

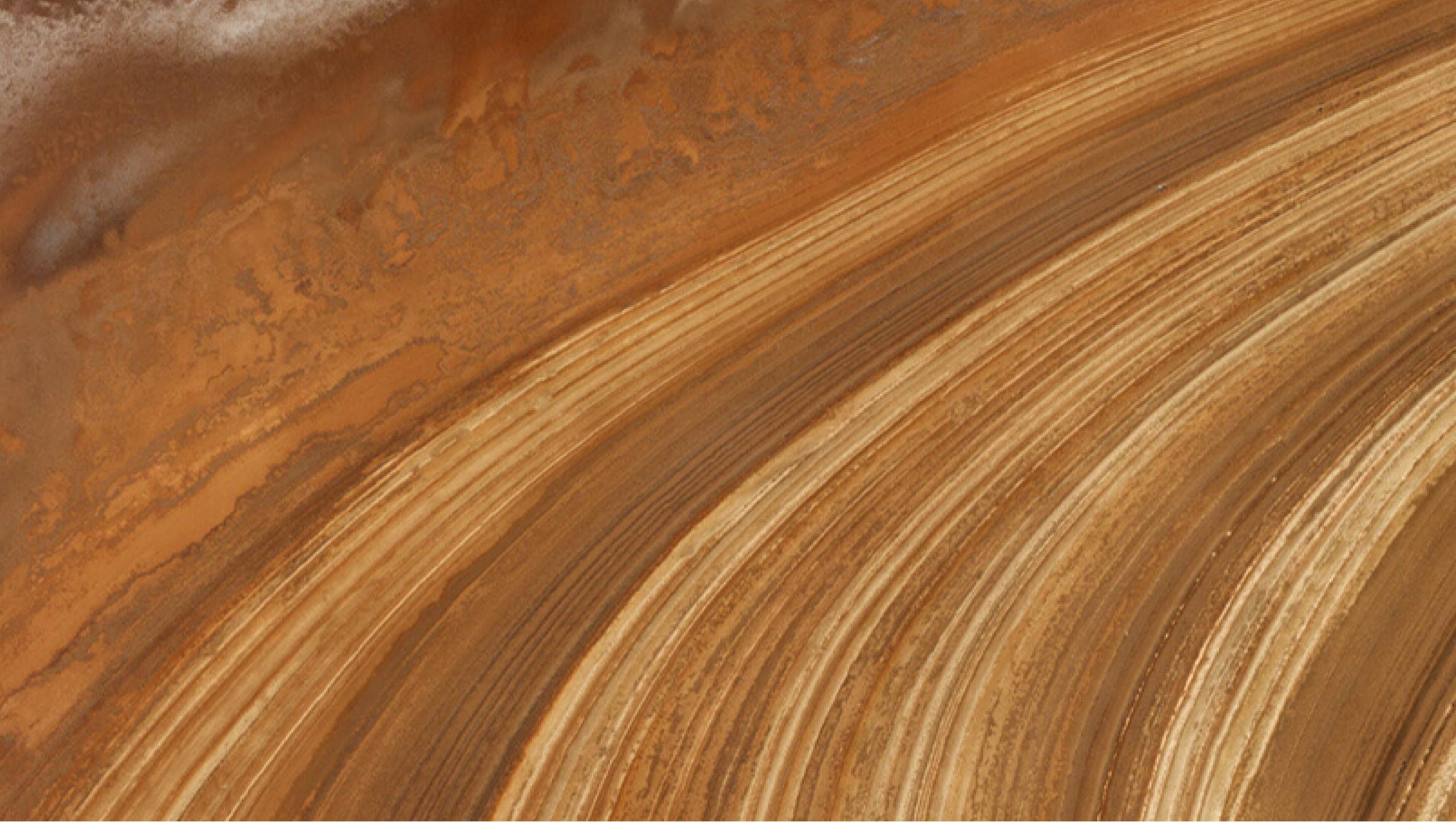
# Ecological forecasting in R

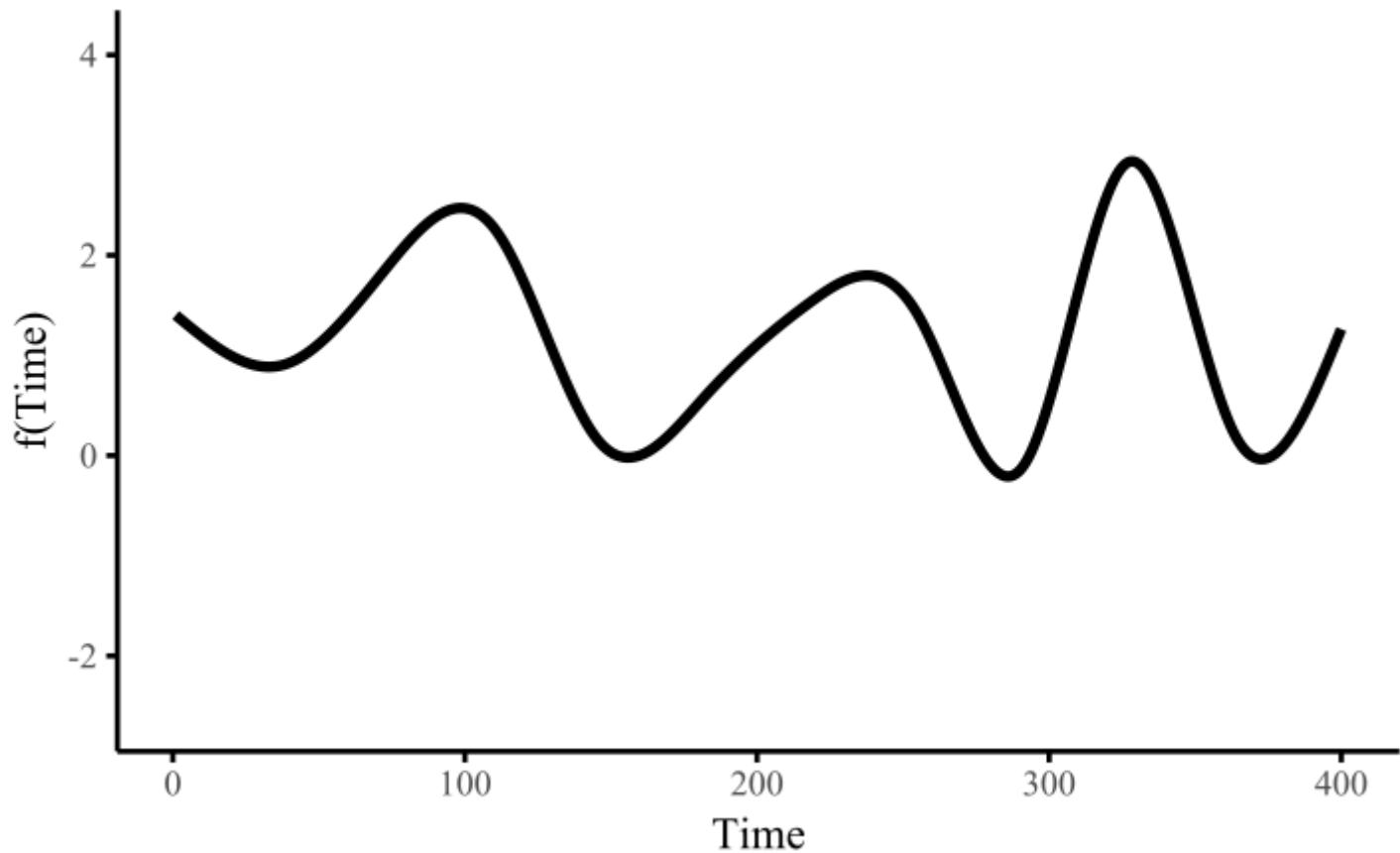
Lecture 2: dynamic GLMs and GAMs

Nicholas Clark

School of Veterinary Science, University of Queensland

0900–1200 CET Monday 4th September, 2023





# Workflow

Press the "o" key on your keyboard to navigate among slides

Access the [tutorial html here](#)

Download the data objects and exercise  script from the html file

Complete exercises and use Slack to ask questions

Relevant open-source materials include:

[An introduction to Bayesian multilevel modeling with brms](#)

[Introduction to Generalized Additive Models with !\[\]\(ec9132f1d27c8919987d92907322654d\_img.jpg\) and mgcv](#)

[Statistical Rethinking 2023 - 04 - Categories & Curves](#)

[Statistical Rethinking 2023 - 12 - Multilevel Models](#)

# This lecture's topics

Useful probability distributions for ecologists

Generalized Linear and Additive Models

Temporal random effects

Temporal residual correlation structures

**When applying statistical modelling to a time series, we aim to estimate parameters for a collection of probability distributions**

**These distributions are indexed by *time* (i.e. the observations are random draws from a set of time-varying distributions)**

**Usually we allow the mean of these distributions to vary over time. But what kinds of distributions are available to us?**

# Useful probability distributions

# Normal (Gaussian)

$$Y_t \sim \text{Normal}(\mu_t, \sigma)$$

## Properties

Real-valued continuous observations (including any decimal)

Unbounded (supports  $-\infty$  to  $\infty$ )

Symmetric spread, controlled by  $\sigma$ , about the mean ( $\mu_t$ )

Nearly all common time series models assume this data distribution

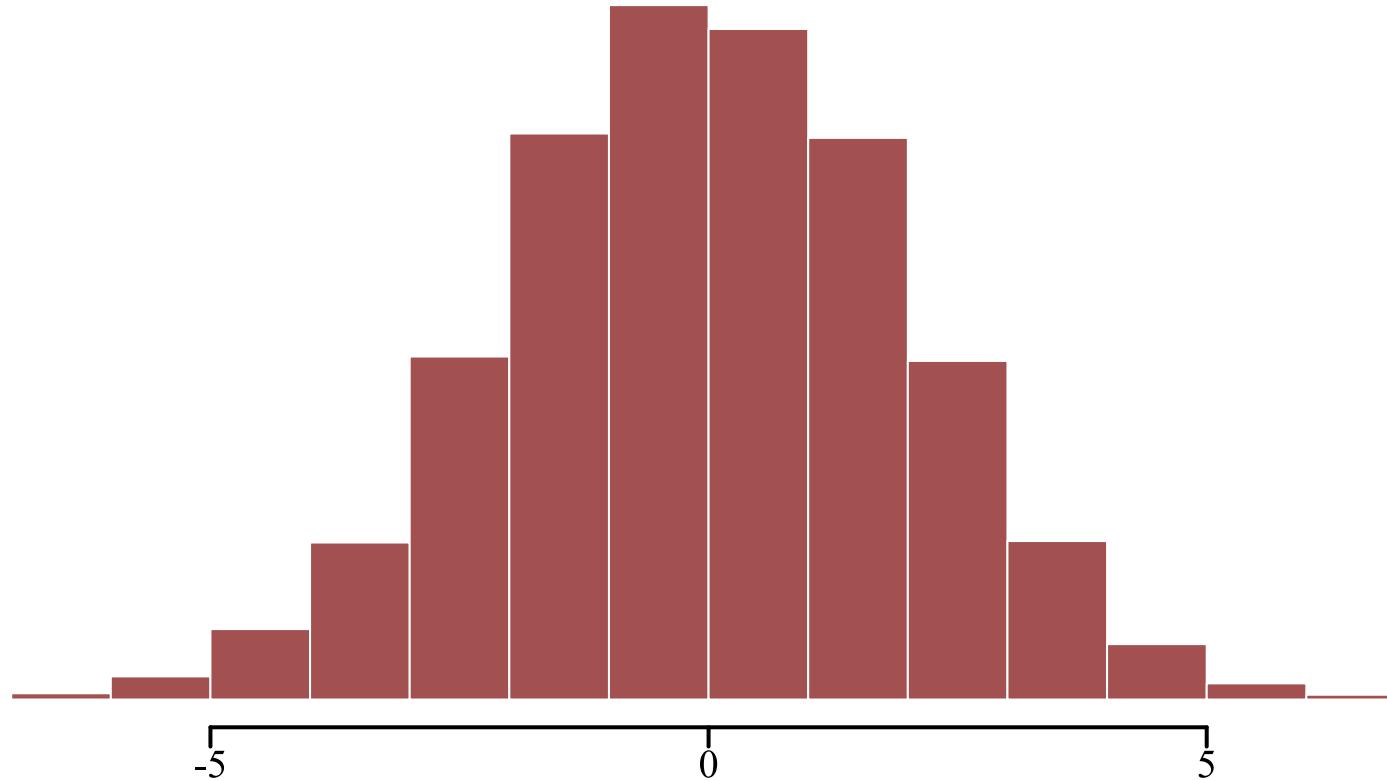
RW, AR, and ARIMA

ETS and TBATS

Meta's Prophet 

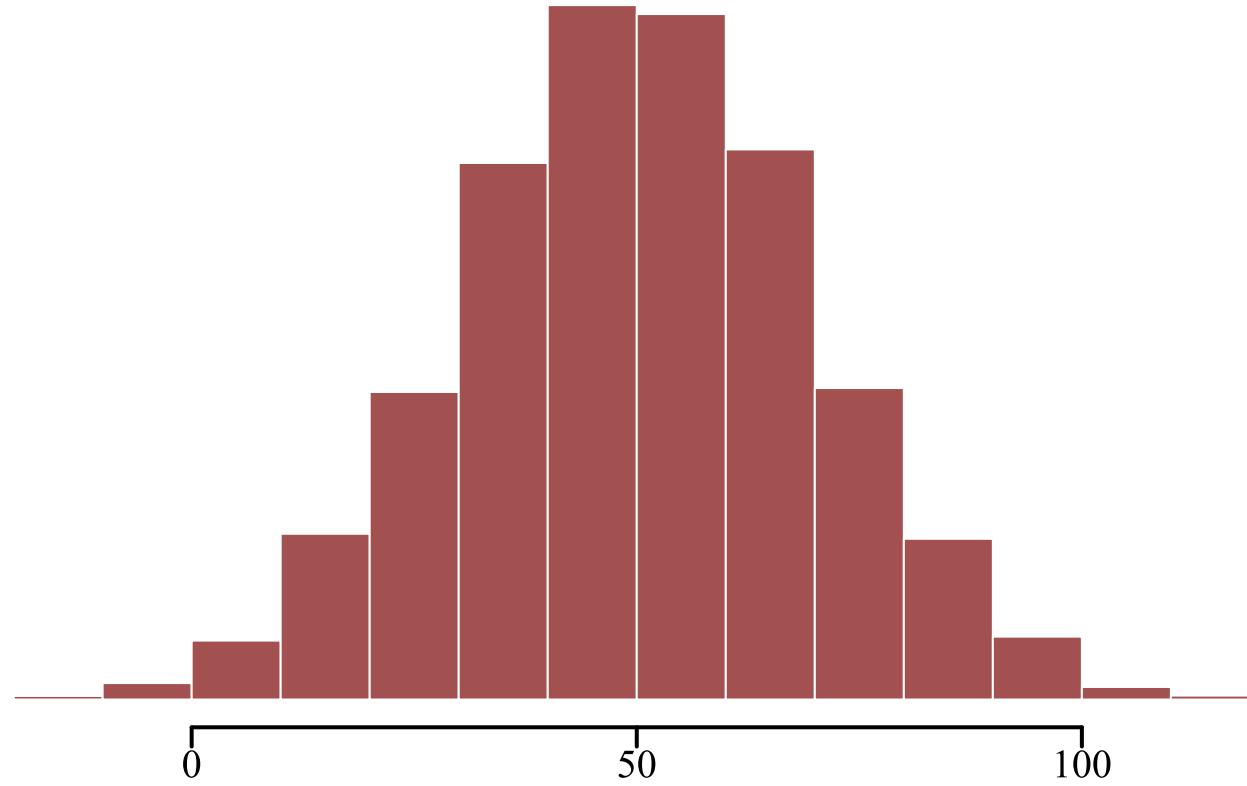
# Normal (Gaussian)

$$Y_t \sim \text{Normal}(0, 2)$$



# Normal (Gaussian)

$$Y_t \sim \text{Normal}(50, 20)$$



# Linear regression

It is common to estimate linear predictors of  $\mu$  with **regression**

$$\mathbf{Y}_t \sim \text{Normal}(\alpha + \beta * \mathbf{X}_t, \sigma)$$

Where:

$\mathbf{X}_t$  represents a design matrix of covariates that contribute linearly to variation in  $\mu_t$

$\alpha$  is an intercept coefficient

$\beta$  is a vector of regression coefficients

$\sigma$  controls the spread of the errors about  $\mu_t$

# ETS(A,A,A)

Exponential smoothing with additive components for trend, seasonality and error assumes a Normal (Gaussian) distribution

$$Y_t \sim \text{Normal}(l_{t-1} + b_{t-1} + s_{t-m}, \sigma)$$

Where:

$l$  gives the value of the level

$b$  gives the value of the trend

$s$  gives the value of the seasonality

$m$  represents the seasonal period

# ARMA( $p, q$ )

ARMA processes also assume Normality

$$Y_t \sim \text{Normal}(c + \sum_{k=1}^p \phi_k(Y_{t-k} - c) + \sum_{i=1}^q \theta_i \epsilon_{t-i}, \sigma)$$

Where:

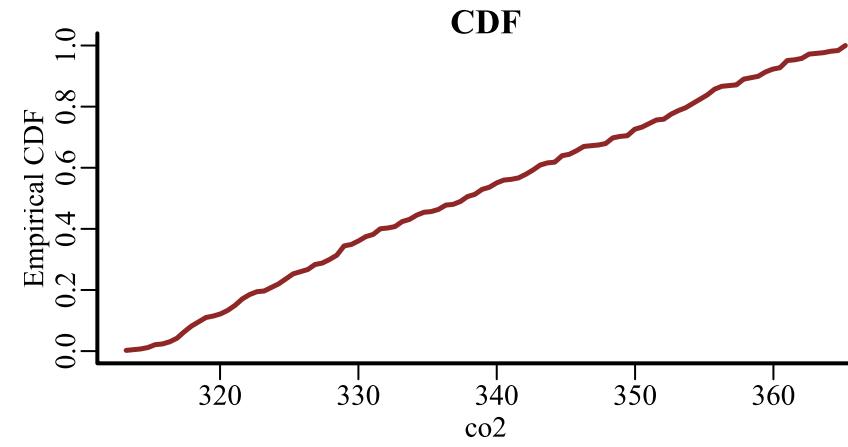
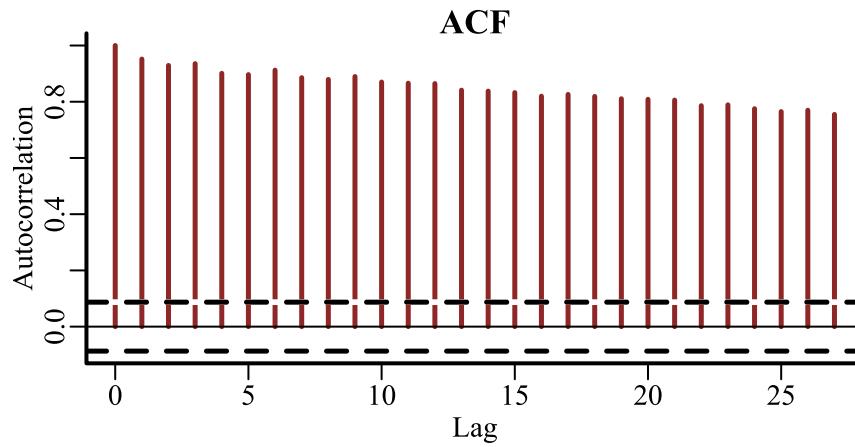
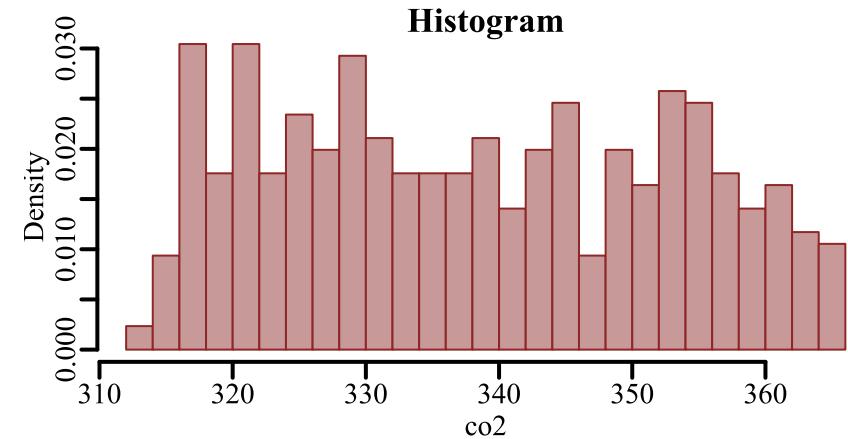
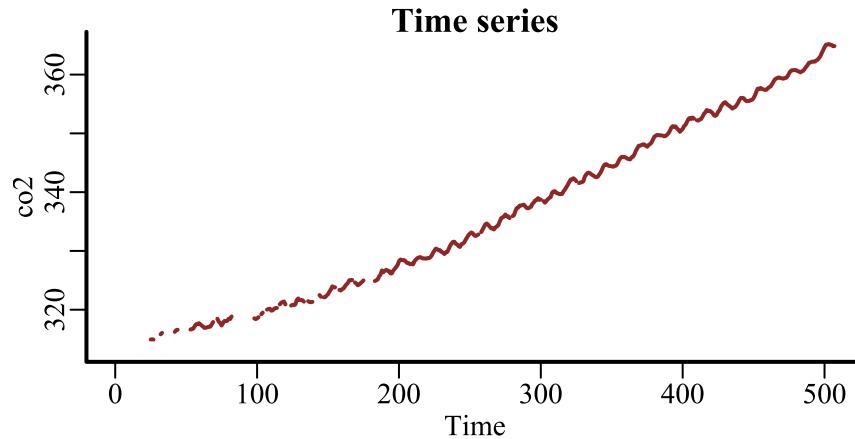
$c$  is a constant (drift parameter)

$p$  and  $q$  gives orders of AR and MA processes

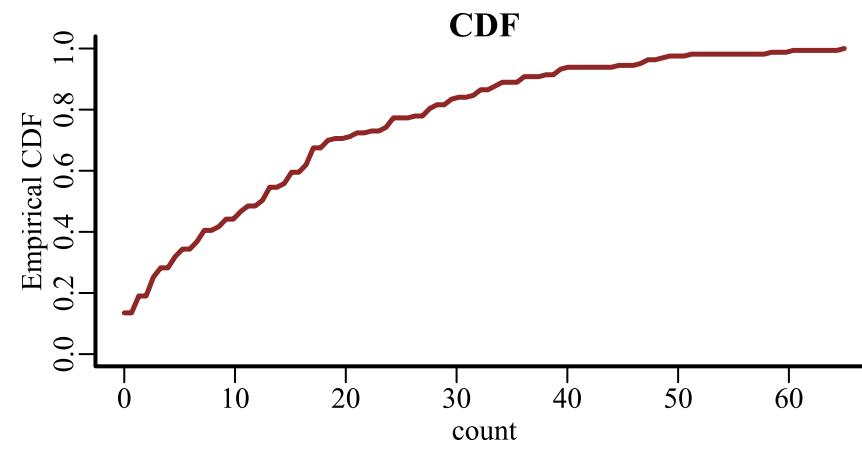
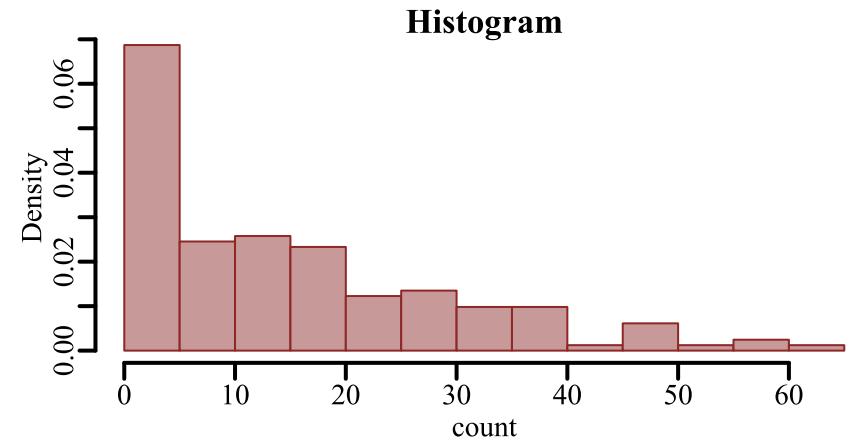
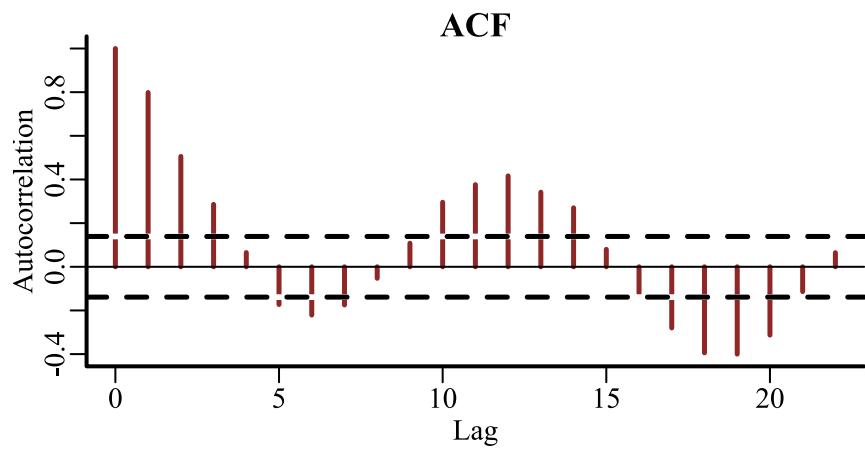
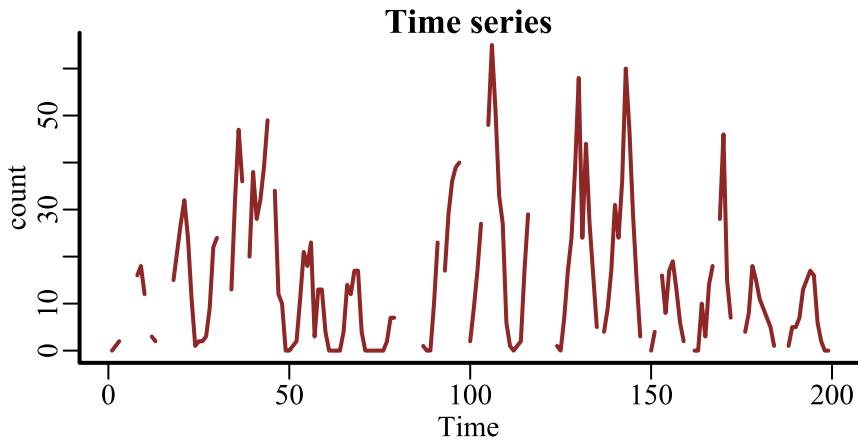
$\phi$  and  $\theta$  are AR and MA coefficients

$\epsilon$  are historical errors (which are  $\text{Normal}(0, \sigma)$ )

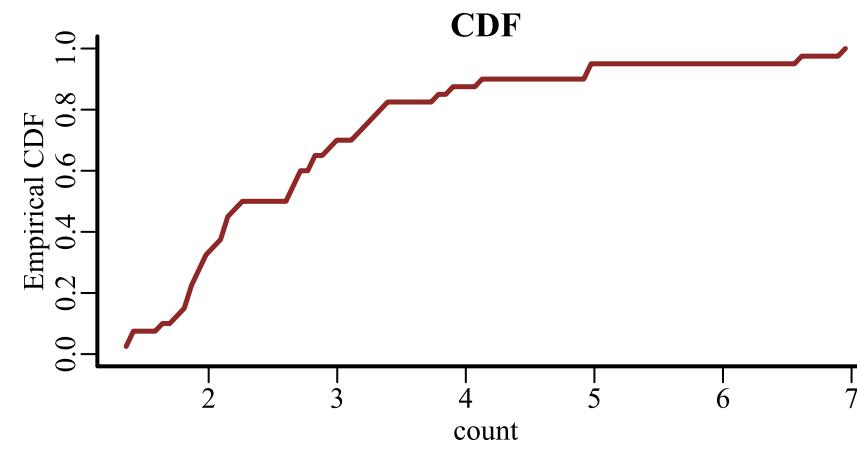
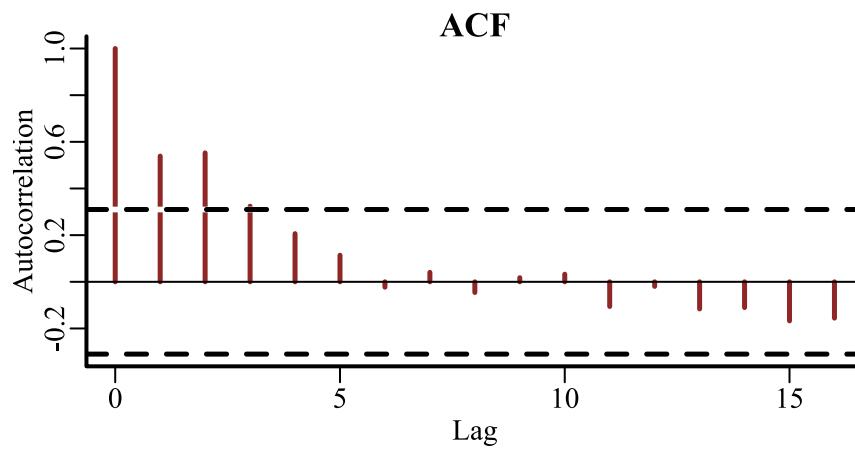
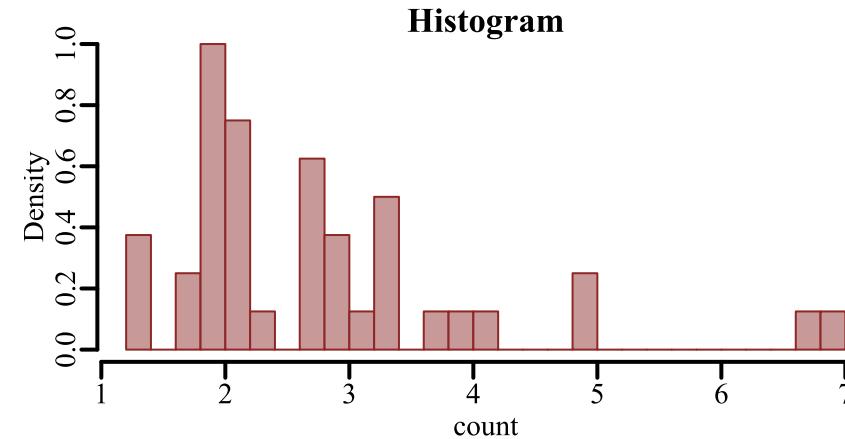
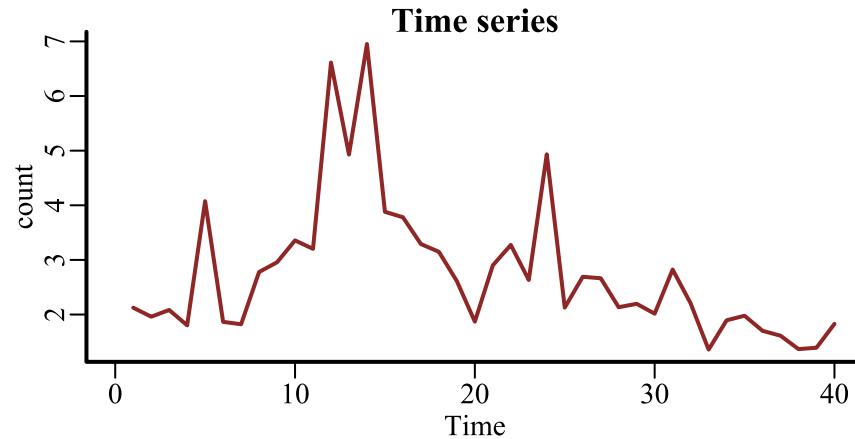
**But most real-world ecological observations, including time series, *are not Gaussian***



Properties of monthly CO<sub>2</sub> measurement time series at the South Pole



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

$$Y_t \sim \text{Poisson}(\lambda_t)$$

## Properties

Discrete, integer-valued observations (including 0)

Lower bound (supports 0 to  $\infty$ )

mean = variance =  $\lambda_t$

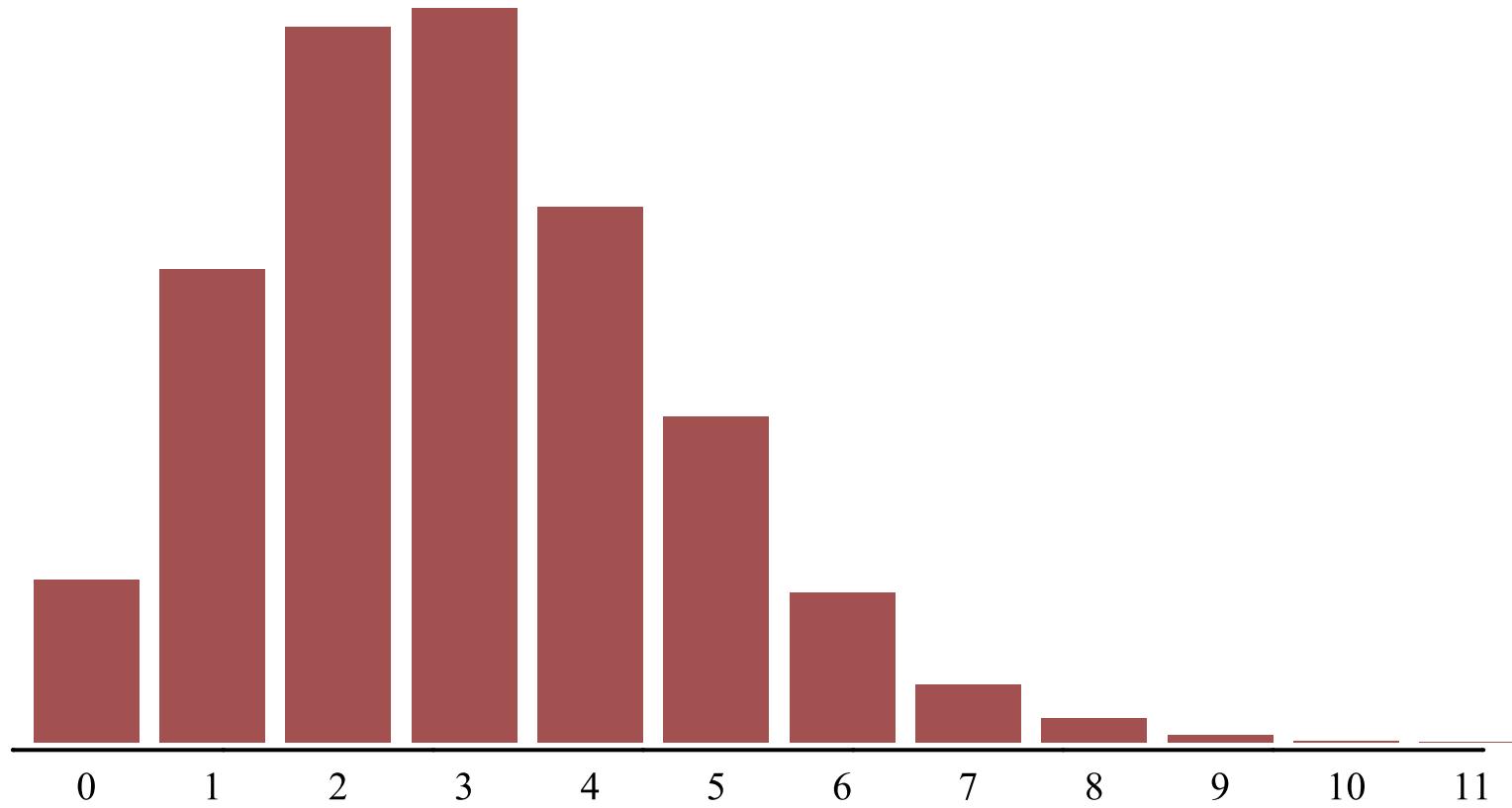
***Virtually no time series models support this distribution***

Most analysts use [log](#) or [Box-Cox](#) transformation

But see the [tscount](#) 

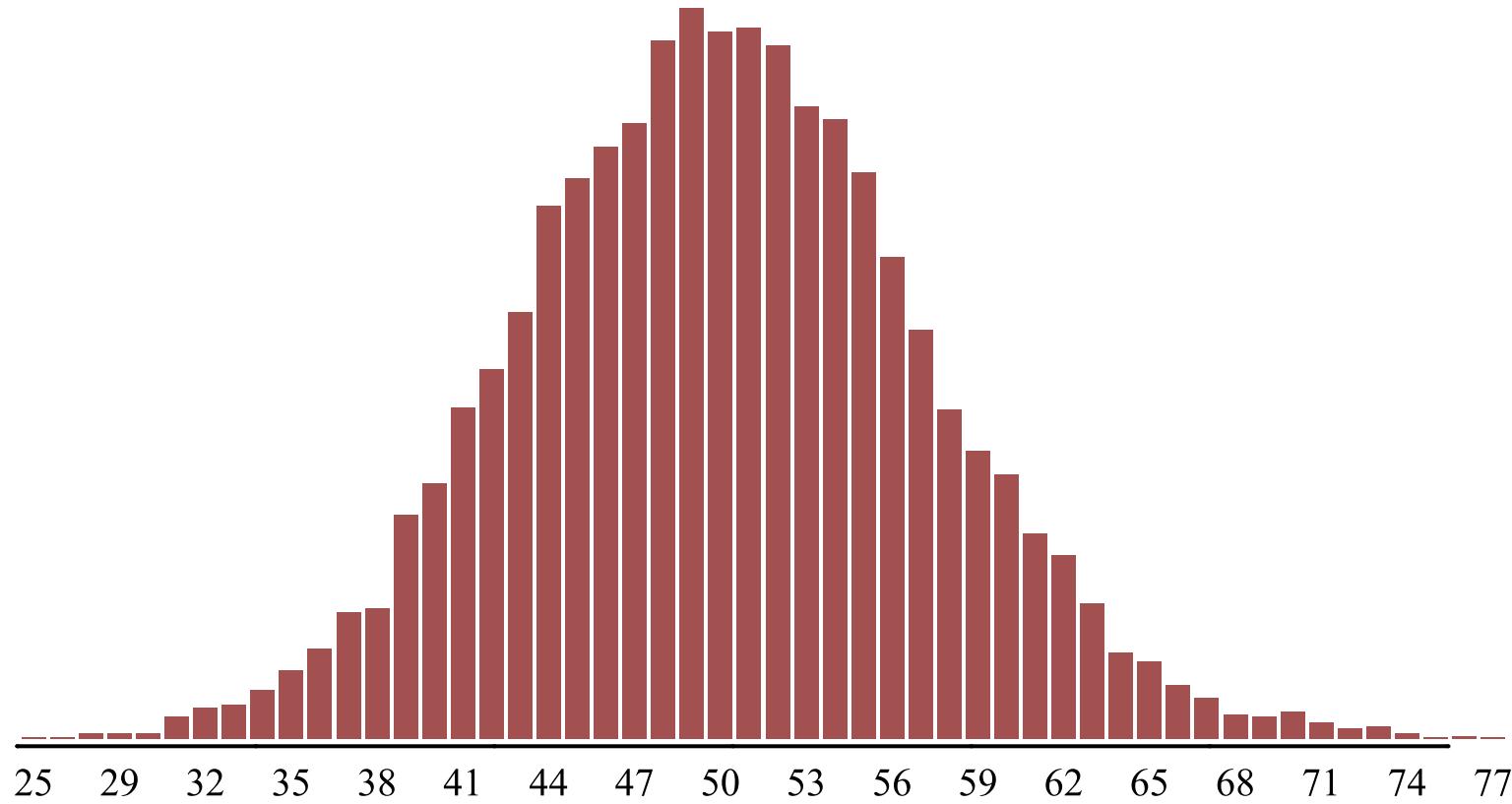
# Poisson

$$Y_t \sim \text{Poisson}(3)$$



# Poisson

$$Y_t \sim \text{Poisson}(50)$$



**How can we model non-Normal data using regression?**

# Generalized linear models

Linear regression can't be trusted to give sensible predictions for non-negative count data (or other types of bounded / discrete / non-Normal data)

We can do better by choosing distributions that obey the constraints on our outcome variables

The idea is to **generalize** the linear regression by replacing parameters from other probability distributions with linear models

This requires a **link function** that transforms from the unbounded scale of the linear predictor to a scale that is appropriate for the parameters being modeled

# Modelling the mean

Most GLMs are used to model the conditional mean ( $\mu_t$ )

$$\mathbb{E}(\mathbf{Y}_t | \mathbf{X}_t) = \mu_t = g^{-1}(\alpha + \mathbf{X}_t \boldsymbol{\beta})$$

Where:

$\mathbb{E}_t$  is the *expected value* of  $\mathbf{Y}_t$  conditional on  $\mathbf{X}_t$

$g^{-1}$  is the *inverse* of the link function

$\alpha$  is an intercept coefficient

$\boldsymbol{\beta}$  is a vector of regression coefficients

# Poisson GLM

A Poisson GLM models the conditional mean with a *log* link

$$\mathbf{Y}_t \sim \text{Poisson}(\lambda_t)$$

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

Where:

$\mathbf{X}_t$  is the matrix of predictor values at time  $t$

$\alpha$  is an intercept coefficient

$\boldsymbol{\beta}$  is a vector of regression coefficients

$$\mathbb{E}(\mathbf{Y}_t | \mathbf{X}_t) = \exp(\alpha + \mathbf{X}_t \boldsymbol{\beta})$$

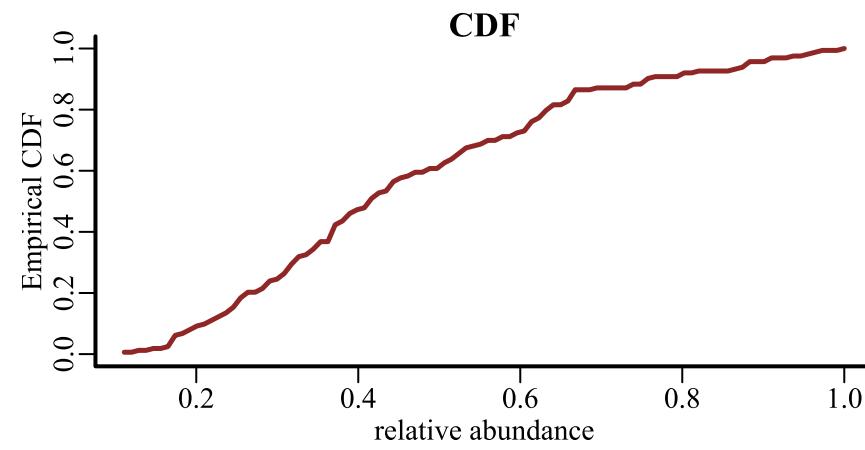
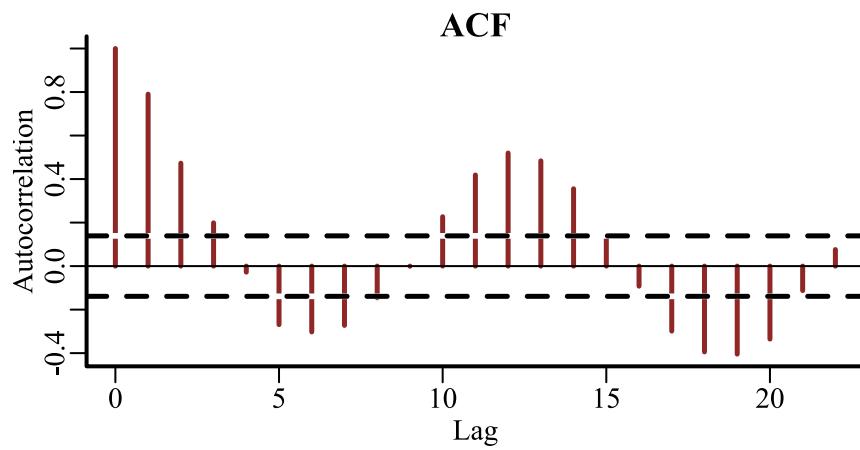
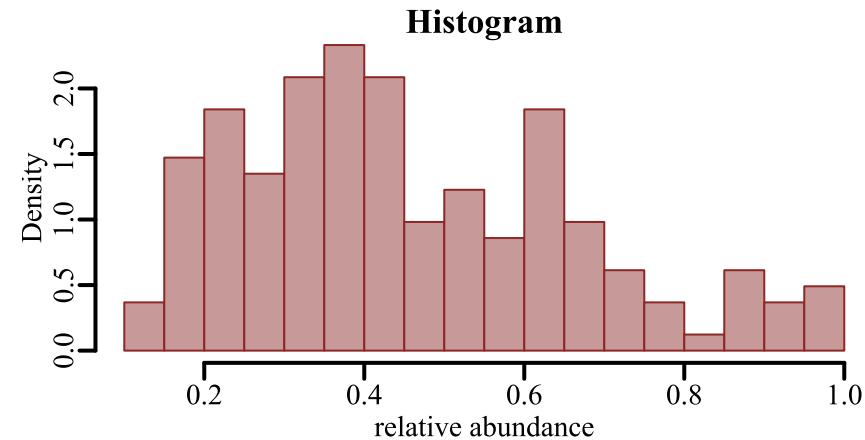
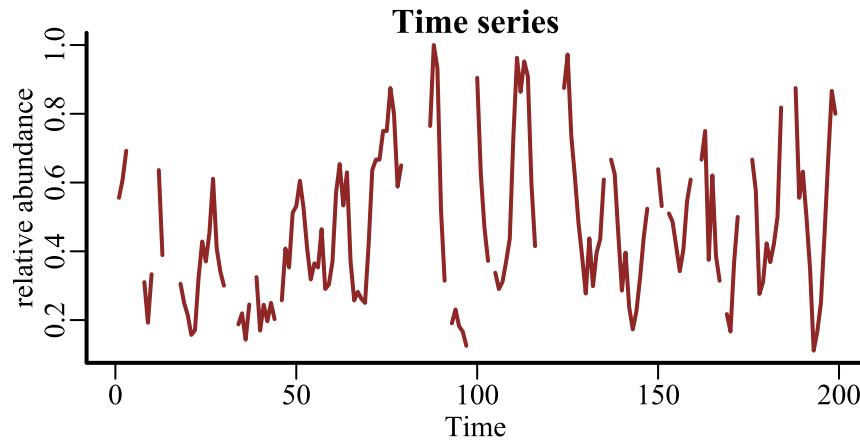
# Poisson GLM

A Poisson GLM models the conditional mean with a *log* link

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

The ***linear predictor component can be hugely flexible***, as we will see in later slides

**What if our data are proportional instead?**



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

# Beta

$$Y_t \sim \text{Beta}(\mu_t, \phi)$$

## Properties

Real-valued continuous observations (including any decimal)

Both lower and upper bounds (supports 0 to 1, noninclusive)

mean =  $\mu_t$

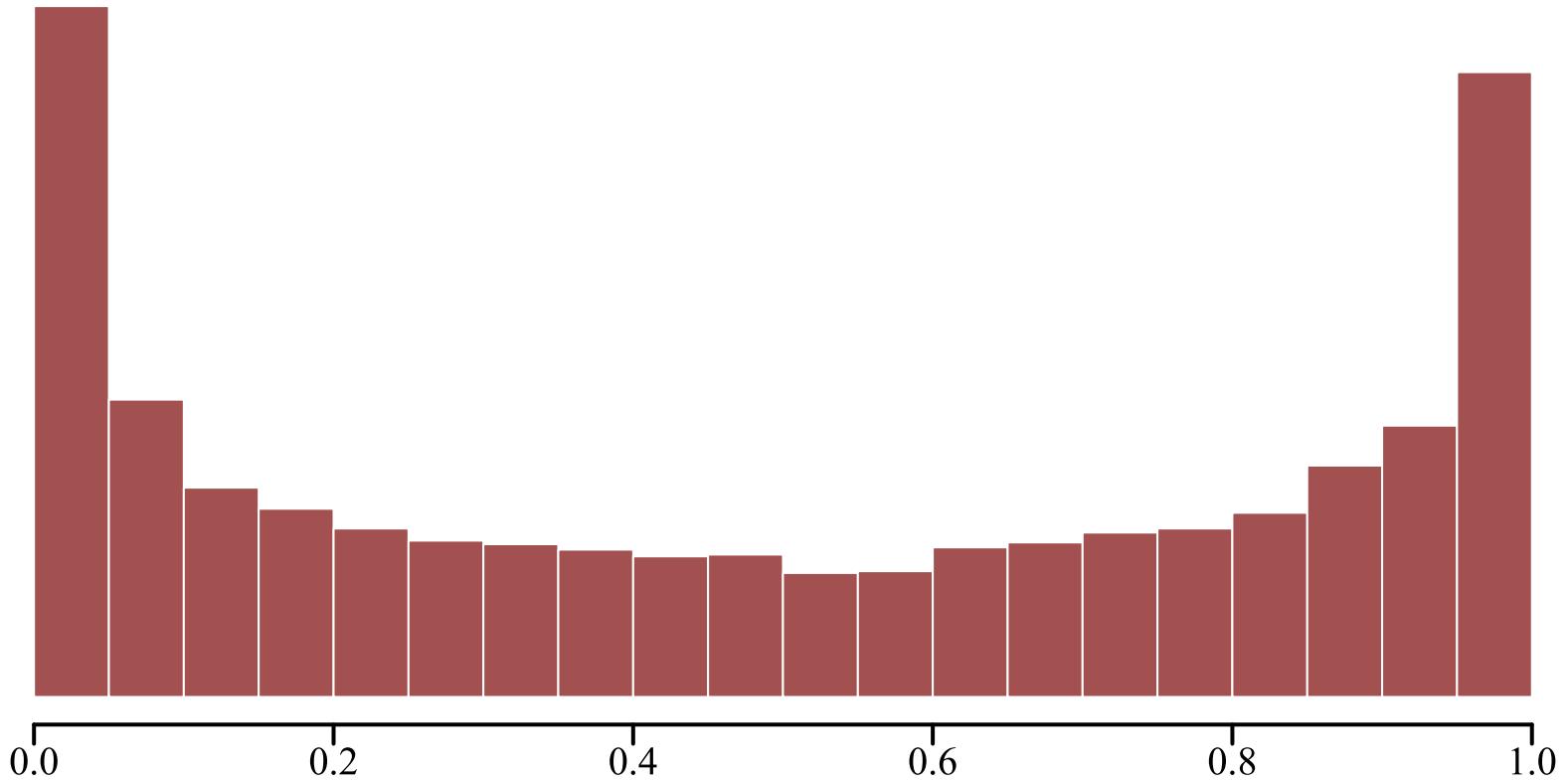
precision =  $\phi$  (higher  $\phi$  = a tighter spread about  $\mu_t$ )

***Virtually no time series models support this distribution***

Most analysts would use logit transformation

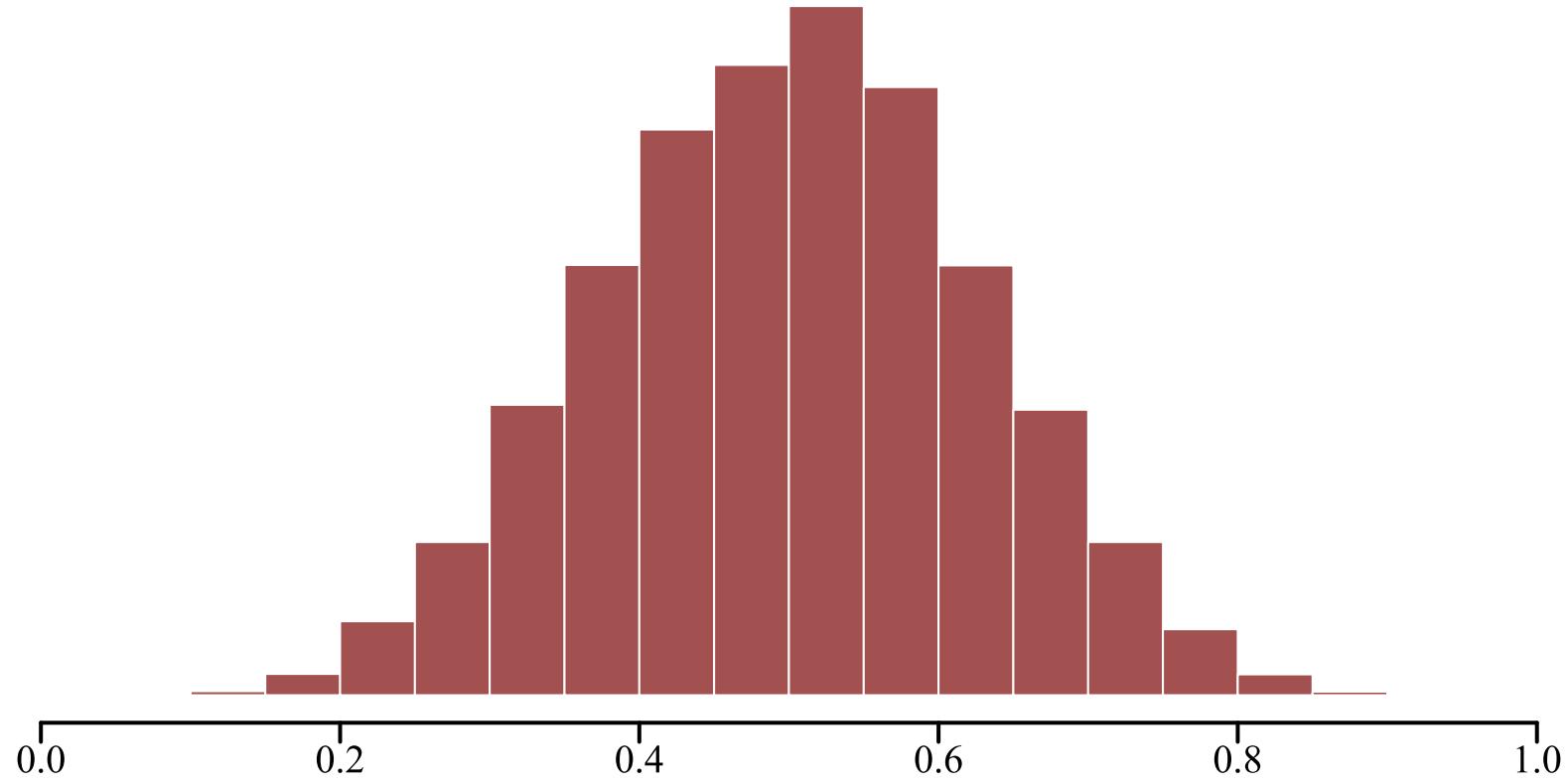
# Beta

$$Y_t \sim \text{Beta}(0.5, 1)$$



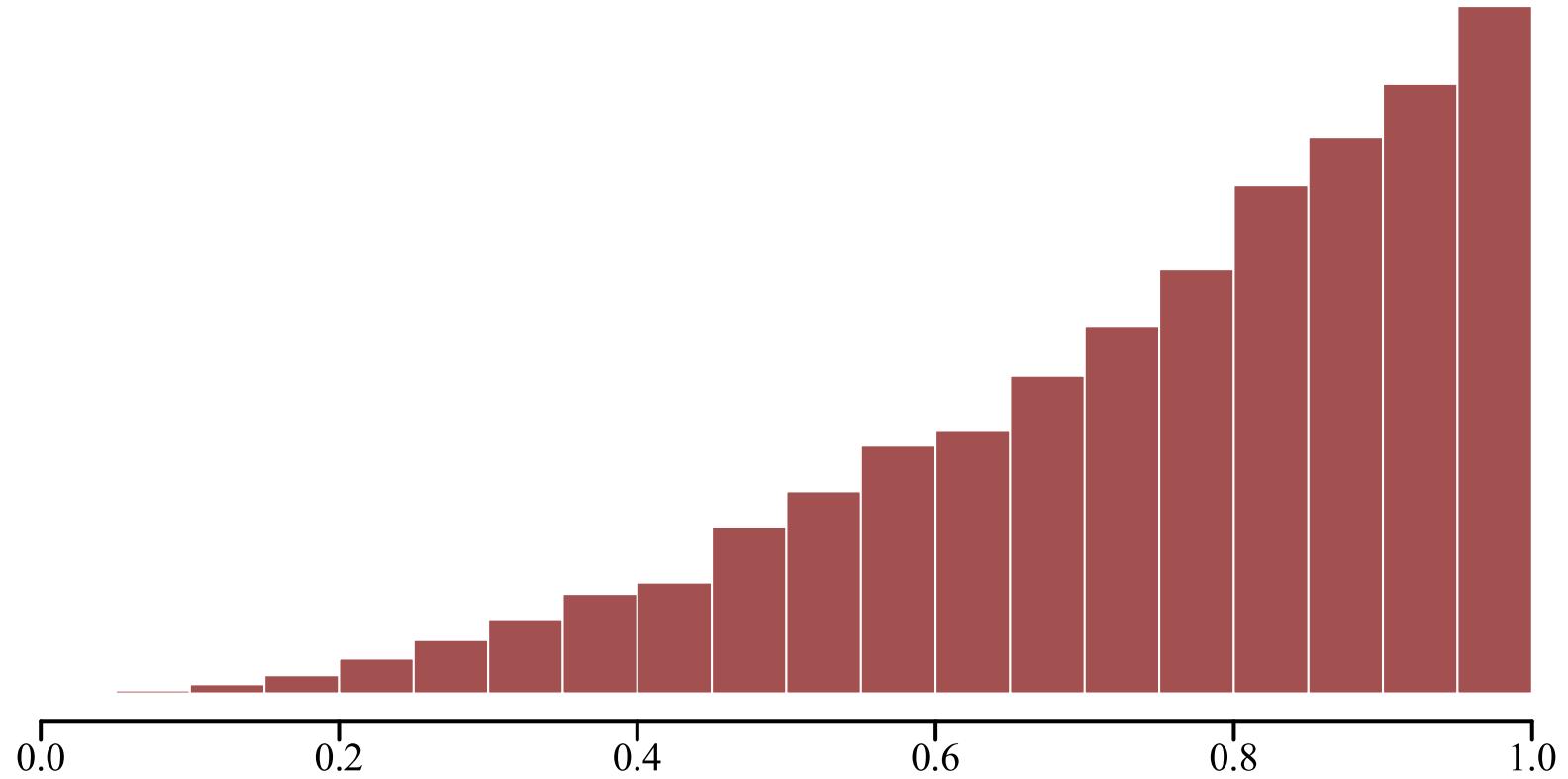
# Beta

$$Y_t \sim \text{Beta}(0.5, 15)$$



# Beta

$$Y_t \sim \text{Beta}(0.75, 4)$$



# Beta GLM

A Beta GLM models the conditional mean with a *logit* link

$$\mathbf{Y}_t \sim \text{Beta}(\mu_t, \phi)$$

$$\begin{aligned}\text{logit}(\mu_t) &= \mathbf{X}_t \boldsymbol{\beta} \\ &= \alpha + \beta_1 \mathbf{x}_{1t} + \beta_2 \mathbf{x}_{2t} + \cdots + \beta_j \mathbf{x}_{jt}\end{aligned}$$

Where:

$\mathbf{X}_t$  is the matrix of predictor values at time  $t$

$\alpha$  is an intercept coefficient

$\boldsymbol{\beta}$  is a vector of regression coefficients

$$\mathbb{E}(\mathbf{Y}_t | \mathbf{X}_t) = \text{logit}^{-1}(\alpha + \mathbf{X}_t \boldsymbol{\beta})$$

# Some other relevant distributions

Many other useful GLM probability distributions exist. Some of these include:

**Negative Binomial** – overdispersed integers in  $(0, 1, 2, \dots)$

**Bernoulli** – presence-absence data in  $\{0, 1\}$

**Student's T** – heavy-tailed (skewed) real values in  $(-\infty, \infty)$

**Lognormal** – heavy-tailed (right skewed) real values in  $(0, \infty)$

**Gamma** – lighter-tailed (less skewed) real values in  $(0, \infty)$

**Multinomial** – integers representing  $K$  unordered categories in  $(0, 1, \dots, K)$

**Ordinal** – integers representing  $K$  ordered categories in  $(0, 1, \dots, K)$

**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**

# Remember these?

Temporal autocorrelation

Lagged effects

***Non-Gaussian data and missing observations***

***Measurement error***

***Time-varying effects***

***Nonlinearities***

***Multi-series clustering***

# Remember these?

Temporal autocorrelation

Lagged effects

Non-Gaussian data and missing observations

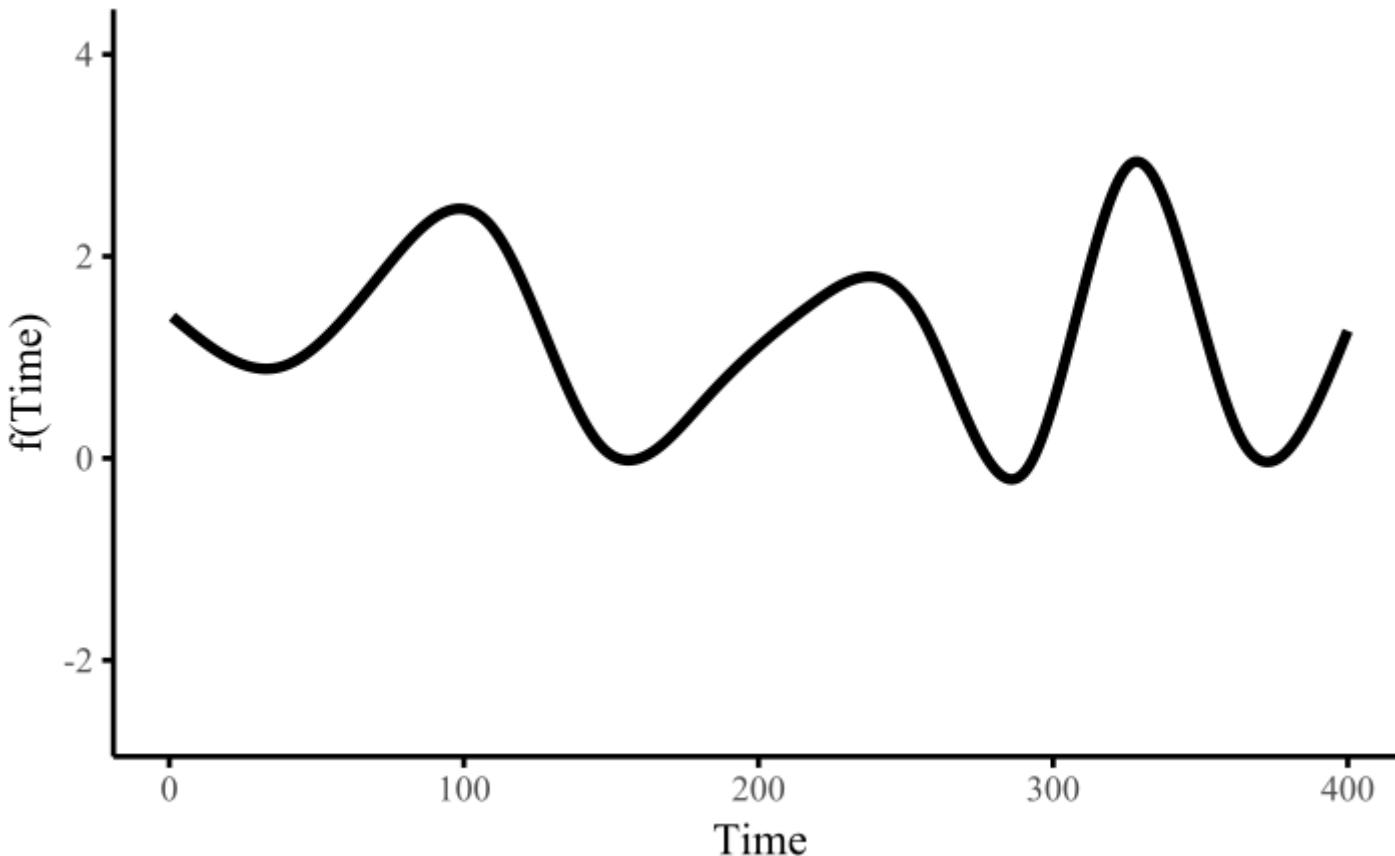
Measurement error

Time-varying effects

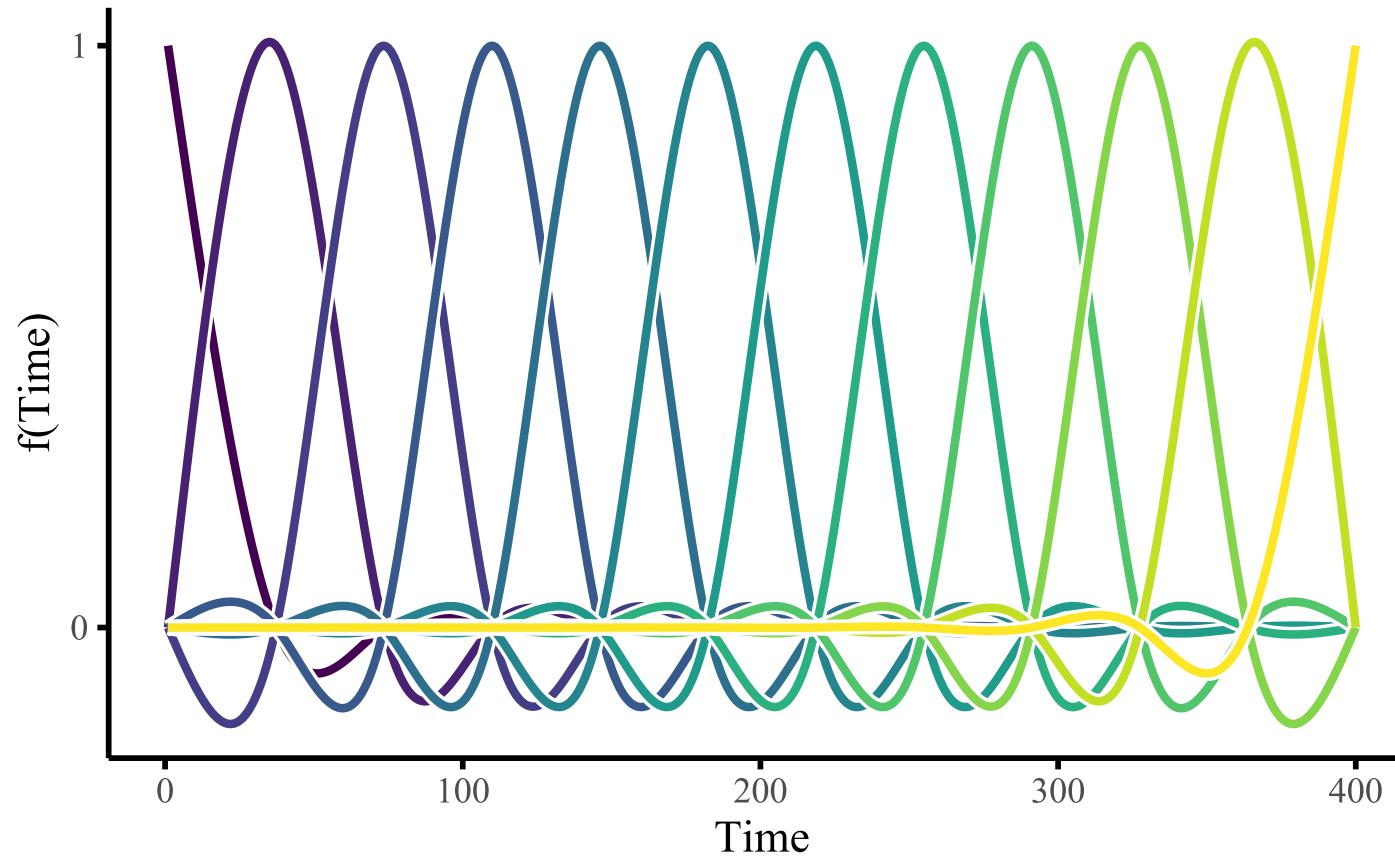
***Nonlinearities***

Multi-series clustering

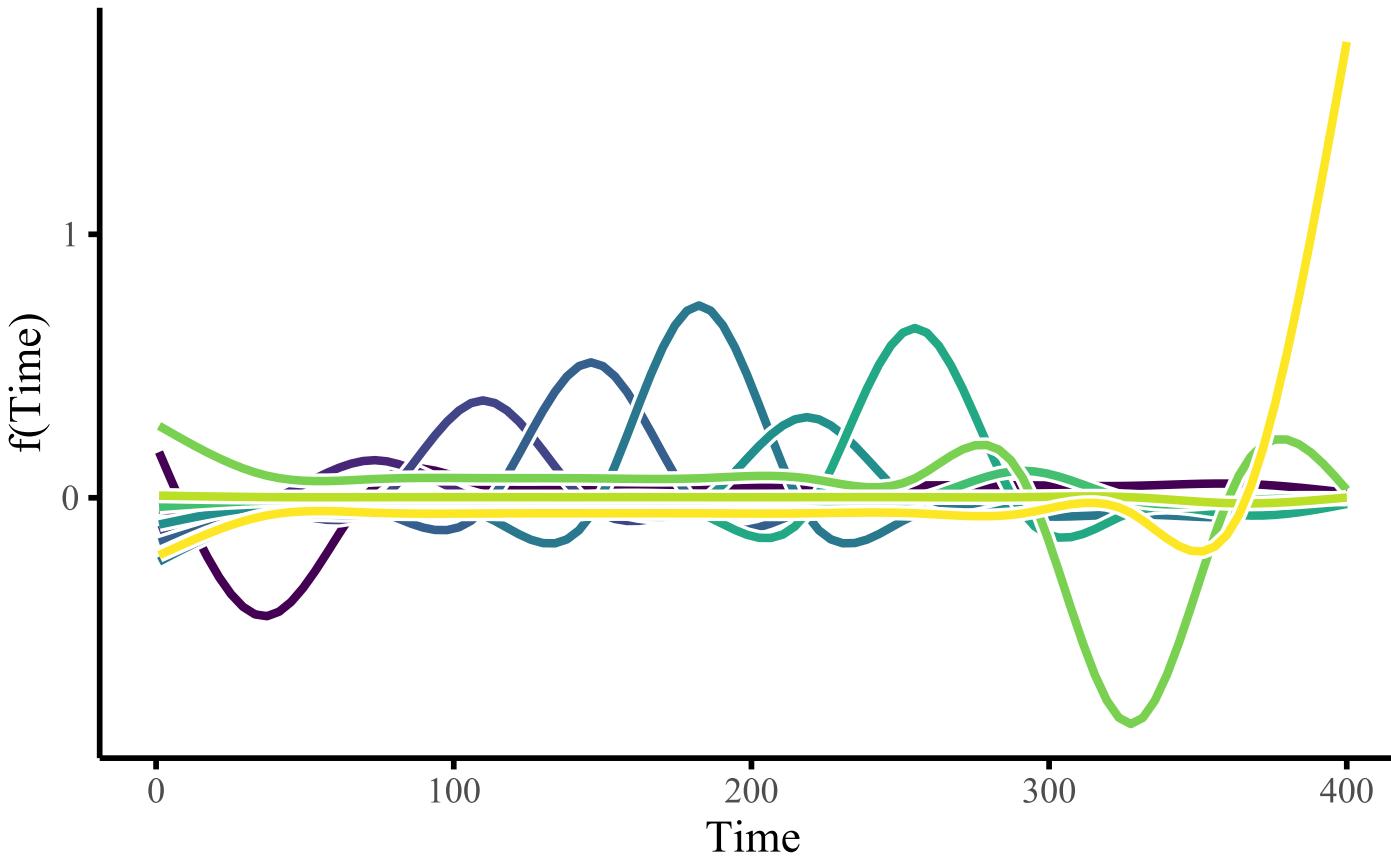
# GAMs use splines ...



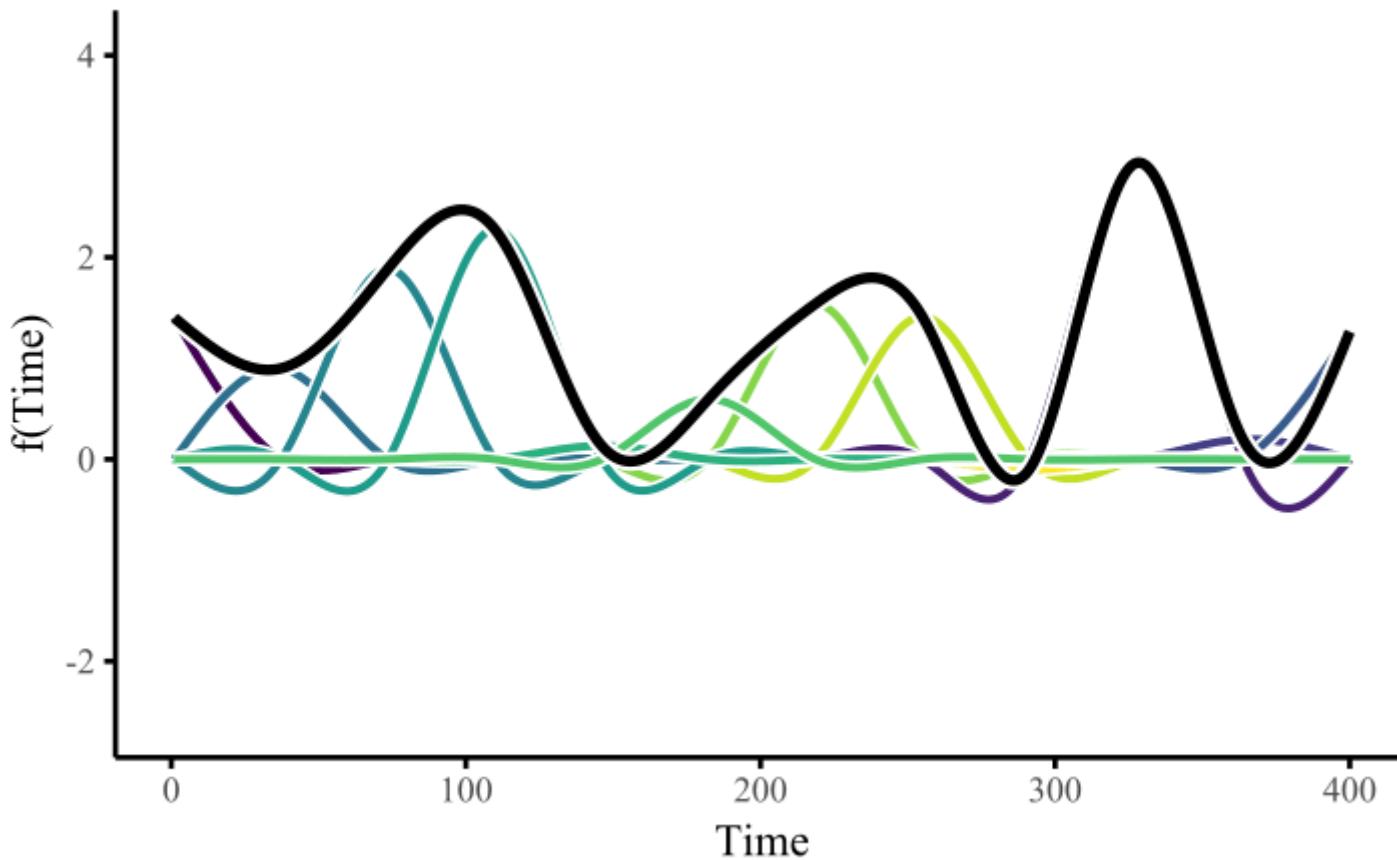
# ... made of basis functions



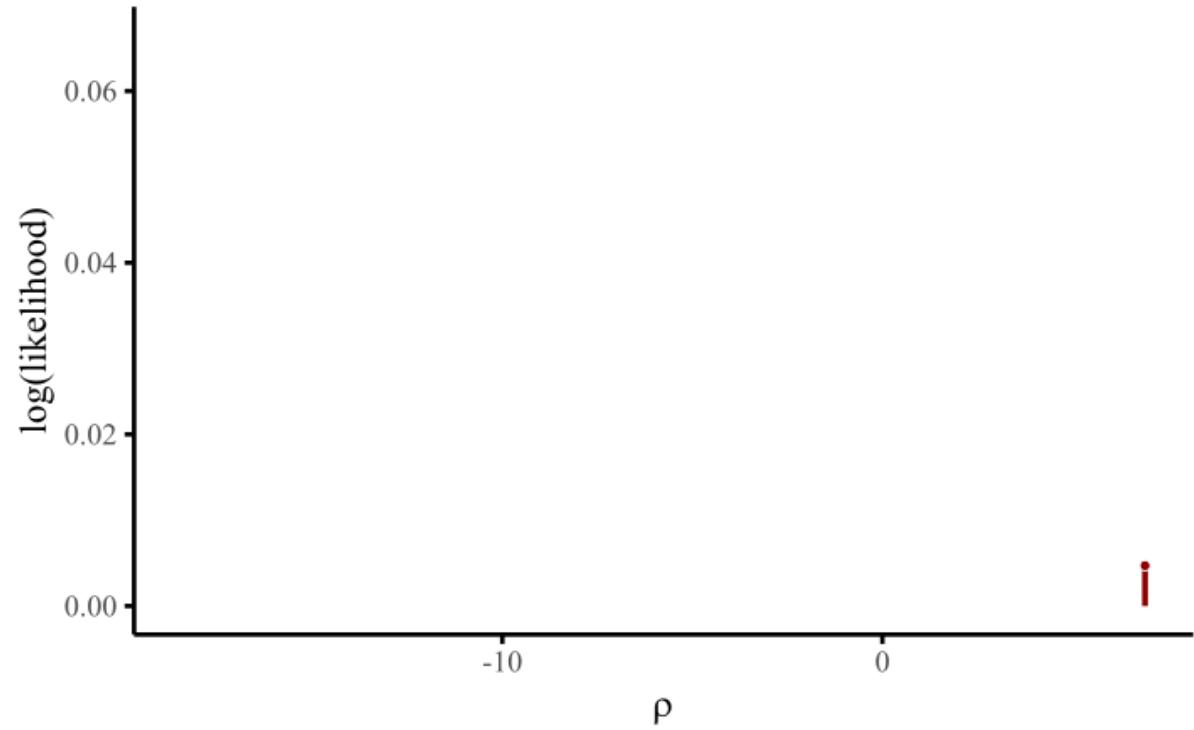
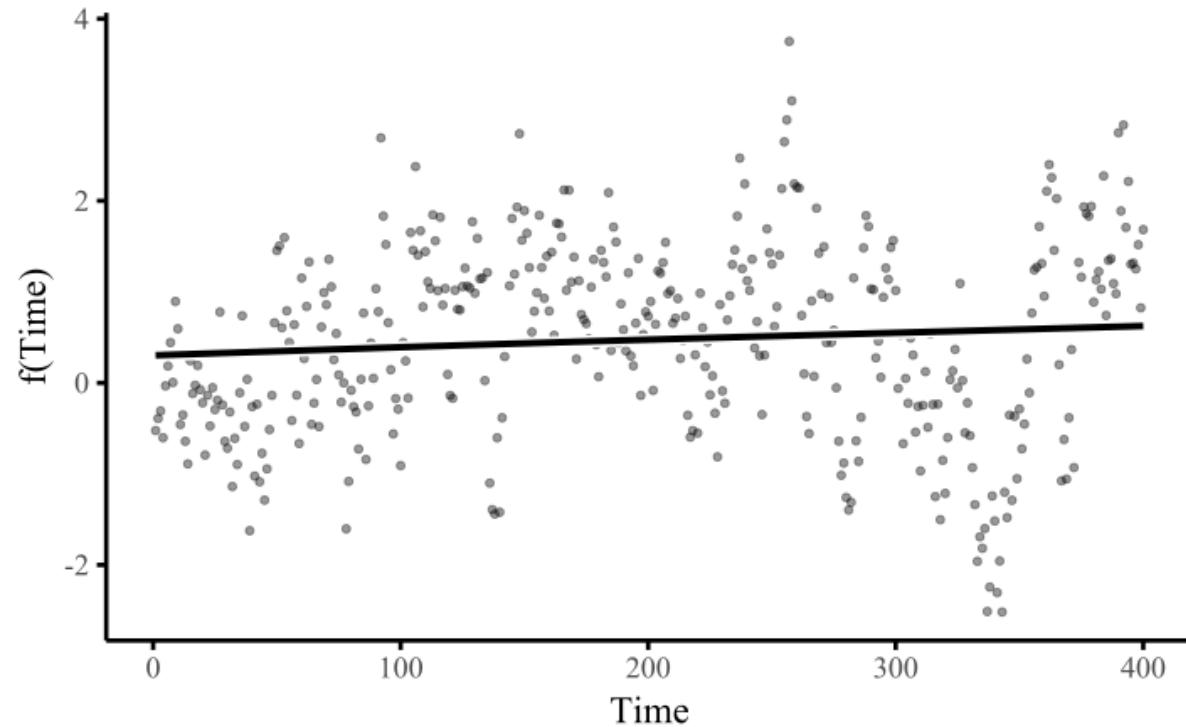
# Weighting basis functions ...



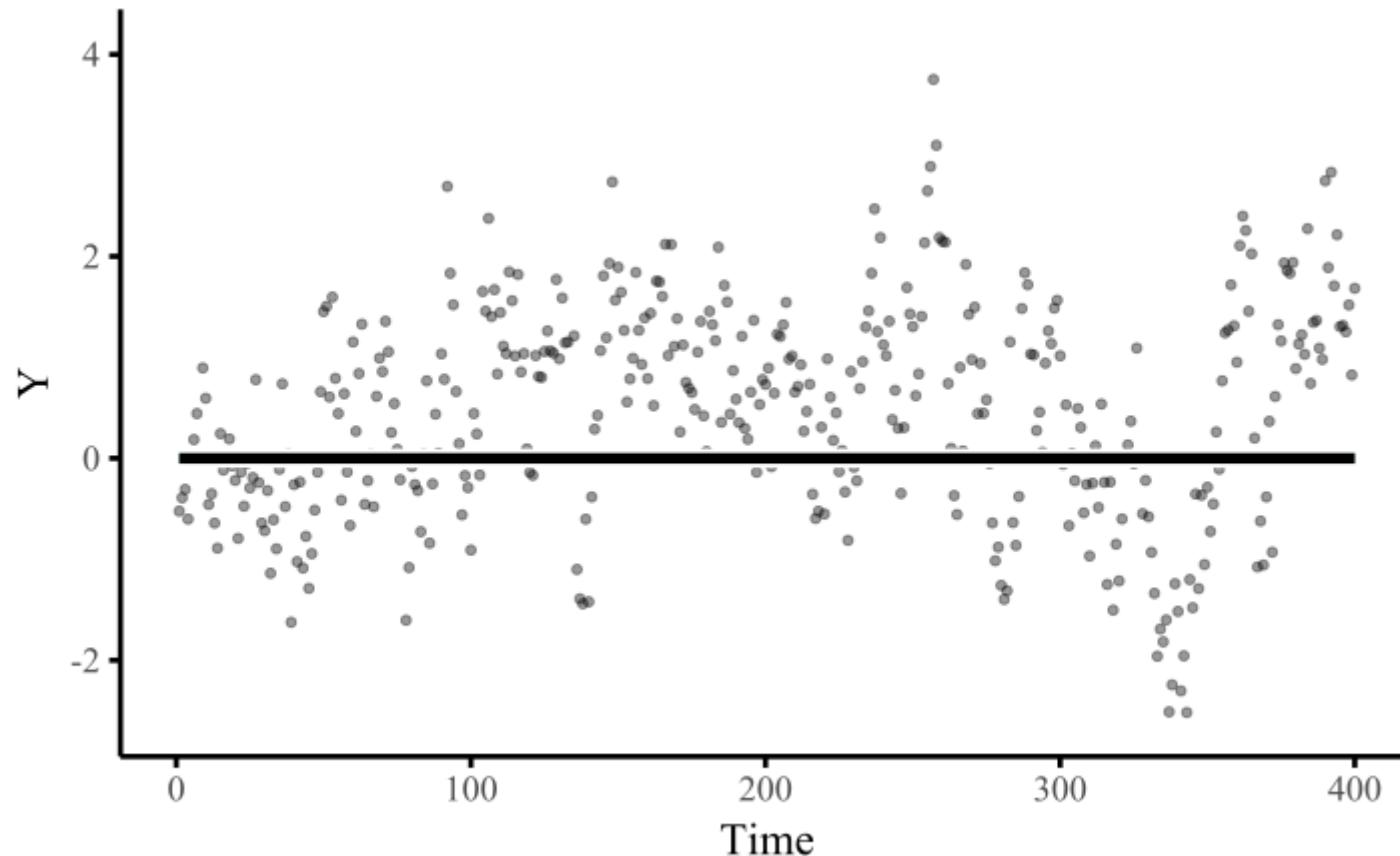
... gives a spline ( $f(x)$ )



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

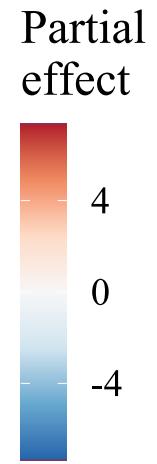
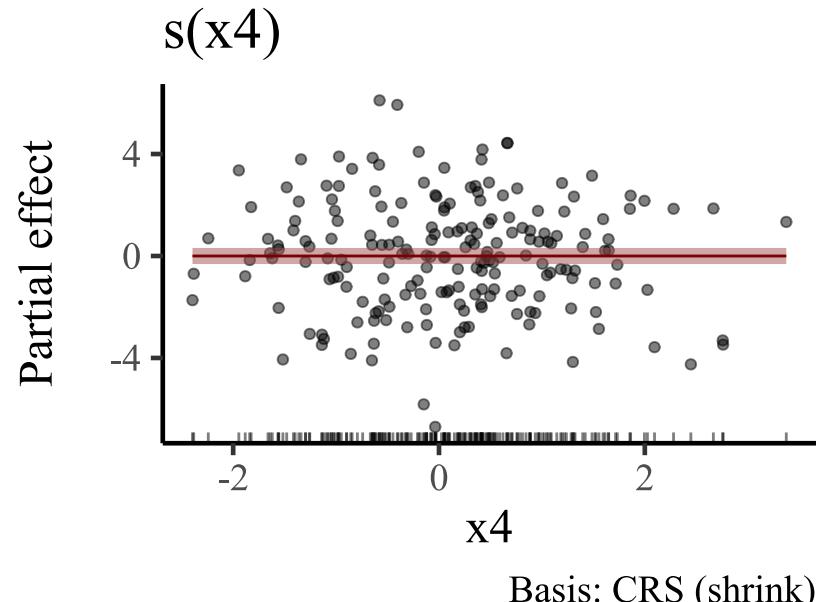
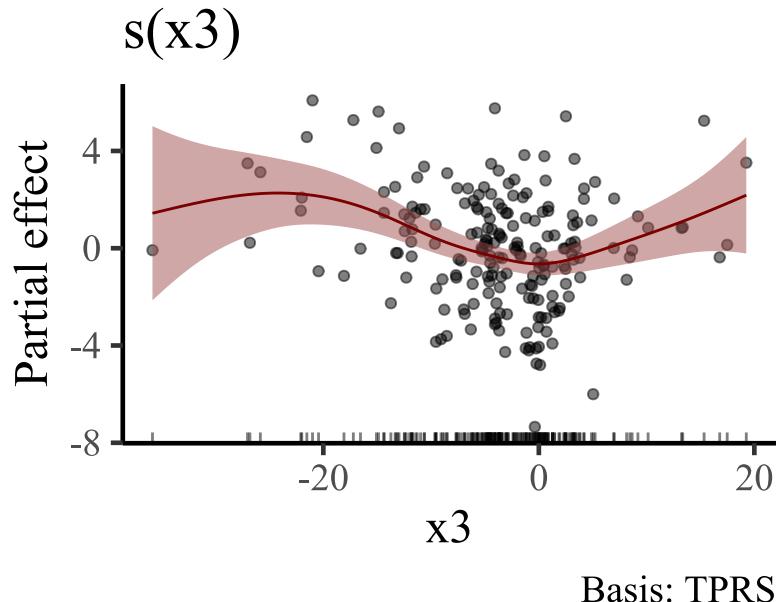
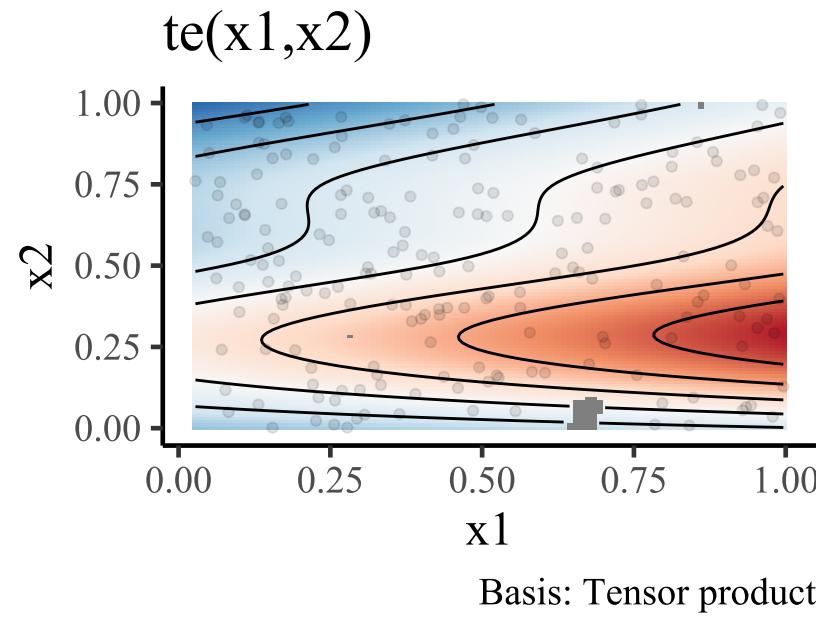
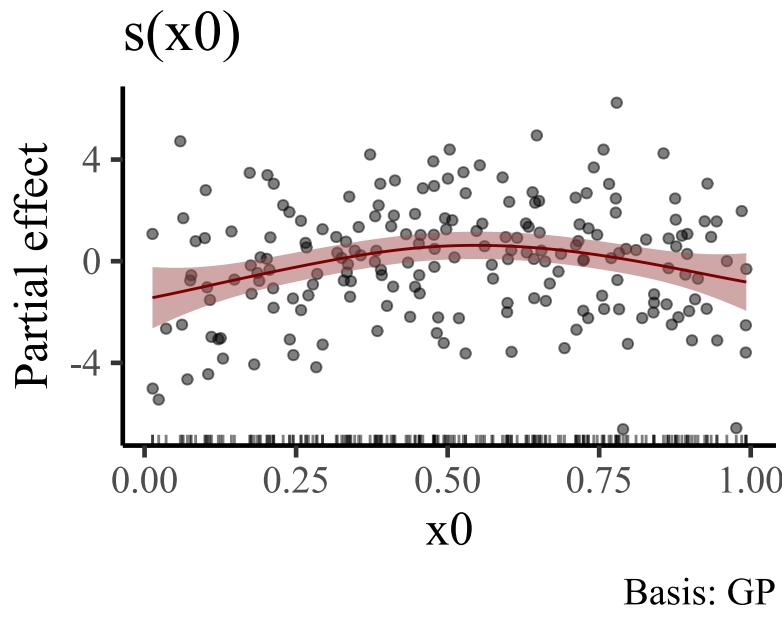


# Penalize $f''(x)$ to learn weights



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*



# GAMs easy to fit in

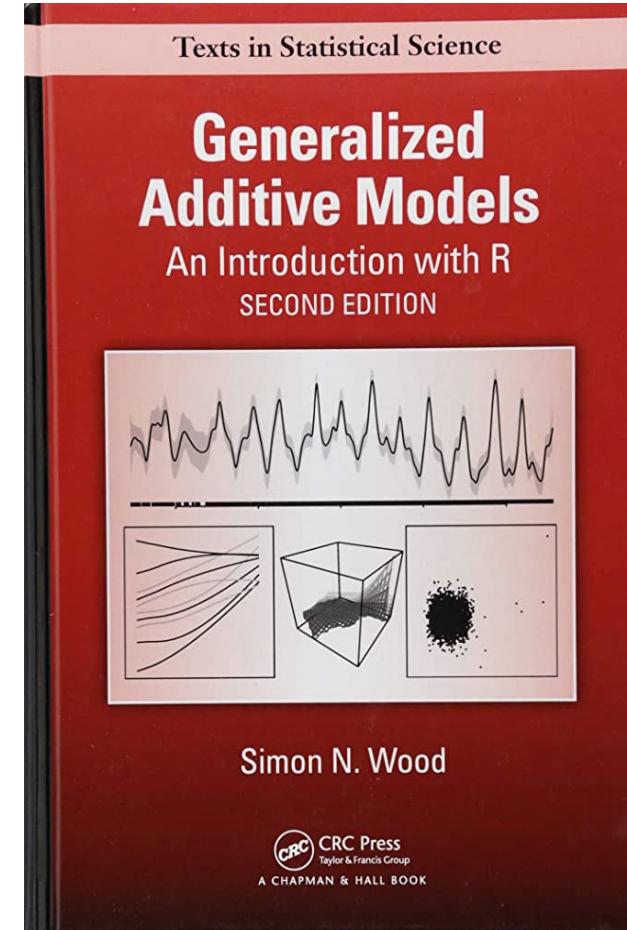
$$\mathbb{E}(Y_t | X_t) = g^{-1}(\alpha + \sum_{j=1}^J f(x_{jt}))$$

Where:

$g^{-1}$  is the *inverse* of the link function

$\alpha$  is the intercept

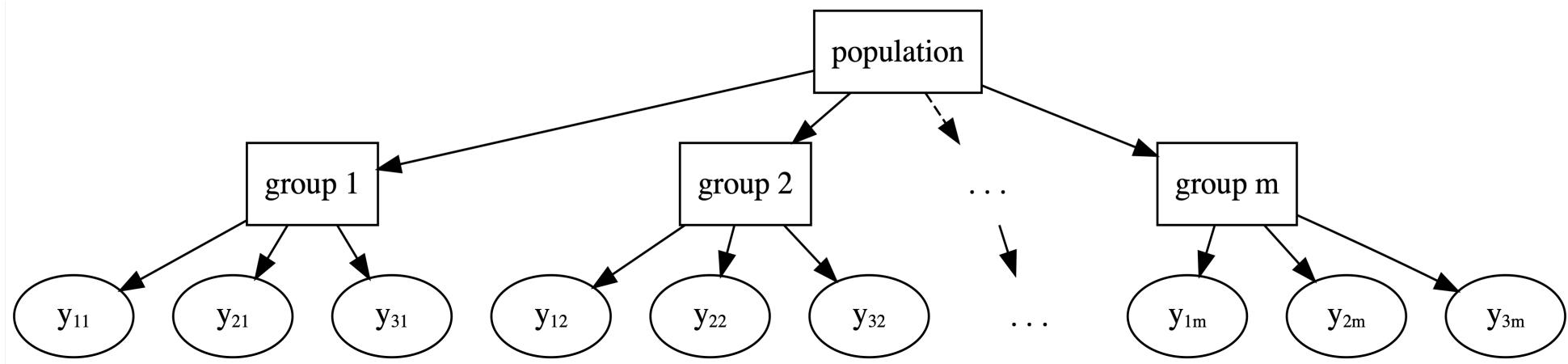
$f(x)$  are potentially nonlinear functions of the  $J$  predictors



**But how can GAMs and GLMs be useful for modelling ecological time series?**

# Temporal random effects

# Random effects are *hierarchical*



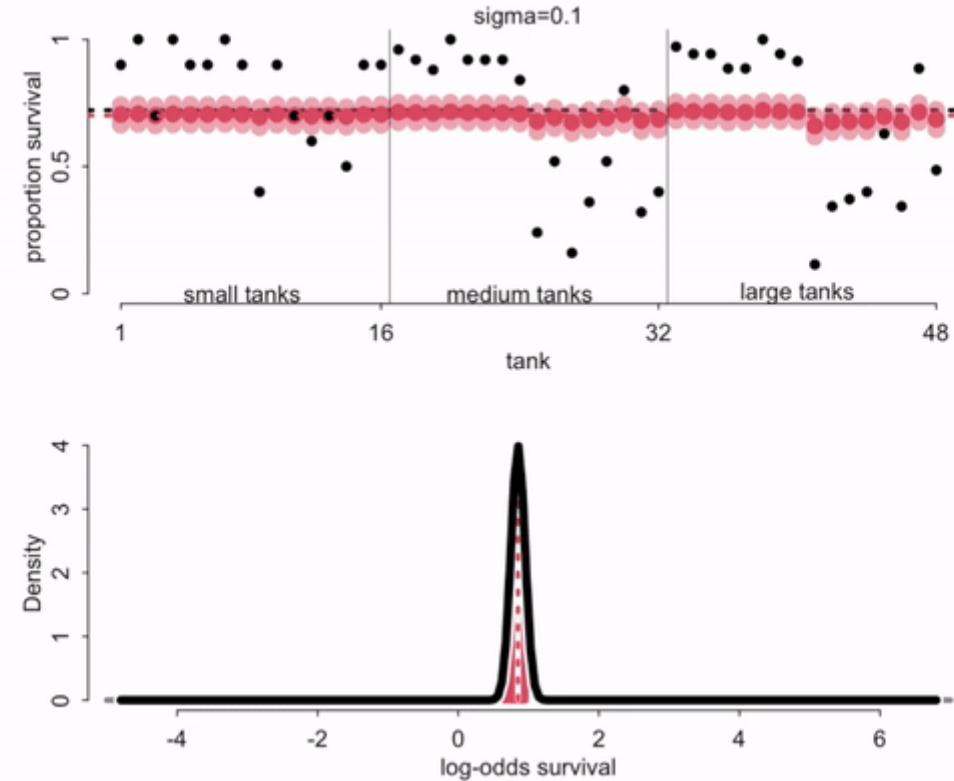
Johnson et al 2021

Hierarchical models *learn from all groups at once* to inform group-level estimates

This induces *regularization*, where noisy estimates are pulled towards the overall mean

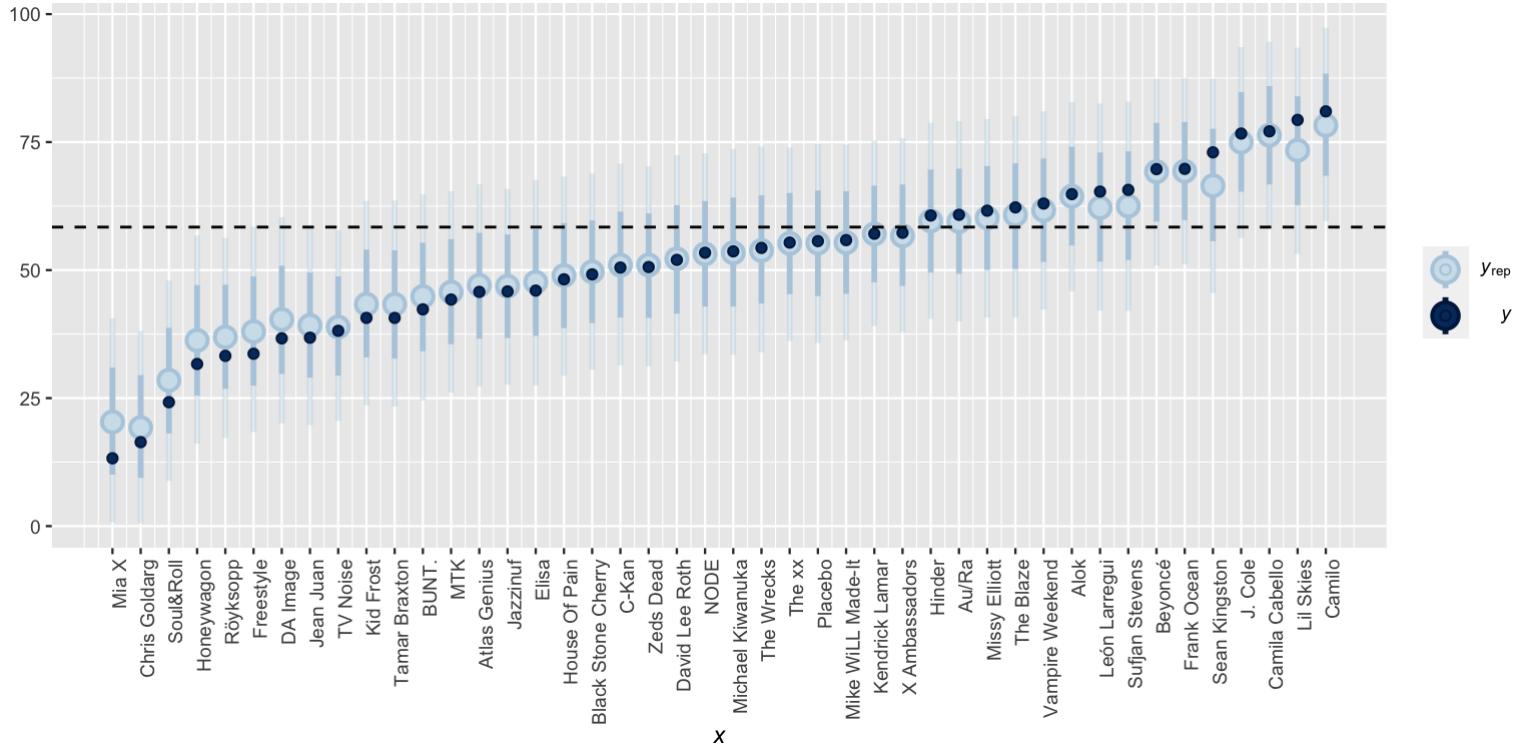
The regularization is known as partial pooling

# Partial pooling in action



McElreath 2023

# Noisy estimates pulled to the mean



Johnson et al 2021

# How can they be modelled?

$$Y_t \sim \text{Poisson}(\lambda_t)$$

$$\log(\lambda_t) = \beta_{year[year_t]}$$

$$\beta_{year} \sim \text{Normal}(\mu_{year}, \sigma_{year})$$

$$\mu_{year} \sim \text{Normal}(0, 1)$$

$$\sigma_{year} \sim \text{Exponential}(2)$$

Where we have multiple time points per year, and:

$\beta_{year}$  are yearly intercepts (*one effect per year*)

$\mu_{year}$  estimates *mean effect among all years*

$\sigma_{year}$  estimates *how much effects vary across years*

# 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  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  
)
```

Where `v` = response, `x`'s = covariates, and `series` = a grouping term

# 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  
)
```

# A random intercept effect

```
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  
)
```

# A random slope effect

```
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  
)
```

# A linear parametric effect

```
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  
)
```

# A one-dimensional smooth

```
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  
)
```

# A two-dimensional smooth

```
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 data (long format)

y	series	time
1	species_1	1
1	species_2	1
NA	species_3	1
NA	species_4	1
1	species_1	2
0	species_2	2
3	species_3	2
4	species_4	2

# Response (NAs allowed)

y	series	time
1	species_1	1
1	species_2	1
NA	species_3	1
NA	species_4	1
1	species_1	2
0	species_2	2
3	species_3	2
4	species_4	2

# Series indicator (as factor)

y	series	time
1	species_1	1
1	species_2	1
NA	species_3	1
NA	species_4	1
1	species_1	2
0	species_2	2
3	species_3	2
4	species_4	2

# Time indicator

y	series	time
1	species_1	1
1	species_2	1
NA	species_3	1
NA	species_4	1
1	species_1	2
0	species_2	2
3	species_3	2
4	species_4	2

# Any other predictors

y	series	time	x0	x1	x2	x3	x4
1	species_1	1	-0.38	A	0.20	1.18	-0.72
1	species_2	1	-0.71	A	-2.67	1.02	0.67
NA	species_3	1	0.05	B	-0.33	0.12	1.50
NA	species_4	1	0.77	B	0.65	0.86	-0.49
1	species_1	2	0.29	A	-0.25	1.18	-0.82
0	species_2	2	0.34	A	-0.15	2.12	0.20
3	species_3	2	-0.38	B	-0.81	1.33	-1.15
4	species_4	2	1.32	B	0.22	-0.72	1.36

# Examples



# The data structure

```
dplyr::glimpse(model_data)
```

```
## #> #> #> Rows: 199  
## #> #> Columns: 6  
## #> $ series <fct> PP,  
## #> PP...  
## #> $ year <fct> 2004, 2004, 2004, 2004, 2004, 2004, 2004, 2004, 2004,  
## #> 20...  
## #> $ time <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,  
## #> 18,...  
## #> $ count <int> 0, 1, 2, NA, 10, NA, NA, 16, 18, 12, NA, 3, 2, NA, NA, 13,  
## #> NA,...  
## #> $ mintemp <dbl> -9.710, -5.924, -0.220, 1.931, 6.568, 11.590, 14.370,  
## #> 16.520, ...  
## #> $ ndvi <dbl> 1.4658889, 1.5585069, 1.3378172, 1.6589129, 1.8536561,
```

# The observations

---

Code    Plot

---

```
# use mvgam's plot utility to view properties of the observations
plot_mvgam_series(data = model_data, y = 'count')
```

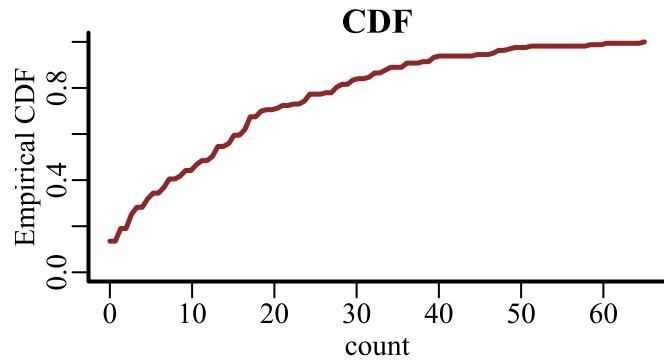
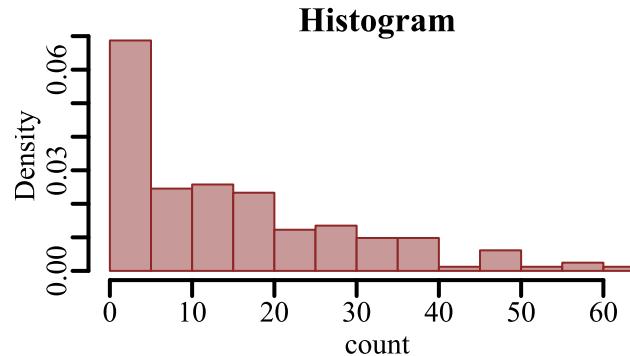
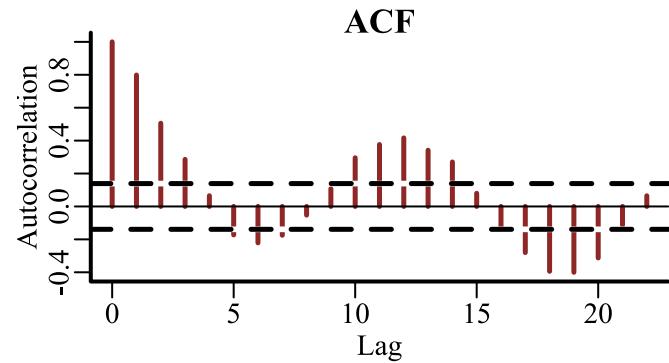
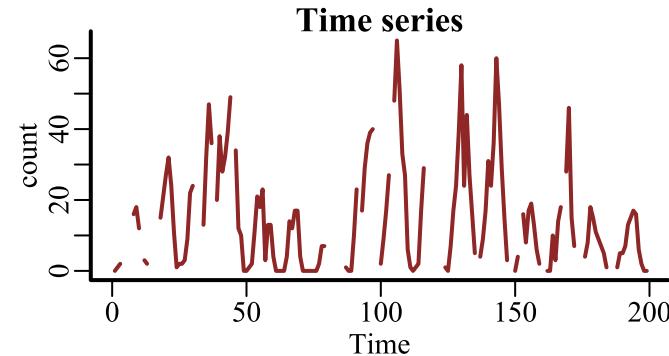
# The observations

---

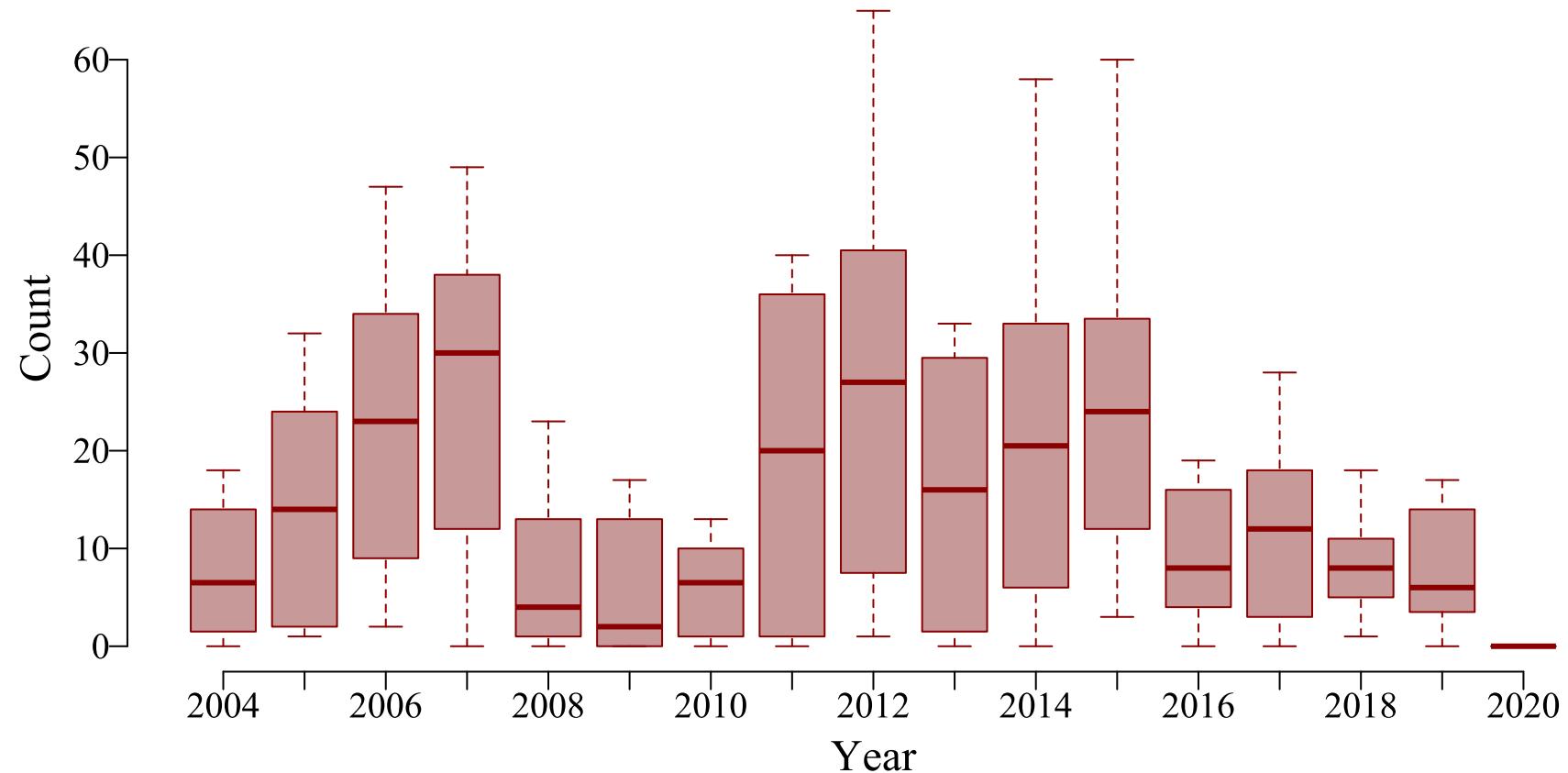
Code

Plot

---



# Yearly heterogeneity



# Yearly random intercepts

```
year_random ← mvgam(count ~  
                      s(year, bs = 're') - 1,  
                      family = poisson(),  
                      data = model_data,  
                      trend_model = 'None',  
                      burnin = 500,  
                      samples = 500,  
                      chains = 4)
```

Random effect basis in `mgcv` language

Global intercept suppressed

# Estimated yearly intercepts

---

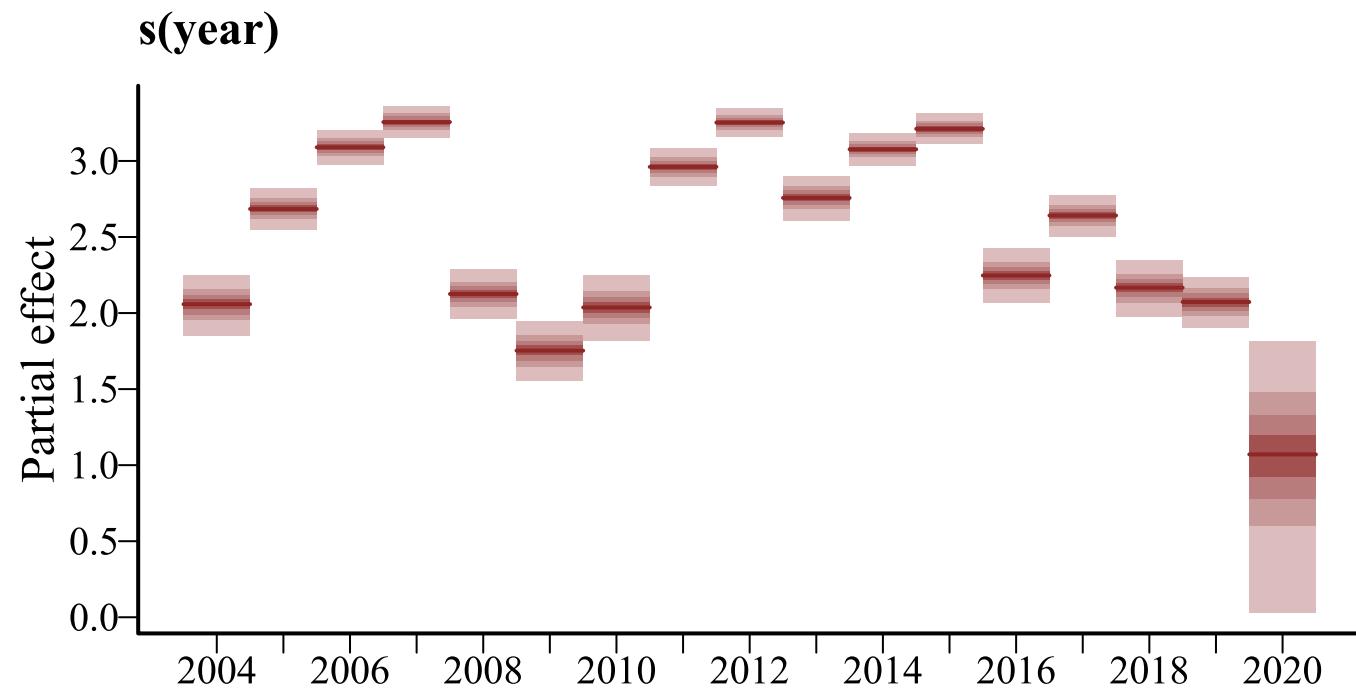
[Code](#)    [Plot](#)

---

```
# plot the random effect posterior estimates
plot(year_random, type = 're')
```

# Estimated yearly intercepts

Code Plot



# Population parameters

---

Code	Means	SDs
------	-------	-----

---

```
# extract population estimates
pop_params ← as.data.frame(year_random,
                           variable = c('mean(year)',
                                         'sd(year)'))

# plot as histograms
hist(pop_params$`mean(year)` , main = expression(mu[year]))

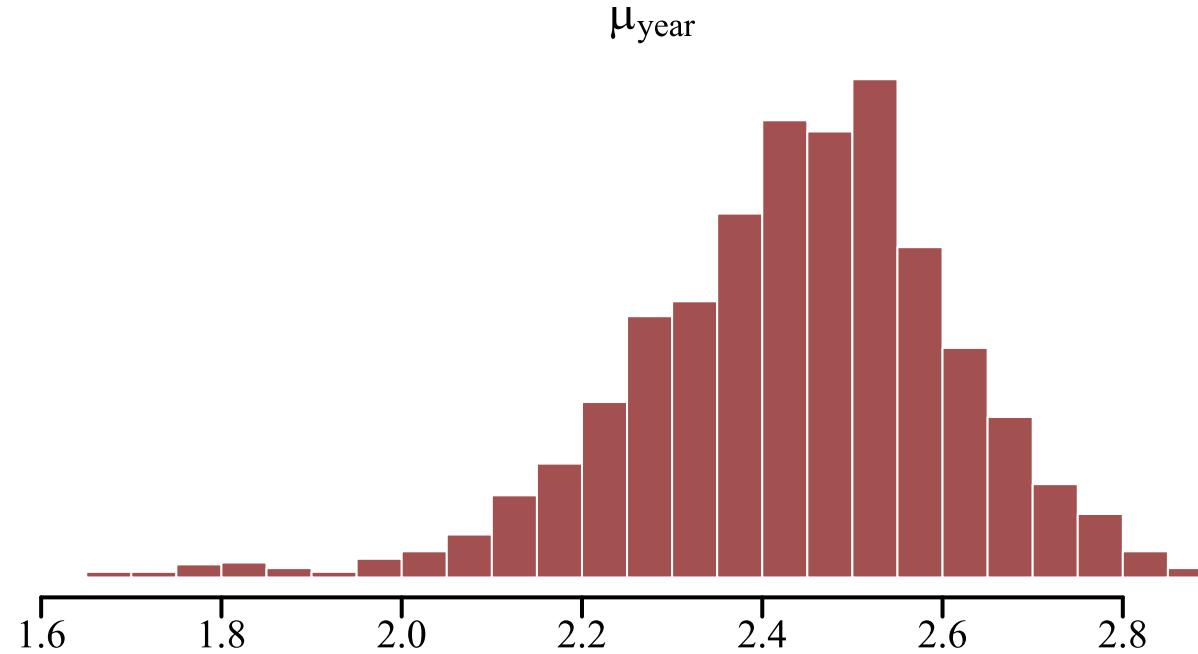
hist(pop_params$`sd(year)` , main = expression(sigma[year]))
```

# Population parameters

---

Code	Means	SDs
------	-------	-----

---



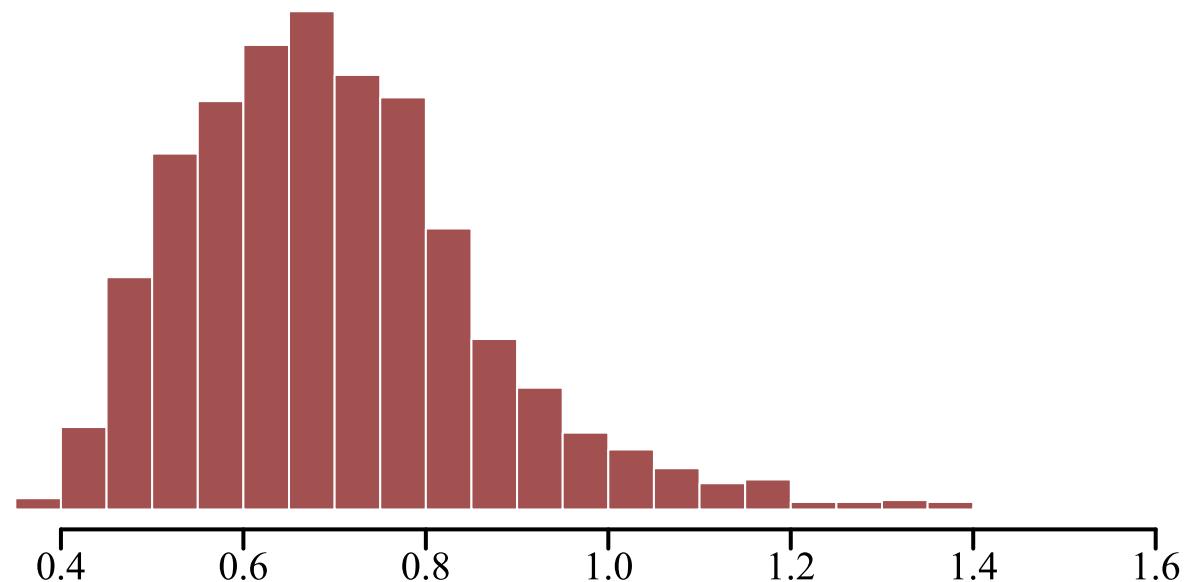
# Population parameters

---

Code	Means	SDs
------	-------	-----

---

$\sigma_{\text{year}}$



# Or using bayesplot

---

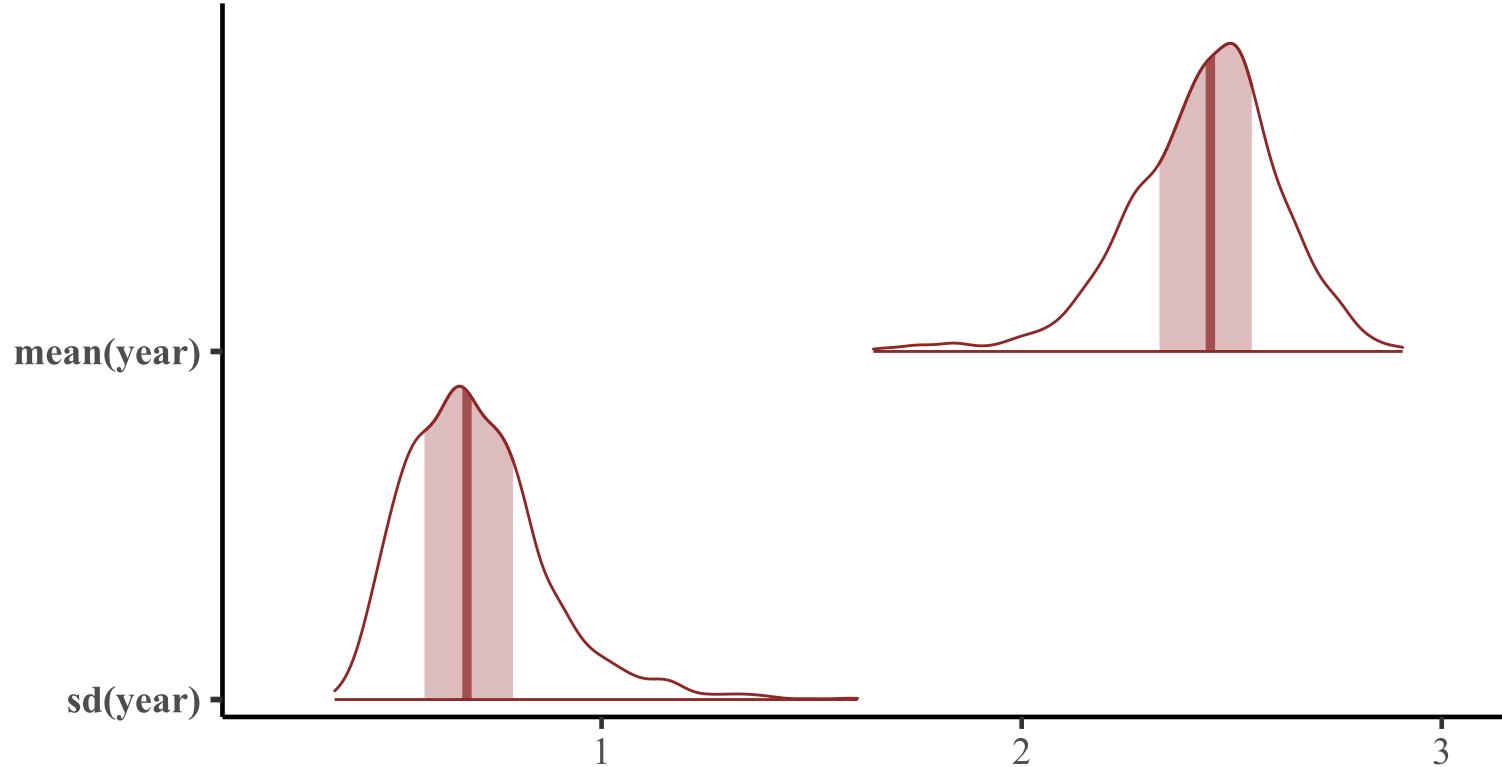
Code    Plot

---

```
# use bayesplot utilities to plot parameter estimates
mcmc_plot(year_random, type = 'areas',
           variable = c('mean(year)', 'sd(year)'))
```

# Or using bayesplot

Code      Plot



# Conditional predictions

---

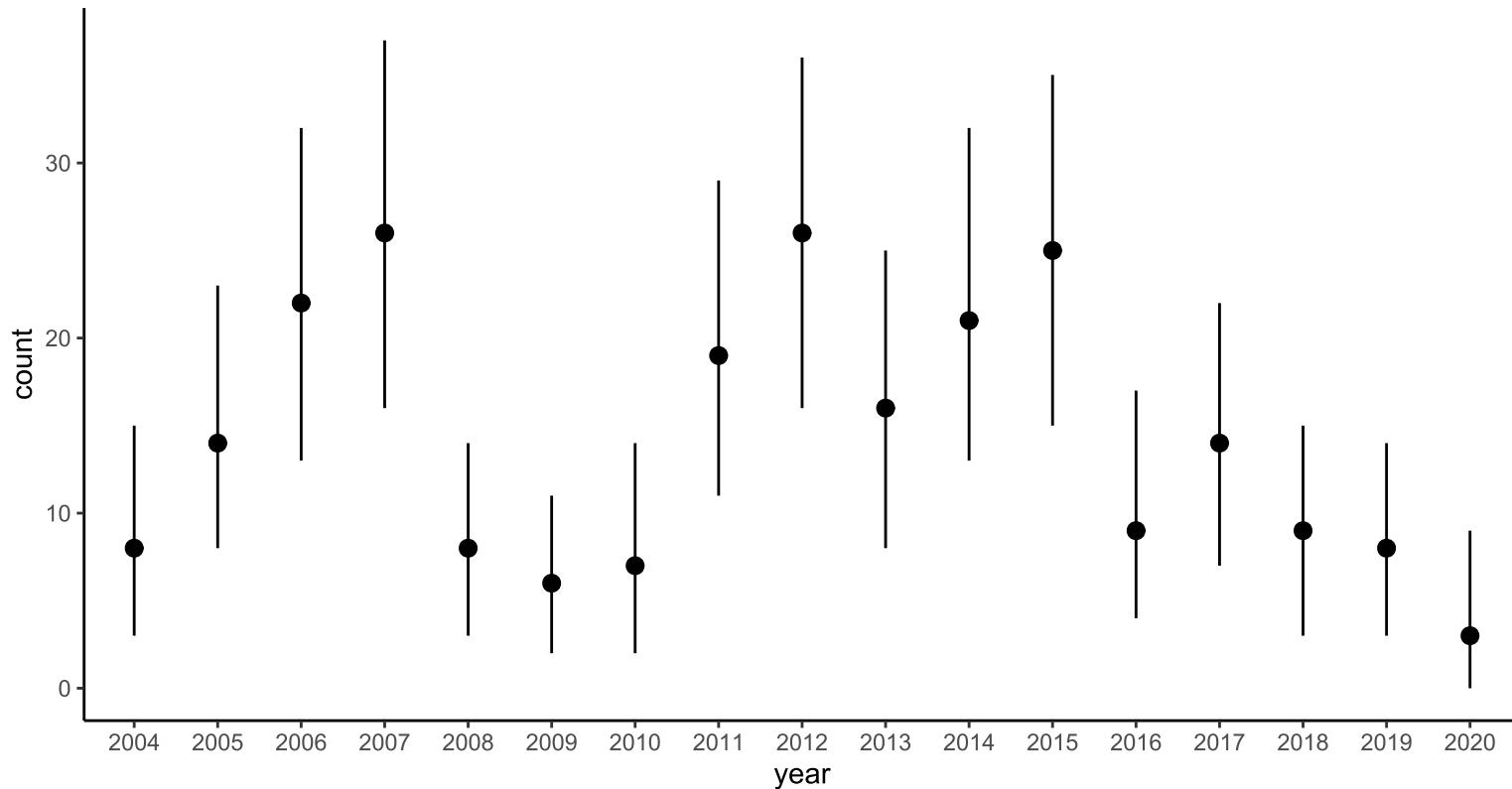
Code    Plot

---

```
# use marginaleffects utilities to plot conditional predictions
library(ggplot2)
plot_predictions(year_random,
                  condition = 'year',
                  n_cores = 2) +
  theme_classic()
```

# Conditional predictions

Code Plot



# Hindcast predictions

---

[Code](#)    [Plot](#)

---

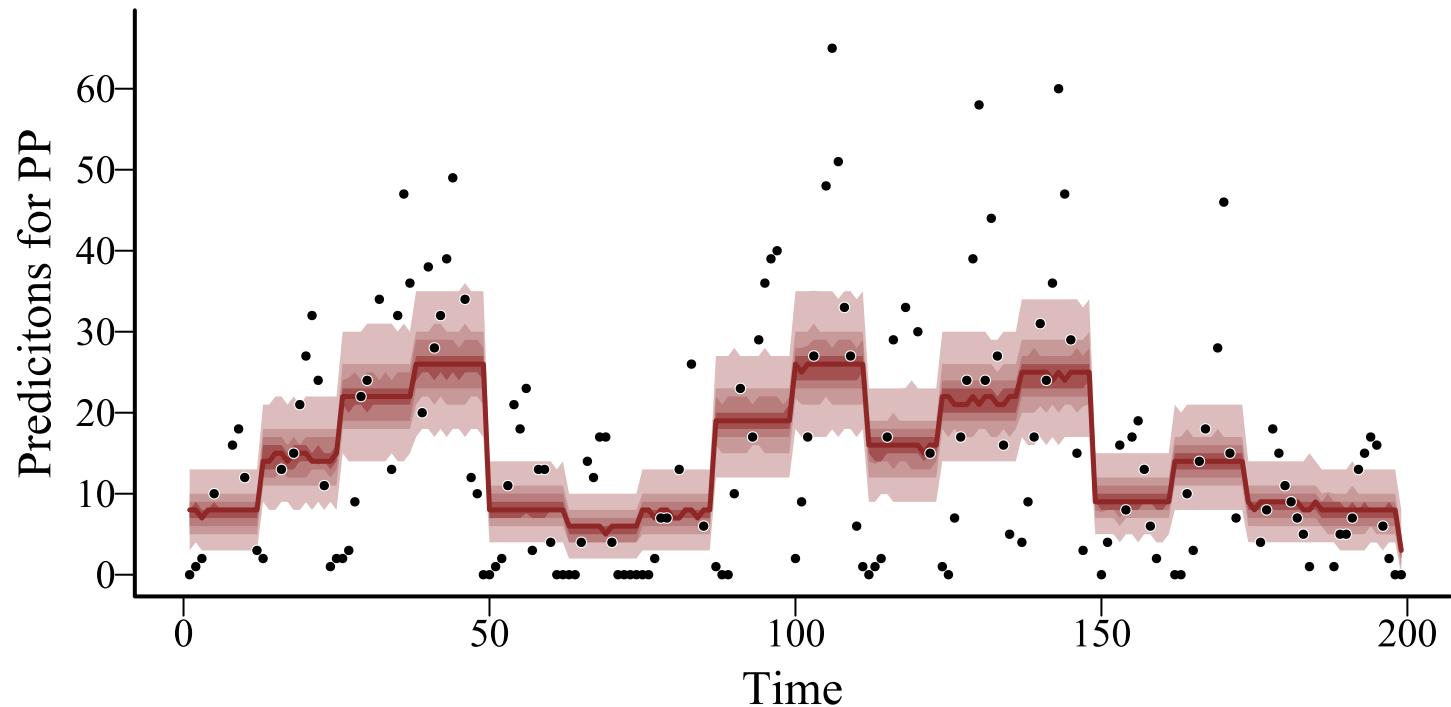
```
# use mvgam's plot to view hindcast predictions
plot(year_random, type = 'forecast')
```

# Hindcast predictions

---

Code Plot

---



# mvgam with yearly smooth

```
model_data %>%  
  dplyr::mutate(year = as.numeric(as.character(year))) → model_data  
  
year_smooth ← mvgam(count ~  
  s(year, bs = 'tp', k = 15),  
  family = poisson(),  
  data = model_data,  
  trend_model = 'None',  
  burnin = 500,  
  samples = 500,  
  chains = 4)
```

A thin plate regression spline of the numeric `year` variable

Retain intercept because smooths are zero-centered

# Coefficients uninterpretable

```
rownames(coef(year_smooth))
```

```
## [1] "(Intercept)" "s(year).1"    "s(year).2"    "s(year).3"    "s(year).4"  
## [6] "s(year).5"   "s(year).6"    "s(year).7"    "s(year).8"    "s(year).9"  
## [11] "s(year).10"  "s(year).11"   "s(year).12"   "s(year).13"   "s(year).14"
```

We **must** use predictions and plots to understand the model

# Estimated yearly smooth

---

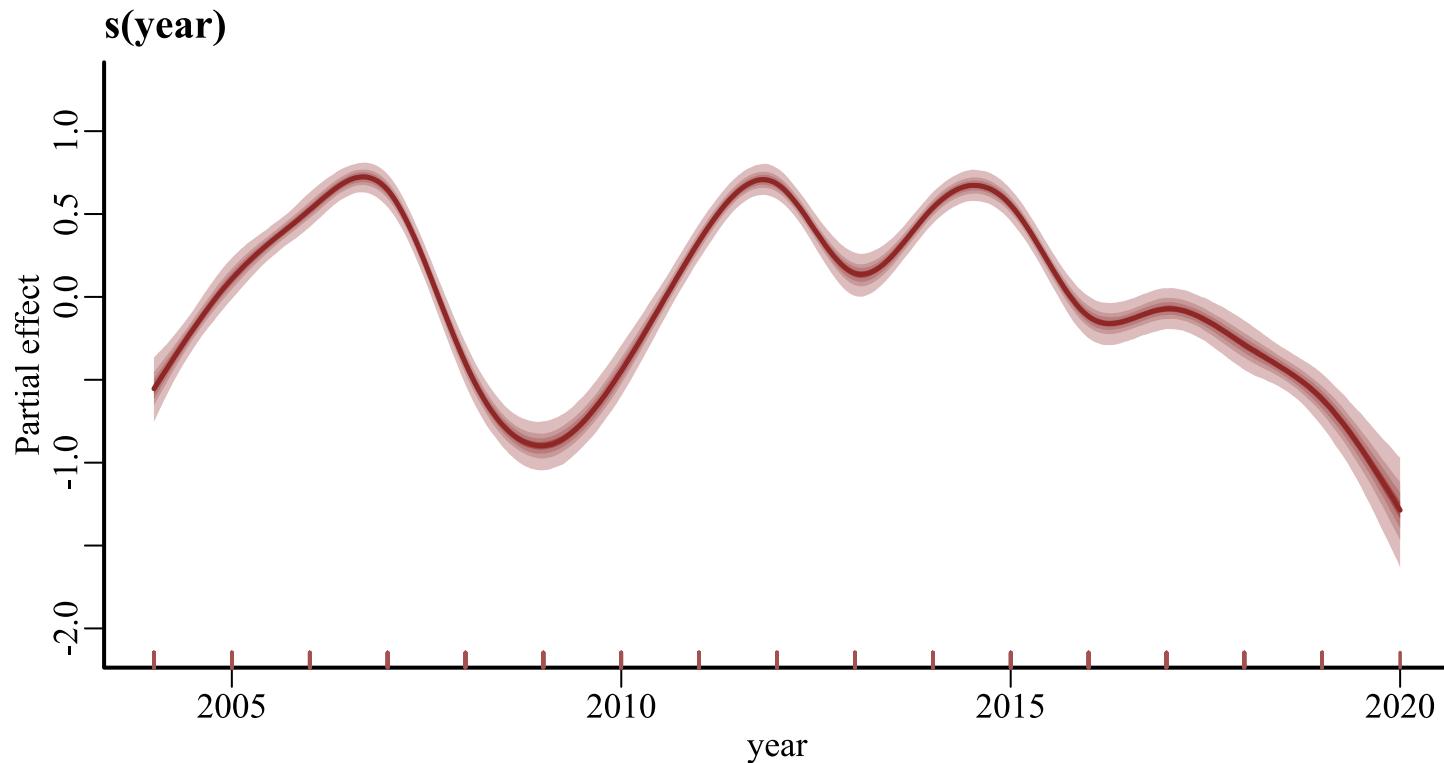
[Code](#) [Plot](#)

---

```
# plot the smooth effect posterior estimates
plot(year_smooth, type = 'smooth')
```

# Estimated yearly smooth

Code      Plot



# Plotting basis functions

---

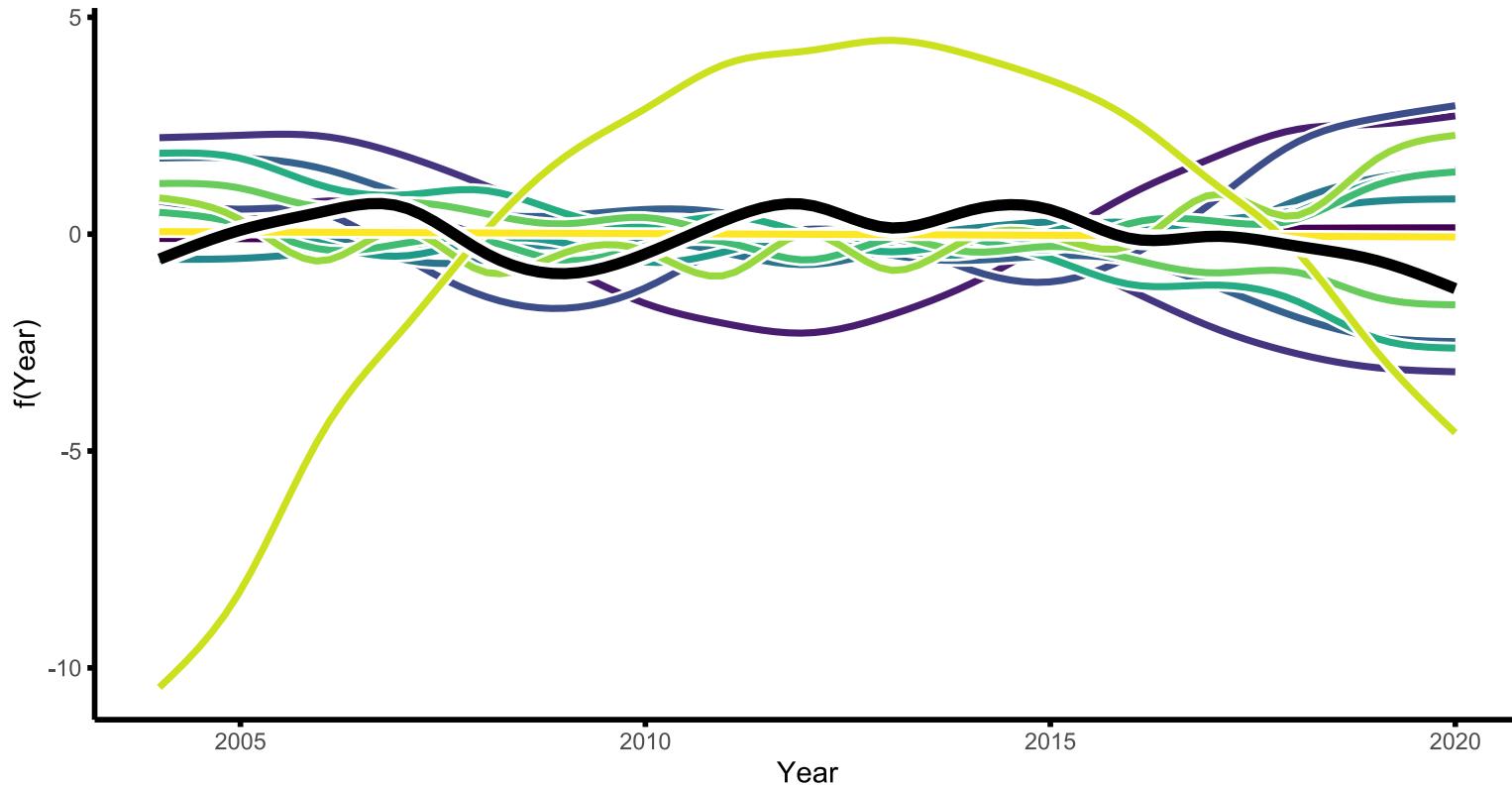
Code    Plot

---

```
# plot the basis functions with gratia
library(ggplot2); library(viridis); library(gratia)
theme_set(theme_classic())
ggplot(basis(year_smooth$mgcv_model),
       aes(x = year, y = value, color = bf)) +
  geom_borderline(linewidth = 1.25, bordercolour = "white") +
  geom_borderline(data = smooth_estimates(year_smooth$mgcv_model),
                  inherit.aes = FALSE,
                  mapping = aes(x = year, y = est), linewidth = 2) +
  scale_color_viridis(discrete = TRUE) +
  theme(legend.position = 'none', axis.line = element_line(size = 1),
        axis.ticks = element_line(colour = "black", size = 1)) +
  ylab('f(Year)') + xlab('Year')
```

# Plotting basis functions

Code Plot



# Rates of change

---

[Code](#)    [Plot](#)

---

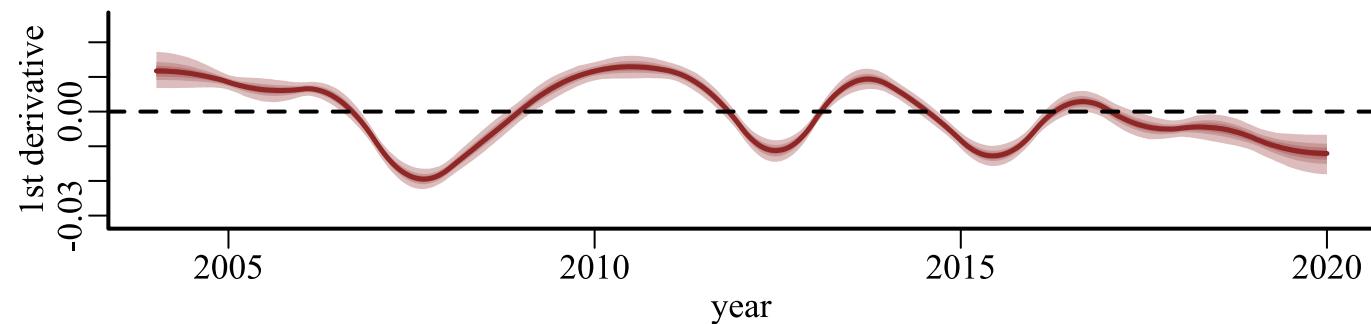
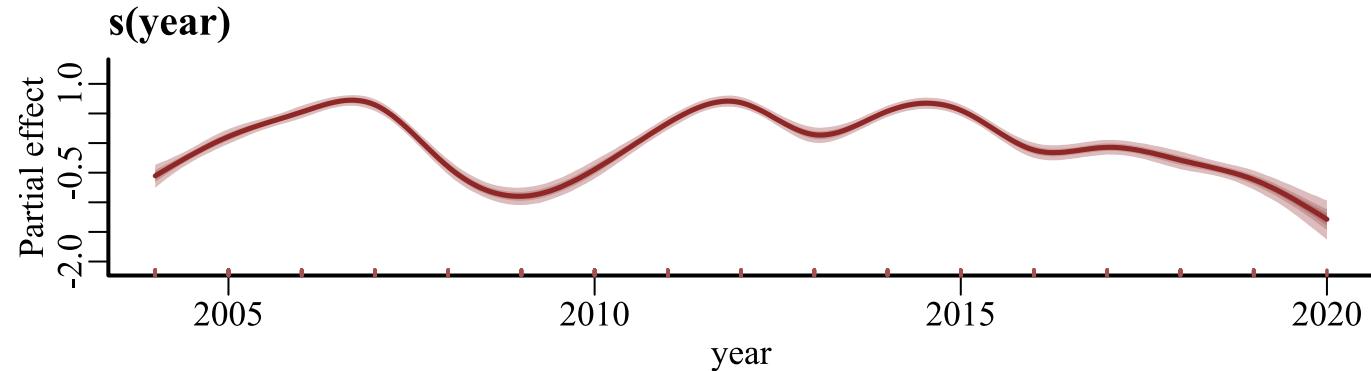
```
# plot the smooth effect posterior estimates
plot(year_smooth, type = 'smooth', derivatives = TRUE)
```

# Rates of change

---

Code      Plot

---



# Conditional predictions

---

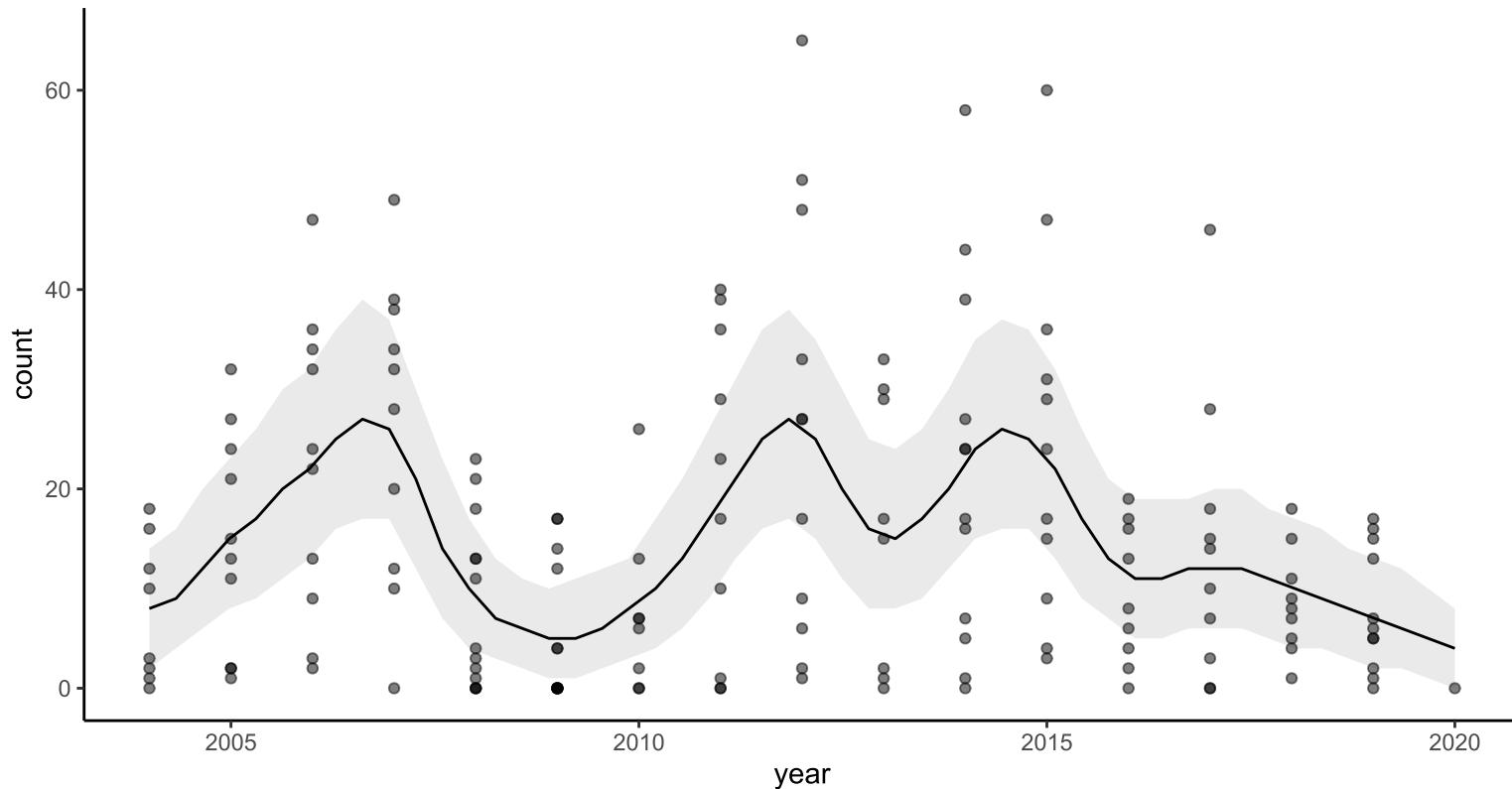
Code    Plot

---

```
# use marginaleffects utilities to plot conditional predictions
library(ggplot2)
plot_predictions(year_smooth,
                  condition = 'year',
                  points = 0.5,
                  n_cores = 2) +
  theme_classic()
```

# Conditional predictions

Code Plot



# Hindcast predictions

---

[Code](#)    [Plot](#)

---

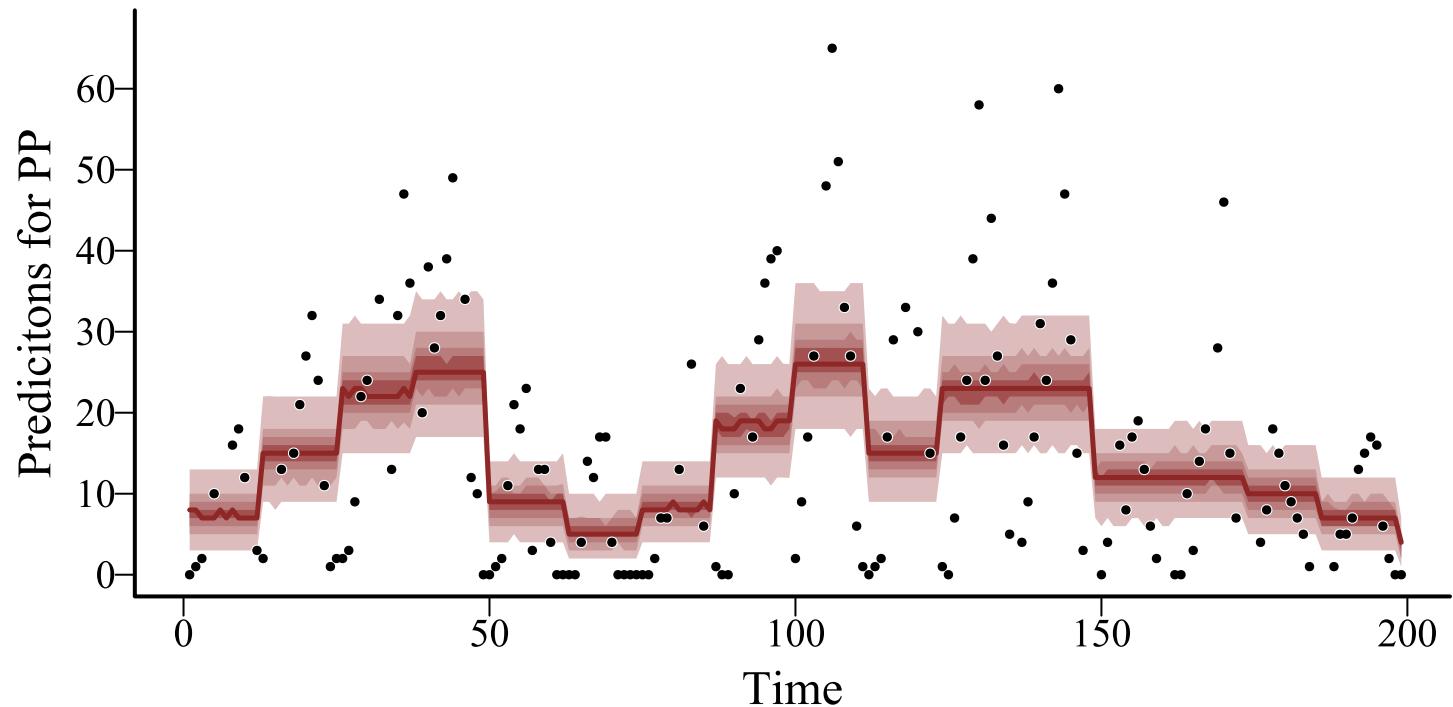
```
# use mvgam's plot to view hindcast predictions
plot(year_smooth, type = 'forecast')
```

# Hindcast predictions

---

Code Plot

---

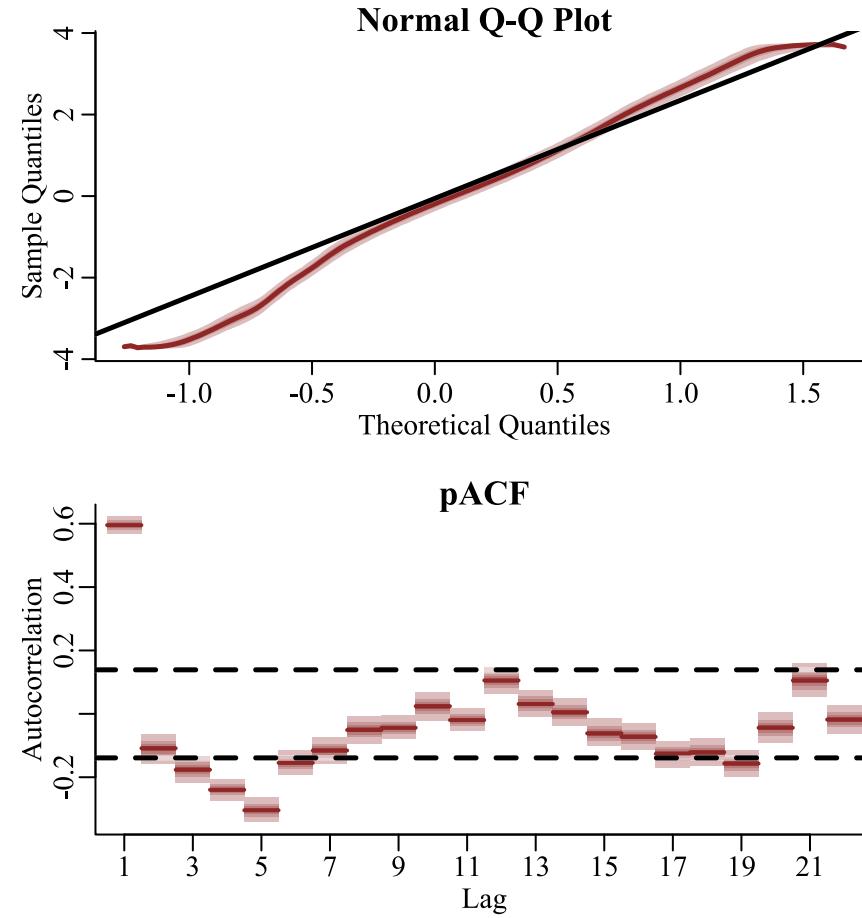
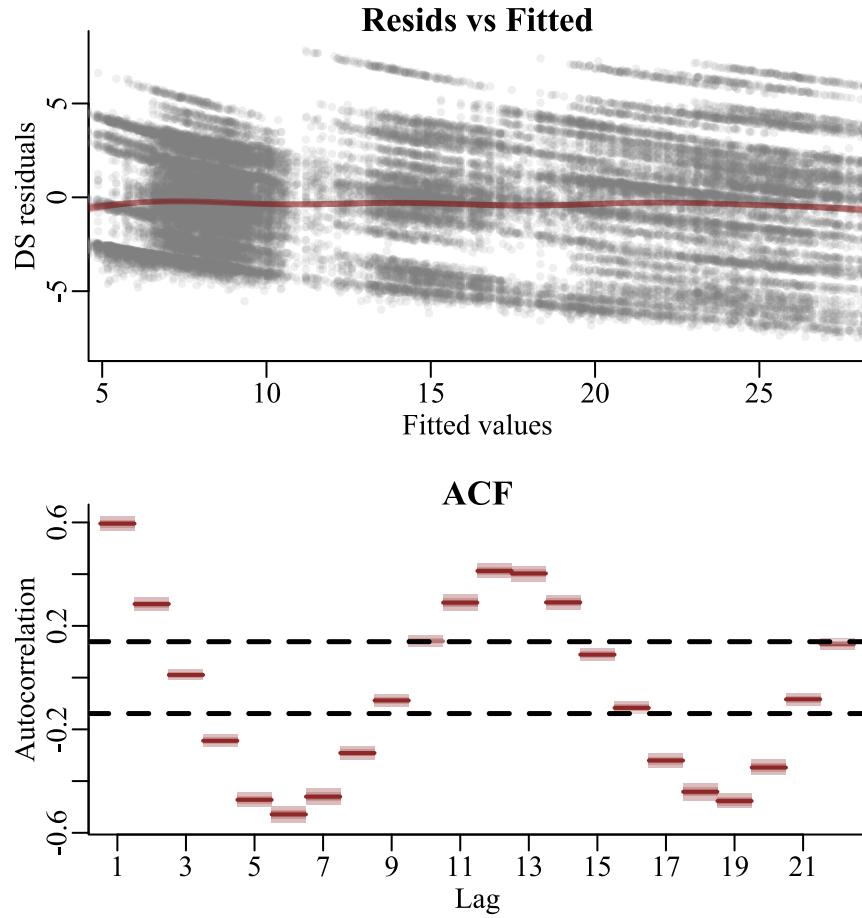


**Forecasts will differ. Why?**

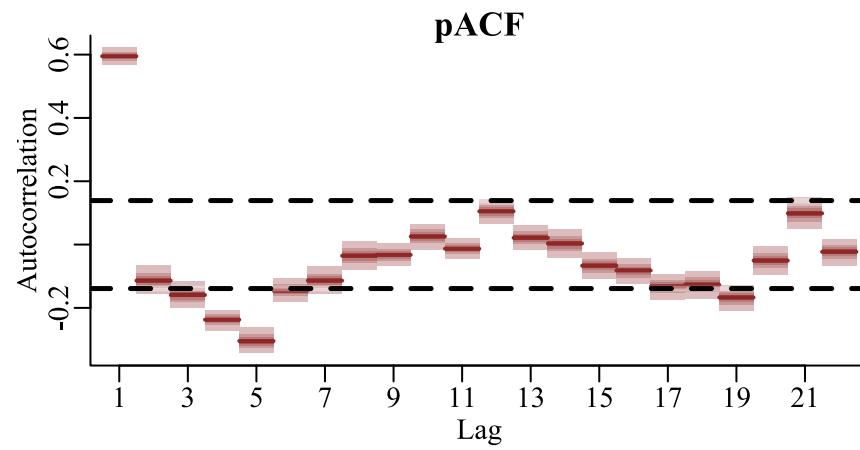
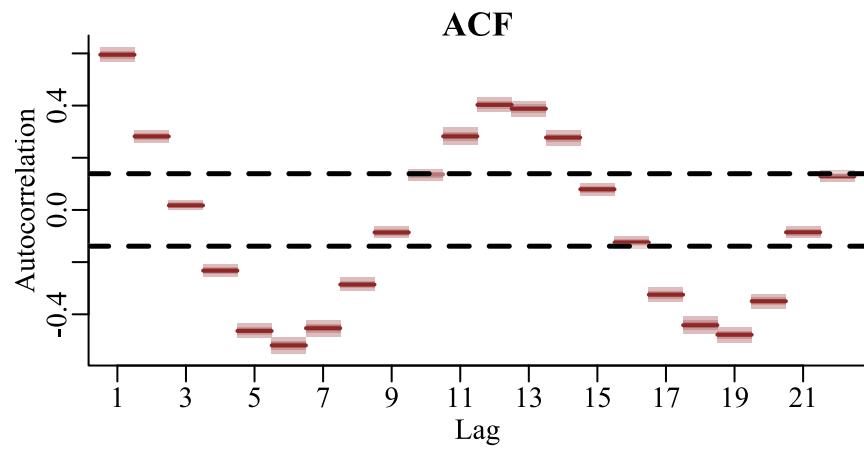
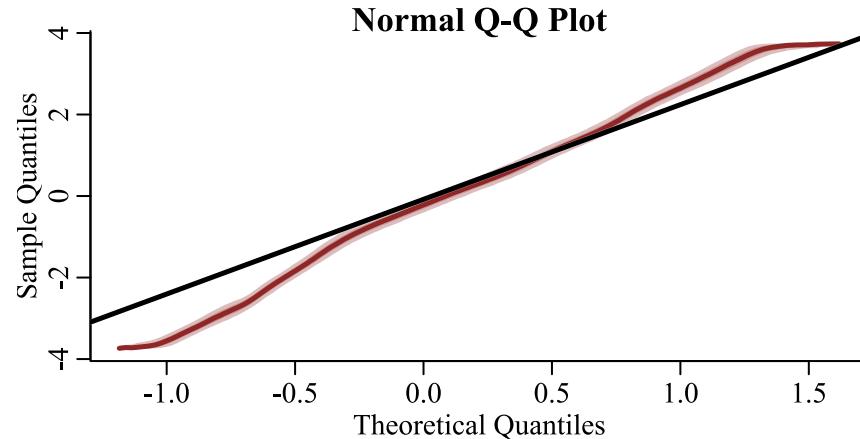
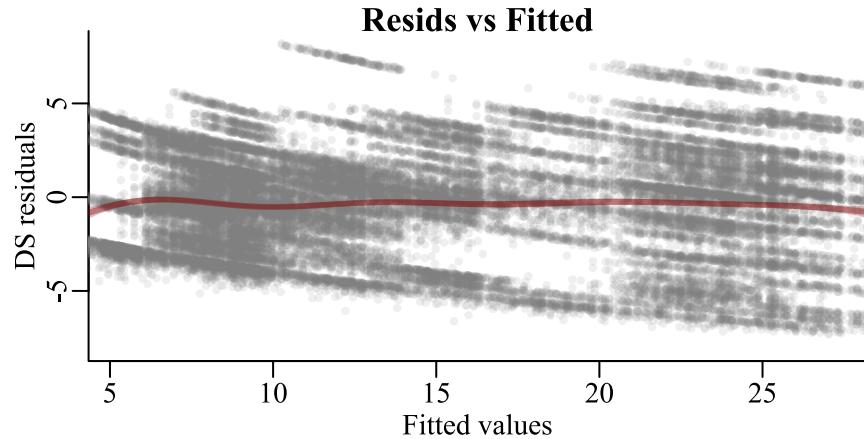
**We will explore this further in the tutorial and in the next lecture**

**But how do model diagnostics look?**

# Random year diagnostics



# Smooth year diagnostics



**Randomized quantile residuals show evidence of unmodelled  
autocorrelation and seasonality**

**How can we deal with the seasonality?**

# Adding a smooth of mintemp

```
year_temp_smooth ← mvgam(count ~  
                           s(year, bs = 'tp', k = 15) +  
                           s(mintemp, bs = 'tp', k = 8),  
                           family = poisson(),  
                           data = model_data,  
                           trend_model = 'None',  
                           burnin = 500,  
                           samples = 500,  
                           chains = 4)
```

A thin plate regression spline of `mintemp`

# Estimated smooths

---

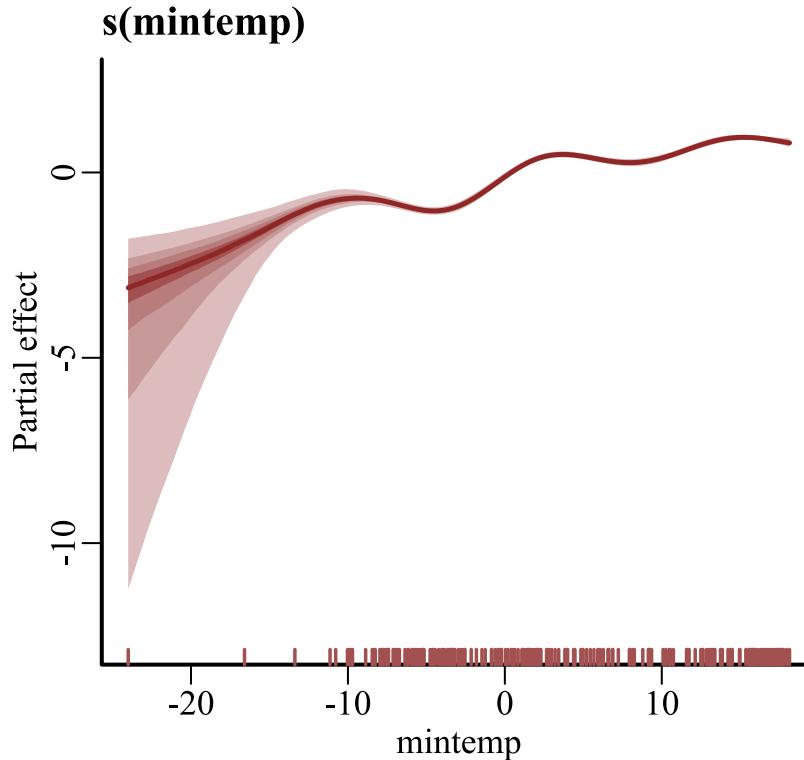
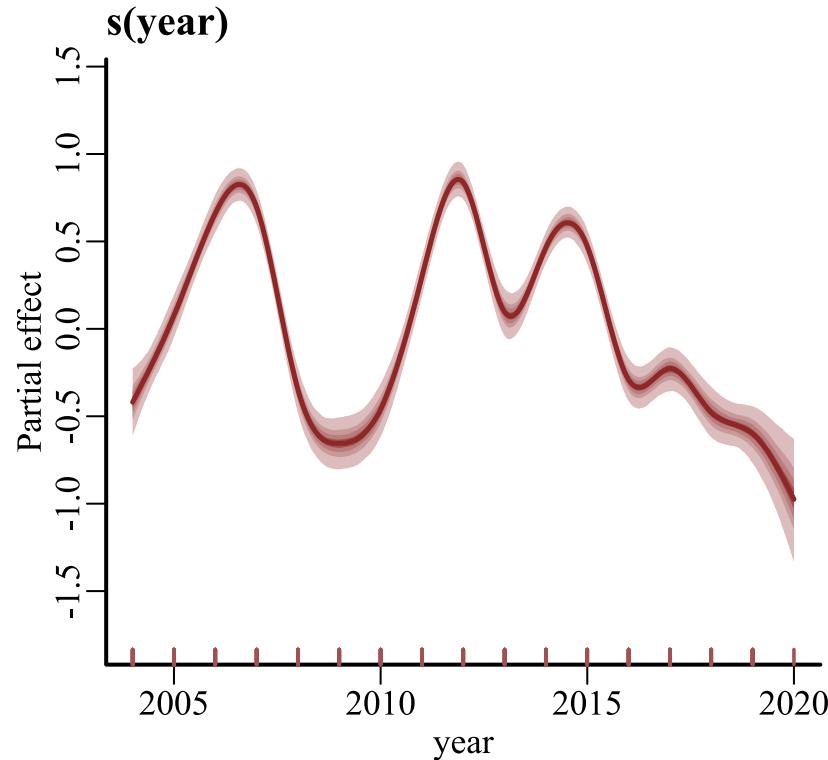
[Code](#)    [Plot](#)

---

```
# use mvgam's plot to view both smooth functions
plot(year_temp_smooth, type = 'smooth')
```

# Estimated smooths

Code      Plot



# Partial residuals

---

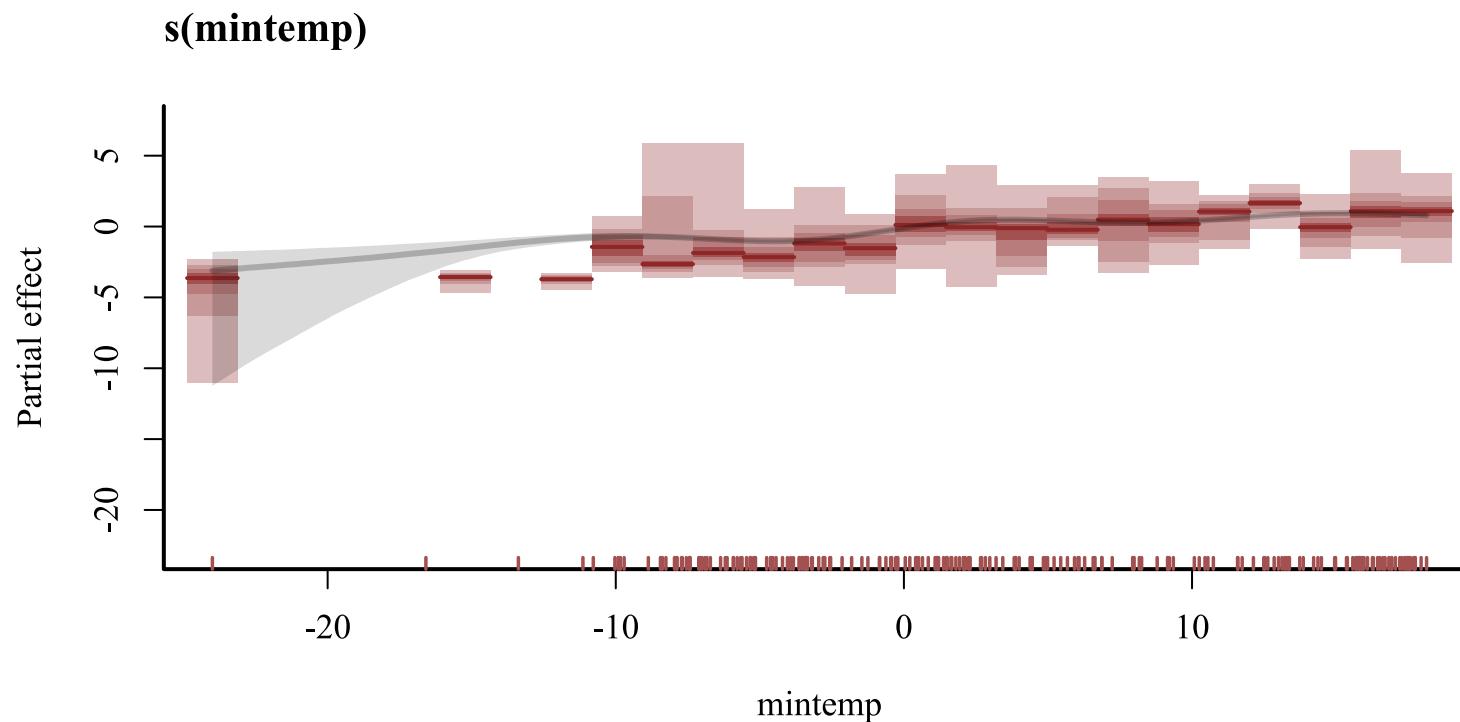
[Code](#)    [Plot](#)

---

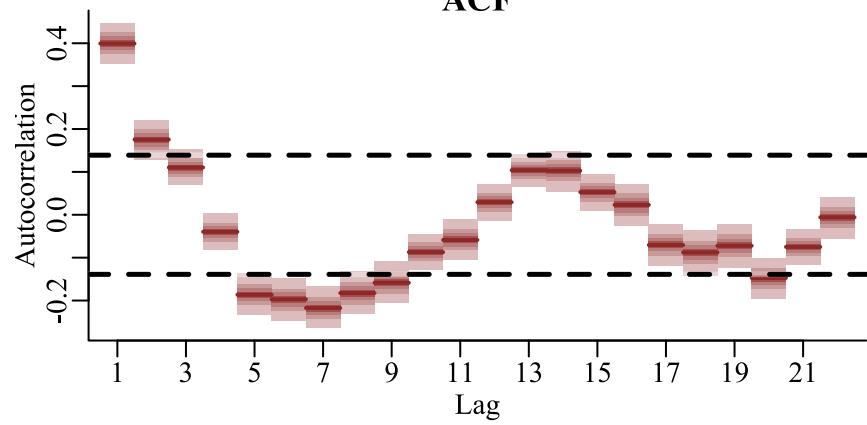
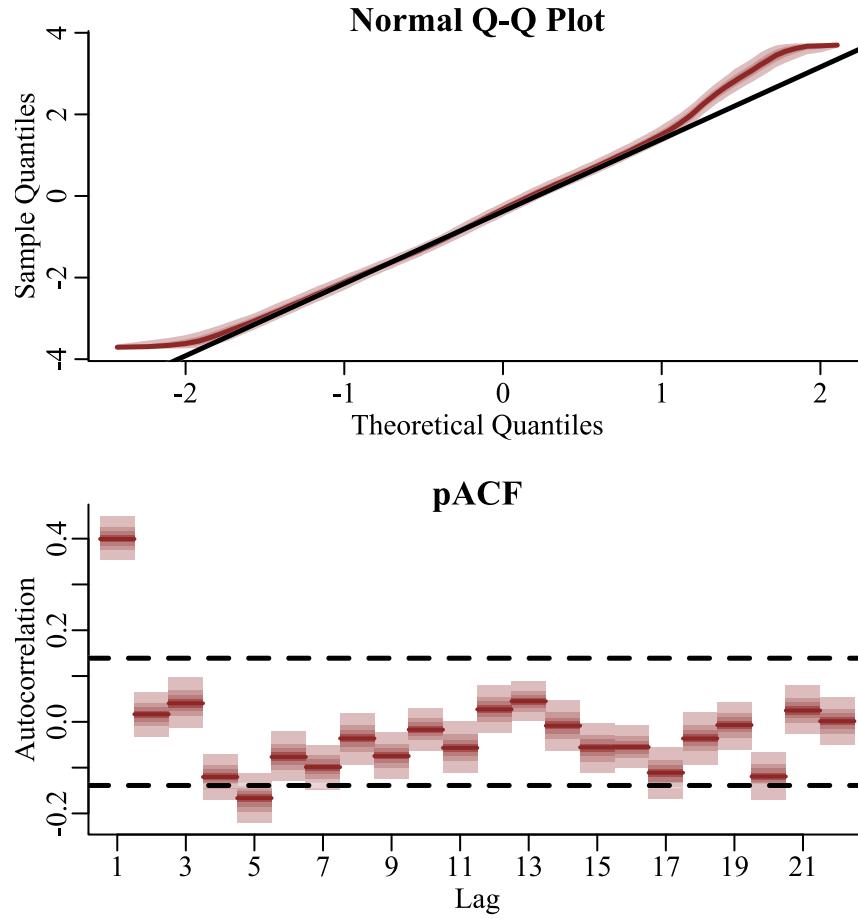
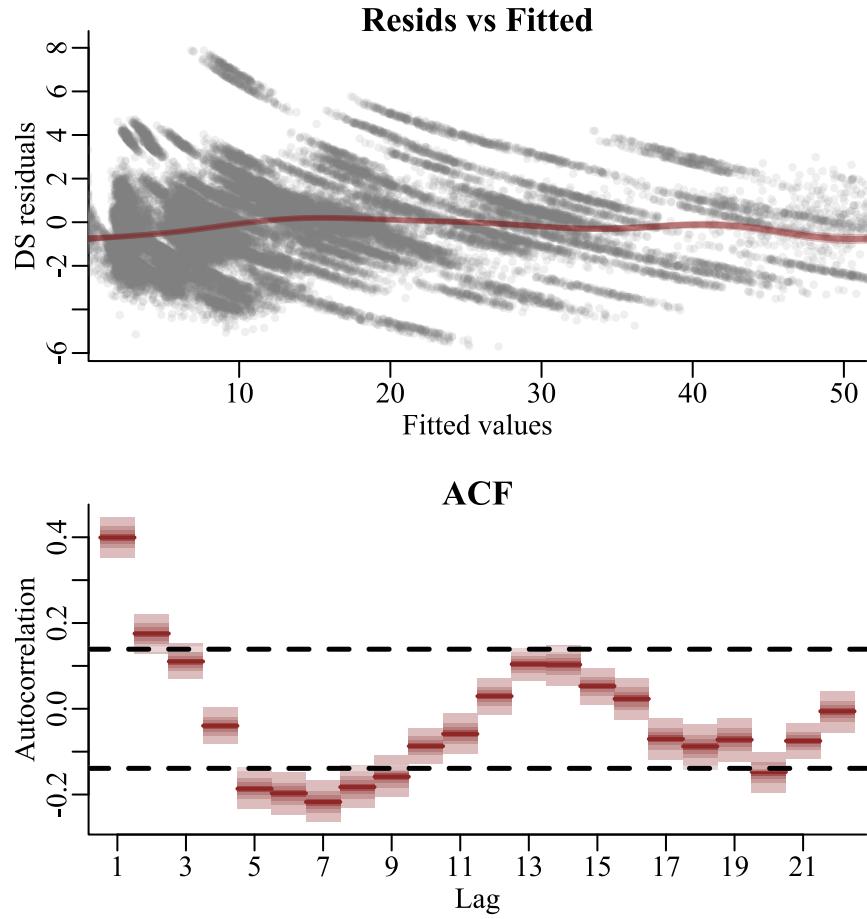
```
# use mgvam's plot_mgvam_smooth to view partial residuals
plot_mgvam_smooth(year_temp_smooth, smooth = 2, residuals = TRUE)
```

# Partial residuals

Code      Plot



# Diagnostics



**Randomized quantile residuals still show evidence of unmodelled autocorrelation**

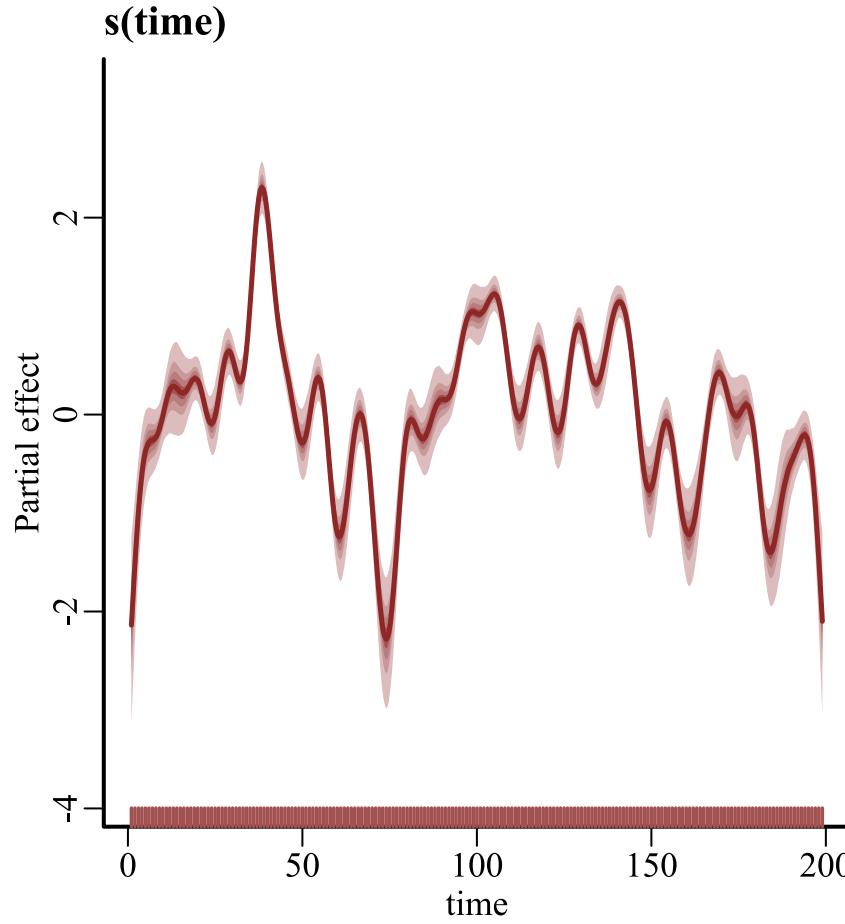
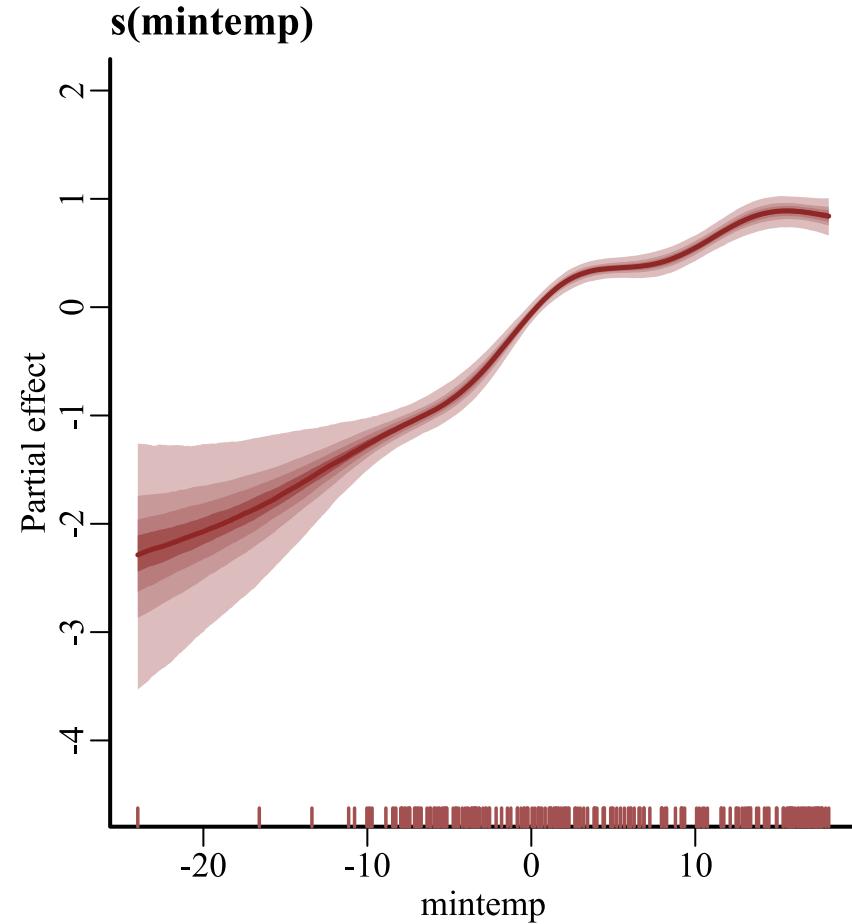
**How can we deal with this?**

# A smooth of time

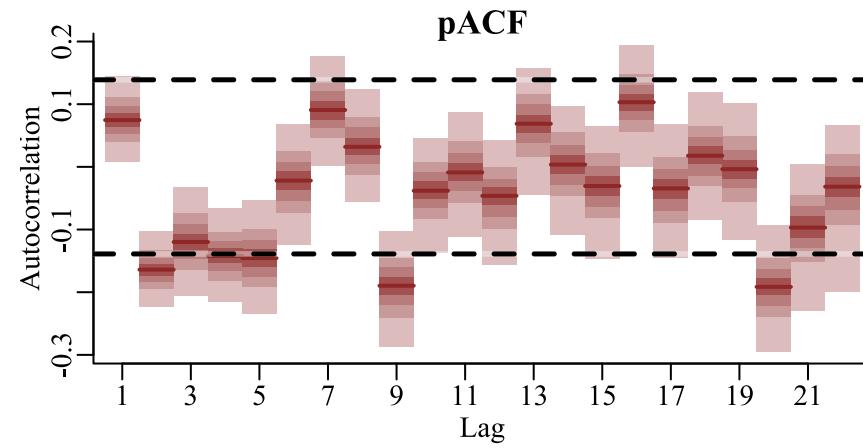
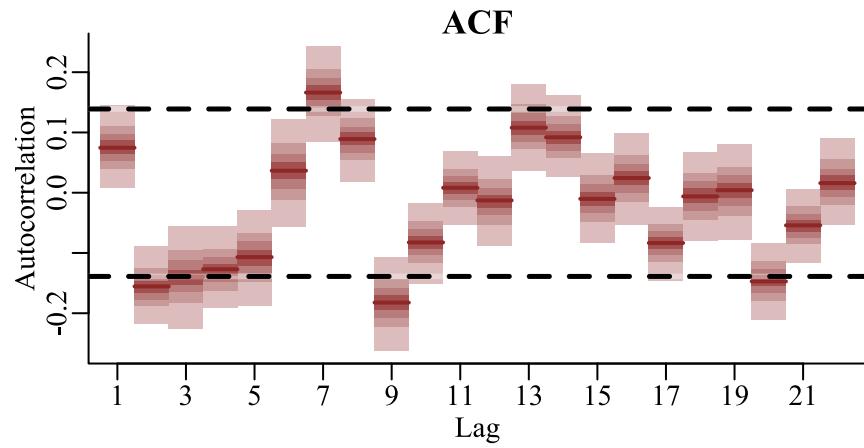
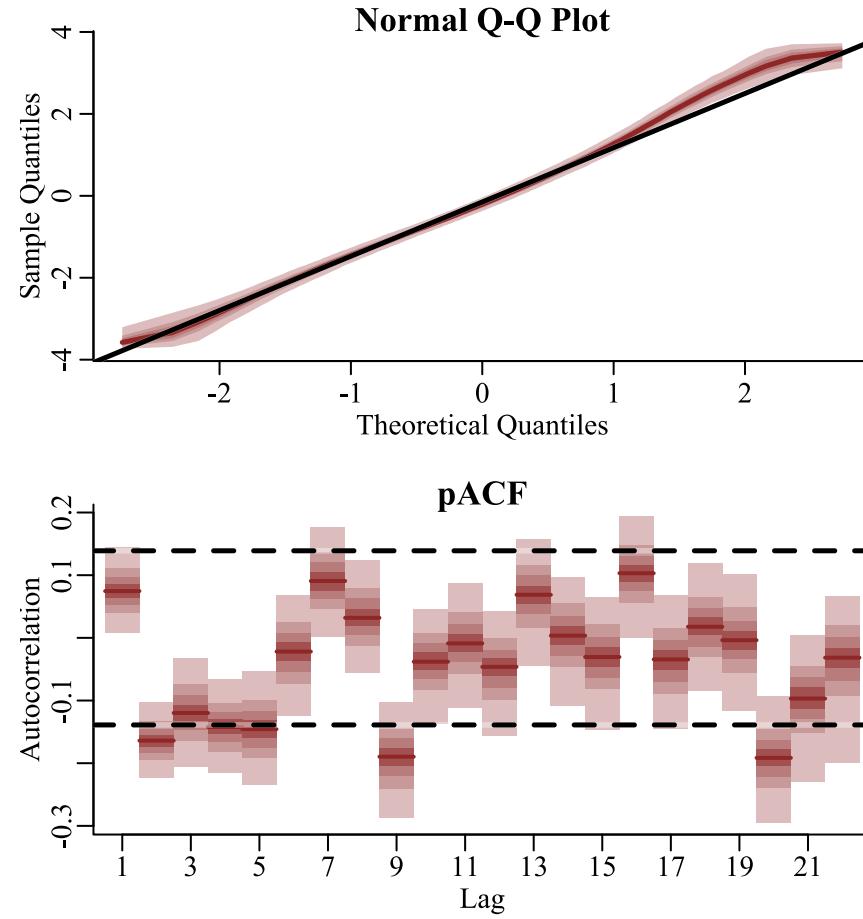
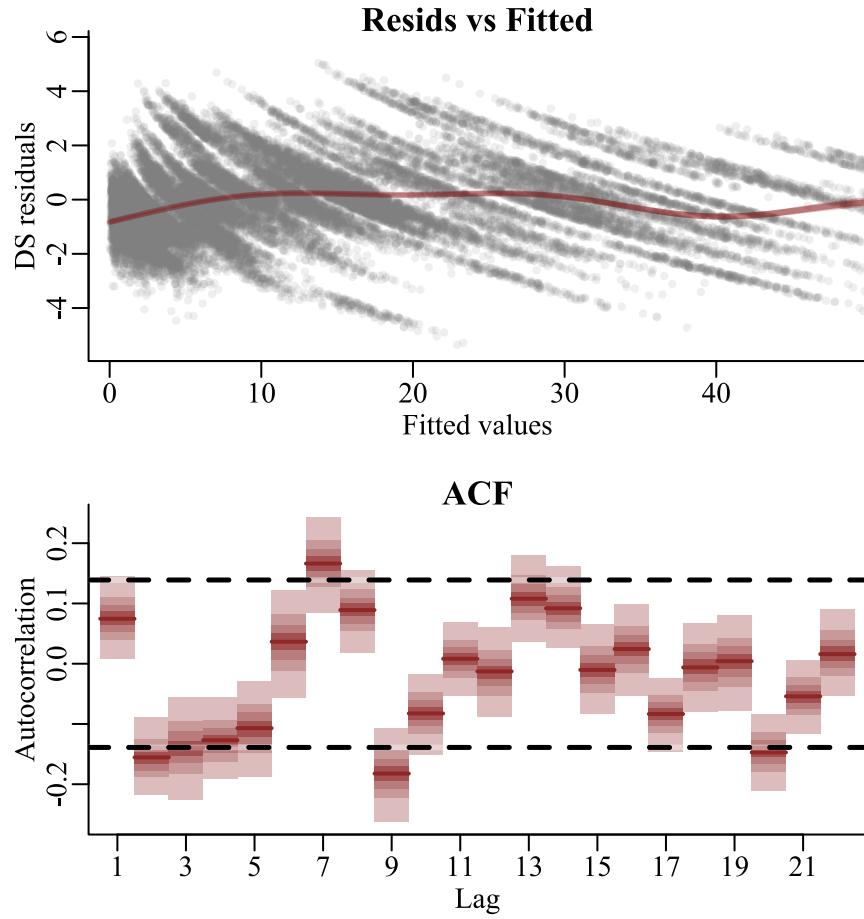
```
temp_time_smooth ← mvgam(count ~  
                           s(mintemp, bs = 'tp', k = 8) +  
                           s(time, bs = 'tp', k = 50),  
                           family = poisson(),  
                           data = model_data,  
                           burnin = 500,  
                           samples = 500,  
                           chains = 4)
```

Replace the spline of `year` with a complex spline of `time` to capture autocorrelation

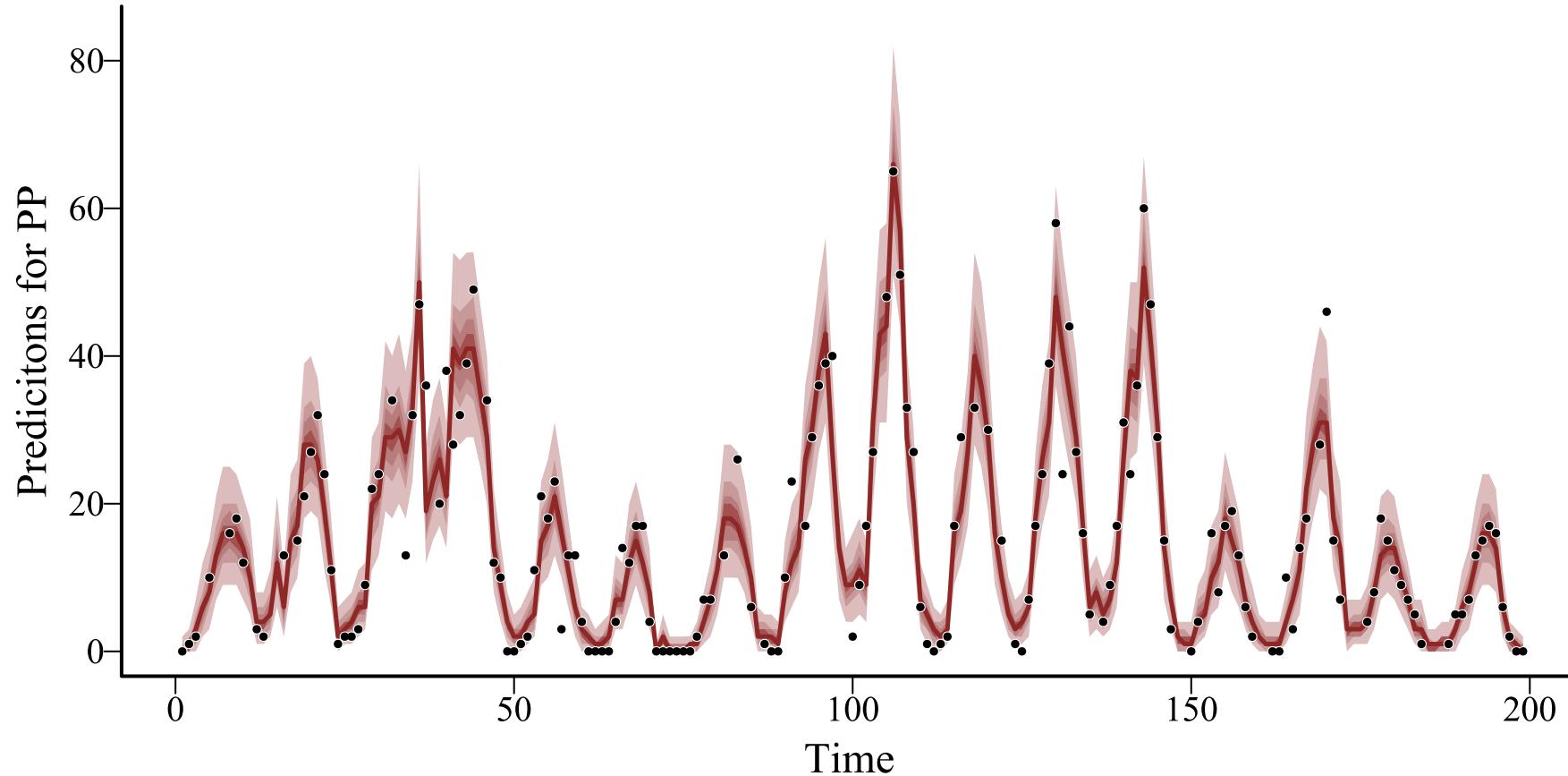
# Updated smooths



# Diagnostics



# Hindcast predictions 😊



**Using an *additive* combination of smooth functions, we have captured a lot of the variation in the observed data**

**But we are dealing with a time series, so we'd like our model to generate sensible forecast predictions**

**As we'll see in the next lecture, this one has some problems**

# In the next lecture, we will cover

Extrapolating splines

Latent autoregressive processes

Latent Gaussian Processes

Dynamic coefficient models