GAM Tutorial

Tae-Jin Yoon

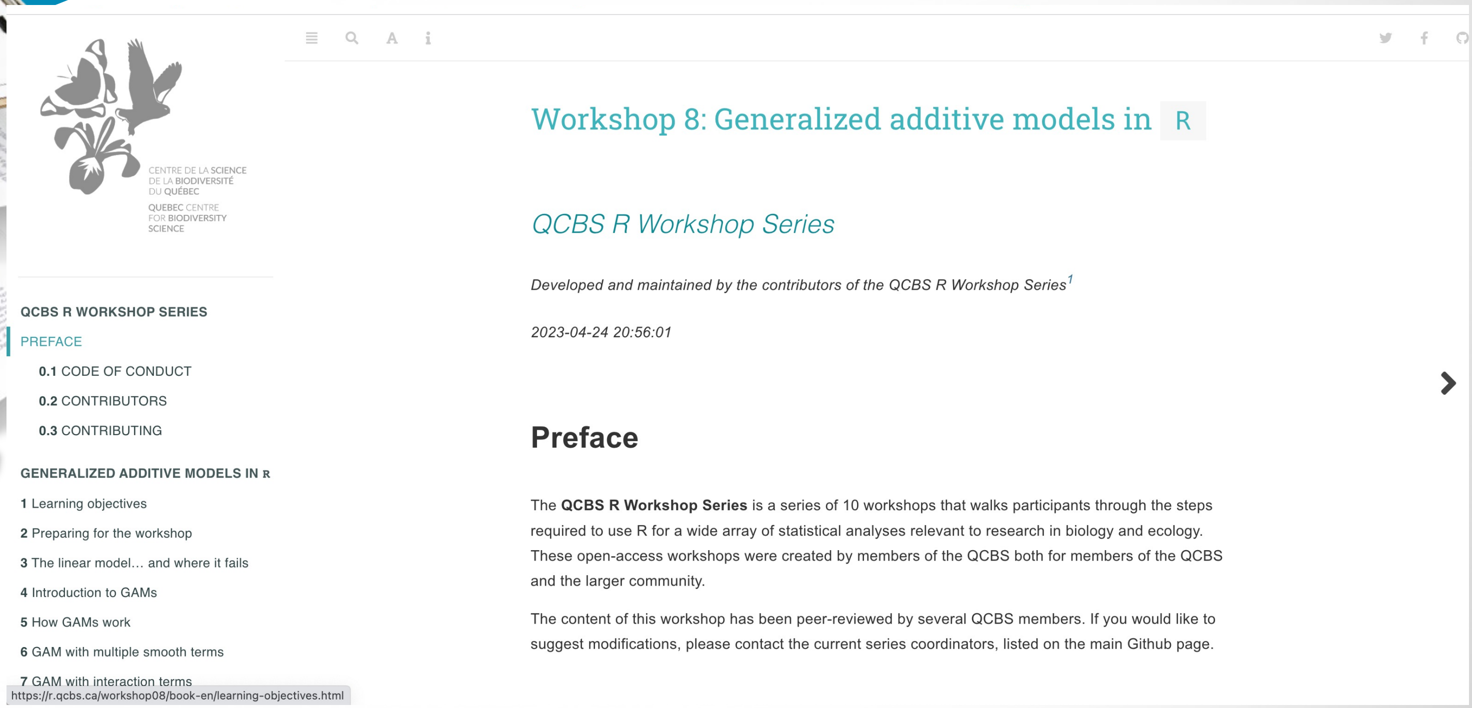
Sungshin Women's University

Workshop abstract

Measurements of human speech often show nonlinear patterns. Formant trajectories and pitch contours are well-known examples of nonlinear patterns. For example, pitch contour typically does not develop linearly over time. F0 contour over a stretch of sentence can be quite fluctuating or wiggly. Unlike a common practice of considering a pre-defined subset of measurements, such as maximum or minimum pitch values, GAMM extends the generalized linear mixed model with a large array of tools for modeling nonlinear dependencies between a response variable and one or more numeric predictors (Wood, 2015, Sóskuthy, 2017, 2021; Wieling, 2018).

In this tutorial, the Generalized Additive Mixed Model (GAMM) will be used with the help of R and its packages, especially tidyverse (Wickham, 2017), mgcv (Wood, 2015), itsadug (van Rij et al., 2015), because the statistical modeling implemented as R packages can capture the underlying dynamic patterns as well as the effects of random factors. The two packages, mgcv and itsadug, are specifically designed for the GAMM modeling and its visualization.

Source



CENTRE DE LA SCIENCE DE LA BIODIVERSITÉ DU QUÉBEC
QUEBEC CENTRE FOR BIODIVERSITY SCIENCE

https://r.qcbs.ca/workshop08/book-en/learning-objectives.html

Workshop 8: Generalized additive models in R

QCBS R Workshop Series

Developed and maintained by the contributors of the QCBS R Workshop Series[1]

2023-04-24 20:56:01

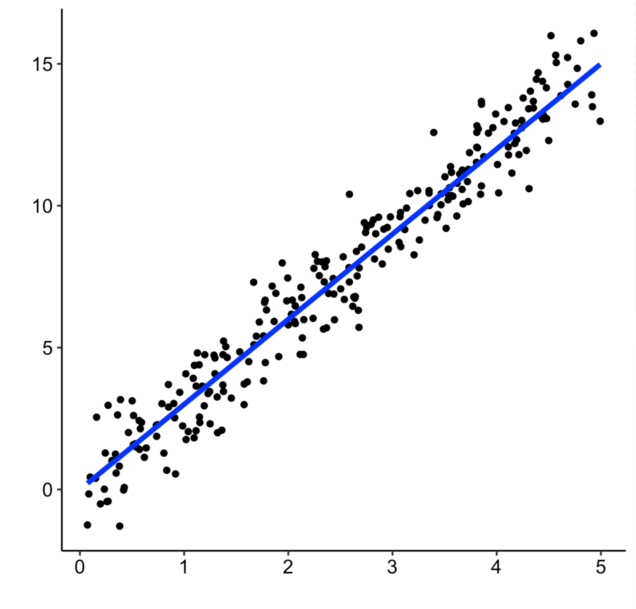
Preface

The **QCBS R Workshop Series** is a series of 10 workshops that walks participants through the steps required to use R for a wide array of statistical analyses relevant to research in biology and ecology. These open-access workshops were created by members of the QCBS both for members of the QCBS and the larger community.

The content of this workshop has been peer-reviewed by several QCBS members. If you would like to suggest modifications, please contact the current series coordinators, listed on the main Github page.

https://r.qcbs.ca/workshop08/book-en/

- GLM
  - Generalized Linear Model
- GAM
  - Generalized Additive Model
- GAMM
  - Generalized Additive Mixed Model



1. There is a linear relationship between response and predictor variables: $y_i = \beta_0 + \beta_1 \times x_i + \epsilon_i$;
2. The error is normally distributed: $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$;
3. The variance of the error is homogeneous (homoscedastic);
4. The errors are independent of each other;

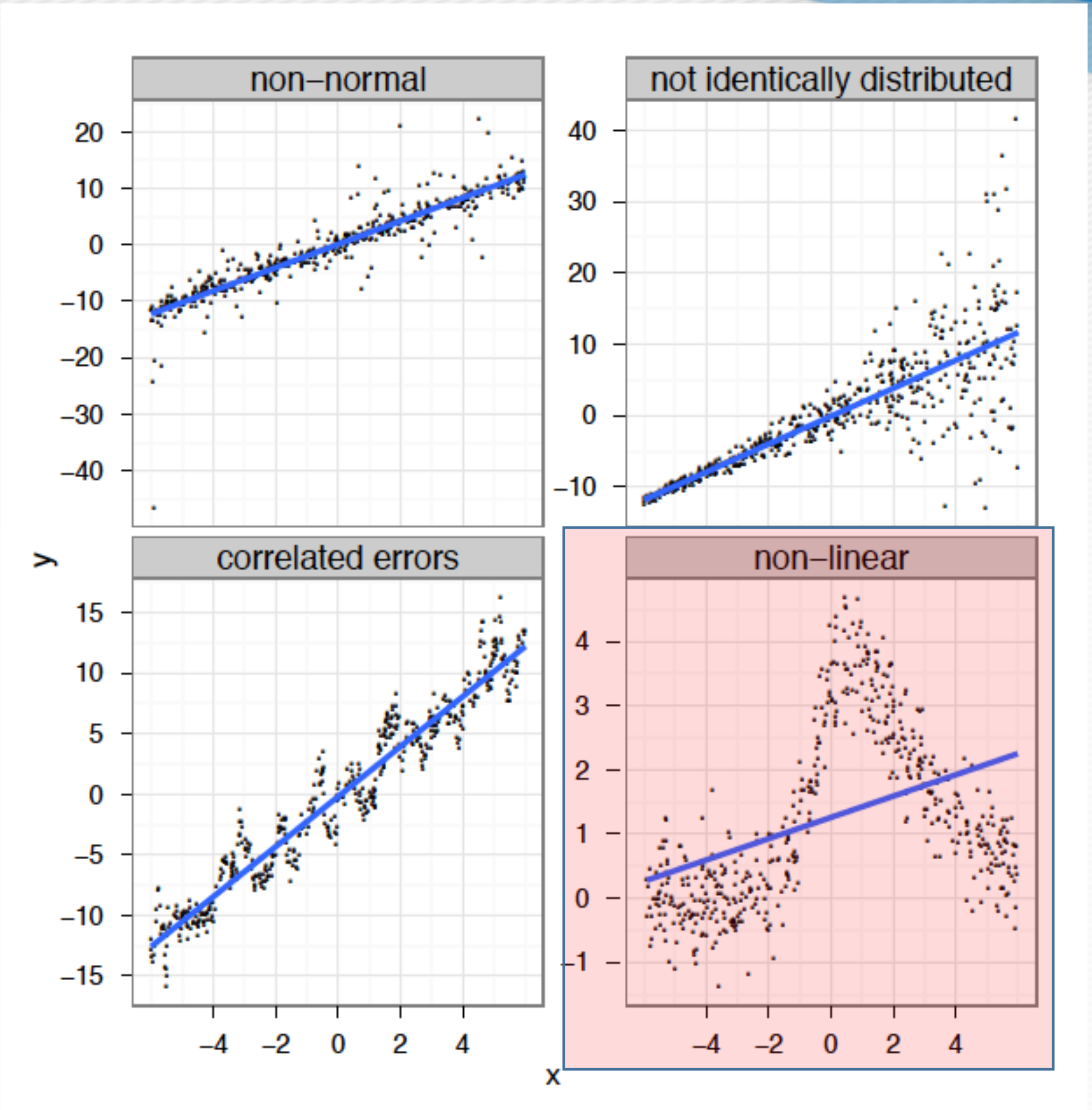
https://r.qcbs.ca/workshop08/book-en/introduction-to-generalized-additive-mixed-models-gamms.html

I. GLM vs. GAM

- An equation for a Gaussian linear model
- $Y = \beta_0 + x_1\beta_1 + \varepsilon,\ \ \varepsilon \sim N(0,\sigma^2)$

- GAM – Generalized Additive Model
- $Y = \beta_0 + f(x_1) + \varepsilon,\ \ \varepsilon \sim N(0,\sigma^2)$

- A combination of linear and smooth terms
- $Y = \beta_0 + x_1\beta_1 + f(x_2) + \varepsilon,\ \varepsilon \sim N(0,\sigma^2)$



I. GLM vs. GAM

GLM vs. GAM

- A linear model tries to fit the best straight line that passes through the data, so it does not work well for all datasets.
- In contrast, a GAM can capture complex relationships by fitting a non-linear smooth function through the data, while controlling how wiggly the smooth can get.

wig·gle | ˈwig(ə)l |

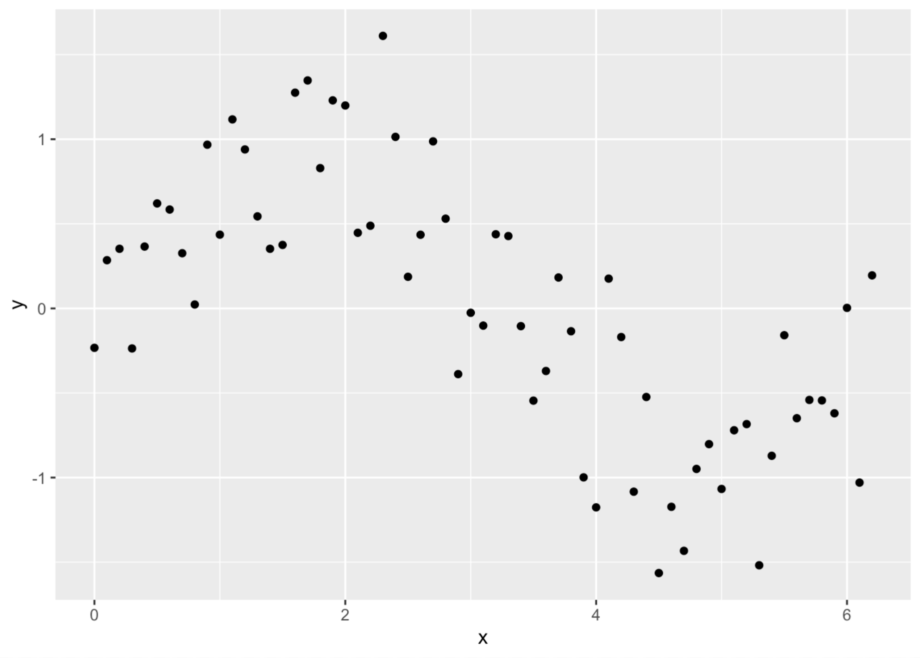
verb

move or cause to move up and down or from side to side with small rapid movements: *[with object] : Stasia wiggled her toes* | *[no object] : my tooth was wiggling around.*

A simple example

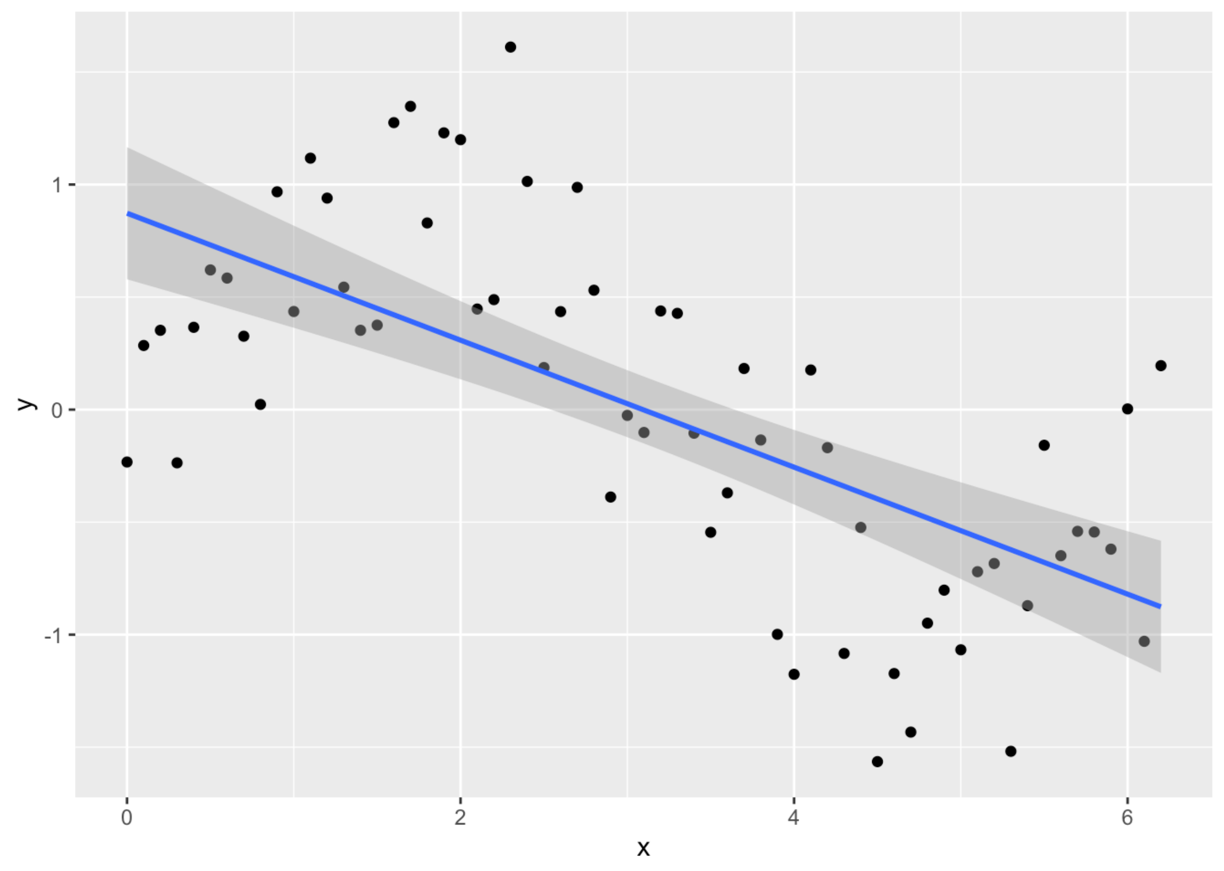
```
x <- seq(0, pi * 2, 0.1)
sin_x <- sin(x)
y <- sin_x + rnorm(n = length(x), mean = 0, sd = sd(sin_x / 2))
sample_data <- data.frame(y, x)
```

```
library(ggplot2)
ggplot(sample_data, aes(x, y)) +
  geom_point()
```



Fitting a normal linear model

```
lm_y <- lm(y ~ x, data = sample_data)

ggplot(sample_data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = lm)
```

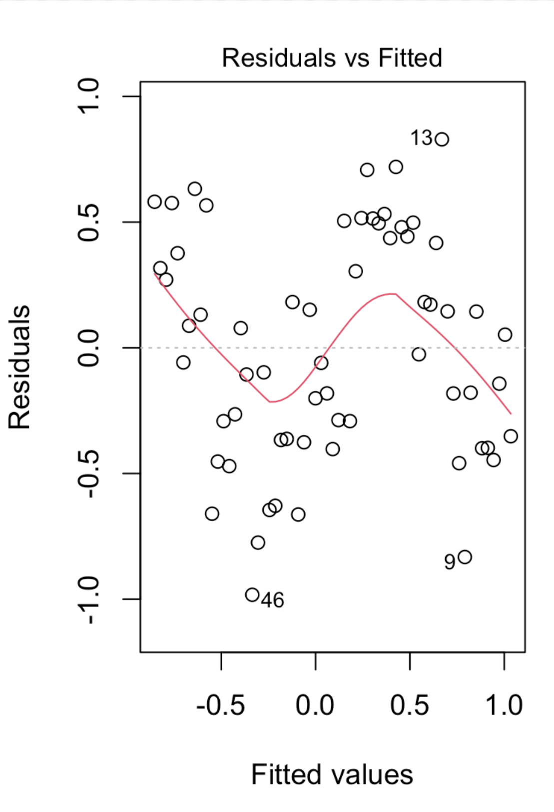


Is the model fit nicely to the data?

```
summary(lm_y)
plot(lm_y, which = 1)
```

```
> summary(lm_y)

Call:
lm(formula = y ~ x, data = sample_data)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.98240 -0.36448 -0.02587  0.42669  0.82909 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.03488    0.11204   9.236 3.40e-13 ***
x           -0.30457    0.03118  -9.769 4.32e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.45 on 61 degrees of freedom
Multiple R-squared:  0.6101,	Adjusted R-squared:  0.6037 
F-statistic: 95.44 on 1 and 61 DF,  p-value: 4.317e-14
```

The residual plot



- The residuals are **not evenly spread** across values of x, and we need to consider a better model.

II. GAM in action

```
install.packages("ggplot2")
install.packages("mgcv")
install.packages("itsadug")
library(ggplot2)
library(mgcv)
library(itsadug)
```

Mixed GAM Computation Vehicle with Automatic Smoothness Estimation

Interpreting Time Series, Autocorrelated Data Using GAMMs

GAM

- In GAM, the relationship between the response variable and the predictors is:

- $Y = \alpha + s(x_1) + s(x_2) + \ldots + \epsilon$

Smooth terms

- Note: the degree of smoothness of **s(x)**, i.e., the optimal shape, is determined automatically (using a generalized cross-validation)

GAM – Data fitting

- Before we consider a GAM, we need to load the package mgcv

Mixed GAM Computation Vehicle with Automatic Smoothness Estimation

```
df <- read.csv("gam_tutorial.csv")
head(df)
df2 <- subset(df, Gender == 2)
gam_model <- gam(VOT ~ s(F0), data = df2)
summary(gam_model)
```

Smooth terms

GAM - Summary

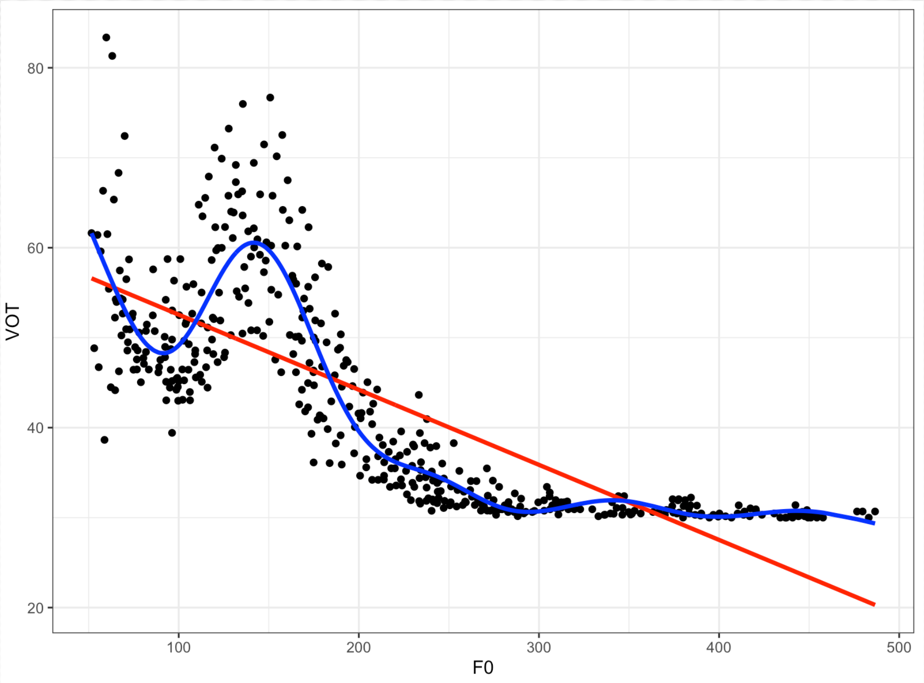
```
> summary(gam_model)

Family: gaussian
Link function: identity

Formula:
VOT ~ s(F0)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  42.8937     0.2471   173.6   <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(F0) 8.908  8.998 214.1  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =   0.81   Deviance explained = 81.4%
GCV = 28.287  Scale est. = 27.669    n = 453
```

effective degrees of freedom (EDF)
Essentially, more EDF imply more complex, wiggly splines.

When a term has an EDF value that is close to 1, it is close to being a linear term. Higher values indicate that the term’s spline is more wiggly, or in other words, highly non-linear.

GAM – Data fitting (Linear & Non-Linear)

```
df <- read.csv("gam_tutorial.csv")
head(df)
df2 <- subset(df, Gender == 2)
linear_model <- gam(VOT ~ F0, data = df2)
summary(linear_model)
data_plot <- ggplot(data = df2, aes(y = VOT, x = F0)) +
  geom_point() + geom_line(aes(y = fitted(linear_model)),
  colour = "red", linewidth = 1.2) +
  theme_bw()
data_plot

gam_model <- gam(VOT ~ s(F0), data = df2)
summary(gam_model)
data_plot <- data_plot + geom_line(aes(y = fitted(gam_model)),
               colour = "blue", size = 1.2)
data_plot
```



Test for linearity using GAM

- How do we test whether the non-linear model offers a significant improvement over the linear model?
- Use **gam()** and **AIC()** to test whether an assumption of linearity is justified.

```
> linear_model <- gam(VOT ~ F0, data = df2)
> smooth_model <- gam(VOT ~ s(F0), data = df2)
> AIC(linear_model, smooth_model)
                   df      AIC
linear_model  3.00000 3143.720
smooth_model 10.90825 2801.451
```

III. A closer look at GAM

III. A closer look at GAM

- Let's look at what GAMs are doing behind the scenes.
- A model containing one smooth function of one covariate, x_i:

- $y_i = f(x_i) + \varepsilon_i$

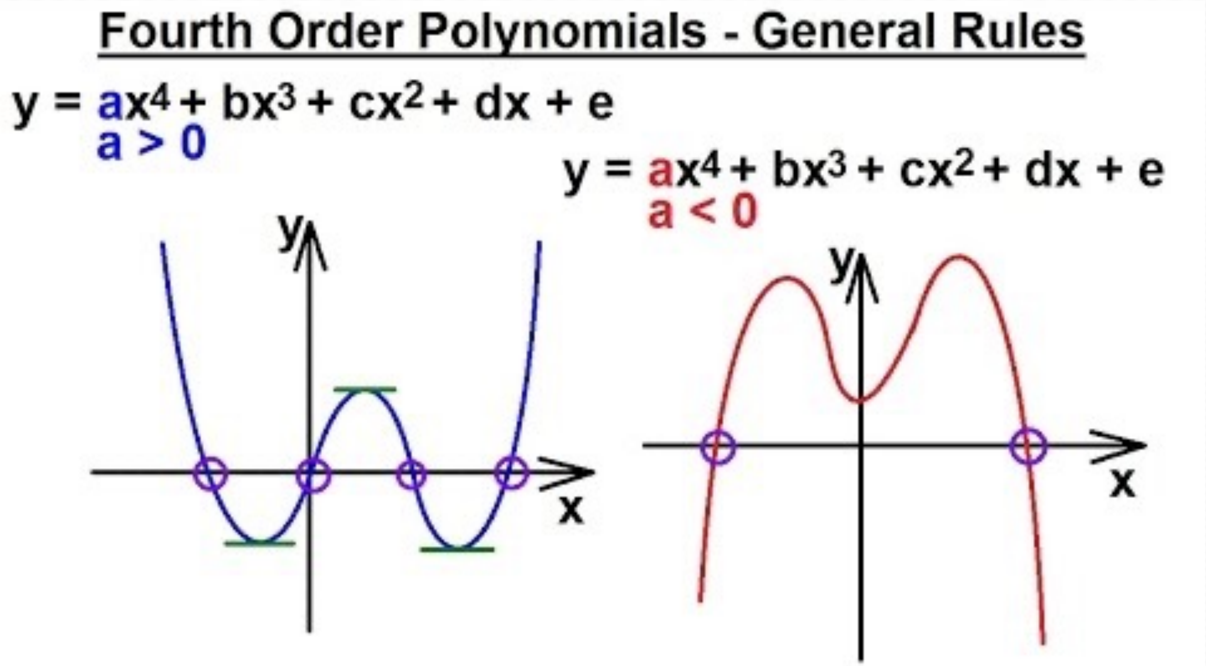
- We need to make $f(x_i)$ as a linear model
- How? - By choosing a basis $b_i(x)$

$$f(x) = \sum_{i=1}^{q} b_i(x) \times \beta_i$$

Example: a polynomial basis

- Suppose that **f** is believed to be a 4th order polynomial.
- Then a basis for this function would be:

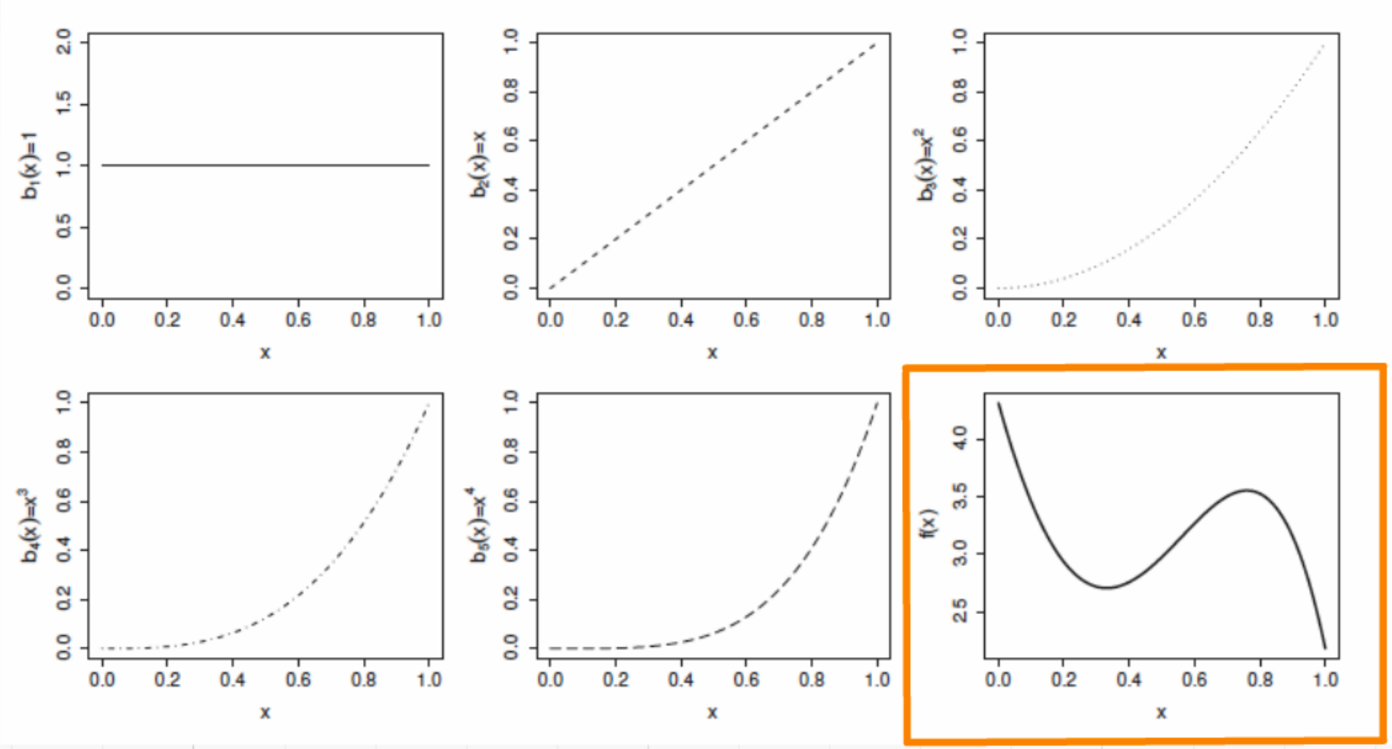
$$b_1(x) = 1, b_2(x) = x, b_3(x) = x^2, b_4(x) = x^3, b_5(x) = x^4$$



Example: a polynomial basis

- Suppose that **f** is believed to be a 4th order polynomial.

$$b_1(x) = 1, b_2(x) = x, b_3(x) = x^2, b_4(x) = x^3, b_5(x) = x^4$$



- The basis functions are each multiplied by a real valued parameter, β_i and are then summed to give the final curve f(x).

- By varying the β_i we can vary the form of f(x) to produce any polynomial function of order 4 or lower.

Example: a polynomial basis

- Suppose that **f** is believed to be a 4th order polynomial.
- Then a basis for this function would be:

$$b_1(x) = 1, b_2(x) = x, b_3(x) = x^2, b_4(x) = x^3, b_5(x) = x^4$$

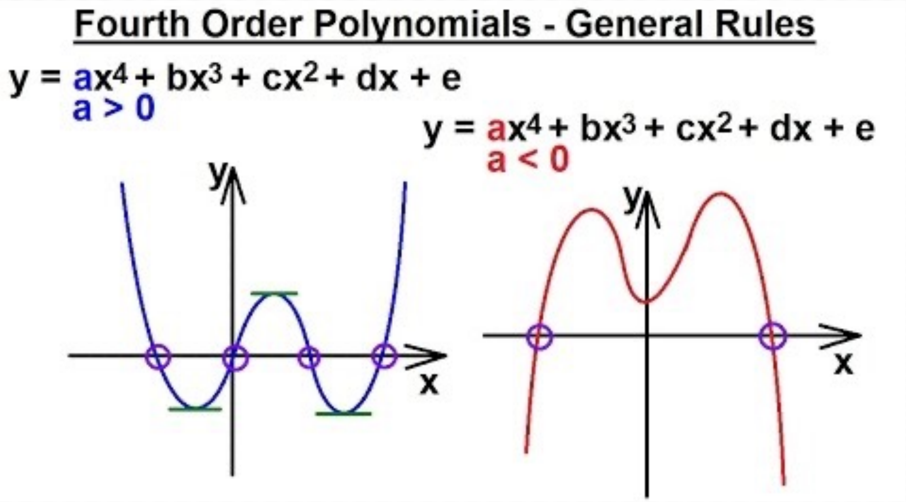
- so that f(x) becomes:

$$f(x)=\beta_1+\beta_2\, x_i + \beta_3\, x^{2i} + \beta_4\, x^3_i + \beta_5\, x^4_i$$

- The full model now becomes:

$$y_i = \beta_1 + x_i\beta_2 + x_i^2\beta_3 + x_i^3\beta_4(x) + x_i^4\beta_5 + \varepsilon_i$$

$F(x_i)$



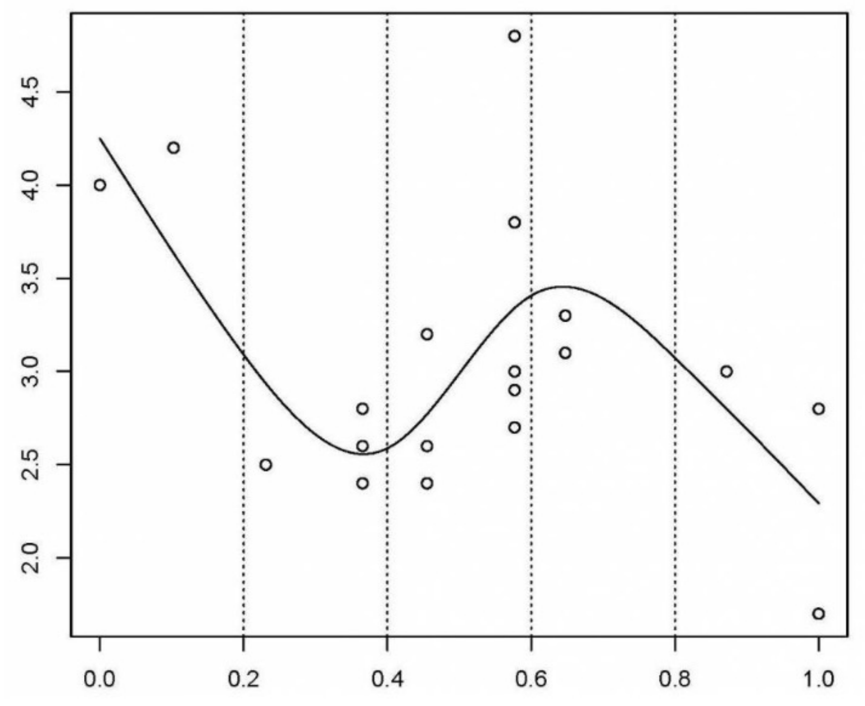
Controlling the degree of smoothing – a "wiggleness" penalty

- Fitting the model by minimizing

$$||y - XB||^2 + \lambda \int_0^1 [f''(x)]^2 dx$$

Least Squares Regression

as λ goes to ∞, the model becomes linear.



IV. Multiple smooth terms

IV. Multiple smooth terms

- How to include both smooth terms and linear terms, multiple smoothed terms and smoothed interactions?

- Simulating data by generating with **mgcv::gamSim()**

```
# ?gamSim
gam_data <- gamSim(eg = 5)
```

`?gamSim`

See the source code for exactly what is simulated in each case.

1. Gu and Wahba 4 univariate term example.
2. A smooth function of 2 variables.
3. Example with continuous by variable.
4. Example with factor by variable.
5. An additive example plus a factor variable.
6. Additive + random effect.
7. As 1 but with correlated covariates.

IV. Multiple smooth terms

```
head(gam_data)
```

```
#           y x0         x1         x2         x3
# 1  4.723147  1 0.02573032 0.70706571 0.69248543
# 2  8.886671  2 0.83272144 0.84997218 0.88974095
# 3 11.196905  3 0.66302652 0.88025265 0.08469529
# 4 10.886068  4 0.11126873 0.80087554 0.15109792
# 5 12.270534  1 0.87969756 0.37692184 0.51467778
# 6  9.020910  2 0.12441532 0.05154493 0.86526950
```

model the response y using the predictors x_0 to x_3

(1) One categorical predictor + one smoothed term

- A basic model with one smoothed term (x_1) and one categorical predictor(X_0), which has 4 levels.

```
basic_model <- gam(y ~ x0 + s(x1), data = gam_data)
basic_summary <- summary(basic_model)
```

`basic_summary$p.table`

the significance table for each linear term

```
> basic_summary$p.table
            Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 8.808226  0.3241680 27.171793 1.950601e-92
x02         2.276046  0.4586280  4.962728 1.035334e-06
x03         3.513064  0.4586547  7.659495 1.454202e-13
x04         5.914444  0.4589142 12.887908 5.726531e-32
```

`basic_summary$s.table`

the significance table for each smoothed term

```
> basic_summary$s.table
      edf Ref.df        F p-value
s(x1)   1      1 123.8262       0
```

Note on estimated degree of freedom

the estimated degrees of freedom

```
> basic_summary$s.table
      edf Ref.df        F p-value
s(x1)   1      1 123.8262       0
```

- Essentially, a larger edf value implies more complex wiggly splines
  - A value close to 1 - a linear term
  - A high value (8 or higher) – highly non-linear

(2) Adding a linear term – x0 + s(x1) + x2

- Add a second term, x2, as a linear relationship with y

```
two_term_model <- gam(y ~ x0 + s(x1) + x2, data = gam_data)
two_term_summary <- summary(two_term_model)
two_term_summary$p.table
```

Linear relationship

```
> two_term_summary$p.table
             Estimate Std. Error     t value      Pr(>|t|)
(Intercept) 11.521479  0.3833732   30.052906 1.048492e-103
x02          1.929761  0.4089605    4.718698  3.309479e-06
x03          3.431480  0.4075483    8.419812  7.215750e-16
x04          5.976612  0.4067285   14.694353  3.085318e-39
x2          -5.243617  0.4964713  -10.561773  4.132346e-23
```

```
two_term_summary$s.table
```

```
> two_term_summary$s.table
           edf   Ref.df        F p-value
s(x1) 3.646514 4.501664 34.99635       0
```

(3) Multiple smoothed term – x0 + s(x1) + s(x2)

```
two_smooth_model <- gam(y ~ x0 + s(x1) + s(x2), data =
gam_data)
two_smooth_summary <- summary(two_smooth_model)
two_smooth_summary$p.table
```

```
> two_smooth_summary$p.table
            Estimate Std. Error   t value      Pr(>|t|)
(Intercept) 8.902740  0.1941003 45.866699 3.579224e-158
x02         1.880008  0.2757636  6.817461  3.590953e-11
x03         3.511055  0.2760496 12.718929  3.473499e-31
x04         5.934435  0.2751611 21.567130  3.142058e-68
```

```
two_smooth_summary$s.table
```

```
> two_smooth_summary$s.table
           edf   Ref.df        F p-value
s(x1) 2.603565 3.232852 97.14407       0
s(x2) 8.098912 8.780162 81.51290       0
```

Comparison of models – AIC()

- Perform an ANOVA to test if the smoothed term is necessary

```
AIC(basic_model, two_term_model, two_smooth_model)
```

```
> AIC(basic_model, two_term_model, two_smooth_model)
                        df      AIC
basic_model       6.000000 2082.866
two_term_model    9.646514 1986.663
two_smooth_model 15.702477 1677.076
```

The best fit model is the model with both smooth terms for x_1 and x_2.

V. GAM with interaction terms

V. GAM with interaction terms

- There are two ways to include interactions between variables:
- For two smoothed variables: s(x1, x2)
- For one smoothed variable and one linear variable (either factor or continuous): use the by argument s(x1, by = x2)

Interaction (1): smoothed and factor variables

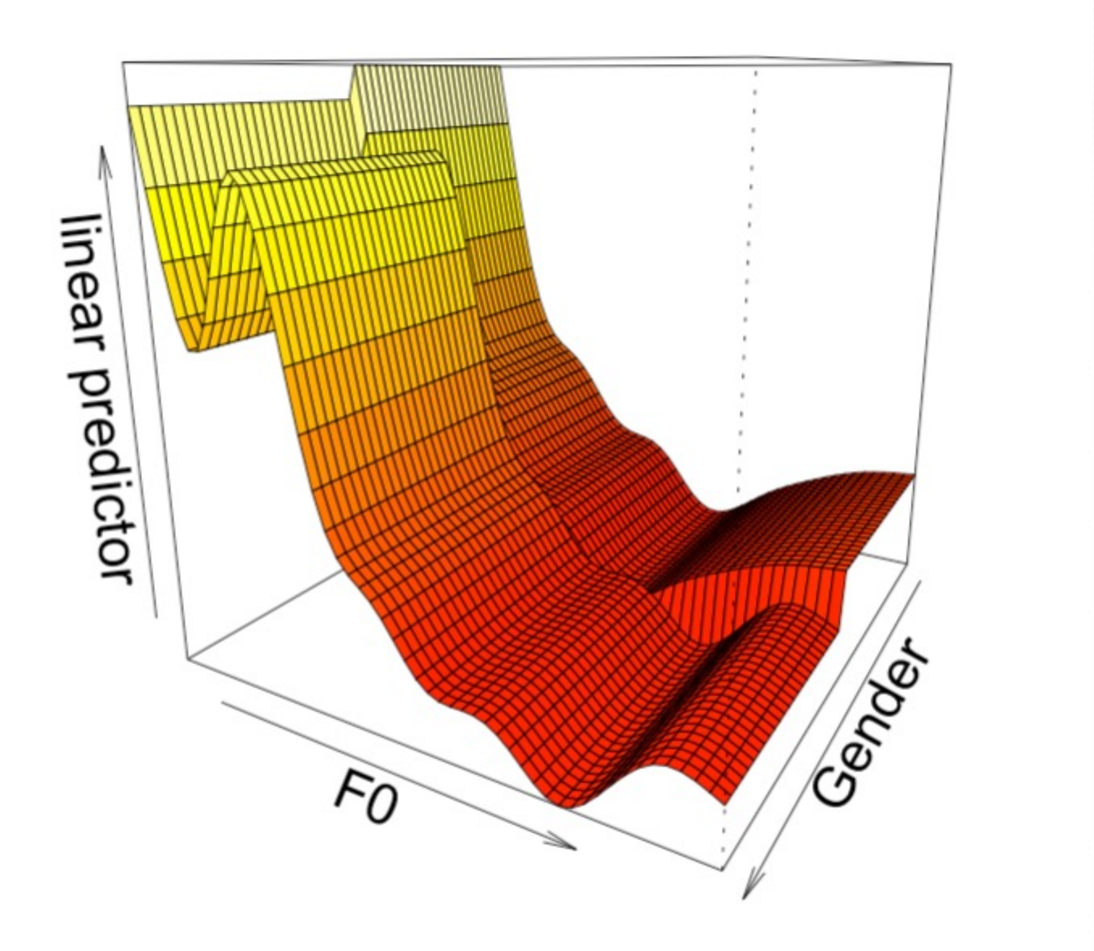
```
df$Gender <- as.factor(df$Gender)
factor_interact <- gam(VOT ~ Gender +
                           s(F0, by=Gender) +
                           s(VDUR),
                           data = df2, method = "REML")
summary(factor_interact)$s.table
```

```
> summary(factor_interact)$s.table
                   edf   Ref.df          F p-value
s(F0):Gender 8.770843 8.975946 181.999211       0
s(VDUR)      7.500649 8.464173   8.853745       0
```

Interaction (1): smoothed and factor variables

- We can also visualize our model in 3D using `vis.gam()`

```
vis.gam(factor_interact, theta=120, n.grid=50, lwd=.4)
```



Interaction (2): two smoothed variables

- The interaction between two smoothed terms

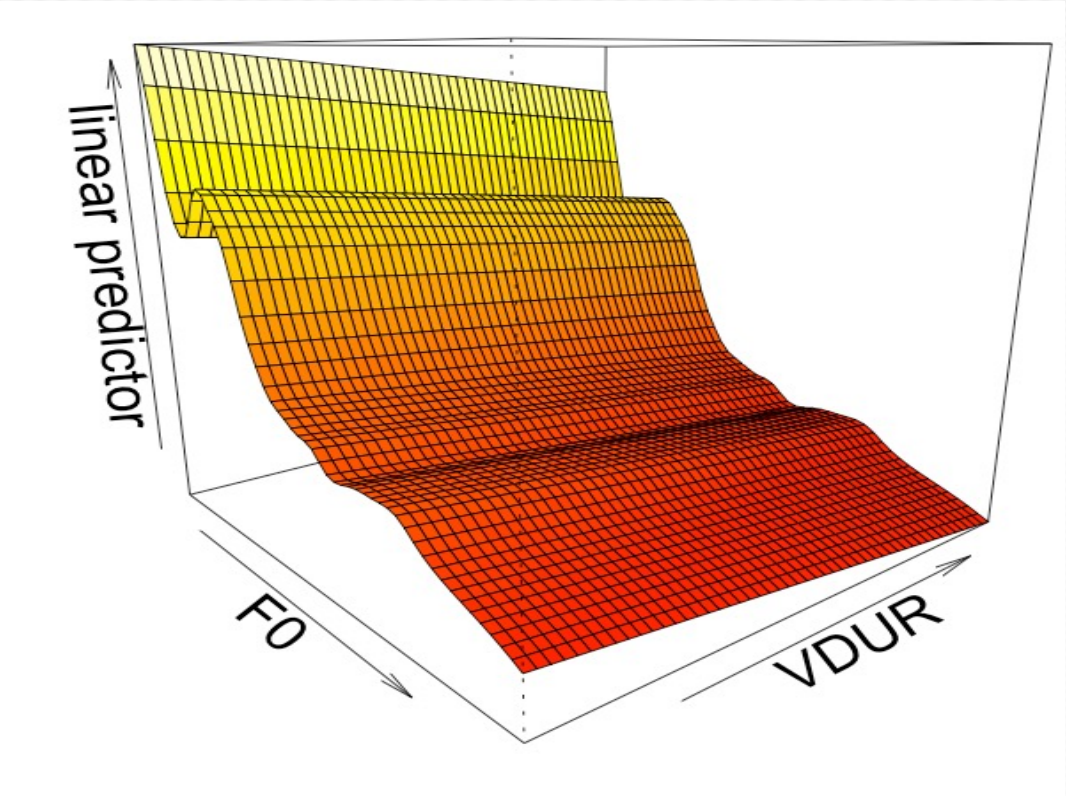
```
smooth_interact <- gam(VOT ~ Gender + s(F0, VDUR),  data = df, method =
"REML")
summary(smooth_interact)$s.table
```

```
> summary(smooth_interact)$s.table
                 edf   Ref.df        F p-value
s(F0,VDUR) 17.02099 21.15907 80.29506       0
```

Smooth term

Visualization - two smoothed variables

```
vis.gam(smooth_interact, view = c("F0", "VDUR"), theta = 50, n.grid = 50,
lwd = .4)
```



Two smooth model vs. smooth interact

```
> AIC(two_smooth_model, smooth_interact)
                       df      AIC
two_smooth_model 19.24943 2735.954
smooth_interact  20.33436 5203.671
```

III. Quick Intro. to GAMM

GAMM

- Generalized additive mixed effect models (GAMMs)
  - a type of statistical model that combines the flexibility of generalized additive models (GAMs) with the ability to account for random effects in mixed-effect models.

a vector of random effects coefficients

$$Y_i = f_1(X_{1,i}) + f_2(X_{2,i}) + \ldots + f_p(X_{p,i}) + Z_i b + \epsilon_i$$

the response variable for the ith observation

the values of the p predictor variables for that observation

random effects

the relationships between each predictor variable and the response variable.

a matrix that specifies the random effects design for the ith observation

Dealing with non-independence

- When observations are not independent, GAMs can be used to incorporate random effects that model independence between observations at the same site.
  - Random effects that model independence among observations from the same site.
- **`bs`** specifies the type of underlying base function
  - **`bs = "re"`** for random intercepts and linear random slopes
  - **`bs = "fs"`** for random smooths

Smoothing term or smoothing functions

A correlation structure to model autocorrelation residuals will not be discussed.

Random effects

Fac → factor coding for the random effect
x0 → continuous fixed effect

- **Random intercepts**
  - Adjust the height of other terms with a constant value
  - **`s(fac, bs="re")`**
- **Random slopes**
  - Adjust the slope of the trend of a numeric predictor
  - **`s(fac, x0, bs="re")`**
- **Random smooths**
  - Adjust the trend of a numeric predictor in a nonlinear way
  - **`s(fac, x0, bs="fs", m=1)`**
    - The argument m=1 sets a heavier penalty for the smooth moving away from 0, causing shrinkage to the mean.

IV. GAMM in action

```r
library(mgcv)
gam_data2 <- gamSim(eg=6)
```

gam_data2

```
##           y        x0         x1        x2        x3         f        f0
## 1  8.411980 0.7369989 0.30941786 0.5079422 0.5440523  9.066472 1.4707805
## 2  9.208443 0.1083441 0.34284917 0.6295774 0.3747799 11.956612 0.6676773
## 3 18.986368 0.9648756 0.24928256 0.4615991 0.3388980 13.867240 0.2202455
## 4 15.279170 0.8198440 0.03304455 0.5583043 0.6934456 17.072441 1.0724811
## 5 16.254626 0.3668214 0.48894089 0.2308583 0.1565436 16.404978 1.8274861
## 6 16.580965 0.8007808 0.67568426 0.6680502 0.5543013 14.247673 1.1715978
##          f1       f2 f3 fac
## 1 1.856765 2.738927  0   1
## 2 1.985158 3.303777  0   2
## 3 1.646357 3.000637  0   3
## 4 1.068322 2.931638  0   4
## 5 2.658818 8.918673  0   1
## 6 3.862708 3.213367  0   2
```

```r
?gamSim
```

See the source code for exactly what is simulated in each case.

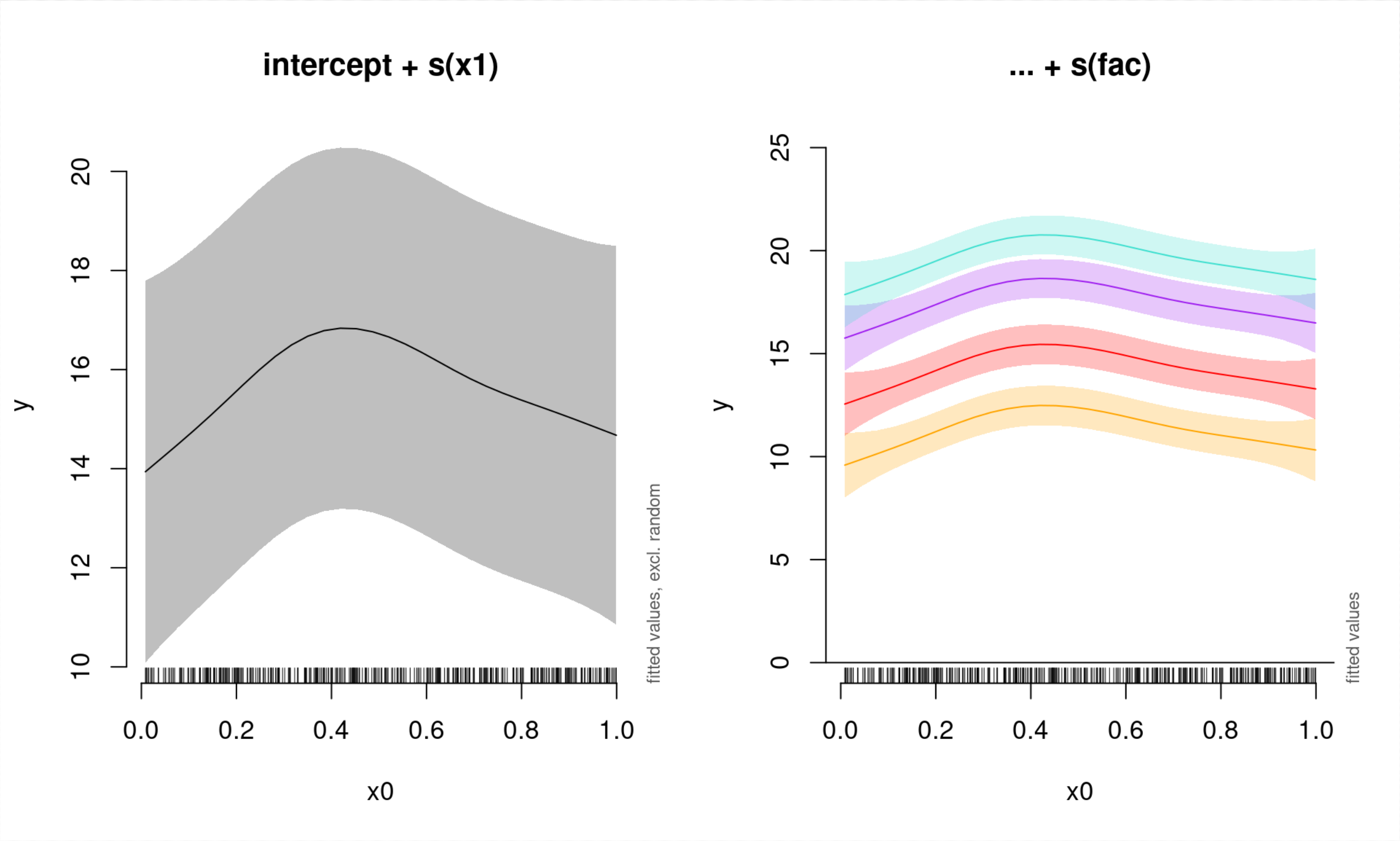
1. Gu and Wahba 4 univariate term example.
2. A smooth function of 2 variables.
3. Example with continuous by variable.
4. Example with factor by variable.
5. An additive example plus a factor variable.
6. Additive + random effect.
7. As 1 but with correlated covariates.

summary(gam_data2)

```
> summary(gam_data2)
       y               x0                x1                 x2          
 Min.   : 3.061   Min.   :0.01308   Min.   :0.001837   Min.   :0.001315  
 1st Qu.:12.032   1st Qu.:0.25813   1st Qu.:0.282795   1st Qu.:0.229992  
 Median :15.746   Median :0.47679   Median :0.511054   Median :0.455470  
 Mean   :15.409   Mean   :0.49506   Mean   :0.512017   Mean   :0.484681  
 3rd Qu.:18.985   3rd Qu.:0.72993   3rd Qu.:0.751293   3rd Qu.:0.743006  
 Max.   :29.552   Max.   :0.99608   Max.   :0.999455   Max.   :0.999931  
       x3                 f                f0                 f1       
 Min.   :0.001642   Min.   : 4.877   Min.   :0.02465   Min.   :1.004  
 1st Qu.:0.207623   1st Qu.:12.203   1st Qu.:0.85471   1st Qu.:1.760  
 Median :0.469656   Median :15.508   Median :1.47856   Median :2.779  
 Mean   :0.486131   Mean   :15.491   Mean   :1.31914   Mean   :3.261  
 3rd Qu.:0.759675   3rd Qu.:18.771   3rd Qu.:1.85990   3rd Qu.:4.493  
 Max.   :0.996272   Max.   :26.705   Max.   :2.00000   Max.   :7.381  
       f2               f3     fac    
 Min.   :0.0000   Min.   :0   1:100  
 1st Qu.:0.7766   1st Qu.:0   2:100  
 Median :2.9121   Median :0   3:100  
 Mean   :3.4114   Mean   :0   4:100  
 3rd Qu.:5.5883   3rd Qu.:0          
 Max.   :8.9186   Max.   :0          
```

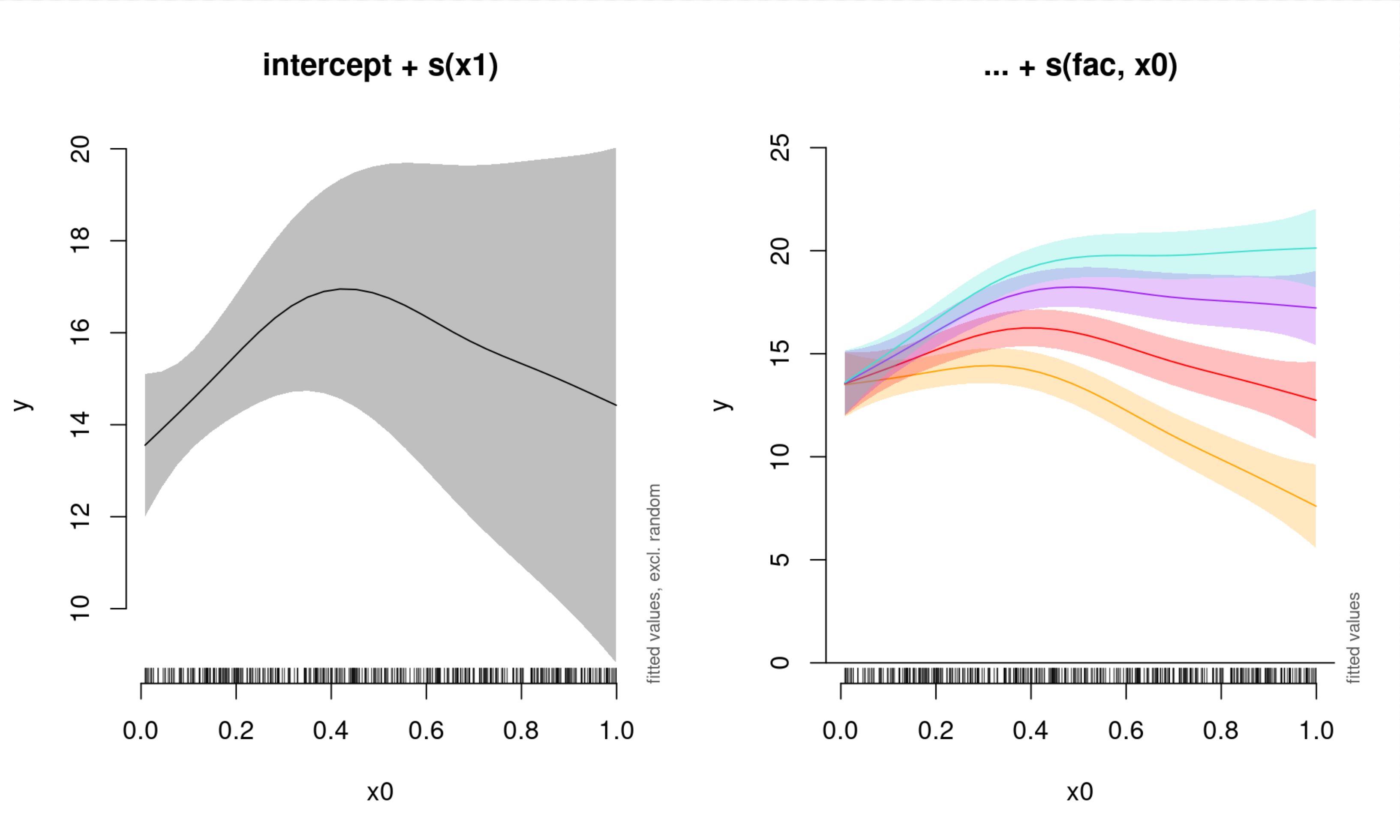
GAMM with a random intercept

gamm_intercept <- gam(y ~ s(x0) + s(fac, bs = "re"), data = gam_data2, method = "REML")



GAMM with a random slope

```
gamm_slope <- gam(y ~ s(x0) + s(x0, fac, bs = "re"), data = gam_data2, method = "REML")
```

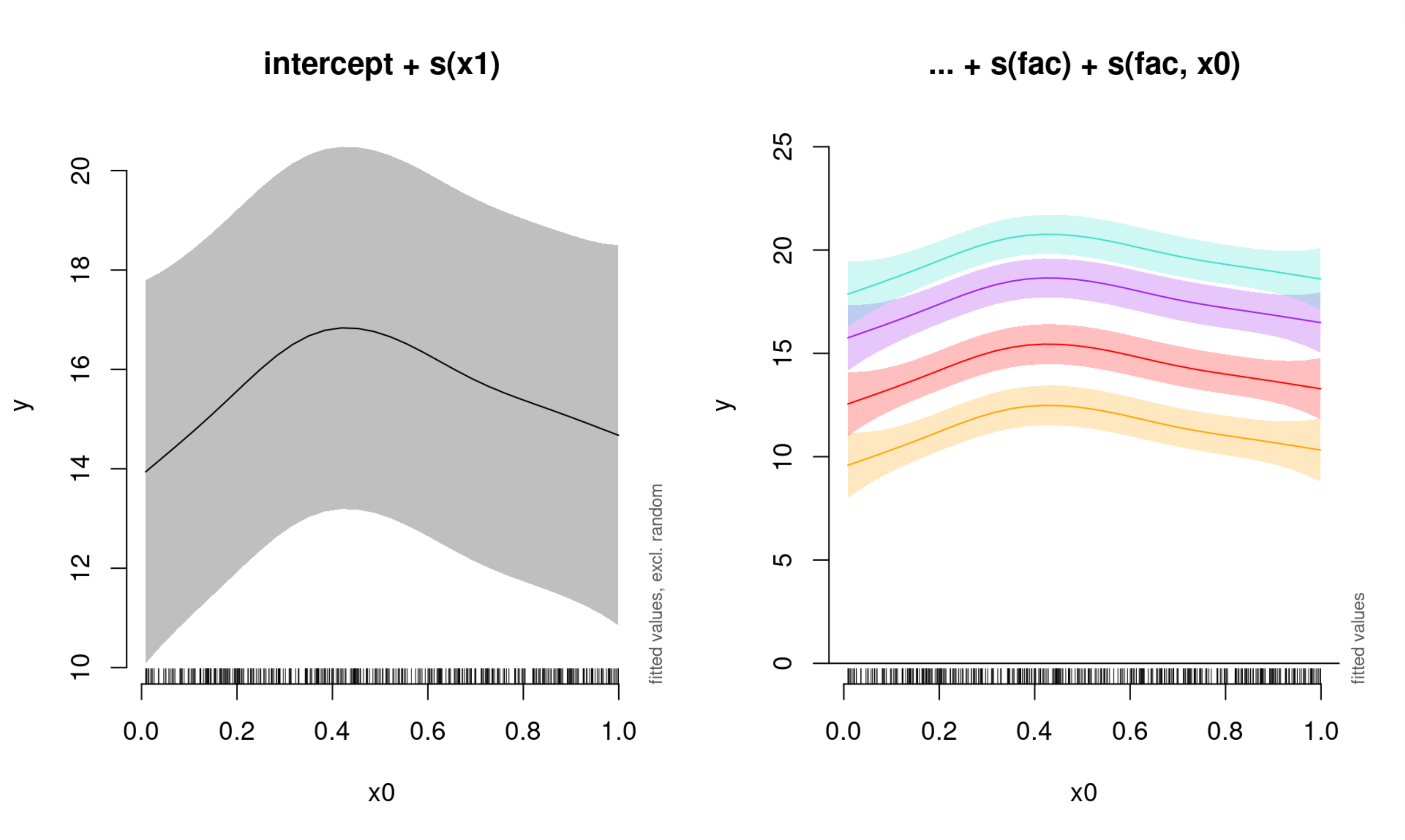


GAMM with a random intercept and slope

```
gamm_int_slope <- gam(y ~ s(x0) + s(fac, bs = "re") + s(fac, x0, bs = "re"), data = gam_data2,
                      method = "REML")
```

random intercept

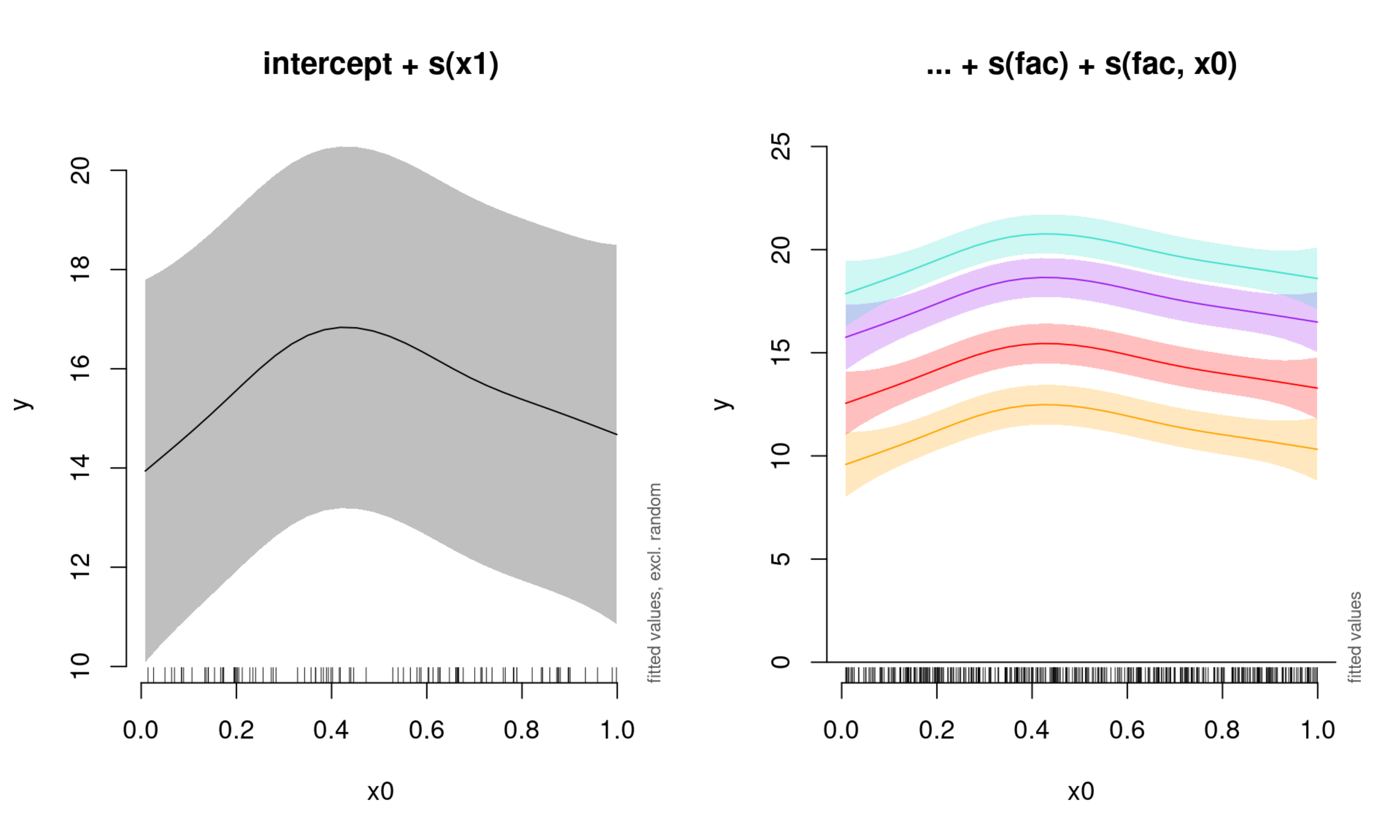
random slope



GAMM with a random smooth

```
gamm_smooth <- gam(y ~ s(x0) + s(x0, fac, bs = "fs", m = 1),
                   data = gam_data2, method = "REML")
```

random smooth



GAMM model comparison

AIC(gamm_intercept, gamm_slope, gamm_int_slope, gamm_smooth)

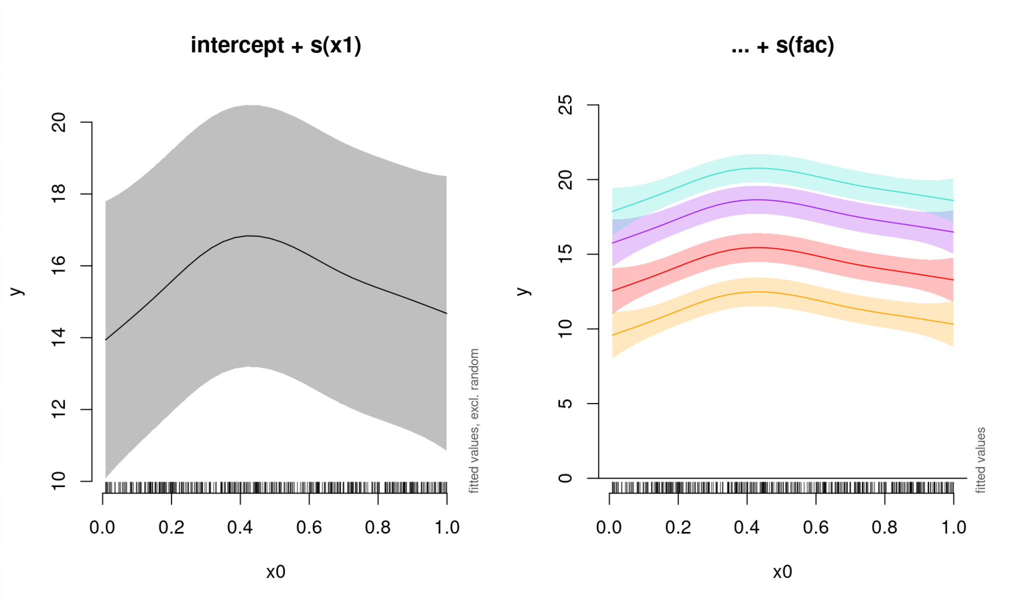
```
##                      df      AIC
## gamm_intercept 8.804002 2229.206
## gamm_slope     8.786560 2290.520
## gamm_int_slope 8.806546 2229.210
## gamm_smooth    8.810264 2229.216
```

Which is the best model among these?

→ A GAMM with a random effect on the intercept

Visualization: GAMM with a random intercept

```
par(mfrow = c(1, 2), cex = 1.1)
# Plot the summed effect of x0 (without random effects)
plot_smooth(gamm_intercept, view = "x0", rm.ranef = TRUE, main = "intercept + s(x0)")


# Plot each level of the random effect
plot_smooth(gamm_intercept, view = "x0", rm.ranef = FALSE, cond = list(fac = "1"), main = "... + s(fac)", col = "orange", ylim =
c(0, 25))

plot_smooth(gamm_intercept, view = "x0", rm.ranef = FALSE, cond = list(fac = "2"), add = TRUE, col = "red")
plot_smooth(gamm_intercept, view = "x0", rm.ranef = FALSE, cond = list(fac = "3"), add = TRUE, col = "purple")
plot_smooth(gamm_intercept, view = "x0", rm.ranef = FALSE, cond = list(fac = "4"), add = TRUE, col = "turquoise")
```



References

- Wood, S. N. 2006. *Generalized Additive Models: An Introduction with r*. 2nd ed. Chapman; Hall/CRC.
- Wood, Simon. 2021. “mgcv: Mixed GAM Computation Vehicle with Automatic Smoothness Estimation.”
- Rij, Jacolien van. 2015. “Overview GAMM Analysis of Time Series Data.” https://jacolienvanrij.com/Tutorials/GAMM.html.
- Ross, Noam. 2019. “Generalized Additive Models in r: A Free Interactive Course.” *Generalized Additive Models in R*. https://noamross.github.io/gams-in-r-course/.
- Simpson, Gavin. 2022. *From the Bottom of the Heap: The Musings of a Geographer*. https://fromthebottomoftheheap.net/.

Resources

- https://environmentalcomputing.net/statistics/gams/
- https://fromthebottomoftheheap.net/blog/
- https://www.youtube.com/watch?v=q4_t8jXcQgc
- https://noamross.github.io/gams-in-r-course/
- http://edinbr.org/edinbr/2017/10/10/october-meeting.html