The two modelling strategies under discussion are

* to log-transform the y and use ordinary least squares; or
* a quasi-likelihood GLM with a log link function and variance proportionate to the mean of the response. (As discussed above, this is the purpose I want to use. Down the track I might want to instead use “to the square of the mean of the response” instead, but that’s not important for illustrative purposes.)

These strategies give similar coefficient estimates. Let’s look at some simulated data (held over from my aborted question to Cross-Validated)

library(tidyverse)

library(scales)

n <- 50

set.seed(123)

d <- tibble(x = rnorm(n, 4, 1)) %>%

mutate(mean = 2 + 0.25 \* x,

sd = sqrt(mean) / 20,

y = exp(mean + rnorm(n, 0, sd))

)

mod1 <- lm(log(y) ~ x, data = d)

mod2 <- glm(y ~ x, family = quasi(link = log, variance = mu), data = d)

# similar coefficients (one is minimising squares on the log scale,

# the other is minimising weighted squares on the original)

rbind(coef(mod1), coef(mod2))

Which gives us:

(Intercept) x

[1,] 2.021225 0.2478562

[2,] 2.016842 0.2496386

So just as a curve-fitting exercise, both methods are pretty good at recovering the original true values of 2 and 0.25 for the intercept and slope respectively in the linear predictor of my data generating process.

A prediction interval differs from a confidence interval in that it includes individual-level randomness *in addition to* the uncertainty in parameter estimates of the model. For example, geom\_smooth() draws a confidence interval, expressing confidence/uncertainty in the position and shape of the line of best fit, but not the range where 95% of new values might be expected to turn up.

**Prediction interval with transformation and ordinary least squares**

Creating a prediction interval from the log-transformed OLS model is a standard Regression 101 exercise. The variance of the estimate for any individual is the sum of the variance of our estimated curve at that point and the residual variance of individuals. Take the square root and you get the standard deviation; assume normality and multiply by 1.96 to get a 95% prediction interval; back transform to the original scale (note that there are bias-adjustment complications we could worry about here but will ignore for today’s purposes).

… generated with this code:

#------------Prediction interval from the log transform version--------------------

sx <- tibble(x = seq(from = 1, to = 7, length.out = 100))

pred1 <- predict(mod1, se.fit = TRUE, newdata = sx)

sx1 <- sx %>%

# calculate standard deviation for the prediction interval and create lower and upper bounds:

mutate(se\_pi = sqrt(pred1$se.fit ^ 2 + pred1$residual.scale ^ 2),

lower\_pi = exp(pred1$fit - 1.96 \* se\_pi),

upper\_pi =exp(pred1$fit + 1.96 \* se\_pi))

p1 <- ggplot(d, aes(x = x)) +

geom\_ribbon(data = sx1, aes(ymin= lower\_pi, ymax = upper\_pi), fill = "steelblue", alpha = 0.1) +

geom\_point(aes(y = y))+

labs(title = "Original data with prediction interval from log transform model")

**Prediction interval with GLM and a log link function**

To do something similar with the generalized linear model with the quasi likelihood response, we need to understand the residual.scale argument returned by predict.glm(). Whether we use predict(..., type = "response") or predict(..., type = "link"), the residual.scale value is the same (0.361 in our case). But 0.361 of what?

residual.scale is described in the help file as “a scalar giving the square root of the dispersion used in computing the standard errors”. How do we use this to make a prediction interval? In our specification of mod2, I told glm that the variance of y was going to be proportional to its mean (family = quasi(link = log, variance = mu)). We can recover the estimated variance at an individual point by the predicted value of y multiplied by the dispersion factor. So this means that in this particular case we can generate the prediction interval as follows. The key part for our discussion is + pred2$fit \* pred2$residual.scale ^ 2) – because pred2$fit is the mean of y at the given point, and pred2$residual\_scale ^ 2 is the dispersion factor that scales the mean to become its variance.

#-----------------prediction interval from the quasi family GLM------------

pred2 <- predict(mod2, se.fit = TRUE, newdata = sx, type = "response")

sx2 <- sx %>%

mutate(se\_pi = sqrt(pred2$se.fit ^ 2 + pred2$fit \* pred2$residual.scale ^ 2),

lower\_pi = pred2$fit - 1.96 \* se\_pi,

upper\_pi = pred2$fit + 1.96 \* se\_pi)

ggplot(d, aes(x = x)) +

geom\_ribbon(data = sx2, aes(ymin= lower\_pi, ymax = upper\_pi), fill = "steelblue", alpha = 0.1) +

geom\_point(aes(y = y))+

labs(title = "Original data with prediction interval from quasi likelihood glm with log link")

And here we go:

Seems simple once you know.

There are subtle but important differences between the prediction intervals from our two methods. Because I wrote the original data generating process, I know that the second one is closer to the “real” model. The log transform model actually increases the variance too quickly as the mean increases, compared to the true process. For higher values of y, the prediction interval is wider than it needs to be. Of course, whether this will be the case with real data will depend on the particular situation.

Note that this method won’t work for all GLMs. In particular, anything with a non-continuous response would definitely need another think. And to use those +/- 1.96 values for the prediction interval, I’ve relied on being prepared to make a simplifying assumption about the normal distribution of response variable for the purposes of the prediction interval, which I hadn’t made in the original estimation of the quasi-likelihood model. There’s no central limit theorem or anything to rely on here, because these are *individual* values of y, and they’ll have whatever shape y usually has.

What matters most here is understanding how that residual.scale disperson estimate works, in relationship to the specification of variance in the original call to quasi(). This relationship to the variance *doesn’t* depend on any additional normality assumption.

Anyway, use this approach with caution. Obviously, if I *really* were prepared to say that y is normally distributed (but with variance increasing proportional to the mean), I could model it explicitly as such rather than mucking around with quasi-likelihoods. So what’s outlined here is very much about pragmatism.