Generalized linear mixed models

Just like LMMs, but adding non-normal response and a link function

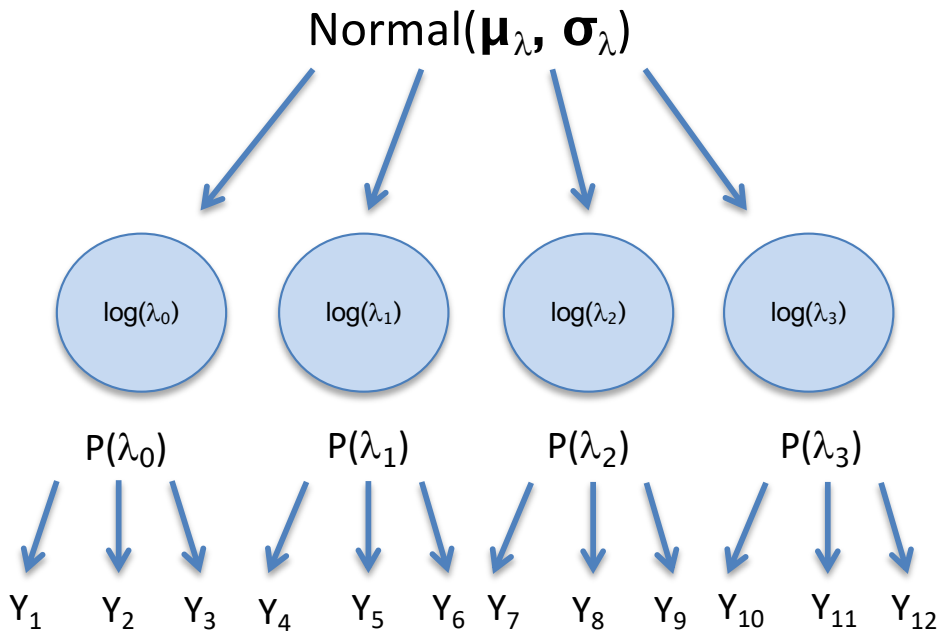
Note: random effects are assumed to be normally distributed **on the link scale**

So we have gaussian variation in the parameters, on the link scale, but non-normal variation of the actual observations

$$\log(\lambda_j) \sim \text{Normal}(\mu_\lambda, \sigma_\lambda)$$
$$Y_i \sim \text{Poisson}(\lambda_{j[i]})$$

$$\text{logit}(p_j) \sim \text{Normal}(\mu_p, \sigma_p)$$
$$Y_i \sim \text{Binomial}(n, p_{j[i]})$$

$$\log(\lambda_j) \sim \text{Normal}(\mu_\lambda, \sigma_\lambda)$$
$$Y_i \sim \text{Poisson}(\lambda_{j[i]})$$



Reminder: Why does normal distribution usually work well for random effects, even if the raw data are clearly non-normal?

Example: roundworm infection in red deer, in Spain

Binary (infected or not); measured on 24 farms with varying # observations

```
table(deer$Farm)

##
##   AL   AU   BA   BE   CB  CRC   HB   LN  MAN   MB   MO   NC   NV   PN   QM
##   15   32   50   13   85    1   17   33   27   34  209   27   20   37   60
##   RF   RN   RO  SAU   SE   TI   TN VISO   VY
##   20   23   30    3   26   19   25   13    7
```

```
tapply(deer$Ecervi, deer$Farm, mean)

##     AL     AU     BA     BE     CB    CRC     HB     LN    MAN     MB
## 0.4000 0.8750 0.8200 1.0000 0.6941 0.0000 0.1176 0.9091 0.4074 0.8824
##     MO     NC     NV     PN     QM     RF     RN     RO    SAU     SE
## 0.3923 0.4815 0.5500 0.9189 0.8000 0.9000 0.3043 0.9333 0.0000 0.7692
##     TI     TN   VISO     VY
## 0.9474 0.7200 0.8462 0.8571
```

```
tapply(deer$Ecervi, deer$Sex, mean)

## female   male
## 0.6942 0.5899
```

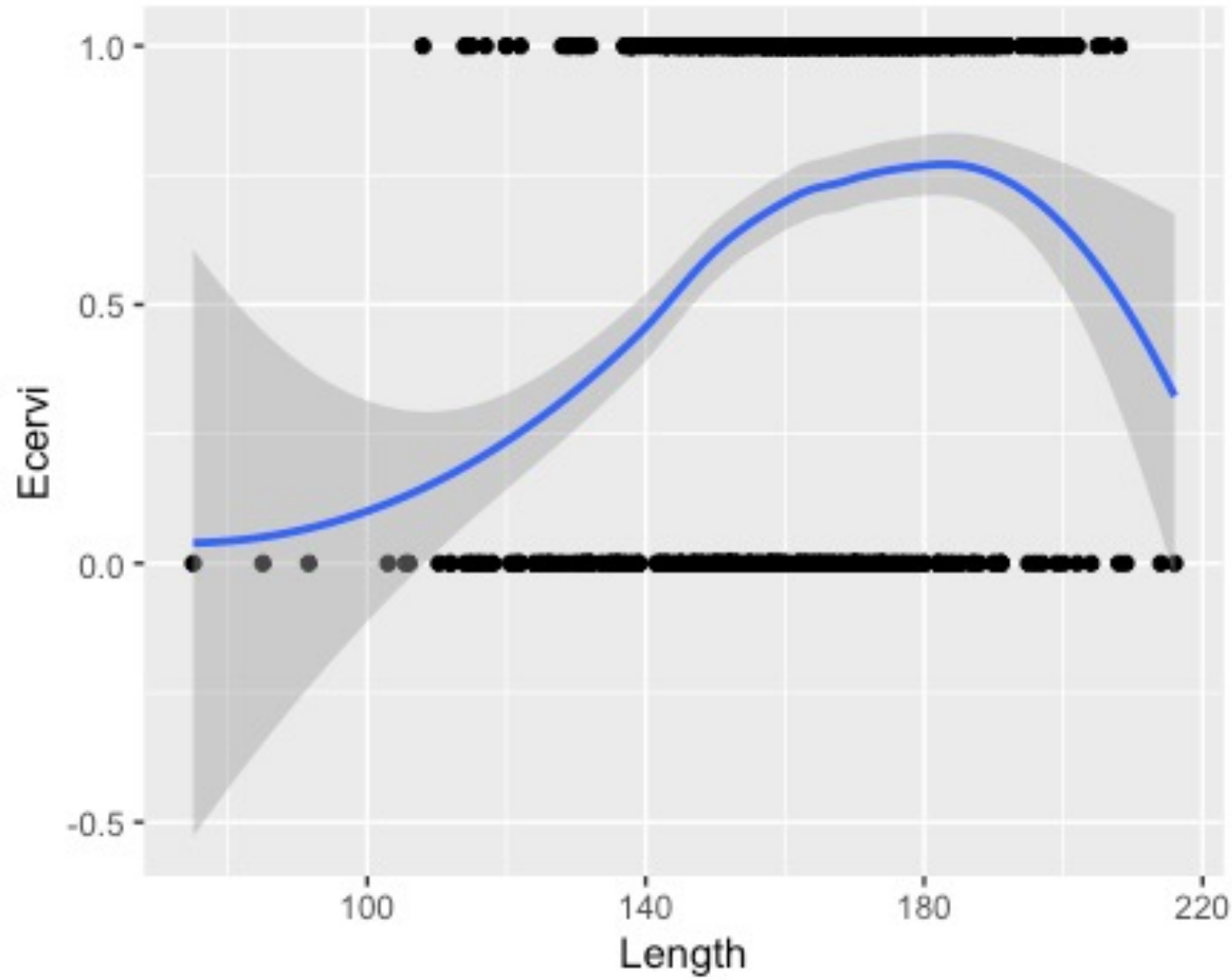
Question: We want to test whether various predictors can explain infection prevalence.

- Should we model this data as binary (i.e., binomial with n = 1), or as a binomial proportion (i.e., binomial with n > 1: what proportion of deer are infected on a farm)?

Example: roundworm infection in red deer, in Spain

Binary (infected or not); measured on 24 farms with varying # observations

```
ggplot(deer, aes(Length, Ecervi)) + geom_point() + geom_smooth()
```



Example: roundworm infection in red deer, in Spain

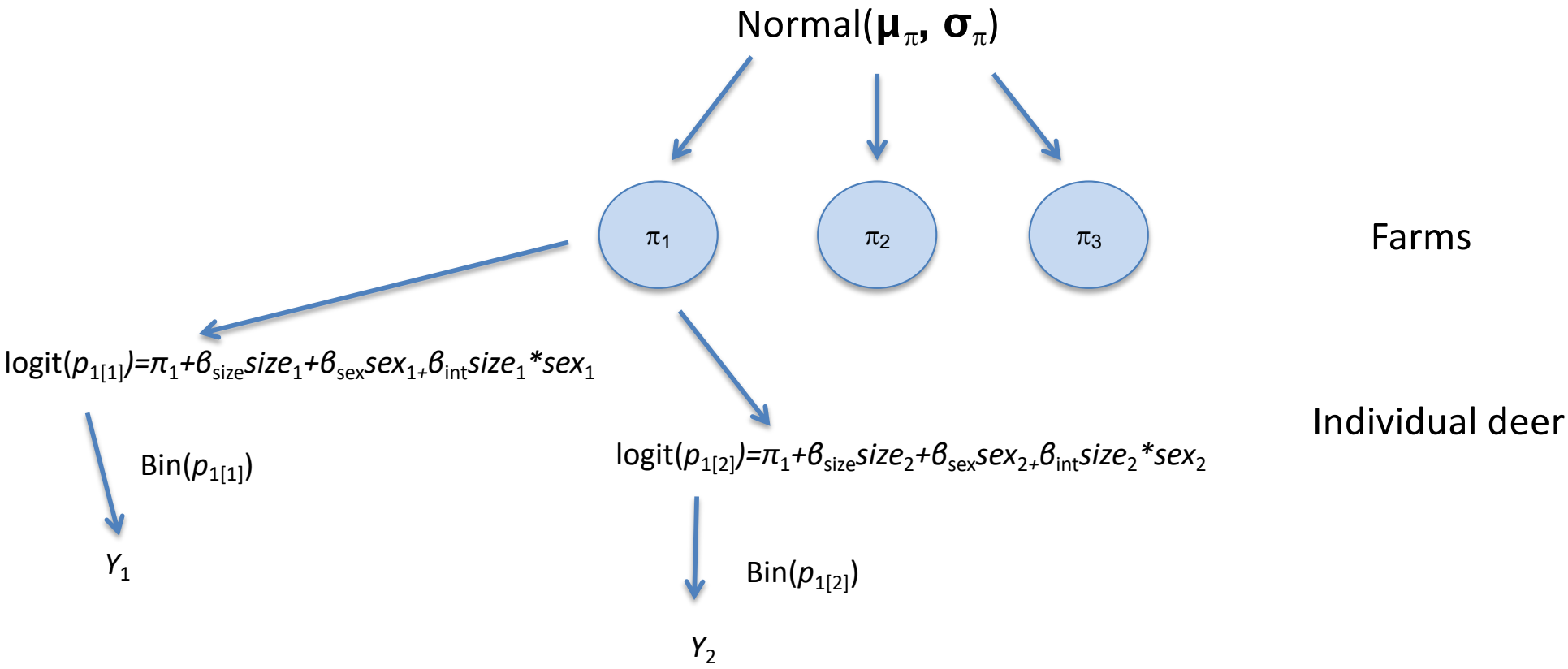
Centering the predictor (Length): this time, so the intercept is interpretable

```
deer$Length = deer$Length - mean(deer$Length)
```

Now the intercept will be probability of infection at mean deer size

Example: roundworm infection in red deer, in Spain

$$\pi_j \sim \text{Normal}(\mu_\pi, \sigma_\pi)$$
$$\text{logit}(p_{j[i]}) = \pi_j + \beta_{size} size_i + \beta_{sex} sex_i + \beta_{int} size_i * sex_i$$
$$Y_i \sim \text{Binomial}(n = 1, p_{j[i]})$$



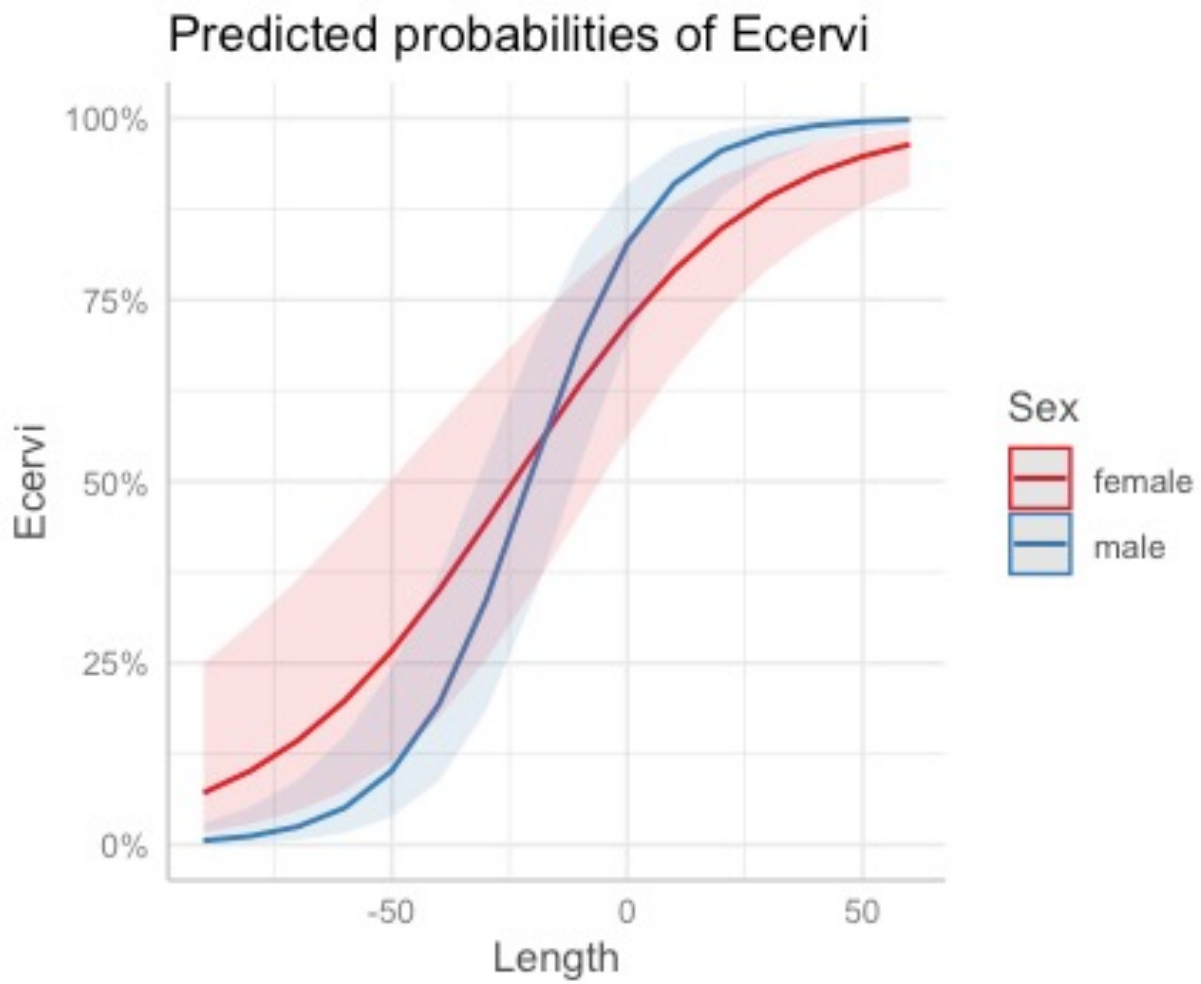
```
mod = glmer(Ecervi ~ Sex*Length + (1|Farm), data = deer, family = binomial)
```

```
summary(mod)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: Ecervi ~ Sex * Length + (1 | Farm)
##    Data: deer
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  Farm   (Intercept) 2.39     1.55
## Number of obs: 826, groups:  Farm, 24
##
## Fixed effects:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     0.93897    0.35600    2.64   0.0084 **
## Sexmale         0.62449    0.22294    2.80   0.0051 **
## Length          0.03896    0.00692    5.63  1.8e-08 ***
## Sexmale:Length  0.03586    0.01141    3.14   0.0017 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

exp(0.94)/(1 + exp(0.94)) = 0.72

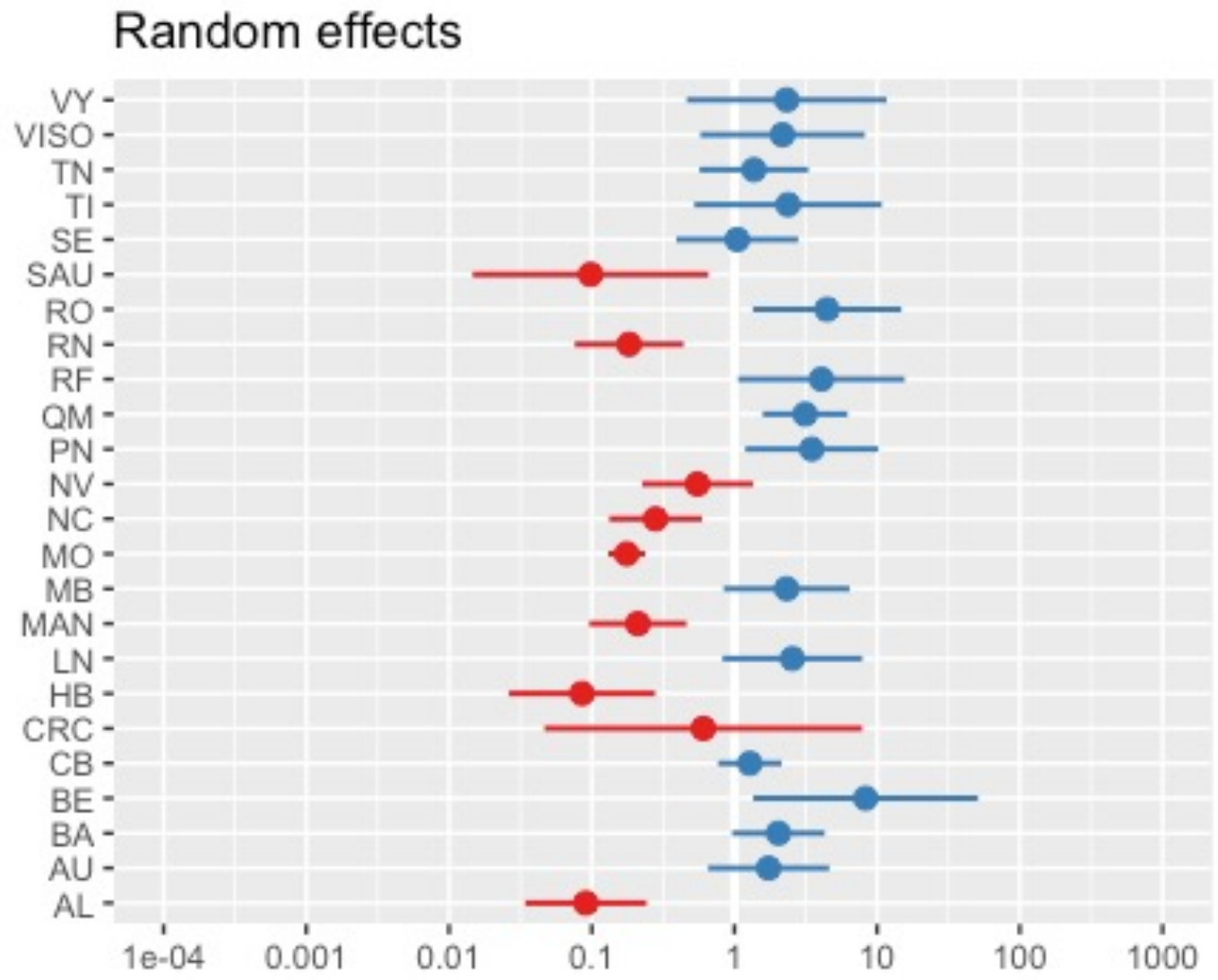
```
mod = glmer(Ecervi ~ Sex*Length + (1|Farm), data = deer, family = binomial)
```



Deer size has a huge effect on infection probability

Males show a somewhat steeper response

```
mod = glmer(Ecervi ~ Sex*Length + (1|Farm), data = deer, family = binomial)
```



Inference with GLMMs

Even harder/trickier than for LMMs

Now we don't have approximate F-tests as an option

Can use LRT, but still anti-conservative, especially at small sample size

Same for AIC

Can use parametric bootstrap, but it can be slow

```
> PBmodcomp(mod, mod.nointer)
Parametric bootstrap test; time: 1414.07 sec; samples: 1000 extremes: 2;
Requested samples: 1000 Used samples: 992 Extremes: 2
large : Ecervi ~ Sex * Length + (1 | Farm)
small : Ecervi ~ Sex + Length + (1 | Farm)
         stat df  p.value
LRT    10.225  1 0.001385 **
PBtest 10.225    0.003021 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Inference with GLMMs

Even harder/trickier than for LMMs

Now we don't have approximate F-tests as an option

Can use LRT, but still anti-conservative, especially at small sample size

Same for AIC

Can use parametric bootstrap, but it can be slow

```
anova(mod, mod.nointer)

## Data: deer
## Models:
## mod.nointer: Ecervi ~ Sex + Length + (1 | Farm)
## mod: Ecervi ~ Sex * Length + (1 | Farm)
##             Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)
## mod.nointer  4 841 860   -416      833
## mod          5 833 856   -411      823  10.2      1     0.0014 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Overdispersion in GLMMs

Previously we used quasi-likelihood, or negative binomial (counts only)

Either is possible with MMs, but no quasi-likelihood in lme4

Can do negative binomial with glmer.nb()

I will explain a third option that uses random effects and can be used with count data or binomial data

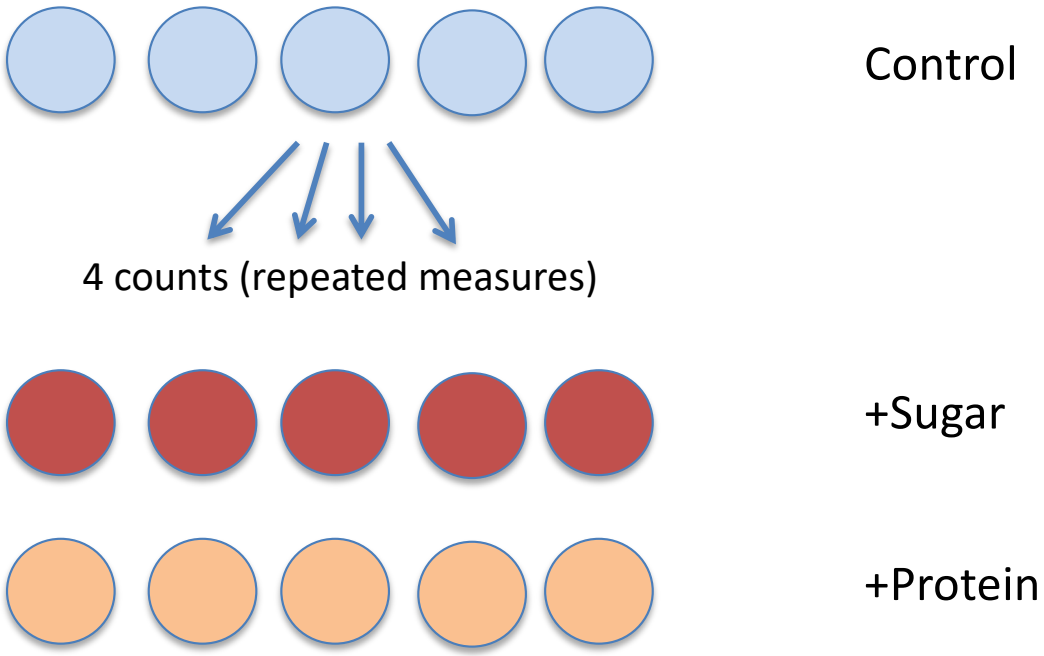
Honeybee supplementation experiment:

Control, +Sugar syrup, +Protein

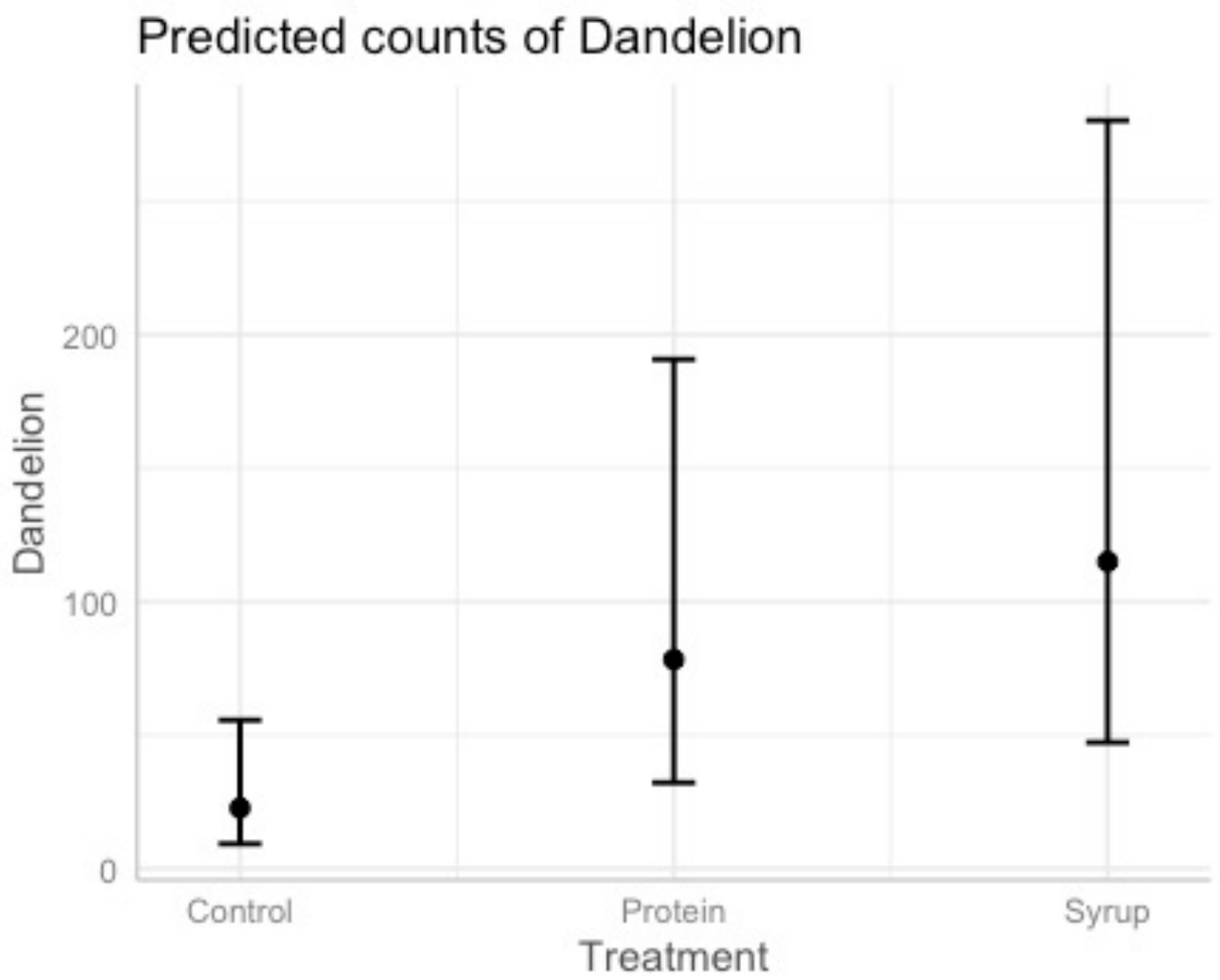
5 hives per treatment

4 measurements per hive

Counts of pollen grains in traps at hive entrance



```
mod = glmer(Dandelion ~ Treatment + (1|Hive), data = pollen, family = poisson)
```



Quantifying overdispersion

- Sum of the squared Pearson residuals, divided by the degrees of freedom

$$\varphi = \frac{\sum_i^n \varepsilon_i^2}{n - p}$$

- *Phi* = dispersion parameter; *n* = number of samples; *p* = number of coefficients
- This was already a rough estimate, but now we have issue with counting parameters in mixed models
- Suggests that this number will often be **too small**

```
library(RVAideMemoire)

overdisp.glmer(mod)

## Residual deviance: 971.236 on 56 degrees of freedom (ratio: 17.343)
```

Negative binomial GLMM

Negative binomial GLMM uses a special function in the lme4 package

```
mod.nb = glmer.nb(Dandelion ~ Treatment + (1|Hive), data = pollen)

overdisp.glmer(mod.nb)

## Residual deviance: 54.634 on 56 degrees of freedom (ratio: 0.97
6)
```

Negative binomial GLMM

Negative binomial instead of observation-level random effect

```
> anova(mod.nb, mod.nb.notreat)
Data: pollen
Models:
mod.nb.notreat: Dandelion ~ 1 + (1 | Hive)
mod.nb: Dandelion ~ Treatment + (1 | Hive)
               Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
mod.nb.notreat  3 640.95 647.23 -317.47   634.95
mod.nb          5 639.23 649.70 -314.61   629.23 5.7163      2    0.05738 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> PBmodcomp(mod.nb, mod.nb.notreat)
Parametric bootstrap test; time: 172.58 sec; samples: 1000 extremes: 102;
Requested samples: 1000 Used samples: 999 Extremes: 102
large : Dandelion ~ Treatment + (1 | Hive)
small : Dandelion ~ 1 + (1 | Hive)
         stat df p.value
LRT    5.7163  2 0.05738 .
PBtest 5.7163    0.10300
---
```

Observation-level random effect, aka individual-level random effect

Another option for overdispersion worth knowing about: the observation-level or individual-level random effect

```
pollen$ID = factor(1:nrow(pollen))
mod.id = glmer(Dandelion ~ Treatment + (1|Hive) + (1|ID), data = pollen,
family = poisson)
```

Random effects are normally distributed on the log link scale

The Poisson mean, lambda, varies among **observations from the same hive**:

log(lambda) ~ Normal(0, sd = SD for ID), or lambda ~ LogNormal

The variance for ID quantifies extra variability in the counts beyond what the Poisson expects

Observation-level random effect, aka individual-level random effect

```
summary(mod.id)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: Dandelion ~ Treatment + (1 | Hive) + (1 | ID)
##    Data: pollen
##
##      AIC      BIC   logLik deviance df.resid
##    642.4    652.9   -316.2    632.4       55
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.4602 -0.1445  0.0223  0.1031  0.6259
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  ID     (Intercept) 0.389    0.624
##  Hive   (Intercept) 1.062    1.031
## Number of obs: 60, groups:  ID, 60; Hive, 15
```

```
overdisp.glmer(mod.id)
```

```
## Residual deviance: 12.721 on 55 degrees of freedom (ratio: 0.231)
```

Observation-level random effect, aka individual-level random effect

```
mod.id.notreat = glmer(Dandelion ~ 1 + (1|Hive) + (1|ID), data = pollen
, family = poisson)
```

```
PBmodcomp(mod.id, mod.id.notreat)
```

```
## Bootstrap test; time: 89.07 sec; samples: 1000; extremes: 110;
## large : Dandelion ~ Treatment + (1 | Hive) + (1 | ID)
## Dandelion ~ 1 + (1 | Hive) + (1 | ID)
##          stat df p.value
## LRT    5.3079  2 0.07037 .
## PBtest 5.3079    0.11089
```

Generalized additive mixed models

Naturally you might want to use random effects, and also let continuous predictors have a nonlinear relationship

Still a frontier, but the software is getting better

Example: revisiting the coyote-wolf hybrid data

1-4 individuals per pack – random effect

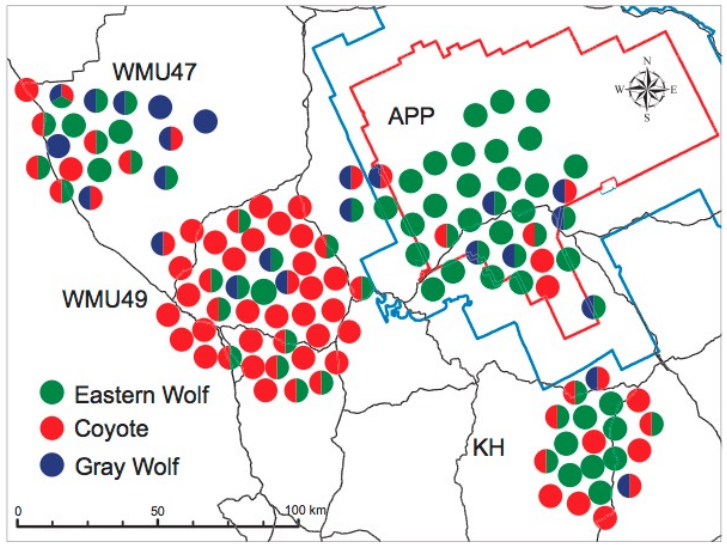


Fig. 5 Study area with resident individuals plotted (approximately) at home range centroids with pie charts showing genotypes based on individual assignment to genetic clusters using Structure and PCA. >1 colour in pie charts indicates admixture. Pie charts are simplified to show 100% ancestry for highly assigned individuals, and 50–50% or 33–33–33% for individuals admixed between 2 or 3 parental clusters, respectively. Also shown are major roads (black lines), APP boundary (red line) and harvest ban buffer area boundary (blue line).

gamm4 combines mgcv and lme4 – also check out **brms**

```
library(gamm4)
mod.gamm = gamm4(CoyoteAncestry.Logit. ~ s(PrimaryRds) + s(sqrt(SecondaryRds))
+ s(TertiaryRds, by = StudyArea) + s(Deer) + s(Moose) + StudyArea, random = ~
(1|PackNumber), data = wolf)
```

Important: model output comes in two parts / slots

```
summary(mod.gamm$gam)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## CoyoteAncestry.Logit. ~ s(PrimaryRds) + s(sqrt(SecondaryRds)) +
##     s(TertiaryRds, by = StudyArea) + s(Deer) + s(Moose) + StudyArea
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -2.647      0.864   -3.06   0.0030 **
## StudyAreaOut    3.053      0.958    3.19   0.0021 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                               edf Ref.df     F p-value
## s(PrimaryRds)                1.00   1.00  0.61 0.43553
## s(sqrt(SecondaryRds))        1.00   1.00 13.38 0.00046 ***
## s(TertiaryRds):StudyAreaAPP 2.61   2.61  1.82 0.15537
## s(TertiaryRds):StudyAreaOut 1.00   1.00  5.10 0.02671 *
## s(Deer)                      1.00   1.00  1.24 0.26892
## s(Moose)                     1.00   1.00  6.37 0.01367 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.583
## lmer.REML = 344.75  Scale est. = 2.3626    n = 85
```

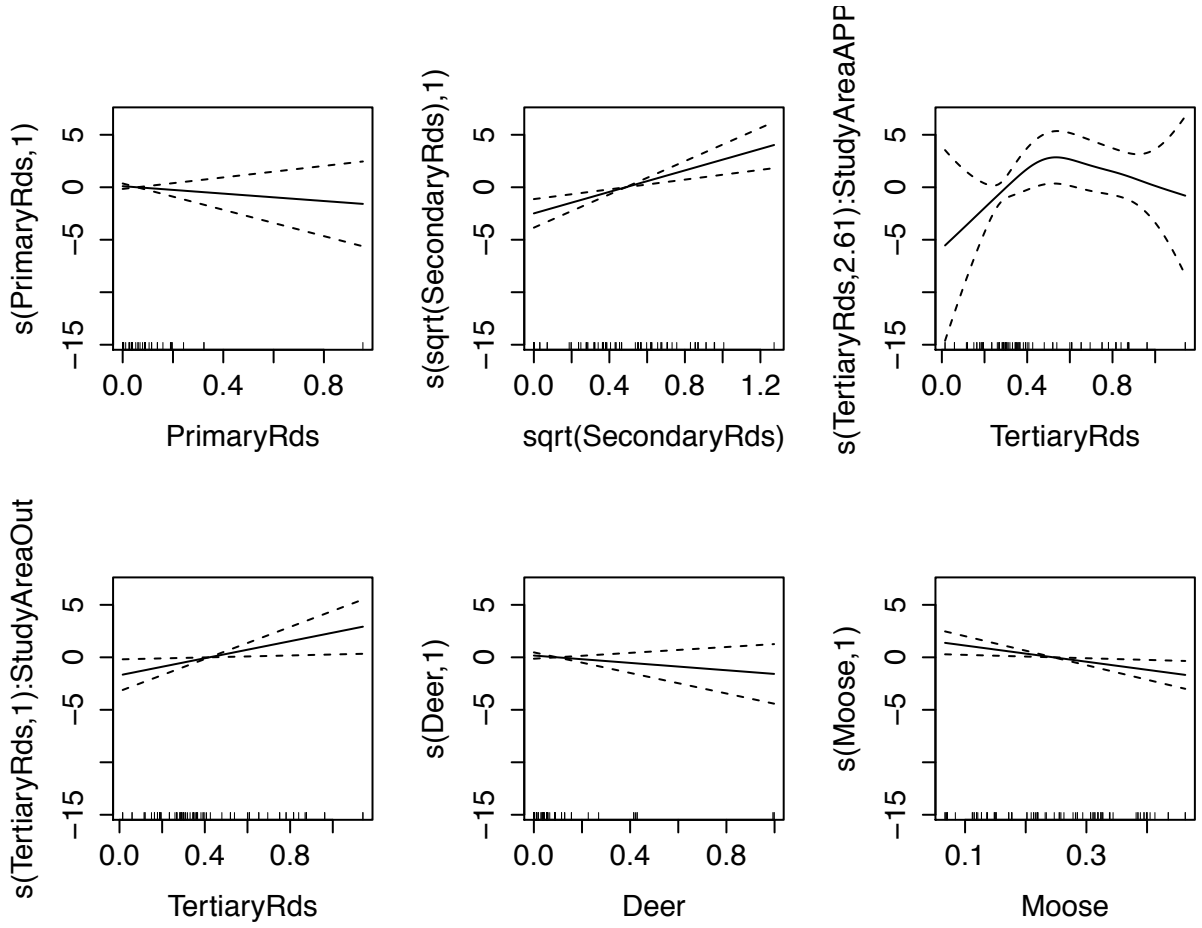
```
summary(mod.gamm$mer)
```

```
## Random effects:
##  Groups     Name                           Variance Std.Dev.
##  PackNumber (Intercept)                      1.38    1.17
##  Xr.4       s(Moose)                         0.00    0.00
##  Xr.3       s(Deer)                          0.00    0.00
##  Xr.2       s(TertiaryRds):StudyAreaOut      0.00    0.00
##  Xr.1       s(TertiaryRds):StudyAreaAPP    200.64   14.16
##  Xr.0       s(sqrt(SecondaryRds))            0.00    0.00
##  Xr         s(PrimaryRds)                    0.00    0.00
##  Residual                                    2.36    1.54
## Number of obs: 85, groups:
## PackNumber, 47; Xr.4, 8; Xr.3, 8; Xr.2, 8; Xr.1, 8; Xr.0, 8; Xr, 8
##
## Fixed effects:
##                                   Estimate Std. Error t value
## X(Intercept)                        -2.647      0.864   -3.06
## XStudyAreaOut                        3.053      0.958    3.19
## Xs(PrimaryRds)Fx1                   -0.216      0.276   -0.78
## Xs(sqrt(SecondaryRds))Fx1            1.511      0.413    3.66
## Xs(TertiaryRds):StudyAreaAPPFx1      1.681      3.968    0.42
## Xs(TertiaryRds):StudyAreaOutFx1      1.071      0.474    2.26
## Xs(Deer)Fx1                         -0.351      0.316   -1.11
## Xs(Moose)Fx1                        -0.840      0.333   -2.52
```

GAMMs use random effects to fit the smoother – variance proportional to curviness

Don’t worry about this output: use for random effects, and for likelihood comparisons

```
par(mfrow = c(2,3))
plot(mod.gamm$gam)
```



Could just refit as LMM

Inference with GAMMs is a bit sketchy: LRTs, AIC, F/t all pretty approximate

```r
mod.gamm.mooselinear = gamm4(CoyoteAncestry.Logit. ~ s(PrimaryRds) + s(sqrt(Se
condaryRds)) + s(TertiaryRds, by = StudyArea) + s(Deer) + Moose + StudyArea, r
andom = ~ (1|PackNumber), data = wolf, REML = FALSE)


mod.gamm.nomoose = gamm4(CoyoteAncestry.Logit. ~ s(PrimaryRds) + s(sqrt(Second
aryRds)) + s(TertiaryRds, by = StudyArea) + s(Deer) + StudyArea, random = ~ (1
|PackNumber), data = wolf, REML = FALSE)


anova(mod.gamm.nomoose$mer, mod.gamm.mooselinear$mer)
```

```
## Data:
## Models:
## mod.gamm.nomoose$mer: NULL
## mod.gamm.mooselinear$mer: NULL
##                            Df AIC BIC logLik deviance Chisq Chi Df
## mod.gamm.nomoose$mer       14 377 411   -174      349
## mod.gamm.mooselinear$mer 15 375 412   -173      345  3.85      1
##                            Pr(>Chisq)
## mod.gamm.nomoose$mer
## mod.gamm.mooselinear$mer       0.05 *
## ---
```

```r
library(MuMIn)
AICc(mod.gamm.mooselinear$mer)
```

```
## [1] 382.1
```

```r
AICc(mod.gamm.nomoose$mer)
```

```
## [1] 383
```