Strategies for `ggplot2::geom_smooth` when outcome domain is not the real line

Holger Sennhenn-Reulen
Northwest German Forest Research Institute (NW-FVA), Germany.

August 22, 2024

This short manuscript gives pragmatic strategies for how to use `ggplot2::geom_smooth` when the outcome domain is not the real line.

Contents

1 Software

1.1 General

We use the statistical software environment *R* (R Core Team, 2024), and R add-on package *ggplot2* (Wickham, 2016) for graphical visualizations.

This document is produced using *Quarto* (Allaire et al., 2024).

2 Organize R Session

```
rm(list = ls())
library("ggplot2")
```

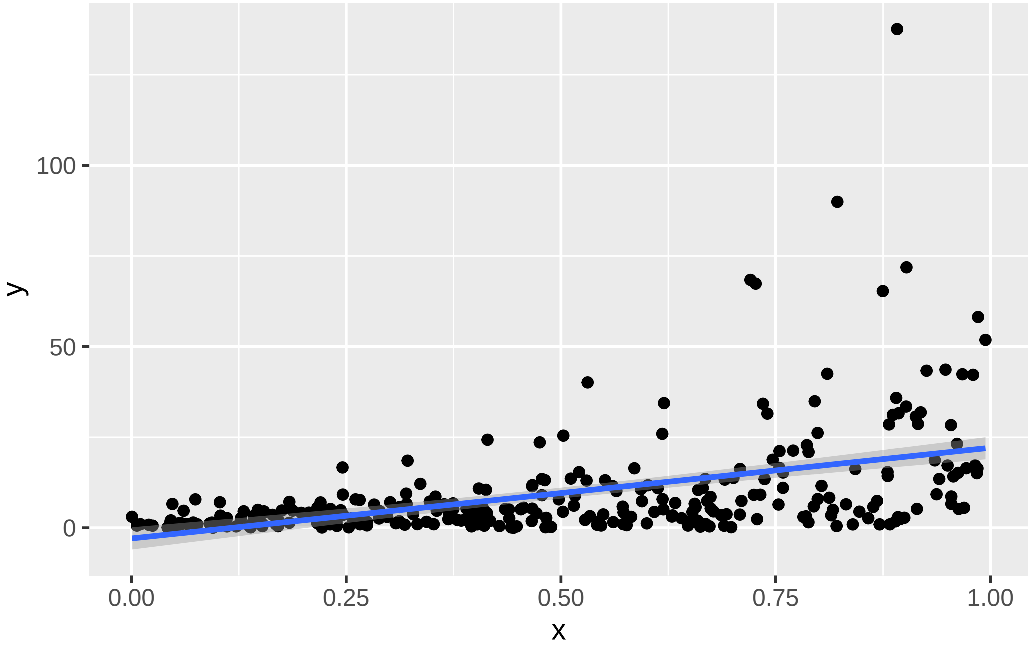
3 Simulation build-up

```
N <- 300
set.seed(123)
x <- runif(N)
y <- rgamma(n = N, shape = 1, scale = exp(.5 + 3*x))
df <- data.frame(x = x, y = y)
```

4 Further R Code

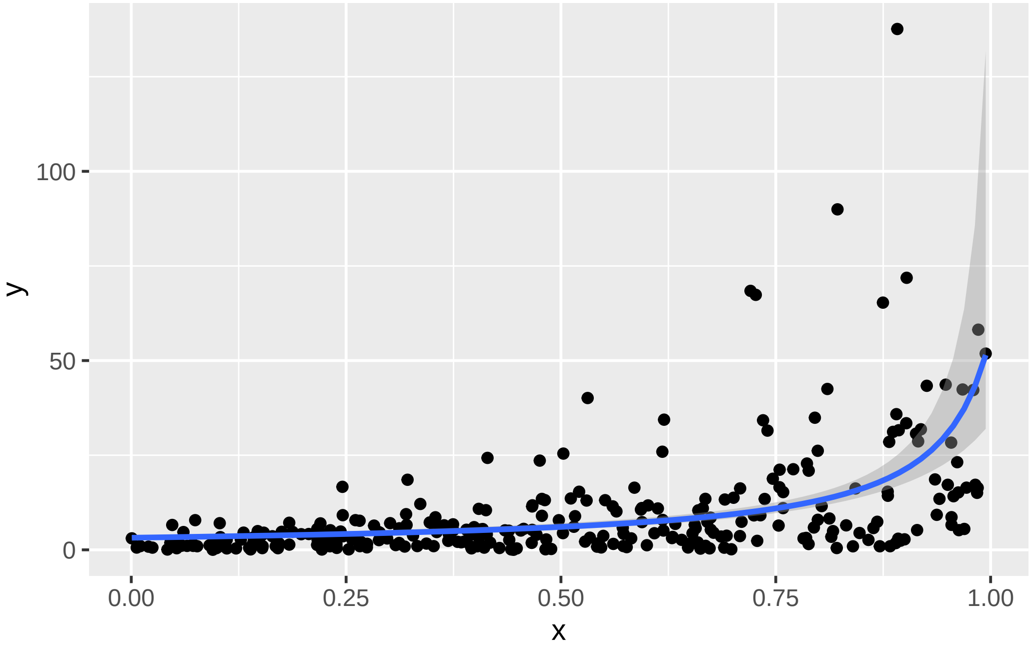
Here, due to the simulation setup, we also get negative values for the conditional expected value if `method = "lm"`:

```
ggplot(data = df, aes(x = x, y = y)) +
  geom_point()+
  geom_smooth(formula = y ~ x, method = "lm")
```



Gamma-GLM for only positive values:

```
ggplot(data = df, aes(x = x, y = y)) +
  geom_point()+
  geom_smooth(formula = y ~ x, method = "gam",
              method.args = list(family = "Gamma"))
```

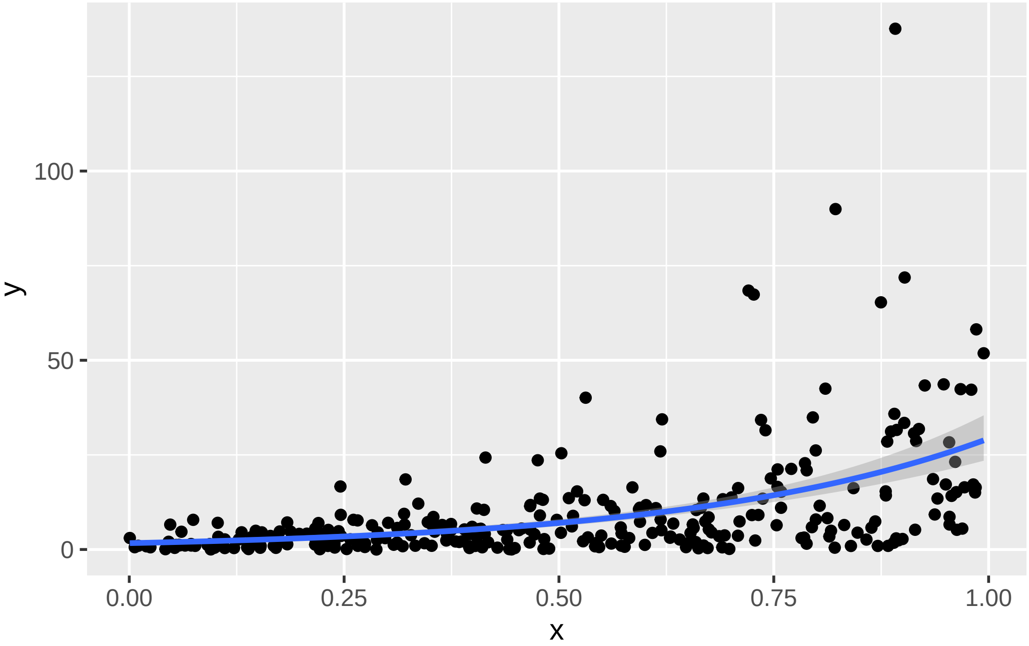


Gamma-GLM no longer works with a value exactly equal to 0:

```
df$y[1] <- 0
```

Alternative: Tweedie-GAM if at least one value is exactly equal to 0:

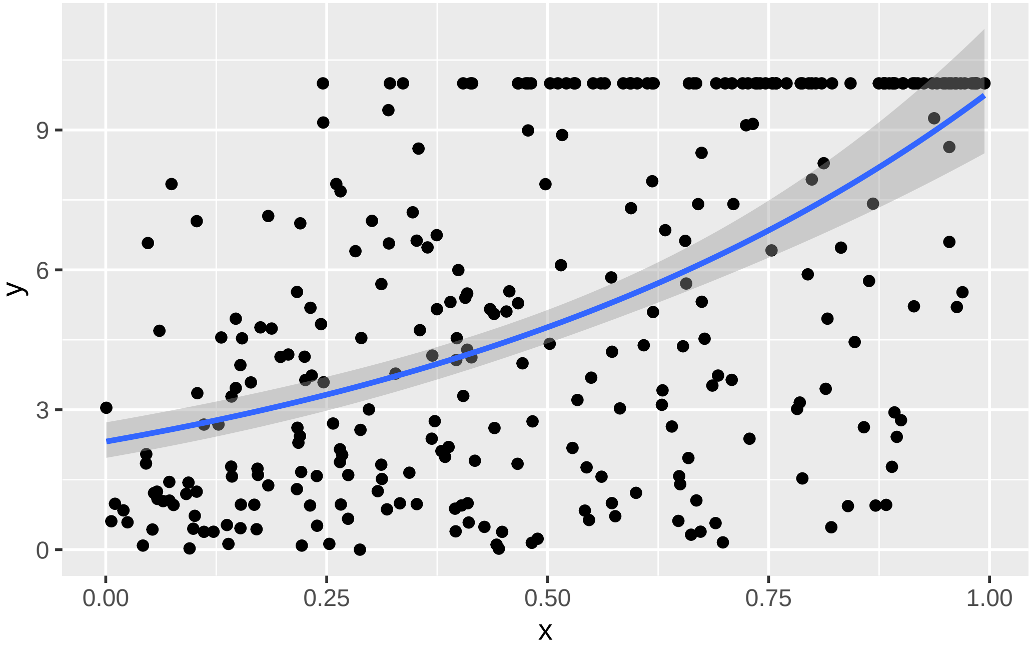
```
ggplot(data = df, aes(x = x, y = y)) +
  geom_point()+
  geom_smooth(formula = y ~ x, method = "gam",
              method.args = list(family = "Tweedie(p=1.5)"))
```



… setting `p=1.5` is a bit rough, but maybe it's okay as a pragmatic solution for now?!

What to do if values between 0 and 10, with exactly equal to 0 and exactly equal to 10?

```
df$y[df$y > 10] <- 10
ggplot(data = df, aes(x = x, y = y)) +
  geom_point()+
  geom_smooth(formula = y ~ x, method = "gam",
              method.args = list(family = "Tweedie(p=1.5)"))
```

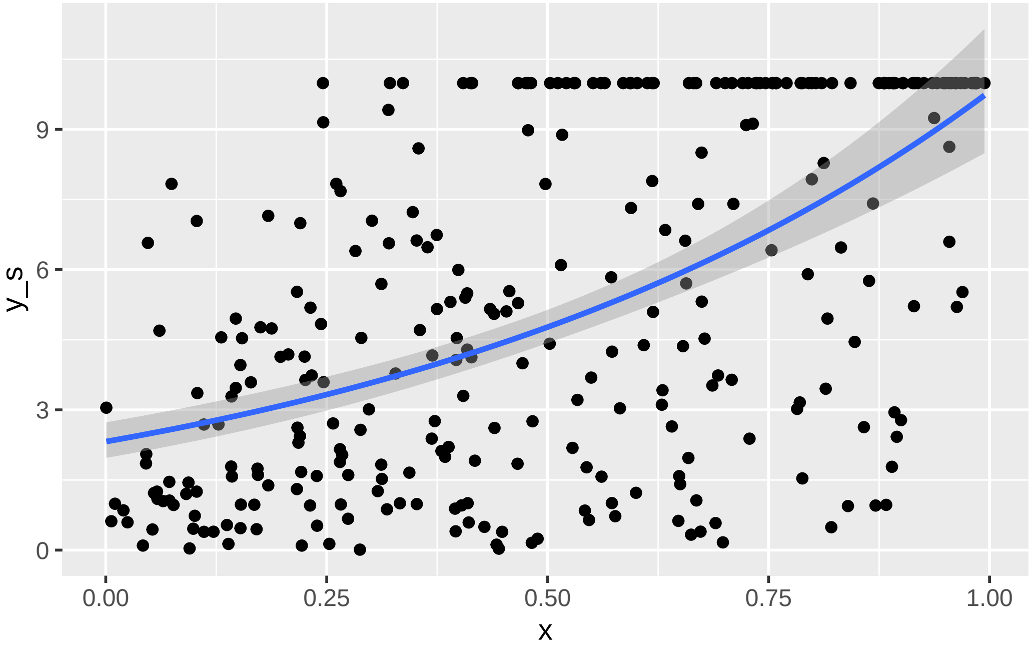


The uncertainty interval goes beyond 10.

Pragmatic strategy in three steps:

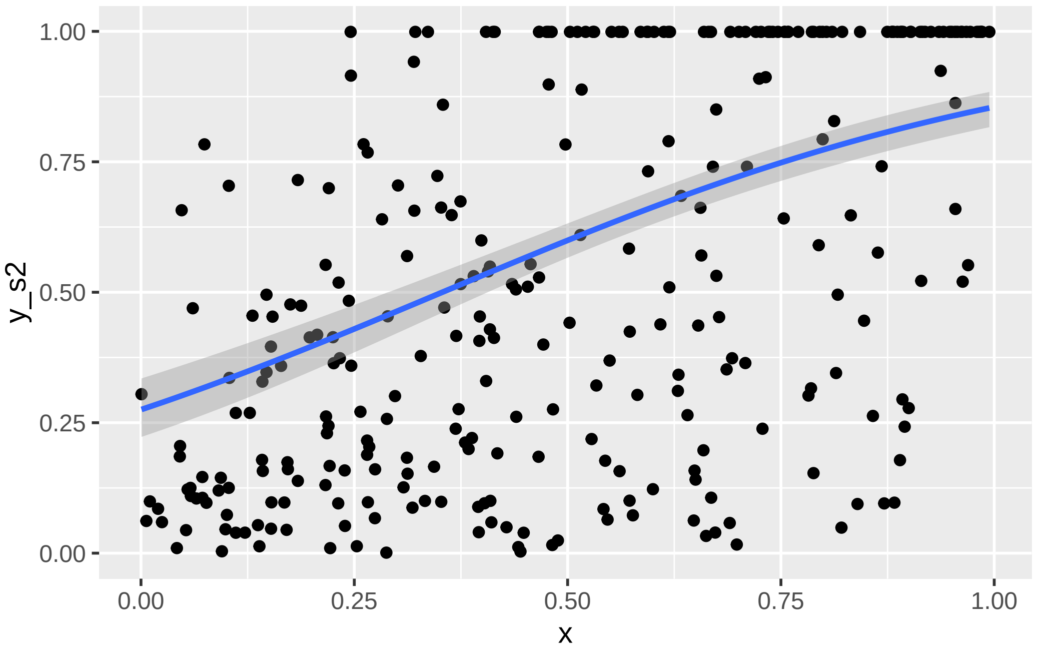
Part a: Transform values so that all valuesin the interval $[c, 10 - c]$, e.g. with `c = 0.01`:

```
df$y_s <- 0.01 + (10 - 2 * 0.01) * df$y / 10
ggplot(data = df, aes(x = x, y = y_s)) +
  geom_point()+
  geom_smooth(formula = y ~ x, method = "gam",
              method.args = list(family = "Tweedie(p=1.5)"))
```



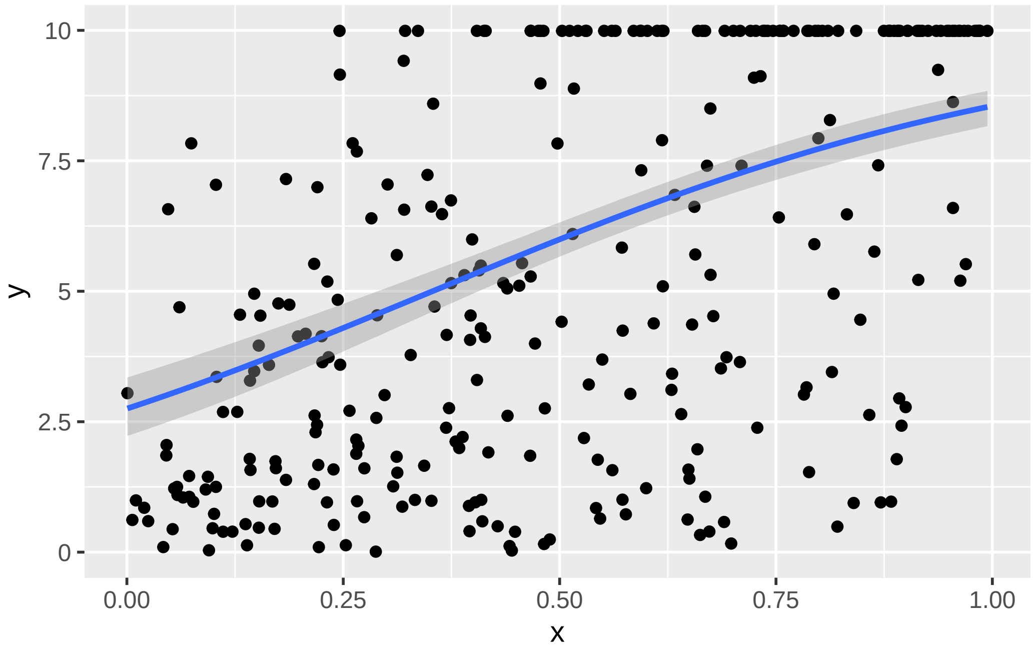
Part b: Now divide all values by 10 in order to be able to apply beta regression:

```
df$y_s2 <- df$y_s / 10
(p <- ggplot(data = df, aes(x = x, y = y_s2)) +
  geom_point()+
  geom_smooth(formula = y ~ x, method = "gam",
              method.args = list(family = "betar")))
```



Part c: rescale y-axis:

```
times_ten <- function(x){
  x * 10
  }
p +
  scale_y_continuous(labels = times_ten) +
  labs(y = "y")
```


10
7.5
5
2.5
0
y
0.00
0.25
0.50
0.75
1.00
x

References

Allaire, J. J., Teague, C., Scheidegger, C., Xie, Y., & Dervieux, C. (2024). *Quarto (Version 1.4.553)*. https://doi.org/10.5281/zenodo.5960048

R Core Team. (2024). *R: A Language and Environment for Statistical Computing (Version 4.4.1)*. R Foundation for Statistical Computing.

Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org