Ecological forecasting with dynamic GAMs

Nicholas Clark

School of Veterinary Science, University of Queensland

1200–1300 Thursday 12th October, 2023

About me

ARC Discovery Early Career Fellow

The University of Queensland

- School of Veterinary Science
- Located in Gatton, Australia

Interested in:

- Quantitative ecology
- Molecular genetics
- Multivariate time series modelling

“Because all decision making is based on what will happen in the future, either under the status quo or different decision alternatives, decision making ultimately depends on forecasts”

Dietze et al. 2018

Can we use common time series models in ecology?

Very easy to apply in 

Hyndman's tools in the `forecast` 📦 are hugely popular and accessible for time series analysis / forecasting

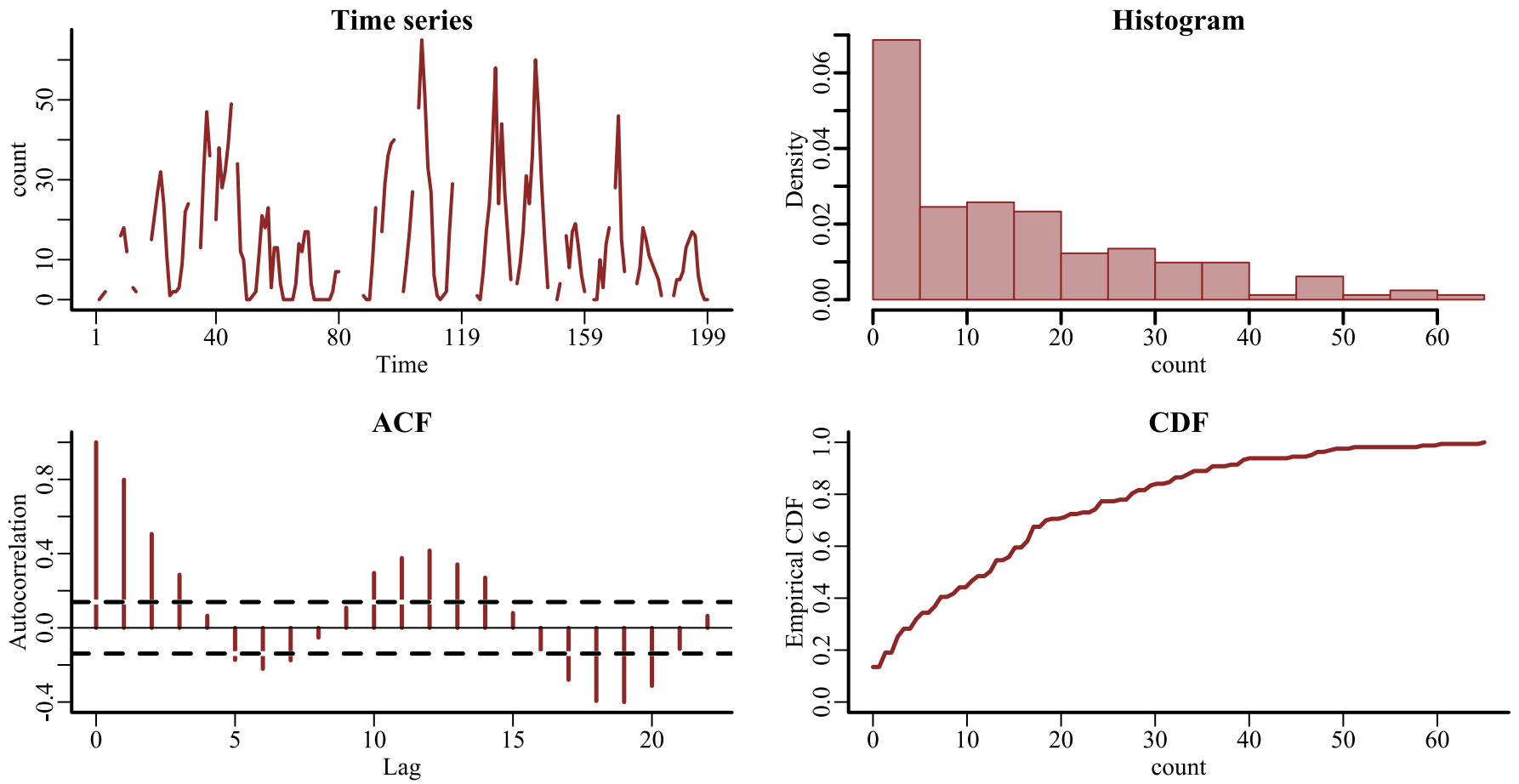
ETS handles many types of seasonality and nonlinear trends

Regression with ARIMA errors includes additive effects of predictors while capturing trends and seasonality

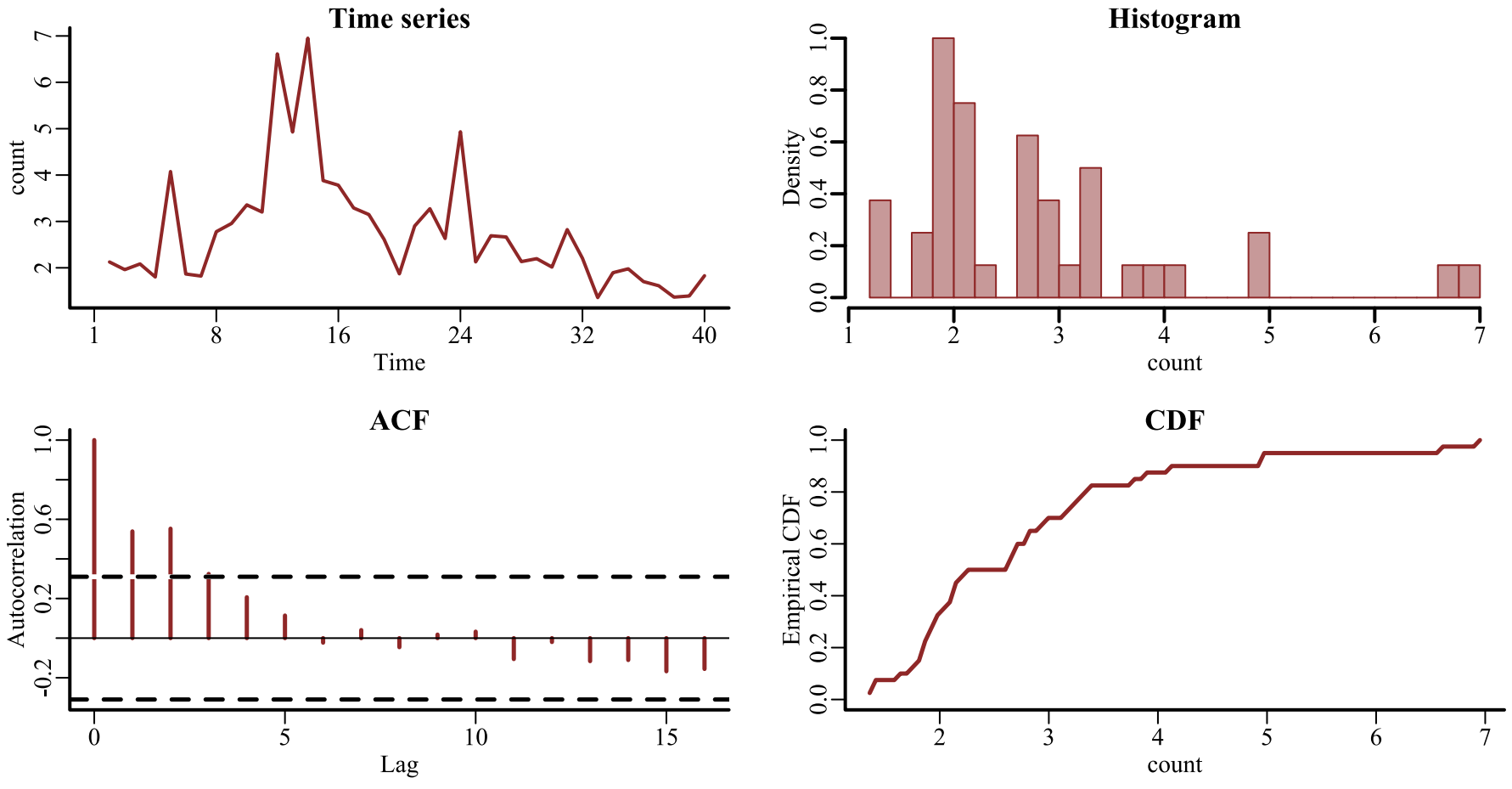
Some of these algorithms can handle missing data

All are fast to fit and forecast, but assume observations are *Gaussian*

But most real-world ecological observations, including time series, *are not Gaussian*



Properties of lunar monthly Desert Pocket Mouse capture time series from a long-term monitoring study in Portal, Arizona, USA



Properties of annual American kestrel abundance time series in British Columbia, Canada

“If our data contains small counts (0,1,2,...), then we need to use forecasting methods that are more appropriate for a sample space of non-negative integers.

***Such models are beyond the scope of this book*”**

Hyndman and Athanasopoulos, Forecasting Principles and Practice

Ok. So now what?

Poisson GLM for counts?

A Poisson GLM models the conditional mean with a log link

$$
\begin{aligned}
\boldsymbol{Y}_t &\sim \text{Poisson}(\lambda_t) \\
log(\lambda_t) &= \boldsymbol{X}_t\beta \\
&= \alpha + \beta_1\boldsymbol{x}_{1t} + \beta_2\boldsymbol{x}_{2t} + \cdots + \beta_j\boldsymbol{x}_{jt}
\end{aligned}
$$

The ***linear predictor component can be hugely flexible***, as we will see in a moment

GLMs allow us to build models that respect the bounds and distributions of our observed data

They traditionally assume the appropriately transformed mean response depends *linearly* on the predictors

But there are many other properties we'd like to model

Properties of ecological series

Temporal autocorrelation

Lagged effects

Non-Gaussian data and missing observations

Measurement error

Time-varying effects

Nonlinearities

Multi-series clustering

Properties of ecological series

Temporal autocorrelation

Lagged effects

Non-Gaussian data and missing observations

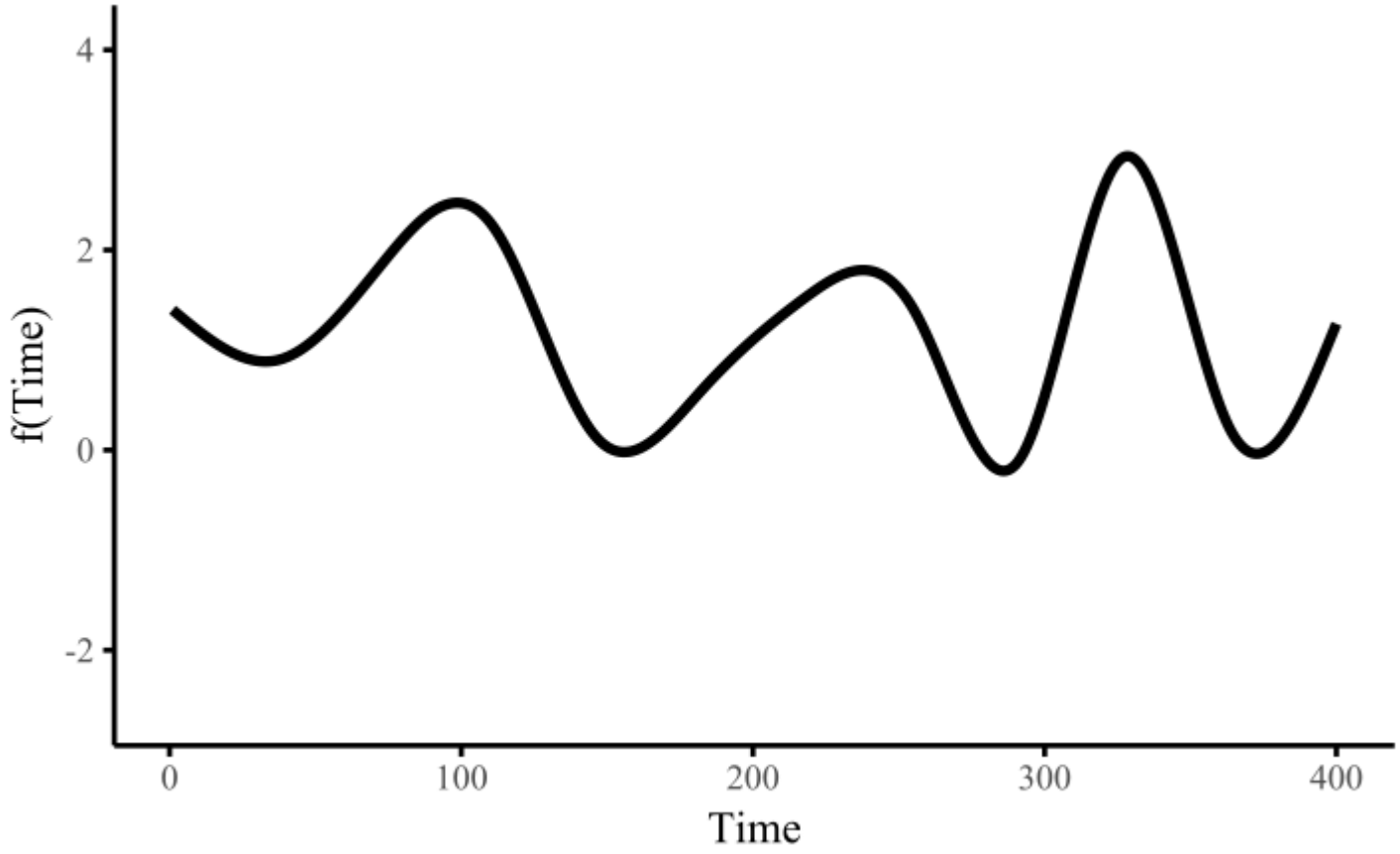
Measurement error

Time-varying effects

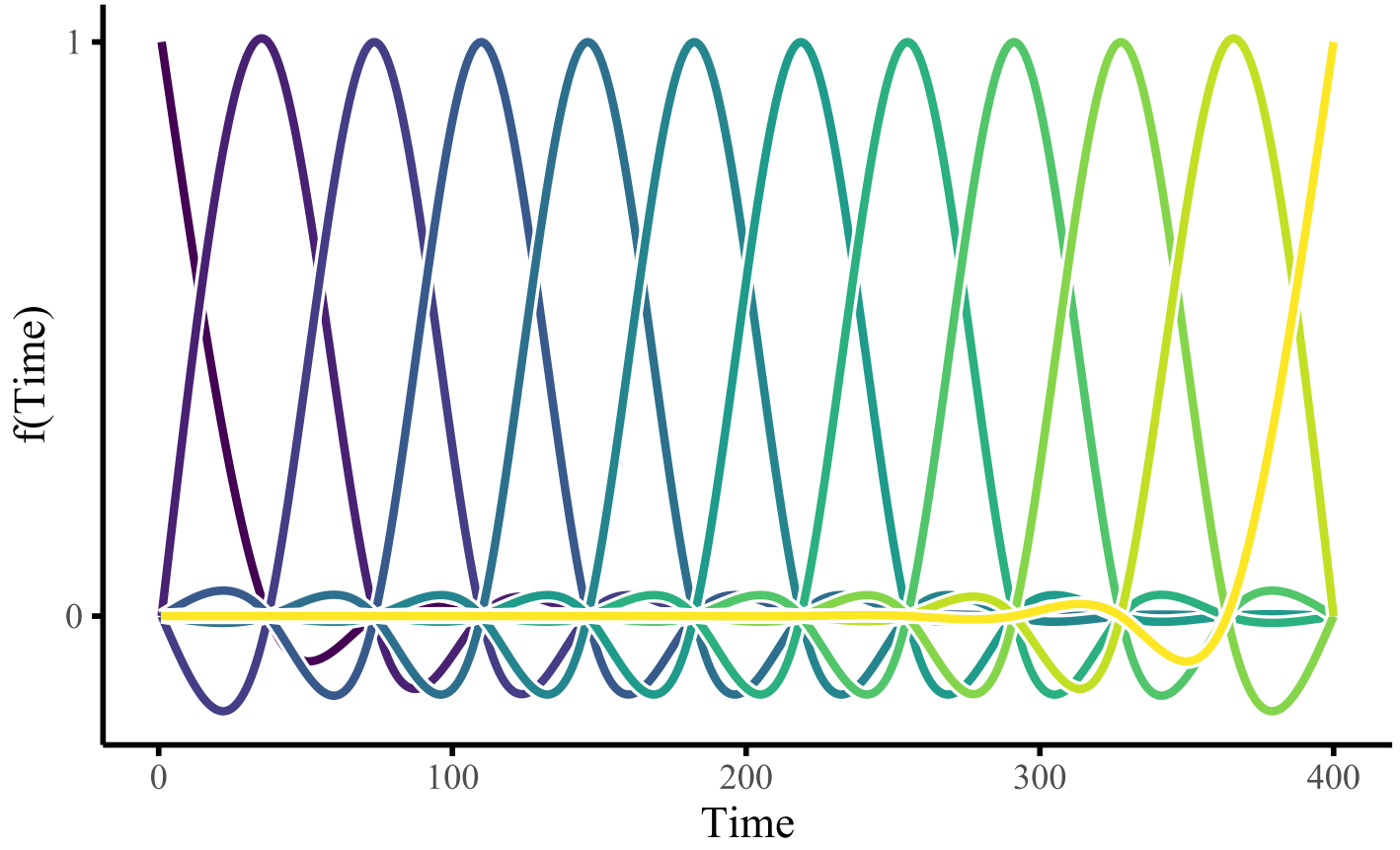
Nonlinearities

Multi-series clustering

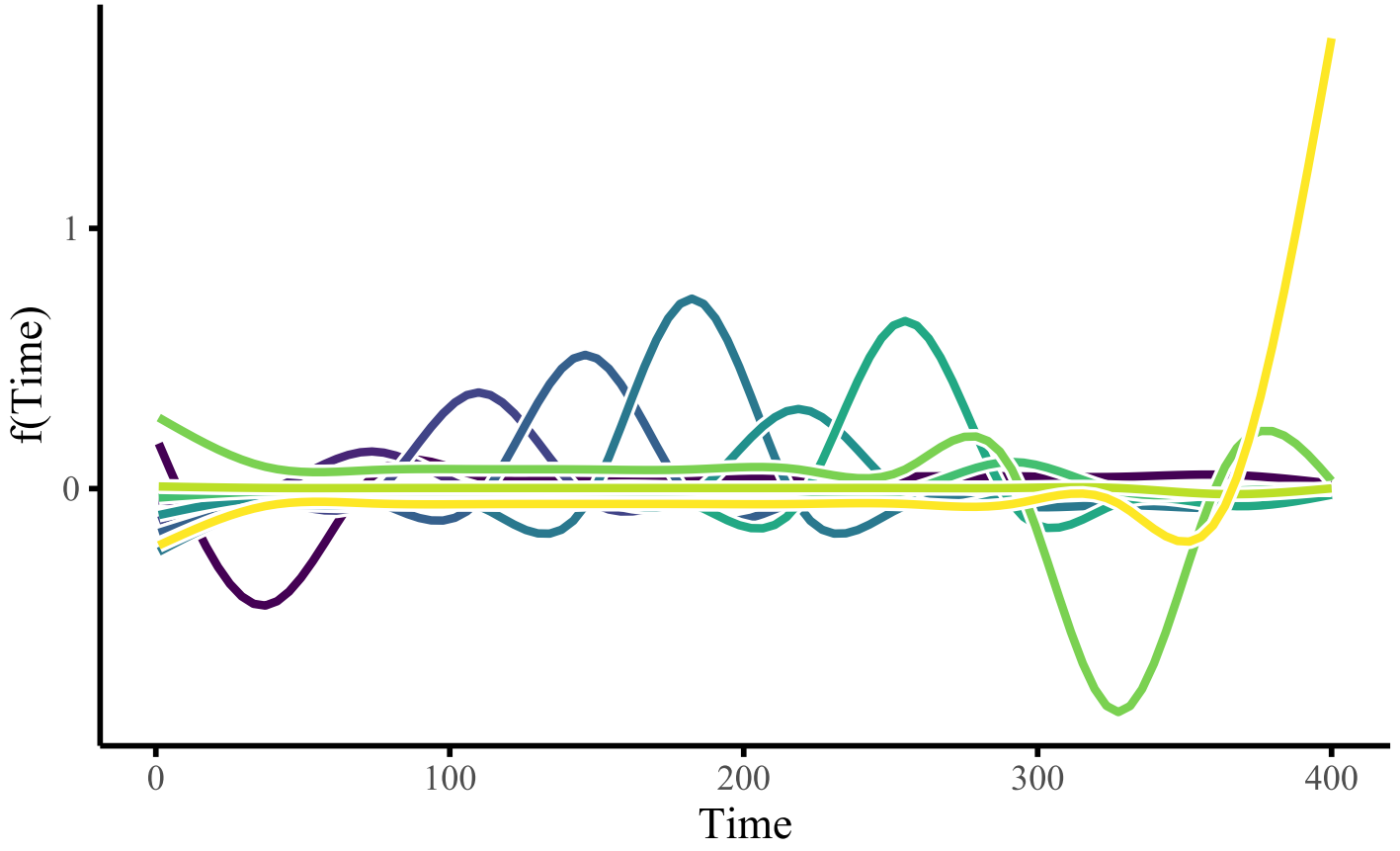
GAMs use splines ...



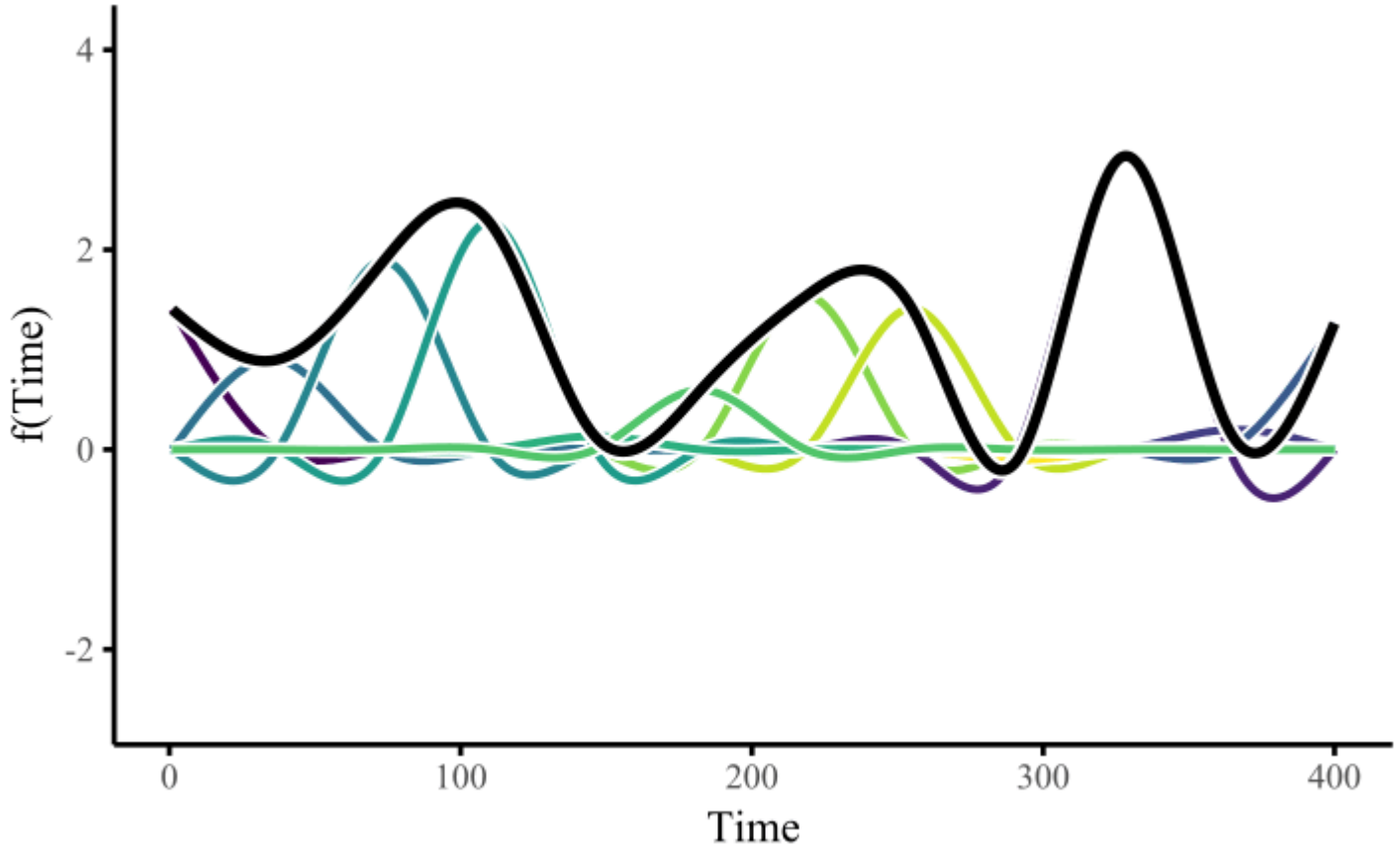
... made of basis functions



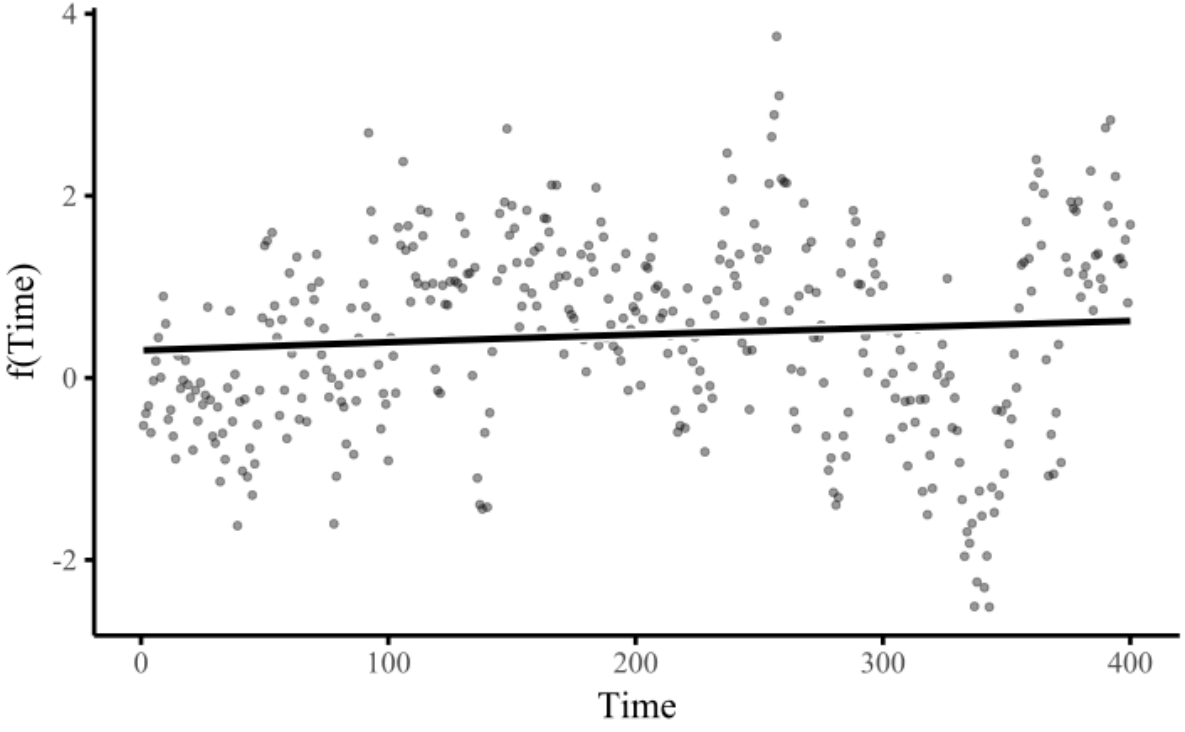
Weighting basis functions ...

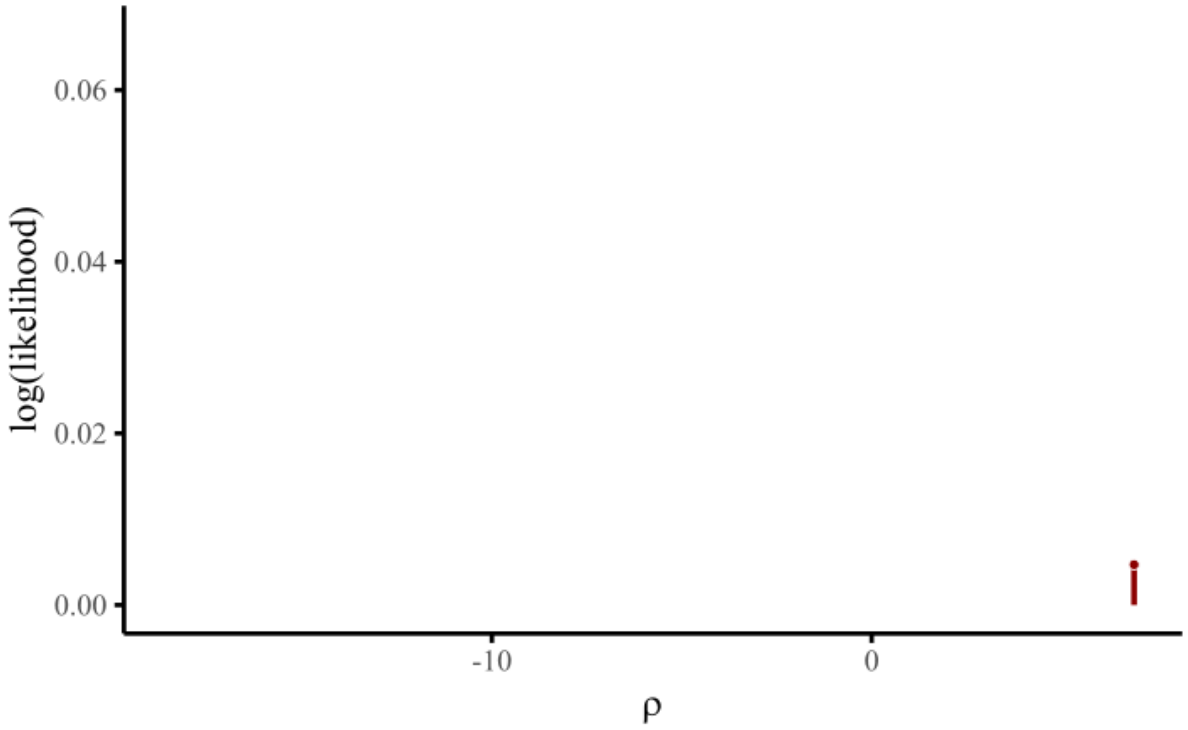


... gives a spline $(f(x))$



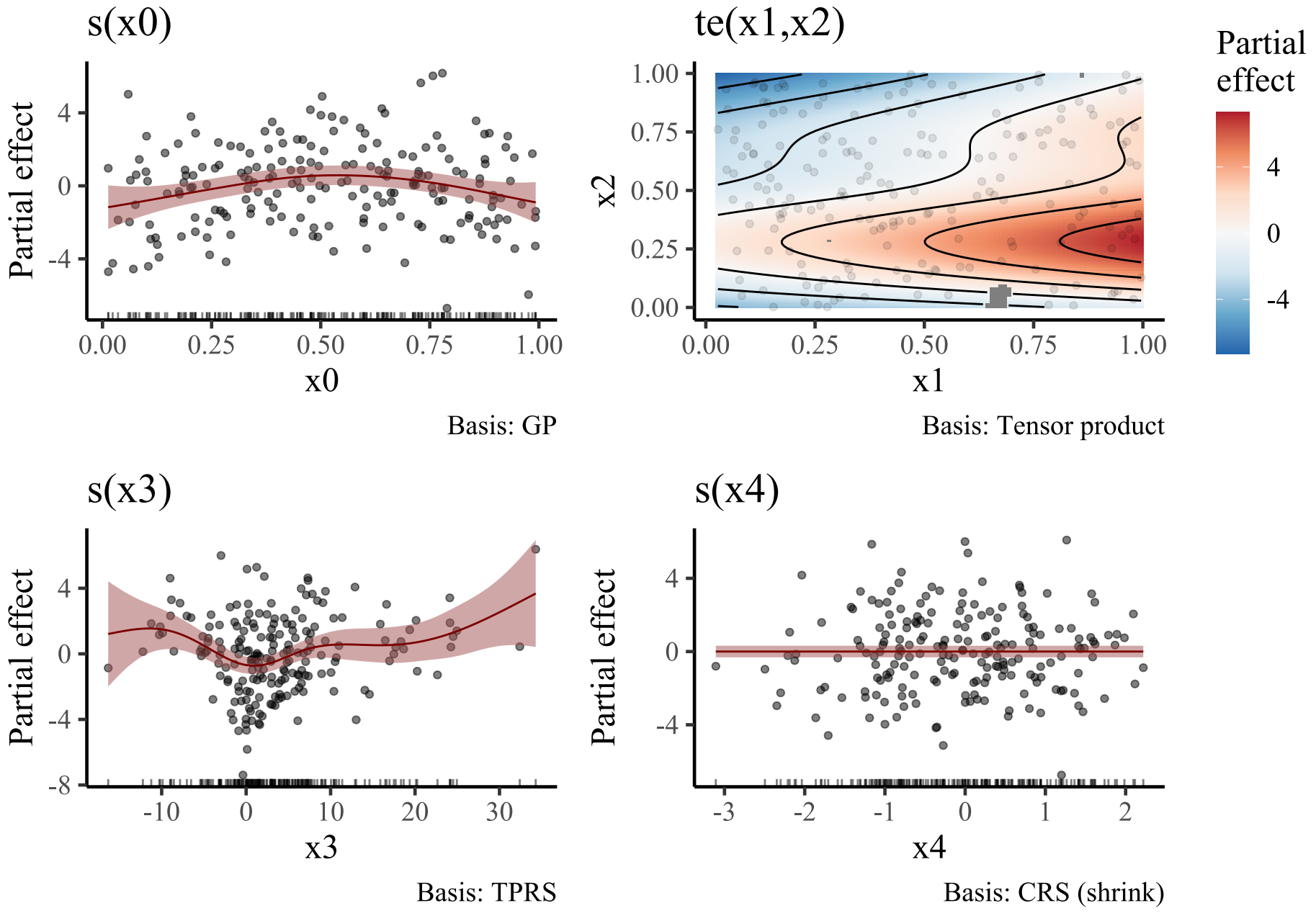
Penalize $f''(x)$ to learn weights





GAMs are just fancy GLMs, where some (or all) of the predictor effects are estimated as (possibly nonlinear) smooth functions

But the complexity they can handle is *enormous*


s(x0)
Partial effect
4
0
-4
0.00
0.25
0.50
0.75
1.00
x0
Basis: GP
te(x1,x2)
x2
1.00
0.75
0.50
0.25
0.00
x1
Basis: Tensor product
Partial effect
4
0
-4
s(x3)
Partial effect
4
0
-4
-8
-10
0
10
20
30
x3
Basis: TPRS
s(x4)
Partial effect
4
0
-4
-3
-2
-1
0
1
2
x4
Basis: CRS (shrink)

Modelling with the `mvgam` 📦

Bayesian framework to fit Dynamic GLMs and Dynamic GAMs

- Hierarchical intercepts, slopes *and smooths*
- Latent dynamic processes
- State Space models with measurement error

Built off the `mgcv` 📦 to construct penalized smoothing splines

Convenient and familiar R formula interface

Uni- or multivariate series from a range of response distributions

Uses Stan for efficient Hamiltonian Monte Carlo sampling

Example of the interface

```
model ← mvgam(
  formula = y ~
    s(series, bs = 're') +
    s(x0, series, bs = 're') +
    x1 +
    s(x2, bs = 'tp', k = 5) +
    te(x3, x4, bs = c('cr', 'tp')),
  data = data,
  family = poisson(),
  trend_model = 'AR1',
  burnin = 500,
  samples = 500,
  chains = 4,
  parallel = TRUE
  )
```

Typical formula syntax

```
model ← mvgam(
  formula = y ~
    s(series, bs = 're') +
    s(x0, series, bs = 're') +
    x1 +
    s(x2, bs = 'tp', k = 5) +
    te(x3, x4, bs = c('cr', 'tp')),
  data = data,
  family = poisson(),
  trend_model = 'AR1',
  burnin = 500,
  samples = 500,
  chains = 4,
  parallel = TRUE
  )
```

Data and response distribution

```
model ← mvgam(
  formula = y ~
    s(series, bs = 're') +
    s(x0, series, bs = 're') +
    x1 +
    s(x2, bs = 'tp', k = 5) +
    te(x3, x4, bs = c('cr', 'tp')),
  data = data,
  family = poisson(),
  trend_model = 'AR1',
  burnin = 500,
  samples = 500,
  chains = 4,
  parallel = TRUE
  )
```

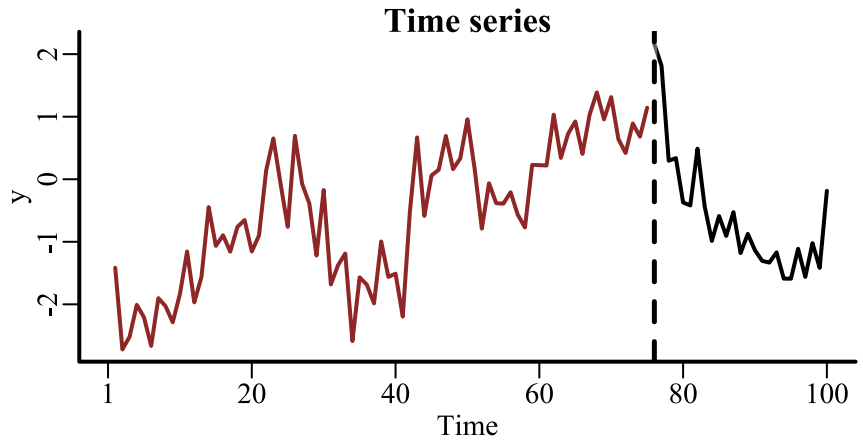
latent dynamics

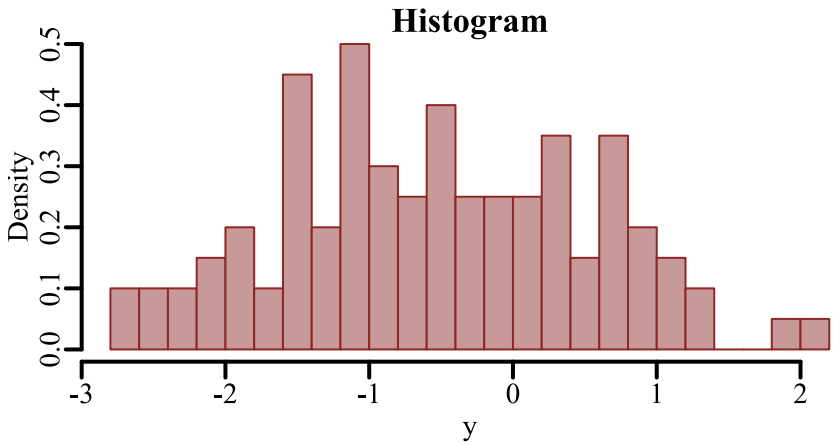
```
model ← mvgam(
  formula = y ~
    s(series, bs = 're') +
    s(x0, series, bs = 're') +
    x1 +
    s(x2, bs = 'tp', k = 5) +
    te(x3, x4, bs = c('cr', 'tp')),
  data = data,
  family = poisson(),
  trend_model = 'AR1',
  burnin = 500,
  samples = 500,
  chains = 4,
  parallel = TRUE
  )
```

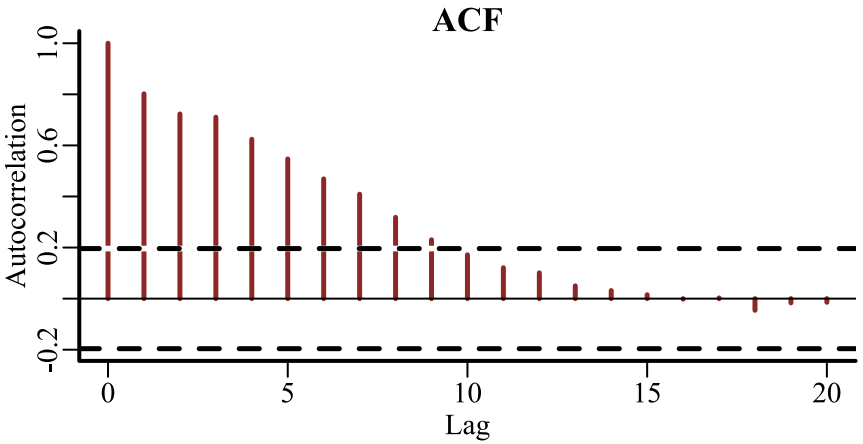
Sampler parameters

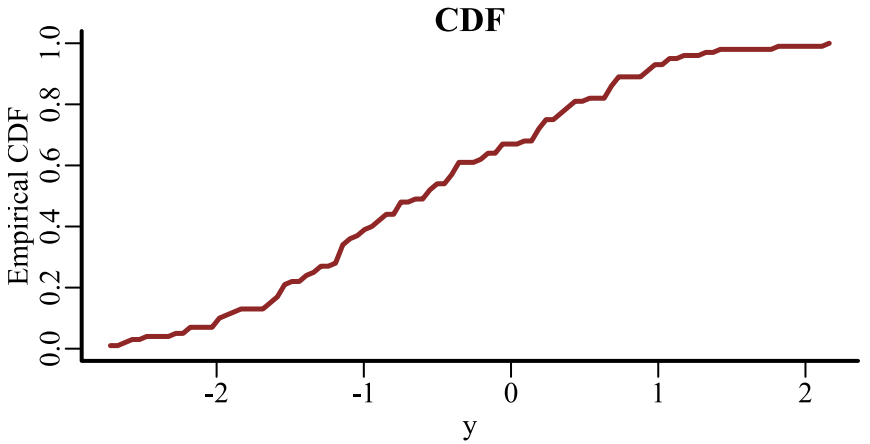
```
model ← mvgam(
  formula = y ~
    s(series, bs = 're') +
    s(x0, series, bs = 're') +
    x1 +
    s(x2, bs = 'tp', k = 5) +
    te(x3, x4, bs = c('cr', 'tp')),
  data = data,
  family = poisson(),
  trend_model = 'AR1',
  burnin = 500,
  samples = 500,
  chains = 4,
  parallel = TRUE
  )
```

Example: simulated data









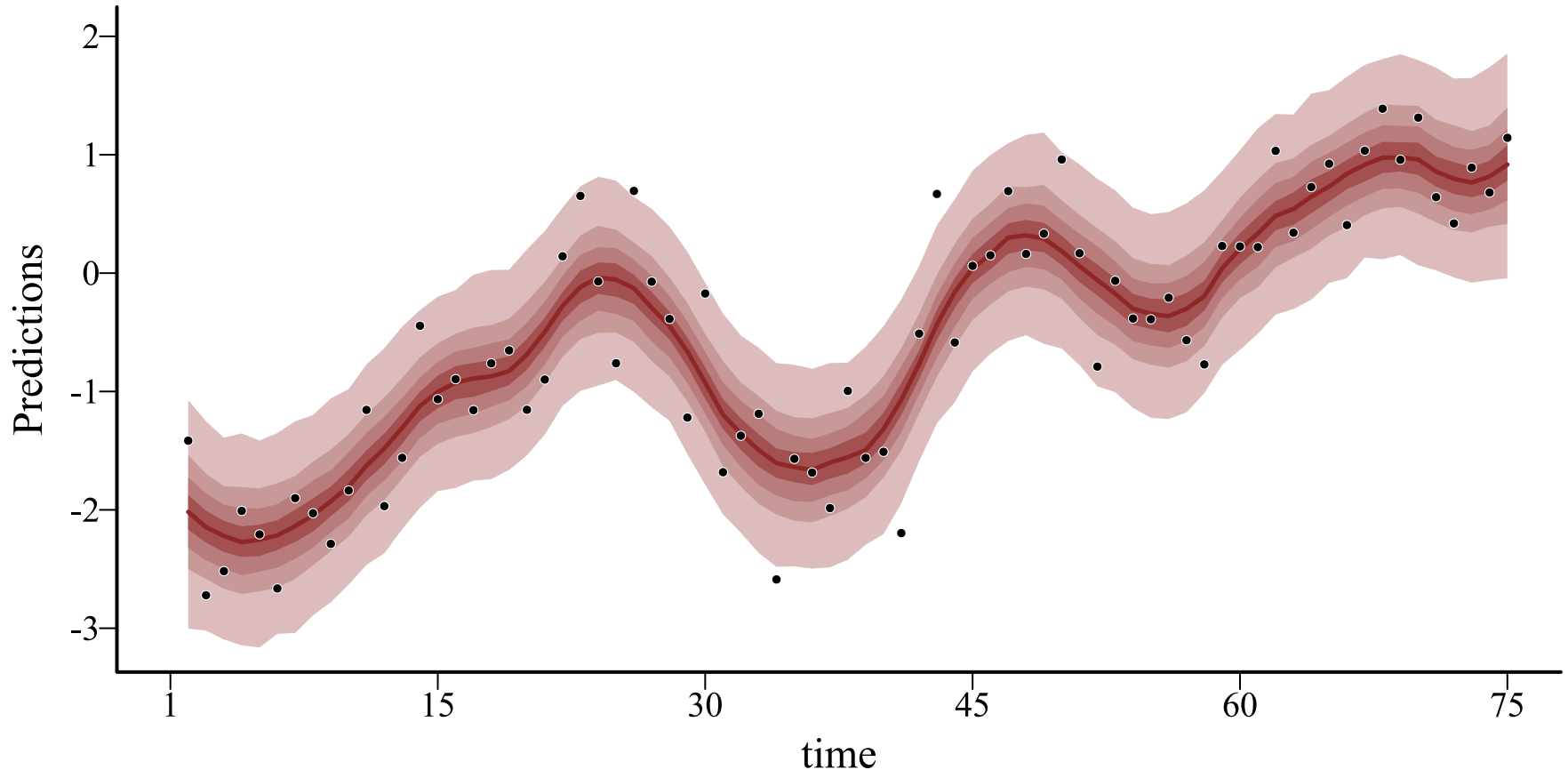
A spline of `time`

```
library(mvgam)
model ← mvgam(y ~
                 s(time, k = 20, bs = 'bs', m = 2),
               data = data_train,
               newdata = data_test,
               family = gaussian())
```

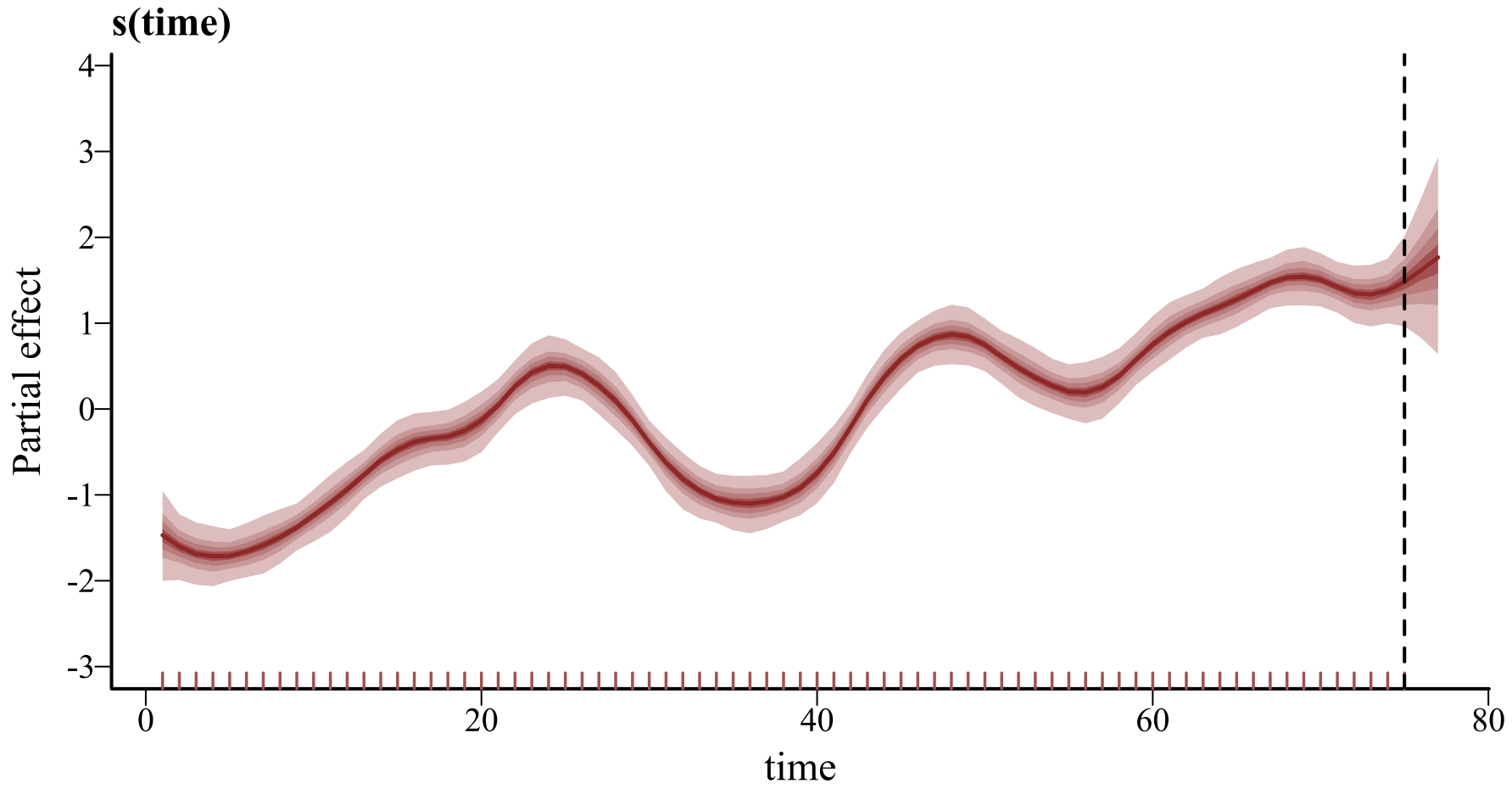
A B-spline (`bs = 'bs'`) with `m = 2` sets the penalty on the second derivative

Use `newdata` argument to generate automatic probabilistic forecasts

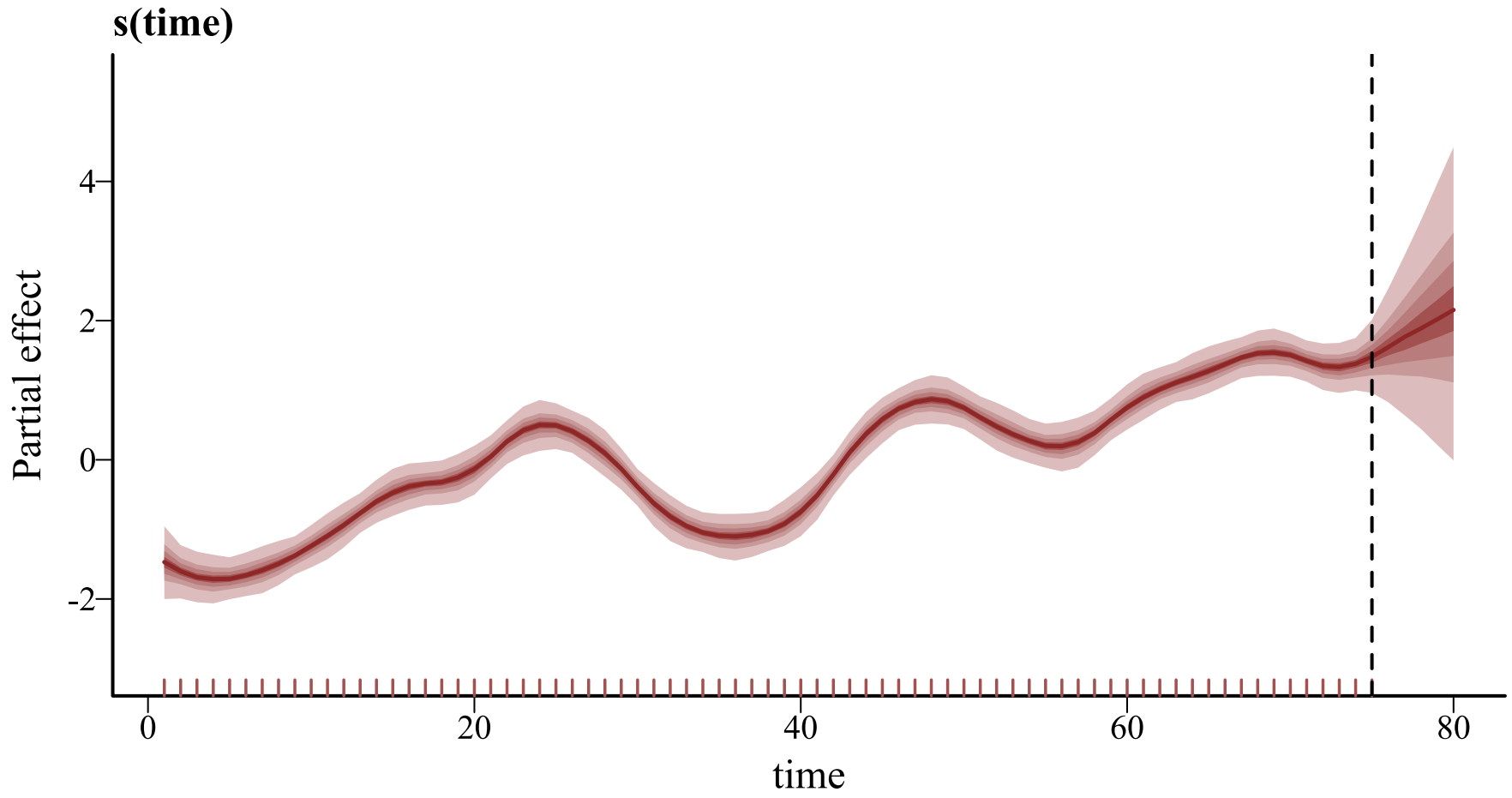
Hindcasts ☺



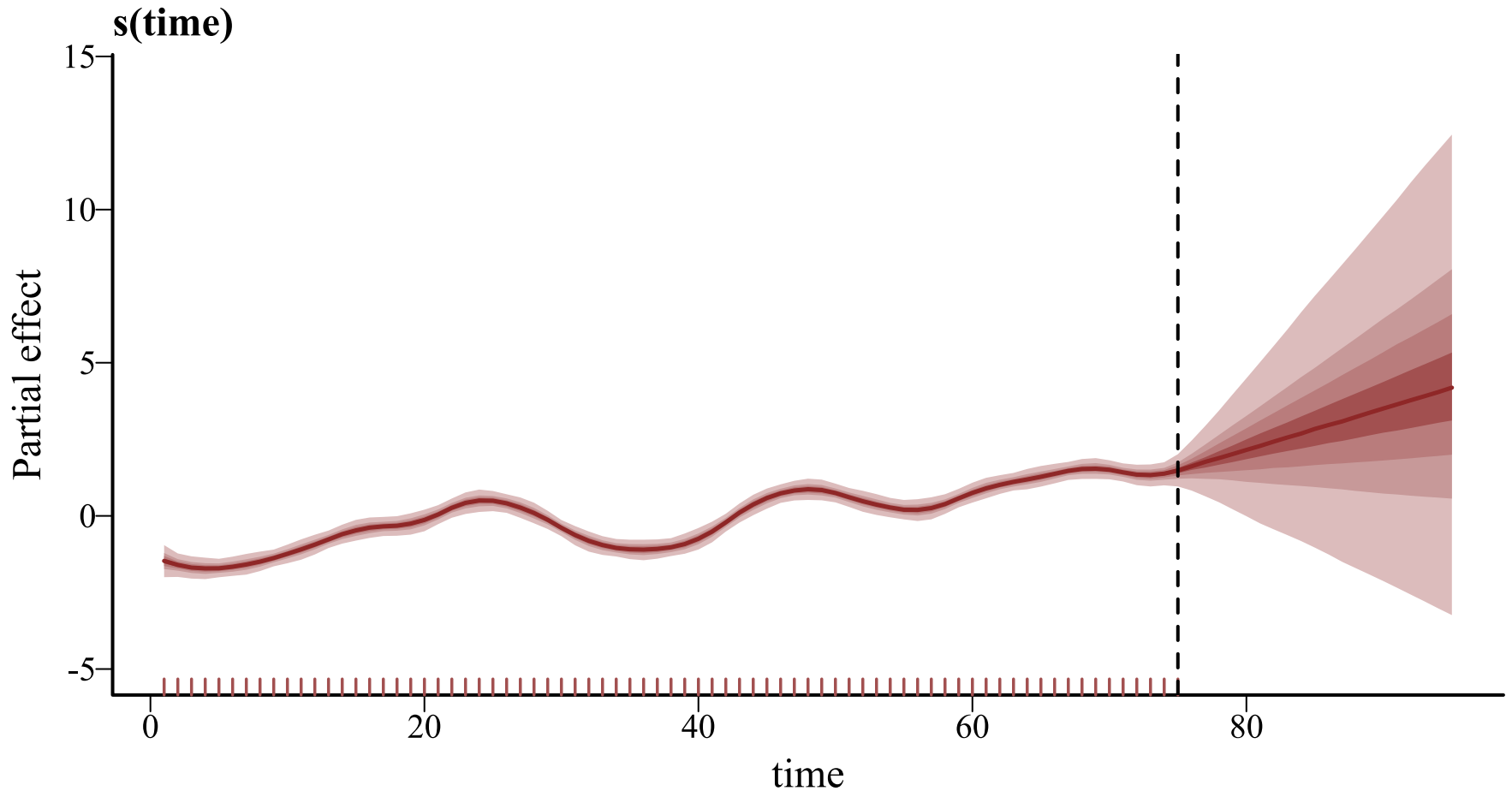
Extrapolate 2-steps ahead ☺



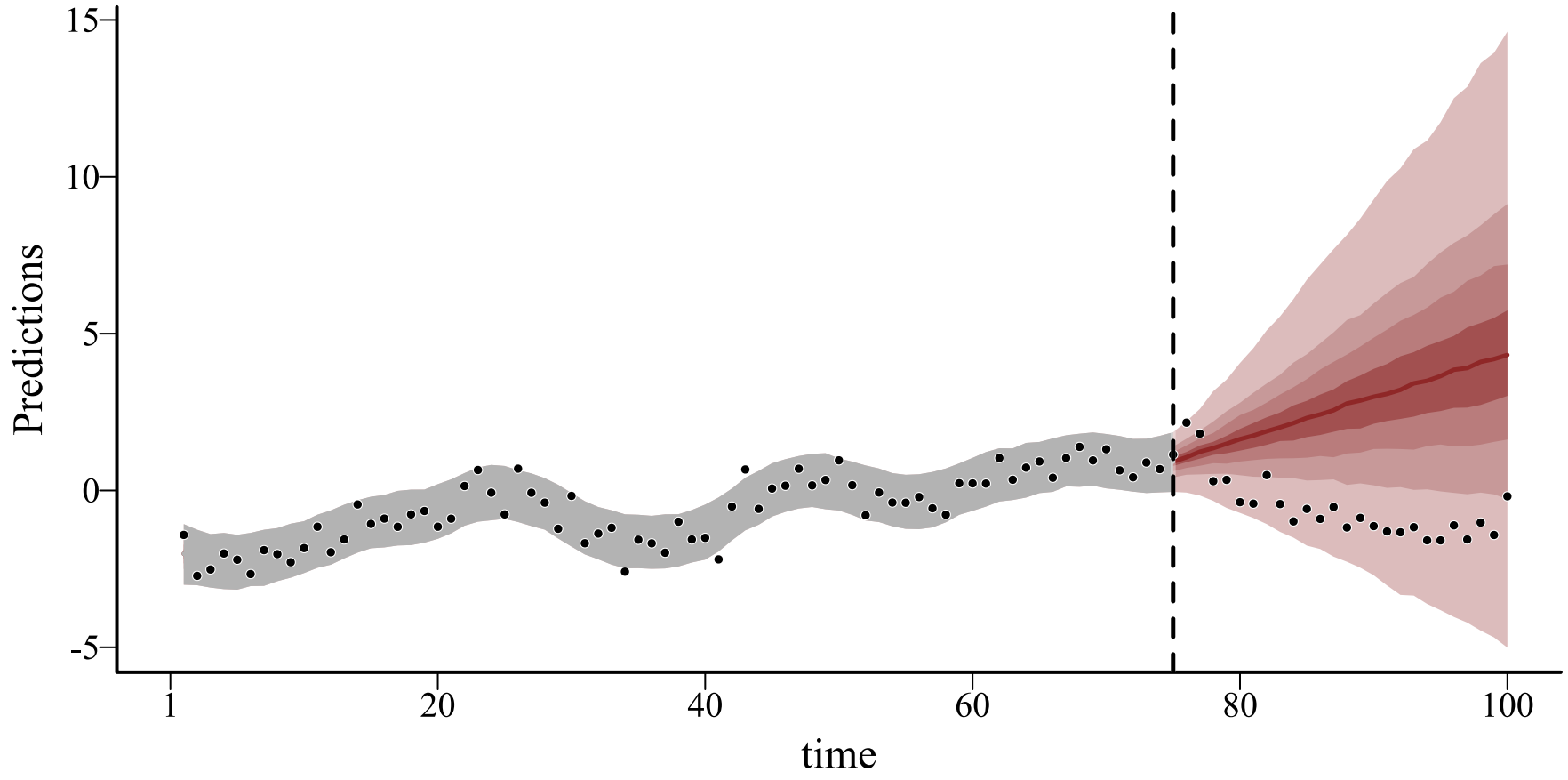
5-steps ahead ☹



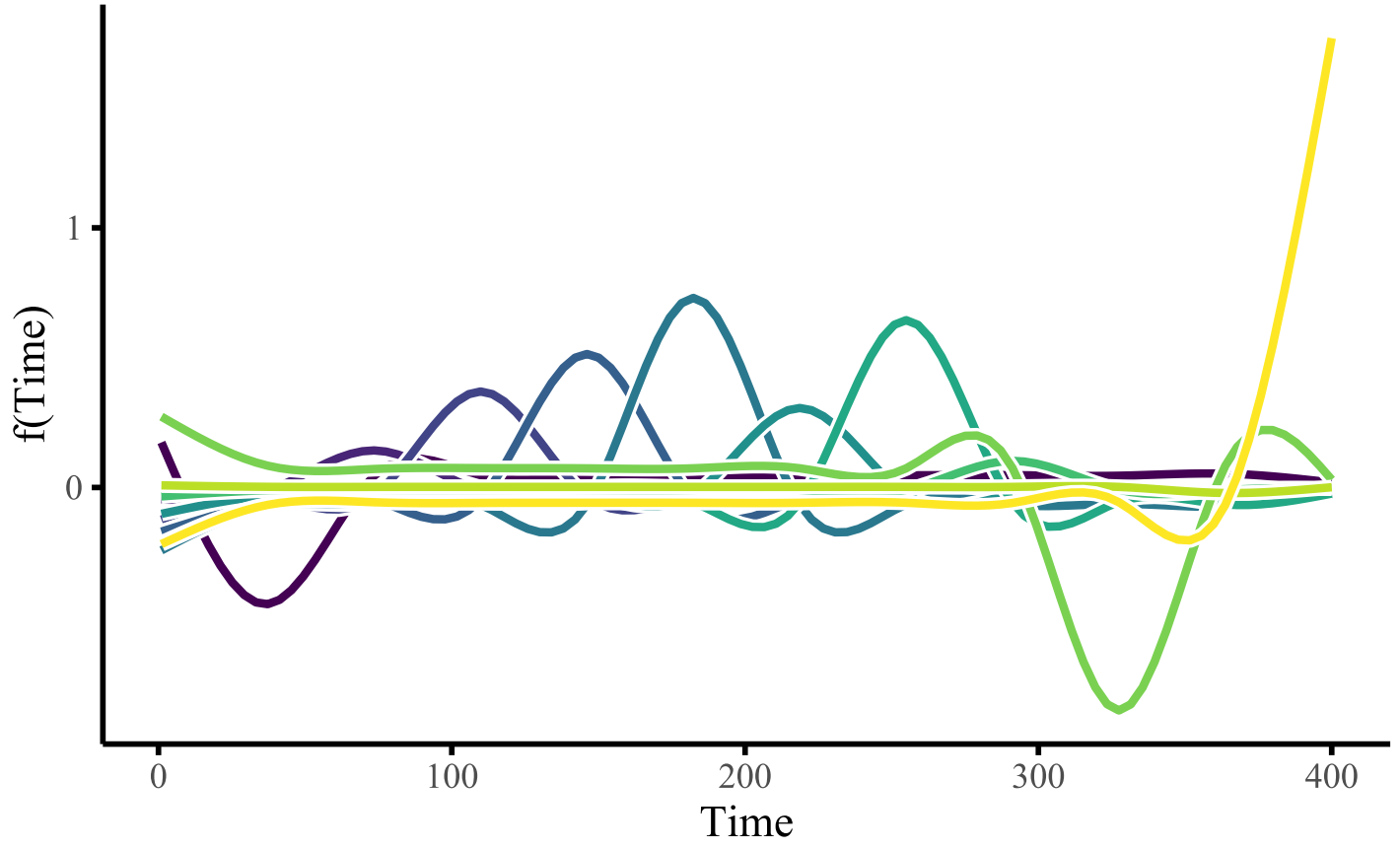
20-steps ahead ☹



Forecasts



Basis functions ⇨ local knowledge



Latent dynamic processes

Latent dynamics in `mvgam` 📦

State-Space models where a latent process evolves to capture unobserved temporal dynamics

- RW
- AR (1-3)
- Gaussian Process (squared exponential kernel)
- Multivariate processes

Can estimate effects of predictors (including splines and random effects) in ***both the observation and latent process models***

Dynamic Poisson GLM

A dynamic Poisson GLM can use ***dynamic latent residuals***

$$
\begin{aligned}
\boldsymbol{Y}_t &\sim \text{Poisson}(\lambda_t) \\
log(\lambda_t) &= \alpha + \cdots + z_t \\
z_t &\sim \text{MVNormal}(0, \Sigma) \\
\Sigma_{t_i,t_j} &= \alpha^2 * exp(-0.5 * ((|t_i - t_j|/\rho))^2)
\end{aligned}
$$

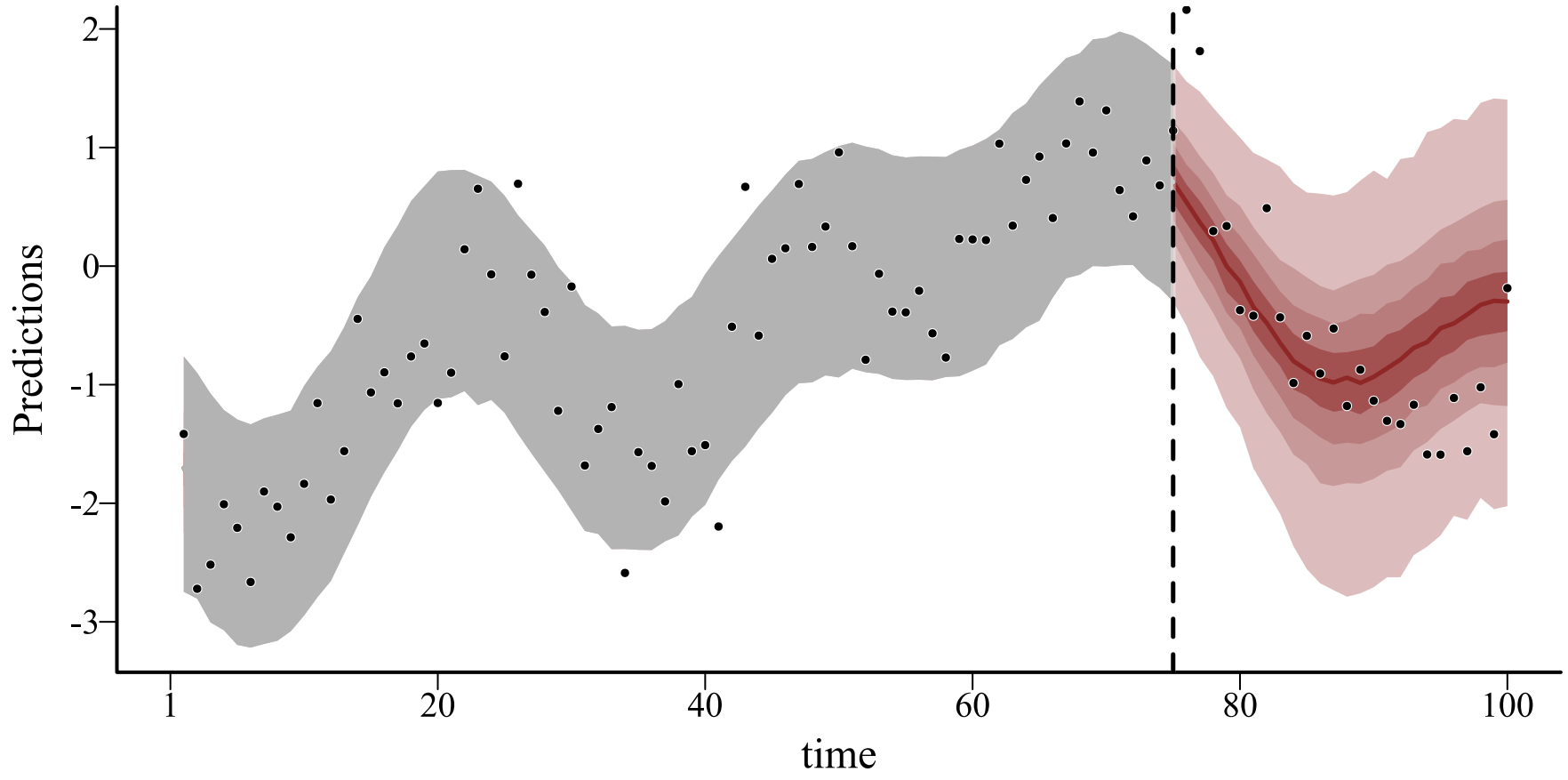
Where:

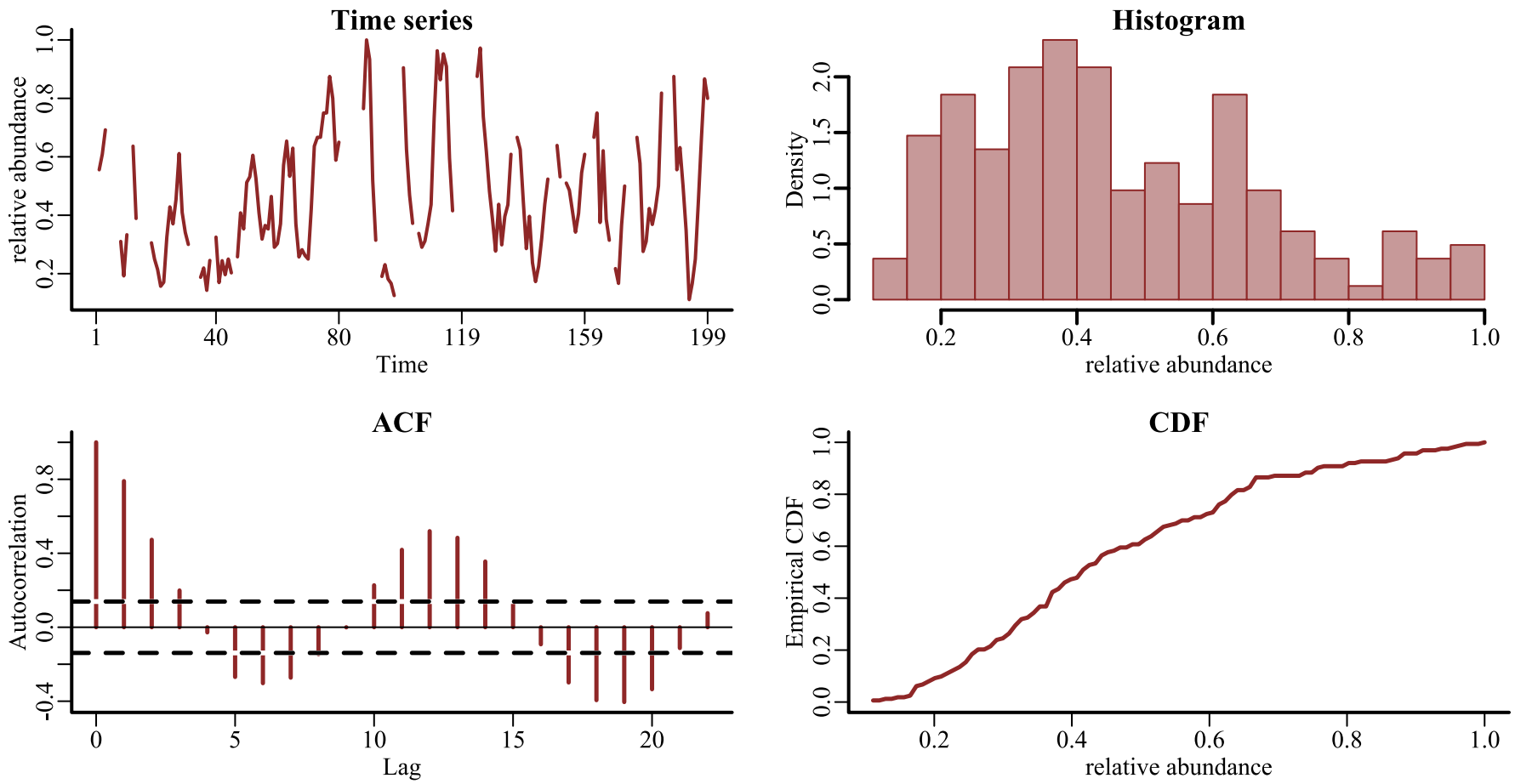
α controls the marginal variability (magnitude) of the function

ρ controls how correlations decay as a function of time lag

Σ is the kernel, in this case a squared exponential kernel

Dynamics ⇨ *global knowledge*





Properties of Merriam's kangaroo rat relative abundance time series from a long-term monitoring study in Portal, Arizona, USA

Dynamic Beta GAM

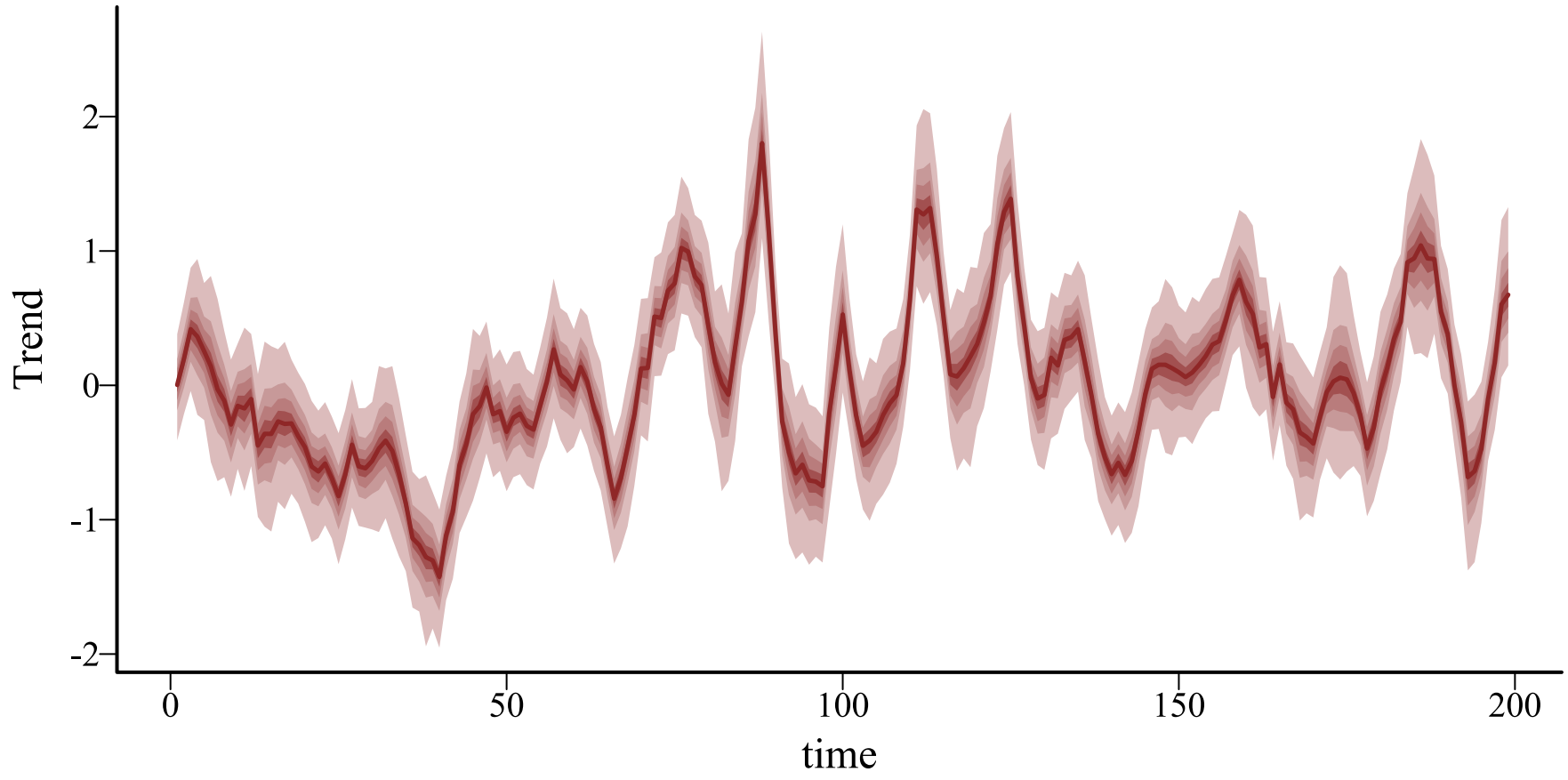
```
mod_beta ← mvgam(relabund ~
                   te(mintemp, ndvi),
                 trend_model = 'AR3',
                 family = betar(),
                 data = dm_data)
```

Beta regression using the `mgcv` 📦 's `betar` family

AR3 dynamic trend model

Multidimensional tensor product smooth function for nonlinear covariate interactions (using `te`)

The latent dynamics

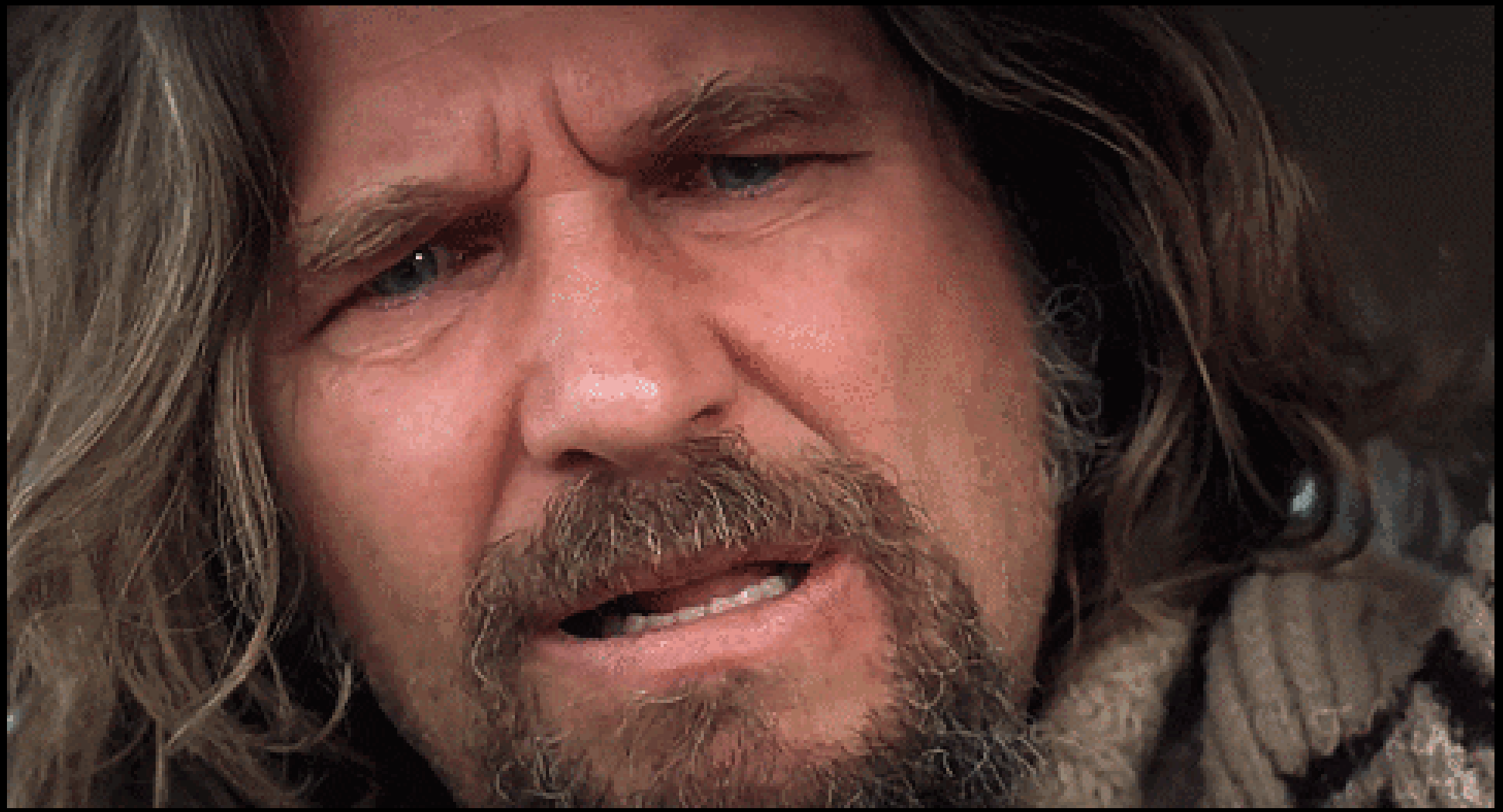


Coefficients?

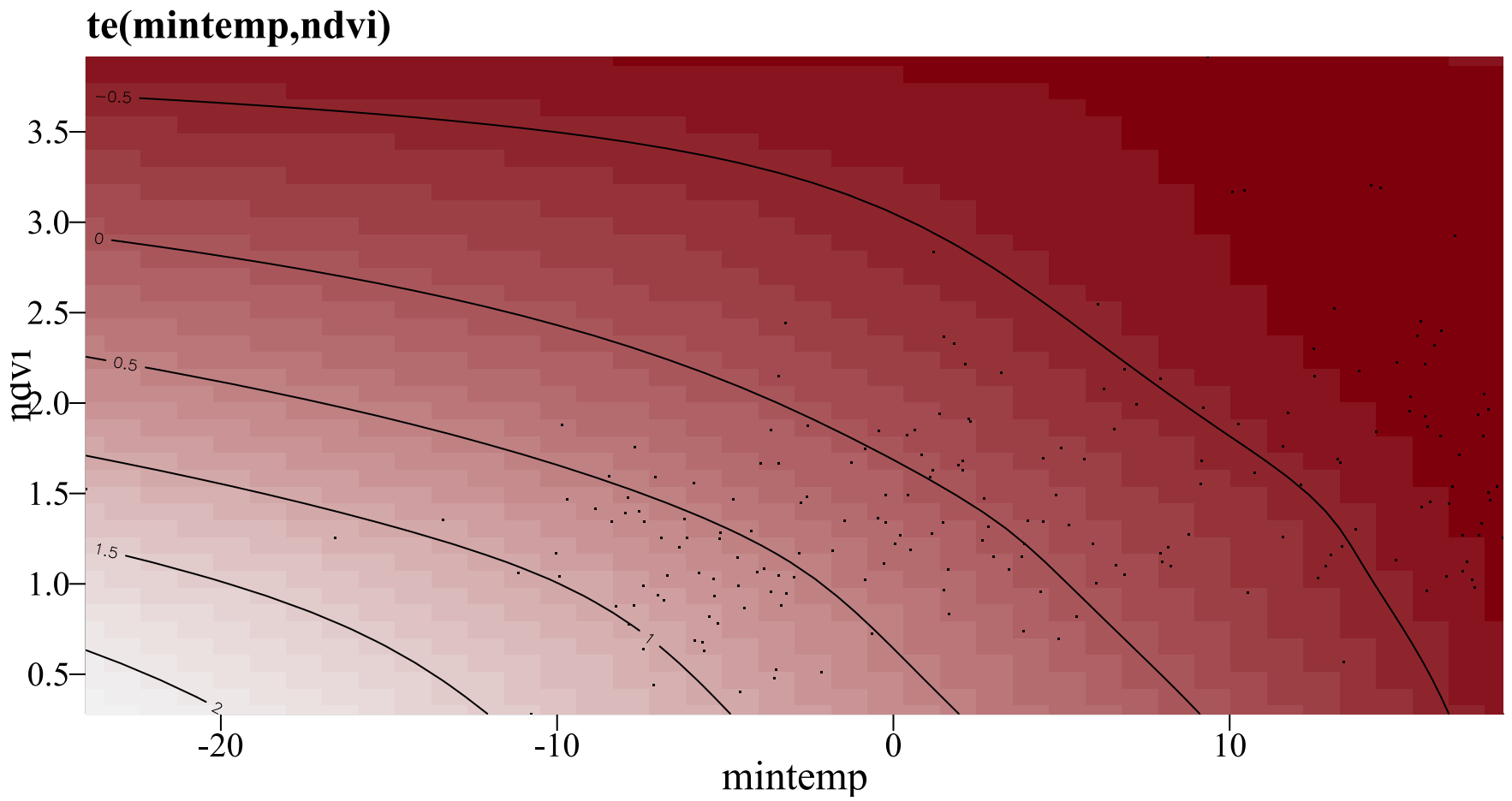
```
coef(mod_beta)
```

```
##                          2.5%         50%        97.5% Rhat n.eff
## (Intercept)         -0.27704943 -0.11949450  0.037417205 1.01   366
## te(mintemp,ndvi).1   0.09777064  1.58677000  2.435108500 1.02   123
## te(mintemp,ndvi).2   0.12734882  1.28906000  2.018242500 1.02   142
## te(mintemp,ndvi).3  -0.02043112  0.87295550  1.707162750 1.01   385
## te(mintemp,ndvi).4  -3.80155025 -0.62739600  1.895769000 1.00   822
## te(mintemp,ndvi).5   0.20000635  0.65926500  1.102021500 1.01   351
## te(mintemp,ndvi).6  -1.87098075 -0.78675750  0.713037000 1.03   136
## te(mintemp,ndvi).7  -0.44178235 -0.05658210  0.489970700 1.03   148
## te(mintemp,ndvi).8  -0.72445805 -0.29076650  0.232700575 1.02   188
## te(mintemp,ndvi).9  -2.08312200 -0.68123850  0.510886500 1.00   600
## te(mintemp,ndvi).10  0.04609435  0.50692150  0.930659100 1.00   795
## te(mintemp,ndvi).11 -0.60098247 -0.21375750  0.212092350 1.02   145
## te(mintemp,ndvi).12 -0.21238740 -0.05144905  0.114180750 1.01   273
## te(mintemp,ndvi).13 -1.62903200 -0.96678800  0.015499635 1.02   136
## te(mintemp,ndvi).14 -1.75806450 -0.72884100  0.195621150 1.01   410
## te(mintemp,ndvi).15 -0.96620883 -0.21764150  0.540176950 1.00   299
## te(mintemp,ndvi).16 -1.71074775 -1.01050500 -0.005658116 1.02   123
## te(mintemp,ndvi).17 -0.52369400 -0.26230500 -0.020589035 1.02   180
## te(mintemp,ndvi).18 -2.60926150 -1.66896500 -0.155125775 1.02   126
```

How do I report this garbage?



Plot the smooth?



Credit @stephenjwild

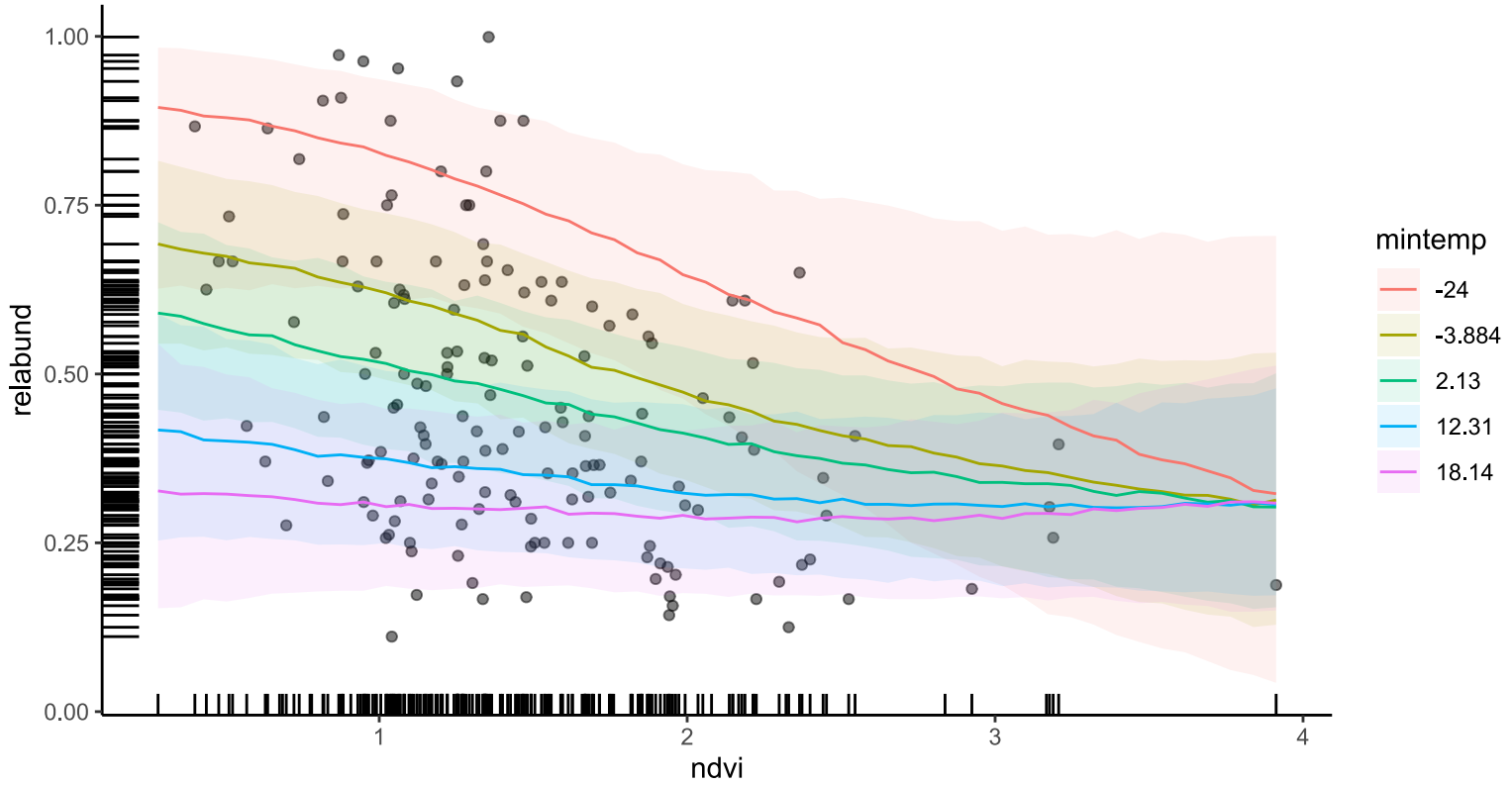
marginaleffects for clarity

Code Plot

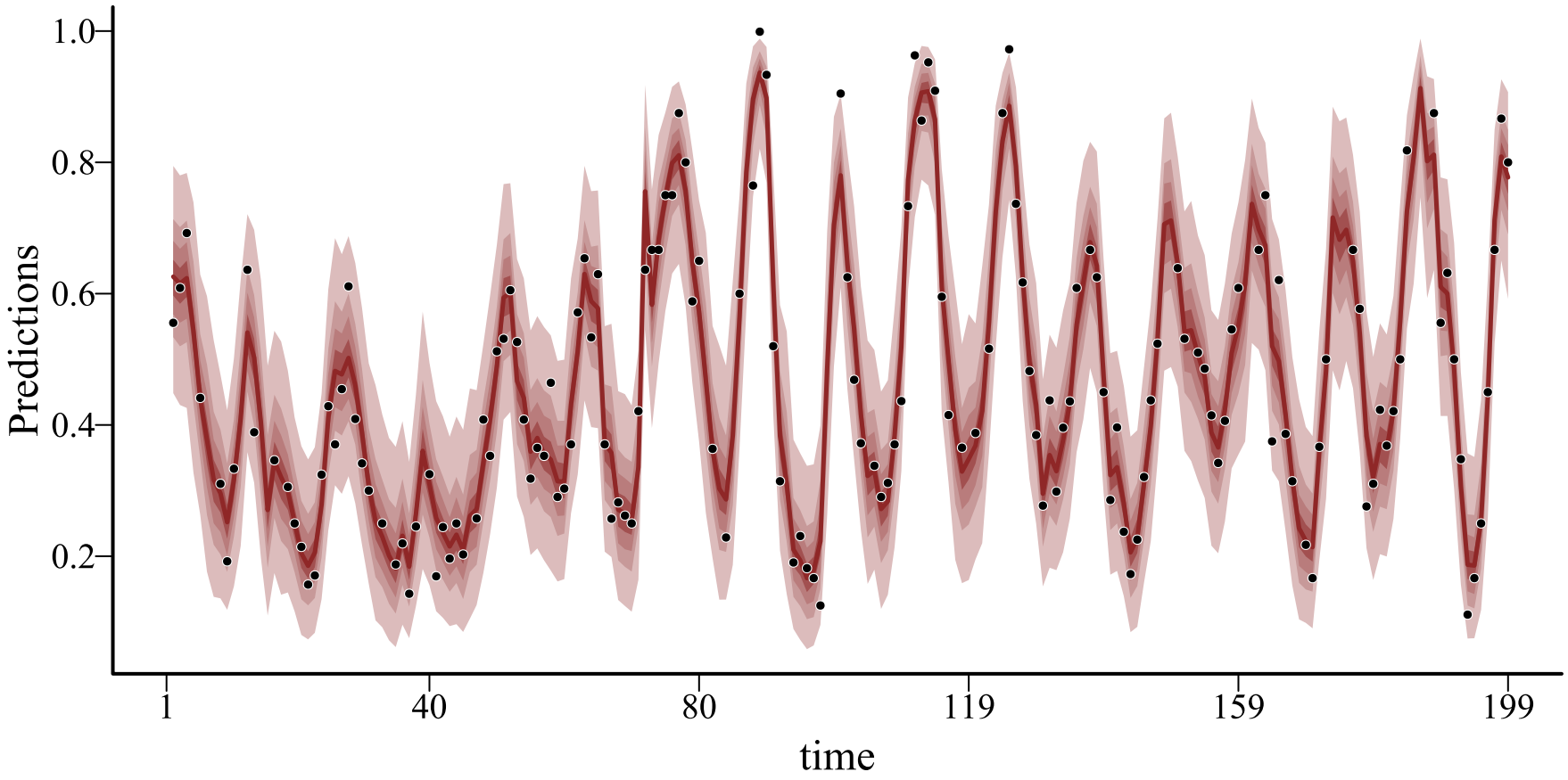
```
# plot conditional effect of BOTH covariates on the outcome scale
plot_predictions(mod_beta, condition = c('ndvi', 'mintemp'),
                 points = 0.5, conf_level = 0.8, rug = TRUE) +
  theme_classic()
```

marginaleffects for clarity

Code Plot



Hindcasts 😊



Multivariate ecological time series

Hierarchical *nonlinear* effects?

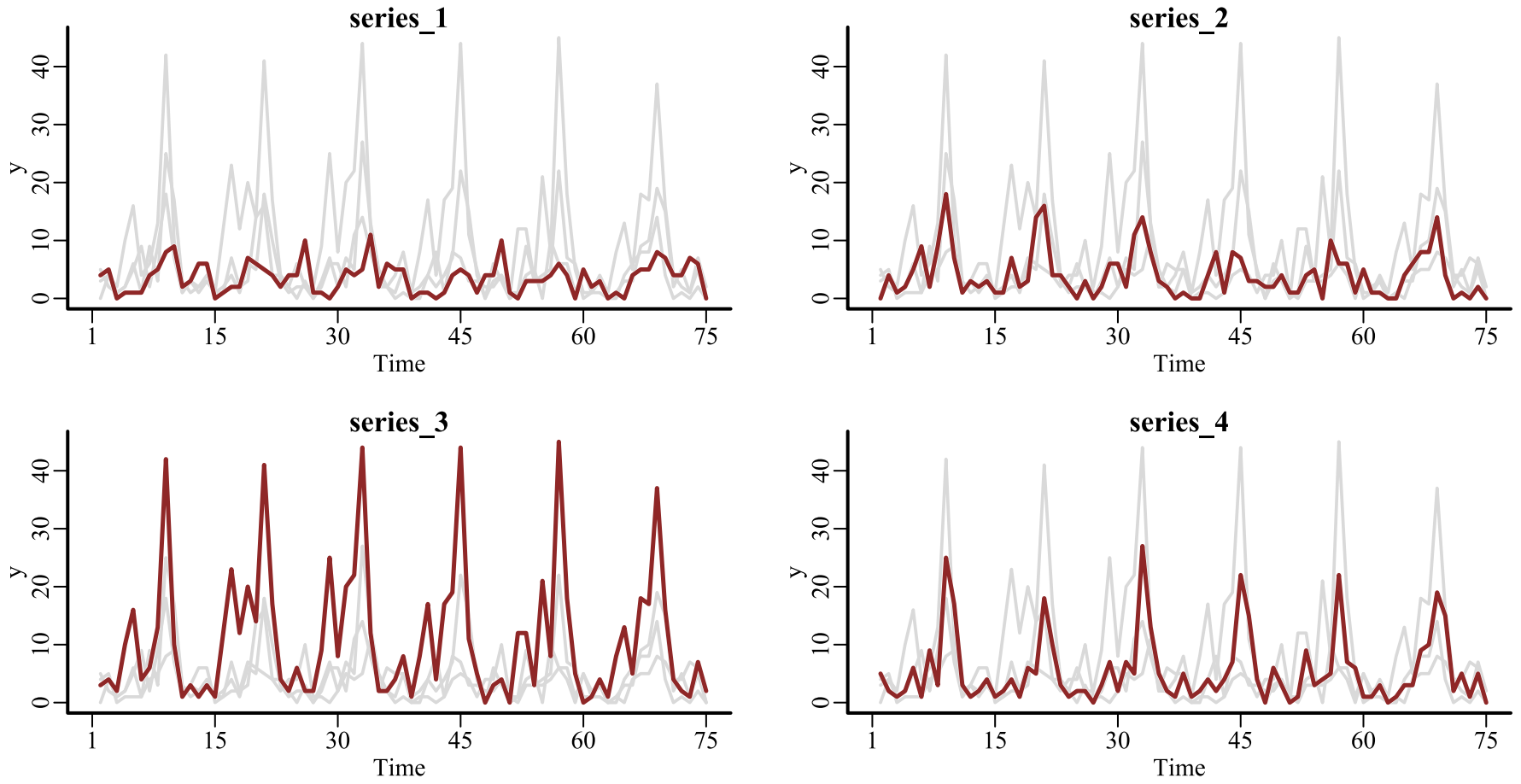
We very often expect to encounter nonlinear effects in ecology

But if we measure multiple species / plots / individuals etc.. through time, we can also encounter hierarchical nonlinear effects

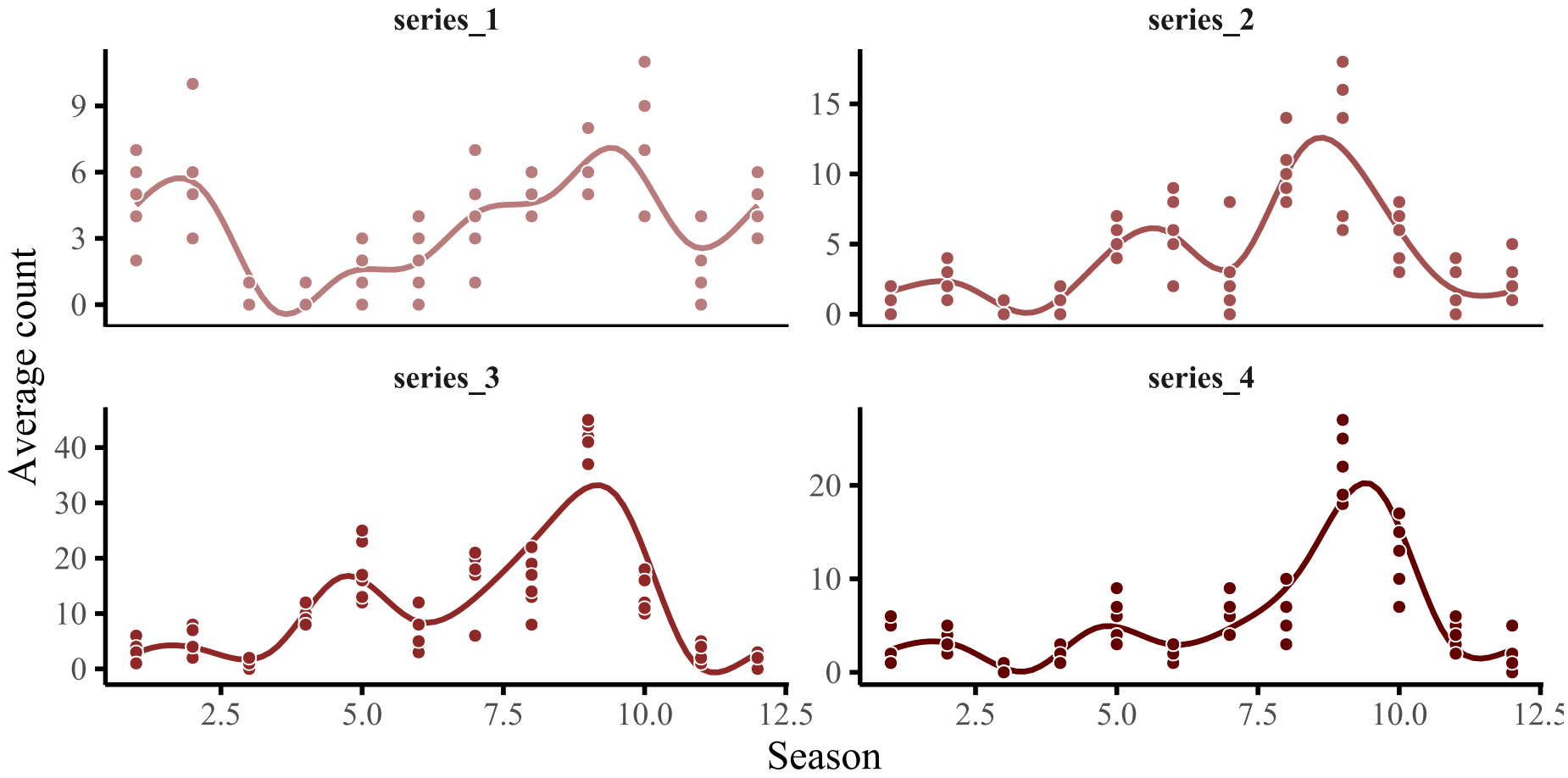
- Same species may respond similarly to environmental change over different sites
- Different species may respond similarly in the same site

Our data may not be rich enough to estimate all effects individually; so what can we do?

Example: simulated data



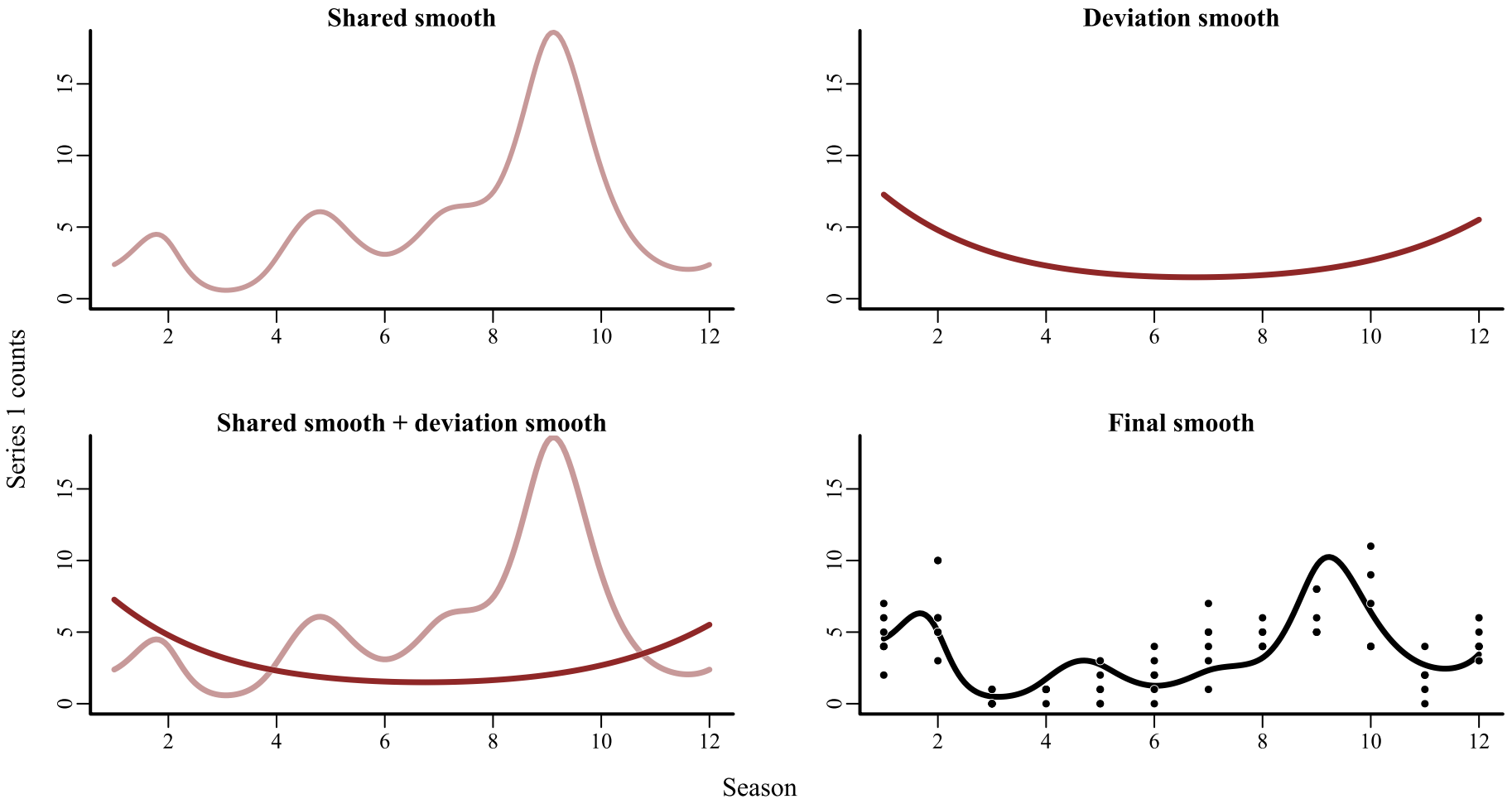
Similar seasonal patterns



Can we somehow estimate the average population smooth *and* a smooth to determine how each series deviates from the population?

Yes! We can use hierarchical GAMs

Decomposing seasonality



How did we model this?

```
mod ← mvgam(y ~
              s(season, bs = 'cc', k = 12) +
              s(season, series, k = 4, bs = 'fs'),
            data = data,
            family = poisson())
```

A ***shared*** smooth of seasonality

This is a group-level smooth, similar to what we might expect the average seasonal function to be in this set of series

How did we model this?

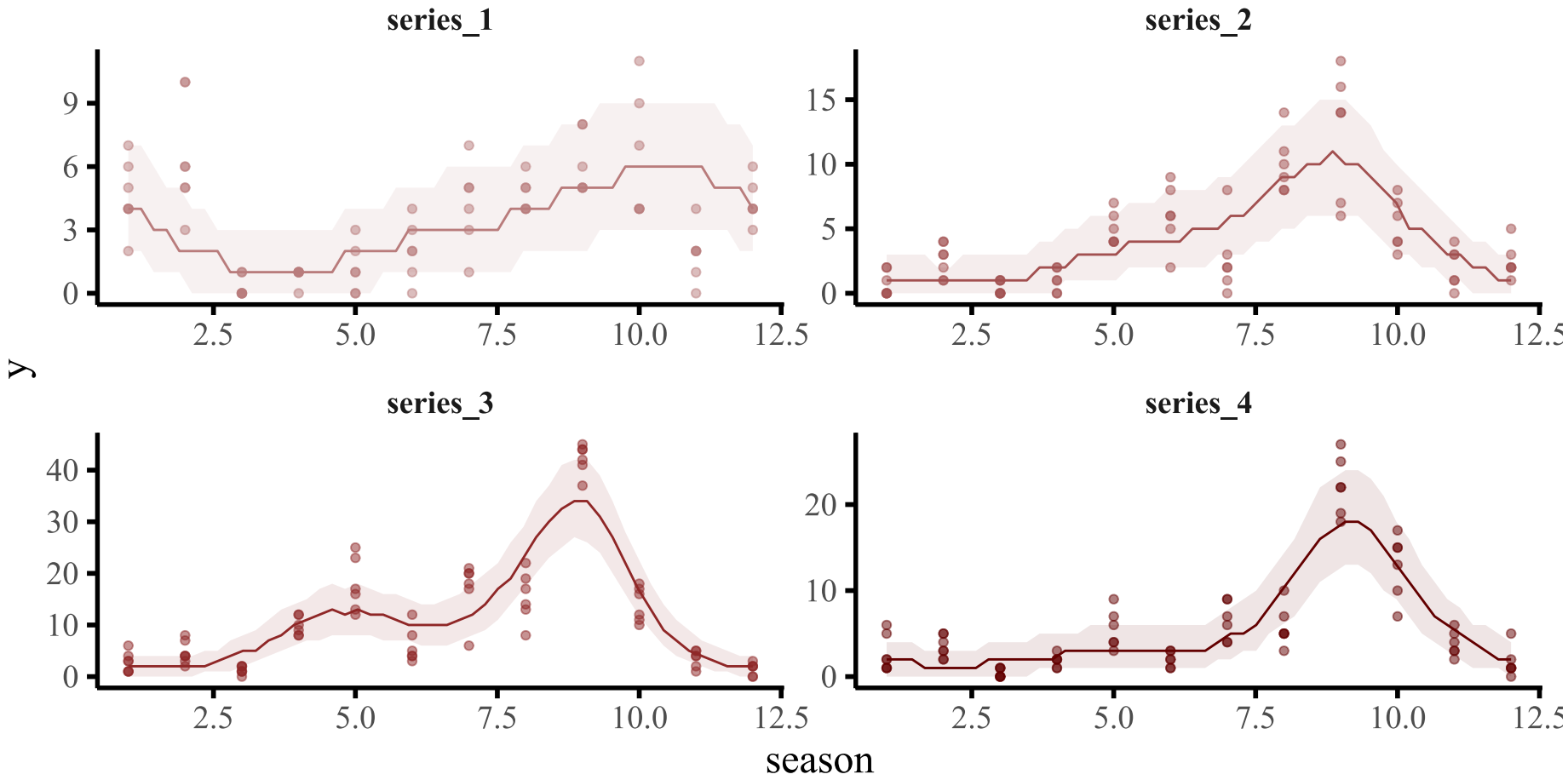
```
mod ← mvgam(y ~
              s(season, bs = 'cc', k = 12) +
              s(season, series, k = 4, bs = 'fs'),
            data = data,
            family = poisson())
```

Series-level ***deviation*** smooths of seasonality, which all share a common smoothing penalty

These are individual-level smooths that capture how each series' seasonal pattern differs from the shared smooth

- There are a number of ways to do this using splines
- See Pedersen et al 2019 for useful guidance

Conditional predictions



Posterior contrasts

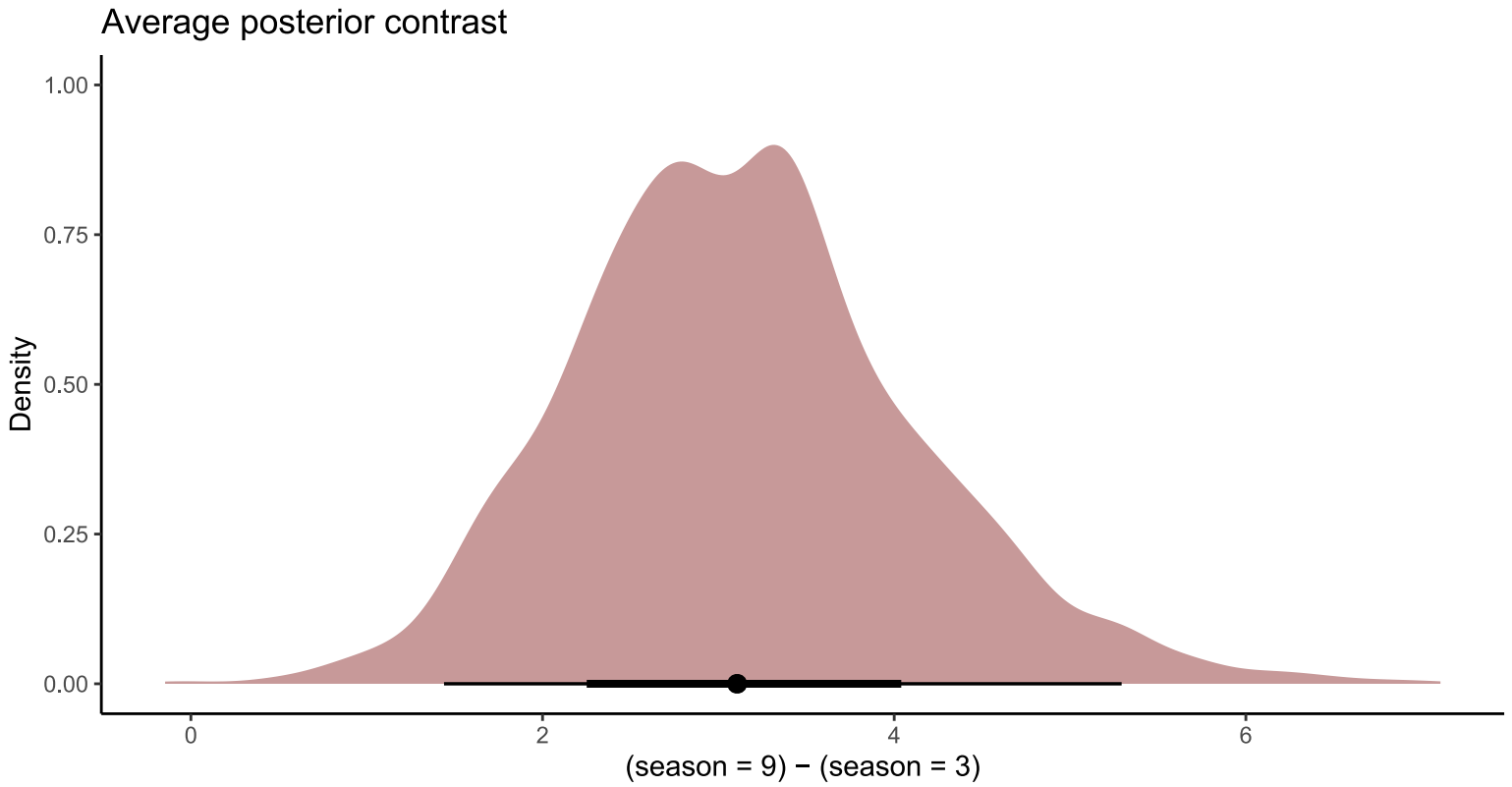
Code Plot

```
# take draws of average comparison between season = 9 vs season = 3
post_contrasts ← avg_comparisons(model,
                                 variables = list(season = c(9, 3)),
                                 proces_error = FALSE) %>%
  posteriordraws()

# use the resulting posterior draw object to plot a density of the
# posterior contrasts
library(tidybayes)
post_contrasts %>% ggplot(aes(x = draw)) +
  # use the stat_halfeye function from tidybayes for a nice visual
  stat_halfeye(fill = "#C79999") +
  labs(x = "(season = 9) - (season = 3)", y = "Density",
       title = "Average posterior contrast") + theme_classic()
```

Posterior contrasts

Code Plot

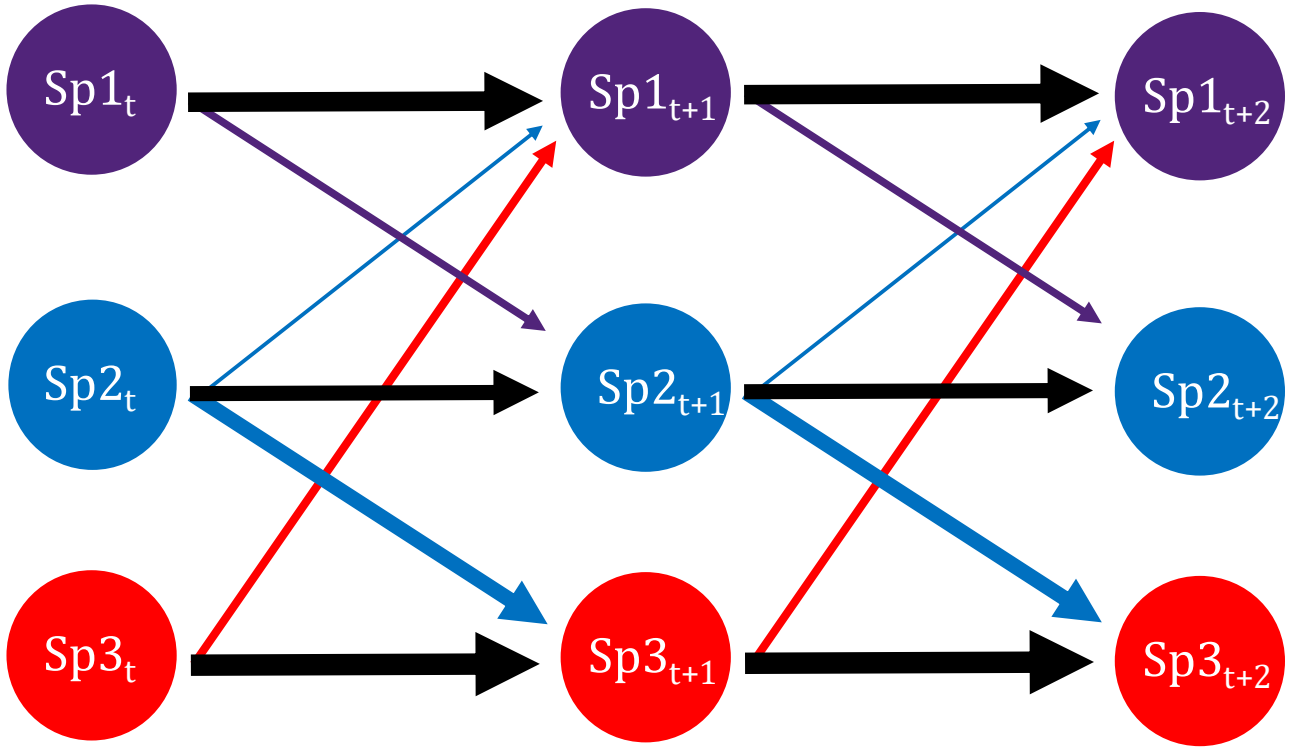


HGAMs offer a solution to estimate the hierarchical, nonlinear effects that we think are common in ecology

This is a huge advantage over traditional time series models

But how can we handle multivariate *dynamic components*?

VARs ⇨ Granger causality



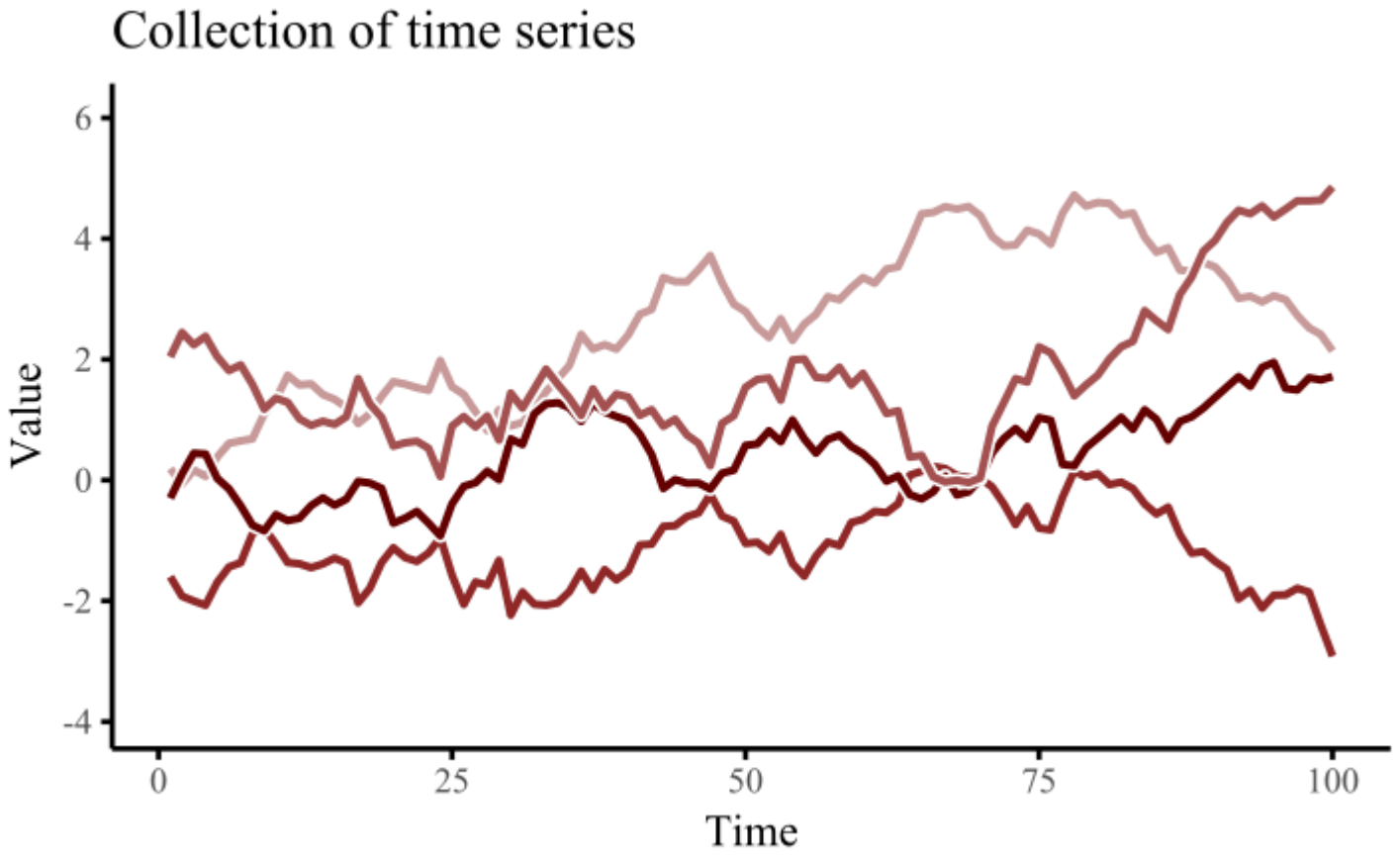
Latent VAR1s in mvgam 📦

```
varmod ← mvgam(y ~ 1,
               trend_model = 'VAR1',
               data = data_train,
               newdata = data_test,
               family = gaussian())
```

If multiple series are included in the data, we can use a VAR1 to estimate latent dynamics

- `trend_model = 'VAR1'`: a VAR1 with uncorrelated process errors (off-diagonals in Σ set to zero)
- `trend_model = 'VAR1cor'`: a VAR1 with possibly correlated process errors

Factors ⇨ induced correlations



Dynamic factors in mvgam 📦

```
dfmod ← mvgam(y ~ 1,
              use_lv = TRUE,
              n_lv = 2,
              trend_model = 'AR',
              data = data_train,
              newdata = data_test,
              family = gaussian())
```

If multiple series are included in the data, we can use a dynamic factor model to estimate latent dynamics

- `n_lv`: the number of latent factors to estimate
- `trend_model`: can be `RW`, `AR1`, `AR2`, `AR3` or `GP`
- factor variances fixed to ensure identifiability

Other uses of `mvgam` 📦

Multiseries models with shared latent states

Uni- and multivariate proper scoring rules for forecast evaluation

Randomised Quantile Residuals to inspect model misspecification

Gaussian Processes to estimate smooth dynamics and time-varying coefficients

User-specified priors for all key parameters

See more on the package website

Thank you