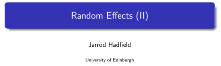
Random Effects (II)

Jarrod Hadfield

University of Edinburgh

Random Effects (II)


Random Effects (II)

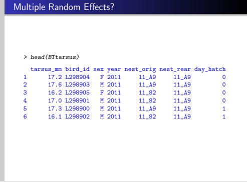
Jarrod Hadfield

University of Edinburgh

OK - on this final day we're going to dig a bit deeper into mixed models - we're going to see how they can be useful for modelling over-dispersion with non-Gaussian data, and how they can be useful for modelling a whole range of interesting biological phenomena. In a day I can only really show you the tip of an iceberg, but hopefully it will give you enough insight that you could imagine using them for a whole range of problems, even if that does involve some effort on your part.

Multiple Random Effects?

```
> head(BTtarsus)

  tarsus_mm bird_id sex year nest_orig nest_rear day_hatch
1      17.2 L298904   F 2011     11_A9     11_A9         0
2      17.6 L298903   M 2011     11_A9     11_A9         0
3      16.2 L298905   F 2011     11_82     11_A9         0
4      17.0 L298901   M 2011     11_82     11_A9         0
5      17.3 L298900   M 2011     11_A9     11_A9         1
6      16.1 L298902   M 2011     11_82     11_A9         1
```

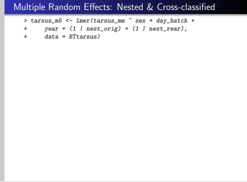

Multiple Random Effects?

```
> head(BTtarsus)

  tarsus_mm bird_id sex year nest_orig nest_rear day_hatch
1      17.2 L298904   F 2011     11_A9     11_A9         0
2      17.6 L298903   M 2011     11_A9     11_A9         0
3      16.2 L298905   F 2011     11_82     11_A9         0
4      17.0 L298901   M 2011     11_82     11_A9         0
5      17.3 L298900   M 2011     11_A9     11_A9         1
6      16.1 L298902   M 2011     11_82     11_A9         1
```

We're going to analyse the data that we had a brief look at yesterday - and just to recap, the data were collected on blue tits over 4 years (`year`: 2011-214) and we have the tarsus length (`tarsus_mm`) of around 3,000 birds. We also have the identity of the bird (`bird_id`) the sex of the bird (`sex`), the nest in which they were laid (`nest_orig`) and the nest in which they hatched and were raised (`nest_orig`). `day_hatch` indicates how many days after the first chick in the nest hatched did the chick hatch. 0 indicates a chick that was amongst the first to hatch, 1 indicates a chick that hatched the day after and 3 indicates a chick that hatched either 2 or 3 days after (we didn't check nests 2 days after hatching). In the data-frame `day_hatch` is numeric.

Multiple Random Effects: Nested & Cross-classified

```
> tarsus_m5 <- lmer(tarsus_mm ~ sex + day_hatch +
+     year + (1 | nest_orig) + (1 | nest_rear),
+     data = BTtarsus)
```

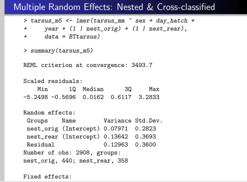

Multiple Random Effects: Nested & Cross-classified

```
> tarsus_m5 <- lmer(tarsus_mm ~ sex + day_hatch +
+     year + (1 | nest_orig) + (1 | nest_rear),
+     data = BTtarsus)
```

The model I've fitted is relatively straightforward, and you probably have an intuitive idea what it means. In terms of random effects I have a set of effects associated with the nest-of-origin and I have another set of effects associated with the nest-of-rearing. The only way I can estimate these effects (and their respective variances) is because I cross-fostered eggs between nests so that chicks from the same nest of origin were raised in multiple nests (2 usually) meaning that the two types of effect are not confounded. If I had not done any cross-fostering then the observations associated with a particular level of the nest-of-origin are all associated with a particular level of the nest-of-rearing and they are confounded. All that could be estimated is their joint effect by fitting `nest` as a random effect. Random effects, after all, are not fundamentally that different from fixed effects; if you had 60 fish, 30 were subjected to a treatment and 30 served as controls, but all the treated fish were in 1 tank and the 30 controls were in another, then tank and treatment are totally confounded. You could test whether the fish in the two groups are different but this could be due to treatment and/or tank effects. You have no way of knowing.

Multiple Random Effects: Nested & Cross-classified

```
> tarsus_m5 <- lmer(tarsus_mm ~ sex + day_hatch +
+     year + (1 | nest_orig) + (1 | nest_rear),
+     data = BTtarsus)

> summary(tarsus_m5)

REML criterion at convergence: 3493.7

Scaled residuals:
    Min      1Q  Median      3Q     Max
-5.2498 -0.5696  0.0162  0.6117  3.2833

Random effects:
 Groups    Name        Variance Std.Dev.
 nest_orig (Intercept) 0.07971  0.2823
 nest_rear (Intercept) 0.13642  0.3693
 Residual              0.12963  0.3600
Number of obs: 2908, groups:
nest_orig, 440; nest_rear, 358

Fixed effects:
```


Multiple Random Effects: Nested & Cross-classified

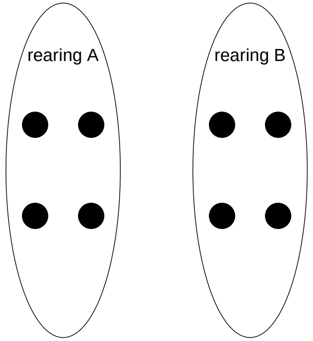
```
> tarsus_m5 <- lmer(tarsus_mm ~ sex + day_hatch +
+     year + (1 | nest_orig) + (1 | nest_rear),
+     data = BTtarsus)

> summary(tarsus_m5)

REML criterion at convergence: 3493.7

Scaled residuals:
    Min      1Q  Median      3Q     Max
-5.2498 -0.5696  0.0162  0.6117  3.2833

Random effects:
 Groups    Name        Variance Std.Dev.
 nest_orig (Intercept) 0.07971  0.2823
 nest_rear (Intercept) 0.13642  0.3693
 Residual              0.12963  0.3600
Number of obs: 2908, groups:
nest_orig, 440; nest_rear, 358

Fixed effects:
```

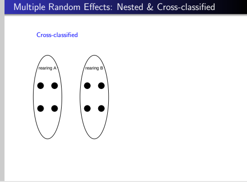
The total variance is the sum of these numbers (0.35), and so our best estimate is that nest-of-origin explains about 23% of the variance, nest-of-rearing explains about 39% of the variance, and the remaining 37% is residual variation; differences in tarsus lengths that cannot be explained by nest-level effects like parental genes and/or behaviour; for example within nest competition or measurement error.

People tend to be quite happy with the model specification, because the design of the experiment is cross-classified. Chicks from a particular nest-of-origin aren't *always* found within the same nest of rearing.

Multiple Random Effects: Nested & Cross-classified

Cross-classified

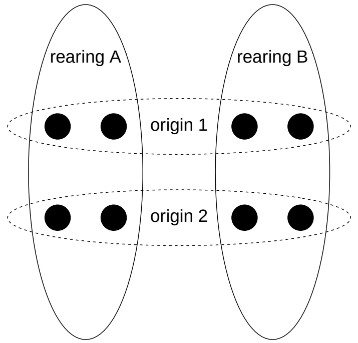


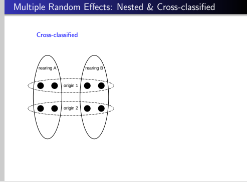


For example, lets say we have two nests of rearing and in each we've measured four chicks.

Multiple Random Effects: Nested & Cross-classified

Cross-classified

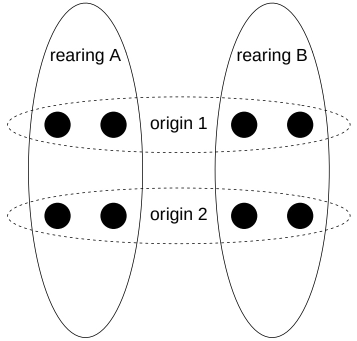




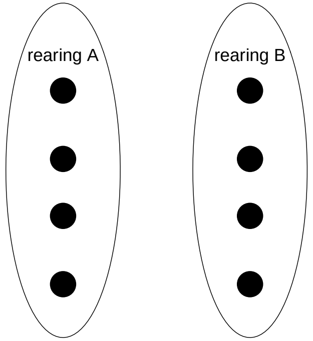
If we group those same chicks by their nests of origin, the design is such that chicks from the same nest of origin can have different nests of rearing, and likewise chicks from the same nest of rearing can have different nests of origin. The design is what's called cross-classified. Often, however, designs aren't cross-classified but nested.

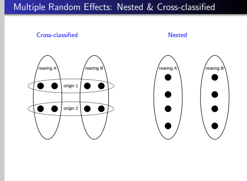
Multiple Random Effects: Nested & Cross-classified

Cross-classified



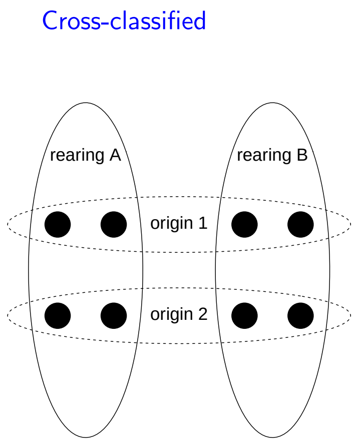
Nested

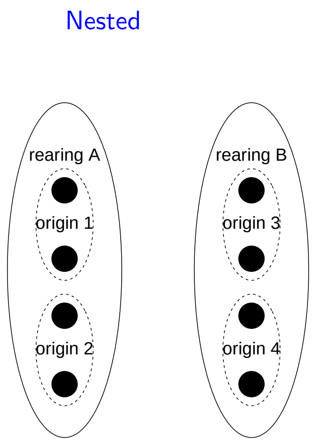


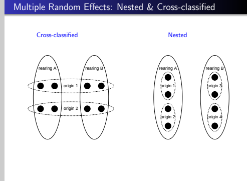


let's imagine 2 nests of rearing again, each containing 4 chicks.

Multiple Random Effects: Nested & Cross-classified



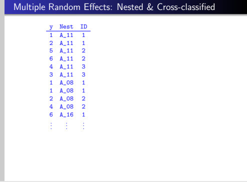




However, if we group them by their nest of origin we find that chicks from the same nest of origin are always found in the same nest of rearing; nest of origin is *nested* within nest of rearing (maybe it wasn't a good idea to use chicks in nests to explained nestedness!). Does it matter for making inferences? Well in both cases you can estimate the variation due to nest of rearing and nest of origin. Obviously so in the cross-classified design, but also in the nested design. In the nested design you could compare how similar chicks in Origin 1 are to each other compared to how similar they are to chicks in Origin 2. This would give you information on the nest-of-origin effects. You could then compare how similar Origin 1 chicks are to Origin 2 chicks, compared to how similar Origin 1 chicks are to Origin 3 and/or Origin 4 chicks, and this would give you information on the nest-of-rearing effects. In the cross-classified design you could, if you wanted, estimate any interaction effects between nest-of-origin and nest-of-rearing because chicks from the same nest-of-origin but in different nests-of-rearing can be compared. In the nested design this is not possible; chicks from Origin 1 are *always* in Rearing A and so you could not tell if they would have been different in a different nest-of-rearing (i.e. an interaction effects). In fact, any interaction effects will be estimated as nest-of-origin effects. However, the main point is if I wanted to just estimate the main effects of nest-of-origin and nest-of-rearing do I need to worry whether my design is nested or not? The short answer is no. When computers and algorithms were less good, specialised algorithms could be used on nested designs that could compute the answer much quicker than general algorithms. Because of this, software provided syntax for telling the computer the design was nested. Now, general algorithms have been developed that automatically work out the form of the design and implement the most efficient strategy for making inferences. However, there remains the worry that if your design is nested you have to tell the computer, and this worry is only partly justified.

Multiple Random Effects: Nested & Cross-classified

y	Nest	ID
1	A_11	1
2	A_11	1
5	A_11	2
6	A_11	2
4	A_11	3
3	A_11	3
1	A_08	1
1	A_08	1
2	A_08	2
4	A_08	2
6	A_16	1
⋮	⋮	⋮

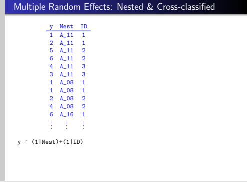

Multiple Random Effects: Nested & Cross-classified

Let's imagine a more common type of nested design where for example I have repeated measurements on the same individual (lets say I've measured each bird twice) and a bird is always found in the same nest. So there are 3 chicks in nest `A_11` , 2 chicks in nest `A_08` and so on, and I've measured each two times.

Multiple Random Effects: Nested & Cross-classified

y	Nest	ID
1	A_11	1
2	A_11	1
5	A_11	2
6	A_11	2
4	A_11	3
3	A_11	3
1	A_08	1
1	A_08	1
2	A_08	2
4	A_08	2
6	A_16	1
⋮	⋮	⋮

```
y ~ (1|Nest)+(1|ID)
```


Multiple Random Effects: Nested & Cross-classified

y	Nest	ID
1	A_11	1
2	A_11	1
5	A_11	2
6	A_11	2
4	A_11	3
3	A_11	3
1	A_08	1
1	A_08	1
2	A_08	2
4	A_08	2
6	A_16	1
⋮	⋮	⋮

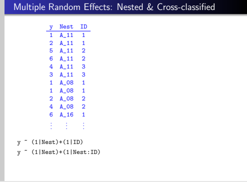
y ~ (1|Nest)+(1|ID)

Why do you think there's a problem with this model? The ID for the 1st chick in nest `A_11` is 1 and the ID for the 1st chick in nest `A_08` is also 1 1. If I fit this model the software is going to treat these four observations as if they are coming from the same chick.

Multiple Random Effects: Nested & Cross-classified

y	Nest	ID
1	A_11	1
2	A_11	1
5	A_11	2
6	A_11	2
4	A_11	3
3	A_11	3
1	A_08	1
1	A_08	1
2	A_08	2
4	A_08	2
6	A_16	1
⋮	⋮	⋮

```
y ~ (1|Nest)+(1|ID)
y ~ (1|Nest)+(1|Nest:ID)
```

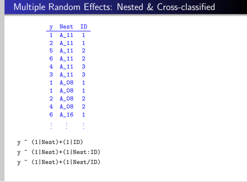


Do you think this model is OK? Yes. It's not an intuitive way of doing it, but when we interact nest with id we are fitting a different effect for each unique combination of `Nest` and `ID` and in this case, each unique combination defines a unique individual.

Multiple Random Effects: Nested & Cross-classified

y	Nest	ID
1	A_11	1
2	A_11	1
5	A_11	2
6	A_11	2
4	A_11	3
3	A_11	3
1	A_08	1
1	A_08	1
2	A_08	2
4	A_08	2
6	A_16	1
⋮	⋮	⋮

```
y ~ (1|Nest)+(1|ID)
y ~ (1|Nest)+(1|Nest:ID)
y ~ (1|Nest)+(1|Nest/ID)
```



Multiple Random Effects: Nested & Cross-classified

y	Nest	ID
1	A_11	1
2	A_11	1
5	A_11	2
6	A_11	2
4	A_11	3
3	A_11	3
1	A_08	1
1	A_08	1
2	A_08	2
4	A_08	2
6	A_16	1
⋮	⋮	⋮

```
y ~ (1|Nest)+(1|ID)
y ~ (1|Nest)+(1|Nest:ID)
y ~ (1|Nest)+(1|Nest/ID)
```

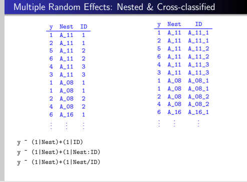
Alternatively, we could use this syntax, that stands for `ID` is nested within `Nest`. But, its fitting exactly the same model as the one above.

Multiple Random Effects: Nested & Cross-classified

y	Nest	ID
1	A_11	1
2	A_11	1
5	A_11	2
6	A_11	2
4	A_11	3
3	A_11	3
1	A_08	1
1	A_08	1
2	A_08	2
4	A_08	2
6	A_16	1
⋮	⋮	⋮

y	Nest	ID
1	A_11	A_11_1
2	A_11	A_11_1
5	A_11	A_11_2
6	A_11	A_11_2
4	A_11	A_11_3
3	A_11	A_11_3
1	A_08	A_08_1
1	A_08	A_08_1
2	A_08	A_08_2
4	A_08	A_08_2
6	A_16	A_16_1
⋮	⋮	⋮

```
y ~ (1|Nest)+(1|ID)
y ~ (1|Nest)+(1|Nest:ID)
y ~ (1|Nest)+(1|Nest/ID)
```


Multiple Random Effects: Nested & Cross-classified

y ~ (1|Nest)+(1|ID)
y ~ (1|Nest)+(1|Nest:ID)
y ~ (1|Nest)+(1|Nest/ID)

What I do, and what I would advocate, is store your data in a less ambiguous way. So give different individuals unique ID's; which I've done here by simply pasting together the `Nest` term with the original `ID` term.

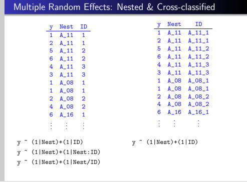
Multiple Random Effects: Nested & Cross-classified

y	Nest	ID
1	A_11	1
2	A_11	1
5	A_11	2
6	A_11	2
4	A_11	3
3	A_11	3
1	A_08	1
1	A_08	1
2	A_08	2
4	A_08	2
6	A_16	1
⋮	⋮	⋮

y	Nest	ID
1	A_11	A_11_1
2	A_11	A_11_1
5	A_11	A_11_2
6	A_11	A_11_2
4	A_11	A_11_3
3	A_11	A_11_3
1	A_08	A_08_1
1	A_08	A_08_1
2	A_08	A_08_2
4	A_08	A_08_2
6	A_16	A_16_1
⋮	⋮	⋮

```
y ~ (1|Nest)+(1|ID)
y ~ (1|Nest)+(1|Nest:ID)
y ~ (1|Nest)+(1|Nest/ID)
```

```
y ~ (1|Nest)+(1|ID)
```



Multiple Random Effects: Nested & Cross-classified

y ~ (1|Nest)+(1|ID)
y ~ (1|Nest)+(1|Nest:ID)
y ~ (1|Nest)+(1|Nest/ID)

y ~ (1|Nest)+(1|ID)

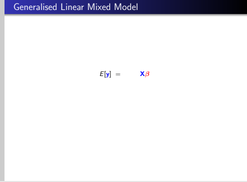
If you do it this way, then you can fit it using simpler and more intuitive sytnax. The model is *identical* to the `(1|Nest:ID)` and `(1|Nest/ID)` models fitted to the original data frame and you do not need to explicitly say your design is nested if your data are organised in a sensible way.

Generalised Linear Mixed Model

We've spent quite a bit of time on a linear mixed model (the response is conditionally normal) and now I'd like to spend a small amount of time considering random effects in a generalised linear mixed model. I don't want to spend too much time on this because I think the interpretation of random effects in the two types of model are not fundamentally different. However, there is one particular type of random effect that is very useful to fit in a non-Gaussian GLMM (I would say you should do it by default) that you should never fit in a Gaussian GLMM, and I think it will be useful to go through it.

Generalised Linear Mixed Model

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

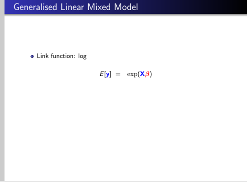


Let's have a quick recap on what a GLM is. First we specify a linear model. We have some predictor data that we can organise into a design matrix, and we can multiply that by our parameter vector. This operation produces a vector, which gives the expected value of the response for each row of the data frame, given the model. However, our expected value might be constrained in some way. If our data are counts then our expected value has to be positive, if our data are binomial, and so we are modelling the average proportion of successes, the expected value has to lie between zero and one (it's a proportion or probability).

Generalised Linear Mixed Model

- Link function: log

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


Generalised Linear Mixed Model

- Link function: log

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

If our data were counts for example, we might deal with these constraints by saying that our linear model is predicting the log of the number of counts, and so we are using a log link function. The inverse of logging is exponentiating, and so this is like saying that after exponentiating our linear we model we are predicting the number of counts.

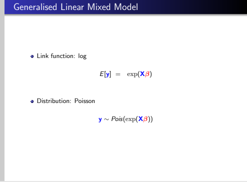
Generalised Linear Mixed Model

- Link function: log

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

- Distribution: Poisson

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



Generalised Linear Mixed Model

- Link function: log

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

- Distribution: Poisson

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

We then have to say what distribution these counts come from, and a Poisson distribution is a common choice. We don't have to specify a variance for the Poisson, because in the Poisson distribution the variance is equal to the mean; it's a single parameter distribution.

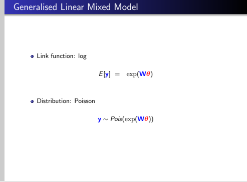
Generalised Linear Mixed Model

- Link function: log

$$E[\mathbf{y}] = \exp(\mathbf{W}\boldsymbol{\theta})$$

- Distribution: Poisson

$$\mathbf{y} \sim Pois(\exp(\mathbf{W}\boldsymbol{\theta}))$$



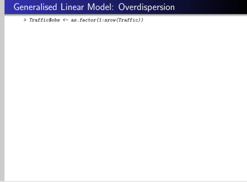
If we want to extend this idea to random effects - to move from GLM to GLMM, then all we have to do is simply add the effects into the model. $\mathbf{W}$ is now the fixed and random effect design matrices combined, and $\boldsymbol{\theta}$ is a vector of fixed and random effects. We don't have to add any new concepts.

Generalised Linear Model: Overdispersion

However, there is one surprising way you can use random effects in a GLMM, and that is to deal with overdispersion.

Generalised Linear Model: Overdispersion

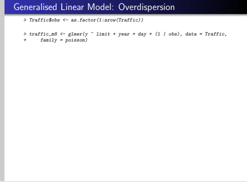
```
> Traffic$obs <- as.factor(1:nrow(Traffic))
```



Let's go back to the Swedish accident data, which we treated as Poisson on Day 3, and where we saw a lot overdispersion. If you remember, there was about 3 times more variability than what we expected from the Poisson distribution, even after accounting for the effect of year, time of year and whether a speed limit was in place or not. Now what I'm going to do is create a new factor called obs which has a unique level for each row of the data frame.

Generalised Linear Model: Overdispersion

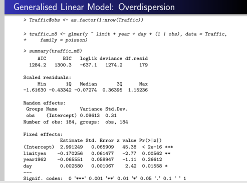
```
> Traffic$obs <- as.factor(1:nrow(Traffic))

> traffic_m8 <- glmer(y ~ limit + year + day + (1 | obs), data = Traffic,
+     family = poisson)
```



And I'm going to fit exactly the same model as before, but I'm going to fit random obs effects.

Generalised Linear Model: Overdispersion

```
> Traffic$obs <- as.factor(1:nrow(Traffic))

> traffic_m8 <- glmer(y ~ limit + year + day + (1 | obs), data = Traffic,
+     family = poisson)

> summary(traffic_m8)
     AIC      BIC   logLik deviance df.resid
  1284.2   1300.3   -637.1   1274.2      179

Scaled residuals:
     Min       1Q   Median       3Q      Max
-1.61630 -0.43342 -0.07274  0.36395  1.15236

Random effects:
 Groups Name        Variance Std.Dev.
 obs    (Intercept) 0.09613  0.31
Number of obs: 184, groups:  obs, 184

Fixed effects:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.991249   0.065909   45.38  < 2e-16 ***
limityes    -0.170256   0.061477   -2.77  0.00562 **
year1962    -0.065551   0.058947   -1.11  0.26612
day          0.002580   0.001067    2.42  0.01558 *
---
```


Generalised Linear Model: Overdispersion

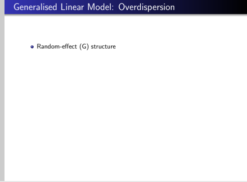
```
> Traffic$obs <- as.factor(1:nrow(Traffic))

> traffic_m8 <- glmer(y ~ limit + year + day + (1 | obs), data = Traffic,
+     family = poisson)

> summary(traffic_m8)
     AIC      BIC   logLik deviance df.resid
  1284.2   1300.3   -637.1   1274.2      179

Scaled residuals:
     Min       1Q   Median       3Q      Max
-1.61630 -0.43342 -0.07274  0.36395  1.15236

Random effects:
 Groups Name        Variance Std.Dev.
 obs    (Intercept) 0.09613  0.31
Number of obs: 184, groups:  obs, 184

Fixed effects:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.991249   0.065909   45.38  < 2e-16 ***
limityes    -0.170256   0.061477   -2.77  0.00562 **
year1962    -0.065551   0.058947   -1.11  0.26612
day          0.002580   0.001067    2.42  0.01558 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

What you can see is the coefficients are very similar to what we got using a GLM with a standard Poisson, a quasipoisson or a negative binomial distribution. There's roughly a 17% reduction in the number of accidents when there's a speed-limit in place. However, you can see that the standard errors are reasonable, and the p-values not stupidly small[1]. In fact they're virtually identical to the quasipoisson model, and particularly the negative binomial model, indicating that the overdispersion has been handled sensibly.

[1] When we fitted a linear mixed model `lmer` didn't report a p-value and we had to obtain it be hand. The reason for this is that the sampling distribution for a fixed effect is not t-distributed when we have uncertainty about variance parameters other than the normal (we don't know what it's distribution is) so the author of the package refused to do a significance test using a t-distribution as the sampling distribution. Fair enough. However, as we have seen, if the variances are well estimated and close to their true value then the sampling distribution is normal and we could use a z-test. For some reason, the author is prepared to do a significance test using this approximate sampling distribution when the response is not Gaussian.

Generalised Linear Model: Overdispersion

- Random-effect (G) structure

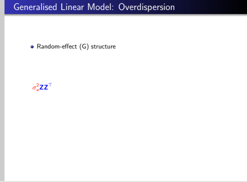


Let's first think why this works in terms of a G-structure. As we saw yesterday we can think of what our random effect specification means in terms of how the expected value of our responses may vary, and whether there's any covariance; if one observation deviates positively from its expectation are there other observations that are also likely to do so (for example because those observations were made on the same day).

Generalised Linear Model: Overdispersion

- Random-effect (G) structure

$\sigma_u^2 \mathbf{Z}\mathbf{Z}^\top$

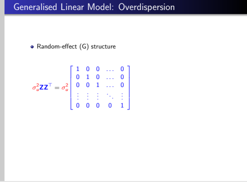


We got this by multiplying our random-effect design matrix in this weird way (its called a cross-product)

Generalised Linear Model: Overdispersion

- Random-effect (G) structure

$$\sigma^2_u \mathbf{Z}\mathbf{Z}^\top = \sigma^2_u \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$


Generalised Linear Model: Overdispersion

- Random-effect (G) structure

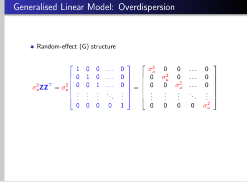
$$\sigma^2_u \mathbf{Z}\mathbf{Z}^\top = \sigma^2_u \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

which in simple models is generating a big matrix equal in dimension to the number of observations (184×184) and putting a one where ever two observations are in the same group defined by the random effect. In this case, we have an identity matrix; there are only 1's along the diagonal because each observation is associated with a unique level of obs.

Generalised Linear Model: Overdispersion

- Random-effect (G) structure

$$\sigma^2_u\mathbf{Z}\mathbf{Z}^\top = \sigma^2_u\begin{bmatrix}1 & 0 & 0 & \dots & 0\\ 0 & 1 & 0 & \dots & 0\\ 0 & 0 & 1 & \dots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}\sigma^2_u & 0 & 0 & \dots & 0\\ 0 & \sigma^2_u & 0 & \dots & 0\\ 0 & 0 & \sigma^2_u & \dots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & 0 & \sigma^2_u\end{bmatrix}$$


Generalised Linear Model: Overdispersion

- Random-effect (G) structure

$$\sigma^2_u\mathbf{Z}\mathbf{Z}^\top = \sigma^2_u\begin{bmatrix}1 & 0 & 0 & \dots & 0\\ 0 & 1 & 0 & \dots & 0\\ 0 & 0 & 1 & \dots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}\sigma^2_u & 0 & 0 & \dots & 0\\ 0 & \sigma^2_u & 0 & \dots & 0\\ 0 & 0 & \sigma^2_u & \dots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & 0 & \sigma^2_u\end{bmatrix}$$

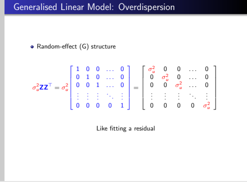
Implying that there is some noise around the expectation for each observation; there's a whole bunch of biology associated with each observation that you haven't measured and that causes it's expected value to deviate from that predicted by the fixed effects; the road was icy, it was Christmas so everyone stayed at home etc. What we can do is infer the magnitude of these effects in aggregate by estimating the amount of noise, essentially by seeing how much more variable our observed outcomes are than what we would expect from our simple Poisson model.

Generalised Linear Model: Overdispersion

- Random-effect (G) structure

$$\sigma^2_u\mathbf{Z}\mathbf{Z}^\top = \sigma^2_u\begin{bmatrix} 1 & 0 & 0 & \dots & 0\\ 0 & 1 & 0 & \dots & 0\\ 0 & 0 & 1 & \dots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \sigma^2_u & 0 & 0 & \dots & 0\\ 0 & \sigma^2_u & 0 & \dots & 0\\ 0 & 0 & \sigma^2_u & \dots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & 0 & \sigma^2_u \end{bmatrix}$$

Like fitting a residual

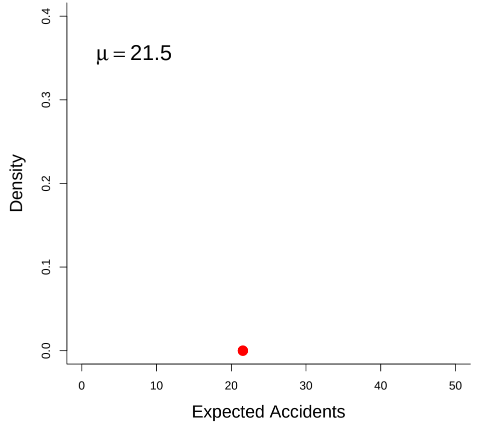

Generalised Linear Model: Overdispersion

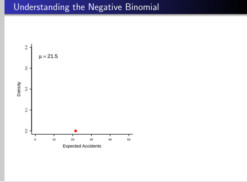
- Random-effect (G) structure

Like fitting a residual

Essentially we're fitting a residual. However, when we assume a normal distribution the residual is the deviation between what we observe (y) and what we expect based on the fixed effects ($\mathbf{X}\beta$). With the Poisson it is the deviation between what the *expected* value of our observations ($E[y]$) are and what we expect based on the fixed effects ($\mathbf{X}\beta$). There are still deviations between what we observe y and its expectation $E[y]$ due to sampling from the Poisson with mean $E[y] = \mathbf{X}\beta + \mathbf{u}$

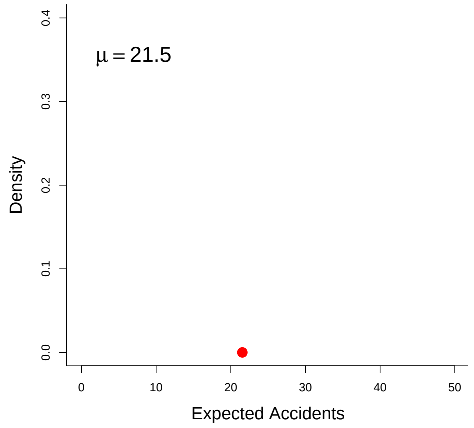
Understanding the Negative Binomial

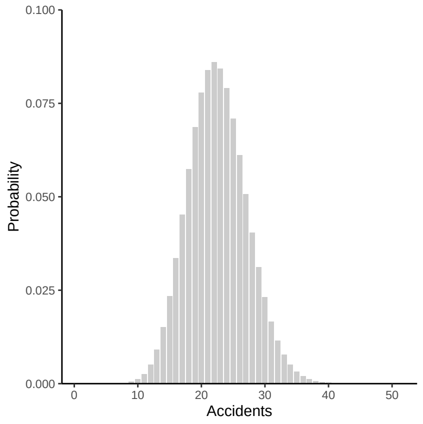


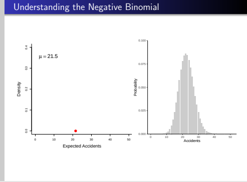


Two days ago we dealt with this overdispersion using a negative binomial model. What follows is exactly the same as what we went through then....
The average number of accidents per day in our data frame is 21.5. First, imagine a Poisson distribution with a mean parameter of 21.5. That does not mean on every day we will actually observe 21.5 accidents (how could we!),

Understanding the Negative Binomial

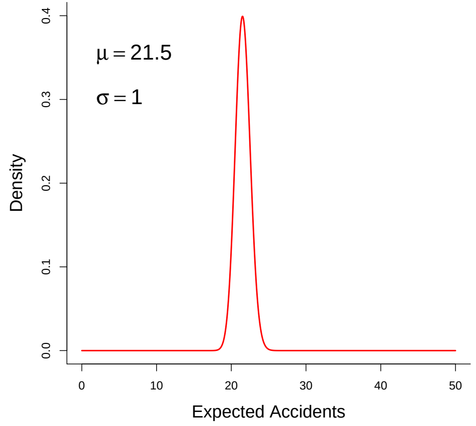


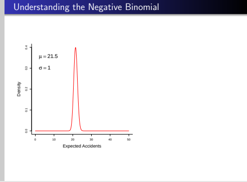




it means the number of accidents over days will be drawn from a Poisson distribution with this mean. Now what is the chance that all days have exactly the same *expected* number of accidents? If we want to think about the Poisson as a binomial with low p and high n, do we think that the number of cars 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.

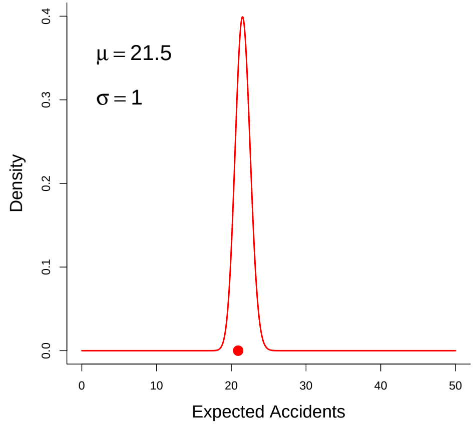
Understanding the Negative Binomial

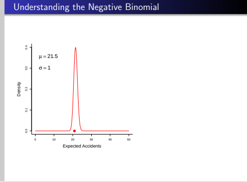




We could imagine then that the expected number of accidents on a particular date isn't just a single number but comes from a distribution of possible numbers. We could say the standard deviation of this distribution is quite small - lets say 1.

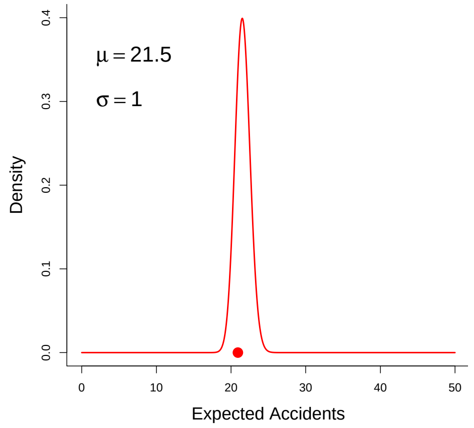
Understanding the Negative Binomial

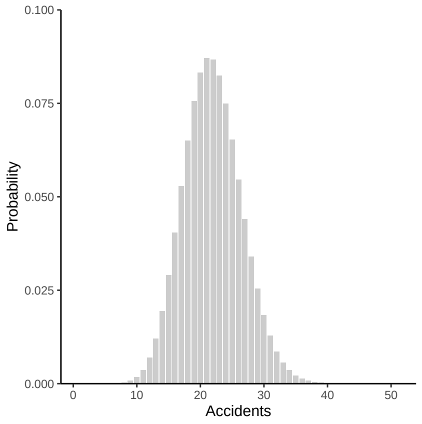


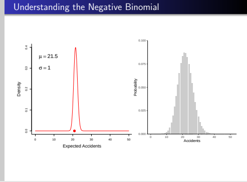


and then we draw a number from this distribution - it will be close to 21.5 but not exactly 21.5 - which gives the expected number of accidents for that date, here it is perhaps 21.6 or 21.7.

Understanding the Negative Binomial

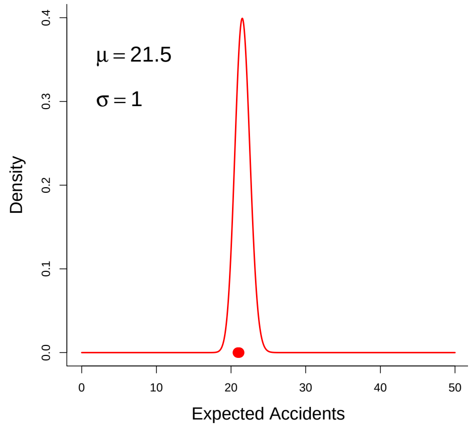


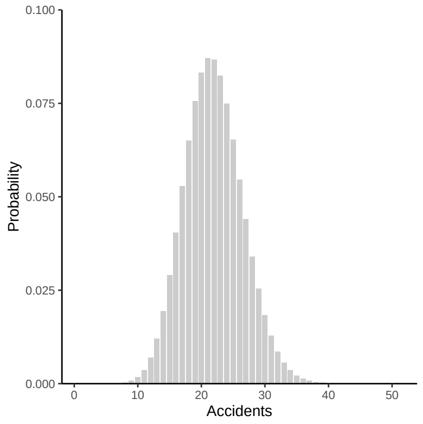


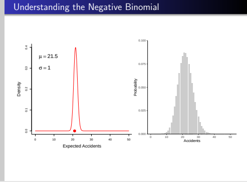


The observed number of accidents on this particular date will then be Poisson distributed with this mean of 21.7. However, that's just one possibility

Understanding the Negative Binomial

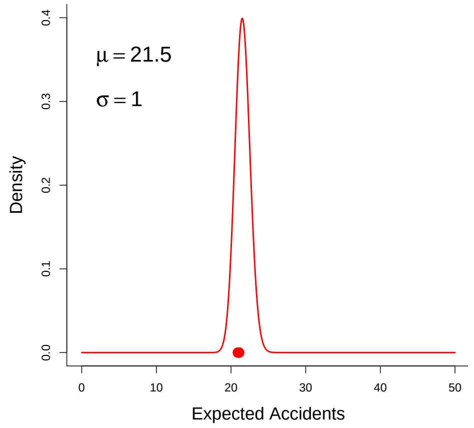


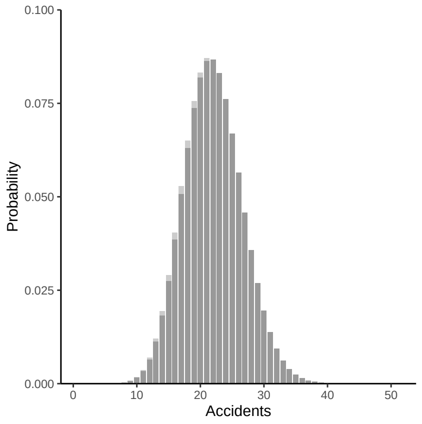


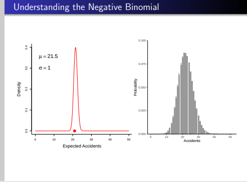


We could have drawn this expectation for example,

Understanding the Negative Binomial

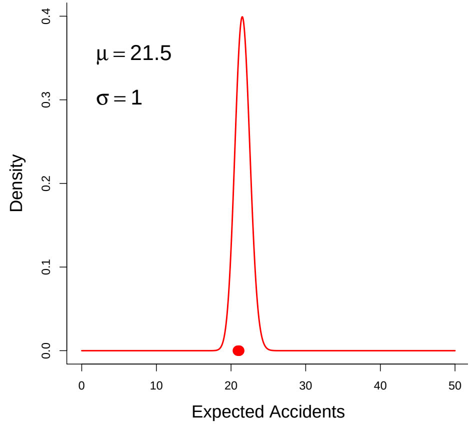


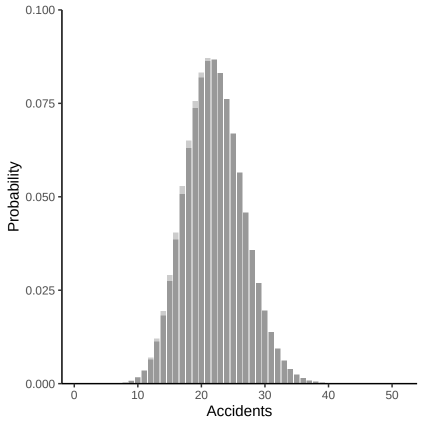


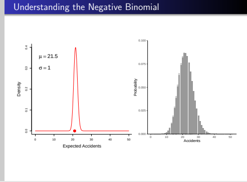


in which case the Poisson would look like this darker distribution with a slightly higher mean (and slightly higher variance)

Understanding the Negative Binomial

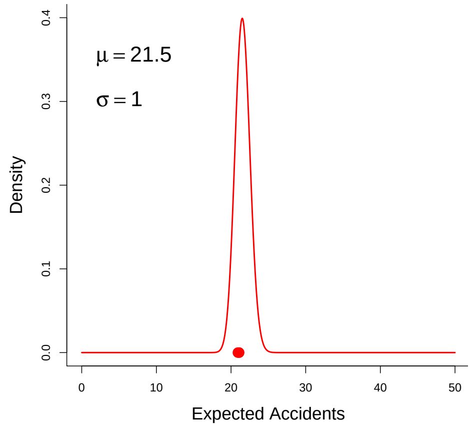


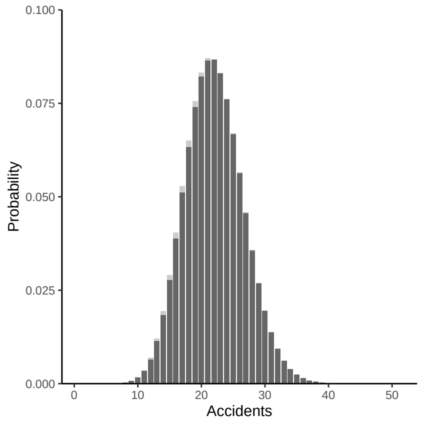


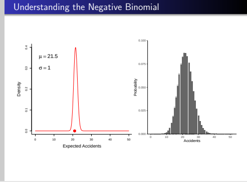


or it could have been this expectation,

Understanding the Negative Binomial

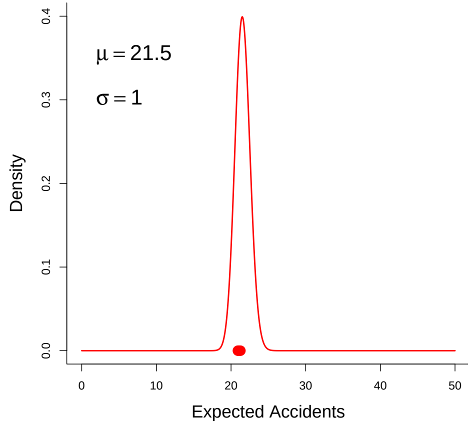


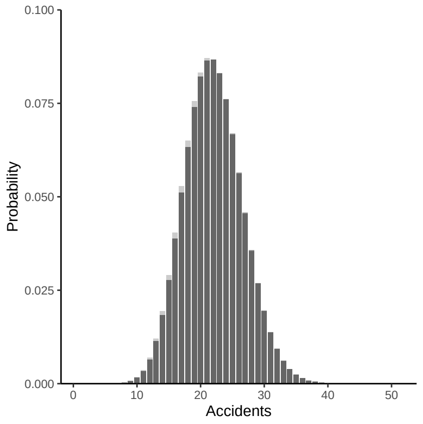


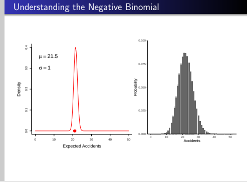


generating another Poisson.

Understanding the Negative Binomial

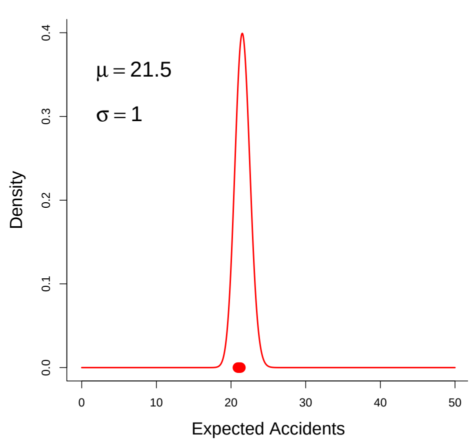


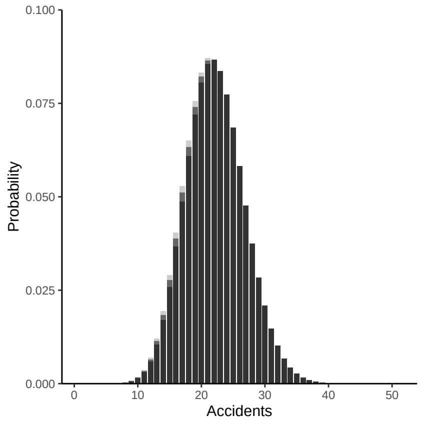


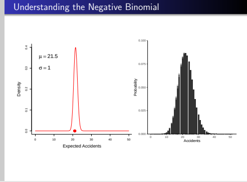


or this expectation,

Understanding the Negative Binomial

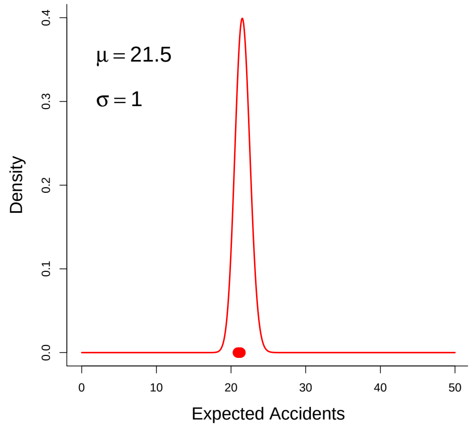


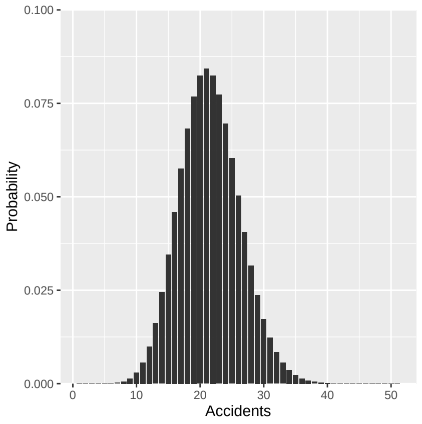


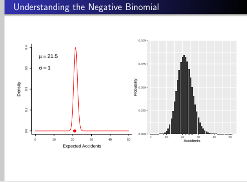


generating another Poisson..... If the distribution of expectations (the red distribution) is a gamma distribution,

Understanding the Negative Binomial

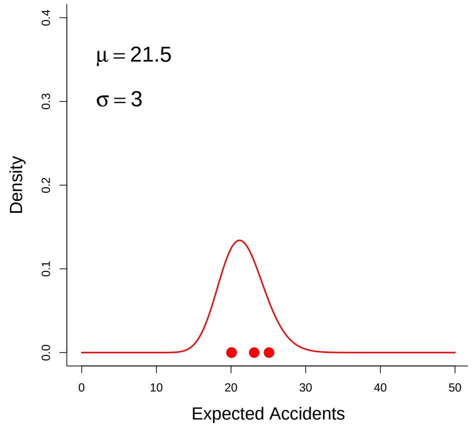


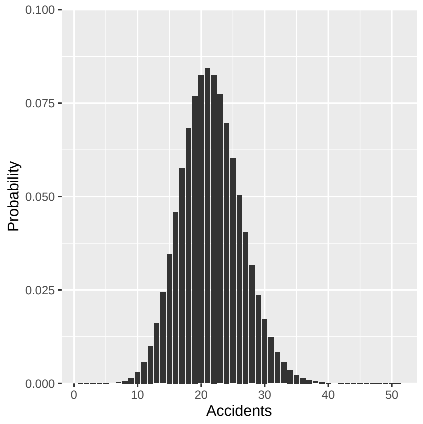


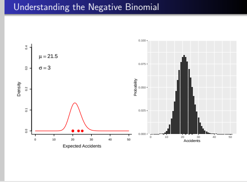


and we imagine combining all these Poisson distributions into a single distribution, then we end up with a negative binomial distribution with a parameter estimating the standard deviation of the underlying gamma distribution; a negative binomial distribution is a mixture of Poisson distributions, whose 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

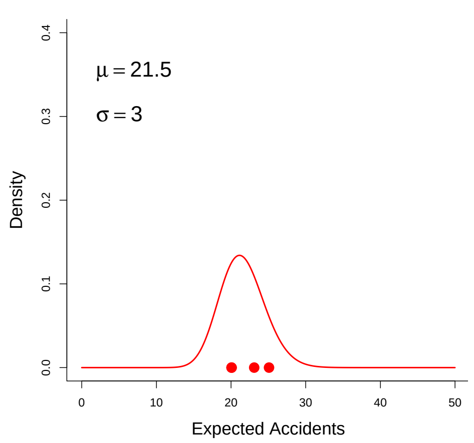


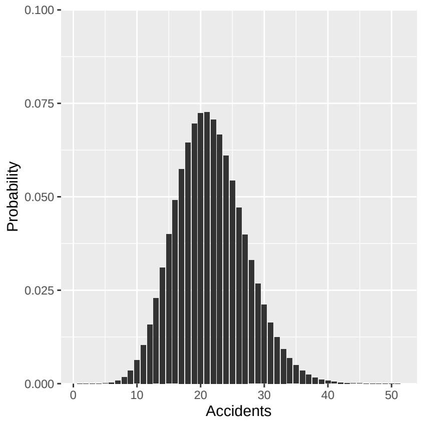


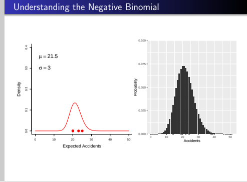


If we up the standard deviation of the gamma to 3, then the distribution of the expected number of accidents will be wider

Understanding the Negative Binomial

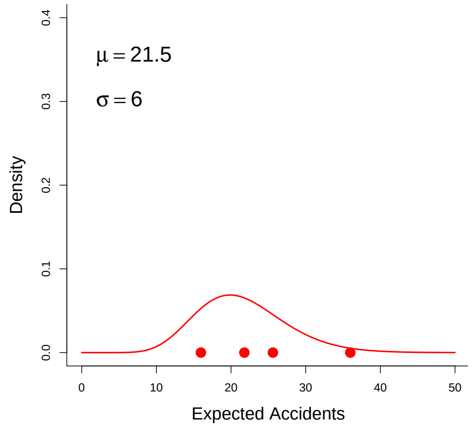


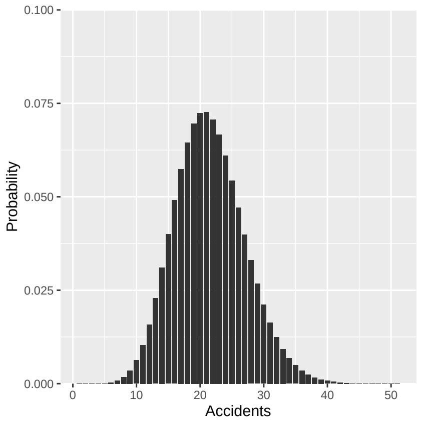


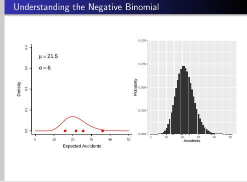


resulting in a mixture of Poisson's with greater variability

Understanding the Negative Binomial

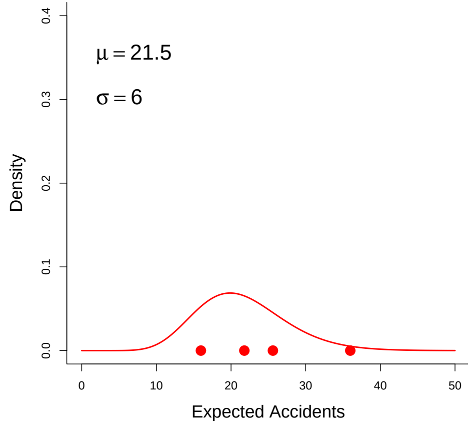


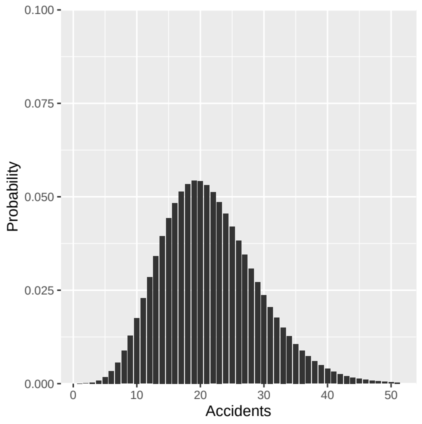


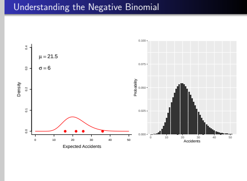


and as we up the standard deviation more and more

Understanding the Negative Binomial



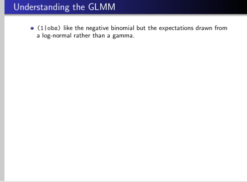




the mixture of Poisson's becomes ever more variable with a longer right hand tail. So by estimating the standard deviation of the underlying gamma distribution (σ) we allow the model to capture any excess variation.

Understanding the GLMM

- `(1|obs)` like the negative binomial but the expectations drawn from a log-normal rather than a gamma.



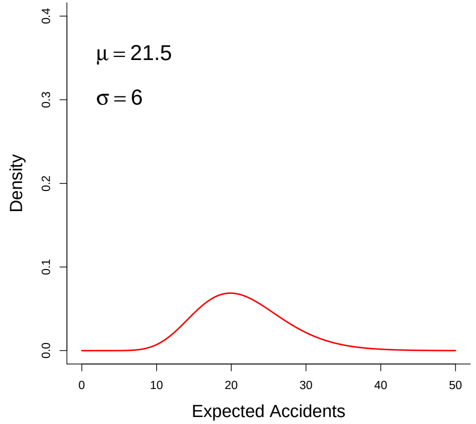
Understanding the GLMM

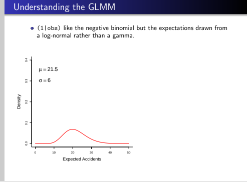
- (1|obs) like the negative binomial but the expectations drawn from a log-normal rather than a gamma.

When we fit an observation-level random effect, we are essentially doing something conceptually similar. However, rather than assuming that the distribution of expected values in the population is gamma distributed we are assuming its log normally distributed (because we assume the random effects are normally distributed on the log-scale (because we used a log link)).

Understanding the GLMM

- `(1|obs)` like the negative binomial but the expectations drawn from a log-normal rather than a gamma.

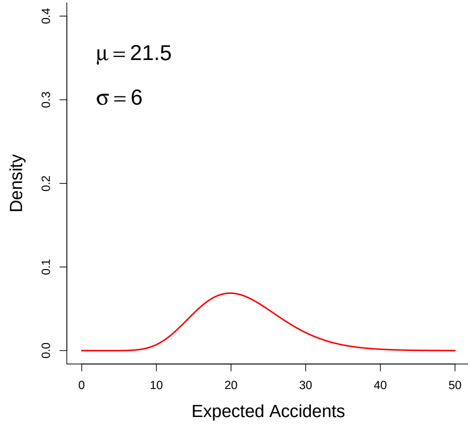


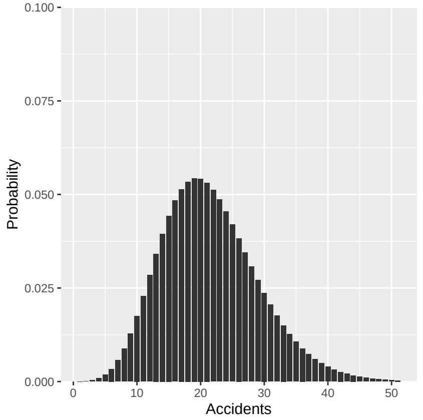


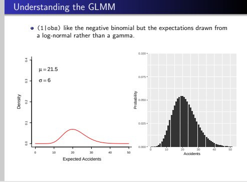
This is a gamma distribution with a mean of 21.5 and a standard deviation of 6.

Understanding the GLMM

- (1|obs) like the negative binomial but the expectations drawn from a log-normal rather than a gamma.



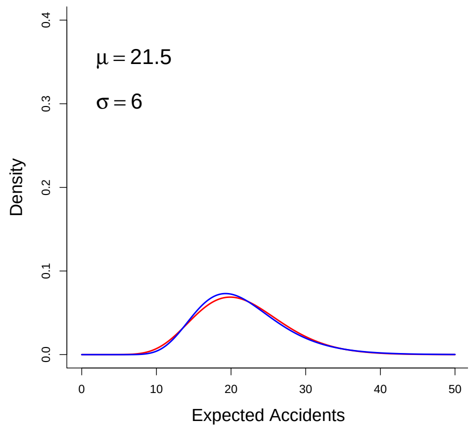


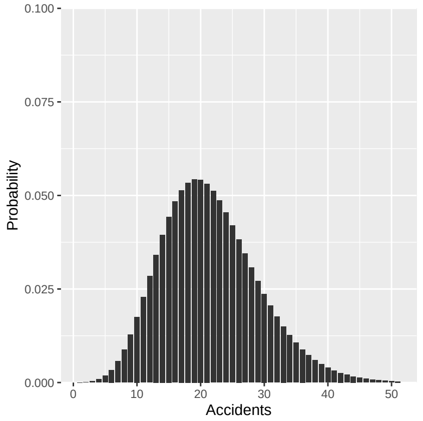


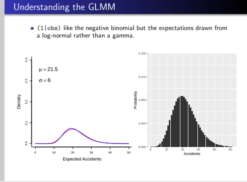
When we blend Poisson distributions whose means vary according to this gamma distribution, we end up with this distribution; the negative binomial.

Understanding the GLMM

- (1|obs) like the negative binomial but the expectations drawn from a log-normal rather than a gamma.



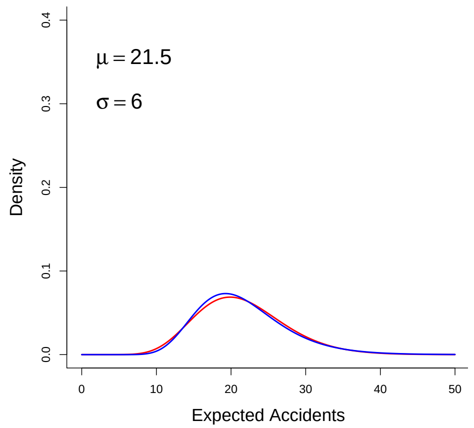


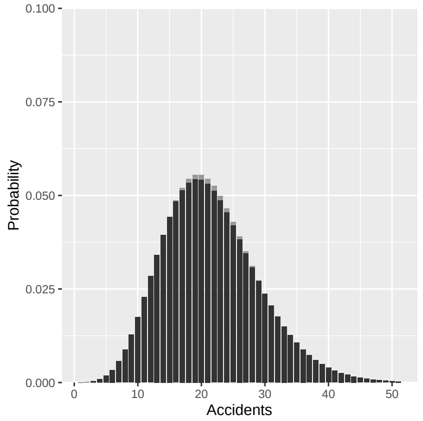


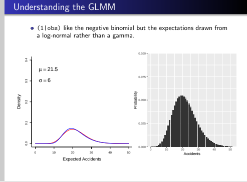
This blue line is a log-normal distribution with the same mean and variance. You can see it is almost identical to the gamma distribution in this instance.

Understanding the GLMM

- (1|obs) like the negative binomial but the expectations drawn from a log-normal rather than a gamma.



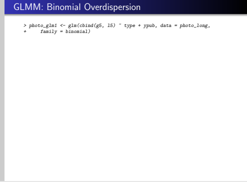




If we blend Poisson distributions whose means vary according to this log-normal distribution, we end up with another distribution which is almost identical to the the negative binomial. You can just see the lighter grey bars poking out, but its very similar and you would probably need millions of data points to tell which of the two distributions your data were really from. Practically then, it makes little difference whether you use the negative binomial or this random effect model (sometimes referred to as the Poisson log-normal).

GLMM: Binomial Overdispersion

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

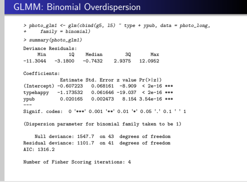

GLMM: Binomial Overdispersion

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

However, the nice thing about this approach is that you can also use it with other distributions. This was a model we fitted earlier to our 44 photos, but rather than analysing the mean score for each photo, we treated it as binomial response with the number of respondents giving the photo a score above 5 as the number of successes.

GLMM: Binomial Overdispersion

```
> photo_glm1 <- glm(cbind(g5, 15) ~ 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) -0.607223   0.068161  -8.909  < 2e-16 ***
typehappy   -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
```


GLMM: Binomial Overdispersion

```
> photo_glm1 <- glm(cbind(g5, 15) ~ 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) -0.607223   0.068161  -8.909  < 2e-16 ***
typehappy   -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 the model as a simple glm, the standard errors are tiny and the p-values stupidly small. We can see that the residual deviance is far greater than the residual degrees of freedom suggesting we have a large amount of overdispersion. Earlier we dealt with this using the quasibinomial fudge factor, but this was a bit unsatisfying because we were no longer thinking about an underlying model generating the data, and as a consequence we couldn't obtain things such as the likelihood.

GLMM: Binomial Overdispersion

```
> photo_long$obs <- as.factor(1:nrow(photo_long))
```

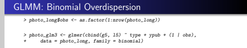
GLMM: Binomial Overdispersion

> photo_long$obs <- as.factor(1:nrow(photo_long))

However, we could create a column in our data frame that gives each row a unique level as before

GLMM: Binomial Overdispersion

```
> photo_long$obs <- as.factor(1:nrow(photo_long))

> photo_glm3 <- glmer(cbind(g5, 15) ~ type + ypub + (1 | obs),
+     data = photo_long, family = binomial)
```



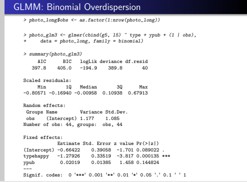
GLMM: Binomial Overdispersion

```
> photo_long$obs <- as.factor(1:nrow(photo_long))

> photo_glm3 <- glmer(cbind(g5, 15) ~ type + ypub + (1 | obs),
+     data = photo_long, family = binomial)
```

and fit these observation-level effects as random.

GLMM: Binomial Overdispersion

```
> photo_long$obs <- as.factor(1:nrow(photo_long))

> photo_glm3 <- glmer(cbind(g5, 15) ~ type + ypub + (1 | obs),
+     data = photo_long, family = binomial)

> summary(photo_glm3)
     AIC      BIC   logLik deviance df.resid
   397.8    405.0   -194.9    389.8       40

Scaled residuals:
     Min       1Q   Median       3Q      Max
-0.80571 -0.16940 -0.00958  0.10938  0.67913

Random effects:
 Groups Name        Variance Std.Dev.
 obs    (Intercept) 1.177    1.085
Number of obs: 44, groups:  obs, 44

Fixed effects:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.66422    0.39058  -1.701 0.089022 .
typehappy   -1.27926    0.33519  -3.817 0.000135 ***
ypub         0.02019    0.01385   1.458 0.144824
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```


GLMM: Binomial Overdispersion

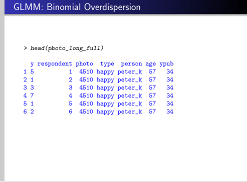
```
> photo_long$obs <- as.factor(1:nrow(photo_long))

> photo_glm3 <- glmer(cbind(g5, 15) ~ type + ypub + (1 | obs),
+     data = photo_long, family = binomial)

> summary(photo_glm3)
     AIC      BIC   logLik deviance df.resid
   397.8    405.0   -194.9    389.8       40

Scaled residuals:
     Min       1Q   Median       3Q      Max
-0.80571 -0.16940 -0.00958  0.10938  0.67913

Random effects:
 Groups Name        Variance Std.Dev.
 obs    (Intercept) 1.177    1.085
Number of obs: 44, groups:  obs, 44

Fixed effects:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.66422    0.39058  -1.701 0.089022 .
typehappy   -1.27926    0.33519  -3.817 0.000135 ***
ypub         0.02019    0.01385   1.458 0.144824
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

and what you find is that there's quite a bit of between-observation variance in the expected probability, and by accounting for it the standard errors have gone up and the p-values look a little more sensible.

The source of overdispersion is easier to understand in this example than it is in the road accidents data I think. Each binomial data point consists of 122 success/failures on a single photo. The probability of success (a score greater than 5) is likely to vary from photo to photo after accounting for type or time in academia because of the person been photographed, or because of the conditions under which that particular photo of that particular person were taken.

GLMM: Binomial Overdispersion

```
> head(photo_long_full)

  y respondent photo  type  person age ypub
1 5          1  4510 happy peter_k  57   34
2 1          2  4510 happy peter_k  57   34
3 3          3  4510 happy peter_k  57   34
4 7          4  4510 happy peter_k  57   34
5 1          5  4510 happy peter_k  57   34
6 2          6  4510 happy peter_k  57   34
```

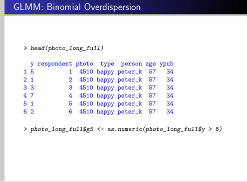

GLMM: Binomial Overdispersion

```
> head(photo_long_full)

  y respondent photo  type  person age ypub
1 5          1  4510 happy peter_k  57   34
2 1          2  4510 happy peter_k  57   34
3 3          3  4510 happy peter_k  57   34
4 7          4  4510 happy peter_k  57   34
5 1          5  4510 happy peter_k  57   34
6 2          6  4510 happy peter_k  57   34
```

We can then think of this 'binomial log-normal' model in a different way. The data we have just analysed only had one row per photograph and we had aggregated the data over respondents. This is the full data set you analysed yesterday, where we have the score (y) each respondent gave each photo; rather than 44 rows this data frame has over 5000 because each photo was scored by around 122 respondents.

GLMM: Binomial Overdispersion

```
> head(photo_long_full)

  y respondent photo  type  person age ypub
1 5          1  4510 happy peter_k  57   34
2 1          2  4510 happy peter_k  57   34
3 3          3  4510 happy peter_k  57   34
4 7          4  4510 happy peter_k  57   34
5 1          5  4510 happy peter_k  57   34
6 2          6  4510 happy peter_k  57   34

> photo_long_full$g5 <- as.numeric(photo_long_full$y > 5)
```

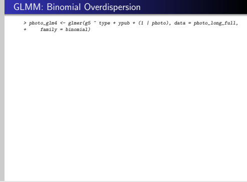

GLMM: Binomial Overdispersion

```
> head(photo_long_full)

  y respondent photo  type  person age ypub
1 5          1  4510 happy peter_k  57   34
2 1          2  4510 happy peter_k  57   34
3 3          3  4510 happy peter_k  57   34
4 7          4  4510 happy peter_k  57   34
5 1          5  4510 happy peter_k  57   34
6 2          6  4510 happy peter_k  57   34

> photo_long_full$g5 <- as.numeric(photo_long_full$y > 5)
```

What we could then do is take each score and turn it into a 1 if the score is over 5 and a zero if it is 5 or under. So rather than having binomial data for each photo we've disaggregated the data into a series of 122 0's or 1's for each photo.

GLMM: Binomial Overdispersion

```
> photo_glm4 <- glmer(g5 ~ type + ypub + (1 | photo), data = photo_long_full,
+      family = binomial)
```

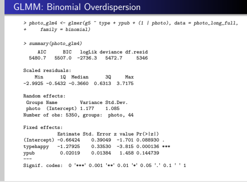

GLMM: Binomial Overdispersion

```
> photo_glm4 <- glmer(g5 ~ type + ypub + (1 | photo), data = photo_long_full,
+      family = binomial)
```

What we can then do is fit a similar model to the one we had before to these Bernoulli data (note you don't need a two column response if you have binomial data with a single trial; passing a vector of zero's and one's is fine) and fit photo effects as random.

GLMM: Binomial Overdispersion

```
> photo_glm4 <- glmer(g5 ~ type + ypub + (1 | photo), data = photo_long_full,
+     family = binomial)

> summary(photo_glm4)

     AIC      BIC   logLik deviance df.resid
  5480.7   5507.0  -2736.3   5472.7     5346

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.9925 -0.5432 -0.3660  0.6313  3.7175

Random effects:
 Groups Name        Variance Std.Dev.
 photo  (Intercept) 1.177    1.085
Number of obs: 5350, groups:  photo, 44

Fixed effects:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.66424    0.39049  -1.701 0.088930 .
typehappy   -1.27925    0.33530  -3.815 0.000136 ***
ypub         0.02019    0.01384   1.458 0.144739
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

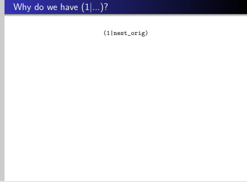

GLMM: Binomial Overdispersion

```
> photo_glm4 <- glmer(g5 ~ type + ypub + (1 | photo), data = photo_long_full,
+     family = binomial)

> summary(photo_glm4)

     AIC      BIC   logLik deviance df.resid
  5480.7   5507.0  -2736.3   5472.7     5346

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.9925 -0.5432 -0.3660  0.6313  3.7175

Random effects:
 Groups Name        Variance Std.Dev.
 photo  (Intercept) 1.177    1.085
Number of obs: 5350, groups:  photo, 44

Fixed effects:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.66424    0.39049  -1.701 0.088930 .
typehappy   -1.27925    0.33530  -3.815 0.000136 ***
ypub         0.02019    0.01384   1.458 0.144739
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

And what we get is *exactly* the same answer as before with an observation-level model on the aggregated (Binomial) data. Not a similar answer, but the exact same answer, because the model is identical. In the aggregate data an observation was at the level of photo, and so it makes sense that these two models coincide. Of course, you might want to put other random effects in both models (for example `person`) and the disaggregated data allows you to explore things like respondent effects which you saw yesterday.
With Bernoulli data, and only Bernoulli data, you don't have to deal with overdispersion. This is not because our model completely captures all the variation in the underlying probability of success or failure amongst observations, it is because any additional variation in the probabilities cannot be estimated from the data. Imagine we are in a room of 100 people and we are told that 20% of the people will be dead the following day. If the people in the room were a random sample from the UK population I would worry - the probability of death is probably pretty constant across people so it probably means *I* have a 20% chance of dying. If on the other hand the room was a hospital ward and I was a visitor, I may not worry too much for *my* safety. There will be a lot of variation in the underlying probability (between visitors and patients for example) and I'm probably at the low probability end. The point is that in the absence of this information (visitor versus patient), the binary data look the same if each person has a 20% chance of dying or if 20 people have a 100% chance of dying and 80 people no chance. In both cases we expect 20 people to be dead and 80 not.

Why do we have (1|...)?

```
(1|nest_orig)
```


Why do we have (1|...)?

(1|nest_orig)

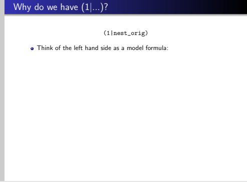
When you fit effects as random in `lmer` you use this syntax; 1 then a ‘pipe‘ then the name of the term, and the whole thing wrapped in brackets. Why? Because that’s just how you do it.

Understanding why you do this is important, because by understanding what it means you open up a whole new range of useful models that can be used to answer biological questions that are hard to answer in any other way.

Why do we have (1|...)?

`(1|nest_orig)`

- Think of the left hand side as a model formula:



Why do we have (1|...)?

(1|nest_orig)

- Think of the left hand side as a model formula:

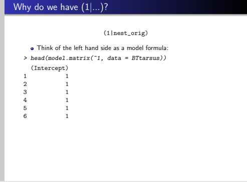
You should interpret the left hand side as a model formula.

Why do we have (1|...)?

(1|nest_orig)

- Think of the left hand side as a model formula:

```
> head(model.matrix(~1, data = BTtarsus))
  (Intercept)
1           1
2           1
3           1
4           1
5           1
6           1
```



So for example the 1 is fitting an intercept and if we look at the design matrix it of course only has one column (the model has a single parameter - the intercept).

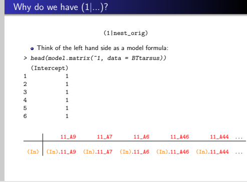
Why do we have (1|...)?

`(1|nest_orig)`

- Think of the left hand side as a model formula:

```
> head(model.matrix(~1, data = BTtarsus))
  (Intercept)
1           1
2           1
3           1
4           1
5           1
6           1
```

	11_A9	11_A7	11_A6	11_A46	11_A44	...
(In)	(In).11_A9	(In).11_A7	(In).11_A6	(In).11_A46	(In).11_A44	...


Why do we have (1|...)?

(1|nest_orig)

- Think of the left hand side as a model formula:

```
> head(model.matrix(~1, data = BTtarsus))
  (Intercept)
1           1
2           1
3           1
4           1
5           1
6           1
```

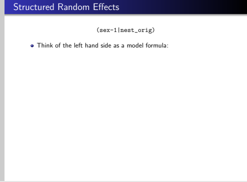
	11_A9	11_A7	11_A6	11_A46	11_A44	...
(In)	(In).11_A9	(In).11_A7	(In).11_A6	(In).11_A46	(In).11_A44	...

What you can then imagine doing is pasting the parameter name (Here I've used `(In)` instead of `(Intercept)`) to each parameter name defined by the model to the right hand side of the pipe (here an effect associated with each level of `nest_orig`) to create a new set of effects to be fitted. In this case you can see there is a one-to-one mapping between our new effects `(In).11.A9` and the `nest_orig`) effects so it is exactly equivalent to fitting `nest_orig` effects.

Structured Random Effects

`(sex-1|nest_orig)`

- Think of the left hand side as a model formula:



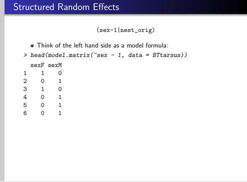
Now let's think of a slightly more complicated scenario. I've fitted sex-1 on the left hand side. I've fitted sex as an effect, but removed the intercept.

Structured Random Effects

(sex-1|nest_orig)

- Think of the left hand side as a model formula:

```
> head(model.matrix(~sex - 1, data = BTtarsus))
  sexF sexM
1    1    0
2    0    1
3    1    0
4    0    1
5    0    1
6    0    1
```


Structured Random Effects

(sex-1|nest_orig)

- Think of the left hand side as a model formula:

```
> head(model.matrix(~sex - 1, data = BTtarsus))
  sexF sexM
1    1    0
2    0    1
3    1    0
4    0    1
5    0    1
6    0    1
```

Again we can look at the design matrix to see what this implies, but for this simple example we probably have a good idea. We are fitting a female and male effect (rather than a female effect, and then a effect of the difference between males and females).

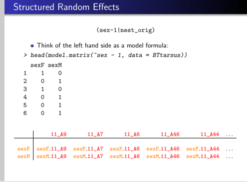
Structured Random Effects

(sex-1|nest_orig)

- Think of the left hand side as a model formula:

```
> head(model.matrix(~sex - 1, data = BTtarsus))
  sexF sexM
1    1    0
2    0    1
3    1    0
4    0    1
5    0    1
6    0    1
```

	11_A9	11_A7	11_A6	11_A46	11_A44	...
sexF	sexF.11_A9	sexF.11_A7	sexF.11_A6	sexF.11_A46	sexF.11_A44	...
sexM	sexM.11_A9	sexM.11_A7	sexM.11_A6	sexM.11_A46	sexM.11_A44	...


Structured Random Effects

(sex-1|nest_orig)

Think of the left hand side as a model formula:

> head(model.matrix(~sex - 1, data = BTtarsus))

sexF sexM
1 1 0
2 0 1
3 1 0
4 0 1
5 0 1
6 0 1

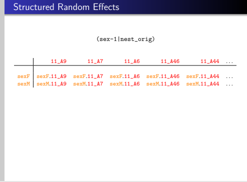
11_A9 11_A7 11_A6 11_A46 11_A44 ...
sexF sexF.11_A9 sexF.11_A7 sexF.11_A6 sexF.11_A46 sexF.11_A44 ...
sexM sexM.11_A9 sexM.11_A7 sexM.11_A6 sexM.11_A46 sexM.11_A44 ...

We now have two effects, and we can paste each one to the effects defined by the right hand side (the `nest_orig` effects again). Now, we have twice as many parameters. We have the expected effect of being from nest 11_A9 if your female, but also the expected effect of being from nest 11_A9 if your male, and this is the case for every nest.

Structured Random Effects

```
(sex-1|nest_orig)
```

	11_A9	11_A7	11_A6	11_A46	11_A44	...
sexF	sexF.11_A9	sexF.11_A7	sexF.11_A6	sexF.11_A46	sexF.11_A44	...
sexM	sexM.11_A9	sexM.11_A7	sexM.11_A6	sexM.11_A46	sexM.11_A44	...



Structured Random Effects

(sex-1|nest_orig)

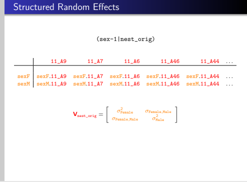
Now we've got the distribution of two sets of effects to worry about, rather than just one set. We could, for example, imagine that the between nest-of-origin variance for females is different from that in males. We might also want to allow for these two sets of effects to be correlated; the nest-of-origin effects on tarsus length are actually due to the effect of genes, and we might expect sisters and brothers to have similar tarsus lengths because they share half their genes and the effects of those genes on female and male tarsus lengths are similar.

Structured Random Effects

`(sex-1|nest_orig)`

	11_A9	11_A7	11_A6	11_A46	11_A44	...
sexF	sexF.11_A9	sexF.11_A7	sexF.11_A6	sexF.11_A46	sexF.11_A44	...
sexM	sexM.11_A9	sexM.11_A7	sexM.11_A6	sexM.11_A46	sexM.11_A44	...

$$\mathbf{V}_{\texttt{nest\_orig}} = \begin{bmatrix} \sigma^2_{\texttt{Female}} & \sigma_{\texttt{Female,Male}} \\ \sigma_{\texttt{Female,Male}} & \sigma^2_{\texttt{Male}} \end{bmatrix}$$


Structured Random Effects

(sex-1|nest_orig)

	11_A9	11_A7	11_A6	11_A46	11_A44	...
sexF	sexF.11_A9	sexF.11_A7	sexF.11_A6	sexF.11_A46	sexF.11_A44	...
sexM	sexM.11_A9	sexM.11_A7	sexM.11_A6	sexM.11_A46	sexM.11_A44	...

$$\mathbf{V}_{\texttt{nest\_orig}} = \begin{bmatrix} \sigma^2_{\texttt{Female}} & \sigma_{\texttt{Female,Male}} \\ \sigma_{\texttt{Female,Male}} & \sigma^2_{\texttt{Male}} \end{bmatrix}$$

So for this problem, rather than estimating a single variance (the variance of `nest_orig` effects) we're trying to estimate a covariance matrix. The first element ($\sigma^2_{\texttt{Female}}$) is the between nest-of-origin variance in tarsus lengths in females (the genetic variance in female tarsus length if you like) and the second element ($\sigma^2_{\texttt{Male}}$) is the between nest-of-origin variance in tarsus lengths in males. The off-diagonal element is the covariance between these two sets of effects. Usually people are more comfortable working with correlations because they know that a correlation of -1 means perfectly dissimilar and a correction of 1 means perfectly similar.

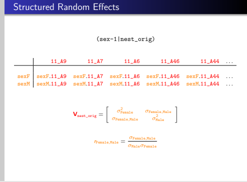
Structured Random Effects

```
(sex-1|nest_orig)
```

	11_A9	11_A7	11_A6	11_A46	11_A44	...
sexF	sexF.11_A9	sexF.11_A7	sexF.11_A6	sexF.11_A46	sexF.11_A44	...
sexM	sexM.11_A9	sexM.11_A7	sexM.11_A6	sexM.11_A46	sexM.11_A44	...

$$\mathbf{V}_{\texttt{nest\_orig}} = \begin{bmatrix} \sigma^2_{\texttt{Female}} & \sigma_{\texttt{Female,Male}} \\ \sigma_{\texttt{Female,Male}} & \sigma^2_{\texttt{Male}} \end{bmatrix}$$

$$r_{\texttt{Female,Male}} = \frac{\sigma_{\texttt{Female,Male}}}{\sigma_{\texttt{Male}}\sigma_{\texttt{Female}}}$$



and this can be obtained by taking the covariance and standardising it by the two standard deviations.

Structured Random Effects

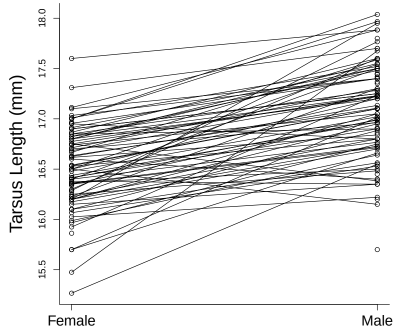
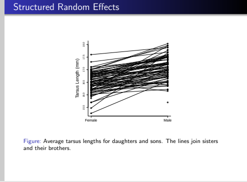


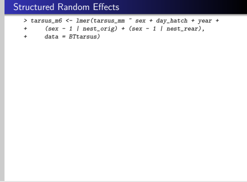
Figure: Average tarsus lengths for daughters and sons. The lines join sisters and their brothers.



What I've done here is I've taken the average tarsus length for a group of sisters (left) and the average tarsus lengths for a group of brothers (right) and then I've drawn lines that connect groups of sisters with their brothers. You can see, even from these averages, that there is a reasonably strong relationship - on average a group of sisters with large tarsi tend to have brothers with large tarsi. However, there is some noise. For example, the sisters with the second lowest mean tarsus length have brothers with tarsi lengths that are among the longest? However, it is very hard to know from this graph whether the correlation due to nest effects (the genetic correlation) is less than one. Perhaps there is only one female in this group, and she just happened to be born late, or got tangled in the nest material, and so she was at a competitive disadvantage, and couldn't grow a large tarsus length. If we could, hypothetically, have sampled a very large number of her sisters perhaps we would find that they had, on average, large tarsi like her brothers.

Structured Random Effects

```
> tarsus_m6 <- lmer(tarsus_mm ~ sex + day_hatch + year +
+      (sex - 1 | nest_orig) + (sex - 1 | nest_rear),
+      data = BTtarsus)
```

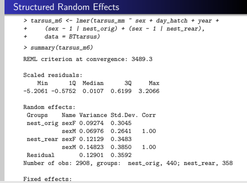

Structured Random Effects

```
> tarsus_m6 <- lmer(tarsus_mm ~ sex + day_hatch + year +
+      (sex - 1 | nest_orig) + (sex - 1 | nest_rear),
+      data = BTtarsus)
```

However, we can factor out the effect of this noise using mixed models. We have the same fixed effects, but I have also fitted sex by `nest_orig` effects and sex by `nest_rear` effects. The latter might be interpreted as asking the question; does the rearing environment effect the tarsus lengths of females and males in the same way?

Structured Random Effects

```
> tarsus_m6 <- lmer(tarsus_mm ~ sex + day_hatch + year +
+      (sex - 1 | nest_orig) + (sex - 1 | nest_rear),
+      data = BTtarsus)

> summary(tarsus_m6)

REML criterion at convergence: 3489.3

Scaled residuals:
    Min      1Q  Median      3Q     Max
-5.2061 -0.5752  0.0107  0.6199  3.2066

Random effects:
 Groups    Name Variance Std.Dev. Corr
 nest_orig sexF 0.09274  0.3045
           sexM 0.06976  0.2641   1.00
 nest_rear sexF 0.12129  0.3483
           sexM 0.14823  0.3850   1.00
 Residual       0.12901  0.3592
Number of obs: 2908, groups:  nest_orig, 440; nest_rear, 358
```


Structured Random Effects

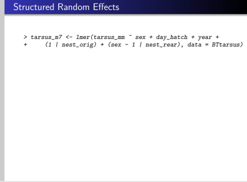
```
> tarsus_m6 <- lmer(tarsus_mm ~ sex + day_hatch + year +
+      (sex - 1 | nest_orig) + (sex - 1 | nest_rear),
+      data = BTtarsus)

> summary(tarsus_m6)

REML criterion at convergence: 3489.3

Scaled residuals:
    Min      1Q  Median      3Q     Max
-5.2061 -0.5752  0.0107  0.6199  3.2066

Random effects:
 Groups    Name Variance Std.Dev. Corr
 nest_orig sexF 0.09274  0.3045
           sexM 0.06976  0.2641   1.00
 nest_rear sexF 0.12129  0.3483
           sexM 0.14823  0.3850   1.00
 Residual       0.12901  0.3592
Number of obs: 2908, groups:  nest_orig, 440; nest_rear, 358

Fixed effects:
```

The summary is a little bit more complex. We have variances as before, so the first variance is the variance in female nest-of-origin effects, and the second variance is the variance in male nest-of-origin effects. You can see they're quite similar. In addition, the correlation between these effects is estimated to be 1.00 - right at the boundary of the possible[1] - indicating the effect of nest-of-origin is likely to be very similar for males and females.

[1]Although correlations outside the range -1 to 1 are not allowed, some programs do let the correlation exceed it's theoretical limit. As with negative variances this may seem nonsensical, but in fact it is possible to understand what it means and be less concerned when it happens.

Structured Random Effects

```
> tarsus_m7 <- lmer(tarsus_mm ~ sex + day_hatch + year +
+      (1 | nest_orig) + (sex - 1 | nest_rear), data = BTtarsus)
```



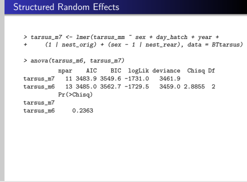
Structured Random Effects

```
> tarsus_m7 <- lmer(tarsus_mm ~ sex + day_hatch + year +
+      (1 | nest_orig) + (sex - 1 | nest_rear), data = BTtarsus)
```

We can fit a simpler model where we do not allow the nest-of-origin effects to differ by sex and we just estimate a single set of nest-of-origin effects common to both males and females.

Structured Random Effects

```
> tarsus_m7 <- lmer(tarsus_mm ~ sex + day_hatch + year +
+     (1 | nest_orig) + (sex - 1 | nest_rear), data = BTtarsus)

> anova(tarsus_m6, tarsus_m7)
          npar    AIC    BIC  logLik deviance  Chisq Df
tarsus_m7   11 3483.9 3549.6 -1731.0   3461.9
tarsus_m6   13 3485.0 3562.7 -1729.5   3459.0 2.8855  2
          Pr(>Chisq)
tarsus_m7
tarsus_m6     0.2363
```



Structured Random Effects

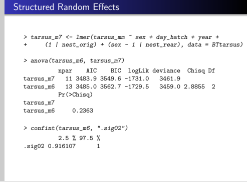
```
> tarsus_m7 <- lmer(tarsus_mm ~ sex + day_hatch + year +
+     (1 | nest_orig) + (sex - 1 | nest_rear), data = BTtarsus)

> anova(tarsus_m6, tarsus_m7)
          npar    AIC    BIC  logLik deviance  Chisq Df
tarsus_m7   11 3483.9 3549.6 -1731.0   3461.9
tarsus_m6   13 3485.0 3562.7 -1729.5   3459.0 2.8855  2
          Pr(>Chisq)
tarsus_m7
tarsus_m6     0.2363
```

A likelihood ratio test of the simple and more complex model tells us that that the more complex model is not significantly better than our simple model. You might think this was obvious from the estimates themselves since the best estimate of the correlation between the two effects was one. However, you will notice that the two models differ by two degrees of freedom because the simple model assumes that the two sets of effects have the same variance *and* a correlation of one (because they are the same effects) whereas the complex model allows the correlation to be less than one but also allows the two sets of effects to have different variances. This likelihood ratio test is therefore testing the joint null hypothesis of equal variance and a correlation of one.

Often people ask whether it is possible to fit a model where the variances are the same but the correlation is allowed to differ from one. In `lmer`, a general way of doing this has not been implemented, but if you are willing to assume that the correlation cannot be negative you can fit the model `(1|nest_orig)+(1|nest_orig:sex)`. If we designate the two variances as σ^2_1 and σ^2_2 the between nest-of-origin variance is $\sigma^2_1+\sigma^2_2$ for both sexes, and the covariance in nest-of-origin effects between the sexes is σ^2_1. The correlation is therefore $\sigma^2_1/(\sigma^2_1+\sigma^2_2)$ and has to be positive because `lmer` doesn't allow negative estimates of σ^2_1.

Structured Random Effects

```
> tarsus_m7 <- lmer(tarsus_mm ~ sex + day_hatch + year +
+     (1 | nest_orig) + (sex - 1 | nest_rear), data = BTtarsus)

> anova(tarsus_m6, tarsus_m7)
          npar    AIC    BIC  logLik deviance  Chisq Df
tarsus_m7   11 3483.9 3549.6 -1731.0   3461.9
tarsus_m6   13 3485.0 3562.7 -1729.5   3459.0 2.8855  2
          Pr(>Chisq)
tarsus_m7
tarsus_m6     0.2363

> confint(tarsus_m6, ".sig02")
          2.5 % 97.5 %
.sig02 0.916107      1
```



Structured Random Effects

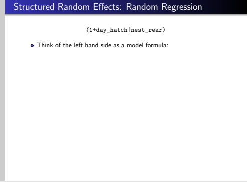
```
> tarsus_m7 <- lmer(tarsus_mm ~ sex + day_hatch + year +
+     (1 | nest_orig) + (sex - 1 | nest_rear), data = BTtarsus)

> anova(tarsus_m6, tarsus_m7)
          npar    AIC    BIC  logLik deviance  Chisq Df
tarsus_m7   11 3483.9 3549.6 -1731.0   3461.9
tarsus_m6   13 3485.0 3562.7 -1729.5   3459.0 2.8855  2
          Pr(>Chisq)
tarsus_m7
tarsus_m6     0.2363

> confint(tarsus_m6, ".sig02")
          2.5 % 97.5 %
.sig02 0.916107      1
```

We can also get confidence intervals on the correlation (in this case through a profile likelihood) and we can see that we can be fairy confident that the correlation between these effects is high - the lower 95% confidence interval is above 0.9 and so we can conclude with reasonable confidence that the genetic effects for male and female tarsus length are extremely similar.

Structured Random Effects: Random Regression

(1+day_hatch|nest_rear)

- Think of the left hand side as a model formula:


Structured Random Effects: Random Regression

(1+day_hatch|nest_rear)

- Think of the left hand side as a model formula:

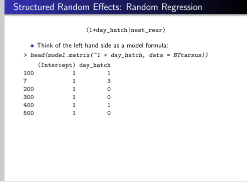
The final topic I want to discuss is something called random regression. In the previous example the model formula we used to the left of the pipe defined a model in which the coefficients referred to categorical predictors (we had a male and female effect). Now I want to go through an example where the term appearing on the left hand side of the pipe is a continuous covariate (it is true that `day_hatch` only takes 3 values (0, 1, and 3) but in the data frame it is coded as numeric).

Structured Random Effects: Random Regression

`(1+day_hatch|nest_rear)`

- Think of the left hand side as a model formula:

```
> head(model.matrix(~1 + day_hatch, data = BTtarsus))
    (Intercept) day_hatch
100           1         1
7             1         3
200           1         0
300           1         0
400           1         1
500           1         0
```



You should be comfortable with the structure of the design matrix: the first column is for the intercept and the second column is simply the covariate for which we would like to estimate a slope. So the model on the left hand side defines a two parameter model, an intercept and a slope.

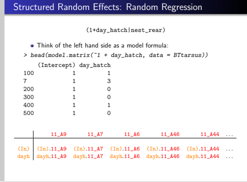
Structured Random Effects: Random Regression

`(1+day_hatch|nest_rear)`

- Think of the left hand side as a model formula:

```
> head(model.matrix(~1 + day_hatch, data = BTtarsus))
    (Intercept) day_hatch
100           1         1
7             1         3
200           1         0
300           1         0
400           1         1
500           1         0
```

	11_A9	11_A7	11_A6	11_A46	11_A44	...
(In)	(In).11_A9	(In).11_A7	(In).11_A6	(In).11_A46	(In).11_A44	...
dayh	dayh.11_A9	dayh.11_A7	dayh.11_A6	dayh.11_A46	dayh.11_A44	...

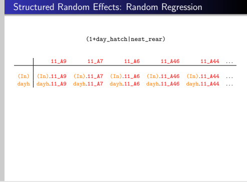

Structured Random Effects: Random Regression
(1+day_hatch|nest_rear)
Think of the left hand side as a model formula:
> head(model.matrix(~1 + day_hatch, data = BTtarsus))
(Intercept) day_hatch
100 1 1
7 1 3
200 1 0
300 1 0
400 1 1
500 1 0
11_A9 11_A7 11_A6 11_A46 11_A44 ...
(In) (In).11_A9 (In).11_A7 (In).11_A6 (In).11_A46 (In).11_A44 ...
dayh dayh.11_A9 dayh.11_A7 dayh.11_A6 dayh.11_A46 dayh.11_A44 ...

As before we can think of forming a table with these two parameters in the rows and the terms on the right-hand side of the pipe (the nest-of-rearing effects) in the columns. We can then paste these names together to get a new set of effects; an intercept for each nest (the expected tarsus length for nest `11_A9` for chicks that hatched on day 0) and a slope for each nest (how the expected tarsus length changes with `day_hatch` in nest `11_A9`). So an intercept and slope for each nest.

Structured Random Effects: Random Regression

`(1+day_hatch|nest_rear)`

	11_A9	11_A7	11_A6	11_A46	11_A44	...
(In)	(In).11_A9	(In).11_A7	(In).11_A6	(In).11_A46	(In).11_A44	...
dayh	dayh.11_A9	dayh.11_A7	dayh.11_A6	dayh.11_A46	dayh.11_A44	...


Structured Random Effects: Random Regression

(1+day_hatch|nest_rear)

	11_A9	11_A7	11_A6	11_A46	11_A44	...
(In)	(In).11_A9	(In).11_A7	(In).11_A6	(In).11_A46	(In).11_A44	...
dayh	dayh.11_A9	dayh.11_A7	dayh.11_A6	dayh.11_A46	dayh.11_A44	...

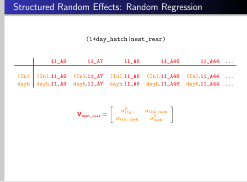
As before we have to then consider how we expect these two sets of effects to vary, and how we expect them to covary.

Structured Random Effects: Random Regression

`(1+day_hatch|nest_rear)`

	11_A9	11_A7	11_A6	11_A46	11_A44	...
(In)	(In).11_A9	(In).11_A7	(In).11_A6	(In).11_A46	(In).11_A44	...
dayh	dayh.11_A9	dayh.11_A7	dayh.11_A6	dayh.11_A46	dayh.11_A44	...

$$\mathbf{V}_{\texttt{nest\_rear}} = \begin{bmatrix} \sigma^2_{\texttt{(In)}} & \sigma_{\texttt{(In),dayh}} \\ \sigma_{\texttt{(In),dayh}} & \sigma^2_{\texttt{dayh}} \end{bmatrix}$$


Structured Random Effects: Random Regression
(1+day_hatch|nest_rear)

The most general model is to estimate the between nest-of-rearing variance in the intercept ($\sigma^2_{\texttt{(In)}}$), which is the between nest variance in tarsus length among chicks that hatch on day 0, and the between nest-of-rearing variance in the slope ($\sigma^2_{\texttt{(dayh)}}$). You might expect variation in the slopes. For example, for those nests where the parents are high quality they might be able to bring so much food to the nest that late hatched chicks (`day_hatch`=3) are not disadvantaged and so they grow at a comparable rate to their nest mates and so there is little change in tarsus length with `day_hatch`. The slope is close to zero. However, for those nests where the parents are low quality they may bring insufficient food for the whole nest and so the early hatched chicks deprive their younger siblings of food causing them to have smaller tarsi. The slope would be negative. Its often hard to know whether biologically whether you expect the intercepts and slopes to covary, but generally its a good idea to fit the covariance.

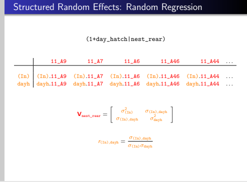
Structured Random Effects: Random Regression

(1+day_hatch|nest_rear)

	11_A9	11_A7	11_A6	11_A46	11_A44	...
(In)	(In).11_A9	(In).11_A7	(In).11_A6	(In).11_A46	(In).11_A44	...
dayh	dayh.11_A9	dayh.11_A7	dayh.11_A6	dayh.11_A46	dayh.11_A44	...

$$\mathbf{V}_{\texttt{nest\_rear}} = \begin{bmatrix} \sigma^2_{\texttt{(In)}} & \sigma_{\texttt{(In)},\texttt{dayh}} \\ \sigma_{\texttt{(In)},\texttt{dayh}} & \sigma^2_{\texttt{dayh}} \end{bmatrix}$$

$$r_{\texttt{(In)},\texttt{dayh}} = \frac{\sigma_{\texttt{(In)},\texttt{dayh}}}{\sigma_{\texttt{(In)}}\sigma_{\texttt{dayh}}}$$

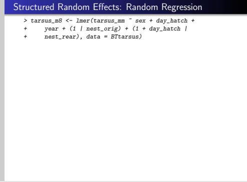

Structured Random Effects: Random Regression

(1+day_hatch|nest_rear)

As before we can get the correlation between intercepts and slopes, but in this instance it is less meaningful than say the correlation between male and female effects.

Structured Random Effects: Random Regression

```
> tarsus_m8 <- lmer(tarsus_mm ~ sex + day_hatch +
+     year + (1 | nest_orig) + (1 + day_hatch |
+     nest_rear), data = BTtarsus)
```

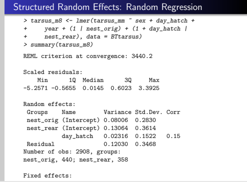


We can fit the model to the data...

Structured Random Effects: Random Regression

```
> tarsus_m8 <- lmer(tarsus_mm ~ sex + day_hatch +
+     year + (1 | nest_orig) + (1 + day_hatch |
+     nest_rear), data = BTtarsus)
> summary(tarsus_m8)
```

```
REML criterion at convergence: 3440.2

Scaled residuals:
    Min      1Q  Median      3Q     Max
-5.2571 -0.5655  0.0145  0.6023  3.3925

Random effects:
 Groups    Name        Variance Std.Dev. Corr
 nest_orig (Intercept) 0.08006  0.2830
 nest_rear (Intercept) 0.13064  0.3614
           day_hatch   0.02316  0.1522   0.15
 Residual              0.12030  0.3468
Number of obs: 2908, groups:
nest_orig, 440; nest_rear, 358

Fixed effects:
```

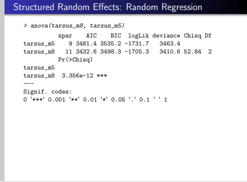

Structured Random Effects: Random Regression

```
> tarsus_m8 <- lmer(tarsus_mm ~ sex + day_hatch +
+     year + (1 | nest_orig) + (1 + day_hatch |
+     nest_rear), data = BTtarsus)
> summary(tarsus_m8)

REML criterion at convergence: 3440.2

Scaled residuals:
    Min      1Q  Median      3Q     Max
-5.2571 -0.5655  0.0145  0.6023  3.3925

Random effects:
 Groups    Name        Variance Std.Dev. Corr
 nest_orig (Intercept) 0.08006  0.2830
 nest_rear (Intercept) 0.13064  0.3614
           day_hatch   0.02316  0.1522   0.15
 Residual              0.12030  0.3468
Number of obs: 2908, groups:
nest_orig, 440; nest_rear, 358

Fixed effects:
```

and what we find is that the between-nest variance in slopes is 0.023. This is often quite hard to interpret because it is in (units of the response per units of the covariate)^2: in this example (mm per day)^2. If we take the square root we get 0.152 which is the between-nest standard deviation in slopes and is in units off mm per day. If we take extreme nests (the 2.5% and 97.5% quantiles) they will differ by approximately 4 standard deviations and so the difference in slopes would be 0.609 mm day. For late hatched chicks with a value of the covariate of 3 this would translate into a difference of 3 times this, so 1.826mm. That's about a 10% change with respect to the mean - not a massive change, but not negligible either. (If the trait was life span I'd be a bit gutted losing 8.1 years). The best estimate of the correlation is pretty low at 0.15

Structured Random Effects: Random Regression

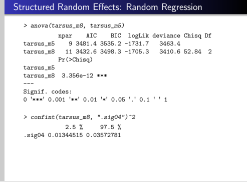
```
> anova(tarsus_m8, tarsus_m5)
          npar    AIC    BIC  logLik deviance Chisq Df
tarsus_m5    9 3481.4 3535.2 -1731.7   3463.4
tarsus_m8   11 3432.6 3498.3 -1705.3   3410.6 52.84  2
          Pr(>Chisq)
tarsus_m5
tarsus_m8  3.356e-12 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



If we do a likelihood ratio test we find that this model is substantially better at explaining our data than the simpler model, and so not surprisingly

Structured Random Effects: Random Regression

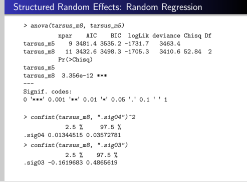
```
> anova(tarsus_m8, tarsus_m5)
          npar    AIC    BIC  logLik deviance Chisq Df
tarsus_m5    9 3481.4 3535.2 -1731.7   3463.4
tarsus_m8   11 3432.6 3498.3 -1705.3   3410.6 52.84  2
          Pr(>Chisq)
tarsus_m5
tarsus_m8  3.356e-12 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> confint(tarsus_m8, ".sig04")^2
            2.5 %     97.5 %
.sig04 0.01344515 0.03572781
```



The confidence intervals on the between-nest variance in slopes is quite tight and well away from zero.

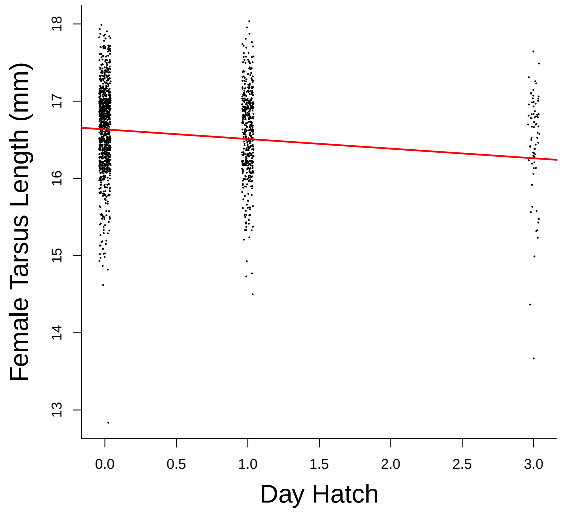
Structured Random Effects: Random Regression

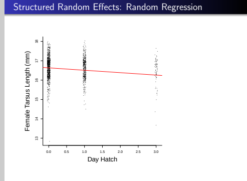
```
> anova(tarsus_m8, tarsus_m5)
          npar    AIC    BIC  logLik deviance Chisq Df
tarsus_m5    9 3481.4 3535.2 -1731.7   3463.4
tarsus_m8   11 3432.6 3498.3 -1705.3   3410.6 52.84  2
          Pr(>Chisq)
tarsus_m5
tarsus_m8  3.356e-12 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> confint(tarsus_m8, ".sig04")^2
             2.5 %     97.5 %
.sig04 0.01344515 0.03572781
> confint(tarsus_m8, ".sig03")
             2.5 %    97.5 %
.sig03 -0.1619683 0.4865619
```



The confidence intervals on the correlation between intercepts and slopes overlaps zero.

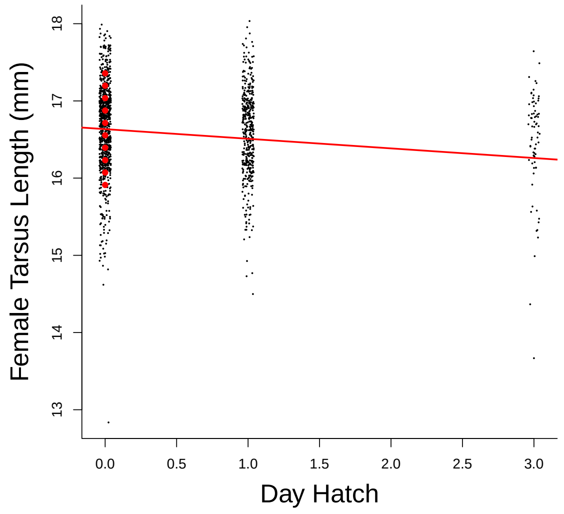
Structured Random Effects: Random Regression

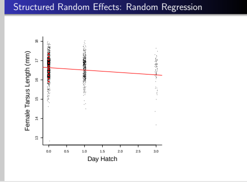




It is sometimes hard for people to understand what these models are doing, so it can help to visualise the model. These are the data, which I've jittered on the x-axis so the observations don't obscure each other. The intercept and slope of the red line are our fixed intercept and `day_hatch` slope effect. It is important that you fit `day_hatch` as a fixed effect also, because this part of the model defines the *average* effect of the covariate on the response. The random effect part of the model models deviations from this average.

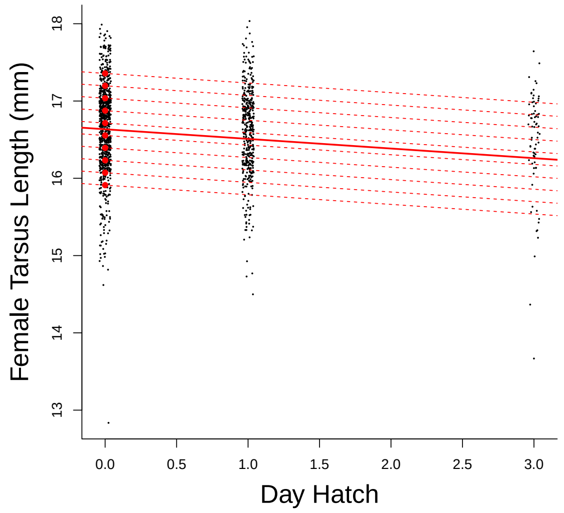
Structured Random Effects: Random Regression

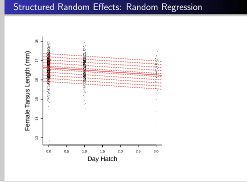




These red dots represent the nest-of-rearing effects at day 0. The highest and lowest point are at the 2.5% and 97.5% quantile of the distribution for the nest-of-origin intercepts. If you like, they represent what you expect the average tarsus length of chicks that hatch on day 0 to be in extreme nests. The actual distribution of tarsus lengths is more extreme than this, because there are additional sources of variation (nest-of-origin effects, residual effects ....). The other red points are equally spaced between the extremes.

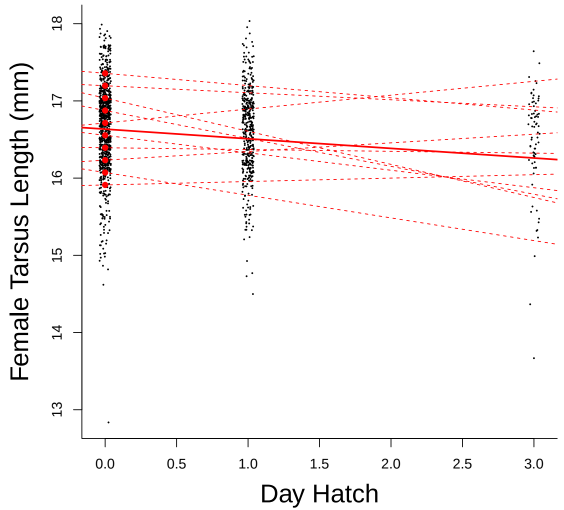
Structured Random Effects: Random Regression

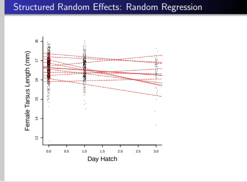




If there was no variation in slope across nests, this is how we would expect the tarsus lengths of chicks to change with `day_hatch`. The lines are all parallel because there is no variation in slope. In fact, in our model we did see variation in slope

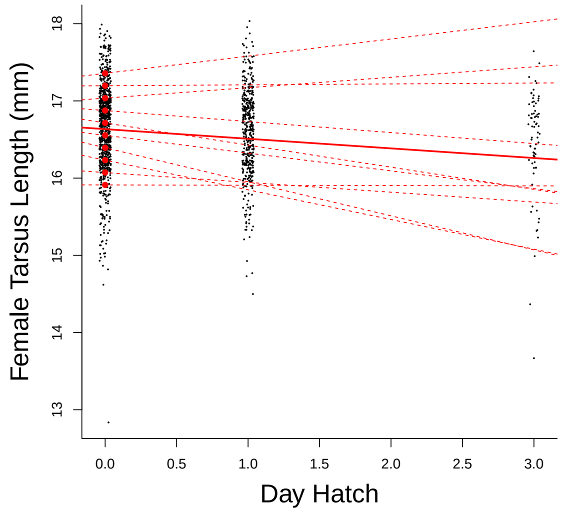
Structured Random Effects: Random Regression

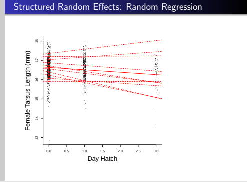




and if we take a random realisation of our model to see what the distribution of slopes would look like, it would be something like this. Some slopes are really quite negative, whereas some have such a positive deviation from the average that the slope is even slightly positive. The slopes look fairly random with respect to these red points because the correlation between intercept and slope is small in magnitude at 0.15.

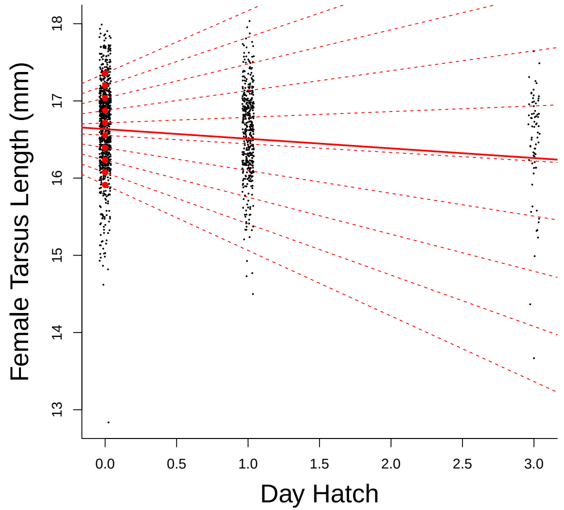
Structured Random Effects: Random Regression





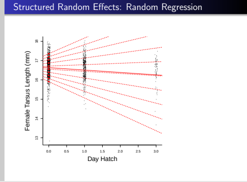
If the correlation between intercept and slope had been exactly zero, the slopes would look even more random with respect to the intercepts. In contrast, if the correlation was estimated to be large in magnitude

Structured Random Effects: Random Regression



Random Effects (II)

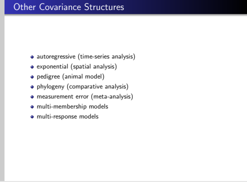
└─Structured Random Effects: Random Regression



For example if the correlation was 1 you would get this type of picture. Large intercepts are associated with large slopes and vice versa.

Other Covariance Structures

- autoregressive (time-series analysis)
- exponential (spatial analysis)
- pedigree (animal model)
- phylogeny (comparative analysis)
- measurement error (meta-analysis)
- multi-membership models
- multi-response models


Other Covariance Structures

- autoregressive (time-series analysis)
- exponential (spatial analysis)
- pedigree (animal model)
- phylogeny (comparative analysis)
- measurement error (meta-analysis)
- multi-membership models
- multi-response models

In a week there is a limit to how much I can teach you, and what I have shown you is only a tiny fraction of what you can do when you understand mixed models well. We've fitted fairly simple models to capture relatively straight forward patterns of covariation in our data. However, there's many types of biological process that cause more complicated patterns of covariance in our data, and there's a whole range of different possibilities for accounting for these types of dependency, estimating their magnitude and hopefully getting some scientific insight into how the world works.