An introduction to GAM(M)s

Stefano Coretta

12/07/2018

Outline

Part 1

- Linear models
- Introduction to GAM theory
- Comparing groups
- Significance testing

Part 2

- Dynamic data
- Random effects
- Interactions

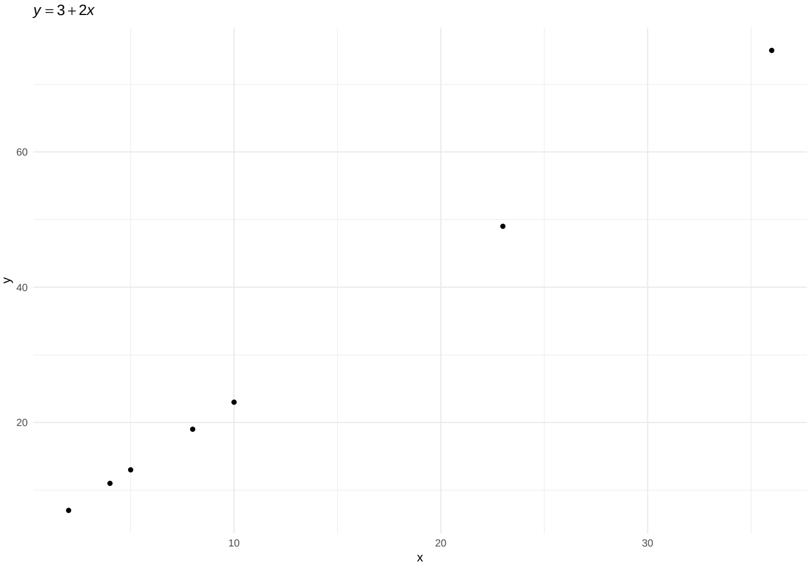
Time travel...

Linear models

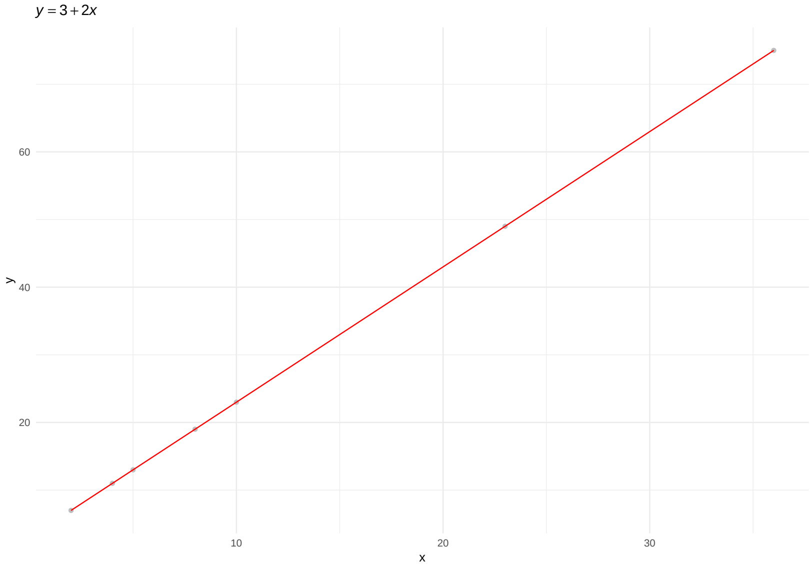
$$y = 3 + 2x$$

where $x = (2, 4, 5, 8, 10, 23, 36)$

Linear models



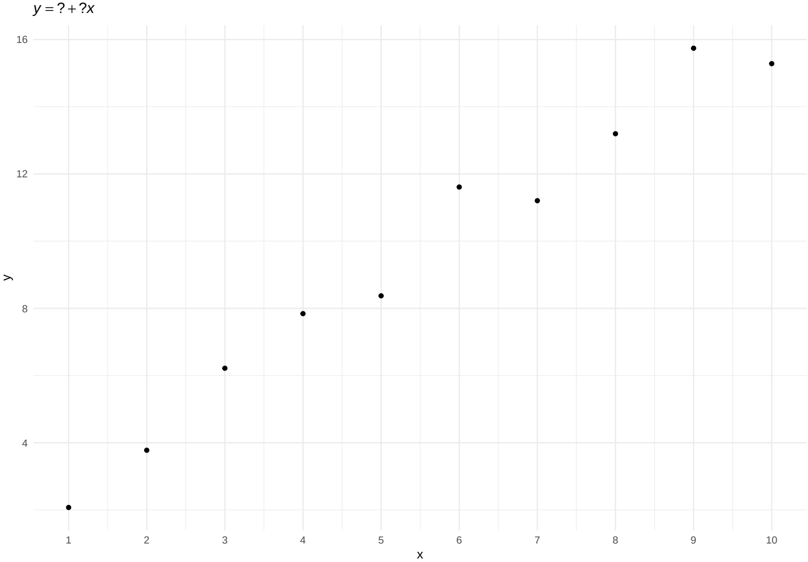
Linear models



Linear models

- In science, we have x and y...
- for example, vowel duration and VOT, speech rate and pitch, etc...

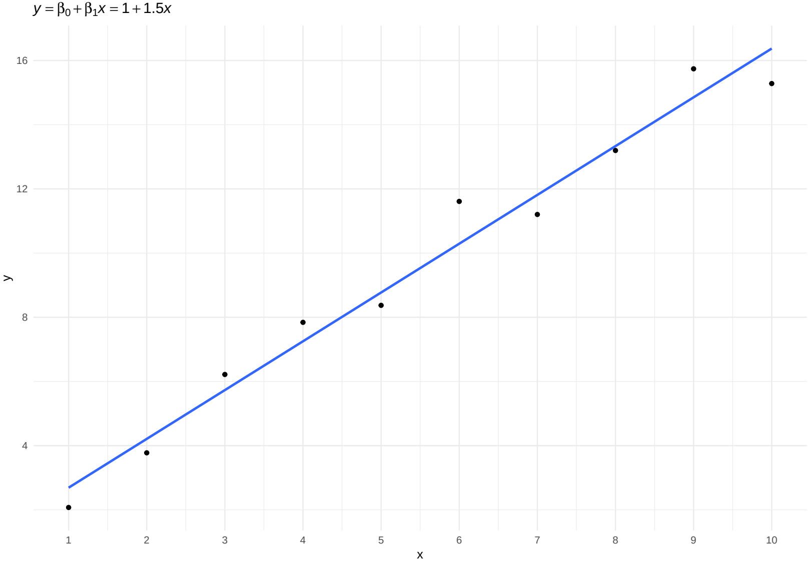
Linear models



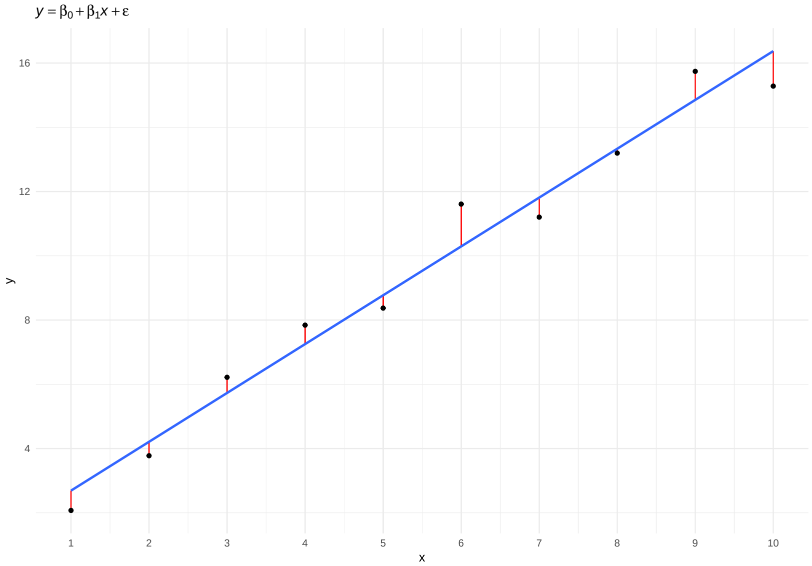
Linear models

- The formula: $y = \beta_0 + \beta_1 x$
  - β_0 is the **intercept**
  - β_1 is the **slope**
- We know x and y
  - we need to estimate β_0, $\beta_1 = \hat{\beta}_0, \hat{\beta}_1$
- We can add more predictors
  - $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$
- `lm(y ~ x, data)` ('y as a function of x')

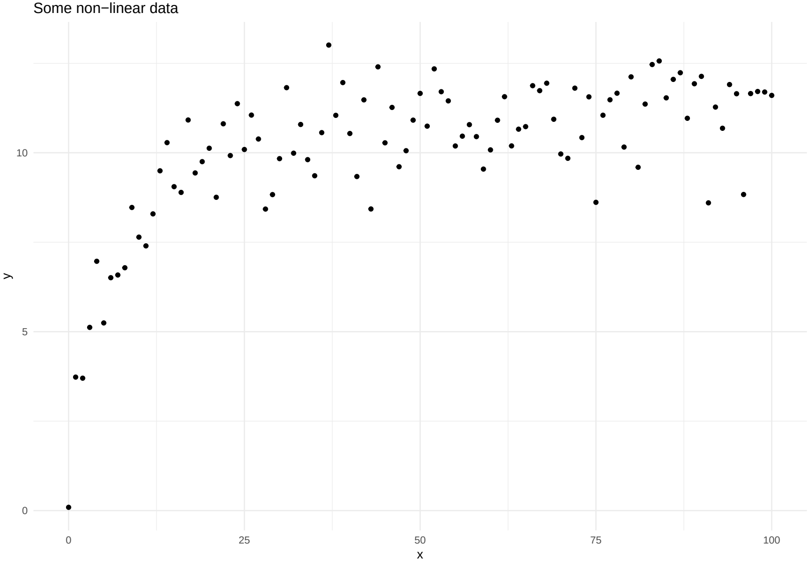
Linear models



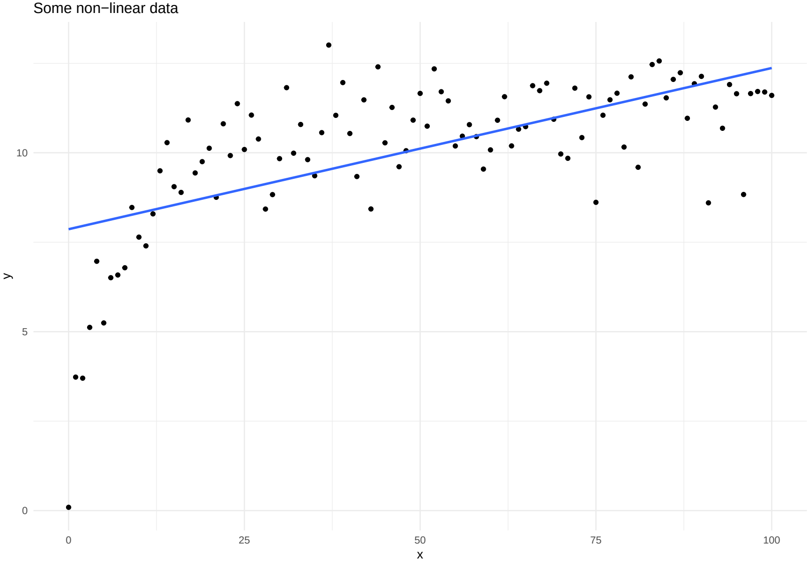
Linear models



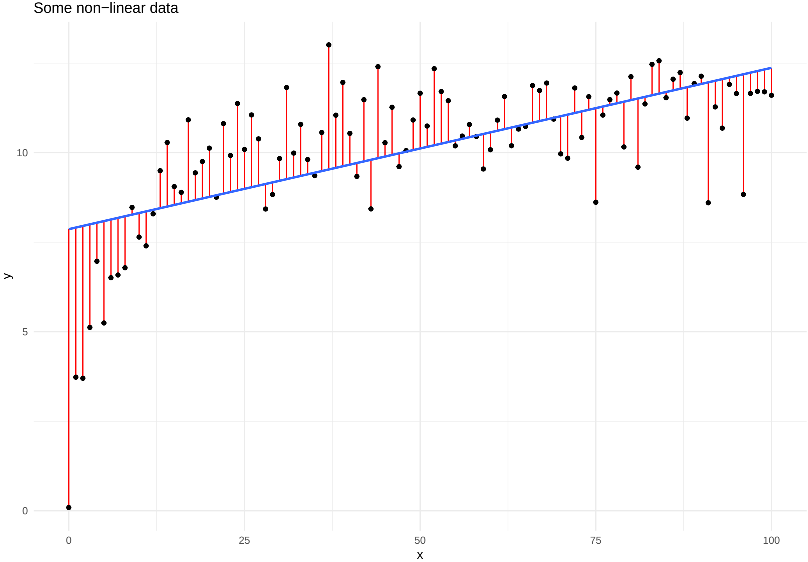
LM with non-linear data



LM with non-linear data



LM with non-linear data

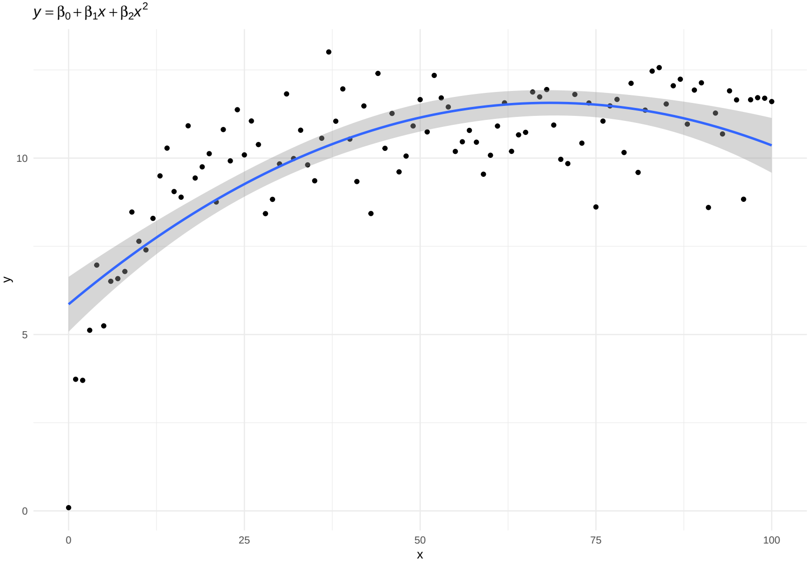


LM with non-linear data

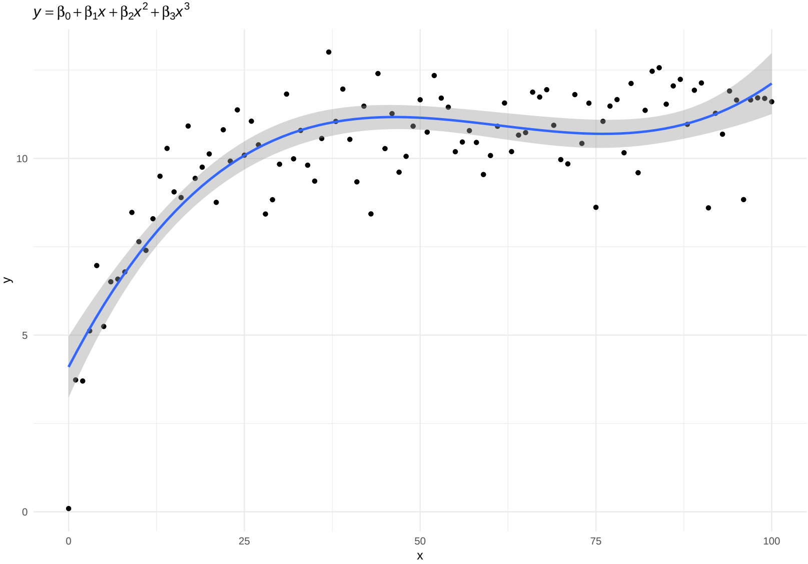
How to account for non-linearity in a linear model?

- Use **higher-degree polynomials**
  - quadratic: $y = \beta_0 + \beta_1 x + \beta_2 x^2$
  - cubic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$
  - nth: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + ... + \beta_n x^n$

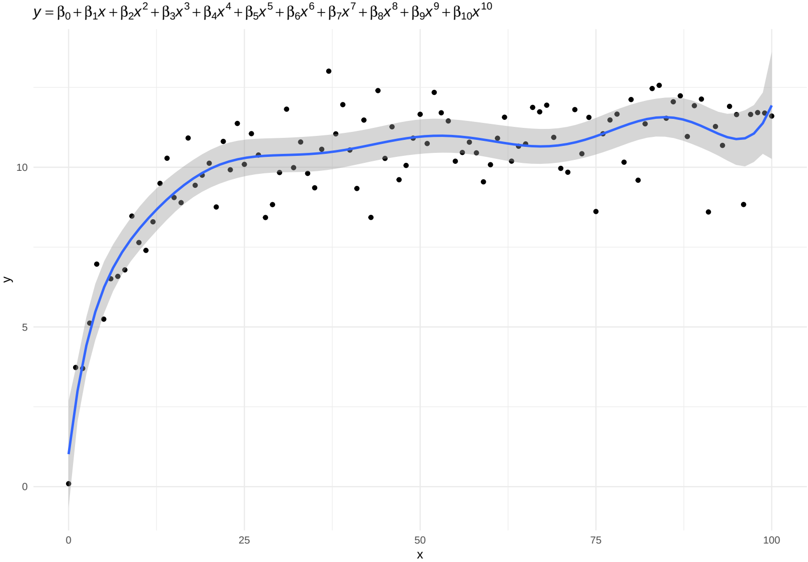
LM with non-linear data



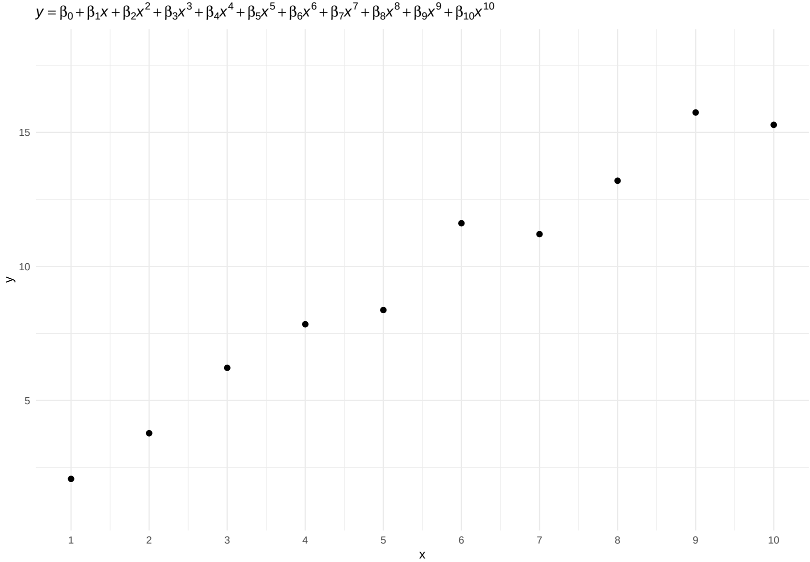
LM with non-linear data



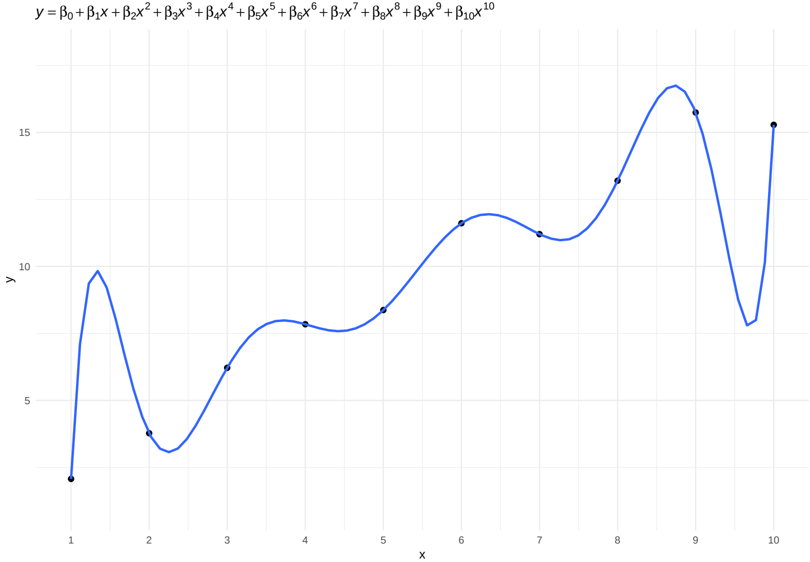
LM with non-linear data



LM with non-linear data



LM with non-linear data



Generalised additive models

- **G**enrealised **A**dditive **M**odel**s**
- $y = f(x) + \epsilon$
  - $f(x) =$ 'some function of x' (or *smooth function*)

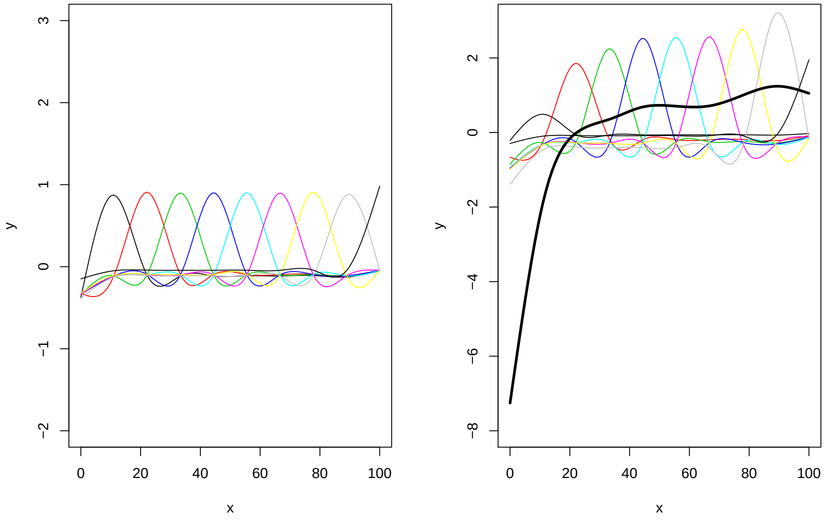
Smooth terms

- LMs have **parametric terms**
  - $\beta_n x_n$
  - `x` in `R`
  - linear effects
- GAMs add (non-parametric) **smooth terms** (or simply smooths, also smoothers)
  - $f(x)$
  - `s(x)` in `R`
  - non-linear effects
- `gam(y ~ s(x), data)`, 'y as *some* function of x'

Smoothing splines, basis, basis functions

- Smooths in GAMs are **smoothing splines**
  - splines are defined piecewise with a set of polynomials
- The set of polynomials is called a **basis**
  - the basis is composed of **basis functions** (the polynomials)
- A spline is the sum of the products of each basis function and its coefficient

Basis functions and knots



Smoothing parameter

- 'Wiggliness' is related to number of basis functions
  - more basis functions, more wiggliness (less smoothing)
- The **smoothing parameter** penalises wiggliness
  - high values = less wiggliness (more smoothing)
  - estimated from the data

Smoothing splines

- There are **several kinds** of splines
  - each with their own basis functions
- Most common
  - *thin plate regression splines*
  - *cubic regression splines*
- For more info, run `?smooth.terms`

A simple GAM

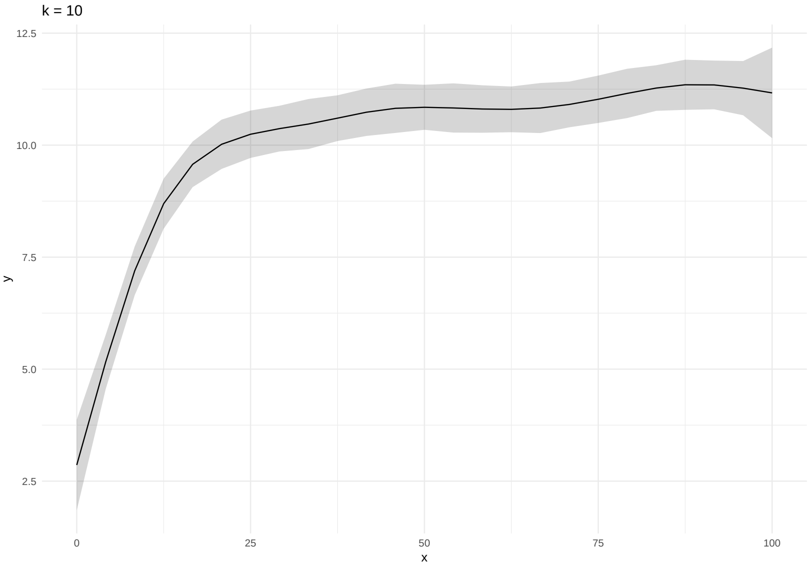
```r
simple <- gam(
  y ~
    s(x, bs = "cr", k = 10),
  data = sim_nl_a
)
```

A simple GAM

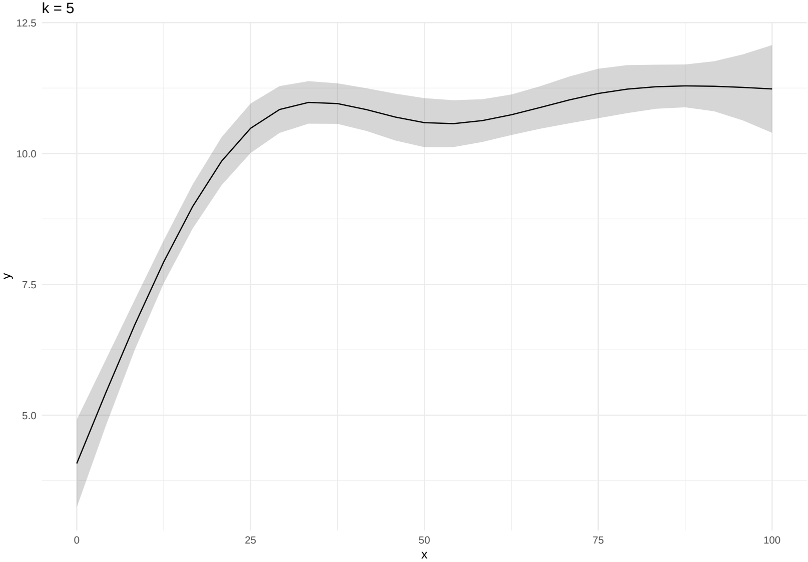
```r
summary(simple)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ s(x, bs = "cr", k = 10)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.1165     0.1028   98.37   <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(x) 6.939   8.01 38.69  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.755   Deviance explained = 77.2%
## GCV = 1.1593  Scale est. = 1.0681    n = 101
```

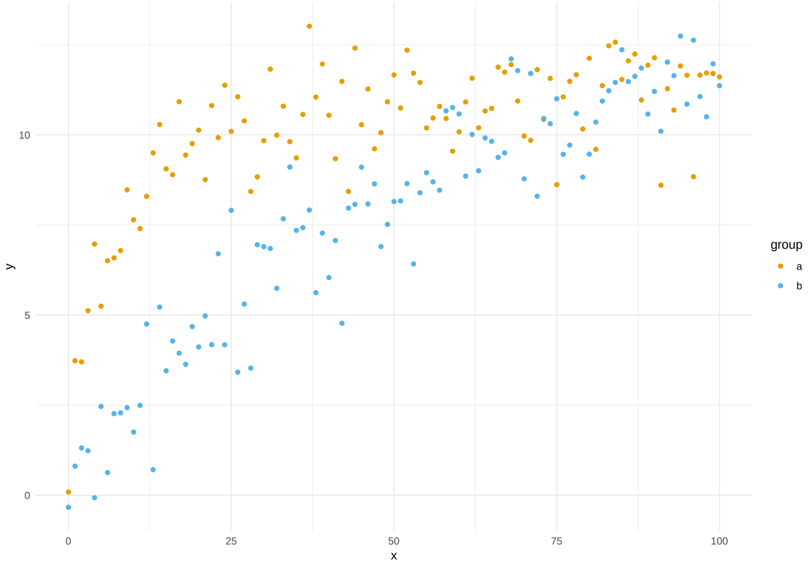
A simple GAM



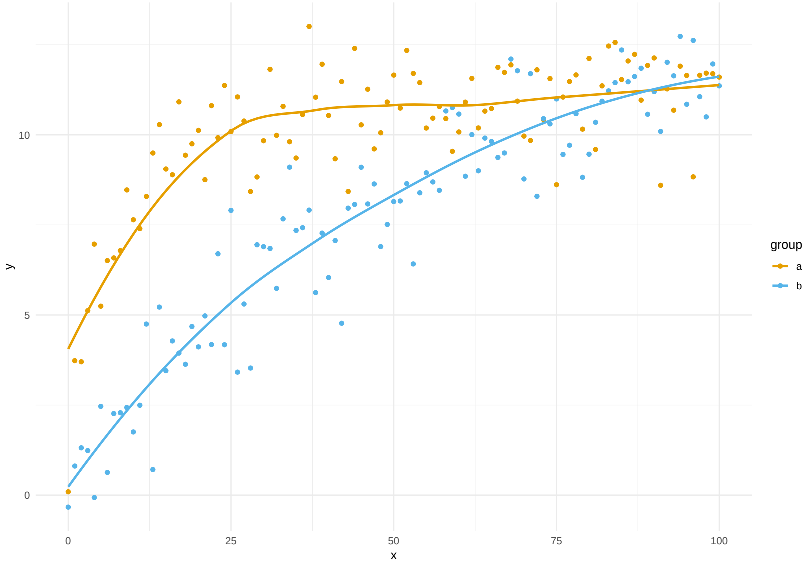
A simple GAM:



Comparing levels



Comparing levels



Comparing levels

- by-variables with ordered factors

```
compare <- gam(
  y ~
    # parametric term
    group +
    # reference smooth
    s(x, bs = "cr", k = 5) +
    # difference smooth
    s(x, bs = "cr", k = 5, by = group),
  data = sim_nl
)
```

Comparing levels

- To use `by`-variables with ordered factors
  - change factor to **ordered factor**
  - change factor contrast to **treatment contrast** (`contr.treatment`)
    - the default in ordered factors is `contr.poly`, this won't work
  - include factor as **parametric term**
  - include a **reference smooth** and a **difference smooth** with the `by`-variable

Comparing levels

```r
sim_nl <- sim_nl %>%
  mutate(group = ordered(group, levels = c("a", "b")))
contrasts(sim_nl$group) <- "contr.treatment"
```

Comparing levels

```r
library(mgcv)


compare <- gam(
  y ~
    # parametric term
    group +
    # reference smooth
    s(x, bs = "cr", k = 5) +
    # difference smooth
    s(x, bs = "cr", k = 5, by = group),
  data = sim_nl
)
```

Comparing levels

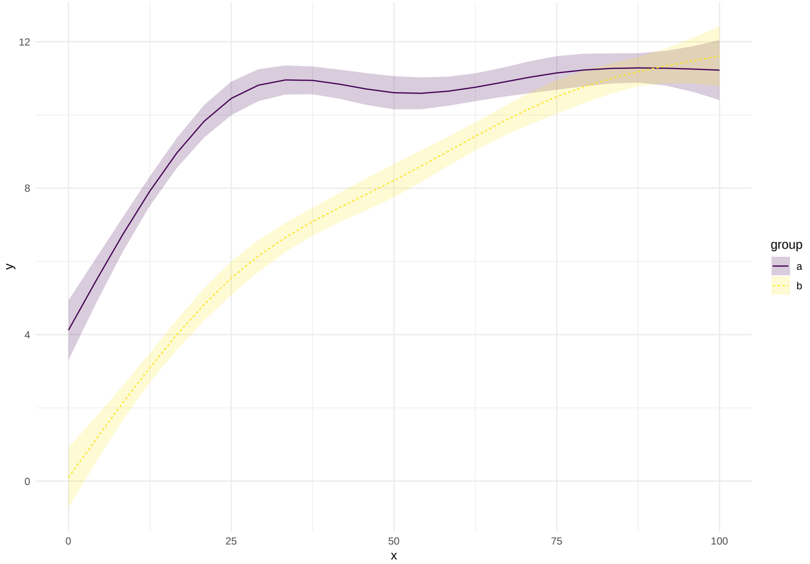
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ group + s(x, bs = "cr", k = 5) + s(x, bs = "cr", k = 5, by = group)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.1165     0.1096   92.34   <2e-16 ***
## groupb       -2.4947     0.1549  -16.10   <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(x)        4.000  4.000 64.99  <2e-16 ***
## s(x):groupb 3.576  3.896 39.67  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.873   Deviance explained = 87.8%
## GCV = 1.2725  Scale est. = 1.2122    n = 202
```

Comparing levels

```r
library(tidymv)

plot_smooths(
  model = compare,
  time_series = x,
  comparison = group
)
```

Comparing levels



Significance testing

- Several ways for testing significance of smooths
- We will used a combined method
  - model comparison with `itsadug::compareML()` of a full and a null model
  - visualisation of the difference smooth with `tidymv::plot_difference()`
    - (you can also use `itsadug::plot_diff()`)
- Caveats
  - models need to be fitted with `method = "ML"`
  - only for testing fixed effects

Significance testing

```r
group_full <- gam(
  y ~
    group +
    s(x, bs = "cr", k = 5) +
    s(x, bs = "cr", k = 5, by = group),
  data = sim_nl,
  method = "ML"
)

group_null <- gam(
  y ~
    s(x, bs = "cr", k = 5),
  data = sim_nl,
  method = "ML"
)
```

Significance testing

```
compareML(group_null, group_full)
```

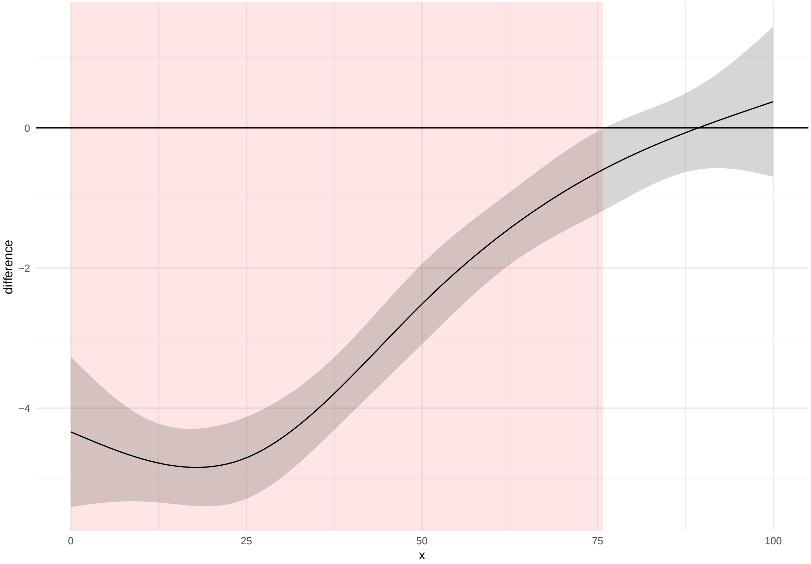
```
## group_null: y ~ s(x, bs = "cr", k = 5)
##
## group_full: y ~ group + s(x, bs = "cr", k = 5) + s(x, bs = "cr", k = 5, by =
##
## Chi-square test of ML scores
## -----
##        Model    Score Edf Difference    Df  p.value Sig.
## 1 group_null 422.4827   3
## 2 group_full 314.0105   6    108.472 3.000  < 2e-16  ***
##
## AIC difference: 221.22, model group_full has lower AIC.
```

Significance testing

- Let's plot the difference smooth with `tidymv::plot_difference()`

```
plot_difference(
  model = group_full,
  time_series = x,
  comparison = list(group = c("b", "a"))
)
```

Significance testing



Practical 1

Dynamic data

- "Dynamic speech analysis is a term used to refer to analyses that look at measureable quantities of speech that **vary in space and/or time**" (Sóskuthy, 2017)
- Examples
  - formant trajectories
  - pitch contours
  - geographic (*diatopic*) variation
  - tongue contours

Dynamic data

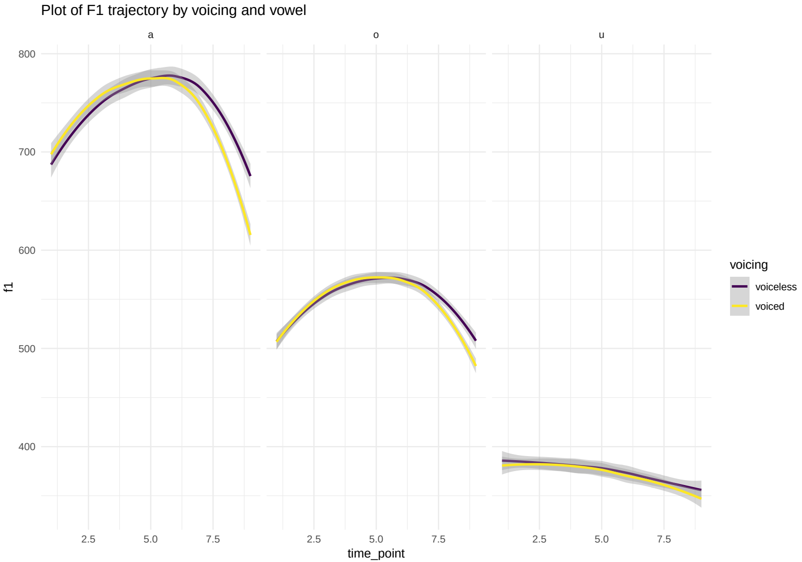
- Two main types
  - **time series data**
  - **spatial data**
- More data ($n > 1000$)
  - use `bam()` (big GAM) instead of `gam()`

Dynamic data

- formant trajectories (time series data)

```
## # A tibble: 10,705 x 13
##    speaker index word  time_point    f1    f2    f3    f0 duration vowel
##    <fct>   <chr> <fct>      <int> <dbl> <dbl> <dbl> <dbl>    <dbl> <ord>
##  1 it01    it01~ pugu           1  308.  797. 2280.  137.     95.2 u
##  2 it01    it01~ pugu           2  315.  779. 2124.  134.     95.2 u
##  3 it01    it01~ pugu           3  316.  786. 2314.  134.     95.2 u
##  4 it01    it01~ pugu           4  314.  789. 2374.  135.     95.2 u
##  5 it01    it01~ pugu           5  313.  737. 2307.  137.     95.2 u
##  6 it01    it01~ pugu           6  305.  717. 2315.  138.     95.2 u
##  7 it01    it01~ pugu           7  291.  713. 2318.  138.     95.2 u
##  8 it01    it01~ pugu           8  280.  733. 2308.  137.     95.2 u
##  9 it01    it01~ pugu           9  287.  784. 2329.  136.     95.2 u
## 10 it01    it01~ pada           1  651. 1119. 2155.  120.    139.  a
## # ... with 10,695 more rows, and 3 more variables: voicing <ord>,
## #   place <ord>, vow_voi <ord>
```

Dynamic data



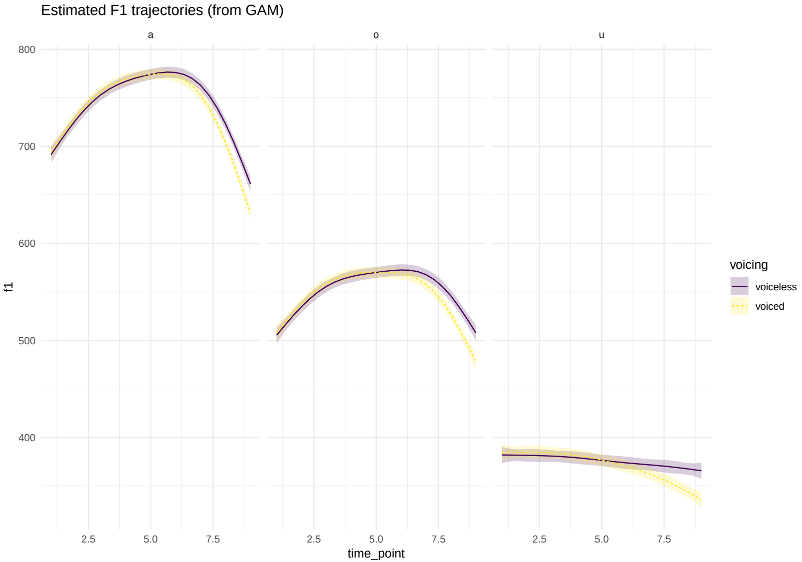
Dynamic data

```
big_gam <- bam(
  f1 ~
    voicing +
    vowel +
    s(time_point, k = 6) +
    s(time_point, k = 6, by = voicing) +
    s(time_point, k = 6, by = vowel),
  data = vowels
)
```

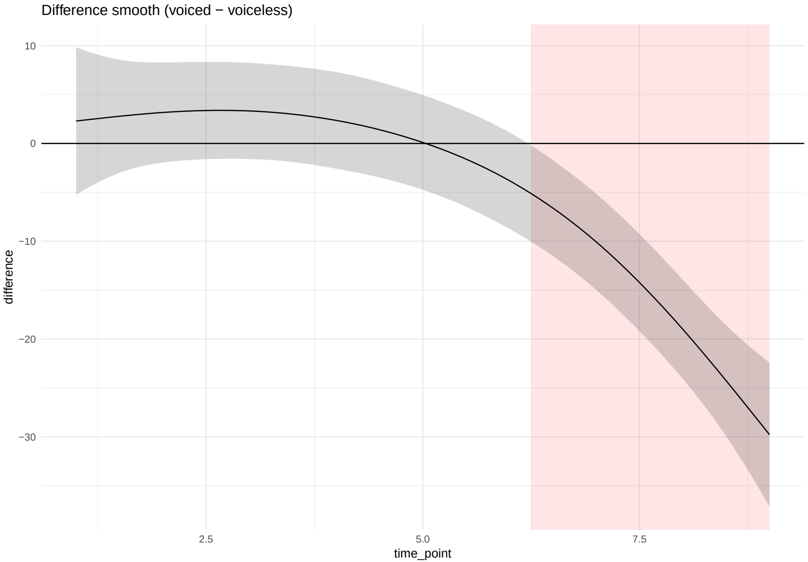
Dynamic data

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## f1 ~ voicing + vowel + s(time_point, k = 6) + s(time_point, k = 6,
##     by = voicing) + s(time_point, k = 6, by = vowel)
##
## Parametric coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept)    737.722      1.476  499.932  < 2e-16 ***
## voicingvoiced   -5.788      1.480   -3.911 9.25e-05 ***
## vowelo        -190.218      1.799 -105.734  < 2e-16 ***
## vowelu        -362.253      1.822 -198.779  < 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_point)                4.816  4.944 140.16  < 2e-16 ***
## s(time_point):voicingvoiced 2.737  3.331  17.28 7.44e-12 ***
## s(time_point):vowelo        3.663  4.266  17.69 5.40e-15 ***
## s(time_point):vowelu        4.610  4.903  83.36  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Dynamic data



Dynamic data



Random effects

- Only **fixed effects** so far...
- **G**eneralised **A**dditive **M**ixed **M**odel (GAMM)
  - fixed + random effects
- Include a **random smooth** term with the **factor smooth interaction** as basis

Random effects

- Factor smooth interaction
  - `bs = "fs"`
  - a smooth is fitted at each level of a factor
- the random effect variable *needs to be a factor*
- `s(time, speaker, bs = "f")`

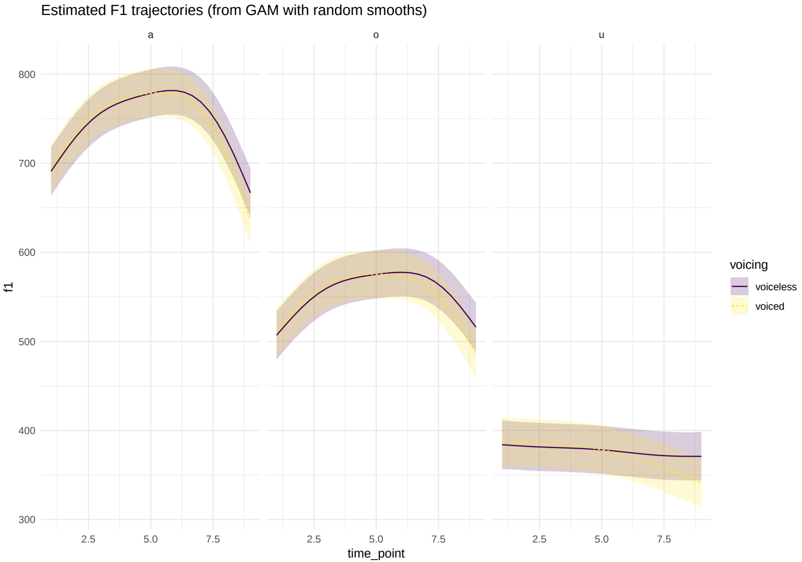
Random effects

```
random_gam <- bam(
  f1 ~
    voicing +
    vowel +
    s(time_point, k = 6) +
    s(time_point, k = 6, by = voicing) +
    s(time_point, k = 6, by = vowel) +
    # random smooth
    s(time_point, speaker, bs = "fs", m = 1),
  data = vowels
)
```

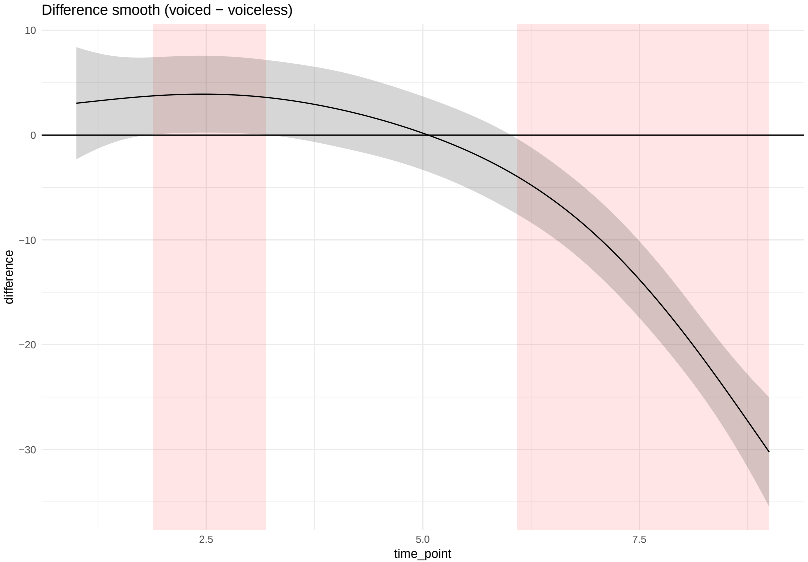
Random effects

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## f1 ~ voicing + vowel + s(time_point, k = 6) + s(time_point, k = 6,
##     by = voicing) + s(time_point, k = 6, by = vowel) + s(time_point,
##     speaker, bs = "fs", m = 1)
##
## Parametric coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept)    741.732     13.384   55.418  < 2e-16 ***
## voicingvoiced   -5.489      1.008   -5.444 5.32e-08 ***
## vowelo        -189.579      1.226 -154.652  < 2e-16 ***
## vowelu        -364.484      1.251 -291.430  < 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_point)                4.853   4.927 152.71  <2e-16 ***
## s(time_point):voicingvoiced  3.174   3.808  33.58  <2e-16 ***
## s(time_point):vowelo         4.194   4.693  37.20  <2e-16 ***
## s(time_point):vowelu         4.811   4.969 189.35  <2e-16 ***
## s(time_point,speaker)       84.021 142.000  86.87  <2e-16 ***
```

Random effects



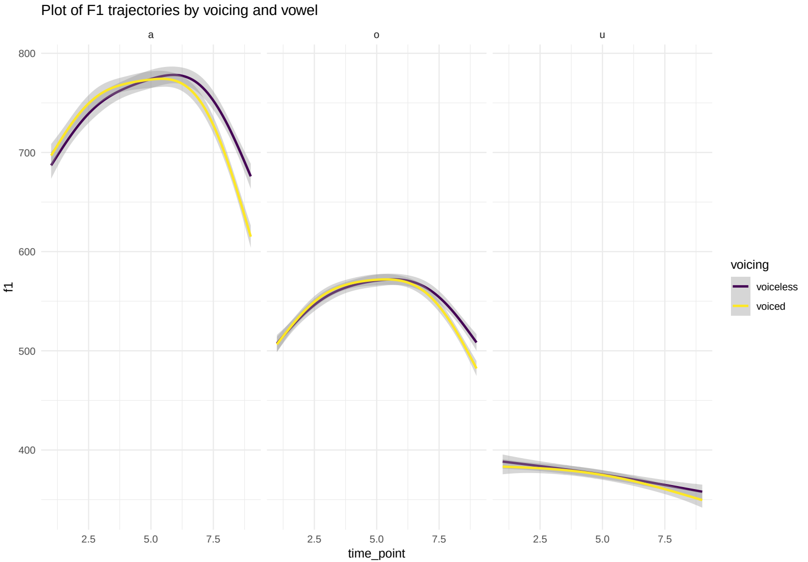
Random effects



Random effects

- You can also include classical random intercepts and slopes
- random intercept
  - `s(speaker, bs = "re")`
- random slope
  - `s(speaker, time_point, "re")`

Interactions



Interactions

- Use factor `by`-variable with the interaction of the terms
- Create the interaction with `interaction()`
  - be sure it is an ordered factor

Interactions

```r
vowels <- vowels %>%
  mutate(
    vow_voi = interaction(vowel, voicing),
    vow_voi = as.ordered(vow_voi)
  )
```

Interactions

```r
vowel_gam <- bam(
  f1 ~
    vow_voi +
    s(time_point, k = 6) +
    s(time_point, by = vow_voi, k = 6) +
    s(time_point, speaker, bs = "fs", m = 1),
  data = vowels
)
```

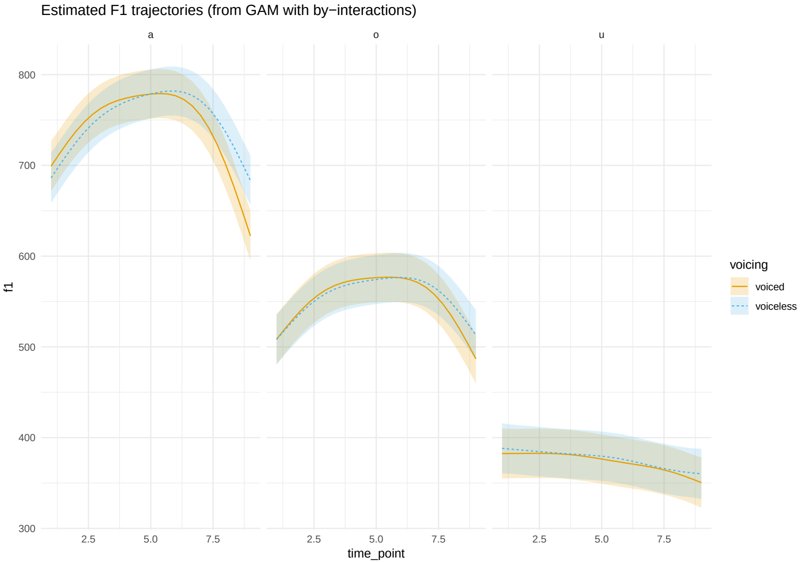
Interactions

```
## Family: gaussian
## Link function: identity
##
## Formula:
## f1 ~ vow_voi + s(time_point, k = 6) + s(time_point, by = vow_voi,
##     k = 6) + s(time_point, speaker, bs = "fs", m = 1)
##
## Parametric coefficients:
##                     Estimate Std. Error  t value Pr(>|t|)
## (Intercept)          743.477     13.408   55.448  < 2e-16 ***
## vow_voio.voiceless -192.215      1.729 -111.157  < 2e-16 ***
## vow_voiu.voiceless -367.254      1.757 -209.026  < 2e-16 ***
## vow_voia.voiced      -9.052      1.732   -5.225 1.77e-07 ***
## vow_voio.voiced    -195.955      1.728 -113.399  < 2e-16 ***
## vow_voiu.voiced    -370.751      1.758 -210.867  < 2e-16 ***
## ---
```

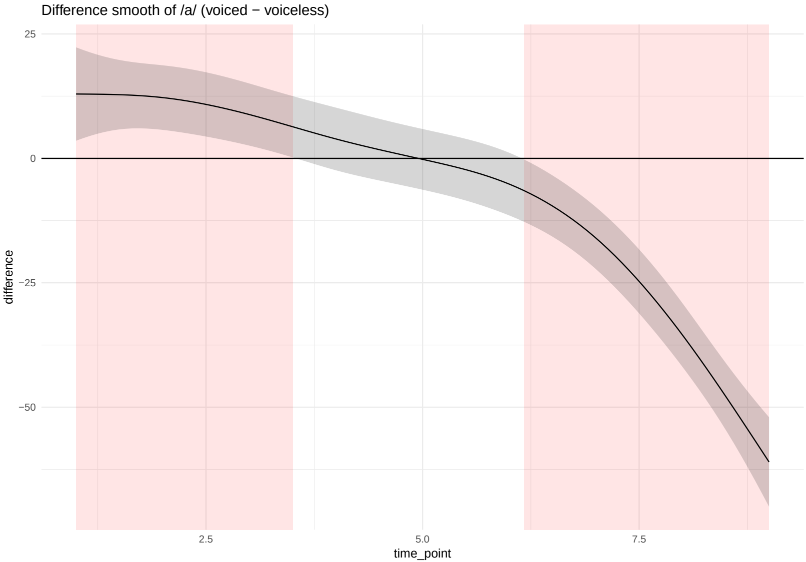
Interactions

```
## Approximate significance of smooth terms:
##                                     edf  Ref.df       F  p-value    
## s(time_point)                     4.838   4.908 113.935  < 2e-16 ***
## s(time_point):vow_voio.voiceless  3.612   4.215  11.604 1.22e-09 ***
## s(time_point):vow_voiu.voiceless  4.668   4.926  89.411  < 2e-16 ***
## s(time_point):vow_voia.voiced     3.715   4.307  43.703  < 2e-16 ***
## s(time_point):vow_voio.voiced     2.863   3.455   8.455 4.79e-06 ***
## s(time_point):vow_voiu.voiced     4.584   4.894  81.688  < 2e-16 ***
## s(time_point,speaker)            84.132 142.000  87.515  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.905   Deviance explained = 90.6%
## fREML =  57607  Scale est. = 2700.4    n = 10705
```

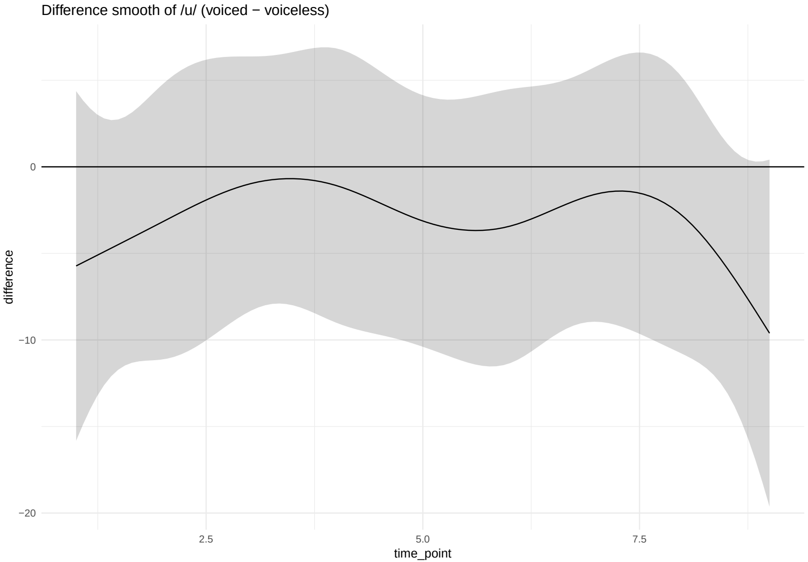
Interactions



Interactions



Interactions



Practical 2

Sóskuthy, Márton. 2017. Generalised additive mixed models for dynamic analysis in linguistics: a practical introduction. arXiv preprint arXiv:1703.05339.