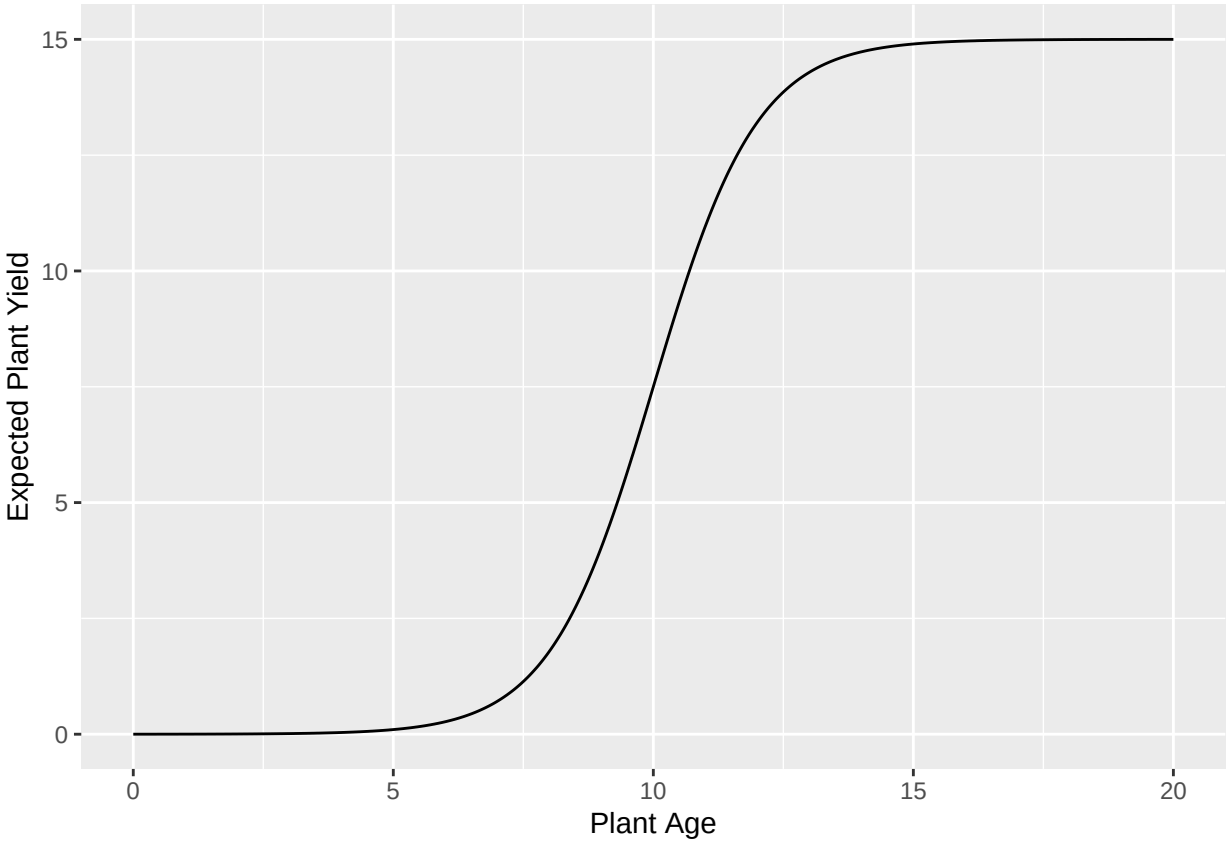
NLS Estimates for a Logistic Curve

A Bias Simulation Study

Katherine Pulham

Recently, a colleague reached out to me with a problem that prompted an interesting statistical question. She had collected yield data from some plants she was studying, and wanted to model the average plant yield using a logistic curve, which was a model suggested by domain knowledge in her field:



She had a few dozen observations, and for each observation she had the plant's age, a yield response variable, and several others. For the time being, she wanted to focus on the relationship between the plant's age and the yield, and to estimate the parameters of the logistic curve which best fit her data. She consulted a statistical consultant, who recommended she use the `nls()` and `SSlogis()` functions in R to fit the model. Hearing this proposal got me thinking.

Nonlinear Least Squares is a generalization of linear regression. For linear regression, we typically assume that the expectation of the response variable is a linear combination of the parameters which are to be estimated, and that the residuals are normally distributed:

$$E[y_i] = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} + \varepsilon_i$$

where the $\varepsilon_i \sim N(0, \sigma^2)$. In such a case, the ordinary least squares estimates $\hat{\beta} = (X^T X)^{-1} X^T Y$ will be the *Best Linear Unbiased Estimator*, or BLUE, for β. However, it is not even necessary to assume that the residuals are normally distributed, all that is really needed is that the residuals are symmetrically distributed. SEE Faraway (2015) for more details on OLS and Gauss-Markov.

The linearity assumption of ordinary least squares is extremely limiting. Often times we can work around this issue by transforming the explanatory or response variables. For instance, say we want the expected value of y to increase exponentially as a single explanatory variable increases. That is, we wish to model:

$$y = ce^{rx} + \varepsilon$$

Observe that by taking the logarithm of both sides, we get

$$\log(y) = \log(c) + rx + \log(\varepsilon)$$

Hence, if we let $\beta_0 = \log(c)$ and $\beta_1 = r$, we see that we can fit the desired model by transforming the response variable in the appropriate way, and performing a linear regression on the transformed variable. Note that if $\log(\varepsilon)$ is normally distributed, then ε will have a log-normal distribution, and the back-transformed mean will instead be a median response instead of a mean.

The reason the OLS estimates are called the ordinary *least squares* estimates is because they are chosen so that they minimize the squared distance from the predicted $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + ... + \hat{\beta}_p x_{ip} = \hat{\beta} x_i$ values and the observed y values. If we let Y be the vector of responses, and $\mathbf{X}$ be the design matrix, the matrix where each column is an explanatory variable and each row is an observation, then the OLS are chosen to minimize:

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \hat{\beta} x_i)^2 = ||Y - \hat{\beta}\mathbf{X}||^2$$

However, there are some functions for which there is no simple transformation. In the case of my colleague the plant scientist, the model we wish to fit is:

$$E[y] = \frac{Asym}{1 + e^{-r(x - x_0)}}$$

This is where nonlinear least squares comes in. If we assume that

$$y = \frac{Asym}{1 + e^{-r(x - x_0)}} + \varepsilon$$

where ε has a normal distribution with zero mean, then we can use a technique which is an alternative to ordinary least squares: Nonlinear Least Squares.

Nonlinear least squares assumes a more broad set of assumptions. In this case, we assume that

$$y = f(x, \theta) + \varepsilon$$

where ε is normally distributed with mean 0. We then wish to estimate the parameters θ. What's helpful about the concept of least squares is that we can still attempt to minimize the equation:

$$\sum_{i=1}^{n} (y_i - f(x_i, \hat{\theta}))^2 = ||Y - f(x, \hat{\theta})||^2$$

However, the implementation of this minimization is technical. It frequently involves using Newton-Rhapson optimization, based on a Taylor series approximation for $f(x, \theta)$ around an initial guess for the parameter vector. To read more about this, see Bates (1988), page 40 for more details.

However, there is an issue. If we were to consider my colleague's model, we would realize that for values of x for which the expected yield $E[y]$ is small, a symmetric distribution for the error term is unfeasible. To make a random variable with a very small mean have a symmetric distribution, we would need either a)

the random variable to have a very small, finite support, with $E[y]$ in the middle of that support, or b) the random variable to be able to take negative values. For our modeled variable, which represents the yield of plants, neither of these two conditions can hold, which means that the symmetric residual assumption of Nonlinear Least Squares is *not* met.

What I wanted to know is this: will this issue give my colleague a model with bias in it, and if so, how much bias? To answer this question, I decided to do a simulation study.

First, it was helpful to define a function for the model:

```
S <- function(x,asym,rate,xmid){
  return(asym/(1+exp(-rate*(x-xmid))))
}
```

To do this simulation study, I would need to pick a distribution for Y. Since the data was continuous, and positive, I decided to use a gamma model. Since I wanted $S(x) = \frac{Asym}{1+e^{-r(x-x_0)}}$ to be the mean of y, $y \sim gamma(k, theta)$ would imply

$$S(x) = k\theta$$

Here θ is taken to be the scale parameter of the gamma random variable. In the notation of the equation we wish to minimize, we would have $\hat{k}, \hat{\theta}$ be the parameters which minimize

$$\sum_{i=1}^{n}(y_i - f(x_i, \hat{\theta}, \hat{k}))^2 = ||Y - f(x, \hat{\theta}, \hat{k})||^2$$

Since we expect larger plants to have bigger variance in the yield they produce, we can add a dispersion parameter to allow the variance to vary along with $S(x)$. We assume $var(y) = \phi S(x)$ for some ϕ, and hence

$$\phi S(x) = k\theta^2$$

Solving these equations for k and θ, we get:

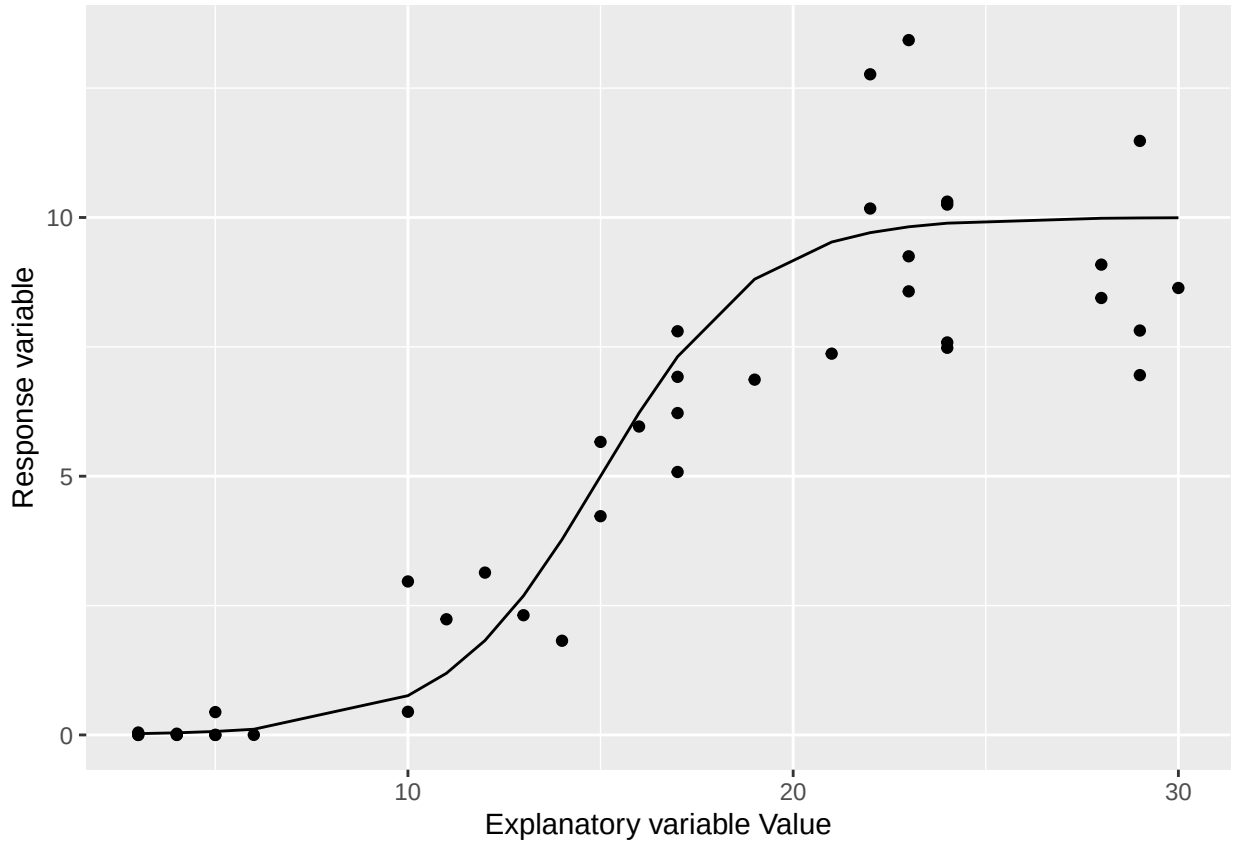
$$k = \frac{S(x)}{\phi}$$

$$\theta = \phi$$

And hence, $y \sim gamma(S(x)/\phi, \phi)$. We can simulate some data using this model to verify that it behaves as desired. Taking a random sampling of the explanatory variables:

```
set.seed(13)

#Pick values for parameters
phi   <- 1/2
tru_asym <- 10
tru_rate <- 1/2
tru_xmid <- 15
n     <- 40

#random sample explanatory variable
x <- sample(1:30,n,replace=TRUE)

#Logistic mean
Ey <- S(x,asym=tru_asym,rate=tru_rate,xmid=tru_xmid)

#simulate response
y <- numeric(length=length(x))
```

```
for(i in 1:n){
  y[i] <- rgamma(1,shape=Ey[i]/phi,rate=1/phi)
} #the rate parameter is the inverse, due to how R calculates the gamma pdf

#Plot
qplot(sort(x),sort(Ey),geom="line")+
  geom_point(aes(x,y))+
  labs(x="Explanatory variable Value",y="Response variable")+
  scale_linetype_manual('',values='solid')
```



Here, the line represents the expected response, and the points represent the simulated response. We can see then that the gamma model we've concocted is behaving as we would expect it to.

Observe that `nls()` can be used to estimate the Nonlinear Least Squares estimates for these parameters. Instead of manually setting guesses for the parameters, we can use the function `SSlogis()` to create a "self-starting" logistic function, and have R guess for us. This is simultaneously less of a hassle, and easier code to teach someone who is not a full-time R programmer:

```
nlsmod <- nls(y~SSlogis(x,Asym,xmid,scal),data=data.frame(x=x,y=y))
coef(nlsmod)
```

```
##      Asym      xmid      scal
##  9.485214 15.027731  2.373379
```

Ok, so we have guesses for the parameters. To get an idea of whether or not these guesses are biased, we will simulate 10,000 samples in this fashion, and check to see what the estimators are converging to. Note,

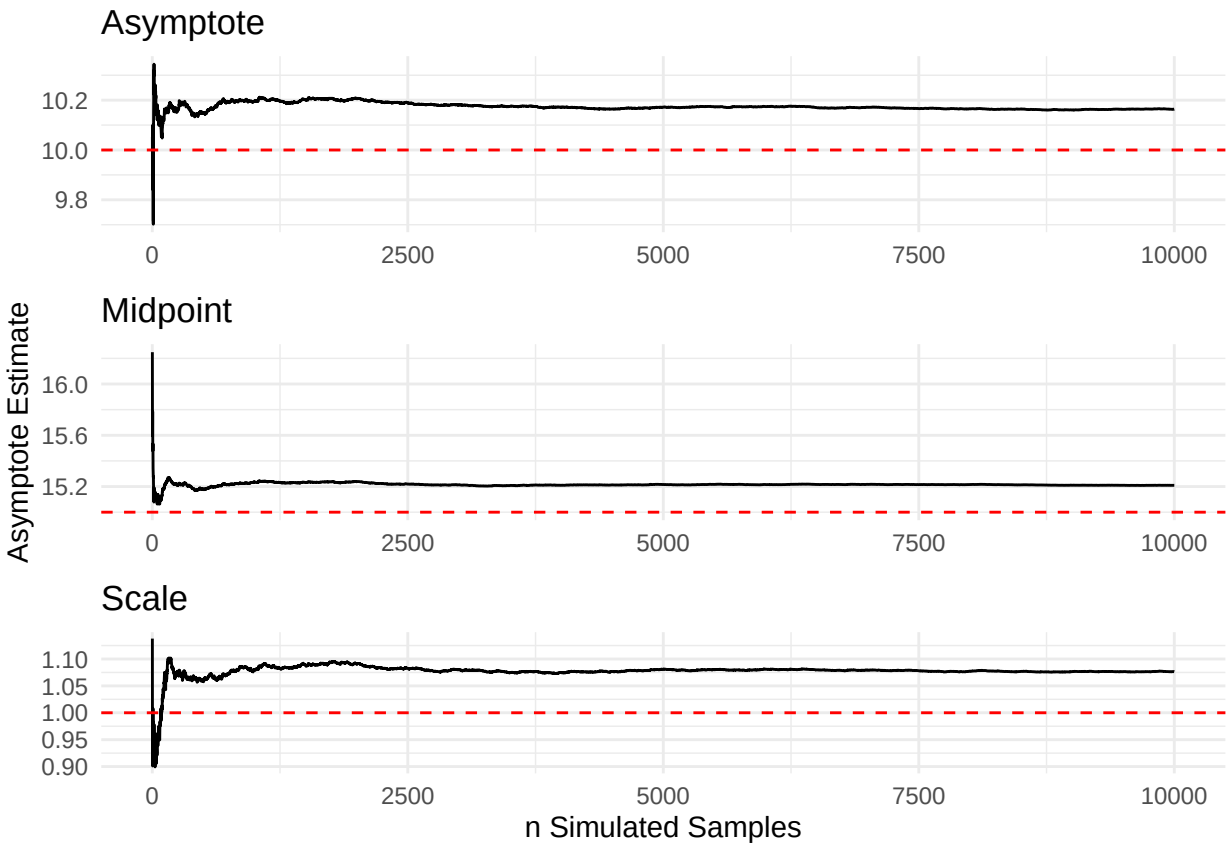
generating this many simulated samples and fitting an NLS model, which has to converge, takes quite a bit of computation time.

```
#set seed for reproducibility
set.seed(42)

#Initialize parameters:
phi   <- 1 #dispersion parameter
tru_asym <- 10
tru_rate <- 1
tru_xmid <- 15
n     <- 40
nsim  <- 10000
estimates <-  data.frame(Asym=rep(0,nsim),xmid=rep(0,nsim),scal=rep(0,nsim))

#for loop simulates nsim different samples using the above parameters,
#storing the fitted coefficients each time
for(i in 1:nsim){
  #we have to be able to handle the occasions where nls and SSlogis fail due to
  #algorithmic issues. Because of this, we will use a while loop to iterate the
  #data generating process and model fitting until a successful model fit occurs.
  exitflag<-FALSE
  while(!exitflag){
    #simulate x,E[y|x],y
    x <- sample(1:30,n,replace=TRUE)
    Ey <- S(x,asym=tru_asym,rate=tru_rate,xmid=tru_xmid)
    y <- numeric(length=length(x))
    for(j in 1:n){
      y[j] <- rgamma(1,shape=Ey[j]/phi,rate=1/phi)
    }

    #perform NLS fit
    #allowing for the algorithm to potentially fail due to singular gradient
    iter_mod <- try(nls(y~SSlogis(x,Asym,xmid,scal),data=data.frame(x=x,y=y)),silent=TRUE)

    #check to see if model fitting was successful. if so, record estimates and exit while loop
    if(length(iter_mod)>1){
      estimates[i,]<-coef(iter_mod)
      exitflag<-TRUE
    }
  }
}
```

Now we're ready to investigate the results of these simulated samples. First, let's see if the simulated parameter estimates converged, and if so, whether or not they converged to the true parameter values which generated the data:

```
mcdf <- estimates%>%
  mutate(Asym=cummean(Asym),xmid=cummean(xmid),scal=cummean(scal),n=1:nsim)%>%
  pivot_longer(1:3,names_to = "parameter",values_to = "estimate")
p4<-ggplot(filter(mcdf,parameter=="Asym"),aes(x=n,y=estimate))+
  geom_line()+
  geom_hline(yintercept = tru_asym,linetype=2,color="red")+
  labs(x=NULL,y="",title="Asymptote")+
```

```
  theme_minimal()
p5<-ggplot(filter(mcdf,parameter=="xmid"),aes(x=n,y=estimate))+
  geom_line()+
  geom_hline(yintercept = tru_xmid,linetype=2,color="red")+
  labs(x=NULL,y="Asymptote Estimate",title="Midpoint")+
  theme_minimal()
p6<-ggplot(filter(mcdf,parameter=="scal"),aes(x=n,y=estimate))+
  geom_line()+
  geom_hline(yintercept = tru_rate,linetype=2,color="red")+
  labs(x="n Simulated Samples",y="",title="Scale")+
  theme_minimal()
grid.arrange(p4,p5,p6,ncol=1)
```

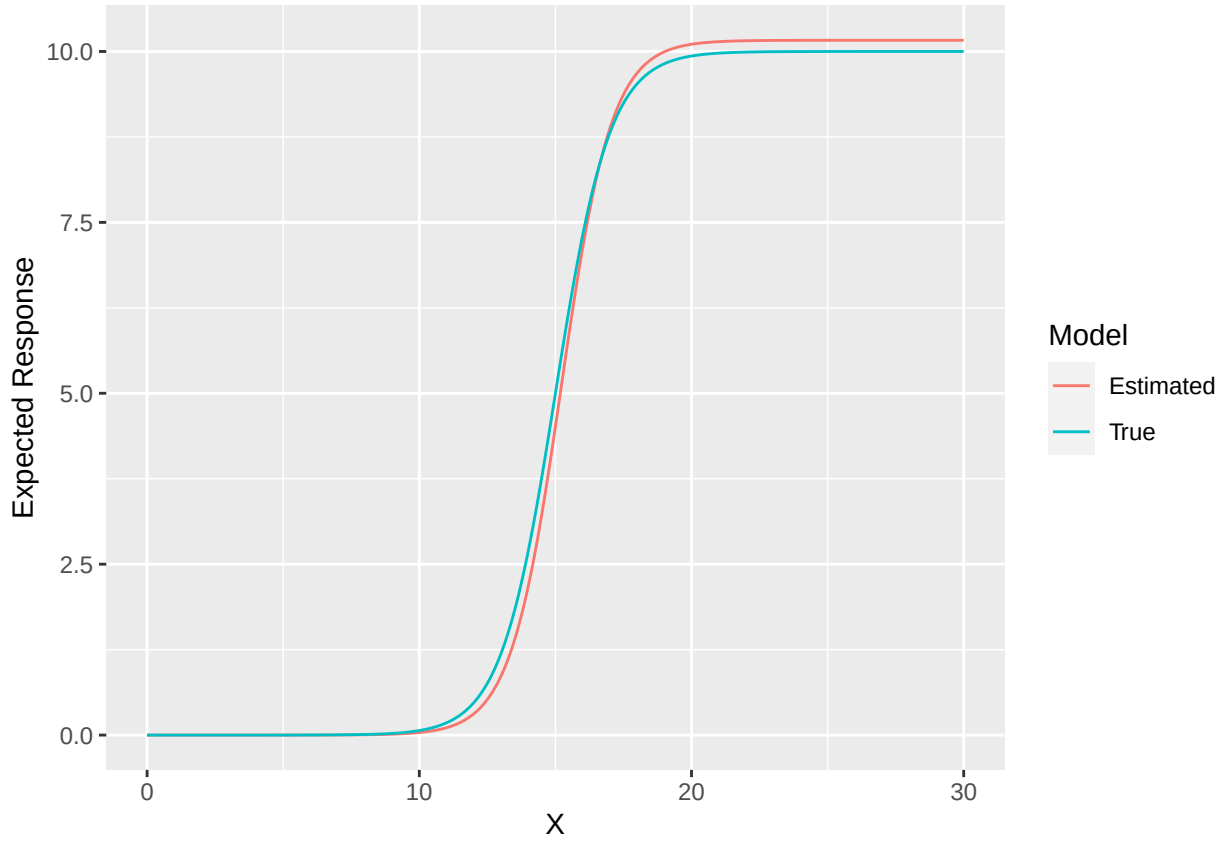


Here, the true values of the parameters which were used to generate these data are represented as horizontal dotted red lines. We can see that all of the estimates appear to have some bias present. Here's a table of the "true" values which were used to simulate the data, and the monte-carlo estimates of the expected value of each NLS estimate:

```
##                          Asymptote Midpoint    Scale
## True Parameter Values      10.0000 15.00000 1.000000
## Estimated Parameter Mean   10.1636 15.20963 1.076594
```

However, just because there is a bias doesn't mean these estimates are *bad* per se. For instance, for the expected estimated values, and the true values, we can plot the logistic curve based on each to see how different the two models are:

```r
t <- seq(0,30,length.out=1000)
true <- S(t,tru_asym,tru_rate,tru_xmid)
estimated <- S(t,mean(estimates$Asym),mean(estimates$scal),mean(estimates$xmid))
data.frame(t=t,True=true,Estimated=estimated)%>%
  pivot_longer(2:3,names_to = "Model",values_to = "y")%>%
  ggplot(aes(x=t,y=y,color=Model))+
  geom_line()+
  labs(x="X",y="Expected Response")
```



In the case of my plant science colleague, this amount of bias is likely a tolerable one. In the future I would love to work on this problem to find unbiased estimates for my colleague. I suspect a maximum likelihood approach would be more appropriate. Perhaps with the correct link function, a generalized linear model can be used to estimate these parameters unbiasedly.

References

Bates, Douglas M. 1988. *Nonlinear Regression Analysis and Its Applications*. John Wiley & Sons.

Faraway, Julian J. 2015. *Linear Models with r*. 2nd ed. CRC Press.