Workshop 7: Generalized linear (mixed) models

QCBS R Workshop Series

Québec Centre for Biodiversity Science

About this workshop

REPO DEV | WIKI 07 | SLIDES 07 | SLIDES 07 | SCRIPT 07

Required packages

- ggplot2
- lme4
- MASS
- vcdExtra
- bbmle
- DescTools

```
install.packages(c('ggplot2', 'lme4', 'MASS', 'vcdExtra', 'bbmle', 'DescT
```

Outline

1. Why be normal? (*Your data is ok; it's the model that's wrong*)
2. GLM with binary and proportion data
3. GLM with count data
4. GLMMs

Why be normal?

Your data is OK!

It is the model that is wrong

Limitations of linear (mixed) models

Load dataset and fit a linear model (`lm()`):

```
# make sure you're in the right working directory
mites ← read.csv('data/mites.csv')
head(mites)
str(mites)
```

The dataset that you have just loaded is a subset of the 'Oribatid mite dataset':

> 70 moss and mite samples;
>
> 5 environmental measurements and abundance of the mite *Galumna* sp. for each site.

Goal: Model the abundance (`abund`), occurrence (`pa`), and proportion (`prop`) of ***Galumna*** as a function of the five environmental variables.

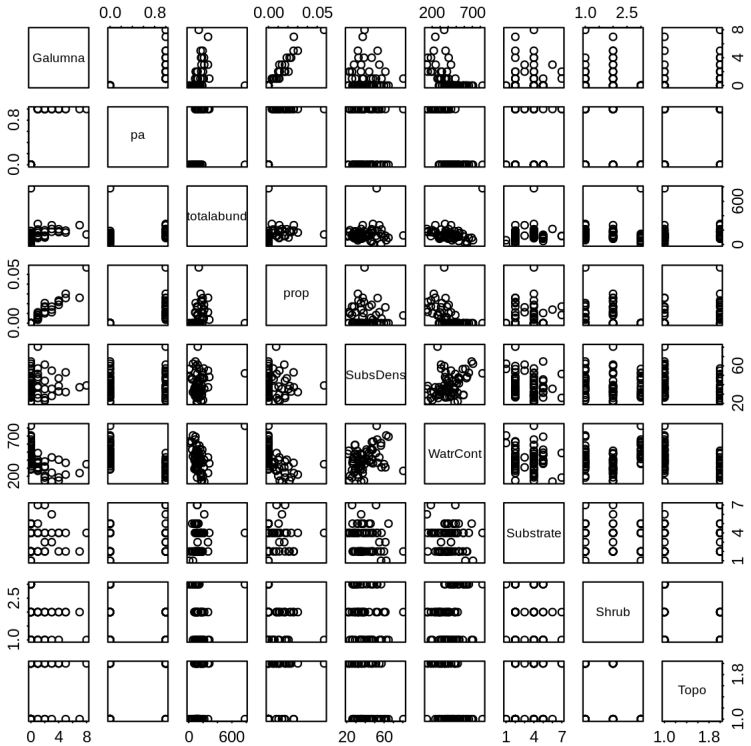
Exploring relationships

Can we see any relationship(s) between *Galumna* and the five environmental variables?

Exploring relationships

Can we see any relationship(s) between *Galumna* and the 5 environmental variables?

```
plot(mites)
```

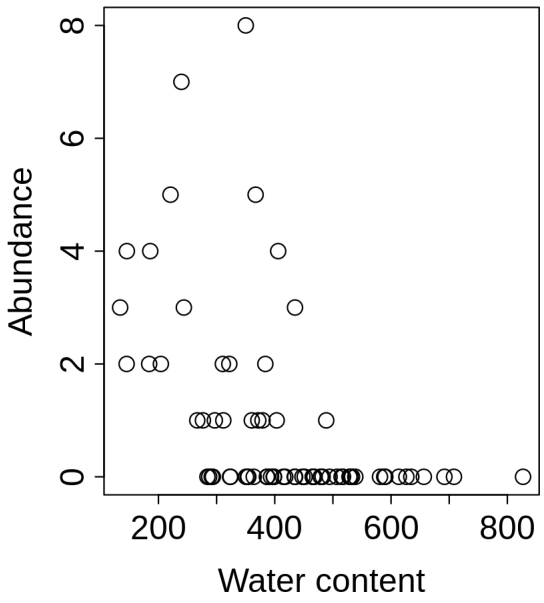


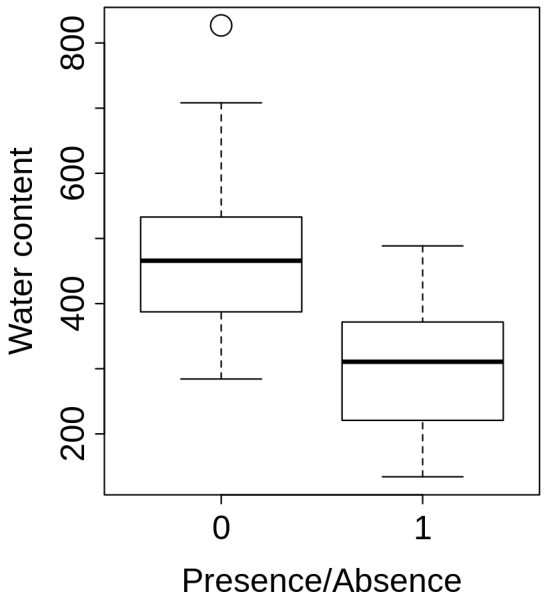
`Galumna` vs `WatrCont`?!

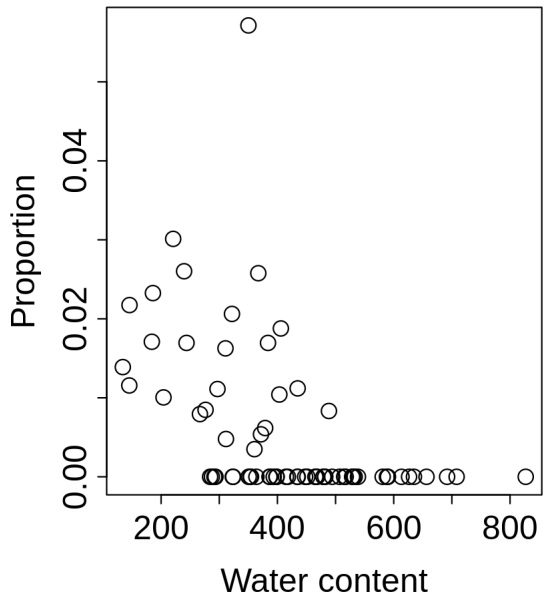
Exploring relationships

A negative relationship between `Galumna` and `WatrCont`?

```
par(mfrow = c(1, 3), cex = 1.4)
plot(Galumna ~ WatrCont, data = mites, xlab = 'Water content', ylab='Ab
boxplot(WatrCont ~ pa,
        data = mites, xlab='Presence/Absence', ylab = 'Water content')
plot(prop ~ WatrCont, data = mites, xlab = 'Water content', ylab='Propo
```







Testing linearity

Fit linear models to test whether `abund`, `pa`, and/or `prop` vary as a function of water content.

```
  lm.abund ← lm(Galumna ~ WatrCont, data = mites)
##   summary(lm.abund)
  lm.pa ← lm(pa ~ WatrCont, data = mites)
##   summary(lm.pa)
  lm.prop ← lm(prop ~ WatrCont, data = mites)
##   summary(lm.prop)
```

```
# Extracting the 4th coefficient

summary(lm.abund)$coefficients[,
#  (Intercept)     WatrCont
# 3.981563e-08 1.206117e-05
summary(lm.pa)$coefficients[, 4]
#  (Intercept)     WatrCont
# 6.030252e-12 4.676755e-08
summary(lm.prop)$coefficients[, 4
#  (Intercept)     WatrCont
```

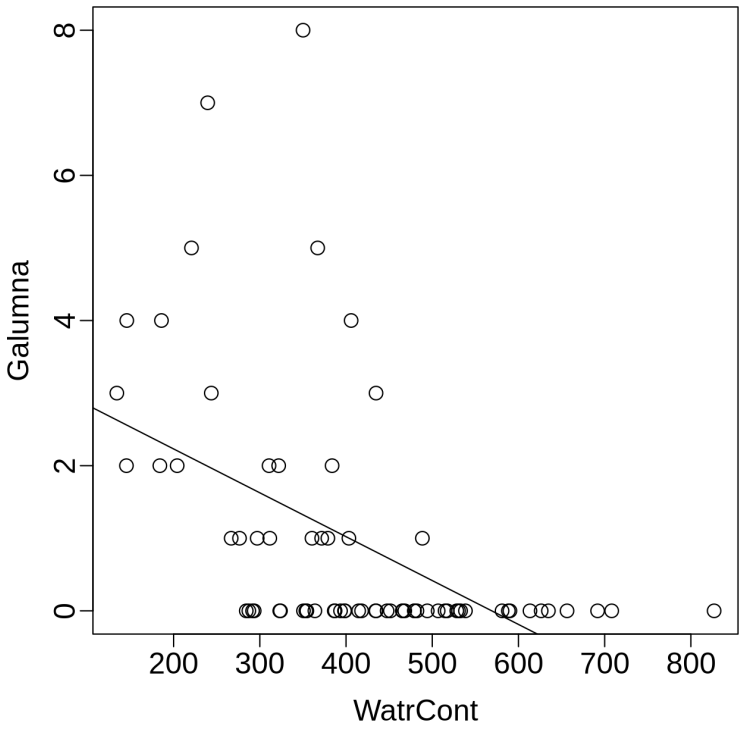
Significant relationship in all models!

But...

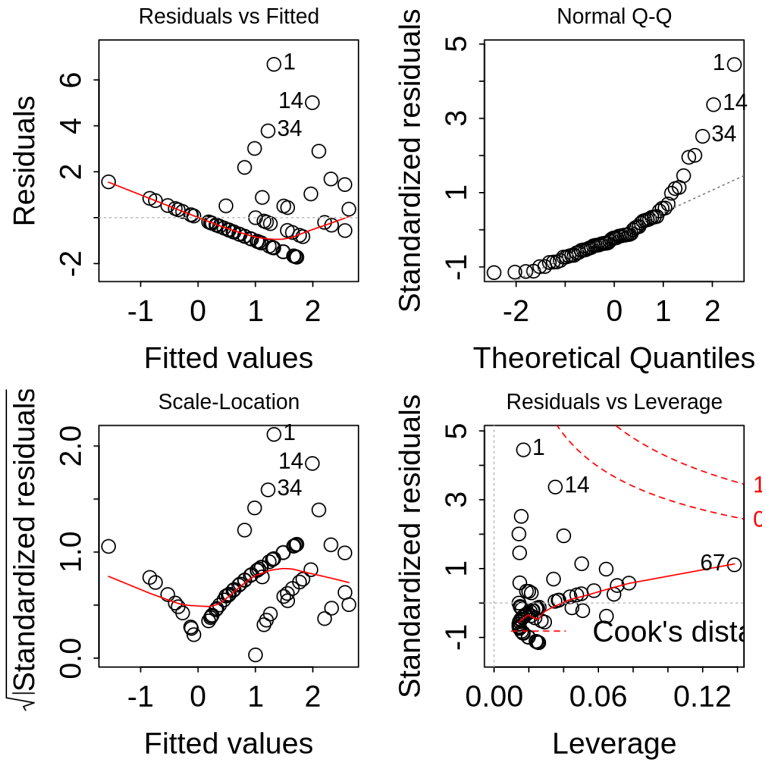
Testing linearity

Significant relationship in all models! **Wait a minute...**

```
plot(Galumna ~ WatrCont, data = m
abline(lm.abund)
```



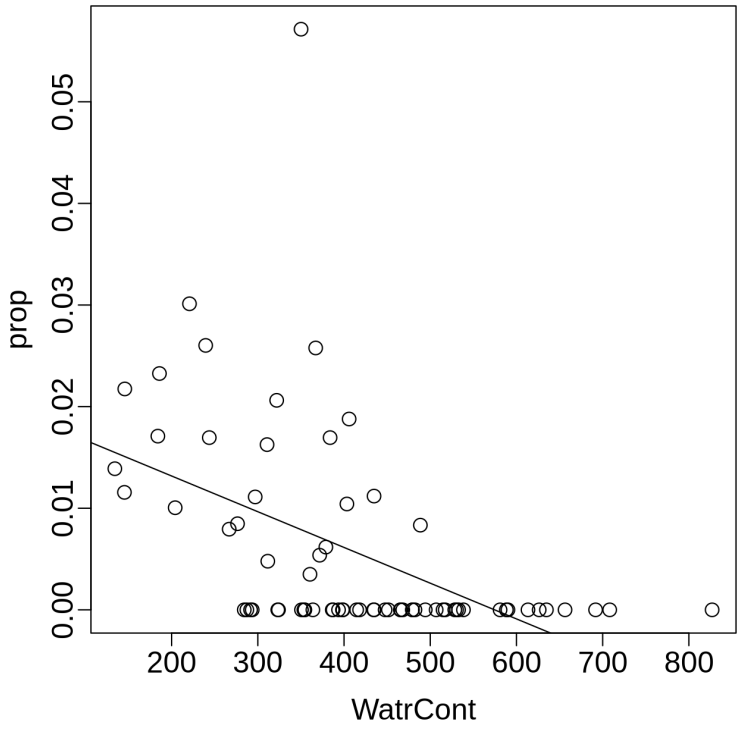

```
par(mfrow = c(2, 2), cex = 1.4)
plot(lm.abund)
```



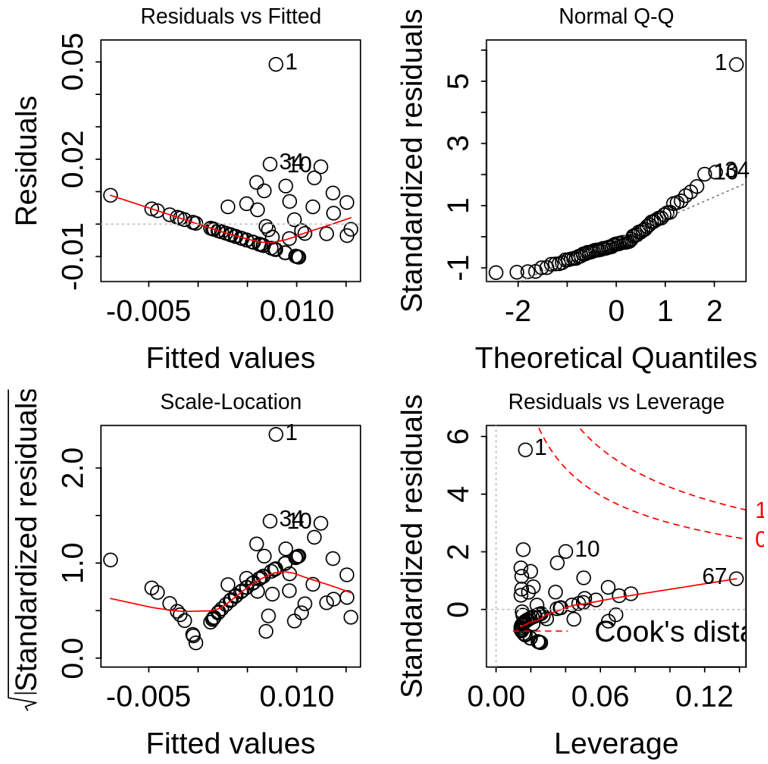
Testing linearity

Even worse for other models (proportion; `prop`):

```
plot(prop ~ WatrCont, data = mite
abline(lm.prop)
```



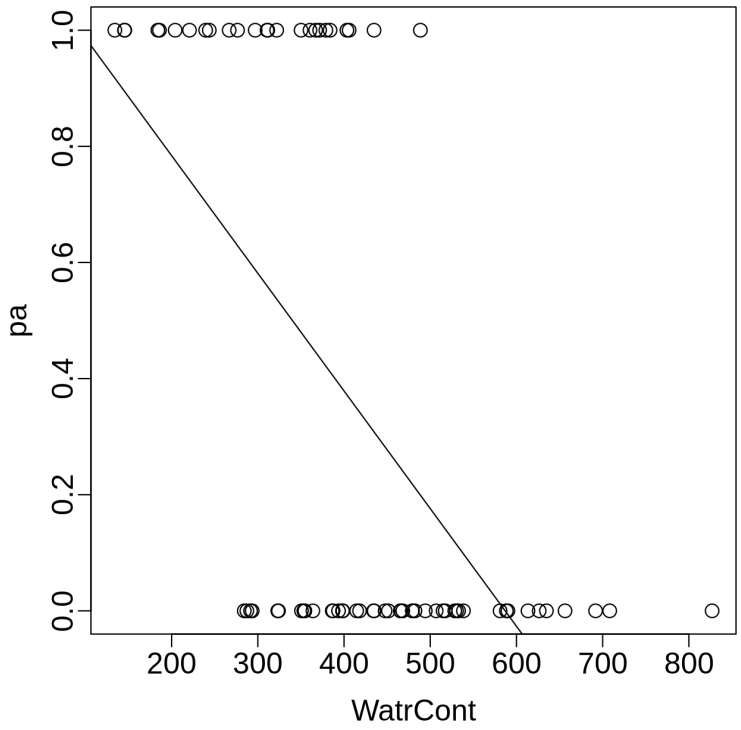

```
par(mfrow = c(2, 2), cex = 1.4)
plot(lm.prop)
```



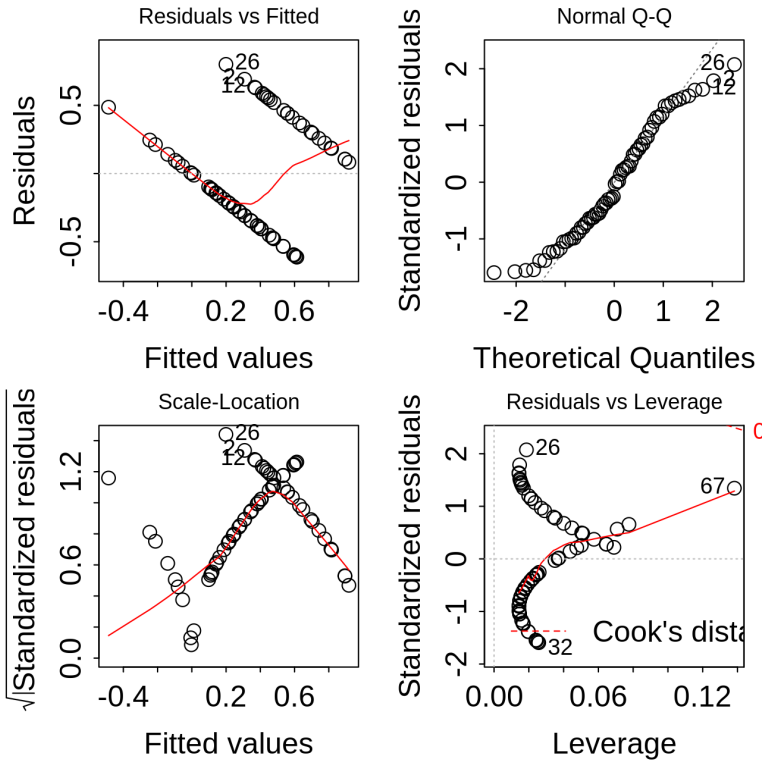
Testing linearity

Even worse for other models (presence-absence; `pa`):

```
plot(pa ~ WatrCont, data = mites)
abline(lm.pa)
```




```
par(mfrow = c(2, 2), cex = 1.4)
plot(lm.pa)
```



Model assumptions

It is common in Ecology that assumptions of homogeneity of variance and normality are not met.

GLM are very useful to deal with these issues!

Let us revisit the assumptions of linear models...

Model assumptions

A linear model can be represented by:

$y = \beta_0 + \beta_1 x_i + \varepsilon$

where:

y_i = predicted value of a response variable

β_0 = intercept

β_1 = slope

x_i = explanatory variable

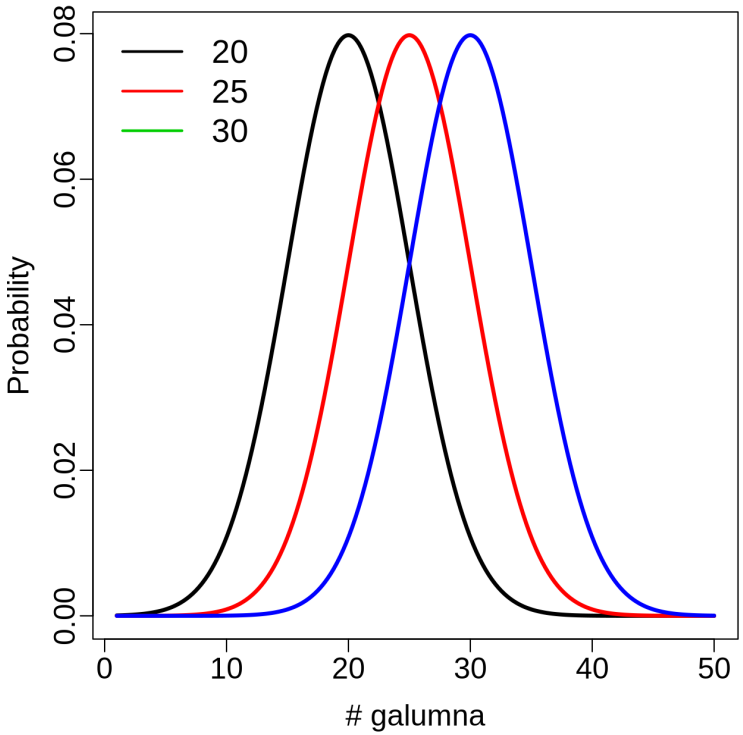
ε_i = model residuals drawn from a normal distribution with a varying mean but a constant variance**

> ***Key point!***
>
> *Residuals (the distance between each observation and the regression line) can be predicted by drawing random values from a normal distribution.*

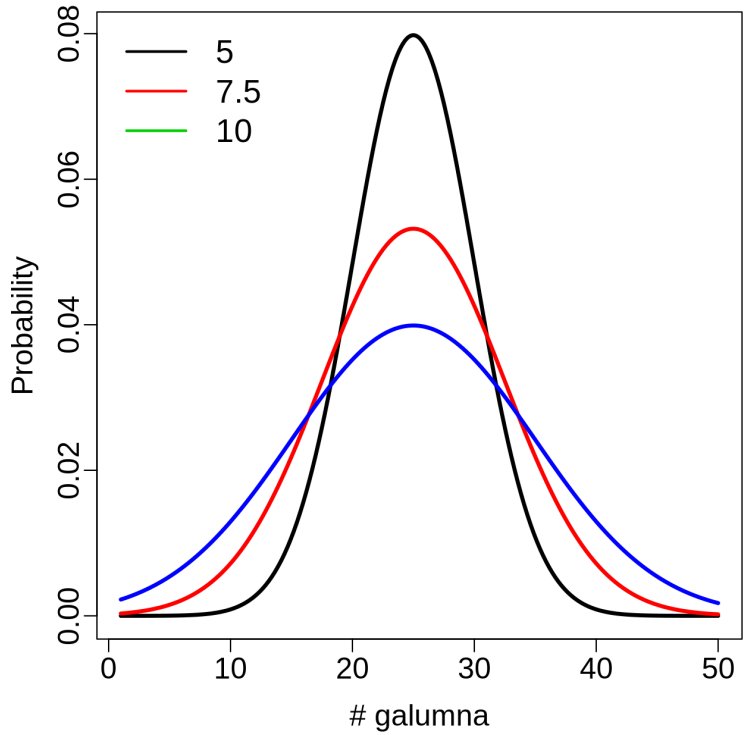
Normally Distributed Residuals

Recall: Normal distributions have two parameters: μ (mean) and σ (variance).

Varying μ, $\sigma = 5$



$\mu = 25$, varying σ



Normally Distributed Residuals

Another way to write the linear model equation is:

$$y_i \sim N(\mu = \beta_0 + \beta_1 X_i, \sigma^2)$$

Which literally means that y_i is drawn from a normal distribution with parameters μ (which depends on x_i) and σ (which has the same value for all Y s).

Let us predict Galumna abundances as a function of water content using the linear model we fitted earlier...

Model prediction

We need regression coefficients (β) and the standard deviation (σ):

```
coef(lm.abund)
#  (Intercept)     WatrCont
#  3.439348672 -0.006044788
summary(lm.abund)$sigma
# [1] 1.513531
```

What are the parameters of the normal distribution used to model y when water content = 300?

$$y_i \sim N(\mu = \beta_0 + \beta_1 X_i, \sigma^2)$$

$$\mu = 3.44 + (-0.006x300) = 1.63$$

$$\sigma = 1.51$$

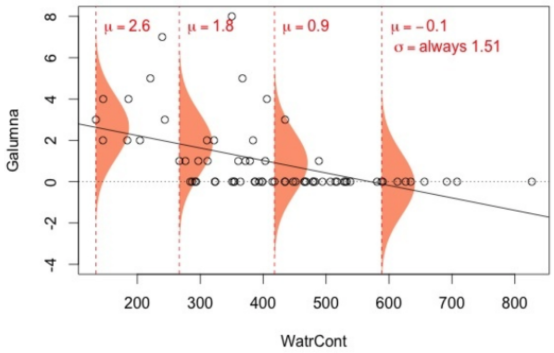
Model prediction

$$y_i \sim N(\mu = \beta_0 + \beta_1 X_i, \sigma^2)$$

$$\mu = 3.44 + (-0.006x300) = 1.63$$

$$\sigma = 1.51$$

- Graphically, this is our model:



At $x = 300$, residuals should follow a normal distribution with $\mu = 1.63$ and $\sigma^2 = 1.51$.

At $x = 400$, we get $\mu = 1.02$ and $\sigma^2 = 1.51$, etc.

Problems:

- σ^2 is not homogeneous, yet `lm()` forces a constant σ^2

Biological data & distributions

Statisticians have described a multitude of distributions that correspond to different types of data.

A distribution provides the probability of observing each possible outcome of an experiment or survey (e.g. $abund = 8$ *Galumna* individuals).

Distributions can be **discrete** (only includes integers or **continuous** (includes fractions).

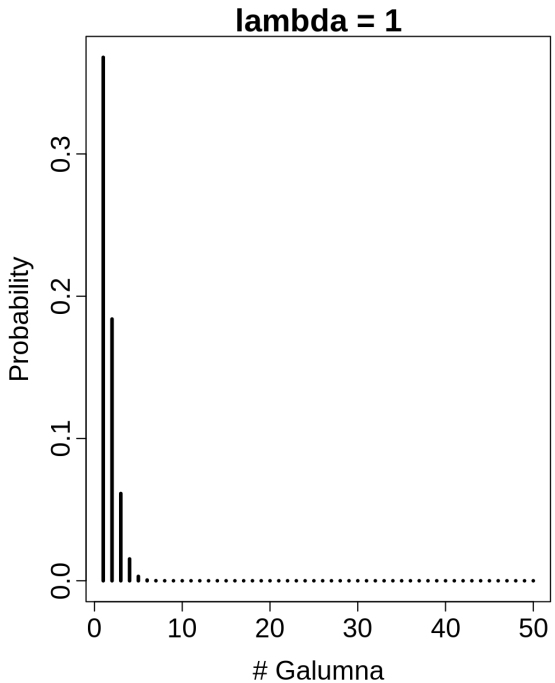
All distributions have **parameters** that dictate the shape of the distribution (e.g. μ and σ^2 for the normal).

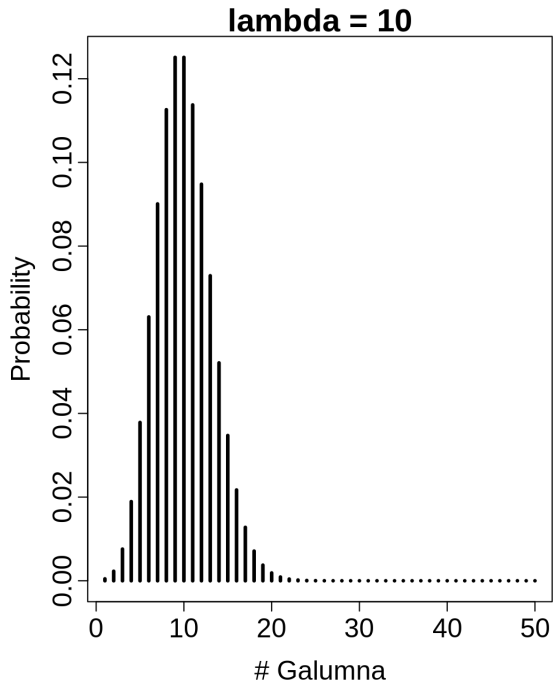
Biological data & distributions

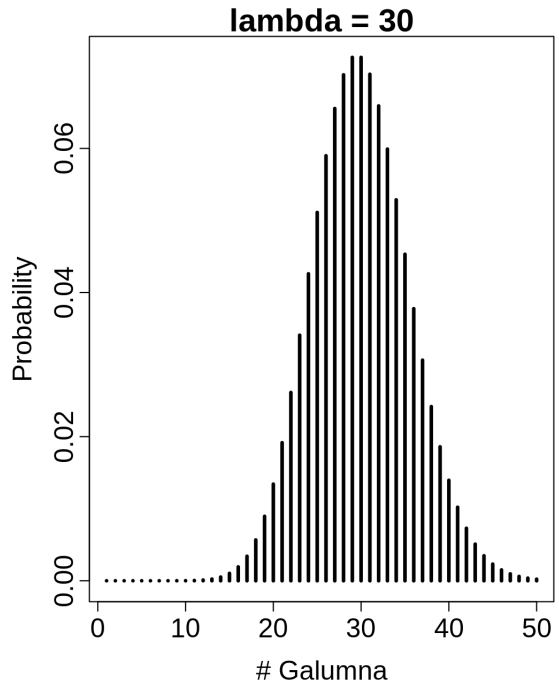
Galumna abund follows a discrete distribution (i.e. it only takes integer values).

A useful distribution to model abundance data is the "Poisson" distribution:

- **Poisson** is a discrete distribution with a single parameter, λ (lambda), which defines both the mean and the variance of the distribution:



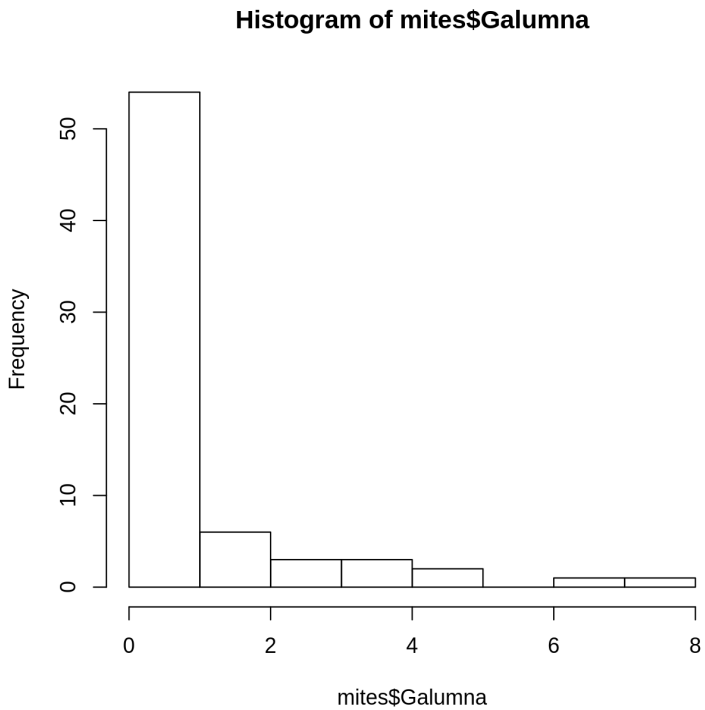




Biological data & distributions

Galumna seems to follow a Poisson distribution with a low value of λ:

```
hist(mites$Galumna)
```



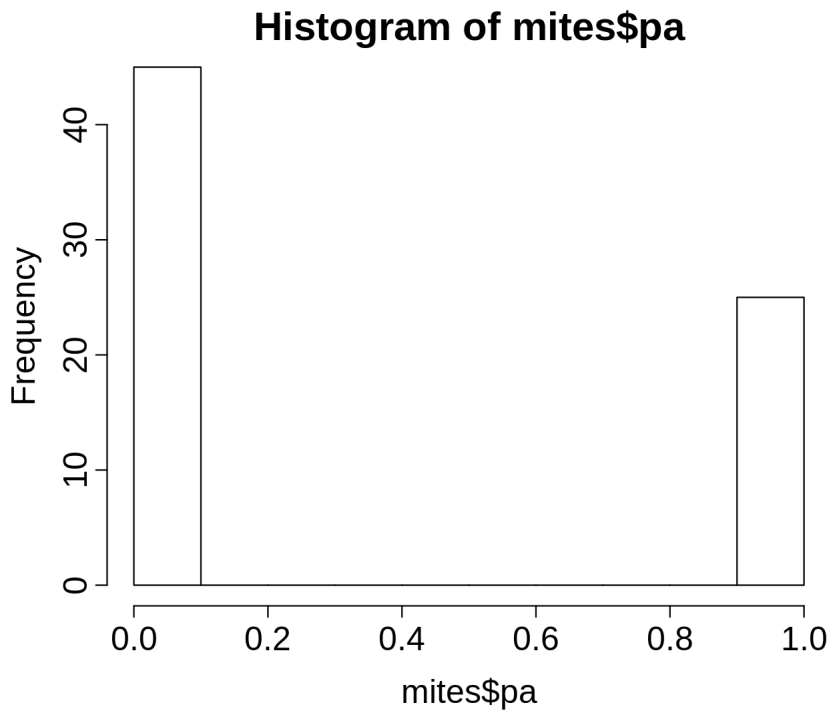

```
mean(mites$Galumna)
# [1] 0.9571429
```

Biological data & distributions

Presence-absence data takes yet another form:

- Only `0`s and `1`s;
- Poisson distribution would not be appropriate to model this variable.

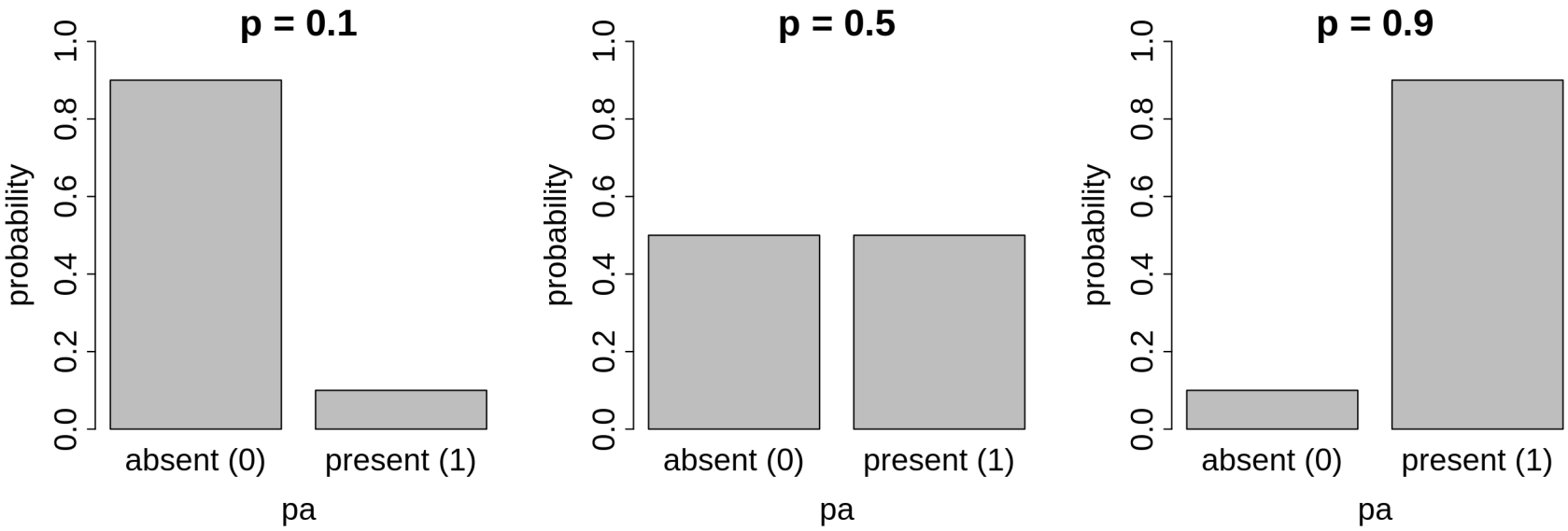
```
hist(mites$pa)
```



Biological data & distributions

“Bernoulli” distribution:

- Only two possible outcomes in its range: success (`1`) or failure (`0`);
- One parameter, p, the probability of success.

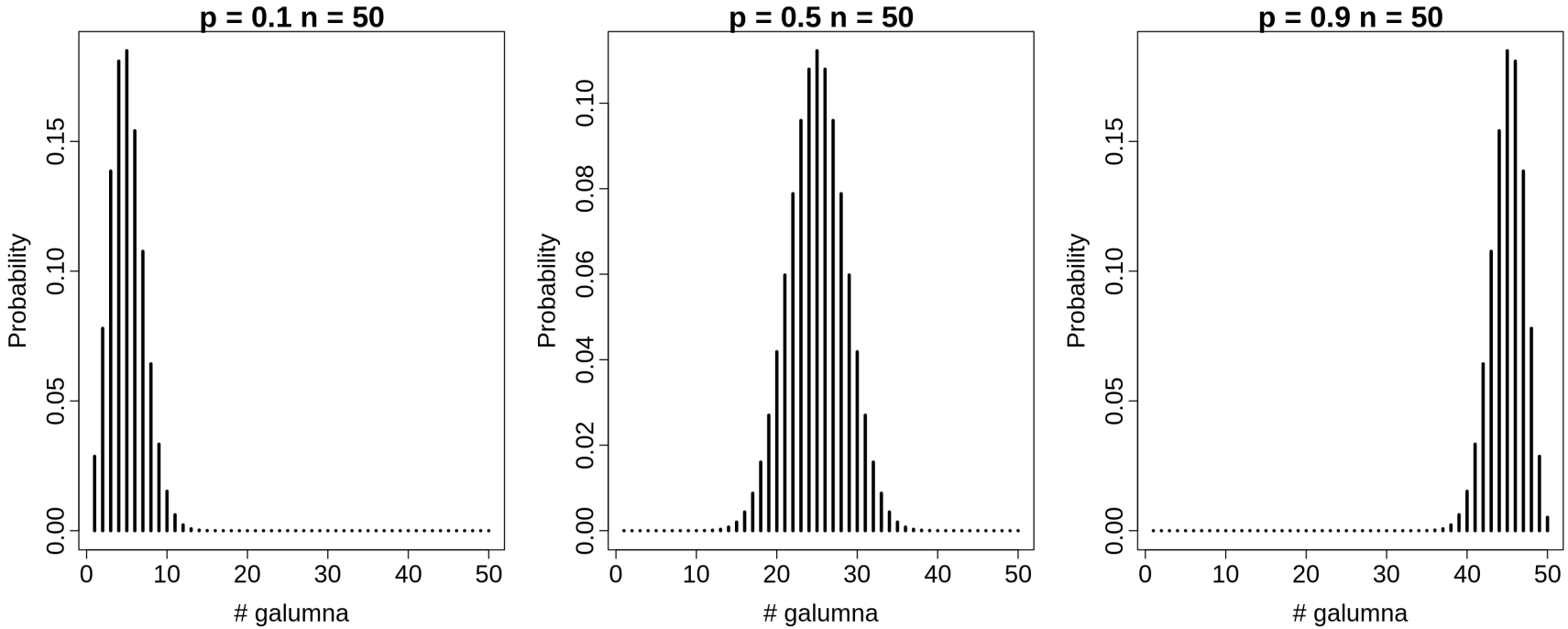


We can use the Bernouilli distribution to calculate the probability Galumna present (`1`) vs. absent (`0`)

Biological data & distributions

Binomial distribution: When there are multiple trials (each with a success/failure), the Bernoulli distribution expands into the binomial distribution.

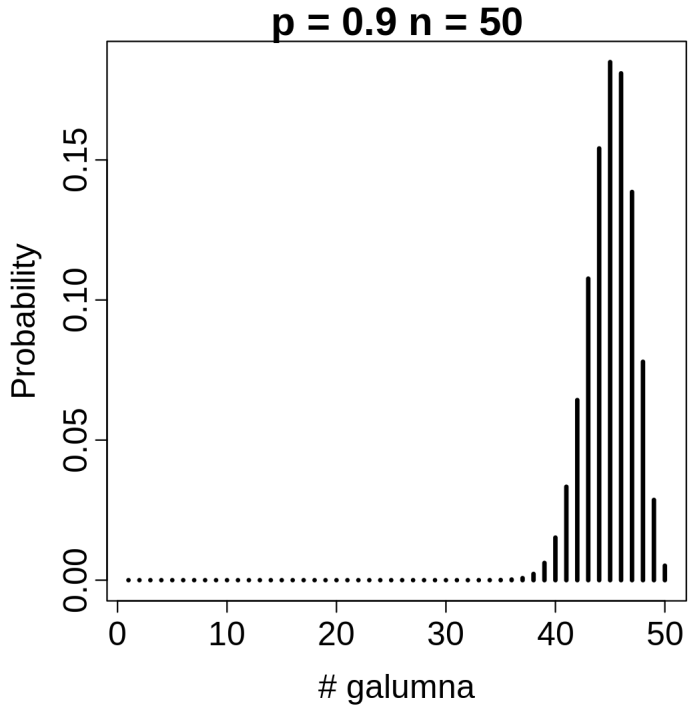
- Additional parameter, n, for number of trials;
- Predicts the probability of observing a given proportion of successes, p, out of a known total number of trials, n.

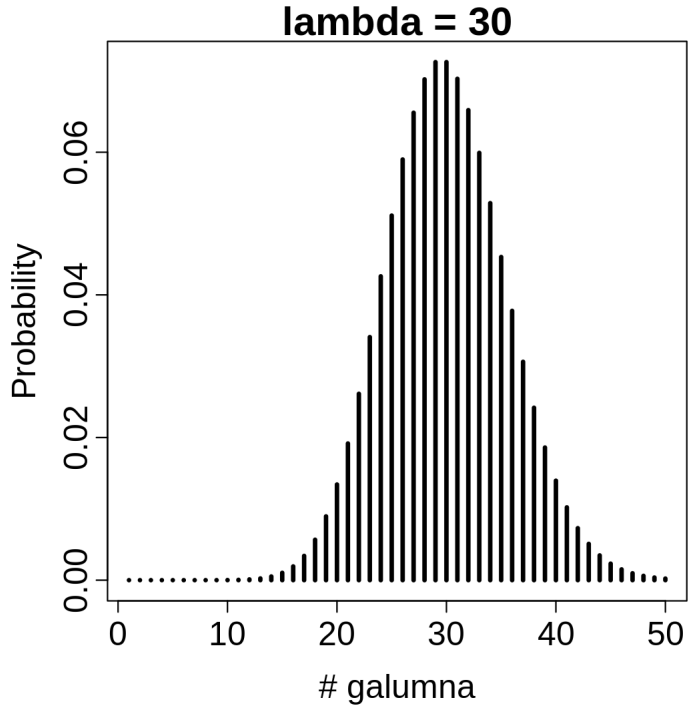


Biological data & distributions

Binomial distribution: used to model data where the number of successes are integers and where the number of trials, n, is known.

Main difference with Poisson distribution: the binomial has an upper limit to its range, corresponding to n. Consequently, it is right-skewed at low p values but left-skewed at high p values





Biological data & distributions

Getting back to our problem... We can switch the distribution of error terms (ε_i) from normal to Poisson:

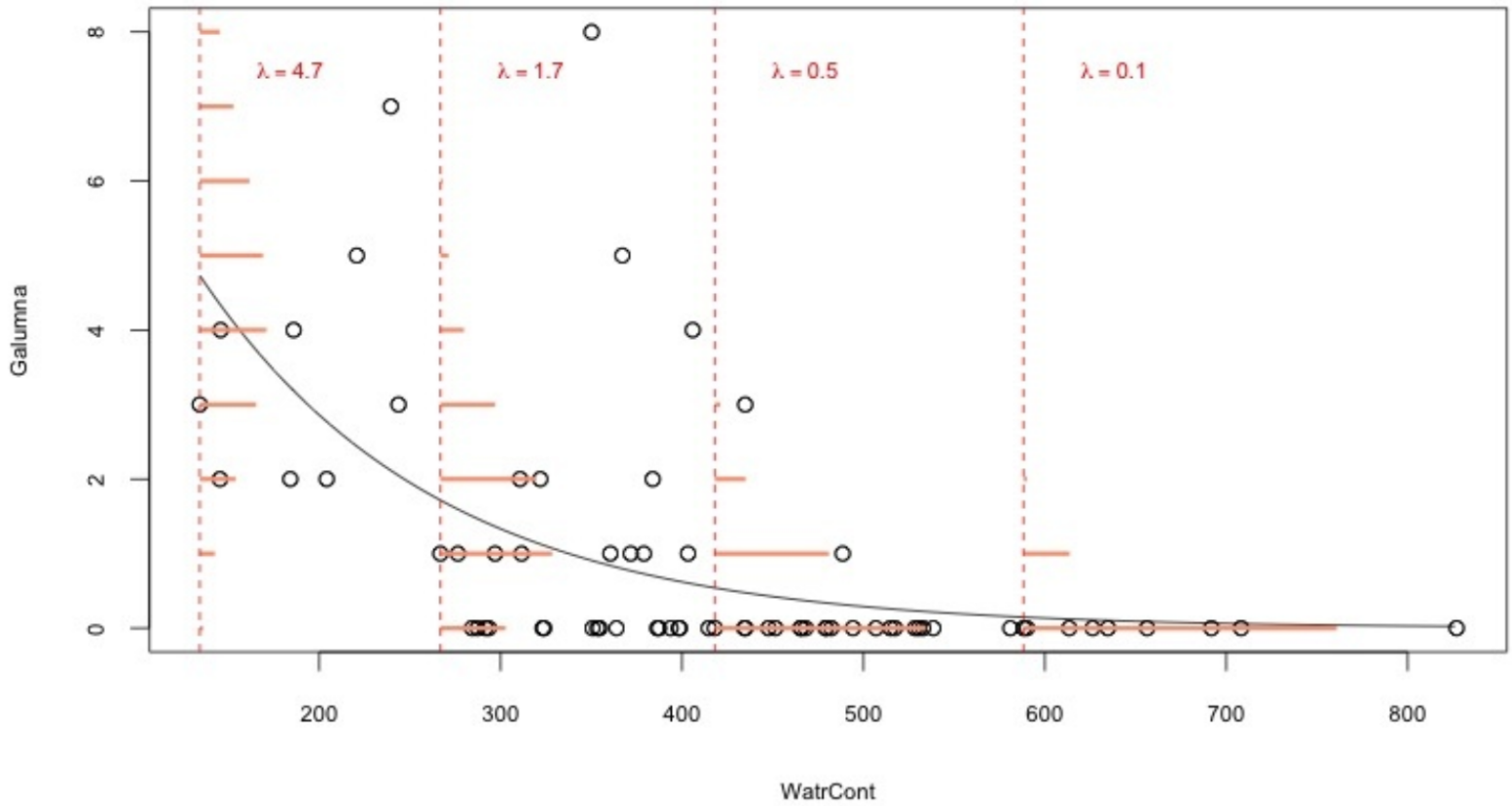
$$y_i \sim Poisson(\lambda = \beta_0 + \beta_1 x_i)$$

Problems solved!

1. λ varies with x (water content) which means that the residual variance will also vary with x. This also means that we have relaxed the homogeneity of variance assumption!
2. Predicted values will now be integers instead of fractions;
3. The model will never predict negative values (Poisson is strictly positive).

Biological data & distributions

This is **almost** a Poisson GLM, which looks like this:



Probabilities (in orange) are now integers, and both the variance and the mean of the distribution decline as λ decreases with increasing water content.

GLM with binary data

Binary variables

A common response variable in ecological datasets is the **binary variable**: we observe a phenomenon X or its "absence".

- Presence or absence of a species
- Presence or absence of a disease
- Success or failure to observe behaviour
- Survival or death of organisms

We wish to determine how $P/A \sim Environment$, for instance.

We often call this situation as a logistic regression or logit model.

Binary variables

In `R`, binary variables are coded with `1` and `0`:

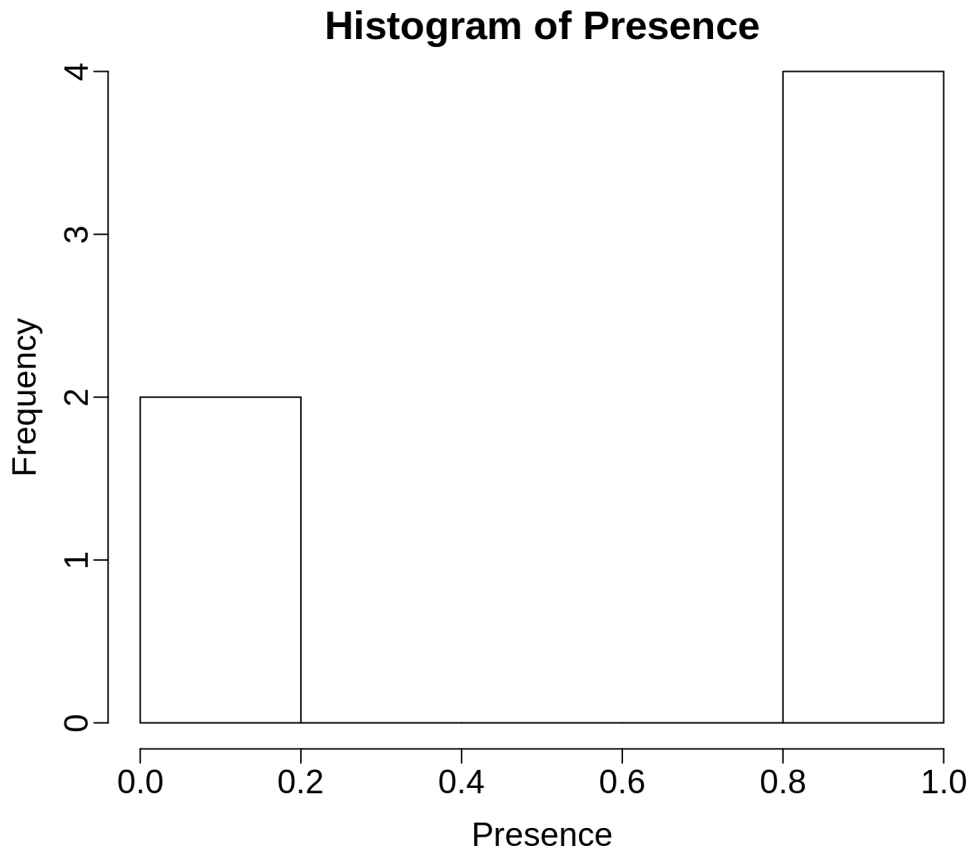
```
#   Site Presence
# 1    A        1
# 2    B        0
# 3    C        1
# 4    D        1
# 5    E        0
# 6    F        1
```

1 = Presence

0 = Absence

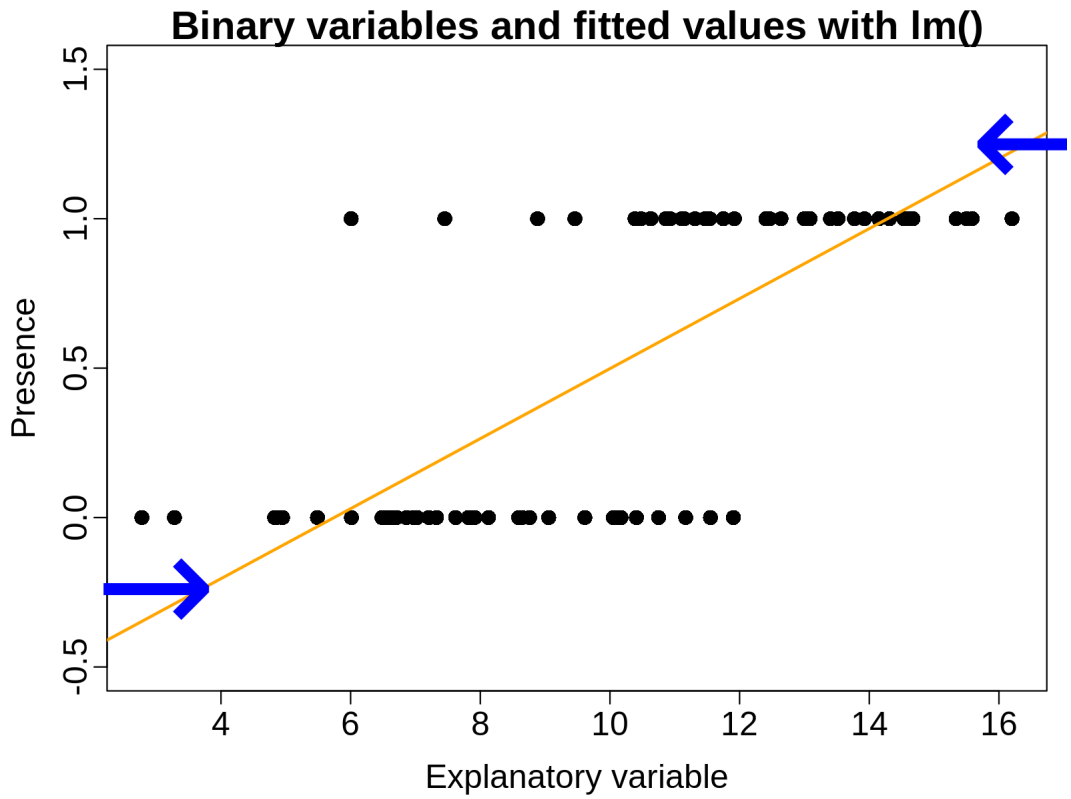
Binary variables

They are clearly not normally distributed!



Binary variables

Under a linear model, expected values can be out of the `[0,1]` range with `lm()`:



Probability distribution

The Bernoulli distribution is well suited for binary response variables.

$$E(Y) = p$$

➔ **Mean of distribution** Probability `(p)` of observing an outcome

$$Var(Y) = p \times (1 - p)$$

➔ **Variance of distribution** Variance decreases as `(p)` is close to `0` or `1`

Logistic regression

The `glm()` function!

```
logit.reg ← glm(formula, data, family)
```

Moving away from traditional linear models (`lm()`)...

In addition to the data (of course!), we need to specify additional elements in a generalized linear model (`glm()`):

1. The error distribution

AND

1. the link function to be used in the model.

In `glm()`, this is set within the `family` argument.

The Link Function

For a simple linear model of a normally distributed continuous response variable, the equation for the expected values is:

$$\mu = x\beta$$

where

- μ is the expected value of the response variable;
- x is the model matrix (*i.e.* your data);
- β is the vector of estimated parameters (*i.e.* the intercept & slope).

$x\beta$ is called the **linear predictor**.

The Link Function

$\mu = x\beta$ is only true for normally distributed data.

If this is not the case, we must use a transformation on the expected values μ.

$$g(\mu) = x\beta$$

where $g(\mu)$ is the link function.

This allows us to relax the normality assumption.

The Link Function

For binary data, the link function is called the **logit**:

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

$\mu =$ expected values (probability that $Y = 1$).

The *logit* function is the logarithm of the odds ($\frac{p}{1-p}$).

The odds puts our expected values on a `0` to `+Inf` scale.

The log-transformation puts our expected values on a `-Inf` to `+Inf`.

➔ The expected values can now be **linearly** related to the linear predictor.

Exercise 1 - Our first generalized linear model, together.

Build a logistic regression model using the `mites` data

```
# setwd('...')

mites ← read.csv("data/mites.csv", header = TRUE)
str(mites)
# 'data.frame':    70 obs. of  9 variables:
#  $ Galumna   : int  8 3 1 1 2 1 1 1 2 5 ...
#  $ pa        : int  1 1 1 1 1 1 1 1 1 1 ...
#  $ totalabund: int  140 268 186 286 199 209 162 126 123 166 ...
#  $ prop      : num  0.05714 0.01119 0.00538 0.0035 0.01005 ...
#  $ SubsDens  : num  39.2 55 46.1 48.2 23.6 ...
#  $ WatrCont  : num  350 435 372 360 204 ...
#  $ Substrate : Factor w/ 7 levels "Barepeat","Interface",..: 4 3 2 4 4
#  $ Shrub     : Factor w/ 3 levels "Few","Many","None": 1 1 1 1 1 1 1 2
#  $ Topo      : Factor w/ 2 levels "Blanket","Hummock": 2 2 2 2 2 2 2 1
```

Exercise 1 - Our first generalized linear model, together.

Let us build a model of the presence of *Galumna* sp. as a function of water content and topography.

```
logit.reg ← glm(pa ~ WatrCont + Topo, data=mites,
                family = binomial(link = "logit"))
```

```
summary(logit.reg)
```

Exercise 1

```
summary(logit.reg)
#
# Call:
# glm(formula = pa ~ WatrCont + Topo, family = binomial(link = "logit"),
#     data = mites)
#
# Deviance Residuals:
#     Min       1Q   Median       3Q      Max
# -2.0387  -0.5589  -0.1594   0.4112   2.0252
#
# Coefficients:
#             Estimate Std. Error z value Pr(>|z|)
# (Intercept)  4.464402   1.670622   2.672 0.007533 **
# WatrCont    -0.015813   0.004535  -3.487 0.000489 ***
# TopoHummock  2.090757   0.735348   2.843 0.004466 **
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# (Dispersion parameter for binomial family taken to be 1)
#
```

Challenge 1

Using the `bacteria` dataset, model the presence of *H. influenzae* as a function of treatment and week of test.

Start with a full model and reduce it to the most parsimonious model.

```
# install.packages("MASS")

library(MASS)
data(bacteria)
str(bacteria)
# 'data.frame':    220 obs. of  6 variables:
#  $ y   : Factor w/ 2 levels "n","y": 2 2 2 2 2 2 1 2 2 2 ...
#  $ ap  : Factor w/ 2 levels "a","p": 2 2 2 2 1 1 1 1 1 1 ...
#  $ hilo: Factor w/ 2 levels "hi","lo": 1 1 1 1 1 1 1 1 2 2 ...
#  $ week: int  0 2 4 11 0 2 6 11 0 2 ...
#  $ ID  : Factor w/ 50 levels "X01","X02","X03",..: 1 1 1 1 2 2 2 2 3 3
#  $ trt : Factor w/ 3 levels "placebo","drug",..: 1 1 1 1 3 3 3 3 2 2 ..
```

Tests of the presence of the bacteria H. influenzae in children with otitis media in the Northern Territory of Australia.

Exercise 1 - Solution

```r
model.bact1 ← glm(y ~ trt * week, data = bacteria, family = binomial)
```

```r
model.bact2 ← glm(y ~ trt + week, data = bacteria, family = binomial)
```

```r
model.bact3 ← glm(y ~ week, data = bacteria, family = binomial)
```

```r
anova(model.bact1, model.bact2, model.bact3, test = "LRT")
# Analysis of Deviance Table
#
# Model 1: y ~ trt * week
# Model 2: y ~ trt + week
# Model 3: y ~ week
#   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
# 1       214     203.12
# 2       216     203.81 -2  -0.6854  0.70984
# 3       218     210.91 -2  -7.1026  0.02869 *
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Interpreting the `glm()` coefficients

Let's go back to the summary of our `logit.reg` model to see the coefficients:

```
logit.reg ← glm(pa ~ WatrCont + Topo, data=mites,
                family = binomial(link = "logit"))
```

```
summary(logit.reg)$coefficients
#                Estimate  Std. Error   z value     Pr(>|z|)
# (Intercept)  4.46440199 1.670622482  2.672299 0.0075333598
# WatrCont    -0.01581255 0.004535069 -3.486728 0.0004889684
# TopoHummock  2.09075654 0.735348234  2.843220 0.0044660283
```

The output indicates that both water content and topography are significant drivers of the presence-absence variable.

But, how do we interpret the slope coefficients?

Interpreting the `glm()` coefficients

The direct interpretation of the coefficients in the logit model is tenuous because of the link function. If the link is *identity*, it is much easier to interpret.

Let us assume that we have a binary outcome y and two covariates x_1 and x_2 (and a constant). The probability of a successful outcome ($y = 1$) is given by:

$$Pr(y_i = 1) = p = g^{-1(\beta_0 + x_{1i}\beta_1 + x_{2i}\beta_2)}$$

where $g^{-1}()$ is the **inverse link function**.

Now let us focus in interpreting the β_1 coefficient.

Identity link

For the identity link, the interpretation is straighforward. For one-unit increase in x_1, β_1 dictates a constant difference in the outcome.

$\Delta y_i = [\beta_0 + (x_{1i} + 1)\beta_1 + x_{2i}\beta_2] - (\beta_0 + x_{1i}\beta_1 + x_{2i}\beta_2)$

$\Delta y_i = \beta_1$

Interpreting the `glm()` coefficients

Logit link

If a linear logistic model has been used with the two covariates x_1 and x_2, we have the model:

$$log(\frac{p}{1-p}) = \beta_0 + x_{1i}\beta_1 + x_{2i}\beta_2$$

for a log odds of a positive response. We may also write that model as follows:

$$\frac{p}{1-p} = exp(\beta_0 + x_{1i}\beta_1 + x_{2i}\beta_2)$$

$$Pr(yi) = \frac{exp(\beta_0 + x_{1i}\beta_1 + x_{2i}\beta_2)}{1 + exp(\beta_0 + x_{1i}\beta_1 + x_{2i}\beta_2)}$$

Great! Do not give up!

Interpreting the `glm()` coefficients

Logit link

Since the inverse link is nonlinear, the coefficient interpretation is difficult.

Let us take a look what happens to the differences for a one-unit change to x_1:

$$\Delta y_i = \frac{\exp[\beta_0 + (x_{1i} + 1)\beta_1 + x_{2i}\beta_2]}{1 + \exp\left[\beta_0 + (x_{1i} + 1)\beta_1 + x_{2i}\beta_2\right]} - \frac{\exp[\beta_0 + x_{1i}\beta_1 + x_{2i}\beta_2]}{1 + \exp\left[\beta_0 + x_{1i}\beta_1 + x_{2i}\beta_2\right]}$$

$$\Delta y_i = \exp\left(\beta_0\right)\exp\left(\beta_1\right)$$

As x increases by one unit, the odds increase by a factor of $\exp(\beta_1)$.

Interpreting the output

Getting back to our `logit.reg` model!

```
logit.reg
#
# Call:  glm(formula = pa ~ WatrCont + Topo, family = binomial(link = "lo
#     data = mites)
#
# Coefficients:
# (Intercept)     WatrCont  TopoHummock
#     4.46440     -0.01581      2.09076
#
# Degrees of Freedom: 69 Total (i.e. Null);  67 Residual
# Null Deviance:        91.25
# Residual Deviance: 48.76     AIC: 54.76
```

For a one-unit increase (or decrease) in water content, we can obtain the odds for the presence of mites.

```
exp(logit.reg$coefficient[2])
#  WatrCont
# 0.9843118
```

Predictive power and goodness-of-fit

Recall that the R^2 value is a measure of how well the model explains the data.

In simple linear models, R^2 can be obtained as the square of the sample correlation equation coefficient (r) between the observed outcomes and the predictive values.

In some of the generalized linear models, we take different approaches.

We can obtain a pseudo-R^2, the analogue of the R^2 for models fitted by maximum likelihood.

We can calculate McFadden's (1973) pseudo-R^2 in this way:

$$\text{pseudo-R}^2 = \frac{\text{null deviance - residual deviance}}{\text{null deviance}}$$

> pseudo-R^2 is the variance explained by the model

Unit deviance, total deviance and null deviance

The unit deviance is a measure of distance between y and μ.

$$d(y, y) = 0$$

$$d(y, \mu) > 0 \quad \forall y \neq \mu$$

The total deviance $D(\mathbf{y}, \hat{\boldsymbol{\mu}})$ of a model with predictions $\hat{\boldsymbol{\mu}}$ of the observation $\mathbf{y}$ is the sum of its unit deviances:

$$D(\mathbf{y}, \hat{\boldsymbol{\mu}}) = \sum_i d(y_i, \hat{\mu}_i)$$

Now, the deviance of a model that has estimates $\hat{\mu} = E[Y|\hat{\theta}_0]$ can be defined by its **likelihood**:

$$D(y, \hat{\mu}) = 2\Big(\log\big(p(y \mid \hat{\theta}_s)\big) - \log\big(p(y \mid \hat{\theta}_0)\big) \Big)$$

with $\hat{\theta}_0$ denoting the fitted values of the parameters in the reduced model, while $\hat{\theta}_s \hat{\theta}_s$ denotes the fitted parameters for the saturated model.

Unit deviance, total deviance and null deviance

Now, the deviance of a model that has estimates $\hat{\mu} = E[Y|\hat{\theta}_0]$ can be defined by its **likelihood**:

$$D(y, \hat{\mu}) = 2\Big(\log\big(p(y \mid \hat{\theta}_s)\big) - \log\big(p(y \mid \hat{\theta}_0)\big)\Big)$$

with $\hat{\theta}_0$ denoting the fitted values of the parameters in the reduced model, while $\hat{\theta}_s \hat{\theta}_s$ denotes the fitted parameters for the saturated model.

The **residual deviance** is defined as 2 times the log-likelihood ratio of the full model compared to the reduced model. (The function below is exactly the same as above!)

$$D(y, \hat{\mu}) = 2\Big(\log\big(p(\textbf{saturated model})\big) - \log\big(p(\textbf{reduced model})\big)\Big)$$

And, the **null deviance** is defined 2 times the log-likelihood ratio of the full model compared to the null model (i.e. predictors are set to 1).

$$D(y, \hat{\mu}) = 2\Big(\log\big(p(\textbf{saturated model})\big) - \log\big(p(\textbf{null model})\big)\Big)$$

Total deviance and null deviance

In R, we can do this as follows:

Let us compare the deviance of your model (residual deviance) to the deviance of a null model (null deviance).

The **null deviance model** here is a model without explanatory variables.

```
null.model ← glm(response.variable ~ 1, family = binomial)
```

The **saturated (or full) deviance model** here is a model with all explanatory variables.

```
full.model ← glm(response.variable ~ ., family = binomial)
```

Predictive power and goodness-of-fit

In `R`, we can extract the residual and null deviances directly from the `glm object`:

```
objects(logit.reg)
#  [1] "aic"              "boundary"          "call"
#  [4] "coefficients"     "contrasts"         "control"
#  [7] "converged"        "data"              "deviance"
# [10] "df.null"          "df.residual"       "effects"
# [13] "family"           "fitted.values"     "formula"
# [16] "iter"             "linear.predictors" "method"
# [19] "model"            "null.deviance"     "offset"
# [22] "prior.weights"    "qr"                "R"
# [25] "rank"             "residuals"         "terms"
# [28] "weights"          "xlevels"           "y"
```

```
pseudoR2 ← (logit.reg$null.deviance - logit.reg$deviance) / logit.reg$nu

pseudoR2
# [1] 0.4655937
```

Predictive power and goodness-of-fit

An adjusted McFadden's pseudo-R2, which penalizes for the number of predictors, can be calculated as below:

$$R^2_{adj} = 1 - \frac{logL(M) - K}{logL(M_{null})}$$

where K corresponds to the additional number of predictors in relation to the null model.

The goodness-of-fit of the logistic regression model can be expressed by some variants of pseudo-R2 statistics, such as Maddala (1983) or Cragg and Uhler (1970) measures.

When talking about *logistic regressions*, low R^2 values are often common.

One way to assess the goodness-of-fit of these models is to use the **Hosmer-Lemeshow goodness-of-fit test**.

It roughly involves:

1. Dividing the data into groups of similar size;
2. Ordering on the predicted probability (or equivalently, the linear predictor);
3. Comparing the observed to expected number of positive responses in each group

Exercise 2 - Calculating the pseudo-R2

The `DescTools::PseudoR2()` function computes several pseudo-R2.

```
logit.reg ← glm(mites$pa ~ mites$WatrCont + mites$Topo,
                family = binomial(link = "logit"))

library(DescTools)

fit ← PseudoR2(logit.reg, which = "all")
fit
#        McFadden     McFaddenAdj        CoxSnell      Nagelkerke   Aldri
#       0.4655937       0.3998373       0.4549662       0.6245898       0
# VeallZimmermann           Efron McKelveyZavoina            Tjur
#       0.6674318       0.5024101       0.7064093       0.5114661      54
#             BIC          logLik          logLik0              G2
#      61.5078819     -24.3811981     -45.6229593      42.4835224
```

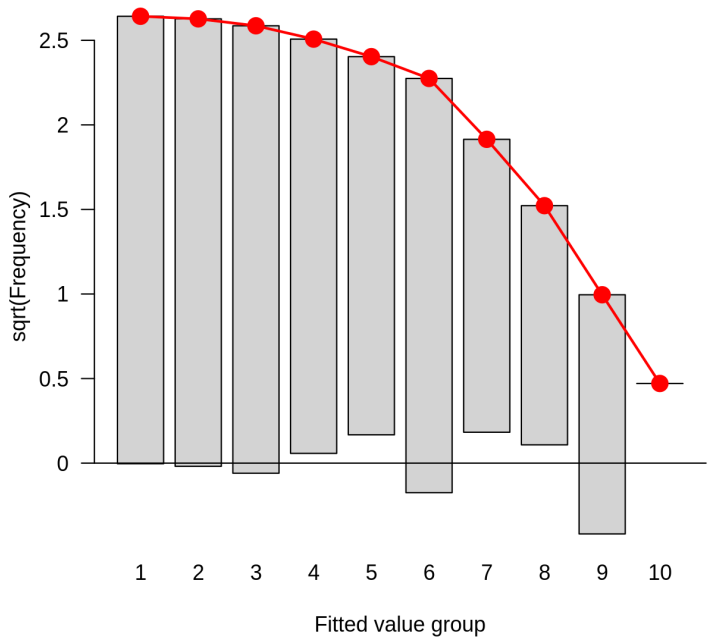
Exercise 2: Performing a Hosmer-Lemeshow goodness-of-fit test

This test is assessing whether or not the observed event rates match expected event rates in subgroups of the model population.

```
library(vcdExtra)

HLtest(logit.reg)
# Hosmer and Lemeshow Goodness-of
#
# Call:
# glm(formula = mites$pa ~ mites$
#  ChiSquare df   P_value
#   3.421693  8 0.9051814
```

```
plot(HLtest(logit.reg))
```



A non-significant value indicates an adequate fit!

Challenge 1

1. Using the model created with `bacteria` dataset, assess goodness-of-fit and predictive power.
2. Think how predictive power could be improved for this model.

Solution

1. Using the model created with `bacteria` dataset, assess goodness-of-fit and predictive power.

```
null.d ← model.bact2$null.deviance

resid.d ← model.bact2$deviance

bact.pseudoR2 ← (null.d - resid.d) / null.d

HLtest(model.bact2)
```

1. Think how predictive power could be improved for this model.

Adding informative explanatory variables could increase the explanatory power of the model.

But, do not be afraid of non-significant results!

Proportion data and GLM

Sometimes, proportion data are more similar to logistic regression than you think...

In discrete counts, we can, for instance, measure the number of presence of individuals in relation to the total number of populations sampled.

We will thus obtain a proportional number of "success" in observing individuals by dividing the counts by the total counts.

> In `glm()`, we have to provide *prior weights* if the response variable is the proportion of successes.

Exercise 3

In R, we have to specify the number of times something happened and the number of times something did not happen:

```
prop.reg ← glm(cbind(Galumna, totalabund - Galumna) ~ Topo + WatrCont,
               data = mites,
               family = binomial)
```

```
summary(prop.reg)
```

Exercise 3

```
summary(prop.reg)
#
# Call:
# glm(formula = cbind(Galumna, totalabund - Galumna) ~ Topo + WatrCont,
#     family = binomial, data = mites)
#
# Deviance Residuals:
#     Min       1Q   Median       3Q      Max
# -1.4808  -0.9699  -0.6327  -0.1798   4.1688
#
# Coefficients:
#             Estimate Std. Error z value Pr(>|z|)
# (Intercept) -3.288925   0.422109  -7.792 6.61e-15 ***
# TopoHummock  0.578332   0.274928   2.104   0.0354 *
# WatrCont    -0.005886   0.001086  -5.420 5.97e-08 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# (Dispersion parameter for binomial family taken to be 1)
#
```

Exercise 3

We can also code the model directly with proportions:

```
prop.reg2 ← glm(prop ~ Topo + WatrCont,
                data = mites,
                family = binomial,
                weights = totalabund)
```

GLM with count data

Modeling count data

What is count data?

First, import the `faramea.csv` into `R`.

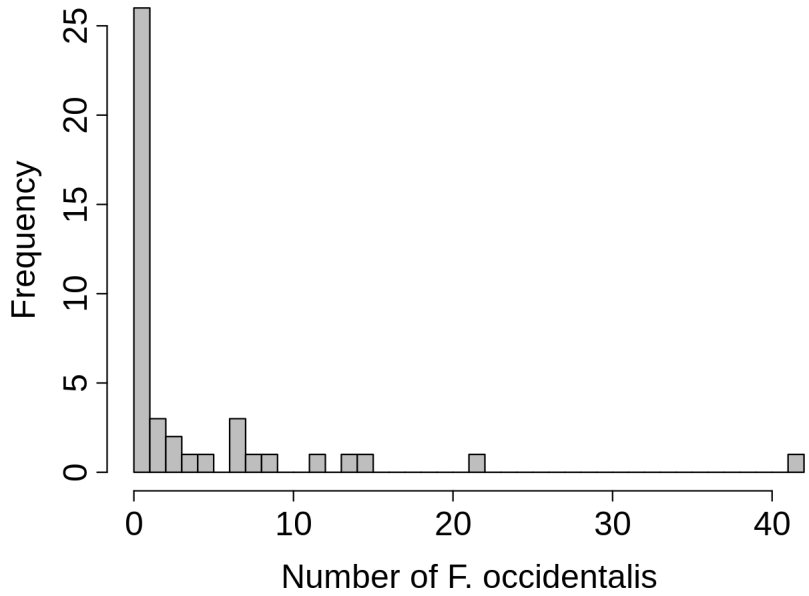
```
faramea ← read.csv('faramea.csv', header = TRUE)
```

The number of trees of the species *Faramea occidentalis* was assessed in 43 quadrats in Barro Colorado Island (Panama). For each quadrat, environmental characteristics (such as elevation and precipitation) were also recorded.

Let's investigate the histogram of the number of *Faramea occidentalis*.

Modeling count data

What is count data?



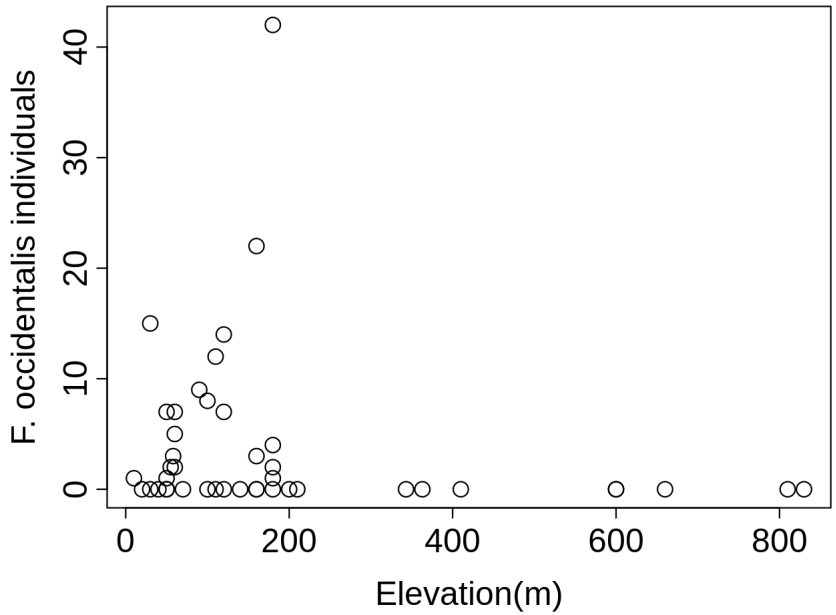
Count data are characterized by:

- Positive values: you do not count -7 individuals
- Integer values: you do not count 7.56 individuals
- Exhibits larger variance for large values

Modeling count data

How to model count data?

Does elevation influence the abundance of *F. occidentalis*?



The **Poisson distribution** seems to adequately fit this data distribution.

We can use the Poisson family in our GLM to begin modeling count data.

The Poisson distribution

The Poisson distribution specifies the probability of a discrete random variable Y and is given by:

$$f(y, \mu) = Pr(Y = y) = \frac{\mu^y \times e^{-\mu}}{y!}$$

$$E(Y) = Var(Y) = \mu$$

Properties:

- μ is the parameter of the Poisson distribution;
- specifies the probably only for integer values;
- probability for negative values is null ($P(Y < 0) = 0$);
- $Var(Y) = \mu$ (allows for heterogeneity).

Poisson GLM behind the scenes

A Poisson GLM will model the value of μ as a function of different explanatory variables.

Three steps

Step 1. We assume Y_i follows a Poisson distribution with mean and variance μ_i.

$$Y_i = Poisson(\mu_i)$$

$$E(Y_i) = Var(Y_i) = \mu_i$$

$$f(y_i, \mu_i) = \frac{\mu_i^{y_i} \times e^{-\mu_i}}{y!}$$

μ_i corresponds to the expected number of individuals

Poisson GLM behind the scenes

Step 2. We specify the systematic part of the model just as in a linear model.

$$\underbrace{\alpha}_{\text{Intercept}} + \underbrace{\beta_1}_{\text{slope of 'Elevation'}} \times \text{Elevation}_i + \underbrace{\beta_2}_{\text{slope of 'Precipitation'}} \times \text{Precipitation}_i$$

Step 3. The link between the mean of Y_i and the systematic part is a logarithmic function can be written as:

$$log(\mu_i) = \alpha + \beta_1 \times \text{Elevation}_i + \beta_2 \times \text{Precipitation}_i$$

or

$$\mu_i = e^{\alpha + \beta_1 \times \text{Elevation}_i + \beta_2 \times \text{Precipitation}_i}$$

or

$$\mu_i = e^{\alpha} \times e^{\beta_1^{\text{Elevation}_i}} \times e^{\beta_2^{\text{Precipitation}_i}}$$

This shows that the impact of each explanatory variable is multiplicative. Increasing Elevation by one increases μ by factor of exp($\beta_{\text{Elevation}}$).

If $\beta j = 0$ then $exp(\beta j) = 1$ and μ is not related to x_j. If $\beta j > 0$ then μ increases if x_j increases; if $\beta j < 0$ then μ decreases if x_j increases.

Fitting a Poisson GLM in R

The function `glm()` allows you to specify a Poisson GLM

```
glm.poisson = glm(Faramea.occidentalis ~ Elevation,
                  data = faramea, family = poisson)
```

The `family` argument specifies the distribution and link function.

As with `lm()` you can access the outputs of the model using the function `summary()`:

```
summary(glm.poisson)
```

Model summary

```
summary(glm.poisson)
#
# Call:
# glm(formula = Faramea.occidentalis ~ Elevation
#     data = faramea)
#
# Deviance Residuals:
#     Min       1Q   Median       3Q      Max
# -3.3319  -2.7509  -1.5451   0.1139  11.3995
#
# Coefficients:
#              Estimate Std. Error z value Pr(>
# (Intercept)  1.7687001  0.1099136  16.092  < 2
# Elevation   -0.0027375  0.0006436  -4.253 2.11
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.
#
# (Dispersion parameter for poisson family taken
#
#     Null deviance: 414.81  on 42  degrees of f
```

Estimates:

Intercept = α

Elevation = β

What about the *null deviance* and the *residual deviance*?

Parameter estimates

In our model the unknown parameters are the intercept (α) and the slope of elevation (β)

$$log(\mu_i) = 1.769 - 0.0027 \times \text{Elevation}_i$$

or

$$\mu_i = e^{1.769 - 0.0027 \times \text{Elevation}_i}$$

The deviance

Remember that to estimate the unknown parameter maximum likelihood estimation is used.

The residual deviance is defined as twice the difference between the log-likelihood of a model that provides a perfect fit and the log-likelihood of our model.

$$\text{residual deviance} = 2\,log(L(y;\,y)) - 2\,log(L(y;\,\mu))$$

In a Poisson GLM, the residual deviance should be close to the residual degrees of freedoms.

388.12 >> 41

> Our residual deviance is much higher than the degrees of freedom of our model!

Overdispersion

For a Poisson distribution, $var[y] = \mu$. However, in practice the apparent variance of the data often exceeds μ, reflecting *overdispersion* in the model parameters.

Overdispersion arises because the mean μ innately varies, even when all the explanatory variables are fixed, or because the events that are being counted are positively correlated.

Are you able to think of a situation in biology that can cause overdispersion?

Why should we care about overdispersion?

> Tests on the explanatory variables will generally appear to be more significant and confidence intervals for the parameters will be narrower than warranted by the data!

Overdispersion

When the residual deviance is higher than the residual degrees of freedom we say that the model is **overdispersed**. We can calculate a overdispersion parameter (ϕ):

$$\phi = \frac{\text{Residual deviance}}{\text{Residual degrees of freedom}}$$

Since the Poisson GLM assumes that $var[y] = \mu$, we will need to find an alternative.

Solutions

1: Correct for overdispersion using a **quasi-Poisson** generalized linear model.

2: Choose another distribution family, such as the **negative binomial**.

Quasi-Poisson GLM

The variance of the distribution will account for **overdispersion**:

$$E(Y_i) = \mu_i$$

$$Var(Y_I) = \phi \times \mu_i$$

where phi is the dispersion parameter.

The **systematic part** and the **link function** remain the same.

Correcting for overdispersion will not affect parameter estimates, but will affect their **significance**. Further, the standard errors of the parameters are multiplied by $\sqrt{\phi}$.

Some marginally significant p-values may no longer hold!

Fitting a quasi-Poisson GLM in R

Create a new `glm.object` using the '`quasipoisson`' family or update the previous one:

```
glm.quasipoisson = glm(Faramea.occidentalis ~ Elevation, data = faramea,
                       family=quasipoisson)

glm.quasipoisson = update(glm.poisson, family = quasipoisson)
```

Fitting a quasi-Poisson GLM in R

```
summary(glm.quasipoisson)
#
# Call:
# glm(formula = Faramea.occidentalis ~ Elevation
#     data = faramea)
#
# Deviance Residuals:
#     Min       1Q   Median       3Q      Max
# -3.3319  -2.7509  -1.5451   0.1139  11.3995
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|
# (Intercept)  1.768700   0.439233   4.027 0.000
# Elevation   -0.002738   0.002572  -1.064 0.293
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.
#
# (Dispersion parameter for quasipoisson family
#
#     Null deviance: 414.81  on 42  degrees of f
```

Same estimates but

The standard errors of the parameters are multiplied by

$$\sqrt{\phi} = 4$$

`0.0006436 * 4 = 0.00257`

<- ϕ

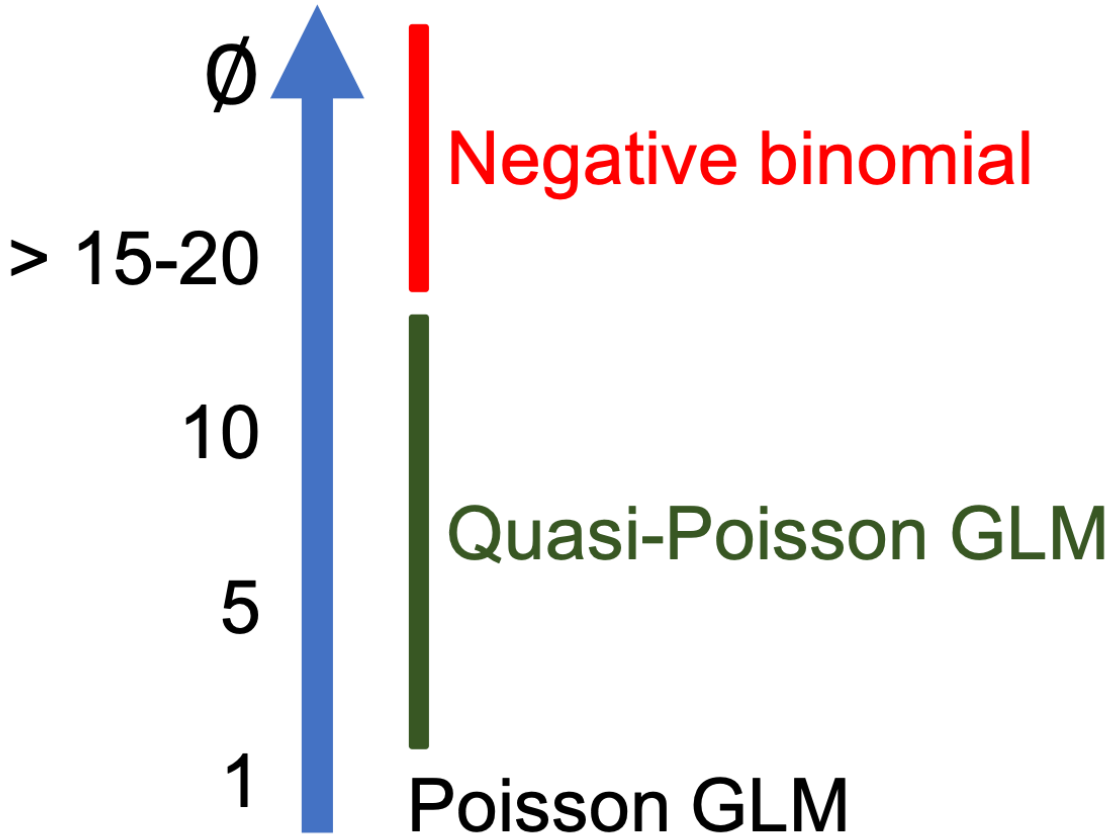
<- AIC is not defined

Fitting a quasi-Poisson GLM in R

Try also deviance analysis to test for the effect of Elevation

```
null.model ← glm(Faramea.occidentalis ~ 1, data = faramea,
                 family = quasipoisson)
anova(null.model, glm.quasipoisson, test = "Chisq")
# Analysis of Deviance Table
#
# Model 1: Faramea.occidentalis ~ 1
# Model 2: Faramea.occidentalis ~ Elevation
#   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
# 1        42     414.81
# 2        41     388.12  1   26.686   0.1961
```

Dispersion parameter



Negative binomial GLM

Negative binomial GLMs are favor when overdispersion is high

- It has **two parameters** μ and k. k controls for the dispersion parameter (smaller k indicates higher dispersion)
- It corresponds to a combination of two distributions (**Poisson** and **gamma**)
- It assumes that the Y_i are Poisson distributed with the mean μ assumed to follow a gamma distribution!

$$E(Y_i) = \mu_i$$

$$Var(Y_i) = \mu_i + \frac{\mu_i^2}{k}$$

Fitting a negative binomial in R

NB is not in the `glm()` function so you need to install and charge the `MASS` package

```
install.packages('MASS')
```

```
glm.negbin = glm.nb(Faramea.occidentalis ~ Elevation, data = faramea)
```

```
summary(glm.negbin)
```

```
#
# Call:
# glm.nb(formula = Faramea.occidentalis ~ Elevation, data = faramea,
#     init.theta = 0.2593107955, link = log)
#
# Deviance Residuals:
#      Min        1Q    Median        3Q       Max
# -1.36748  -1.17564  -0.51338  -0.05226   2.25716
#
# Coefficients:
#             Estimate Std. Error z value Pr(>|z|)
# (Intercept)  2.369226   0.473841    5.00 5.73e-07 ***
# Elevation   -0.007038   0.002496   -2.82  0.00481 **
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# (Dispersion parameter for Negative Binomial(0.2593) family taken to be 1
#
#     Null deviance: 41.974  on 42  degrees of freedom
# Residual deviance: 36.343  on 41  degrees of freedom
# AIC: 182.51
#
# Number of Fisher Scoring iterations: 1
#
```

theta $= k$

Plotting the final GLM model

Step 1 Plot the data and the use estimates of the parameters to draw model line

$$\mu_i = e^{2.369 - 0.007 \times Elevation_i}$$

Use `summary()` to get the parameters

```
summary(glm.negbin)$coefficients[1, 1]
summary(glm.negbin)$coefficients[2, 1]
```

Step 2 Use the standard errors to build the confidence envelope

```
summary(glm.negbin)$coefficients[1, 2]
summary(glm.negbin)$coefficients[2, 2]
```

$$\text{Upper limit} = e^{[\alpha - 1.96 \times SE_\alpha] + [\beta - 1.96 \times SE_\beta] \times \text{Elevation}_i}$$

$$\text{Upper limit} = e^{[\alpha + 1.96 \times SE_\alpha] + [\beta + 1.96 \times SE_\beta] \times \text{Elevation}_i}$$

```r
pp ← predict(glm.negbin, newdata = data.frame(Elevation = 1:800), se.fit
linkinv ← family(glm.negbin)$linkinv ## inverse-link function
pframe$pred0 ← pp$fit
pframe$pred ← linkinv(pp$fit)
sc ← abs(qnorm((1-0.95)/2))  ## Normal approx. to likelihood
pframe ← transform(pframe, lwr = linkinv(pred0-sc*pp$se.fit), upr = link

plot(faramea$Elevation, faramea$Faramea.occidentalis, ylab = 'Number of F
lines(pframe$pred, lwd = 2)
lines(pframe$upr, col = 2, lty = 3, lwd = 2)
lines(pframe$lwr, col = 2, lty = 3, lwd = 2)
```

Challenge 3

Use the `mites` dataset! Model the abundance of the species Galumna as a function of the substrate characteristics (water content `WatrCont` and density `SubsDens`)

- Do you need to account for overdispersion?
- Which covariates have a significant effect?
- Select the best model!

```
mites ← read.csv("data/mites.csv", header = TRUE)
```

Challenge 3: tips

Drop each term in turn and compare the full model with a nested model using the command:

```
drop1(MyGLM, test = "Chi")
```

Specify manually a nested model, call it for example MyGLM2, and use the command:

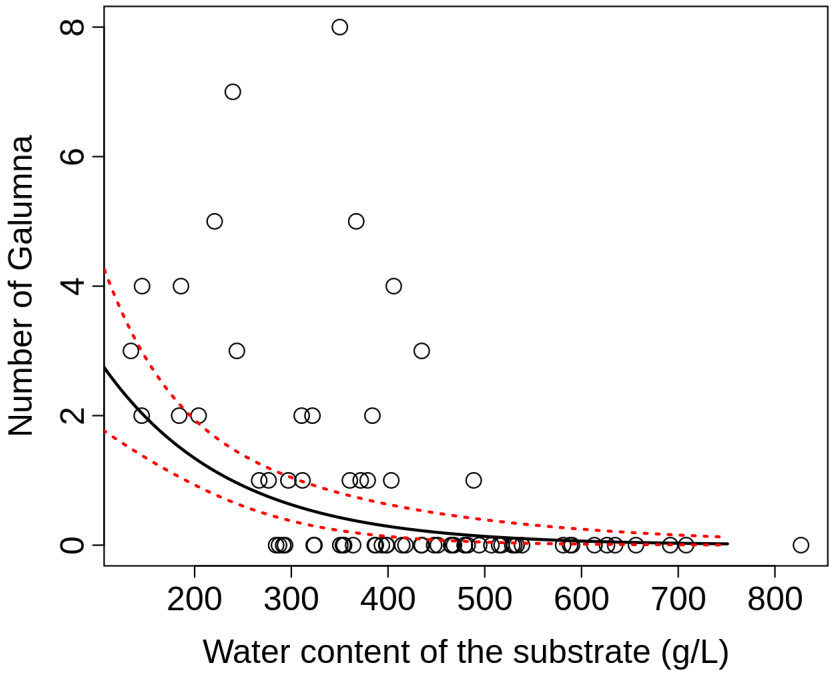
```
anova(MyGLM, MyGLM2, test = "Chi")
```

Challenge 3: solution

```r
# Poisson GLM
glm.p = glm(Galumna~WatrCont+SubsDens, data=mites, family=poisson)
# quasi-Poisson GLM
glm.qp = update(glm.p,family=quasipoisson)
# model selection
drop1(glm.qp, test = "Chi")
# Single term deletions
#
# Model:
# Galumna ~ WatrCont + SubsDens
#          Df Deviance scaled dev.  Pr(>Chi)
# <none>         101.49
# WatrCont  1   168.10      31.711 1.789e-08 ***
# SubsDens  1   108.05       3.125   0.07708 .
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# or
glm.qp2 = glm(Galumna~WatrCont, data=mites, family=quasipoisson)
anova(glm.qp2, glm.qp, test="Chisq")
```

Challenge 3: solution



Other distributions

- **Logit transformation of the data** often used with `lm(m)` for percentages or proportions when the binomial distribution is not appropriate. When not selections from fixed quantities (e.g. percent cover, school grades, etc).
- **Log-normal distribution in glm**, avoids having to log-transform the data.
- **Gamma distribution**. Similar to log-normal, more versatile.
- **Tweedie distribution**. Versatile family of distributions. Useful for data with a mix of zeros and positive values (not necessarily counts).
- **Zero-inflated Poisson or zero-inflated negative binomial**. When the data comprise an excess number of zeros, that arise from a different process than the process that generates the counts.

GLMMs

Review: Linear Mixed Models

Review of LMM Workshop:

- Structure in the dataset or correlation among observations can result in **lack of independence among observations** sampled from same sites or time points
- Account for this by including r**andom effect terms**

Random effects:

- Parameter is a sample from the population, i.e. the subjects you happen to be working with
- Explains the variance of the response variable

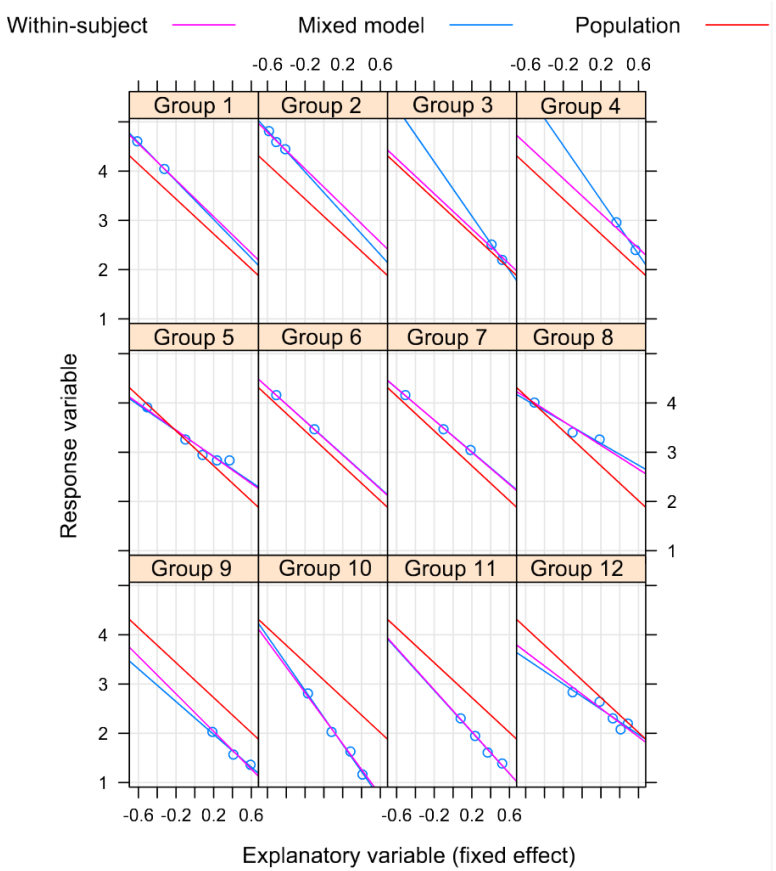
Fixed effects:

- Parameter is reproducible, i.e. would be the same across studies
- Explain the mean of the response variable

Review: Linear Mixed Models

Shrinkage estimates

- Random effects are often called **shrinkage estimates** because they represent a weighted average of the data and the overall fit (fixed effect)
- The random effect shrinkage toward the overall fit (fixed effect) is more severe if the within-group variability is large compared to the among-group variability



Generalized Linear Mixed Models

Extension of GLMs to account for additional structure in dataset

Follows similar steps introduced in LMM Workshop

1. LMMs incorporate random effects
2. GLMs handle non-normal data (letting errors take on different distribution families - e.g. Poisson or negative binomial)

How to run a GLMM in R

Import the `Arabidopsis` dataset `banta_totalfruits.csv` into R.

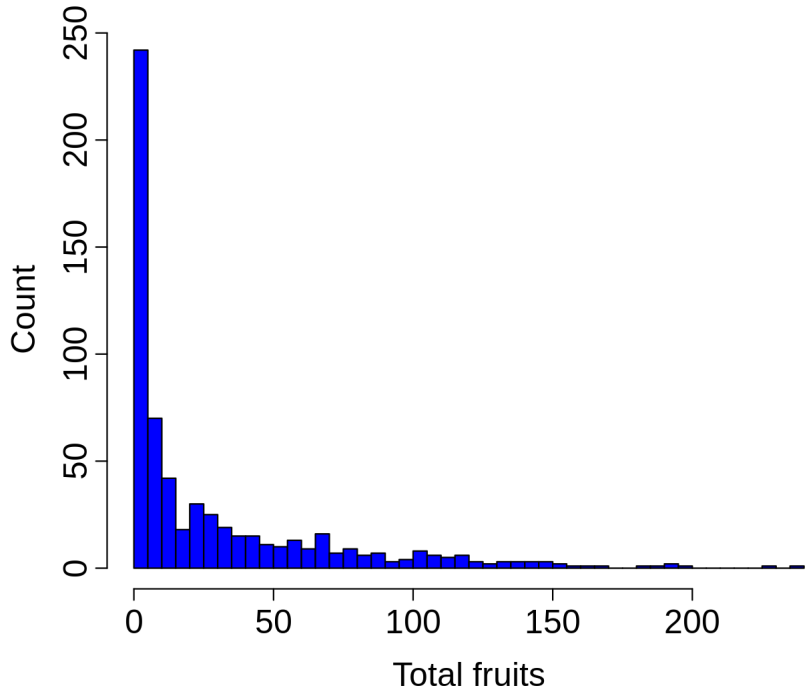
```
dat.tf ← read.csv("banta_totalfruits.csv")
```

```
# popu factor with a level for each population
# gen factor with a level for each genotype
# nutrient factor with levels for low (value = 1) or high (value = 8)
# amd factor with levels for no damage or simulated herbivory
# total.fruits integer indicating the number of fruits per plant
```

The effect of nutrient availability and herbivory (**fixed effects**) on the fruit production of the mouse-ear cress (Arabidopsis thaliana) was evaluated by measuring 625 plants across 9 different populations, each comprised of 2 to 3 different genotypes (**random effects**)

Choosing error distribution

The response variable is count data which suggests to that a **Poisson distribution** should be used (i.e. the variance is equal to the mean)



However, as we will soon see, the variance increases with the mean much more rapidly than expected under the Poisson distribution

Exploring variance

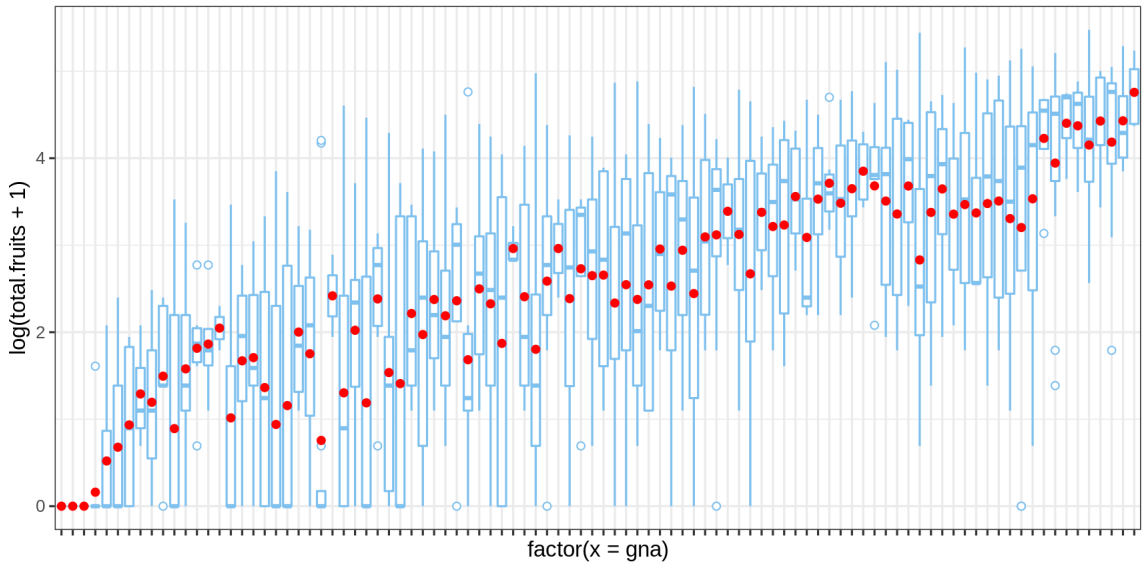
To illustrate heterogeneity in variance we will first create boxplots of the response variable versus different environmental factors

Let's create new variables that represents every combination of **nutrient** x **clipping** x **random factor**

```
dat.tf ← within(dat.tf,
{
  # genotype x nutrient x clipping
  gna ← interaction(gen,nutrient,amd)
  gna ← reorder(gna, total.fruits, mean)
  # population x nutrient x clipping
  pna ← interaction(popu,nutrient,amd)
  pna ← reorder(pna, total.fruits, mean)
})
```

Exploring variance

```
# Boxplot of total fruits vs genotype x nutrient x clipping interaction
library(ggplot2)
ggplot(data = dat.tf, aes(factor(x = gna),y = log(total.fruits + 1))) +
  geom_boxplot(colour = "skyblue2", outlier.shape = 21,
  outlier.colour = "skyblue2") +
  theme_bw() + theme(axis.text.x=element_blank()) +
  stat_summary(fun.y=mean, geom="point", colour = "red")
```

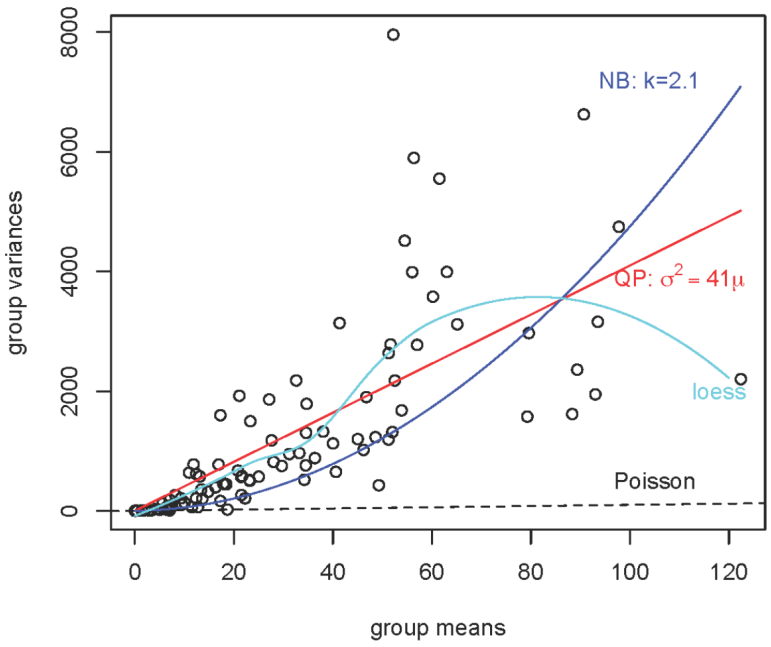


Similarly, the boxplot of total fruits vs population x nutrient x clipping interaction shows a large amount of heterogeneity among populations.

Choosing error distribution

As we just saw, there is a large amount of heterogeneity among group variances even when the response variable is transformed

If we plot the **group variances vs group means** (example with genotype x nutrient x clipping grouping shown here), we can appreciate that the Poisson family is the least appropriate distribution (i.e. variances increase much faster than the means)



NB = negative binomial

QP = quasi-Poisson

loess = Locally weighted regression smoothing

Poisson GLMM

Given the mean-variance relationship, we will most likely need a model with overdispersion

- but let's start with a Poisson model:

To run a GLMM in R we simply need to use the `glmer()` function of the lme4 package

```
library(lme4)
mp1 ← glmer(total.fruits ~ nutrient*amd + rack + status +
              (1|popu)+
              (1|gen),
              data = dat.tf, family = "poisson")
```

Random effects: `(1|popu)` contains a random intercept shared by measures that have the same value for `popu`

Overdispersion check

We can check for overdispersion using the `overdisp_fun()` function (Bolker *et al*. 2011) which divides the Pearson residuals by the residual degrees of freedom and tests whether ratio is greater than unity

```
# Download the glmm_funs.R code from the wiki page and source it to run t
source(file="data/glmm_funs.R")
# Overdispersion?
overdisp_fun(mp1)
#       chisq       ratio           p        logp
# 15755.86833    25.57771     0.00000 -6578.47027
```

- Ratio is significantly >> 1
- As expected, we need to try a different distribution where the variance increase more rapidly than the mean

Negative binomial GLMM (Poisson-gamma)

Recall that the negative binomial (or Poisson-gamma) distribution meets the assumption that the **variance is proportional to the square of the mean**

```
mnb1 ← glmer.nb(total.fruits ~ nutrient*amd + rack + status +
                  (1|popu)+
                  (1|gen),
                  data=dat.tf, control=glmerControl(optimizer="bobyqa"))
# Control argument specifies the way we optimize the parameter values
```

```
# Overdispersion?
overdisp_fun(mnb1)
```

←Ratio is now much closer to 1 although p-value is still less than 0.05

Poisson-lognormal GLMM

- Another option is the **Poisson-lognormal** distribution.
- This can be achieved simply by placing an observation-level random effect in the formula.

```
mpl1 ← glmer(total.fruits ~ nutrient*amd + rack + status +
               (1|X) +
               (1|popu)+
               (1|gen),
data=dat.tf, family="poisson",
control = glmerControl(optimizer = "bobyqa"))
```

`(1|X)` deals with overdisp. by adding **observation-level random effects**

```
overdisp_fun(mpl1)
#         chisq         ratio             p          logp
#  1.775363e+02  2.886768e-01  1.000000e+00 -3.755952e-73
```

Ratio now meets our criterion

Poisson-lognormal GLMM

Visualization the model parameters: A graphical representation of the model parameters can be obtained using the `coefplot2()` function from the `coefplot2` package:

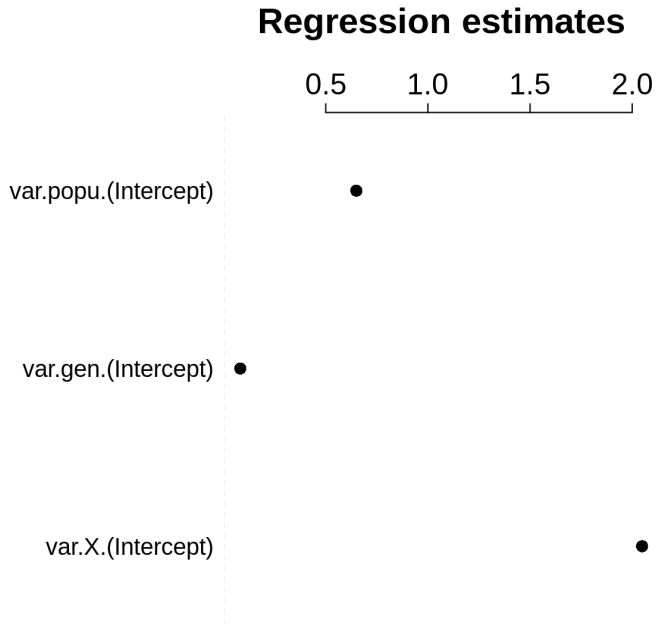
⚠ This package is not on CRAN! We install it from GittHub using the remotes package.

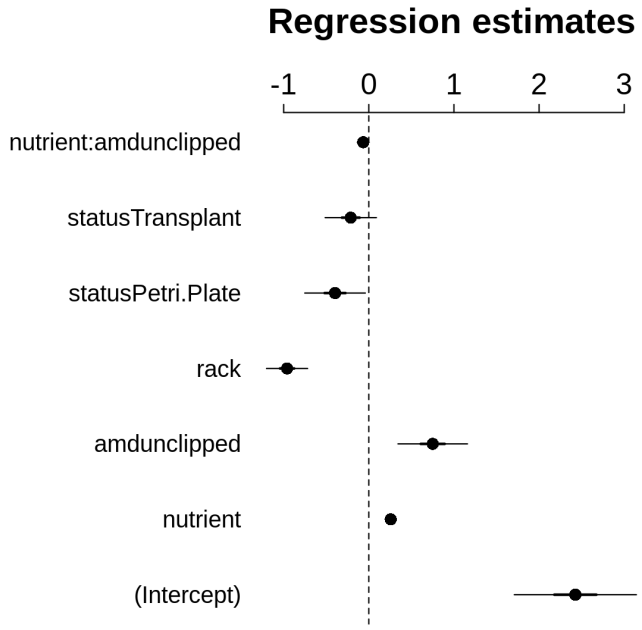
```
if (!require("coefplot2"))
  remotes::install_github("palday/coefplot2", subdir = "pkg")
library(coefplot2)
```

Poisson-lognormal GLMM

```
# Variance terms
coefplot2(mpl1, ptype = "vcov", i
```

```
# Fixed effects
coefplot2(mpl1, intercept = TRUE)
```





Note: error bars are only shown for the fixed effects because glmer doesn't provide information on uncertainty of variance .

Visualization the random effects

You can extract the random effect predictions using `ranef()` and plot them using a `dotplot()` from the `lattice` package

Regional variability among populations:

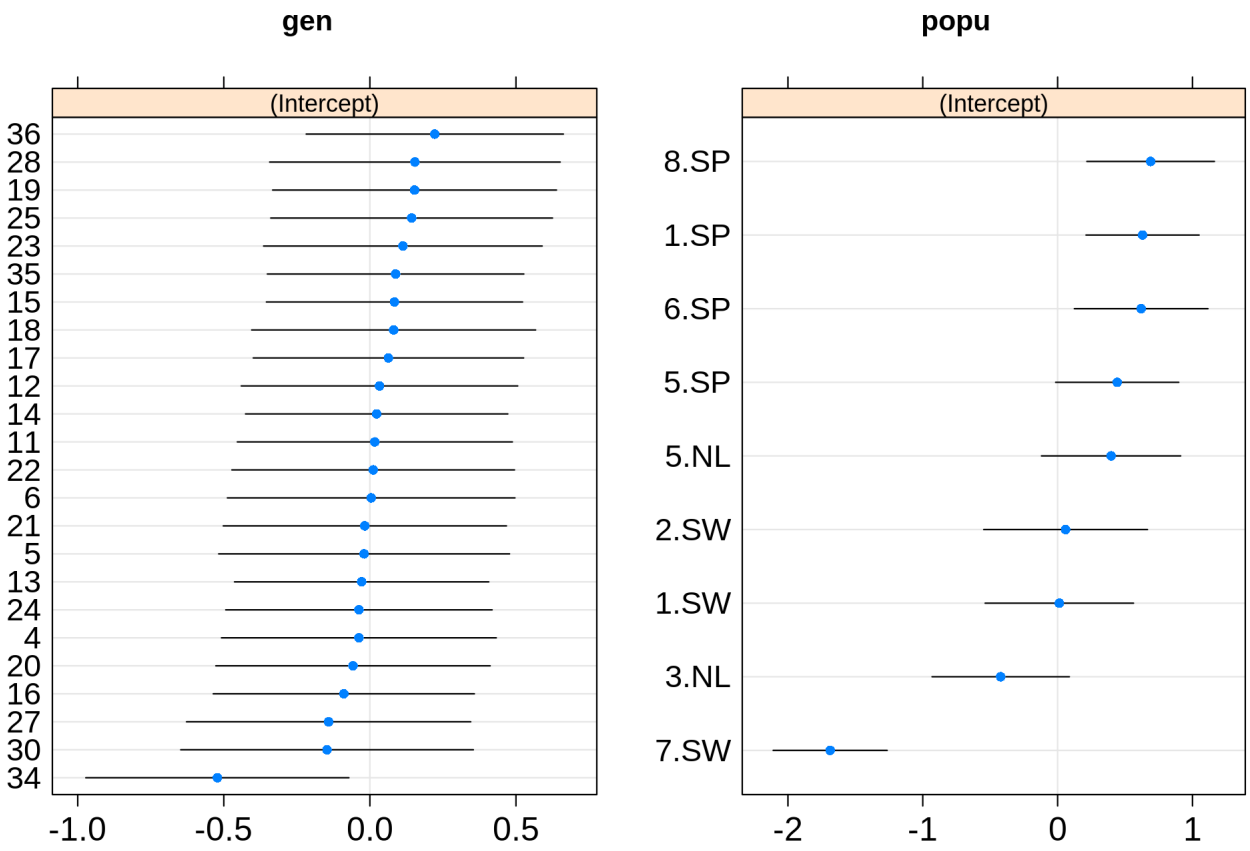
- Spanish populations (SP) larger values than Swedish (SW) or Dutch (NL)

Difference among genotypes largely driven by genotype 34

```
library(gridExtra)
library(lattice)
# dotplot code
pp ← list(layout.widths=list(left.padding=0, right.padding=0),
          layout.heights=list(top.padding=0, bottom.padding=0))
r2 ← ranef(mpl1, condVar = TRUE)
d2 ← dotplot(r2, par.settings = pp)

grid.arrange(d2$gen, d2$popu, nrow = 1)
```

Visualization the random effects



Model selection

The same methods can be used with a glmm or lmm to choose between models with various random intercepts and/or random slopes and to choose fixed effects to keep in final model.

- an **information theoretic approach** (e.g., AICc - Workshop 5)
- a **frequentist approach** (where the significance of each term is evaluated using `anova()` and the likelihood ratio test; LRT)

Model selection

We first code potential models and compare them using AICc*:

```
mpl2 ← update(mpl1, . ~ . - rack) # model without rack
mpl3 ← update(mpl1, . ~ . - status) # model without status
mpl4 ← update(mpl1, . ~ . - amd:nutrient) # without amd:nutrient interac
bbmle::ICtab(mpl1, mpl2, mpl3, mpl4, type = c("AICc"))
#      dAICc df
# mpl1  0.0  10
# mpl4  1.4  9
# mpl3  1.5  8
# mpl2 55.0  9
```

NB: We do not cover all possible models above, however, the interaction `amd:nutrient` can only be evaluated if both amd and nutrient (i.e., the main effects) are included in the model.

Model selection

Alternatively, we can use `drop1()` and `dfun()` functions to evaluate our fixed effects (`dfun()` converts the AIC values returned by the `drop1()` into ΔAIC values)

```
dd_LRT ← drop1(mpl1,test="Chisq")
(dd_AIC ← dfun(drop1(mpl1)))
# Single term deletions
#
# Model:
# total.fruits ~ nutrient * amd + rack + status + (1 | X) + (1 |
#     popu) + (1 | gen)
#              Df   dAIC
# <none>           0.000
# rack          1 55.083
# status        2  1.612
# nutrient:amd  1  1.444
```

- Strong **rack** effect (dAIC = 55.08 if we remove this variable)
- Effects of **status** and **interaction** term are weak (dAIC < 2)

Model selection

```
mpl2 ← update(mpl1, . ~ . - and:nutrient)
# Use AIC
mpl3 ← update(mpl2, . ~ . - rack) # no rack or
mpl4 ← update(mpl2, . ~ . - status) # no status
mpl5 ← update(mpl2, . ~ . - nutrient) # no nutr
mpl6 ← update(mpl2, . ~ . - amd) # no clipping
# bbmle::ICtab(mpl2, mpl3, mpl4, mpl5, mpl6,
#              type = c("AICc"))

# Or use drop1
dd_LRT2 ← drop1(mpl2,test="Chisq")
dd_AIC2 ← dfun(drop1(mpl2))
```

```
library(bbmle)
ICtab(mpl2, mpl3 ,
      mpl5, mpl6,
      type = c("AI
#      dAICc df
# mpl2  0.0  10
# mpl5  0.0  10
# mpl4  1.5  8
# mpl6 10.6  9
# mpl3 55.0  9
```

Both the main effects of **nutrient** and **clipping** are strong (large change in AIC of 135.6 (`mpl5`) and 10.2 (`mpl6`) if either nutrient or clipping are dropped, respectively).

- Final model includes the fixed nutrient and clipping effects, rack, and observation-level random e ffects `(1|X)` to account for over-dispersion

Up for a challenge?

Use the `inverts` dataset (larval development times (`PLD`) of 74 marine invertebrate and vertebrate species reared at different temperatures and time), answer the following questions:

- What is the effect of feeding type and climate (fixed effects) on `PLD`?
- Does this relationship vary among taxa (random effects)?
- What is the best distribution family for this count data?
- Finally, once you determined the best distribution family, re-evaluate your random and fixed effects.

Additional GLMM resources

Books:

- B. Bolker (2009) Ecological Models and Data in R. Princeton University Press.
- A. Zuur et al. (2009) Mixed Effects Models and Extensions in Ecology with R. Springer.

Websites:

- GLMM for ecologists (http://glmm.wikidot.com) *A great website on GLMM with a Q&A section!*

Thank you for attending this workshop!