Generalized Additive Models

Fitting Models to Data, Not Data to Models
Model Fitting Series - With Applications in R

Jesse Ghashti
Source code by Stefano Mezzini

October 28, 2025

Centre for Scholarly Communication
The University of British Columbia | Okanagan Campus | Syilx Okanagan Nation Territory

Welcome, feel free to get ready while we wait to start...

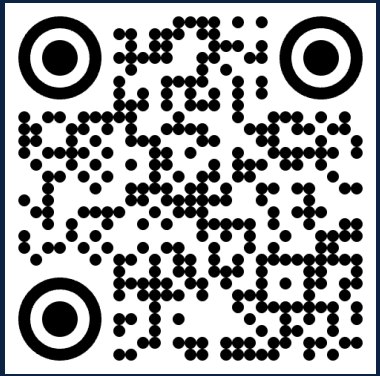
Install Required Packages

```
packages <- c("dplyr", "mgcv", "ggplot2",
              "gratia", "janitor", "tidyr")
toInstall <- packages[!(packages %in%
                          installed.packages()[,"Package"])]
if(length(toInstall)) install.packages(toInstall)

library('dplyr')    # for data wrangling
library('mgcv')     # for modeling
library('ggplot2')  # for fancy plots
library('janitor')  # for data cleaning
library('gratia')   # ggplot-based graphics
library('tidyr')
```

Workshop Series Overview

Session	Topic	Date/Time
1	Simple Linear Regression	Oct 7, 9:00 AM
2	Fitting Linear Models in R	Oct 8, 10:30 AM
3	Multiple Linear Regression in R	Oct 16, 4:00 PM
4	Interaction Terms & Hierarchical Linear Models	Oct 21, 11:00 AM
5	Generalized Linear Models	Oct 23, 4:00 PM
6	Generalized Additive Models (GAMs)	Oct 28, 11:00 AM
7	Interpreting & Predicting from GAMs	Oct 29, 10:30 AM
8	Hierarchical GAMs	Nov 4, 12:00 PM
9	Penalized Models	Nov 18, 11:00 AM
10	Survival Models	Nov 25, 11:00 AM
11	Nonparametric Models	Dec 2, 11:00 AM

New Here?

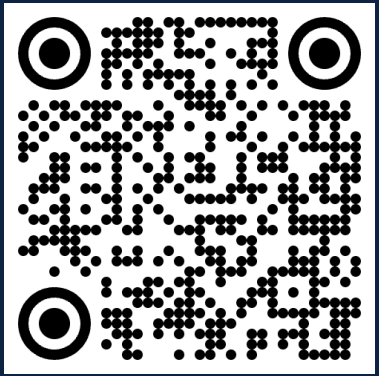


<———————-

New to R? Check out the Fundamentals of R series!

GitHub code and slides for today's workshop (and previous workshops)

———————————>



Alternatively, code/slides available at the bottom of `https://csc-ubc-okanagan.github.io/workshops/`

Last Time (Workshop 5) — Quick Recap

Key Concepts

- Learned about GLMs: Family + Link + Linear Predictor
- Moved beyond normal distributions (Gamma, Poisson, Binomial)
- Used link functions to map constrained means to unconstrained scales
- Interpreted coefficients on link vs response scales
- Key insight: Choose family based on response type!

Today: Generalized Additive Models (GAMs)

Today we will...

- Move beyond linear relationships
- Understand smooth functions and basis expansions
- Learn when to use GAMs vs GLMs
- Fit smooth effects of time in `ChickWeight`
- Compare LM, GLM, and GAM predictions
- Visualize smooths with `gratia::draw()`

Today we require...

```
library('dplyr')   # for data wrangling
library('tidyr')   # for expand_grid
library('mgcv')    # for modeling
library('ggplot2') # for fancy plots
library('gratia')  # for ggplot-based model graphics
theme_set(theme_classic(base_size = 15))
```

Why Do We Need GAMs?

Limitations of GLMs

GLMs assume:

- Effects are **linear on the link scale**
- Example: $\log(\mu) = \beta_0 + \beta_1 \text{Time}$ means exponential growth
- But what if growth rate changes over time?
- What if the relationship is more complex?

Real data often has nonlinear patterns:

- Growth curves with saturation
- Seasonal patterns
- Dose-response curves
- Temperature effects (optimal ranges)

GAMs add flexibility by replacing linear terms with smooth functions

What is a Generalized Additive Model?

From GLM to GAM

GLM:

$$g(\mu) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots$$

GAM:

$$g(\mu) = \beta_0 + f_1(x_1) + f_2(x_2) + \cdots$$

where f_j are **smooth functions** estimated from the data

Simply put, we let the data tell determine the shape of the relationship, not force it to be linear.

Remember: all GLMs are GAMs, but not all GAMs are GLMs. Similarly, all LMs are GLMs.

What are Smooth Functions?

Building Smooths

Basis functions:

- Smooths are built from basis functions (like polynomial terms)
- Controlled by the basis dimension which controls flexibility

Penalization:

- Prevents overfitting
- Balances fit and complexity

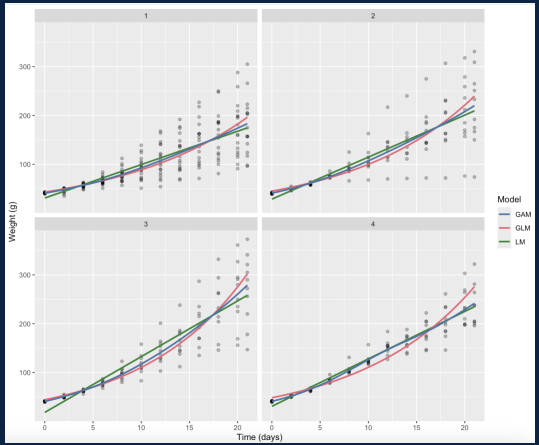
Syntax in R

- `s(x)`: smooth of x
- `s(x, k = 10)`: limit flexibility
- `s(x, bs = 're')`: random effect
- `s(x, y)`: 2-dimensional smooth
- `s(x, by = group)`: group-specific smooth

GAMs automatically determine how flexible the smooth should be.

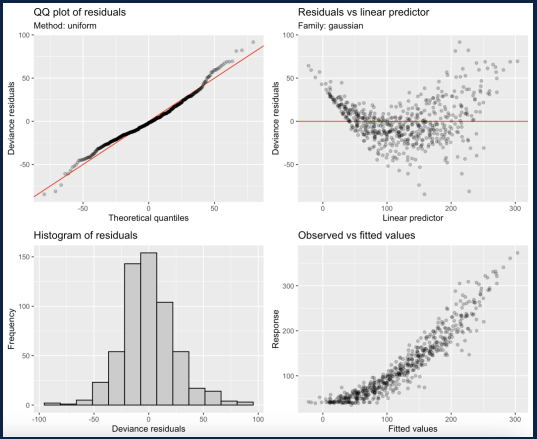
Exploratory Visualization: LM vs GLM vs GAM

```r
# Plot the data and add exploratory models
ggplot(ChickWeight, aes(Time, weight)) +
  facet_wrap(~ Diet) +
  geom_point(alpha = 0.3) +
  # Two weeks ago: Linear Model
  geom_smooth(method = 'lm', formula = y ~ x, aes(color = 'LM'),
              se = FALSE, lwd = 1) +
  # Last week: Gamma GLM
  geom_smooth(method = 'glm', formula = y ~ x, aes(color = 'GLM'),
              method.args = list(family = Gamma(link = 'log')),
              se = FALSE, lwd = 1) +
  # This week: Gamma GAM with smooth
  geom_smooth(method = 'gam', formula = y ~ s(x), aes(color = 'GAM'),
              method.args = list(family = Gamma(link = 'log')),
              se = FALSE, lwd = 1) +
  khroma::scale_color_bright(name = 'Model') +
  labs(x = 'Time (days)', y = 'Weight (g)')
```



Review: Linear Model (Workshop 4)

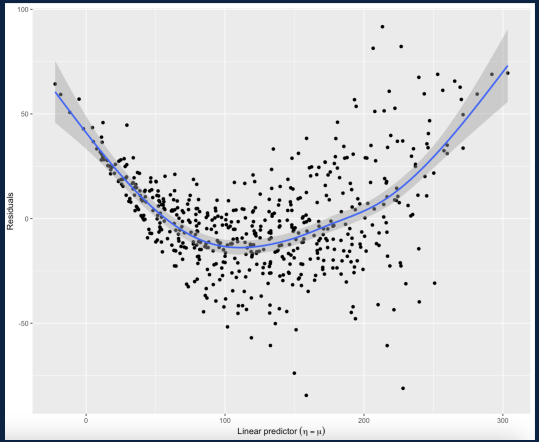
```r
# All LMs are GLMs with gaussian()
m_cw_lm <- gam(
  formula = weight ~
    Time +                # linear effect
    Diet +                # intercept shift
    Time:Diet +           # different slopes
    s(Chick, bs = 're'),# random effects
  family = gaussian(),  # normal errors
  data = ChickWeight,
  method = 'ML')

appraise(m_cw_lm, point_alpha = 0.25)
```



Linear Model: Residual Patterns

```r
# check residuals vs linear predictor
ggplot(mapping = aes(
    x = predict(m_cw_lm, type = 'link'),
    y = residuals(m_cw_lm))) +
  geom_point() +
  geom_smooth(method = 'gam') +
  labs(x = expression(
    Linear~predictor~(eta == mu)),
       y = 'Residuals')
```

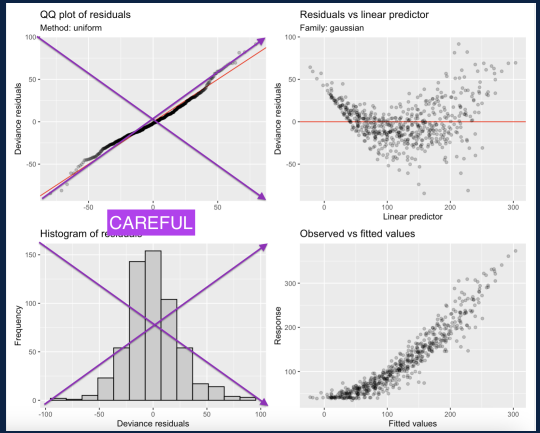
Notice Heteroscedasticity (variance increases with fitted values)



Review: Gamma GLM (Workshop 5)

```r
# Gamma HGLM from last week
# All GLMs are GAMs
m_cw_glm <- gam(
  formula = weight ~
    Time +  # linear on LINK scale
    Diet +
    Time:Diet +
    s(Chick, bs = 're'),
  family = Gamma(link = 'log'),
  data = ChickWeight,
  method = 'ML')

appraise(m_cw_glm, point_alpha = 0.25)
```

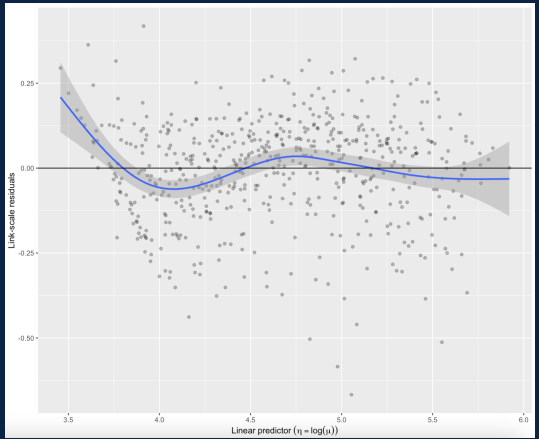
Gamma handles variance-mean relationship (remember the caution of the diagnostics plots here)



Gamma GLM: Residual Patterns

```r
# check residuals vs linear predictor
ggplot(mapping = aes(
    x = predict(m_cw_glm, type = 'link'),
    y = residuals(m_cw_glm))) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = 'gam') +
  geom_hline(yintercept = 0) +
  labs(x = expression(
    Linear~predictor~(eta == log(mu))),
       y = 'Link-scale residuals')
```

Notice there is still some pattern — maybe relationship isn't perfectly linear on log scale?

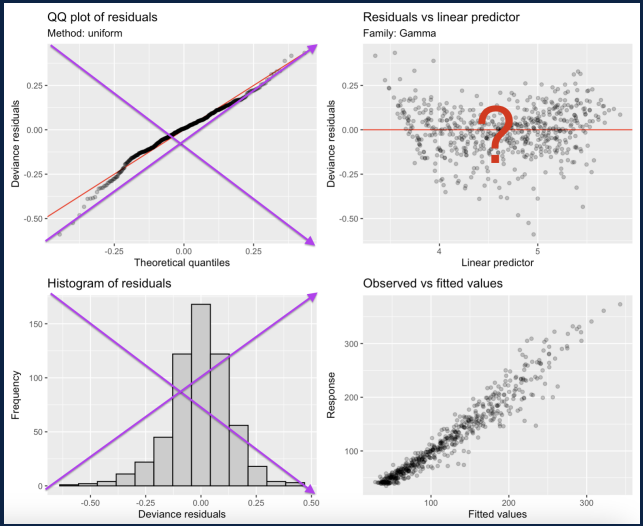


Fitting a GAM: Diet-Specific Smooths

```r
# Gamma HGAM with no common smooth of time
m_cw_gam <- gam(
  formula = weight ~
    s(Time, Diet, bs = 'fs') +  # smooth effect for each diet
    Diet +                      # maintain different intercepts
    s(Chick, bs = 're'),        # random chick effects
  family = Gamma(link = 'log'),
  data = ChickWeight,
  method = 'REML')              # Restricted ML (preferred for smooths)

appraise(m_cw_gam, point_alpha = 0.25)
```

`bs = 'fs'` creates a "factor smooth" — separate smooth for each level of Diet with shared smoothness penalty

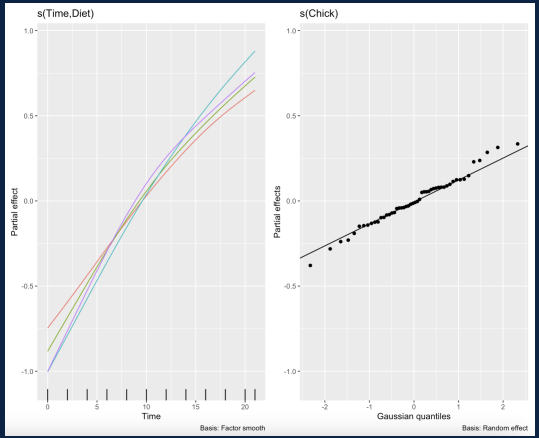
GAM Diagnostics: Diet-Specific Smooths



Visualizing the Smooth Terms

```
# visualize smooth terms
# draw(m_cw_gam)
# with fixed scales for comparison
draw(m_cw_gam, scales = 'fixed')
```

- Smooth function on y-axis
- Time on x-axis
- Separate curve for each Diet
- Confidence bands



Hierarchical GAM: Common + Deviations

Modeling Philosophy

Instead of completely separate smooths, we can model:

- A **common smooth** that all groups share
- **Group-specific deviations** from that common pattern
- **Individual-level deviations** for repeated measures

This is more efficient and helps identify shared patterns

$$f_{\text{Diet}_j}(t) = f_{\text{common}}(t) + f_{\text{Diet}_j}(t) + f_{\text{Chick}_i}(t)$$

Fitting a Hierarchical GAM

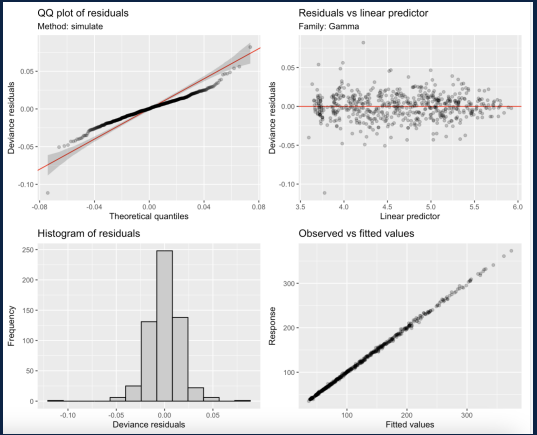
```r
# Gamma HGAM with common smooth of time
# Using bam() for faster computation with large datasets
m_cw_gam <- bam(
  formula = weight ~
    s(Time) +                    # COMMON smooth effect of time
    s(Time, Diet, bs = 'fs') +   # Diet-specific deviations
    s(Time, Chick, bs = 'fs') +  # Chick-level deviations
    Diet,                        # Parametric diet effect
  family = Gamma(link = 'log'),
  data = ChickWeight,
  method = 'fREML',              # fast REML for bam()
  discrete = TRUE)               # required for fREML

appraise(m_cw_gam, n_simulate = 1000, method = 'simulate', point_alpha = 0.25)
```

`bam()` is optimized for large datasets. Here we use simulation-based diagnostics.

Hierarchical GAM: Diagnostic Plots

Notice how our fitted values are nearly perfect... overfitting?
–> we will see next time.

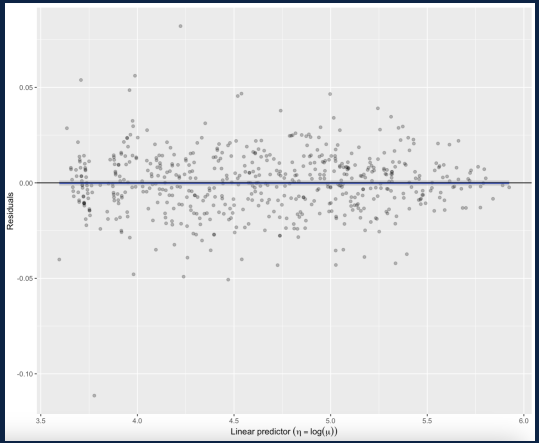
Using 1000 simulations for reference bands



Hierarchical GAM: Residual Check

```
ggplot(mapping = aes(
    x = predict(m_cw_gam,
                type = 'link'),
    y = residuals(m_cw_gam))) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = 'gam') +
  geom_hline(yintercept = 0) +
  labs(x = expression(
    Linear~predictor~(eta == log(mu))),
       y = 'Residuals')
```

No more systematic patterns



Model Summary 1 of 2: LM and GLM

LM

```
Family: gaussian
Link function: identity

Formula:
weight ~ Time + Diet + Time:Diet + s(Chick, bs = "re")

Parametric coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  31.5081     5.9358   5.308 1.63e-07 ***
Time          6.7130     0.2584  25.982  < 2e-16 ***
Diet2        -2.8745    10.2342  -0.281    0.779
Diet3       -13.2577    10.2342  -1.295    0.196
Diet4        -0.3983    10.2430  -0.039    0.969
Time:Diet2    1.8961     0.4285   4.425 1.17e-05 ***
Time:Diet3    4.7099     0.4285  10.992  < 2e-16 ***
Time:Diet4    2.9495     0.4340   6.795 2.92e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
           edf Ref.df     F p-value
s(Chick) 41.15     46 9.879  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.873   Deviance explained = 88.3%
-ML =   2744  Scale est. = 643.72    n = 578
```

GLM

```
Family: Gamma
Link function: log

Formula:
weight ~ Time + Diet + Time:Diet + s(Chick, bs = "re")

Parametric coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.777906   0.038080  99.210  < 2e-16 ***
Time        0.067876   0.001558  43.578  < 2e-16 ***
Diet2       0.051339   0.065687   0.782  0.43481
Diet3       0.019151   0.065687   0.292  0.77074
Diet4       0.101373   0.065737   1.542  0.12365
Time:Diet2  0.008199   0.002582   3.175  0.00158 **
Time:Diet3  0.022041   0.002582   8.536  < 2e-16 ***
Time:Diet4  0.014585   0.002616   5.576 3.93e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
           edf Ref.df     F p-value
s(Chick) 41.83     46 12.08  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.907   Deviance explained = 93.2%
-ML = 2493.3  Scale est. = 0.023364  n = 578
```

Model Summary 2 of 2: GAM and HGAM

GAM

```
Family: Gamma
Link function: log

Formula:
weight ~ s(Time, Diet, bs = "fs") + Diet + s(Chick, bs = "re")

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.47447    0.03731 119.927  < 2e-16 ***
Diet2        0.13470    0.06373   2.114 0.035031 *
Diet3        0.24531    0.06373   3.849 0.000133 ***
Diet4        0.24563    0.06375   3.853 0.000131 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
               edf Ref.df      F p-value
s(Time,Diet) 12.81     36 249.42  <2e-16 ***
s(Chick)     42.79     46  14.59  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.925   Deviance explained = 94.5%
-REML = 2474.6  Scale est. = 0.01964   n = 578
>
```

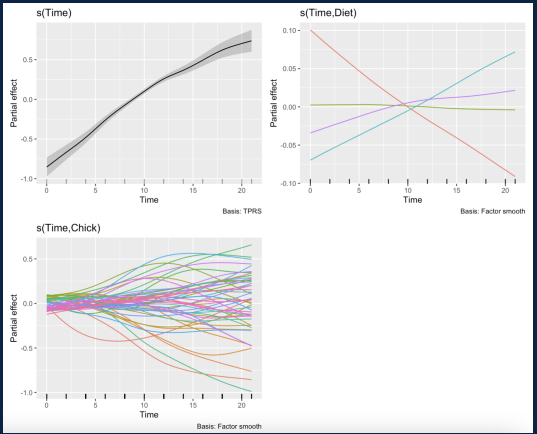
HGAM

```
Family: Gamma
Link function: log

Formula:
weight ~ s(Time) + s(Time, Diet, bs = "fs") + s(Time, Chick,
    bs = "fs") + Diet

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.41465    0.06963  63.404   <2e-16 ***
Diet2        0.15164    0.10195   1.487   0.1380
Diet3        0.25649    0.10195   2.516   0.0124 *
Diet4        0.26254    0.10195   2.575   0.0105 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                edf Ref.df      F p-value
s(Time)       8.400   8.76 32.902  <2e-16 ***
s(Time,Diet)  3.233  38.00  0.161  <2e-16 ***
s(Time,Chick) 281.392 483.00 92.912  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.998   Deviance explained = 99.9%
fREML = -766.85  Scale est. = 0.00055821  n = 578
>
```

Visualizing All Smooth Components I

```
# All smooth terms
draw(m_cw_gam)
```

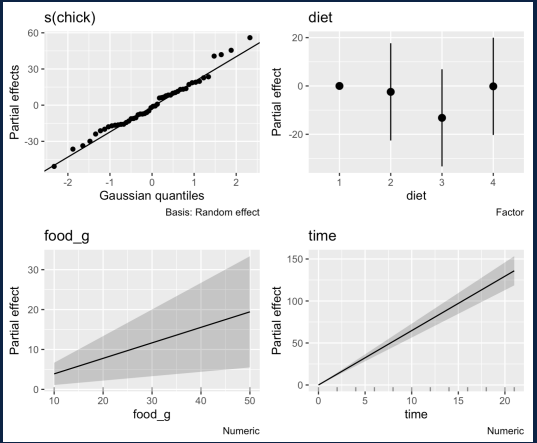
- `s(Time)`: the common growth pattern shared by all diets
- `s(Time,Diet)`: how each diet deviates from that common pattern (4 facets, one per diet)
- `s(Time,Chick)`: individual chick deviations (many facets, one per chick)
- Each plot has its own y-axis scale (auto-scaled for that term)



Visualizing All Smooth Components II

```r
# With fixed scales
draw(m_cw_gam, scales = 'fixed')
```

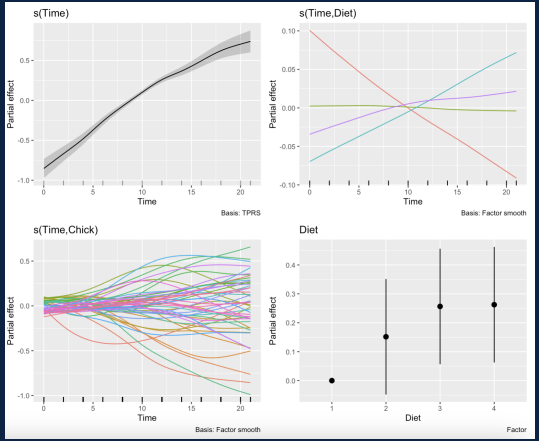
- Same as previous, but now all plots share the same y-axis scale
- Lets you compare the magnitude of effects across smooths
- You'll see if diet-level variation is bigger/smaller than chick-level variation



Visualizing All Smooth Components III

```r
# Include parametric terms
draw(m_cw_gam, parametric = TRUE)
```

- Shows smooths (like first) plus the parametric effects
- Adds a plot for Diet showing the intercept differences between diets
- Helpful to see both the smooth and non-smooth parts of your model together



Why Are Intercepts So Different?

```
# Why are estimated intercepts so different?
coef(m_cw_lm)['(Intercept)']         # mean weight for diet 1 at t = 0 is 31.50807
exp(coef(m_cw_glm)['(Intercept)'])   # mean weight for diet 1 at t = 0 is 43.72437
exp(coef(m_cw_gam)['(Intercept)'])   # mean weight at t = 21/2 is 82.65286
```

Explanation

- **LM & GLM:** Intercept = mean at Time = 0 (birth weight)
- **GAM with smooths:** mgcv centers smooths by default
- The smooth `s(Time)` is centered, so intercept represents the average weight at mean time ($\approx$ 10.5 days)
- This is for numerical stability and interpretation

The point here is to not compare intercepts directly across different smooth specifications

Making Predictions from the GAM

```r
# Create new data for predictions
newd <- expand_grid(
  Time = seq(0, 21, length.out = 400),
  Diet = unique(ChickWeight$Diet),  # fixed effects
  Chick = 'new chick')              # random effects

# Predict from GAM (must exclude random chick effect for new chick)
# NOTATION BELOW
# predict(object = m_cw_gam, newdata = # # newd, type = 'link',
#         se.fit = TRUE,
#         discrete = FALSE,                    # required for bam() predictions
#         exclude = c('s(Time,Chick)')) %>%    # exclude chick-level deviations
#  as.data.frame() %>%
#  rename(gam_fit = fit, gam_se = se.fit) %>%
#  mutate(gam_est = exp(gam_fit),
#          gam_lwr95 = exp(gam_fit - 1.96 * gam_se),  # CI on link scale
#          gam_upr95 = exp(gam_fit + 1.96 * gam_se))
```

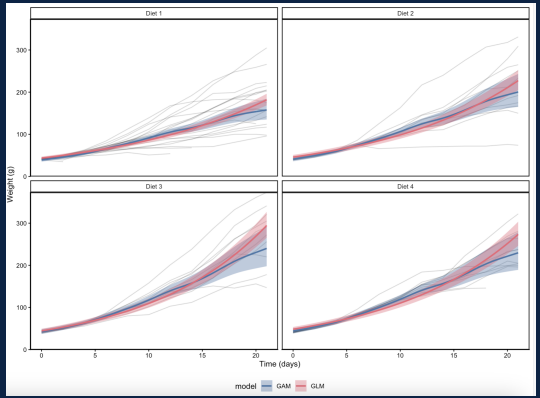
When using `discrete = TRUE` in `bam()`, must set `discrete = FALSE` in `predict()` and exclude random smooths for new levels

Comparing LM, GLM, and GAM Predictions

```r
preds <-
  bind_cols(
    newd,
    # LM predictions
    predict(object = m_cw_lm, newdata = newd, type = 'response', se.fit = TRUE) %>%
      as.data.frame() %>%
      rename(lm_fit = fit, lm_se = se.fit) %>%
      mutate(lm_est = lm_fit, lm_lwr95 = lm_fit - 1.96 * lm_se, lm_upr95 = lm_fit + 1.96 * lm_se),
    # Gamma GLM predictions
    predict(object = m_cw_glm, newdata = newd, type = 'link', se.fit = TRUE) %>%
      as.data.frame() %>%
      rename(glm_fit = fit, glm_se = se.fit) %>%
      mutate(glm_est = exp(glm_fit), glm_lwr95 = exp(glm_fit - 1.96 * glm_se), glm_upr95 = exp(glm_fit + 1.96 * glm_se)),
    # Gamma GAM predictions
    predict(object = m_cw_gam, newdata = newd, type = 'link',
            se.fit = TRUE, discrete = FALSE, exclude = c('s(Time,Chick)')) %>%
      as.data.frame() %>%
      rename(gam_fit = fit, gam_se = se.fit) %>%
      mutate(gam_est = exp(gam_fit), gam_lwr95 = exp(gam_fit - 1.96 * gam_se), gam_upr95 = exp(gam_fit + 1.96 * gam_se))) %>%
  pivot_longer(cols = c(lm_fit, lm_se, lm_est, lm_lwr95, lm_upr95,
                        glm_fit, glm_se, glm_est, glm_lwr95, glm_upr95,
                        gam_fit, gam_se, gam_est, gam_lwr95, gam_upr95),
               names_to = c('model', 'parameter'), names_sep = '_') %>%
  pivot_wider(names_from = parameter, values_from = value) %>%
  mutate(model = toupper(model))
```

Visualizing All Three Models

```
# remove [preds$model %in% c("GLM", "GAM"),] to see
# LM predictions as well
ggplot(preds[preds$model %in% c("GLM", "GAM"),]) +
  facet_wrap(~ paste('Diet', Diet)) +
  geom_line(aes(Time, weight,
                group = Chick),
            ChickWeight, alpha = 0.2) +
  geom_ribbon(aes(Time,
                  ymin = lwr95,
                  ymax = upr95,
                  fill = model),
              alpha = 0.4) +
  geom_line(aes(Time, est,
                color = model),
            lwd = 1) +
  geom_hline(yintercept = 0,
             color = 'grey') +
  xlab('Time (days)') +
  scale_y_continuous('Weight (g)',
                     expand = c(0, 0)) +
  khroma::scale_color_bright() +
  khroma::scale_fill_bright() +
  theme_classic() +
  theme(legend.position = 'bottom',
        panel.grid = element_blank(),
        panel.border = element_rect(NA, "black", 1))
```



Key Differences Across Models

LM

- Linear growth
- Can predict negative
- Constant variance
- Symmetric CIs
- Simplest

GLM (for gamma)

- Exponential growth
- Always positive
- Variance $\propto$ mean
- Asymmetric CIs
- Better fit

GAM (for gamma)

- Flexible growth
- Always positive
- Variance $\propto$ mean
- Asymmetric CIs
- Best fit (of the three)

Notice that GAM captures the nonlinear growth patterns (slower at extremes) that GLM misses!

When Should You Use GAMs?

Use GAMs when...

You should consider GAMs if:

- Relationships are clearly nonlinear
- You don't want to assume a specific functional form
- Exploratory data analysis shows curvature
- You have enough data
- Interpretability of exact relationship isn't critical

Stick with GLMs when...

GLMs might be better if:

- Linear relationship is reasonable
- You need simple interpretation
- Sample size is small
- You want to test specific hypotheses about slopes

Interpreting GAMs: A Little Trickier

Challenges with Interpretation

GLM coefficients:

- Single number per variable
- Direct interpretation: "1 unit increase multiplies mean by $\exp(\beta)$"

GAM smooths:

- Entire *function* per variable
- No single coefficient to report
- Must visualize or describe the shape
- Effect differs across the range of x

The trade-off is that GAMs sacrifice simplicity of interpretation for flexibility and better fit

Checking if Smooths are Adequate

```
# Check if k (basis dimension) is large enough
gam.check(m_cw_gam)

# 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(Time)          9.00   8.40    0.97    0.24
# s(Time,Diet)    40.00   3.23    0.97    0.18
# s(Time,Chick) 500.00 281.39    0.97    0.22
```

What to look for:

- **k-index:** Should be close to `1`
- **p-value:** High p-value is good (smooth has enough flexibility)
- **edf vs k':** If edf (effective degrees of freedom) is close to k' (basis size), might need to increase k

If inadequate, try to re-fit with larger `k`, e.g., `s(Time, k = 15)`

Concurvity: GAM's Version of Collinearity

What is Concurvity?

Collinearity: Linear dependence between predictors

Concurvity: Nonlinear dependence between smooth terms

Problems it causes:

- Unstable coefficient estimates
- Wide confidence intervals
- Difficulty attributing effects to specific smooths

```
# Check concurvity
concurvity(m_cw_gam, full = TRUE)
```

Values close to 1 indicate high concurvity — consider simplifying model

Common Types of Smooths in mgcv

Smooth Type	When to Use
`s(x)`	Default — thin plate regression spline
`s(x, bs = 'cr')`	Cubic regression spline (faster for 1D)
`s(x, bs = 'cc')`	Cyclic cubic spline (for periodic data)
`s(x, bs = 're')`	Random effect (like in lme4)
`s(x, y)`	2D smooth (spatial or interaction surfaces)
`s(x, by = g)`	Separate smooth for each level of g
`s(x, g, bs = 'fs')`	Factor smooth (shared penalty)
`ti(x, y)`	Tensor product interaction (no main effects)

You Try: Fitting GAMs

Exercise: Explore smooths

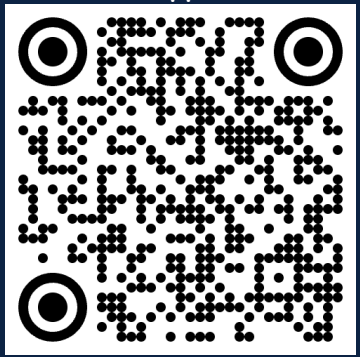
```r
# Option 1: Motorcycle crash data (classic GAM example)
library(MASS)
?mcycle  # head acceleration vs time after impact

# Fit GAM with smooth of time
m1 <- gam(accel ~ s(times), data = mcycle, family = gaussian())
# Try: plot(m1), appraise(m1), draw(m1)

# Option 2: Wage data (non-monotonic age effects)
library(ISLR)
?Wage  # wage vs age, year, education

# Compare linear vs smooth effect of age
m2a <- gam(wage ~ age, data = Wage, family = Gamma(link = 'log'))
m2b <- gam(wage ~ s(age), data = Wage, family = Gamma(link = 'log'))
# Compare with AIC(m2a, m2b)

# try the following:
# 1. Fit both a GLM and GAM
# 2. Use draw() and visualize smooths
# 3. Use gam.check() and verify smooth adequacy
# 4. Compare predictions, make a decision?
```

Key Takeaways

- GAMs extend GLMs by replacing linear terms with smooth functions
- Use `s(x)` for smooth terms, `s(x, bs = 'fs')` for factor smooths
- Smooths are automatically penalized to prevent overfitting
- `bam()` with `method = 'fREML'` for large datasets
- Always check smooth adequacy with `gam.check()`
- Hierarchical GAMs model common patterns + group deviations
- GAMs trade interpretability for flexibility — use when relationships are clearly nonlinear

What's Next?

Additional Questions?
Book an Appointment!



Additional Resources:
Generalized Additive Models: An Introduction with R (Wood, 2017)

Next Workshop:

Interpreting & Predicting from GAMs
– more on predicting
– revisiting model fit metrics (ex: AIC, BIC, RMSE, etc.)

Thank You!

Questions?

Workshop Materials:
`https://github.com/csc-ubc-okanagan/ubco-csc-modeling-workshop`

Contact:
jesse.ghashti@ubc.ca