GLMs with a Poisson

Bob O'Hara

Contents

The Poisson distribution and log-linear models

First, a video.

link

This week you will:

- learn about log-linear models
- learn about over-dispersion
  - when there is more error than you expect

A Model for Counts: Fishing

Imagine sitting on the banks of the river Seine in Paris, fishing. Fish swim past randomly at a constant rate. If we catch fish for an hour, how many fish do we catch? Because they swim past randomly, the number will vary. it will also depend on the density of fish (and how fast they swim etc. etc.). But if we catch them at rate μ, the mean number we catch in time t will be $\lambda = \mu t$. The actual number will vary, and will follow a Poisson distribution:

$$Pr(N = r|\lambda) = \frac{\lambda^r e^{-\lambda}}{r!}$$

Figure 1: Anglers by Raoul Dufy

Now, as some of you may know, the French for fish is poisson (and the French call April 1st poisson d'avril, so beware). But the Poisson distribution is actually named after *Baron* Siméon Denis Poisson.

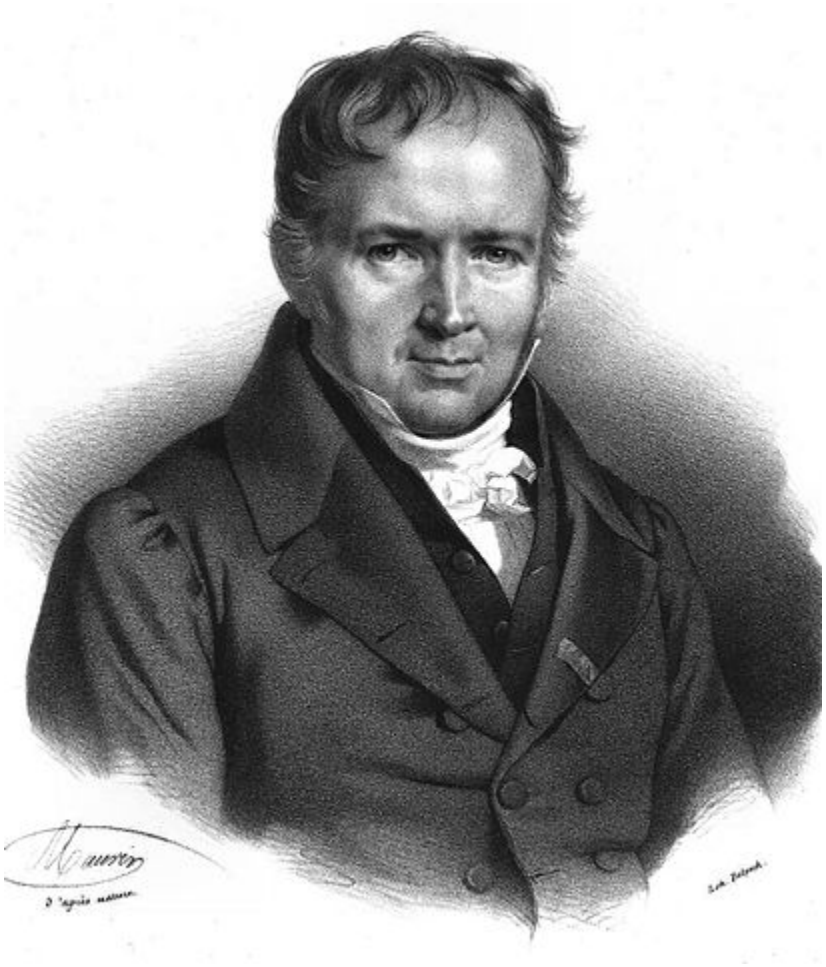


Figure 2: Siméon Denis Poisson

By François Séraphin Delpech - http://web4.si.edu/sil/scientific-identity/display_results.cfm?alpha_sort=W, Public Domain, https://commons.wikimedia.org/w/index.php?curid=536305

Count data is common, for example the numbers of murders, offspring, bacterial/fungal colonies, and of course the number of people infected with a disease. So the ideas we discuss here will have a wide range of uses, and also form the basis of a lot of different models for different types of data.

The Poisson distribution

So, what does the Poisson distribution look like? You can take a look! Here is some code to simulate 1000 data points from a Poisson distribution with a mean of 1.5. The `plot(table(...))` line of code plots the frequency of each count, e.g. the number of simuations with 0 counts (this is a histogram, but the plots are a bit nicer than using `hist()`):

```
Mean <- 1.5
Pois <- rpois(1000, Mean)
plot(table(Pois), lwd=8, lend=3)
```

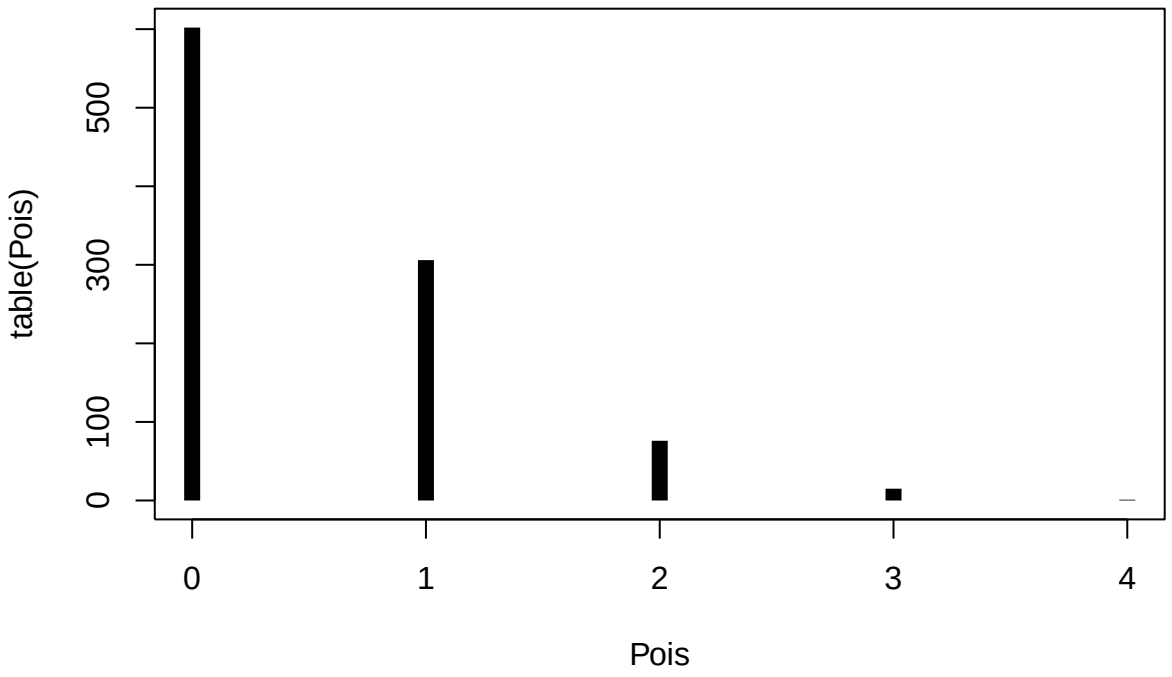
Use this code and change the value of Mean to see what happens to the shape of the distribution when

- the mean is less than 1?
- the mean equals 1
- the mean is above 1
- the mean gets large (i.e. a lot bigger than 1: chose your own values)?

Answer: the mean is less than 1?

We can try with 0.5:

```
Mean <- 0.5
Pois <- rpois(1000, Mean)
plot(table(Pois), lwd=8, lend=3)
```

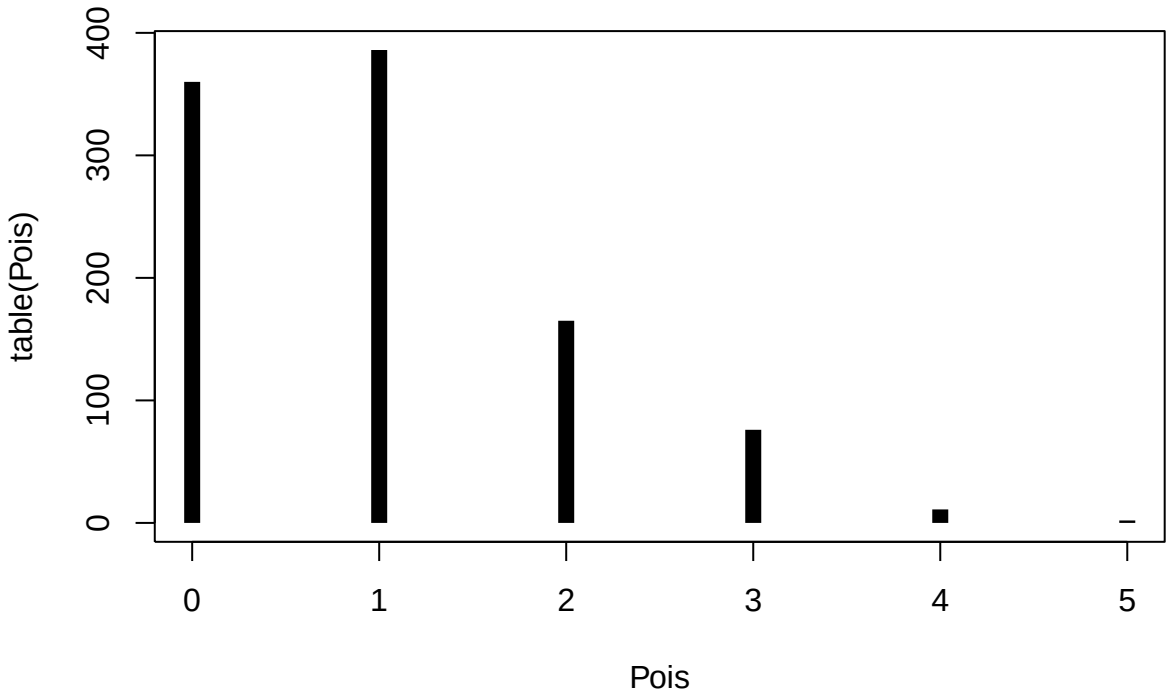


The most common value is 0, and as the number of counts goes up, their frequency decreases.

Answer: the mean equals 1?

Now we plug in 1 for `Mean`:

```
Mean <- 1
Pois <- rpois(1000, Mean)
plot(table(Pois), lwd=8, lend=3)
```

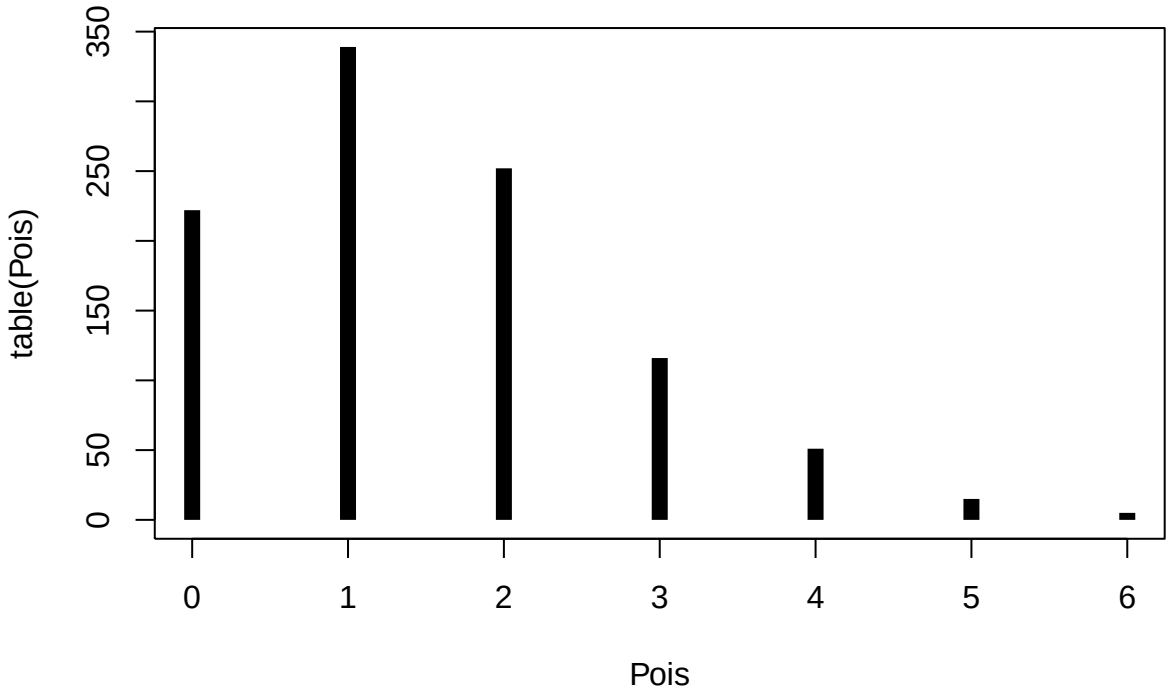


Both 0 and 1 are equally common (you will see some random variation, of course. If you run the code a few times, you'll see that the heights of the lines for 0 and 1 are usually about the same, but either can be a bit larger), and as the number of counts goes up, their frequency decreases.

Answer: the mean is larger than 1?

We can try a mean of 1.5:

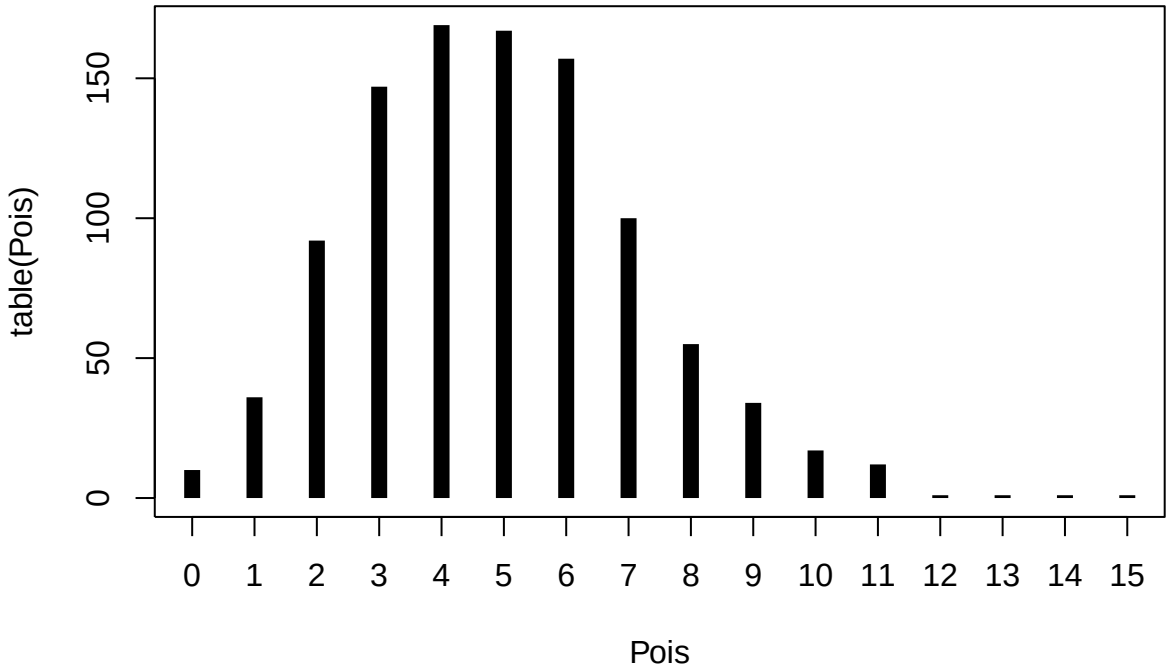
```
Mean <- 1.5
Pois <- rpois(1000, Mean)
plot(table(Pois), lwd=8, lend=3)
```



Here the most common value (the mode) is 1, and the distribution tails off above that.

How about a mean of 5?

```
Mean <- 5
Pois <- rpois(1000, Mean)
plot(table(Pois), lwd=8, lend=3)
```



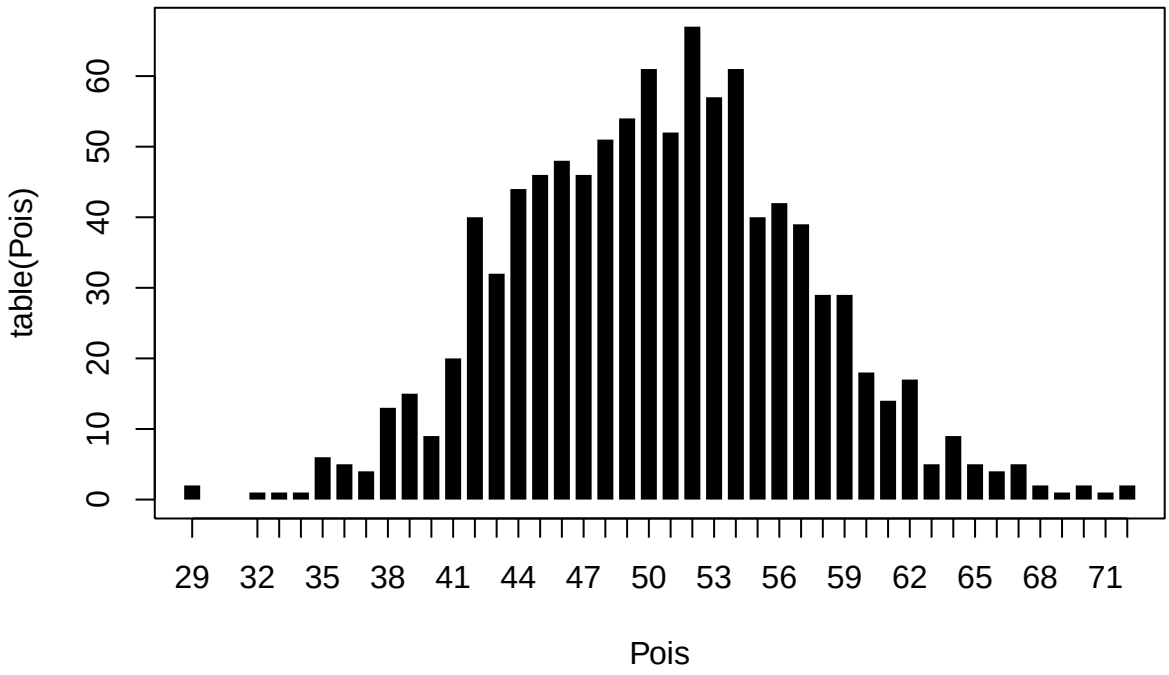
Now the mode is about 5 (but 4 and 6 are also fairly likely, so they might have larger values in another simulation). Again, this tails off for larger values.

Also notice that this distribution is positively skewed - above the mode the values spread out more.

Answer: the mean is large?

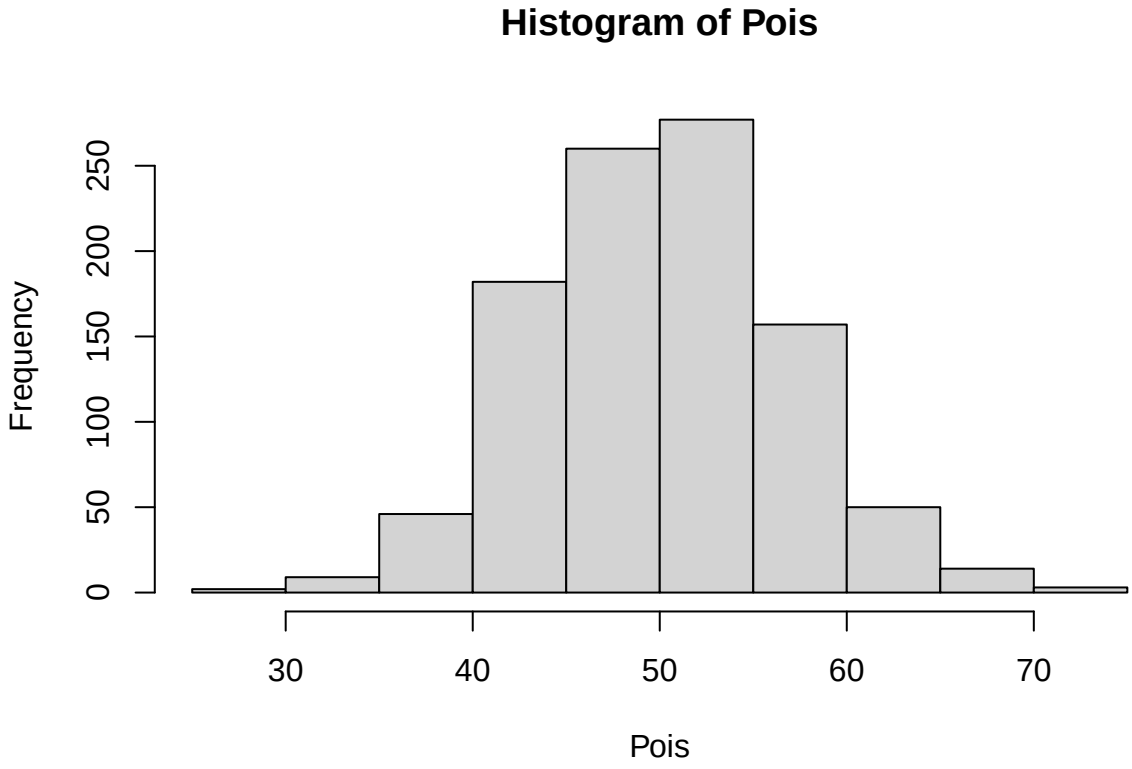
Let's go big with 50:

```
Mean <- 50
Pois <- rpois(1000, Mean)
plot(table(Pois), lwd=8, lend=3)
```



The plot looks messier, so now a histogram does work better:

```
hist(Pois)
```



The histogram looks more like a normal distribution.

Some useful facts about the Poisson distribution

Some useful facts about the Poisson distribution. You'll be AMAZED at number 4

1. **When the mean is large, it is approximately a normal distribution.** This means that if you have large counts, you might as well model the data with a normal distribution (it is also commom to log transform it first).
2. **If two random variables are Poisson distributed, the number of one type conditioned on the total follows a binomial distirbution**. OK, I need to explain this. Imaagine we were back to catching Parisian fish, and we catch two species, say perch and pike. We might be interested in what proportion are perch. If both follow Poisson distributions, and we catch N_{perch} and N_{pike}, then N_{perch} *given* $N_{\text{perch}} + N_{\text{pike}}$ follows a binomial distribution, with $P = \lambda_{\text{perch}}/(\lambda_{\text{perch}} + \lambda_{\text{pike}})$. The upshot of this is that if we have a data analysis problem where we want to look at proportions, we can actually use a Poisson distribution instead. Whether we want to depends on us: sometimes it is easier to use a Poisson, and sometimes a binomial.
3. Following on from this, if we have a binomial distribution with a small p, the number of "successes" approximately follows a Poisson distribution with $\lambda = Np$. This is also useful: in epidemiology there are a lot of rare diseases, and it is easier to model the numbers rather than proportions, and to talk about risks as proportions (e.g. "relative risk"). So diseases can be (say) twice as common.

Is the Poisson a GLM?

The log-likelihood is $l(N = r|\lambda) = r \log \lambda - \lambda - log(t!)$, so how does this compare to a GLM likelihood? Let's use some colours!

The log-likelihood:

$$l(N = r|\lambda) = r\log \lambda - \lambda - \log(t!)$$

The GLM likelihood:

$$l(\theta|y) = \frac{y\theta - b(\theta)}{a(\phi)} - c(\phi, y)$$

So $a(\phi) = 1$

and $\theta = \log \lambda$

Thus this is a GLM, and the natural link function is a log link. It is very rare than any other link function is used.

Interpretation

The log link also makes the interpretation straightforward. If we are counting something, the process is multiplicative. So if we double the effort, we double the number of counts we expect. On the log scale this is additive:

$$\log(\lambda) = \alpha + \beta x$$
$$\lambda = e^{\alpha+\beta x} = e^{\alpha} e^{\beta x}$$

if we make x a dummy variable, i.e. 0 or 1, then if β doubles the mean (i.e. $\lambda = 2e^{\alpha}$), $\beta = \log(2) = 0.69$.

This is why these models are sometimes called **log-linear models**, because they are linear on the log scale.

There are some additional aides to interpretaion. If a coefficient is small, it is (approximately) the percent increase. So $e^{\alpha+\beta}$ means an increase by $e^{\beta} \approx 1 + \beta$ times (if β is small). So a coefficient of 0.01 is equal to about a 1% increase. A coefficient of 0.1 is about a 10.5% increase, so the approximation is getting worse.

Also, the coefficients are symmetrical. A value of +0.01 increases the mean by $e^{0.01}$ times, and a value of -0.01 *decreases* the mean by $e^{0.01}$ times. This means that the sign only shows the direction, the absolute value show the strength.

Model Fitting

Model fitting is easy, we can use the `glm()` function, almost like we did with the binomial distribution:

```
mu <- seq(1,2, length=10)
Count <- rpois(length(mu), mu)
m1 <- glm(Count ~1, family=poisson("log"))
m1a <- glm(Count ~1, family="poisson")
```

We just need to tell R to use the Poisson family. It will use the log link by default. We can use `summary()`, `coef()`, `confint()`, `anova()` etc just as we have before.

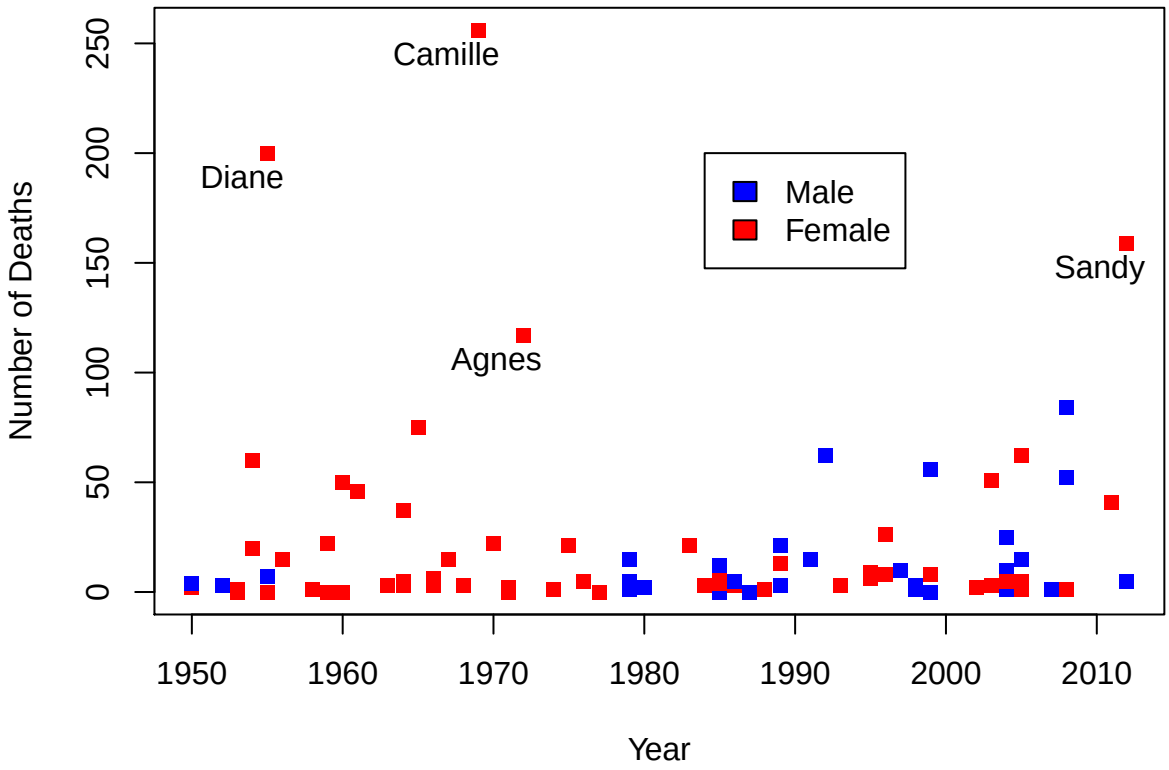
An Example: Himmicanes

A few years a go a strange paper appeared in PNAS that suggested that hurricanes in the USA with female names caused more deaths than those with male names. This was rather a surprising result, so was worth checking. Fortunately, the authors of the paper made their data available. This is the data (you will need this code soon):

```
File <- "https://www.math.ntnu.no/emner/ST2304/2019v/Week13/Himmicanes.csv"
HimmData <- read.csv(File, stringsAsFactors=FALSE)
# Select hurricanes with > 100 deaths
BigH <- which(HimmData$alldeaths>100)
# Make a factor out of the Gender variable
HimmData$GenderF <- factor(c("Male", "Female")[1+HimmData$Gender])
```

First we can plot the data:

```
plot(HimmData$Year, HimmData$alldeaths, col=HimmData$ColourMF,
     type="p", pch=15, xlab="Year",
     ylab="Number of Deaths")
text(HimmData$Year[BigH], HimmData$alldeaths[BigH],
     HimmData$Name[BigH], adj=c(0.8,1.5))
legend(1984,200,c("Male","Female"),fill=c("blue","red"))
```



We can see that the 4 hurricanes that caused the most death had female names. But also note that 3 of them occured between 1956 and 1978, when only female names were used.

The data contains the following variables:

- **Year**: Year
- **Name**: Hurricane's name
- **Gender**: Gender (0: Male, 1: Female)
- **MasFem**: A scoring of how feminine the name sounds. We won't use this here
- **Minpressure**: minimum air pressure in the hurricane (a measure of stength)
- **Category**: Category of hurricane (larger is more severe)
- **NDAM**: Normalised damage (i.e. how much the hurricane cost, corrected for inflation etc.)
- **alldeaths**: Number of deaths

The aim is to predict the number of deaths, and see if it is explaned by the gender assigned to the hurricane. It should also depend on the severity of the hurricane, which we can measure with minimum pressure, normalised damage or Category.

We will model the number of deaths. these are counts, so a Poisson distribution is called for. This then suggests a log link function. But which variables to use? Using Gender rather than MasFem will make the interpretation easier (and doesn't change the results), so we will us that. But should we use minimum pressure, normalised damage or Category? Note that all 3 are correlated with each other, as we might expect. So you should pick one to use in this analysis.

You should fit the model, using whichever variable you decided to use. You only need to use the main effects, no interactions.

Fit the model! What estimates do you get? In particular, is there an effect of gender?

Panic Button, if you want a hint

Your code should look like this:

```
A_Model <- glm(Count ~X, data=SomeData, family="poisson")
The_Same_Model <- glm(Count ~X, data=SomeData, family=poisson())
```

You need to work out what to plug in where.

Answers if you used minimum pressure

This is the code to fit the model, and the summary:

```
mod.minpres <- glm(alldeaths ~ GenderF+Minpressure, family="poisson", data=HimmData)
summary(mod.minpres)
```

```
##
## Call:
## glm(formula = alldeaths ~ GenderF + Minpressure, family = "poisson",
##     data = HimmData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.0023  -3.9983  -2.3865   0.2403  30.2263
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 39.302945   1.086442  36.176   <2e-16 ***
## GenderFMale -0.472112   0.054982  -8.587   <2e-16 ***
## Minpressure -0.037740   0.001143 -33.012   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 4031.9  on 91  degrees of freedom
## Residual deviance: 2882.4  on 89  degrees of freedom
## AIC: 3213.3
##
## Number of Fisher Scoring iterations: 6
```

It suggests a large effect of gender: -0.47, which suggests that a male hurricane has exp(-0.47) = 0.62 the deaths. Or, equivalently, a female hurricane has exp(- -0.47) = 1.6 times more deaths. The confidence interval is -0.58 to -0.37, which would suggest the effect is definitely large.

Notice also that a higher pressure leads to less deaths - a change of 1 Mb changes the expected number of deaths by exp(-0.04) = 0.96 times. A lower presure means a stronger hurricane, so this makes intuitive sense. Also note that 0.96 is roughly a 4% decrease, so (as noted above) we can get an approximate estimate of the percent change from just from the estimate.

Answers if you used hurricane Category

With Category we should probably use it as a factor. Although it is ordered (i.e. a category 4 hurricane is more powerful than a category 3), we don't know if the change from category 2 to category 3 should be the same as from category 3 to category 4. If we treat it as continuous, we could use polynomials to adjust for this, but we can "spend" a couple of parameters by making it categorical, and not having to worry.

```
mod.Cat <- glm(alldeaths ~ GenderF+factor(Category), family="poisson", data=HimmData)
summary(mod.Cat)
```

```
##
## Call:
```

```
## glm(formula = alldeaths ~ GenderF + factor(Category), family = "poisson",
##     data = HimmData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -7.4087  -4.8534  -3.1476   0.3887  25.5639
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        2.74213    0.04883  56.159  < 2e-16 ***
## GenderFMale       -0.62280    0.05574 -11.172  < 2e-16 ***
## factor(Category)2  0.57003    0.06482   8.794  < 2e-16 ***
## factor(Category)3  0.21635    0.06567   3.295 0.000985 ***
## factor(Category)4  0.84395    0.09479   8.903  < 2e-16 ***
## factor(Category)5  2.59045    0.07318  35.398  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 4031.9  on 91  degrees of freedom
## Residual deviance: 2977.4  on 86  degrees of freedom
## AIC: 3314.3
##
## Number of Fisher Scoring iterations: 6
```

Which suggests a large effect of gender: -0.62, which suggests that a male hurricane has exp(-0.62) = 0.54 the deaths. Or, equivalently, a female hurricane has exp(- -0.62) = 1.86 times more deaths. The confidence interval is -0.73 to -0.51, which woud suggest the effect is large.

Also notice that the effects of category become more positive as the category goes up. In other words, higher categories lead to more deaths as we might expect: higher categories are stronger hurricanes.

Answers if you used normalised damage

Finally, damage:

```
mod.NDAM <- glm(alldeaths ~ GenderF+NDAM, family="poisson", data=HimmData)
summary(mod.NDAM)
```

```
##
## Call:
## glm(formula = alldeaths ~ GenderF + NDAM, family = "poisson",
##     data = HimmData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.8660  -4.2486  -3.0442   0.4558  24.0392
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.716e+00  3.178e-02  85.480   <2e-16 ***
## GenderFMale -5.368e-01  5.496e-02  -9.766   <2e-16 ***
## NDAM         3.630e-05  8.576e-07  42.333   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 4031.9  on 91  degrees of freedom
## Residual deviance: 2731.7  on 89  degrees of freedom
## AIC: 3062.6
##
## Number of Fisher Scoring iterations: 6
```

Which suggests a large effect of gender: -0.54, which suggests that a male hurricane has exp(-0.54) = 0.58 the deaths. Or, equivalently, a female hurricane has exp(- -0.54) = 1.71 times more deaths. The confidence interval is -0.65 to -0.43, which woud suggest the effect is large.

Also notice that the effect of damage is positive, so more damage means more deaths.

Summary of Answers

Whichever hurricane severity measure you use, you find that stronger hurricanes lead to more deaths, which is not surprising. What is surprising is that gender seems to have a large effect, and the narrow coinfidence intervals suggest we can be fairly certain about it. But this is not the full story.

For those poor souls who want to hear me explain the answers, here is the video

Model Comparison

We can compare models in the same way as before.

```
mod.null <- glm(alldeaths ~ 1, family="poisson", data=HimmData)
mod.onlyNDAM <- glm(alldeaths ~ NDAM, family="poisson", data=HimmData)
anova(mod.null, mod.onlyNDAM, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: alldeaths ~ 1
## Model 2: alldeaths ~ NDAM
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1        91     4031.9
## 2        90     2836.3  1   1195.7 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Compare the models with and without gender: does there seem to be an effect?

Hint

Fit and compare these models (for NDAM, but this also works for the others): `alldeaths ~ NDAM` and `alldeaths ~ GenderF+NDAM`

Answer

Yes, there definitely seems to be an effect. The smallest deviance is about 80, with 1 degree of freedom:

```
# Fit the additional models we need
mod.onlyminpres <- glm(alldeaths ~ Minpressure, family="poisson", data=HimmData)
mod.onlyCat <- glm(alldeaths ~ factor(Category), family="poisson", data=HimmData)
mod.onlyNDAM <- glm(alldeaths ~ NDAM, family="poisson", data=HimmData)

# Compare the models with ANOVA
anova(mod.onlyminpres, mod.minpres, test="Chisq")
```

```
## Analysis of Deviance Table
##
```

```
## Model 1: alldeaths ~ Minpressure
## Model 2: alldeaths ~ GenderF + Minpressure
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1        90     2962.4
## 2        89     2882.4  1    79.96 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(mod.onlyCat, mod.Cat, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: alldeaths ~ factor(Category)
## Model 2: alldeaths ~ GenderF + factor(Category)
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1        87     3115.2
## 2        86     2977.3  1   137.79 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(mod.onlyNDAM, mod.NDAM, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: alldeaths ~ NDAM
## Model 2: alldeaths ~ GenderF + NDAM
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1        90     2836.2
## 2        89     2731.7  1   104.51 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Model Checking

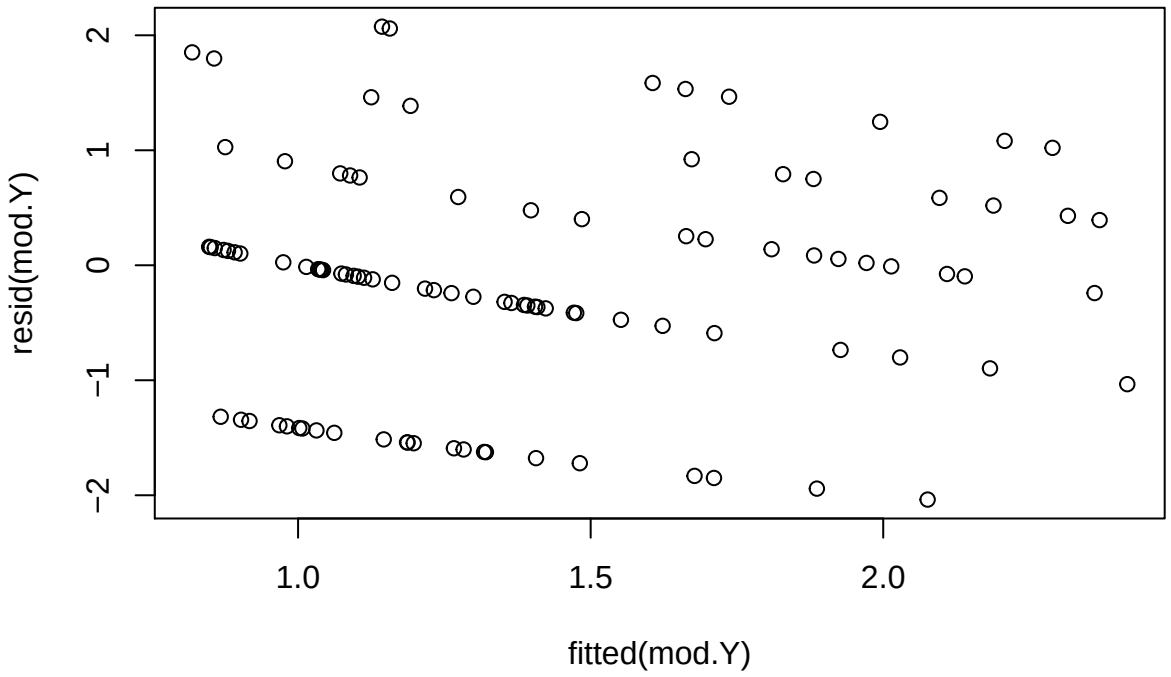
(there is a video walkthrough of this code at the end if the section)

Even if we fit a model, it might not be a good one. There are all the usual problems we've seen (outliers, non-linearity etc.). We can check them in the same way we check other residuals, although with some checks we can run into problems of non-normality. For example, we can get some weird patterns, like this

```
N <- 100
x <- runif(N)

Y <- rpois(N, exp(0+1*x))
mod.Y <- glm(Y ~ x, family = poisson())

plot(fitted(mod.Y), resid(mod.Y))
```



Plots like this can still be useful, though. But we will look at another aspect of model fit.

One property of the Poisson distribution is that the mean equals the variance. But there is no reason why the counts we observe have to do that. Indeed, we often (perhaps usually) see that the variance is larger than the mean.

We can see what happens when we have over-dispersion by simulating some data without and with over-dispersion. To do this we want to create some data that is from a Poisson distribution, i.e. a GLM is the correct model. We will then alter the model by adding some extra variation (this will also help you understand how a GLM is made).

We will use 1000 data points (because it's a nice large number), and a single covariate which we will call `X`.

```
N <- 1e3
X <- rnorm(N)
```

Now, to build up the GLM. Follow these steps:

1. Calculate the linear predictor. You will need an intercept and a slope (for the effect of X). Choose some sane values (e.g. 1.2 for each).

Panic Button, if you want a hint

The linear predictor will look like $\alpha + \beta X$

Answer

You might use different variables names and values, but that's OK.

```
alpha <- 1.2
beta <- 1.2
eta.noOD <- alpha + beta*X
```

2. Use the inverse of the link function to calculate the expected values for each (simulated) data point

Panic Button, if you want a hint

The link function is the log link, so we want the inverse of that.

Answer

```
Mean.noOD  <- exp(eta.noOD)
```

3. Simulate each data point from a Poisson distribution

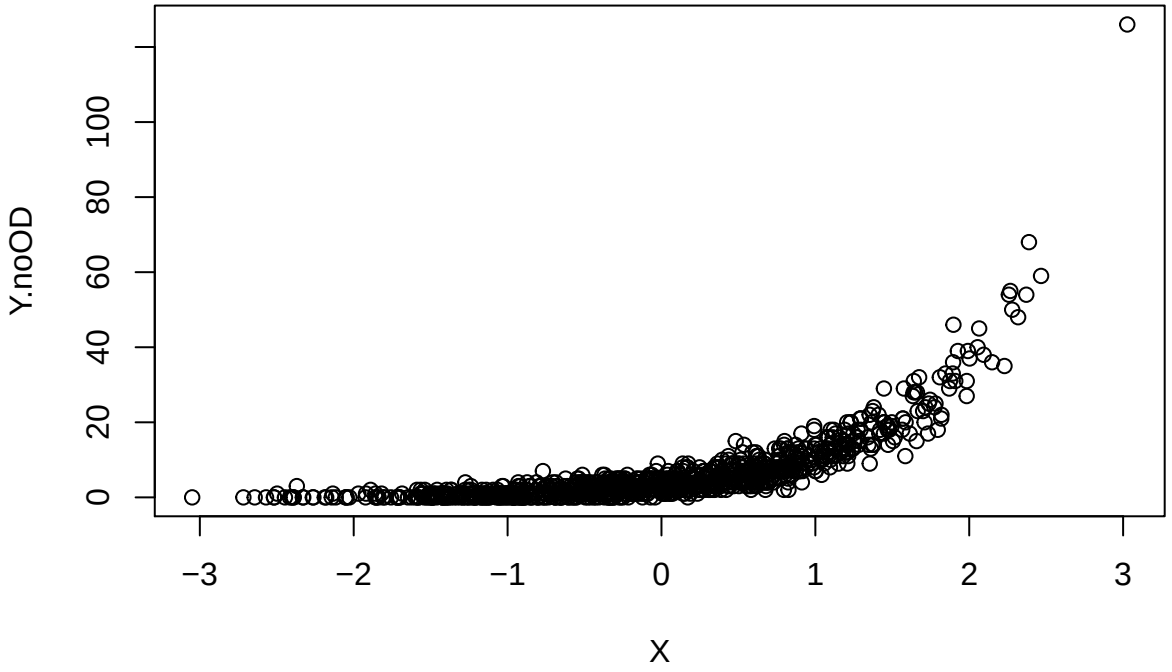
Panic Button, if you want a hint

Use the `rpois()` function

Answer (includes a bonus plot)

```
Y.noOD <- rpois(length(Mean.noOD), Mean.noOD)

plot(X, Y.noOD)
```



You should now have some nice poisson-y data.

Next you need to simulate some over-dispersed data. There are lots of ways to do this. Here we will add a second covariate that doesn't go into the model: it is probably also how a lot of overdispersion occurs in real data sets.

We need to create a new variable:

```
X.unobs <- rnorm(N)
```

1. Calculate the linear predictor with both the new and old `X`. You will need a regression coefficient (=slope) for the `X.unobs`. I would suggest a value of 0.5:

Hint

The linear predictor will look like $\alpha + \beta_1 X_1 + \beta_2 X_2$. You could also simply add the effect of the new covariate to the old linear predictor.

Answer

```
beta.unobs <- 0.5
eta.OD <- eta.noOD + beta.unobs*X.unobs
```

2. Use the inverse of the link function to calculate the expected values for each (simulated) data point

Hint

The link function is the log link, so we want the inverse of that.

Answer

```
Mean.OD <- exp(eta.OD)
```

3. Simulate each data point from a Poisson distribution

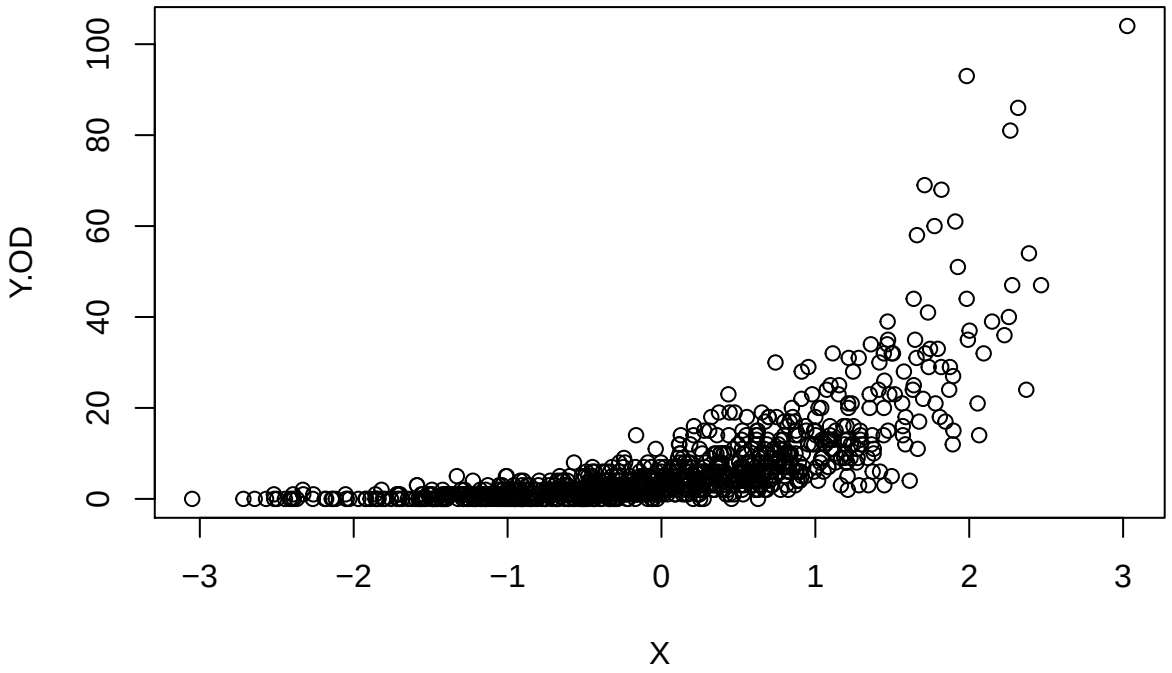
Hint

Use the `rpois()` function

Answer

```
Y.OD <- rpois(length(Mean.OD), Mean.OD)

plot(X, Y.OD)
```



So now we have some data from both a "good" model and one with over-dispersion. The first thing to note is that the variances are different: for the parameter estimates I chose the variance for the good model is 79.9, and for the over-dispersed model it is 107.9.

But what happens when we fit the model?

Fit a model to the data without over-dispersion, with `X` as a predictor. Then fit the same model with overdispersion. Look at the summaries of the two models, in particular the coefficients, their standard errors, and the residual deviance. What effects does overdispersion have (and what doesn't it have?)

Panic Button, if you want a hint

The code should look something like this:

```
mod.Y <- glm(Y ~ x, family = poisson())
summary(mod.Y)
```

You need to find the relevant terms in the summary.

If you have done this and are not sure what's real, and what is noise, run the same code a few times with different simulations of the data, and see what patterns keep on appearing.

Answer

First, the model with no overdispersion

```
Model.noOD <- glm(Y.noOD ~ X, family = poisson())
summary(Model.noOD)
```

```
##
## Call:
## glm(formula = Y.noOD ~ X, family = poisson())
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8670  -0.9363  -0.1624   0.6596   3.4291
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.21082    0.01938   62.47   <2e-16 ***
## X            1.18403    0.01449   81.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 8320.2  on 999  degrees of freedom
## Residual deviance: 1107.0  on 998  degrees of freedom
## AIC: 3932.5
##
## Number of Fisher Scoring iterations: 5
```

And with overdispersion

```
Model.OD <- glm(Y.OD ~ X, family = poisson())
summary(Model.OD)
```

```
##
## Call:
## glm(formula = Y.OD ~ X, family = poisson())
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.1262  -1.2351  -0.3825   0.5784   7.6463
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.33685    0.01819   73.48   <2e-16 ***
## X            1.15071    0.01378   83.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 10037.7  on 999  degrees of freedom
## Residual deviance:  2528.6  on 998  degrees of freedom
```

```
## AIC: 5365.9
##
## Number of Fisher Scoring iterations: 5
```

Your results might be a bit different, because random number generators are random. But what you should see are:

- the parameter estimates are the same (plus or minus some random error)
- the standard errors are the same (plus or minus some random error)
- the residual deviance is higher for the overdispersed data.

Here is a video of me waffling on whilst presenting the code.

Having overdispersion is like having more residual variation. The overall effect of more unexplained variation should be to make things less precise. But we don't see that here. Does this matter? Well, we can look at what the distribution of parameters should be by doing these simulations lots of times, and looking at the standard deviation

```
# Function to simulate the data with overdispersion, and return the coefficient for the effect of X
# betaOD=0 means no overdispersion
SimOverDisp <- function(alpha=1, betaX=1, betaOD=0, N=1e3) {
  X <- rnorm(N)
  X.unobs <- rnorm(N)

  eta <- alpha + betaX*X + betaOD*X.unobs
  Y <- rpois(length(eta), exp(eta))

  Model <- glm(Y ~ X, family = poisson())
  coef(Model)["X"]
}

# First run the simulations without overdispersion, and then with overdispersion
Repbeta0 <- replicate(1e2, SimOverDisp(betaOD=0))
Repbeta1 <- replicate(1e2, SimOverDisp(betaOD=1))
# And look at the standard deviations of the estimates
c(NoOD=sd(Repbeta0), OD=sd(Repbeta1))
```

```
##       NoOD         OD
## 0.01587353 0.09843870
```

What we see is that the variance should be larger - we should see wider standard errors.

The uncertainty in the estimates increases but this isn't seen in the confidence intervals. But the residual deviance does increase. The residual deviance summarises the amount of unexplained variation, and because the variance of the Poisson distribution is determined by the mean, the amount of unexplained variation should also be determined by the mean.

Once the maths is done (not by us!), it turns out that with no overdispersion, the deviance should (roughly) equal the residual degrees of freedom. Indeed, it should follow a chi-squared distribution, with degrees of freedom equal to the residual degrees of freedom. So we can use that to test whether there is overdispersion.

We can do this with the **deviance ratio**, which is the Deviance divided by the degrees of freedom. This should equal about 1, but obviously there will be variation. As a rule of thumb, values below about 1.2 suggst that even if there is overdispersion, it shouldn't make big difference. If the null hypothesis is true (i.e. there is no overdispersion), the deviance should follow a chi-squared distribution, so we can use that as a test.

```
(Dispersion <- deviance(Model.noOD)/df.residual(Model.noOD))
```

```
## [1] 1.109231
```

```
pchisq(q = deviance(Model.noOD), df.residual(Model.noOD), lower.tail = FALSE)
```

```
## [1] 0.008863996
```

Note that we need `lower.tail = FALSE` in `pchisq()` because we are interested in knowing if the dispersion is too big: this is a one-tailed test.

What evidence is there of over-dispersion in the Himmicanes data? Test it for one of the covariates

Answers

Minimum pressure:

```
(Dispersion.minpres <- deviance(mod.minpres)/df.residual(mod.minpres))
```

```
## [1] 32.38654
```

```
pchisq(q = deviance(mod.minpres), df.residual(mod.minpres),
       lower.tail = FALSE)
```

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

Categories:

```
(Dispersion.Cat <- deviance(mod.Cat)/df.residual(mod.Cat))
```

```
## [1] 34.62037
```

```
pchisq(q = deviance(mod.Cat), df.residual(mod.Cat),
       lower.tail = FALSE)
```

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

Normalised damage:

```
(Dispersion.NDAM <- deviance(mod.NDAM)/df.residual(mod.NDAM))
```

```
## [1] 30.69371
```

```
pchisq(q = deviance(mod.NDAM), df.residual(mod.NDAM),
       lower.tail = FALSE)
```

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

Regardless of which model we chose, the deviance ratio is huge, which suggests there is a lot of overdispersion. The p-values are so small they are effectively 0.

Dealing With Overdispersion

Let's have another video!

Overdispersion is so common that a few ways have been eveloped to deal with it:

- Correct in the likelihood
- Use a mixed model (not in this course, sorry)
- Use a different distribution

Correct the likelihood

The general form for a GLM likelihood is

$$l(\theta|y) = \frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi)$$

Thus far we have not discussed ϕ much. But here we need it. One way to correct for overdispersion is to write $a(\phi) = c\phi$, and to estimate c. We can do this with the **deviance ratio**:

```
Dispersion <- deviance(Model.OD)/df.residual(Model.OD)
```

This should equal about 1, but obviously there will be variation. As a rule of thumb, values below about 1.2 suggst that even if there is overdispersion, it shouldn't make big difference.

We use the dispersion estimate to correct the calcualtions of standard errors by plugging it into the summary:

```
summary(mod.OD, dispersion = Dispersion)
```

The effect is to increase standard errors by sqrt(Dispersion):

```
# Standard errors without over-dispersion
summary(Model.OD)$coefficients[,"Std. Error"]
# Standard errors without over-dispersion, multiplied by sqrt of dispersion
summary(Model.OD)$coefficients[,"Std. Error"]*sqrt(Dispersion)
# Standard errors with over-dispersion
summary(Model.OD, dispersion =
                Dispersion)$coefficients[,"Std. Error"]
```

We can also use the dispersion in `anova()`:

```
anova(Model.OD, dispersion = Dispersion, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: poisson, link: log
##
## Response: Y.OD
##
## Terms added sequentially (first to last)
##
##
##      Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                  999    10037.7
## X     1   7509.1       998     2528.6 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For the Himmicanes data, what happens to the results if you correct the analysis by correcting the estimates with overdispersion? What happens with the model comparison, and with the estimates of the gender effect and its standard error?

Hint

Look at both `summary()` and `anova()`

Answer

Regardless of which model we chose, the deviance ratio is over 30, which is huge.

```
(MinPresDR <- deviance(mod.minpres)/df.residual(mod.minpres))
```

```
## [1] 32.38654
```

```
(CatDR <- deviance(mod.Cat)/df.residual(mod.Cat))
```

```
## [1] 34.62037
```

```
(NDAMDR <- deviance(mod.NDAM)/df.residual(mod.NDAM))
```

```
## [1] 30.69371
```

The models are compared here:

```
anova(mod.onlyminpres, mod.minpres, test="Chisq", dispersion = MinPresDR)
```

```
## Analysis of Deviance Table
##
## Model 1: alldeaths ~ Minpressure
## Model 2: alldeaths ~ GenderF + Minpressure
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1        90     2962.4
## 2        89     2882.4  1    79.96   0.1161
```

```
anova(mod.onlyCat, mod.Cat, test="Chisq", dispersion = CatDR)
```

```
## Analysis of Deviance Table
##
## Model 1: alldeaths ~ factor(Category)
## Model 2: alldeaths ~ GenderF + factor(Category)
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1        87     3115.2
## 2        86     2977.3  1   137.79  0.04604 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(mod.onlyNDAM, mod.NDAM, test="Chisq", dispersion = NDAMDR)
```

```
## Analysis of Deviance Table
##
## Model 1: alldeaths ~ NDAM
## Model 2: alldeaths ~ GenderF + NDAM
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1        90     2836.2
## 2        89     2731.7  1   104.51    0.065 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value goes up, so it is on the edge of significance. Basically, the evidence for an effect is much weaker.

We will only look at the estimates from the minimum pressure analysis

```
Disp.minpres <- deviance(mod.minpres)/df.residual(mod.minpres)
summary(mod.minpres, dispersion = Disp.minpres)$coefficients
```

```
##                Estimate  Std. Error   z value     Pr(>|z|)
## (Intercept) 39.30294459 6.182849986  6.356768 2.060425e-10
## GenderFMale -0.47211174 0.312899987 -1.508826 1.313432e-01
## Minpressure -0.03774036 0.006506034 -5.800825 6.598962e-09
```

The estimated gender effect doesn't change, but the standard error is larger, and suggests the direction of effect is difficult to determine. Unfortunately `confint()` doesn't take a dispersion term.

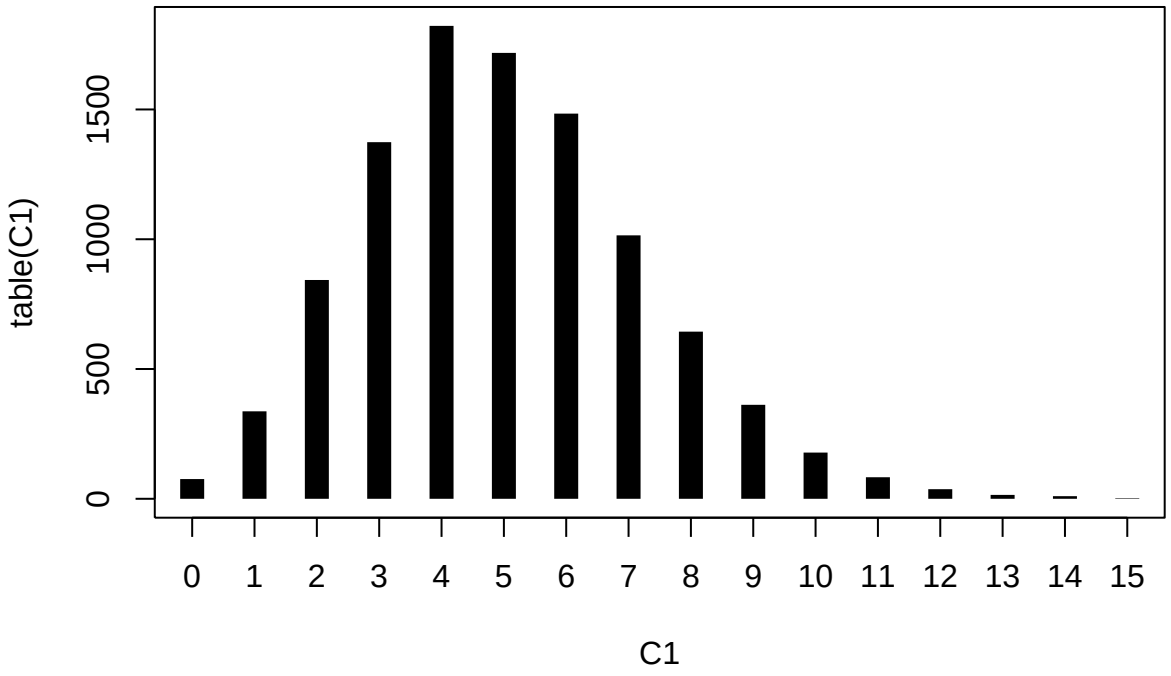
Use a different distribution

Correcting the dispersion is one approach, but one disadvantage it has it that it makes predictions more difficult. Another approach is to use a distribution that is more flexible, so that it can account for this extra

variation. One distribution that is commonly used is the negative binomial distribution. There are several ways of deriving the negative binomial distribution, but the one that is useful here is as a mixture of Poisson distirbutions.

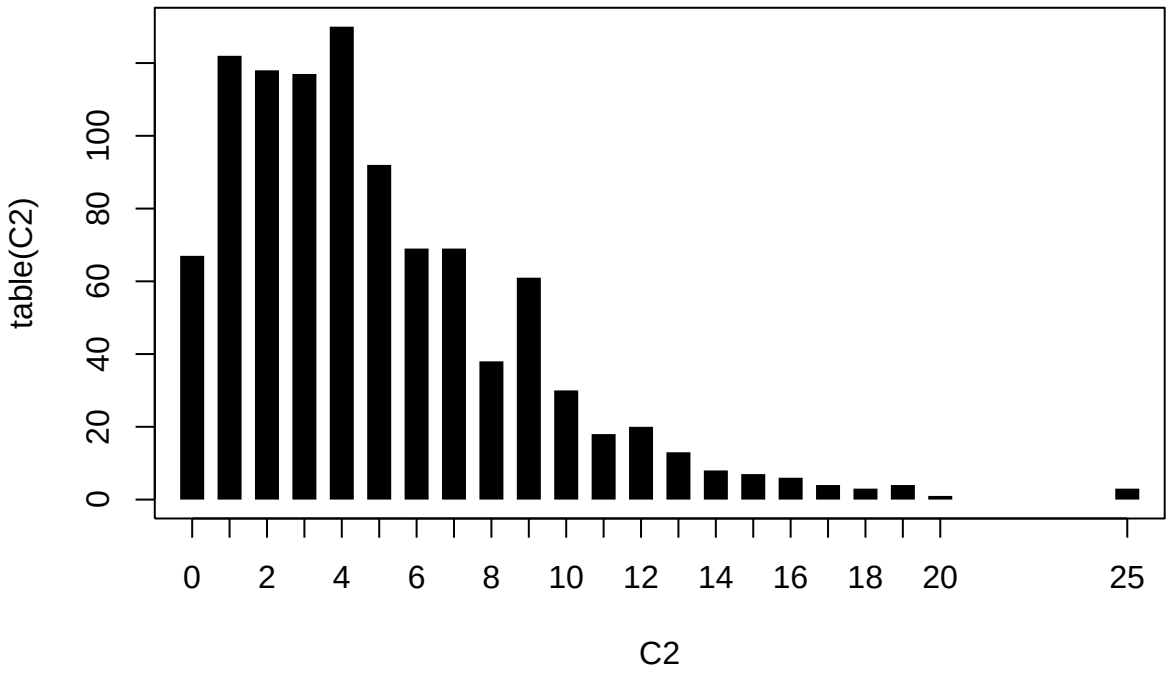
With a Poisson distribution we have a mean, λ, and this defines the distribution:

```
lambda <- 5
C1 <- rpois(1e4, lambda)
plot(table(C1), lwd=12, lend=3)
```



But what if it varies? In particular, what if the mean is allowed to follow a chi-squared distribution?

```
df <- 5
# the expected value of a chi-squared is its degrees of freedom, so we divide by df to give an expected
lambda <- 5*rchisq(1e4, df)/df
C2 <- rpois(1e3, lambda)
plot(table(C2), lwd=12, lend=3)
```



We see that there is more variation. This makes sense if you think that this is a mixture of Poisson distributions, some with a mean above 5, some below.

The amount of extra variation is determined by the degrees of freedom parameter. The parameter has to be positive.

How does the variance of the distribution change as the degrees of freedom parameter increases? Use the code above and try different values of df.

Hint

You don't need to plot the data each time: just use a different value of df (it has to be positive, though) and calculate the variance of `C2`

Answer

Because I'm lazy, I used a function to calculate the variance, and pass different degrees of freedom to it:

```
CalcVar <- function(df) {
  lambda <- 5*rchisq(1e4, df)/df
  C <- rpois(1e3, lambda)
  var(C)
}
CalcVar(1)
```

```
## [1] 55.52911
```

```
CalcVar(5)
```

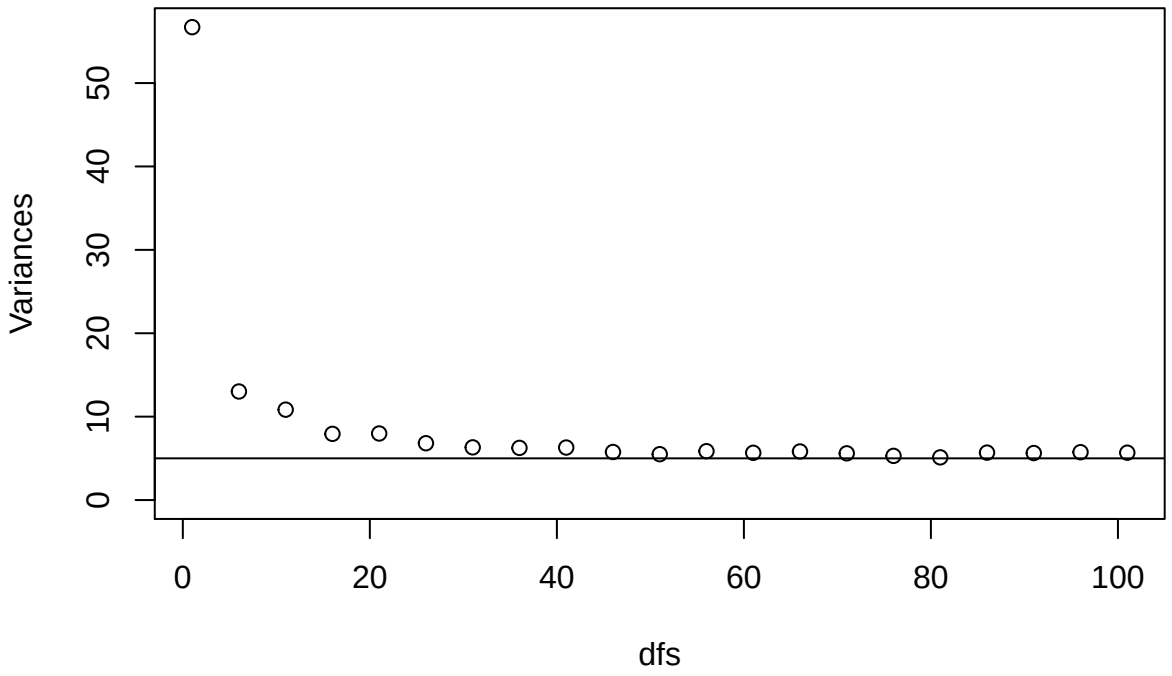
```
## [1] 15.35822
```

```
CalcVar(50)
```

```
## [1] 6.153558
```

So we see that the variance decreases. We can look in a bit more detail using `sapply()` to loop over the values. We know that the variance for this Poisson distribution is 5, so we can add that line:

```r
dfs <- seq(1, 101, by=5)
Variances <- sapply(dfs, CalcVar)
plot(dfs, Variances, ylim=c(0, Variances[1]))
abline(h=5)
```



We can see that the variance decreases, and at about 20 is close to the variance of a Poisson distribution. In fact, with an infinite degrees of freedom, the negative binomial is the same as a Poisson distribution.

So, now we know that we can use a negative binomial distribution to model over-dispersed count data, how do we do it? The internal workings are a bit more complicated: if we know the degrees of freedom parameter (which is called θ), the negative binomial distribution is a GLM. But it has to be estimated. Of couse we can simply let the computer do the work.

We fit the model in almost the same way as we do for other GLMs, but with the `glm.nb()` function in the `MASS` package. We need this because the function has to flip between estimating θ and fitting the GLM for the value of θ it has estimated. It does this until the change in the likeihood get very small.

```r
# Import the library first
library(MASS)
mod.NB <- glm.nb(Y.OD ~ X)
# Could also do this, to call the function from the library directly:
# mod.NB <- MASS::glm.nb(Y.OD ~ X)
summary(mod.NB)
```

```
##
## Call:
## glm.nb(formula = Y.OD ~ X, init.theta = 4.204813879, link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9766  -0.9204  -0.2537   0.4210   3.0093
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)  1.31943    0.02525   52.26   <2e-16 ***
## X            1.18192    0.02648   44.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(4.2048) family taken to be 1)
##
##     Null deviance: 3781.3  on 999  degrees of freedom
## Residual deviance: 1050.8  on 998  degrees of freedom
## AIC: 4611.6
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  4.205
##           Std. Err.:  0.365
##
##  2 x log-likelihood:  -4605.598
```

The summary gives the same information, but also has an estimate of θ, which is 4.2, and its standard error (0.36). If we just want θ, we can get it with `mod.NB$theta`, or `theta.ml(mod.NB)`.

What happens if you fit an negative binomial distribution to the Himmicanes data? What happens with the model comparison, and with the estimates of the gender effect and its standard error?

Hint

Fit the model with `glm.mb()`, use `summary()` to look at the parameters, and `anova()` to compare the models.

Negative Binomial Answer

First we can look at the values of θ

```
library(MASS)
NB.minpres <- glm.nb(alldeaths ~ Minpressure+GenderF, data=HimmData)
NB.Cat <- glm.nb(alldeaths ~ factor(Category)+GenderF, data=HimmData)
NB.NDAM <- glm.nb(alldeaths ~ NDAM+GenderF, data=HimmData)
NB.minpres$theta
```

```
## [1] 0.5426327
```

```
NB.Cat$theta
```

```
## [1] 0.525983
```

```
NB.NDAM$theta
```

```
## [1] 0.6835657
```

Regardless of which model we chose, the estimate of θ is small, suggesting overdispersion.

We can test whether Gender has an effect. Note that I have been cunning here: in the models above I put `GenderF` last. because the test is sequential, the line of the `anova()` tests the effect of adding gender to the model above (i.e. the model with Minpressure, Categors or NDAM). I also only show the line for gender (which is what `["GenderF",]`does: it selects that row)

```
anova(NB.minpres)["GenderF",]
```

```
## Warning in anova.negbin(NB.minpres): tests made without re-estimating 'theta'
```

```
## Analysis of Deviance Table
##
```

```
## Model: Negative Binomial(0.5426), link: log
##
## Response: alldeaths
##
## Terms added sequentially (first to last)
##
##
##         Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## GenderF  1   3.7064        89      107.2   0.0542 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(NB.Cat)["GenderF",]
```

```
## Warning in anova.negbin(NB.Cat): tests made without re-estimating 'theta'
```

```
## Analysis of Deviance Table
##
## Model: Negative Binomial(0.526), link: log
##
## Response: alldeaths
##
## Terms added sequentially (first to last)
##
##
##         Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## GenderF  1   4.5457        86     107.52    0.033 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(NB.NDAM)["GenderF",]
```

```
## Warning in anova.negbin(NB.NDAM): tests made without re-estimating 'theta'
```

```
## Analysis of Deviance Table
##
## Model: Negative Binomial(0.6836), link: log
##
## Response: alldeaths
##
## Terms added sequentially (first to last)
##
##
##         Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## GenderF  1  0.63483        89     104.79   0.4256
```

We can see that the effect becomes much more likey to have been an effect of random error.

We will only look at the estimates from the minimum pressure analysis.

```
summary(NB.minpres)$coefficients
```

```
##                Estimate Std. Error   z value     Pr(>|z|)
## (Intercept) 32.07171144 7.35771711  4.358922 1.307050e-05
## Minpressure -0.03013278 0.00762929 -3.949618 7.827593e-05
## GenderFMale -0.62408179 0.30964702 -2.015462 4.385626e-02
```

We can see that the estimated gender effect is still negative (i.e. it suggests that Male hurricanes kill fewer people), but the standard error is wider: -1.2 to 0. This suggests the estimated effects could be between

1.01 and 3.4 times higher with a female name. i.e. if could be huge (3 times!) or almost nothing (about 1% higher).

More model Checking: Residuals

Now we know how to account for excess variation, we can do some more model checking, and arrive at the punch-line of all of this. R can calculate residuals for you:

```
resid(mod.minpres)[1:5]
```

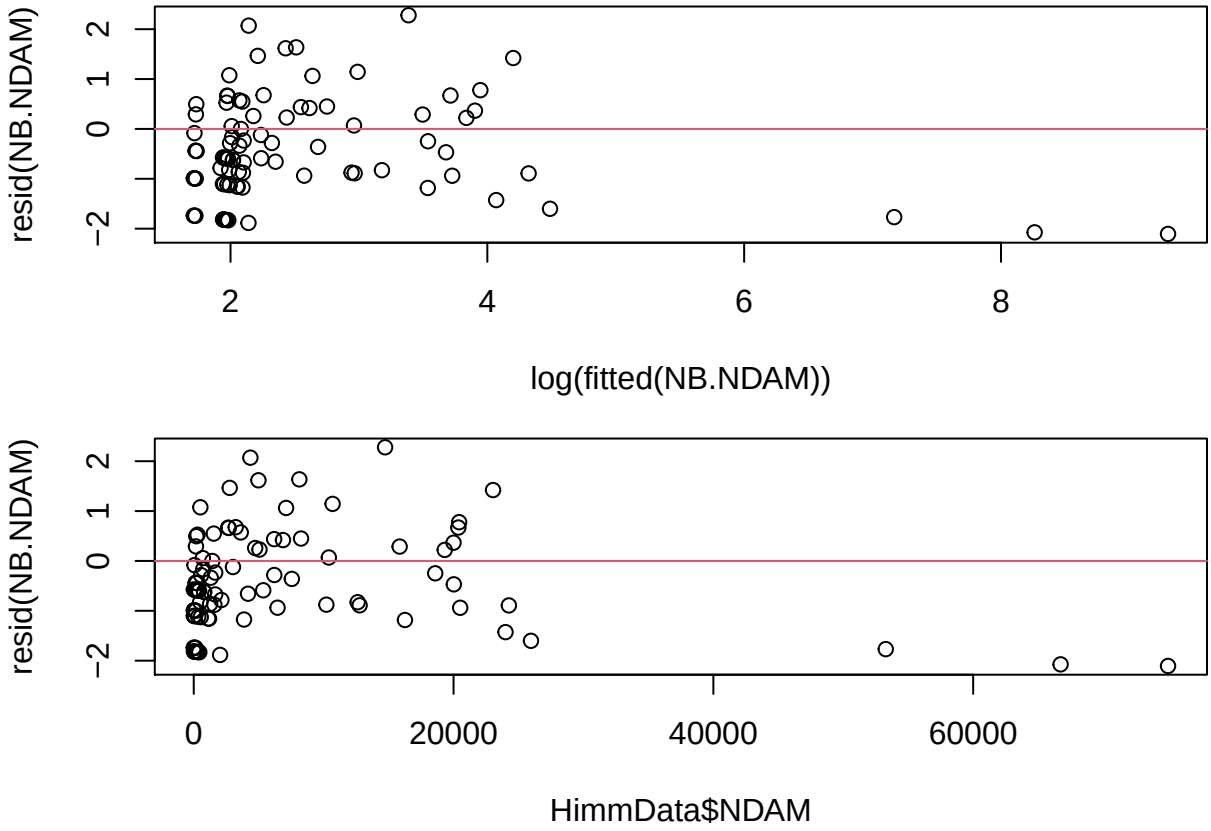
```
##         1         2         3         4         5
## -5.446324 -3.647283 -1.065315 -3.078851 -4.099720
```

```
resid(NB.minpres)[1:5]
```

```
##          1          2          3          4          5
## -1.2477639 -0.7570847 -0.4189013 -1.1621443 -1.8195728
```

We can look at them for the Himmicanes data. We'll look at the negative binomial model, with NDAM as a covariate (the pattern with the other covariates isn't as obvious: take a look!)

```
par(mfcol=c(2,1), mar=c(4.1,4.1,1,1))
plot(log(fitted(NB.NDAM)), resid(NB.NDAM))
abline(h=0, col=2)
plot(HimmData$NDAM, resid(NB.NDAM))
abline(h=0, col=2)
```



When I looked at these , I thought it looked like it might be curved.

How could we improve the model?

Help! Help!

Look back at the model checking week. The solutions are often similar, even if the data are a bit different.

Negative Binomial Answer

We could add a quadratic term into the model:

```
library(MASS)
NB.NDAM.quad <- glm.nb(alldeaths ~ NDAM + I(NDAM^2) + GenderF,
                       data=HimmData)
anova(NB.NDAM.quad)
```

```
## Warning in anova.negbin(NB.NDAM.quad): tests made without re-estimating 'theta'
```

```
## Analysis of Deviance Table
##
## Model: Negative Binomial(0.8628), link: log
##
## Response: alldeaths
##
## Terms added sequentially (first to last)
##
##
##           Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                         91     194.97
## NDAM       1   67.041        90     127.93 2.659e-16 ***
## I(NDAM^2)  1   24.030        89     103.90 9.487e-07 ***
## GenderF    1    0.411        88     103.49    0.5217
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now in the test, there is a large quadratic effect, and no evidence for an effect of gender.

```
summary(NB.NDAM.quad)
```

```
##
## Call:
## glm.nb(formula = alldeaths ~ NDAM + I(NDAM^2) + GenderF, data = HimmData,
##     init.theta = 0.8628335691, link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0310  -1.0656  -0.4226   0.2403   2.2692
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.558e+00  1.754e-01   8.884  < 2e-16 ***
## NDAM         1.859e-04  2.230e-05   8.335  < 2e-16 ***
## I(NDAM^2)   -1.988e-09  3.584e-10  -5.547 2.91e-08 ***
## GenderFMale -1.645e-01  2.531e-01  -0.650    0.516
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(0.8628) family taken to be 1)
##
##     Null deviance: 194.97  on 91  degrees of freedom
## Residual deviance: 103.49  on 88  degrees of freedom
```

```
## AIC: 649.63
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  0.863
##           Std. Err.:  0.136
##
##  2 x log-likelihood:  -639.635
```

And the direction of the Gender effect is uncertain.

Basically, there was never a Gender effect, just a curved effect of hurricane strength. this is why you should check your models: if you don't, someone else might and find out you're wrong.

Summary

You should now have some idea about the following:

- what a Poisson distribution is, and what sorts of problems it is used for
- how to fit a GLM with a Poisson distribution
- how to interpt the results of fitting a GLM with a Poisson distribution
- what over-dispersion is
- how to detect overdispersion
- how to correct for overdispersion, e.g. by fitting a model with a negative binomial distribution.