An introduction to GAM(M)s

Stefano Coretta

12/07/2018

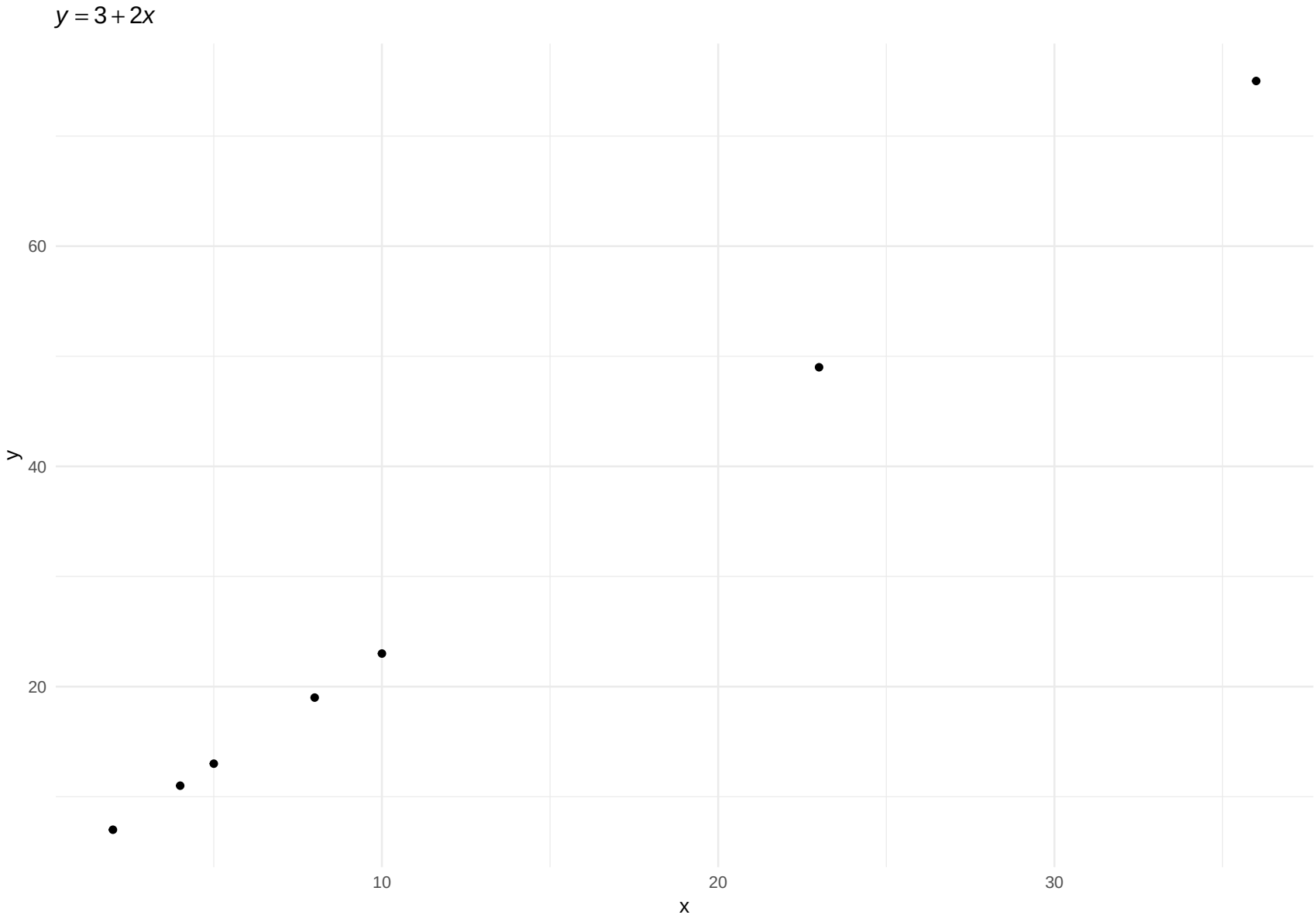
Time travel. . .

Linear models

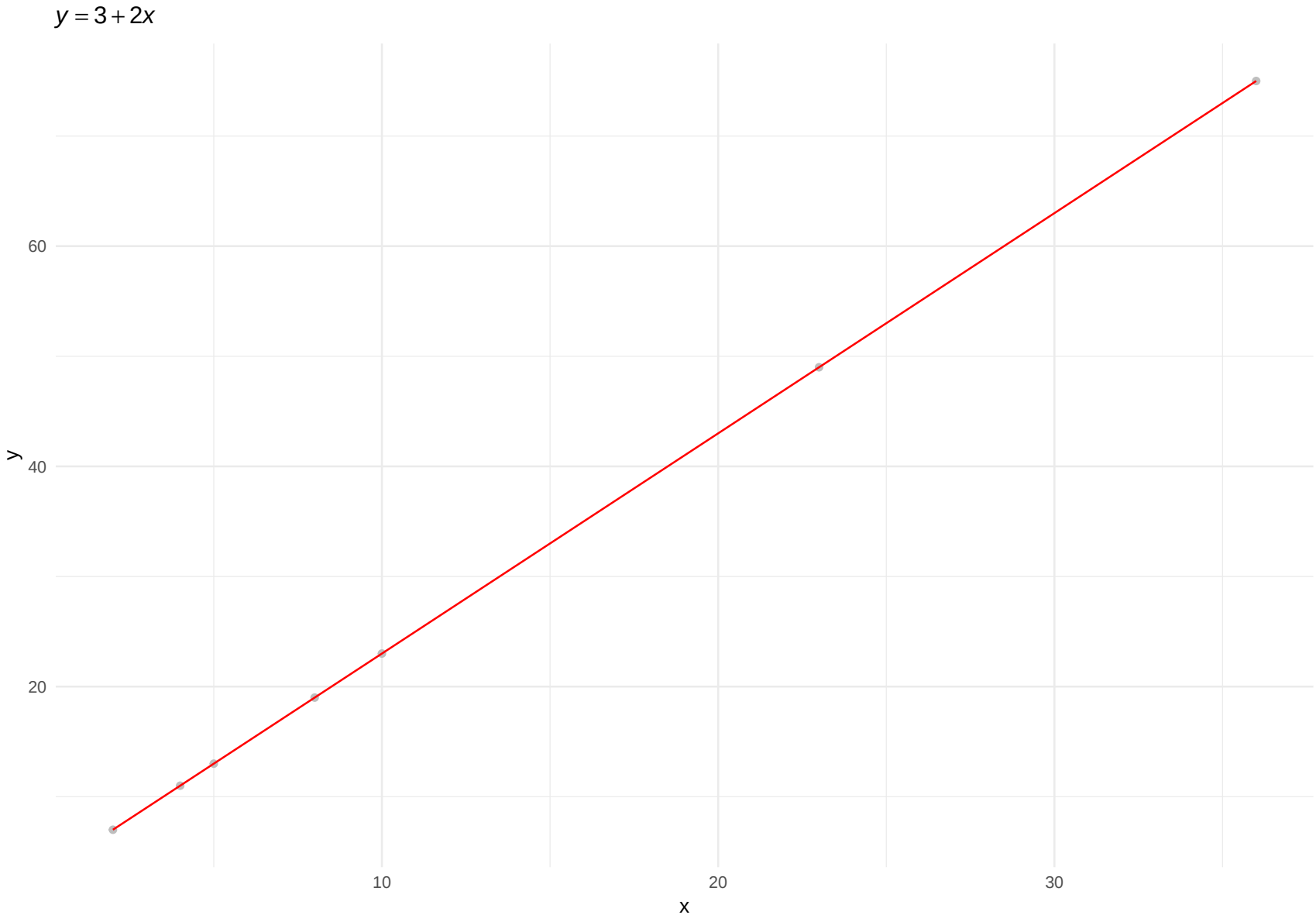
$$y = 3 + 2x$$

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

Linear models



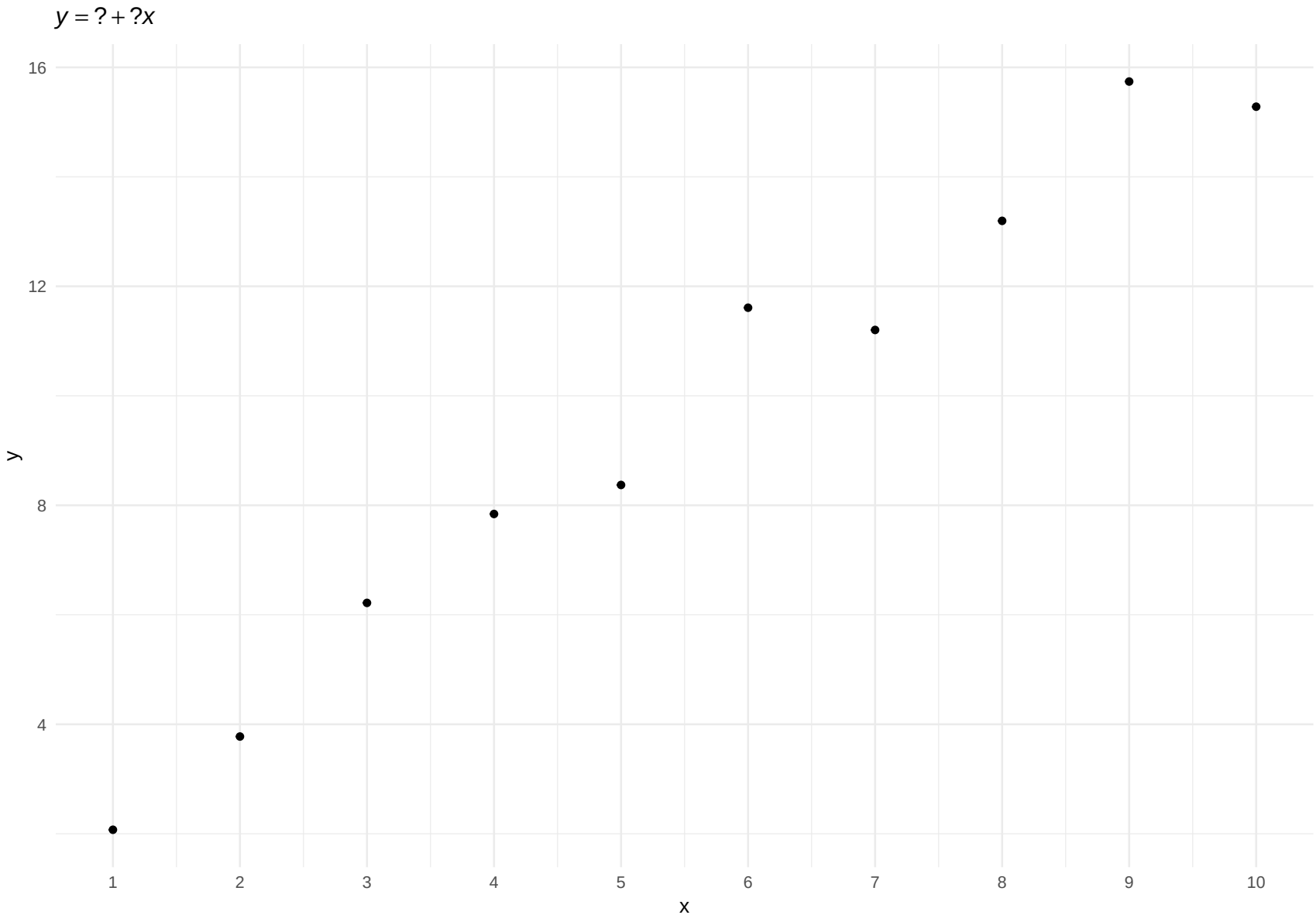
Linear models



Linear models

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

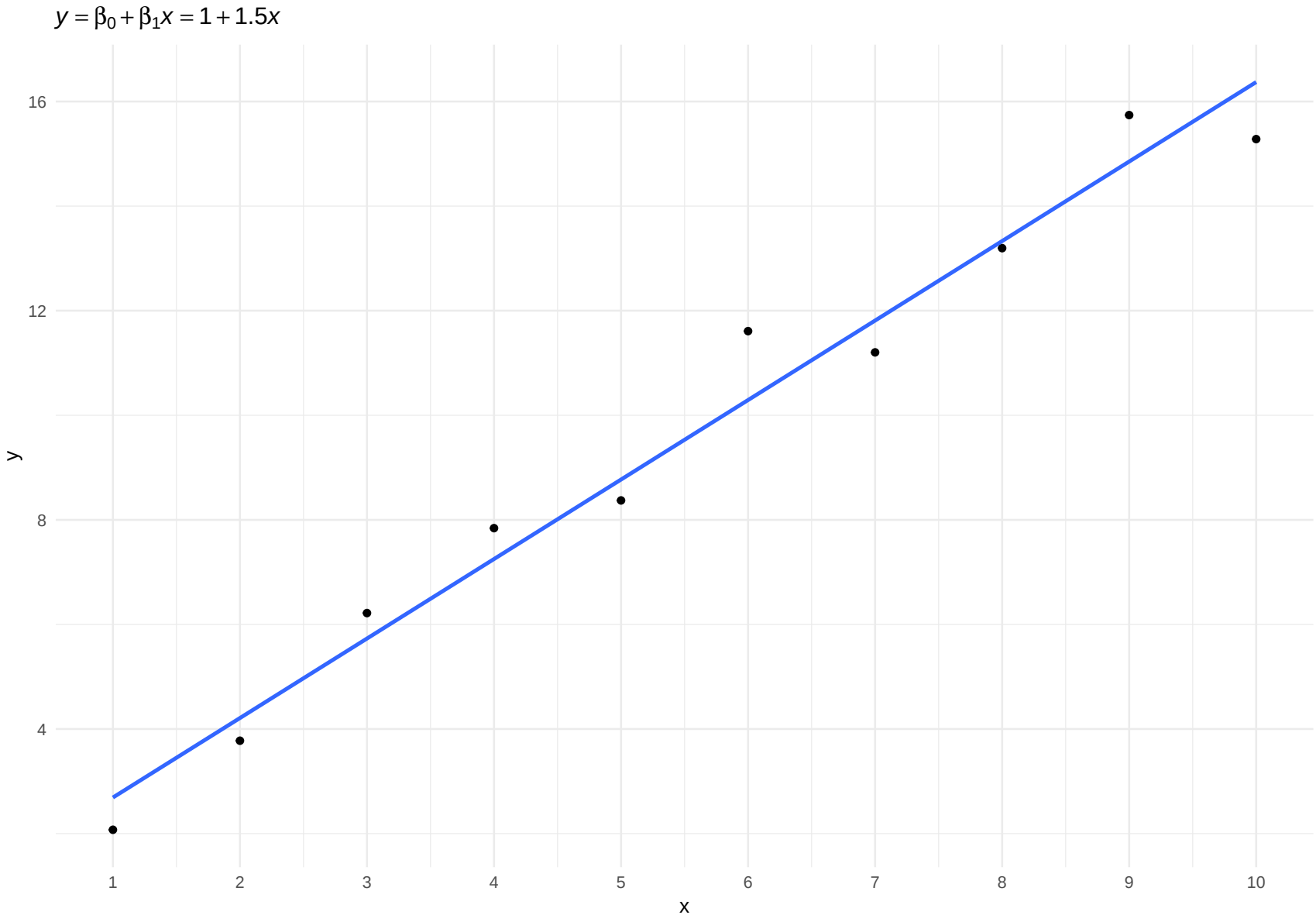
Linear models



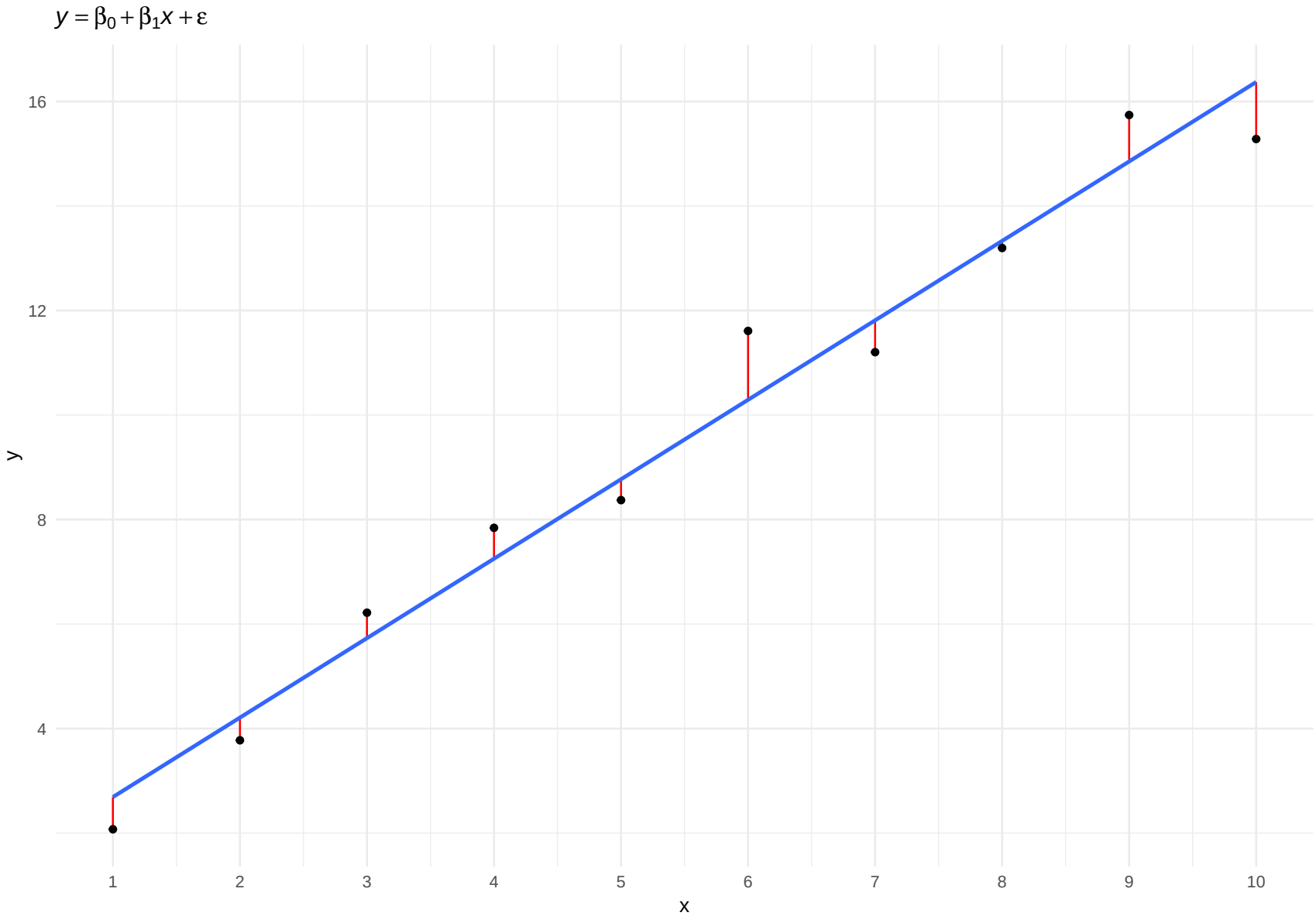
Linear models

- 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')

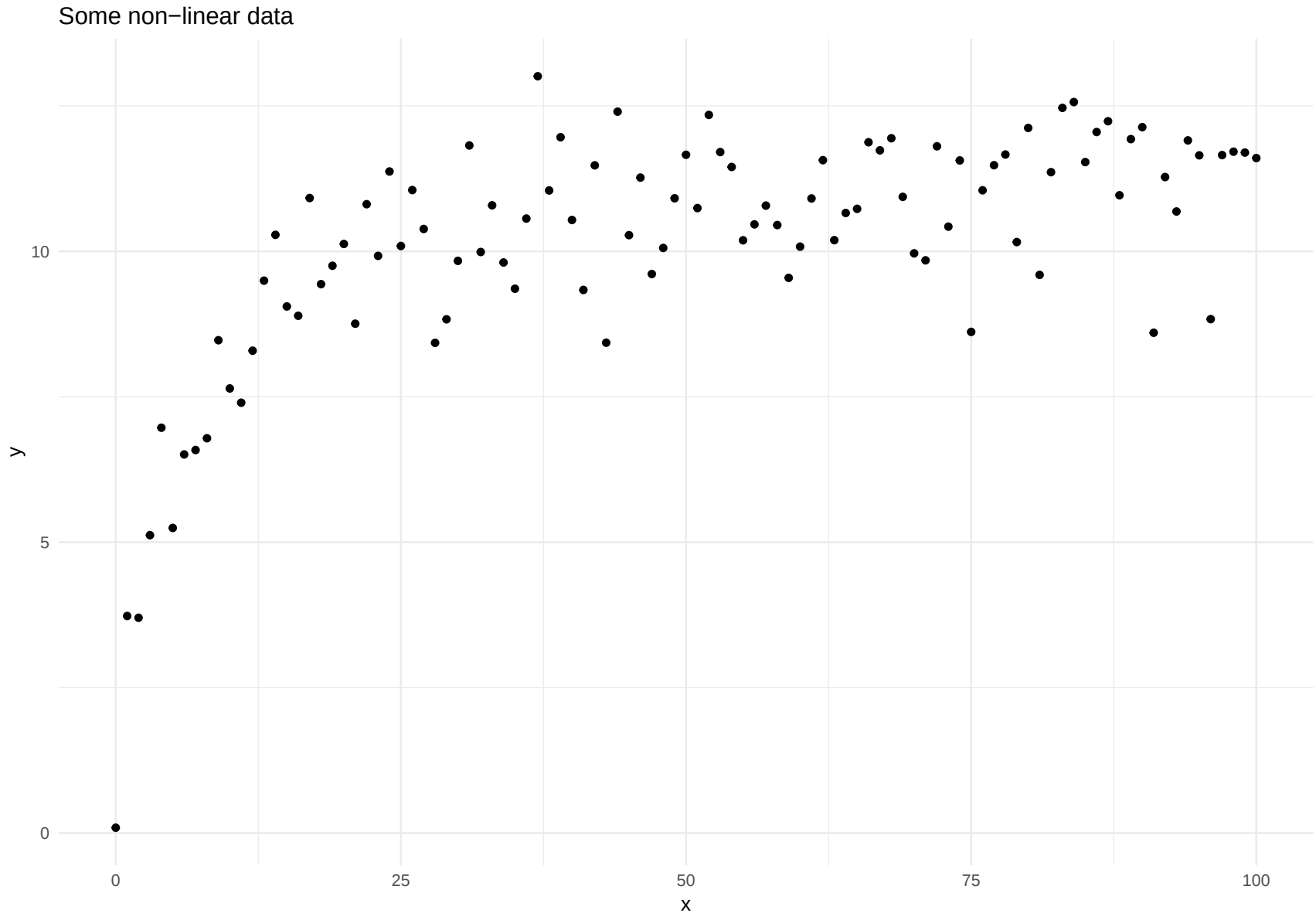
Linear models



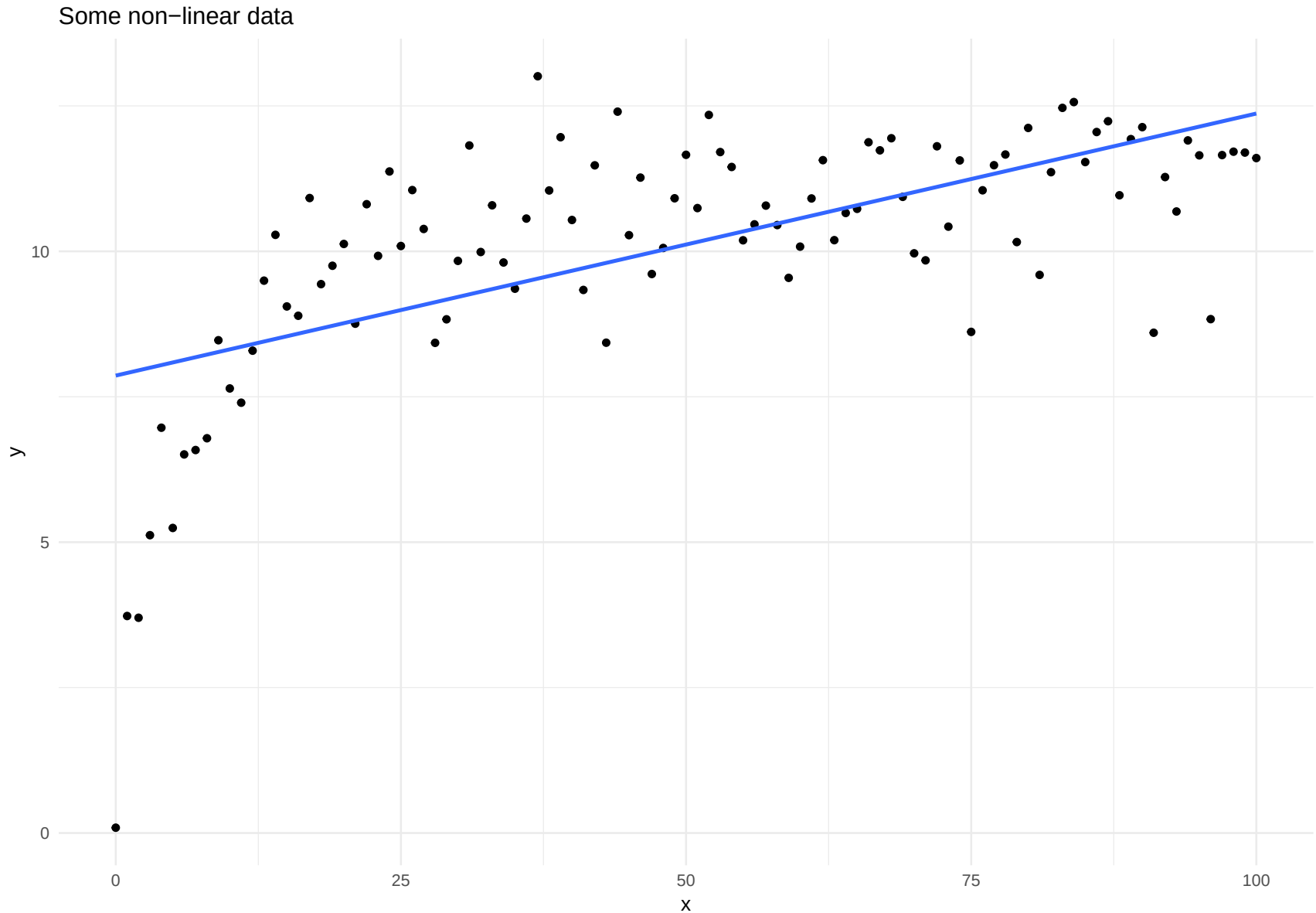
Linear models



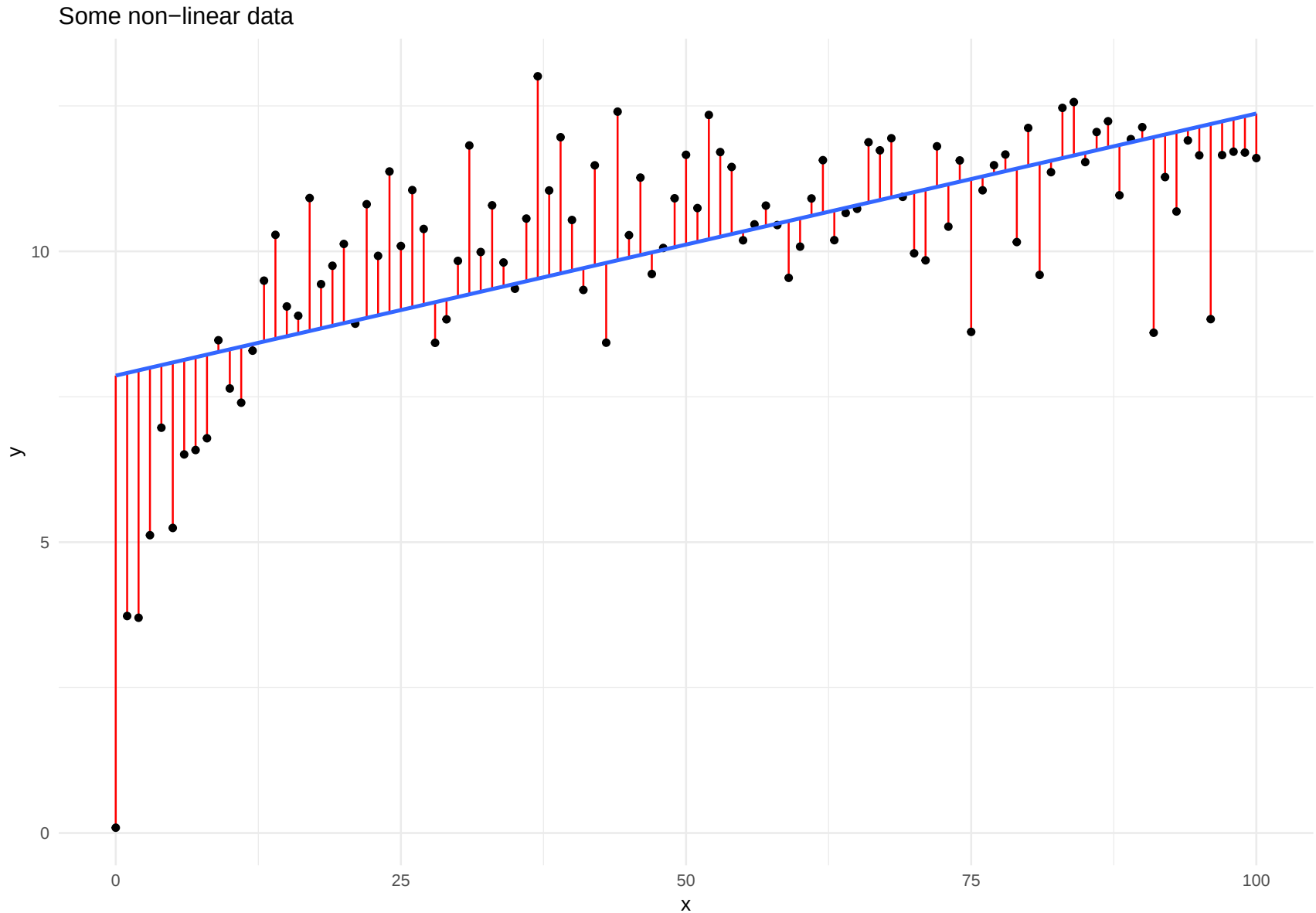
LM with non-linear data



LM with non-linear data



LM with non-linear data

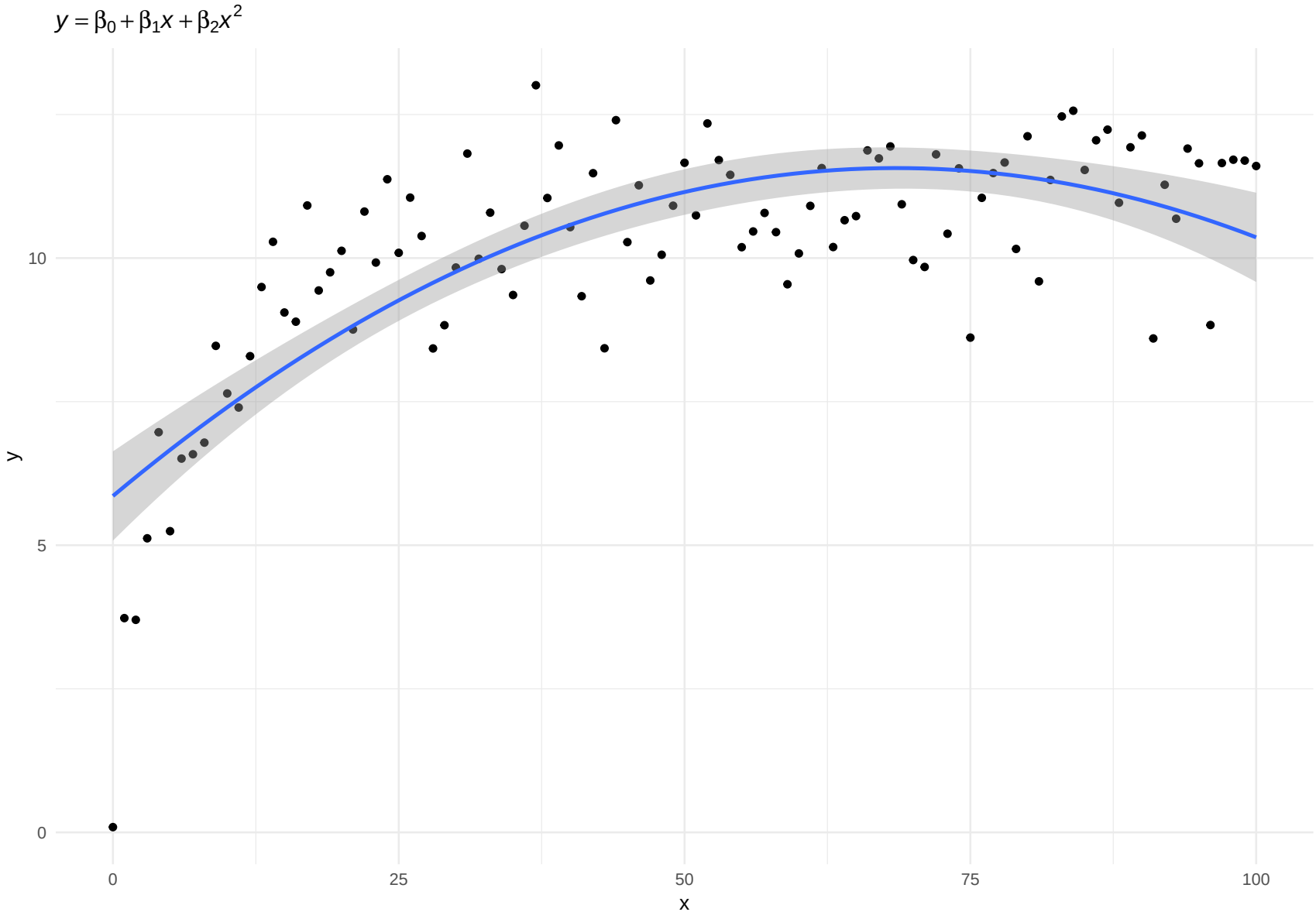


LM with non-linear data

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

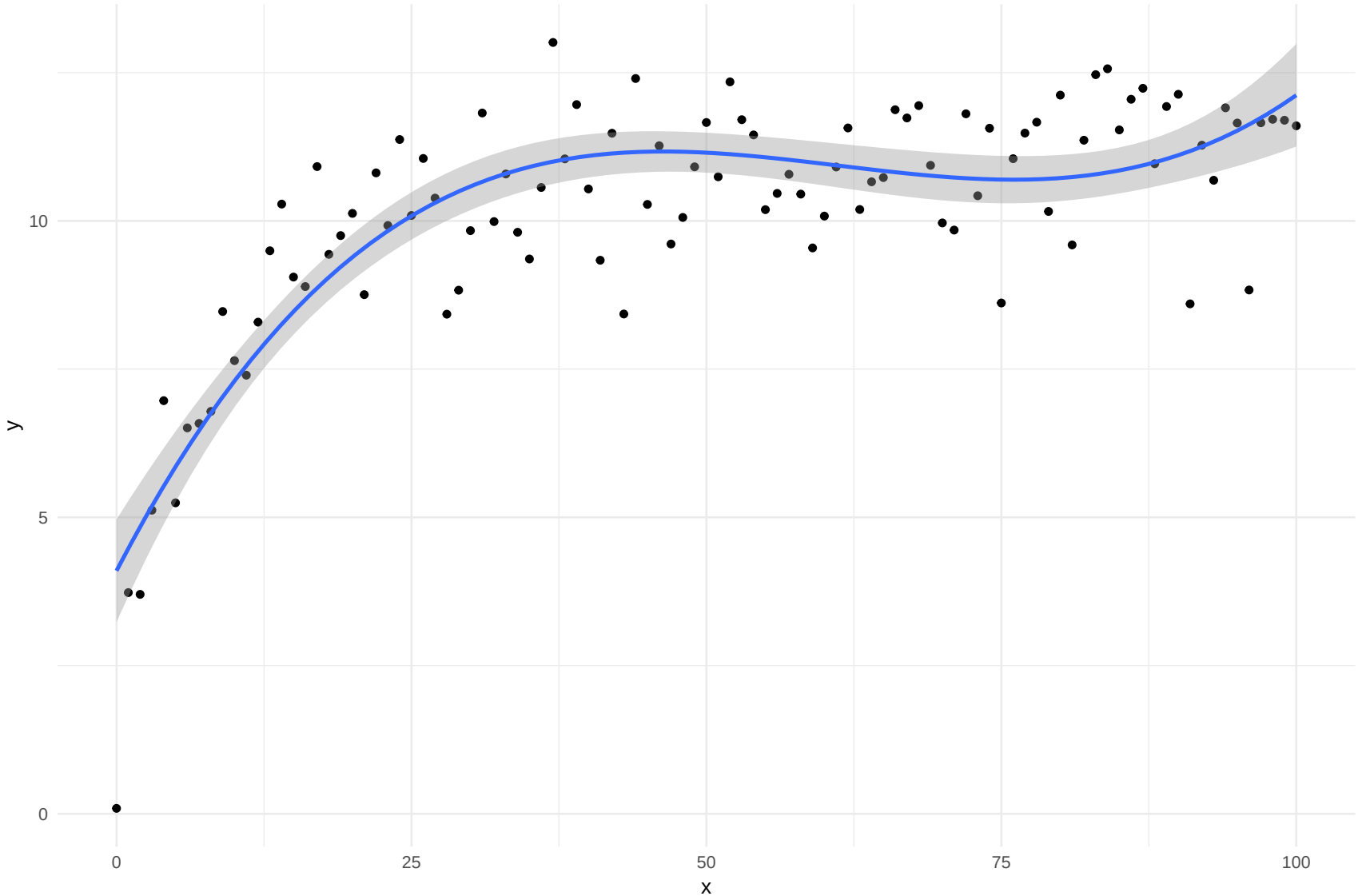
- Use **higher-degree polynomials**
  - quadratic: $y = \beta_0 + \beta_1 x + \beta_2 x^2$
  - cubic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$
  - nth: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + ... + \beta_n x^n$

LM with non-linear data



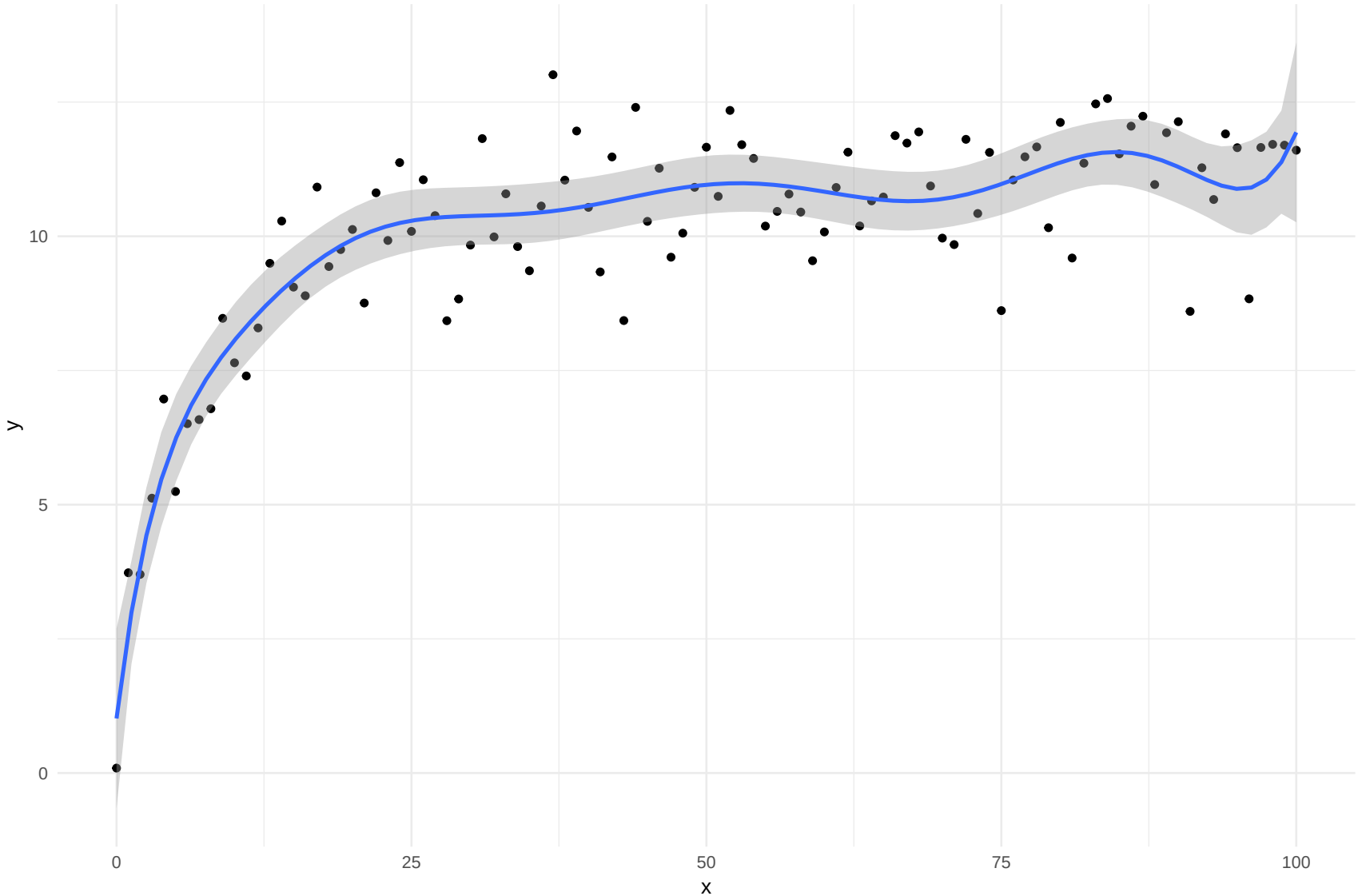
LM with non-linear data

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



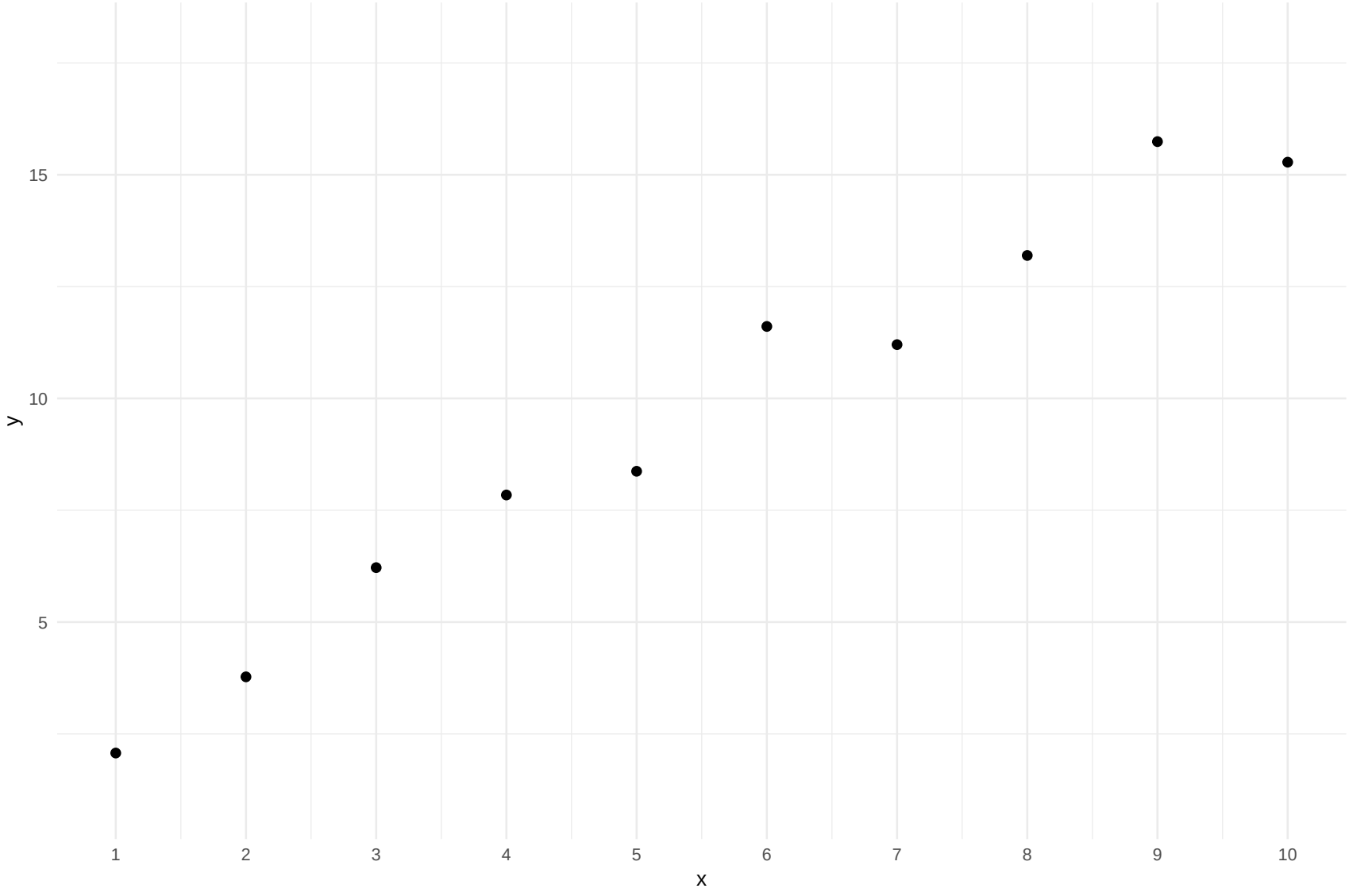
LM with non-linear data

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



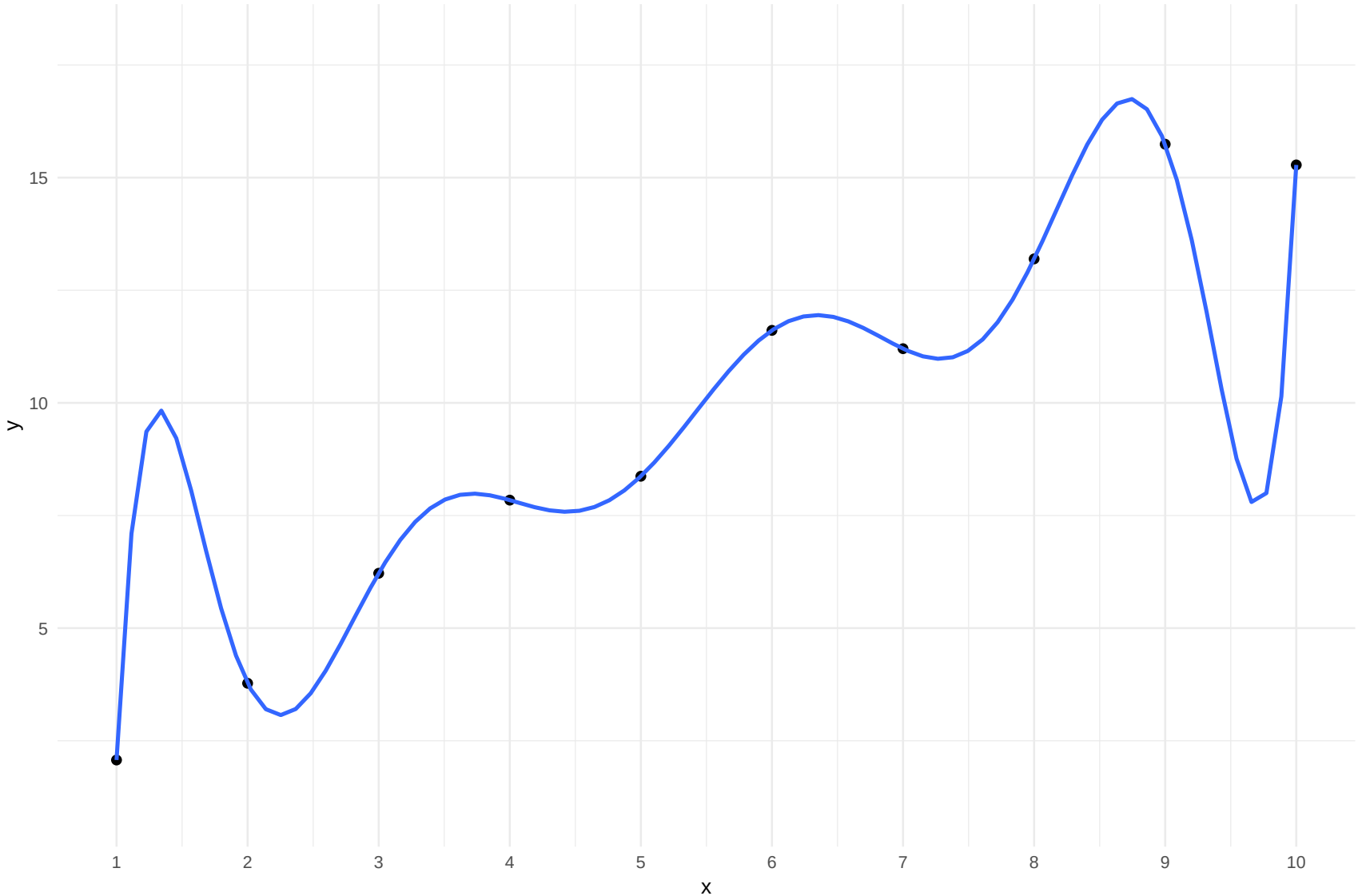
LM with non-linear data

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



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*)

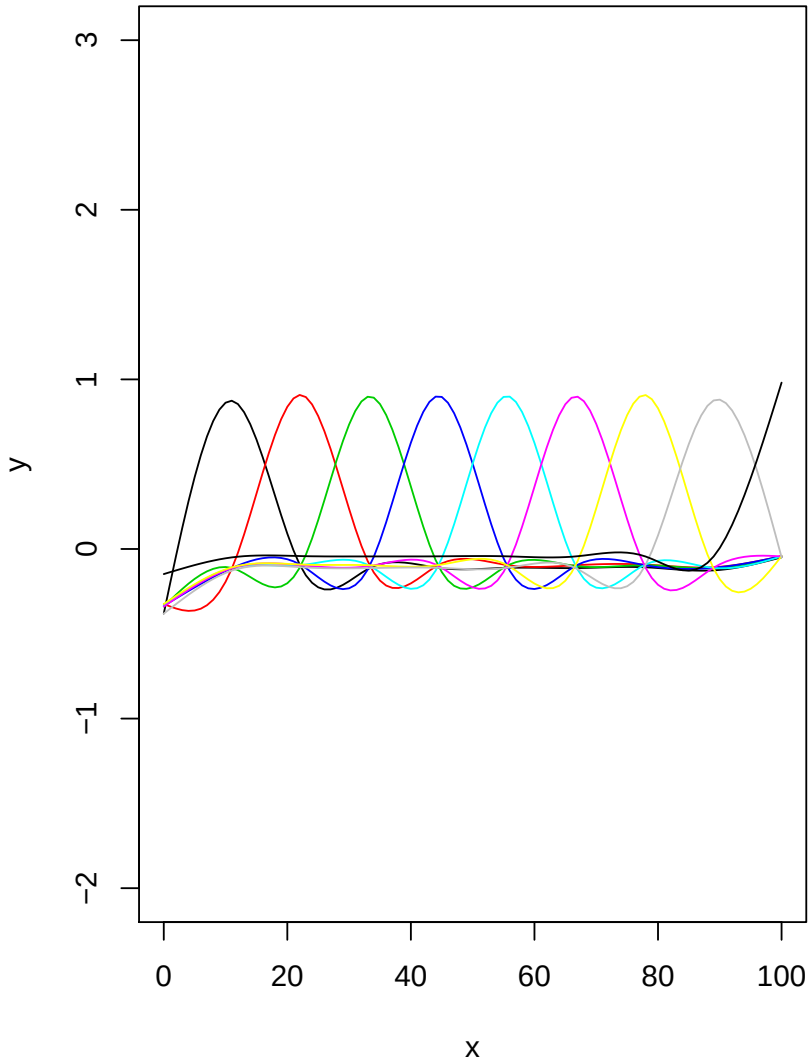
Smooth terms

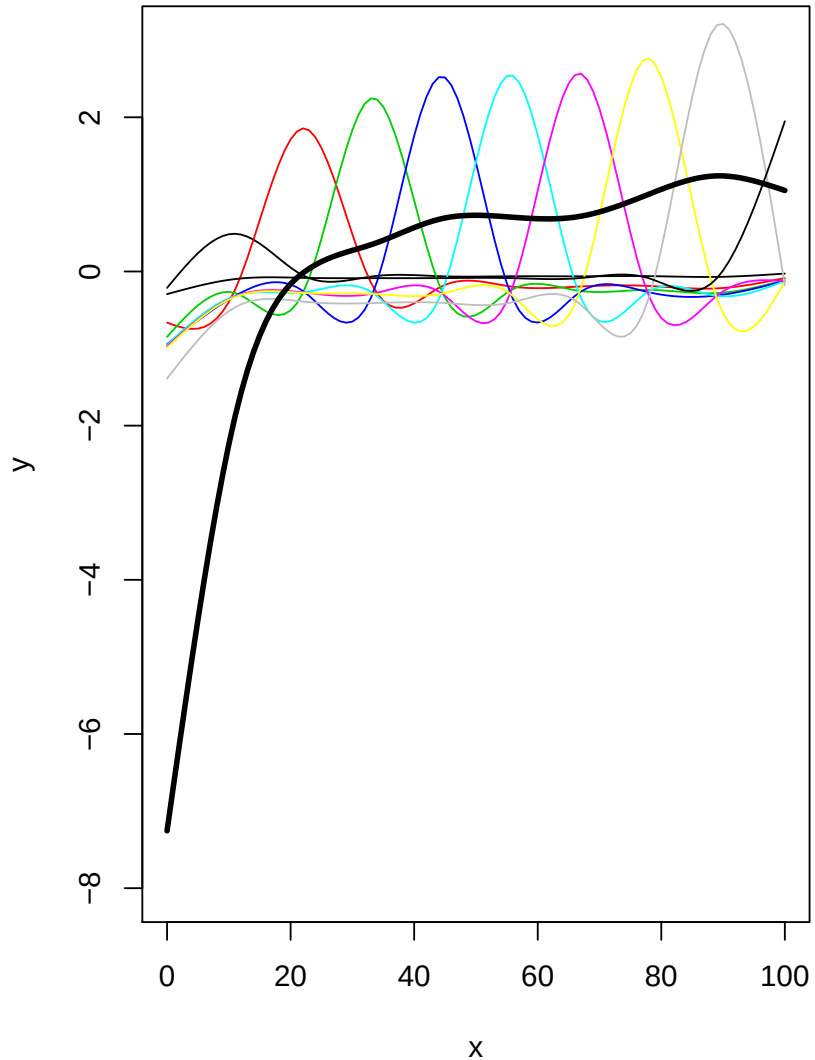
- LMs have **parametric terms**
  - $\beta_n x_n$
  - `x` in `R`
  - linear effects
- GAMs add (non-parametric) **smooth terms** (or simply smooths, also smoothers)
  - $f(x)$
  - `s(x)` in `R`
  - non-linear effects
- `gam(y ~ s(x), data)`, 'y as *some* function of x'

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

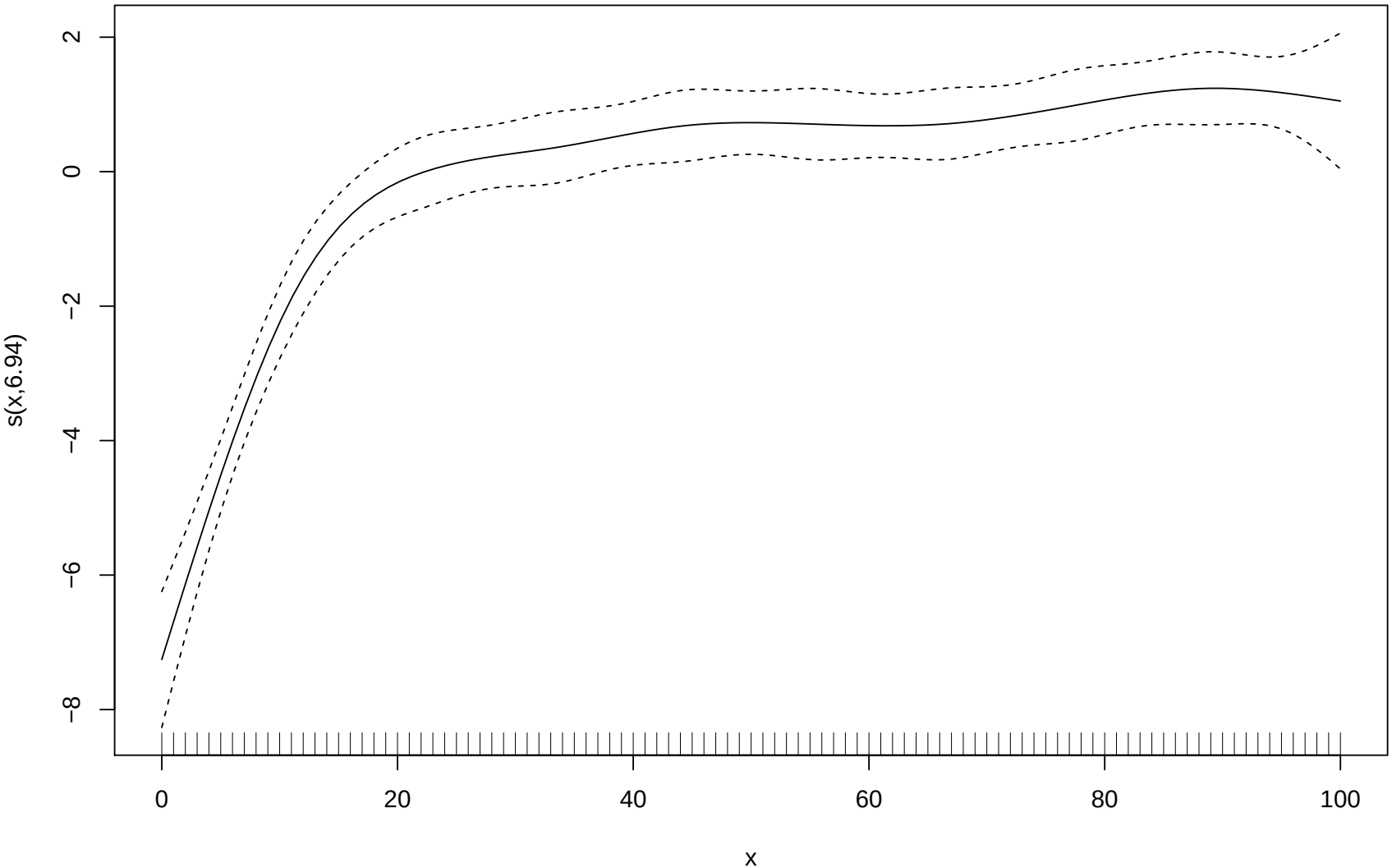
- There are **several kinds** of splines
  - each with their own basis functions
- Most common
  - *thin plate regression splines*
  - *cubic regression splines*
- For more info, run `?smooth.terms`

A simple GAM

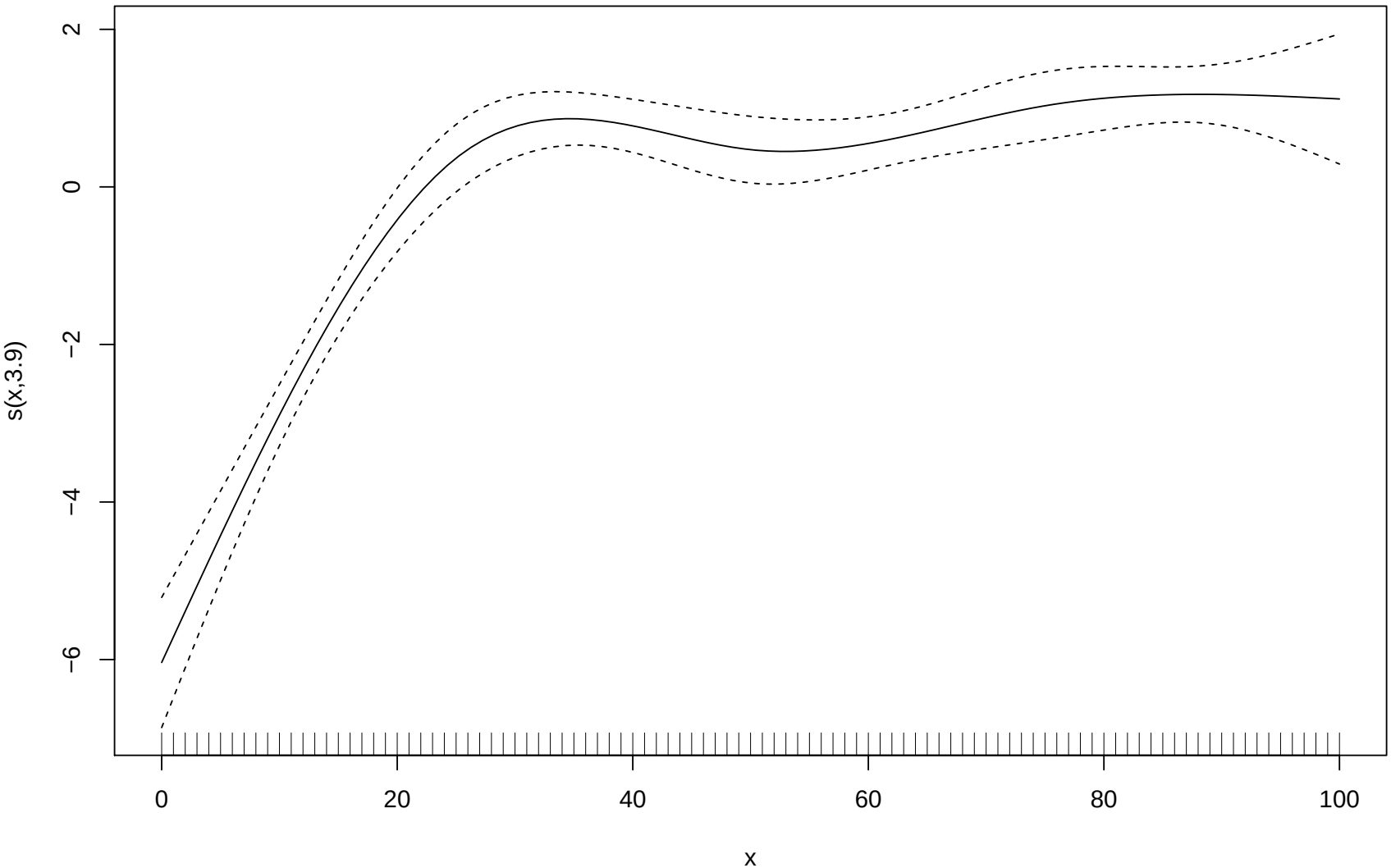
```r
simple <- gam(y ~ s(x, bs = "cr", k = 10), data = sim_nl_a)
summary(simple)
```

```
##
## 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
```

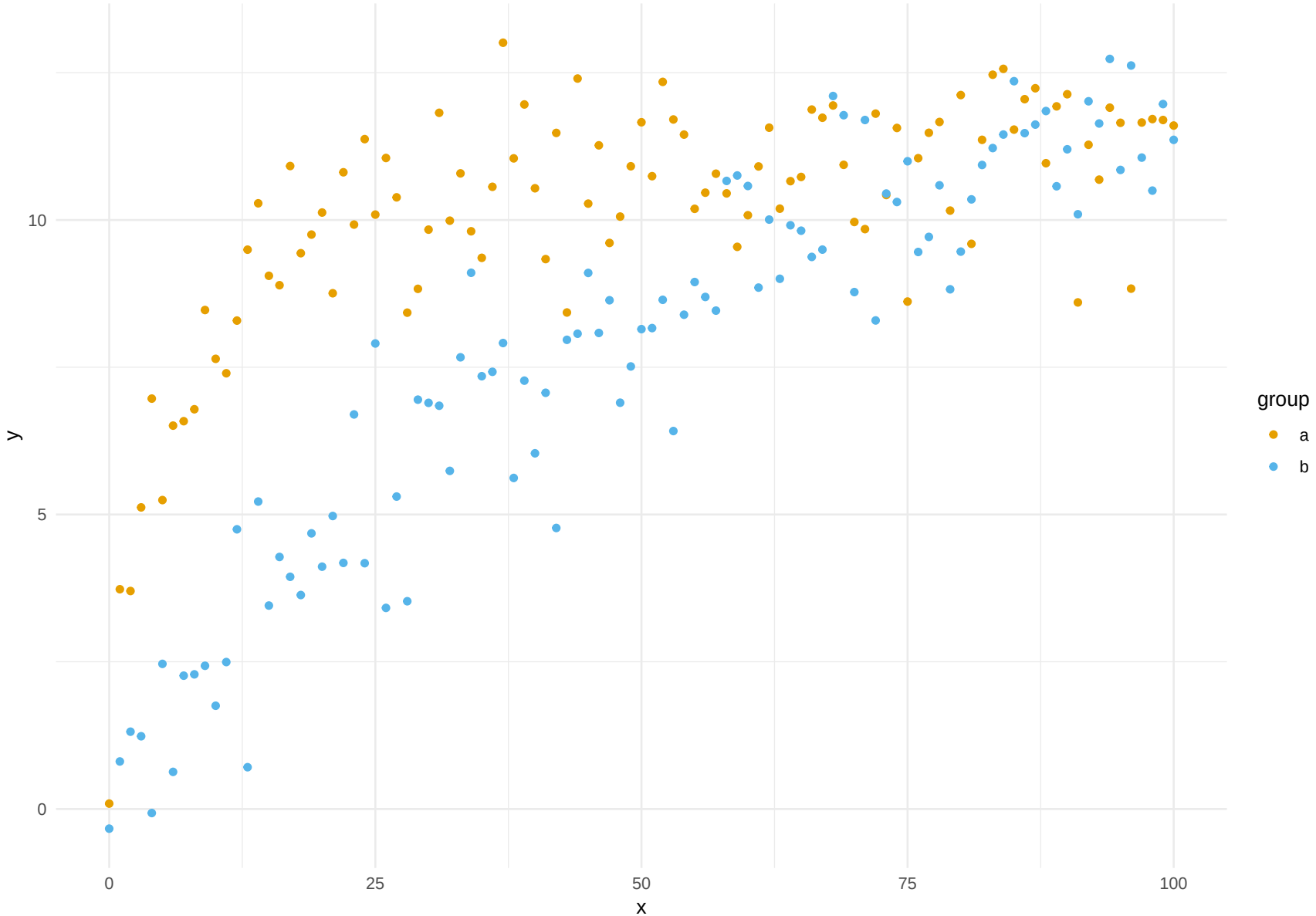
A simple GAM



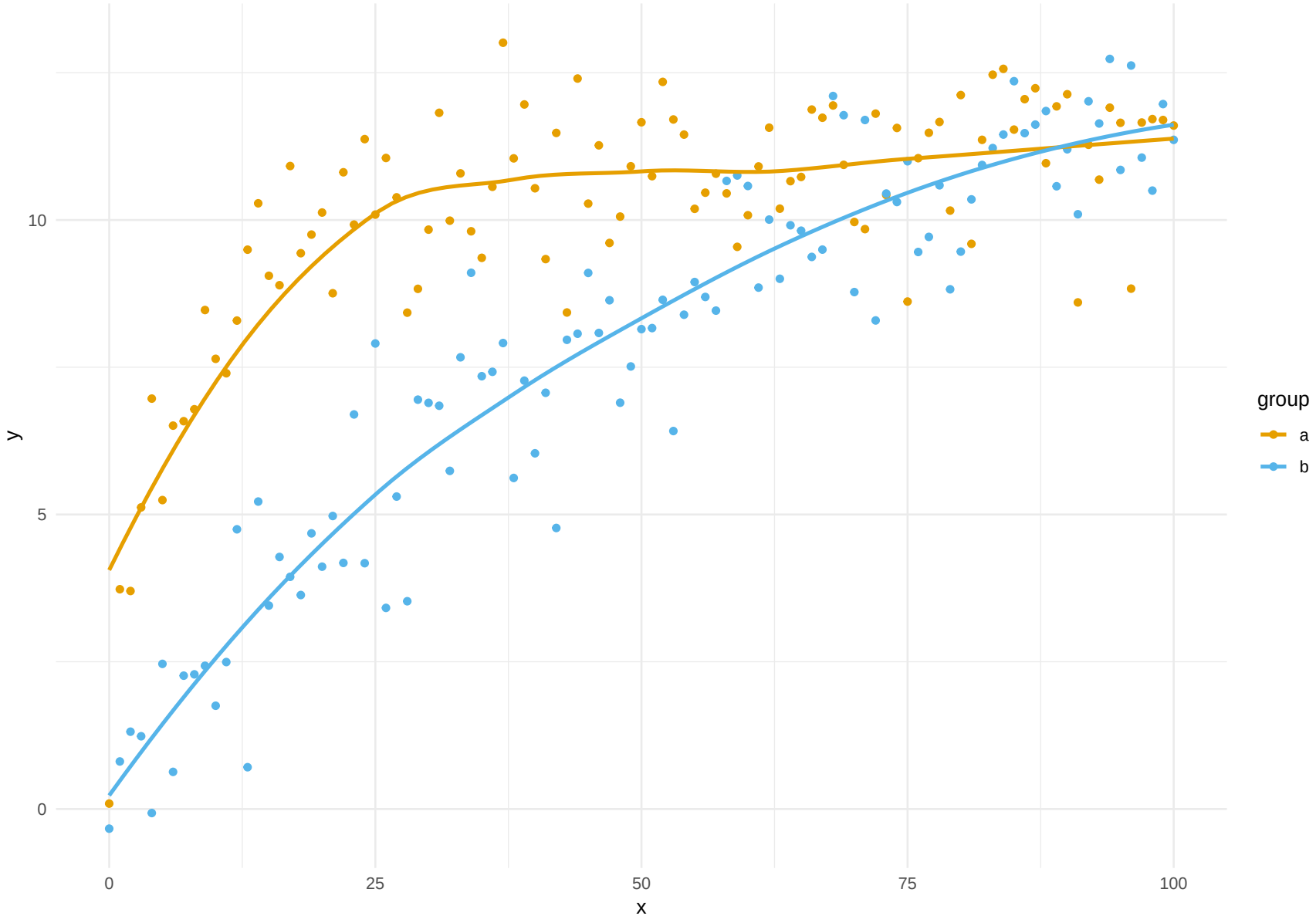
A simple GAM



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

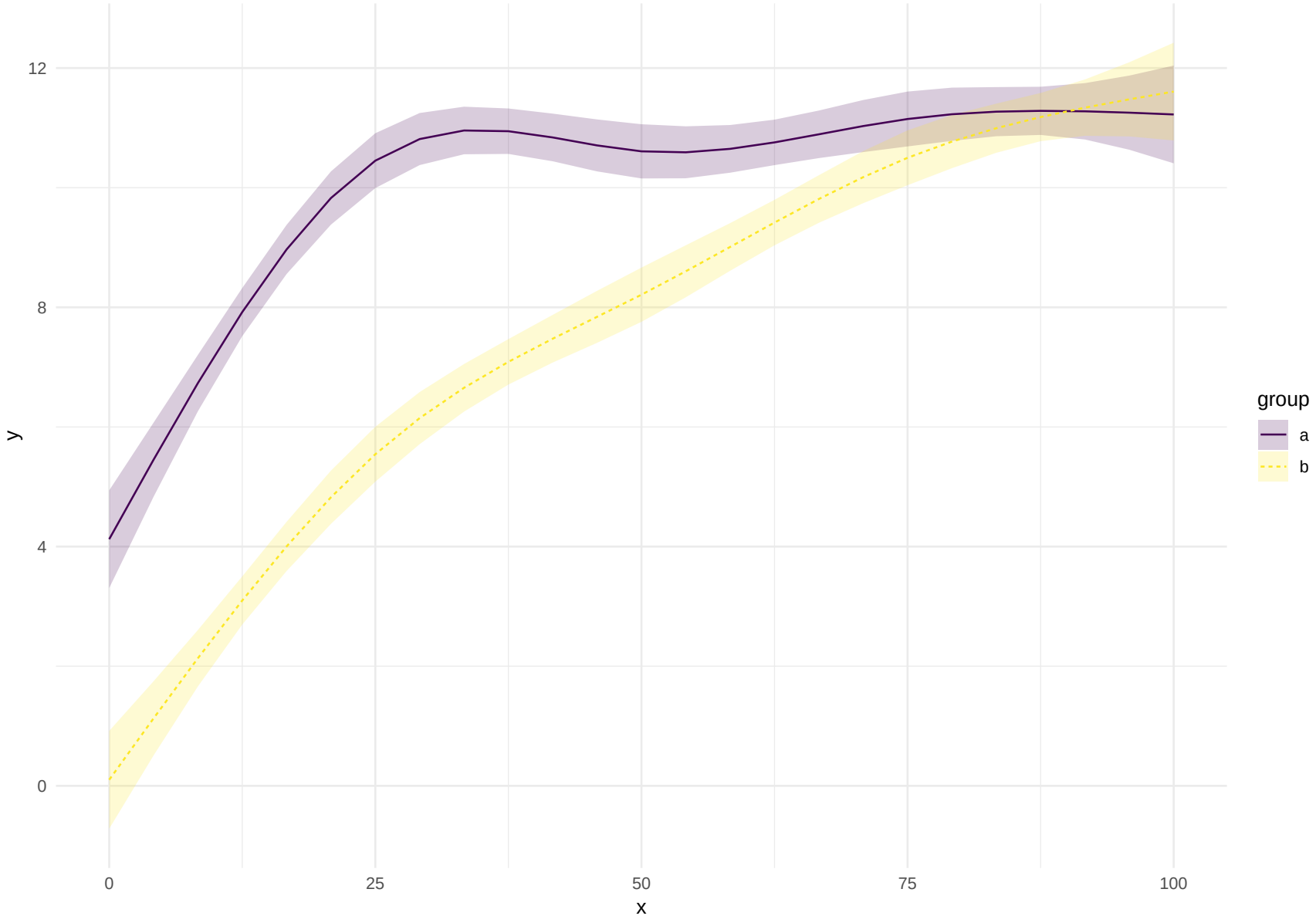
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ group + s(x, bs = "cr", k = 5) + s(x, bs = "cr", k = 5, by = group)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.1165     0.1096   92.34   <2e-16 ***
## groupb       -2.4947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

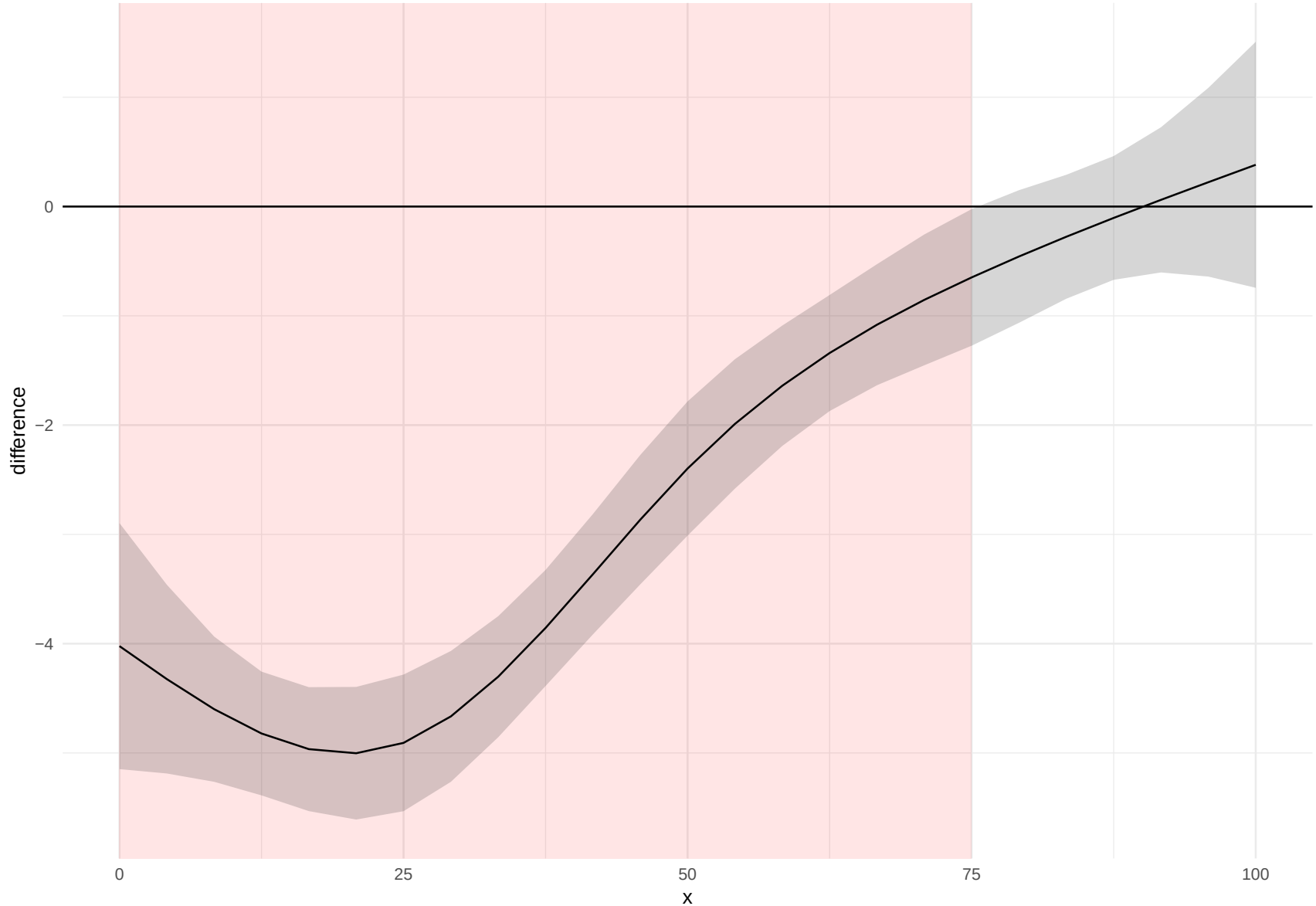
```
## compare_0: y ~ s(x, bs = "cr", k = 5)
##
## compare_1: y ~ group + s(x, bs = "cr", k = 5) + s(x, bs = "cr", k = 5, by =
##
## Chi-square test of ML scores
## -----
##       Model    Score Edf Difference    Df  p.value Sig.
## 1 compare_0 422.4827   3
## 2 compare_1 314.0105   6    108.472 3.000  < 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
- two main types
  - **time series data**
  - **spatial data**

Dynamic data

- more data ($n > 1000$)
- use `bam()` (big GAM)

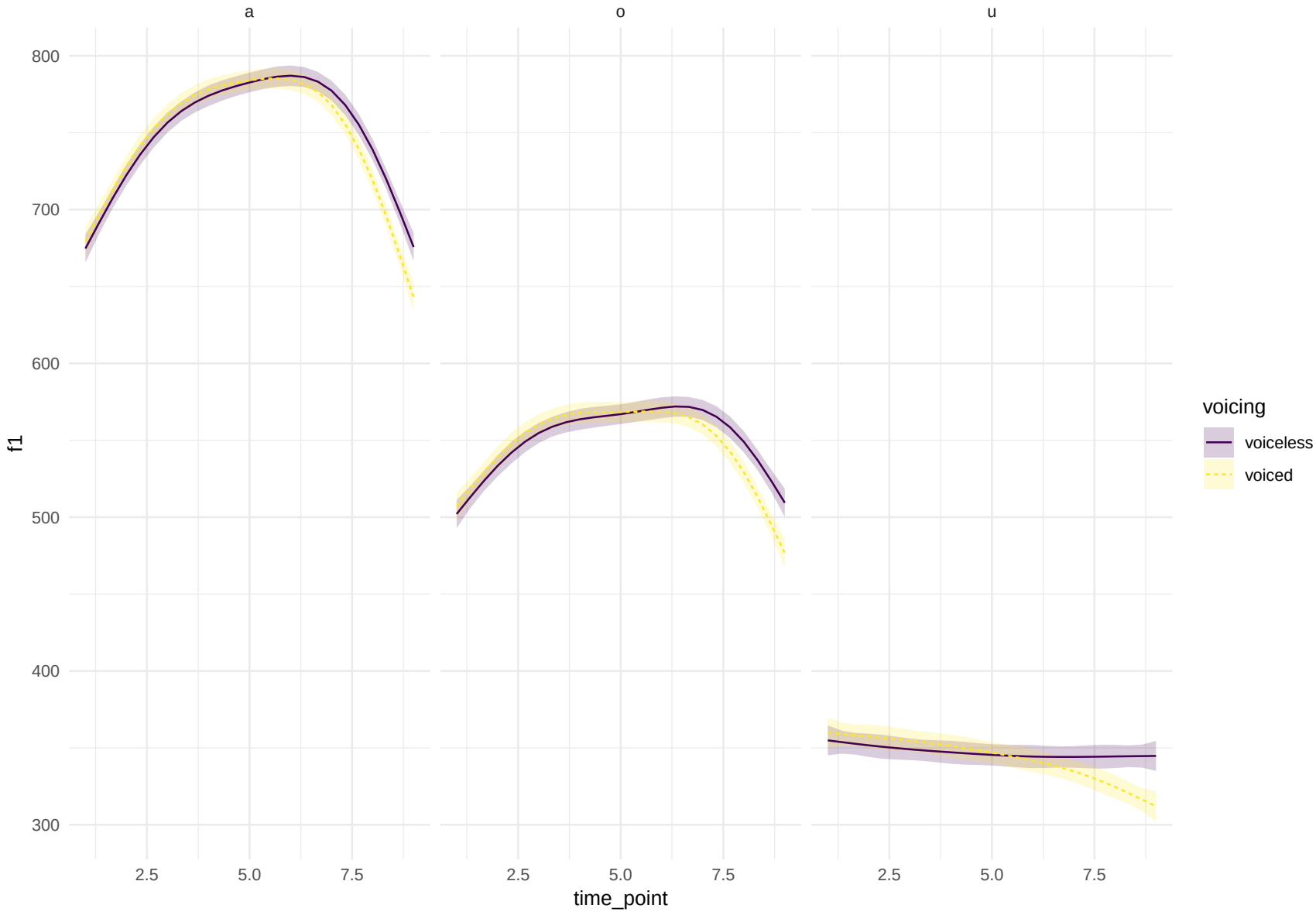
Dynamic data

```
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
)
```

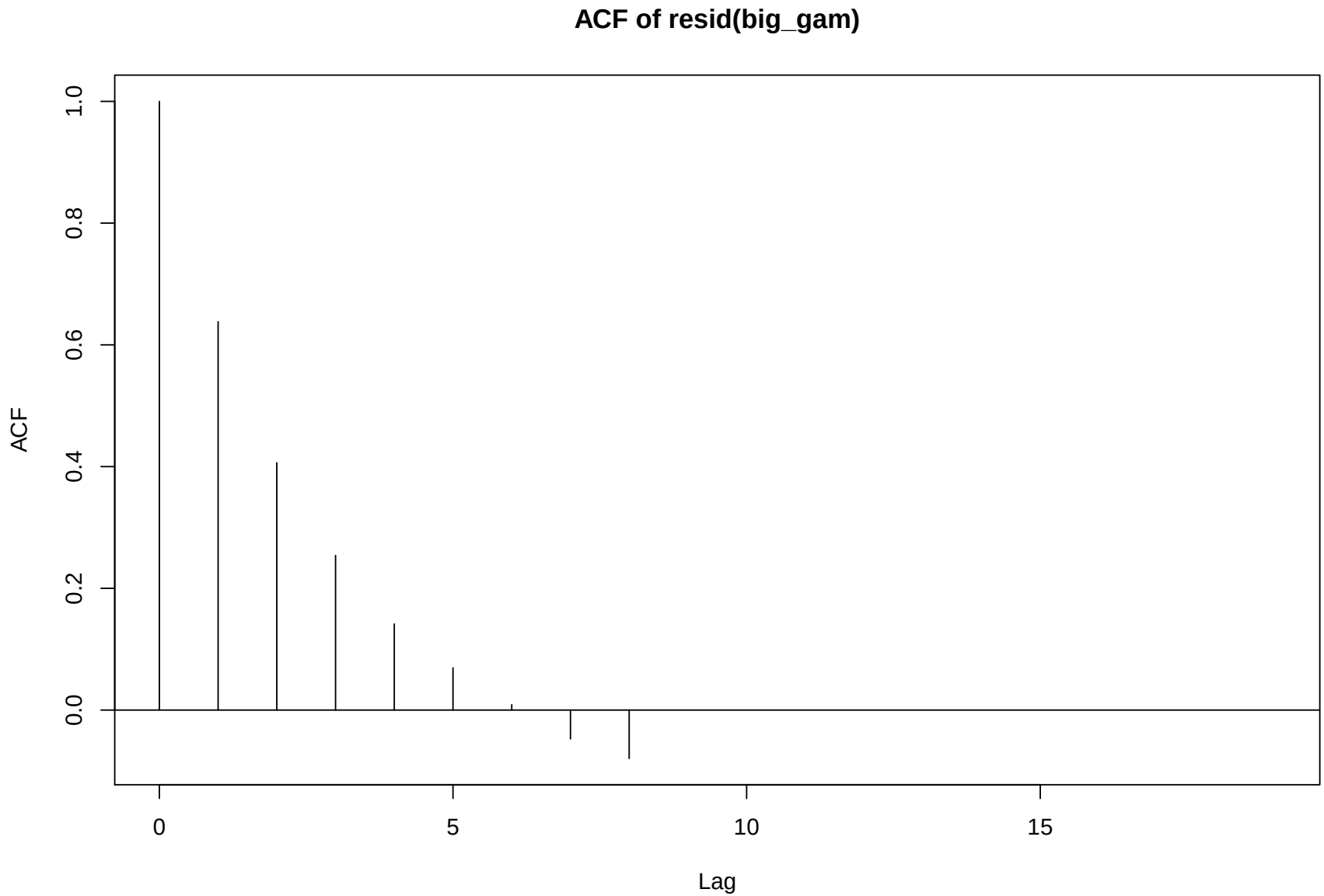
Dynamic data

```
##
## 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



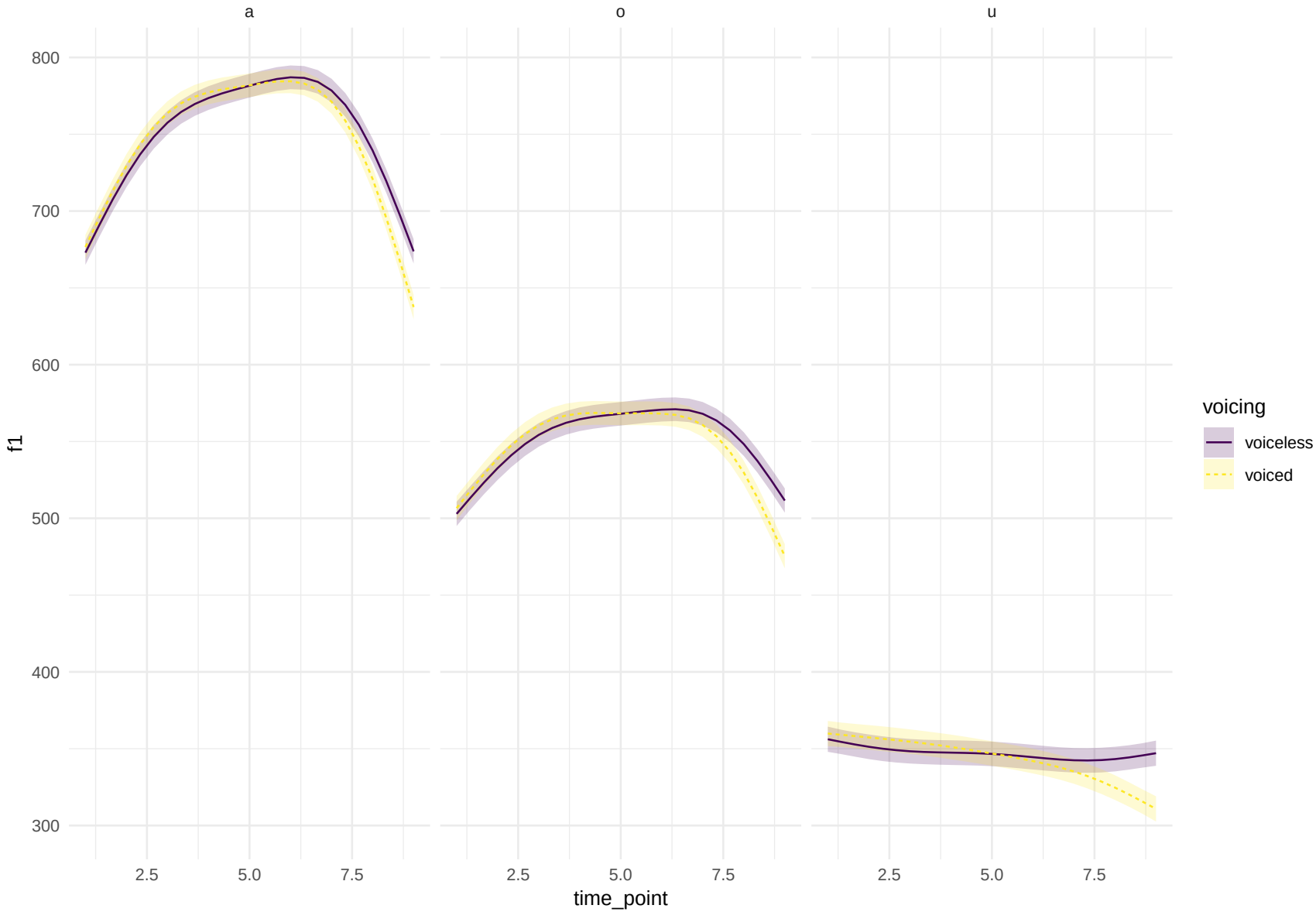
Autocorrelation



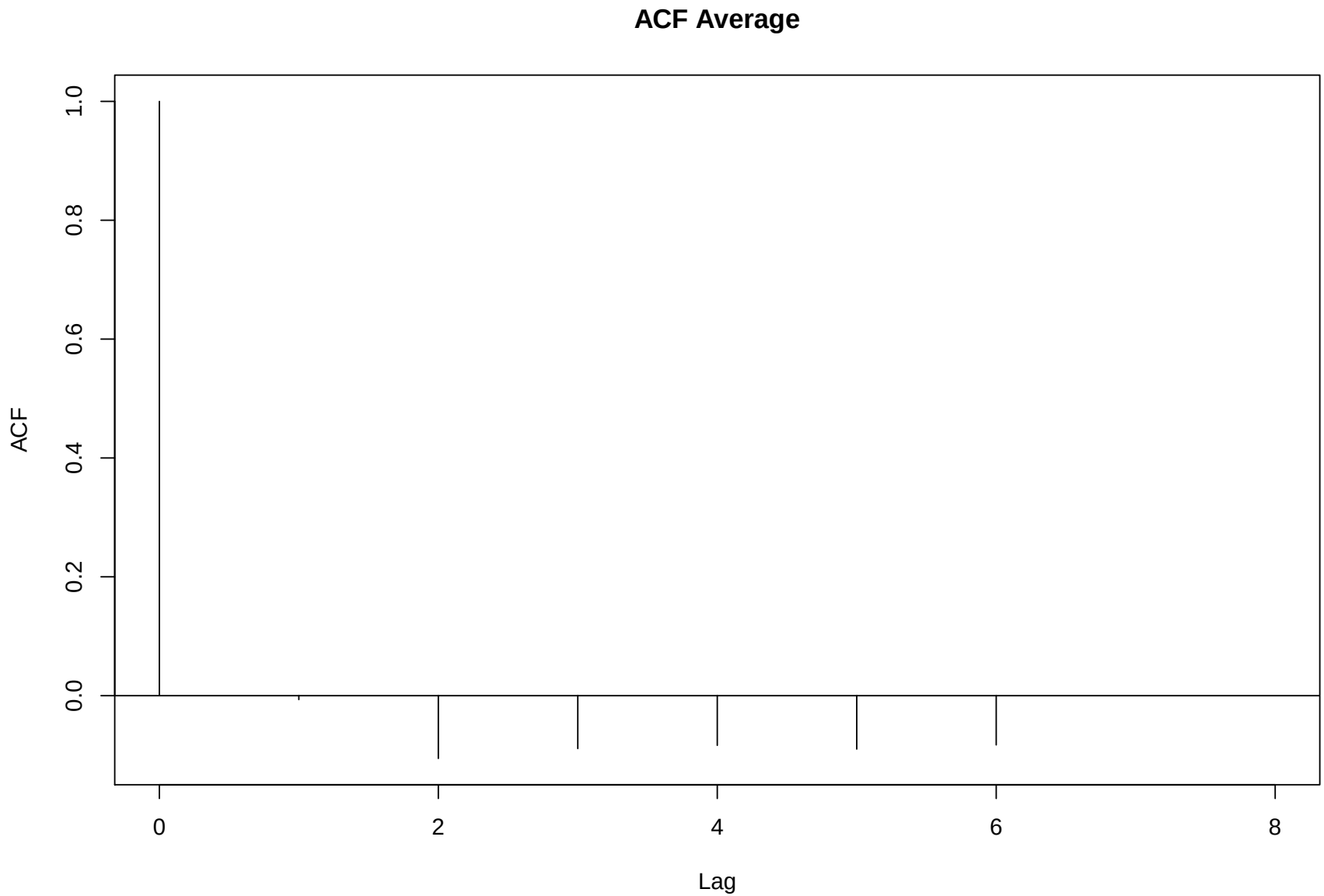
Autocorrelation

Autocorrelation

Autocorrelation



Autocorrelation



Random effects

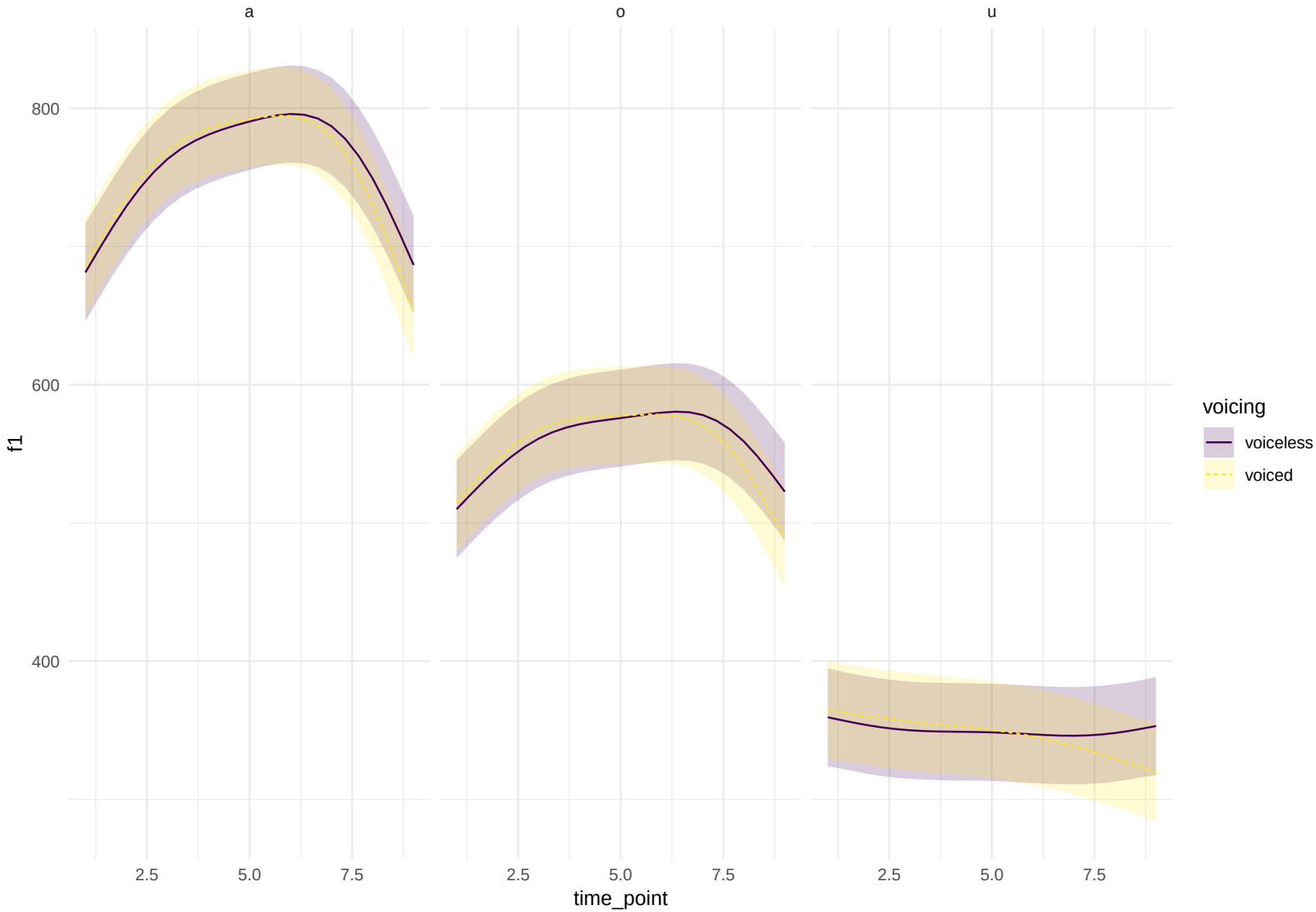
[introduce why GAMM with two ms]

Random effects

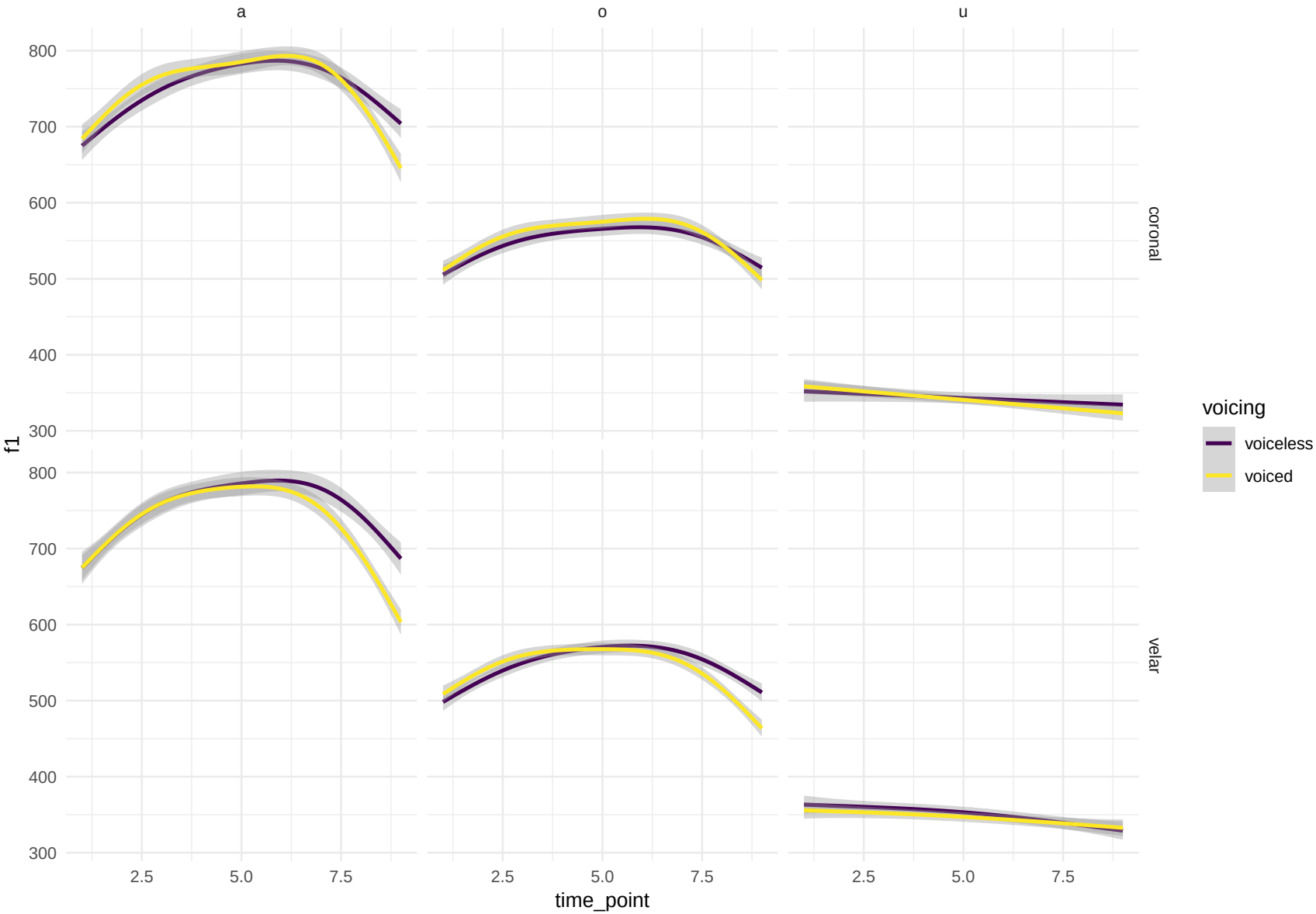
```
## Warning in gam.side(sm, X, tol = .Machine$double.eps^0.5): mo
## repeated 1-d smooths of same variable.

##
## Family: gaussian
## Link function: identity
##
## Formula:
## f1 ~ voicing + vowel + s(time_point, k = 6) + s(time_point, k
##     by = voicing) + s(time_point, k = 6, by = vowel) + s(time
##     speaker, bs = "fs", m = 1)
##
## Parametric coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
```

Random effects



Interactions



Interactions

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## f1 ~ vow_voi + s(time_point, k = 6) + s(time_point, by = vow_
##     k = 6)
##
## Parametric coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)               746.087      2.059  362.34  < 2e-16 ***
## vow_voio.voiceless       -201.537      2.906  -69.36  < 2e-16 ***
## vow_voiu.voiceless       -399.511      2.986 -133.80  < 2e-16 ***
```

Practical 2