An introduction to GAM(M)s

Stefano Coretta

12/07/2018

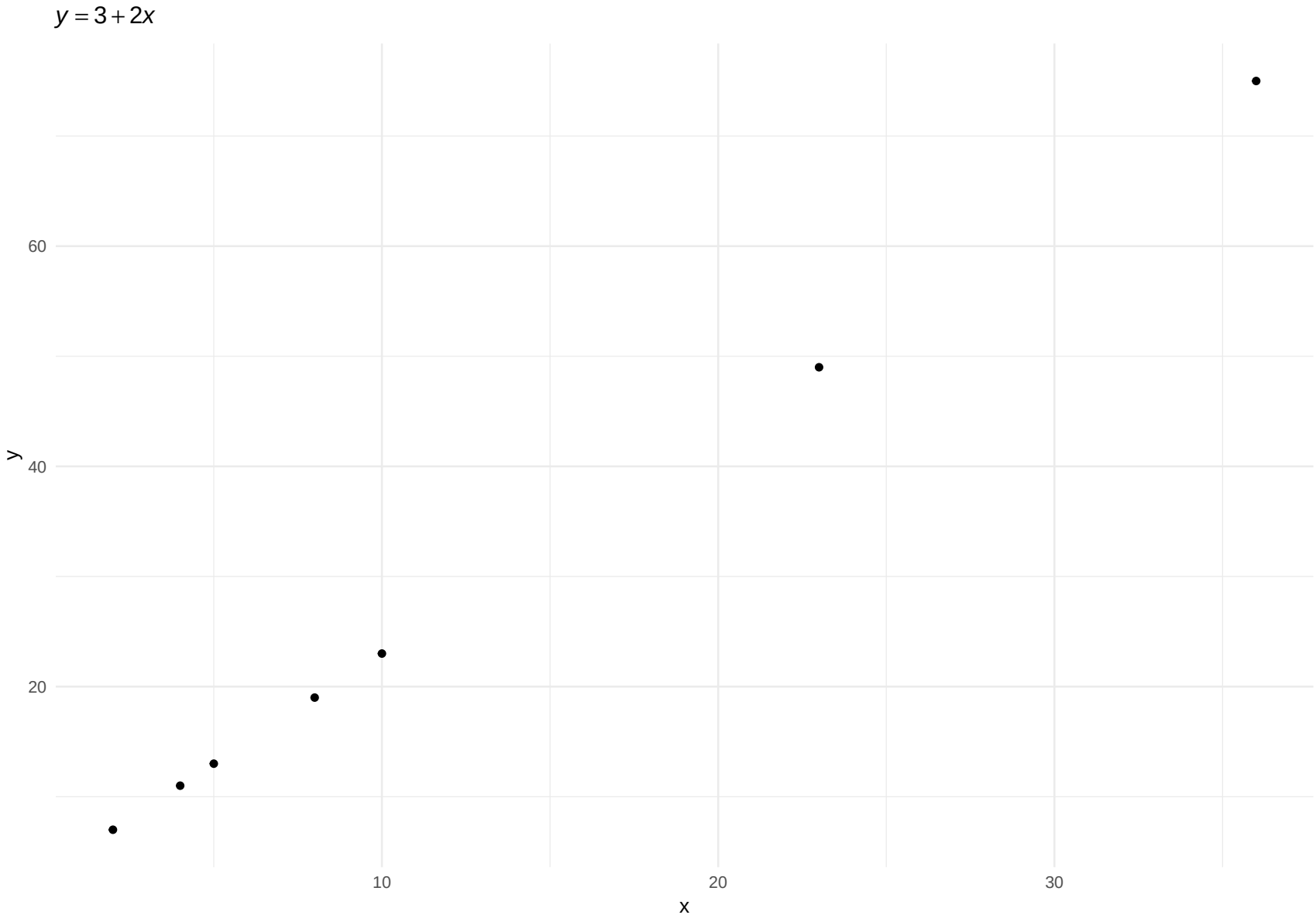
Time travel. . .

Linear models

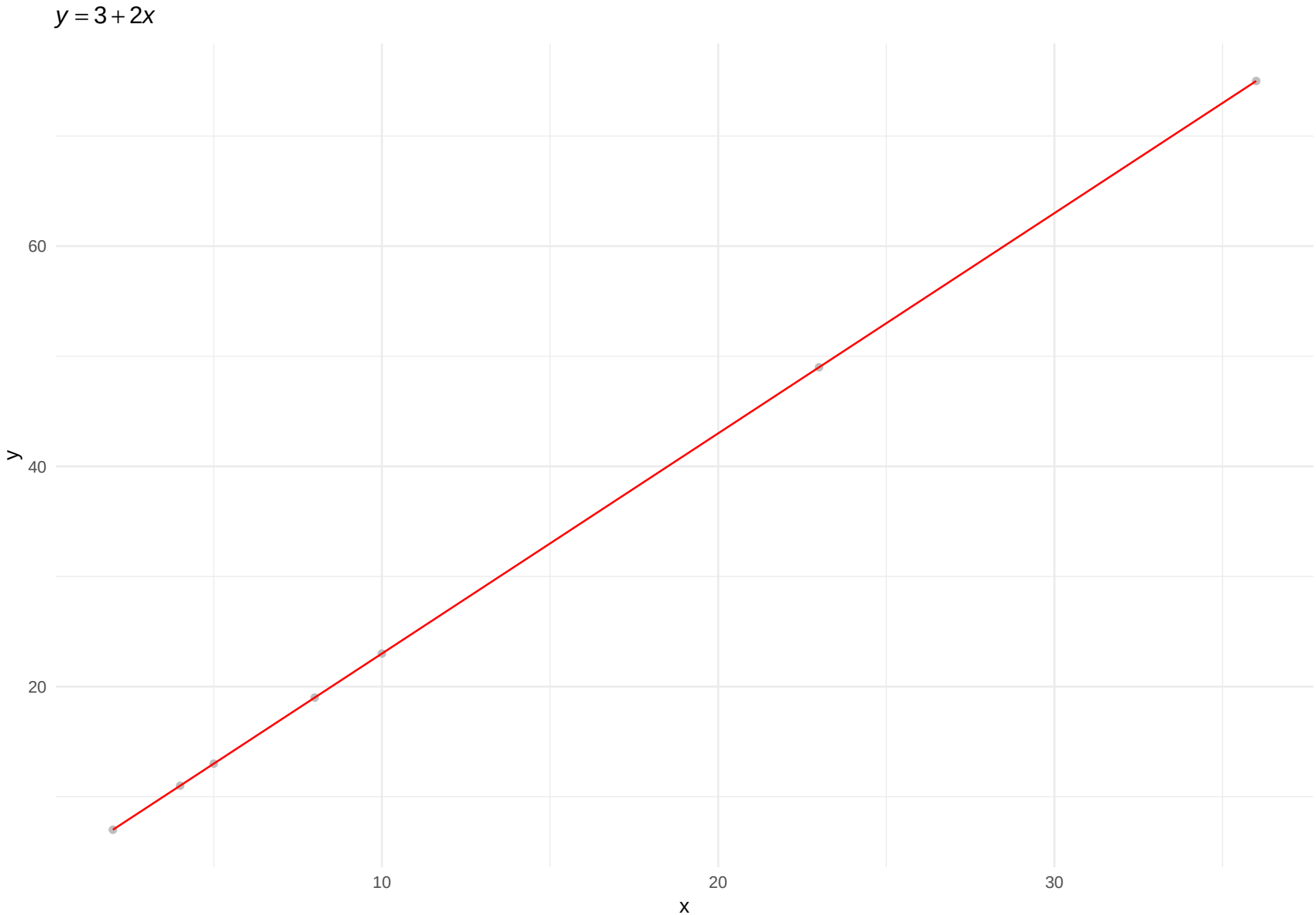
$$y = 3 + 2x$$

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

Linear models



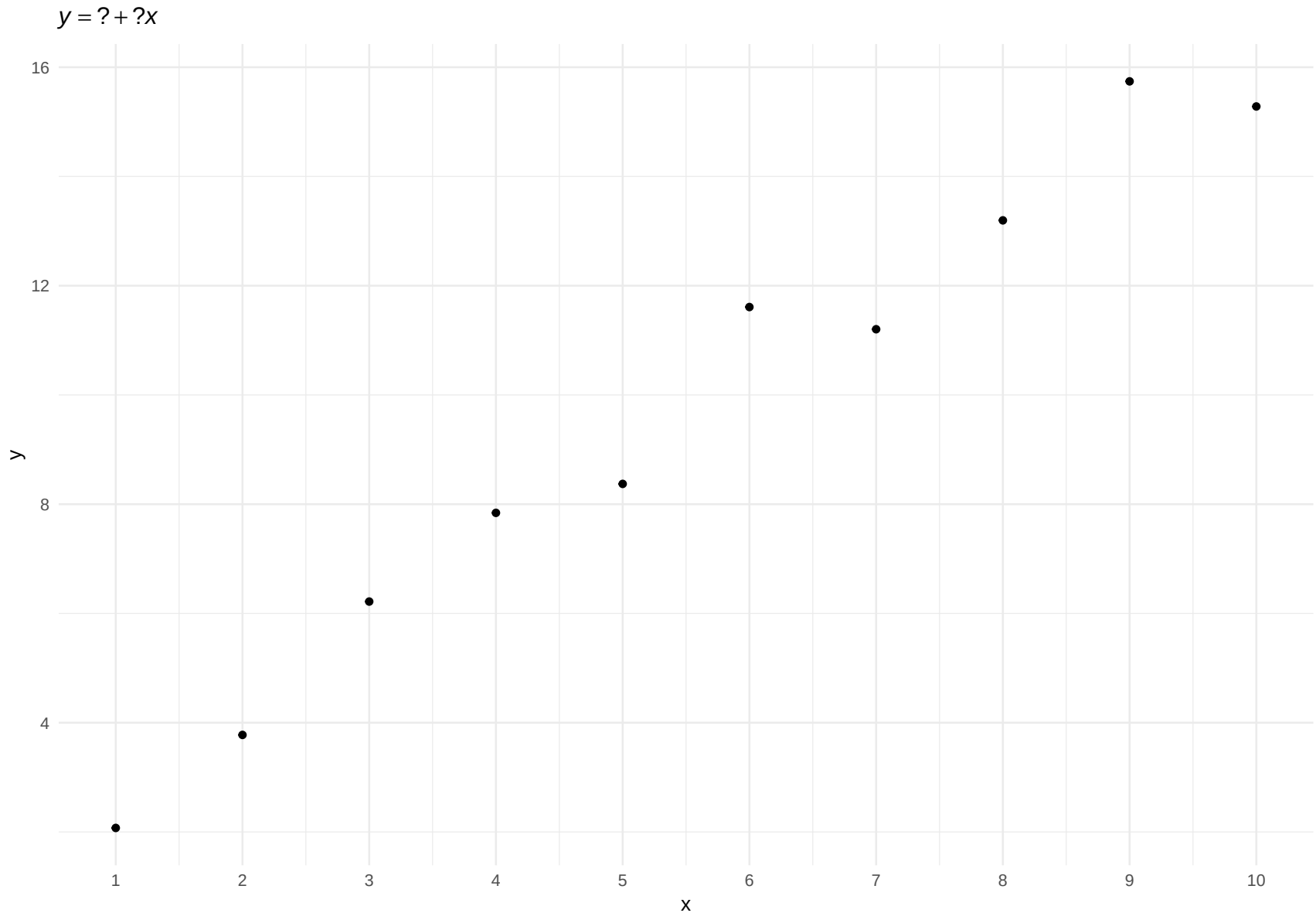
Linear models



Linear models

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

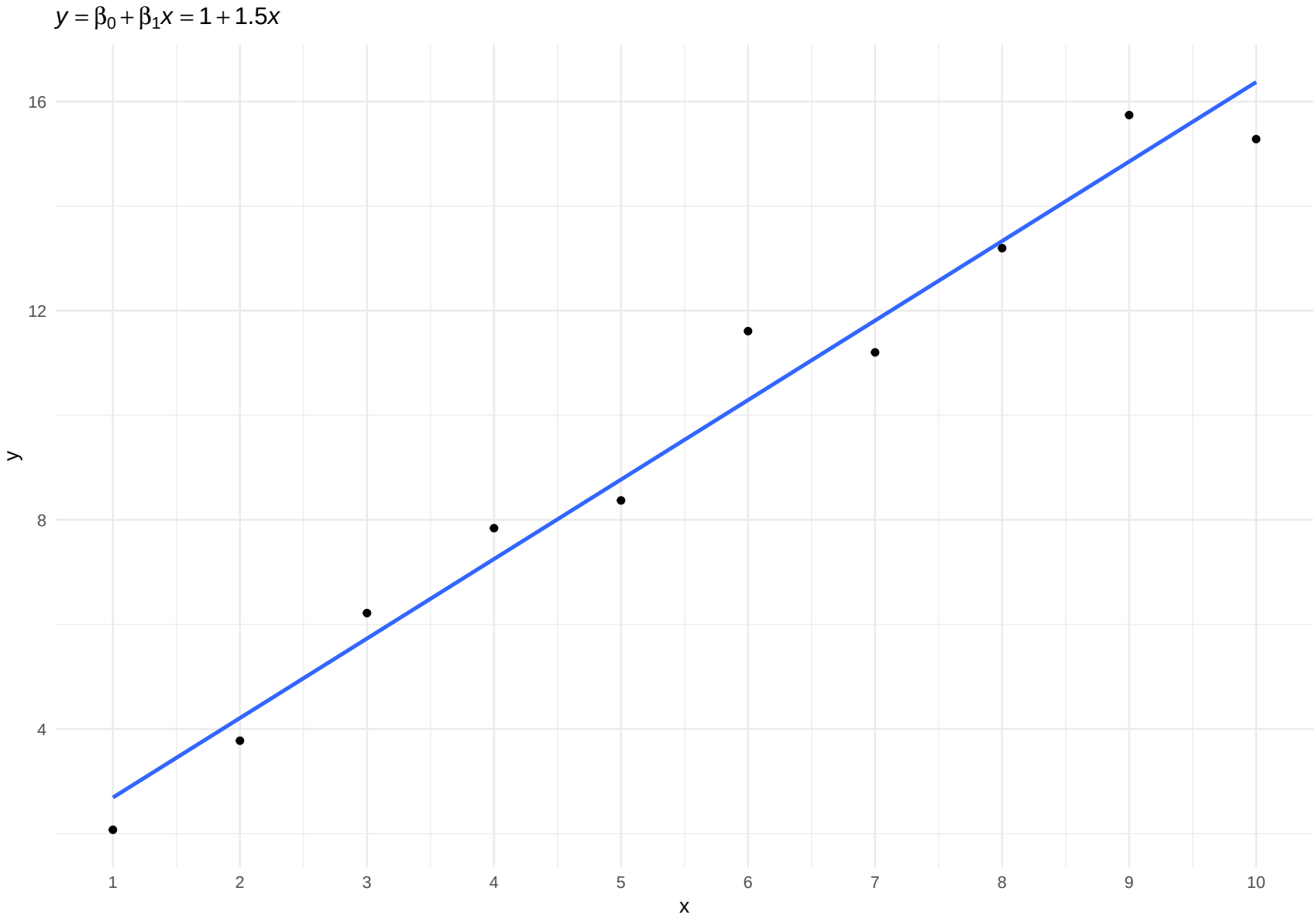
Linear models



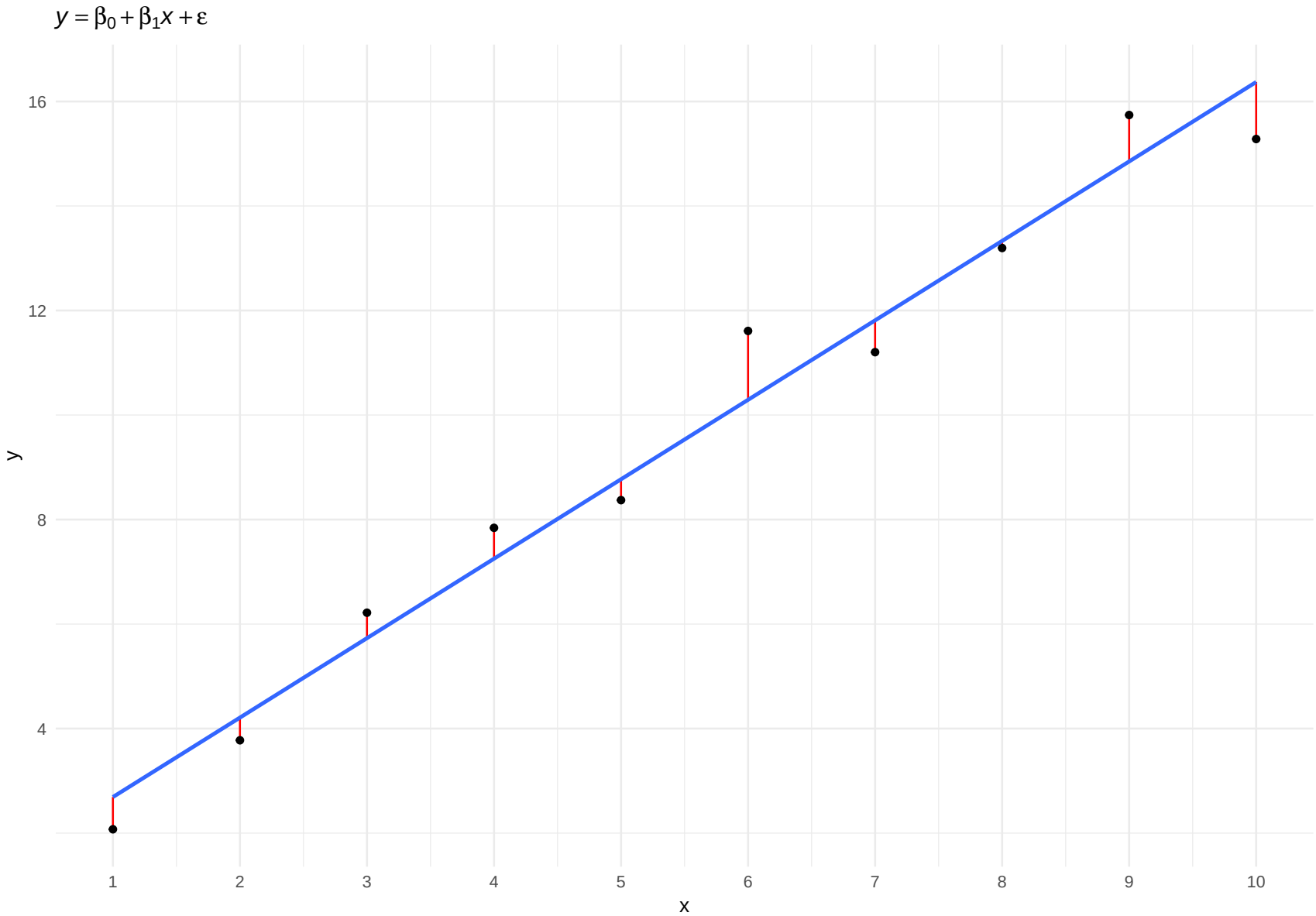
Linear models

- 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')

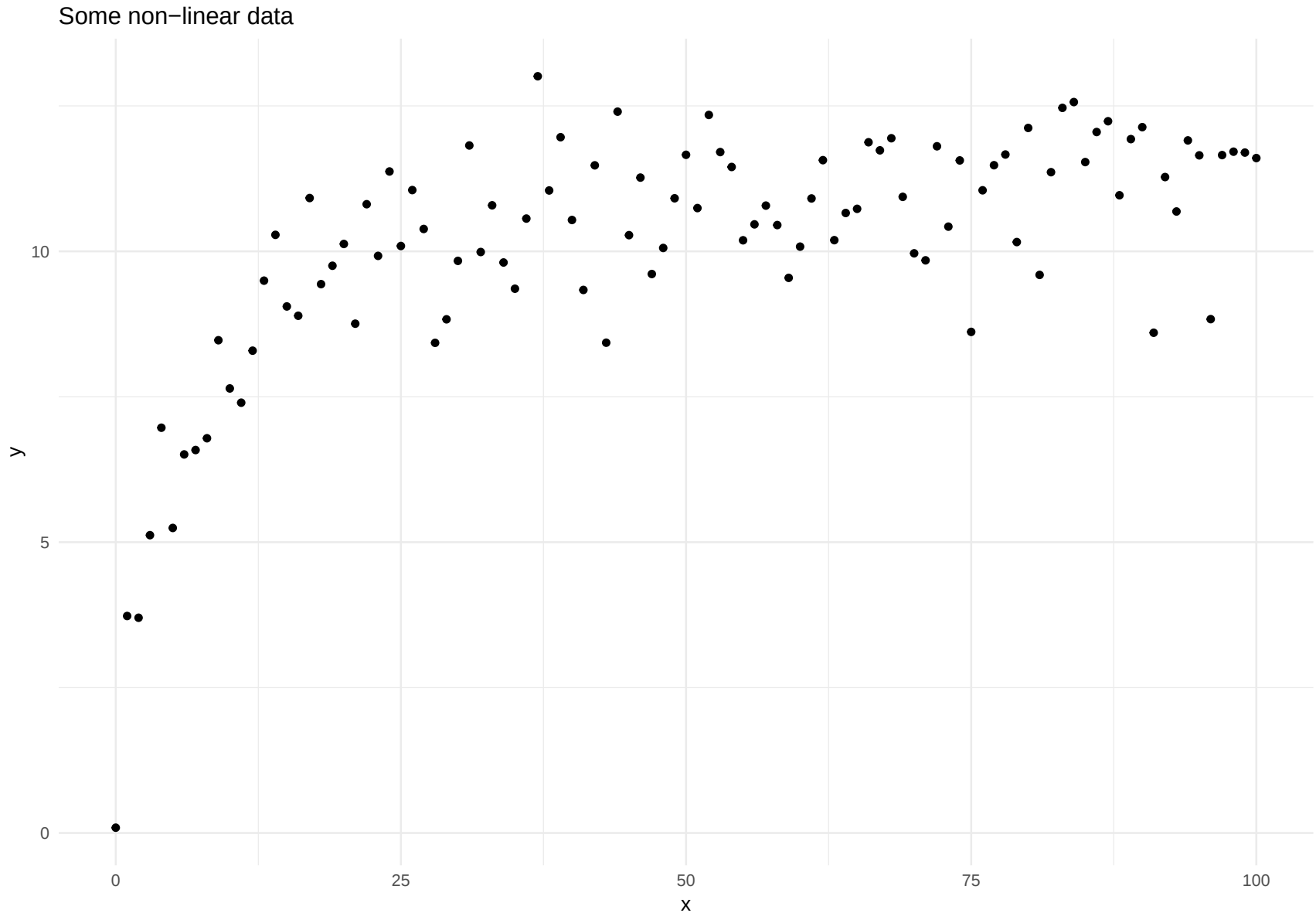
Linear models



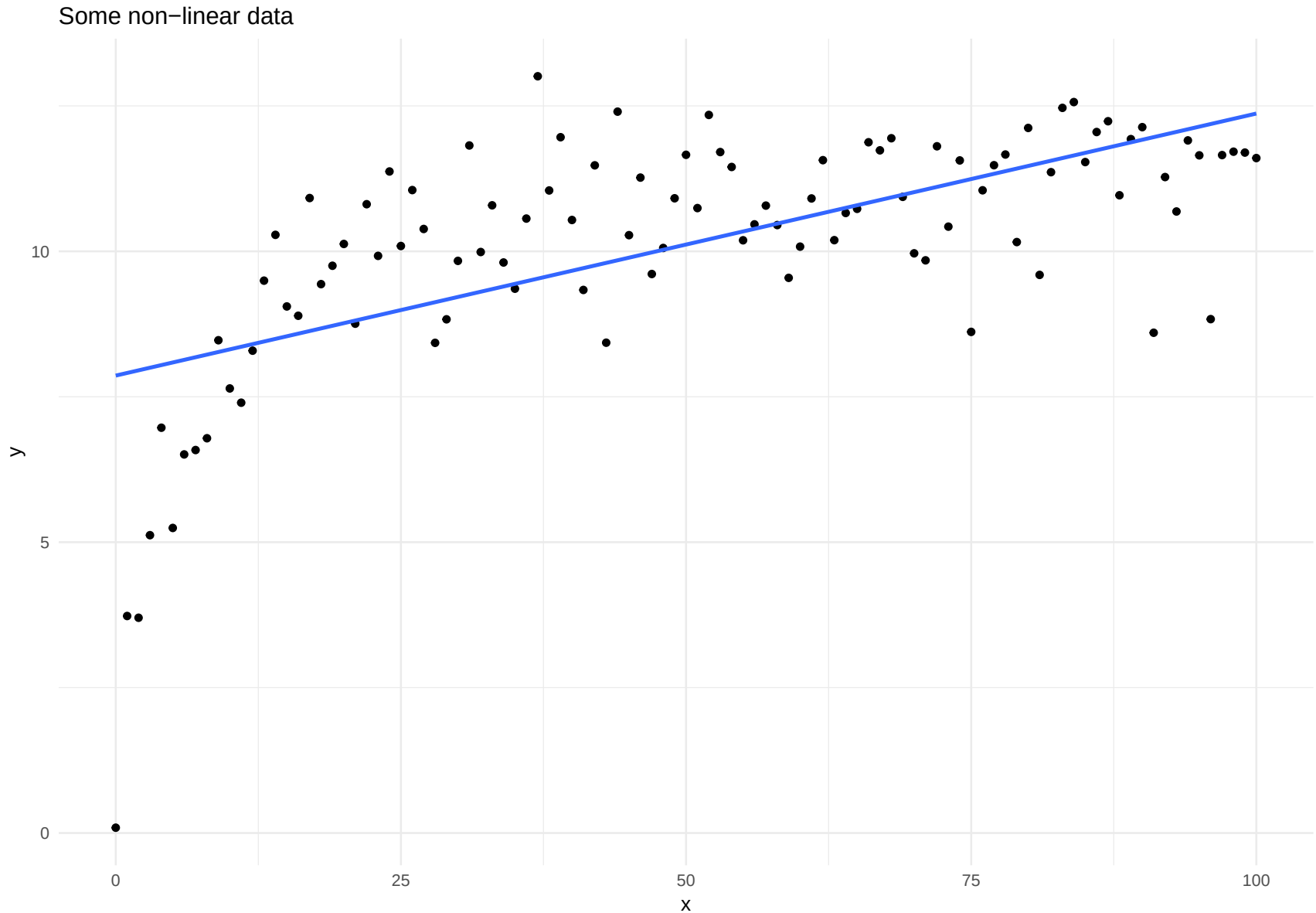
Linear models



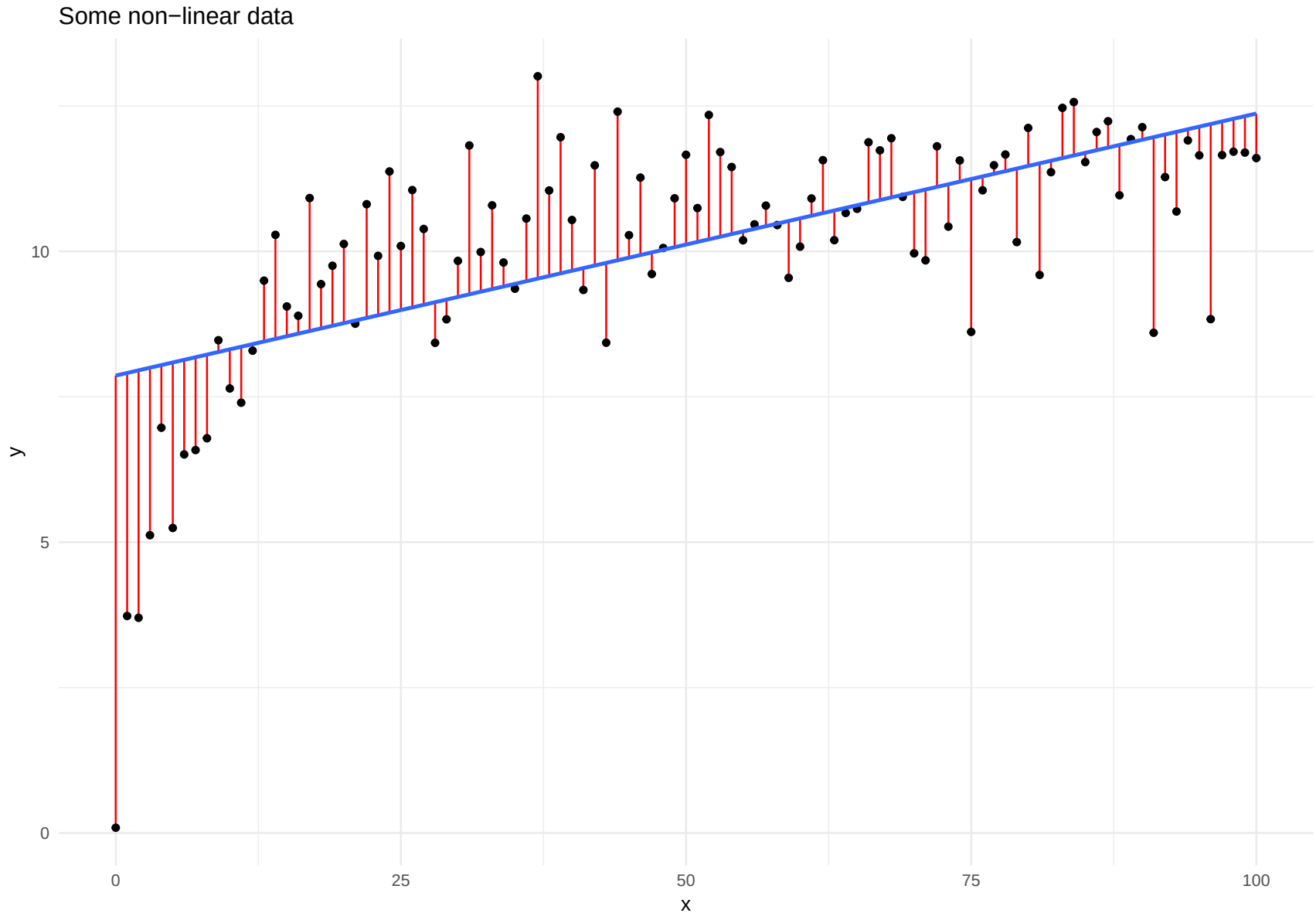
LM with non-linear data



LM with non-linear data



LM with non-linear data

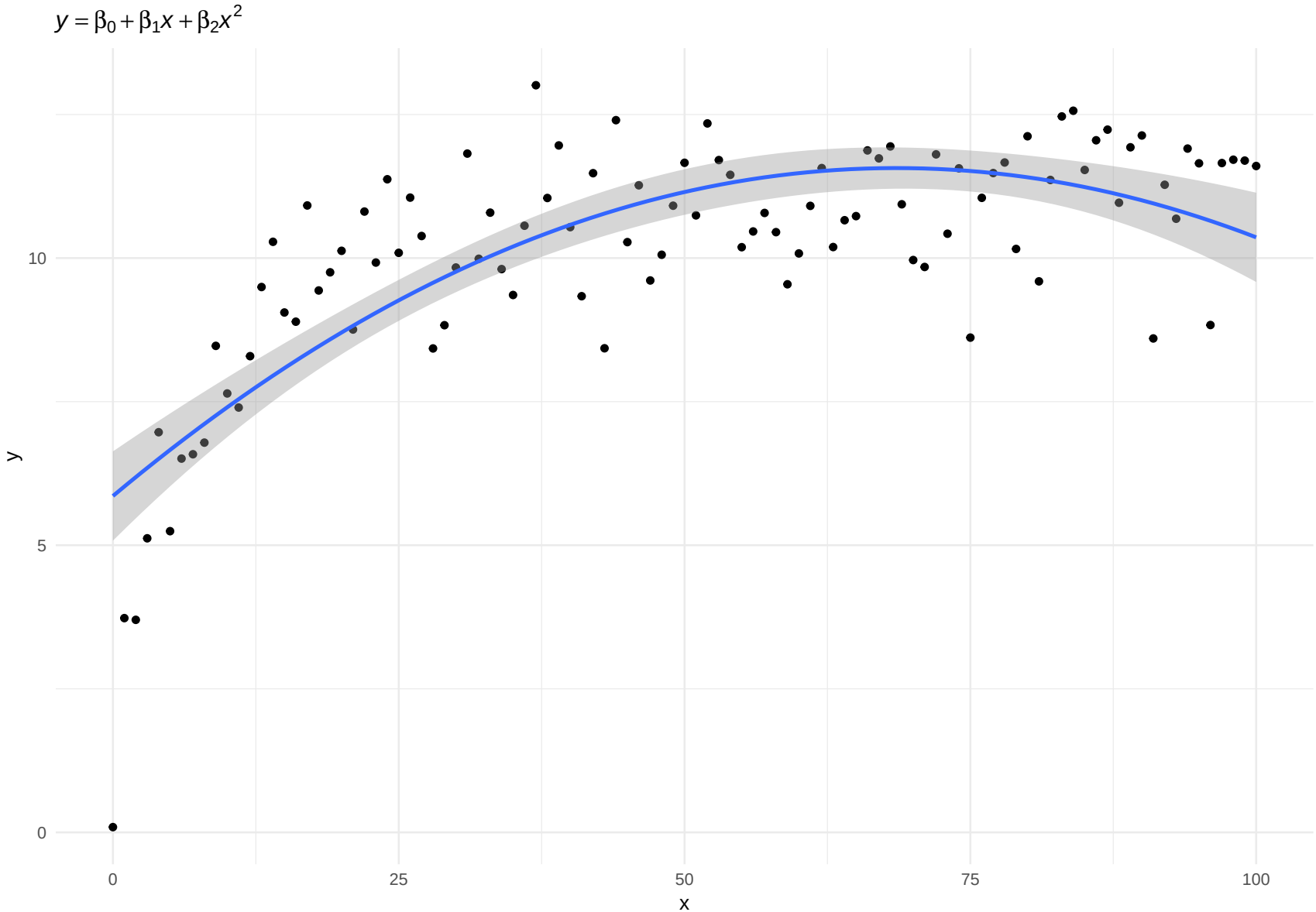


LM with non-linear data

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

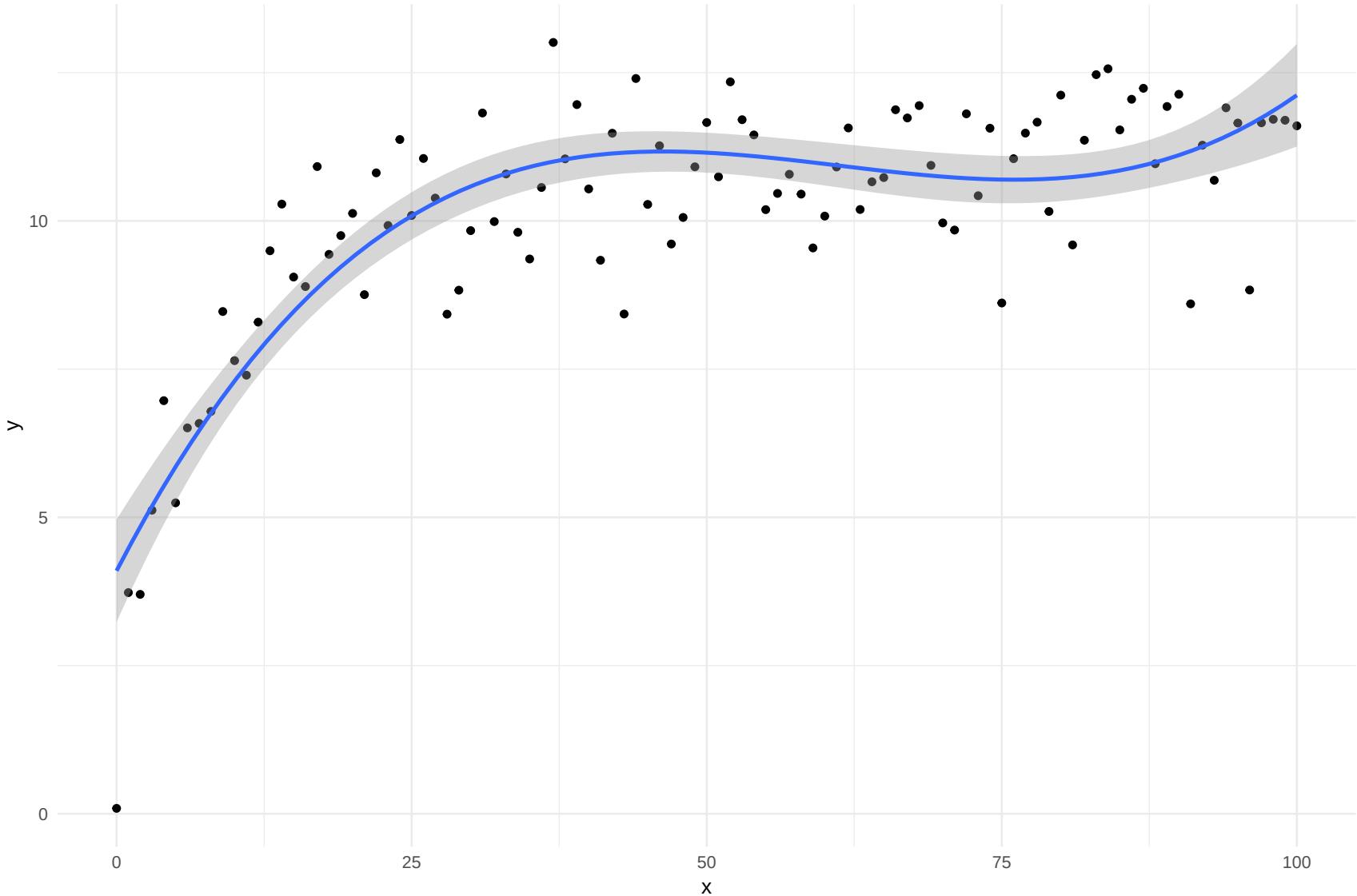
- Use **higher-degree polynomials**
  - quadratic: $y = \beta_0 + \beta_1 x + \beta_2 x^2$
  - cubic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$
  - nth: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + ... + \beta_n x^n$

LM with non-linear data



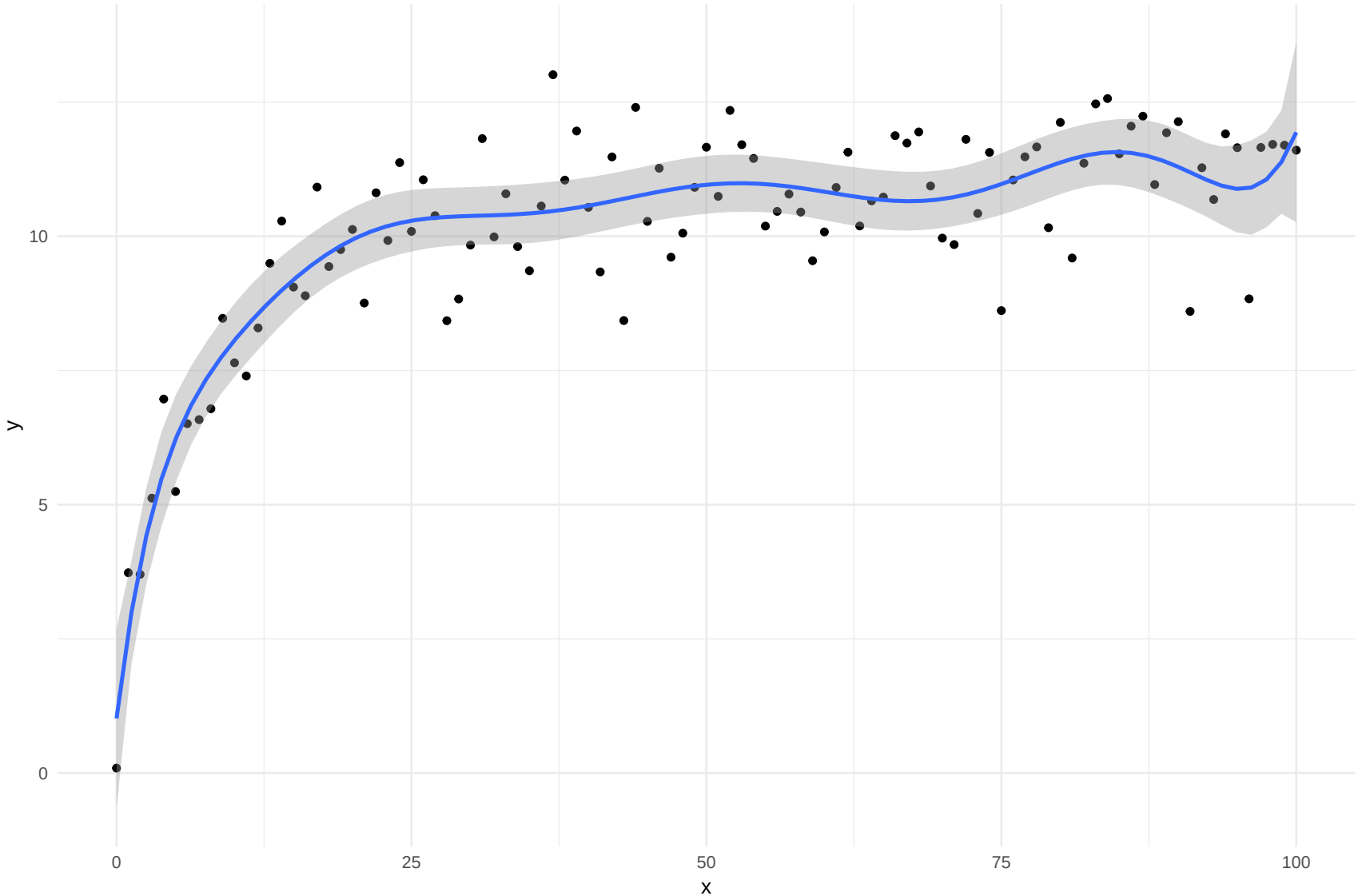
LM with non-linear data

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$



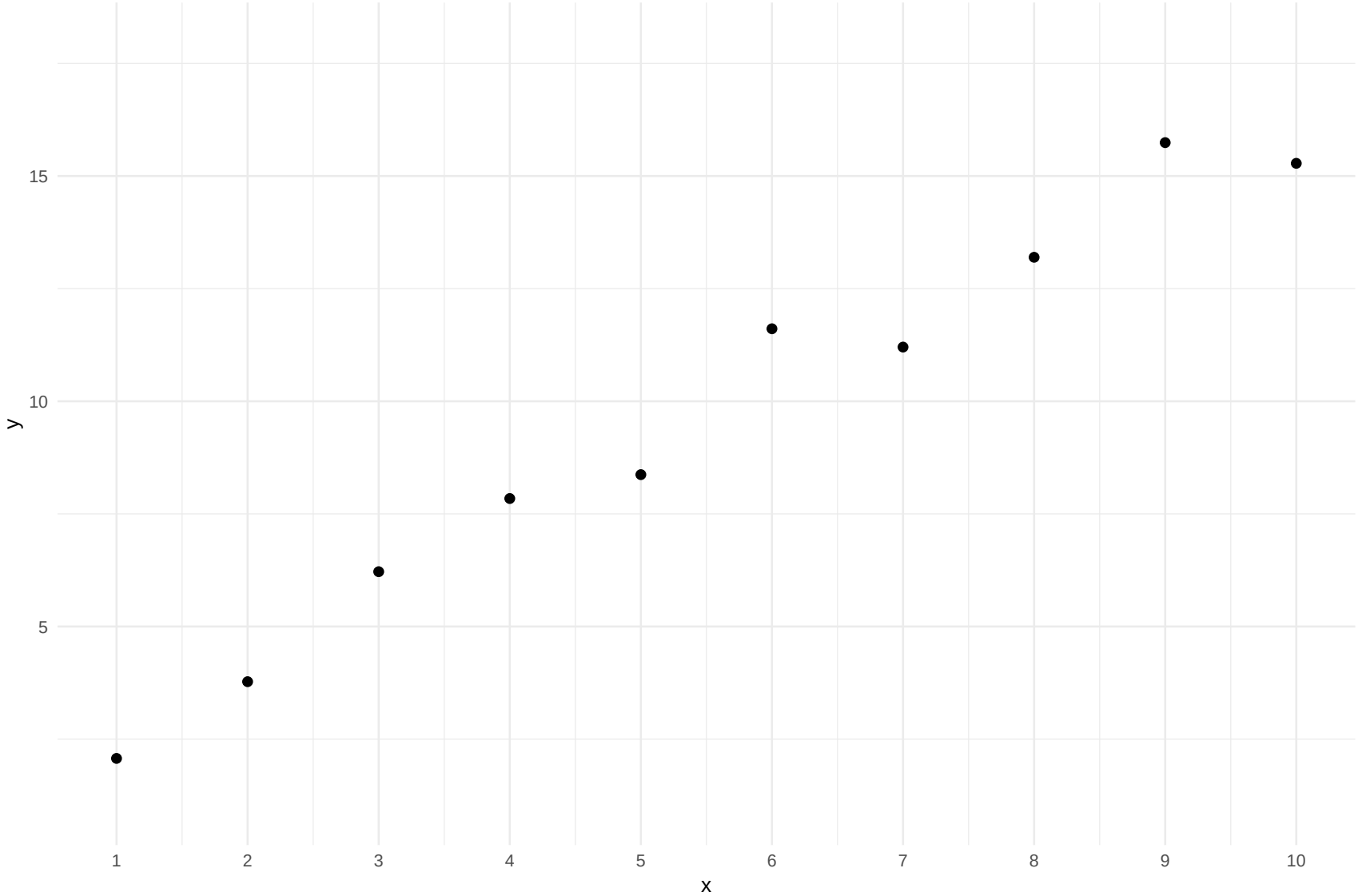
LM with non-linear data

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \beta_5 x^5 + \beta_6 x^6 + \beta_7 x^7 + \beta_8 x^8 + \beta_9 x^9 + \beta_{10} x^{10}$$



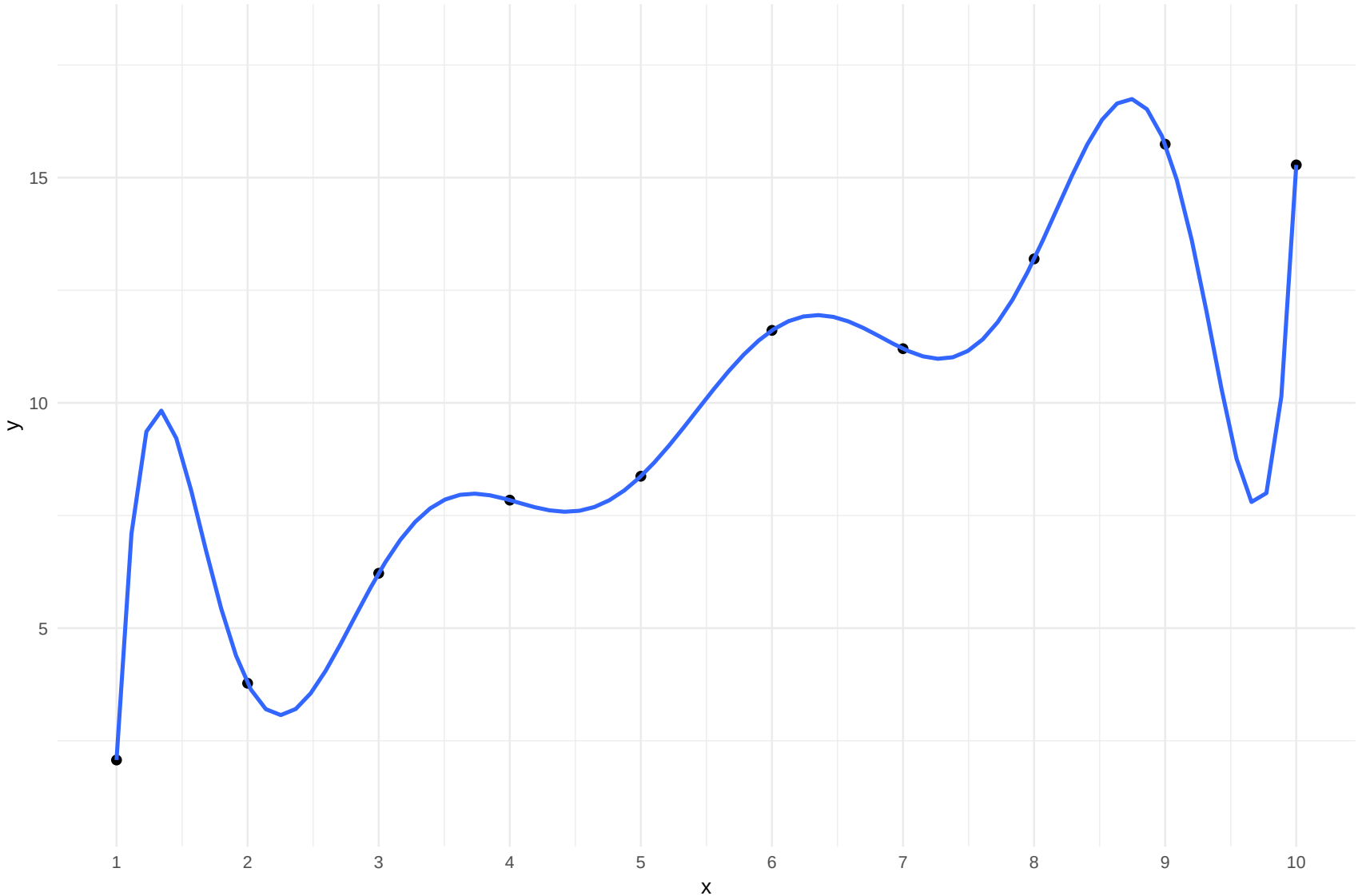
LM with non-linear data

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \beta_5 x^5 + \beta_6 x^6 + \beta_7 x^7 + \beta_8 x^8 + \beta_9 x^9 + \beta_{10} x^{10}$$



LM with non-linear data

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \beta_5 x^5 + \beta_6 x^6 + \beta_7 x^7 + \beta_8 x^8 + \beta_9 x^9 + \beta_{10} x^{10}$$



Generalised additive models

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

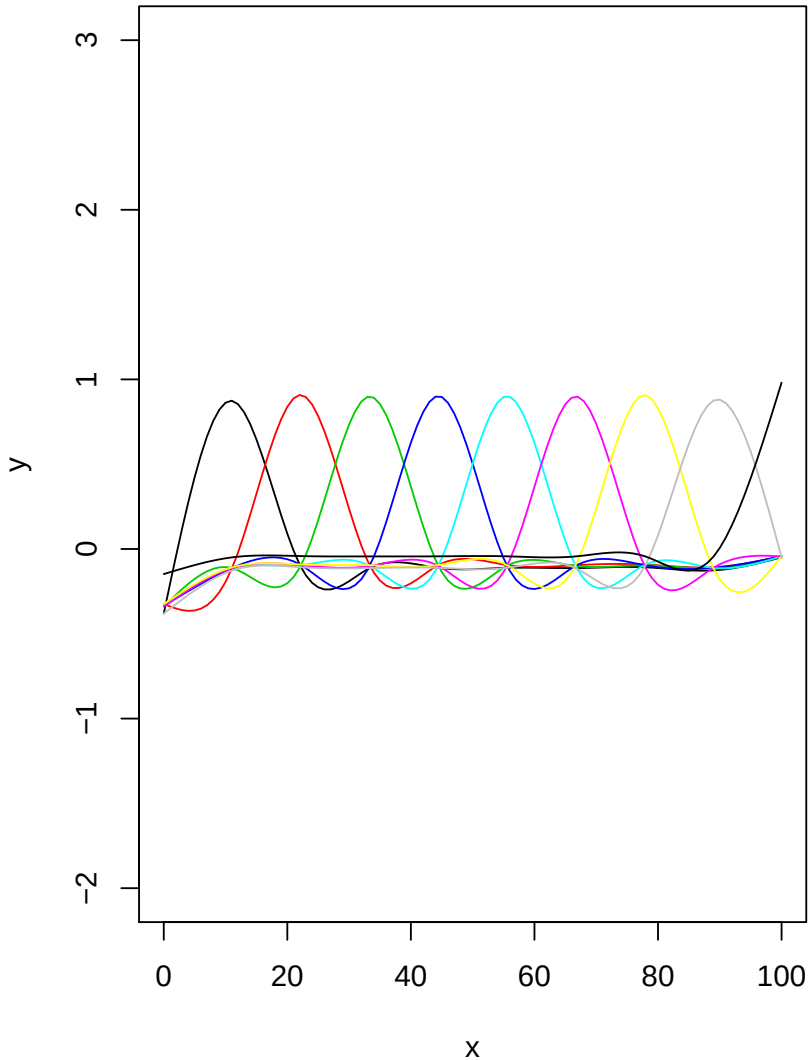
Smooth terms

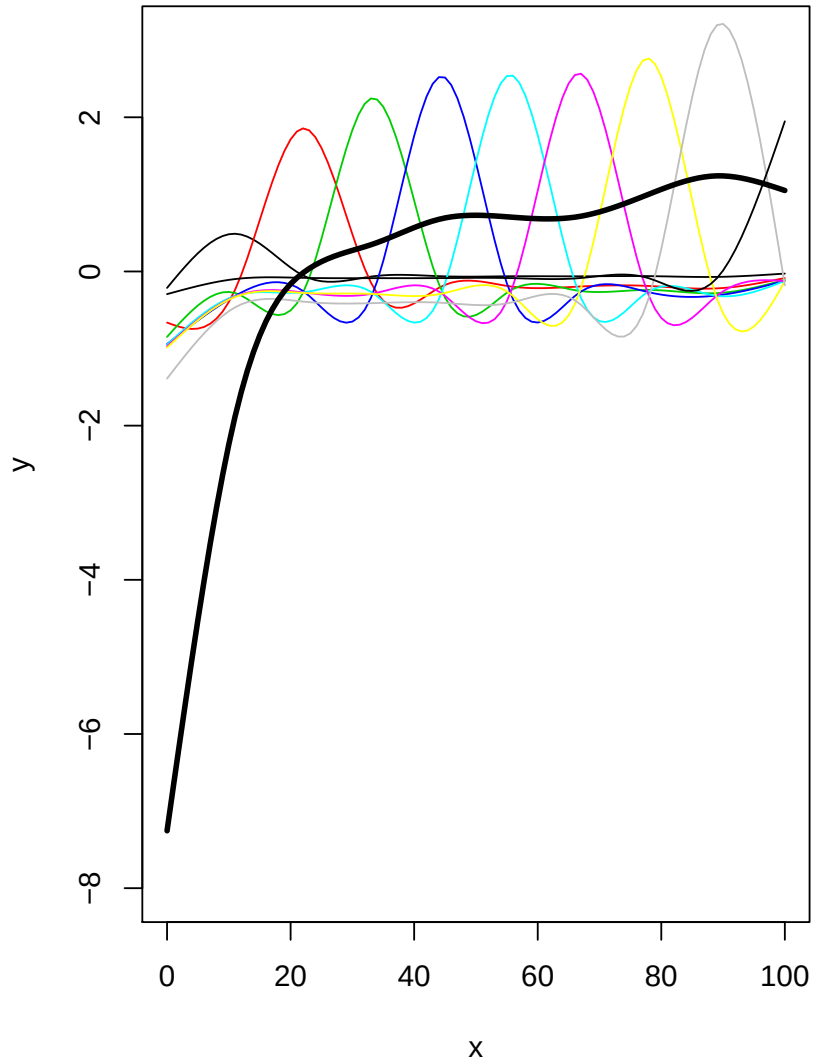
- LMs have **parametric terms**
  - $\beta_n x_n$
  - `x` in `R`
  - linear effects
- GAMs add (non-parametric) **smooth terms** (or simply smooths, also smoothers)
  - $f(x)$
  - `s(x)` in `R`
  - non-linear effects
- `gam(y ~ s(x), data)`, 'y as *some* function of x'

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





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

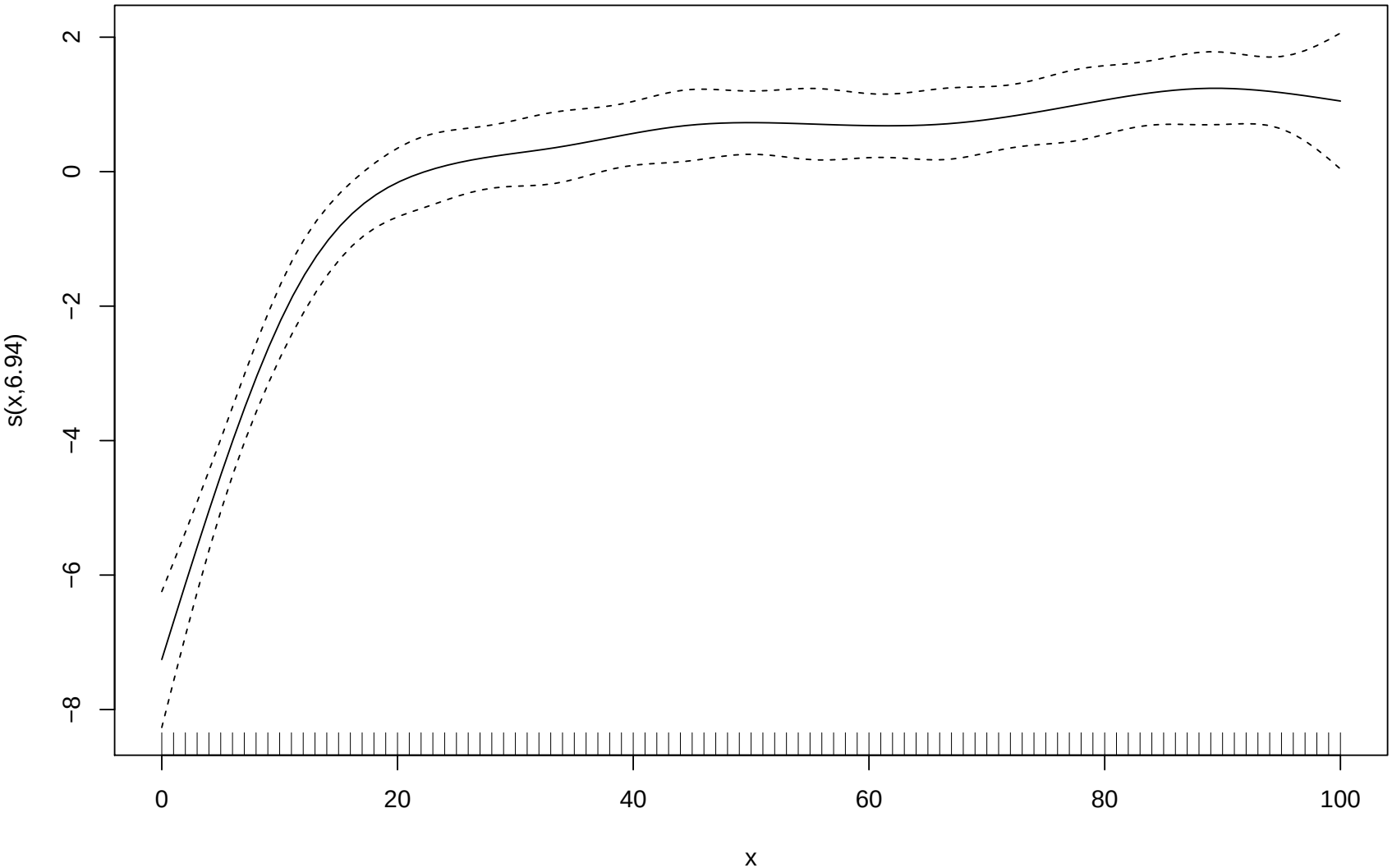
- There are **several kinds** of splines
  - each with their own basis functions
- Most common
  - *thin plate regression splines*
  - *cubic regression splines*
- For more info, run `?smooth.terms`

A simple GAM

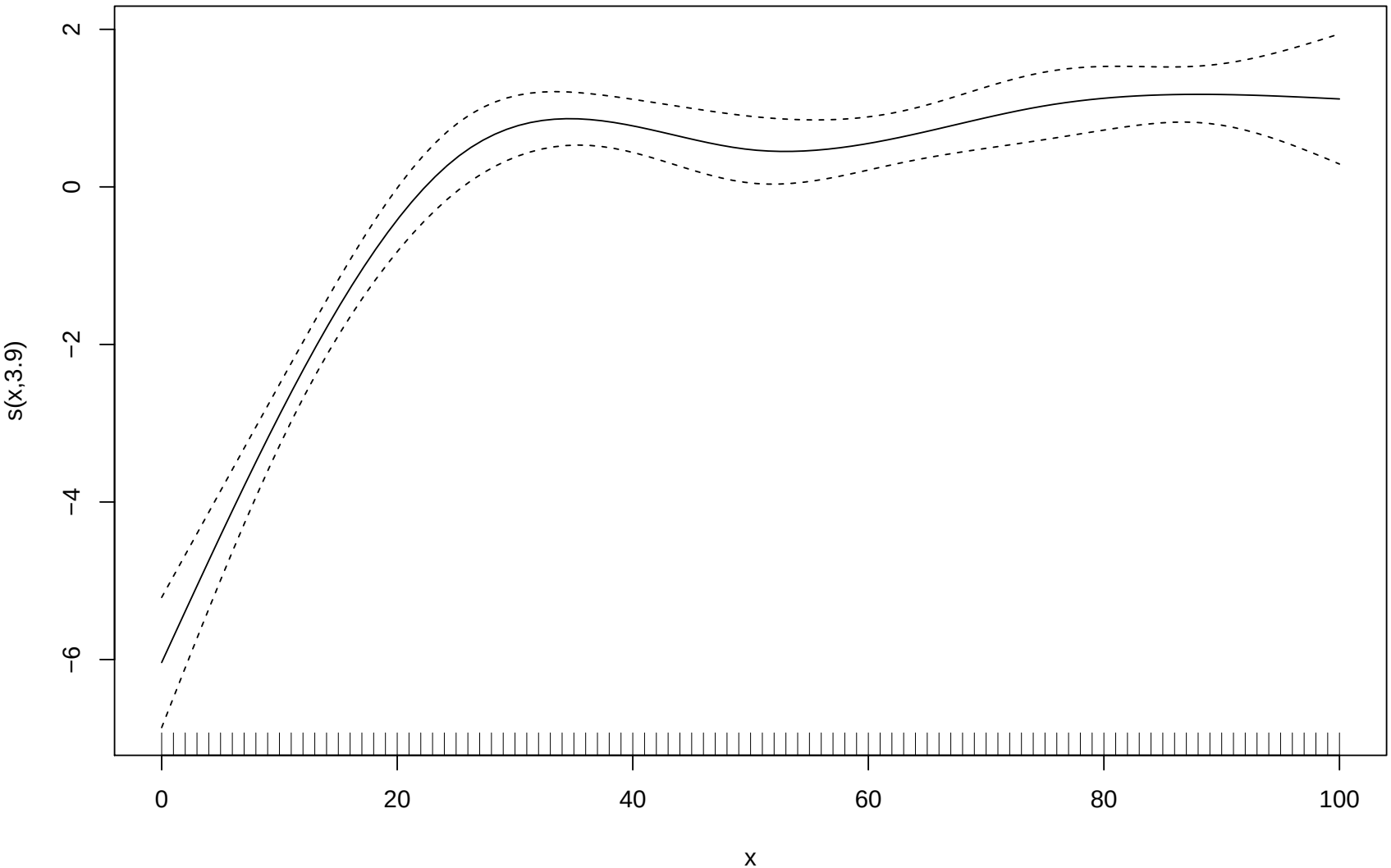
```r
simple <- gam(y ~ s(x, bs = "cr", k = 10), data = sim_nl_a)
summary(simple)
```

```
##
## 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
```

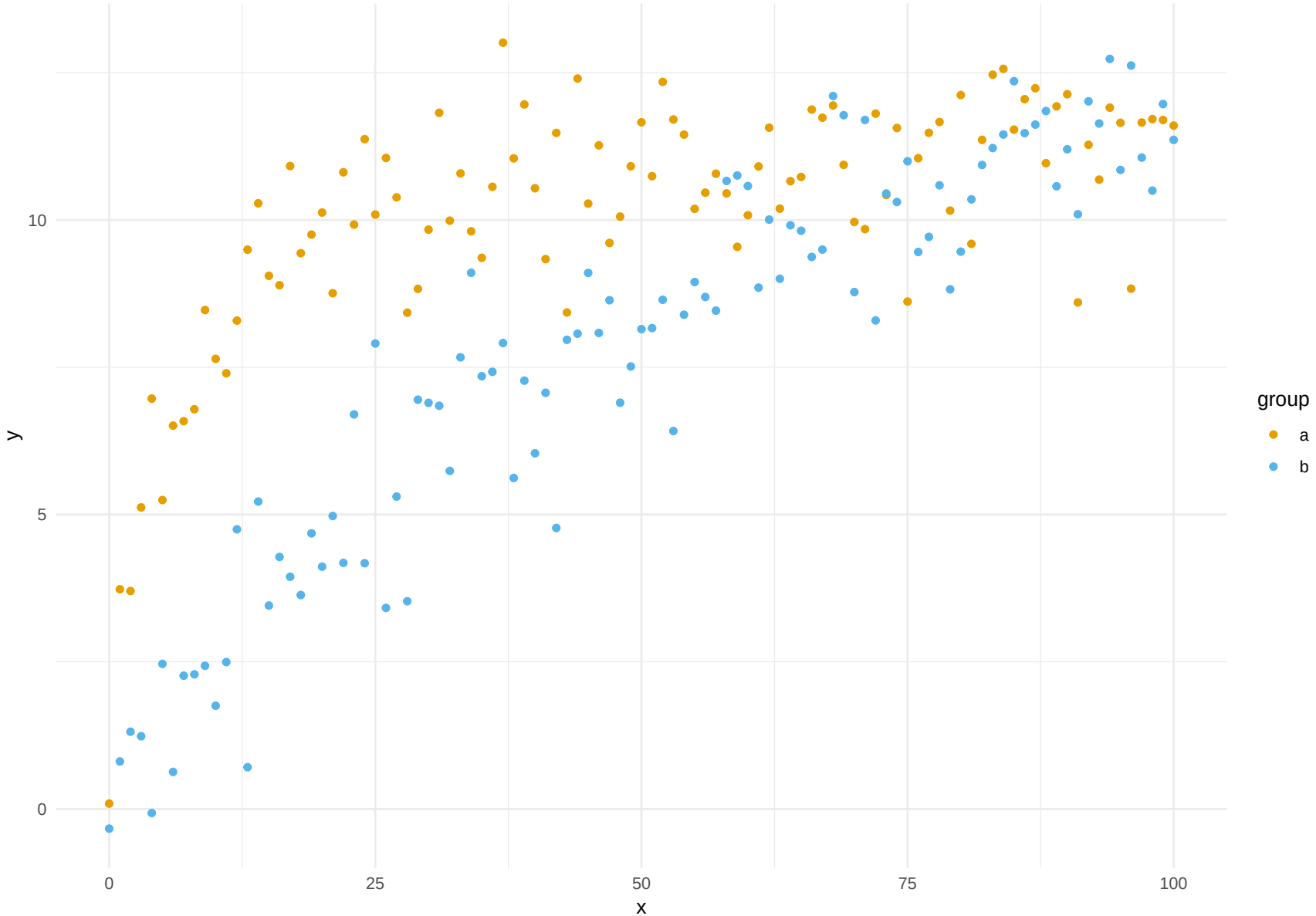
A simple GAM



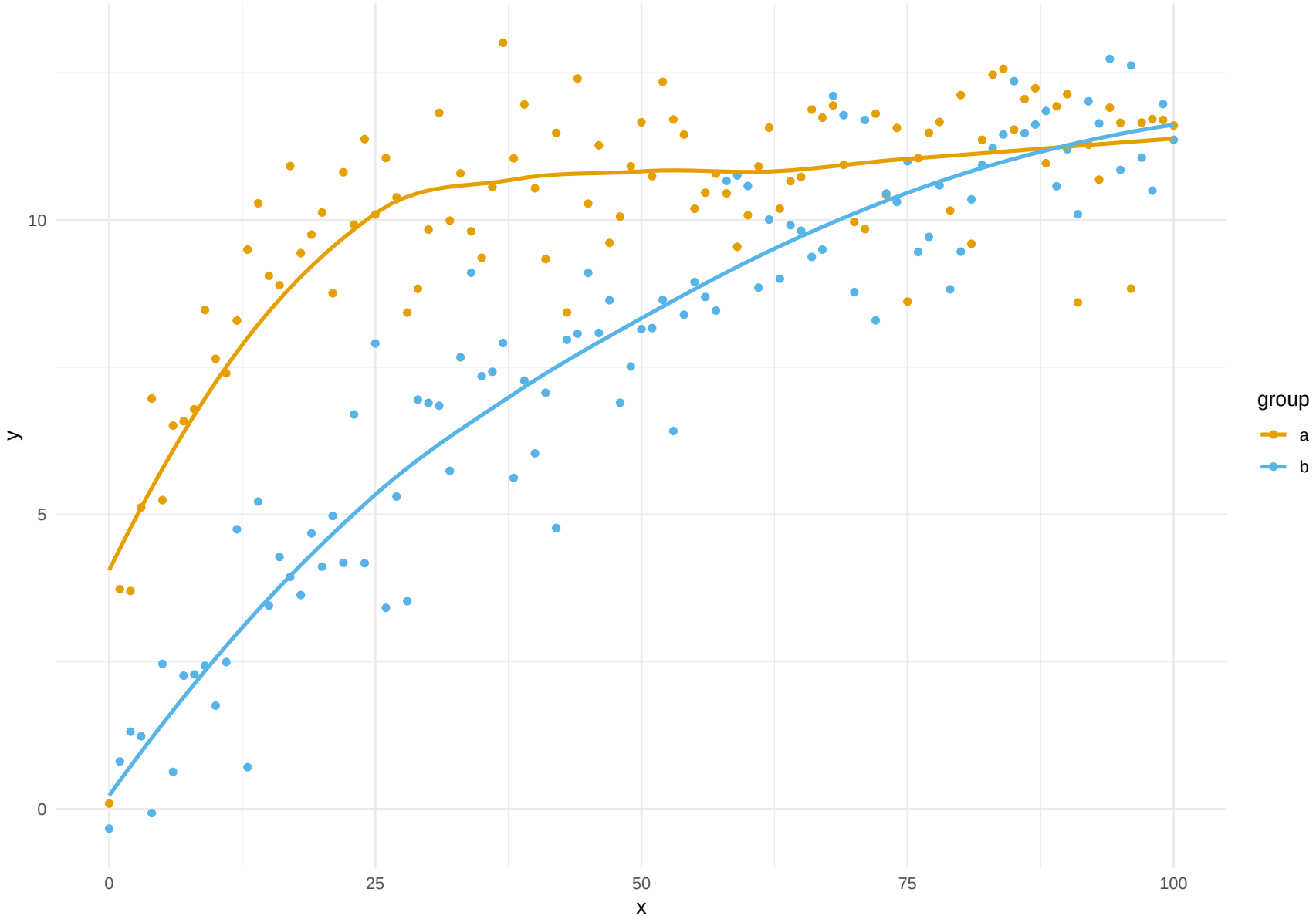
A simple GAM



Comparing levels



Comparing levels



Comparing levels

- by-variables with ordered factors

```
compare <- gam(
  y ~
    group +
    s(x, bs = "cr", k = 5) +
    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

```
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 ~
    group +
    s(x, bs = "cr", k = 5) +
    s(x, bs = "cr", k = 5, by = group),
  data = sim_nl
)
```

Comparing levels

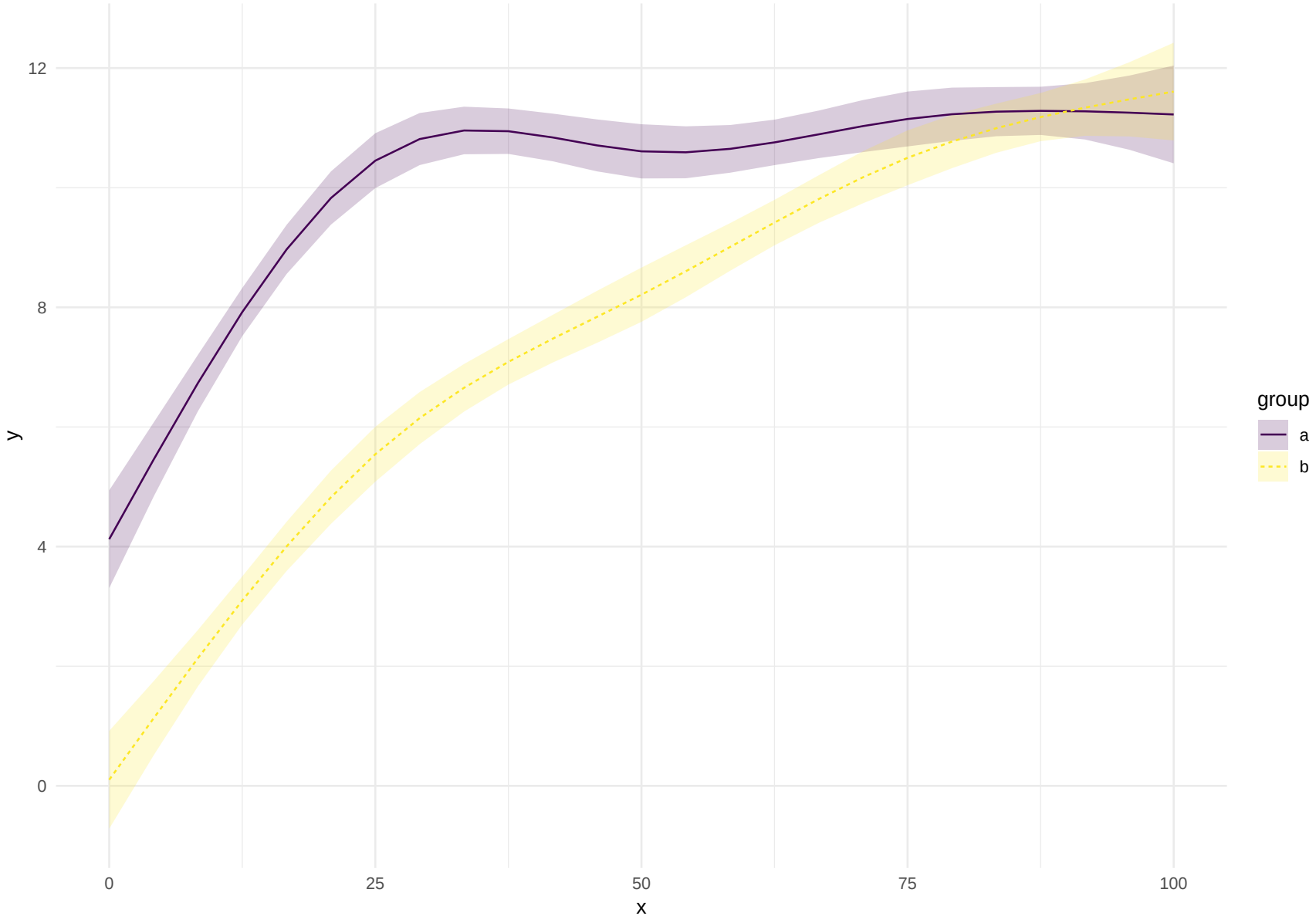
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ group + s(x, bs = "cr", k = 5) + s(x, bs = "cr", k = 5, by = group)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.1165     0.1096   92.34   <2e-16 ***
## groupb       -2.4947     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

```
library(tidymv)


plot_smooths(compare, x, 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()`)

[say that you need to use ML]

Significance testing

```
compare_1 <- gam(
  y ~
    group +
    s(x, bs = "cr", k = 5) +
    s(x, bs = "cr", k = 5, by = group),
  data = sim_nl,
  method = "ML"
)

compare_0 <- gam(
  y ~
    s(x, bs = "cr", k = 5),
  data = sim_nl,
  method = "ML"
)
```

Significance testing

```
compareML(compare_0, compare_1)
```

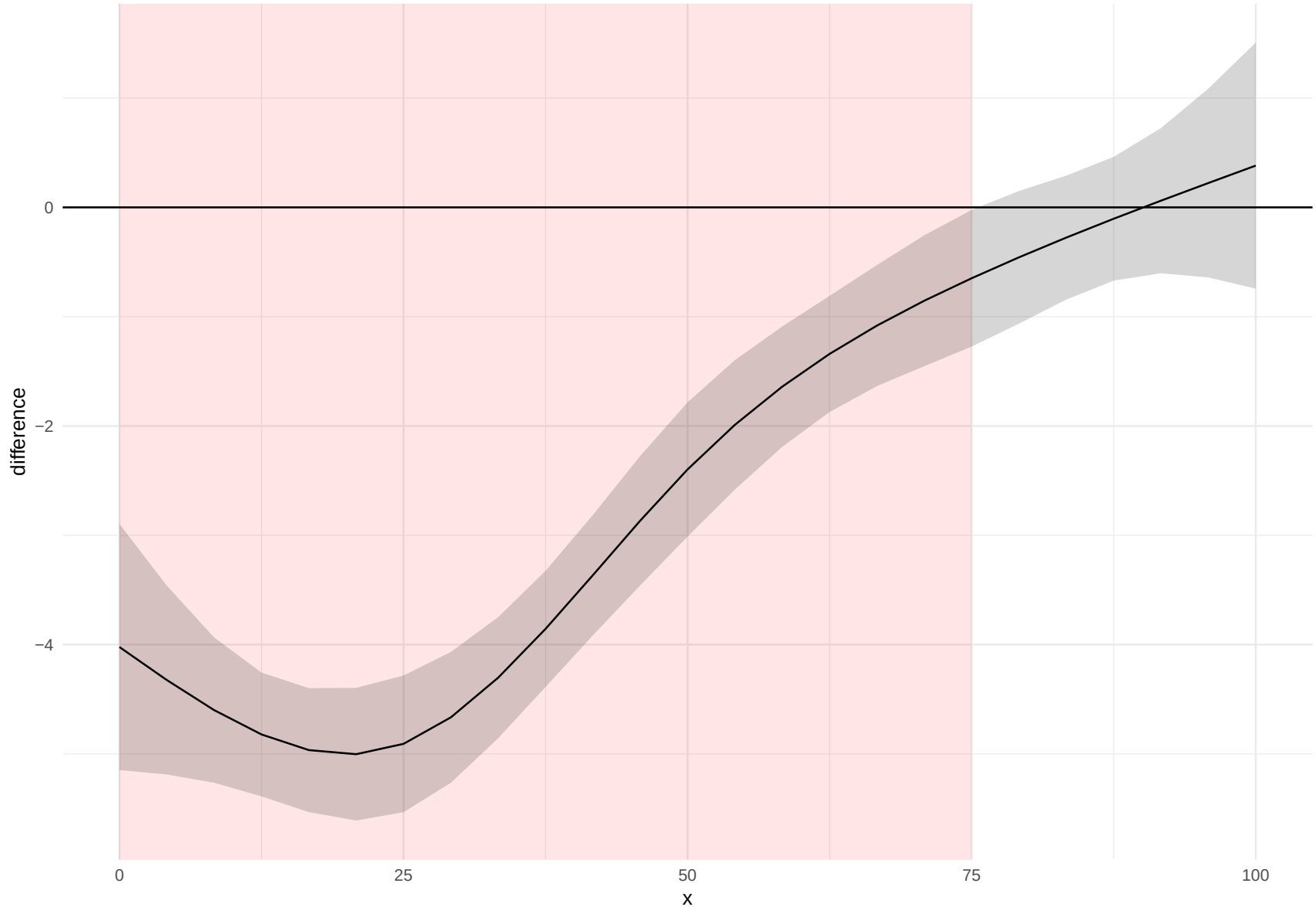
```
## compare_0: y ~ s(x, bs = "cr", k = 5)
##
## compare_1: y ~ group + s(x, bs = "cr", k = 5) + s(x, bs = "cr", k = 5, by =
##
## Chi-square test of ML scores
## -----
##       Model    Score Edf Difference    Df  p.value Sig.
## 1 compare_0 422.4827   3
## 2 compare_1 314.0105   6    108.472 3.000  < 2e-16  ***
##
## AIC difference: 221.22, model compare_1 has lower AIC.
```

Significance testing

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

```
plot_difference(compare, x, 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**" [@soskuthy2017]
- Examples
  - formant trajectories
  - pitch contours
  - geographic (*diatopic*) variation
  - tongue contours

Dynamic data

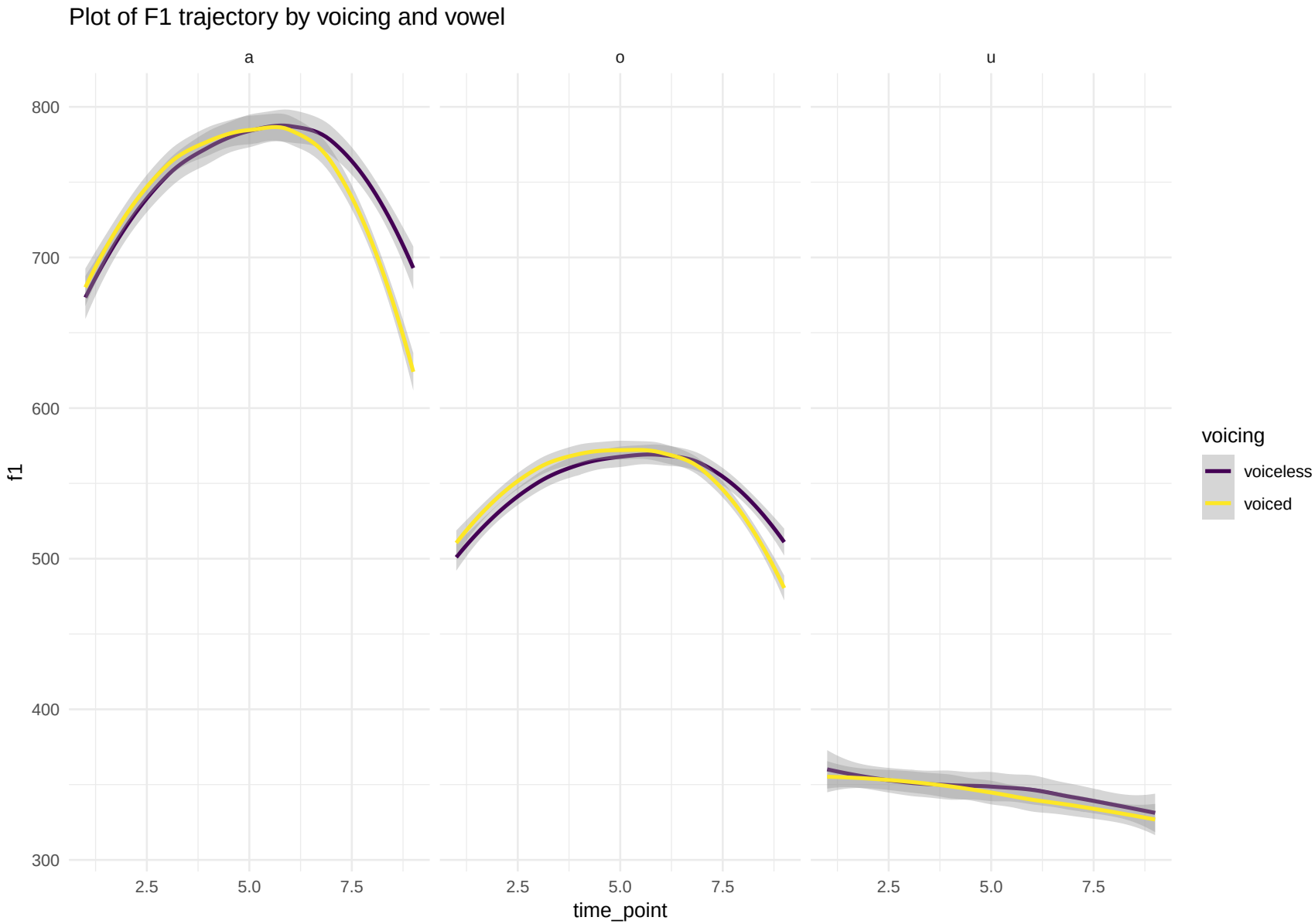
- Two main types
  - **time series data**
  - **spatial data**
- More data ($n > 1000$)
  - use `bam()` (big GAM) instead of `gam()`

Dynamic data

- formant trajectories (time series data)

```
## # A tibble: 6,165 x 9
##    speaker index time_point    f1 duration vowel voicing place   vow_voi
##    <fct>   <int>      <int> <dbl>    <dbl> <ord> <ord>   <ord>   <ord>
##  1 it01        1          1  308.     95.2 u     voiced  velar   u.voiced
##  2 it01        1          2  315.     95.2 u     voiced  velar   u.voiced
##  3 it01        1          3  316.     95.2 u     voiced  velar   u.voiced
##  4 it01        1          4  314.     95.2 u     voiced  velar   u.voiced
##  5 it01        1          5  313.     95.2 u     voiced  velar   u.voiced
##  6 it01        1          6  305.     95.2 u     voiced  velar   u.voiced
##  7 it01        1          7  291.     95.2 u     voiced  velar   u.voiced
##  8 it01        1          8  280.     95.2 u     voiced  velar   u.voiced
##  9 it01        1          9  287.     95.2 u     voiced  velar   u.voiced
## 10 it01        2          1  651.    139.  a     voiced  coronal a.voiced
## # ... with 6,155 more rows
```

Dynamic data



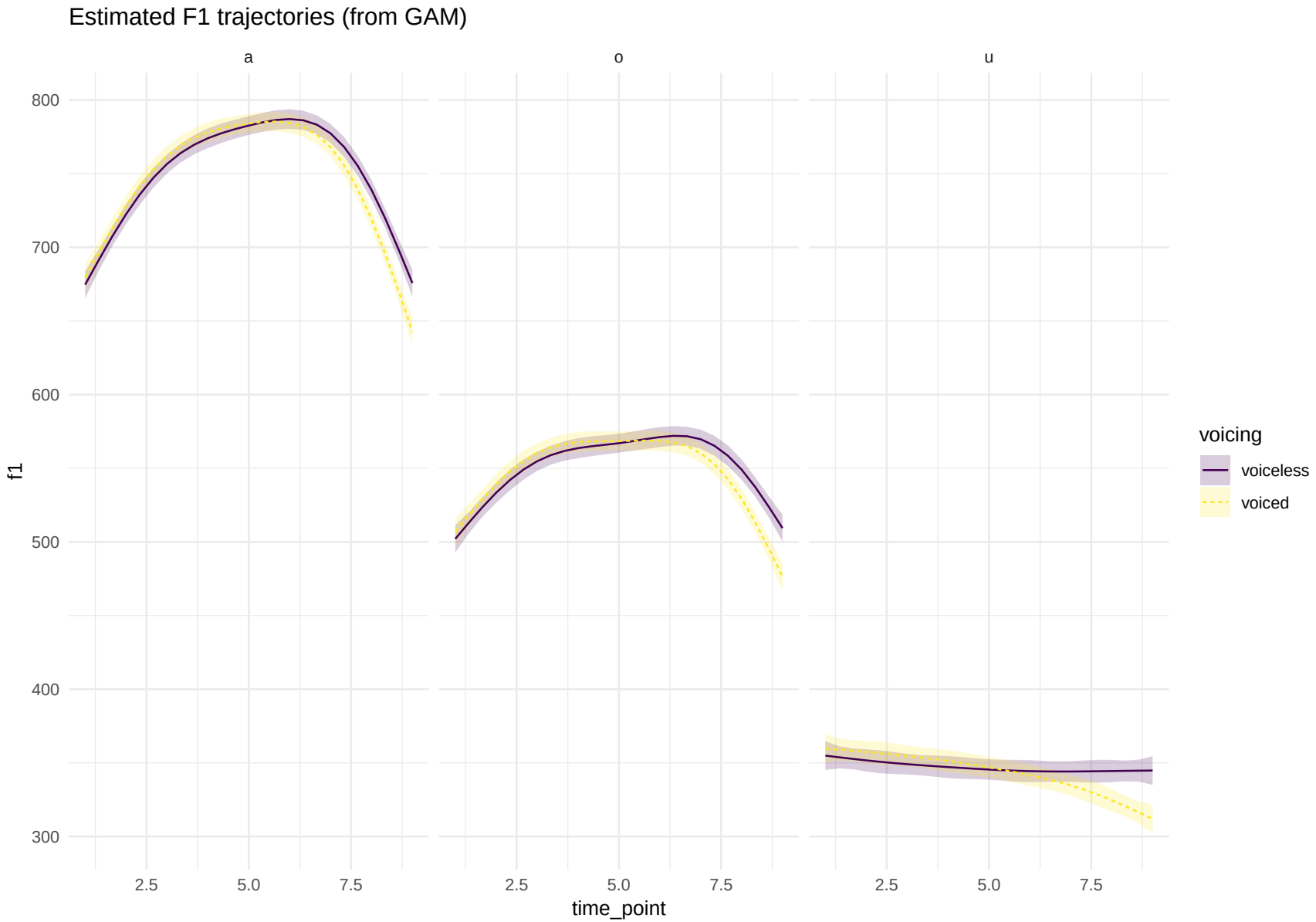
Dynamic data

```
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
)
```

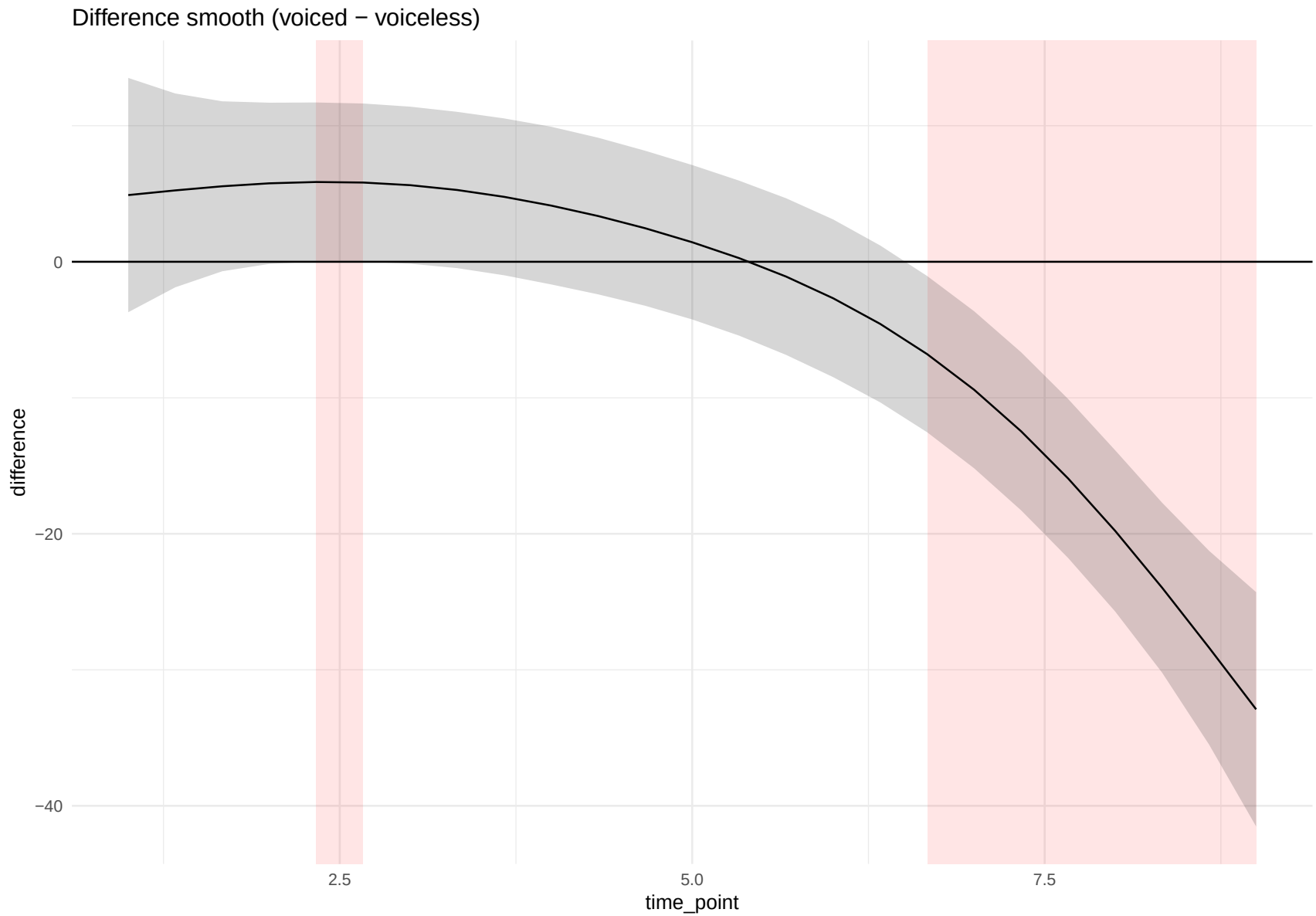
Dynamic data

```
##
## 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)    743.273      1.693  439.010   <2e-16 ***
## voicingvoiced   -4.768      1.713   -2.783   0.0054 **
## vowelo        -196.604      2.066  -95.157   <2e-16 ***
## vowelu        -395.951      2.118 -186.909   <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)                4.810  4.942 129.07 < 2e-16 ***
## s(time_point):voicingvoiced 2.807  3.407  16.70 1.2e-11 ***
## s(time_point):vowelo        3.652  4.255  17.01 2.6e-14 ***
## s(time_point):vowelu        4.621  4.907  87.63 < 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")`

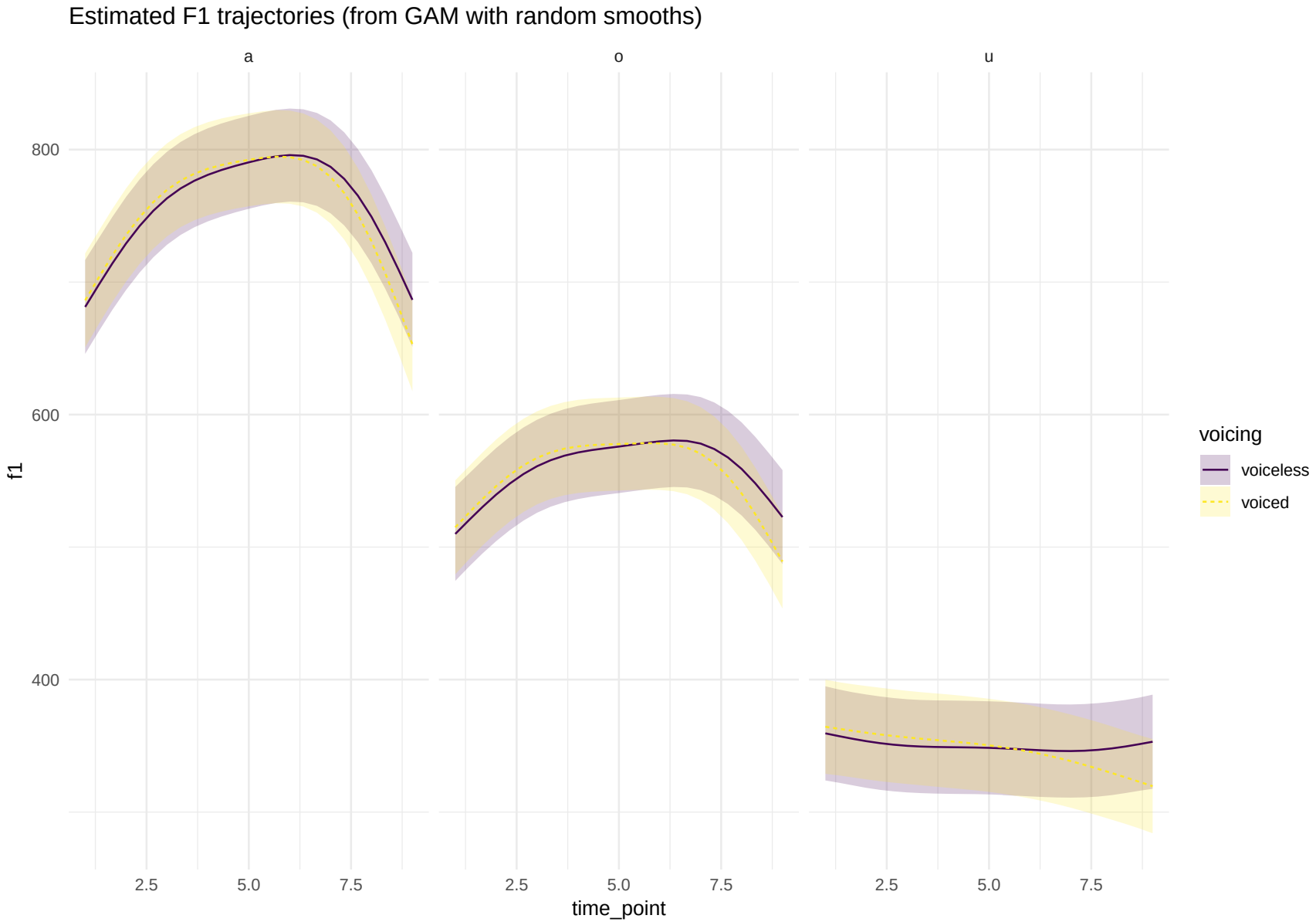
Random effects

```
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
)
```

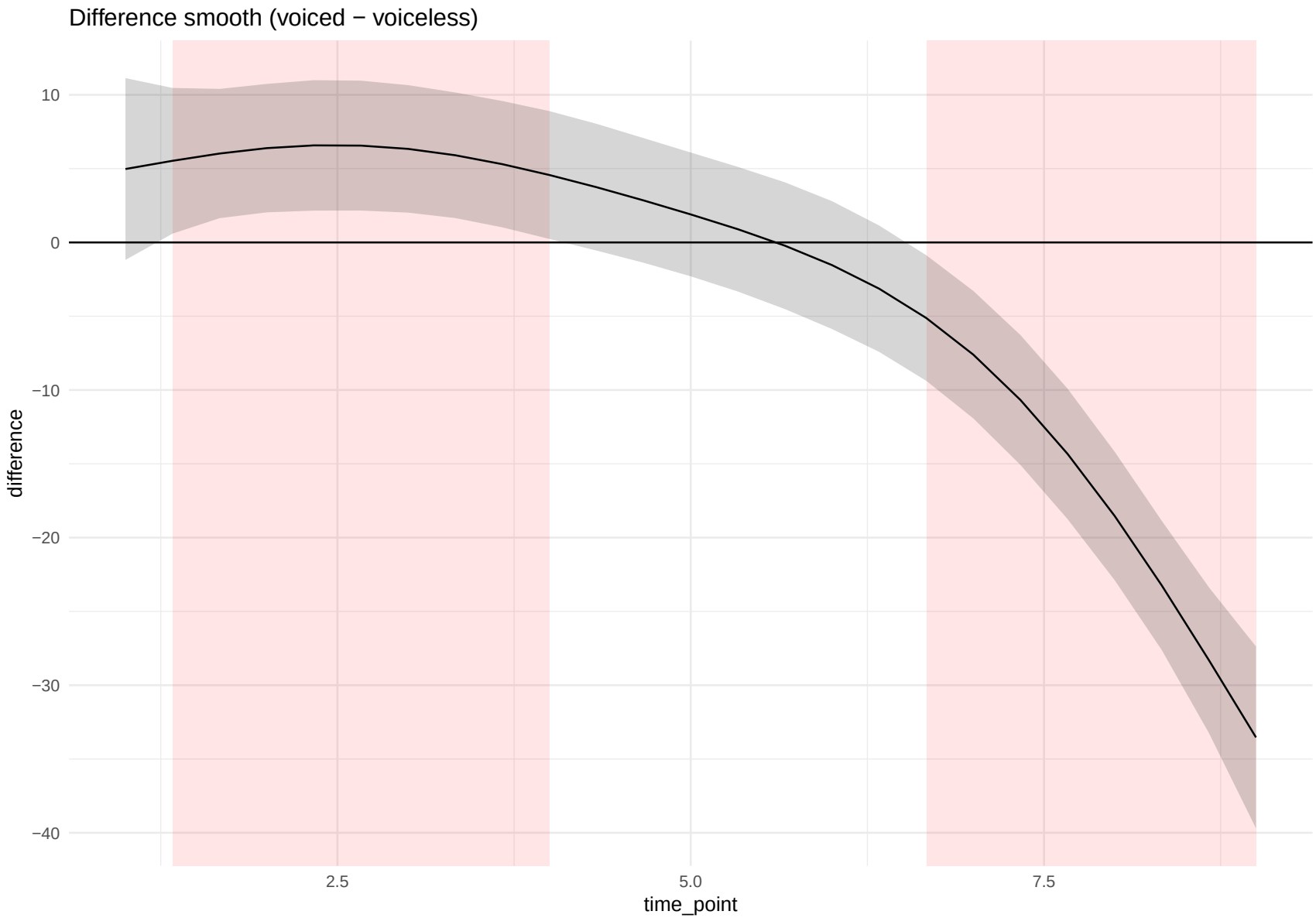
Random effects

```
##
## 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)    751.515     17.533   42.863  < 2e-16 ***
## voicingvoiced   -4.117      1.164   -3.537 0.000407 ***
## vowelo        -196.254      1.403 -139.835  < 2e-16 ***
## vowelu        -401.003      1.457 -275.209  < 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)                4.833  4.914 128.74  <2e-16 ***
## s(time_point):voicingvoiced  3.417  4.053  31.06  <2e-16 ***
## s(time_point):vowelo         4.155  4.666  35.14  <2e-16 ***
## s(time_point):vowelu         4.814  4.970 193.99  <2e-16 ***
## s(time_point,speaker)       45.710 79.000  90.77  <2e-16 ***
## ---
```

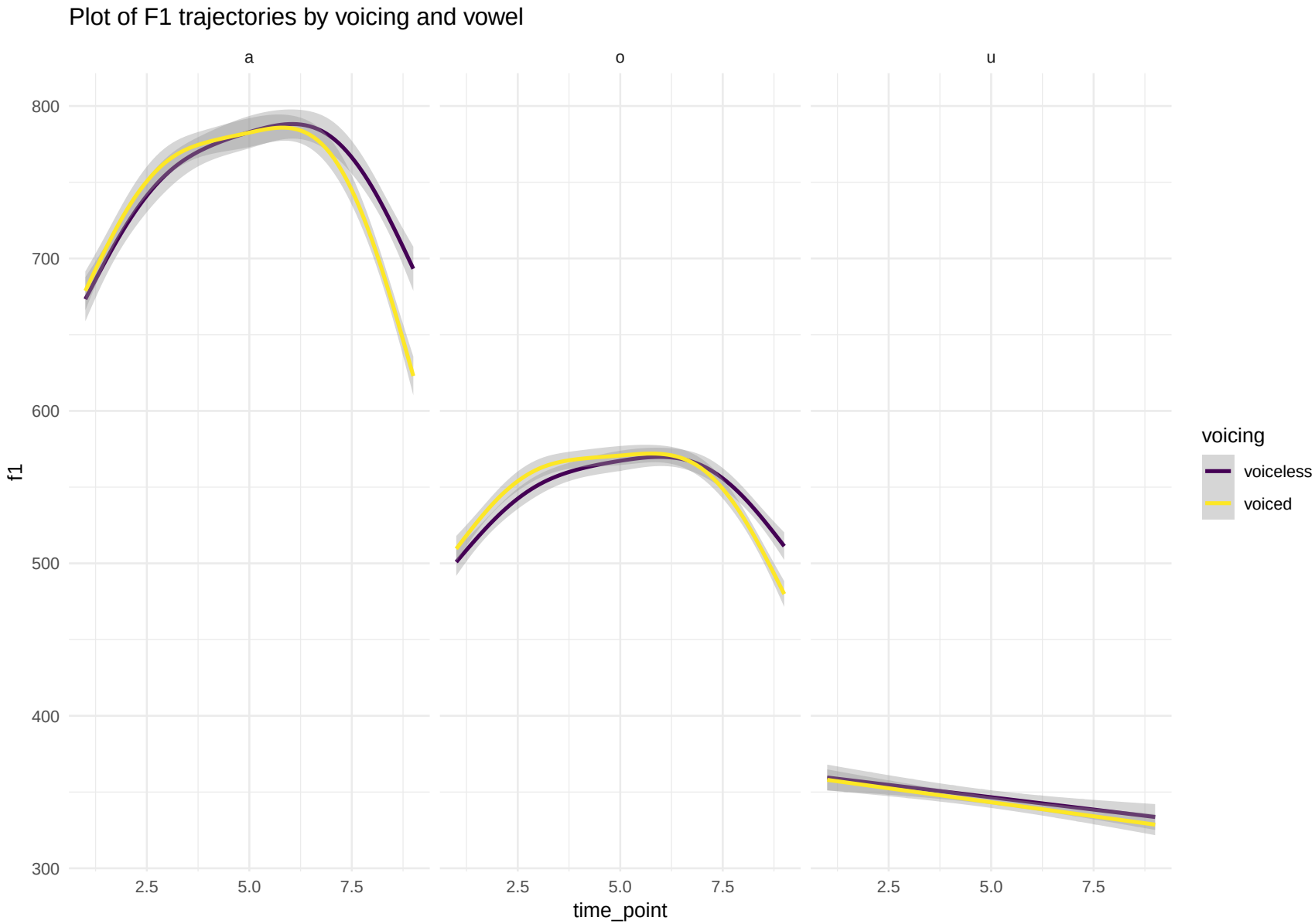
Random effects



Random effects



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

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

Interactions

```
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
)
```

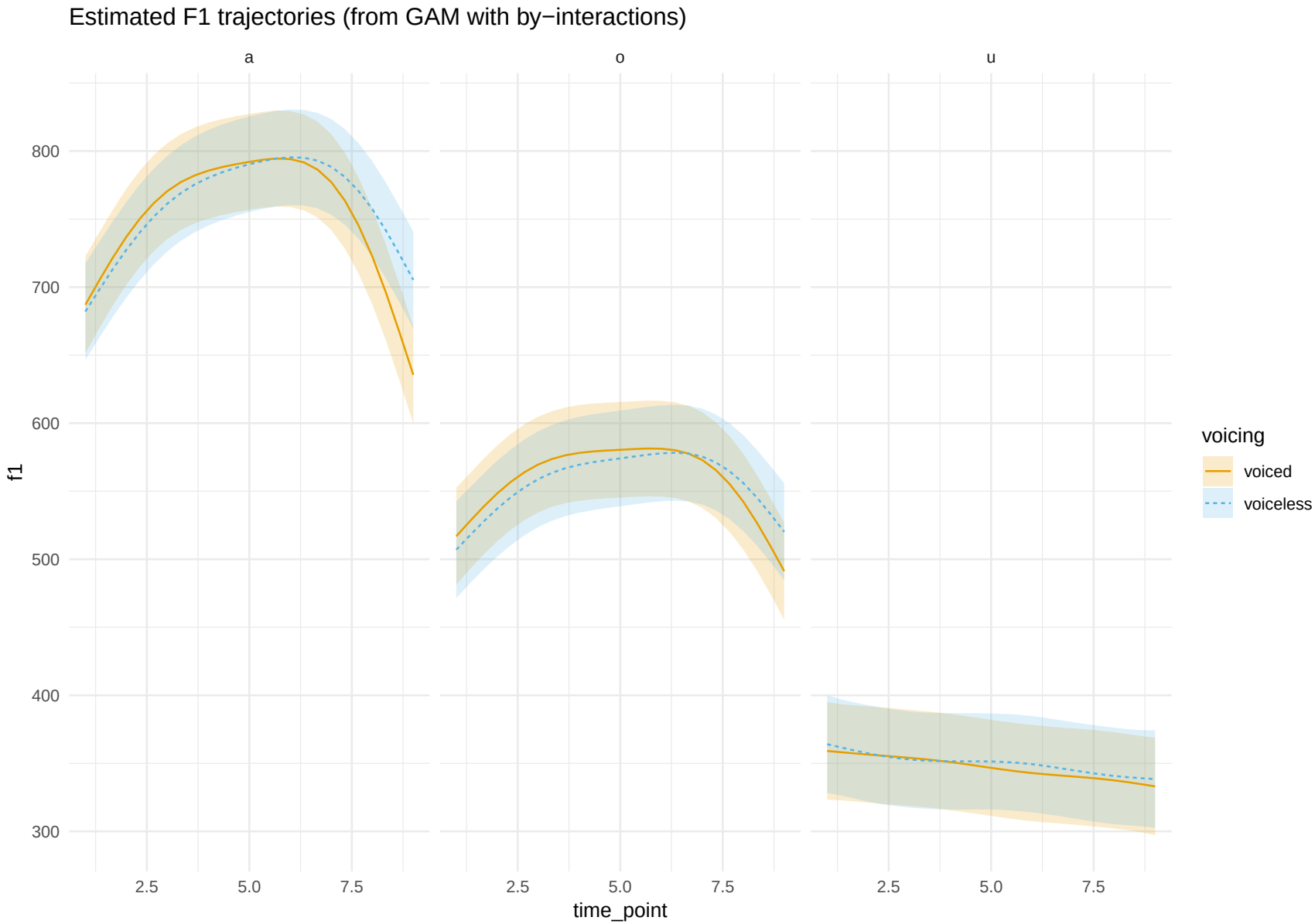
Interactions

```
## 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)         754.213     17.546   42.986  < 2e-16 ***
## vow_voio.voiceless -201.306      1.967 -102.352  < 2e-16 ***
## vow_voiu.voiceless -404.157      2.035 -198.626  < 2e-16 ***
## vow_voia.voiced      -9.593      1.975   -4.857 1.22e-06 ***
## vow_voio.voiced    -200.731      1.975 -101.620  < 2e-16 ***
## vow_voiu.voiced    -407.434      2.035 -200.237  < 2e-16 ***
## ---
```

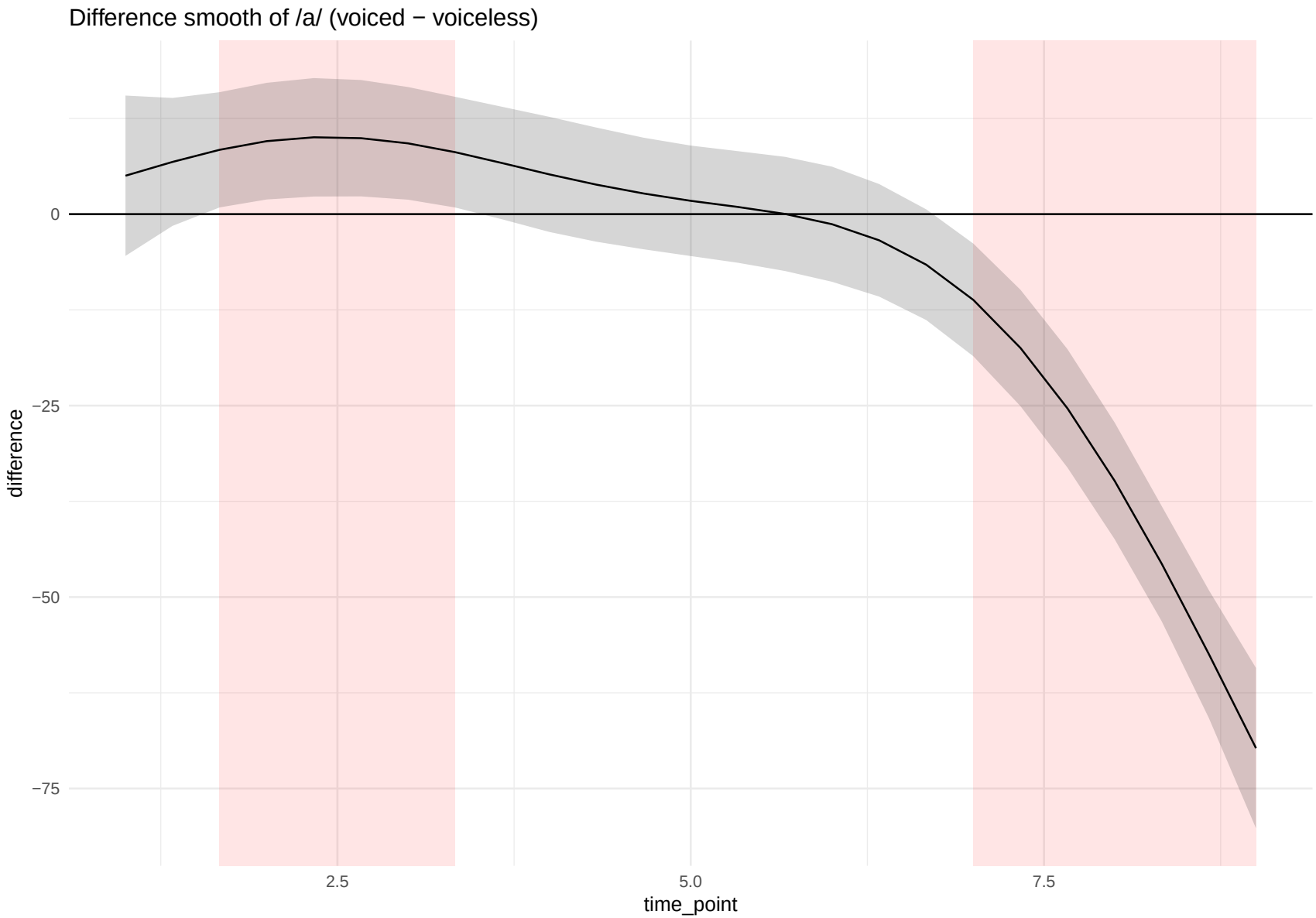
Interactions

```
## Approximate significance of smooth terms:
##                                    edf Ref.df     F  p-value
## s(time_point)                    4.805  4.886 93.77  < 2e-16 ***
## s(time_point):vow_voio.voiceless 3.535  4.142 13.51 3.81e-11 ***
## s(time_point):vow_voiu.voiceless 4.634  4.913 88.08  < 2e-16 ***
## s(time_point):vow_voia.voiced    4.067  4.591 36.28  < 2e-16 ***
## s(time_point):vow_voio.voiced    2.790  3.374 23.25 5.86e-16 ***
## s(time_point):vow_voiu.voiced    4.576  4.890 88.70  < 2e-16 ***
## s(time_point,speaker)           45.865 79.000 91.85  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.934   Deviance explained = 93.5%
## fREML =  32346  Scale est. = 2062.7    n = 6165
```

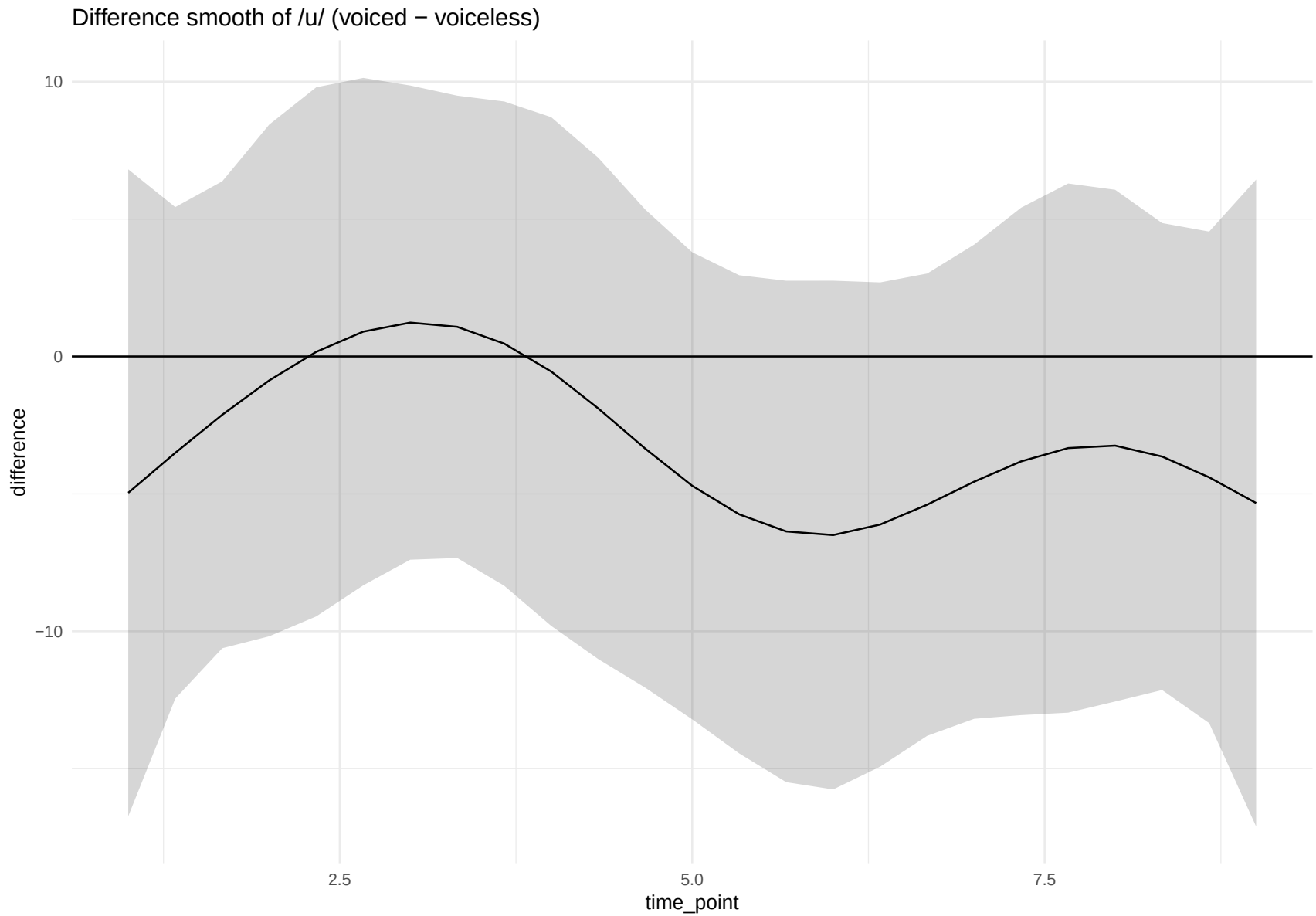
Interactions



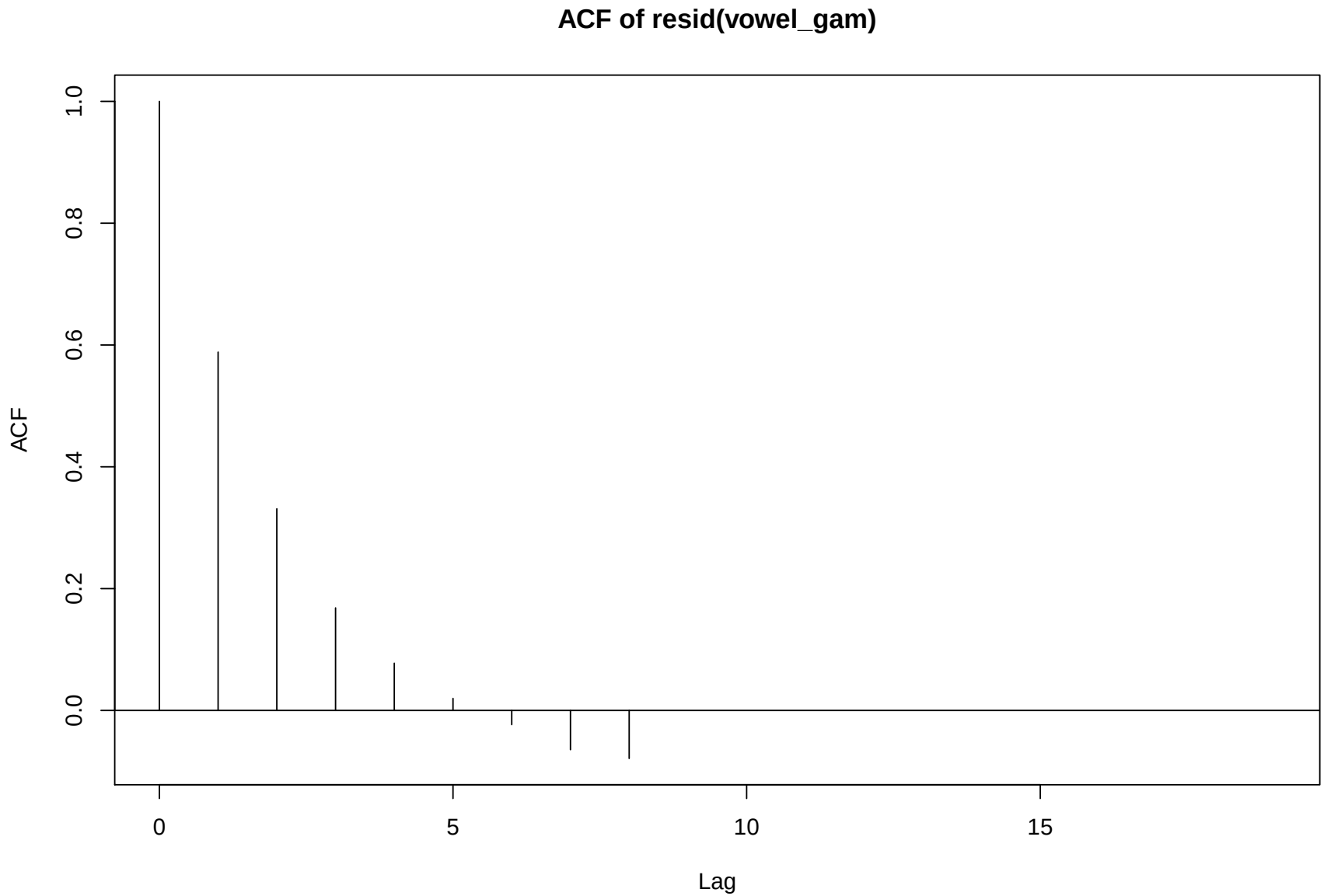
Interactions



Interactions



Residuals autocorrelation



Residuals autocorrelation

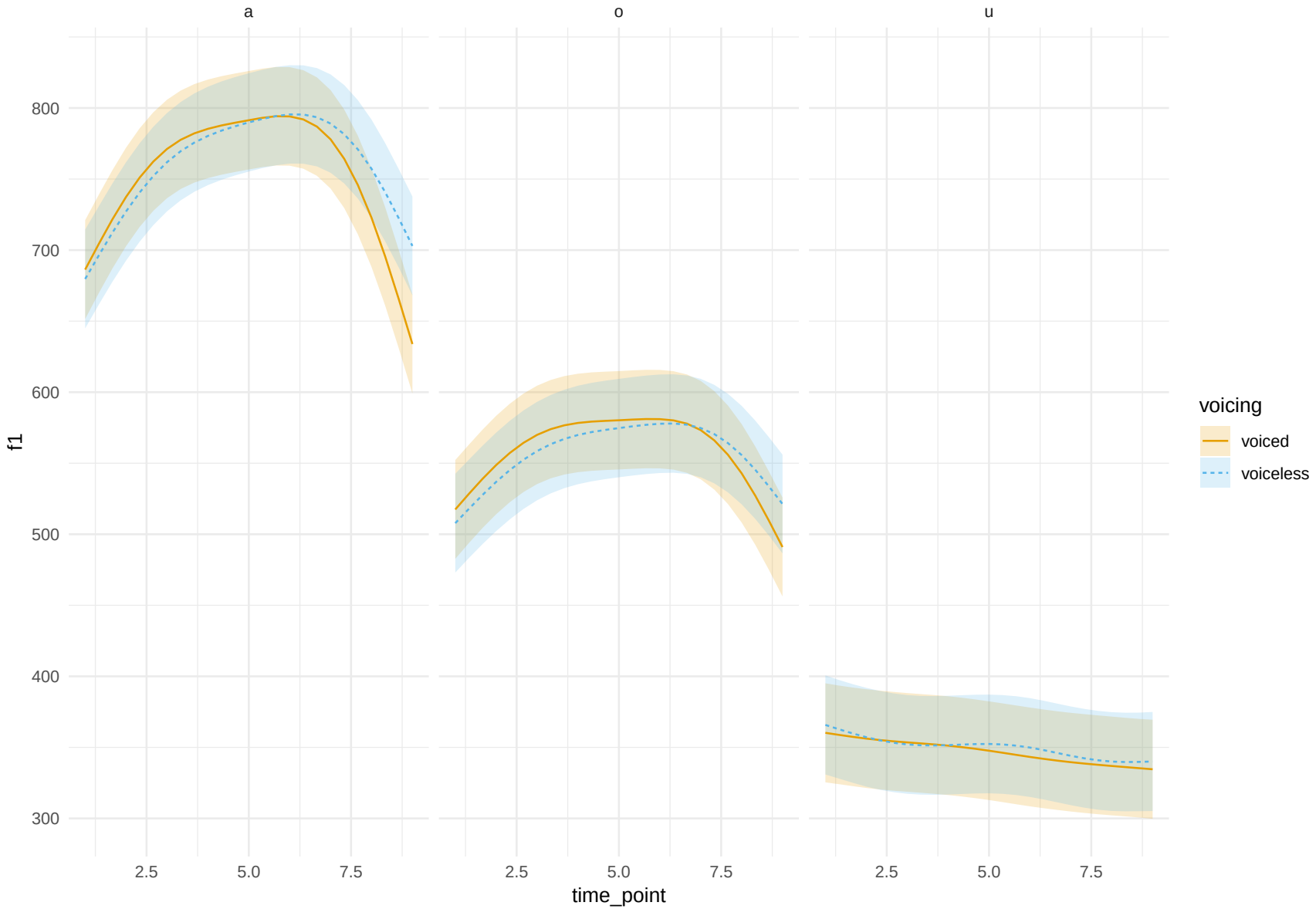
```
vowels <- vowels %>%
  arrange(speaker, index, time_point) %>%
  create_event_start("index")
```

Residuals autocorrelation

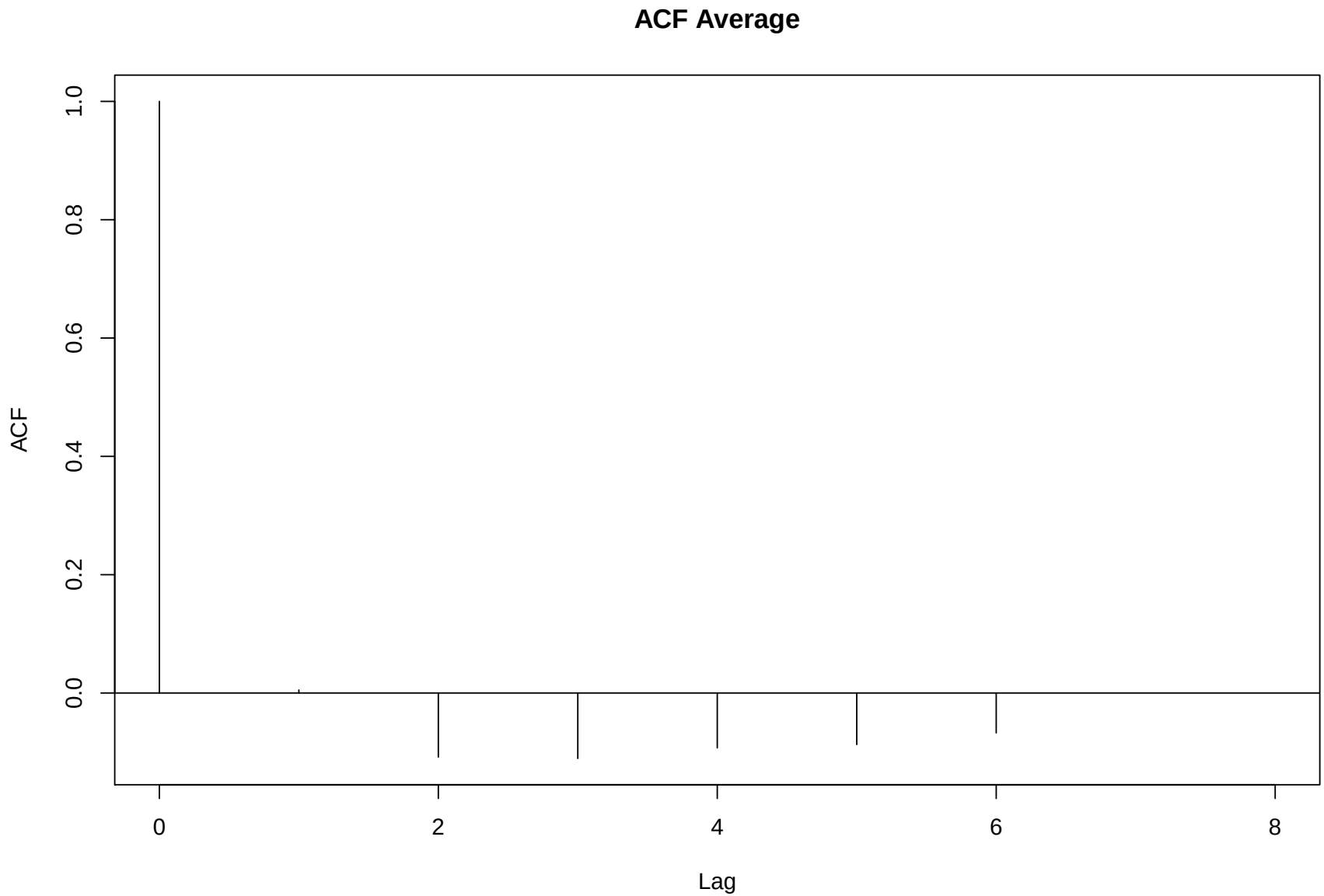
```
rho <- start_value_rho(vowel_gam)


ar_gam <- bam(
  f1 ~
    vow_voi +
    s(time_point, k = 6) +
    s(time_point, k = 6, by = vow_voi) +
    s(time_point, speaker, bs = "fs", m = 1),
  data = vowels,
  rho = rho,
  AR.start = vowels$start.event
)
```

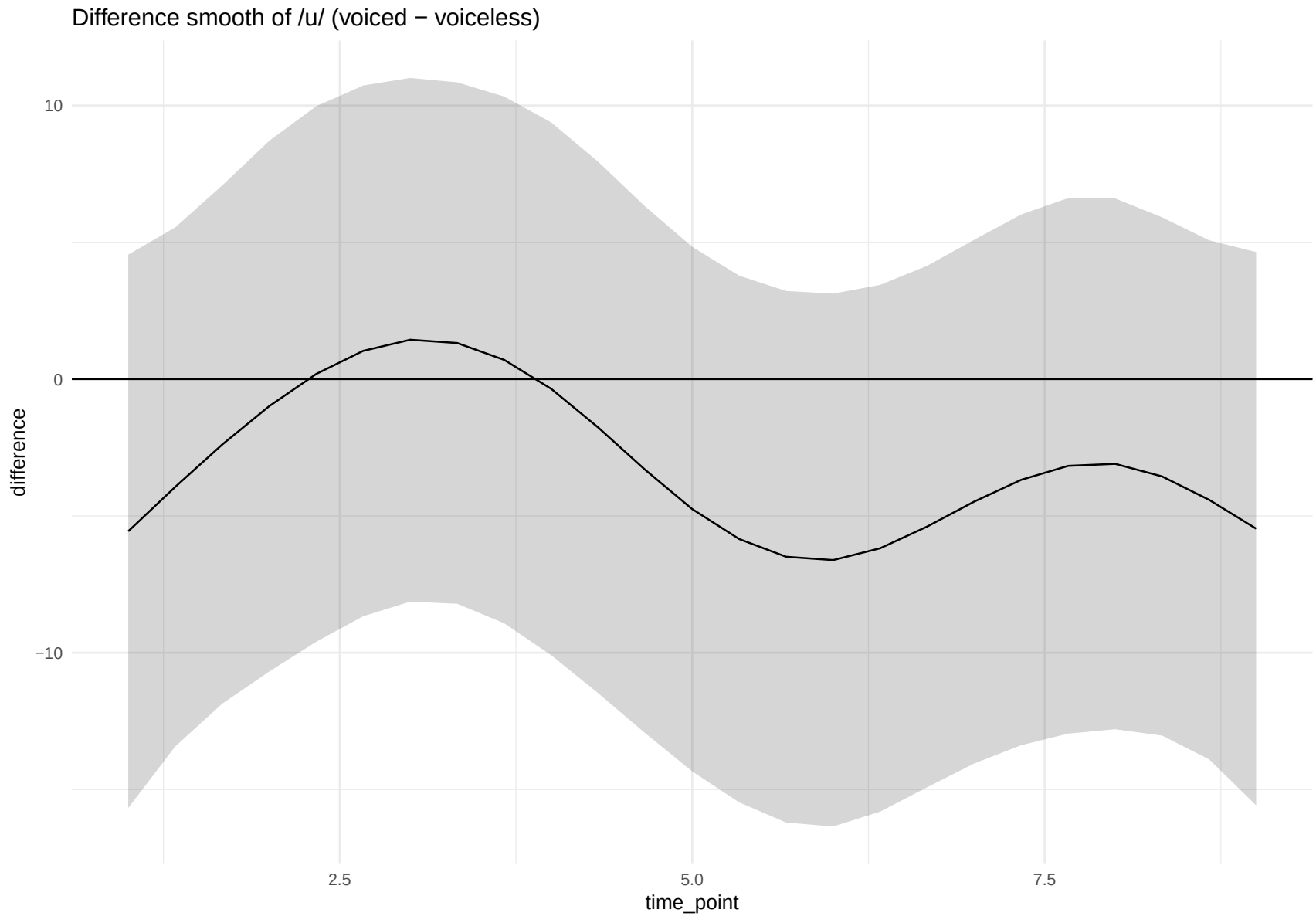
Residuals autocorrelation



Residuals autocorrelation



Residuals autocorrelation



Practical 2