Generalized Linear Models

We have seen linear regression and logistic regression, both of which are examples of generalized linear models (GLMs), but there are many other possible generalized linear models.

GLMs are typically characterized by three components:

1. *linear prediction of predictors ($X\beta$)*

2. *A link function for transforming the linear predictor to match the support of the response.*

3. *A probability distribution for the data.*

A generalized linear model can also include other parameters such as variance or overdispersion terms and/or cutpoints for latent responses.

When formally writing out a GLM, it is important to include these three components:

$$y \sim Bernoulli(\pi) \tag{1}$$
$$logit(\pi) = XB \tag{2}$$

Chapter 15 in ROS mentions another set of GLMs which we will cover:

- Poisson and negative binomial models for count data
- logistic-binomial model, where y_i is a set of n_i Bernoulli trials and a related (beta-binomial model)
- The probit model for binary data
- Multinomial logit (and probit models) for categorical data, both ordered and unordered
- Robust regression, using non-normal errors for continuous data.

Count Regression

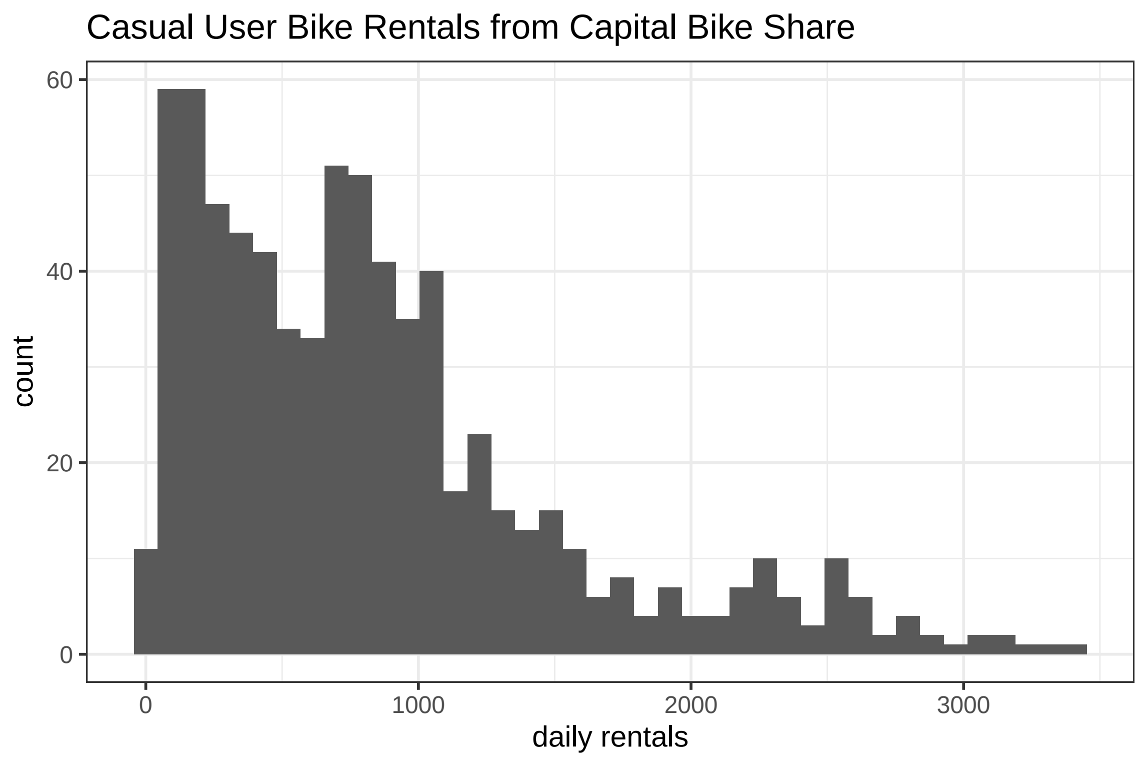
Count regression is applicable with the response term is a non-negative integer. These scenarios generally do not have a maximum value, such as the number of MSU students that are infected with COVID-19, which are more appropriately modeled using a binomial framework.

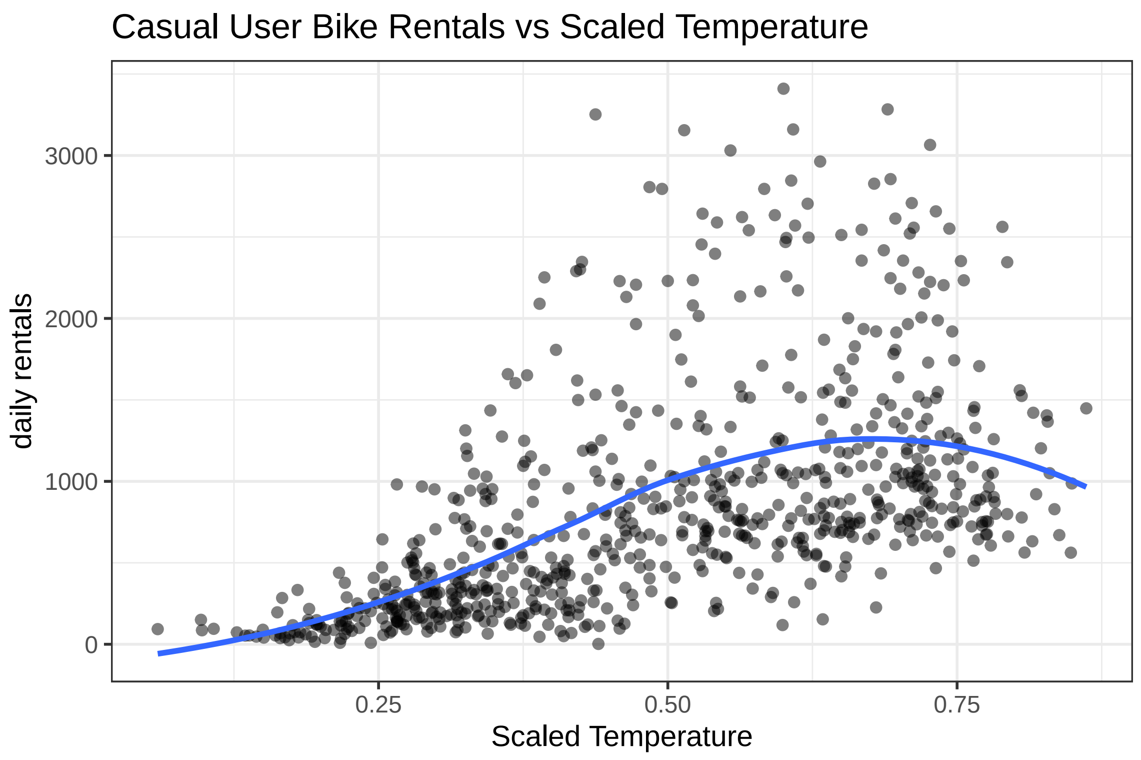
The motivating example that we will use for this scenario is a dataset that contains the total number of daily bike rentals from the Capital Bikeshare System in Washington, DC.

```
bikes <- read_csv("https://raw.githubusercontent.com/STAT506/GLM_Lectures/main/daily_bike.csv")
```

```
## Parsed with column specification:
## cols(
##   instant = col_double(),
##   dteday = col_character(),
##   season = col_double(),
##   yr = col_double(),
##   mnth = col_double(),
##   holiday = col_double(),
##   weekday = col_double(),
##   workingday = col_double(),
##   weathersit = col_double(),
##   temp = col_double(),
##   atemp = col_double(),
##   hum = col_double(),
##   windspeed = col_double(),
##   casual = col_double(),
##   registered = col_double(),
##   cnt = col_double()
## )
```

```
bikes <- bikes %>% mutate(temp_centered = scale(temp))
```


Casual User Bike Rentals from Capital Bike Share
count
daily rentals
0
20
40
60
0
1000
2000
3000


Casual User Bike Rentals vs Scaled Temperature
daily rentals
Scaled Temperature
0
1000
2000
3000
0.25
0.50
0.75

The starting point for count regression is the Poisson model:

$$y_i \sim Poisson(\mu_i) \tag{3}$$
$$\log(\mu_i) = X_i\beta \tag{4}$$

one implication of this model is that a one unit increase in $X\beta$ would have a logarithmic relationship with y

Furthermore, the Poisson distribution also has a single parameter that controls the mean ***and*** *variance of the distribution, so the standard deviation implied by the model would be $\sqrt(\mu_i)$*

When there is more variance in the data than the data would suggest, this phenomenon is referred to as overdispersion

In the presence of overdispersion, negative binomial regression is a common solution. *Note, sometimes a quasi-Poisson distribution is used too*

$$y_i \sim NegativeBinomial(\mu_i, \phi) \tag{5}$$
$$\log(\mu_i) = X_i\beta \tag{6}$$

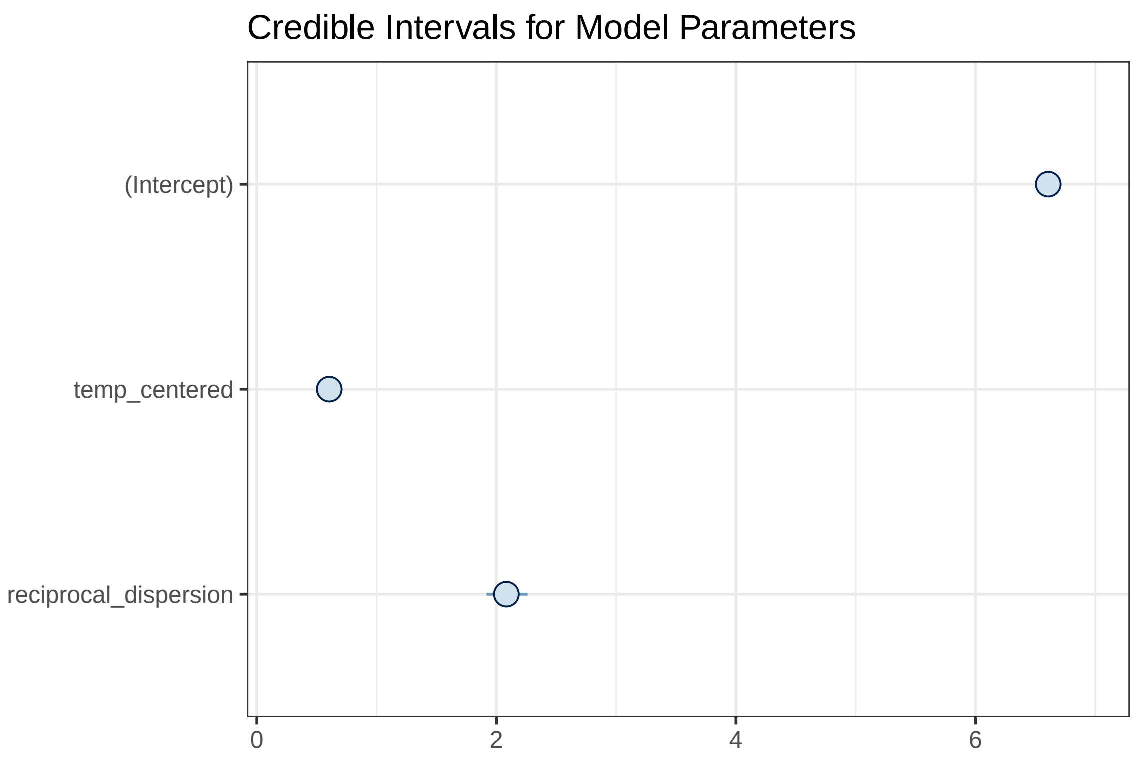
The parameter ϕ can account for additional dispersion in the model. Formally, the standard deviation of $y|x = \sqrt{E(y|x) + E(y|x)/\phi}$. So when $\phi \to \infty$ this limit results in a Poisson distribution.

Model Fitting and Intepreting Coefficients

```r
nb_model <- stan_glm(casual ~ temp_centered, family = neg_binomial_2, data = bikes, refresh = 0)
print(nb_model)
```

```
## stan_glm
##  family:       neg_binomial_2 [log]
##  formula:      casual ~ temp_centered
##  observations: 731
##  predictors:   2
## ------
##               Median MAD_SD
## (Intercept)   6.6    0.0   
## temp_centered 0.6    0.0   
## 
## Auxiliary parameter(s):
##                       Median MAD_SD
## reciprocal_dispersion 2.1    0.1   
## 
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```r
plot(nb_model) + theme_bw() + ggtitle('Credible Intervals for Model Parameters')
```



The intercept can be interpreted as the expected outcome (on the log scale) when the other predictors are zero. Or, more intuitively the exponential of the intercept is the expected outcome when the other predictors are zero.

So in the case $\exp(\beta_0) = 741$ with a credible interval of (705, 778).

The slope coefficient in this model represents the expected difference in y (but again in the logarithmic scale).

Given that the model can be simplified as $y \sim NB(\exp(\beta_0 + \beta_1 x), \phi)$, the coefficient can also be interpreted in a multiplicative way. In particular $\exp(\beta_1)$ is the multiplicative change in y for a one unit change in x. *Again, think about utility of scaling or meaningful units.*

So with this model $\exp(\beta_1) = 1.83$ with a credible interval of (1.73, 1.94).

Count GLMS with Exposure

In many situations there is an "exposure term". With our bike share example, the exposure would be one day for each observation, so the exposure essentially cancels out.

In other situations, the exposure could vary and the model could explicitly include this in the model.

$$y_i \sim NB(u_i \theta_i, \psi),$$

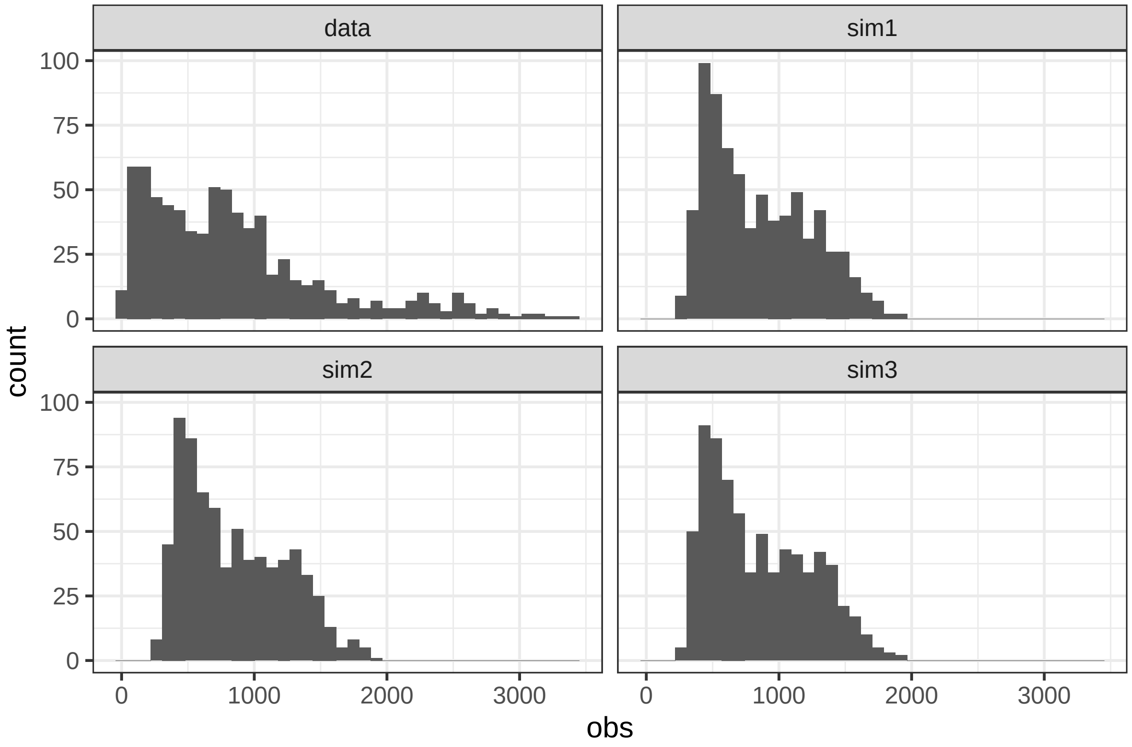
where θ_i is a rate per unit exposure and u_i is the exposure.

As with other regression frameworks, posterior predictive checks can be a useful tool for model checking.

```r
pois_model <- stan_glm(casual ~ temp_centered, family = poisson, data = bikes, refresh = 0)
pp <- posterior_predict(pois_model)
```

This can either be done visually

```r
tibble(obs = c(bikes$casual, pp[1,], pp[2,], pp[3,]),
       group = rep(c('data','sim1','sim2','sim3'), each = ncol(pp))) %>%
  ggplot(aes(x = obs)) + geom_histogram(bins = 40) + facet_wrap(.~ group) + theme_bw()
```



or by comparing summary statistics between the simulated datasets and the observed data.

```r
sim_max = apply(pp,1,max)
data_max = max(bikes$casual)
tibble(sim_max = sim_max) %>% ggplot(aes(x = sim_max)) + geom_histogram(bins = 50 ) +
  geom_vline(xintercept = data_max) + theme_bw() +
  ggtitle("Comparison of maximum value from simulation vs model fit")
```

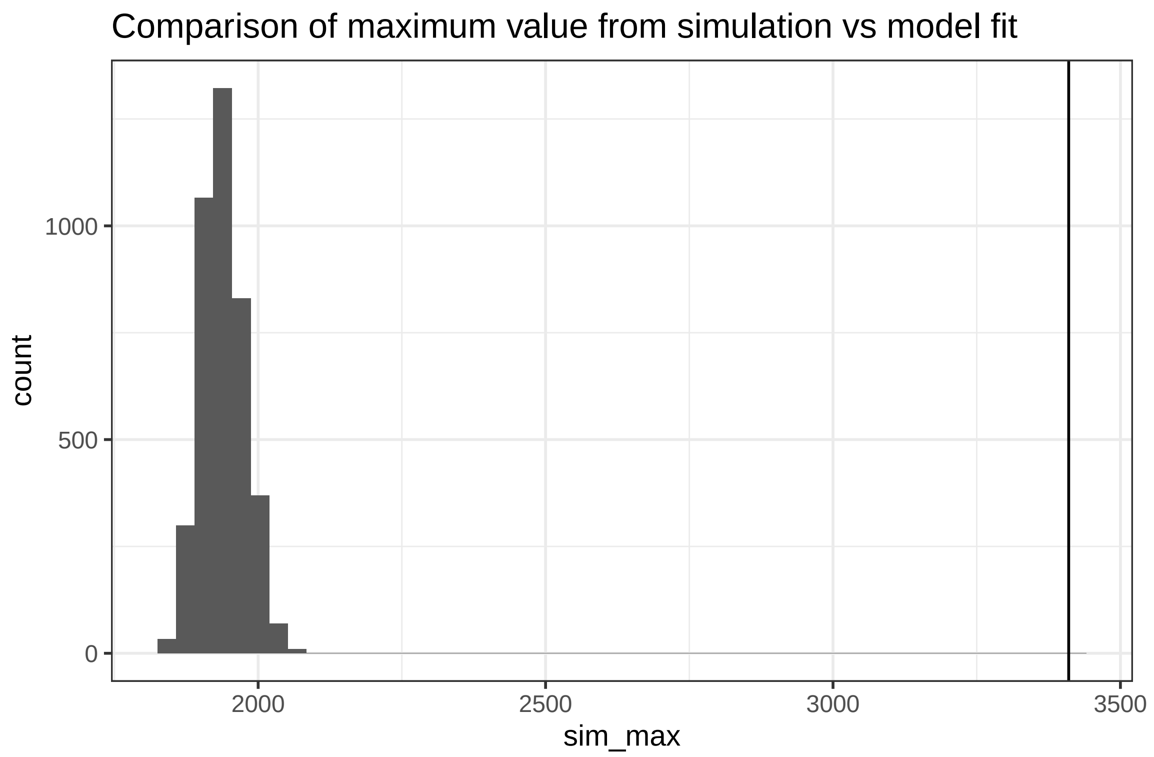


Figure 1: Vertical line represents maximum value in observed dataset