# Nonlinear Modelling in R with GAMS

*Miao Cai*

*1/9/2019*

# 1 Introduction to Generalized Additive Models

Trade-offs in Model Building

- Linear models
- GAMs
- Black-Box ML

```r
library(mgcv)
gamfit = gam(y ~ s(x), data = dat)
```

The flexible smooths in GAM are constructed of many smaller functions, which are called basis functions. Each smooth is a sum of a number of basis functions and each basis function is multiplied by a coefficient, each of which is a parameter in the model

## 1.1 Basis Functions and Smoothing

The flexibility of GAM makes it easy to overfit the data.

- Close to the data (avoiding under-fitting)
- Not fitting the noise (avoid overfitting)

### 1.1.1 Smoothing parameter

The complexity of likelihood, or how much the curve changes its shape is meausred by **wiggliness**

$$\text{Fit} = \text{Likelihood} - \lambda * \text{Wiggliness}$$

The key is balance the trade-off between Likelihood and Wiggliness. The smoothing parameter $\lambda$ is used to control the balance.

Normally we let the package chooses the smoothing parameter.

```r
# Setting a fixed smoothing parameter
gam(y ~ s(x), data = dat, sp = 0.1)
gam(y ~ s(x, sp = 0.1), data = dat)

# Smoothing via restricted maximum likelihood
gam(y ~ s(x), data = dat, method = "REML")
```

`REML` is recommended as the smoothing algorithm. It is most likely to give stable and reliable results.

### 1.1.2 Number of basis functions

Set the number of basis functions

```r
gam(y ~ s(x, k = 3), data = dat, method = "REML")
gam(y ~ s(x, k = 10), data = dat, method = "REML")
```

Use the defaults

```r
gam(y ~ s(x), data = dat, method = "REML")
```

We can test if the number of basis functions is adequate using statistical tests.

### 1.2 Multivariate GAMs

- The `mgcv` package does not use character variables
- Normally we give splines to continuous variables, while keep categorical variables as they are in GAMs.
- Set `by` parameter, we can specify different smoothing in different categories.

```r
model4b <- gam(hw.mpg ~ s(weight, by = fuel) + fuel, data = mpg,
               method = "REML")
```

## 2 Interpreting GAMs

A good way to interpret significant smooth terms in GAMs is: **A significant smooth term is the one where you cannot draw a horizontal line through the 95% confidence interval.**

Note that a high EDF (effective degrees of freedom) doesn't mean significance or vice versa. In the example model, the price term is non-linear but non-significant, meaning that it has some complexity, but there isn't certainty to the shape or direction of its effect.

The plots generated by `mgcv`'s `plot()` function are partial effect plots. That is, they show the component effect of each of the smooth or linear terms in the model, which add up to the overall prediction.

```r
plot(gam_model, select = c(2, 3))
plot(gam_model, pages = 1)
plot(gam_model, pages = 1, all.terms = TRUE)
```

The first option we have when making our plots is which partial effect to show.

- The `select` argument chooses which terms we plot, with the default being all of them.
- Normally, each plot gets its own page, but using the pages argument, you can decide how many total pages to spread plots across.
- Finally, by default we only see the smooth plots, but by setting `all.terms = TRUE`, we can display partial effects of linear or categorical terms as well.

```r
plot(gam_model, rug = TRUE)
plot(gam_model, residuals = TRUE)
plot(gam_model, rug = TRUE, residuals = TRUE,
     pch = 1, cex = 1)
```

We often want to show data alongside model predictions.

- The `rug` argument puts X-values along the bottom of the plot.
- The residuals argument puts partial residuals on the plot.

Partial residuals are the difference between the partial effect and the data, after all other partial effects have been accounted for.

## 2.1 Showing standard errors

```r
plot(gam_model, se = TRUE)
plot(gam_model, shade = TRUE)
```

By default, plot will put standard errors on your plots. These show the 95% confidence interval for the mean shape of the effect. It is often preferable to use shading rather than lines to show these intervals, which can be achieved using the `shade = TRUE` argument.

## 2.2 Transforming standard errors

```r
plot(gam_model, seWithMean = TRUE)
plot(gam_model, seWithMean = TRUE, shift = coef(gam_model)[1])
```

It is often useful to plot the standard errors of a partial effect term combined with the standard errors of the model intercept. This is because the confidence intervals at the mean value of a variable can be very tiny, and don't reflect the overall uncertainty in our model. Using the `seWithMean` argument adds in this uncertainty.

To make the plot even more interpretable, it is useful to shift the scale so that the intercept is included. Using the `shift` argument, we can shift the scale by value of the intercept, which is the first coefficient of the model. Now, the partial effect plot has a more natural interpretation: it shows us the prediction of the output, assuming other variables are at their average values.

## 2.3 Model checking