# An Introduction to Generalized Linear Models

Emma Grossman, Leah Marcus, Emily Palmer, Katherine Pulham, Andrew Rumments

2021-03-19

# Contents

Chapter 1

# About this book

Welcome to our bookdown book on Generalized Linear Models. This book is the result of the authors' Reading and Conference Class at Oregon State University Winter term 2021. We hope you find it useful. The code generating this book can be found at https://github.com/GeneralizedRegressionModelingAgency /GRMA. If you find a problem or have questions, we encourage you to file an issue in our Github Repository.

To learn this material ourselves, we used **Generalized Linear Models With Examples in R** by Peter K. Dunn and Gordon K. Smyth(Dunn and Smyth, 2018). If you are looking for more background reading or examples and applications, we recommend this book, as it is a good, accessible, resource to understand these models.

A big thank you to James Molyneux for leading and organizing our reading and conference class.

## 1.1 About the Authors

Emma Grossman is a graduate student finishing her MS degree in Statistics at Oregon State University. She enjoys using machine learning to solve statistical problems and loves programming in R. You can find more about her on her website.

Leah Marcus is a graduate student obtaining her MS in Statistics. Her interests are statistical modeling applications, with a focus in Economics and baseball. You can find her on LinkedIn.

Emily Palmer is a graduate student in statistics studying statistical methods in microbiome research. You can find more of her work on her GitHub, or follow her on Twitter.

Katherine Pulham is a graduate student and early-career researcher in the field of statistics. Her research interests are point processes, statistical computing, regression methods and Bayesian statistics. You can find her on LinkedIn.

Andrew Rumments exists and is cool

## 1.2 Find a problem? Have a question?

The best way to notify the authors of a problem is to file an issue on this book's [GitHub repository.

# Chapter 2

# Introduction

## 2.1 Linear models - a kind of Generalized Linear Model

If you have fit a linear model before, congratulations! You have already fit (one case) of a generalized linear model (GLM). Over the course of this book, we will explore the framework of Generalized Linear Models, why the Linear Model is a special case of a GLM, and two common type of GLMs, including logistic regression (for binary/binomial count data) and poisson regression (for count data).

Let's review why we would fit a linear model. For linear regression, given our data, (if we make some key assumptions 2.2 ), we can perform inference or prediction by assuming that our response value forms a linear relationship with our explanatory variable (or variables). Generalized linear models expand upon the linear model assumptions to increase the kinds and ways we can use data to answer questions.

The reasoning behind linear models can be intuitive; if we have two continuous variables, and make a scatterplot our data, and see this:

We might want to fit a straight line through the cloud of points, i.e. modeling the relationship linearly.

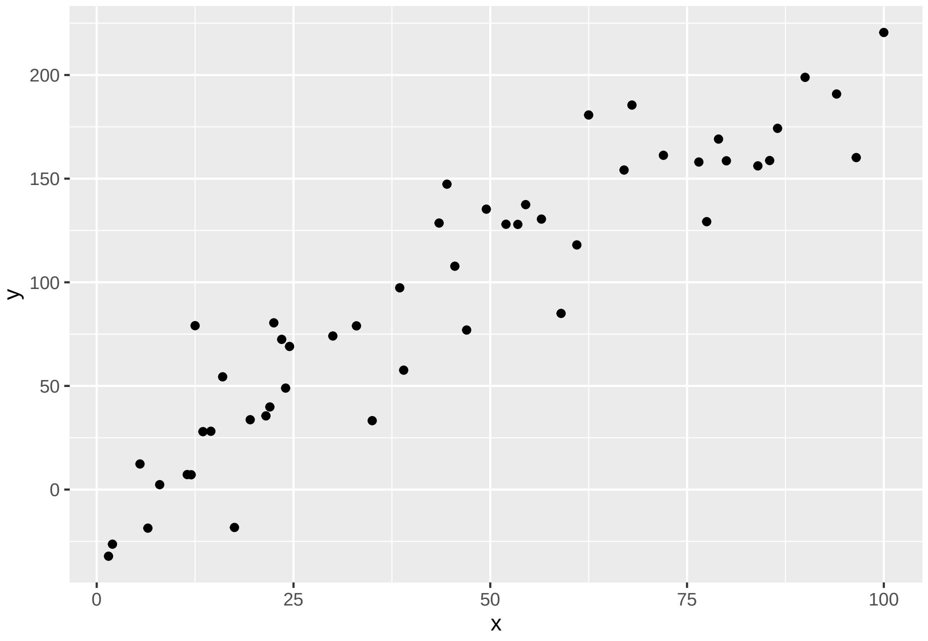


Figure 2.1: A scatterplot of points for two numeric variables.

To interpret this relationship and make predictions, we need to know the slope and intercept of this line. This is done by minimizing the least squares, which will be explored in Chapter 3 @ref{linear}. (will it? should it?) Linear models can also have multiple explanatory variables ($X_1, ... X_p$ instead of just one $X$), and this becomes multiple linear regression. The visualization of this kind of data is more difficult, and for example purposes, we will only use one explanatory variable, $X$.

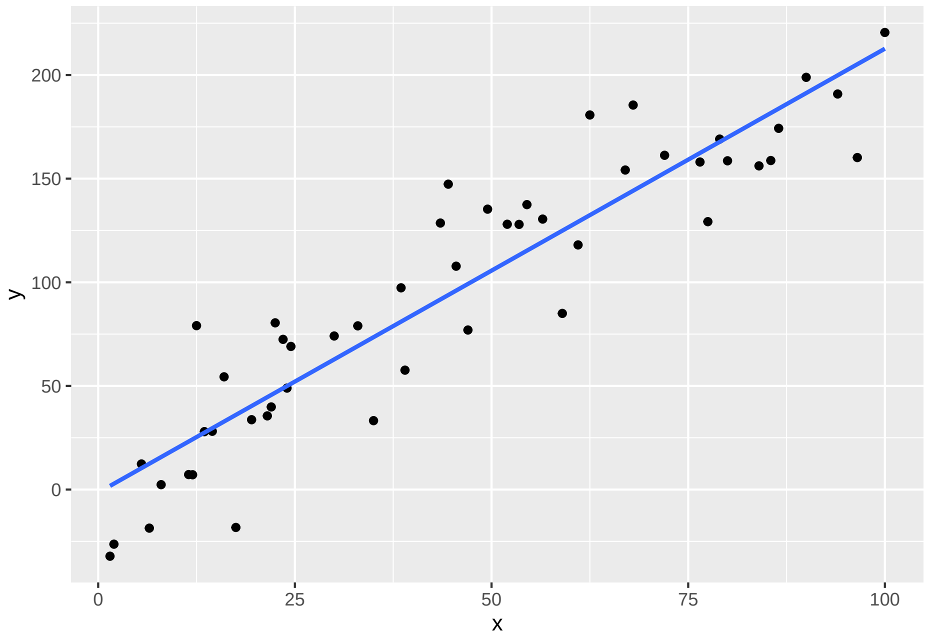


Figure 2.2: A scatterplot of points now with a *line of best fit* added.

## 2.2 Assumptions of Linear Models

A linear model might very well be a good model if we have data like that shown in Figures 2.1 and 2.2. However, there are many cases where it might be inappropriate to use a linear model. To understand these cases, we first review the assumptions of linear models.

Linear models assume:

- The relationship between the explanatory variables and the response is linear.
- The samples are independent.
- The errors are normally distributed with mean 0 and constant variance.

You might have seen these assumptions written in notation as such.

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

where $(x_i, y_i)$ represents the observed values for observation $i = 1, 2, \ldots, n$ and

$$\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$$

In words, this means that the errors are independent and identically distributed from a normal distribution with mean 0 and constant variance $\sigma^2$ (Notice how there is no subscript $i$ for the variance. This means that all our our $y_i$ values will have the same variation, regardless of what their associated values of $x_i$ are). If we are performing multiple linear regression with up to $p$ explanatory variables then this first assumption becomes $y_i = \beta_0 + \beta_1 x_{1,i} + ... + \beta_p x_{p,i} + \epsilon_i$. That is, our response variable, $Y$, is a *linear function* of the explanatory variables $X_1, X_2, ..., X_p$.

## 2.3 Reframing Linear Models as Generalized Linear Models

To see how the linear model is just a special type of Generalized Linear Model, we will slightly modify how we think of these assumptions, so as to be able to more easily see the parallels. Instead of the response $y_i$, now consider the mean response $\mu_i$, which for a given value of one predictor $x_i$ is the mean, or expected value of all responses with the value of that explanatory variable.

Then our assumptions become:

$$\mu_i = \beta_0 + \beta_1 x_i \tag{2.1}$$

$$y_i \overset{\text{iid}}{\sim} N(\mu_i, \sigma^2) \tag{2.2}$$

We refer to the first condition, Equation (2.1), as the **Systematic Component** and the second condition, Equation (2.2), as the **Random Component**. In this framework, the systematic component describes how the explanatory variables impact some parameter, in this case the mean. Once we model how the mean parameter changes depending on the explanatory variable, the random component then describes the variability we see in the response given the value of the parameter.

Generalized Linear Models are inspired by this idea of generalizing the random and systematic component of linear models. More broadly, a Generalized Regression Model has a systematic component

$$g(\mu_i) = \beta_0 + \beta_1 x_i + \epsilon_i$$

where, in order to generalize the systematic component, we use a link function $g(y)$ which then requires some function of the response to be linearly related to our explanatory variables.

The random component can then be written in general as:

$$y_i \overset{\text{iid}}{\sim} EDM(g(\mu_i), \phi)$$

Where EDM stands for a probability distribution from the Exponential Dispersion Family (EDM). We discuss these EDMs in greater detail in the next chapter but we can note for now that the Normal, Binomial, and Poisson distributions all belong to the Exponential Dispersion Family. The EDM distribution we use depends on the mean response, as well as a dispersion parameter $\phi$. For many commonly use GLMs, and the ones discussed in this book, this $\phi$ is known.

We note that normal linear models fall easily into this framework, where $g(y_i) = y_i$ the identity function, and use the Normal distribution as our random component.

Deciding on what Random and Systematic component to use in a model requires …. us to have some idea about both the possible values our response variable takes as well as having some idea about which probability distribution might have generated our response.

## 2.4 What happens when we break the assumptions of linear models

How can we tell when these assumptions are violated? We can examine both the random and systematic component and see if our data can/should be modeled in such a way. This largely comes from knowledge of the data.

Linear models are generally robust, and can be reasonable when assumptions are not exactly met. However, if we know assumptions are not met, and how they are not met, it is appropriate to use a more appropriate model for the data.

## 2.5 Parameter estimation

Another difference between linear models and generalized linear models is the way we estimate the parameters $\beta$. For linear models, we find the minimize the sums of squares from the predictor to the response(s). This has a closed form solution, and can be calculated by hand (if one **really** wanted to). For generalized linear models, we estimate the $\beta$ parameters using Maximum Likelihood Estimation. For the Normal linear model case, this is equivalent to minimizing the sums of squares. However, for other GLMs, there is no closed form solution, which requires us to perform an iterative algorithm to land at the parameter estimates. Luckily this is all done behind the scenes in R. Those interested in learning more about how parameter estimation works for GLMs can read Chapter 6 of (Dunn and Smyth, 2018).

## 2.6 LMs and GLMs in R

Fitting GLMs in R is very similar to fitting linear models in R. For linear models, we use the function `lm()`. For generalized regression models, we use the function `glm()`. Both require a formula input, but `glm()` also requires the user to specify the random and systematic part of a GLM, by specifying the `family` and `link` function. Help for fitting a GLM can be found by `?glm()` and reading more about the family argument.

Let's use some simulated data to see how GLMs are fit in R, and

```
head(sim_data)
```

```
##           y     x1    x2
## 1 -725.9119  32.00 264.5
## 2 -865.7442 -24.25 266.8
## 3 -735.8747  51.50 283.6
## 4 -372.6265 241.25 284.6
## 5 -530.3268 127.25 249.0
## 6 -386.1976 217.25 269.6
```

First, fit a linear model the normal way, using `lm()`

```
lm_mod <- lm(y ~ x1 + x2, data = sim_data)
summary(lm_mod)
```

```
## 
## Call:
## lm(formula = y ~ x1 + x2, data = sim_data)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -45.373 -15.786  -0.426  13.828  46.333 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  8.64298   24.72986   0.349    0.728    
## x1           1.95640    0.02618  74.721   <2e-16 ***
## x2          -3.04431    0.09788 -31.101   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 22.18 on 72 degrees of freedom
## Multiple R-squared:  0.9901, Adjusted R-squared:  0.9898 
## F-statistic:  3584 on 2 and 72 DF,  p-value: < 2.2e-16
```

Now we show that we can use `glm()` the same way, specifying `family = "gaussian"`, which gives the identity link.

```
glm_mod <- glm(y ~ x1 + x2, data = sim_data, family = "gaussian")
summary(glm_mod)
```

```
## 
```

```
## Call:
## glm(formula = y ~ x1 + x2, family = "gaussian", data = sim_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -45.373  -15.786   -0.426   13.828   46.333
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.64298   24.72986   0.349    0.728
## x1           1.95640    0.02618  74.721   <2e-16 ***
## x2          -3.04431    0.09788 -31.101   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 491.8093)
##
##     Null deviance: 3560648  on 74  degrees of freedom
## Residual deviance:   35410  on 72  degrees of freedom
## AIC: 682.64
##
## Number of Fisher Scoring iterations: 2
```

As you can see from the summary outputs, we get results which are very similar, as we should, since both functions fit the same model. However, some outputs of our linear model are slightly different when we compare them to the GLM version. For instance, we no longer have values for the residual standard errors, R-squared values and F-statistics. Instead, in the GLM, we have summary values like the null and residual deviance along with the AIC. We'll talk more about these differences in future chapters. But for now, know that many of the functions we used for linear models, like `coef()`, `predict()`, etc, will perform as expected, when we pass a generalized linear model fitted using `glm()` in as the first argument.

## 2.7 Some definitions

We close this chapter with some definitions of common terminology and explanation of notation that will be used throughout this text.

Predictor - the thing on the y-axis Explanatory variable - the stuff on the x-axis. Note that we can have more than one (but won't plot it then), and then this becomes multiple variable regression.

Something that is an estimated quantity will have a hat over it. For example, we might assume that there is some 'true' (but unknown) linear relationship between our explanatory variables and our predictor.

$$y = \beta_0 + \beta_1 x$$

From our sample data, we use a linear model to make an estimate of $\beta_0$ and $\beta_1$, so our estimate/best guess of this true model relationship is

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

We of course want our $\hat{\beta}_0$ and $\hat{\beta}_1$ to be a 'good' and 'close' estimate of the unknown quantities $\beta_0$ and $\beta_1$. Ideas of what 'good' and 'close' mean will be covered in the next section.

## 2.8 Conclusion

Linear models are not always the best tool for describing relationship in data. Luckily we can generalize the ideas and framework developed in linear models to hold for more general cases to create GLMs. Using a more general framework and more general assumptions allows us to build tools that will hold for all GRMs. The most notable of these that we will further explore are GRMs for binary data (ch4) and count data (ch5)

## 2.9 Examples

Perhaps some examples of data and students can tell what type of data it should be modeled by?

Explore the following case of where a linear model is NOT appropriate, and then fit a `glm()` call in R using an appropriate family and link function.

Note that this example might take a minute to load.

Chapter 3

# How are GLMs "different"?

## 3.1 Introdution

So, we've talked about the issues that linear models can run into. The question now is how do we deal with these issues? What we're going to need to do is expand the type of model we're trying to fit. In linear regression we assumed two things: that the response variable $Y_i$ is distributed normally, with constant variance $\sigma^2$, and that the mean of the response variable is a linear combination of the explanatory variables. These two assumptions can be stated as

1. $Y_i \sim \mathcal{N}(\mu_i, \sigma^2)$
2. $\mu_i = \beta_0 + \beta_1 X_{i,1} + ... + \beta_k X_{i,k}$

In this chapter we're going to make our model more general by expanding these two assumptions. The first assumption, which we will call the random component, is going to change from $Y$ being distributed normally to $Y$ being distributed according to *some probability family.* The second assumption is going to change from $\mu_i$ directly equaling the linear predictor to *some function* of $\mu_i$ being equal to this linear predictor.

## 3.2 Assumptions of a GLM

GLMs are made up of two components: a random component, and a structural component. In general, what we're saying is that the response variable of interest is a random variable that follows a specific probability distribution (random component). This probability distribution is, in some way, related to a linear combination of the explanatory variables (systematic component). This linear combination of the explanatory variables is where the "linear" in "generalized linear model" comes from. In linear regression, which is a special case of the generalized linear model, the random component is that Y comes from a normal distribution: $Y_i \sim N(\mu_i, \sigma^2)$ and the systematic component is that the mean is some linear combination of the explanatory variables: $\mu_i = \beta_0 + \beta_1 X_{1i} + ... + \beta_k X_{ki}$. With GLMs, our goal is to extend this framework so that we're not just limited to the normal distribution for the random component of our model, for reasons we discussed in the last chapter.

However, when we fit these models, we need to be sure of a couple of things. We need to ensure that for a linear combination of explanatory variables, we can identify which distribution the response variable comes from. We also need to ensure that the parameters of that distribution we're trying to fit are estimable. To ensure that we're able to properly fit these models, GLMs consider a specific kind of family of distributions for the random component: the Exponential Dispersion Model.

## 3.3 Framework

### 3.3.1 Exponential Dispersion models

An exponential dispersion model is a specific type of random variable, whose pdf follows a specific form:

$$f_Y(y) = a(y, \phi) exp\left[\frac{y\theta - \kappa(\theta)}{\phi}\right]$$

in this form, $\theta$ is called the *canonical parameter*, and $\phi$ is called the *dispersion parameter*. For our purposes, the function $a(y, \phi)$ is not of much interest, but it is needed to guarantee that $f_Y(y)$ integrates to 1, and is therefore a valid probability density function. $\kappa(\theta)$ is called the *cumulant* function, and will be useful to us in estimation. Another term for an exponential dispersion model is to say that the family of random variables is an exponential family.

A surprising, and fortunate, number of families of distributions are exponential dispersion models. Notably, some of them are

- Normal random variables
- Bernoulli random variables
- Binomial random variables
- Poisson random variables
- Exponential random variables
- Gamma random variables
- Negative binomial random variables

We'll spare the details for most of these families, but to show the general idea for how we decide whether or not a family of random variables is an exponential dispersion model, we shall consider the poisson random variable.

**Example:** For a poisson random variable, the pmf is written as

$$f_Y(y) = e^{-\lambda}\frac{\lambda^y}{y!}$$

by applying the identity $x = e^{log(x)}$ to the numerator, we see that this is equivalent to

$$f_Y(y) = \frac{1}{y!}exp\left[-\lambda + ylog(\lambda)\right] = \frac{1}{y!}exp\left[\frac{ylog(\lambda) - \lambda}{1}\right]$$

and we see that the poisson random variable is an exponential dispersion model with dispersion parameter $\phi = 1$, with canonical parameter $\theta = log(\lambda)$ and with cumulant function $\kappa(\theta) = \lambda = e^{\theta}$. Notice how we left out the $\frac{1}{y!}$ term of the exponential because it was not needed to put the function into this important form. $\square$

### 3.3.2 Properties of EDMs

Once we can get a probability distribution function into the exponential dispersion model form, we can connect this form to both the mean and variance of the random variable. The expected value (mean) of the random variable is simply the first derivative of the cumulant function with respect to the canonical parameter:

$$E[Y] = \mu = \frac{d}{d\theta}\kappa(\theta)$$

The cumulant function is also related to the variance of the random variable. The variance of the random variable is the dispersion parameter multiplied by the second derivative of the cumulant function with respect to the canonical parameter:

$$var(Y) = \phi\frac{d^2}{d\theta^2}\kappa(\theta)$$

The second part of this expression is an important quantity, called the variance function. Notice that it is equal to the first derivative of the expected value of Y as well:

$$V(\mu) = \frac{d^2}{d\theta^2}\kappa(\theta) = \frac{d}{d\theta}\mu$$

It is worth noting that, in addition to helping us estimate properties of $Y$, the variance function uniquely determines the family of distributions (type of random variable) for a given EDM. For instance, following our previous example, since $\kappa(theta) = e^{\theta}$, the variance function is $V(\mu) = \frac{d^2}{d\theta^2}e^{\theta} = e^{\theta} = \lambda = \mu$. What this means is that *any* EDM with variance function $V(\mu) = \mu$ will be a poisson random variable.

### 3.3.3 Linking the EDM to the explanatory data

Recall, just for a second, the goal of constructing these models. We have a response variable, $Y_i$, and a collection of explanatory variables $X_1, X_2, X_3, ...X_k$. We want to be able to look at a combination of the explanatory variables and draw some conclusions about $Y$. Perhaps we want to predict Y with a point estimator. If we make this sort of prediction, it's also of interest to know how precise that estimate will be, so we may wish to find an interval estimate for the prediction as well. Ultimately, all of these things come from the distribution of $Y$, so the thing that is of interest is to be able to know what the probability distribution of $Y$ is given the input values of the $X_i$'s.

As stated before, the L in GLM stands for linear, and these explanatory variables are where that linearity comes into play. In GLMs, we're assuming that the quantity we'll use to predict the response variable $Y$ is a linear combination of the explanatory data $\beta_0 + \beta_1 X_1 + ... + \beta_k X_k$. We will call this quantity the linear predictor; a common shorthand way of writing it is to use the greek letter $\eta = \beta_0 + \beta_1 X_1 + ... + \beta_k X_k$. In practice, we often have multiple repetitions of the explanatory variables, where $Y_i$ is a random variable who's distribution is somehow linked to the covariates $X_{1i}, X_{2i}, X_{3i}, ...X_{ki}$. In this case, we will denote the separate linear predictors as $\eta_i = X_{1i}, X_{2i}, X_{3i}, ...X_{ki}$. Note that although the variables may change, the coefficients $\beta_0,\ \beta_1,\ ...\ \beta_k$ are the same for every $\eta_i$. These $\beta$ coefficients are the thing we must estimate to fit our GLM.

The question remains of *how* we connect $\eta$ to the distribution of $Y$. First, we have to suppose what kind of distribution $Y$ is coming from (is it a poisson random variable? Binomial?) and then we need to find some function g() such that the expected value $E[Y] = \mu$ is simply $g(\mu) = \eta$. For this, we have to place a couple restrictions on g. First, g must be a strictly monotonic function (strictly increasing or strictly decreasing) from some subset of the real numbers onto the set of all values that $\mu$ could be. We require the monotonicity to ensure that we don't have multiple separate means being linked to the same linear predictor. This function also has to be differentiable to make sure that the tools we use to estimate $\mu$ don't break. In practice, these requirements don't come up very much, since typically there are a couple of link functions that get used for each family of probability densities.

One special link function for each EDM family is the *canonical link function*. For an EDM family of distributions, the canonical link function is the function $g(\mu)$ that satisfies $\eta = \theta = g(\mu)$.

The canonical link function isn't the only valid link function. Take for example the binomial family of distributions, and let $Y \sim Binom(n, p)$, for some known n. Note that $\mu = p$. In this case, the set of possible values of $p$ is the unit interval $(0, 1)$. The canonical link function for this family is the logit function:

$$g(p) = log\left(\frac{p}{1-p}\right)$$

However, there are a couple of other link functions that satisfy the required assumptions. Notably, we have the probit function:

$$g(p) = \Phi^{-1}(p)$$

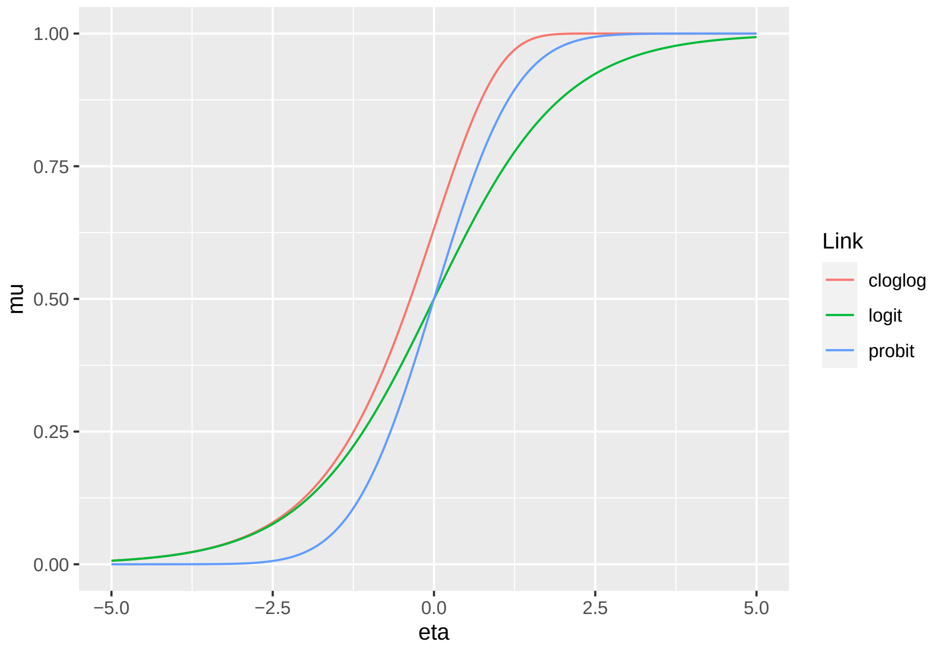
which is just the inverse of the normal CDF $\Phi$. In other words, $\Phi^{-1}(p) = \xi$ where $\xi$ is the real number that satisfies $P(Z \leq \xi) = p$ with $Z$ being a standard normal random variable (mean 0 and variance 1).

One more notable link function for the binomial family is complimentary log-log (or c-log-log) model. This link function is

$$g(p) = log(-log(1-p))$$

All three of these link functions map the real numbers to the unit interval (0,1). Note that since $g(\mu) = \eta$, and since these link functions are invertible (guaranteed by differentiability and strict monotonicity), we can express this as $\mu = g^{-1}(\eta)$. Often times this second form is a more intuitive way to think about how the linear predictors relate to the mean response.

It can be seen that all three of these link functions are sigmoid functions, but that they have slightly different properties:



The consequences of these differences will not be discussed here, this example exists purely to illustrate that an EDM family can have multiple distinct link functions. The consequenses of these varying link functions varies from family to family.

### 3.3.4 Formal definition of a GLM

Formally, a Generalized Linear Model is made of two components: the *probability family* and the *link function*. Given a set of data with response variable $Y$ and explanatory variables $X_1, ...X_k$, we wish to build a Generalized Linear Model. We assume that each $Y_i$ follows a probability distribution from a given EDM family of distribution with mean $\mu_i$ and dispersion parameter $\phi$: $\mu_i$ $Y_i \sim EDM(\mu_i, \phi)$, where $\mu_i$ is such that $g(\mu_i) = \beta_0 + \beta_1 X_{i,1} + ...\beta_k X_{i,k}$ for the link function $g(\mu_i)$ and some vector of parameters $(\beta_0 ... \beta_k)$. We assume all of this is true, and then estimate the parameters $(\beta_0 ... \beta_k)$ using the data and maximum likelihood estimation algorithms. In this book, we will leave these estimation algorithms "under the hood" for brevity's sake, and focus on some common applications of these GLMs. Generally, to fit one of these models in R, you will need to know the family and the link function, as defined above.

# Chapter 4

# Linear Models

## 4.1 Introduction

At this point, you are likely familiar with linear regression. As discussed before, linear regression models are a special case of generalized regression model that we use when the data are normally distributed and have constant variance. We can think of linear regression models in the same terms we think of other regression models.

The two components of a regression model are the random component and the systematic component and for linear regression,

$$\begin{cases} \mathrm{var}[y_i] = \sigma^2/w_i \\ \mu_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ji} \end{cases}$$

where $w_i$ are prior weights and $w_i$ and $\mathrm{E}[y_i] = \mu_i$ are known.

When our linear regression has two $\beta_j$ coefficients and the systematic compnent looks like $\mu = \beta_0 + \beta_1 x_1$, it is called *simple linear regression*. If we have more than two $\beta_j$ coefficients, our regression model is called *multiple linear regression model* or *multiple regression model*.

When all prior weights $w_i$ are equal to one, our regression model is refered to as *ordinary linear regression model* as opposed to when our prior weights $w_i$ have values other than one and is called a *weighted linear regression model*.

As mentioned before, the assumptions belonging to linear regression are:

1. The relationship between $\mu$ and each explanatory variable is **linear**.
2. The unexplained variation in our response is constant, otherwise known as **constant variance**.
3. Each datam is **independent** of all other data points.

ALRIGHT, LETS SEE WHAT THIS POPULAR STATISTICAL METHOD REALLY IS
YET ANOTHER LINEAR MODEL
imgflip.com

Figure 4.1: It's linear models all the way down. Posted on reddit in r/statisticsmemes by u/not_really_redditing.

## 4.2 A “Good” Example of Simple Linear Regression

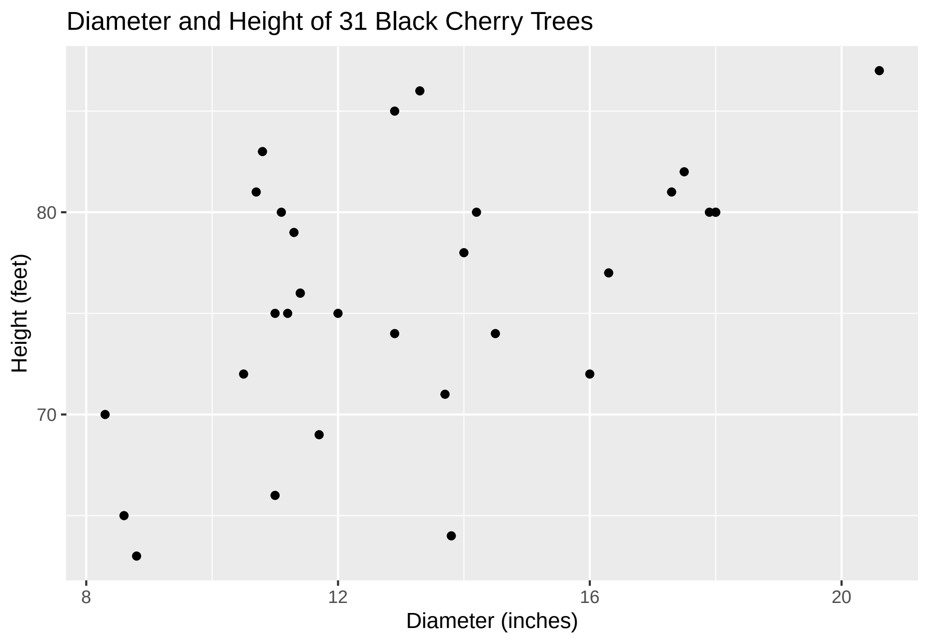
`trees` is a data set that comes with R. It has three variables, the diameter (mistakenly named girth in the data set), height and volume of each of the thirty-one trees in the data set. If we just type `data(trees)` into the Console, R will retreive it for us. This is the data set I’m going to use to showcase a good example of simple linear regression. At this time, let’s also make sure we have the `tidyverse` (which includes `ggplot` for graphing) loaded.

```
library(tidyverse)
data(trees)

# Let's change the name from Girth to Diameter:
trees <- trees %>%
  mutate(Diameter = Girth) %>%
  select(-Girth)

head(trees)
```

```
##   Height Volume Diameter
## 1     70   10.3      8.3
## 2     65   10.3      8.6
## 3     63   10.2      8.8
## 4     72   16.4     10.5
## 5     81   18.8     10.7
## 6     83   19.7     10.8
```

Let’s look at the scatterplot between these two variables, Diameter and Height.

```
ggplot(trees)+
  geom_point(aes(x = Diameter, y =  Height))+
  xlab("Diameter (inches)")+
  ylab("Height (feet)")+
  ggtitle("Diameter and Height of 31 Black Cherry Trees")
```



When plotting the explanatory variable (Diameter) against the response variable (Height), we are looking for linearity, since that is one of our conditions of fitting a linear model to our data. Though the relationship between these two variables seems to be moderate or moderately-weak, it does indeed look linear.

The next condition is constant variance, which we evaluate by looking at the residuals. We first need to produce those residuals which we do by fitting the model. The `lm` function automatically saves residuals, among other information, about the model:

```r
mod <- lm(Height ~ Diameter, data = trees)
summary(mod)
```

```
##
## Call:
## lm(formula = Height ~ Diameter, data = trees)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.5816  -2.7686   0.3163   2.4728   9.9456
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  62.0313     4.3833  14.152 1.49e-14 ***
## Diameter      1.0544     0.3222   3.272  0.00276 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.538 on 29 degrees of freedom
## Multiple R-squared:  0.2697, Adjusted R-squared:  0.2445
## F-statistic: 10.71 on 1 and 29 DF,  p-value: 0.002758
```

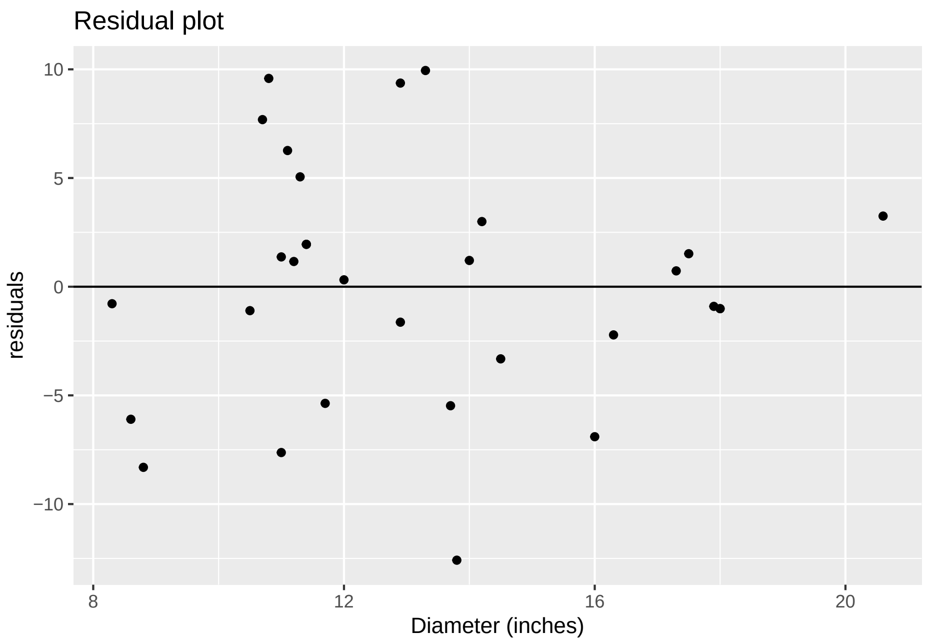
We can view those residuals by typing `head(mod$residuals)`, let's just look at the first six to get an idea of them.

```
head(mod$residuals)
```

```
##         1         2         3         4         5         6
## -0.782575 -6.098886 -8.309759 -1.102186  7.686940  9.581503
```

Next, we should view the actual residuals to check for normality and constant variance.

```
ggplot()+
  geom_point(aes(x = trees$Diameter, y = mod$residuals))+
  geom_hline(yintercept = 0)+
  xlab("Diameter (inches)")+
  ylab("residuals")+
  ggtitle("Residual plot")
```



When examining the residual plot, we are looking for two things: randomness and constant variance. Both are indications that our residuals are $N(0, \sigma^2)$. If we don't see randomness, i.e. there is a pattern, that indicates that are data are not linear. Constant variance is a necessary condition because when we assume the residuals are $N(0, \sigma^2)$, the variance parameter is one estimate, $\sigma^2$. If we see the variance of the residuals changing, then that indicates that the one variance parameter isn't sufficient and since a normal distribution only has one variance parameter, it might not be the best model for the data.

There does not seem to be a pattern to the residuals, so they look random. This again suggests our data are linear. As for constant variance, however, it seems have more variation in the middle than we do at the ends of our graph. It isn't too bad though and we can proceed.

Our last condition is independence. We evaluate independence by looking at how the data were collected. We can find out more information about the trees data set by typing `?trees` into the Console, which will bring up the help file on the data set. This data comes from 31 black cherry trees, felled for timber. Though we don't have much information, the trees could be independent if they were randomly selected to be measured.

The `trees` data set isn't perfect for linear regression, but it meets most of our assumptions. I'll now move on to an example of linear regression with a data set that is not linear.

## 4.3 A "Bad" Example of Simple Linear Regression

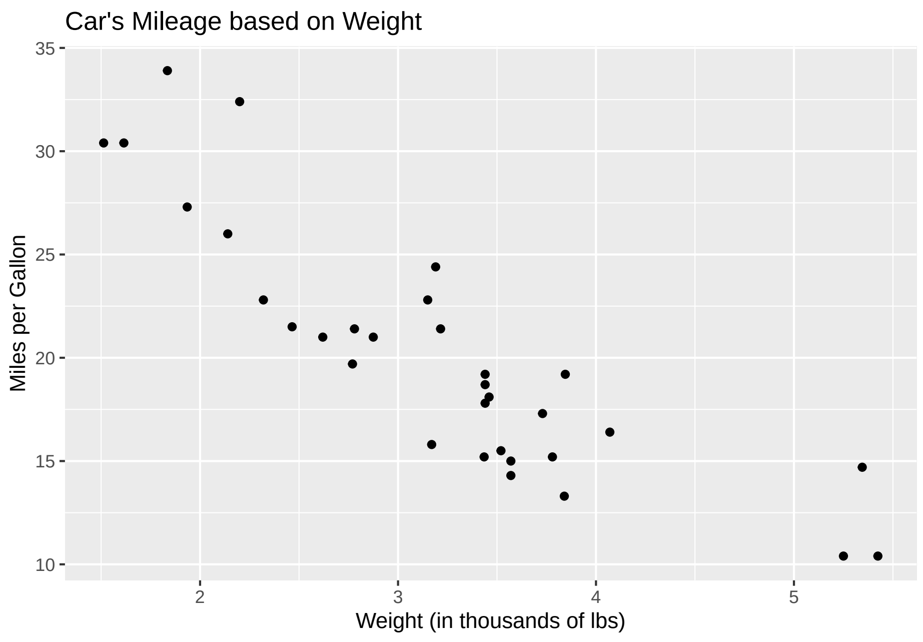
First, we load in the `mtcars` data set, look at the first six obervations of each variable.

```r
library(tidyverse)
data(mtcars)
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Now, let's take a look at a scatterplot of two variables in `mtcars`.

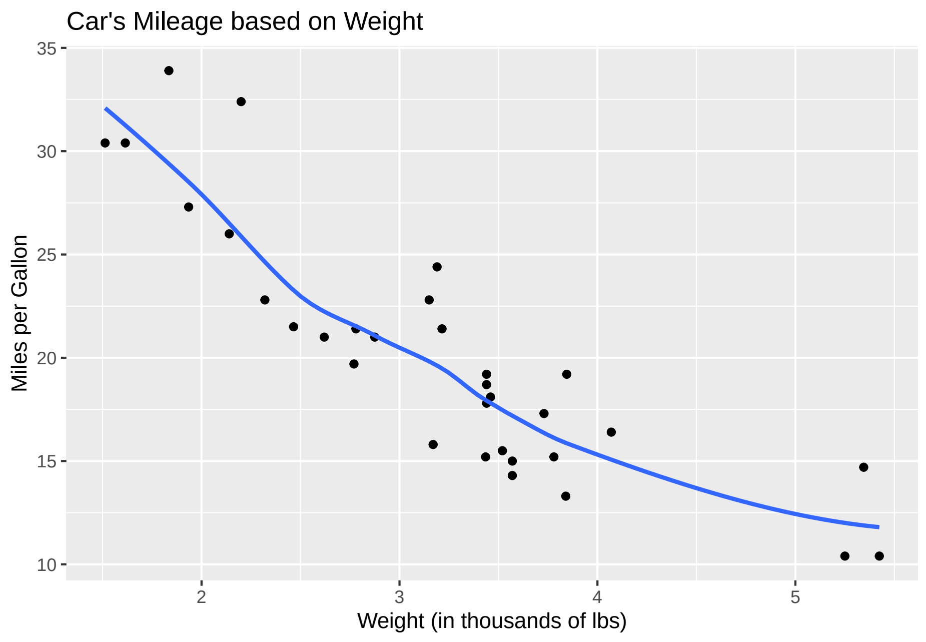
```r
ggplot(mtcars)+
  geom_point(aes(x = wt, y = mpg))+
  xlab("Weight (in thousands of lbs)")+
  ylab("Miles per Gallon")+
  ggtitle("Car's Mileage based on Weight")
```



We can see that this graph looks mostly linear, there seems to be a strong, negative relationship between a car's weight and and its mpg. There is some nonlinearity because of the three points for cars with a weight above 5,000 lbs; the graph look a bit curved so we should be cautious as we proceed.

If we add a smoothed line to the plot, we can see that it is indeed curved.

```
ggplot(mtcars)+
  geom_point(aes(x = wt, y = mpg))+
  geom_smooth(aes(x = wt, y = mpg), se = FALSE)+
  xlab("Weight (in thousands of lbs)")+
  ylab("Miles per Gallon")+
  ggtitle("Car's Mileage based on Weight")
```



This graph addresses one of the assumptions of linear regression: linearity. Next, let's check whether there is constant variation. We do this with a residual plot. First, we have to fit a model to our data in order to obtain residuals.

To fit a model, we use the `lm` function.

```
fit <- lm(mpg ~ wt, data = mtcars)
summary(fit)
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
## wt           -5.3445     0.5591  -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7446
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

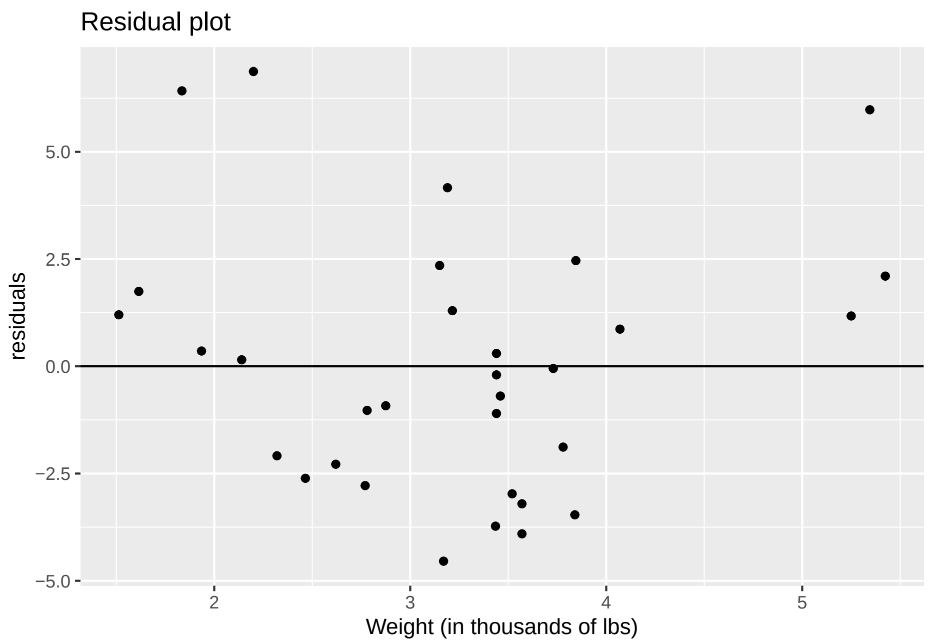
Among other things, the `lm` function saves the residuals.

```
head(fit$residuals)
```

```
##         Mazda RX4     Mazda RX4 Wag        Datsun 710    Hornet 4 Drive
##        -2.2826106        -0.9197704        -2.0859521         1.2973499
## Hornet Sportabout           Valiant
##        -0.2001440        -0.6932545
```

We can plot the residuals against the weight and look for randomness and constant variance.

```
ggplot()+
  geom_point(aes(x = mtcars$wt, y = fit$residuals))+
  geom_hline(yintercept = 0)+
  xlab("Weight (in thousands of lbs)")+
  ylab("residuals")+
  ggtitle("Residual plot")
```



The graph above seems to have a curved pattern. The cars that weigh the least have large residuals, all above the zero line. The residuals of the cars that weigh between 2,250 and 4,250 lbs are lower and seem to be centered on the zero line. The cars that weigh the heaviest also have large residuals. So, there seems to be a parabola shape to our data, indicating nonlinearity.

As for constant variance, we could go either way. As we look at different car weights, the distance between the top-most point and the bottom-most point is fairly similar indicating we could have constant variance. That might be enough evidence for some folks, but others might point out that the though the cars that weigh the least and the cars that weigh between 2,250 and 4,250 lbs are similar, cars that weigh the most have very small variance comparatively.

Lastly, we need independence. We evaluate independence by checking the data and how it was obtained. If you type `?mtcars` into the Console, the Help file with pop up for the `mtcars` data set. Here, we learn that the data came from a 1981 textbook on biometrics. While there isn't much information on the data here, we can look to the data for answers. The data records many different attributes of certain make and models of cars. In general, one car of a particular make and model is unlikely to influence the attributes of another car. The only exception I can think of is cars of the same make. Perhaps two cars made by Mazda might not be totally independent, but this would be a more serious concern if we had observations of two cars with the same make and model but different years. So, this last condition is likely met.

In general, using `wt` to explain `mpg` in the `mtcars` data set could have gone worse. The violations of the assumptions were quite small and some folks might find that thee transgressions to be ignorable. We should not take violations lightly, however, especially since there are steps we can take if our data isn't meeting the assumptions. Transformations are always an option but if we have data of a particular kind, e.g. binary data or count data, we can use generalized linear models instead.

## 4.4 Summary

Often we want to fit our data with a model to better understand our response variable or to try to predict new events. The ideal case (because it is common, widely used and easy to interpret) is that we can fit a linear model to the data. In order to do so, the relationship between the response variable and the explanatory variable(s) needs to be linear, we need to have constant variance, and the data needs to be independent.

If these condtions are not met, then there are steps we can take. One of which is transformations, we will not be going over that. The other is to fit a generalized linear model, which doesn't necessarily assume the data come from a normal distribution.

# Chapter 5

# Logistic Regression

We've asked you thus far to take our word regarding Generalized Linear Models (GLMs). In this chapter, we're going to take a look at a certain type of data that we know violates our assumptions: binomial data. Here we'll examine binomial data, see why ordinary least sqares falls apart, and consider some alternative methods (spoiler: it's going to be one of our generalized linear models).

## 5.1 What is Binomial Data?

### 5.1.1 Refresher: Bernoulli and Binomial

Bernoulli and Binomial random variables are some of the most important to consider, because they get at real the roots of representing the stata of the world. A Bernoulli random variable considers the simplest possible datum: whether something is, or it isn't. Numerically, we represent this as 1 or 0, just like computer data. And while I can use this to measure yes/no type data, such as "does a person own at least one pet?", an astute observer will realise the same idea can apply to a variety concepts including:

- success / failure,
- wins / losses,
- heads / tails,
- exists / doesn't exist
- has / doesn't have
- is a member of a group of interest / is not a member
- or even, was my prediction realized?

Often, whatever resulted in creating this single Bernoulli instance might be something that repeats. In fact, if we hope to apply statistics to it, there needs to be many instances. If we can assume a constant probability across all iterations of the events we're interested in, but want to ask questions about them in aggregate, that's Bernoulli Binomial (or just binomial for short). Some examples of questions that fall into binomial data would be:

- is this strategy effective (i.e. does it affect the rates of success or failure across many trials?)
- is this baseball team better than another (i.e. do they win more games, not just a game?)
- if I bet money on getting four heads in a row, what kind of odds makes that a good gamble?
- if a mailcarrier is moving to a new route, if I know how many homes are $\mu$-probable to have an angry dog, what's the liklihood that mail carrier will encounter an angry dog?
- if $\mu$-proportion of students have a medical condition I'm interested in, how many students will I need to talk to in order for it to be more likely than not I'll talk to x of them? Way more likely than not? Almost certain?
- if my predictions are p-percent accurate, and the cost of failure is m dollars, is it worth spending more money for more accurate predictions?

Hopefully you'll agree with me that questions of this type are actually extremely common.

### 5.1.2 Representing the Bernoulli distribution

Different texts use different language, so it's to explain how we'll be talking about Bernoulli and binomial distributions here. First, here's the /Bernoulli distribution/ probability mass function.

$$\mathcal{P}(y;\mu) = \mu^y(1-\mu)^{1-\mu},$$

where $y \in \{0,1\}, 0 \leq \mu \leq 1$.

This might be a bit different than what you've seen before, so lets talk about each variable.

- $\mathcal{P}$: (`\\mathcal\{P\}` in TeX) is the probability mass function of a certain outcome. $\mathcal{P}(y;\mu)$ then is the probability of event $y$ when the success has a probability of $\mu$.
- $y$: is the number of successes. Since this is a single trial, it can only be 0 or 1 (described by the statement $y \in \{0,1\}$).
  - Basically, it lets us choose whether we're predicting success or predicting failure all in one function.
  - If we're interested in the liklihood of an event not occuring $y = 0$, $\mu^0 = 1$, and the PMF simplifies to $\mathcal{P}(1,\mu) = 1-\mu$.
  - Conversely, if we're interetsed in an event occuring, $y-1$, $(1-\mu)^{(1-1)} = 1$, and the PMF simplifies to $\mathcal{P}(1,\mu) = \mu$.
- $\mu$: (`\mu` in TeX) is the probability of the event. But wait, isn't $\mu$ the mean, and also the expected value? **Yes, exactly!** Those are all the same in this PMF, so we use the same variable. Because many text books need

you to consider those ideas separatey so you can discover that relationship, you might see it recorded elsewhere as $p$, but since demonstrating that is beyond our scope here, we'll use $\mu$ to reduce complexity.

- It complicates using the formula elsewhere, but we could rewrite this as:

$$\mathcal{P}(y;\mu) = \begin{cases} \mu & \text{if } y = 1 \\ 1-\mu & \text{if } y = 0 \end{cases}, 0 \leq \mu \leq 1$$

### 5.1.3 Representing the binomial distribution

Intuitively, you likely recall that if $P(\text{heads on one coinflip}) = \frac{1}{2}$, then

$$P(\text{heads twice on two coinflips}) = P(\text{heads on one coinflip})^2 = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$$

. However, just like the above PMF accounts for yes and no, our binomial PMF $\mathcal{P}$ needs to account for yes and no, how many yesses we're looking for, and how many trials there are overall. So here, we'll use:

$$\mathcal{P}(y;\mu,m) = \binom{m}{my}\mu^y(1-\mu)^{m(1-y)}$$

where $y = 0, 1/m, 2/m, ..., 1, m$ is a positive integer, and $0 < \mu < 1$.

Like before, this format might be a bit different than what you may have seen elsewhere, so let's break it down.

- A reminder, $\binom{n}{k}$, (written `\binom{n}{k}` in TeX) is a combination $n$ choose $k$. More detail here.
- $\mathcal{P}$ (`\\mathcal\{P\}` in TeX) is the probability mass function (PMF) aka probability function. $\mathcal{P}(y;\mu,m)$ then is the probability function of a binomial distribution for event $y$ in the case of $m$ Bernoulli trials with each trial having a probability of $\mu$.
- $m$ here is simply the number of trials easy.
- $y$ here measures the *proportion* of successes (and actually it really is above as well–I'll get to that in a moment).
  - For example, if there are 4 trials and we want to see two successes, $y/m = 2/4 = 0.5$
  - This interacts interestingly in the combinatorial component. That $my$ will always equal the numerator of $y$. In the previous example then, the *choose* part of $m$ choose $my$ (aka $\binom{m}{my}$) will equal $\binom{4}{2}$.
- If $y = 0$, $\binom{m}{m0} = \binom{m}{0} = 1$ (any choose 0, like any number raised to 0 equals 1). So, looking at this formula, if I want to know the odds of not getting any heads on four coin flips, then
  -

$$\mathcal{P}(y;\mu,m) = \binom{m}{my}\mu^y(1-\mu)^{m(1-y)} = \binom{4}{4\times 0}\mu^0(1-\mu)^{4(1-0)} = 1\times1\times(1-\mu)^4$$

- So assuming a fair coin where $\mu = \frac{1}{2}$, then $\mathcal{P}(y; \mu, m) = \mu^4 = (\frac{1}{2})^4 = \frac{1}{16}$, which fits our expectations. Neat!

### 5.1.4 What does a bunch of binomial data look like then?

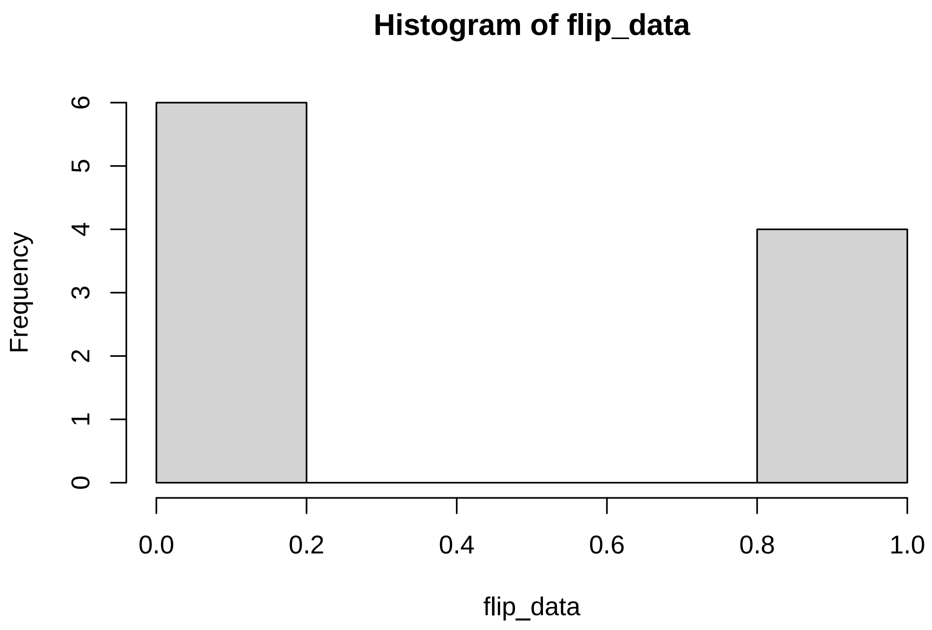
```
possibilities <- c(0,1) # either heads (1) or tails (0).
                        # c(...) makes a list of whatever you include in the ...
probabilities <- c(0.5, 0.5)
                        # a 50% chance of heads, a 50% chance of tails.
(flip_data = sample(x = possibilities, size = 10, replace=TRUE, prob=probabilities))
```

```
##  [1] 1 1 0 0 0 0 1 1 0 0
```

```
                        # x is the vector of possible outcomes.
                        # size is the number of coin flips I want.
                        # we are flipping coin, not drawing from a deck, so ne
                        #   second flip is just as likely to get heads or tails
                        #   as the first: we *do* replace.
                        # prob is a vector of probabilities for each x
                        # encasing the whole thing in parentheses shows the
                        #   value of the variable you set as you set it
```

So zeros and ones, we expect that (sorry if you're the 2 in $2^{10}$ readers who see all zeroes or ones). Lets check a simple histogram.

```
hist(flip_data)         # hist() generates a simple historam. We could make
```



```
                         # this prettier with ggplot2, but we don't need that right
                         # now.
```

## 5.2 Why doesn't OLS work for Binomial Data?

"Hey," you may be saying to the intense discomfort of those around you as you suddenly begin speaking to your book, "Yeah you–you're being intellectually dishonest!" The people around you register but do not address your sudden outburst. "Of course *that's* hard to fit–there is no meaningful relationship!" Acquaintances and strangers in earshot begin to gather their belongings closer to them and some plot in hushed tones to pay their check and depart. Perhaps a family pet or roommate registers this common occurance nonplused and proceeds with their day.

Stop making a scene. You're right, fine. Maybe this will help. Picture five dice of different numbers of sides: 4, 6, 8, 12, and 20 sides. I want to know if the number of sides increases the liklihood of getting a 4 or above. Does the likelihood or getting a 4 or above increase with the number of sides? Our (and hopefully your) intuitive assumption is that yes. Obviously. So lets try to examine some data.

```
# here I'll write a function and then use the map function to apply that
# situation multiple times.This is a great technique for simulation and,
# once you're acquainted with it, can lead to more efficient and/or easier#
# to read code than complicated nested for-loops.

# first, we define a simple function that returns a 1 if a roll of a simulated
# die with a variable number of 'sides' equals a target number 'tn' or higher.

library(tidyverse)

simulate_die_roll <- function(sides = 4, tn = 4) { # this starts describing
                                                   # the function and creates
                                                   # two parameters, sides
                                                   # and tn, which will be set
                                                   # to 4 but can be changed
                                                   # when the function is called
  roll <- sample(1:sides, size = 1) # roll a die, size 1 to 'sides'.
  if(roll >= tn){1} else {0} # if roll >= target number, then return 1,
                             # otherwise, return 0.
}

# with this simple (and poorly error-proofed) function defined, we can use it
# to sample some data.
```

```
dice <- c(4, 6, 8, 12, 20) # gather our dice
results <- 0 # blank the results variable, in case I was sloppy elsewhere
nsimsperdie <- 10 # number of rolls per die

(d4 <- unlist(map(1:nsimsperdie, ~ simulate_die_roll(sides = 4, tn=4))))
```

```
##  [1] 0 0 0 1 0 0 0 0 0 1
```

```
(d6 <- unlist(map(1:nsimsperdie, ~ simulate_die_roll(sides = 6, tn=4))))
```

```
##  [1] 1 1 0 1 1 1 1 0 1 0
```

```
(d8 <- unlist(map(1:nsimsperdie, ~ simulate_die_roll(sides = 8, tn=4))))
```

```
##  [1] 0 0 1 0 1 1 0 0 0 0
```

```
(d12 <- unlist(map(1:nsimsperdie, ~ simulate_die_roll(sides = 12, tn=4))))
```

```
##  [1] 1 0 1 1 1 1 1 1 1 1
```

```
(d20 <- unlist(map(1:nsimsperdie, ~ simulate_die_roll(sides = 20, tn=4))))
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```

## 5.3 Link functions we can use

## 5.4 ED50

# Chapter 6

# Poisson Regression

After learning about one type of GLM using Binomial data in the previous section, this chapter will explore another common type of data where GLMs are used - Poisson data.

## 6.1 What is Poisson data?

To begin, we will start with the types of Poisson data that you may come across. Poisson data occurs when there is a need to model counts. Count data appears often in real world applications, so modeling this data accurately is an important skill to have. The counts of events being modeled should be independent and the upper limit of the counts should be much greater than the majority of the counts - this limit may not even exist. Possion GLMs can also be used to model rates when a Binomial GLM would not be appropriate. This occurs when populations are large and the rate of occurance is very small, usually less than 1%. The Poisson distribution has probability function P(y|$\mu$) = $e^{-\mu}\mu^{y}/y!$ for y = 0, 1, 2,... and $\mu > 0$.

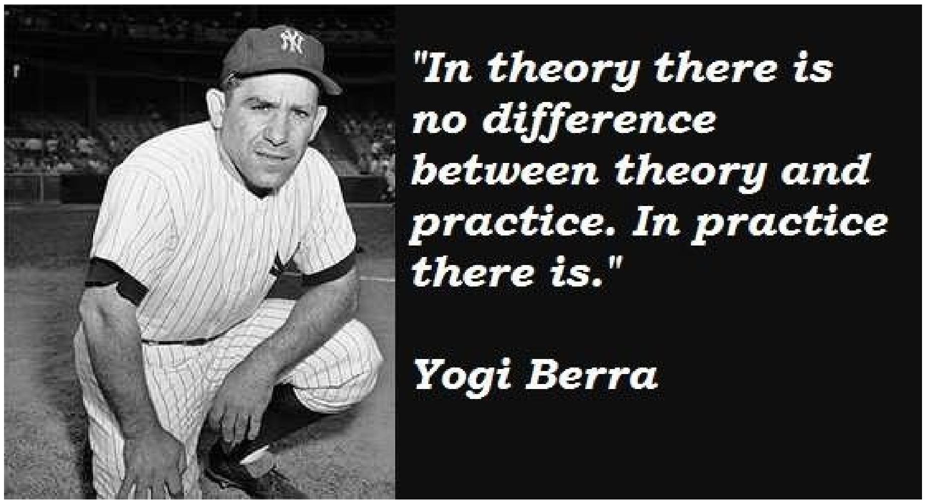
## 6.2 Why ordinary least squares does not work for Poisson data

Similar to other types of GLMs that you have seen previously, ordinary least squares (OLS) regression cannot be used for Poisson data. Count data violates the constant variance assumption of OLS because as the counts get smaller and approach 0, the variance of the response must decrease, and as the counts grow larger, the variance of the response will tend to increase. The normal distribution is also not adequate for modeling the random component of count data because counts are both non-negative and discrete. The Poisson distribution is a more suitable option to model count data.

## 6.3 Link functions for Poisson GLM's

The most common link function used for Poisson regression is the logrithmic link function because it guarentees that $\mu$ will be greater than 0. It also makes interpretation of the regression parameters more straightforward, since they are interpreted as having multiplicative effects. The systematic component of a Poisson GLM is $\mu = e^{\beta_0+\beta_1 x_1+\ldots+\beta_p x_p}$ and increasing $x_j$ by one will increase $\mu$ by a factor of $\beta_j$. Other link functions, while less commmon, can be used for a Poisson GLM. The identity link function, where $\eta = \mu$, or the sqrt link function, where $\eta = \sqrt{\mu}$, are two additional options. When all of the explanatory variables are quantitative, the data are modeled by a Poisson regression model using one of the three previous link functions. Additionally, quantile residuals should be used to asses the model fit since the data are discrete. Count data can also be represented in contingency tables so that observations can be cross-classified in their respective categories. These data are modeled using a log-linear model, where all of the explanatory variables are qualitative.

## 6.4 Poisson Example


"In theory there is no difference between theory and practice. In practice there is."
Yogi Berra

Now that we have had a chance to go over some of the characteristics of Poisson GLMs, let's look at an example involving baseball data. The dataset that we will use is a subset of data from the Lahman package, which is an extensive collection of baseball statistics since the 1800s. This package is an excellant resource for fellow baseball fans to practice fitting various GLM models, including Binomial regressions. The dataset that we will be focusing on is the batting statistics from every World Series during the liveball era. If you do not have the following packages, please take a moment to install them using the install.packages command. To start, we will need to load the Lahman package and the tidyverse package to access our World Series data, which is a subset of the Postseason dataset:

```
require(Lahman)
```

```
## Loading required package: Lahman
```

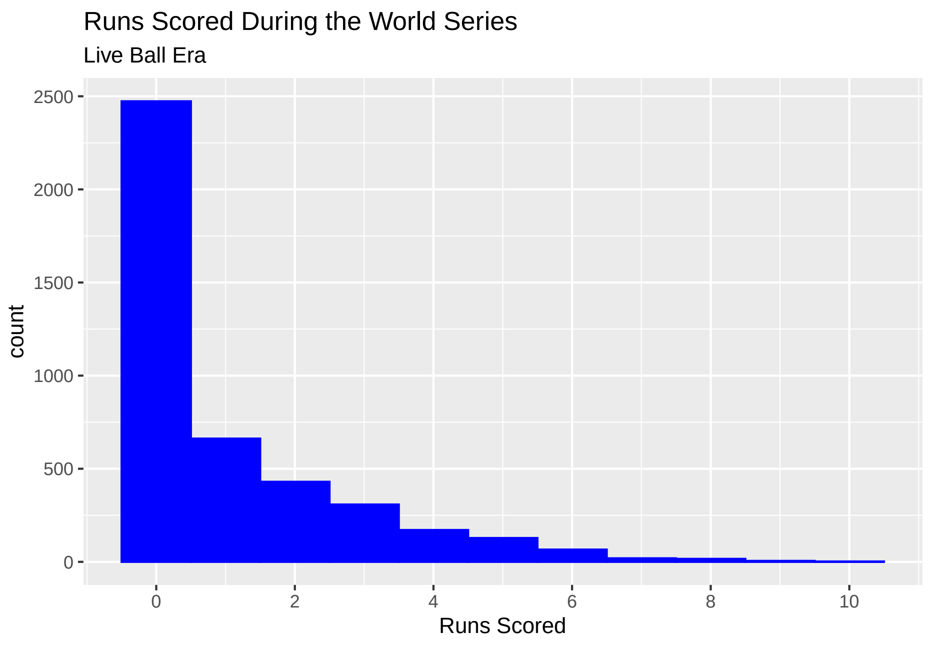
```
require(tidyverse)
data(BattingPost)
```

Now we can manipulate the data to focus only on the World Series batting statistics from the liveball era:

```
liveball_ws <- BattingPost %>%
  filter(yearID >= 1920) %>%
  filter(round == "WS")
```

Our response variable is going to be the runs scored by each player during each World Series from the liveball era. Initially, we can imagine that this will be a good canidate for Poisson regresssion because runs are a quantitative count. The number of runs scored by each player is also probably 0 to 1 typically, but there will be cases where a higher number of runs are occasionally scored and technically there is no upper limit to how many runs that is. Let's graph the response variable and see if this seems to be the case:

```
ggplot(data = liveball_ws, aes(x = R)) +
  geom_histogram(color = "blue", fill = "blue", binwidth = 1) +
  xlab("Runs Scored") +
  labs(title = "Runs Scored During the World Series",
       subtitle = "Live Ball Era") +
  scale_x_continuous(breaks = c(0, 2, 4, 6, 8, 10))
```



This indeed looks like a great candidate for Poisson regression. We could even imagine overlaying a Poisson distribution with a $\mu$ parameter of about 1 and see that it follows our response variable well. In fact, the mean number of runs scored in this dataset is 1.04, which would be the estimated $\mu$ value. Let's now fit a model using batting-related outcome variables as our explanatory variables. We will model runs scored using the number of hits, doubles, triples, and home runs a player recored in each World Series:

```
runmod <- glm(R ~ H + X2B + X3B + HR, data = liveball_ws, family = "poisson")
summary(runmod)
```

```
##
## Call:
## glm(formula = R ~ H + X2B + X3B + HR, family = "poisson", data = liveball_ws)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.7850  -0.8260  -0.8260   0.4824   3.8405
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.075579   0.027460 -39.170  < 2e-16 ***
## H            0.253262   0.005771  43.884  < 2e-16 ***
## X2B          0.044604   0.017234   2.588  0.00965 **
## X3B          0.087291   0.036760   2.375  0.01757 *
## HR           0.266188   0.015666  16.991  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 9503.5  on 4279  degrees of freedom
## Residual deviance: 4161.1  on 4275  degrees of freedom
## AIC: 8949.7
##
## Number of Fisher Scoring iterations: 5
```

All of the variables here are significant at the standard .05 significance level, so they are all important for modeling the number of runs scored. Another point of interest to note from the model summary output is that the residual deviance of 4119 is smaller than its degress of freedom 4225 and much smaller than the null deviance of 9383. We will discuss why we should look at the residual deviance versus its degrees of freedom in the next section, but for now, the key takeaway is that the Poisson distribution was a good choice to model our data. Additionally, compared to our higher null deviance, our model does a good job of explaining the variability of runs scored. If baseball is a topic that interests you, I would encourage you to download the Lahman package and see if you can improve on this model by lowering the AIC.

## 6.5 Problems of overdispersion and solutions

A characteristic of the Poisson distribution is that the mean $\mu$ equals the variance. However, when modeling real data, this assumption can be a bit $fishy$, since the variance is usually greater than $\mu$, resulting in overdispersion. Overdispersion occurs because the events being counted have a positive correlation and are not completely independent. We did not have to worry about overdispersion in our previous example, but you will likely come across it in other datasets. Overdispersion is an issue for Poisson GLMs because the standard errors of the $\beta_j$ estimates will be underestimated, making the model's explanatory variables appear more significant than they actually are. Models fit using the Poisson family can be checked for overdispersion by comparing the residual deviance to the residual degrees of freedom. If the residual deviance is greater than the degrees of freedom, the model is overdispersed, and if the residual deviance is much less than the degrees of freedom, the model is underdispersed, a less frequent occurence. A large residual deviance compared to the residual degrees of freedom could also indicate a lack of fit, but this can be checked by eliminating outliers and fitting the model with the most explanatory variables possible. The residual deviance will still be large if overdispersion is the issue. Similarly, the Pearson goodness-of-fit statistic can be compared to the residual degrees of freedom to check for overdispersion. If the counts are small, so asymptotic approximations might not be accurate, large goodness-of-fit statistics generally indicate a poor model fit.

If there is overdispersion after fitting a Poisson GLM, the model can be fit using other related families. By modeling the data using a hierarchical model to add more variability, $\mu$ can be treated as a random variable, resulting in the response variable of interest following a negative binomial distribution. The expectation of $y_i$ is $\mu_i$ and the variance of $y_i$ is now $\mu_i + \psi\mu_i^2$, where $\psi\mu_i^2$ is the added overdispersion term and $\psi$ is larger when overdispersion is greater. The MASS package contains the function glm.nb() to model negative binomial GLMs. The standard link function is the log-link function so that $\mu > 0$, and similar to the Poisson GLM, quantile residuals should be used. Quasi-Poisson models are another alternative to Poisson GLMs when there is overdispersion. The variance of $y_i$ is now $\phi\mu_i$, where $\phi > 1$ represents overdispersion and raises standard errors to a factor of $\sqrt{\phi}$. Quasi-Poisson models can be fit in R by family = quasipoisson() using the usual glm function. Quantile residuals cannot be found because the quasi-Poisson model is not a probability model but standardized deviance residuals can be examoned.

# Bibliography

Dunn, P. K. and Smyth, G. K. (2018). *Generalized linear models with examples in R*. Springer.