

Moving Beyond Linearity

Generalized Additive Models (GAM)

Andreas Scharmüller

AG Landscape Ecology

2020-11-09 (updated: 2021-03-05)

Linear Regression

Linear Model (LM)

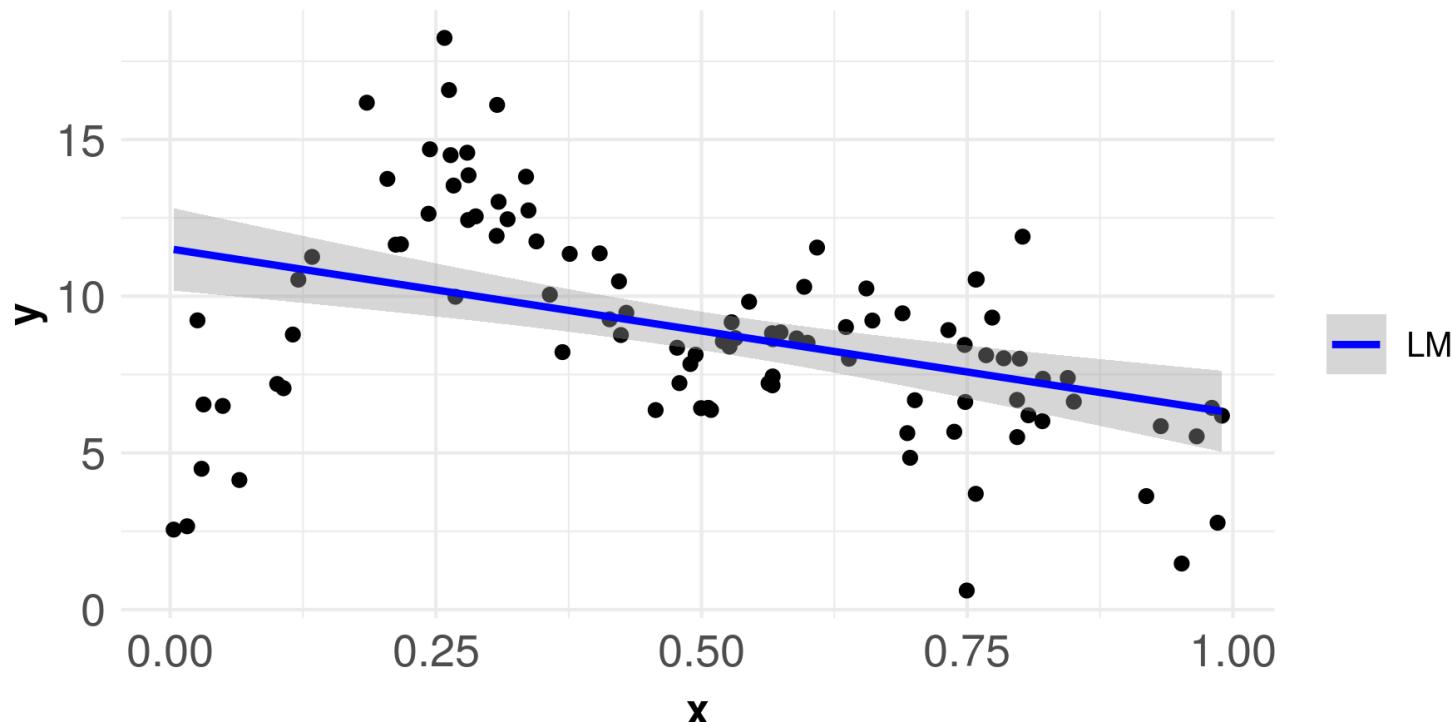
- Easy to interpret
- Confined to linear relationships
- Normally distributed responses

$$y_i = \beta_0 + \beta_1 x_{1i} + \epsilon_i, \epsilon \sim N(0, \sigma^2)$$

Linear Model (LM)

```
lm(y ~ x,  
  data = data)
```

R2: 0.17 AIC: 513.9895637



Generalized Linear Models (GLM)

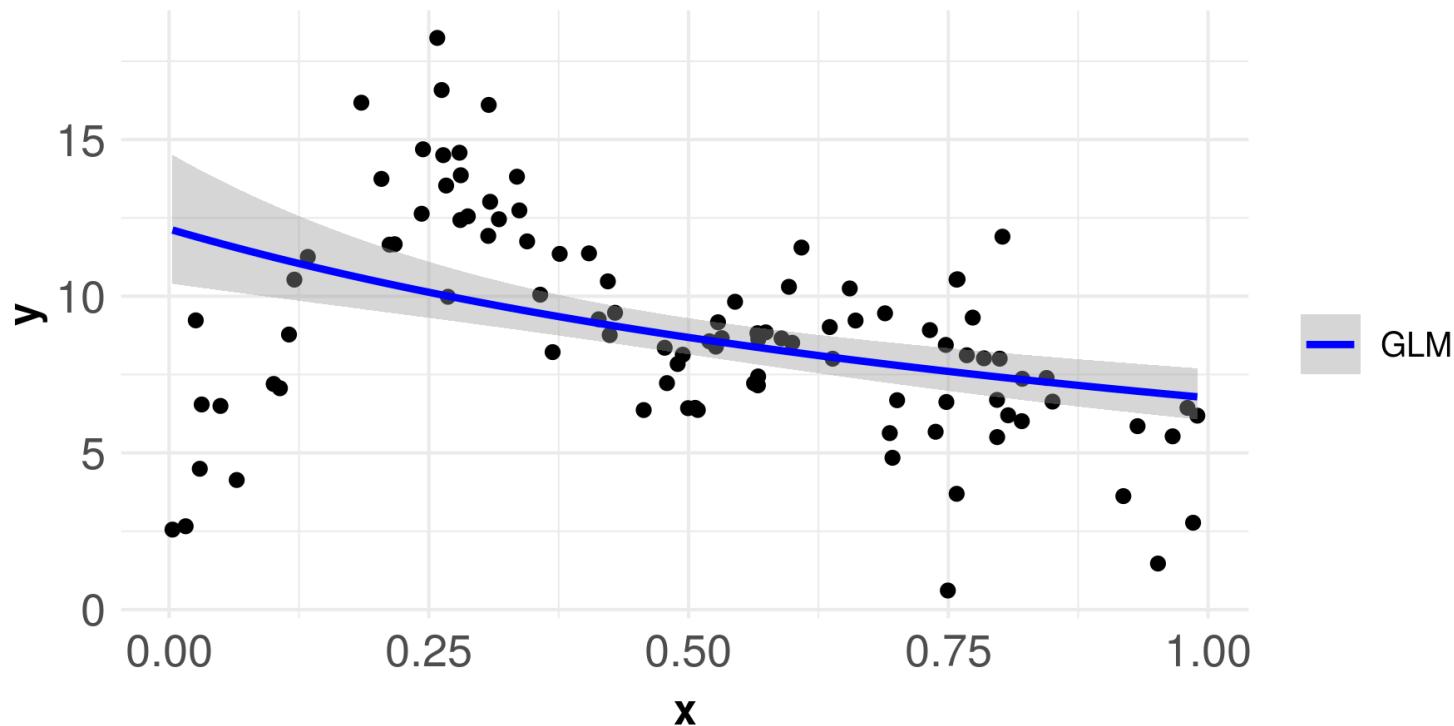
- Additional distributions (Poisson, Gamma, Binomial, etc.)

$$\mathbb{E}(y_i) = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \epsilon_i$$

Generalized Linear Models (GLM)

```
glm(y ~ x,  
    data = data,  
    family = 'Gamma')
```

R2: 0.13 AIC: 531



Polynomial Regression

- Extra predictors, obtained by raising the original predictors to a power
- Specific patterns, not very flexible
 - Global not Local
- Might lead to poor residuals, predictions, extrapolations
 - Especially at the boundaries

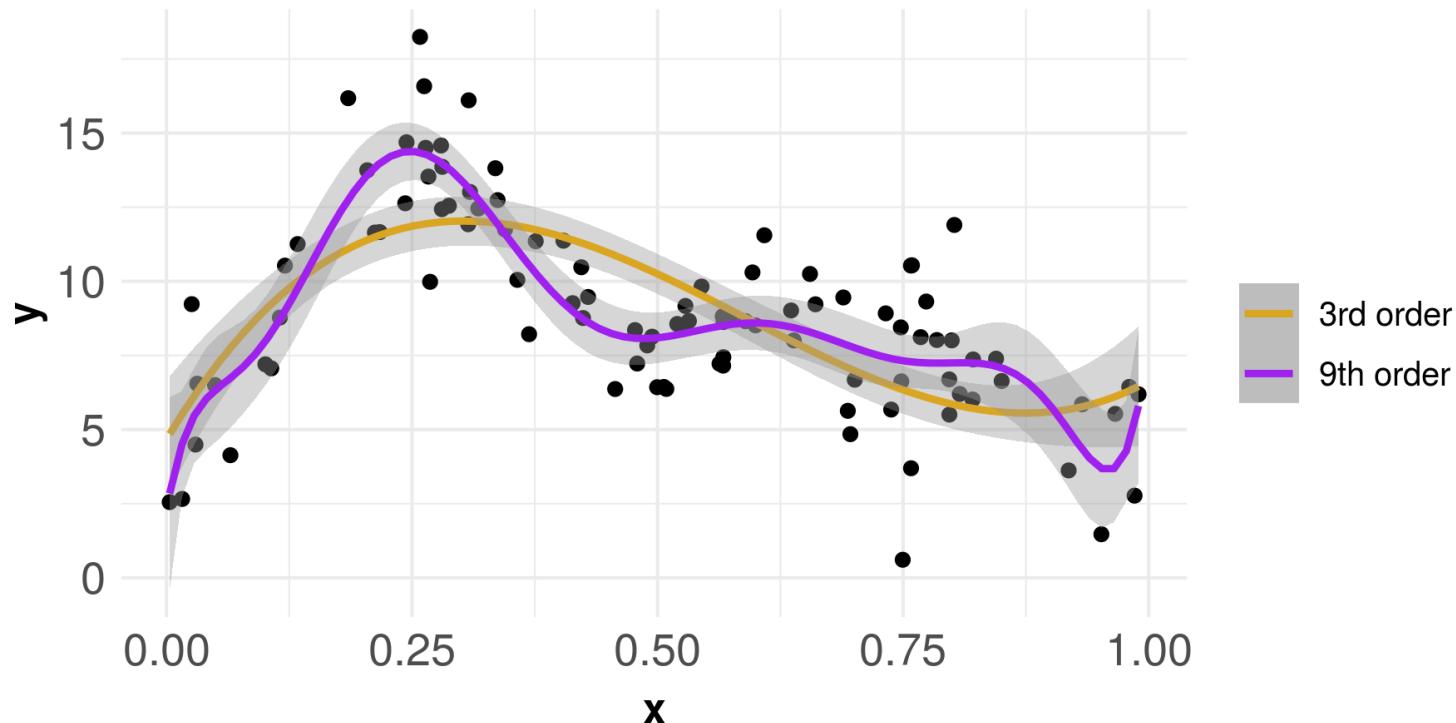
$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{1i}^2 + \beta_3 x_{1i}^3 + \epsilon_i$$

Polynomial Regression

```
lm(y ~ poly(x, 3),  
  data = data)
```

3rd order | R2: 0.5 AIC: 467

9th order | R2: 0.71 AIC: 425



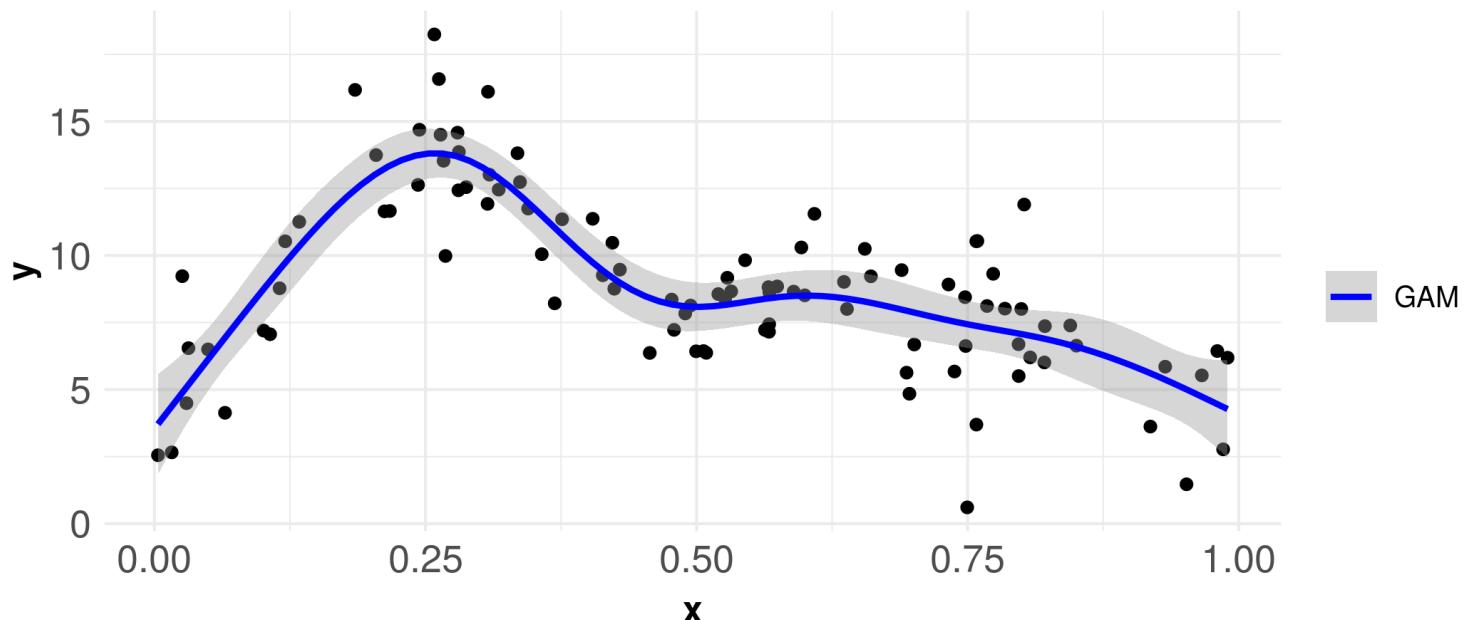
Generalized Additive Models (GAMs)

Generalized Additive Models (GAM)

```
require(mgcv)
gam(y ~ s(x),
    data = data)
```

Dev. explained: 0.69

AIC: 425



Generalized Additive Models (GAMs)

Linear Models

GAMs

Black-Box ML



<https://noamross.github.io/gams-in-r-course>

Generalized Additive Models (GAMs)

140 char vrsn

- 1 GAMs are just GLMs
- 2 GAMs fit wiggly terms
- 3 use + s(foo) not foo in frmla
- 4 use method = "REML"
- 5 gam.check()

— Dr Gavin Simpson 😊🇪🇺 (@ucfagls) March 16, 2017

<https://www.fromthebottomoftheheap.net>

Generalized Additive Models (GAMs)

$$y_i = \beta_0 + f_1(x_{1i}) + f_2(x_{2i}) + f_3(x_{3i}, x_{4i}) + \dots + \epsilon_i, \epsilon \sim N(0, \sigma^2)$$

$$\mathbb{E}(Y) = g^{-1} \left(\beta_0 + \sum_{j=1}^J f_j(x_j) \right)$$

$$f_j(x_j) = \sum_{k=1}^K \beta_{j,k} b_{j,k}(x_j)$$

- Smoothing function: $f()$

- Splines

- Kernels: K

- K Basis functions: $b_{j,k}$

- K Coefficients: $\beta_{j,k}$

- Smoothing parameter

Splines



<https://www.core77.com/posts/55368/When-Splines-Were-Physical-Objects>

Splines

- Splines replace a predictor variable, with a set of synthetic predictor variables.
- Family of functions or transformations that can be applied to x
 - Polynomial basis function

$$f_j(x_i) = x_i^j$$

- Piecewise constant basis function:

$$f_j(x_i) = I(c_j \leq x_i \leq c_{j+1})$$

- Penalized splines reduce wigginess

<https://www.youtube.com/watch?v=ENxTrFf9a7c&t=2226>

<https://www.tjmahr.com/random-effects-penalized-splines-same-thing/>

Splines

Example: Airquality

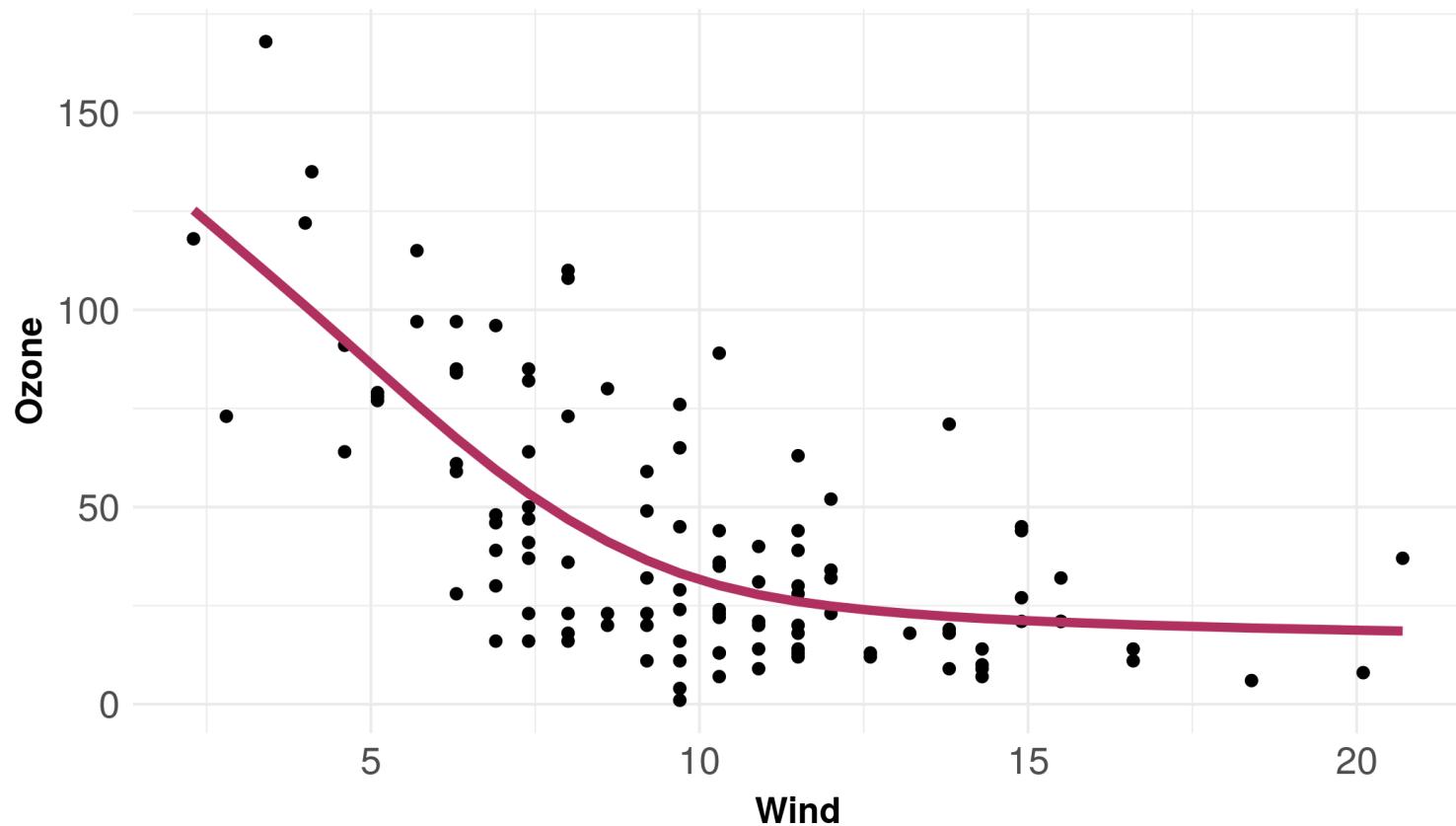
```
# data
airqu = na.omit(airquality)
head(airqu)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1     41      190  7.4   67     5    1
## 2     36      118  8.0   72     5    2
## 3     12      149 12.6   74     5    3
## 4     18      313 11.5   62     5    4
## 7     23      299  8.6   65     5    7
## 8     19       99 13.8   59     5    8
```

```
# model
air1 = gam(Ozone ~ s(Wind, bs = 'cr', k = 7), # Cubic Regr. Spline
            data = airqu,
            method = 'REML')
```

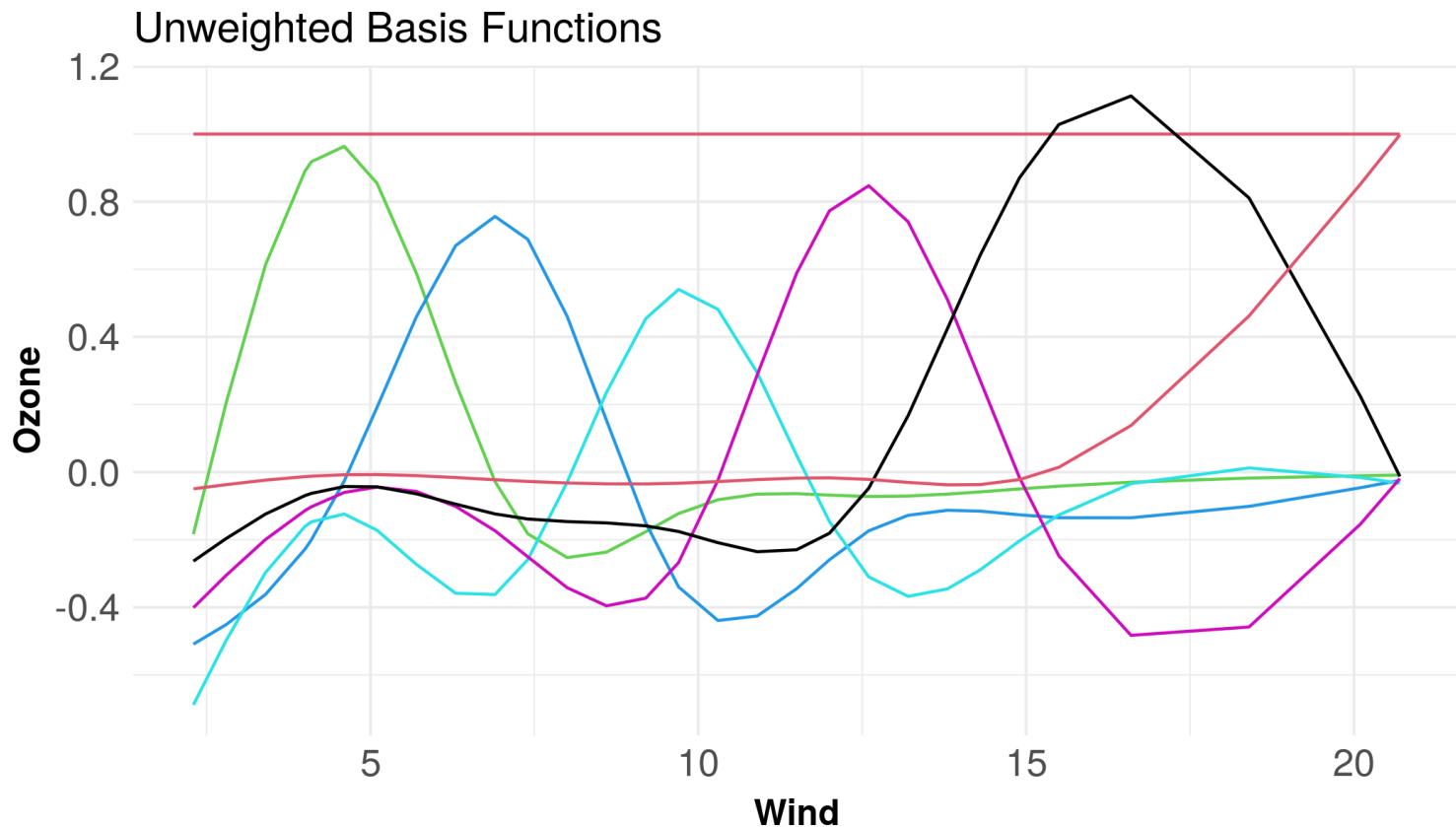
Splines

Example: Airquality



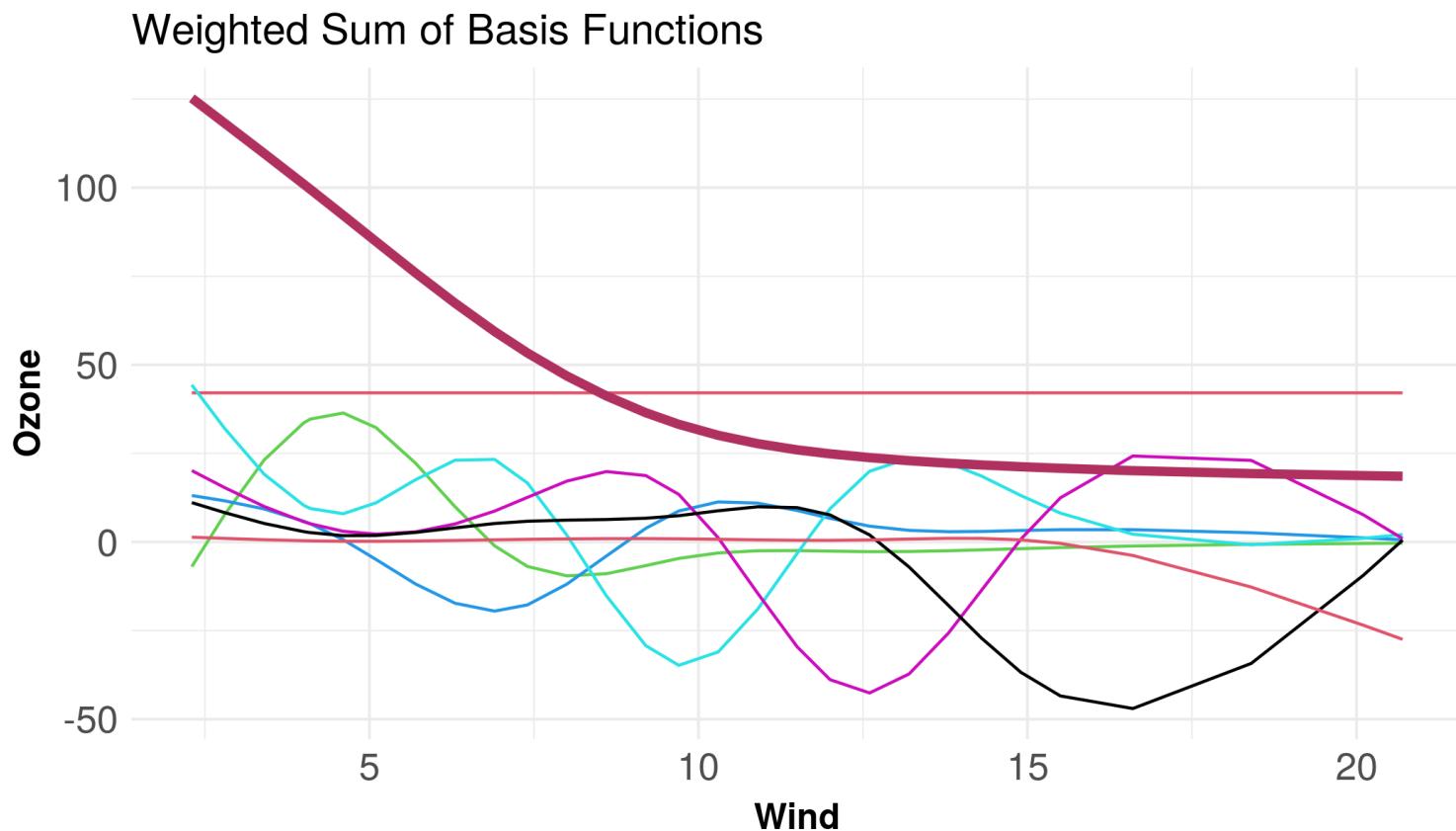
Splines

```
predict(air1, type = 'lpmatrix')
```



Splines

```
predict(air1, type = 'lpmatrix') %*% diag(coef(air1))
```



Splines

Splines

Complexity

```
lm1 = lm(y ~ x, data = sim)
coef(lm1)
```

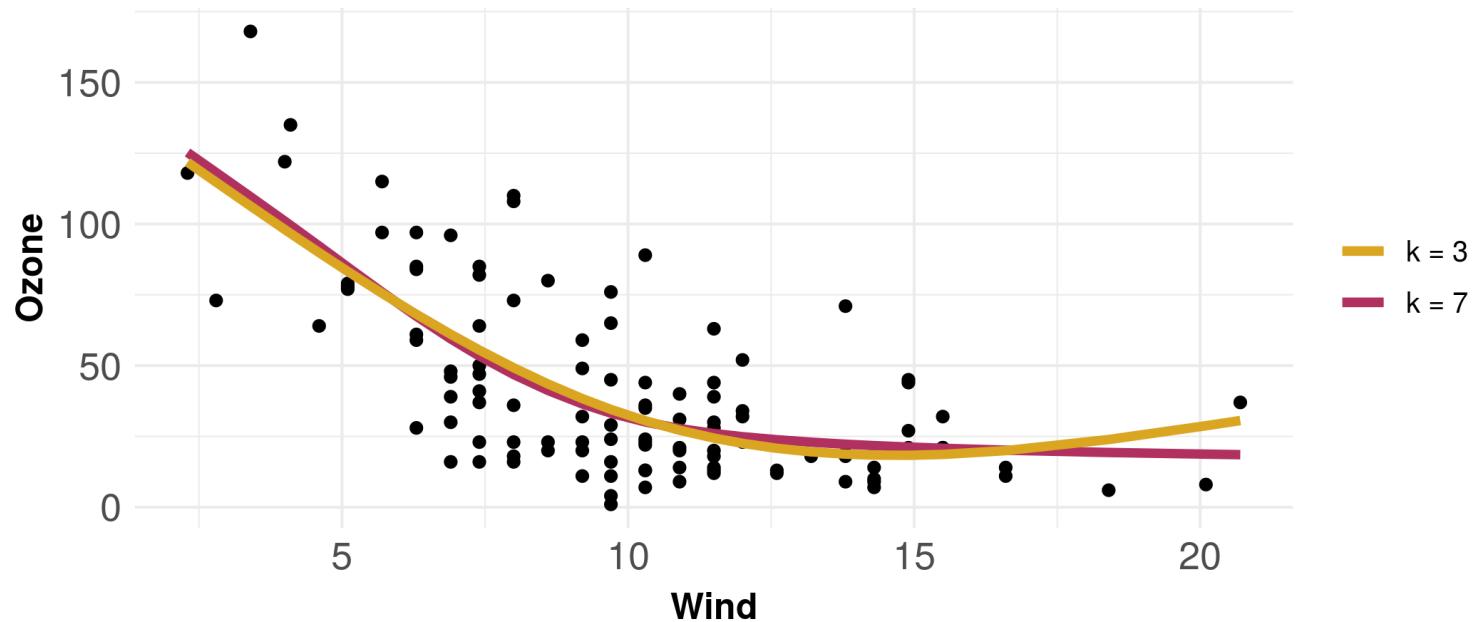
```
## (Intercept)          x
##   11.509559    -5.235207
```

```
gam1 = gam(y ~ s(x, k = 7), data = sim)
coef(gam1)
```

```
## (Intercept)      s(x).1      s(x).2      s(x).3      s(x).4      s(x).5
##   8.885758    -8.004486   -12.522057   -5.455329   -3.854084   19.889658
##      s(x).6
##   6.737313
```

Splines

```
air2 = gam(Ozone ~ s(Wind, bs = 'cr', k = 3), # Cubic Regr. Spline  
           data = airqu,  
           method = 'REML')
```



Splines - Knots

Check for k:

```
gam.check(air2)
```

```
k.check(air1) # k = 7
```

```
##           k'      edf   k-index p-value
## s(Wind)    6 3.427874 0.8576004  0.0525
```

```
k.check(air2) # k = 3
```

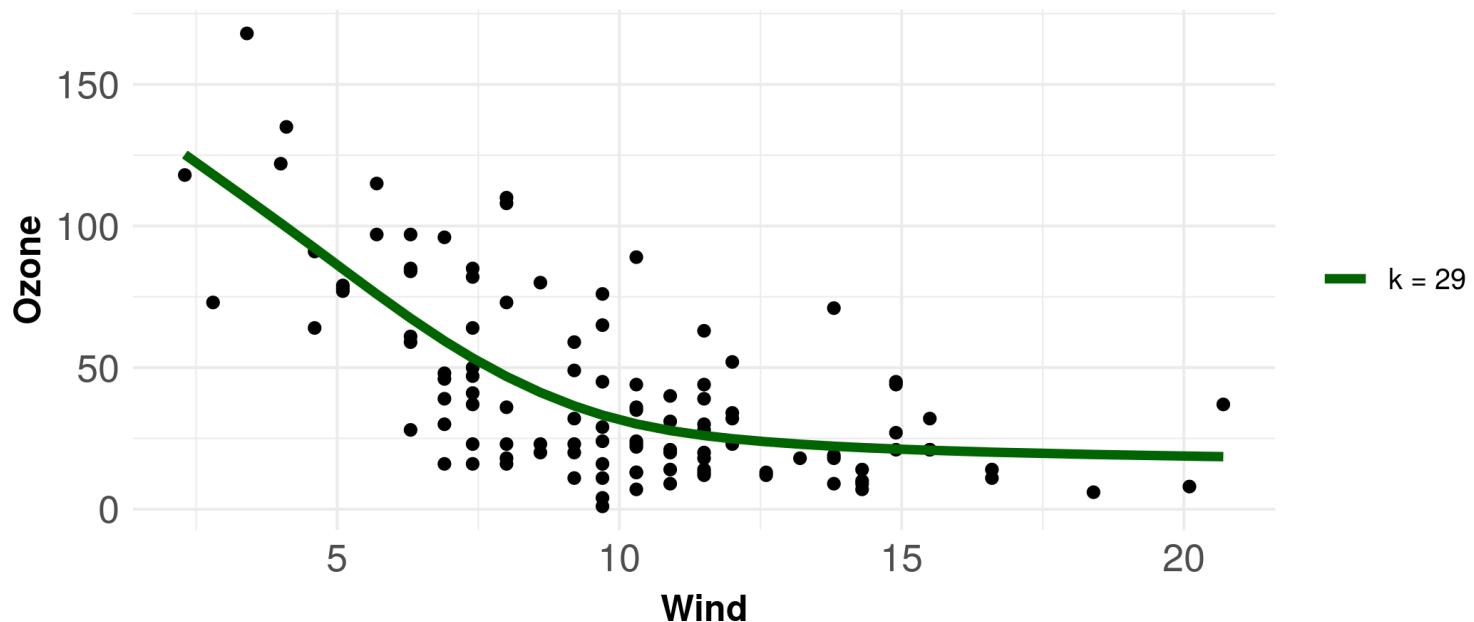
```
##           k'      edf   k-index p-value
## s(Wind)    2 1.966923 0.8369715  0.0225
```

If p-value is low, k might be too small

Overfitting?

Large basis size (knots) lead to overfitting?

```
air3 = gam(Ozone ~ s(Wind, bs = 'cr',  
                      k = length(unique(airqu$Wind))), # 29  
           data = airqu,  
           method = 'REML')
```



Overfitting?

SOLUTION:

Penalize departure from smoothness:

$$P(f) = \int f''(x)^2 dx = \beta^T \mathbf{S} \beta$$

<https://www.maths.ed.ac.uk/~swood34/mgcv/tampere/basis-penalty.pdf>

Smoothing

$$Fit = Likelihood - \lambda \times Wiggliness$$

$$\hat{\beta} = \arg \min_{\beta} \|y - X\beta\|^2 + \lambda \beta^T S \beta$$

- Likelihood: How well a GAM captures patterns in the data
- Wigginess: Complexity of a smooth
 - Penalty matrix: S
- Smoothing parameter λ is optimized in `gam()`
 - Controls the trade-off between Likelihood and Wigginess

<https://noamross.github.io/gams-in-r-course>

<https://www.maths.ed.ac.uk/~swood34/mgcv/tampere/basis-penalty.pdf>

Smoothing

Smoothing parameter

```
gam(y ~ s(x1, sp = NULL),  
     data = data,  
     sp = NULL,  
     method = 'REML') # GCV.Cp, ML
```

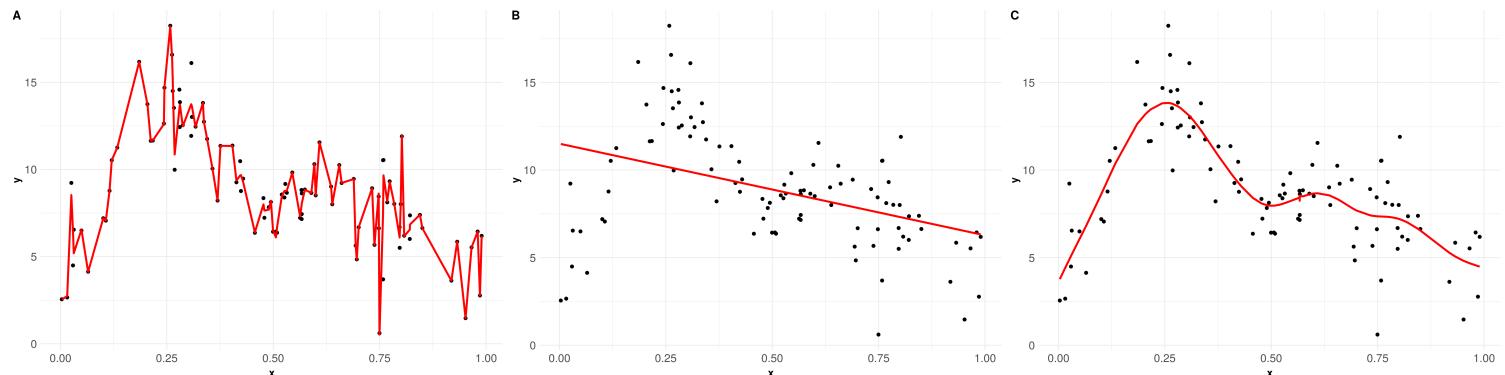
- Smoothing parameter estimation method: `method =`
 - Recommended: `method = 'REML'`¹

[1] Wood et al. (2011)

Maximum Likelihood Algorithm: <https://www.youtube.com/watch?v=XepXtl9YKwc>

Smoothing

Smoothing parameter



- A: `sp = 0`
- B: `sp = 1e5`
- C: `method = 'REML'`

The R-package does that for you!

Smoothness selection algorithms:

<https://www.maths.ed.ac.uk/~swood34/mgcv/tampere/smoothness.pdf>

GAMs in R

GAMs in R

gam package

- Original GAM R-package

mgcv package

- Most used package

qgam package

- Quantile GAMs. Model e.g. the 75% percentile

gamlss package

- Model not only the location & scale, but also the shape (e.g. kurtosis)

brms package

- Bayesian GAM approach
- runs STAN

The mgcv package

`mgcv:::gam()`

- Standard function

`mgcv:::bam()`

- Reduces RAM-overhead

`mgcv:::gamm()`

- Uses the `nlme` package for random effects

`gamm4:::gamm4()`

- Uses the `lme4` package for random effects

GAMs in R

R-function:

```
require(mgcv)

gam(y ~ s(x1, bs = 'tp', k = -1) + x2, # formula
     data = data, # data
     family = 'gaussian', # family object
     method = 'REML', # default: 'GCV.Cp'
     sp = NULL) # smoothing parameter
```

GAMs in R

Formula:

```
require(mgcv)

# gam(y ~
#       s(# smooth term s(), te()
#           x1, # predictor
#           bs = 'tp', # spline basis
#           k = -1, # number of basis functions (i.e. knots)
#           sp = NULL # smoothing parameter
#       )
#       # + x2, # linear term
#       # data = data,
#       # family = 'gaussian',
#       # method = 'REML' # default: 'GCV.Cp'
#       # sp = NULL)
```

Smooth terms

Smooth terms

Two additive smooths

```
gam(y ~ s(x1) + s(x2))
```

Smooth-interactions

```
gam(y ~ s(x1, x2)) # common way to declare spatial data  
gam(y ~ s(x1, by = fac))
```

Tensor product smooths

```
gam(y ~ te(x1, x2, k = c(4,8))) # interaction on different scales
```

Splines

- Thin Plate Regression Splines (TPRS)
 - Default
 - Computationally more demanding (CRS)
- Cubic Regression Splines (CRS)
 - Computationally less demanding
 - Cyclic Cubic Regression Splines
- Random Effects
- Different penalty matrices: S

```
gam(y ~ s(x, bs = 'tp'),  
     data = data)
```

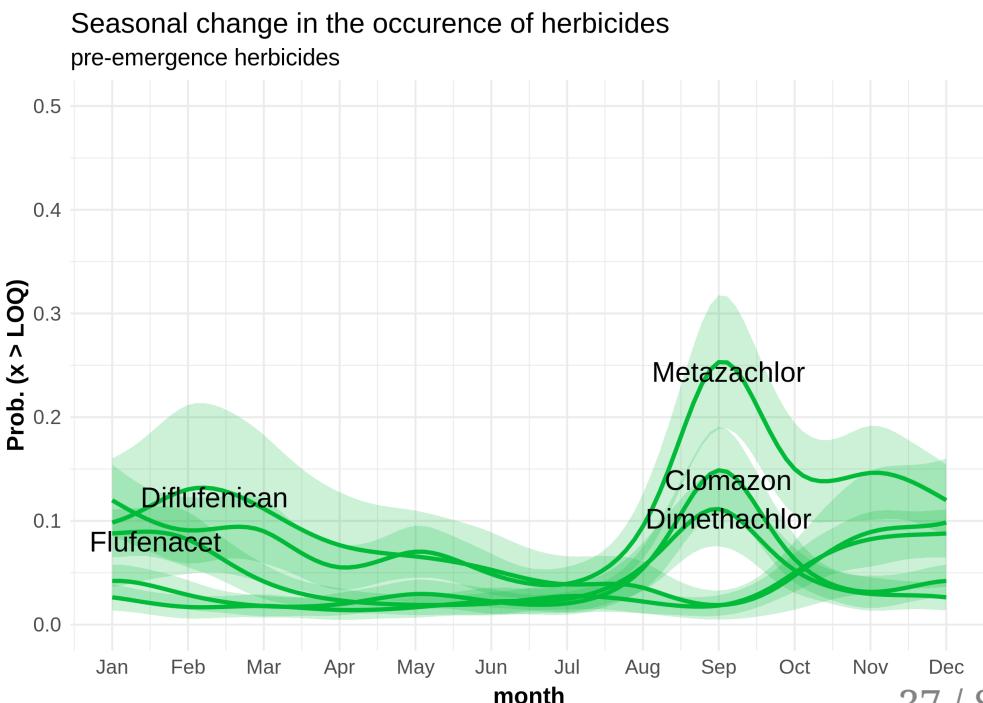
<https://stats.stackexchange.com/questions/305338/adaptive-gam-smooths-in-mgcv>

Splines

Cyclic Cubic regression splines

- Cyclical data (e.g. seasons)

```
gam(y ~ s(x, bs = 'cc'),  
     data = data)
```

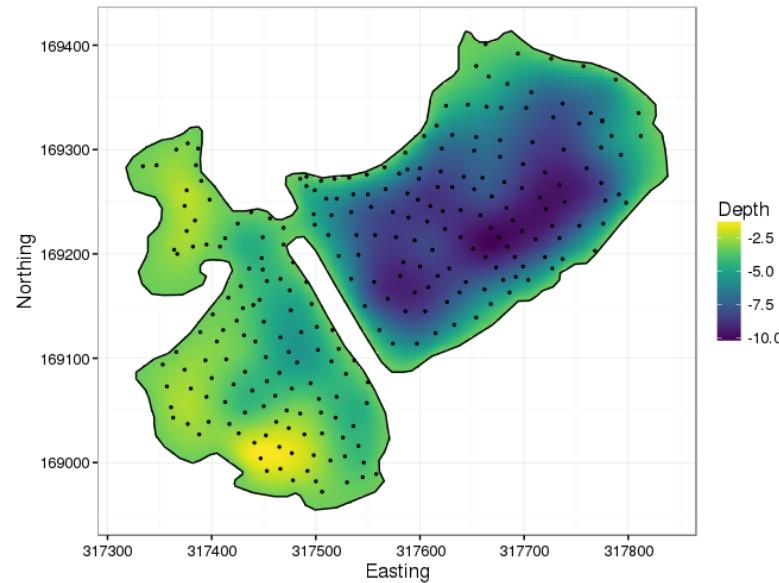


Splines

Soap Films

- Boundary polygons can be introduced
- Spatial models

```
gam(y ~ s(x, y, bs = 'so', xt = list(bnd = boundary_polygon)),  
     data = data)
```



Random Effects

Splines

Discrete random effects

- Classes (e.g. age, sex)
- Sites, states, rivers, lakes
- Mesocosm groups
- No need to set k (equals number of levels)

```
gam(y ~ s(x) + s(fac, bs = 're'),  
     data = data)
```

gamm() & gamm4()

- Calls `nlme::lme()`

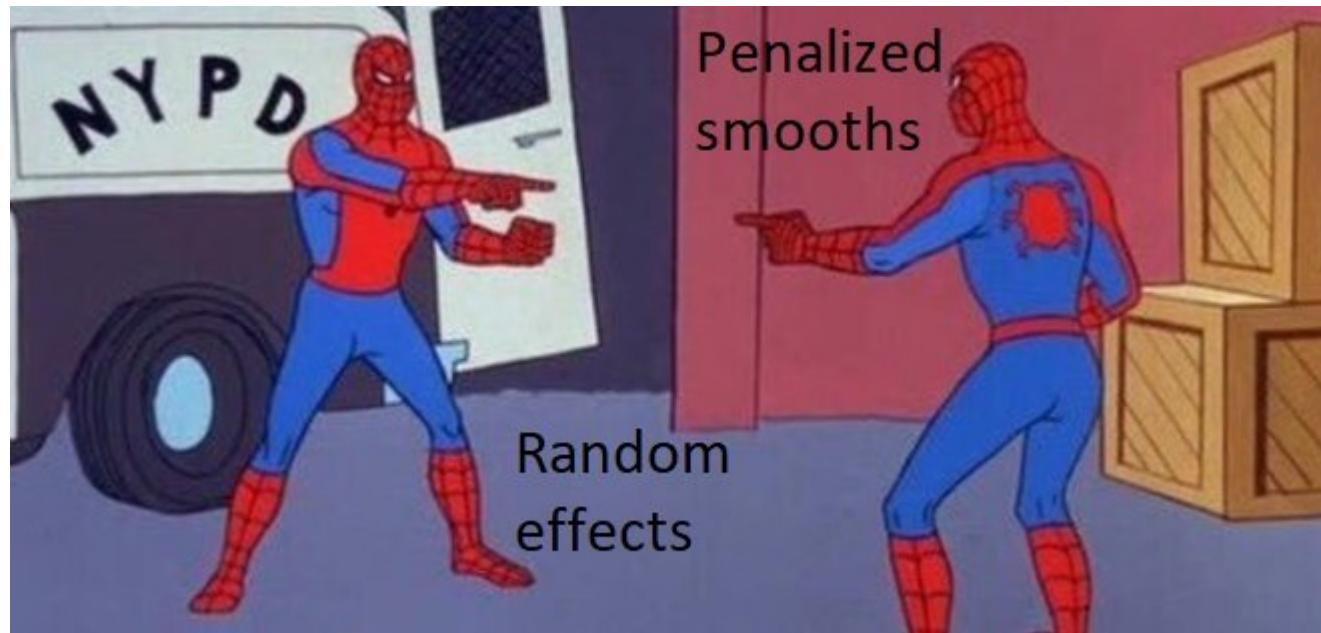
```
mgcv::gamm(y ~ s(x),  
            data = data,  
            random = list(fac = ~1))
```

- Calls `gamm4::gamm4()`

```
gamm4::gamm4(y ~ s(x),  
              data = data,  
              random = list(fac = ~ 1 | fac))
```

Random Effects

"Random effects and penalized splines are the same thing."



<https://twitter.com/ericJpedersen/status/1293508069016637440>

<https://www.tjmahr.com/random-effects-penalized-splines-same-thing>

Model example

Model example

```
require(gamair)
```

```
data('mack')
```

```
head(mack, n = 3)
```

```
##   egg.count egg.dens b.depth   lat   lon  time salinity flow s.depth temp
## 1          0         0    4342 44.57 -4.65  8.23   35.72   417    104
## 2          0         0    4334 44.57 -4.48  9.68   35.74   405     98
## 3          0         0    4286 44.57 -4.30 10.90   35.74   377    101
##   temp.20m net.area country vessel vessel.haul      c.dist
## 1    15.0    0.242       SP    COSA           22 0.8395141
## 2    15.4    0.242       SP    COSA           23 0.8591926
## 3    15.9    0.242       SP    COSA           24 0.8930153
```

- Response: `egg.count`
- Covariates:
 - Sea bed depth at sampling location: `b.depth`
 - Water salinity: `salinity`

<https://cran.r-project.org/web/packages/gamair/gamair.pdf>

Model example

```
mod = gam(egg.count ~ s(b.depth) + s(salinity),  
          data = mack,  
          family = 'nb')
```

Model example

Summary

```
summary(mod)
```

Predict

```
predict(mod)
```

AIC

```
AIC(mod)
```

Checking

```
gam.check(mod)  
k.check(mod)
```

Model summary

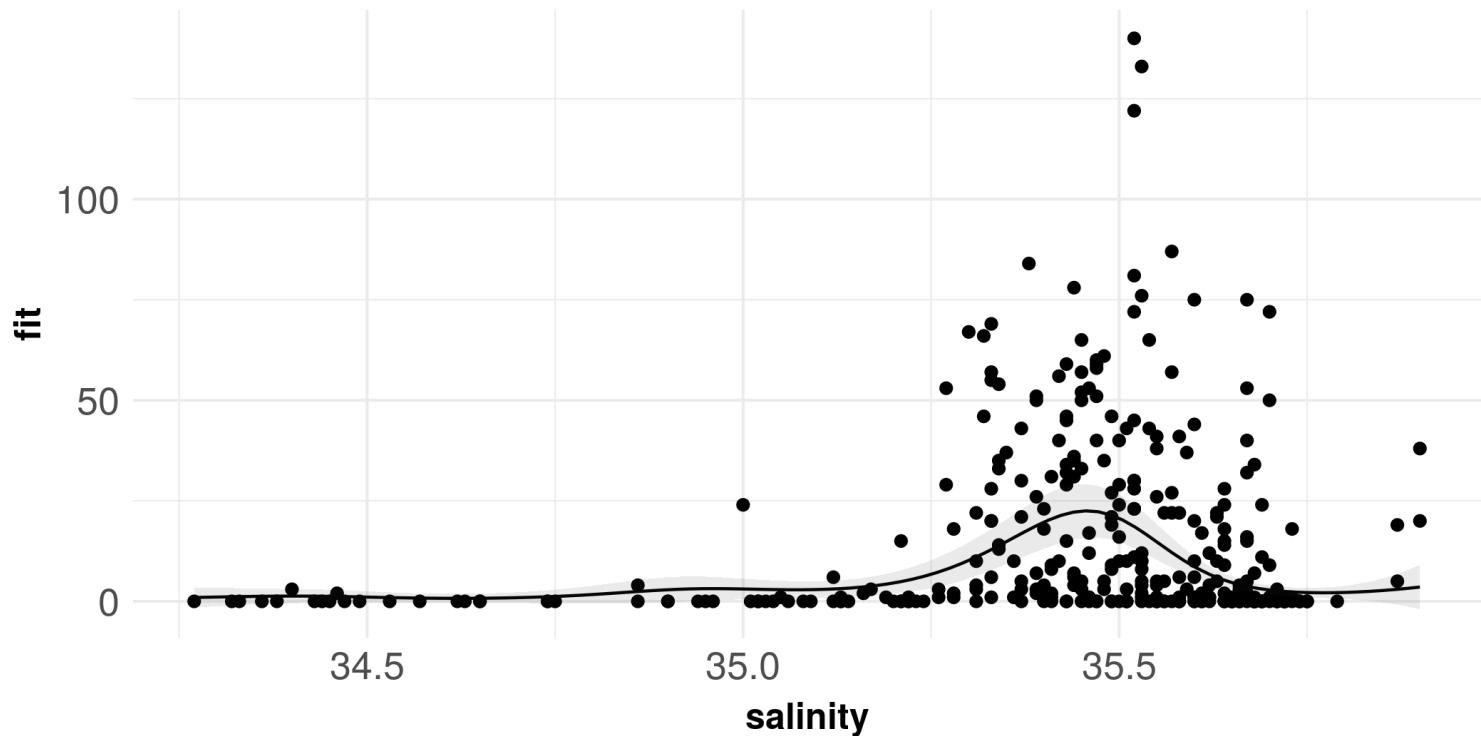
```
##  
## Family: Negative Binomial(0.455)  
## Link function: log  
##  
## Formula:  
## egg.count ~ s(b.depth) + s(salinity)  
##  
## Parametric coefficients:  
##             Estimate Std. Error z value Pr(>|z|)  
## (Intercept) 2.03383   0.08967  22.68 <2e-16 ***  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Approximate significance of smooth terms:  
##             edf Ref.df Chi.sq p-value  
## s(b.depth) 7.045  8.076 99.21 <2e-16 ***  
## s(salinity) 6.672  7.740 90.90 <2e-16 ***  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## R-sq.(adj) =  0.151    Deviance explained = 37.4%  
## -REML = 1047.9    Scale est. = 1            n = 330
```

Model prediction

```
# newdata
new = with(mack,
           expand.grid(b.depth = mean(b.depth),
                        salinity = seq(min(salinity, na.rm = TRUE),
                                       max(salinity, na.rm = TRUE),
                                       length.out = 100)))
setDT(new) # convert to data.table
# predict
prd = predict(mod,
              newdata = new,
              type = 'response',
              exclude = 's(b.depth)', # exclude of a variable
              se.fit = TRUE) # include standard errors
# update newdata
new[ , `:=`  
  (fit = prd$fit,  
   lwr = prd$fit - (1.96 * prd$se.fit),  
   upr = prd$fit + (1.96 * prd$se.fit)) ]
```

Model plot

```
ggplot(new, aes(y = fit, x = salinity)) +  
  geom_line() +  
  geom_point(data = mack, aes(y = egg.count)) +  
  geom_ribbon(aes(ymin = lwr, ymax = upr), alpha = 0.1)
```



Model checking

Model checking

Model selection

- Adds additional shrinkage on perfectly smooth terms
- Shrinkage Smoothers
 - `bs = 'ts'`, `bs = 'cs'`
- Double penalty approach
 - `select = TRUE`
 - Adds second penalty to all terms
 - Penalizes the NULL space
 - "Shrinkage", similar to Ridge Regression, Lasso

```
gam(y ~ s(x1) + s(x2, bs = 'ts') + s(x3),  
    select = TRUE,  
    data = data,  
    method = 'REML')
```

Marra & Wood (2011)

Model selection

- AIC
- Expert judgement
- Computational time
- **Inferential goals of the study**

<https://stats.stackexchange.com/questions/274151/anova-to-compare-models>

Exercise 1

GAM/exercise_pesticides.Rmd

Visualization

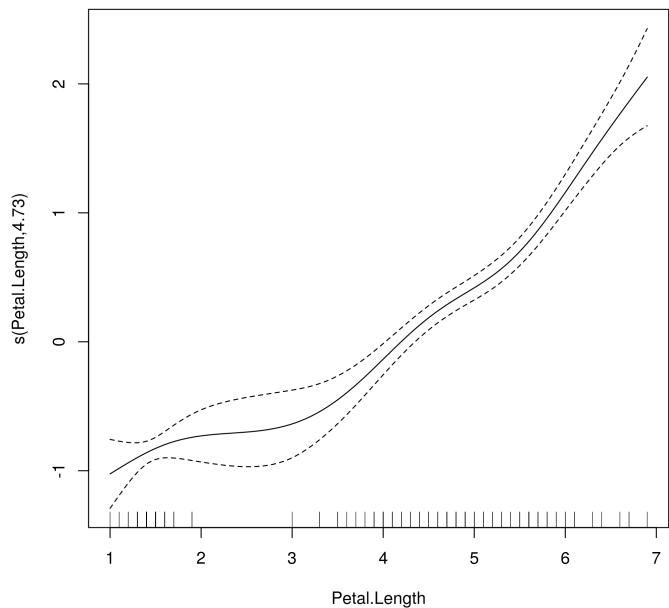
Visualization

```
irl = gam(Sepal.Length ~ s(Petal.Length),  
          data = iris,  
          family = 'gaussian',  
          method = 'REML')
```

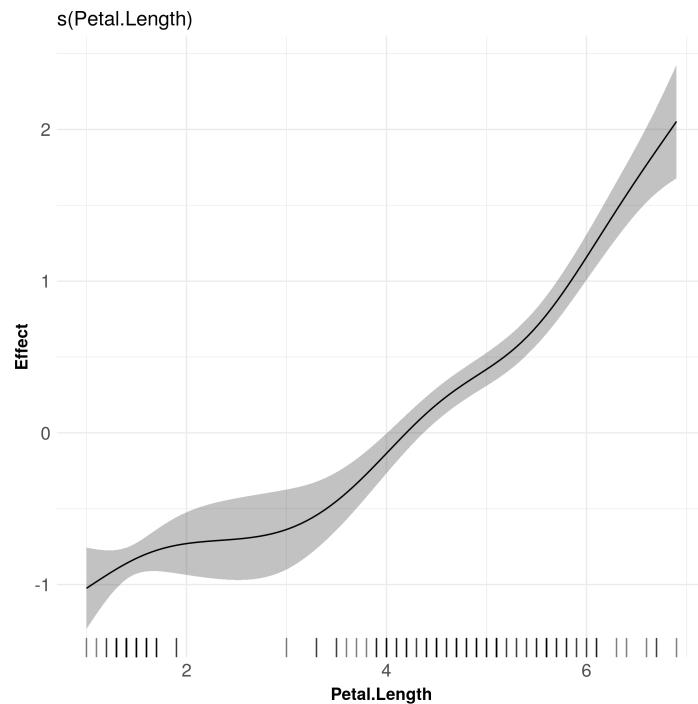
Visualization

Partial effect plots

```
require(mgcv)
plot(ir1, pages = 1)
```



```
require(gratia)
draw(ir1)
```



Visualization

Multiple covariates

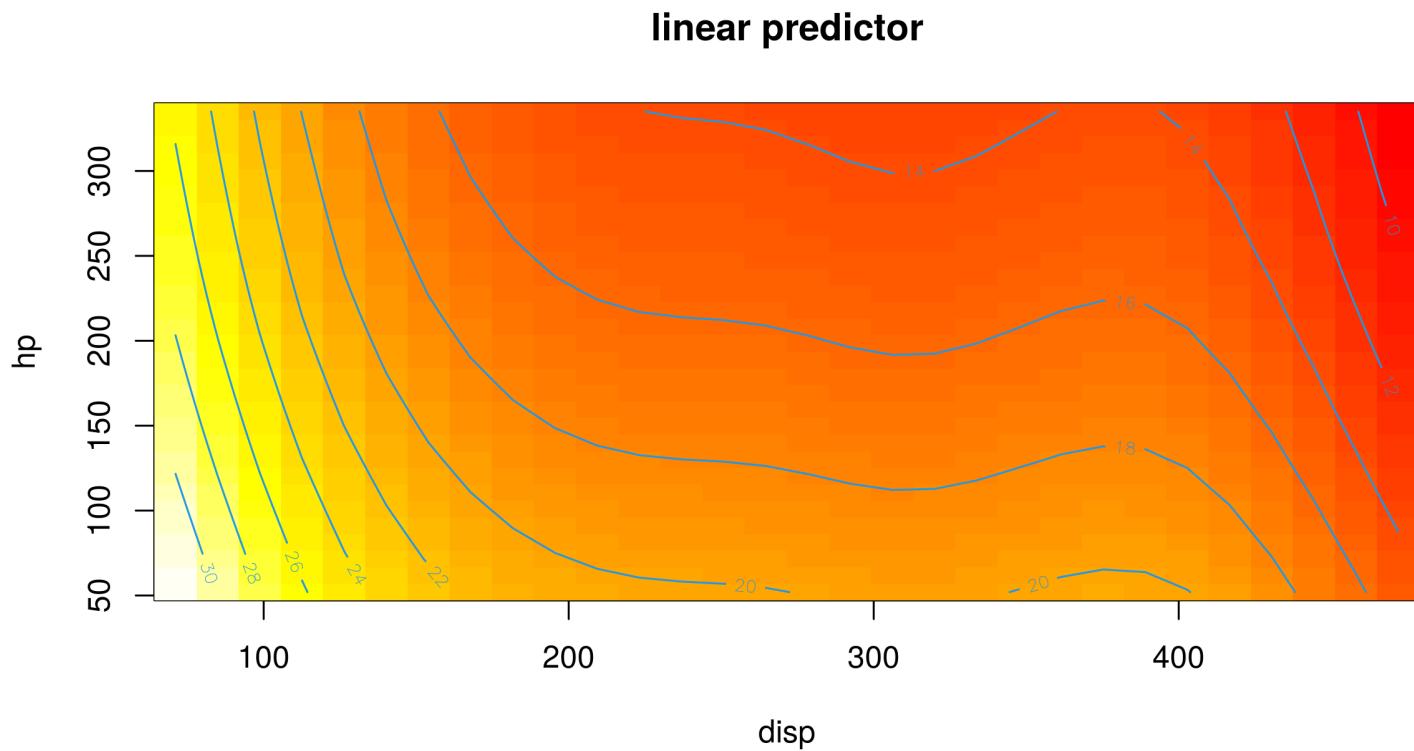
```
mt1 = gam(mpg ~ s(disp) + s(hp),  
          data = mtcars,  
          family = 'gaussian',  
          method = 'REML')
```

Visualization

```
vis.gam(mt1, # GAM object
         view = c("disp", "hp"), # variables
         plot.type = "persp", # 3D plot
         theta = 135, # horizontal rotation
         phi = 10, # phi vertical rotation
         r = 10) # zoom
```

Visualization

```
vis.gam(mt1, # GAM object  
        view = c("disp", "hp"), # variables  
        plot.type = "contour") # contour plot or heatmap
```



Hierachical / Mixed-effect GAMs

Hierarchical GAMs

- Hierarchical data: grouped data (factor!)
- Non-linear relationships
- Global function
- Group-specific function



Hierarchical generalized additive models in ecology: an introduction with mgcv

Eric J. Pedersen^{1,2}, David L. Miller^{3,4}, Gavin L. Simpson^{5,6} and
Noam Ross⁷

¹ Northwest Atlantic Fisheries Center, Fisheries and Oceans Canada, St. John's, NL, Canada

² Department of Biology, Memorial University of Newfoundland, St. John's, NL, Canada

³ Centre for Research into Ecological and Environmental Modelling, University of St Andrews,
St Andrews, UK

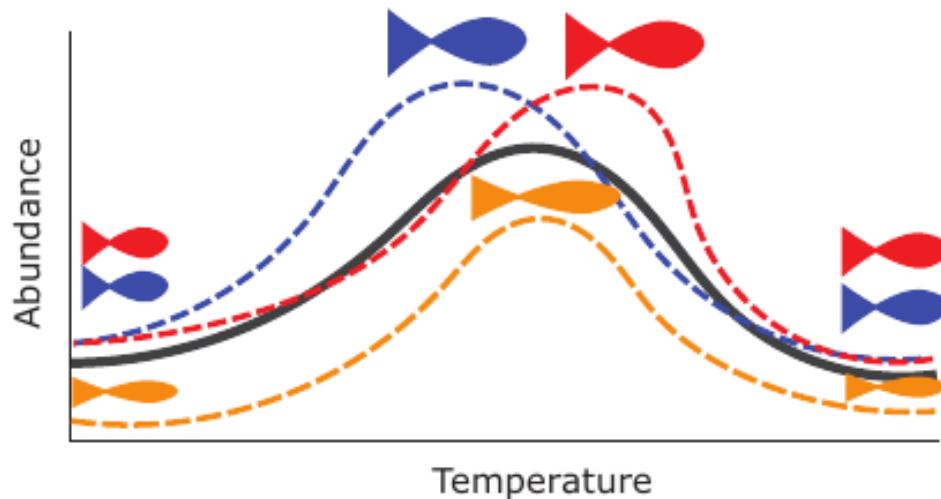
⁴ School of Mathematics and Statistics, University of St Andrews, St Andrews, Scotland, UK

⁵ Institute of Environmental Change and Society, University of Regina, Regina, SK, Canada

⁶ Department of Biology, University of Regina, Regina, SK, Canada

⁷ EcoHealth Alliance, New York, NY, USA

Hierarchical GAMs



al. (2019)

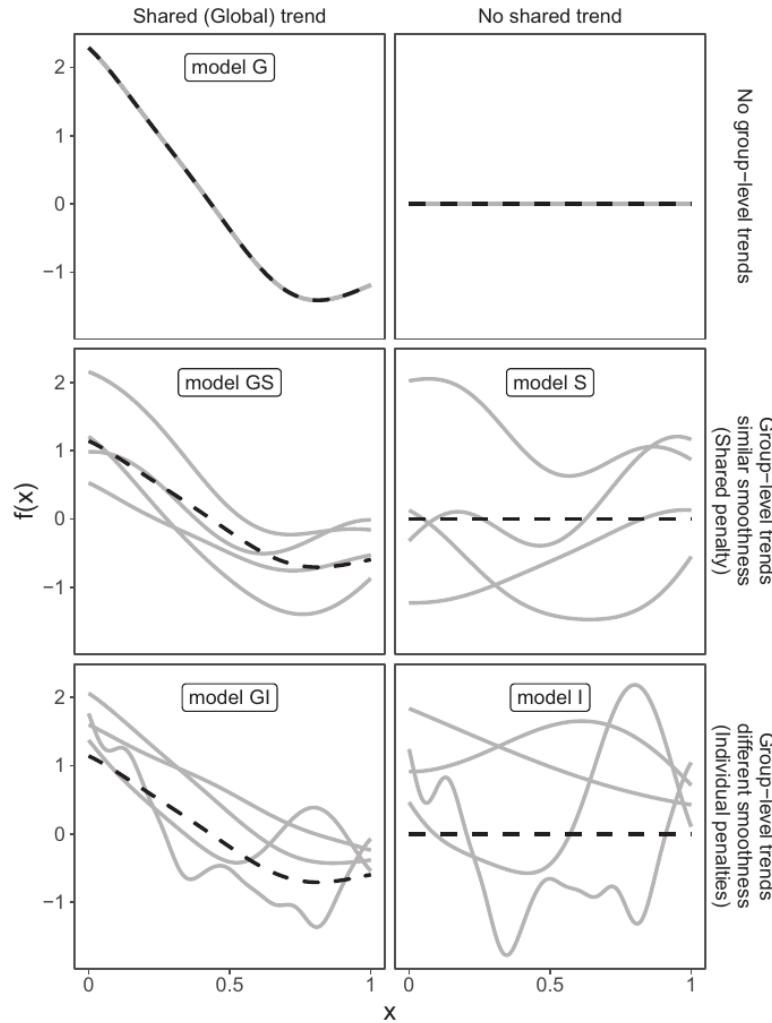
Pedersen et

- Estimating a function for each species throws away shared information
 - Highly noisy estimates
- Ignoring inherent grouping would miss individual optima

Hierarchical GAMs

- Should there be a common smoother or a smoother for each group?
- Do all group-specific smoothers have the **same wiggliness** or should each group wiggly independently (own smoothing parameter)?
- Will the smoothers for each group have a **similar shape**?

Hierarchical GAMs



Hierarchical GAM (HGAM)

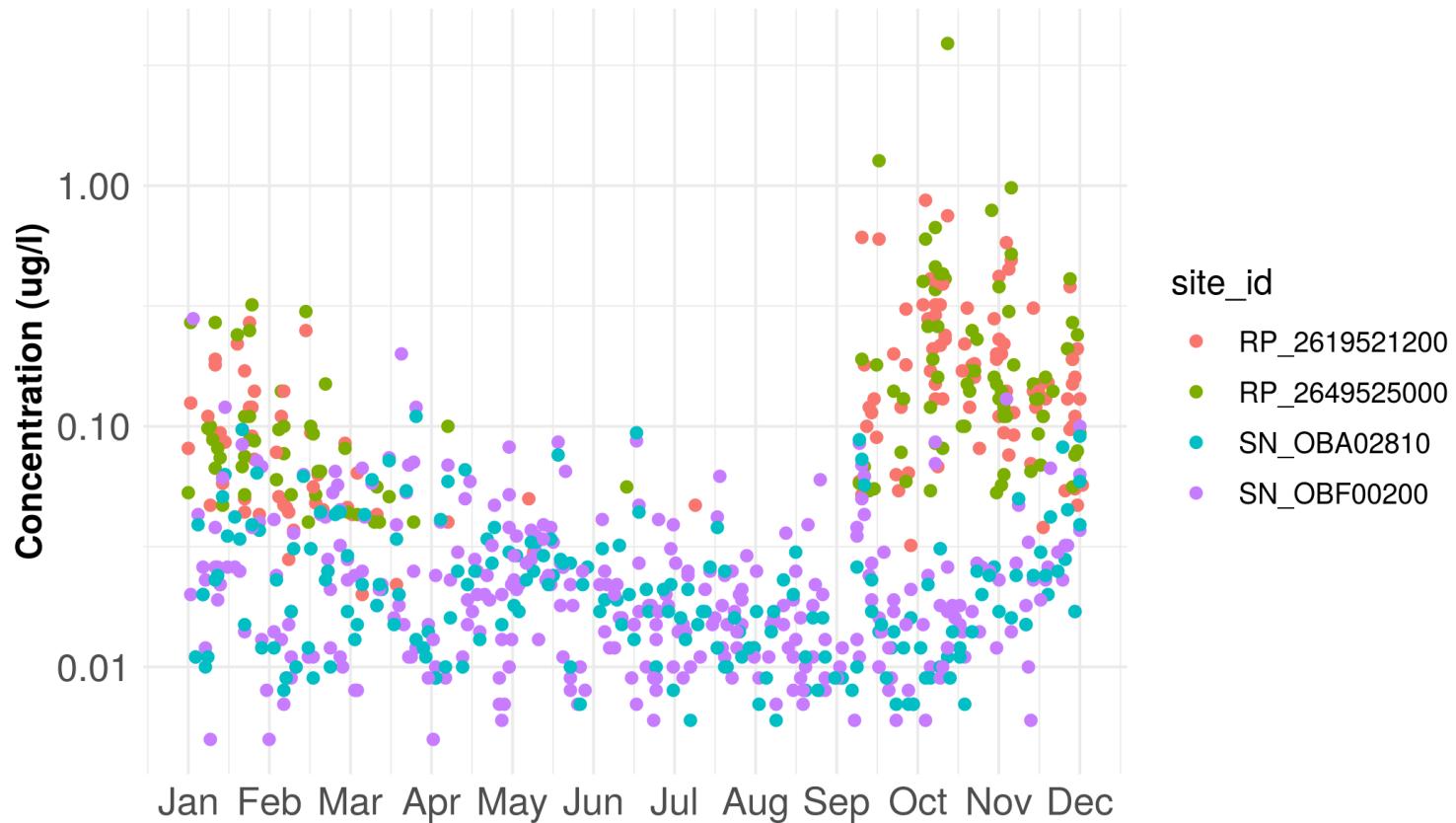
Chlortoluron

```
dat[ , head(.SD, 2), site_id]
```

```
##          site_id value_fin      date       name doy site_id_f
## 1: RP_2619521200    0.110 2011-11-28 Chlortoluron 332 RP_2619521200
## 2: RP_2619521200    0.310 2011-12-12 Chlortoluron 346 RP_2619521200
## 3: RP_2649525000    0.052 2010-02-22 Chlortoluron  53 RP_2649525000
## 4: RP_2649525000    0.140 2010-02-08 Chlortoluron  39 RP_2649525000
## 5: SN_OBF00200     0.068 2001-01-31 Chlortoluron  31 SN_OBF00200
## 6: SN_OBF00200     0.053 2001-03-01 Chlortoluron  60 SN_OBF00200
## 7: SN_OBA02810     0.022 2010-04-25 Chlortoluron 115 SN_OBA02810
## 8: SN_OBA02810     0.012 2010-04-07 Chlortoluron  97 SN_OBA02810
```

Hierarchical GAM (HGAM)

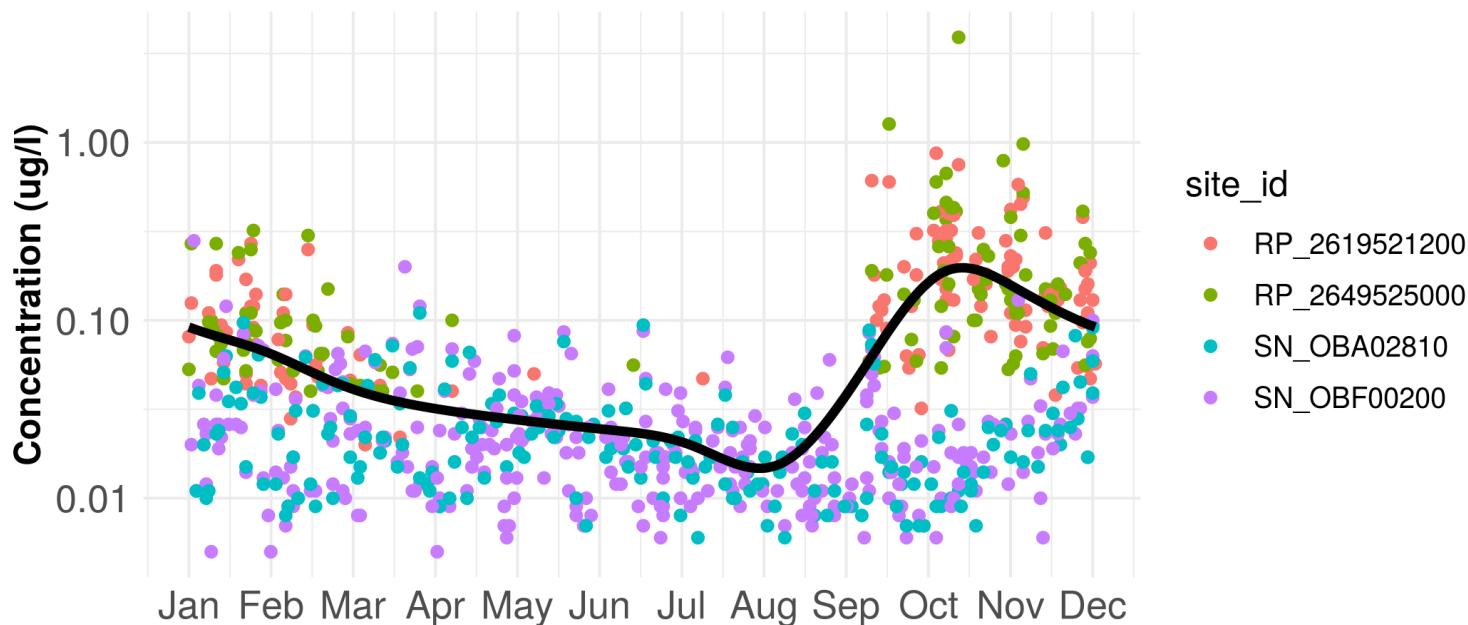
Chlortoluron



No HGAM

```
m0 = gam(value_fin ~ s(doy, bs = 'cc', k = 12),  
         data = dat,  
         family = Gamma(link = 'log'))
```

Chlortoluron (0)
AIC: -3014; Dev. expl.: 0.41



HGAM - Global Smooth

```
mG = gam(value_fin ~  
           s(doy, bs = 'cc', k = 12) +  
           s(site_id_f, bs = 're'),  
           data = dat,  
           family = Gamma(link = 'log'))
```

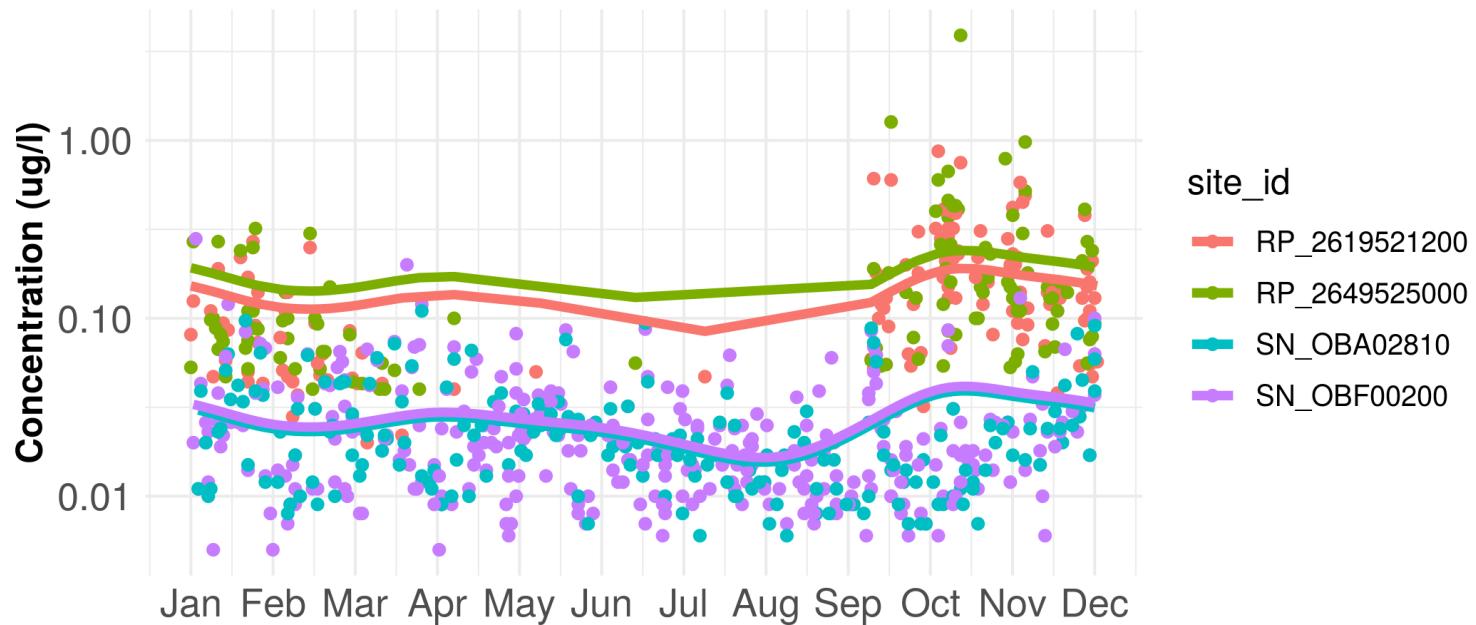
- Single global smooth term for each variable
- Level-individual **random effect intercepts** (`bs = 're'`)
- Mixed models (like in `lme4`, `nlme`)

HGAM - Global Smooth

```
mG = gam(value_fin ~  
           s(doy, bs = 'cc', k = 12) +  
           s(site_id_f, bs = 're'),  
           data = dat,  
           family = Gamma(link = 'log'))
```

Chlortoluron (G)

AIC: -3452; Dev. expl.: 0.64



HGAM - Global + Shared Group Level Smooth

```
mGS = gam(value_fin ~  
            s(doy, bs = 'cc', k = 12) +  
            s(doy, site_id_f, bs = 'fs', k = 12, xt=list(bs='cc'))  
            s(site_id_f, bs = 're'), # not explicitly needed  
            data = dat,  
            family = Gamma(link = 'log'))
```

- Factor-smooth interaction (bs = 'fs')
- Single global smooth term for each variable
- Varying slopes
- Copy of each set of basis functions,
- Penalizing functions too far away from the average
- **Similar wigginess**, one smoothing parameter

HGAM - Global + Shared Group Level Smooth

```
mGS = gam(value_fin ~  
            s(doy, bs = 'cc', k = 12) +  
            s(doy, site_id_f, bs = 'fs', k = 12, xt=list(bs='cc'))  
            s(site_id_f, bs = 're'), # not explicitly needed  
            data = dat,  
            family = Gamma(link = 'log'))
```

HGAM - Global + Individual Group Level Smooth

```
mGI = gam(value_fin ~  
            s(doy, bs = 'cc', k = 12) +  
            s(doy, by = site_id_f, bs = 'cc', k = 12) +  
            s(site_id_f, bs = 're')),  
            data = dat,  
            family = Gamma(link = 'log'))
```

- `s(x, by = fac)` and `s(fac, bs = 're')`
- Similar to GS
- Varying slopes
- **Different wiggliness**, multiple smoothing parameters
- Useful: Different wiggliness expected

HGAM - Global + Individual Group Level Smooth

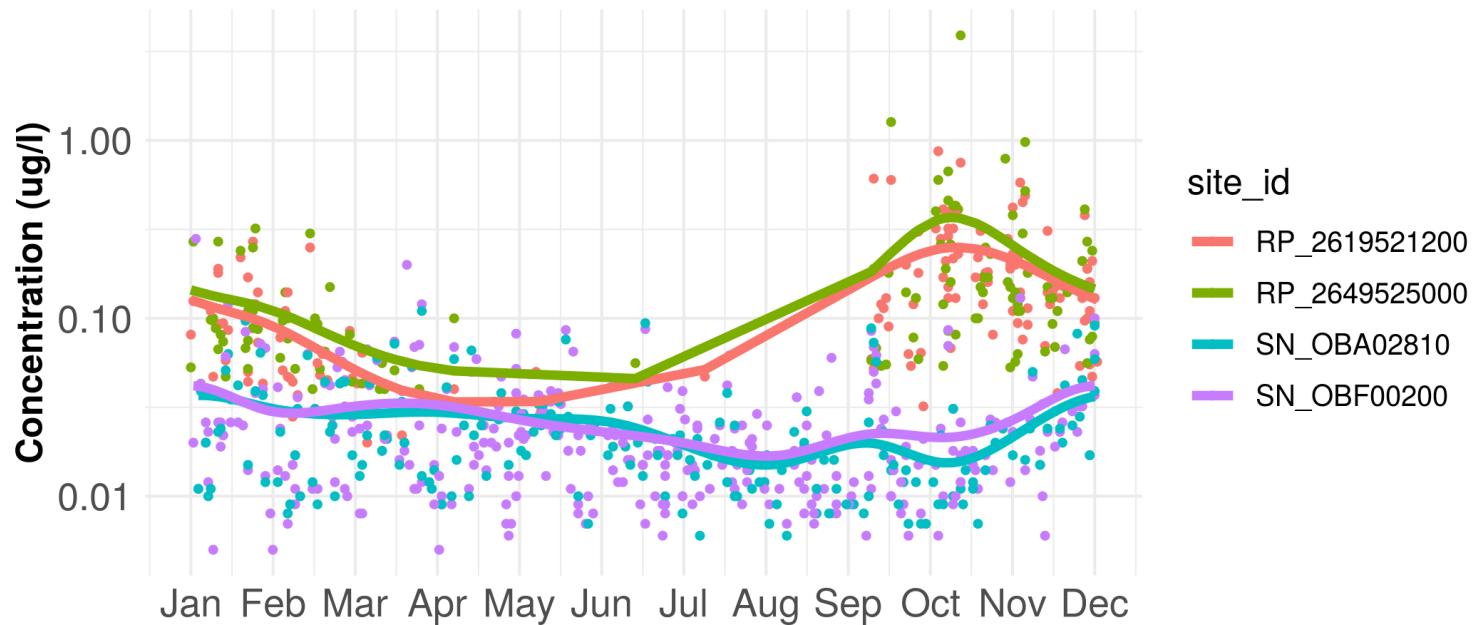
```
mGI = gam(value_fin ~  
            s(doy, bs = 'cc', k = 12) +  
            s(doy, by = site_id_f, bs = 'cc', k = 12) +  
            s(site_id_f, bs = 're')),  
            data = dat,  
            family = Gamma(link = 'log'))
```

HGAM - Only Shared Group Level Smooth

```
mS = gam(value_fin ~  
           s(doy, site_id_f, bs = 'fs', k = 12, xt=list(bs='cc')) +  
           s(site_id_f, bs = 're'), # not explicitly needed  
           data = dat,  
           family = Gamma(link = 'log'))
```

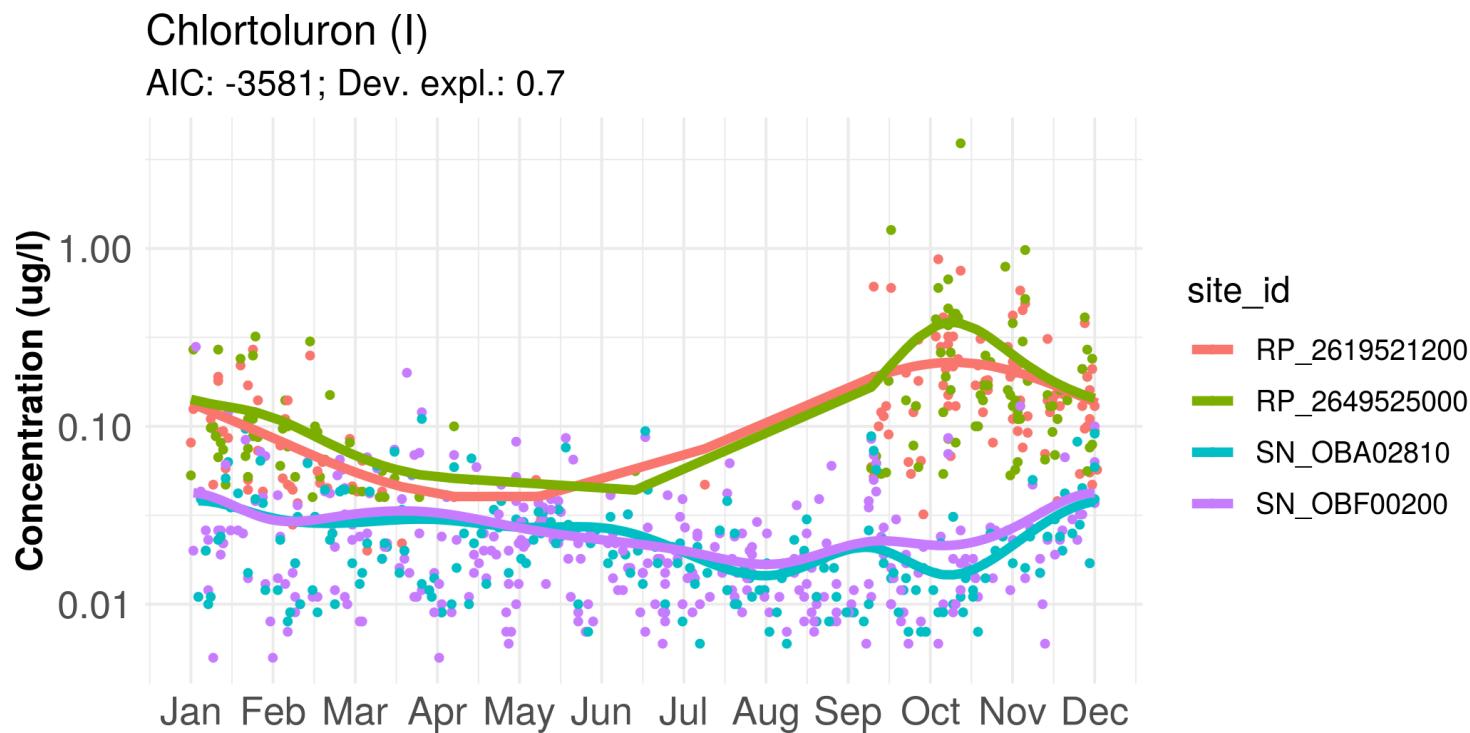
Chlortoluron (S)

AIC: -3578; Dev. expl.: 0.7



HGAM - Only Individual Group Level Smooth

```
mI = gam(value_fin ~  
           s(doy, by = site_id_f, bs = 'cc', k = 12) +  
           s(site_id_f, bs = 're'),  
           data = dat,  
           family = Gamma(link = 'log'))
```



Exercise 2

GAM/exercise_co2uptake.Rmd

Tutorials, Blogs & Videos

- Generalized Additive Models in R (Noam Ross)
 - <https://noamross.github.io/gams-in-r-course>
- Doing magic and analyzing seasonal time series with GAM (Generalized Additive Model) in R
 - <https://petolau.github.io/Analyzing-double-seasonal-time-series-with-GAM-in-R/>
- From the Bottom of the Heap - Blog by Gavin Simpson
 - <https://www.fromthebottomoftheheap.net/>
- Introduction lecture - By Gavin Simpson
 - <https://www.youtube.com/watch?v=sgw4cu8hrZM>
- Introducing gratia
 - <https://www.fromthebottomoftheheap.net/2018/10/23/introducing-gratia/>
- Noam Ross Github
 - <https://github.com/noamross/gam-resources>
- Environmental computing
 - <http://environmentalcomputing.net/intro-to-gams>
- Random effects and penalized splines are the same thing
 - <https://www.tjmahr.com/random-effects-penalized-splines-same-thing>

Slides

- OLAT
- <https://andschar.github.io/teaching>

Made with

- <https://github.com/rstudio/rmarkdown>
- <https://github.com/yihui/knitr>
- <https://github.com/yihui/xaringan>

References

Pedersen et al. (2019)

Marra & Wood (2011)

Wood (2011)

Burnham & Anderson (1998)

Thank you for your attention!

Andreas Scharmüller

@andschar

<https://andschar.github.io>