Generalised Linear Models

Jarrod Hadfield

University of Edinburgh

Generalised Linear Models

Jarrod Hadfield

University of Edinburgh

OK - yesterday we looked in detail at the linear model and hopefully got a good feel for the form of the model and how it works.

Linear Model

- Makes no assumption about the distribution of the predictors.

Generalised Linear Model
Linear Model
Makes no assumption about the distribution of the predictors.

There were two important points about the linear model that I did not make explicit, but are important to understand. The first is that we make no assumptions about the distribution of the predictors and this is because we are modelling the distribution of the response variable *conditional* on the set of predictor variables we have measured.

Generalised Linear Model

Linear Model

- Makes no assumption about the distribution of the predictors.
- Does assume that the response is normally distributed *conditional on the model*.

Generalised Linear Model

Linear Model

- Makes no assumption about the distribution of the predictors.
- Does assume that the response is normally distributed *conditional on the model*.

The second point is that I was often quite lazy in saying that the linear model assumes that our response variable is normally distributed, but what I should have said is that *conditional* on the model our response variable is normally distributed, which people usually interpret as saying that we assume the residuals from the model are normally distributed, not the response itself. This interpretation is fine when the response is conditionally normal, but not for other types of distribution. To get a better understanding of what I mean by *conditional* it is necessary to understand three aspects of a multivariate distribution; the joint, marginal and conditional distribution.

Generalised Linear Model

Linear Model

- Makes no assumption about the distribution of the predictors.
- Does assume that the response is normally distributed *conditional on the model.*

	Y_1	Y_2	Y_3
P_1	0.30	0.15	0.05
P_2	0.15	0.10	0.05
P_3	0.05	0.05	0.10

Joint Distribution

Generalised Linear Model

Linear Model

- Makes no assumption about the distribution of the predictors.
- Does assume that the response is normally distributed *conditional on the model*

	Y_1	Y_2	Y_3
P_1	0.30	0.15	0.05
P_2	0.15	0.10	0.05
P_3	0.05	0.05	0.10

Joint Distribution

Let's imagine we have two random variables Y and P and for simplicity lets say they can only take one of three outcomes. These entries are the probability of observing a particular combination, and so the probability of observing Y_1 and P_1 is 0.3. These numbers characterise the joint distribution - they tell us the probability of observing particular combinations of the two variables. The ellipses I plotted in Lecture 2 (for the sampling distribution of two `lm` coefficients) are graphical representations of the joint distribution for a pair of continuous variables.

Generalised Linear Model

Linear Model

- Makes no assumption about the distribution of the predictors.
- Does assume that the response is normally distributed *conditional on the model.*

	Y_1	Y_2	Y_3
P_1	0.30	0.15	0.05
P_2	0.15	0.10	0.05
P_3	0.05	0.05	0.10

Joint Distribution

$Pr(Y)$	0.50	0.30	0.20

Marginal Distribution

Generalised Linear Model

Linear Model

- Makes no assumption about the distribution of the predictors.
- Does assume that the response is normally distributed *conditional on the model*

	Y_1	Y_2	Y_3
P_1	0.30	0.15	0.05
P_2	0.15	0.10	0.05
P_3	0.05	0.05	0.10

Joint Distribution

$Pr(Y)$	0.50	0.30	0.20

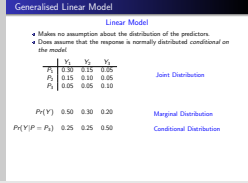
Marginal Distribution

The marginal distribution of a variable is its probability distribution ignoring the other variable, and we can obtain this for Y by adding up the probabilities in each column.

Linear Model

- Makes no assumption about the distribution of the predictors.
- Does assume that the response is normally distributed *conditional on the model.*

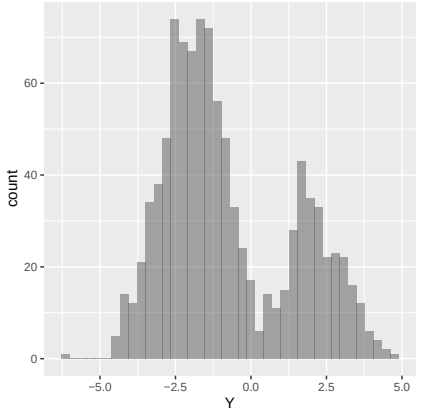
	Y_1	Y_2	Y_3	
P_1	0.30	0.15	0.05	Joint Distribution
P_2	0.15	0.10	0.05	
P_3	0.05	0.05	0.10	
$Pr(Y)$	0.50	0.30	0.20	Marginal Distribution
$Pr(Y\|P = P_3)$	0.25	0.25	0.50	Conditional Distribution



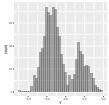
We can also define the probability distribution of Y conditional on some value of P, so here we have the probability of Y given that $P = P_3$. We've taken the final row of the joint probability distribution and rescaled the probabilities so they add up to one. In a linear model, when we say our response is normal, what we mean is that for a given value of the predictor variable, lets say P_3 we expect the possible outcomes for Y to have a probability distribution described by a normal distribution.

Linear Model

- Makes no assumption about the distribution of the predictors.
- Does assume that the response is normally distributed *conditional on the model.*



Generalised Linear Model
Linear Model
Makes no assumption about the distribution of the predictors.
Does assume that the response is normally distributed *conditional on the model*

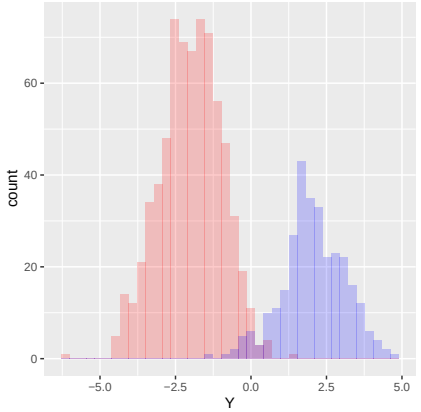


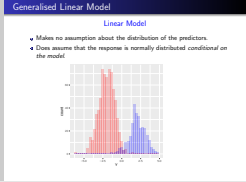
For example, are these data normally distributed? Clearly not, amongst other things the distribution is clearly bimodal, whereas the normal has a single mode. However, if we knew that these data were collected from two groups, males and females for example,

Generalised Linear Model

Linear Model

- Makes no assumption about the distribution of the predictors.
- Does assume that the response is normally distributed *conditional on the model.*





and we looked at the distribution of the data conditional on this information, we might be prepared to say that the conditional distribution of the data are close to being normal and this satisfies the assumption of the linear model. In the absence of knowledge about which observations were from males and which from females, we wouldn't be able to condition on sex and we would be left with a bimodal distribution (the marginal distribution with respect to sex) that would not meet the assumptions of the linear model.

Generalised Linear Model

Generalised Linear Model

- Assumes that the response follows some *specified* distribution *conditional on the model.*

Generalised Linear Model

Generalised Linear Model

- Assumes that the response follows some *specified* distribution *conditional on the model.*

Next, we're going to relax the assumption that our response variable is normally distributed (or conditionally normal) using something called generalised linear models, or GLMs.

Generalised Linear Model

Generalised Linear Model

- Assumes that the response follows some *specified* distribution *conditional on the model.*

- Normal
- Bernoulli
- Binomial
- Multinomial
- Poisson
- Gamma

Generalised Linear Model

Generalised Linear Model

- Assumes that the response follows some *specified* distribution *conditional on the model.*

- Normal
- Bernoulli
- Binomial
- Multinomial
- Poisson
- Gamma

These models allow data to come from these distributions here. The normal we've seen.

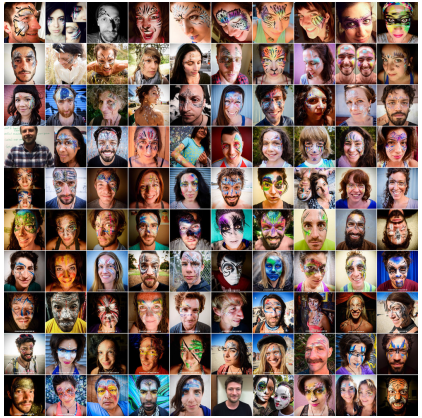
- The Bernoulli you are familiar with - the outcomes are zeros and ones like tosses of a coin,
- and the binomial you're probably familiar with too: it's the same as the Bernoulli but for each observation you flip the coin multiple times for each observation and count the number of heads versus tails (or successes versus failures). For example it might be the number of male and female offspring in a nest and I have collected data on multiple nests.
- The multinomial, which you're probably less familiar with is the same as the binomial but rather than having two possible outcomes such as heads or tails, male or female, you have three possible outcomes: Coke, Pepsi and Irn-bru, for example.
- The Poisson distribution we'll come to, its a distribution that results in count data, and the
- gamma distribution (which we'll touch on briefly) is for continuous data that takes on positive values and has a tail to the right. It's useful for modelling things like waiting times - how long do I have to wait before some event happens, like a bus arrives or the Grim Reaper.

The distributions that are here are from the exponential family. You don't need to know what this means, the key thing is that these distributions have some properties that mean we can obtain maximum likelihood estimates in a relatively straightforward way - nearly in the same way that we obtained maximum likelihood estimates when we assumed the model to be Gaussian. This used to matter a lot when computers were crap but now we have access to fast computers and good optimisation algorithms and so the range of distributions we can accommodate in GLM-like models goes beyond these distributions. Technically, a generalised linear model only allows these distributions but actually the general structure of the model can be applied to a much broader range of distributions and it is the general structure of the model I want to focus on.

Poisson

Let's start with the Poisson. If people have count data they generally start by assuming their data are Poisson. Count data don't have to be from a Poisson distribution any more than continuous data must come from a normal distribution, but like the normal it often seems to work well if we add a bit of flexibility.

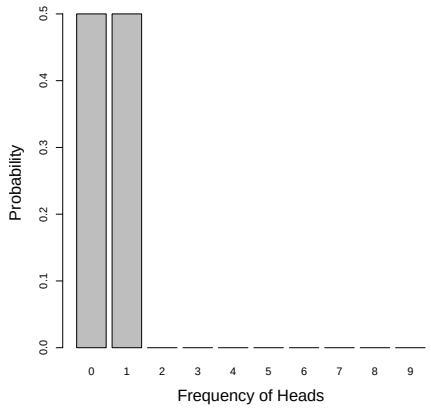
Poisson



Poisson

So how does the Poisson distribution arise? Imagine I dumped you at a large festival and asked you to wander around until you'd bumped into x number of people (100 lets say) and of those people you bumped into, I asked you to record the number of people you recognised. Let's say that I asked each of you to do this and for each of you the probability of bumping into someone you knew was the same. The distribution of how many people you knew and how many people you didn't know would be binomial with the probability being the chance that you knew some one. In the data we saw in the practical, I had set up many arrays of photos like this and asked people in IEB to say how many people they recognised. Each array was different, but on average two of the photos were of people from IEB.

Poisson



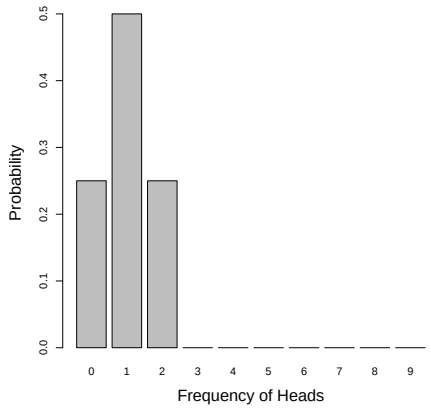
Probability=0.5

Poisson

Probability=0.5

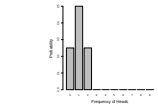
In Lecture 1 I showed you that if you take the average, or the sum, of a large collection of numbers it starts to look like a Gaussian pretty quickly irrespective of what distribution the underlying numbers come from. The example I gave was coin flipping - so a Bernoulli variable of zero and one where the probability of a one is 0.5. With one flip we have two outcomes,

Poisson



Probability=0.5

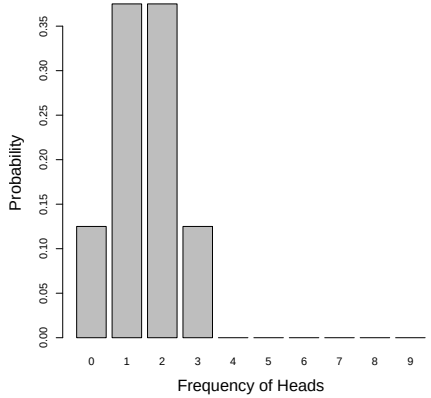
Poisson



Probability=0.5

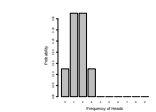
with two flips we have three outcomes

Poisson



Probability=0.5

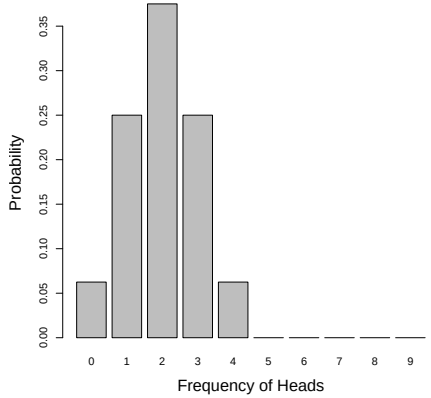
Poisson



Probability=0.5

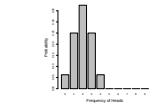
and so on ...

Poisson



Probability=0.5

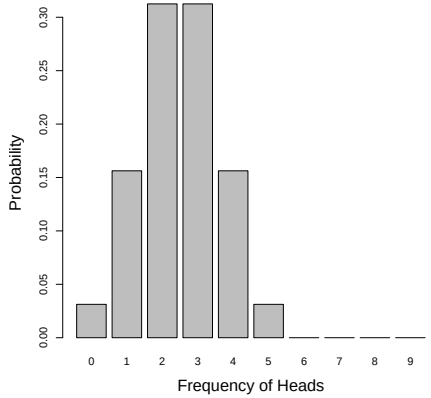
Poisson



Probability=0.5

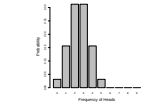
and so on ...

Poisson



Probability=0.5

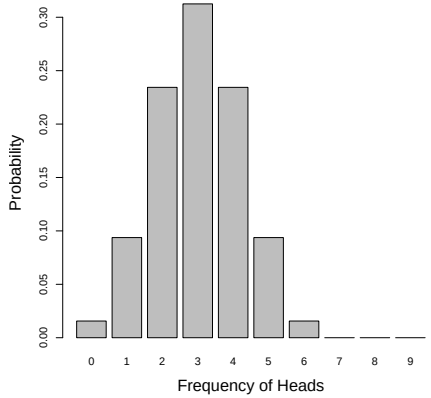
Poisson



Probability=0.5

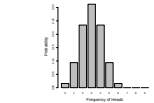
and so on ...

Poisson



Probability=0.5

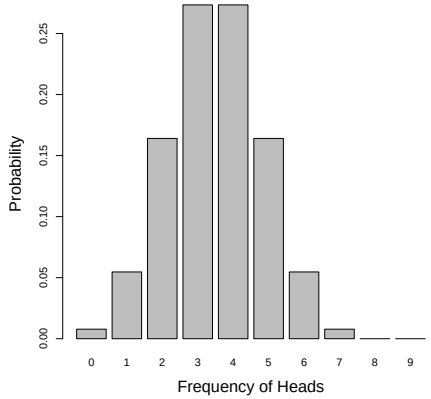
Poisson



Probability=0.5

and so on ...

Poisson



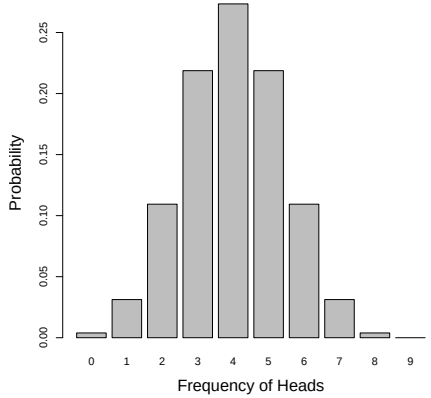
Probability=0.5

Poisson

Probability=0.5

and so on ...

Poisson



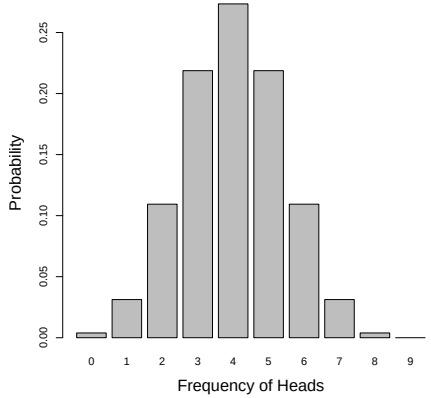
Probability=0.5

Poisson

Probability=0.5

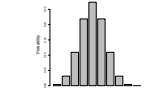
and so on ...

Poisson



Probability=0.1

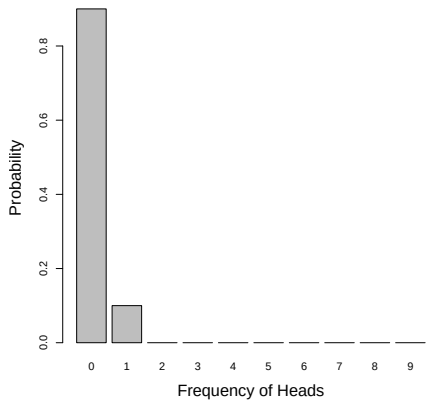
Poisson



Probability=0.1

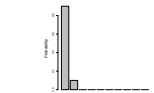
Now let's imagine that the probability was 0.1 instead of 0.5 and we redid the same process.

Poisson



Probability=0.1

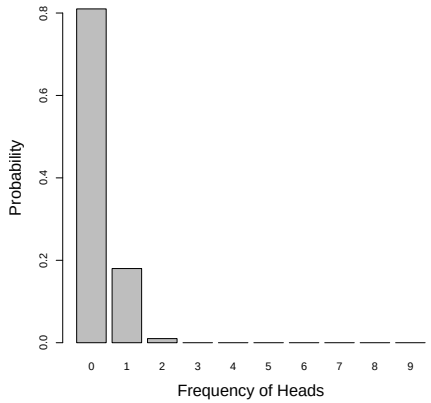
Poisson



Probability=0.1

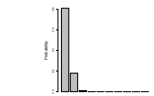
If we flipped the coin once there are two outcomes - a zero with 90% and one with 10%.

Poisson



Probability=0.1

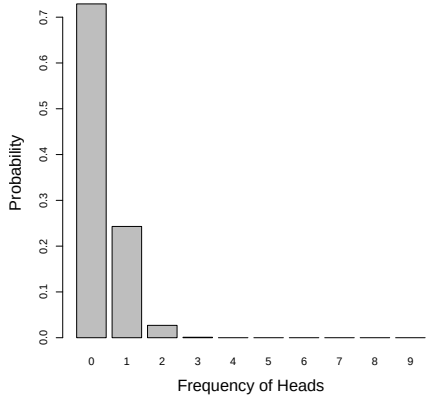
Poisson



Probability=0.1

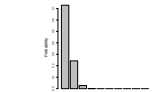
If we flipped the coin twice there are three outcomes.

Poisson



Probability=0.1

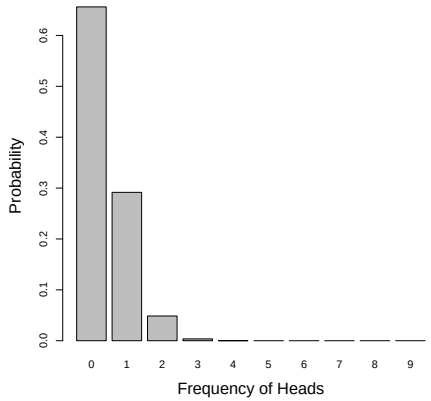
Poisson



Probability=0.1

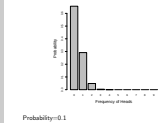
and so on.

Poisson



Probability=0.1

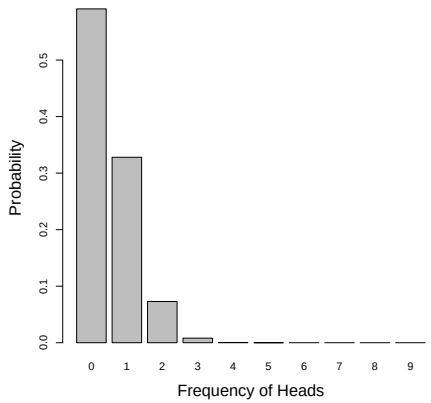
Poisson



Probability=0.1

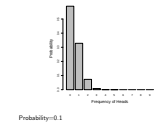
and so on.

Poisson



Probability=0.1

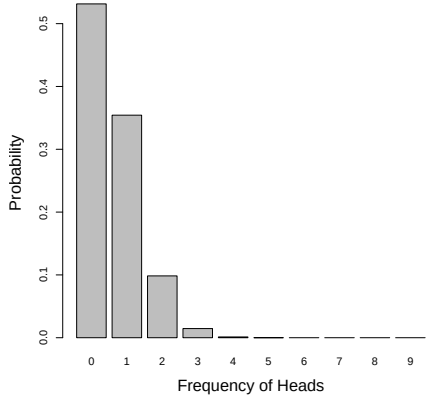
Poisson



Probability=0.1

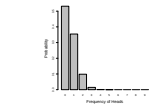
and so on.

Poisson



Probability=0.1

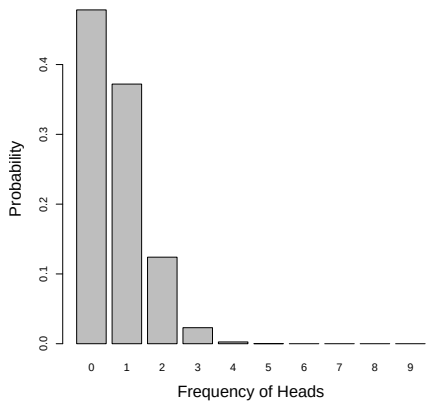
Poisson



Probability=0.1

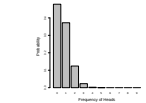
and so on.

Poisson



Probability=0.1

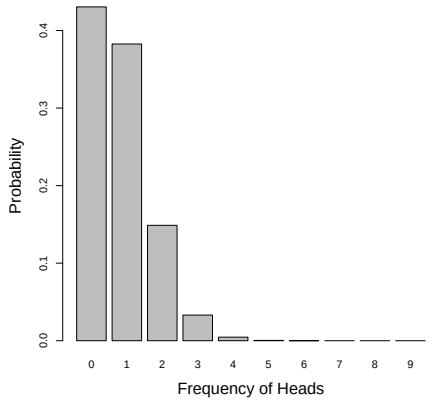
Poisson



Probability=0.1

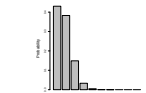
and so on.

Poisson



Probability=0.1

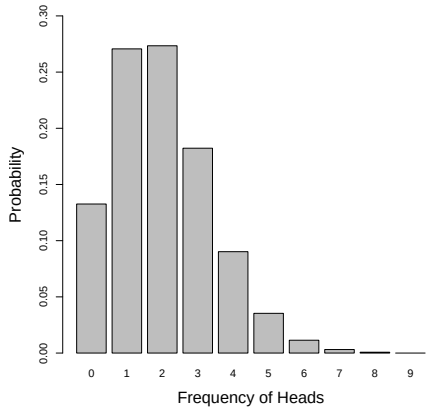
Poisson



Probability=0.1

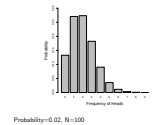
It doesn't look very normal yet, but if we flipped the coin a greater number of times it would eventually look like a normal distribution. If the probability was even smaller it would take an even larger number of flips before the distribution started to look normal.

Poisson



Probability=0.02, N=100

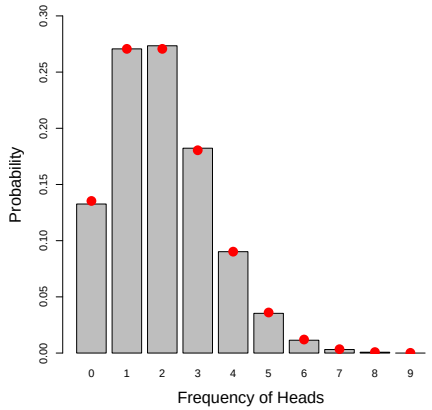
Poisson



Probability=0.02, N=100

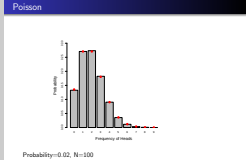
If our example of the photos, the number of trials is quite large (100). However, despite 100 trials we're are still quite far from a normal because the probability of a success (recognising someone) is so small (2/100=0.02). The Poisson distribution assumes that this probability is very very very small and the number of trials is very very large and is parameterised by the expected number of successes (2 in this instance; n multiplied by the probability).

Poisson



Probability=0.02, N=100

Poisson



Probability=0.02, N=100

These red points are the probabilities for each frequency obtained from the Poisson mass function with mean (or rate) 2 (In R you would use the function `dpois` with `mean=2` to obtain these numbers). You can see that although the frequencies are generated under a binomial distribution with these parameters (p=0.02, n=100) the Poisson distribution provides a good fit even though the probability isn't very very small and the number of trials isn't very very large.

So the Poisson is the number of successes you expect when the chance of something happening is very unlikely but you've had a large number of goes. The number of new mutations in a genome would be a good example ($p = 10^{-8}$ and $n = 6 \times 10^{-9}$), but as we've seen the Poisson distribution can still recapitulate the distribution of successes very well even when p and n are not so extreme.

Generalised Linear Model

$$E[\mathbf{y}] = \mathbf{X}\beta$$

Generalised Linear Model

$E[\mathbf{y}] = \mathbf{X}\beta$

OK - so before we constructed a model for the mean of a Gaussian, and I represented it in a compact form in terms of matrices. Now we could do the same for the mean of our counts, but we have a problem. With out some constraints on the parameters our model could predict negative numbers and that would be a logical impossibility.

Generalised Linear Model

$$\log(E[\mathbf{y}]) = \quad \mathbf{X}\beta$$

Generalised Linear Model

$\log(E[\mathbf{y}]) = \mathbf{X}\beta$

What we can do is we can say, rather than predicting the mean, our linear model could predict some function of the mean, let's say the log of the mean. This would solve our problem because if we predict a negative number on the log scale when we put it back on the arithmetic scale we have a positive number.

Generalised Linear Model

- Link function: log

$$\log(E[\mathbf{y}]) = \quad \mathbf{X}\beta$$

Generalised Linear Model

- Link function: log

$\log(E[\mathbf{y}]) = \mathbf{X}\beta$

This transformation is called the link function, in this particular case our link function was the log transform, but we're quite free to choose any transformation we like. Each distribution comes with its own 'preferred' transform, so the preferred transform for the Poisson distribution is the log transform.

Generalised Linear Model

- Link function: log

$$\begin{aligned} \log(E[\mathbf{y}]) &= \mathbf{X}\beta \\ E[\mathbf{y}] &= \log^{-1}(\mathbf{X}\beta) \end{aligned}$$

Generalised Linear Model

- Link function: log

$$\begin{aligned} \log(E[\mathbf{y}]) &= \mathbf{X}\beta \\ E[\mathbf{y}] &= \log^{-1}(\mathbf{X}\beta) \end{aligned}$$

If we want we can also take the inverse of this function so that we can think about predicting the expectation once again.

Generalised Linear Model

- Link function: log

$$\begin{aligned} \log(E[\mathbf{y}]) &= \mathbf{X}\beta \\ E[\mathbf{y}] &= \log^{-1}(\mathbf{X}\beta) \\ E[\mathbf{y}] &= \exp(\mathbf{X}\beta) \end{aligned}$$

Generalised Linear Model

- Link function: log

$$\begin{aligned} \log(E[\mathbf{y}]) &= \mathbf{X}\beta \\ E[\mathbf{y}] &= \log^{-1}(\mathbf{X}\beta) \\ E[\mathbf{y}] &= \exp(\mathbf{X}\beta) \end{aligned}$$

The inverse of logging is exponentiating. So we take our linear model to get some prediction and then we exponentiate that prediction to get the expected number of counts.

Generalised Linear Model

- Link function: log

$$\begin{aligned} \log(E[\mathbf{y}]) &= \mathbf{X}\beta \\ E[\mathbf{y}] &= \log^{-1}(\mathbf{X}\beta) \\ E[\mathbf{y}] &= \exp(\mathbf{X}\beta) \end{aligned}$$

- Distribution: Poisson

$$\mathbf{y} \sim Pois(\exp(\mathbf{X}\beta))$$

Generalised Linear Model

- Link function: log

$$\begin{aligned} \log(E[\mathbf{y}]) &= \mathbf{X}\beta \\ E[\mathbf{y}] &= \log^{-1}(\mathbf{X}\beta) \\ E[\mathbf{y}] &= \exp(\mathbf{X}\beta) \end{aligned}$$

- Distribution: Poisson

$$\mathbf{y} \sim Pois(\exp(\mathbf{X}\beta))$$

The second thing we need to do is specify a distribution other than the normal, and we're going to use the Poisson. Note that we have not specified a variance here like we did with the normal. The Poisson distribution only has a single parameter, the mean, and the variability around the mean is equal to the variance.

Linear Model

```
> data(Traffic, package = "MASS")
```

A Swedish Experiment: On some days make everyone drive to the speed limit on others let everyone drive as fast as they want. Count how many accidents there are.

Linear Model

> data(Traffic, package = "MASS")
A Swedish Experiment: On some days make everyone drive to the speed limit on others let everyone drive as fast as they want. Count how many accidents there are.

We'll start with a lovely little experiment done by the Swedish government in the early 60's. It was very simple, in 1961 they made everyone drive to the speed limit on some days, and then on other days they let everyone drive as fast as they wanted, and then they just counted how many accidents happened. Then the next year they repeated the experiment just to make sure. It makes you wonder what they were doing to their lab mice during that period.

Linear Model

```
> data(Traffic, package = "MASS")
```

A Swedish Experiment: On some days make everyone drive to the speed limit on others let everyone drive as fast as they want. Count how many accidents there are.

```
> Traffic[c(1, 2, 184), ]
    year day limit  y
1   1961   1    no  9
2   1961   2    no 11
184 1962  92   yes  9
```


Linear Model

> data(Traffic, package = "MASS")
A Swedish Experiment: On some days make everyone drive to the speed limit on others let everyone drive as fast as they want. Count how many accidents there are.

> Traffic[c(1, 2, 184),]
year day limit y
1 1961 1 no 9
2 1961 2 no 11
184 1962 92 yes 9

This is the data frame for the 1st, 2nd and last observation. In total the number of accidents (y) was collected on 92 days in each year, where day 1 or day 50 in 1961 is comparable to day 1 or day 50 in 1962: it's the same day of the week, the same month and so on. And some days there was a speed limit "yes" and some days there wasn't.

Generalised Linear Model

```
> traffic_m1 <- glm(y ~ limit + year + day, data = Traffic)
```

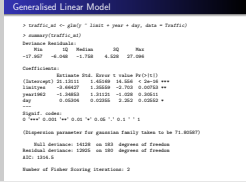

Generalised Linear Model

> traffic_m1 <- glm(y ~ limit + year + day, data = Traffic)

Let's start by fitting a standard Gaussian model. We're going to use the `glm` function rather than the `lm` function but it fits the same model since the default argument for the distribution is normal. You should be familiar with most of the output.

Generalised Linear Model

```
> traffic_m1 <- glm(y ~ limit + year + day, data = Traffic)

> summary(traffic_m1)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-17.957   -6.048   -1.758    4.528   27.096

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 21.13111    1.45169  14.556  < 2e-16 ***
limityes    -3.66427    1.35559  -2.703  0.00753 **
year1962    -1.34853    1.31121  -1.028  0.30511
day          0.05304    0.02355   2.252  0.02552 *
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 71.80587)

    Null deviance: 14128  on 183  degrees of freedom
Residual deviance: 12925  on 180  degrees of freedom
AIC: 1314.5

Number of Fisher Scoring iterations: 2
```

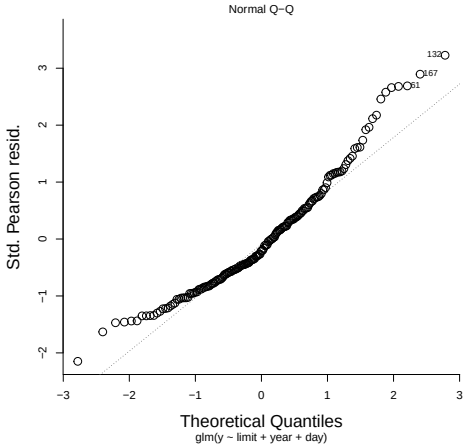
On day 0 of 1961 in the absence of a speed limit we expected 21 road accidents. Had there been a speed limit we would have expected about 3.7 less accidents (about a 17% reduction) and had it been 1962 rather than 1961 we would have expected 1.3 less accidents. It seems unlikely that the true effect of a speed limit was zero, but it could well be that there was no systematic difference between the years and that the 1.3 fewer accidents in 1962 was just the result of chance. The number of accidents rose over the course of the year such that by day 92 there was about 4.9 (92*0.05) more accidents per day than at the start of the year, and it seems possible that this is not due to chance alone.

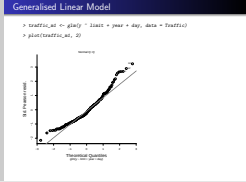
One difference between the functions `glm` and `lm` is that the R^2 of the model is not reported and instead of the residual standard deviation (called residual standard error in `lm`) we get something called the dispersion. This is the residual variance. If we take its square root we get the standard deviation which is about 8 accidents so the number of accidents on any given day is unlikely to deviate from it's expectation by more than twice this ($\pm$17 accidents).

Very good - we could stop there. But, we assumed the data were conditionally Gaussian and we never checked whether this assumption is valid or not.

Generalised Linear Model

```
> traffic_m1 <- glm(y ~ limit + year + day, data = Traffic)
> plot(traffic_m1, 2)
```

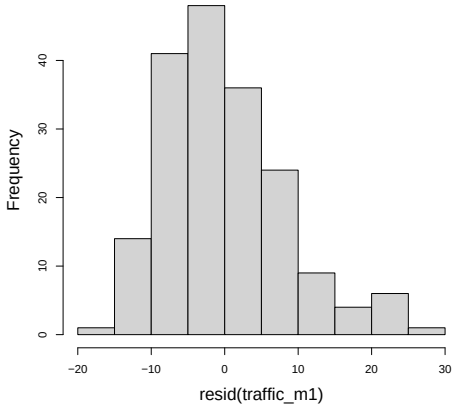




Let's have a look at a diagnostic qq-plot. It's not perfect, but I've seen a lot worse. There's a deficit of small values and quite a large excess of large values: the residuals seem to be right skewed. We can see this

Generalised Linear Model

```
> traffic_m1 <- glm(y ~ limit + year + day, data = Traffic)
> hist(resid(traffic_m1))
```



Generalised Linear Model
> traffic_m1 <- glm(y ~ limit + year + day, data = Traffic)
> hist(resid(traffic_m1))

by plotting a histogram of the residuals - the right skew is evident. To be honest, I wouldn't be overly worried about this model but we can do a bit more work to see if we can improve it.

Generalised Linear Model

```
> traffic_m2 <- glm(log(y) ~ limit + year + day, data = Traffic)
```

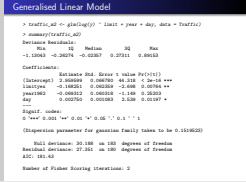

Generalised Linear Model

> traffic_m2 <- glm(log(y) ~ limit + year + day, data = Traffic)

Right skew is natural for count data where negative values do not exist, but it can also be quite a common feature of continuous data. We saw earlier that the Gaussian distribution arises if we average or add a number of quantities together: the hundreds of genetic differences between people in this room, or the thousands of events in your childhood, that add or subtract a few tenths of a millimetre. However if these genes or events do not act additively but multiplicatively - so rather than adding or subtracting a few tenths of a millimetre they make you a tenth of a *percent* bigger or smaller then the resulting distribution is log-normal. If we take the log of this distribution we end up back with the normal distribution. So we've tried that here.

Generalised Linear Model

```
> traffic_m2 <- glm(log(y) ~ limit + year + day, data = Traffic)

> summary(traffic_m2)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.13043  -0.26274  -0.02357   0.27311   0.89153

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.959599   0.066780  44.318  < 2e-16 ***
limityes    -0.168251   0.062359  -2.698  0.00764 **
year1962    -0.069312   0.060318  -1.149  0.25203
day          0.002750   0.001083   2.539  0.01197 *
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.1519523)

    Null deviance: 30.188  on 183  degrees of freedom
Residual deviance: 27.351  on 180  degrees of freedom
AIC: 181.43

Number of Fisher Scoring iterations: 2
```



Generalised Linear Model

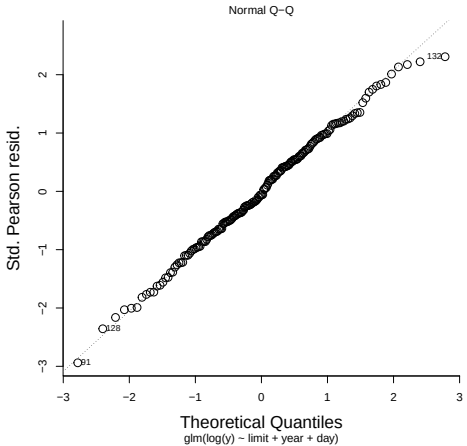
```
> traffic_m2 <- glm(log(y) ~ limit + year + day, data = Traffic)

> summary(traffic_m2)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.13043  -0.26274  -0.02357   0.27311   0.89153

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.959599   0.066780  44.318  < 2e-16 ***
limityes    -0.168251   0.062359  -2.698  0.00764 **
year1962    -0.069312   0.060318  -1.149  0.25203
day          0.002750   0.001083   2.539  0.01197 *
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.1519523)

    Null deviance: 30.188  on 183  degrees of freedom
Residual deviance: 27.351  on 180  degrees of freedom
AIC: 181.43

Number of Fisher Scoring iterations: 2
```

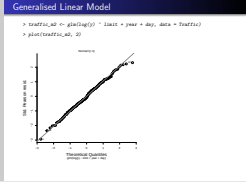
The coefficients appear entirely different to what we had before, but that is because they are no longer looking at additive differences but proportional differences. If you exponentiate the coefficient you get the proportional change in the outcome, so for example `exp(coef(traffic_m2)["limityes"])` gives a value of 0.85; under a speed limit we expect 85% the number of accidents as we would without a speed limit, a 15% reduction[1]. In many ways working with proportional changes is easier. If someone told me there were 3.7 less accidents in Sweden when a speed limit is in place I would find it hard to know whether this is a big or small difference without knowing something about road accidents rates generally. If I was told a speed limit results in a 15% reduction I can immediately see that this is quite a substantial change.

[1] If x is small, $exp(x)$ is approximately equal to $1 + x$ (likewise $\log(1 + x) \approx x$ when x is small). Consequently, if the magnitude of the coefficients is not too large (between -0.2 and 0.2) then you can actually just multiply them by 100 and read it as the percentage change (16.8%≈15% reduction).

Generalised Linear Model

```
> traffic_m2 <- glm(log(y) ~ limit + year + day, data = Traffic)
> plot(traffic_m2, 2)
```

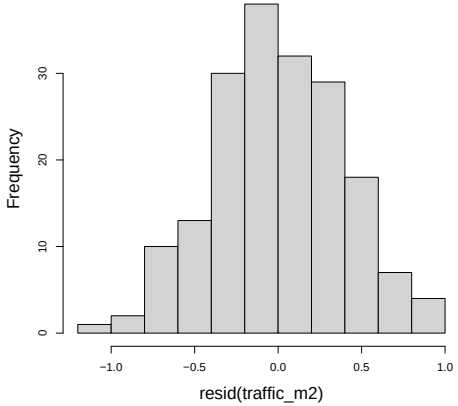


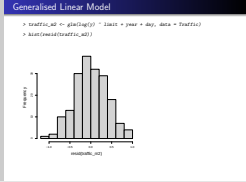


The new model also seems to do a better job at capturing the features of the data. The quantile plot is nicer and hardly deviates from the 1:1 line

Generalised Linear Model

```
> traffic_m2 <- glm(log(y) ~ limit + year + day, data = Traffic)
> hist(resid(traffic_m2))
```





and accordingly a histogram of the residuals shows them to be pretty close to normality. However, because these are count data people usually go straight for a Poisson distribution, and in some ways treating it as Poisson is a bit more satisfying; it acknowledges the fact that the response has to be integer valued (you can't have 3.5 accidents) and can't be negative (which could happen in the Gaussian model when the response is not logged). Also, if there had been days when there were zero accidents, simply logging the data would have caused issues because the log of 0 is minus infinity.

Generalised Linear Model

```
> traffic_m3 <- glm(y ~ limit + year + day, data = Traffic, family = poisson)
```


Generalised Linear Model

> traffic_m3 <- glm(y ~ limit + year + day, data = Traffic, family = poisson)

So we can simply switch from using a Normal distribution to using a Poisson distribution through the `family` argument. We haven't specified a link function so the default (canonical) link is used; the log link.

Generalised Linear Model

```
> traffic_m3 <- glm(y ~ limit + year + day, data = Traffic, family = poisson)

> summary(traffic_m3)
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.1774  -1.4067  -0.4040   0.9725   4.9920

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.0467406  0.0372985  81.685  < 2e-16 ***
limityes    -0.1749337  0.0355784  -4.917 8.79e-07 ***
year1962    -0.0605503  0.0334364  -1.811   0.0702 .
day          0.0024164  0.0005964   4.052 5.09e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 625.25  on 183  degrees of freedom
Residual deviance: 569.25  on 180  degrees of freedom
AIC: 1467.2

Number of Fisher Scoring iterations: 4
```


Generalised Linear Model

> traffic_m3 <- glm(y ~ limit + year + day, data = Traffic, family = poisson)

> summary(traffic_m3)
Deviance Residuals:
Min 1Q Median 3Q Max
-4.1774 -1.4067 -0.4040 0.9725 4.9920

Coefficients:
Estimate Std. Error z value Pr(>|z|)
(Intercept) 3.0467406 0.0372985 81.685 < 2e-16 ***
limityes -0.1749337 0.0355784 -4.917 8.79e-07 ***
year1962 -0.0605503 0.0334364 -1.811 0.0702 .
day 0.0024164 0.0005964 4.052 5.09e-05 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 625.25 on 183 degrees of freedom
Residual deviance: 569.25 on 180 degrees of freedom
AIC: 1467.2

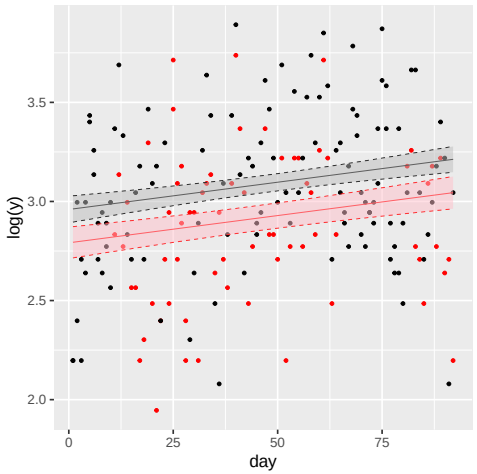
Number of Fisher Scoring iterations: 4

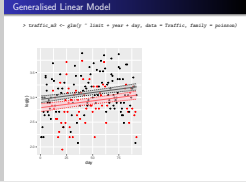
The estimates are very similar to what we got under the Gaussian model when we logged the data, which makes sense because they are both on the log scale and are modelling proportional changes[1]. However, the standard errors are much smaller, and consequently so are the p-values. I love the Poisson! The number of accidents certainly increases over the year, and perhaps there are even differences between year.

[1] A word of caution: $E[log(y)]$ does not equal $log(E[y])$. When we logged the response and fitted a standard linear model we are trying to predict $E[log(y)]$. When treating the data as Poisson and using the log-link we are trying to predict $log(E[y])$. In general, $E[log(y)]$ is less than $log(E[y])$ by an amount the depends on the variance of y. If $log(y)$ is normally distributed with mean μ and variance σ^2, $E[log(y)] = \mu$ by definition, but $log(E[y]) = \mu + \sigma^2/2$. Because of this, we only expect the coefficients from a linear model on logged count data to be approximately the same as the coefficients from a Poisson glm with log-link.

Generalised Linear Model

```
> traffic_m3 <- glm(y ~ limit + year + day, data = Traffic, family = poisson)
```

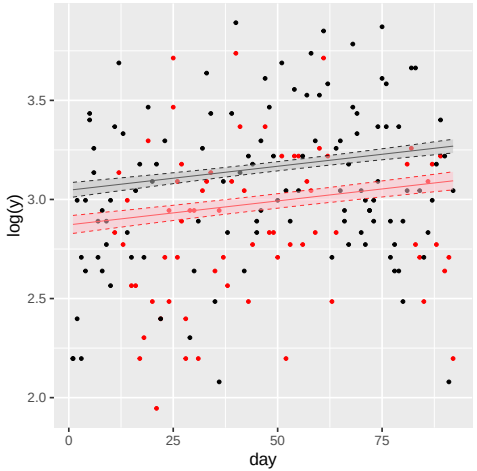


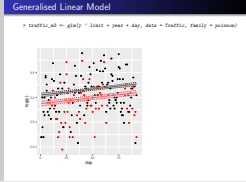


We can see this graphically too. These are our data (on the log scale) with red data points being days where there was a speed limit and black being days without a speed limit. The lines are the estimates of the expected number of accidents plus and minus the standard errors from our model where we log-transformed the response and specified Gaussian errors. I'll show you how to plot the data and the predictions on the arithmetic scale shortly, but for now I think this illustrates the point.

Generalised Linear Model

```
> traffic_m3 <- glm(y ~ limit + year + day, data = Traffic, family = poisson)
```

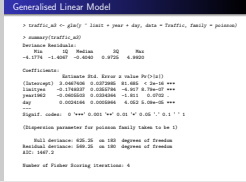




If we look at the equivalent plot for the Poisson model, we can see that the estimates are very similar but the uncertainty around them is considerably less. Is this because the Poisson glm is more powerful than a standard linear model? Probably not.

Generalised Linear Model

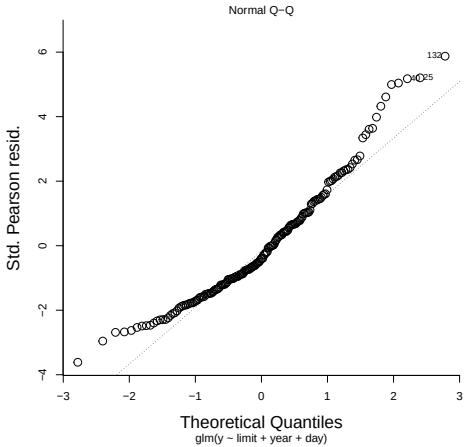
```
> traffic_m3 <- glm(y ~ limit + year + day, data = Traffic, family = poisson)

> summary(traffic_m3)
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.1774  -1.4067  -0.4040   0.9725   4.9920

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.0467406  0.0372985  81.685  < 2e-16 ***
limityes    -0.1749337  0.0355784  -4.917 8.79e-07 ***
year1962    -0.0605503  0.0334364  -1.811   0.0702 .
day          0.0024164  0.0005964   4.052 5.09e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 625.25  on 183  degrees of freedom
Residual deviance: 569.25  on 180  degrees of freedom
AIC: 1467.2

Number of Fisher Scoring iterations: 4
```

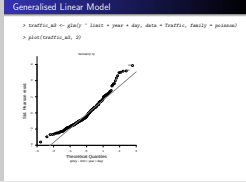



In a Poisson distribution the variance is equal to the expectation, but for most count data the variance in the response usually exceeds the expectation. This is known as overdispersion and you can see here that the ratio of the residual deviance to the residual degrees of freedom is about 3.2 which means, roughly speaking, there is more than 3 times as much variation in the residuals than what we expect. Now its easy to see why this might happen: imagine you didn't have data on whether a speed limit was in place or not, and you didn't know what year or day the observations were made. If you then just fitted the model with an intercept the variability caused by these terms is still manifest in the data but we have no way of controlling for it and the residual variability will increase. Now even if you have this information how likely is it that there are not other things out there that cause variation in road accidents that you were not able to measure. Very unlikely I think, I could imagine a whole suite of things such as a particular day was icy or not, or it was a holiday so there were less cars on the road, and such like.

Generalised Linear Model

```
> traffic_m3 <- glm(y ~ limit + year + day, data = Traffic, family = poisson)
> plot(traffic_m3, 2)
```

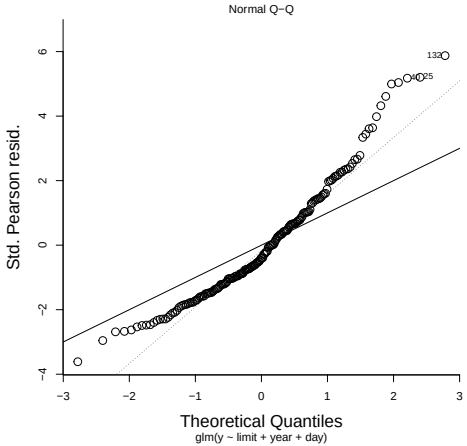




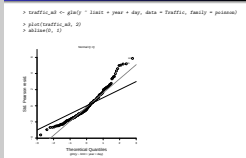
We can see this in the qq-plot too. At first it doesn't look too bad, but this is because the dashed line in the qq-plot is NOT the 1:1 line.

Generalised Linear Model

```
> traffic_m3 <- glm(y ~ limit + year + day, data = Traffic, family = poisson)

> plot(traffic_m3, 2)
> abline(0, 1)
```



Generalised Linear Model



If we add the 1:1 line we can see that we have an excess of extreme values compared to what we expect.

Generalised Linear Model

```
> traffic_m3 <- glm(y ~ limit + year + day, data = Traffic, family = poisson)

> sim_max <- apply(simulate(traffic_m3, n = 1000), 2, max)
```

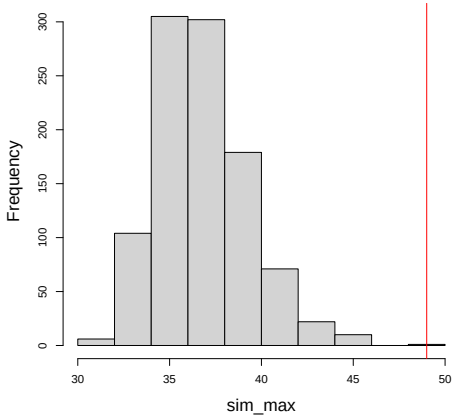

Generalised Linear Model

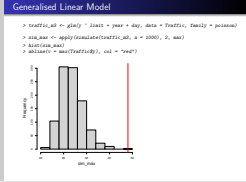
> traffic_m3 <- glm(y ~ limit + year + day, data = Traffic, family = poisson)
> sim_max <- apply(simulate(traffic_m3, n = 1000), 2, max)

A very useful function in R is `simulate`. We can pass it our model and it will generate `n` replicate data sets assuming that the underlying model is true and the parameters are equal to our (maximum likelihood) estimates. For each replicate data set we can then summarise features of the distribution and compare it to what we see empirically in the data we have actually collected. What I have done here is saved the maximum number of accidents that we would expect to see in our data if our model were true. We can then plot a histogram of this number and compare it to the maximum number we actually observe.

Generalised Linear Model

```
> traffic_m3 <- glm(y ~ limit + year + day, data = Traffic, family = poisson)

> sim_max <- apply(simulate(traffic_m3, n = 1000), 2, max)
> hist(sim_max)
> abline(v = max(Traffic$y), col = "red")
```





And indeed, on one day we observed 49 accidents which is very very unlikely to have occurred under our model. This is one way to understand why overdispersion results in too narrow standard errors and an increased (often massively so) type I error rate. These days where there are 49 accidents are incredibly unlikely given the model. They would be even more unlikely under the null model where the probability of getting a value of 49 or higher is 3e-07 `1-ppois(48,mean(Traffic$y))`. Any thing additional in the model that could explain such a rare occurrence would come out as significant because there simply isn't any flexibility in the null model to accommodate such occurrences. If the extreme value happened to be associated with a particular level of a categorical predictor or happened to be associated with an extreme value of some continuous predictor, then the coefficients associated with these predictors may well come out as significant. However, under a more plausible null model the extreme observations may not be too surprising and there may be little support for the predictors having an effect on the response. A more plausible model, and one that we've alluded to, would be to allow the *expected* number of accidents to vary across sampling points due to unmeasured variables. This would allow the variation in the number of *observed* accidents to exceed the predicted mean based on the measured variables.

Generalised Linear Model: Overdispersion

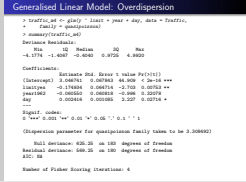
```
> traffic_m4 <- glm(y ~ limit + year + day, data = Traffic,
+     family = quasipoisson)
```


Generalised Linear Model: Overdispersion
> traffic_m4 <- glm(y ~ limit + year + day, data = Traffic,
+ family = quasipoisson)

One possible solution is to bring in the fudge factor: and use a quasipoisson distribution.

Generalised Linear Model: Overdispersion

```
> traffic_m4 <- glm(y ~ limit + year + day, data = Traffic,
+     family = quasipoisson)
> summary(traffic_m4)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.1774  -1.4067  -0.4040   0.9725   4.9920

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.046741   0.067843  44.909  < 2e-16 ***
limityes    -0.174934   0.064714  -2.703  0.00753 **
year1962    -0.060550   0.060818  -0.996  0.32078
day          0.002416   0.001085   2.227  0.02716 *
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasipoisson family taken to be 3.308492)

    Null deviance: 625.25  on 183  degrees of freedom
Residual deviance: 569.25  on 180  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 4
```

And you can see that although the parameter estimates haven't changed, the standard errors have got larger and the p-values are now much more moderate. In addition, the dispersion parameter is no longer 1 but is 3.308. This number reflects how much excess variation there is compared to what we expect (i.e. there is 3.308 times more variation than we expect from the standard Poisson). The sampling variance of the coefficients is then simply increased by this factor (the standard errors are multiplied by the square-root of this factor)[1]. In general it works well, and in fact this summary is largely recapitulating what the model on the log-transformed data told us.

The other thing to notice is that something called AIC is now "not available" and neither is the likelihood - if you try `logLik(traffic_m4)` it will also return `NA`. When you specify "quasipoisson" you are no longer fitting a parametric model to the data. This, I think, is a serious drawback because it makes it hard for us to understand the process by which the data came to be (we couldn't simulate it for example) and it makes it harder to assess the models adequacy and compare it to other models.

[1] Recall from the first day's lecture that the sampling variance of the mean was σ^2_e/n. The sampling variance of a linear model coefficient is a little more complicated but it is still proportional to the residual variance. Consequently, if we increase the residual variance by a factor x (e.g. 3.308), then the sampling variance needs to increases by x and the sampling deviation (the standard error) needs to increases by $\sqrt{x}$.

Generalised Linear Model: Overdispersion

```
> traffic_m5 <- MASS::glm.nb(y ~ limit + year + day, data = Traffic)
```

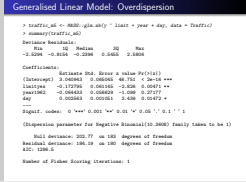

Generalised Linear Model: Overdispersion

> traffic_m5 <- MASS::glm.nb(y ~ limit + year + day, data = Traffic)

An alternative, and widely used option, is to assume the data come from a negative binomial distribution. You can't fit this distribution using the function `glm` but there are libraries out there that allow you to work the negative binomial, such as the `MASS` package.

Generalised Linear Model: Overdispersion

```
> traffic_m5 <- MASS::glm.nb(y ~ limit + year + day, data = Traffic)
> summary(traffic_m5)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5294  -0.8154  -0.2396   0.5455   2.5806

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.040943   0.065045  46.751  < 2e-16 ***
limityes    -0.172795   0.061145  -2.826  0.00471 **
year1962    -0.064433   0.058629  -1.099  0.27177
day          0.002563   0.001051   2.439  0.01472 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(10.3608) family taken to be 1)

    Null deviance: 202.77  on 183  degrees of freedom
Residual deviance: 184.19  on 180  degrees of freedom
AIC: 1286.5

Number of Fisher Scoring iterations: 1
```

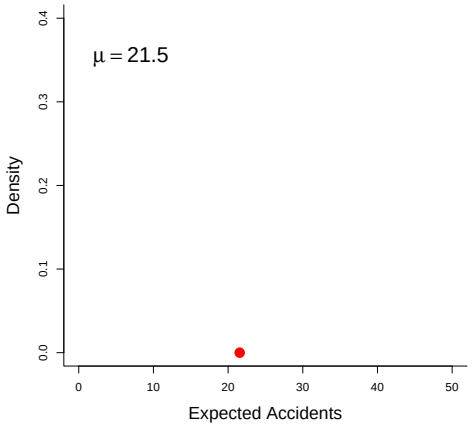


Generalised Linear Model: Overdispersion

```
> traffic_m5 <- MASS::glm.nb(y ~ limit + year + day, data = Traffic)
> summary(traffic_m5)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5294  -0.8154  -0.2396   0.5455   2.5806

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.040943   0.065045  46.751  < 2e-16 ***
limityes    -0.172795   0.061145  -2.826  0.00471 **
year1962    -0.064433   0.058629  -1.099  0.27177
day          0.002563   0.001051   2.439  0.01472 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(10.3608) family taken to be 1)

    Null deviance: 202.77  on 183  degrees of freedom
Residual deviance: 184.19  on 180  degrees of freedom
AIC: 1286.5

Number of Fisher Scoring iterations: 1
```

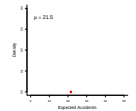
You can see that again the estimates and standard errors are very similar to the normal model applied to the log of the counts, and also the quasipoisson model. However, unlike the quasi-poisson model, there is an additional parameter that is estimated that allows for overdispersion. Most people don't think too much about this parameter, but its useful to understand what it means; its the shape parameter of a gamma-distribution[1].

[1] I always find the standard parameterisation of the negative binomial hard to interpret. The coefficients in the table are the logged expectations as in the Poisson glm with log link, and so exponentiating them you get the mean of the underlying gamma (`shape*scale`). Getting the variance of the underlying gamma is a bit more work. The reported dispersion parameter is the shape parameter of the gamma (in the output it is called `theta` which is very annoying because `theta` is usually used to refer to the scale parameter rather than the shape parameter), and because the variance in the gamma distribution is `shape*scale`^{2} then the variance is the mean squared divided by the reported shape parameter. In this example, the expected number of accidents on day 0 of 1961 without a speed limit is `exp(coef(traffic_m5)["(Intercept)"])`=20.92. The variance is then 20.92^2/`traffic_m5$theta` = 42.26. In terms of the actual outcomes (rather than the expectations) there is also the additional Poisson variance which is equal to the mean. Adding the gamma variance to the Poisson variance gives 20.92 + 42.26 = 63.19 which isn't too far from the variance (the dispersion) in the first normal model fitted using `glm` (71.8).

Understanding the Negative Binomial

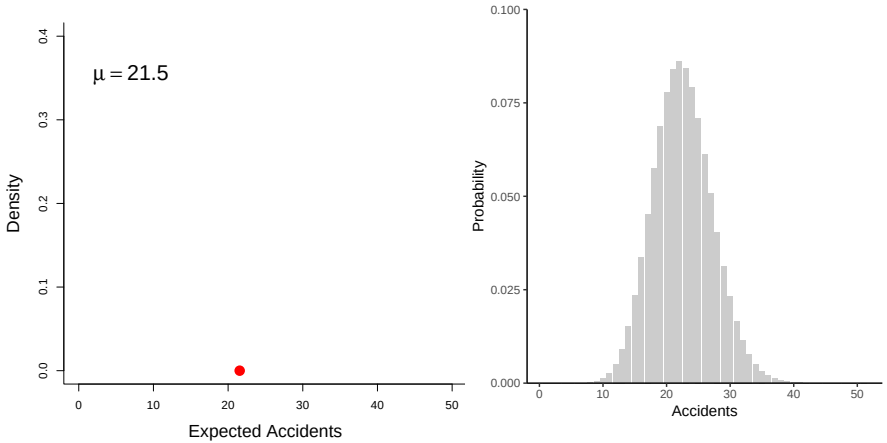


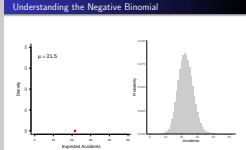
Understanding the Negative Binomial



The average number of accidents per day in our data frame is 21.5. First, imagine a Poisson distribution with a mean parameter of 21.5. That does not mean on every day we will actually observe 21.5 accidents (how could we!),

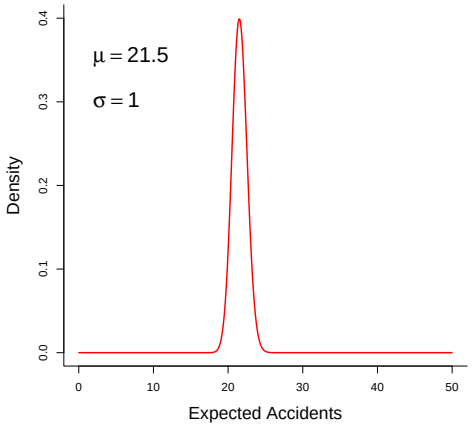
Understanding the Negative Binomial



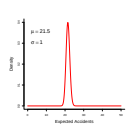


it means the number of accidents over days will be drawn from a Poisson distribution with this mean. Now what is the chance that all days have exactly the same *expected* number of accidents? If we want to think about the Poisson as a binomial with low p and high n, do we think that the number of cars on the road is the same from day to day (is n really fixed?) and do we really think the probability of any one car having an accident is constant from day to day (is p really fixed?). Probably not - I think we would expect pn to vary across dates.

Understanding the Negative Binomial

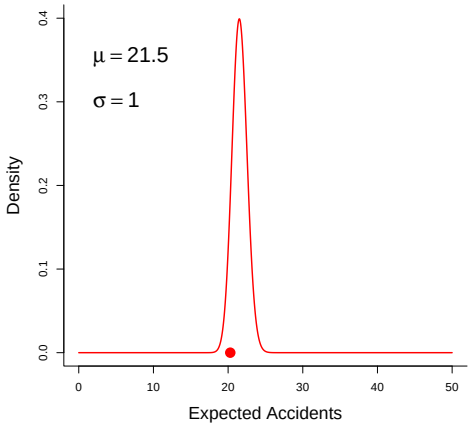


Understanding the Negative Binomial

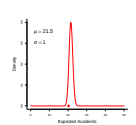


We could imagine then that the expected number of accidents on a particular date isn't just a single number but comes from a distribution of possible numbers. We could say the standard deviation of this distribution is quite small - lets say 1.

Understanding the Negative Binomial

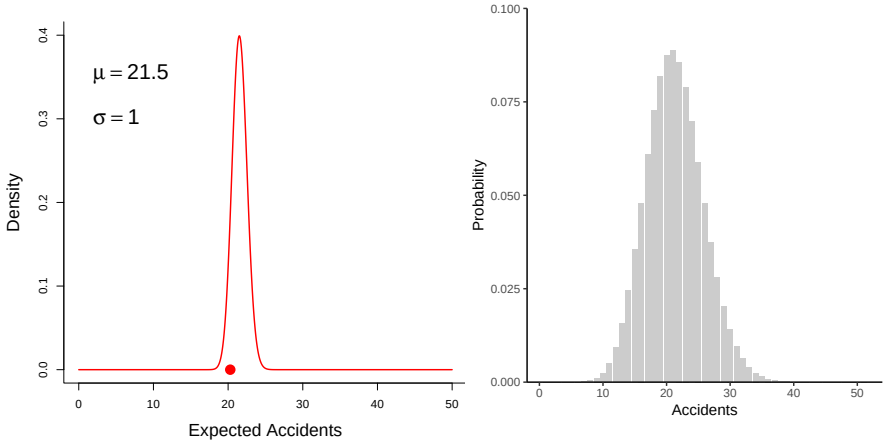


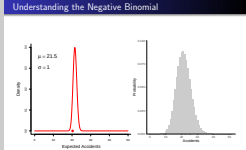
Understanding the Negative Binomial



and then we draw a number from this distribution - it will be close to 21.5 but not exactly 21.5 - which gives the expected number of accidents for that date, here it is perhaps 21.6 or 21.7.

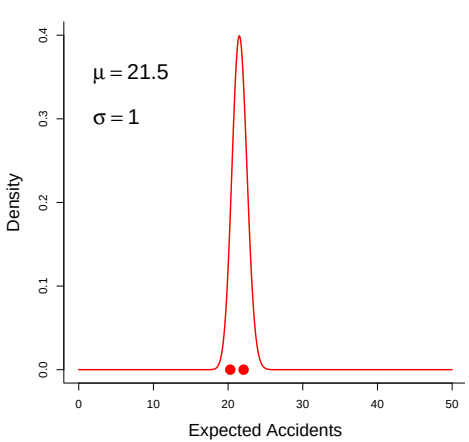
Understanding the Negative Binomial

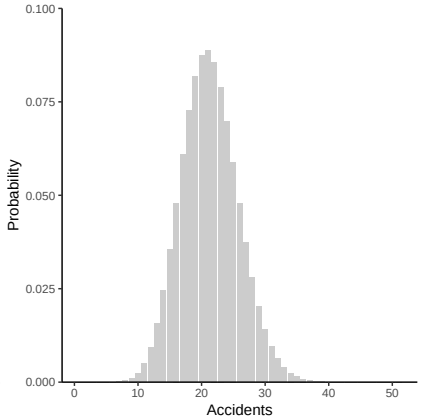




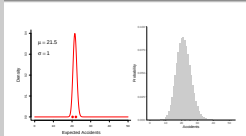
The observed number of accidents on this particular date will then be Poisson distributed with this mean of 21.7. However, that's just one possibility

Understanding the Negative Binomial



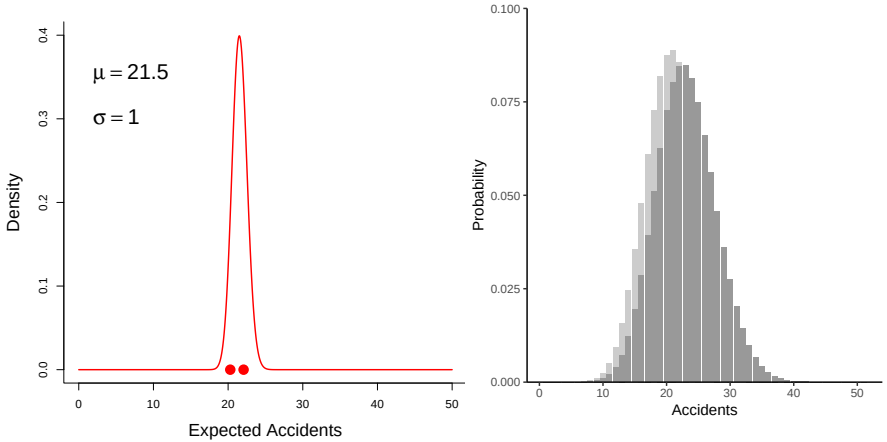


Understanding the Negative Binomial

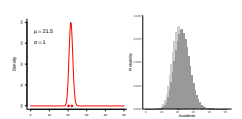


We could have drawn this expectation for example,

Understanding the Negative Binomial

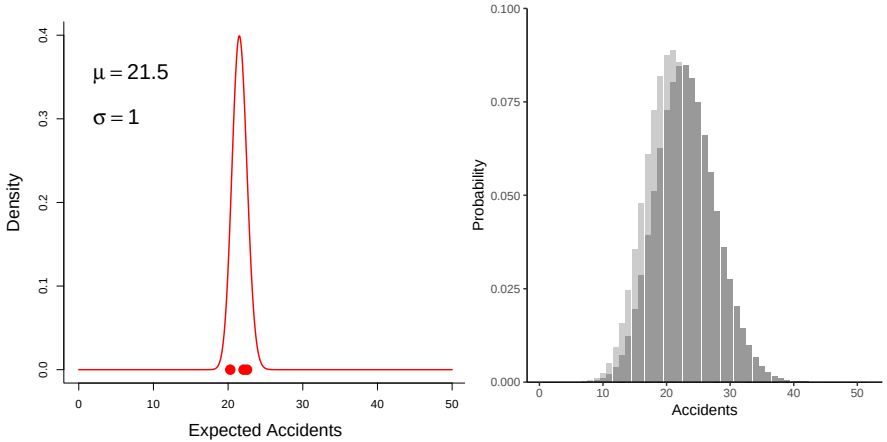


Understanding the Negative Binomial

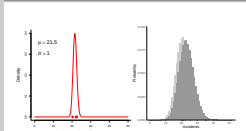


in which case the Poisson would look like this darker distribution with a slightly higher mean (and slightly higher variance)

Understanding the Negative Binomial

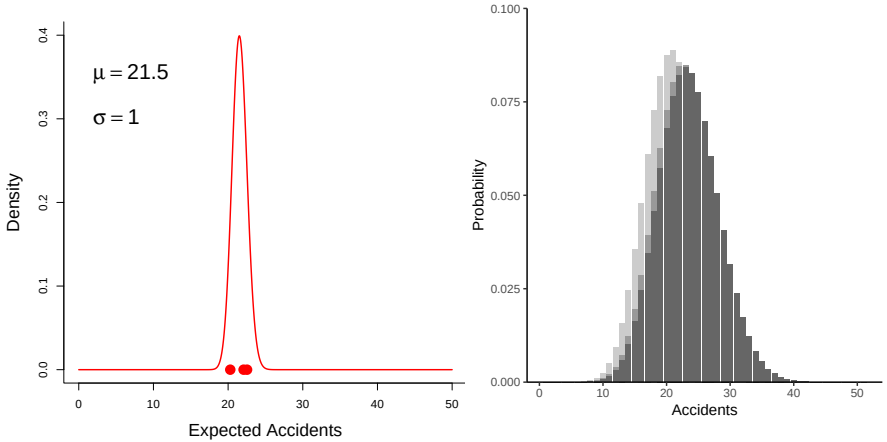


Understanding the Negative Binomial

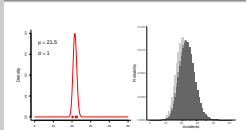


or it could have been this expectation,

Understanding the Negative Binomial

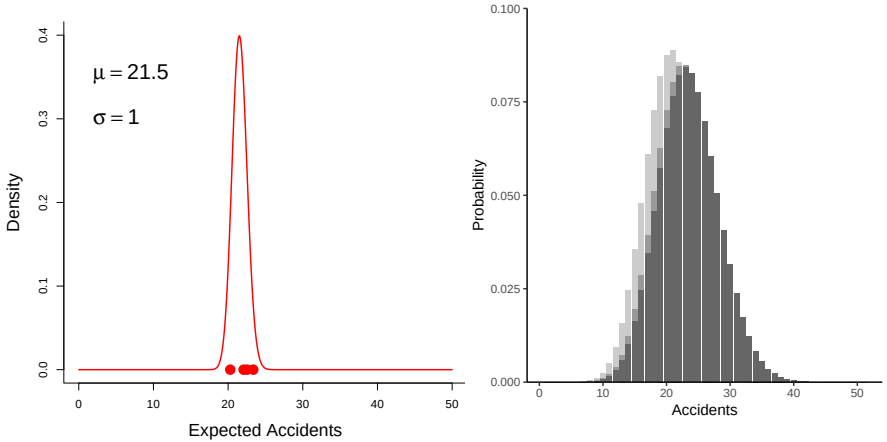


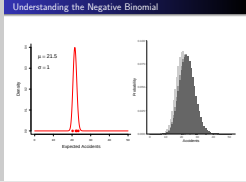
Understanding the Negative Binomial



generating another Poisson.

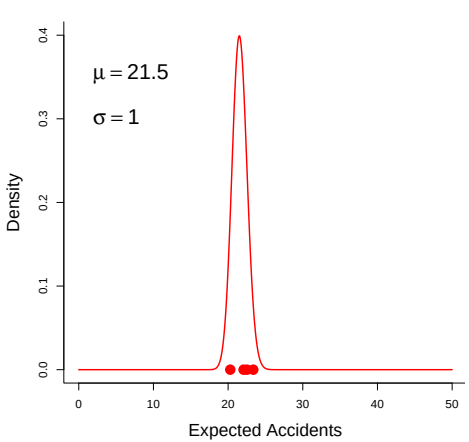
Understanding the Negative Binomial

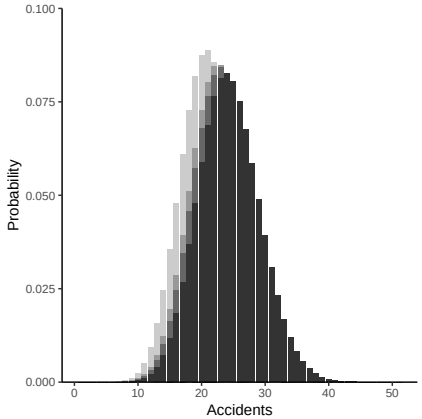




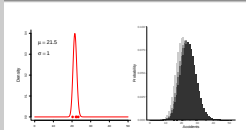
or this expectation,

Understanding the Negative Binomial



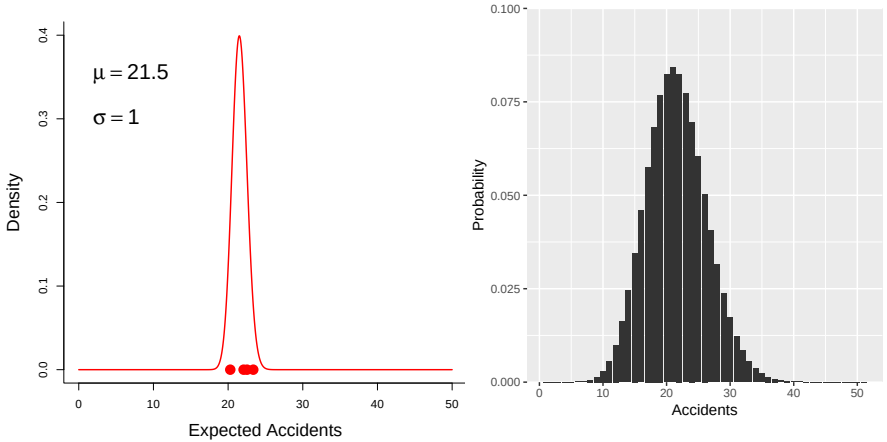


Understanding the Negative Binomial

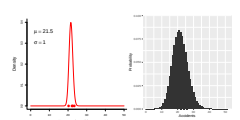


generating another Poisson..... If the distribution of expectations (the red distribution) is a gamma distribution,

Understanding the Negative Binomial

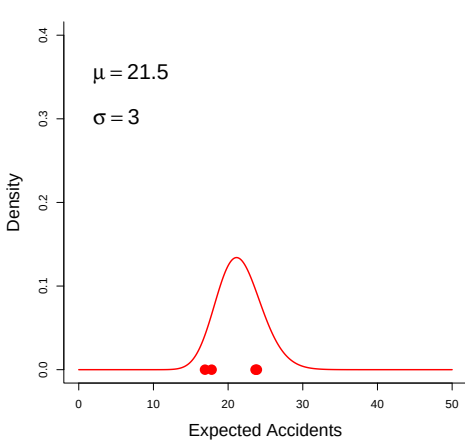


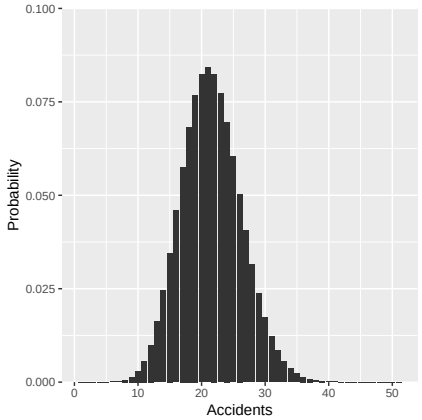
Understanding the Negative Binomial



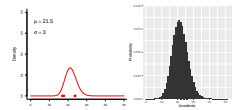
and we imagine combining all these Poisson distributions into a single distribution, then we end up with a negative binomial distribution with a parameter estimating the standard deviation of the underlying gamma distribution; a negative binomial distribution is a mixture of Poisson distributions, whose means are distributed according to a gamma distribution. When we move on to mixed-effect models we'll see how can use this idea to deal with over-dispersion for other types of distribution such as the binomial.

Understanding the Negative Binomial



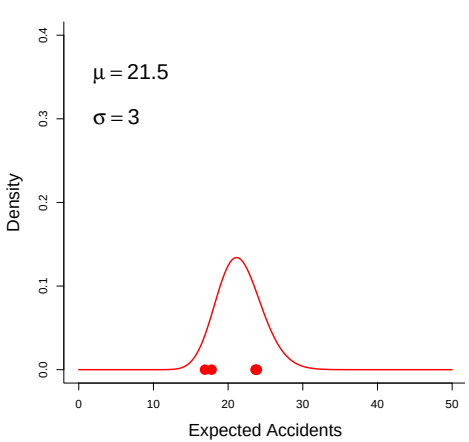


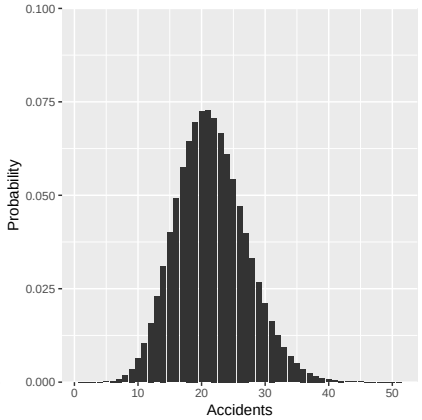
Understanding the Negative Binomial

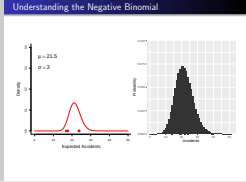


If we up the standard deviation of the gamma to 3, then the distribution of the expected number of accidents will be wider

Understanding the Negative Binomial

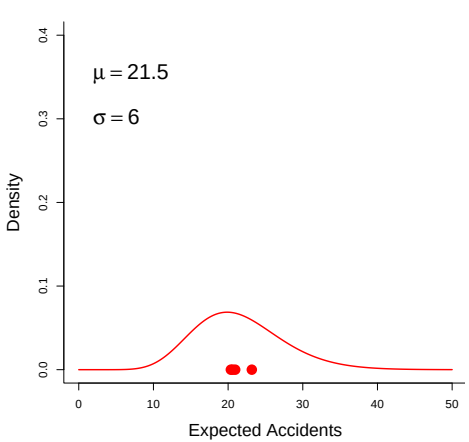


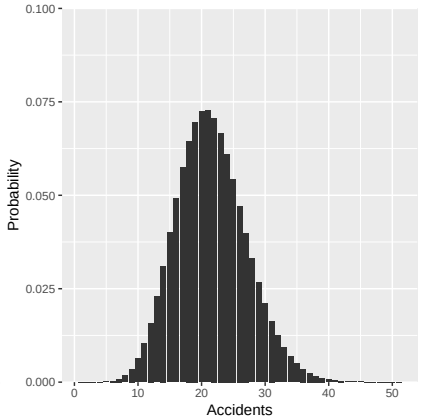


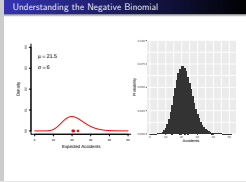


resulting in a mixture of Poisson's with greater variability

Understanding the Negative Binomial

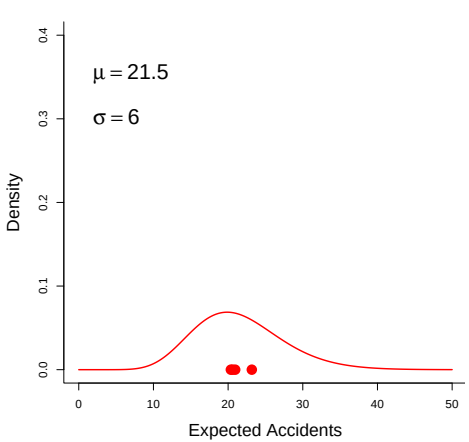


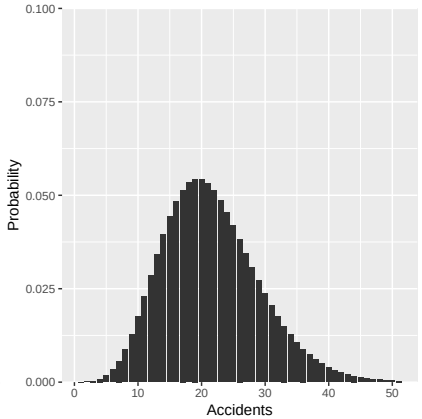




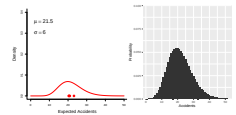
and as we up the standard deviation more and more

Understanding the Negative Binomial



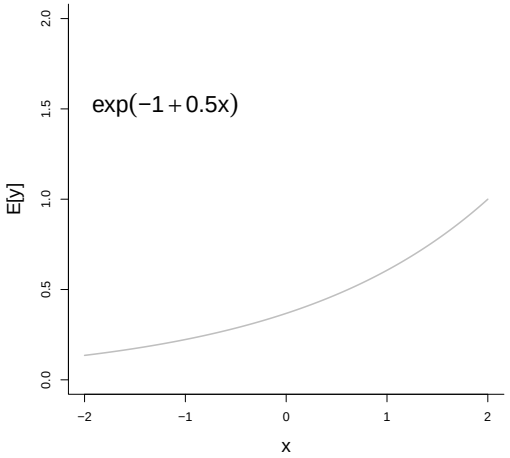


Understanding the Negative Binomial

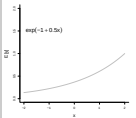


the mixture of Poisson's becomes ever more variable with a longer right hand tail. So by estimating the standard deviation of the underlying gamma distribution (σ) we allow the model to capture any excess variation.

Link functions: log link



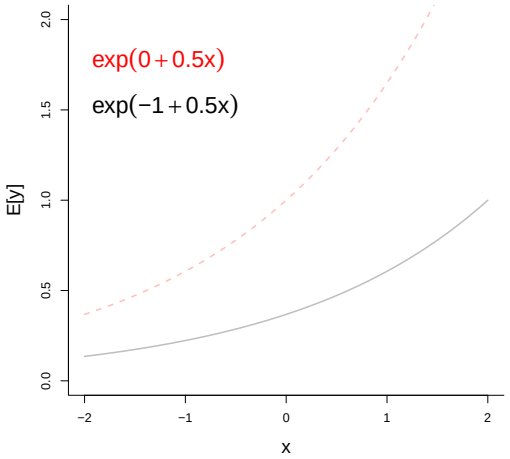
Link functions: log link



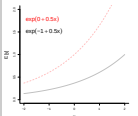
We touched briefly on how to interpret the coefficients from a model with log-link and I said that if we exponentiate the coefficient we get the proportional change in the outcome due to some factor or by changing the covariate by one unit. Many people haven't absorbed this information, and because of this they rarely look at the coefficients in the model they've fitted, they just go straight to the p-values. This is a shame, because the coefficients are telling you something much more valuable than the p-values.

On the y-axis we have our prediction on the data (arithmetic) scale (the expected number of counts) and on the x-axis we have the value of some covariate x. Imagine this log-linear model where the intercept is -1 and the coefficient associated with the covariate x is 0.5.

Link functions: log link

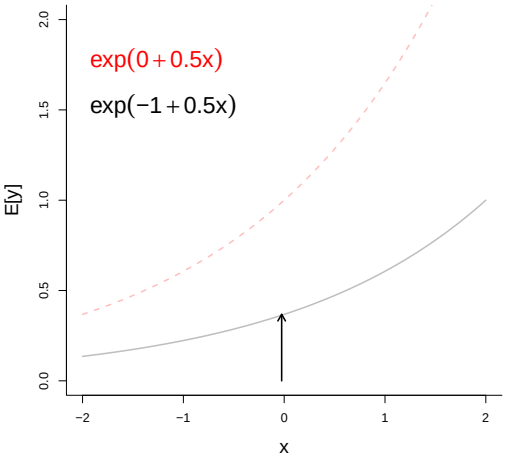


Link functions: log link

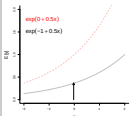


Imagine then that you also have a categorical predictor that you fit as a main effect and the log-linear model for data associated with this second category is this. You can see there is no interaction between the covariate and the factor because the regression coefficent is 0.5 for both groups. However, there is quite a sizeable main effect and the associated coefficient would be 1 in this instance; 0 is 1 unit higher than the intercept of -1.

Link functions: log link

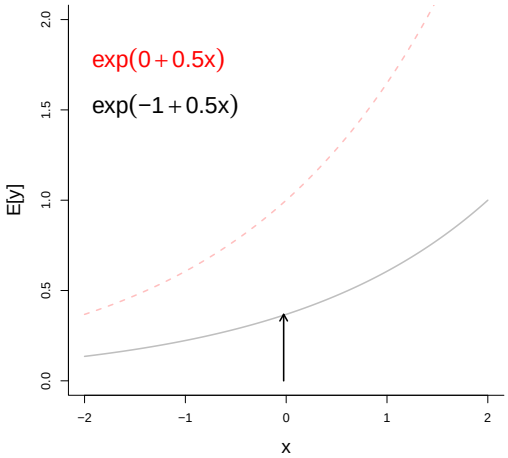


Link functions: log link

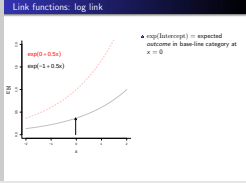


When the value of the covariate is 0 the expected outcome for the base-line catogory is around 0.4

Link functions: log link

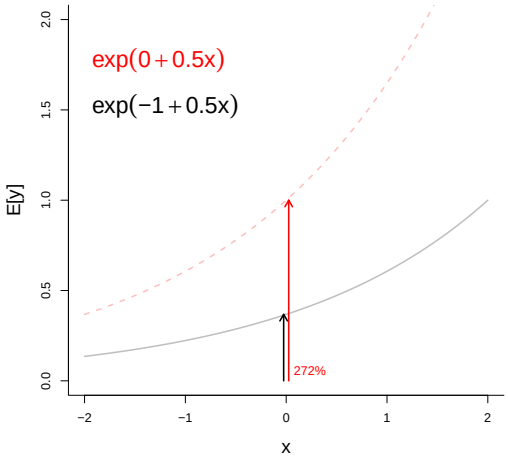


- $\exp(\text{Intercept}) =$ expected *outcome* in base-line category at $x = 0$

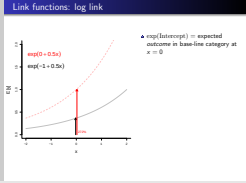


and we can obtain this by exponentiating the intercept (exp(−1)=0.37)

Link functions: log link

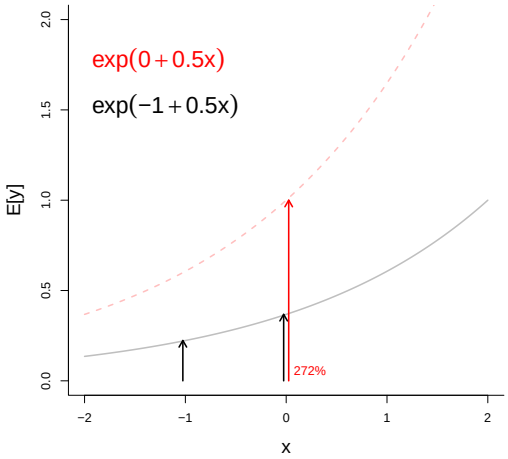


- $\exp(\text{Intercept}) =$ expected *outcome* in base-line category at $x = 0$

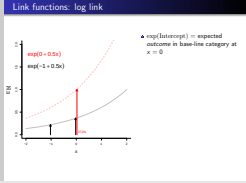


If we took the expected response in the base-line category when x=0 and compared that to the red category, the proportional change in the response between the two groups when x=0 is about 270% in this instance; the expected response is 2.7 times higher in the red group.

Link functions: log link

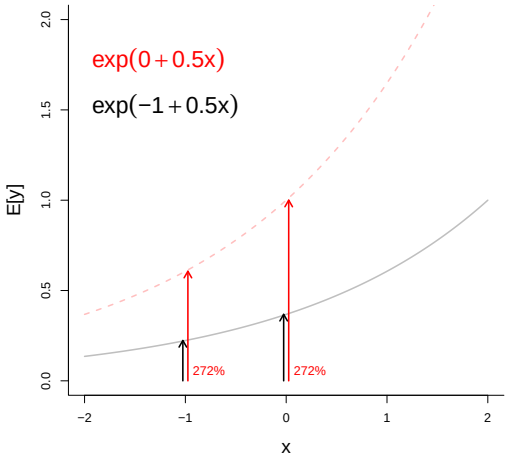


- $\exp(\text{Intercept}) =$ expected *outcome* in base-line category at $x = 0$

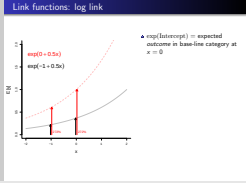


If we took the expected response in the base-line category when `x=-1`

Link functions: log link

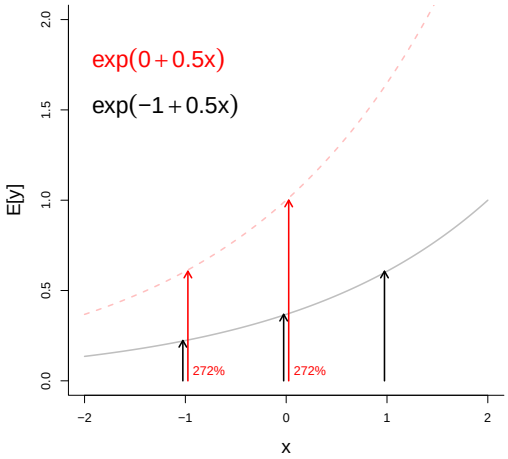


- $\exp(\text{Intercept}) =$ expected *outcome* in base-line category at $x = 0$

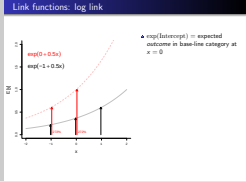


the expected response is again 2.7 times higher in the red group, and this

Link functions: log link

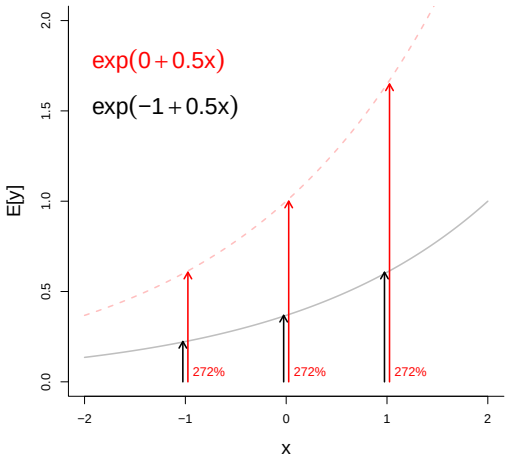


- $\exp(\text{Intercept}) =$ expected *outcome* in base-line category at $x = 0$

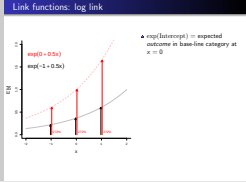


is true for any value of the covariate;

Link functions: log link

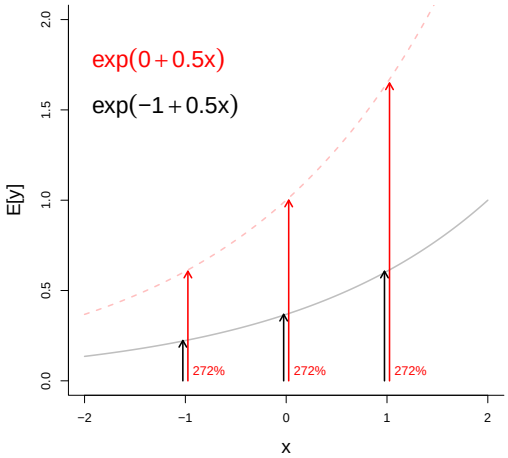


- $\exp(\text{Intercept}) =$ expected *outcome* in base-line category at $x = 0$

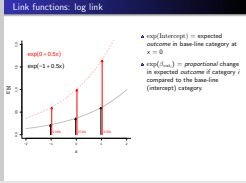


we always see the same proportional change.

Link functions: log link

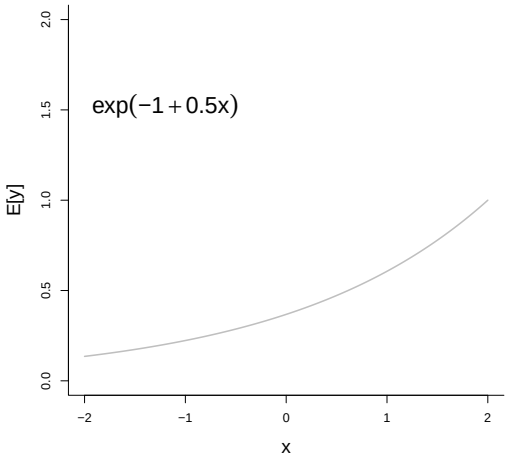


- $\exp(\text{Intercept}) =$ expected *outcome* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i}) =$ *proportional* change in expected *outcome* if category i compared to the base-line (intercept) category.



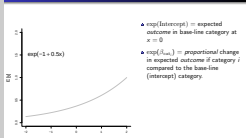
If we exponentiate the coefficient associated with the *difference* between the two groups we get this number; in this example the coefficient would be 1 and `exp(1)`=2.72.

Link functions: log link



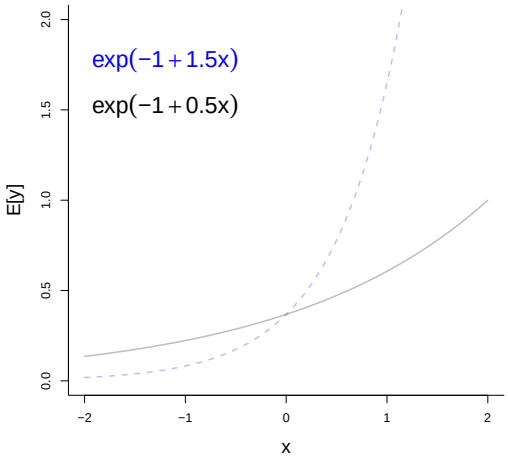
- $\exp(\text{Intercept}) =$ expected *outcome* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i}) =$ *proportional* change in expected *outcome* if category i compared to the base-line (intercept) category.

Link functions: log link



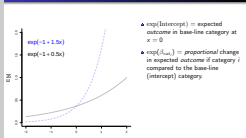
Lets imagine a slightly different scenario. Lets have the same model for the first category, but

Link functions: log link



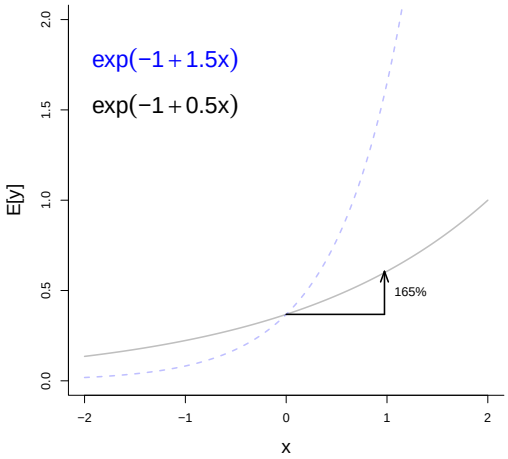
- $\exp(\text{Intercept}) =$ expected *outcome* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i}) =$ *proportional* change in expected *outcome* if category i compared to the base-line (intercept) category.

Link functions: log link

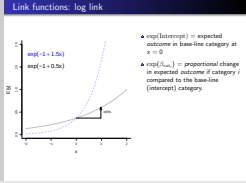


lets assume no main effect (so both groups have the same intercept) but the slopes for the two categories vary.

Link functions: log link

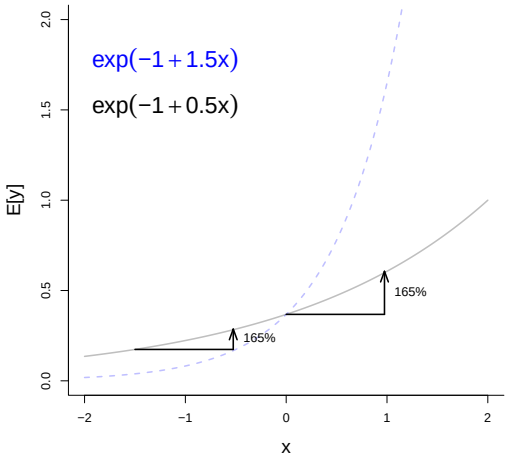


- $\exp(\text{Intercept}) =$ expected *outcome* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i}) =$ *proportional* change in expected *outcome* if category i compared to the base-line (intercept) category.

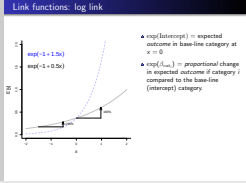


If we increased the value of the covariate from 0 to 1 in the base-line category the expected outcome increases by a factor of 1.65.

Link functions: log link

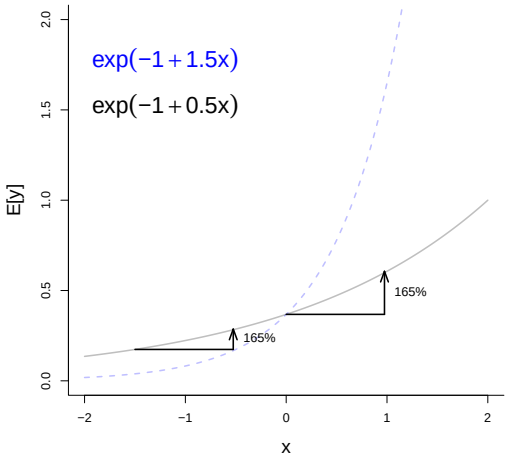


- $\exp(\text{Intercept})$ = expected *outcome* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i})$ = *proportional* change in expected *outcome* if category i compared to the base-line (intercept) category.

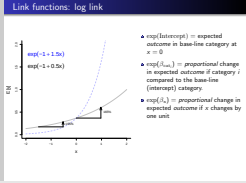


and this would be true if we compared the outcome between any two values of `x` that differed by one unit. So the expected outcome when `x=-0.5` compared to that when `x=-1.5` also differs by a factor of 1.65.

Link functions: log link

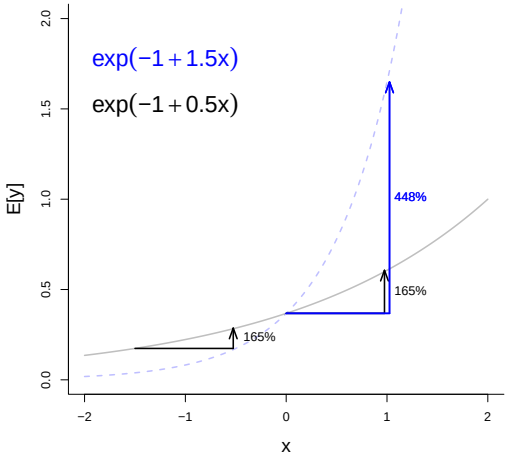


- $\exp(\text{Intercept}) =$ expected *outcome* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i}) =$ *proportional* change in expected *outcome* if category i compared to the base-line (intercept) category.
- $\exp(\beta_{\text{x}}) =$ *proportional* change in expected *outcome* if x changes by one unit

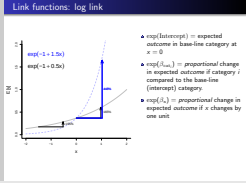


Again, you can obtain this number by exponentiating the relevant coefficient, which in this case is 0.5 `exp(0.5)`=1.65

Link functions: log link

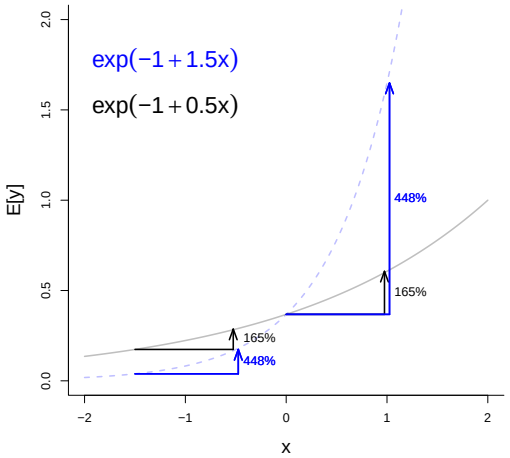


- $\exp(\text{Intercept}) =$ expected *outcome* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i}) =$ *proportional* change in expected *outcome* if category i compared to the base-line (intercept) category.
- $\exp(\beta_{\text{x}}) =$ *proportional* change in expected *outcome* if x changes by one unit

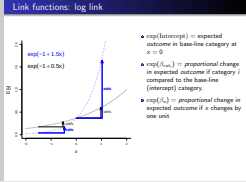


If we exponentiate 1.5 we get 4.48 indicating that if we increase the value of the covariate in the blue group by one unit the outcome would change by a factor of 4.48.

Link functions: log link

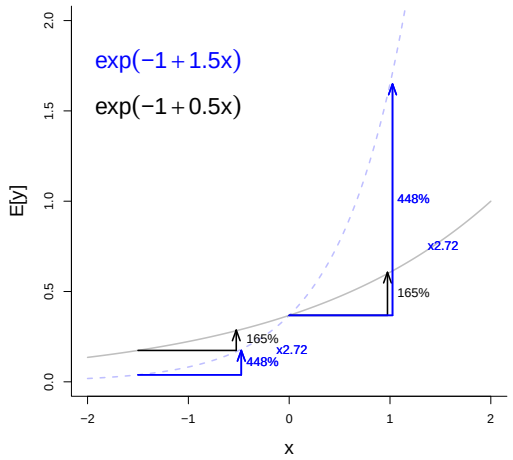


- $\exp(\text{Intercept}) =$ expected *outcome* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i}) =$ *proportional* change in expected *outcome* if category i compared to the base-line (intercept) category.
- $\exp(\beta_{\text{x}}) =$ *proportional* change in expected *outcome* if x changes by one unit

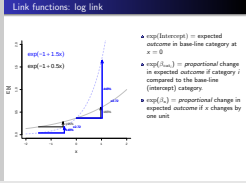


and again, this would be true for any value of the covariate.

Link functions: log link

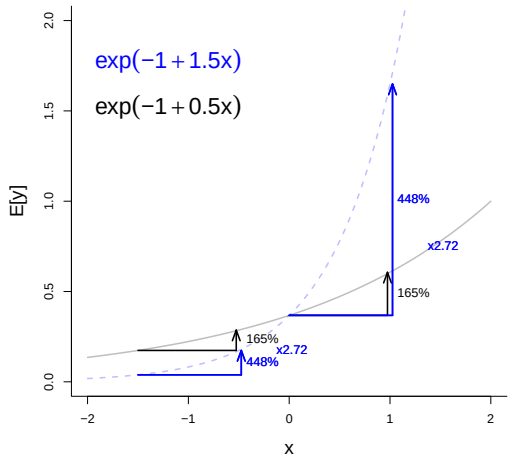


- $\exp(\text{Intercept}) =$ expected *outcome* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i}) =$ *proportional* change in expected *outcome* if category i compared to the base-line (intercept) category.
- $\exp(\beta_{\text{x}}) =$ *proportional* change in expected *outcome* if x changes by one unit

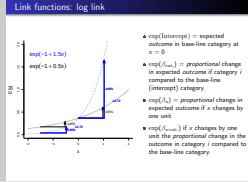


In this example, we have an interaction; the effect of the covariate differs between groups, and for a unit change in the covariate the change in the outcome is is 2.72 times higher in the blue group than it is in the base-line group, and this is true for any value of `x`.

Link functions: log link



- $\exp(\text{Intercept})$ = expected *outcome* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i})$ = *proportional* change in expected *outcome* if category i compared to the base-line (intercept) category.
- $\exp(\beta_{\text{x}})$ = *proportional* change in expected *outcome* if x changes by one unit
- $\exp(\beta_{\text{x:cat}_i})$ if x changes by one unit the *proportional* change in the outcome in category i compared to the base-line category.



If we exponetiate the coefficient associated with the *difference* in slopes, which in this case is 1.5-1 we get this value.

Bernoulli and Binomial

$$E[\mathbf{y}] = \mathbf{X}\beta$$

Bernoulli and Binomial

$E[\mathbf{y}] = \mathbf{X}\beta$

OK - so that was a Poisson glm. Next, we'll consider a binomial glm where we are trying to model the proportion of successes in a number of trials. We have our linear model as before, with the predictor data we've collected organised into this design matrix, which is then multiplied by a vector of location parameters. In a standard linear model the aim is to predict the expected mean of the response directly. However, here our response is value between 0 and 1, and so we need some way to constrain our model to predict values in this interval.

Bernoulli and Binomial

- Link function: logit

$$\text{logit}(E[\mathbf{y}]) = \quad \mathbf{X}\beta$$

Bernoulli and Binomial

- Link function: logit

$\text{logit}(E[\mathbf{y}]) = \mathbf{X}\beta$

There are a huge array of transforms that we could use, but the most natural one and the one most commonly used is the logit transform, which turns values between 0 and 1 into numbers that can range anywhere from minus infinity to positive infinity.

Bernoulli and Binomial

- Link function: logit

$$\begin{aligned} \text{logit}(E[\mathbf{y}]) &= \mathbf{X}\beta \\ E[\mathbf{y}] &= \text{logit}^{-1}(\mathbf{X}\beta) \end{aligned}$$

Bernoulli and Binomial

- Link function: logit

$$\begin{aligned} \text{logit}(E[\mathbf{y}]) &= \mathbf{X}\beta \\ E[\mathbf{y}] &= \text{logit}^{-1}(\mathbf{X}\beta) \end{aligned}$$

Again we could take the inverse of the logit and think about a linear model ($\mathbf{X}\beta$) which is then transformed onto the probability scale using the inverse logit function.

Bernoulli and Binomial

- Link function: logit

$$\begin{aligned} \text{logit}(E[\mathbf{y}]) &= \mathbf{X}\boldsymbol{\beta} \\ E[\mathbf{y}] &= \text{logit}^{-1}(\mathbf{X}\boldsymbol{\beta}) \\ E[\mathbf{y}] &= \texttt{plogis}(\mathbf{X}\boldsymbol{\beta}) \end{aligned}$$

Bernoulli and Binomial

- Link function: logit

$$\begin{aligned} \text{logit}(E[\mathbf{y}]) &= \mathbf{X}\boldsymbol{\beta} \\ E[\mathbf{y}] &= \text{logit}^{-1}(\mathbf{X}\boldsymbol{\beta}) \\ E[\mathbf{y}] &= \texttt{plogis}(\mathbf{X}\boldsymbol{\beta}) \end{aligned}$$

For reasons I don't want to get into now the inverse logit function is the cumulative distribution function (cdf) for the logistic distribution, which is implemented in the R function `plogis` (remember the p prefacing a distribution usually denotes a function that is returning the cdf).

Bernoulli and Binomial

- Link function: logit

$$\begin{aligned} \text{logit}(E[\mathbf{y}]) &= \mathbf{X}\beta \\ E[\mathbf{y}] &= \text{logit}^{-1}(\mathbf{X}\beta) \\ E[\mathbf{y}] &= \texttt{plogis}(\mathbf{X}\beta) \end{aligned}$$

- Distribution: Binomial

$$\mathbf{y} \sim \mathit{Binom}(\texttt{plogis}(\mathbf{X}\beta), n = n)$$

Bernoulli and Binomial

Link function: logit

$$\begin{aligned} \text{logit}(E[\mathbf{y}]) &= \mathbf{X}\beta \\ E[\mathbf{y}] &= \text{logit}^{-1}(\mathbf{X}\beta) \\ E[\mathbf{y}] &= \texttt{plogis}(\mathbf{X}\beta) \end{aligned}$$

Distribution: Binomial

$$\mathbf{y} \sim \mathit{Binom}(\texttt{plogis}(\mathbf{X}\beta), n = n)$$

Given our model for the mean, we need to then to specify what distribution the data come from, which of course is the binomial. Unlike the Poisson the Binomial distribution has an additional term n which is the number of trials that were conducted for which the successes were recorded. However, this number is not a parameter - it is known - and so like the Poisson the Binomial is parameterised solely in terms of the mean; the probability of success.

Bernoulli and Binomial

- Link function: logit

$$\begin{aligned} \text{logit}(E[\mathbf{y}]) &= \mathbf{X}\beta \\ E[\mathbf{y}] &= \text{logit}^{-1}(\mathbf{X}\beta) \\ E[\mathbf{y}] &= \texttt{plogis}(\mathbf{X}\beta) \end{aligned}$$

- Distribution: Binomial

$$\mathbf{y} \sim \mathit{Binom}(\texttt{plogis}(\mathbf{X}\beta), n = n)$$

- Distribution: Bernoulli

$$\mathbf{y} \sim \mathit{Binom}(\texttt{plogis}(\mathbf{X}\beta), n = 1)$$

Bernoulli and Binomial

Link function: logit

$$\begin{aligned} \text{logit}(E[\mathbf{y}]) &= \mathbf{X}\beta \\ E[\mathbf{y}] &= \text{logit}^{-1}(\mathbf{X}\beta) \\ E[\mathbf{y}] &= \texttt{plogis}(\mathbf{X}\beta) \end{aligned}$$

Distribution: Binomial

$$\mathbf{y} \sim \mathit{Binom}(\texttt{plogis}(\mathbf{X}\beta), n = n)$$

Distribution: Bernoulli

$$\mathbf{y} \sim \mathit{Binom}(\texttt{plogis}(\mathbf{X}\beta), n = 1)$$

A special case of the binomial is the Bernoulli where there is only one trial and so the outcomes are either 0 or 1.

Generalised Linear Model: Binomial

```
> photo_glm1 <- glm(cbind(g5, l5) ~ type + ypub, data = photo_long,
+     family = binomial)
```

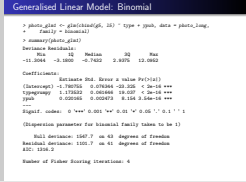

Generalised Linear Model: Binomial

> photo_glm1 <- glm(cbind(g5, l5) ~ type + ypub, data = photo_long,
+ family = binomial)

Here I've fitted a similar analysis to the one we fitted to the average grumpy scores. We allow for a difference between photos scored under grumpy versus happy conditions, and we allow for the grumpy scores to change as a continuous function of ypub; the number of years since the person photographed started publishing. The key difference between this model and the previous one is that the response is no longer the mean grumpy score as assessed by 122 people, but the number of people that gave the photo a score of greater than 5 and the number of people that gave the photo a score of 5 or less. Which ever category comes first within `cbind` (a score of greater than 5 in this case) defines a 'success' and it is the probability of this that we are modelling when we specify `family="binomial"`. Note we give the number in the two categories as the response, rather than something like the average number of successes and the number of trials.

Generalised Linear Model: Binomial

```
> photo_glm1 <- glm(cbind(g5, l5) ~ type + ypub, data = photo_long,
+     family = binomial)
> summary(photo_glm1)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-11.3044   -3.1800   -0.7432    2.9375   12.0952

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.780755   0.076344 -23.325  < 2e-16 ***
typegrumpy   1.173532   0.061646  19.037  < 2e-16 ***
ypub         0.020165   0.002473   8.154 3.54e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1547.7  on 43  degrees of freedom
Residual deviance: 1101.7  on 41  degrees of freedom
AIC: 1316.2

Number of Fisher Scoring iterations: 4
```



Generalised Linear Model: Binomial

> photo_glm1 <- glm(cbind(g5, l5) ~ type + ypub, data = photo_long,
+ family = binomial)
> summary(photo_glm1)
Deviance Residuals:
Min 1Q Median 3Q Max
-11.3044 -3.1800 -0.7432 2.9375 12.0952

Coefficients:
Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.780755 0.076344 -23.325 < 2e-16 ***
typegrumpy 1.173532 0.061646 19.037 < 2e-16 ***
ypub 0.020165 0.002473 8.154 3.54e-16 ***
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1547.7 on 43 degrees of freedom
Residual deviance: 1101.7 on 41 degrees of freedom
AIC: 1316.2

Number of Fisher Scoring iterations: 4

If we run this model and summarise it, our eyes immediately shoot over to the p-values. And everything is very very significant. Perhaps we're not so surprised by the effect of grumpy - we saw it before when we analysed the mean scores, but the effect of ypub is also very significant whereas in the Gaussian model it was not. That seems surprising given that we've essentially dichotomised our continuous data (>5 or <5) before modelling the average, and you would expect that turning continuous data into 0's or 1's must surely result in the loss of information. As before, most of you won't have even bothered to look at the coefficients; you may look at the sign and see that you're more likely to get a grumpy score above 5 if the photo was taken under grumpy conditions, and the grumpy score goes down the longer someone has been publishing, but that's about the limit of any interpretation. This is a shame, because as with the log-link the coefficients with a logit-link have a relatively straightforward interpretation; if you exponentiate them you get the proportional change in the probability of success versus failure; the odds ratio. If we exponentiate the grumpy coefficient we get 3.233, so the number of scores greater than 5 compared to those less than 5 has increased by a factor of 3 for photos taken under grumpy conditions; quite a big effect. If the coefficient is quite close to zero the value after exponentiating is close to the value prior to exponentiating + 1, so we can see that for every year someone has been publishing the probability that their average score is greater than 5 compared to less than 5 decreases by about 2%: `exp(0.02)=1.02`.

You can also see the statement that the dispersion parameter for the binomial is taken to be 1. This is because for the binomial distribution the variance of the number of successes is simply a function of the probability of success and the number of trials (it is $np(1-p)$) so we don't need an additional parameter to model the variance. However, you can see that if we take the residual deviance and divide it by the residual degrees of freedom we get a number far in excess of 1 (27) implying that the variance in the number of success is far greater than what we expect from binomial sampling alone.

Cause of Binomial Overdispersion

Cause of Binomial Overdispersion

When we modelled count data, we thought about overdispersion being generated by variation in the *expected* number of counts that was not captured in our model (by speed-limit and day of year in the case of the Swedish road accidents). In some ways, overdispersion is easier to understand in the Binomial model. We have assumed that variation across photos in the *probability* of someone scoring them above 5 can solely be explained by the `type` of photo and `fpub`.

Cause of Binomial Overdispersion

```
> photo_long[c(13, 19), ]
          y 15  g5   type photo   person age fpub ypub
13 7.885246 15 107 grumpy  4521 darren_o  38 2003   14
19 5.319672 73  49 grumpy  4527  craig_w  38 2003   14
```

Cause of Binomial Overdispersion

> photo_long[c(13, 19),]
y 15 g5 type photo person age fpub ypub
13 7.885246 15 107 grumpy 4521 darren_o 38 2003 14
19 5.319672 73 49 grumpy 4527 craig_w 38 2003 14

For example, the thirteenth and nineteenth photos in our data frame are these two. When we look at the predictor data for these photos we can see that for those terms in the model (type and ypub) these photos have exactly the same values; they were both taken under grumpy conditions and the person start publishing 14 years before I took their photo. Conditional on our model then, the probability of success (a score greater than 5) is assumed to be the same for both photos.

Cause of Binomial Overdispersion

```
> photo_long[c(13, 19), ]
          y l5  g5   type photo   person age fpub ypub
13 7.885246 15 107 grumpy  4521 darren_o  38 2003   14
19 5.319672 73  49 grumpy  4527  craig_w  38 2003   14
```

plogis(1*-1.78+1*1.17+14*0.02)=0.42

Cause of Binomial Overdispersion

> photo_long[c(13, 19),]
y l5 g5 type photo person age fpub ypub
13 7.885246 15 107 grumpy 4521 darren_o 38 2003 14
19 5.319672 73 49 grumpy 4527 craig_w 38 2003 14

plogis(1*-1.78+1*1.17+14*0.02)=0.42

To get the linear predictor we can take the intercept (-1.78), add the `grumpy` coefficient (1.17) and then add 14 multiplied by the `ypub` coefficient (0.02). We can then take the inverse logit transformation (`plogis`) of the linear predictor to get the probability of success (0.42)[1].

[1] Of course, you wouldn't usually do this by hand, you would use the `predict` function in `R`.

Cause of Binomial Overdispersion

```
> photo_long[c(13, 19), ]
          y 15  g5   type photo   person age fpub ypub
13 7.885246 15 107 grumpy  4521 darren_o  38 2003   14
19 5.319672 73  49 grumpy  4527  craig_w  38 2003   14
```

plogis(1*-1.78+1*1.17+14*0.02)=0.42

```
> qbinom(c(0.025, 0.975), size = 122, prob = 0.42)
[1] 41 62
```


Cause of Binomial Overdispersion

> photo_long[c(13, 19),]
y 15 g5 type photo person age fpub ypub
13 7.885246 15 107 grumpy 4521 darren_o 38 2003 14
19 5.319672 73 49 grumpy 4527 craig_w 38 2003 14

plogis(1*-1.78+1*1.17+14*0.02)=0.42
> qbinom(c(0.025, 0.975), size = 122, prob = 0.42)
[1] 41 62

We can then ask, if the probability of success is 0.42 and the number of trials is 122 (15+107=73+49=122) then in what range do we expect the number of successes to be 95% of the time: what are the lower 2.5% and upper 2.5% quantiles of the distribution? And the answer is somewhere between 41 and 62. The second photo, of Craig, is consistent with this since 49 respondents gave this photo a score greater than 5. However, the number of respondents that gave the first photo, of Darren, a score greater than 5 is exceptionally high (107) if the true probability of success is really 0.42. If we looked at the scores for more photos with these properties (photos taken under grumpy conditions of people who had published their first paper 14 years ago) then we might find that although on average the proportion of scores greater than 5 was close to 0.42, there might be several photos where the proportion of scores was improbably high, as with Darren's, or improbably low. The variance in the scores across photos with the same properties would be greater than what we expect given a probability of 0.42 (the expected variance is np(1-p)=29.71).

In this example, I think it is easy to see why this might be the case. Perhaps Darren looks intrinsically grumpier than Craig? Perhaps wearing glasses makes some one look more grumpy? Because the photo of Darren was taken just at the moment his eyes were half closed, perhaps this makes him look more grumpy? We can imagine a great number of reasons why the probability of getting a score greater than 5 exceeds 0.42 for this particular photo of Darren. Essentially, the overdispersion is caused by the probability of success varying over photos in ways that are not captured by the model.

Generalised Linear Model: Binomial

```
> photo_glm2 <- glm(cbind(g5, l5) ~ type + ypub, data = photo_long,
+     family = quasibinomial)
```

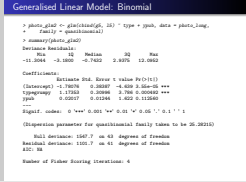

Generalised Linear Model: Binomial

> photo_glm2 <- glm(cbind(g5, l5) ~ type + ypub, data = photo_long,
+ family = quasibinomial)

We can fit a quasibinomial model that deals with this excess variation by inflating the sampling variances of the coefficients by the excess

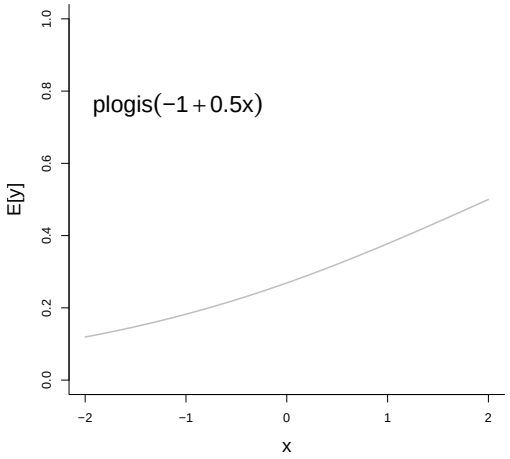
Generalised Linear Model: Binomial

```
> photo_glm2 <- glm(cbind(g5, l5) ~ type + ypub, data = photo_long,
+     family = quasibinomial)
> summary(photo_glm2)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-11.3044   -3.1800   -0.7432    2.9375   12.0952

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.78076    0.38387  -4.639 3.55e-05 ***
typegrumpy   1.17353    0.30996   3.786 0.000492 ***
ypub         0.02017    0.01244   1.622 0.112560
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasibinomial family taken to be 25.28215)

    Null deviance: 1547.7  on 43  degrees of freedom
Residual deviance: 1101.7  on 41  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 4
```

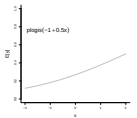


Generalised Linear Model: Binomial

and we can see that the dispersion parameter is close to 27, the standard errors have been inflated by the square-root of the dispersion parameter, and the p-values are much less extreme. However, again no AIC is reported because the quasi model is not really specifying a distribution for the response, its just correcting our standard errors so they are more reasonable. When we modelled count data, we thought about overdispersion being generated by variation in the *expected* number of counts that was not captured in our model (by speed-limit and day of year in the case of the Swedish road accidents). Using this idea, we then modelled the distribution of the *expected* number of counts as gamma-distributed which leads us to the negative binomial model for the *actual* number of counts. In some ways, overdispersion is even easier to understand in the binomial model; we have assumed that variation amongst photos in the *probability* of someone scoring them above 5 can solely be explained by the type of photo and ypub. If we took two photos of the same type, and of two people who had been publishing the same amount of time, this model assumes that any differences in the *actual* number of times each photo had been scored greater than 5 is due to binomial sampling around the same probability. As with the Poisson model, it is easy to imagine that the probability is very likely to vary from photo to photo due to other things not in the model, for example the probability might vary because some people look grumpier than others, or because someone had just been told an exceptionally bad bit of news and were exceptionally grumpy. A beta-binomial model is probably the most natural way of allowing the underlying probabilities to vary. Like the negative binomial, you can think about pulling p from a distribution (the beta distribution) and then given p you sample the actual numbers of success and failures using a binomial. However, beta-binomial models are not common because they are a bit of a pain to use, but I'll show you a conceptually similar approach later when we discuss mixed models. For now, let's go back to interpreting the coefficients when we use the logit link.

Link functions: logit link

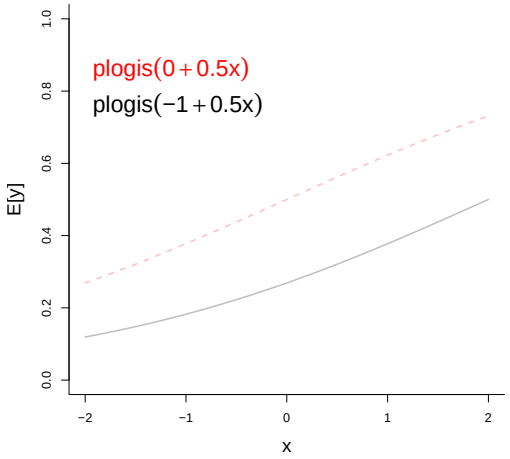


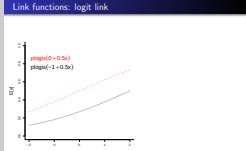
Link functions: logit link



On the y-axis we have our distribution parameter (the binomial probability, p) and on the x-axis we have the value of some covariate x. We have a logit-linear model with an intercept of -1 and a coefficient of 0.5 associated with the covariate.

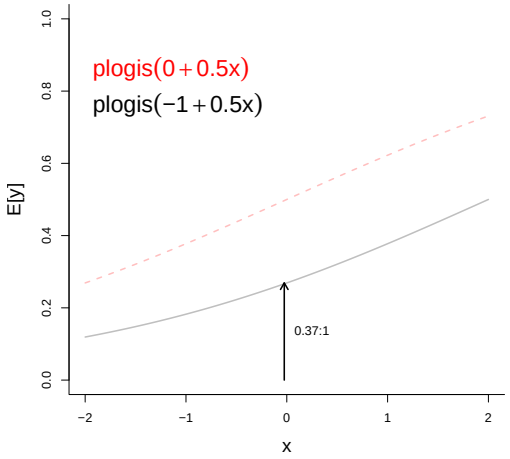
Link functions: logit link

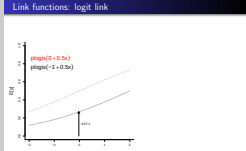




We can also imagine that the observations belong to two groups, and the probability of success for the red group follow this model where the intercept is one unit higher.

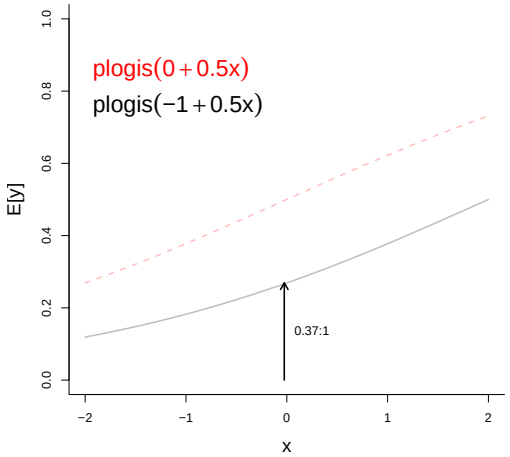
Link functions: logit link



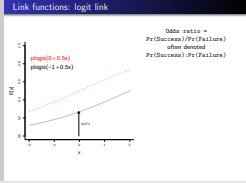


If we exponentiate the intercept we get a value `exp(-1)`=0.37 implying that when `x`=0 we should see 0.37 successes for every failure. Note that `exp(-1)` is NOT the probability of success, that is given by `plogis(-1)`=0.27, exponentiating the coefficients tells us

Link functions: logit link

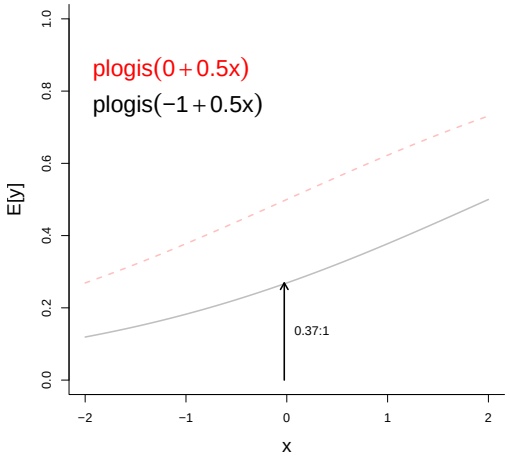


`Odds ratio = Pr(Success)/Pr(Failure) often denoted Pr(Success):Pr(Failure)`



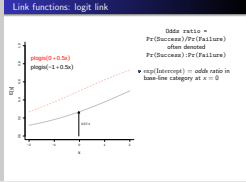
something about the odds-ratio, which is the probability of success divided by the probability of failure. This quantity is well understood by gamblers, where its usually referred to as simply the 'odds' or the 'betting odds' and its usually scaled to be more interpretable. So for example if you have 2:1 odds your twice as likely to win as loose.

Link functions: logit link



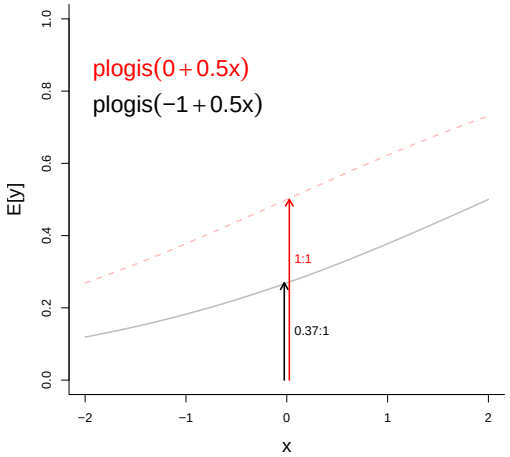
`Odds ratio = Pr(Success)/Pr(Failure) often denoted Pr(Success):Pr(Failure)`

- $\exp(\text{Intercept}) =$ *odds ratio* in base-line category at $x = 0$



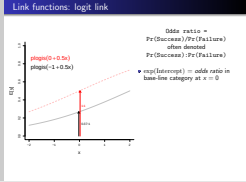
so exponentiating the intercept gives us the odds ratio when x=0.

Link functions: logit link



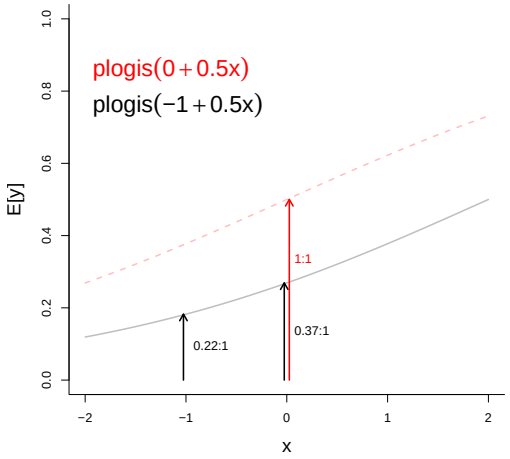
`Odds ratio = Pr(Success)/Pr(Failure)` often denoted `Pr(Success):Pr(Failure)`

- $\exp(\text{Intercept}) =$ *odds ratio* in base-line category at $x = 0$



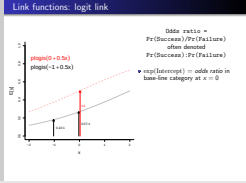
`exp(0)=1` and so for the red group the odds of success are 50:50 (1:1)

Link functions: logit link



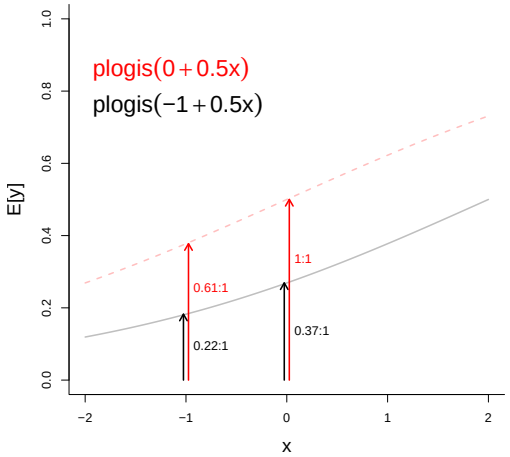
Odds ratio = Pr(Success)/Pr(Failure) often denoted Pr(Success):Pr(Failure)

- $\exp(\text{Intercept}) =$ *odds ratio* in base-line category at $x = 0$



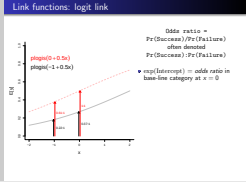
If you calculate the odds at different values of the covariate, clearly they change, so down here a success for the black group is 4 or 5 times less likely than a failure

Link functions: logit link



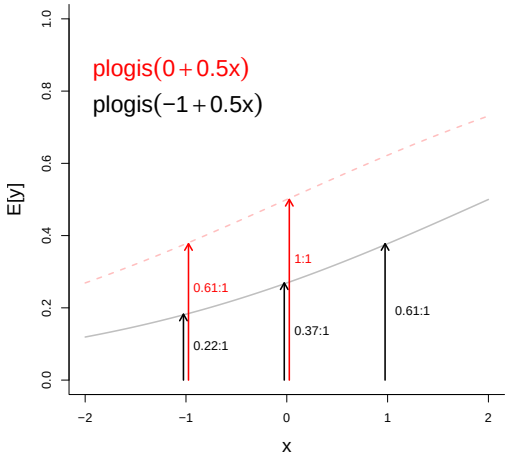
`Odds ratio = Pr(Success)/Pr(Failure) often denoted Pr(Success):Pr(Failure)`

- $\exp(\text{Intercept}) =$ *odds ratio* in base-line category at $x = 0$



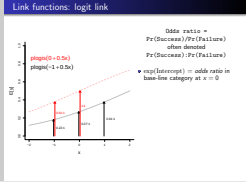
and for the red group a success is now 60% as likely as a failure.

Link functions: logit link



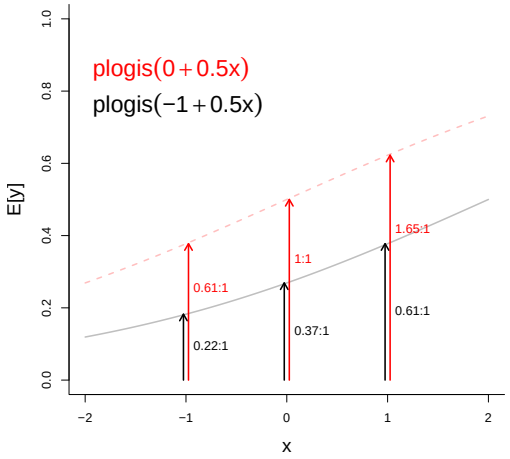
`Odds ratio = Pr(Success)/Pr(Failure) often denoted Pr(Success):Pr(Failure)`

- $\exp(\text{Intercept}) =$ *odds ratio* in base-line category at $x = 0$



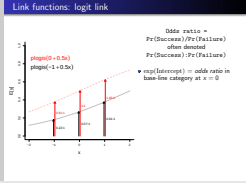
For higher values of x the odds increase

Link functions: logit link



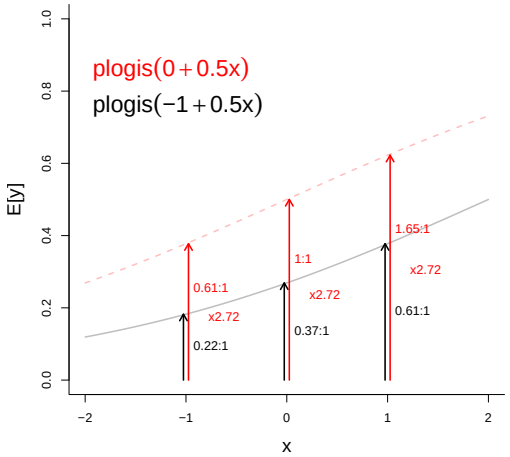
`Odds ratio = Pr(Success)/Pr(Failure)` often denoted `Pr(Success):Pr(Failure)`

- $\exp(\text{Intercept}) =$ *odds ratio* in base-line category at $x = 0$



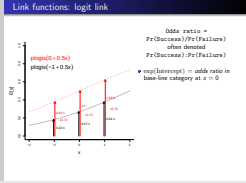
in both cases, but there's not an obvious relationship between the three sets of odds for the red group and the black group.

Link functions: logit link



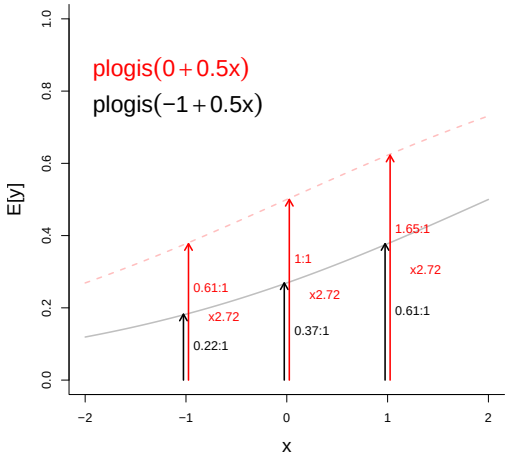
`Odds ratio = Pr(Success)/Pr(Failure)` often denoted `Pr(Success):Pr(Failure)`

- $\exp(\text{Intercept}) =$ *odds ratio* in base-line category at $x = 0$



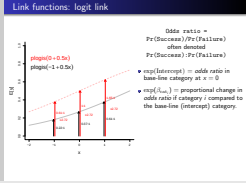
However, if you look at the relative change in odds, it is constant across the range of `x`; the odds of success have increased by a factor of 2.72 in each case.

Link functions: logit link



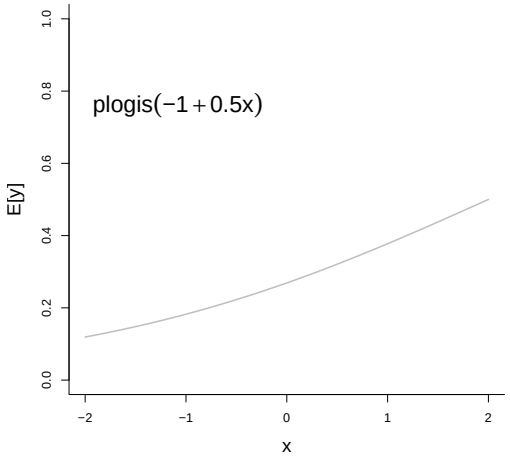
`Odds ratio = Pr(Success)/Pr(Failure)` often denoted `Pr(Success):Pr(Failure)`

- $\exp(\text{Intercept}) =$ *odds ratio* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i}) =$ proportional change in *odds ratio* if category *i* compared to the base-line (intercept) category.



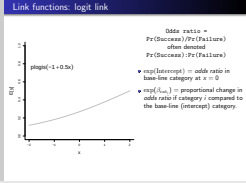
Its 2.72 because the difference in the intercepts of the two groups is 1 (0 - -1 = 1) and `exp(1)`=2.72. So if we exponentiate a coefficient associated with a *difference* between two groups (i.e. a coefficient associaited with a categorical predictor) we get the relative change in odds: you're 2.72 times as likely to win versus lose. In this example, the change in odds is constant across x because there's no interaction, the coefficient associated with x is the same in both groups (0.5).

Link functions: logit link



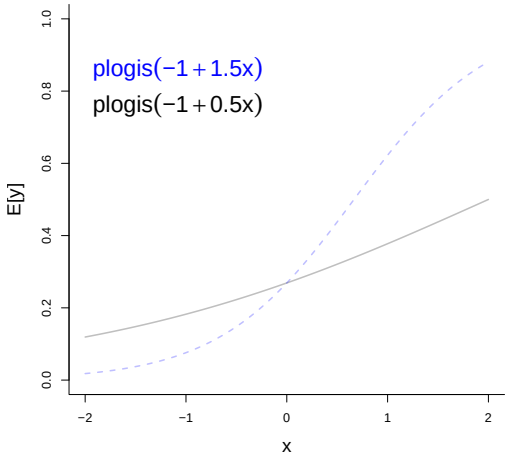
`Odds ratio = Pr(Success)/Pr(Failure)` often denoted `Pr(Success):Pr(Failure)`

- $\exp(\text{Intercept}) =$ *odds ratio* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i}) =$ proportional change in *odds ratio* if category *i* compared to the base-line (intercept) category.



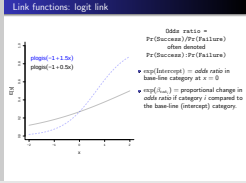
Let's imagine another scenario where the logit-linear model is the same for the black group,

Link functions: logit link



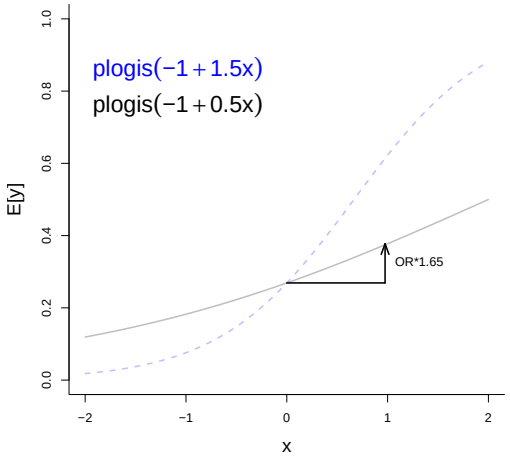
`Odds ratio = Pr(Success)/Pr(Failure)` often denoted `Pr(Success):Pr(Failure)`

- $\exp(\text{Intercept}) =$ *odds ratio* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i}) =$ proportional change in *odds ratio* if category *i* compared to the base-line (intercept) category.



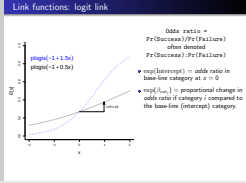
but we now have a blue group with the same intercept, but a different slope.

Link functions: logit link



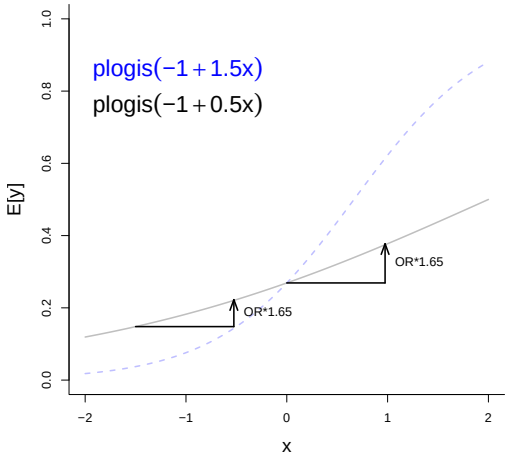
`Odds ratio = Pr(Success)/Pr(Failure)` often denoted `Pr(Success):Pr(Failure)`

- $\exp(\text{Intercept}) =$ *odds ratio* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i}) =$ proportional change in *odds ratio* if category *i* compared to the base-line (intercept) category.



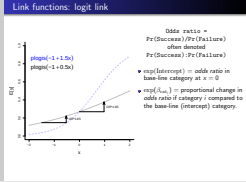
If we look at the odds-ratio for the black group when x has increased by one unit from 0, we see that the odds ratio has increased by a factor of 1.65

Link functions: logit link



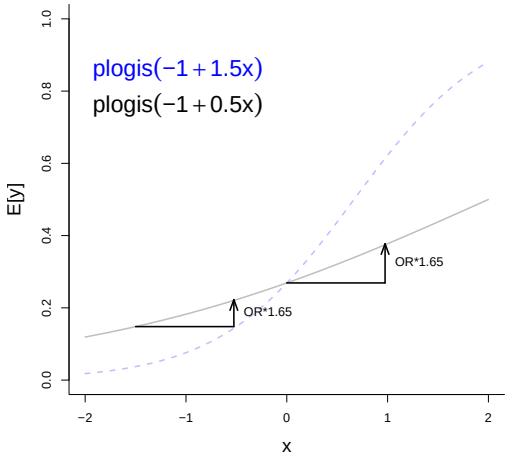
`Odds ratio = Pr(Success)/Pr(Failure)` often denoted `Pr(Success):Pr(Failure)`

- $\exp(\text{Intercept}) =$ *odds ratio* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i}) =$ proportional change in *odds ratio* if category *i* compared to the base-line (intercept) category.



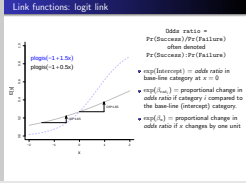
and this is true if you compare the odds ratio at any two values of `x` that differ by one unit.

Link functions: logit link



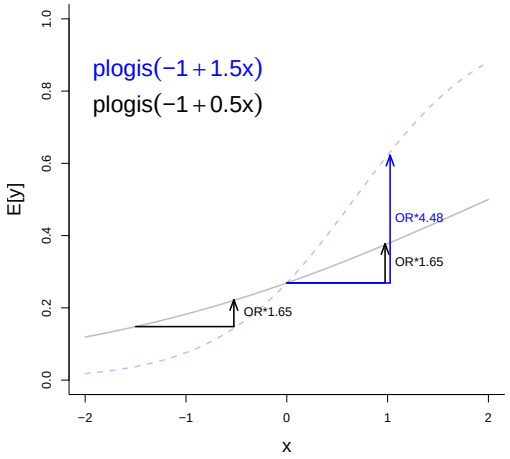
`Odds ratio = Pr(Success)/Pr(Failure)` often denoted `Pr(Success):Pr(Failure)`

- $\exp(\text{Intercept}) =$ *odds ratio* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i}) =$ proportional change in *odds ratio* if category *i* compared to the base-line (intercept) category.
- $\exp(\beta_{\text{x}}) =$ proportional change in *odds ratio* if x changes by one unit



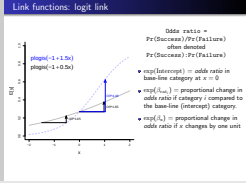
We get 1.65 by expoentiating the coefficient associated with the slope `0.5`.

Link functions: logit link



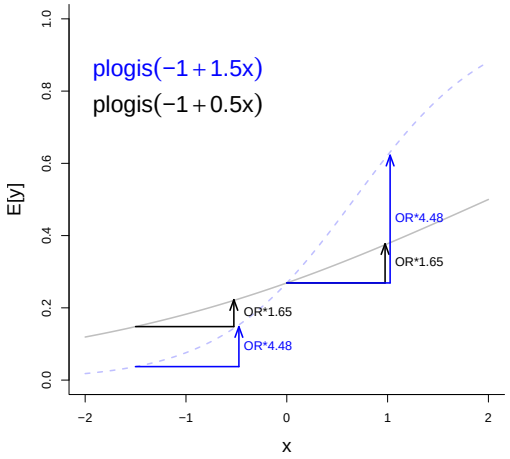
`Odds ratio = Pr(Success)/Pr(Failure)` often denoted `Pr(Success):Pr(Failure)`

- $\exp(\text{Intercept}) =$ *odds ratio* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i}) =$ proportional change in *odds ratio* if category *i* compared to the base-line (intercept) category.
- $\exp(\beta_{\text{x}}) =$ proportional change in *odds ratio* if x changes by one unit



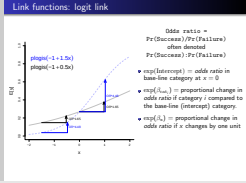
The slope for the blue group is 1.5 and so if we exponentiate this we see we get a 4.48 fold increase in the odds ratio by increase the value of x by one unit,

Link functions: logit link



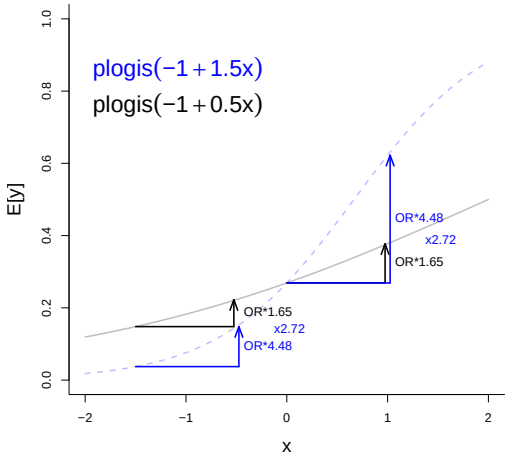
`Odds ratio = Pr(Success)/Pr(Failure)` often denoted `Pr(Success):Pr(Failure)`

- $\exp(\text{Intercept}) =$ *odds ratio* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i}) =$ proportional change in *odds ratio* if category *i* compared to the base-line (intercept) category.
- $\exp(\beta_{\text{x}}) =$ proportional change in *odds ratio* if *x* changes by one unit



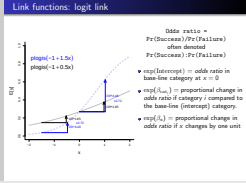
and this holds across the range of x.

Link functions: logit link



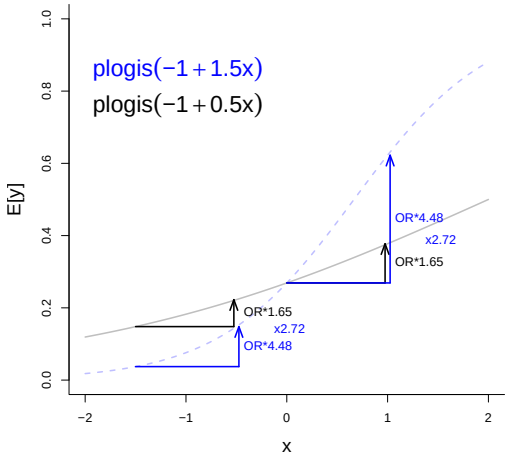
`Odds ratio = Pr(Success)/Pr(Failure)` often denoted `Pr(Success):Pr(Failure)`

- $\exp(\text{Intercept}) =$ *odds ratio* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i}) =$ proportional change in *odds ratio* if category *i* compared to the base-line (intercept) category.
- $\exp(\beta_{\text{x}}) =$ proportional change in *odds ratio* if x changes by one unit



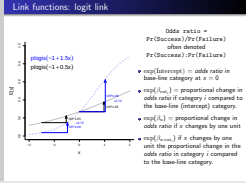
In this example, we have an interaction as the slopes differ between groups, and for a unit change in the covariate the change in the odds is 2.72 times higher in the blue group than it is in the base-line group, and this is true for any value of `x`.

Link functions: logit link



`Odds ratio = Pr(Success)/Pr(Failure)` often denoted `Pr(Success):Pr(Failure)`

- $\exp(\text{Intercept}) =$ *odds ratio* in base-line category at $x = 0$
- $\exp(\beta_{\text{cat}_i}) =$ proportional change in *odds ratio* if category *i* compared to the base-line (intercept) category.
- $\exp(\beta_{\text{x}}) =$ proportional change in *odds ratio* if x changes by one unit
- $\exp(\beta_{\text{x:cat}_i})$ if x changes by one unit the proportional change in the *odds ratio* in category *i* compared to the base-line category.



If we exponentiate the coefficient associated with the *difference* in slopes, which in this case is 1.5-0.5=1 we get this value.