Generalized Additive Models

What can they do for you?

Cesko Voeten

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

Outline

Setup...

```r
if (packageVersion('base') < '4.1.0') {
        stop('Your R is too old, please update')
}
pkgs <- c('knitr','mgcv','MASS','dplyr','ggplot2','magrittr','languageR')
if (any(missing <- !sapply(pkgs,\(x) do.call('require',list(x))))) {
        stop('Please install ',paste(pkgs[missing],collapse=', '))
}
opts_chunk$set(out.extra='keepaspectratio',
               fig.align='center',
               out.height='0.7\\textheight',
               fig.width=10,
               fig.height=4)
```

What problem do GAMs solve?

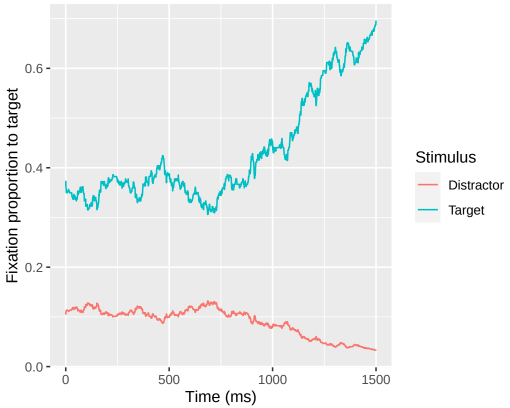
The messiness of reality

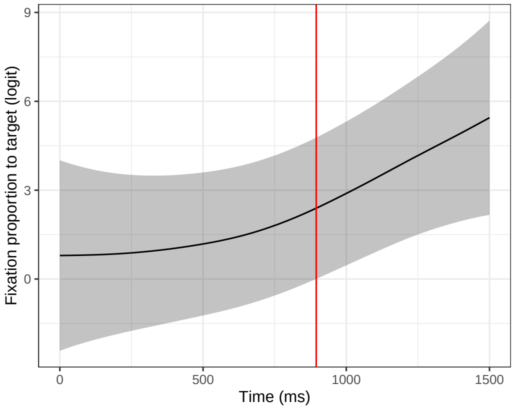
- Most statistical techniques used by linguists assume a linear relationship between response(s) and predictor(s)
- Unfortunately, for data from human subjects (as is most data from linguistics), reality doesn't agree:
  - phonetics: the vocal tract does not snap from one segment into the next
  - psycholinguistics: all sorts of nonlinear effects, e.g. fatigue over trials
  - sociolinguistics: regional variation can be spread out over a 2-D map
  - neurolinguistics: topographical distribution of ERPs is spread out over the surface of a sphere
  - Etc
- See e.g. Baayen et al. (2017)

GAMs

- GAMs are a form of regression analysis that can model nonlinearities
- They are closely related (in fact, equivalent) to mixed-effects models
- They are especially useful for time-series data, such as acoustics, articulation, eyetracking, etc.

Example: eyetracking data with onset competitor presented in −12 dB noise (Hintz et al., 2021)

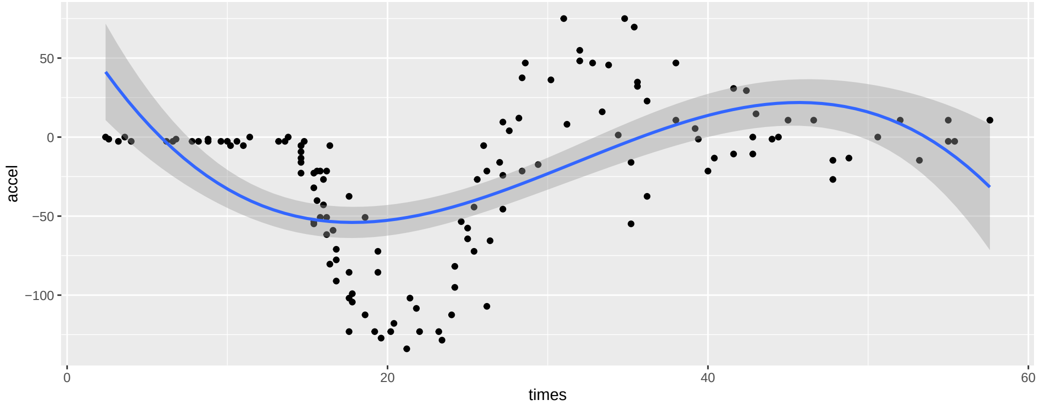




Smoothing splines

Fitting a spline

```r
ggplot(mcycle,aes(times,accel)) +
        geom_point() +
        stat_smooth(method='lm',formula=y ~ x + I(x^2) + I(x^3))
```



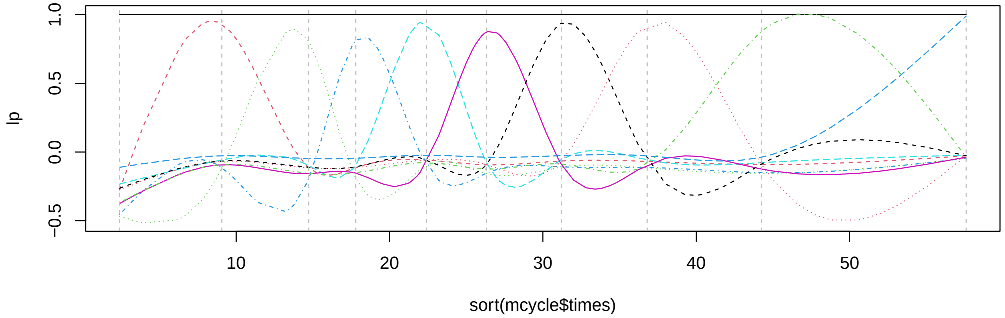
Penalized splines

- We've just invented a very basic spline with four basis functions (intercept, x, x^2, x^3)
- Is this complexity sufficient (cf. underfitting) and warranted by the data (cf. overfitting)?
- Model comparisons could be used to take out, e.g., the quadratic and cubic components as a whole, but what if we need just a little bit of both?
- Or what if we need a different basis function altogether, e.g. a spline on the sphere?
- Enter GAMs

More complex spline bases

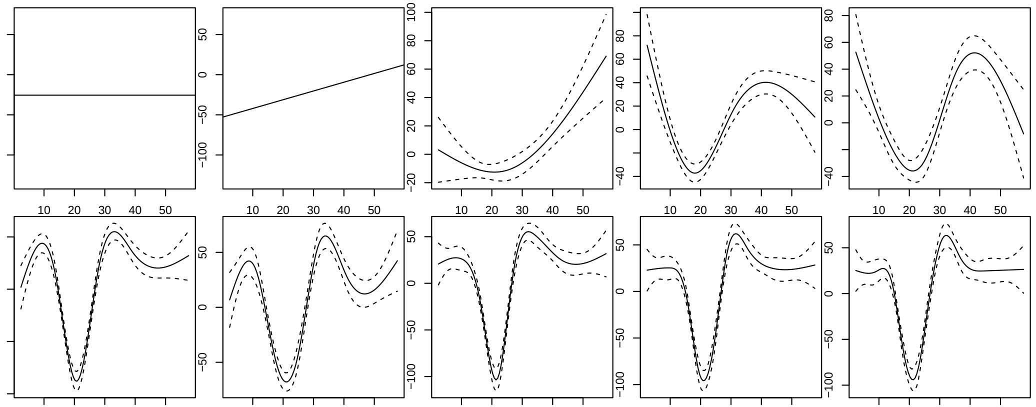
E.g.: the cubic regression spline — here given ten basis functions:

```r
model <- gam(accel ~ s(times,bs='cr',k=10),data=mcycle,method='ML')
lp <- predict(model,type='lpmatrix')
matplot(sort(mcycle$times),lp,type='l')
abline(v=model$smooth[[1]]$xp,lty=2,col='gray')
```



Fitted to data

```r
par(mfrow=c(2,5),mar=rep(1,4))
plot(accel ~ times,mcycle,type='n'); abline(lm(accel ~ 1,mcycle))
plot(accel ~ times,mcycle,type='n'); abline(lm(accel ~ times,mcycle))
for (k in 3:10) plot(gam(accel~s(times,bs='cr',k=k),data=mcycle),rug=FALSE)
```



Penalized splines, again

- A GAM is a penalized GLM that automatically determines a smoothing parameter for each spline (called 'smooths' in GAM parlance)
- Each of our spline's 10 basis functions is given a regression coefficient
- However, these 10 regression coefficients are penalized by the one smoothing parameter
- S.p. == 0: fit the data exactly (very wiggly spline, overfitted). S.p. == Inf: reduce to a straight line (very smooth spline, underfitted)
- The smoothing parameter is selected automatically through, e.g., REML — no work for you!

Relation to the mixed-effects model

- There is an important analogy between GAMs and mixed-effects models
- Random intercepts can be viewed as "splines" that generate a single line for every subject, item, etc.
- Random slopes produce interactions
- The GAM smoothing parameter is inversely related to the mixed-effects-model's variance component
- Each spline coefficient is a "subject" from a "population" of possible splines ('a convenient fiction to implement a smoother'; Hodges, 2016)
- Therefore, every mixed-effects model is also a GAM and every GAM is also a mixed-effects model

Functions for fitting GAMs

In package `mgcv`:

- `gam`: focus for today
- `bam`: version of `gam` optimized for very big datasets
- `gamm`: uses `nlme::lme` to fit the GAM as a mixed-effects model, so you can use `nlme`'s correlation structures
- `jagam`: generates JAGS code

In package `gamm4`:

- `gamm4`: uses `lme4` to fit the GAM as a mixed-effects model; has advantages for model comparisons, but in practice often misconverges badly

In package `brms`:

- `brm`: understands GAM syntax by fitting the GAM as a mixed-effects model

`mgcv`

Fitting our first gam

```
model <- gam(accel ~ s(times,bs='tp',k=10,m=2),data=mcycle,method='ML')
summary(model)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## accel ~ s(times, bs = "tp", k = 10, m = 2)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -25.546      1.951  -13.09   <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(times) 8.625  8.958 53.4  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.783   Deviance explained = 79.7%
```

What just happened?

- `s()` fits a smooth to the predictor mentioned in its argument(s), here `times`
- `bs='tp'`: use a thin-plate regression spline as the basis (this is the default); fits a bendy line (or, in 2D, a bendy sheet, or in 3D, a bendy cube, etc)
- `k=10`: ten basis functions
- `m=2`: second-derivative penalty (this is the default)
- `method='ML'` because this performs best in simulations, unless there are random effects in which case you should prefer `method='REML'`

Possible spline bases

Lots of possible bases (see `?smooth.construct`), but I want to focus on four here:

`bs='tp'` the thin-plate regression spline

`s(times,bs='tp',k=10)`

`bs='sos'` splines on the sphere (use with `m=-1` or `m=-2` please)

`s(lat,lon,bs='sos',k=30,m=-1)`

`bs='re'` random intercepts (with factor argument) or random slopes (with factor and numeric arguments)

`s(subject,bs='re'); s(subject,covariate,bs='re')`

Wrong: `s(subject,factor,bs='re')`; use `s(subject,by=factor,bs='re')` instead

`bs='fs'` random smooths (use with first-derivative penalties please!)

`s(subject,covariate,bs='fs',xt='tp',k=10,m=1)`

A simple 1-D smooth

```
s(x)
```

or, with the defaults explicitly filled in:

```
s(x,bs='tp',k=9,m=2)
```

This creates a bendy line.

Interactions — smooth+smooth

Some bases support isotropic interactions, e.g. this creates a 2D bendy sheet:

```
s(x,y,bs='tp')
```

But if not isotropic, or if different splines should be combined, `te` can be used to construct a tensor product:

```
te(Point,PhonolDiversity,PhonolAwareness,bs=c('tp','tp','tp'))
```

(also see `ti` and `t2` — the former is useful for model comparisons, the latter can be used to construct random tensor products: `t2(Subject,Point,PhonolDiversity, bs=c('re','tp','tp'),k=c(10,10,10),m=c(1,1,1),full=TRUE)`)

Interactions — smooth+parametric

Smooths can interact with factor variables using the `by` argument:

```
s(time,bs='tp',by=factor)
```

The above creates a separate smooth by every factor level; you also need to include `factor` as a main effect. If `factor` is an ordered factor, the first level is skipped.

The `by` argument can also be used with covariates, in which case the covariate directly multiplies the smooth:

```
s(time,bs='tp',by=covariate)
```

In some cases, 0/1 dummy variables can be useful here. Do not also add them as main effects.

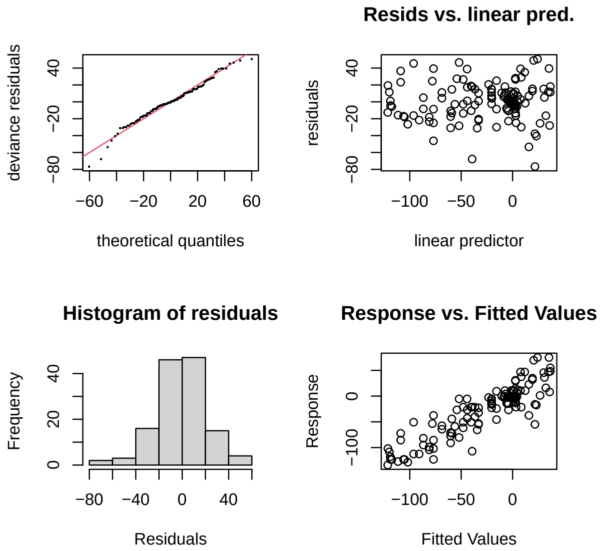
More exotic interactions possible — see `?gam.models`

Checking assumptions

Checking assumptions

```
gam.check(model)

##
## Method: ML   Optimizer: outer newton
## full convergence after 7 iterations.
## Gradient range [-1.508465e-06,1.227471e-06]
## (score 622.2919 & scale 506.3487).
## Hessian positive definite, eigenvalue range [3.331059,66.73635].
## Model rank =  10 / 10
##
## 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(times) 9.00 8.63    1.15    0.96
```

Checking assumptions



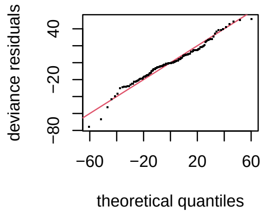
k-values

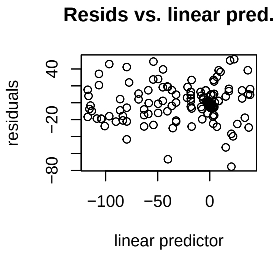
- We have too few basis functions: `k'` is 9, and the edf are `8.63`
- Having too few basis functions leads to oversmoothing
- Having too many basis functions is OK, because the basis functions that are not useful will get penalized to 0 by the influence from the smoothing parameter

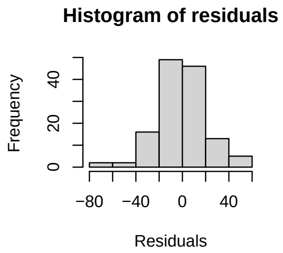
Refit

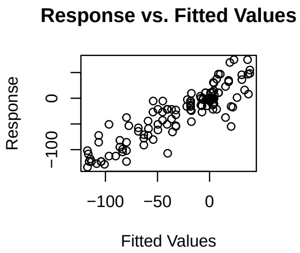
```
model <- gam(accel ~ s(times,bs='tp',k=30),data=mcycle,method='ML')
gam.check(model)

##
## Method: ML   Optimizer: outer newton
## full convergence after 5 iterations.
## Gradient range [-6.794927e-08,4.457834e-08]
## (score 624.9558 & scale 510.6724).
## Hessian positive definite, eigenvalue range [5.194377,67.10617].
## Model rank =  30 / 30
##
## 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(times) 29.0 12.6    1.17    0.98
```

Refit

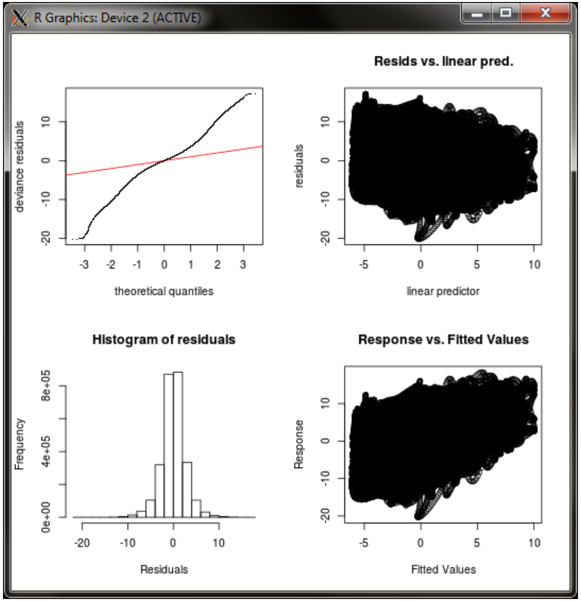

deviance residuals
40
−20
−80
−60
−20
20
60
theoretical quantiles

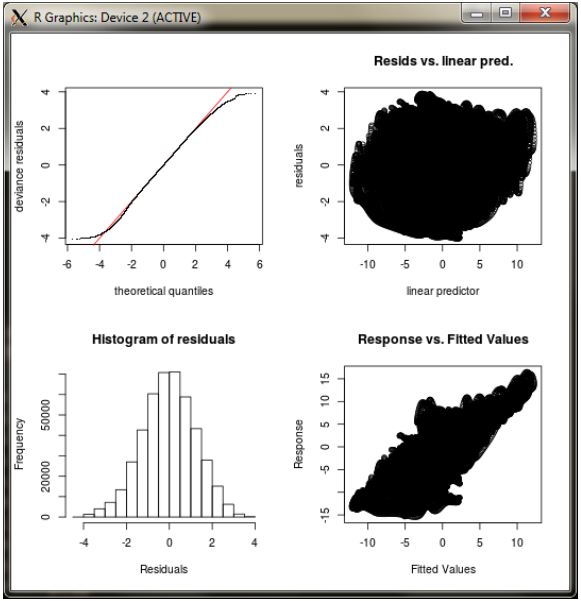

Resids vs. linear pred.
residuals
40
−20
−80
−100
−50
0
linear predictor


Histogram of residuals
Frequency
40
20
0
−80
−40
0
40
Residuals


Response vs. Fitted Values
Response
0
−100
−100
−50
0
Fitted Values

Wrong family





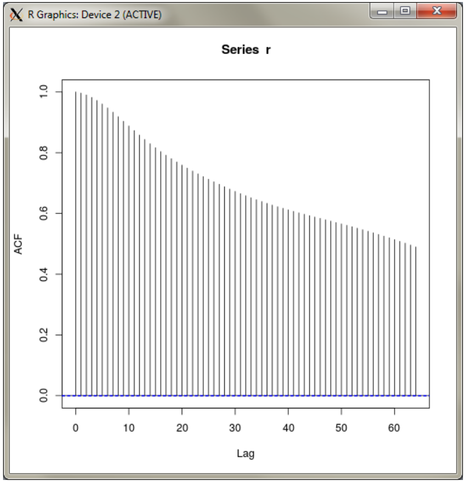
Families in `mgcv`

See `?family.mgcv`:

- All regular GLM families
- Extended families: `ocat`, `tw`, `nb`, `betar`, `scat`, `ziP`. Can be fitted through `bam` and `gam` with ML and REML estimation.
- General families: `cox.ph`, `gammals`, `gaulss`, `gevlss`, `gumbls`, `shash`, `ziplss`, `mvn`, `multinom`. Take `lists` of multiple formulas, and can only be fitted through `gam` with REML estimation only.

Autocorrelation

Specifically for time-series data: check that you indeed modeled all of the time-series structure, even with appropriate k values (`acf(resid(model))`)



Requires data to be sorted by time last!

Addressing autocorrelation

- The ideal option: 'event smooths' (Harald Baayen)
  - Random smooths for every time series (e.g. subject–item combination) in your data
  - Computationally devastating
- Also OK: `gamm` with a suitable `correlation` argument
  - Can only use regular exponential families
  - Convergence can be problematic
- Also OK: `bam` with an AR(1) model
  - Can fit simple AR(1) models only, but this is often already sufficient
  - Note: resulting residuals not corrected, so don't worry that they will show little change
  - Drawback: correlation is given by you, not estimated
  - See `?bam` on how to use arguments `AR.start` and `rho`; N.B. in the generalized case you must use `discrete=TRUE`

Model comparisons

Model comparisons

Don't.

Model comparisons

Don't.Well…

- For linear additive models (Gaussian errors, identity link), generally OK
- For generalized additive models, tricky due to the use of penalized quasi-likelihood: a computational trick used to make the likelihood easier to compute
- Used by `gam` (except with `outer` iteration), `gamm`, and `bam`
- For linear models, the penalized quasi-likelihood and the real likelihood are equivalent, but not so in the generalized case: model comparisons invalid!

Outer iteration

1. Set up the bases
2. Try a set of smoothing parameters
    1. Try a set of model coefficients given the smoothing parameters
    2. Repeat 1 until the model coefficients are optimal
3. Repeat 2 until the smoothing parameters are optimal

PQL

1. Set up the bases
2. Try a set of smoothing parameters and model coefficients
3. Repeat 2 until all parameters are optimal

Alternatives to model comparisons

- Double penalties: `select=TRUE`
- Shrinkage splines: `bs='ts'` instead of `bs='tp'`
- Hypothesis testing
- Parsimonious models

But if you must... my `R` package `buildmer` can do model comparisons for GAMs, and errors out appropriately if you're falling into the PQL trap

Inference

Hypothesis testing

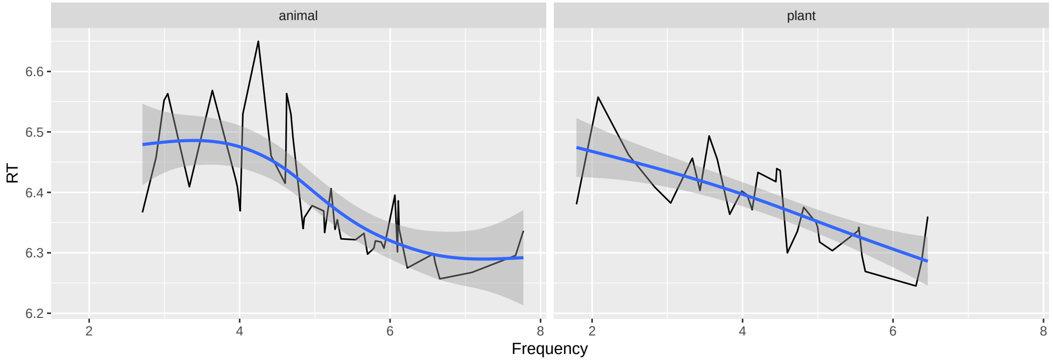
- Is my effect significant over the entire time course? $\rightarrow$ `summary` will tell you
- Where is my effect significant? $\rightarrow$ needs some work
- Two options: difference smooths and posterior inference

Difference smooths

- The difference-smooth method constructs two planned comparisons:
  1. A reference smooth (similar to an intercept)
  2. A difference smooth (similar to a contrast)
- The difference smooth is fitted as a parameter in the model
- Answer the question: ‘is there a smooth difference between two conditions?’

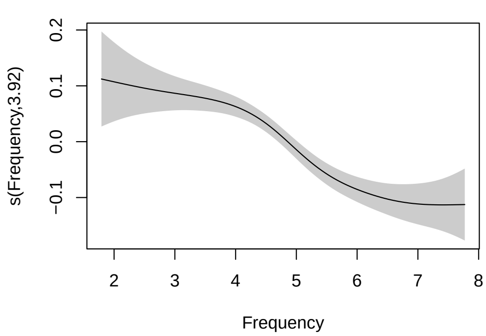
Difference smooths

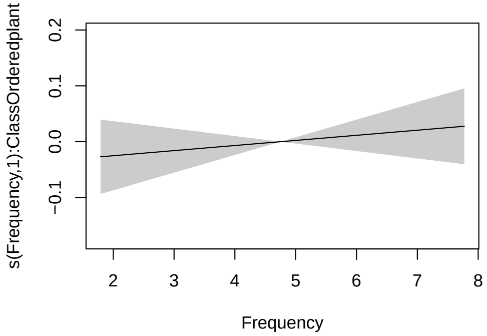
```
group_by(lexdec, Class, Frequency) |>
        summarize(RT = mean(RT)) |>
        ggplot(aes(Frequency,RT)) |>
        add(facet_wrap(~Class)) |>
        add(geom_line()) |>
        add(stat_smooth(method='gam'))
```



Difference smooths

```r
lexdec$ClassOrdered <- ordered(lexdec$Class) |> C(treatment)
model <- gam(RT ~ ClassOrdered + s(Frequency) + s(Frequency,by=ClassOrdered),
             data=lexdec,method='ML')
plot(model,rug=FALSE,shade=TRUE)
```



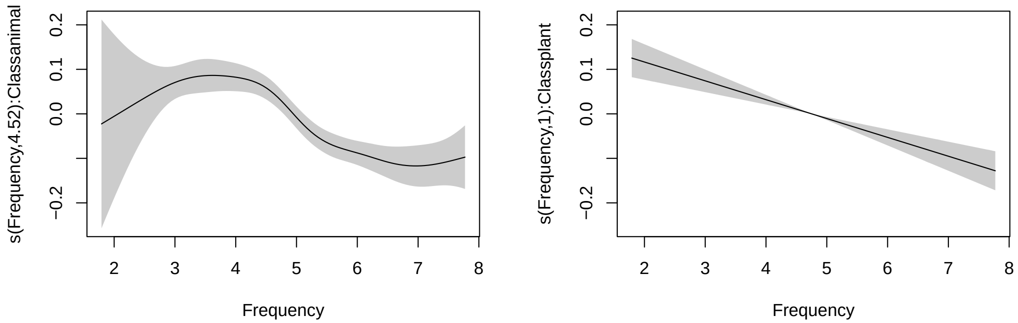


Posterior inference

- Posterior inference constructs post-hoc comparisons
- Constructs two independent smooths per factor combination:
  1. A smooth for animals
  2. A smooth for plants
- After fitting the model, the user computes the difference between the fitted smooths and tests it against zero

Fitting the model

```r
model <- gam(RT ~ Class + s(Frequency,by=Class),data=lexdec,method='REML')
plot(model,rug=FALSE,shade=TRUE)
```



Posterior inference: the procedure

We are going to perform a completely standard Wald test of the difference between two fitted smooths

1. We obtain the linear predictors from the model for both animals and plants over a grid of, e.g., 100 points
2. We subtract the two
3. We optionally zero out random effects
4. We multiply them with the model coefficients
5. We compute a 95% credible interval for the difference (N.B.: requires REML fit!)

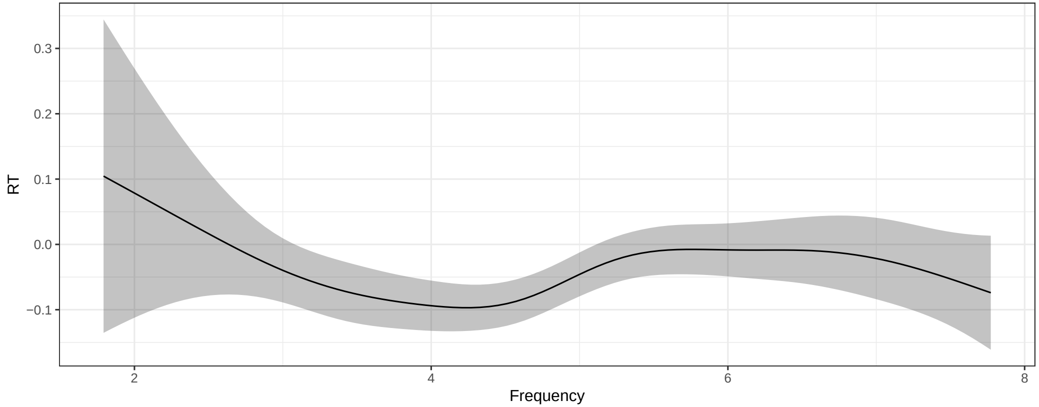
Pro-Tip: this is all automated by function `plot_diff` from package `itsadug` (but with limitations, so eventually you'll come back here!)

Computing posterior differences

```
minmax <- range(lexdec$Frequency)
newdata <- data.frame(Frequency = seq(minmax[1],minmax[2],length.out=100))
newdata.plants  <- transform(newdata, Class = 'plant')
newdata.animals <- transform(newdata, Class = 'animal')
lp.plants  <- predict(model,newdata=newdata.plants,type='lpmatrix')
lp.animals <- predict(model,newdata=newdata.animals,type='lpmatrix')
lp.diff <- lp.plants - lp.animals
newdata <- mutate(newdata,
                  RT = lp.diff %*% coef(model),
                  SE = sqrt(diag(lp.diff %*% vcov(model) %*% t(lp.diff))),
                  cimin = RT - 1.96 * SE,
                  cimax = RT + 1.96 * SE)
```

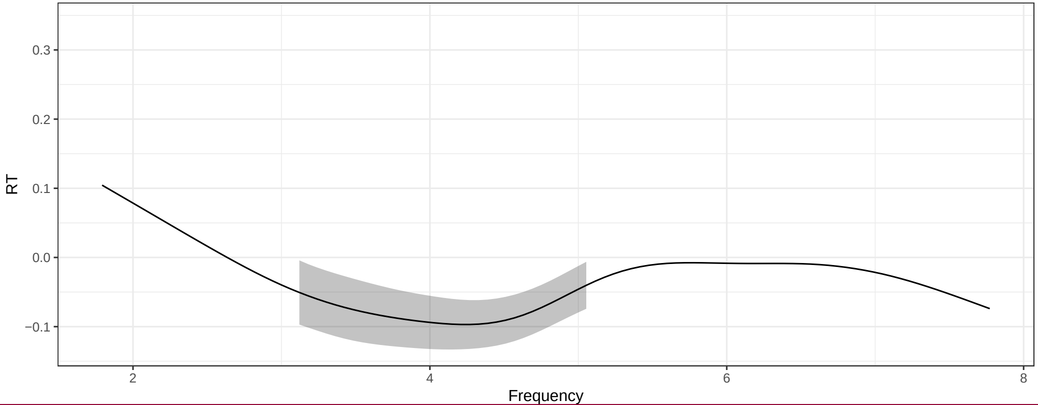
Computing posterior differences

```r
ggplot(newdata,aes(Frequency,RT)) +
        geom_ribbon(aes(ymin=cimin,ymax=cimax),alpha=.3) +
        geom_line() +
        theme_bw()
```



Highlighting significance

```r
newdata$cimin[sign(newdata$cimin) != sign(newdata$cimax)] <- NA
ggplot(newdata,aes(Frequency,RT)) +
        geom_ribbon(aes(ymin=cimin,ymax=cimax),alpha=.3) +
        geom_line() +
        theme_bw()
```



Wrapping up

What did you (hopefully) learn today?

1. What problem do GAMs solve?
2. Smoothing splines
3. `mgcv`
4. Checking assumptions
5. Model comparisons
6. Inference
7. Wrapping up

Getting your feet wet

- Today was really a crash course
- Hopefully, you have now received enough baggage from me to get started analyzing your own data
- Next week: working with real data
- If you have data of your own: bring them, try to analyze them yourself, and I'll be there to help you if you get stuck
- For those of you who don't have any original data, I'll arrange two example problems

Moving on from here

- Go through Jacolien van Rij's excellent but slightly dated tutorial: https://jacolienvanrij.com/Tutorials/GAMM.html
- Read Wood (2017)
- Look into SCAMs
- Look into quantile GAMs

Thank you for your attention!

References

Baayen, H., Vasishth, S., Kliegl, R., & Bates, D. (2017). The cave of shadows: Addressing the human factor with generalized additive mixed models. *Journal of Memory and Language*, *94*, 206–234. https://doi.org/10.1016/j.jml.2016.11.006

Hintz, F., Voeten, C. C., McQueen, J. M., & Scharenborg, O. (2021). The effects of onset and offset masking on the time course of non-native spoken-word recognition in noise. In T. Fitch, C. Lamm, H. Leder, & K. Tessmar (Eds.), *Proceedings of the 43rd annual conference of the Cognitive Science Society (CogSci 2021)*. Cognitive Science Society. https://doi.org/10.31234/osf.io/nx748

Hodges, J. S. (2016). *Richly parameterized linear models: Additive, time series, and spatial models using random effects*. CRC Press.

Wood, S. N. (2017). *Generalized additive models: An introduction with R* (2nd ed.). Chapman & Hall/CRC.