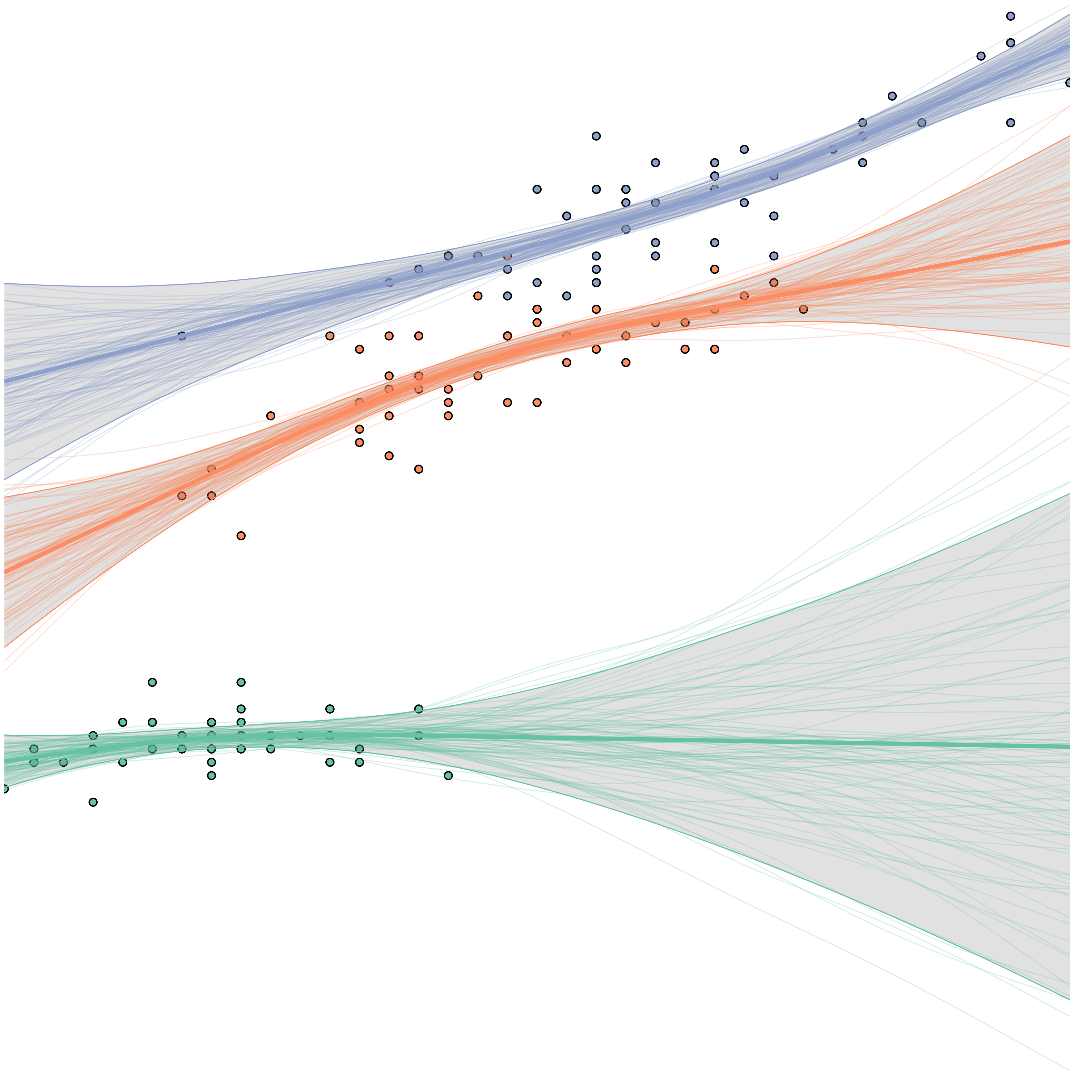
Why use GAMs for (G)LMs?

Asger Svenning



Introduction

In this short example I hope to convince you that `mgcv` and the `gam` function is useful for more than just GAMs, but can - and indeed should - often be used in the case where the relationship under interest is linear (i.e. where a LM would usually be used).

I will show an example of how an unknown non-linear confounding variable can easily be accounted for using a GAM, while either including it or not in a standard LM will lead to biased (incorrect) estimates - which in the worst case can lead to opposite conclusions.

Sampling a smooth confounding curve

To illustrate the problem, we will first need to define a way to generate a non-linear function. Later we will sample a new random non-linear function for each experiment, which is then used as the relationship between the confounding variable and the response.

To do this, we generate N evenly spaced control points, c, along the z-axis, and then sample N random values from a uniform distribution to generate the s-values. The position of the control points on the z-axis is then subtracted from the s-values to give a slight overall negative trend on average:

$$\begin{aligned} c_z &\sim \mathcal{U}(z_{\text{min}}, z_{\text{max}}), & z_{\text{min}} &= -0.1, & z_{\text{max}} &= 1.1 \\ c_s &\sim \mathcal{U}(s_{\text{min}}, s_{\text{max}}) - z, & s_{\text{min}} &= -1, & s_{\text{max}} &= 1 \end{aligned}$$

This is done to shift the distribution of confounding functions in such a way that the confounding effect is more likely to be opposite the true effect of x on y, which for this experiment will be positive one-to-one relationship.

We then use the `splinefun` function to generate a function, f_s, that smoothly interpolates between the control points:

$$f_s(z) = \text{splinefun}(c_z, c_s, \text{method} = \text{"natural"})$$

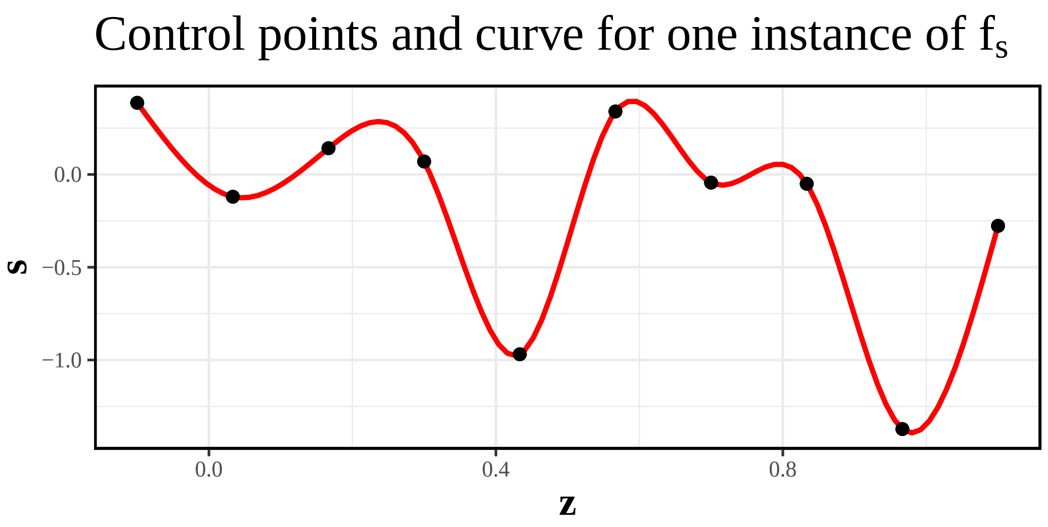
```r
get_sfun <- function(
    zlim = c(-0.1, 1.1),
    slim = c(-1, 1),
    ctrl.pts=10,
    plot=F
  ) {
  z <- seq(zlim[1], zlim[2], length.out = ctrl.pts)
  c <- runif(ctrl.pts, min = slim[1], max = slim[2]) - z

  sfun <- splinefun(z, c, method = "natural")
  if (plot) plot_smooth(sfun, z, c, zlim)

  return(invisible(sfun))
}
```

Example smooth confounding curve

The following plot shows an example of a smooth confounding curve generated using the `get_sfun` function.



The experiment

We will now define a simple experiment where we generate a dataset with three variables: x, z, and y. Both x and z are generated as a linear function of a latent variable l, with a small amount of uniform noise added. This also means that x and z are highly collinear (in this case the correlation coefficient $\rho \approx 0.95$), a case where common wisdom is to exclude one of the variables from the model to avoid multicollinearity. However, as we will see later, in the case of confounding variables excluding the variable can lead to biased estimates, and thus incorrect conclusions.

The response variable y is then generated as a linear function of x, a randomly sampled non-linear function of $f_s(z)$, and some normally distributed noise:

$$y = x + 3 \cdot f_s(z) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1/8)$$

The specific form of the predictor-response relationship was chosen to ensure that the effect of the confounding variable, z, is stronger than the effect of the predictor of interest, x, while the noise is kept small enough that residual patterns in the diagnostic plots are very clear.

We then fit three models to the data:

- A GAM model with y as a function of x and a smooth term for z: `gam(y ~ x + s(z))`.
- A linear model with y as a function of x and z: `lm(y ~ x + z)`.
- A linear model with y as a function of x only: `lm(y ~ x)`.

The `experiment` function below generates the data, fits the models, and returns the estimated coefficient for x and the standard error for each model. We also calculate the error (difference between the estimated coefficient and the true value of 1) and the z_{error}-score, $\tilde{\mu_x}-\mu_x/\widetilde{\sigma(\mu_x)}$, for each model.

Note: It is very important to take notice of the fact that f_s is a random non-linear function, and that for each iteration of the experiment a new f_s is sampled. This is done to ensure that the results hold in general for non-linear confounding variables, and not just for a specific shape of f_s.

```r
experiment <- function(plot=F) {
  data <- tibble(
    # Sample the latent variable
    l = runif(500),
    # Generate x and z as linear functions of l with a small amount of
    # independent uniform noise to ensure they are highly, but not
    # perfectly, collinear
    x = l + runif(500, -1, 1) / 10,
    z = l + runif(500, -1, 1) / 10,
    # Generate y as a linear function of x and a sampled non-linear function
    # of z, with some normally distributed noise
    y = x + 5 * get_sfun()(z) + rnorm(500, 1/8)
  )

  if (plot) dp <- plot_data(data)

  # Fit the three models
  gam_mod <- bam(y ~ x + s(z), data = data, method = "fREML", discrete=T)
  lin_mod1 <- lm(y ~ x + z, data = data)
  lin_mod2 <- lm(y ~ x, data = data)

  if (plot) plot_models(gam_mod, lin_mod1, lin_mod2, dp)

  # Extract and summarize the results for each model
  res <- tibble(
    type = c("gam", "lin1", "lin2"),
    model = list(gam_mod, lin_mod1, lin_mod2),
    result = map(model, ~ broom::tidy(.x, parametric=T))
  ) %>%
    select(!model) %>%
    unnest(result) %>%
    select(type, term, estimate, std.error) %>%
    filter(term == "x") %>%
    mutate(
      error = estimate - 1,
      z.error = error / std.error
    )

  return(invisible(res))
}
```

Example diagnostics

Before we dive into the results, let's look at the diagnostics for the three models fitted to a particular instantiation of the data.

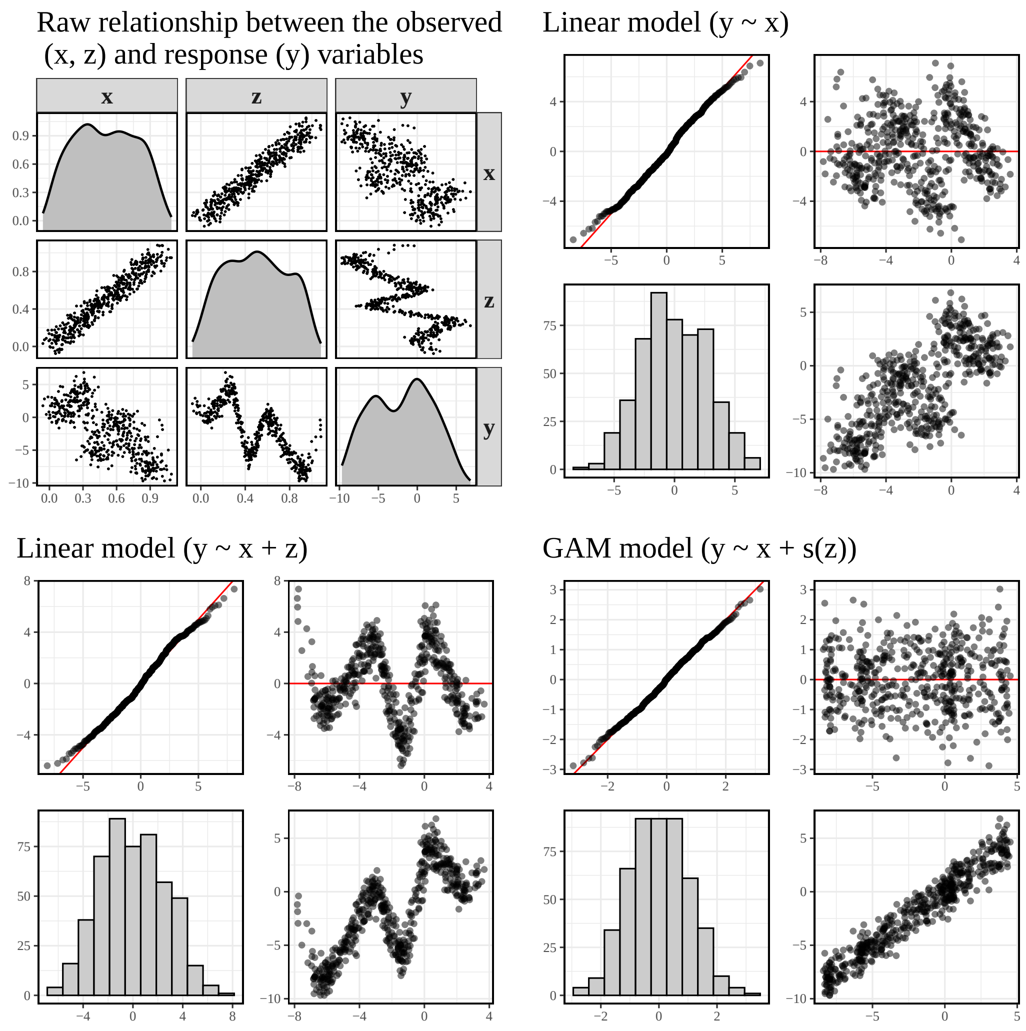


Figure 1: (TR, BL & BR) Diagnostic plots for the three models with sub-panels; QQ-plot (TL), residuals-fitted/linear predictor (TR), residual histogram (BL) and observed-fitted (BR). (TL) Pairwise scatter plot and histograms for x, y and z.

As can be clearly be seen by inspecting the diagnostic plots, the GAM model captures the relationship between y and z well, leading to acceptable model diagnostics, whereas the linear models would have to be rejected based on the diagnostics. However, does this also result in incorrect estimates of the coefficient for x (μ_x)?

Results

To answer the question about whether the GAM model leads to better estimates of the coefficient for x, we will run the `experiment` function 2000 times and compare the results for the three models. Remember that for each iteration we sample a new smooth function f_s and generate new data, this is done to ensure that the results hold in general for non-linear confounding variables.

Summary of results

The table below shows the average estimated coefficient for x ($\overline{\tilde{\mu_x}}$), the mean absolute error ($\overline{|\tilde{\mu_x} - \mu_x|}$), the rate of confident opposite conclusions ($\overline{\text{sign}(\tilde{\mu_x}) \neq \text{sign}(\mu_x) \wedge |{}^{\tilde{\mu}_x}/_{\widetilde{\sigma(\mu_x)}}| > 1.96}$), and the rate of estimates within the confidence interval ($\overline{{}^{|\tilde{\mu_x} - \mu_x|}/_{\widetilde{\sigma(\mu_x)}} < 1.96}$) for the three models.

Model	Mean estimated slope	Mean absolute error	Rate of confident opposite conclusions	Rate of estimate within confidence interval
$y \sim x + s(z)$	1.00	0.48	0.00%	95.00%
$y \sim x + z$	0.94	1.63	3.85%	81.20%
$y \sim x$	-3.96	5.11	84.30%	5.45%

The three models also differ in the rate of rejecting the null hypothesis of $\mu_x = 0$, with the GAM model rejecting the null hypothesis in 41% of the cases, the linear model with $y \sim x + z$ rejecting the null hypothesis in 24% of the cases, and the linear model with $y \sim x$ rejecting the null hypothesis in 92% of the cases.

Expected versus observed error distribution

The following plot shows the distribution of the z_{error}-score, ${}^{\tilde{\mu_x} - \mu_x}/_{\widetilde{\sigma(\mu_x)}}$, for the three models. The blue line shows the expected distribution of the z_{error}-score under the hypothesis of $\mu_x = 1$, i.e. a standard normal distribution. The green dashed line shows the median z_{error}-score for each model, and the red line shows the expected value of the z_{error}-score (0).

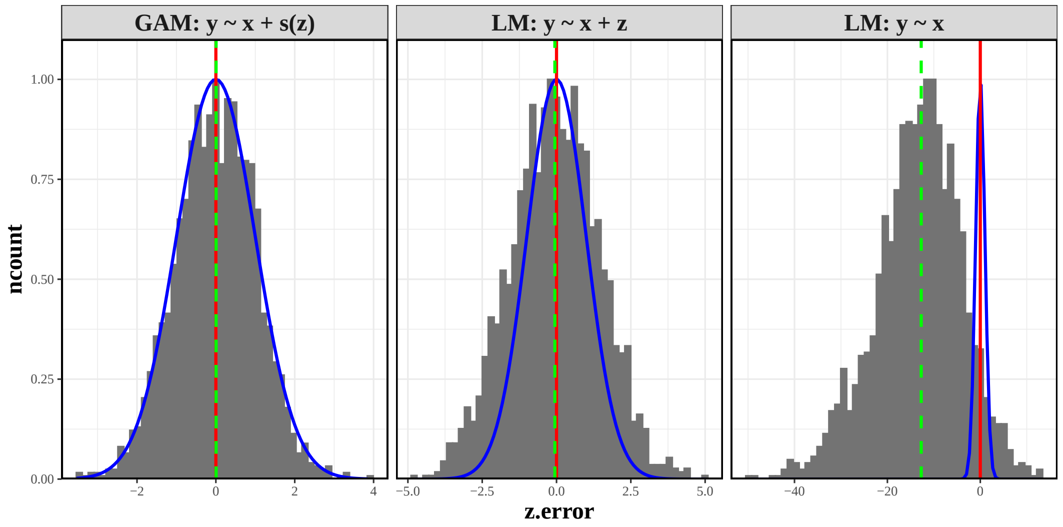


Figure 2: Distribution of the z_error-score, $\tilde{\mu_x} - \mu_x / \widetilde{\sigma(\mu_x)}$, for the three models, with the expected distribution under the null hypothesis of $\mu_x = 1$ (blue), the median z_error-score (green dashed) and the expected value of the z_error-score (red).

Conclusion

As clearly demonstrated the GAM model accurately accounts for the non-linear confounding variable, leading to unbiased estimates of the coefficient for x and accurate confidence intervals. The purely linear models, on the other hand, fail to account for the confounding variable, leading to incorrect confidence intervals in the average case, and in the worst case, opposite conclusions. It also demonstrates how properly modelling confounding variables can lead to higher statistical power, even though the model is using more degrees of freedom.

This simple example illustrates how GAMs can be used to account for non-linear confounding variables, even when the relationship of interest is linear. This is especially important in observational studies where the relationship between the confounding variable and the response is unknown, and thus cannot be accounted for using a linear model.