Introduction to Generalized Additive Models with R and mgcv

Gavin Simpson

1000–1230 CST (1600–1830 UTC) July 30th, 2020

Welcome

Logistics

Slides

Slidedeck: bit.ly/gam-webinar Sources: bit.ly/gam-webinar-git

Direct download a ZIP of everything: bit.ly/gam-webinar-zip

Unpack the zip & remember where you put it

Q & A

Add questions to Google Doc: bit.ly/gam-webinar-qa

Recording

Livestream will be recorded — youtu.be/sgw4cu8hrZM

Donation

In lieu of not charging for this webinar, if you are able to, financially, please make a donation to the University of Regina's **Student Emergency Fund**

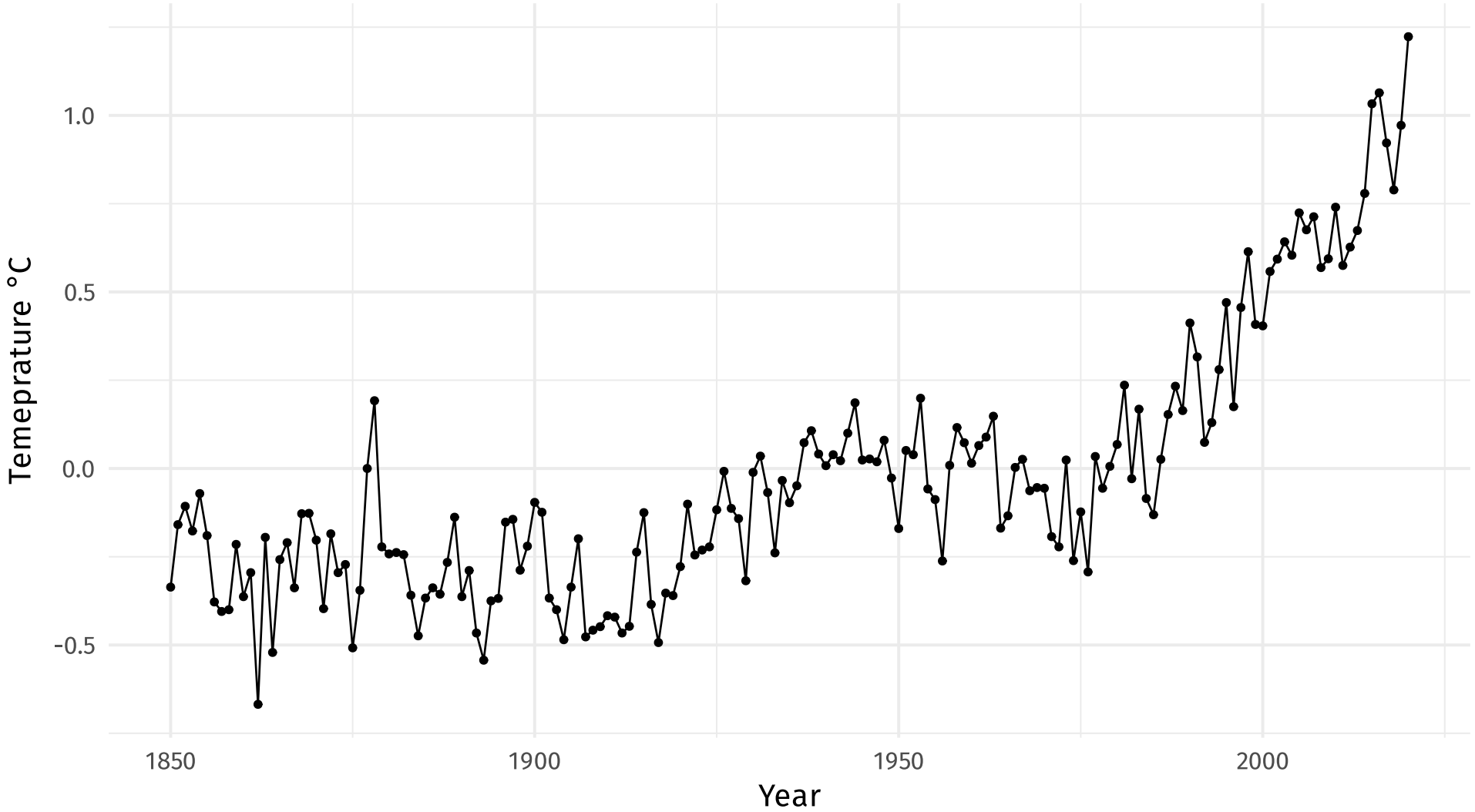
https://giving.uregina.ca/student-emergency-fund

Today's topics

- What are GAMs?
- How to fit GAMs in R with **mgcv**
- Model Checking
- Model Diagnostics
- Examples

Motivating example

HadCRUT4 time series



Linear Models

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

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

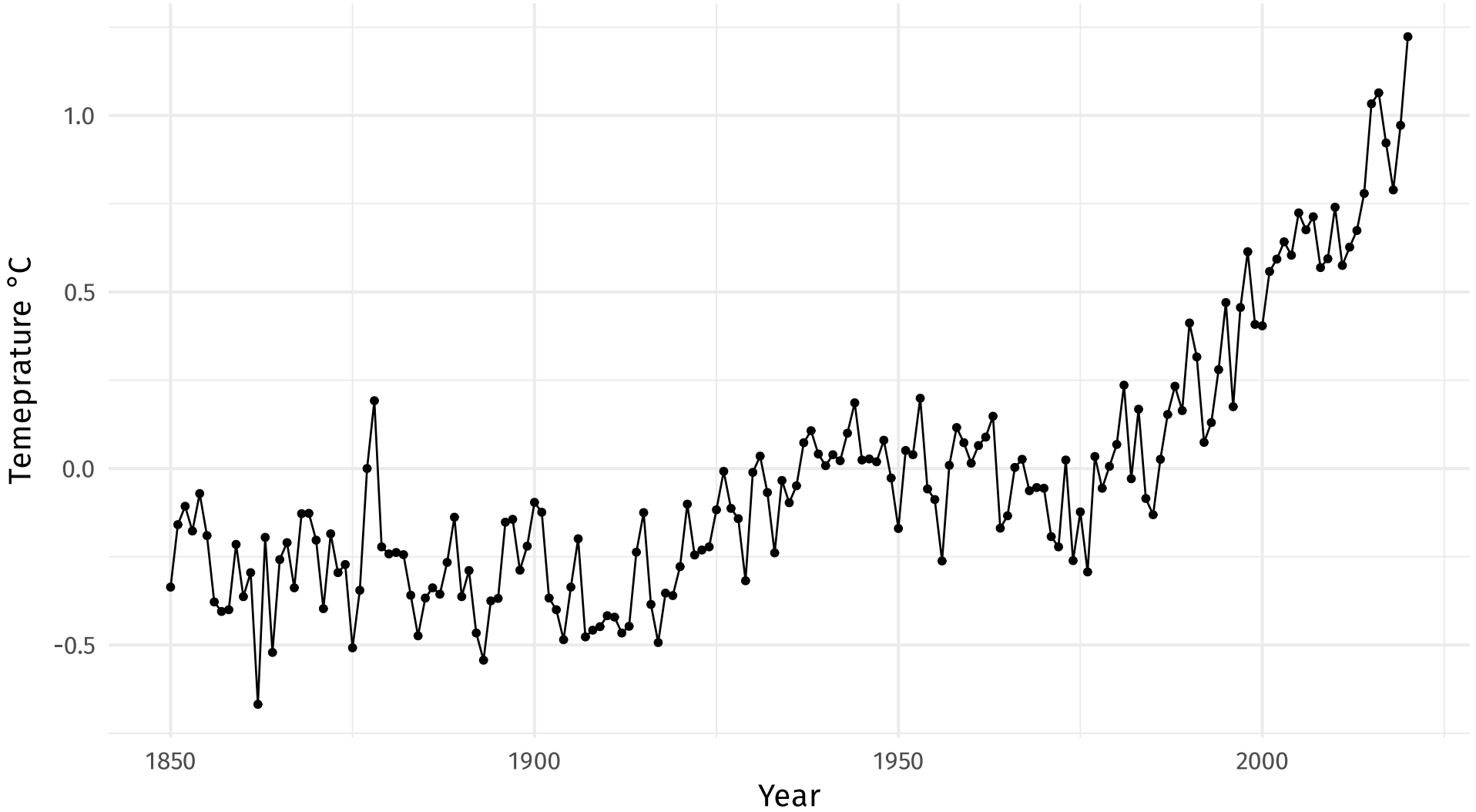
Assumptions

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

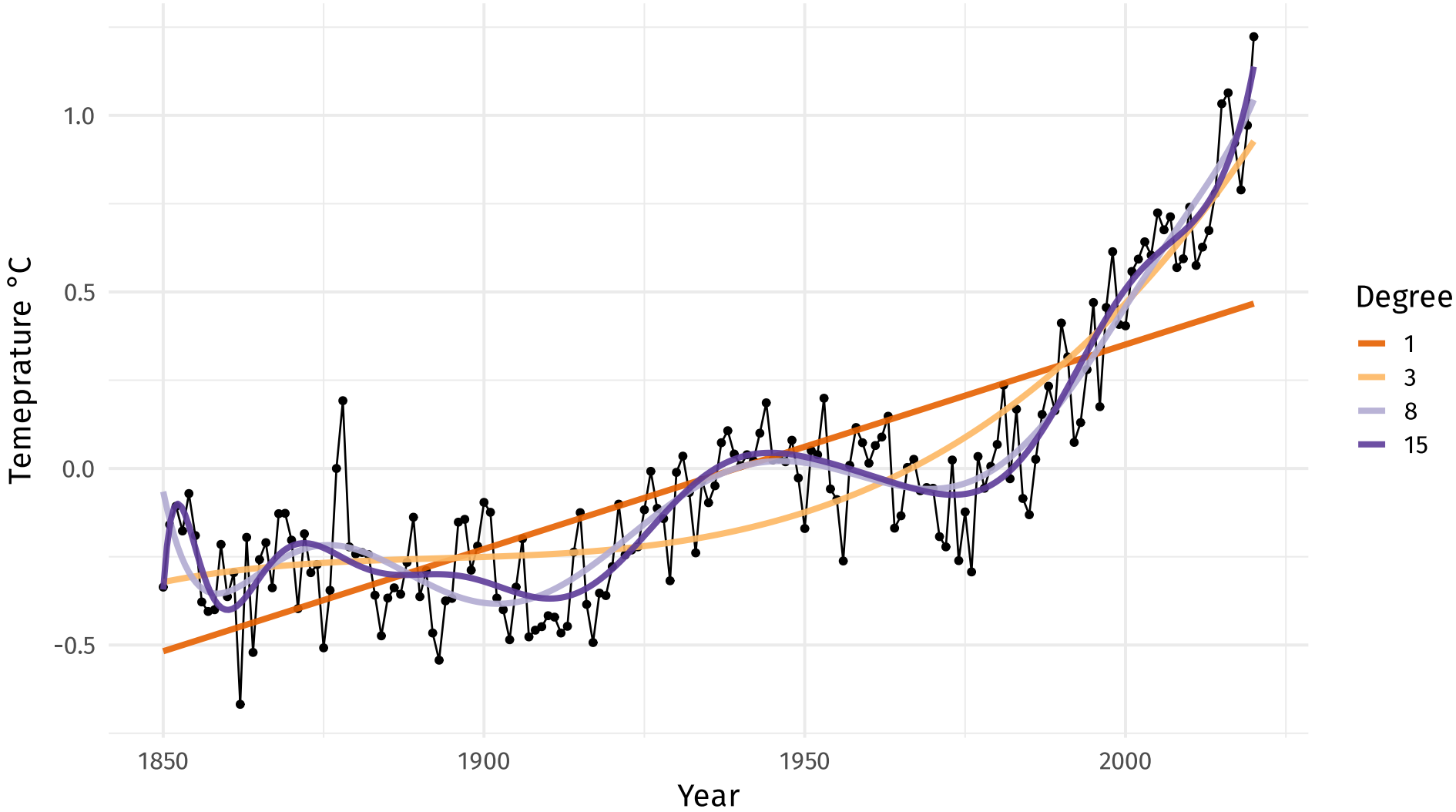
An **additive** model address the first of these

Why bother with anything more complex?

Is this linear?



Polynomials perhaps...



Polynomials perhaps...

We can keep on adding ever more powers of x to the model — model selection problem

Runge phenomenon — oscillations at the edges of an interval — means simply moving to higher-order polynomials doesn't always improve accuracy

GAMs offer a solution

HadCRUT data set

```
library('readr')
library('dplyr')
URL ←  "https://bit.ly/hadcrutv4"
gtemp ← read_delim(URL, delim = ' ', col_types = 'nnnnnnnnnnnn', col_names = FALSE) %>%
    select(num_range('X', 1:2)) %>% setNames(nm = c('Year', 'Temperature'))
```

File format

HadCRUT data set

```
gtemp
```

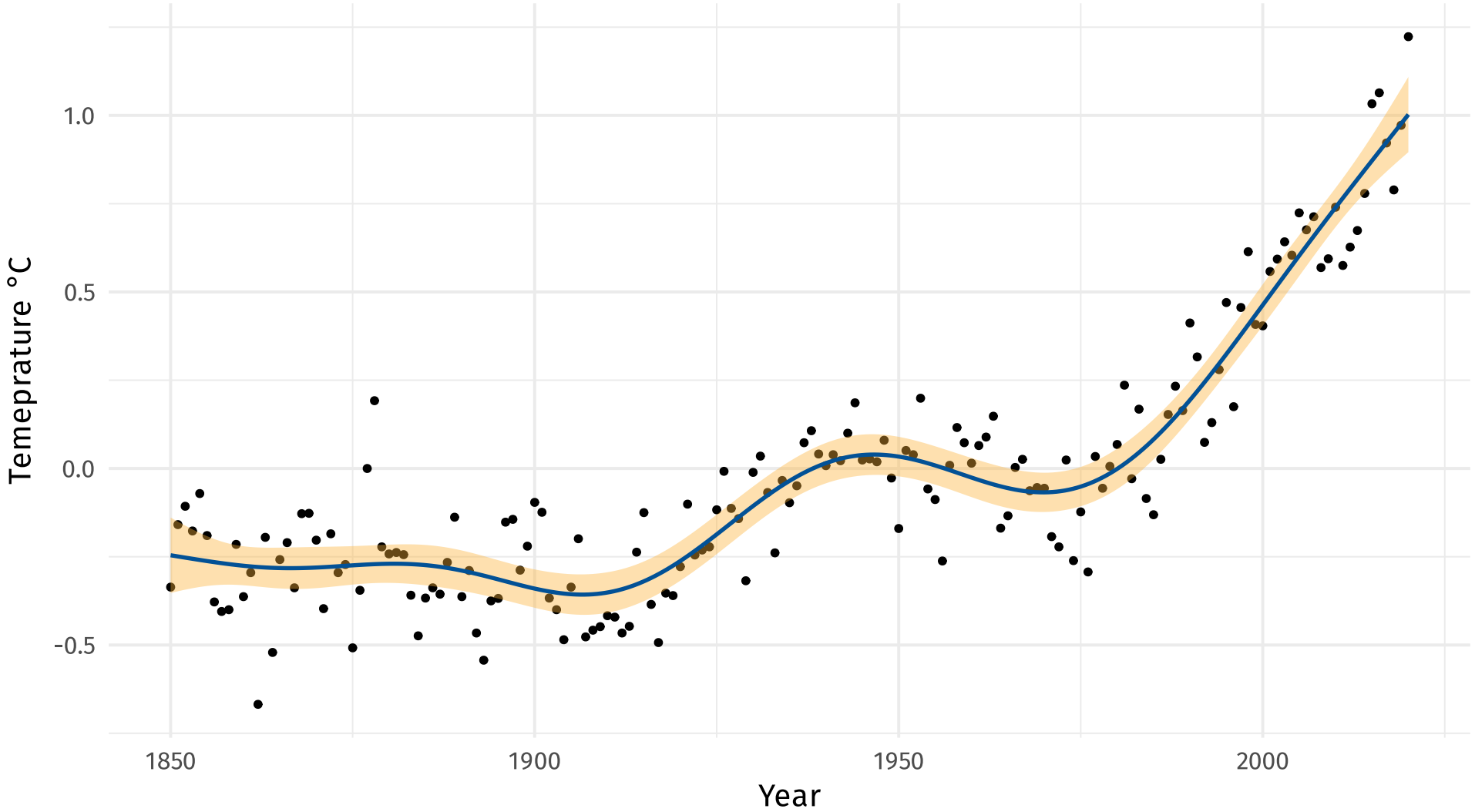
```
## # A tibble: 171 x 2
##     Year Temperature
##    <dbl>       <dbl>
##  1  1850      -0.336
##  2  1851      -0.159
##  3  1852      -0.107
##  4  1853      -0.177
##  5  1854      -0.071
##  6  1855      -0.19
##  7  1856      -0.378
##  8  1857      -0.405
##  9  1858      -0.4
## 10  1859      -0.215
## # … with 161 more rows
```

Fitting a GAM

```
library('mgcv')
m ← gam(Temperature ~ s(Year), data = gtemp, method = 'REML')
summary(m)
```

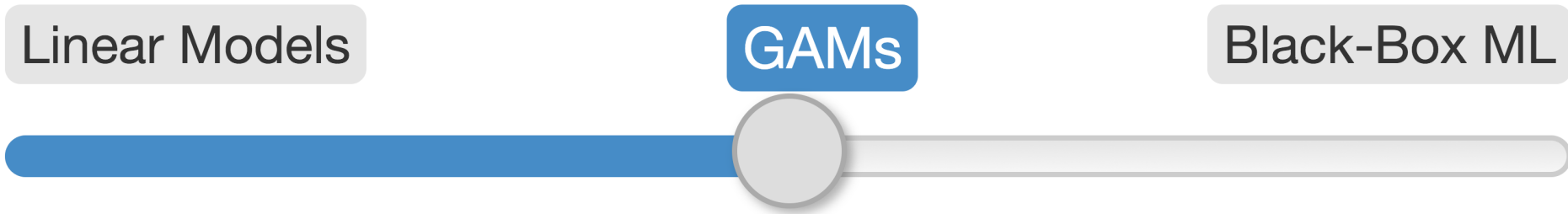
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Temperature ~ s(Year)
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.025327   0.009847  -2.572    0.011 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##           edf Ref.df     F p-value
## s(Year) 7.784   8.62 138.5  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.875   Deviance explained = 88.1%
## -REML = -89.311  Scale est. = 0.016581  n = 171
```

Fitted GAM



GAMs

Generalized Additive Models



Source: GAMs in R by Noam Ross

How is a GAM different?

In LM 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

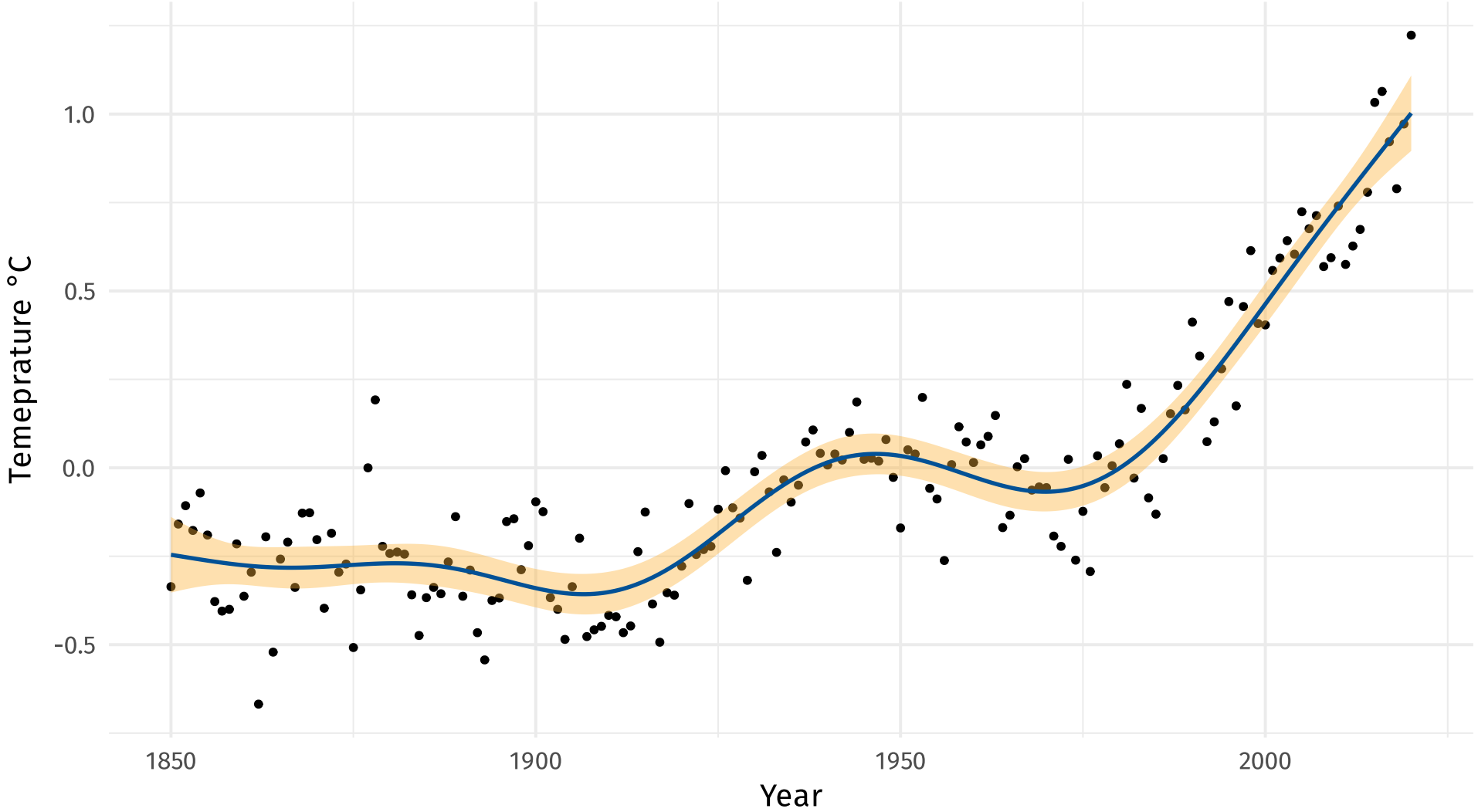
Fitting a GAM in R

```
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*

How did `gam( )` *know*?

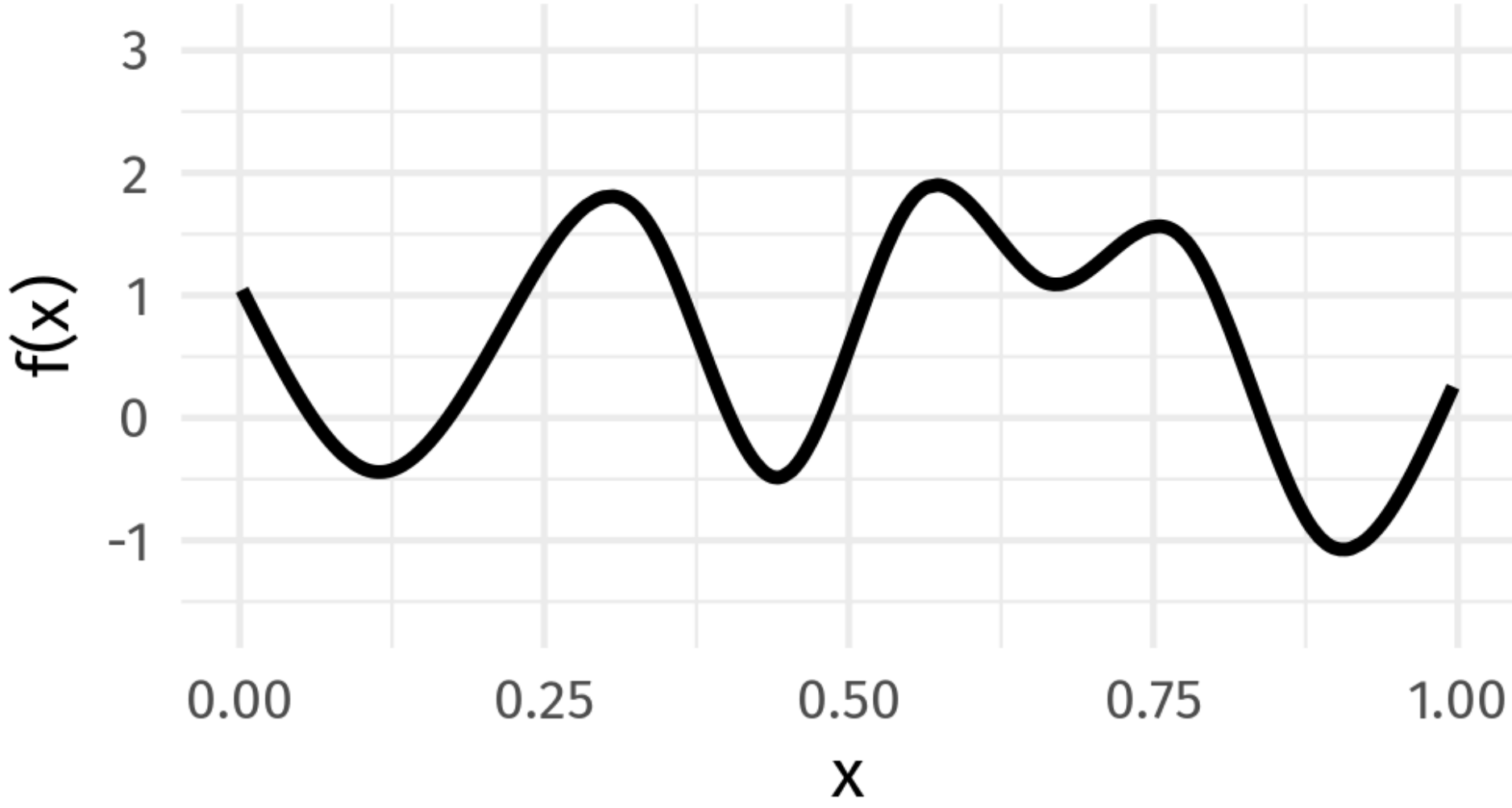


What magic is this?

Rob Potter


WIGGLY
THINGS

Wiggly things



Basis expansions

In the polynomial models we used a polynomial basis expansion of $\boldsymbol{x}$

- $\boldsymbol{x}^0 = \mathbf{1}$ — the model constant term
- $\boldsymbol{x}^1 = \boldsymbol{x}$ — linear term
- $\boldsymbol{x}^2$
- $\boldsymbol{x}^3$
- ...

Splines

Splines are *functions* composed of simpler functions

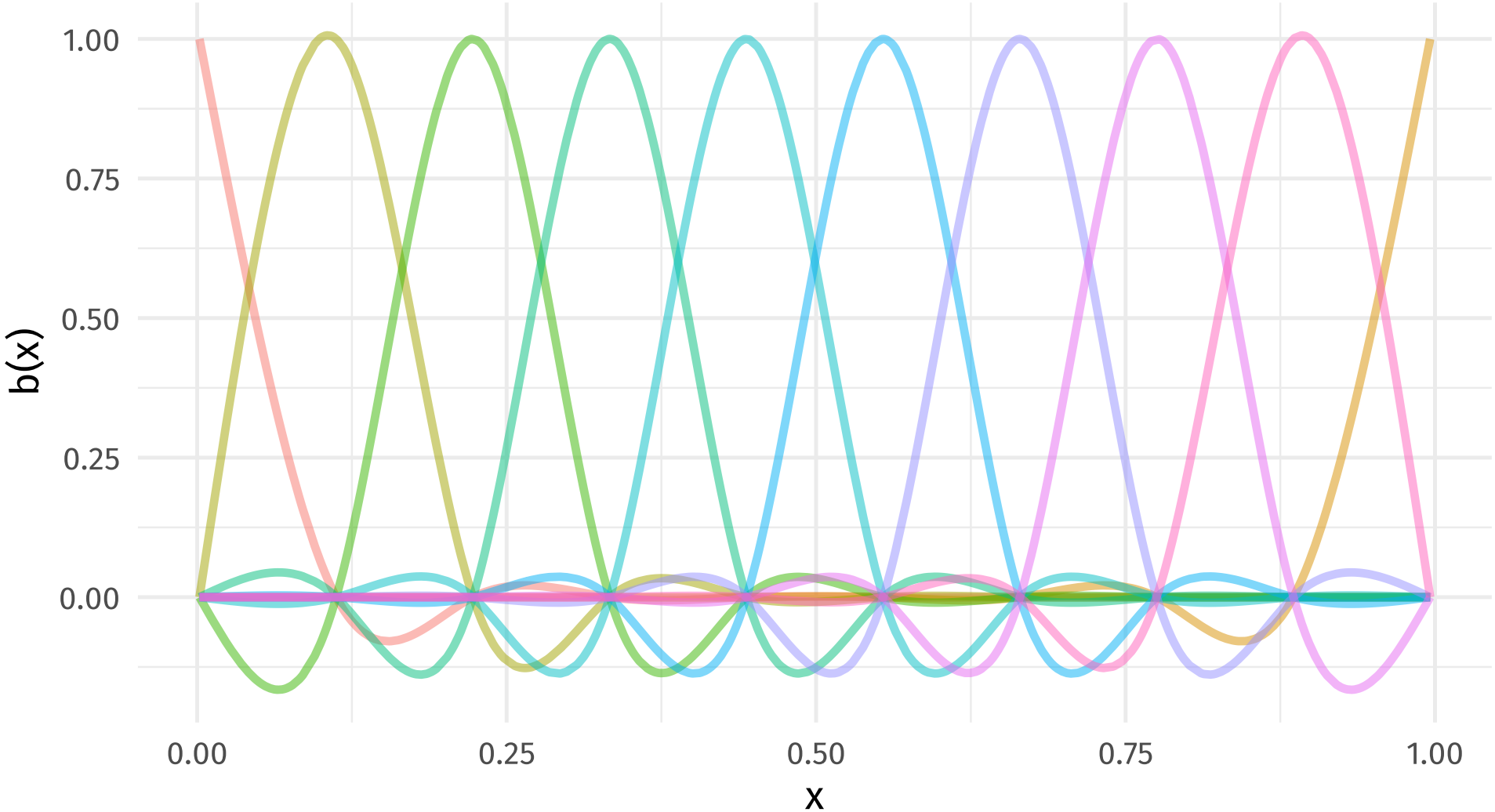
Simpler functions are *basis functions* & the set of basis functions is a *basis*

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

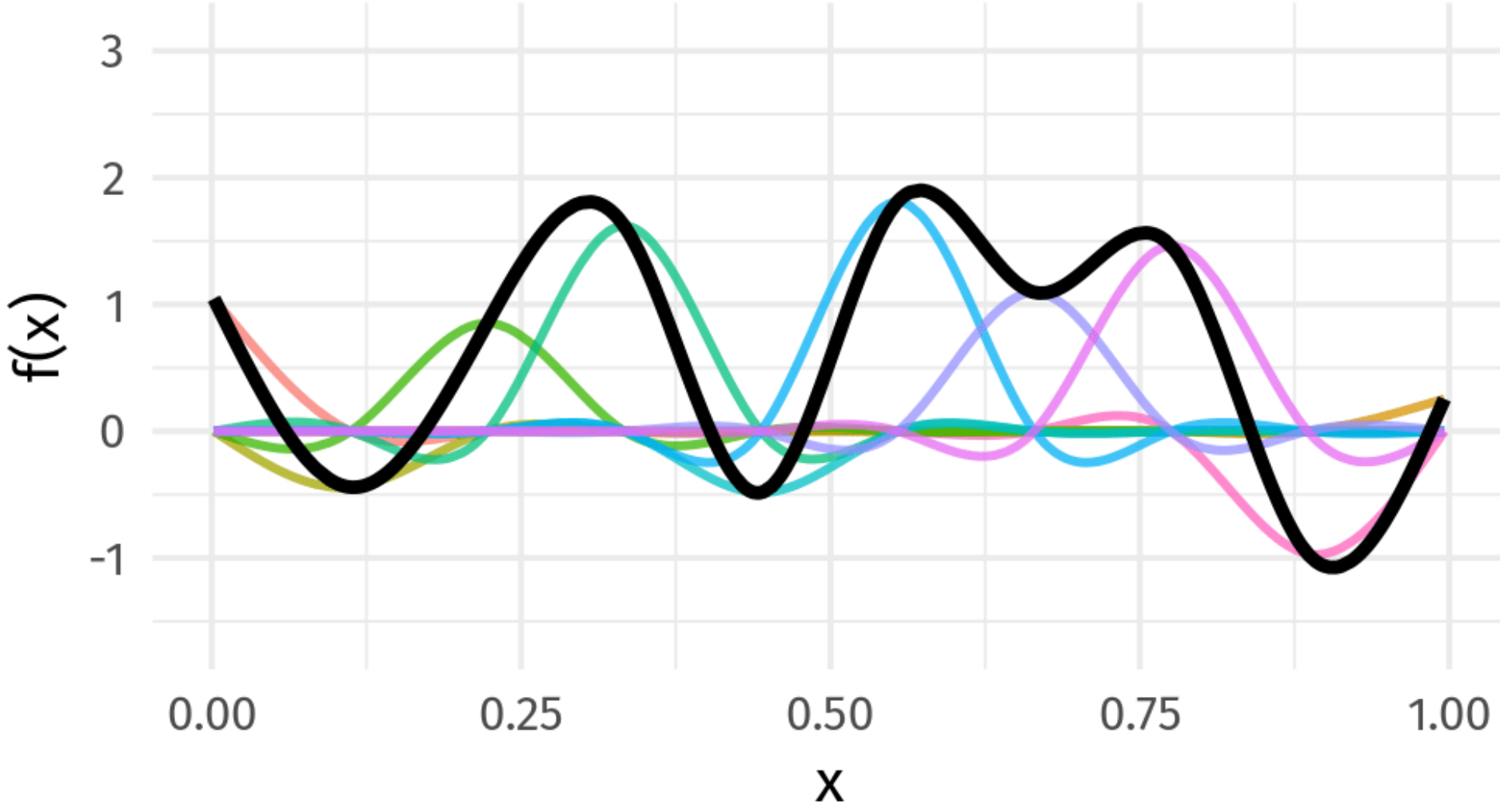
Resultant spline is a the sum of these weighted basis functions, evaluated at the values of x

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

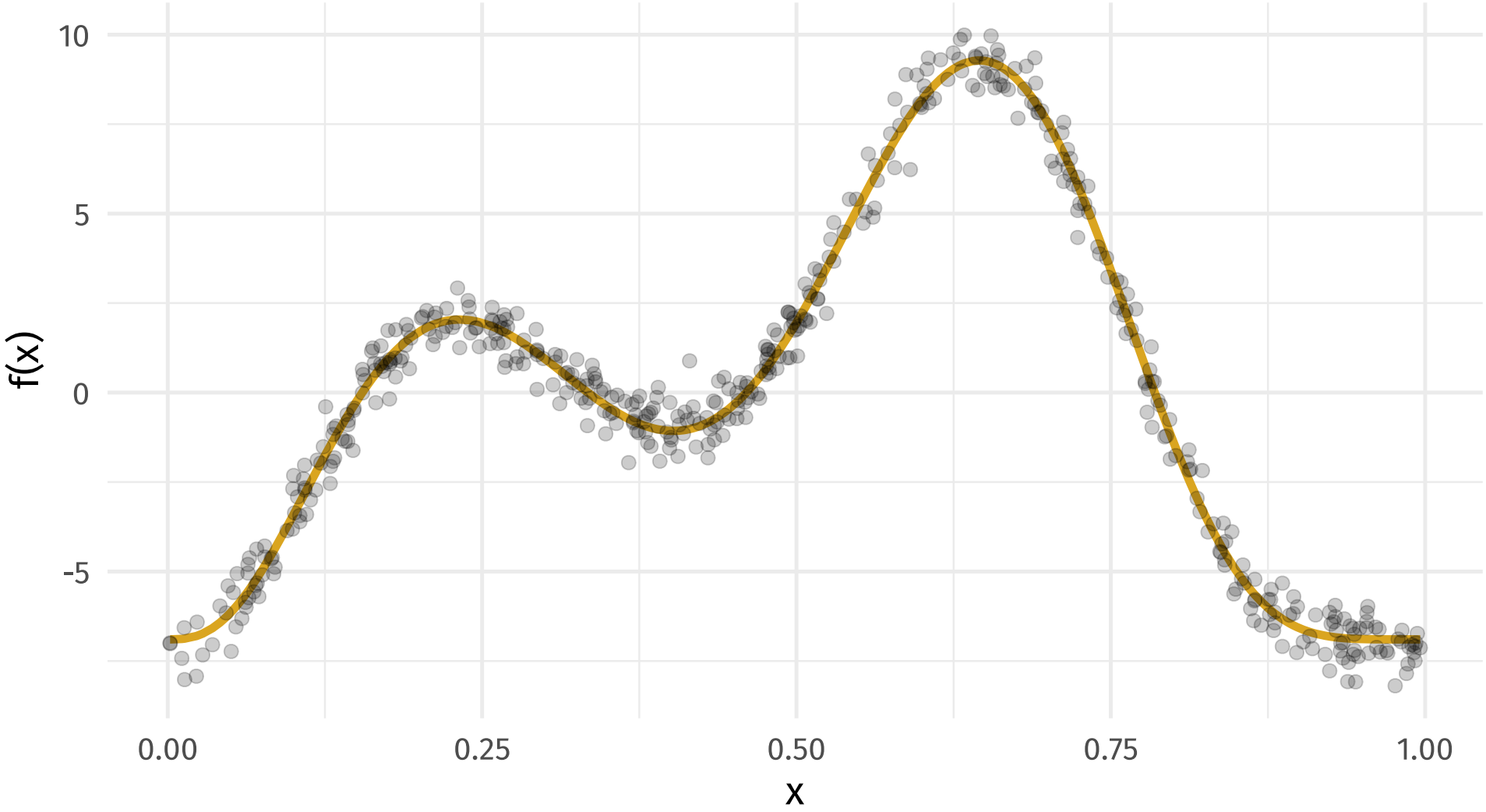
Splines formed from basis functions



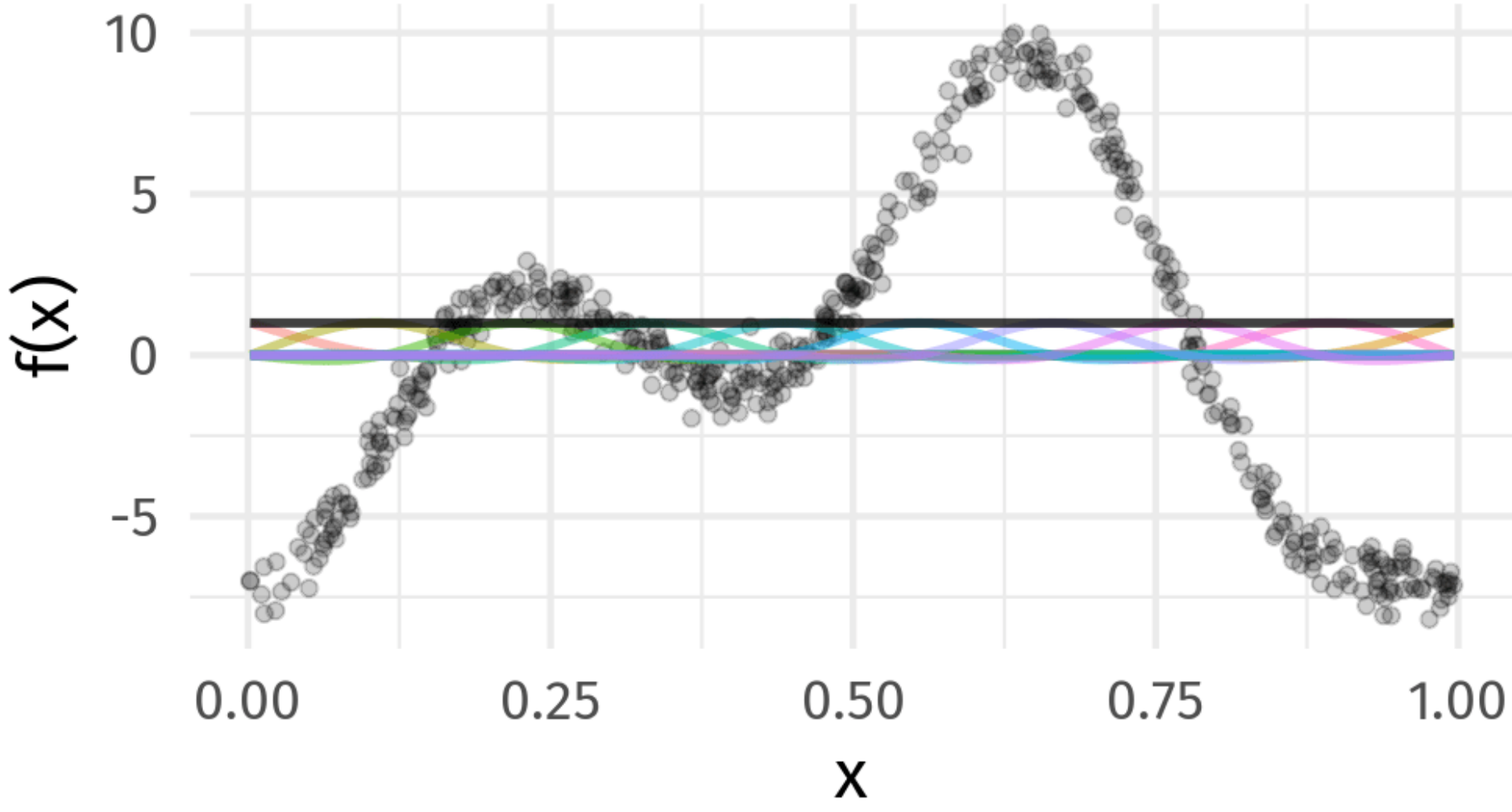
Weight basis functions ⇨ spline



How do GAMs learn from data?



Maximise penalised log-likelihood ⇨ β



Avoid overfitting our sample

How wiggly?

$$\int_{\mathbb{R}} [f'']^2 dx = \beta^{\mathsf{T}} \mathbf{S} \beta$$

Penalised fit

$$\mathcal{L}_p(\boldsymbol{\beta}) = \mathcal{L}(\boldsymbol{\beta}) - \frac{1}{2}\lambda\boldsymbol{\beta}^\mathsf{T}\mathbf{S}\boldsymbol{\beta}$$

Wiggliness

$$\int_{\mathbb{R}} [f'']^2 dx = \boldsymbol{\beta}^\mathsf{T} \mathbf{S} \boldsymbol{\beta} = W$$

(Wiggliness is 100% the right mathy word)

We penalize wiggliness to avoid overfitting

Making wiggliness matter

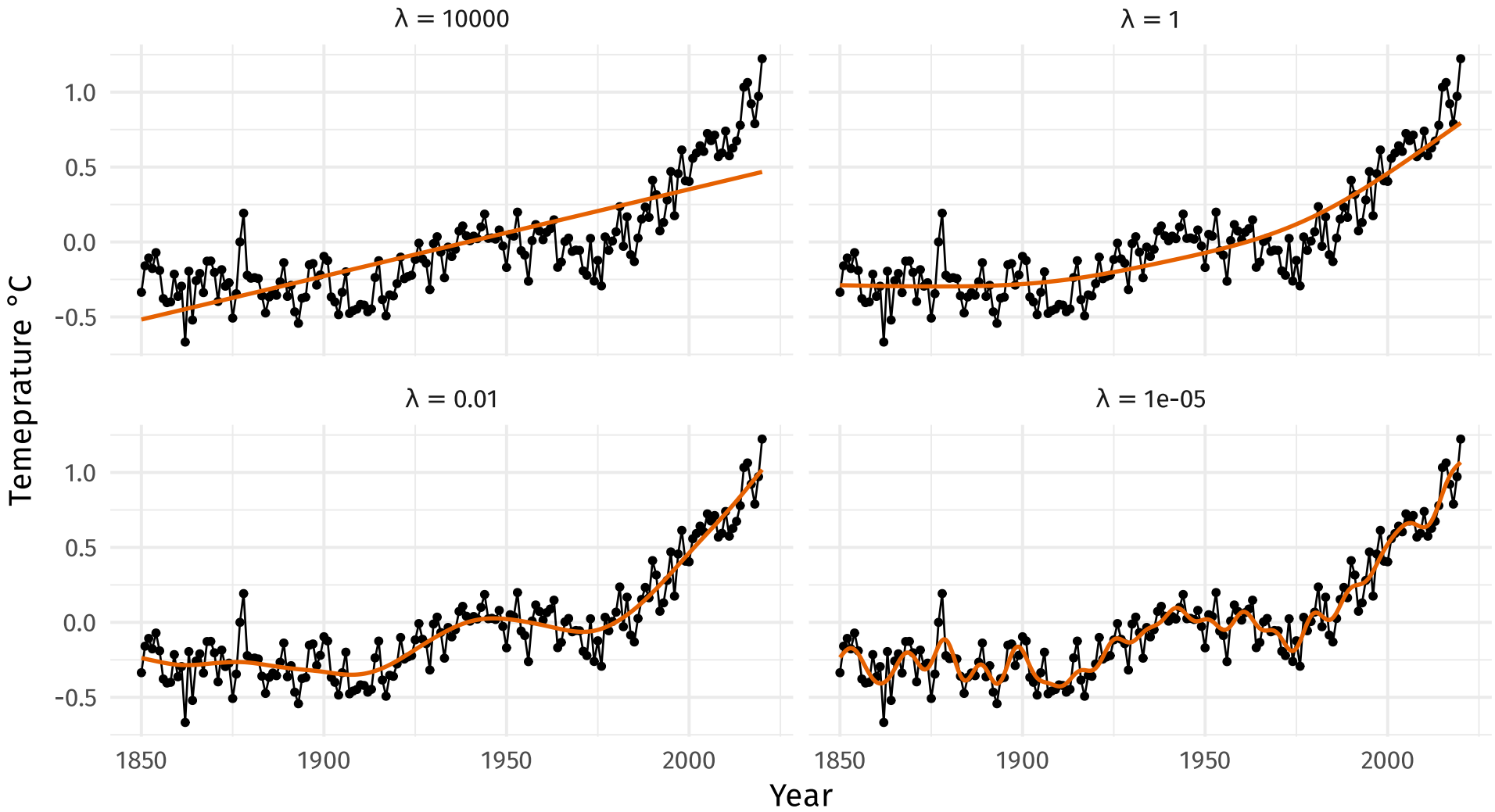
W measures **wiggliness**

(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

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

HadCRUT4 time series



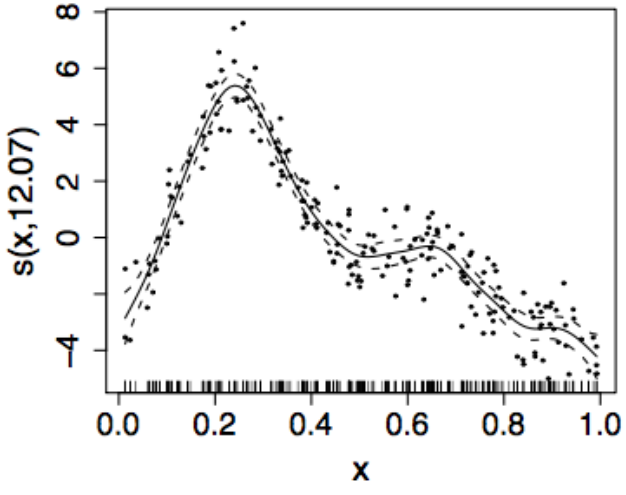
Picking the right wiggliness

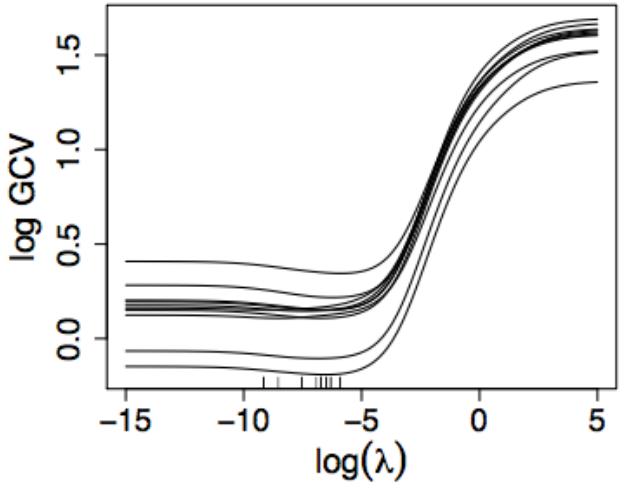
Two ways to think about how to optimize λ:

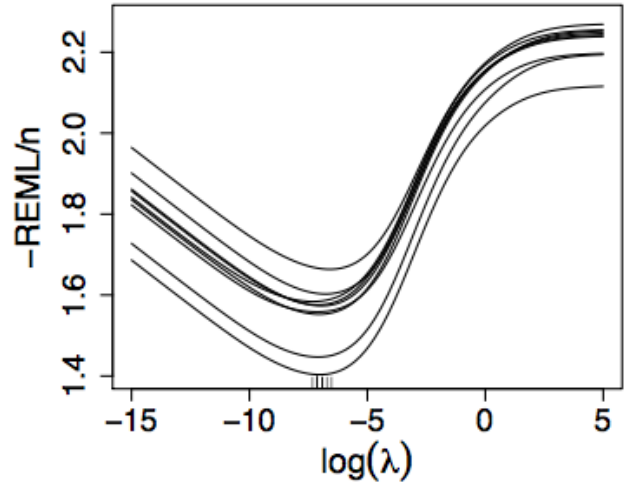
- Predictive: Minimize out-of-sample error
- Bayesian: Put priors on our basis coefficients

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

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







Maximum allowed 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

Maximum allowed wiggliness

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

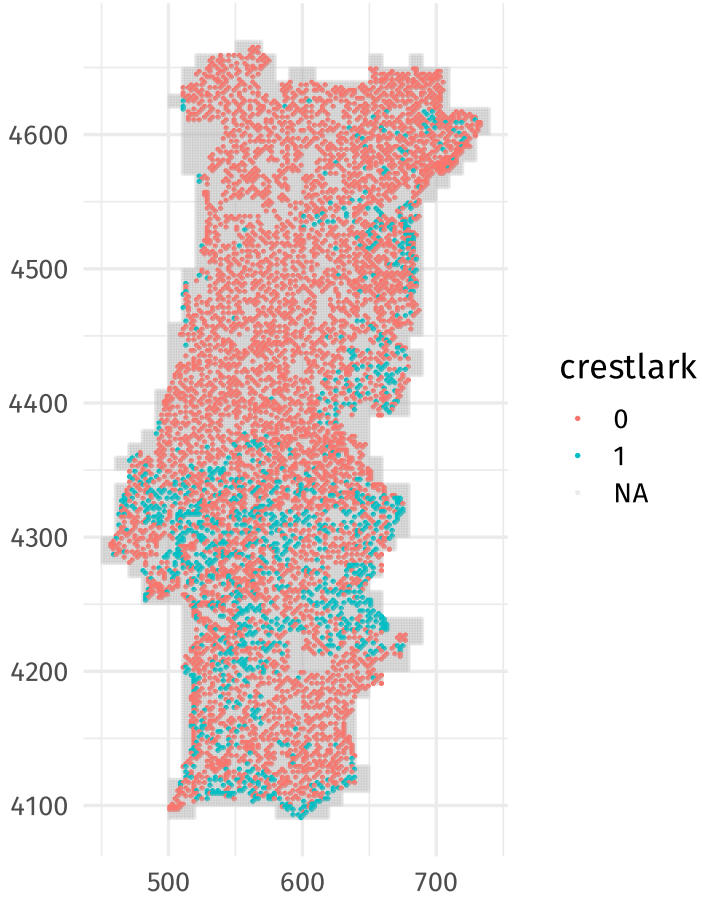
1. GAMs give us a framework to model flexible nonlinear relationships
2. Use little functions (**basis functions**) to make big functions (**smooths**)
3. Use a **penalty** to trade off wiggliness/generality
4. Need to make sure your smooths are **wiggly enough**

Example

Portugese larks

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

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



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(e, n, k = 100)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.24184    0.07785   -28.8   <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(e,n) 74.04  86.46  857.3  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
' ' 1
##
## R-sq.(adj) =  0.234   Deviance explained = 25.9%
## -REML = 2499.8  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

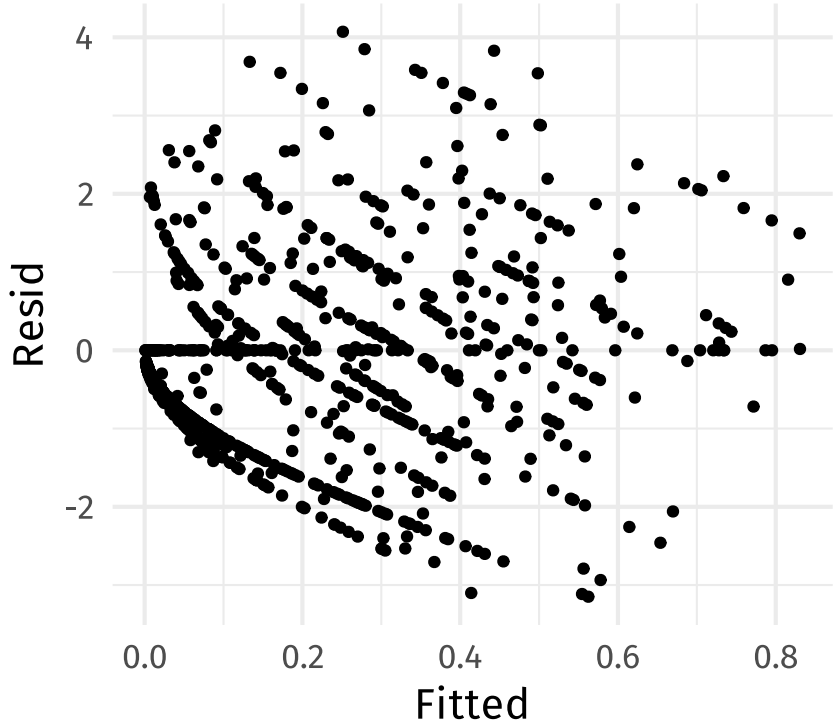
```
crest3 ← gam(cbind(crestlark, N - crestlark) ~
                 s(e, n, k = 100),
             data = bird2, family = quasibinomial,
             method = 'REML')
```

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

Model residuals don't look too bad

Bands of points due to integers

Some overdispersion — $\varphi = 2.32$



A cornucopia of smooths

A cornucopia of smooths

The type of smoother is controlled by the `bs` argument (think *basis*)

The default is a low-rank thin plate spline `bs = 'tp'`

Many others available

- Cubic splines `bs = 'cr'`
- P splines `bs = 'ps'`
- Cyclic splines `bs = 'cc'` or `bs = 'cp'`
- Adaptive splines `bs = 'ad'`
- Random effect `bs = 're'`
- Factor smooths `bs = 'fs'`
- Duchon splines `bs = 'ds'`
- Spline on the sphere `bs = 'sos'`
- MRFs `bs = 'mrf'`
- Soap-film smooth `bs = 'so'`
- Gaussian process `bs = 'gp'`

A bestiary of conditional distributions

A GAM is just a fancy GLM

Simon Wood & colleagues (2016) have extended the *mgcv* methods to some non-exponential family distributions

- `binomial()`
- `poisson()`
- `Gamma()`
- `inverse.gaussian()`
- `nb()`
- `tw()`
- `mvn()`
- `multinom()`
- `betar()`
- `scat()`
- `gaulss()`
- `ziplss()`
- `twlss()`
- `cox.ph()`
- `gamals()`
- `ocat()`

Smooth interactions

Two ways to fit smooth interactions

1. Bivariate (or higher order) thin plate splines
   - `s(x, z, bs = 'tp')`
   - Isotropic; single smoothness parameter for the smooth
   - Sensitive to scales of `x` and `z`
2. Tensor product smooths
   - Separate marginal basis for each smooth, separate smoothness parameters
   - Invariant to scales of `x` and `z`
   - Use for interactions when variables are in different units
   - `te(x, z)`

Tensor product smooths

There are multiple ways to build tensor products in *mgcv*

1. `te(x, z)`
2. `t2(x, z)`
3. `s(x) + s(z) + ti(x, z)`

`te()` is the most general form but not usable in `gamm4::gamm4()` or *brms*

`t2()` is an alternative implementation that does work in `gamm4::gamm4()` or *brms*

`ti()` fits pure smooth interactions; where the main effects of `x` and `z` have been removed from the basis

Factor smooth interactions

Two ways for factor smooth interactions

1. `by` variable smooths
   - entirely separate smooth function for each level of the factor
   - each has it's own smoothness parameter
   - centred (no group means) so include factor as a fixed effect
   - `y ~ f + s(x, by = f)`
2. `bs = 'fs'` basis
   - smooth function for each level of the function
   - share a common smoothness parameter
   - fully penalized; include group means
   - closer to random effects
   - `y ~ s(x, f, bs = 'fs')`

Random effects

When fitted with REML or ML, smooths can be viewed as just fancy random effects

Inverse is true too; random effects can be viewed as smooths

If you have simple random effects you can fit those in `gam()` and `bam()` without needing the more complex GAMM functions `gamm()` or `gamm4::gamm4()`

These two models are equivalent

```
m_nlme ← lme(travel ~ 1, data = Rail, ~ 1 | Rail, method = "REML")

m_gam  ← gam(travel ~ s(Rail, bs = "re"), data = Rail, method = "REML")
```

Random effects

The random effect basis `bs = 're'` is not as computationally efficient as *nlme* or *lme4* for fitting

- complex random effects terms, or
- random effects with many levels

Instead see `gamm()` and `gamm4::gamm4()`

- `gamm()` fits using `lme()`
- `gamm4::gamm4()` fits using `lmer()` or `glmer()`

For non Gaussian models use `gamm4::gamm4()`

Model checking

Model checking

So you have a GAM:

- How do you know you have the right degrees of freedom?
  `gam.check()`
- Diagnosing model issues: `gam.check()` part 2

GAMs are models too

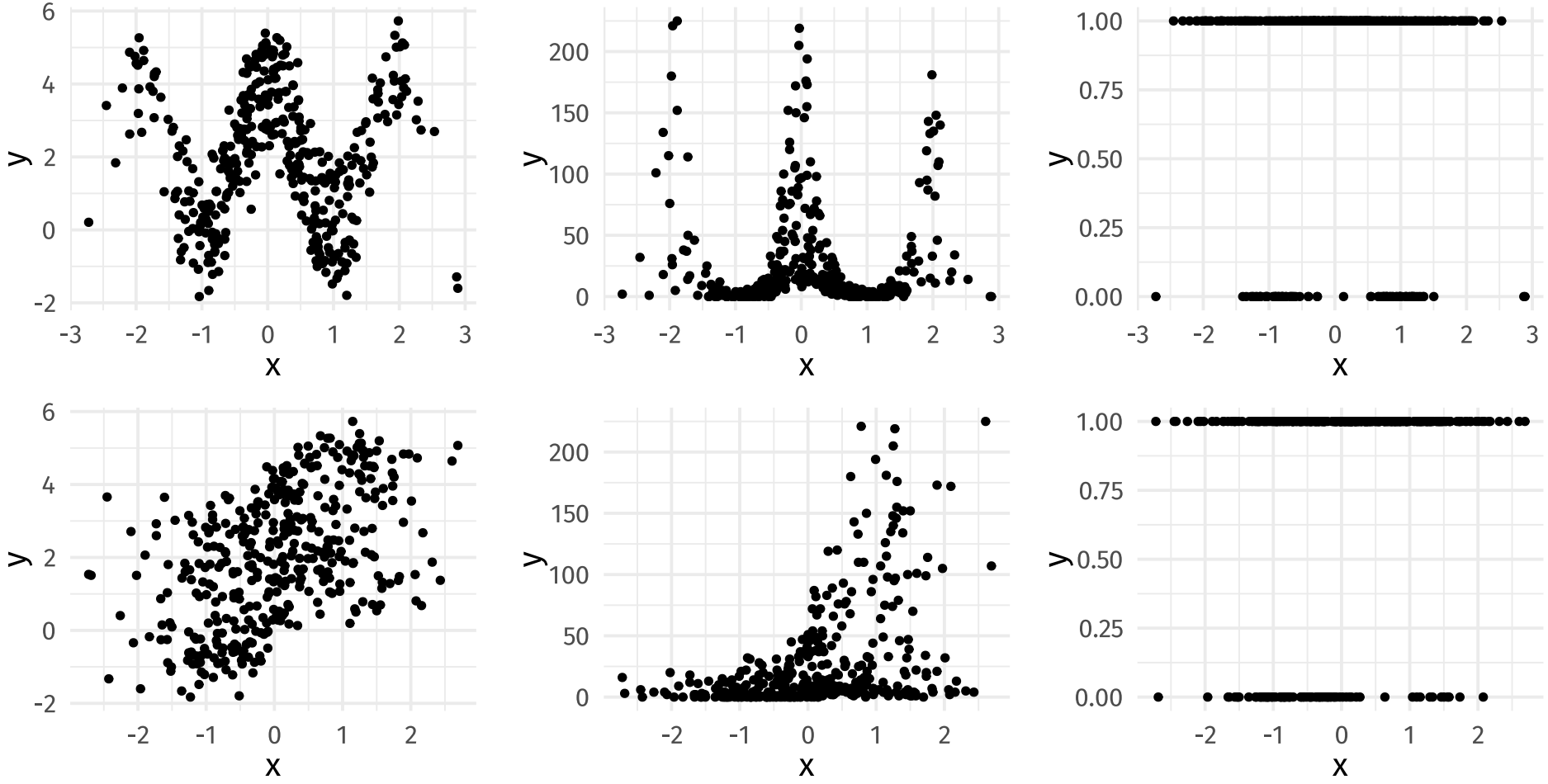
How accurate your predictions will be depends on how good the model is

How do we test how well our model fits?

Simulated data

```
set.seed(2)
n ← 400
x1 ← rnorm(n)
x2 ← rnorm(n)
y_val ← 1 + 2*cos(pi*x1) + 2/(1+exp(-5*(x2)))
y_norm ← y_val + rnorm(n, 0, 0.5)
y_negbinom ← rnbinom(n, mu = exp(y_val),size=10)
y_binom ← rbinom(n,1,prob = exp(y_val)/(1+exp(y_val)))
```

Simulated data



gam.check() part 1: do you have the right functional form?

How well does the model fit?

- Many choices: k, family, type of smoother, ...
- How do we assess how well our model fits?

Basis size *k*

- Set `k` per term
- e.g. `s(x, k=10)` or `s(x, y, k=100)`
- Penalty removes "extra" wigglyness
  - *up to a point!*
- (But computation is slower with bigger `k`)

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    0.83
## ---
## 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)
```

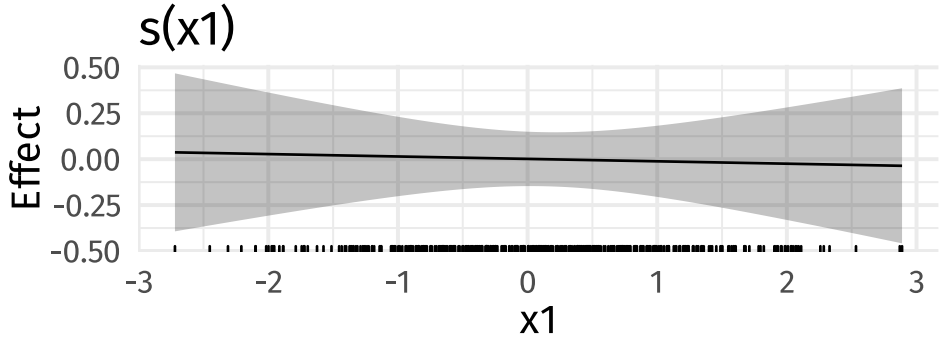
```
##
## Method: REML   Optimizer: outer newton
## full convergence after 11 iterations.
## Gradient range [-5.658609e-06,5.392657e-06]
## (score 345.3111 & scale 0.2706205).
## Hessian positive definite, eigenvalue range [0.967727,198.6299].
## Model rank =  15 / 15
##
## Basis dimension (k) checking 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    0.38
## s(x2)  3.00  2.98    0.86    0.01 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

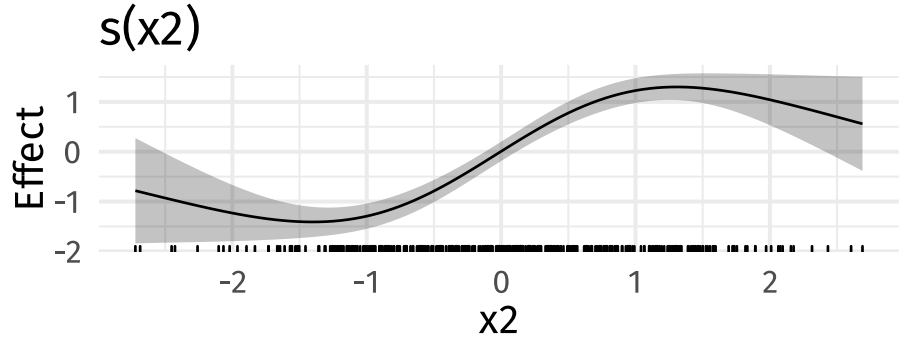
Checking basis size

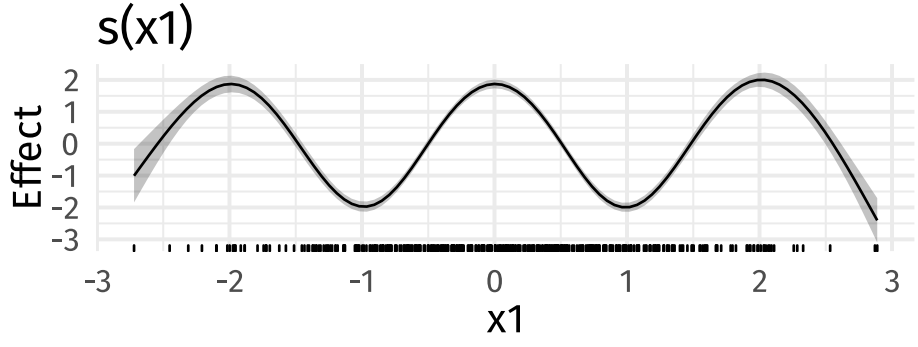
```
norm_model_3 ← gam(y_norm ~ s(x1, k = 12) + s(x2, k = 12),method = 'REML')
gam.check(norm_model_3)
```

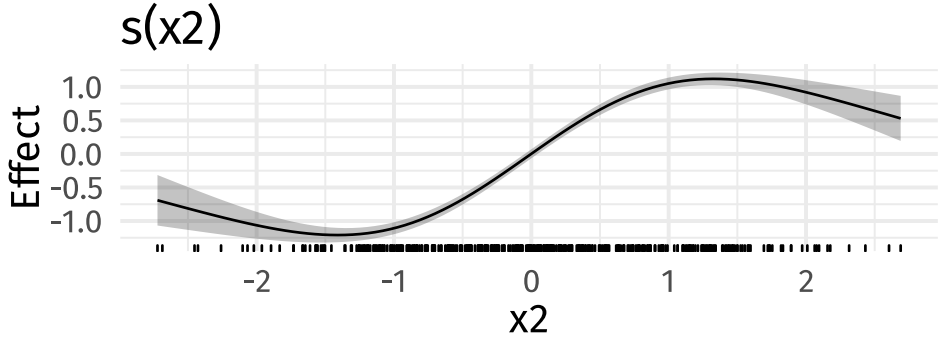
```
##
## Method: REML   Optimizer: outer newton
## full convergence after 8 iterations.
## Gradient range [-1.136192e-08,6.812328e-13]
## (score 334.2084 & scale 0.2485446).
## Hessian positive definite, eigenvalue range [2.812271,198.6868].
## Model rank =  23 / 23
##
## Basis dimension (k) checking 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    0.31
## s(x2) 11.00  7.95    0.95    0.15
```

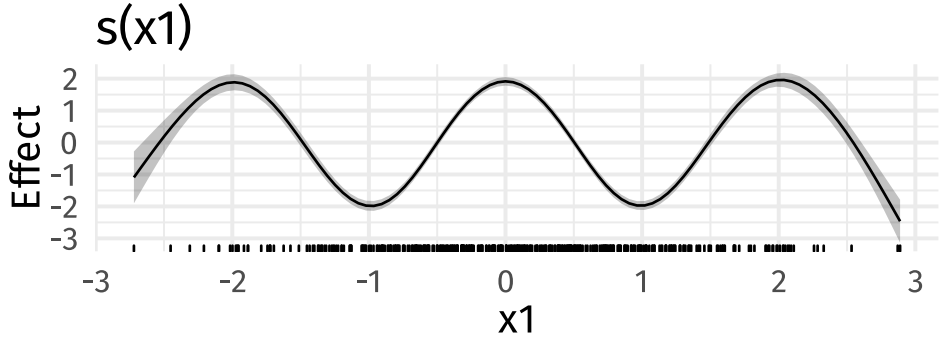
Checking basis size

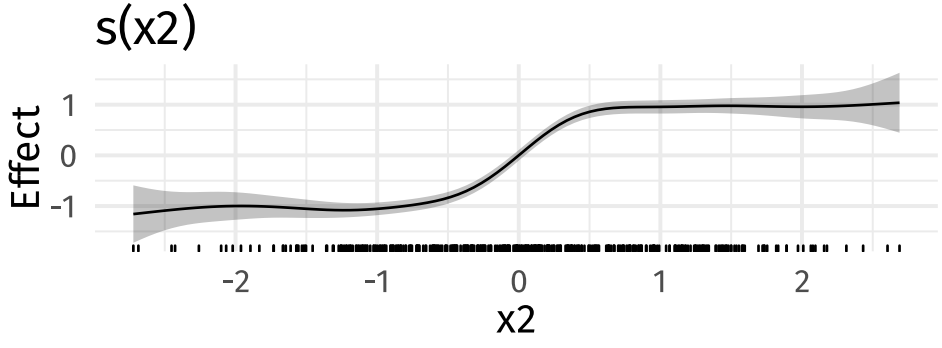












Model diagnostics

Using gam.check() part 2: visual checks

gam.check() plots

`gam.check()` creates 4 plots:

1. Quantile-quantile plots of residuals. If the model is right, should follow 1-1 line
2. Histogram of residuals
3. Residuals vs. linear predictor
4. Observed vs. fitted values

`gam.check()` uses deviance residuals by default

Gaussian data, Gaussian model

```
norm_model ← gam(y_norm ~ s(x1, k=12) + s(x2, k=12), method = 'REML')
gam.check(norm_model, rep = 500)
```

Negative binomial data, Poisson model

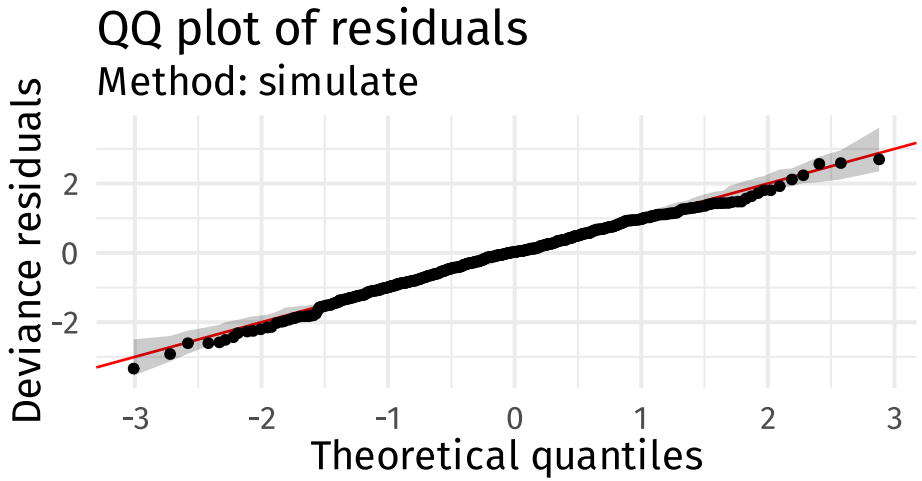
```
pois_model ← gam(y_negbinom ~ s(x1, k=12) + s(x2, k=12), family=poisson, method= 'REML')
gam.check(pois_model, rep = 500)
```

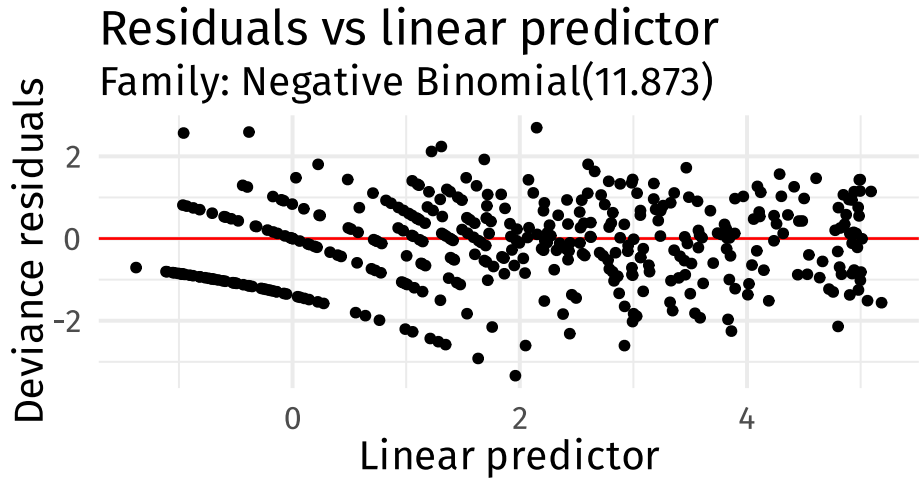
NB data, NB model

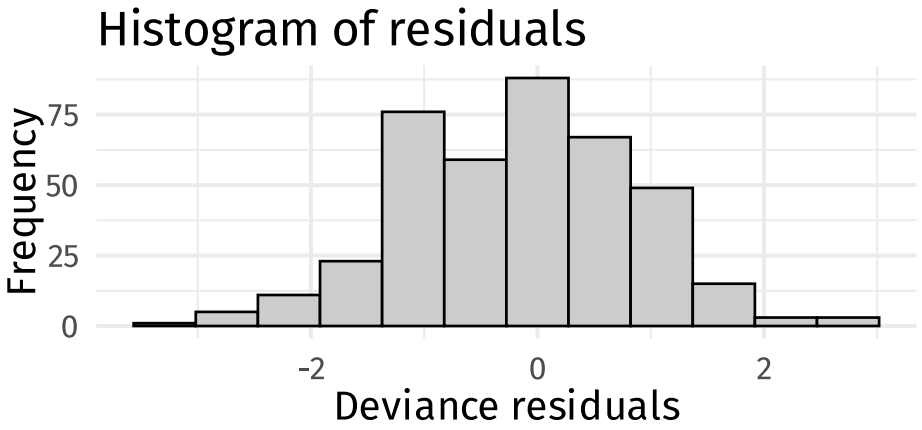
```
negbin_model ← gam(y_negbinom ~ s(x1, k=12) + s(x2, k=12), family = nb, method = 'REML')
gam.check(negbin_model, rep = 500)
```

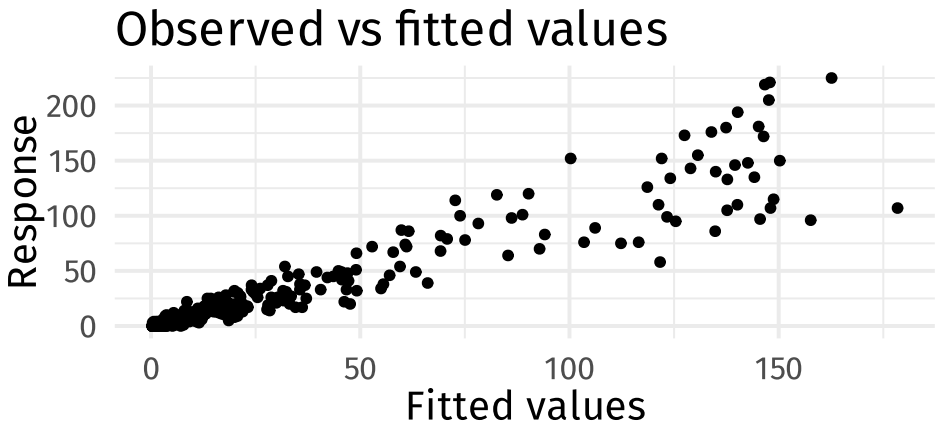
NB data, NB model

```
appraise(negbin_model, method = 'simulate')
```









Model selection

Model selection

Model (or variable) selection — an important area of theoretical and applied interest

- In statistics we aim for a balance between *fit* and *parsimony*
- In applied research we seek the set of covariates with strongest effects on y

We seek a subset of covariates that improves *interpretability* and *prediction accuracy*

Shrinkage & additional penalties

Shrinkage & additional penalties

Smoothing parameter estimation allows selection of a wide range of potentially complex functions for smooths...

But, cannot remove a term entirely from the model because the penalties used act only on the *range space* of a spline basis. The *null space* of the basis is unpenalised.

- **Null space** — the basis functions that are smooth (constant, linear)
- **Range space** — the basis functions that are wiggly

Shrinkage & additional penalties

mgcv has two ways to penalize the null space, i.e. to do selection

- *double penalty approach* via `select = TRUE`
- *shrinkage approach* via special bases for
  - thin plate spline (default, `s( ... , bs = 'ts')`),
  - cubic splines (`s( ... , bs = 'cs')`)

double penalty tends to works best, but applies to all smooths *and* doubles the number of smoothness parameters to estimate

Other shrinkage/selection approaches *are available* in other software

Empirical Bayes...?

$\mathbf{S}_j$ can be viewed as prior precision matrices and λ_j as improper Gaussian priors on the spline coefficients.

The impropriety derives from $\mathbf{S}_j$ not being of full rank (zeroes in $\mathbf{\Lambda}_j$).

Both the double penalty and shrinkage smooths remove the impropriety from the Gaussian prior

Empirical Bayes...?

- **Double penalty** — makes no assumption as to how much to shrink the null space. This is determined from the data via estimation of λ_j^*
- **Shrinkage smooths** — assumes null space should be shrunk less than the wiggly part

Marra & Wood (2011) show that the double penalty and the shrinkage smooth approaches

- performed significantly better than alternatives in terms of *predictive ability*, and
- performed as well as alternatives in terms of variable selection

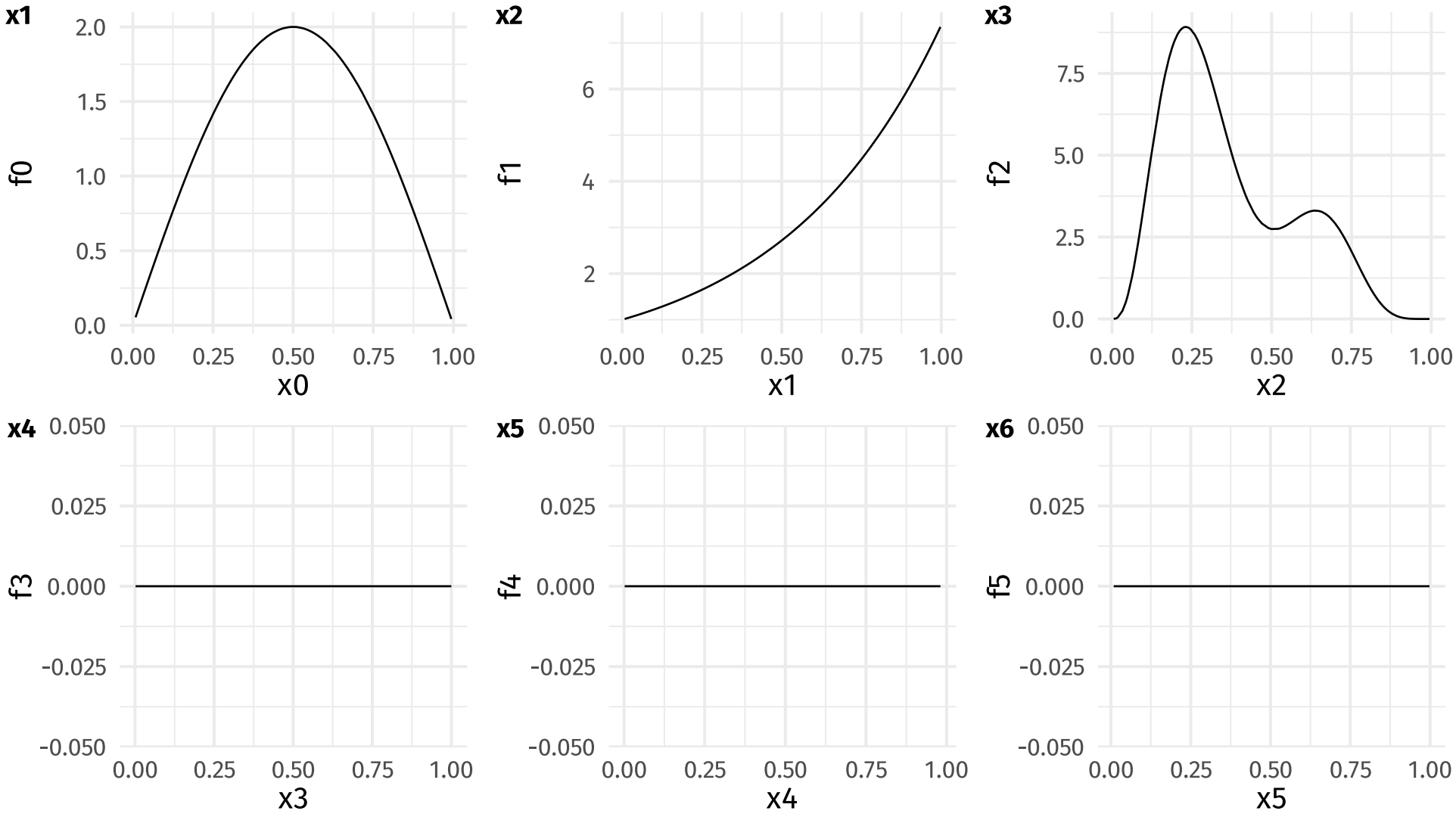
Example

- Simulate Poisson counts
- 4 known functions (left)
- 2 spurious covariates (`runif()` & not shown)

```
## an example of automatic model selection via null space penalization
set.seed(3)
n ← 200
dat ← gamSim(1, n=n, scale=.15, dist='poisson', verbose = FALSE) ## simulate data
dat ← transform(dat, x4 = runif(n, 0, 1), x5 = runif(n, 0, 1),
                f4 = rep(0, n), f5 = rep(0, n))   ## spurious
```

```
b ← gam(y ~ s(x0) + s(x1) + s(x2) + s(x3) +
            s(x4) + s(x5),
        data = dat, family = poisson, method = 'REML',
        select = TRUE)
```

Example


x1
f0
2.0
1.5
1.0
0.5
0.0
0.00 0.25 0.50 0.75 1.00
x0
x2
f1
6
4
2
0.00 0.25 0.50 0.75 1.00
x1
x3
f2
7.5
5.0
2.5
0.0
0.00 0.25 0.50 0.75 1.00
x2
x4
f3
0.050
0.025
0.000
-0.025
-0.050
0.00 0.25 0.50 0.75 1.00
x3
x5
f4
0.050
0.025
0.000
-0.025
-0.050
0.00 0.25 0.50 0.75 1.00
x4
x6
f5
0.050
0.025
0.000
-0.025
-0.050
0.00 0.25 0.50 0.75 1.00
x5

Example

```
summary(b)
```

```
##
## Family: poisson
## Link function: log
##
## Formula:
## y ~ s(x0) + s(x1) + s(x2) + s(x3) + s(x4) + s(x5)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.21758    0.04082   29.83   <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(x0) 1.7655088      9   5.264  0.0397 *
## s(x1) 1.9271040      9  65.356  <2e-16 ***
## s(x2) 6.1351414      9 156.204  <2e-16 ***
## s(x3) 0.0002849      9   0.000  0.4068
## s(x4) 0.0003044      9   0.000  1.0000
## s(x5) 0.1756926      9   0.195  0.2963
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.545   Deviance explained = 51.6%
## -REML = 430.78  Scale est. = 1         n = 200
```

Example

```
library('gratia'); draw(b, scales = 'fixed')
```

Confidence intervals for smooths

Confidence intervals for smooths

`plot.gam()` produces approximate 95% intervals (at +/- 2 SEs)

What do these intervals represent?

Nychka (1988) showed that standard Wahba/Silverman type Bayesian confidence intervals on smooths had good **across-the-function** frequentist coverage properties

When *averaged* over the range of covariate, 1 - α coverage is approximately 1 - α

Confidence intervals for smooths

Marra & Wood (2012) extended this theory to the generalised case and explain where the coverage properties failed:

Mustn't over-smooth too much, which happens when λ_j are over-estimated

Two situations where this might occur

1. where true effect is almost in the penalty null space, $\hat{\lambda}_j \to \infty$
   - ie. close to a linear function
2. where $\hat{\lambda}_j$ difficult to estimate due to highly correlated covariates
   - if 2 correlated covariates have different amounts of wiggliness, estimated effects can have degree of smoothness *reversed*

Don't over-smooth

> In summary, we have shown that Bayesian componentwise variable width intervals... for the smooth components of an additive model **should achieve close to nominal *across-the-function* coverage probability**&hellip

Basically

1. Don't over smooth, and
2. Effect of uncertainty due to estimating smoothness parameter is small

Confidence intervals for smooths

Marra & Wood (2012) suggested a solution to situation 1., namely true functions close to the penalty null space.

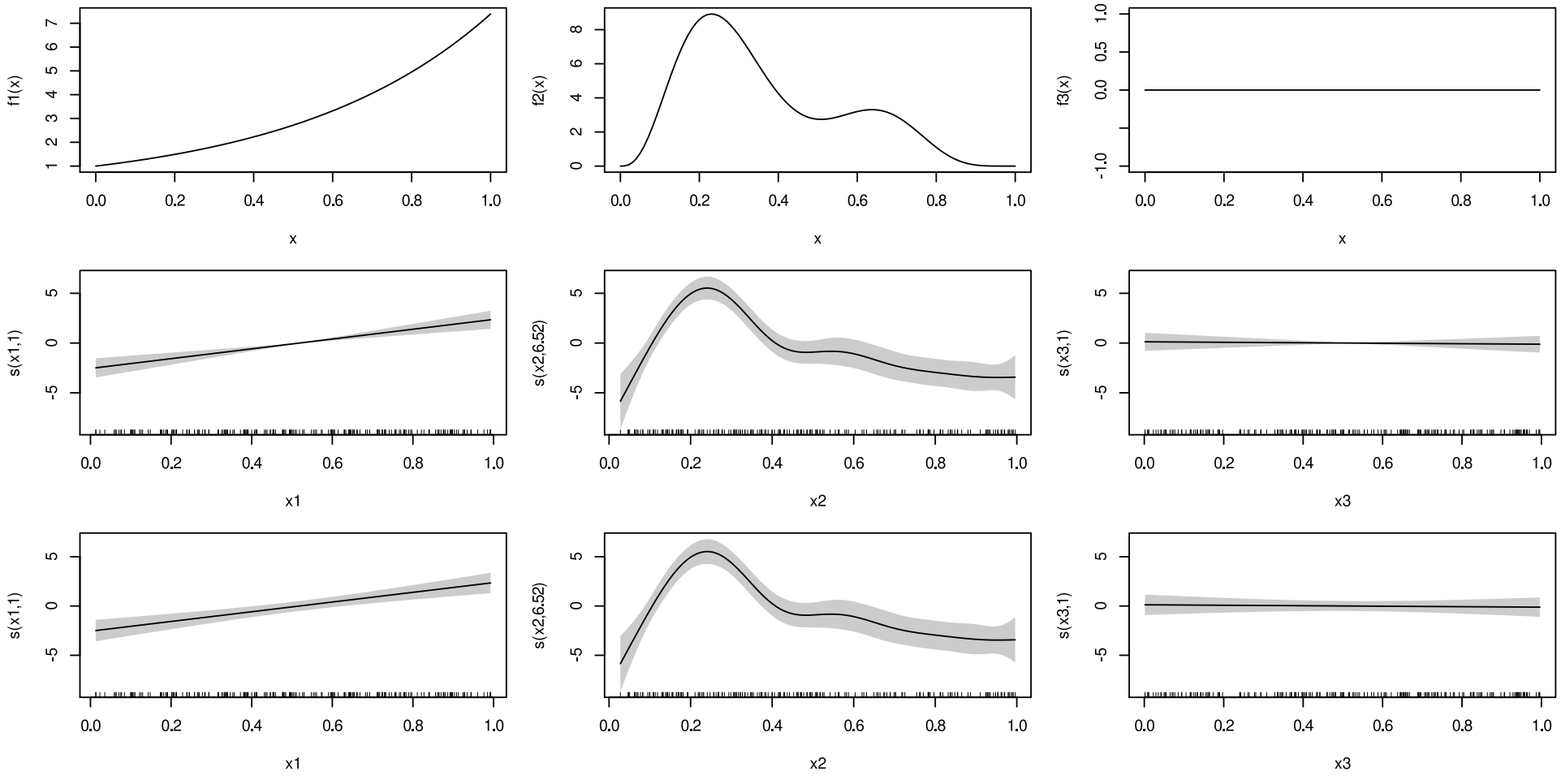
Smooths are normally subject to *identifiability* constraints (centred), which leads to zero variance where the estimated function crosses the zero line.

Instead, compute intervals for j th smooth as if it alone had the intercept; identifiability constraints go on the other smooth terms.

Use

- `seWithMean = TRUE` in call to `plot.gam()`
- `overall_uncertainty = TRUE` in call to `gratia::draw()`

Example


f1(x)
x
f2(x)
x
f3(x)
x
s(x1,1)
x1
s(x2,6.52)
x2
s(x3,1)
x3
s(x1,1)
x1
s(x2,6.52)
x2
s(x3,1)
x3

p values for smooths

Example

```
summary(b)
```

```
##
## Family: poisson
## Link function: log
##
## Formula:
## y ~ s(x0) + s(x1) + s(x2) + s(x3) + s(x4) + s(x5)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.21758    0.04082   29.83   <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(x0) 1.7655088      9   5.264  0.0397 *
## s(x1) 1.9271040      9  65.356  <2e-16 ***
## s(x2) 6.1351414      9 156.204  <2e-16 ***
## s(x3) 0.0002849      9   0.000  0.4068
## s(x4) 0.0003044      9   0.000  1.0000
## s(x5) 0.1756926      9   0.195  0.2963
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.545   Deviance explained = 51.6%
## -REML = 430.78  Scale est. = 1         n = 200
```

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 ∞

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 unpenalized smooths

The results of Nychka (1988) and Marra & Wood (2012) break down if smooth terms are unpenalized

This include i.i.d. Gaussian random effects, (e.g. `bs = "re"`)

Wood (2013b) proposed instead a test based on a likelihood ratio statistic:

- the reference distribution used is appropriate for testing a H_0 on the boundary of the allowed parameter space...
- ...in other words, it corrects for a H_0 that a variance term is zero

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

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 λ_j is used, with the EDF replacing k, the number of model parameters
- This *conditional* AIC tends to select complex models, especially those with random effects, as the EDF ignores that λ_j are estimated
- Wood et al (2016) suggests a correction that accounts for uncertainty in λ_j

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

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
```

Example

Atmospheric CO_2

```
data(co2s)
head(co2s)
```

```
##       co2 c.month month
## 1      NA       1     1
## 2      NA       2     2
## 3      NA       3     3
## 4      NA       4     4
## 5      NA       5     5
## 6 313.21       6     6
```

Atmospheric CO_2

```
ggplot(co2s, aes(x = c.month, y = co2)) + geom_line()
```

Atmospheric CO_2 — fit naive GAM

```
b ← gam(co2 ~ s(c.month, k=300, bs="cr"), data = co2s, method = 'REML')
summary(b)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## co2 ~ s(c.month, k = 300, bs = "cr")
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.382e+02  4.444e-03   76111   <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(c.month) 205.3  241.4 44646  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =      1   Deviance explained =  100%
## -REML = 36.492  Scale est. = 0.0084334  n = 427
```

Atmospheric CO_2 — predict

Predict the next 36 months

```
pd ← with(co2s, data.frame(c.month = 1:(nrow(co2s)+36)))
pd ← cbind(pd, predict(b, pd, se = TRUE))
pd ← transform(pd, upr = fit + (2*se.fit), lwr = fit - (2 * se.fit))
```

Atmospheric CO_2 — predict

```
ggplot(pd, aes(x = c.month, y = fit)) +
    geom_ribbon(aes(ymin = lwr, ymax = upr), alpha = 0.2) +
    geom_line(data = co2s, aes(c.month, co2), col = 'red') +
    geom_line(alpha = 0.5)
```

Atmospheric CO_2 — better model

Decompose into

1. a seasonal smooth
2. a long term trend

```
b2 ← gam(co2 ~ s(month, bs = "cc") + s(c.month, bs = "cr", k = 300),
          data = co2s, method = 'REML',
          knots = list(month = c(0.5, 12.5)))
```

Atmospheric CO_2 — better model

```
summary(b2)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## co2 ~ s(month, bs = "cc") + s(c.month, bs = "cr", k = 300)
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.382e+02  5.435e-03   62229   <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(month)     7.251    8.0   169.9  <2e-16 ***
## s(c.month) 104.823  128.8 55572.6  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =      1   Deviance explained =  100%
## -REML = -101.49  Scale est. = 0.012615  n = 427
```

Atmospheric CO_2 — predict

```
nr ← nrow(co2s)
pd2 ← with(co2s, data.frame(c.month = 1:(nr+36),
                            month = rep(1:12, length.out=nr+36)))
pd2 ← cbind(pd2, predict(b2, pd2, se = TRUE))
pd2 ← transform(pd2, upr = fit + (2*se.fit), lwr = fit - (2 * se.fit))
```

Atmospheric CO_2 — predict

```
ggplot(pd2, aes(x = c.month, y = fit)) +
    geom_ribbon(aes(ymin = lwr, ymax = upr), alpha = 0.2) +
    geom_line(data = co2s, aes(c.month, co2), col = 'red') +
    geom_line(alpha = 0.5)
```

Example

Galveston Bay

Cross Validated question 

> I have a dataset of water temperature measurements taken from a large waterbody at irregular intervals over a period of decades. (Galveston Bay, TX if you're interested)

https://stats.stackexchange.com/q/244042/1390

Galveston Bay

```
galveston ← read_csv(here('data', 'gbtemp.csv')) %>%
    mutate(datetime = as.POSIXct(paste(DATE, TIME), format = '%m/%d/%y %H:%M', tz = "CDT"),
           STATION_ID = factor(STATION_ID), DoY = as.numeric(format(datetime, format = '%j')),
           ToD = as.numeric(format(datetime, format = '%H')) +
               (as.numeric(format(datetime, format = '%M')) / 60))
galveston
```

```
## # A tibble: 15,276 x 13
##    STATION_ID DATE  TIME  LATITUDE LONGITUDE  YEAR MONTH   DAY SEASON
##    <fct>      <chr> <tim>    <dbl>     <dbl> <dbl> <dbl> <dbl> <chr>
##  1 13296      6/20… 11:04     29.5     -94.8  1991     6    20 Summer
##  2 13296      3/17… 09:30     29.5     -94.8  1992     3    17 Spring
##  3 13296      9/23… 11:24     29.5     -94.8  1991     9    23 Fall
##  4 13296      9/23… 11:24     29.5     -94.8  1991     9    23 Fall
##  5 13296      6/20… 11:04     29.5     -94.8  1991     6    20 Summer
##  6 13296      12/1… 10:15     29.5     -94.8  1991    12    17 Winter
##  7 13296      6/29… 11:17     29.5     -94.8  1992     6    29 Summer
##  8 13305      3/24… 11:53     29.6     -95.0  1987     3    24 Spring
##  9 13305      4/2/… 13:39     29.6     -95.0  1987     4     2 Spring
## 10 13305      8/11… 15:25     29.6     -95.0  1987     8    11 Summer
## # … with 15,266 more rows, and 4 more variables: MEASUREMENT <dbl>,
## #   datetime <dttm>, DoY <dbl>, ToD <dbl>
```

Galveston Bay model description

$$\begin{aligned}\mathrm{E}(y_i) = \alpha + f_1(\mathrm{ToD}_i) + f_2(\mathrm{DoY}_i) + f_3(\mathrm{Year}_i) + f_4(\mathrm{x}_i, \mathrm{y}_i) + \\ f_5(\mathrm{DoY}_i, \mathrm{Year}_i) + f_6(\mathrm{x}_i, \mathrm{y}_i, \mathrm{ToD}_i) + \\ f_7(\mathrm{x}_i, \mathrm{y}_i, \mathrm{DoY}_i) + f_8(\mathrm{x}_i, \mathrm{y}_i, \mathrm{Year}_i)\end{aligned}$$

- α is the model intercept,
- $f_1(\mathrm{ToD}_i)$ is a smooth function of time of day,
- $f_2(\mathrm{DoY}_i)$ is a smooth function of day of year ,
- $f_3(\mathrm{Year}_i)$ is a smooth function of year,
- $f_4(\mathrm{x}_i, \mathrm{y}_i)$ is a 2D smooth of longitude and latitude,

Galveston Bay model description

$$\begin{aligned}\mathrm{E}(y_i) = \alpha + f_1(\mathrm{ToD}_i) + f_2(\mathrm{DoY}_i) + f_3(\mathrm{Year}_i) + f_4(\mathrm{x}_i, \mathrm{y}_i) + \\ f_5(\mathrm{DoY}_i, \mathrm{Year}_i) + f_6(\mathrm{x}_i, \mathrm{y}_i, \mathrm{ToD}_i) + \\ f_7(\mathrm{x}_i, \mathrm{y}_i, \mathrm{DoY}_i) + f_8(\mathrm{x}_i, \mathrm{y}_i, \mathrm{Year}_i)\end{aligned}$$

- $f_5(\mathrm{DoY}_i, \mathrm{Year}_i)$ is a tensor product smooth of day of year and year,
- $f_6(\mathrm{x}_i, \mathrm{y}_i, \mathrm{ToD}_i)$ tensor product smooth of location & time of day
- $f_7(\mathrm{x}_i, \mathrm{y}_i, \mathrm{DoY}_i)$ tensor product smooth of location day of year&
- $f_8(\mathrm{x}_i, \mathrm{y}_i, \mathrm{Year}_i$ tensor product smooth of location & year

Galveston Bay model description

$$\begin{aligned}\mathrm{E}(y_i) = \alpha + f_1(\mathrm{ToD}_i) + f_2(\mathrm{DoY}_i) + f_3(\mathrm{Year}_i) + f_4(\mathrm{x}_i, \mathrm{y}_i) + \\ f_5(\mathrm{DoY}_i, \mathrm{Year}_i) + f_6(\mathrm{x}_i, \mathrm{y}_i, \mathrm{ToD}_i) + \\ f_7(\mathrm{x}_i, \mathrm{y}_i, \mathrm{DoY}_i) + f_8(\mathrm{x}_i, \mathrm{y}_i, \mathrm{Year}_i)\end{aligned}$$

Effectively, the first four smooths are the main effects of

1. time of day,
2. season,
3. long-term trend,
4. spatial variation

Galveston Bay model description

$$\begin{aligned}\mathrm{E}(y_i) = \alpha + f_1(\mathrm{ToD}_i) + f_2(\mathrm{DoY}_i) + f_3(\mathrm{Year}_i) + f_4(\mathrm{x}_i, \mathrm{y}_i) + \\ f_5(\mathrm{DoY}_i, \mathrm{Year}_i) + f_6(\mathrm{x}_i, \mathrm{y}_i, \mathrm{ToD}_i) + \\ f_7(\mathrm{x}_i, \mathrm{y}_i, \mathrm{DoY}_i) + f_8(\mathrm{x}_i, \mathrm{y}_i, \mathrm{Year}_i)\end{aligned}$$

whilst the remaining tensor product smooths model smooth interactions between the stated covariates, which model

1. how the seasonal pattern of temperature varies over time,
2. how the time of day effect varies spatially,
3. how the seasonal effect varies spatially, and
4. how the long-term trend varies spatially

Galveston Bay — full model

```
knots ← list(DoY = c(0.5, 366.5))
m ← bam(MEASUREMENT ~
             s(ToD, k = 10) +
             s(DoY, k = 12, bs = 'cc') +
             s(YEAR, k = 30) +
             s(LONGITUDE, LATITUDE, k = 100, bs = 'ds', m = c(1, 0.5)) +
             ti(DoY, YEAR, bs = c('cc', 'tp'), k = c(12, 15)) +
             ti(LONGITUDE, LATITUDE, ToD, d = c(2,1), bs = c('ds','tp'),
                m = list(c(1, 0.5), NA), k = c(20, 10)) +
             ti(LONGITUDE, LATITUDE, DoY, d = c(2,1), bs = c('ds','cc'),
                m = list(c(1, 0.5), NA), k = c(25, 12)) +
             ti(LONGITUDE, LATITUDE, YEAR, d = c(2,1), bs = c('ds','tp'),
                m = list(c(1, 0.5), NA), k = c(25, 15)),
          data = galveston, method = 'fREML', knots = knots,
          nthreads = c(4, 1), discrete = TRUE)
```

Galveston Bay — simpler model

```
m.sub ← bam(MEASUREMENT ~
            s(ToD, k = 10) +
            s(DoY, k = 12, bs = 'cc') +
            s(YEAR, k = 30) +
            s(LONGITUDE, LATITUDE, k = 100, bs = 'ds', m = c(1, 0.5)) +
            ti(DoY, YEAR, bs = c('cc', 'tp'), k = c(12, 15)),
          data = galveston, method = 'fREML', knots = knots,
          nthreads = c(4, 1), discrete = TRUE)
```

Galveston Bay — simpler model?

```
AIC(m, m.sub)
```

```
##              df      AIC
## m     448.2885 58577.37
## m.sub 240.6117 59181.10
```

Galveston Bay — simpler model?

```
anova(m, m.sub, test = 'F')
```

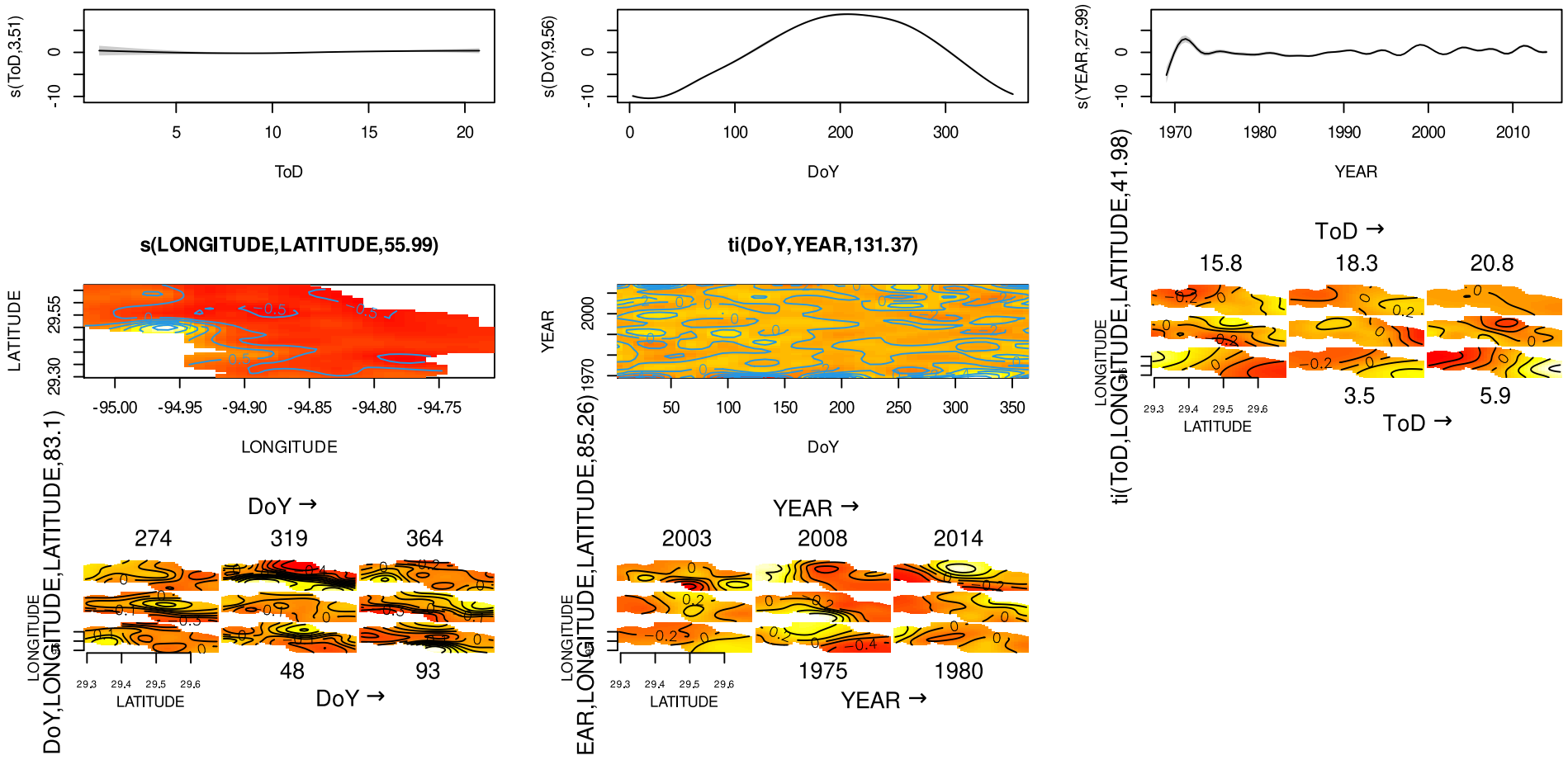
```
## Analysis of Deviance Table
##
## Model 1: MEASUREMENT ~ s(ToD, k = 10) + s(DoY, k = 12, bs = "cc") + s(YEAR,
##     k = 30) + s(LONGITUDE, LATITUDE, k = 100, bs = "ds", m = c(1,
##     0.5)) + ti(DoY, YEAR, bs = c("cc", "tp"), k = c(12, 15)) +
##     ti(LONGITUDE, LATITUDE, ToD, d = c(2, 1), bs = c("ds", "tp"),
##         m = list(c(1, 0.5), NA), k = c(20, 10)) + ti(LONGITUDE,
##     LATITUDE, DoY, d = c(2, 1), bs = c("ds", "cc"), m = list(c(1,
##         0.5), NA), k = c(25, 12)) + ti(LONGITUDE, LATITUDE, YEAR,
##     d = c(2, 1), bs = c("ds", "tp"), m = list(c(1, 0.5), NA),
##     k = c(25, 15))
## Model 2: MEASUREMENT ~ s(ToD, k = 10) + s(DoY, k = 12, bs = "cc") + s(YEAR,
##     k = 30) + s(LONGITUDE, LATITUDE, k = 100, bs = "ds", m = c(1,
##     0.5)) + ti(DoY, YEAR, bs = c("cc", "tp"), k = c(12, 15))
##   Resid. Df Resid. Dev      Df Deviance      F    Pr(>F)
## 1     14736      39029
## 2     15019      41722 -283.15  -2692.5 3.6147 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Galveston Bay — full model summary

```
summary(m)
```

Galveston Bay — full model plot

```
plot(m, pages = 1, scheme = 2, shade = TRUE)
```



Galveston Bay — full model plot

```
draw(m, scales = 'free')
```

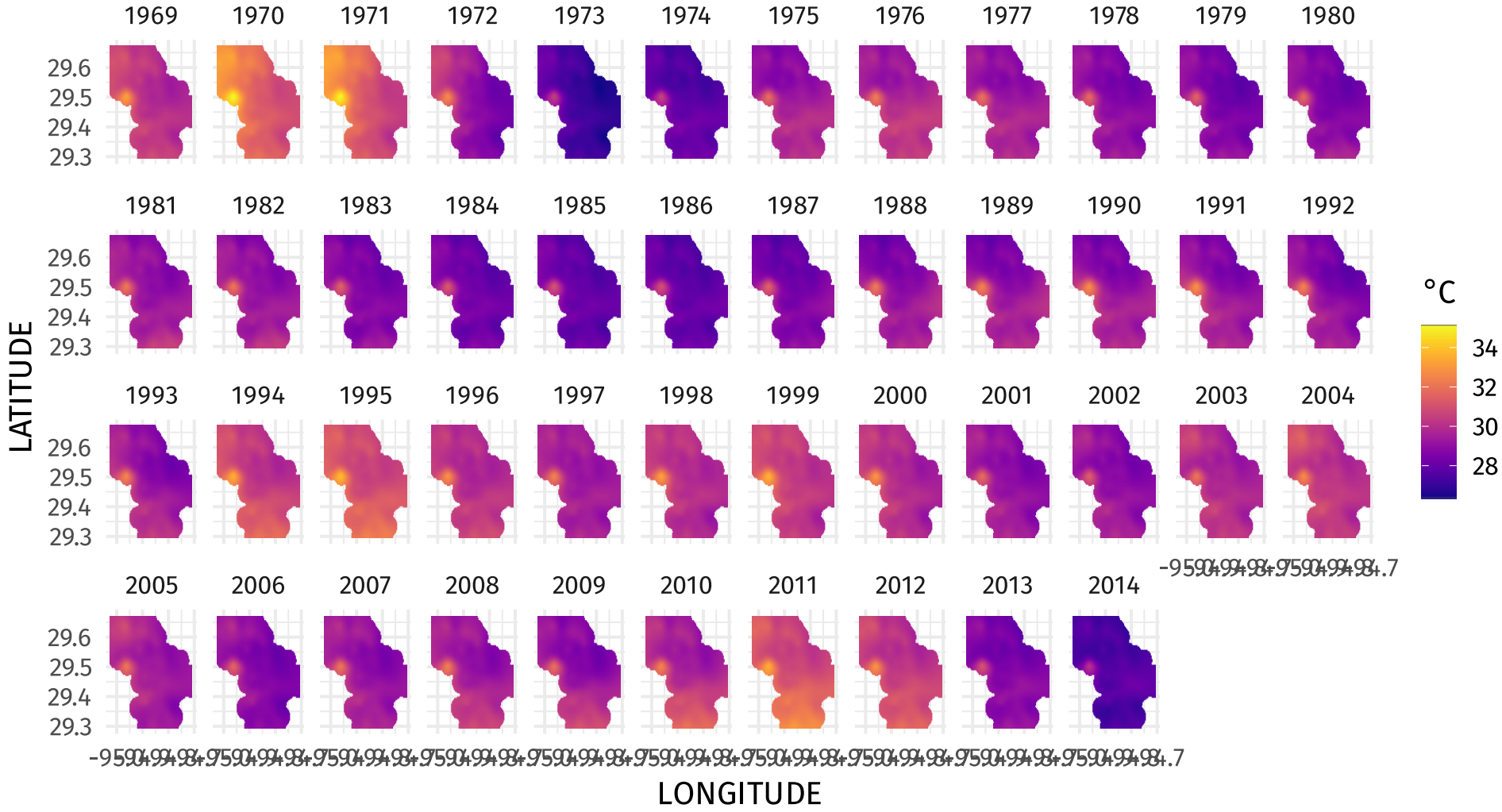
Galveston Bay — predict

```
pdata ← with(galveston,
             expand.grid(ToD = 12,
                         DoY = 180,
                         YEAR = seq(min(YEAR), max(YEAR), by = 1),
                         LONGITUDE = seq(min(LONGITUDE), max(LONGITUDE), length = 100),
                         LATITUDE  = seq(min(LATITUDE), max(LATITUDE), length = 100)))
fit ← predict(m, pdata)
ind ← exclude.too.far(pdata$LONGITUDE, pdata$LATITUDE,
                      galveston$LONGITUDE, galveston$LATITUDE, dist = 0.1)
fit[ind] ← NA
pred ← cbind(pdata, Fitted = fit)
```

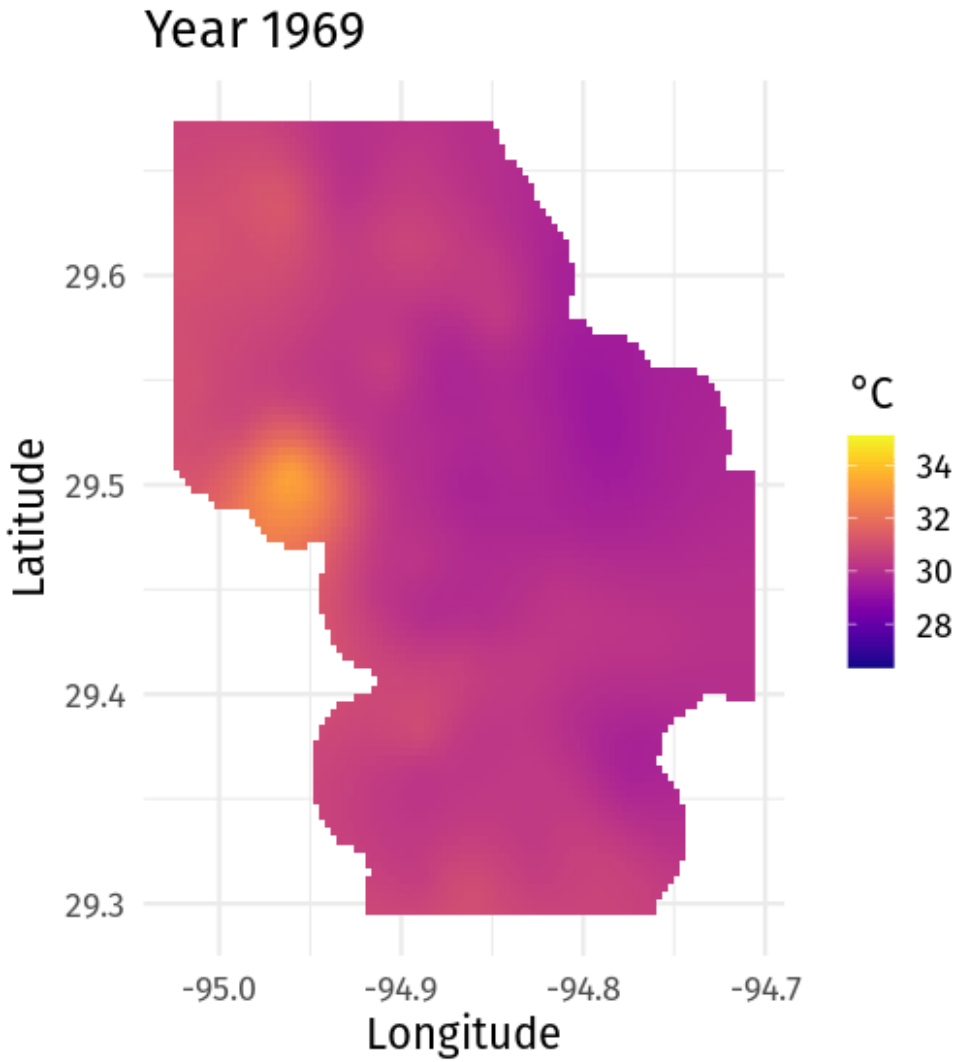
Galveston Bay — plot

```
plt ← ggplot(pred, aes(x = LONGITUDE, y = LATITUDE)) +
    geom_raster(aes(fill = Fitted)) + facet_wrap(~ YEAR, ncol = 12) +
    scale_fill_viridis(name = expression(degree*C), option = 'plasma', na.value = 'transparent') +
    coord_quickmap() +
    theme(legend.position = 'right')
plt
```

Galveston Bay — plot



Galveston Bay —



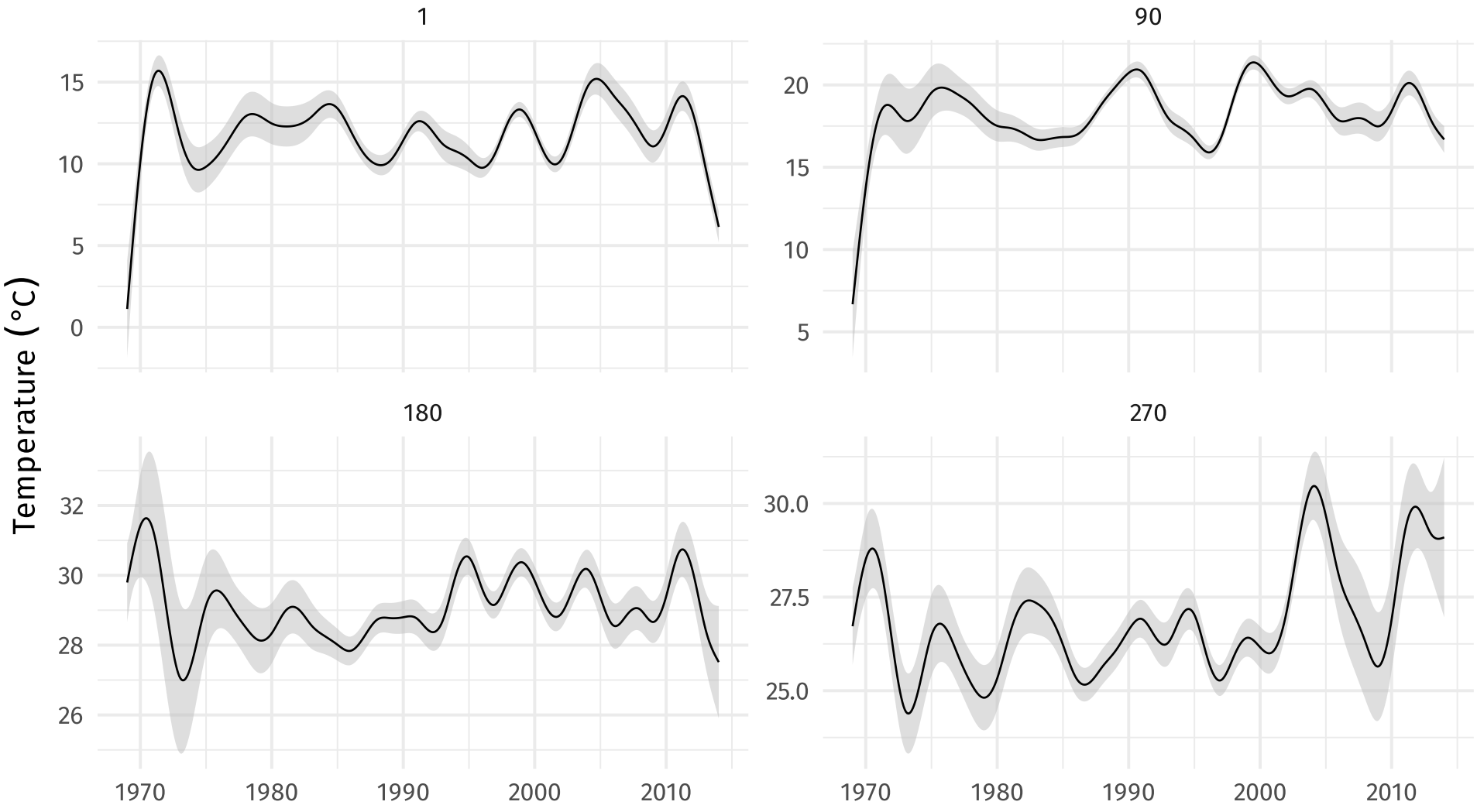
Galveston Bay — plot trends

```
pdata ← with(galveston,
             expand.grid(ToD = 12,
                         DoY = c(1, 90, 180, 270),
                         YEAR = seq(min(YEAR), max(YEAR), length = 500),
                         LONGITUDE = -94.8751,
                         LATITUDE  = 29.50866))

fit ← data.frame(predict(m, newdata = pdata, se.fit = TRUE))
fit ← transform(fit, upper = fit + (2 * se.fit), lower = fit - (2 * se.fit))
pred ← cbind(pdata, fit)

plt2 ← ggplot(pred, aes(x = YEAR, y = fit, group = factor(DoY))) +
    geom_ribbon(aes(ymin = lower, ymax = upper), fill = 'grey', alpha = 0.5) +
    geom_line() + facet_wrap(~ DoY, scales = 'free_y') +
    labs(x = NULL, y = expression(Temperature ~ (degree * C)))
plt2
```

Galveston Bay — plot trends



Next steps

Read Simon Wood's book!

Lots more material on our ESA GAM Workshop site

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

Noam Ross' free GAM Course

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

A couple of papers:

1. Simpson, G.L., 2018. Modelling Palaeoecological Time Series Using Generalised Additive Models. Frontiers in Ecology and Evolution 6, 149. https://doi.org/10.3389/fevo.2018.00149
2. Pedersen, E.J., Miller, D.L., Simpson, G.L., Ross, N., 2019. Hierarchical generalized additive models in ecology: an introduction with mgcv. PeerJ 7, e6876. https://doi.org/10.7717/peerj.6876

Also see my blog: www.fromthebottomoftheheap.net

Acknowledgments

Funding

Fellow GAM colleagues

David Miller

Eric Pedersen

Noam Ross

Slides

- HTML Slide deck bit.ly/gam-webinar © Simpson (2020) 
- RMarkdown Source

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 (2017) *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC. (2nd Edition)
- Wood (2013a) *Biometrika* **100**(1) 221–228.
- Wood (2013b) *Biometrika* **100**(4) 1005–1010.
- Wood et al (2016) *JASA* **111** 1548–1563