Generalised additive models

An introduction with R

Gavin Simpson (David Miller, Eric Pedersen, Noam Ross)

2018-01-31

What are GAMs?

Generalized **A**dditive **M**odel

- Generalized because they can handle response distibutions beyond the normal or Gaussian
    - **mgcv** can handle many types of data that lie even beyond traditional GLMs
- Additive — terms simply **add** together
- Model — a GAM has lots of theory for doing inference

Linear Models

$$y_i \sim \mathcal{N}(\mu_i, \sigma^2)$$

$$\mu_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_j x_{ji}$$

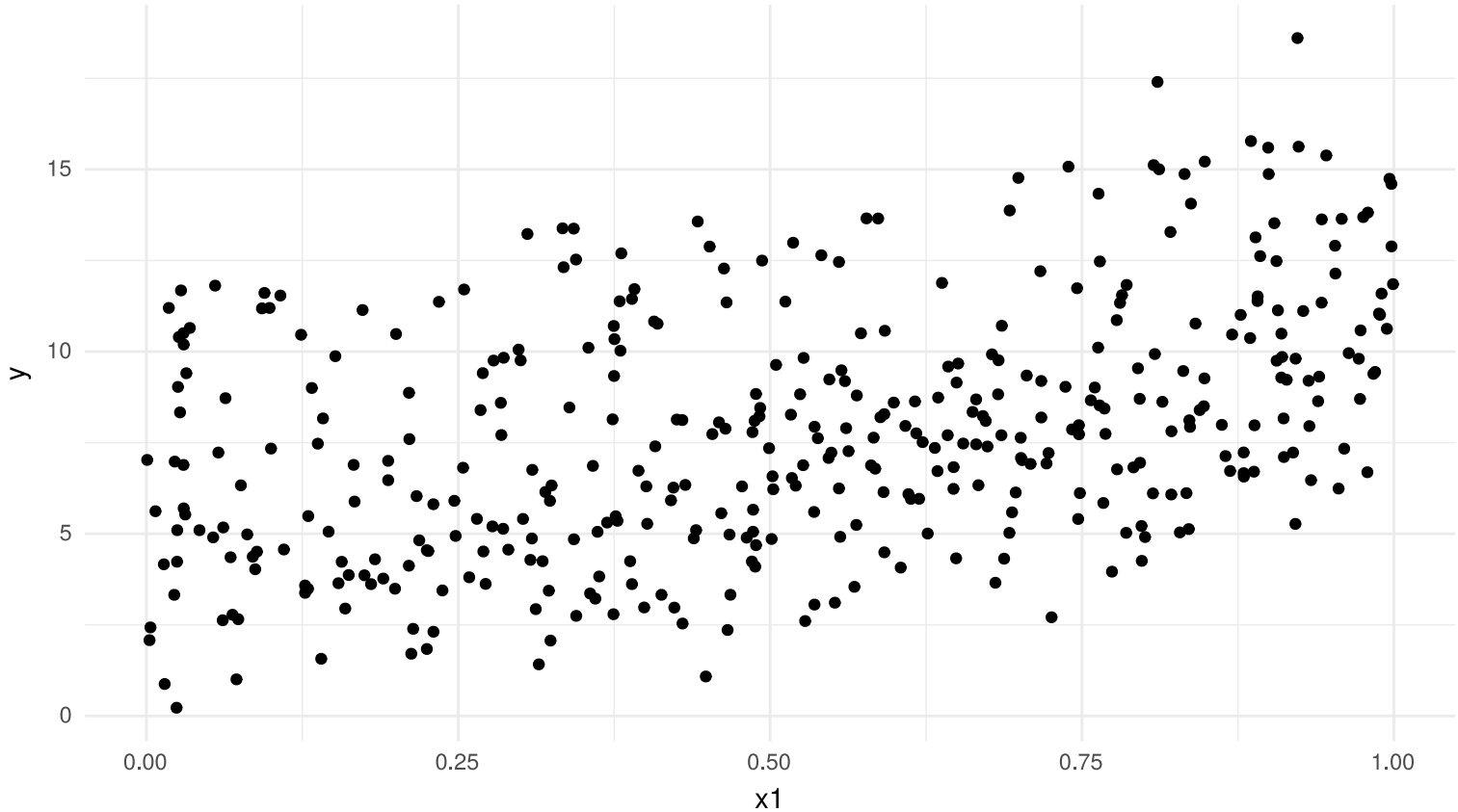
Assumptions

- linear effects of covariates are good approximation of the true effects
- conditional on the values of covariates, $y_i|\mathbf{X} \sim \mathcal{N}(0, \sigma^2)$
- this implies all observations have the same *variance*
- $y_i|\mathbf{X}$ are *independent*

An **additive** model address the first of these

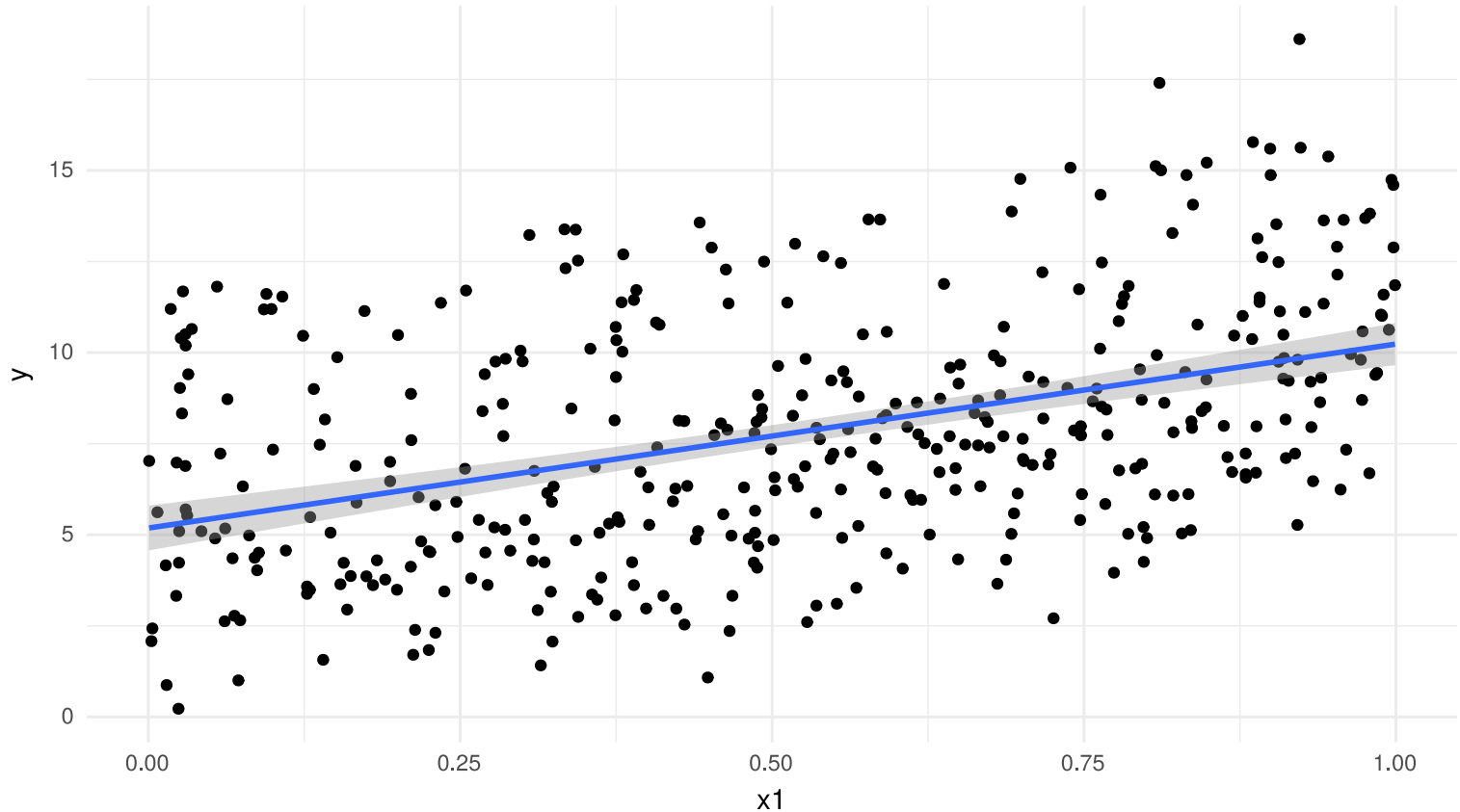
Why bother with anything more complex?

Is this linear?



Is this linear? Maybe?

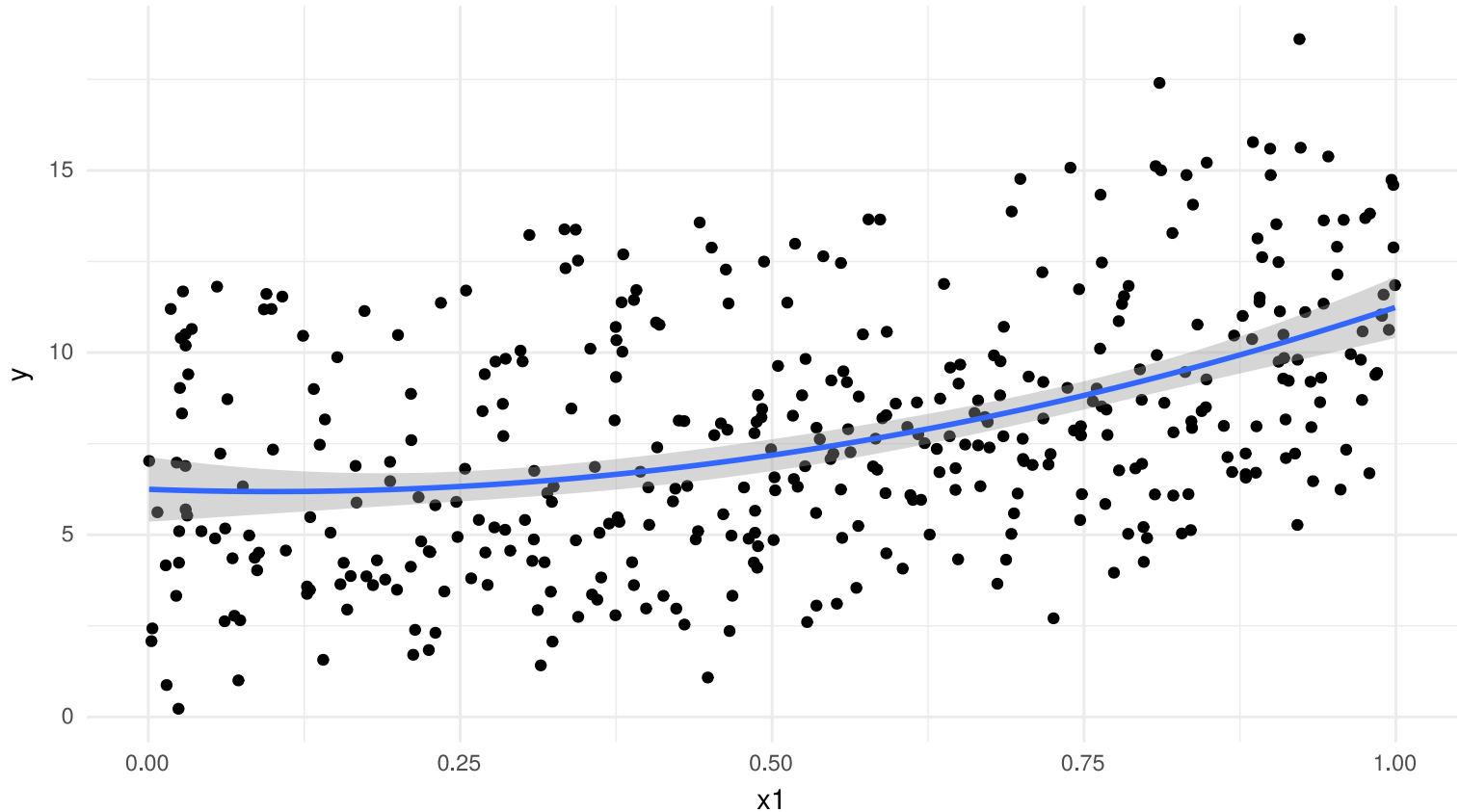
```
lm(y ~ x1, data=dat)
```



What can we do about it?

Adding a quadratic term?

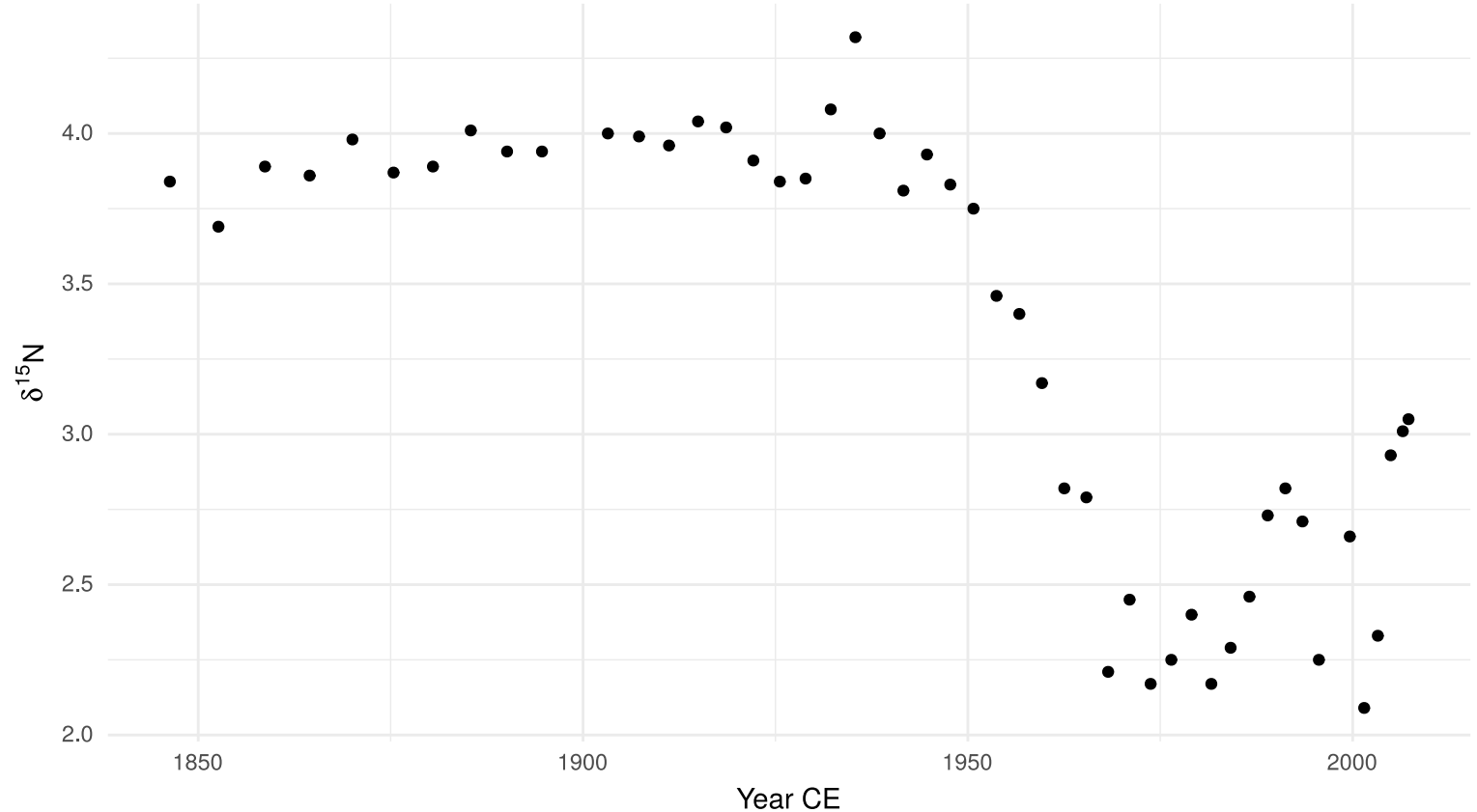
```
lm(y ~ poly(x1, 2), data = dat)
```



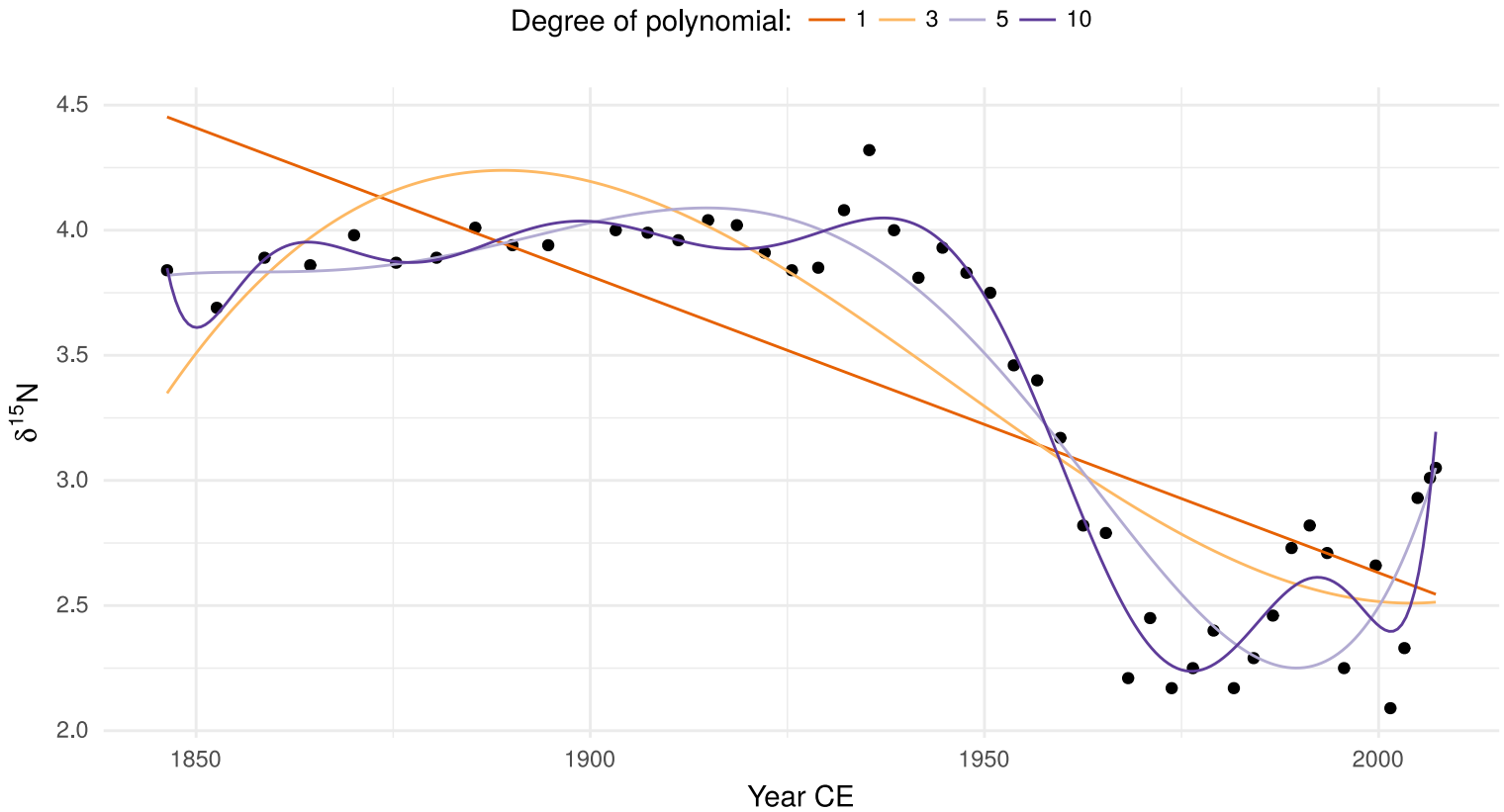
What about real data?

Small Water

What degree polynomial should we use here?



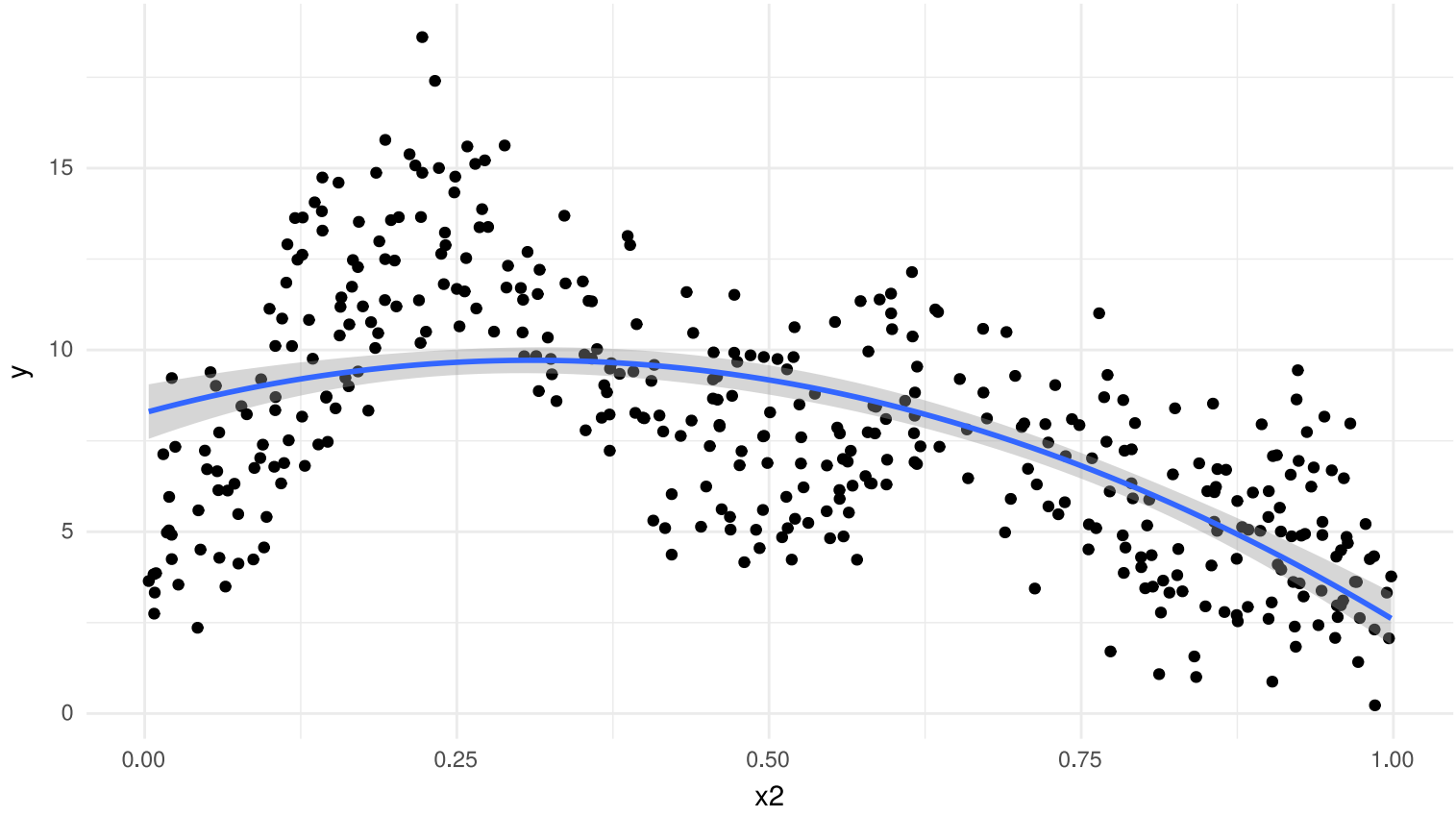
Polynomials



Is this sustainable?

Adding in quadratic (and higher terms) *can* make sense — feels a bit *ad hoc*

Better if we had a **framework** to deal with these issues?



This is where GAMs are useful

How is a GAM different?

In GLM we model the mean of data as a sum of linear terms:

$$y_i = \beta_0 + \sum_j \beta_j x_{ji} + \epsilon_i$$

A GAM is a sum of *smooth functions* or *smooths*

$$y_i = \beta_0 + \sum_j s_j(x_{ji}) + \epsilon_i$$

where $\epsilon_i \sim N(0, \sigma^2)$, $y_i \sim \text{Normal}$ (for now)

Call the above equation the **linear predictor** in both cases.

Fitting a GAM in R

```
model <- gam(y ~ s(x1) + s(x2) + te(x3, x4), # formuala describing model
             data = my_data_frame,           # your data
             method = 'REML',                # or 'ML'
             family = gaussian)              # or something more exotic
```

`s()` terms are smooths of one or more variables

`te()` terms are the smooth equivalent of *main effects + interactions*

```
  Depth   d13C  TotalC d15N TotalN DryWeight     Year
1   0.2 -27.57  806.49 3.05  64.21       8.2 2007.254
2   0.4 -27.67  949.33 3.01  73.26       7.6 2006.510
3   0.8 -27.63 1305.52 2.93  93.25      11.6 2004.941
4   1.2 -27.62 1136.04 2.33  86.09       9.6 2003.269
5   1.6 -27.48 1028.27 2.09  93.80      10.9 2001.496
6   2.0 -27.39  809.91 2.66  79.98       9.9 1999.626
```

Fit the GAM

```
sw <- gam(d15N ~ s(Year), data = small, method = 'REML')
```

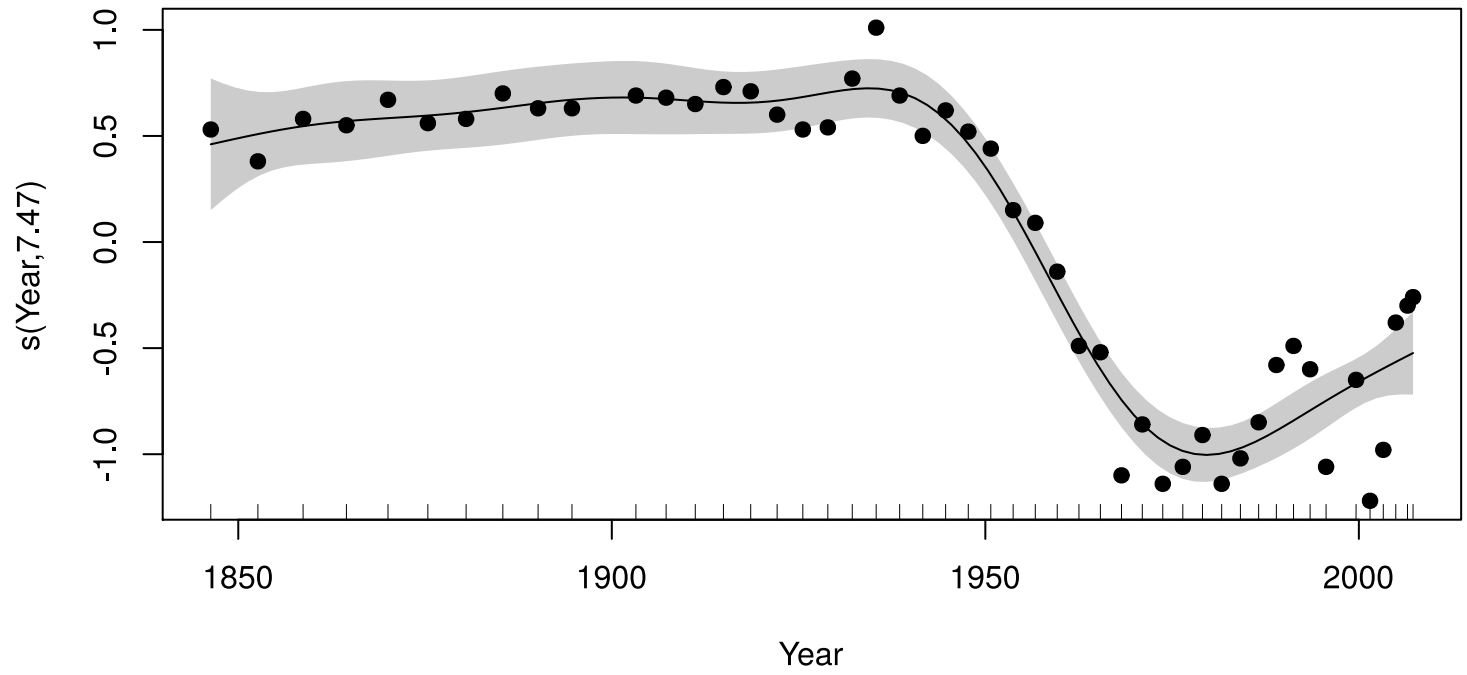
Look at the model summary

```
summary(sw)
```

```
Family: gaussian
Link function: identity

Formula:
d15N ~ s(Year)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.30958    0.02805     118   <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(Year) 7.466  8.416 70.13  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.926   Deviance explained = 93.8%
-REML = 4.8282  Scale est. = 0.037771  n = 48
```

Plot the fitted smooth

```
plot(sw, shade = TRUE, residuals = TRUE, pch = 19)
```



What magic is this?

Rob Potter

Splines

Splines are *functions* composed of simpler functions

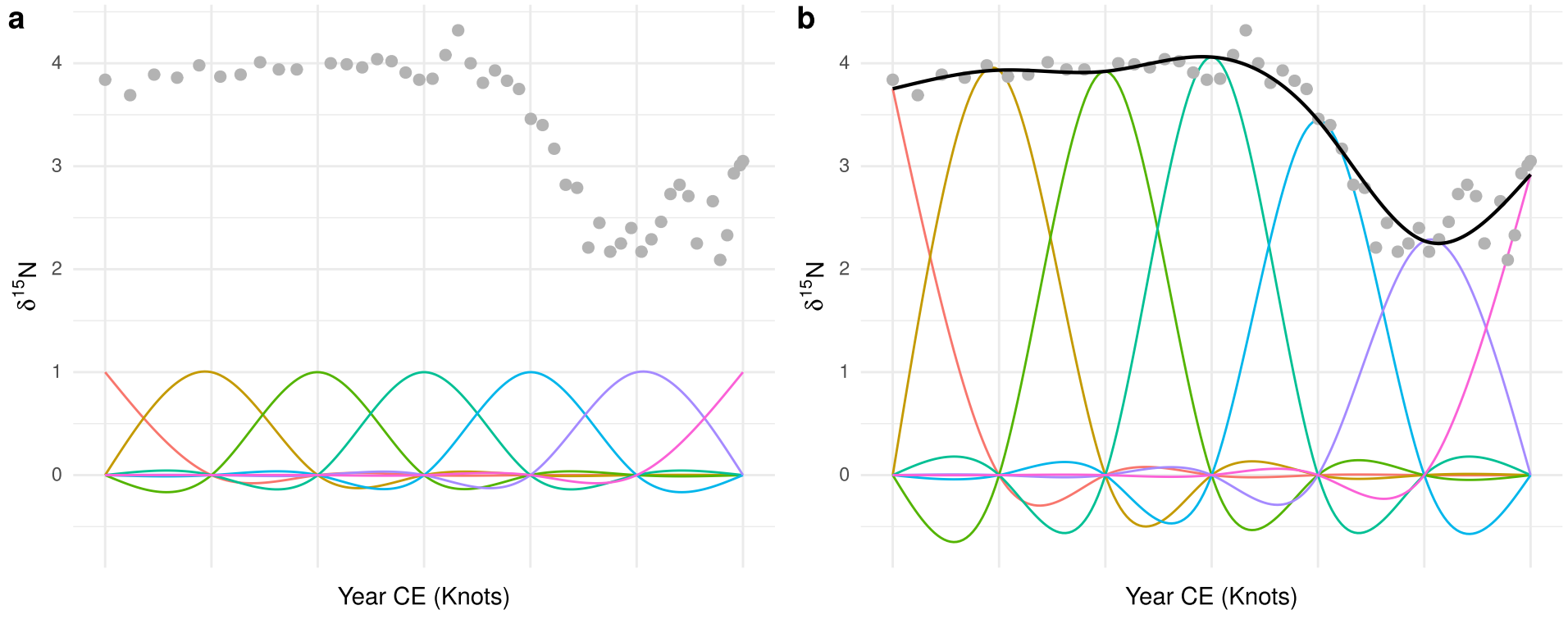
Simpler functions are *basis functions* & the set of basis functions is a *basis*

When we model using splines, each basis function b_k has a coefficient β_k

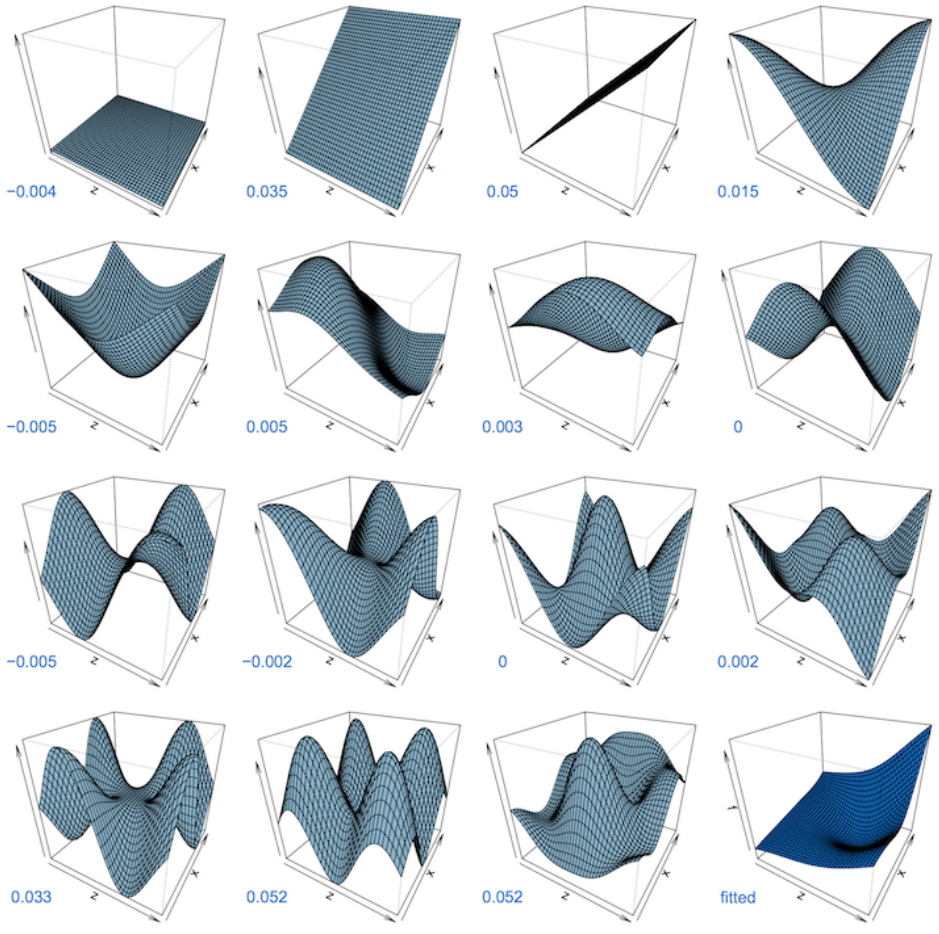
Resultant spline is a the sum of these weighted basis functions, evaluated at the values of x

$$s(x) = \sum_{k=1}^{K} \beta_k b_k(x)$$

Basis functions

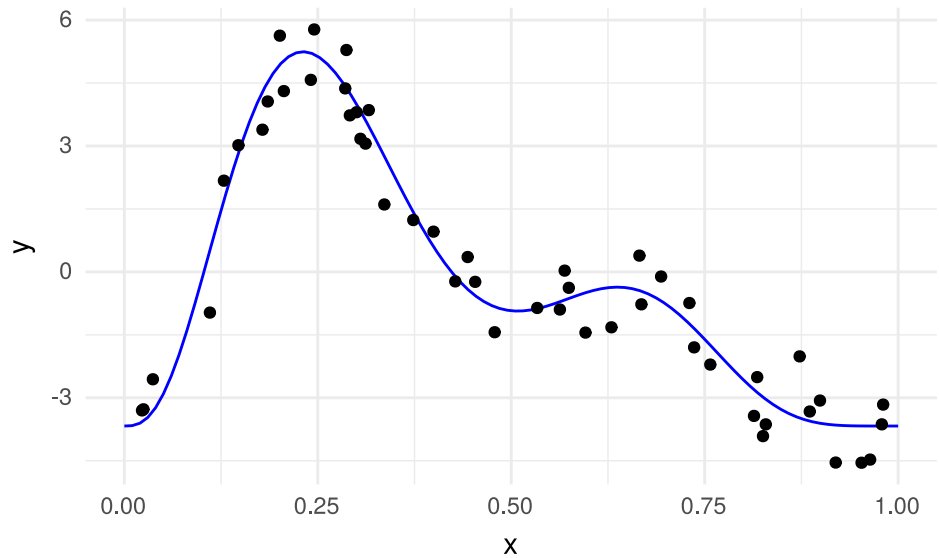


Basis functions



How do we avoid overfitting?

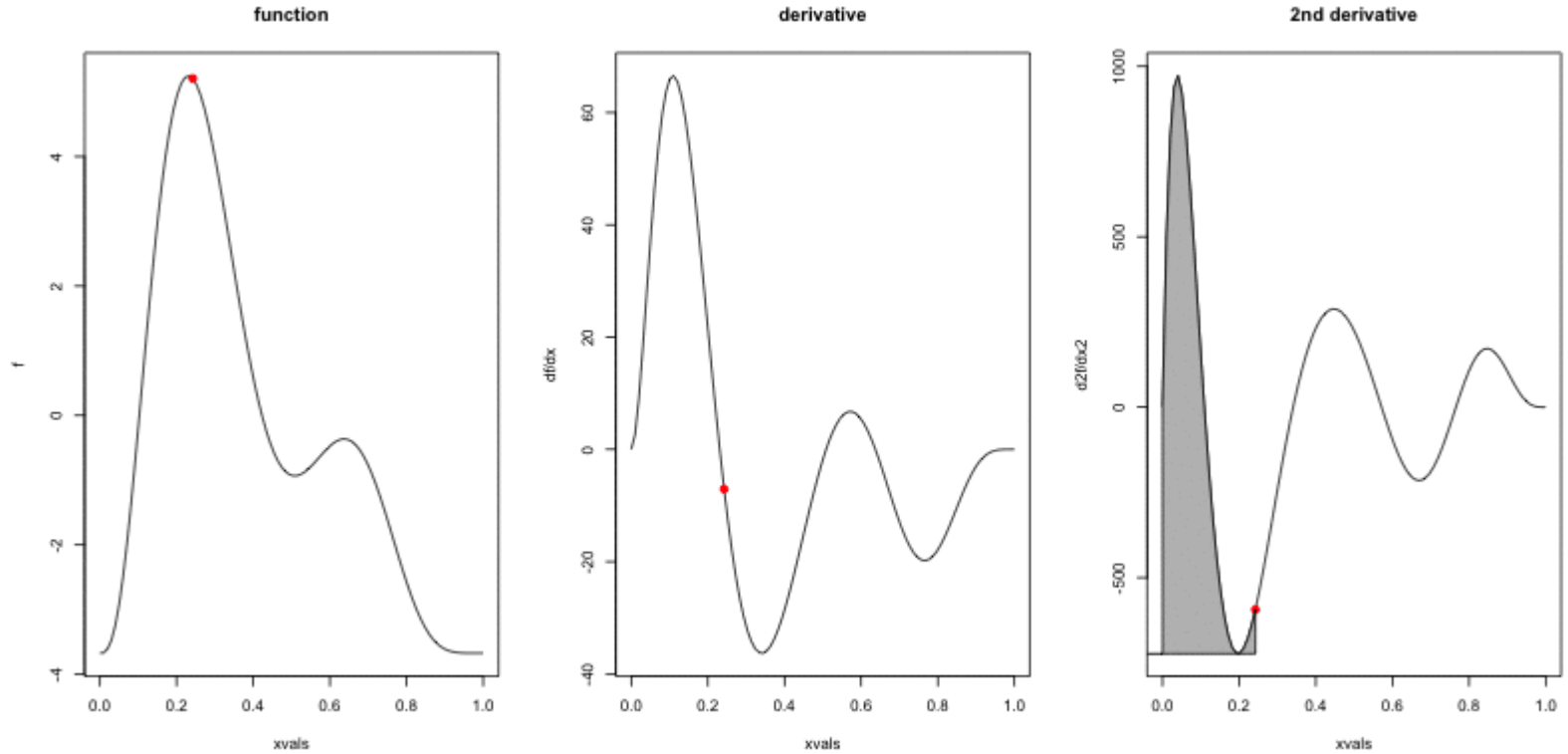
Avoiding overfitting



- Want a line that is *close* to all the data (high likelihood)
- Splines *could* just fit every data point, but this would be overfitting. We know there is *error*
- Easy to overfit — *smooth* curve.
- How do we measure smoothness? Calculus!


WIGGLY
THINGS

Measuring wiggliness



What was that grey bit? Wigglyness

$$\int_{\mathbb{R}} \left(\frac{\partial^2 f(x)}{\partial^2 x} \right)^2 \mathrm{d}x = \boldsymbol{\beta}^{\mathrm{T}} S \boldsymbol{\beta} = W$$

(Wigglyness is 100% the right mathy word)

We penalize wiggliness to avoid overfitting

Making wigglyness matter

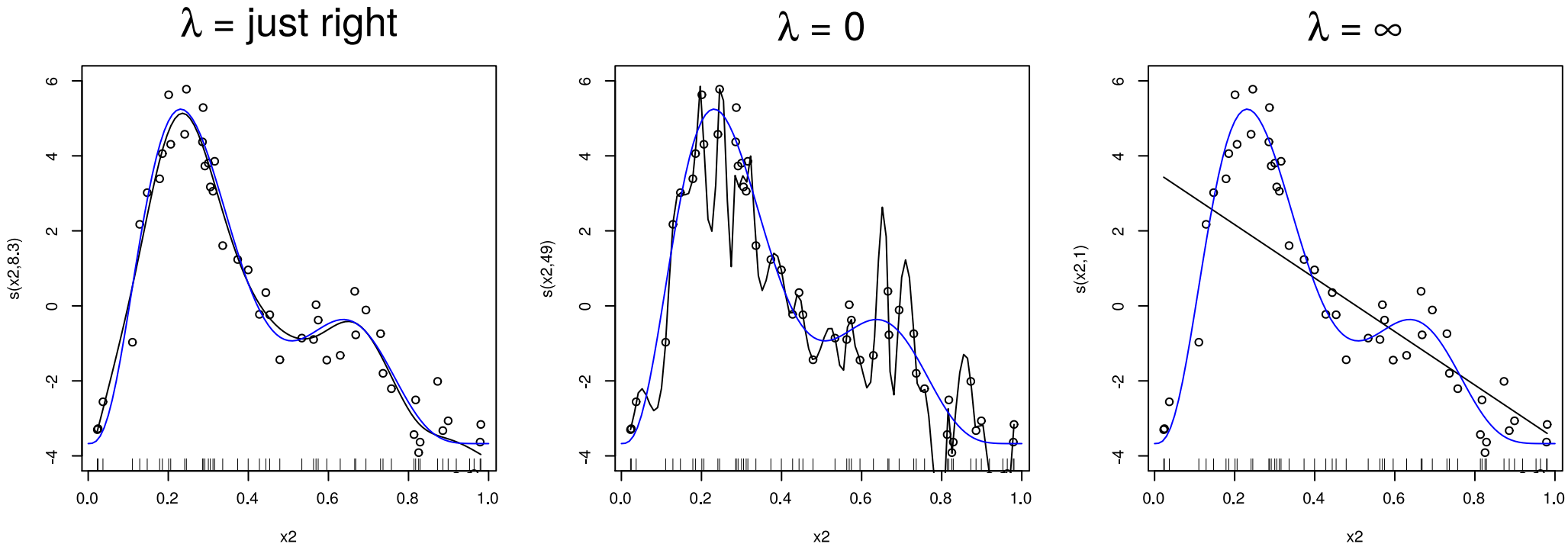
W measures **wigglyness**

(log) likelihood measures closeness to the data

We use a **smoothing parameter** λ to define the trade-off, to find the spline coefficients B_k that maximize the **penalized** log-likelihood

$$\mathcal{L}_p = \log(\text{Likelihood}) - \lambda W$$

Picking the right wiggliness



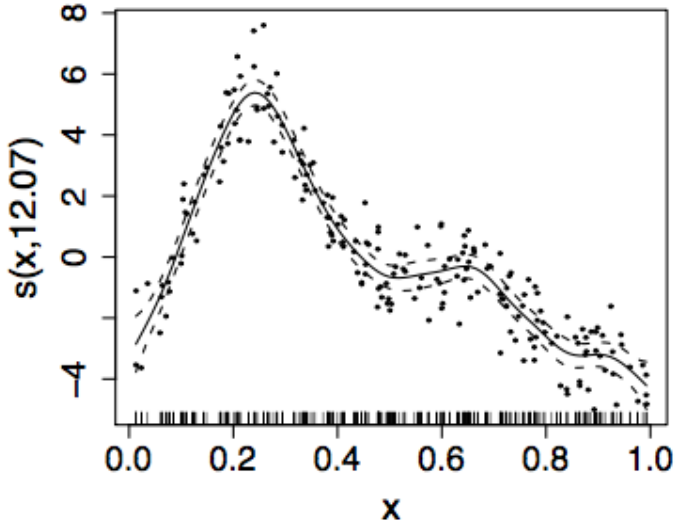
Picking the right wiggliness

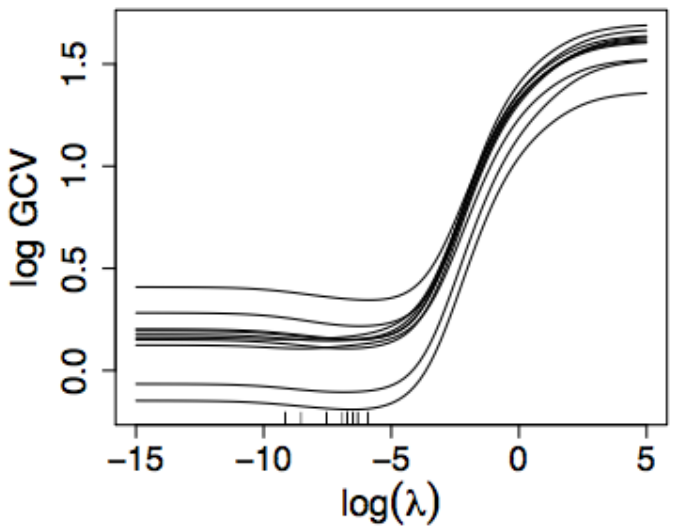
Two ways to think about how to optimize λ:

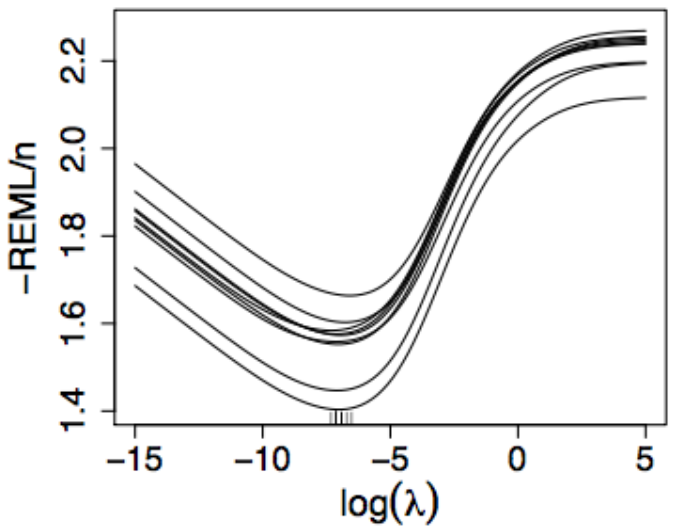
- Predictive: Minimize out-of-sample error
- Bayesian: Put priors on our basis coefficients

Many methods: AIC, Mallow's C_p, GCV, ML, REML

- **Practically**, use **REML**, because of numerical stability
- Hence `gam(..., method="REML")`







Maximum wiggliness

We set **basis complexity** or "size" k

This is *maximum wigglyness*, can be thought of as number of small functions that make up a curve

Once smoothing is applied, curves have fewer **effective degrees of freedom (EDF)**

EDF < k

k must be *large enough*, the λ penalty does the rest

Large enough — space of functions representable by the basis includes the true function or a close approximation to the tru function

Bigger k increases computational cost

In **mgcv**, default k values are arbitrary — after choosing the model terms, this is the key user choice

Must be checked! — `gam.check()`

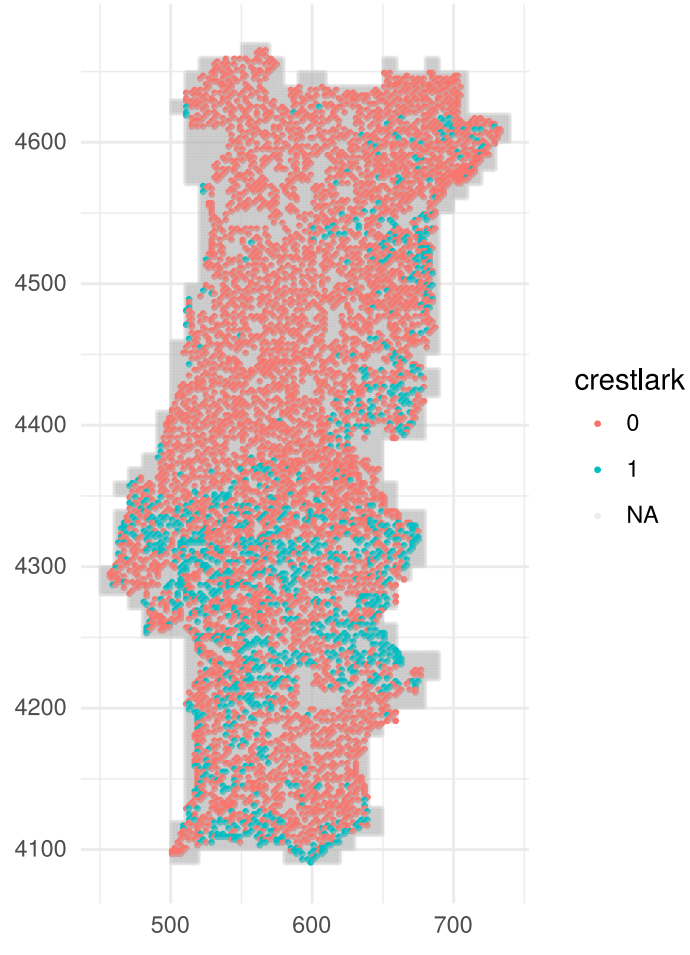
GAM summary so far

1. GAMs give us a framework to model flexible nonlinear relationships
2. Use little functions (**basis functions**) to make big functions (**smooths**)
3. Use a **penalty** to trade off wiggliness/generality
4. Need to make sure your smooths are **wiggly enough**

Portugese larks

```
library('gamair')
data(bird)

bird <- transform(bird,
            crestlark = factor(crestlark),
            linnet = factor(linnet),
            e = x / 1000,
            n = y / 1000)
head(bird)
```

```
      QUADRICULA TET crestlark linnet      x       y   e
13705       NG56   E      <NA>   <NA> 551000 4669000 551
13710       NG56   J      <NA>   <NA> 553000 4669000 553
13715       NG56   P      <NA>   <NA> 555000 4669000 555
13720       NG56   U      <NA>   <NA> 557000 4669000 557
13725       NG56   Z      <NA>   <NA> 559000 4669000 559
13880       NG66   E      <NA>   <NA> 561000 4669000 561
```



Portugese larks — binomial GAM

```
crest <- gam(crestlark ~ s(e, n, k = 100),
             data = bird,
             family = binomial,
             method = 'REML')
```

$s(e, n)$ indicated by `s(e, n)` in the formula

Isotropic thin plate spline

`k` sets size of basis dimension; upper limit on EDF

Smoothness parameters estimated via REML

```
summary(crest)
```

```
Family: binomial
Link function: logit

Formula:
crestlark ~ s(x, y, k = 100)

Parametric coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.24192    0.07743  -28.95   <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(x,y) 74.05  86.62  859.3  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.234   Deviance explained = 25.8%
-REML = 2501.6  Scale est. = 1         n = 6457
```

Portugese larks — binomial GAM

Model checking with binary data is a pain — residuals look weird

Alternatively we can aggregate data at the `QUADRICULA` level & fit a binomial count model

```
## convert back to numeric
bird <- transform(bird,
                  crestlark = as.numeric(as.character(crestlark)),
                  linnet = as.numeric(as.character(linnet)))
## some variables to help aggregation
bird <- transform(bird, tet.n = rep(1, nrow(bird)),
                  N = rep(1, nrow(bird)), stringsAsFactors = FALSE)
## set to NA if not surveyed
bird$N[is.na(as.vector(bird$crestlark))] <- NA
## aggregate
bird2 <- aggregate(data.matrix(bird), by = list(bird$QUADRICULA),
                   FUN = sum, na.rm = TRUE)
## scale by Quads aggregated
bird2 <- transform(bird2, e = e / tet.n, n = n / tet.n)

## fit binomial GAM
crest2 <- gam(cbind(crestlark, N - crestlark) ~ s(e, n, k = 100),
              data = bird2, family = binomial, method = 'REML')
```

Model checking

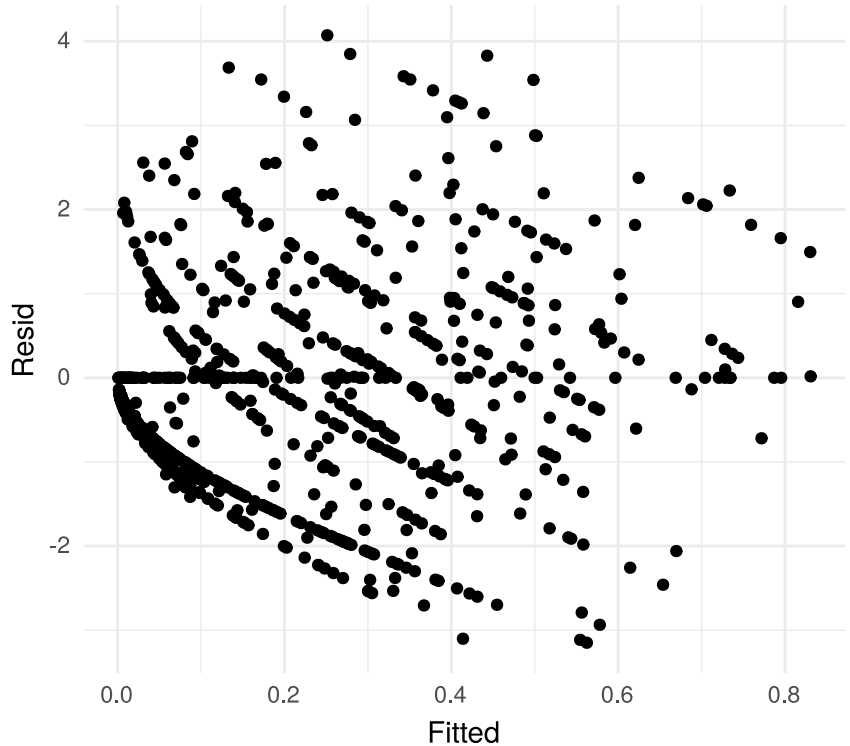
Model residuals don't look too bad

Bands of points due to integers

```
crest3 <- gam(cbind(crestlark, N - crestlark) ~
                  s(e, n, k = 100),
              data = bird2, family = quasibinomial,
              method = 'REML')
```

Some overdispersion — φ = 2.32

```
ggplot(data.frame(Fitted = fitted(crest2),
                  Resid = resid(crest2)),
       aes(Fitted, Resid)) + geom_point()
```



Checking basis size

```
norm_model_1 = gam(y_norm~s(x1,k=4)+s(x2,k=4),method= "REML")
gam.check(norm_model_1)
```

```
Method: REML   Optimizer: outer newton
full convergence after 8 iterations.
Gradient range [-0.0003467788,0.0005154578]
(score 736.9402 & scale 2.252304).
Hessian positive definite, eigenvalue range [0.000346021,198.5041].
Model rank =  7 / 7

Basis dimension (k) checking results. Low p-value (k-index<1) may
indicate that k is too low, especially if edf is close to k'.

        k'  edf k-index p-value
s(x1) 3.00 1.00    0.13  <2e-16 ***
s(x2) 3.00 2.91    1.04    0.79
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Checking basis size

```
norm_model_2 = gam(y_norm~s(x1,k=12)+s(x2,k=4),method= "REML")
gam.check(norm_model_2)
```

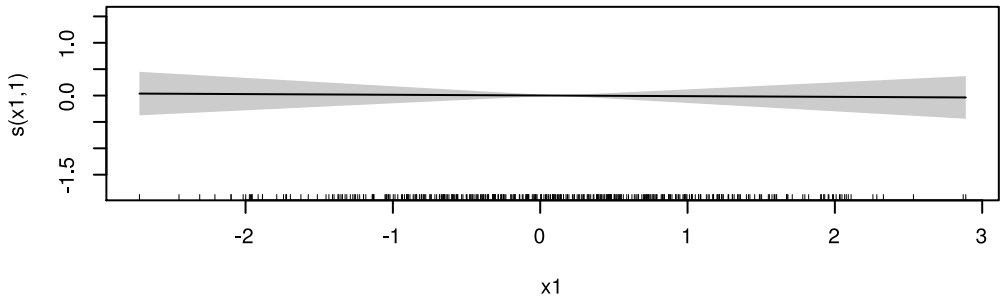
```
Method: REML   Optimizer: outer newton
full convergence after 11 iterations.
Gradient range [-5.658609e-06,5.392657e-06]
(score 345.3111 & scale 0.2706205).
Hessian positive definite, eigenvalue range [0.967727,198.6299].
Model rank =  15 / 15

Basis dimension (k) checking results. Low p-value (k-index<1) may
indicate that k is too low, especially if edf is close to k'.

         k'   edf k-index p-value
s(x1) 11.00 10.84    0.99    0.41
s(x2)  3.00  2.98    0.86  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

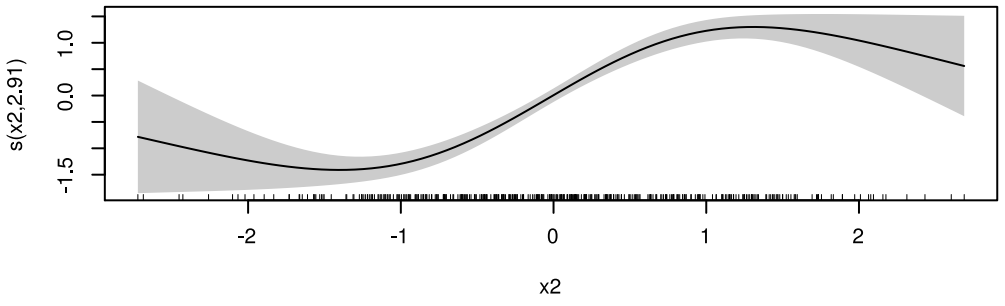
Checking basis size

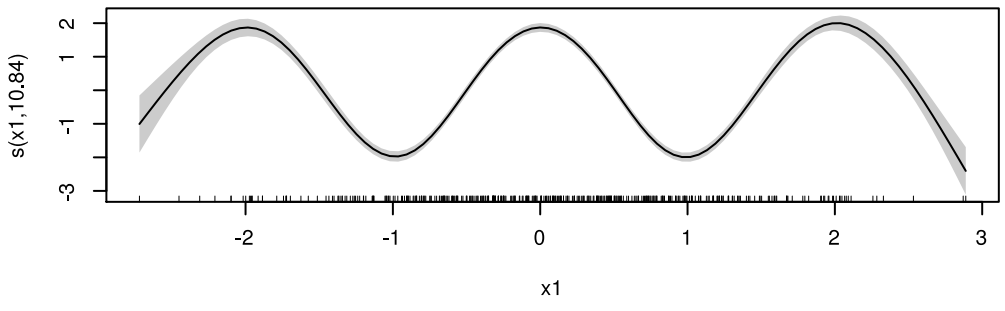
```
norm_model_3 = gam(y_norm~s(x1,k=12)+s(x2,k=12),method= "REML")
gam.check(norm_model_3)
```

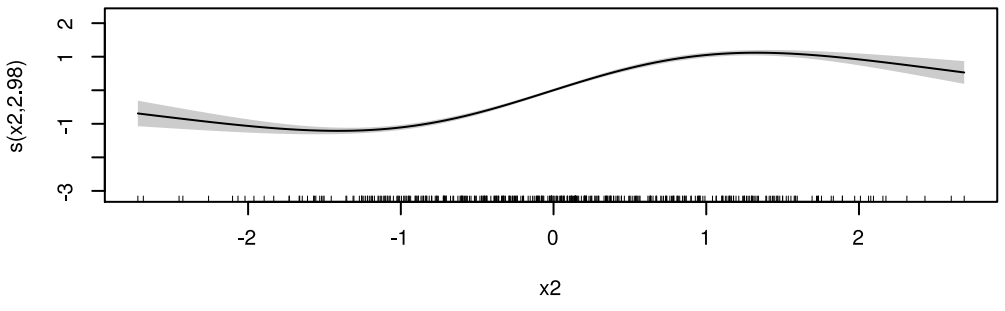
```
Method: REML   Optimizer: outer newton
full convergence after 8 iterations.
Gradient range [-5.103686e-05,-1.089481e-08]
(score 334.2084 & scale 0.2485446).
Hessian positive definite, eigenvalue range [2.812257,198.6868].
Model rank =  23 / 23

Basis dimension (k) checking results. Low p-value (k-index<1) may
indicate that k is too low, especially if edf is close to k'.

         k'   edf k-index p-value
s(x1) 11.00 10.85    0.98    0.28
s(x2) 11.00  7.95    0.95    0.12
```

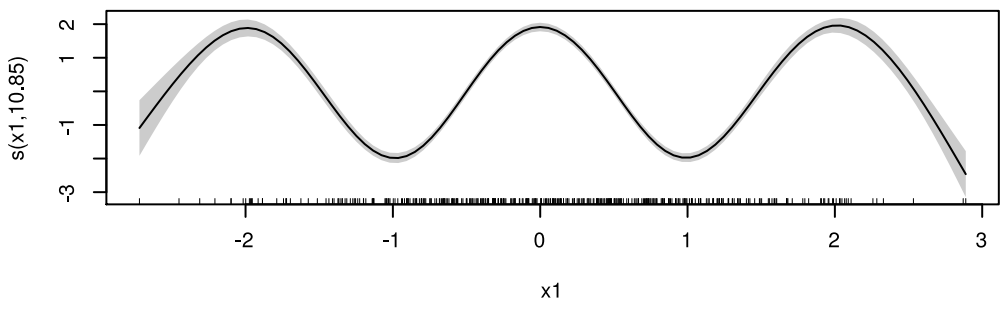
Checking basis size

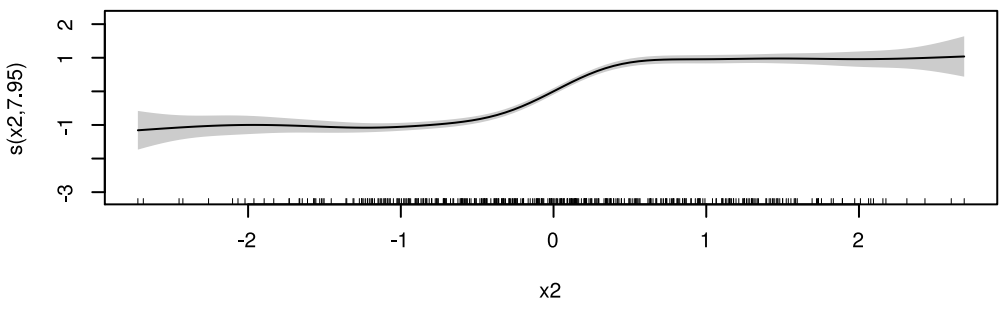












p values for smooths

p values for smooths

p values for smooths are approximate:

1. they don't account for the estimation of λ_j — treated as known, hence *p* values are biased low
2. rely on asymptotic behaviour — they tend towards being right as sample size tends to ∞

The approach described in Wood (2006) is "*no longer recommended*"!

p values for smooths

...are a test of **zero-effect** of a smooth term

Default *p* values rely on theory of Nychka (1988) and Marra & Wood (2012) for confidence interval coverage

If the Bayesian CI have good across-the-function properties, Wood (2013a) showed that the *p* values have

- almost the correct null distribution
- reasonable power

Test statistic is a form of χ^2 statistic, but with complicated degrees of freedom

p values for smooths

Have the best behaviour when smoothness selection is done using ML, then REML.

Neither of these are the default, so remember to use `method = "ML"` or `method = "REML"` as appropriate

anova()

mgcv provides an `anova()` method for `"gam"` objects:

1. Single model form: `anova(m1)`
2. Multi model form: `anova(m1, m2, m3)`

anova() — single model form

This differs from `anova()` methods for `"lm"` or `"glm"` objects:

- the tests are Wald-like tests as described for `summary.gam()` of a $\mathbf{H}_0$ of zero-effect of a smooth term
- these are not *sequential* tests!

```
b1 <- gam(y ~ x0 + s(x1) + s(x2) + s(x3),
          method = "REML")
anova(b1)
```

```
Family: gaussian
Link function: identity

Formula:
y ~ x0 + s(x1) + s(x2) + s(x3)

Parametric Terms:
   df     F  p-value
x0  3 26.94 1.57e-14

Approximate significance of smooth terms:
        edf Ref.df      F  p-value
s(x1) 1.000  1.001 26.682 5.83e-07
s(x2) 6.694  7.807 18.755  < 2e-16
s(x3) 1.000  1.000  0.068    0.795
```

anova() — multi model form

The multi-model form should really be used with care — the *p* values are really *approximate*

```
b1 <- gam(y ~ s(x0) + s(x1) + s(x2) + s(x3) + s(x4) + s(x5),
          data = dat, family=poisson, method = "ML")
b2 <- update(b1, . ~ . - s(x3) - s(x4) - s(x5))
anova(b2, b1, test = "Chisq")
```

```
Analysis of Deviance Table

Model 1: y ~ s(x0) + s(x1) + s(x2)
Model 2: y ~ s(x0) + s(x1) + s(x2) + s(x3) + s(x4) + s(x5)
  Resid. Df Resid. Dev     Df Deviance Pr(>Chi)
1    186.23     248.97
2    183.34     248.01 2.8959  0.96184    0.795
```

Don't used for testing random effect splines!

AIC for GAMs

- Comparison of GAMs by a form of AIC is an alternative frequentist approach to model selection
- Rather than using the marginal likelihood, the likelihood of the β_j *conditional* upon $\boldsymbol{\lambda}_j$ is used, with the EDF replacing $\boldsymbol{k}$, the number of model parameters
- This *conditional* AIC tends to select complex models, especially those with random effects, as the EDF ignores that $\boldsymbol{\lambda}_j$ are estimated
- Wood et al (2016) suggests a correction that accounts for uncertainty in $\boldsymbol{\lambda}_j$

$$AIC = -2\mathcal{L}(\hat{\beta}) + 2\mathrm{tr}(\widehat{\mathcal{I}}V_{\beta}')$$

AIC

In this example, x_3, x_4, and x_5 have no effects on y

```
AIC(b1, b2)
```

```
         df      AIC
b1 15.03493 847.7961
b2 12.12435 842.9368
```

When there is *no difference* in compared models, accepts larger model ~16% of the time: consistent with probability AIC chooses a model with 1 extra spurious parameter $Pr(\chi^2_1 > 2)$

```
pchisq(2, 1, lower.tail = FALSE)
```

```
[1] 0.1572992
```

Next steps

Read Simon Wood's book

Tonnes more material on our ESA GAM Workshop site

https://noamross.github.io/mgcv-esa-workshop/

Also see my blog: www.fromthebottomoftheheap.net

References

- Marra & Wood (2011) *Computational Statistics and Data Analysis* **55** 2372--2387.
- Marra & Wood (2012) *Scandinavian Journal of Statistics, Theory and Applications* **39**(1), 53--74.
- Nychka (1988) *Journal of the American Statistical Association* **83**(404) 1134--1143.
- Wood (2006, 2017) *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC. (New 2nd Edition)
- Wood (2013a) *Biometrika* **100**(1) 221--228.
- Wood (2013b) *Biometrika* **100**(4) 1005--1010.
- Wood et al (2016) *JASA* **111** 1548--1563