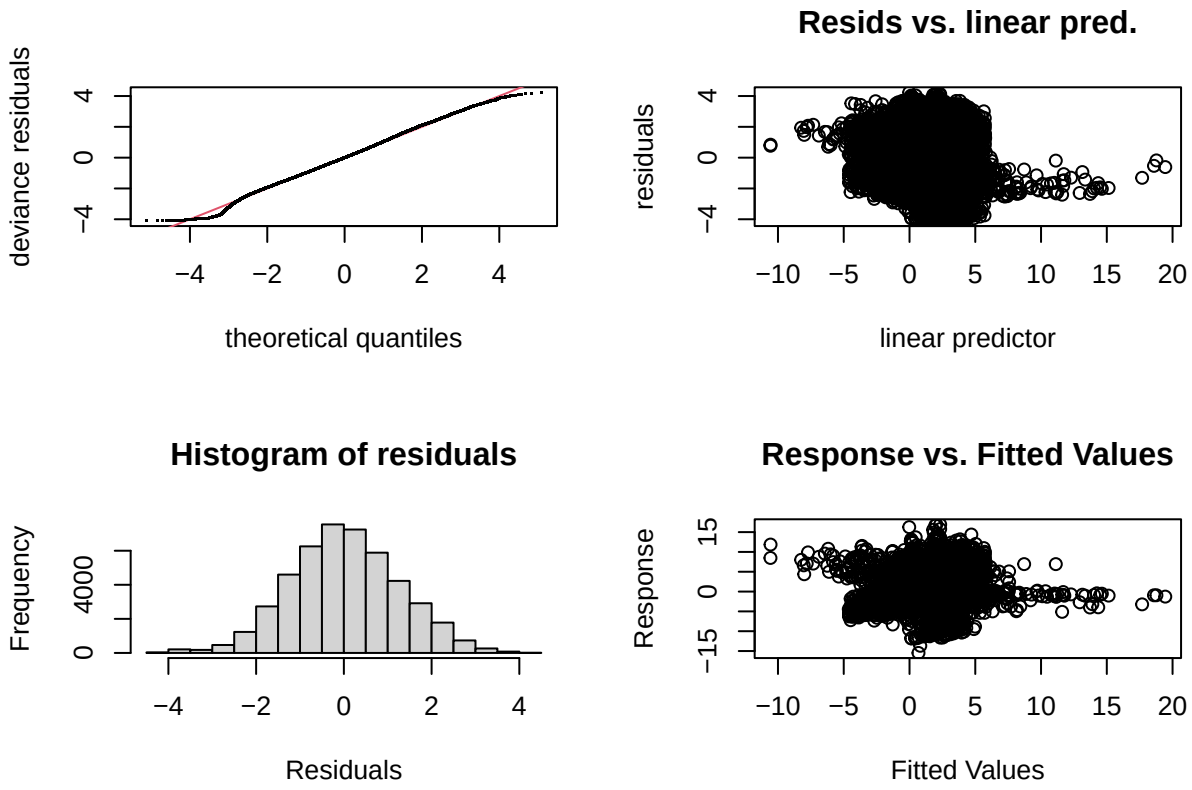
The data set with suffix `v2` is sorted by file and vowtime, and it contains a column `start_event`.

The GAMM is already estimated, and during knitting this R notebook, the GAMM is read from disk.

```
set.seed(20230307) # for replicability
# commented out for knitting
# system.time(
#   m1s4.gamm <- bam( measure.c4 ~ s(vowtime,by=cons,k=20) + cons +
#                         ti(vowtime,ppn,bs="fs",m=1), # re slopes
#                     rho=.85, # from pilot models
#                     gamma=1.6, # Wood, 2017, p.186
#                     AR.start=start.event,
#                     family=scat(min.df=3), # scaled t distrib
#                     # nthreads = 2, # 2 cores
#                     discrete=TRUE,
#                     data=carrier2B194v2[-f0jumps,], # ignoring f0 jumps
#                     weights=wght ) # using weights
# ) # ~42 seconds, no warnings
# save( m1s4.gamm, file="./m1s4gamm20230307.Rdata" ) # for knitting
load( file="./m1s4gamm20230307.Rdata" ) # for knitting
```

Check the first GAMM. . .

```
gam.check(m1s4.gamm) # no problems with edf; k-index<1 but p looks good
```



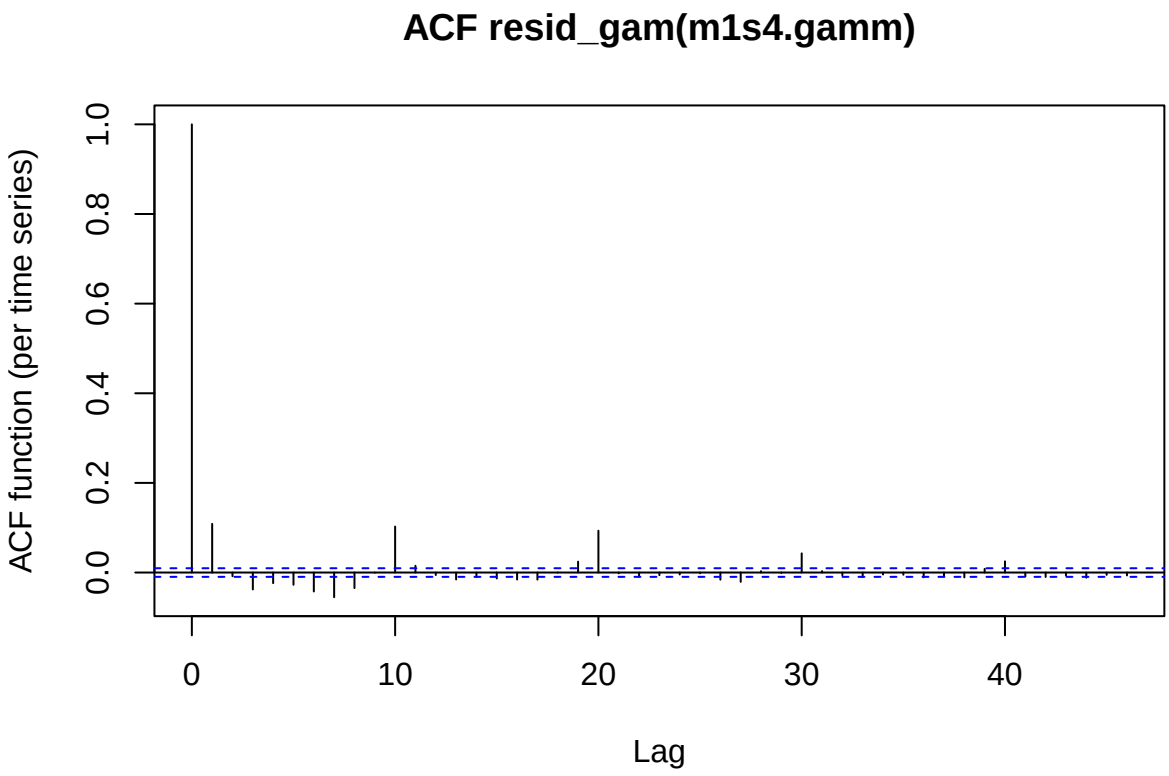

```
##
## Method: fREML   Optimizer: perf chol
## $grad
## [1] -4.943468e-11  1.806217e-10 -9.760992e-11  9.410428e-10  1.225403e-09
## [6] -3.085177e-10  2.078515e-04
##
## $hess
##               [,1]          [,2]          [,3]          [,4]          [,5]
## [1,]  1.091415e+00 -0.0908640401  4.585196e-01 -7.780449e-03 -2.806434e-02
## [2,] -9.086404e-02  1.7491431745  1.050469e-01  8.034416e-02  9.465030e-02
## [3,]  4.585196e-01  0.1050469313  2.438677e+00 -6.484448e-02 -8.825529e-02
## [4,] -7.780449e-03  0.0803441566 -6.484448e-02  7.494096e+00 -6.382863e-02
## [5,] -2.806434e-02  0.0946502985 -8.825529e-02 -6.382863e-02  6.342548e+00
## [6,]  3.470085e-01  0.4937039114 -3.081329e-01 -1.949834e-02  3.516751e-02
## [7,]  8.503064e-05  0.0001455148 -9.594322e-05 -1.462563e-05 -8.011016e-06
##               [,6]          [,7]
## [1,]   0.34700851  8.503064e-05
## [2,]   0.49370391  1.455148e-04
## [3,]  -0.30813286 -9.594322e-05
## [4,]  -0.01949834 -1.462563e-05
## [5,]   0.03516751 -8.011016e-06
## [6,] 170.99469361  1.678771e-02
## [7,]   0.01678771  2.115831e-04
```

```
##
## Model rank =  500 / 500
##
## 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(vowtime):consm  19.00   4.40    1.01    0.73
## s(vowtime):consb  19.00   5.50    1.01    0.71
## s(vowtime):consf  19.00   5.49    1.01    0.71
## s(vowtime):consp  19.00  16.35    1.01    0.77
## s(vowtime):consv  19.00  14.18    1.01    0.72
## ti(vowtime,ppn)  400.00 384.55      NA      NA
```

```
summary( m1s4.gamm )
```

```
##
## Family: Scaled t(3,1.005)
## Link function: identity
##
## Formula:
## measure.c4 ~ s(vowtime, by = cons, k = 20) + cons + ti(vowtime,
##     ppn, bs = "fs", m = 1)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.94527    0.04506  20.978  < 2e-16 ***
## consb       -0.44386    0.05424  -8.184 2.82e-16 ***
## consf        0.59221    0.05929   9.989  < 2e-16 ***
## consp        2.23082    0.05634  39.594  < 2e-16 ***
## consv        0.80073    0.05650  14.172  < 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(vowtime):consm   4.398   5.898  32.672  <2e-16 ***
## s(vowtime):consb   5.502   7.350  25.073  <2e-16 ***
## s(vowtime):consf   5.491   7.317   7.772  <2e-16 ***
## s(vowtime):consp  16.355  17.625 136.271  <2e-16 ***
## s(vowtime):consv  14.177  16.033  54.326  <2e-16 ***
## ti(vowtime,ppn)  384.550 400.000  83.367  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.374   Deviance explained = 38.4%
## fREML =  33984  Scale est. = 1         n = 46392
```

```
m1s4.gamm.acf <- acf_resid(m1s4.gamm) # |rho|<.1, good
```




```
# gamtabs(m1s4.gamm)
```

Rank check is OK.

The smoothing functions require about 5 to 15 knots. (Although this may suggest possible overfitting of the f0 contour, over 11 points per vowel token, models with lower `k` did not pass the check of `edf` being substantially lower than $k - 1$).

Deviance explained is 38.4 % (which is higher than with `family=gaussian` GAMM during piloting).

The residuals look good, and autocorrelation is very low.

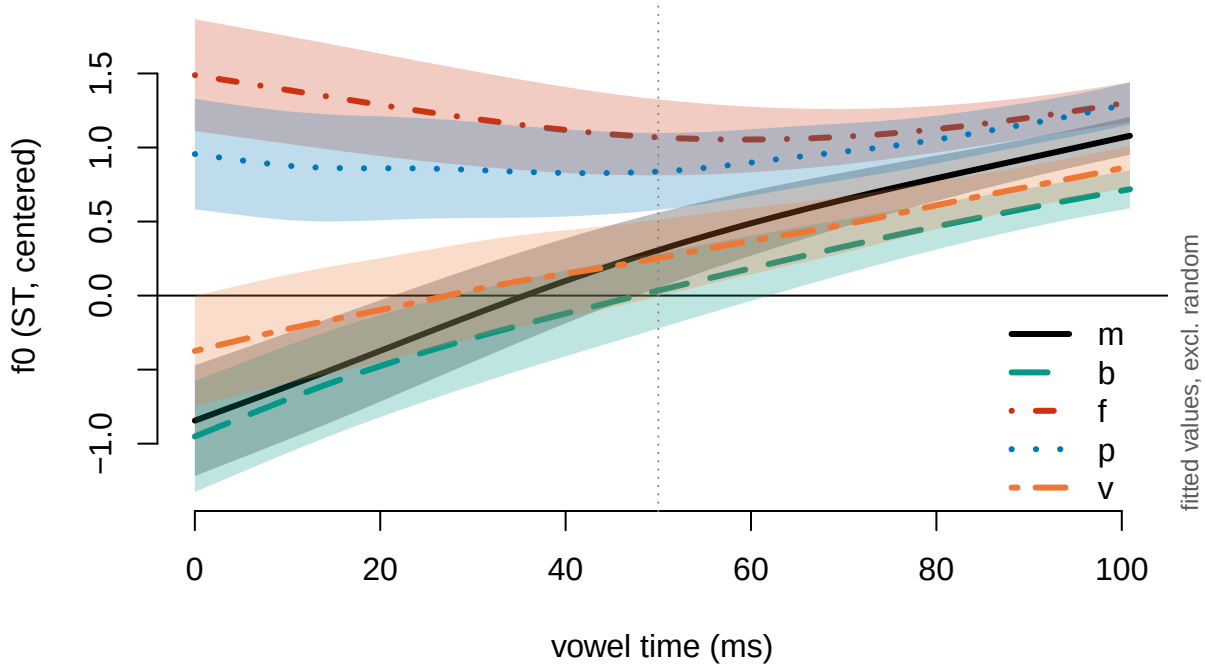
The summary data are reported in Appendix Table I of the manuscript.

Figure 1

```
# load(file="./m1s4.gamm.Rdata")
require(itsadug)
# P50 <- 97 # added 2023.01.11 for completeness; P50 of vowel durations
plot_smooth( m1s4.gamm,
             view="vowtime", plot_all="cons",
             rug=FALSE, col=cp, lwd=3,
             lty=c(1,5,4,3,6), # m b f p v
             legend_plot_all = c(x=50,y=0),
             xlim=c(0,P50*1.04), # allow plot room
             xlab="vowel time (ms)", ylab="f0 (ST, centered)")
```

```
## Summary:
##   * cons : factor; set to the value(s): b, f, m, p, v.
##   * vowtime : numeric predictor; with 30 values ranging from 0.000000 to 100.880000.
##   * ppn : factor; set to the value(s): BR03. (Might be canceled as random effect, check below.)
##   * NOTE : The following random effects columns are canceled: ti(vowtime,ppn)
##
```

```
legend( x=c(85,100), y=c(0,-1.5), col=cp, lwd=3, bty="n",
        lty=c(1,5,4,3,6), # m b f p v
        legend=levels(carrier2B194v2$cons) ) -> fig1legend
abline(v=50,col="grey50",lty=3)
```




```
pdf( file="./m1s4gamm.v20230307.pdf", width=6, height=4)
plot_smooth( m1s4.gamm,
             view="vowtime", plot_all="cons",
             rug=FALSE, col=cp, lwd=3,
             lty=c(1,5,4,3,6), # m b f p v
             xlim=c(0,P50*1.04), # allow plot room
             xlab="vowel time (ms)", ylab="f0 (ST, centered)")
```

```
## Summary:
##   * cons : factor; set to the value(s): b, f, m, p, v.
##   * vowtime : numeric predictor; with 30 values ranging from 0.000000 to 100.880000.
##   * ppn : factor; set to the value(s): BR03. (Might be canceled as random effect, check below.)
##   * NOTE : The following random effects columns are canceled: ti(vowtime,ppn)
##
```

```
legend( x=c(85,100), y=c(0,-1.5), col=cp, lwd=3, bty="n",
        lty=c(1,5,4,3,6), # m b f p v
        legend=levels(carrier2B194v2$cons) ) -> fig1legend
dev.off()
```

```
## pdf
##   2
```

Fig. 1. (Color online). Fitted contours of f0 (in semitones, centered by speaker) over time (in ms from vowel onset) as fitted by nonlinear smooth functions, broken down by preceding consonant, with 95% confidence intervals. Random effects are ignored in the fitted f0 values. [file m1s4gamm.v20230307.pdf].

> Figure 1 shows several interesting patterns. Firstly, compared to the baseline condition of f0 after /m/, the f0 after unvoiced consonants /f, p/ starts relatively higher at vowel onset (by about 2.0 and 1.5 semitones, respectively, see Figure 1). In contrast, secondly, the f0 after voiced consonants /v, b/ does not differ from that after /m/. At vowel onset, the confidence bands of f0 after unvoiced consonants /b, p/ are at least 1 semitone higher than those after voiced /v, f/. Third, these voicing effects of the preceding consonant on f0 are only transitory, and are no longer noticeable after approximately 50 ms in the vowel (when the 95% confidence intervals of /m/ and /p/ start to overlap, see Figure 1).

Report pseudo-R-squared

addition 2023.06.11 suggested by Rev.1

Compare the model above with a simpler model, from which random effects of talkers are dropped entirely. As suggested by Rev.1, we use the `MuMIn` package for this comparison.

```
require(mgcv)
require(itsadug)
set.seed(20230611)
system.time(
  m1s5.gamm <- bam( measure.c4 ~ s(vowtime,by=cons,k=20) + cons,
                    # no random effects! in this null model
                  rho=.85, # from pilot models
                  gamma=1.6, # Wood, 2017, p.186
                  AR.start=start.event,
                  family=scat(min.df=3), # scaled t distrib
                  # nthreads = 2, # 2 cores
                  discrete=TRUE,
                  data=carrier2B194v2[-f0jumps,], # ignoring f0 jumps
                  weights=wght ) # using weights
) # ~3 seconds, no warnings
```

```
##    user  system elapsed
##   0.914   0.063   0.978
```

```
save( m1s5.gamm, file="./m1s5gamm20230611.Rdata" ) # for knitting
# load( file="./m1s5gamm20230611.Rdata" ) # for knitting
require(MuMIn)
```

```
## Loading required package: MuMIn
```

```
citation("MuMIn")
```

```
## To cite package 'MuMIn' in publications use:
##
##   Bartoń K (2023). _MuMIn: Multi-Model Inference_. R package version
##   1.47.5, <https://CRAN.R-project.org/package=MuMIn>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {MuMIn: Multi-Model Inference},
##     author = {Kamil Bartoń},
##     year = {2023},
##     note = {R package version 1.47.5},
##     url = {https://CRAN.R-project.org/package=MuMIn},
##   }
##
## ATTENTION: This citation information has been auto-generated from the
## package DESCRIPTION file and may need manual editing, see
## 'help("citation")'.
```

```
r.squaredLR( m1s4.gamm, null=m1s5.gamm ) # null model without random effects
```

```
## [1] 0.4775785
## attr(,"adj.r.squared")
## [1] 0.5034723
```

A comparison by likelihoods (Bartón, 2023) of the general GAMM and of a null model without any random effects of talkers indicated that in the general GAMM, 48% of the variance in the centered F0 values was due to random effects of talkers (adjusted R2 0.50).

Same model now with relative time points

Rev.3 suggestedd to use as predictor not abs `vowtime` but relative `type_measure`; otherwise the model remains the same as model `m1s4` above.

Note however that with only 11 time points, we need to restrict the smoothing to k<11 knots, here k=10 has been chosen.

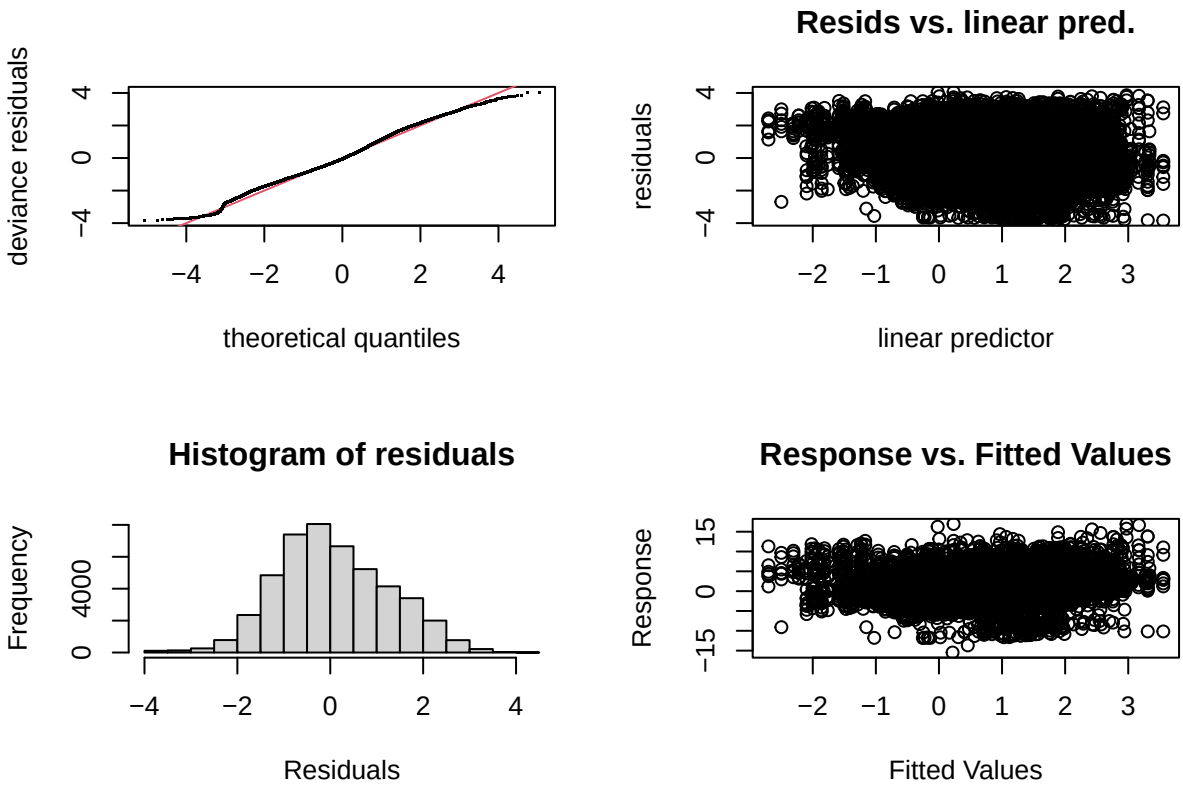
```
require(mgcv)
carrier2B194v2$reltime <- as.integer ( carrier2B194v2$type_measure) *10
# do not save the data set
set.seed(20230611) # for replicability
# commented out for knitting
system.time(
```

```
  m1s6.gamm <- bam( measure.c4 ~ s(reltime,by=cons,k=10) + cons +
                       ti(reltime,ppn,bs="fs",m=1), # re slopes
                    rho=.85, # from pilot models
                    gamma=1.6, # Wood, 2017, p.186
                    AR.start=start.event,
                    family=scat(min.df=3), # scaled t distrib
                    # nthreads = 2, # 2 cores
                    discrete=TRUE,
                    data=carrier2B194v2[-f0jumps,], # ignoring f0 jumps
                    weights=wght ) # using weights
) # ~17 seconds, no warnings
```

```
##    user  system elapsed
##   8.459   0.231   8.692
```

```
save( m1s6.gamm, file="./m1s6gamm20230611.Rdata" ) # for knitting
load( file="./m1s6gamm20230611.Rdata" ) # for knitting
```

```
gam.check(m1s6.gamm) # no problems with edf; k-index<1 but p looks good
```




```
##
## Method: fREML   Optimizer: perf chol
## $grad
## [1] -2.366596e-11 -1.207345e-11 -6.569839e-11 -1.743420e-04  1.612976e-11
```

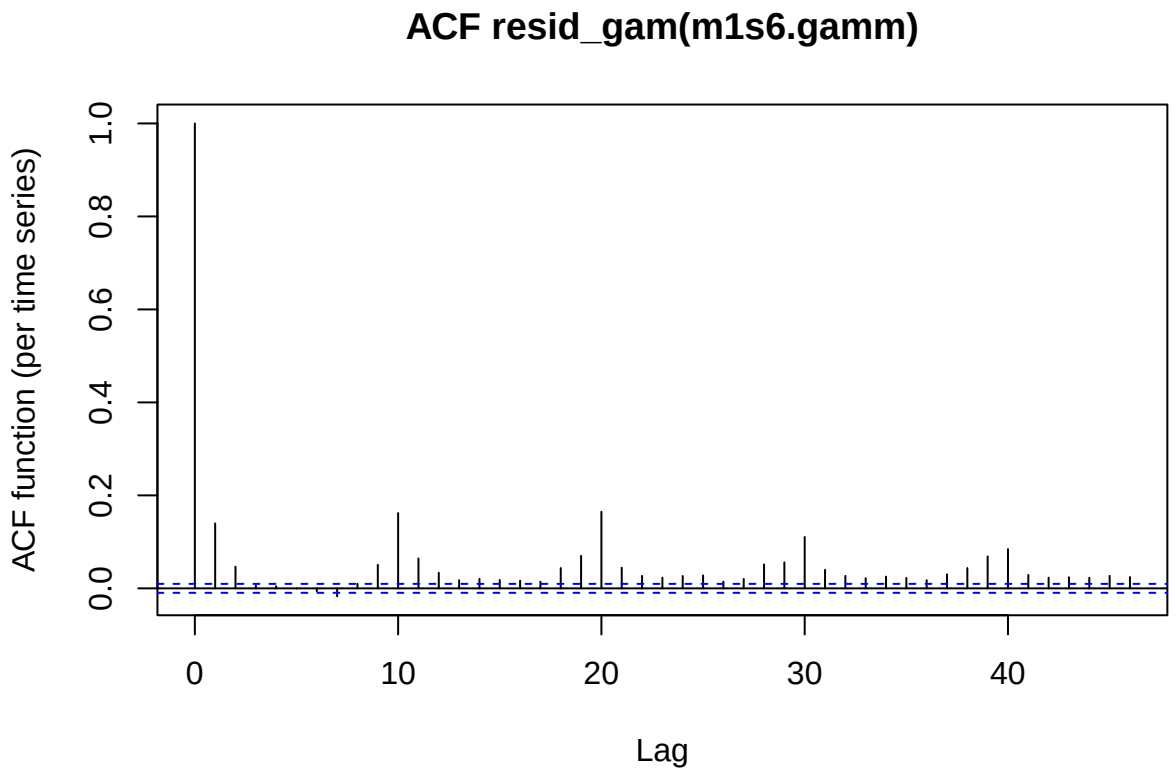
```
## [6]  6.459885e-09  1.818530e-04
##
## $hess
##               [,1]          [,2]          [,3]          [,4]          [,5]
## [1,]  2.294191e+00 -3.985064e-02  6.623764e-02  2.126878e-05 -7.048638e-02
## [2,] -3.985064e-02  2.162170e+00  3.680964e-02  1.218037e-05 -4.542763e-02
## [3,]  6.623764e-02  3.680964e-02  2.096331e-01 -3.175836e-05  6.107839e-02
## [4,]  2.126878e-05  1.218037e-05 -3.175836e-05  1.743181e-04  1.963157e-05
## [5,] -7.048638e-02 -4.542763e-02  6.107839e-02  1.963157e-05  1.744579e+00
## [6,] -1.639526e-02  1.309946e-02 -1.488568e-04 -3.088549e-05  2.931352e-02
## [7,] -8.717055e-09  1.750710e-07  6.855313e-10 -6.732617e-11  2.138316e-07
##               [,6]          [,7]
## [1,] -1.639526e-02 -8.717055e-09
## [2,]  1.309946e-02  1.750710e-07
## [3,] -1.488568e-04  6.855313e-10
## [4,] -3.088549e-05 -6.732617e-11
## [5,]  2.931352e-02  2.138316e-07
## [6,]  1.541685e+02  4.165596e-04
## [7,]  4.165596e-04  1.818555e-04
##
## Model rank =  450 / 450
##
## 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(reltime):consm   9.00   4.70    0.99    0.39
## s(reltime):consb   9.00   5.31    0.99    0.40
## s(reltime):consf   9.00   1.79    0.99    0.41
## s(reltime):consp   9.00   1.00    0.99    0.39
## s(reltime):consv   9.00   4.15    0.99    0.42
## ti(reltime,ppn)  400.00 342.70      NA      NA
```

```
summary( m1s6.gamm )
```

```
##
## Family: Scaled t(3.464,1.438)
## Link function: identity
##
## Formula:
## measure.c4 ~ s(reltime, by = cons, k = 10) + cons + ti(reltime,
##     ppn, bs = "fs", m = 1)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.47493    0.04358  10.897  < 2e-16 ***
## consb       -0.26141    0.06254  -4.180 2.92e-05 ***
## consf        1.04682    0.06406  16.343  < 2e-16 ***
## consp        0.77421    0.06186  12.515  < 2e-16 ***
## consv        0.03431    0.06292   0.545    0.585
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
```

```
##                       edf  Ref.df      F p-value
## s(reltime):consm   4.704   6.085 23.558  <2e-16 ***
## s(reltime):consb   5.309   6.743 18.176  <2e-16 ***
## s(reltime):consf   1.794   2.293 47.403  <2e-16 ***
## s(reltime):consp   1.000   1.001 39.314  <2e-16 ***
## s(reltime):consv   4.149   5.436  8.637  <2e-16 ***
## ti(reltime,ppn)  342.699 400.000 14.318  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.134   Deviance explained = 13.4%
## fREML =  31605  Scale est. = 1         n = 46392
```

```
m1s6.gamm.acf <- acf_resid(m1s6.gamm) # |rho|<.1, good
```




```
# gamtabs(m1s6.gamm)
```

```
# load(file="./m1s4.gamm.Rdata")
require(itsadug)
# P50 <- 97 # added 2023.01.11 for completeness; P50 of vowel durations
# plot_smooth( m1s6.gamm,
#              view="type_measure", plot_all="cons",
#              rug=FALSE, col=cp, lwd=3,
#              lty=c(1,5,4,3,6), # m b f p v
# #              legend_plot_all = c(x=50,y=0),
```

```
# #              xlim=c(0,P50*1.04), # allow plot room
#              xlab="relative vowel time", ylab="f0 (ST, centered)")
# legend( x=c(85,100), y=c(0,-1.5), col=cp, lwd=3, bty="n",
#         lty=c(1,5,4,3,6), # m b f p v
#         legend=levels(carrier2B194v2$cons) ) -> fig1legend
# abline(v=50,col="grey50",lty=3)
```

Without visualisation, the lower deviance explained (13% vs 38%) is a compelling reason to prefer the absolute-time GAMM (m1s4) over the relative-time alternative (m1s6).

Model 2: after fricatives

For fricatives, we include the degree of voicing of the preceding consonant as a predictor. This requires some data wrangling...

Wrangle voicing data

The predictor `cons` denoting the preceding word-initial consonant is a factor with `m` as baseline.

```
levels(carrier2B194v2$cons) # triple check
```

```
## [1] "m" "b" "f" "p" "v"
```

Next, we include the degree of voicing of the preceding consonant (`voicing_A`) as a variable in the data set. We do this for *all* preceding consonants. For /m/ the raw degree of voicing is always 100%.

The degree of voicing itself is inspected first, by taking only a single observation for each vowel token, at vowel *midpoint* (where `type_measure==0.5`). (This turns out to be safer than using the vowel *onset* where `type_measure==0`).

```
carrier2consA <- subset( carrier2B194v2, # for cons data
                         type_measure==0.5 )
```

Summary of raw `voicing_A`

Each of the five preceding consonants in this study occurred in 9 tokens (see Appendix of manuscript) spoken by 100 speakers.

```
require(dplyr)
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:nlme':
##
##     collapse

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
carrier2consA %>%
  group_by(cons) %>%
  summarize(mean=mean(voicing_A), sd=sd(voicing_A), n=length(voicing_A) )
```

```
## # A tibble: 5 x 4
##   cons   mean    sd     n
##   <fct> <dbl> <dbl> <int>
## 1 m     100    0      900
## 2 b      86.1 20.7    898
## 3 f      17.1  9.92   896
## 4 p      21.5  9.82   899
## 5 v      51.9 32.9    899
```

```
carrier2plosA <- subset( carrier2consA, cons %in% c("p","b") )
carrier2fricA <- subset( carrier2consA, cons %in% c("f","v") )
```

Note that for the fricatives' raw degree of voicing, SD for /v/ is considerably higher than for /f/.

Centering of `voicing_A`

First we center the raw `voicing_A` predictor.

Here we use the max voicing value, 100%, as the reference value for centering. This means that the max voicing value of 100%, as found for /m/, is centered to zero. The polarity of the scaled voicing is preserved: higher values denote more voicing. (This centering is an artefact of other unreported analyses.)

> For easier interpretation, the centered scale from −100 to 0 will be uncentered to run from 0 to 100 in Figure 2.

Because the `voicing_A` parameter is bimodally distributed, no scaling is done. The centered values are stored as a separate column `voi.c` in a new data frame, with suffix `v3`, which is also saved.

```
carrier2B194v3 <- cbind( carrier2B194v2,
                         voi.c = scale(carrier2B194v2$voicing_A,
                               center=100, scale=FALSE) )
save( carrier2B194v3, file="./carrier2B194v3.Rdata" )
```

We extract two data sets for the two consonant categories, with a single observation per vowel token (at `type_measure`=0.5, i.e., at vowel midpoint). These datasets are named `A`, and these contain consonant data (a single row per token).

We also extract two similar data sets, for the two consonant categories, with all observations per vowel token. These datasets are named `B`, and these contain vowel data (multiple rows per token).

```
carrier2plosA <- subset( carrier2B194v3,
                         (cons %in% c("p","b")) & (type_measure==0.5) )
carrier2plosB <- subset( carrier2B194v3,
                         (cons %in% c("p","b")) )
carrier2fricA <- subset( carrier2B194v3,
                         (cons %in% c("f","v")) & (type_measure==0.5) )
carrier2fricB <- subset( carrier2B194v3,
                         (cons %in% c("f","v")) )
```

For consistency, we re-use a previously prepared dataset for vowels after fricatives...

```
load( file="./carrier2fricBB.Rdata" ) # fric+V f0 data
```

Run GAMM 2

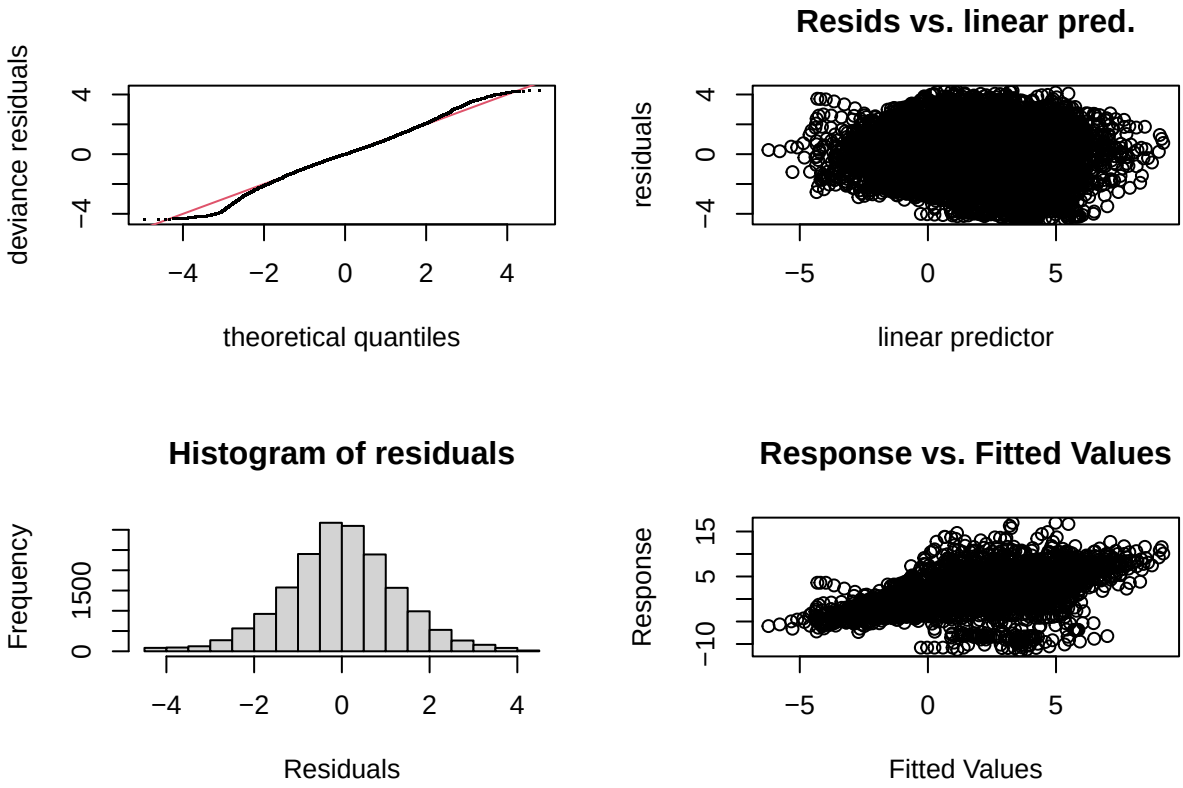
... and we use the same random seed as used in the analysis reported in the manuscript. Here we load the previously estimated second GAMM from disk, rather than computing it again (which would take >5 hours).

```
## voicing_A, raw, pooled over consonants, carrier2fricBB
set.seed(20221127) # for replicability
# commented out for knitting
# print( paste("Estimating fricative models m100 and m101 started at",Sys.time()) )
# system.time( # null
#   fric.m100fs3.gamm <- bam( measure.c4 ~ s(vowtime,by=cons) + cons +
#                               s(vowtime,ppn,by=cons,bs="fs",m=2) + # random slopes
#                               s(voi.c,by=cons,k=20), # voicing, raw
#                             AR.start=start.event,
#                             discrete=TRUE, # for speeding up
#                             family="scat",
#                             # nthreads = 4, # 4 cores
#                             data=carrier2fricB, weights=wght ) # BB not A
# ) # 624 seconds; null model, warnings ignored
# system.time( # full
#   fric.m101fs3.gamm <- bam( measure.c4 ~ cons +
#                               te(voi.c,vowtime,by=cons,k=20) +
#                               s(vowtime,ppn,by=cons,bs="fs",m=2), # random slopes
#                             AR.start=start.event,
#                             discrete=TRUE, # for speeding up
#                             family="scat",
#                             # nthreads = 4, # 4 cores
#                             data=carrier2fricBB, weights=wght ) # B not A
# ) # 19201 seconds; full model, no warnings
# print( paste("Finished at",Sys.time()) )
# save( fric.m100fs3.gamm,
#       file="./fric.m100fs3.gamm.Rdata" ) # comment for knitting
```

```
# save( fric.m101fs3.gamm,
#       file="./fric.m101fs3.gamm.Rdata" ) # comment for knitting
load( file="./fric.m100fs3.gamm.Rdata" ) # uncomment for knitting
load( file="./fric.m101fs3.gamm.Rdata" ) # uncomment for knitting
```

Check the second GAMM...

```
gam.check(fric.m101fs3.gamm)
```




```
##
## Method: fREML   Optimizer: perf chol
## $grad
##  [1] -1.418228e-03  5.026081e-05 -3.765534e-03 -2.133399e-02 -2.522959e-03
##  [6] -1.136350e-03 -1.675728e-03 -7.852637e-04  1.137196e-02  1.692326e-02
##
## $hess
##               [,1]          [,2]          [,3]          [,4]          [,5]
##  [1,]  3.608106e+00  1.606470e-01 -4.216457e-18  3.693197e-18  1.057500e-01
##  [2,]  1.606470e-01  4.904173e+00  4.083191e-17 -3.576470e-17  7.938073e-01
##  [3,] -5.560109e-18 -4.256887e-17  4.220561e+00  1.966127e-02 -4.544472e-16
##  [4,] -7.897059e-19 -1.820933e-17  1.966127e-02  2.390601e+00  3.384723e-16
##  [5,]  1.057500e-01  7.938073e-01  4.375509e-16 -3.832511e-16  4.742530e+01
##  [6,] -8.126781e-03  9.895818e-02  2.253976e-16 -1.974259e-16  3.628121e+00
##  [7,] -6.317145e-02 -5.125737e-02  5.563625e-16 -4.873182e-16  1.564897e+00
```

```
##  [8,]  7.946351e-19 -1.027209e-17  3.298790e-03 -5.281822e-02  4.187839e-16
##  [9,] -5.337840e-19  1.698471e-19  2.378834e-01  1.565248e-01  7.982457e-17
## [10,] -3.683043e-19  1.039198e-17 -1.380822e-01  3.962085e-01 -7.891008e-17
##                [,6]          [,7]          [,8]          [,9]         [,10]
##  [1,] -8.126781e-03 -6.317145e-02 -2.273978e-19  5.567336e-20 -4.516273e-18
##  [2,]  9.895818e-02 -5.125737e-02  2.202106e-18 -5.391375e-19  4.373532e-17
##  [3,]  2.312168e-16 -4.647946e-16  3.298790e-03  2.378834e-01 -1.380822e-01
##  [4,]  1.903151e-17  2.622957e-16 -5.281822e-02  1.565248e-01  3.962085e-01
##  [5,]  3.628121e+00  1.564897e+00  2.359756e-17 -5.777346e-18  4.686635e-16
##  [6,]  4.570050e+01 -4.605255e+00  1.215592e-17 -2.976111e-18  2.414248e-16
##  [7,] -4.605255e+00  3.949396e+01  3.000519e-17 -7.346113e-18  5.959233e-16
##  [8,] -9.675705e-17  1.208703e-16  2.879336e+01  2.552140e+00  2.248425e+00
##  [9,]  2.556909e-17 -2.579910e-17  2.552140e+00  4.672115e+01 -4.325250e+00
## [10,]  6.538053e-17 -2.767518e-16  2.248425e+00 -4.325250e+00  3.823573e+01
##
## Model rank =  2800 / 2800
##
## 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
## te(voi.c,vowtime):consv  399.0   46.9    0.99    0.12
## te(voi.c,vowtime):consf  399.0   45.9    0.98    0.07 .
## s(vowtime,ppn):consv    1000.0  329.6    1.00    0.61
## s(vowtime,ppn):consf    1000.0  291.7    1.00    0.52
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
print( summary( fric.m101fs3.gamm ) -> fric.m101fs3.gamm.summary )
```

```
##
## Family: Scaled t(3,0.827)
## Link function: identity
##
## Formula:
## measure.c4 ~ cons + te(voi.c, vowtime, by = cons, k = 20) + s(vowtime,
##     ppn, by = cons, bs = "fs", m = 2)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.7129     0.1894   3.764 0.000167 ***
## consf         0.6269     0.3204   1.957 0.050411 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                            edf Ref.df      F  p-value
## te(voi.c,vowtime):consv  46.91  60.01  7.893  < 2e-16 ***
## te(voi.c,vowtime):consf  45.92  56.74 11.496  < 2e-16 ***
## s(vowtime,ppn):consv    329.65 961.00 15.782  < 2e-16 ***
## s(vowtime,ppn):consf    291.73 906.00 24.293 0.000254 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## R-sq.(adj) =  0.444   Deviance explained = 51.1%
## fREML =  33650  Scale est. = 1         n = 18312
```

```
anova( fric.m100fs3.gamm, fric.m101fs3.gamm, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: measure.c4 ~ s(vowtime, by = cons) + cons + s(vowtime, ppn, by = cons,
##     bs = "fs", m = 2) + s(voi.c, by = cons, k = 20)
## Model 2: measure.c4 ~ cons + te(voi.c, vowtime, by = cons, k = 20) + s(vowtime,
##     ppn, by = cons, bs = "fs", m = 2)
##   Resid. Df Resid. Dev     Df Deviance  Pr(>Chi)
## 1     17530      59630
## 2     17456      59245 73.201   384.75 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Rank check is OK.

Two of the k-indices have $p < .05$, but the edf are always well below $k - 1$.

Deviance explained is 51.1 % .

The model containing the interaction effect provides a far better fit than the model without, $\chi^2(73) = 385, p < .001$.

Figure 2

This is a two-panel figure, left after /v/, right after /f/. We also need the values of `voi.c` from the fricative consonant dataset.

We suppress value labels along the centered (shifted) axis of degree of voicing, and add axis labels later.

```
par( mfcol=c(1,2)) -> op # two panels
require(vioplot) # add half violin plot along Y axis
```

```
## Loading required package: vioplot
```

```
## Loading required package: sm
```

```
## Package 'sm', version 2.2-5.7: type help(sm) for summary information
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
# /v/ panel
thecons <- "v"
# load(file="./carrier2fricA.Rdata") # consonant data, already loaded when knitting
ymarks <- carrier2fricA[carrier2fricA$cons==thecons, "voi.c"]
fvisgam( fric.m101fs3.gamm,
         view=c("vowtime","voi.c"),
         cond=list(cons=c(thecons)),
         xlab="vowel time (ms)",
         yaxt="n",
         # ylab=paste("pct. voicing of /",thecons,"/",sep=""),
         ylab="",
         contour.col="black", # not cp[5] # v is level 5 of cons
         color="gray",
         xlim=c(2,P50*.8), # P50 of vowel duration
         ylim=c(-97,-3), # range of voi.c of /v/
          zlim=c(-1,2),
         show.diff=TRUE, col.diff="white", alpha.diff=.99,
         main=paste("after /",thecons,"/",sep=""),
         add.color.legend = FALSE,
         n.grid=15,
         rm.ranef=TRUE ) -> fricv.fvisgam
```

```
## Summary:
##  * cons : factor; set to the value(s): v.
##  * voi.c : numeric predictor; with 15 values ranging from -97.000000 to -3.000000.
##  * vowtime : numeric predictor; with 15 values ranging from 2.000000 to 77.600000.
##  * ppn : factor; set to the value(s): BR02. (Might be canceled as random effect, check below.)
##  * NOTE : The following random effects columns are canceled: s(vowtime,ppn):consv,s(vowtime,ppn):cons
##
```

```
##  * Note: Shaded areas indicate areas that include 0 within the 95%CI (f parameter).
```

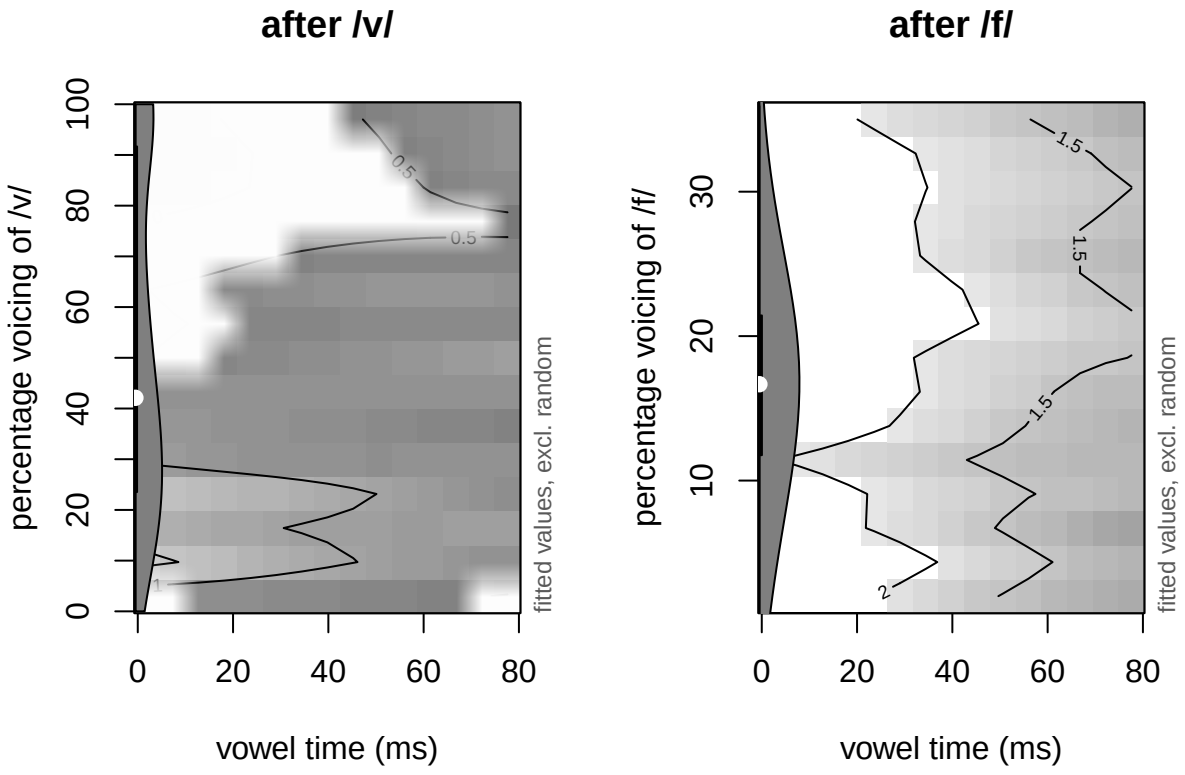
```r
# abline( h=cons.pc1.medians[thecons], lty="dashed", lwd=2, col=cp[3] )
axis( side=2, at=seq(-100,+0,by=10),
      labels=seq(-100,0,by=10)+100 )
mtext( paste("percentage voicing of /",thecons,"/", sep=""), side=2, line=2.5 )
# rug( ymarks, side=2, ticksize=0.04, quiet=TRUE, col="black" )
vioplot( I(ymarks), at=-0.5, xaxt="n", side="right", add=T,
         # yaxt="n", ylim=c(-95,-5), # range of voi.c of /v/
         h=h.select(ymarks,method="cv"), # h=6
         col="grey50",
         plotCentre="point", wex=14 )
# /f/ panel
thecons <- "f"
ymarks <- carrier2fricA[carrier2fricA$cons==thecons, "voi.c"]
fvisgam( fric.m101fs3.gamm,
         view=c("vowtime","voi.c"),
         cond=list(cons=c(thecons)),
         xlab="vowel time (ms)",
         yaxt="n",
         # ylab=paste("pct. voicing of /",thecons,"/",sep=""),
         ylab="",
         contour.col="black", # not cp[5] # v is level 5 of cons
```

```
        color="gray",
        xlim=c(2,P50*.8), # P50 of vowel duration
        ylim=c(-98,-65), # range of voi.c of /v/
        zlim=c(-1,2),
        show.diff=TRUE, col.diff="white", alpha.diff=.99,
        main=paste("after /",thecons,"/",sep=""),
        add.color.legend = FALSE,
        n.grid=15,
        rm.ranef=TRUE ) -> fricf.fvisgam
```

```
## Summary:
##  * cons : factor; set to the value(s): f.
##  * voi.c : numeric predictor; with 15 values ranging from -98.000000 to -65.000000.
##  * vowtime : numeric predictor; with 15 values ranging from 2.000000 to 77.600000.
##  * ppn : factor; set to the value(s): BR02. (Might be canceled as random effect, check below.)
##  * NOTE : The following random effects columns are canceled: s(vowtime,ppn):consv,s(vowtime,ppn):cons
##
```

```
##  * Note: Shaded areas indicate areas that include 0 within the 95%CI (f parameter).
```

```
# abline( h=cons.pc1.medians[thecons], lty="dashed", lwd=2, col=cp[3] )
axis( side=2, at=seq(-90,0,by=10),
      labels=seq(-90,0,by=10)+100 )
mtext( paste("percentage voicing of /",thecons,"/", sep=""), side=2, line=2.5 )
# rug( ymarks, side=2, ticksize=0.04, quiet=TRUE, col="black" )
# h.select(ymarks,method="cv") # CV yields h=6 as smoothing param
vioplot( I(ymarks), at=-0.5, xaxt="n", side="right", add=T,
        h=h.select(ymarks,method="cv"), # h=6
        col="grey50",
        plotCentre="point", wex=14*1.5 )
```



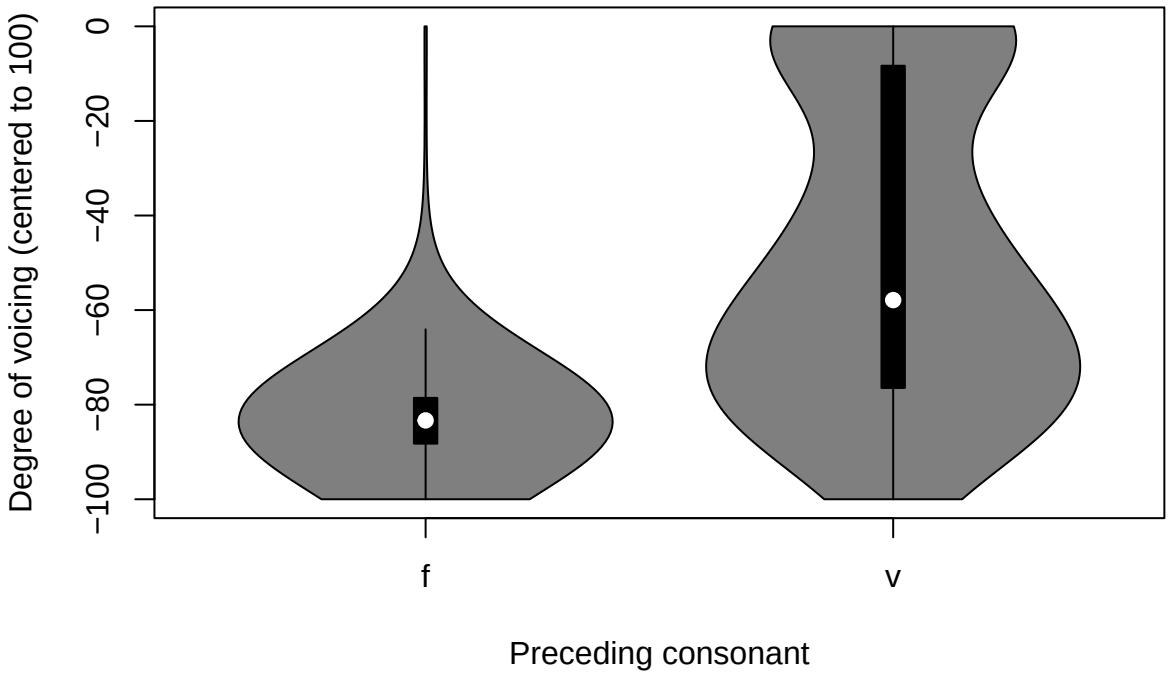

```
par(op) # restore earlier graphics settings
```

Fig. 2. Contour plots showing the interaction between vowel time (in ms, first part of vowel only) and the degree of voicing of the preceding consonant (in percentage points, rescaled), on f0 (in semitones, centered, lighter shade indicates higher f0). The lefthand panel shows the interaction for vowel tokens after /v/; the righthand panel shows the interaction after /f/; voicing axes are scaled differently across panels for clarity. The density of the degree of voicing is shown along the left axis (with quartiles and median). Random effects have been ignored in the fitted f0 values. White areas have an essentially flat contour. [file fric101gamm20230303.pdf]

Supplementary figure

This figure highlights the difference in distributions of the degrees of voicing of /v/ and /f/.

```
require(vioplot)
voidata <- subset(carrier2fricA, cons %in% c("v","f") )
voidata$cons <- as.character(voidata$cons) # to exclude missing levels of cons
vioplot( voi.c~cons, data=voidata,
         xlab="Preceding consonant",
         ylab="Degree of voicing (centered to 100)" )
```



Model 3: after plosives

For consistency (and as with fricative consonants), we re-use a previously prepared dataset for vowels after plosives...

```
load( file="./carrier2plosBB.Rdata" ) # plos+V f0 data
```

Run GAMM 3

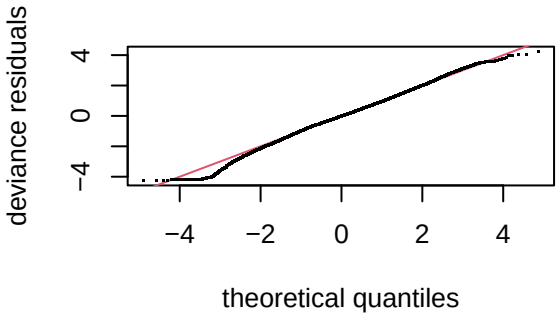
... and we use the same random seed as used in the analysis reported in the manuscript. Here we load the previously estimated second GAMM from disk, rather than computing it again (which would take >5 hours).

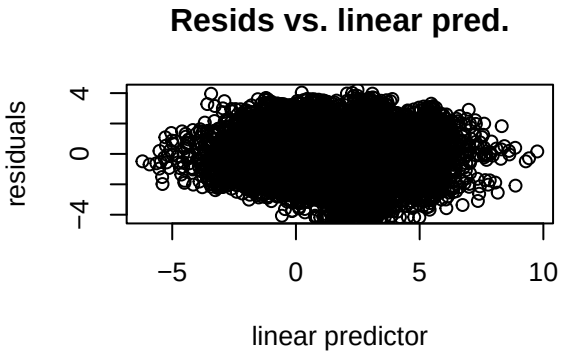
```
## VOT_ms, raw, pooled over consonants, carrier2plosBB
set.seed(20221127) # for replicability
# commented out for knitting
# print( paste("Estimating plosive models m100 and m101 started at",Sys.time()) )
# system.time( # null
#   plos.m100fs3.gamm <- bam( measure.c4 ~ s(vowtime,by=cons) + cons +
#                             s(vowtime,ppn,by=cons,bs="fs",m=2) + # random slopes
#                             s(VOT_ms,by=cons,k=20), # VOT, raw
```

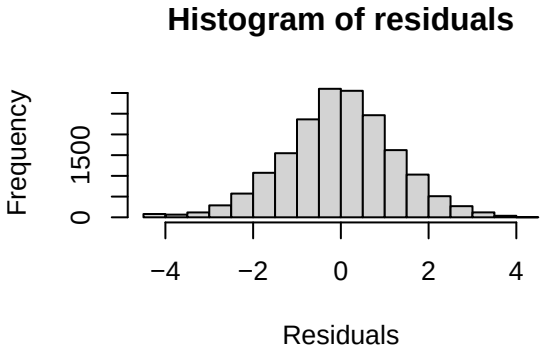
```
#                             AR.start=start.event,
#                             discrete=TRUE, # for speeding up
#                             family="scat", nthreads = 4, # 4 cores
#                             data=carrier2plosBB, weights=wght ) # BB not AA
# ) #
# system.time( # full
#   plos.m101fs3.gamm <- bam( measure.c4 ~ cons +
#                               te(VOT_ms,vowtime,by=cons,k=20) +
#                               s(vowtime,ppn,by=cons,bs="fs",m=2), # random slopes
#                             AR.start=start.event,
#                             discrete=TRUE, # for speeding up
#                             family="scat", nthreads = 4, # 4 cores
#                             data=carrier2plosBB, weights=wght ) # BB not AA
# ) # seconds
# print( paste("Finished at",Sys.time()) )
# save( plos.m100fs3.gamm,
#       file="./plos.m100fs3.gamm.Rdata" ) # comment for knitting
# save( plos.m101fs3.gamm,
#       file="./plos.m101fs3.gamm.Rdata" ) # comment for knitting
# do not load if models are still in global environment
set.seed(20221129) # for replicability
load( file="./plos.m100fs3.gamm.Rdata" ) # comment for knitting
load( file="./plos.m101fs3.gamm.Rdata" ) # comment for knitting
```

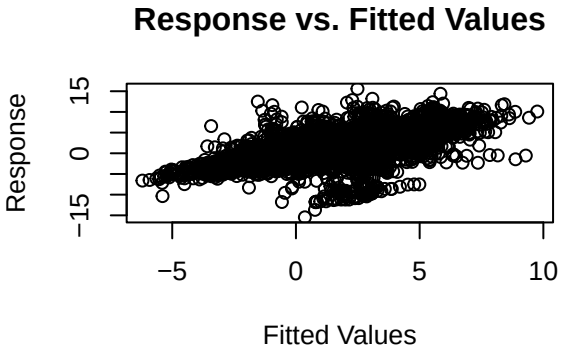
Check model . . .

```
gam.check(plos.m101fs3.gamm) # good
```










```
##
## Method: fREML   Optimizer: perf chol
## $grad
##  [1]  1.427009e-05  2.123156e-05 -1.866202e-06 -1.950109e-07  1.493397e-04
##  [6]  5.611950e-06  2.432947e-05  5.402316e-05  6.326533e-06  2.759540e-05
##
## $hess
##                [,1]          [,2]          [,3]          [,4]          [,5]
##  [1,]  2.384161e+00  8.946567e-01  2.323486e-15  2.831975e-17 -6.999196e-02
##  [2,]  8.946567e-01  4.256864e+00  2.009181e-15  2.448885e-17  2.079399e+00
##  [3,] -2.113029e-16  1.427746e-15  1.716976e+00  5.320408e-02  1.300408e-15
##  [4,]  1.412771e-17 -1.476231e-17  5.320408e-02  5.351250e-02 -2.009044e-17
##  [5,] -6.999196e-02  2.079399e+00 -3.320996e-15 -4.047788e-17  5.161239e+01
##  [6,] -6.295552e-02  2.378950e-02 -4.485795e-15 -5.467500e-17  3.091688e+00
##  [7,]  1.958563e-02 -4.050209e-02 -1.494917e-14 -1.822076e-16  2.874024e+00
##  [8,]  7.497276e-17  3.319637e-16 -7.981285e-03  6.592489e-02 -6.034281e-16
##  [9,] -3.313557e-16  1.159356e-16 -6.821563e-02  7.508715e-03  1.764277e-16
## [10,] -4.153403e-16  3.128558e-17 -7.870389e-02  2.156061e-02  2.755611e-16
##                [,6]          [,7]          [,8]          [,9]          [,10]
##  [1,] -6.295552e-02  1.958563e-02  3.287098e-17 -2.744488e-16 -6.343757e-16
##  [2,]  2.378950e-02 -4.050209e-02  2.842442e-17 -2.373232e-16 -5.485617e-16
##  [3,] -6.469235e-15 -1.329403e-14 -7.981285e-03 -6.821563e-02 -7.870389e-02
##  [4,] -5.951350e-17 -2.341480e-16  6.592489e-02  7.508715e-03  2.156061e-02
##  [5,]  3.091688e+00  2.874024e+00 -4.698302e-17  3.922741e-16  9.067235e-16
##  [6,]  4.583427e+01 -5.012644e+00 -6.346174e-17  5.298594e-16  1.224746e-15
##  [7,] -5.012644e+00  3.894842e+01 -2.114899e-16  1.765787e-15  4.081536e-15
```

```
##  [8,] -1.193344e-15 -4.395797e-15  1.980468e+01  2.669774e+00  2.115169e+00
##  [9,]  3.814901e-16  1.420920e-15  2.669774e+00  4.680073e+01 -4.597098e+00
## [10,]  4.455636e-16  1.150530e-15  2.115169e+00 -4.597098e+00  3.835403e+01
##
## Model rank =  2800 / 2800
##
## 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
## te(VOT_ms,vowtime):consb  399   45    1.00    0.68
## te(VOT_ms,vowtime):consp  399   17    1.01    0.77
## s(vowtime,ppn):consb     1000  338    1.01    0.89
## s(vowtime,ppn):consp     1000  283    1.01    0.91
```

```
print( summary( plos.m101fs3.gamm ) -> plos.m101fs3.gamm.summary )
```

```
##
## Family: Scaled t(3,0.824)
## Link function: identity
##
## Formula:
## measure.c4 ~ cons + te(VOT_ms, vowtime, by = cons, k = 20) +
##     s(vowtime, ppn, by = cons, bs = "fs", m = 2)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.5634     0.1935   2.912  0.00359 **
## consp        -0.4935     0.9462  -0.522  0.60201
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                             edf Ref.df      F p-value
## te(VOT_ms,vowtime):consb  45.03  59.50  4.039  <2e-16 ***
## te(VOT_ms,vowtime):consp  16.97  18.62 11.266  <2e-16 ***
## s(vowtime,ppn):consb     338.35 943.00 16.014  <2e-16 ***
## s(vowtime,ppn):consp     283.42 317.00 68.508   0.985
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.484   Deviance explained = 49.9%
## fREML =  33060  Scale est. = 1         n = 18318
```

```
anova( plos.m100fs3.gamm, plos.m101fs3.gamm, test="Chisq") # p=.06
```

```
## Analysis of Deviance Table
##
## Model 1: measure.c4 ~ s(vowtime, by = cons) + cons + s(vowtime, ppn, by = cons,
##     bs = "fs", m = 2) + s(VOT_ms, by = cons, k = 20)
## Model 2: measure.c4 ~ cons + te(VOT_ms, vowtime, by = cons, k = 20) +
##     s(vowtime, ppn, by = cons, bs = "fs", m = 2)
##   Resid. Df Resid. Dev     Df Deviance Pr(>Chi)
```

```
## 1     17548      57894
## 2     17500      57831 47.662   63.365  0.06349 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Rank check is OK.

The k-indices are also OK (>1).

Deviance explained is 49.9 % .

The model containing the interaction effect does not provide a significantly better fit than the model without, $\chi^2(48) = 63.3, p = .063$.

Figure 3

This is a two-panel figure, left after /b/, right after /p/. We also need the values of raw `VOT_ms` (in ms) from the plosive consonant dataset `carrier2plosA`.

```
par( mfcol=c(1,2)) -> op # two panels
require(vioplot) # add half violin plot along Y axis
# /b/ panel
thecons <- "b"
# load("~/carrier2plosA.Rdata") # consonant data, already loaded when knitting
ymarks <- carrier2plosA[carrier2plosA$cons==thecons, "VOT_ms"]
fvisgam( plos.m101fs3.gamm,
         view=c("vowtime","VOT_ms"),
         cond=list(cons=c(thecons)),
         xlab="vowel time (ms)",
         ylab=paste("VOT of /",thecons,"/ (ms)",sep=""),
         xlim=c(3,P50*.8), # P50 of vowel duration
         ylim=c(-150,-50), # range of VOT of plosives
         zlim=c(-3,3),
         show.diff=TRUE, col.diff="white", alpha.diff=.99,
         main=paste("after /",thecons,"/",sep=""),
         add.color.legend = FALSE,
         n.grid=12,
         rm.ranef=TRUE, color="gray" ) -> plosb.fvisgam
```

```
## Summary:
##  * cons : factor; set to the value(s): b.
##  * VOT_ms : numeric predictor; with 12 values ranging from -150.000000 to -50.000000.
##  * vowtime : numeric predictor; with 12 values ranging from 3.000000 to 77.600000.
##  * ppn : factor; set to the value(s): BR01. (Might be canceled as random effect, check below.)
##  * NOTE : The following random effects columns are canceled: s(vowtime,ppn):consb,s(vowtime,ppn):cons
##
```

```
##  * Note: Shaded areas indicate areas that include 0 within the 95%CI (f parameter).
```

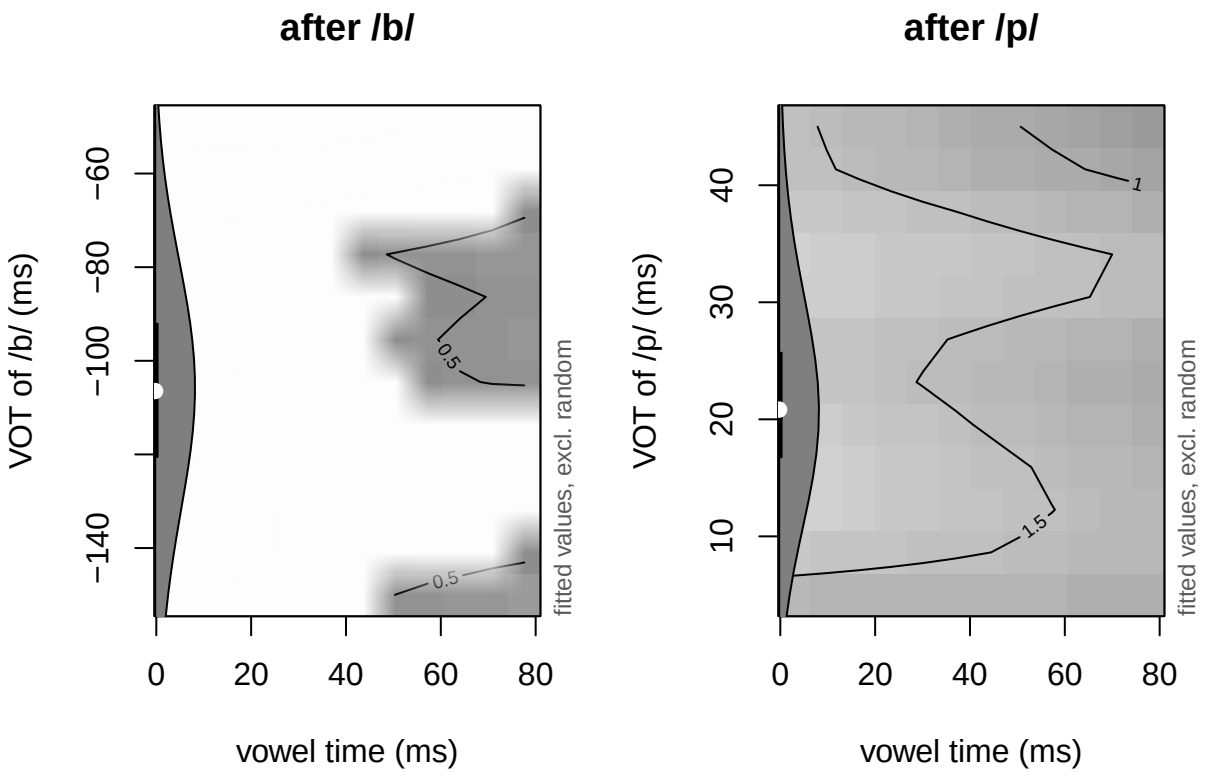
```
vioplot( I(ymarks), at=-0.25, xaxt="n", side="right", add=T,
         h=h.select(ymarks,method="cv"), # h=6
         col="grey50",
         plotCentre="point", wex=14*1.5 )
```

```r
# /p/ panel
thecons <- "p"
ymarks <- carrier2plosA[carrier2plosA$cons==thecons, "VOT_ms"]
fvisgam( plos.m101fs3.gamm,
         view=c("vowtime","VOT_ms"),
         cond=list(cons=c(thecons)),
         xlab="vowel time (ms)",
         ylab=paste("VOT of /",thecons,"/ (ms)",sep=""),
         #          contour.col=cp[5], # v is level 5 of cons
         #          color="gray",
         xlim=c(3,P50*.8), # using P50 of vowel duration
         ylim=c(5,45), # range of VOT of plosives
         zlim=c(-2.5,2.5), # range of semitone scale
         show.diff=TRUE, col.diff="white", alpha.diff=.99,
         main=paste("after /",thecons,"/",sep=""),
         add.color.legend = FALSE,
         n.grid=12,
         rm.ranef=TRUE, color="gray" ) -> plosp.fvisgam
```

```
## Summary:
##  * cons : factor; set to the value(s): p.
##  * VOT_ms : numeric predictor; with 12 values ranging from 5.000000 to 45.000000.
##  * vowtime : numeric predictor; with 12 values ranging from 3.000000 to 77.600000.
##  * ppn : factor; set to the value(s): BR01. (Might be canceled as random effect, check below.)
##  * NOTE : The following random effects columns are canceled: s(vowtime,ppn):consb,s(vowtime,ppn):cons
##
```

```
##  * Note: Shaded areas indicate areas that include 0 within the 95%CI (f parameter).
```

```r
vioplot( I(ymarks), at=-0.25, xaxt="n", side="right", add=T,
         h=h.select(ymarks,method="cv"), # h=6
         col="grey50",
         plotCentre="point", wex=14*1.5 )
```




```
par(op) # restore earlier graphics settings
```

Fig. 3. Contour plots showing the interaction between vowel time (in ms, first part of vowel only) and voice onset time (VOT, in ms), on f0 (in semitones, centered, lighter shade indicates higher f0). The lefthand panel shows the interaction for vowel tokens after /b/; the righthand panel shows the interaction after /p/; VOT axes are scaled differently across panels for clarity. The density of VOTs is shown along the left axis (with quartiles and median). Random effects have been ignored in the fitted f0 values. White areas have an essentially flat contour. [file plos101gamm20230303.pdf]

Cleanup

```
rm( fric.m100fs3.gamm, plos.m100fs3.gamm )
rm( op )
```

Admin

```r
sessionInfo()
```

```
## R version 4.3.1 (2023-06-16)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Ventura 13.5.2
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib;  LAPACK v
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: Europe/Amsterdam
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] vioplot_0.4.0     zoo_1.8-12        sm_2.2-5.7.1      dplyr_1.1.2
## [5] MuMIn_1.47.5      itsadug_2.4.1     plotfunctions_1.4 mgcv_1.8-42
## [9] nlme_3.1-162
##
## loaded via a namespace (and not attached):
##  [1] vctrs_0.6.3      cli_3.6.1        knitr_1.43       rlang_1.1.1
##  [5] xfun_0.40        highr_0.10       generics_0.1.3   glue_1.6.2
##  [9] htmltools_0.5.6  stats4_4.3.1     fansi_1.0.4      rmarkdown_2.24
## [13] grid_4.3.1       tibble_3.2.1     evaluate_0.21    fastmap_1.1.1
## [17] yaml_2.3.7       lifecycle_1.0.3  compiler_4.3.1   pkgconfig_2.0.3
## [21] lattice_0.21-8   digest_0.6.33    R6_2.5.1         tidyselect_1.2.0
## [25] utf8_1.2.3       pillar_1.9.0     splines_4.3.1    magrittr_2.0.3
## [29] Matrix_1.5-4.1   tools_4.3.1
```

```r
citation() # base R
```

```
## To cite R in publications use:
##
##   R Core Team (2023). _R: A Language and Environment for Statistical
##   Computing_. R Foundation for Statistical Computing, Vienna, Austria.
##   <https://www.R-project.org/>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {R: A Language and Environment for Statistical Computing},
##     author = {{R Core Team}},
##     organization = {R Foundation for Statistical Computing},
##     address = {Vienna, Austria},
##     year = {2023},
##     url = {https://www.R-project.org/},
##   }
```

```
##
## We have invested a lot of time and effort in creating R, please cite it
## when using it for data analysis. See also 'citation("pkgname")' for
## citing R packages.
```

```
citation("mgcv")
```

```
## 2011 for generalized additive model method; 2016 for beyond exponential
## family; 2004 for strictly additive GCV based model method and basics of
## gamm; 2017 for overview; 2003 for thin plate regression splines.
##
##   Wood, S.N. (2011) Fast stable restricted maximum likelihood and
##   marginal likelihood estimation of semiparametric generalized linear
##   models. Journal of the Royal Statistical Society (B) 73(1):3-36
##
##   Wood S.N., N. Pya and B. Saefken (2016) Smoothing parameter and model
##   selection for general smooth models (with discussion). Journal of the
##   American Statistical Association 111:1548-1575.
##
##   Wood, S.N. (2004) Stable and efficient multiple smoothing parameter
##   estimation for generalized additive models. Journal of the American
##   Statistical Association. 99:673-686.
##
##   Wood, S.N. (2017) Generalized Additive Models: An Introduction with R
##   (2nd edition). Chapman and Hall/CRC.
##
##   Wood, S.N. (2003) Thin-plate regression splines. Journal of the Royal
##   Statistical Society (B) 65(1):95-114.
##
## To see these entries in BibTeX format, use 'print(<citation>,
## bibtex=TRUE)', 'toBibtex(.)', or set
## 'options(citation.bibtex.max=999)'.
```

```
citation("itsadug")
```

```
## To cite package 'itsadug' in publications use:
##
##   van Rij J, Wieling M, Baayen R, van Rijn H (2022). "itsadug:
##   Interpreting Time Series and Autocorrelated Data Using GAMMs." R
##   package version 2.4.1.
##
## A BibTeX entry for LaTeX users is
##
##   @Misc{,
##     title = {{itsadug}: Interpreting Time Series and Autocorrelated Data Using GAMMs},
##     author = {Jacolien {van Rij} and Martijn Wieling and R. Harald Baayen and Hedderik {van Rijn}},
##     year = {2022},
##     note = {R package version 2.4.1},
##   }
```

```
citation("MuMIn")
```

```
## To cite package 'MuMIn' in publications use:
##
##   Bartoń K (2023). _MuMIn: Multi-Model Inference_. R package version
##   1.47.5, <https://CRAN.R-project.org/package=MuMIn>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {MuMIn: Multi-Model Inference},
##     author = {Kamil Bartoń},
##     year = {2023},
##     note = {R package version 1.47.5},
##     url = {https://CRAN.R-project.org/package=MuMIn},
##   }
##
## ATTENTION: This citation information has been auto-generated from the
## package DESCRIPTION file and may need manual editing, see
## 'help("citation")'.
```

1. The Supplementary Materials in this file accompany the following published article:

Pinget, A.-F. & Quené, H. (2023). Effects of obstruent voicing on vowel fundamental frequency in Dutch. *Journal of the Acoustical Society of America*, 154 (4), 2124–2136. [https://doi.org/10.1121/10.0021070].

2. The data set used in this script, and saved output files, are available at https://doi.org/10.24416/UU01-2XJTYX.

If you use the data or use the software, we kindly ask you to cite (1) the article and/or (2) the package containing the supplementary materials.

HQ 20230308 and 20230611 and 20231009