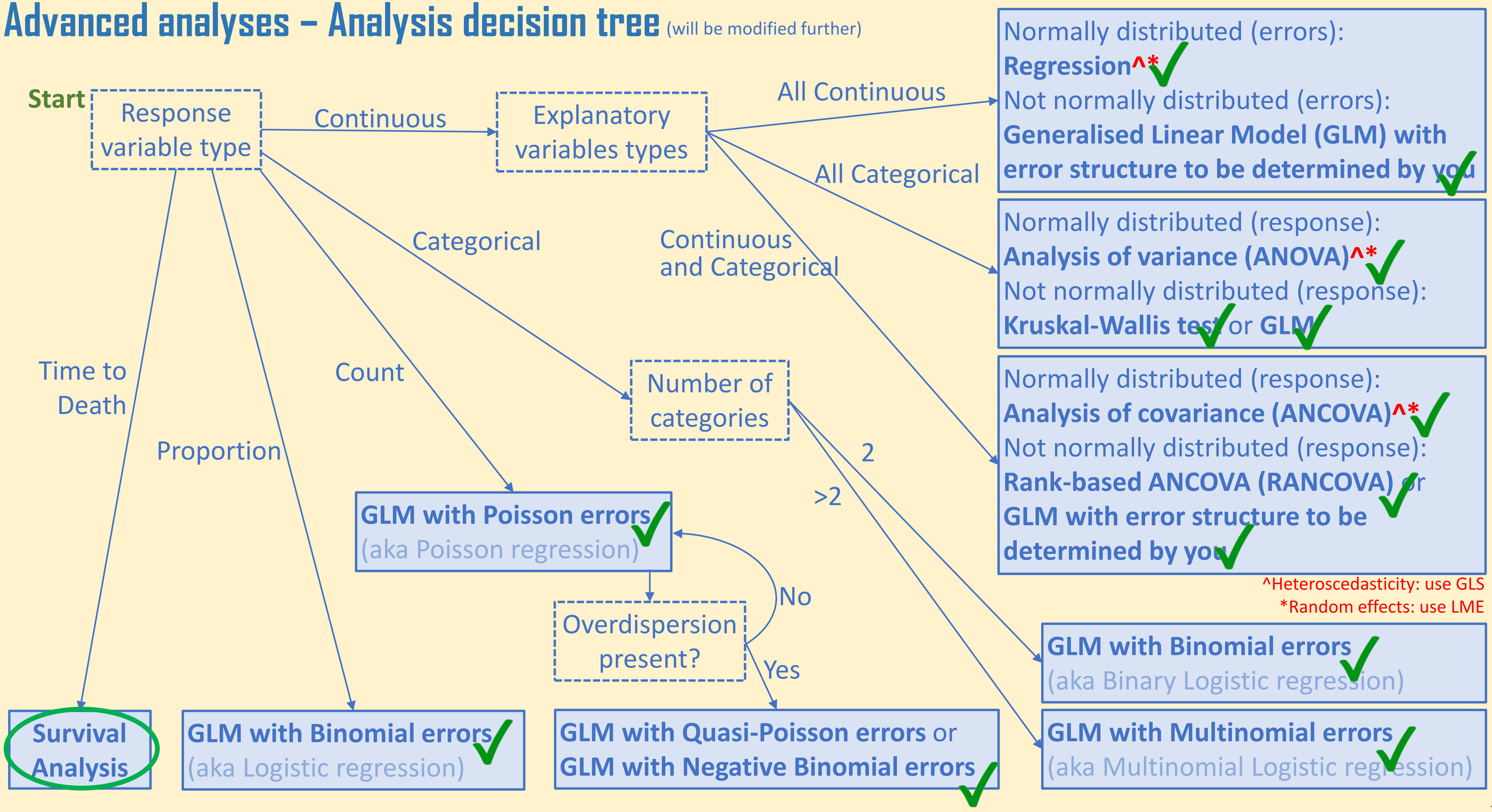
Survival Analysis & GLMM

Lecture 8

LSM3257

AY22/23; Sem 2 | Ian Z.W. Chan

Advanced analyses – Analysis decision tree (will be modified further)


Start
Response variable type
Continuous
Explanatory variables types
All Continuous
Normally distributed (errors):
Regression^* ✓
Not normally distributed (errors):
Generalised Linear Model (GLM) with error structure to be determined by you ✓
All Categorical
Normally distributed (response):
Analysis of variance (ANOVA)^* ✓
Not normally distributed (response):
Kruskal-Wallis test ✓ or GLM ✓
Continuous and Categorical
Normally distributed (response):
Analysis of covariance (ANCOVA)^* ✓
Not normally distributed (response):
Rank-based ANCOVA (RANCOVA) ✓ or
GLM with error structure to be determined by you ✓
^Heteroscedasticity: use GLS
*Random effects: use LME
Categorical
Number of categories
2
GLM with Binomial errors ✓
(aka Binary Logistic regression)
>2
GLM with Multinomial errors ✓
(aka Multinomial Logistic regression)
Count
GLM with Poisson errors ✓
(aka Poisson regression)
Overdispersion present?
No
Yes
GLM with Quasi-Poisson errors or GLM with Negative Binomial errors ✓
Proportion
GLM with Binomial errors ✓
(aka Logistic regression)
Time to Death
Survival Analysis

Summary (Learning Objectives)

Survival Analysis

- What it does, censoring and hazard functions

- Fitting, simplifying and interpreting: Gamma glm() and **survreg()**

- Plotting Kaplan-Meier curves using survfit() and autoplot()

Generalised Linear Mixed Models

- What is a GLMM?: error families, random errors and estimation methods

- Functions: **glmer()**, glmmPQL() and stan_glmer()

- Fitting and simplifying

- Solving errors

Survival Analysis

What is Survival Analysis?

A Survival Analysis models the time taken for an event to happen (e.g. death)

- Variance tends to increase exponentially with increasing values, data tends to be right skewed

- In some cases, it is equivalent to a GLM with an exponential or Gamma error distribution.

time to spot an animal

2 classes of survival models:

used in clinical studies, not really what we are interested in

- Nonparametric: focuses on the individual and cannot make predictions for the future.

- Parametric: models the entire population and cane make predictions for the future—in ecology we are more interested in this.

this is what we are interested in

Examples:

1) Predicting **how long an animal will survive** (number of days) based on its colour (categorical) and body mass (continuous).

2) Predicting **how long it will take for a storm to happen** (number of months) based on average sea surface temperature (continuous).

Censoring

Tells R that even though the datapoint stops, the event did not occur.

- R will treat the datapoint differently: it will use only the fact that the event had not occurred by that time to inform the model.

lets say we are doing experiment for 10 days, but if the event hasnt occured after 10 days,
you just put 10 to keep the data but censoring can tell R that this is not occurance
same for let s say mesed up sampling point for one

Most common examples of when you need to censor your data:

1) The experiment ends and the event did not occur (e.g. the specimen didn't die).

2) There is a problem with the datapoint and it needed to be removed (e.g. unplanned experimental error or problem).

You will need to tell R which datapoints are censored (0), e.g. actually still alive, and which are not (1), e.g. really dead, when you do the analysis.

- "0": don't use this information

- "1": use this information

Survivorship and Hazard functions

Survivorship can be…

Type I: mortality rates **increase** with time (e.g. adult humans).

Type II: mortality rates are **constant**.

Type III: mortality rates **decrease** with time (e.g. fish over their life cycle).

We first choose different hazard functions to reflect these survivorships.

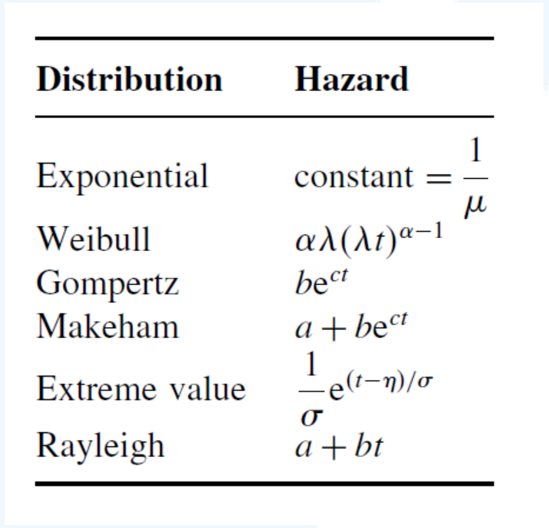
- Constant hazard: exponential distribution
- Hazard functions that vary with time…

Rayleigh: simplest, hazard increases linearly.

Makeham: most useful for humans.

Weibull: very flexible.

Loglogistic: consistently performs well.



Distribution	**Hazard**
Exponential	constant $= \frac{1}{\mu}$
Weibull	$\alpha\lambda(\lambda t)^{\alpha-1}$
Gompertz	be^{ct}
Makeham	$a + be^{ct}$
Extreme value	$\frac{1}{\sigma}e^{(t-\eta)/\sigma}$
Rayleigh	$a + bt$

We then compare the different models using AIC() to see which is best.

2 options to do a Survival Analysis

1) **GLM with Gamma errors**. Can only handle models with no censoring and constant hazard.

Uses `glm()` from Base R.

2) **Parametric survival analysis**. Can handle models with censoring and different hazard functions. We are more interested in this and I will be focusing on it.

Uses `survreg()` from the “survival” package.

Note: you could also do nonparametric survival analysis (we will not be covering this) using Cox proportional hazards models with coxph() from the “survival” package.

- This can handle models with censoring but not varying survivorship.

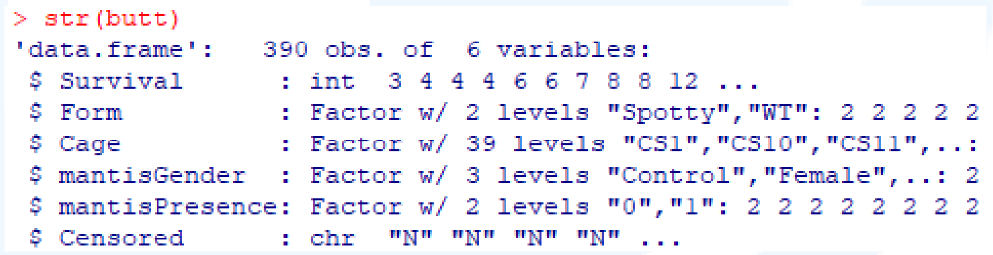
- The code is very similar but the interpretation of the results is different.

Example – Preparing the data

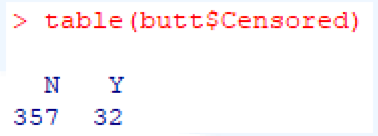
Let’s analyse how butterfly <Survival> (in days) is affected by how a butterfly looks <Form> in the presence of mantid predators <mantisPresence>. The <Cage> that the butterfly came from is a random effect.

Reading in the dataset

```
butt=read.table("buttSurvival.txt", header=T)
butt$Form=as.factor(butt$Form)
butt$Cage=as.factor(butt$Cage)
butt$mantisGender=as.factor(butt$mantisGender)
butt$mantisPresence=as.factor(butt$mantisPresence)
str(butt)
table(butt$Censored)
```



```
> str(butt)
'data.frame':   390 obs. of  6 variables:
 $ Survival      : int  3 4 4 4 6 6 7 8 8 12 ...
 $ Form          : Factor w/ 2 levels "Spotty","WT": 2 2 2 2 2
 $ Cage          : Factor w/ 39 levels "CS1","CS10","CS11",..:
 $ mantisGender  : Factor w/ 3 levels "Control","Female",..: 2
 $ mantisPresence: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2
 $ Censored      : chr  "N" "N" "N" "N" ...
```

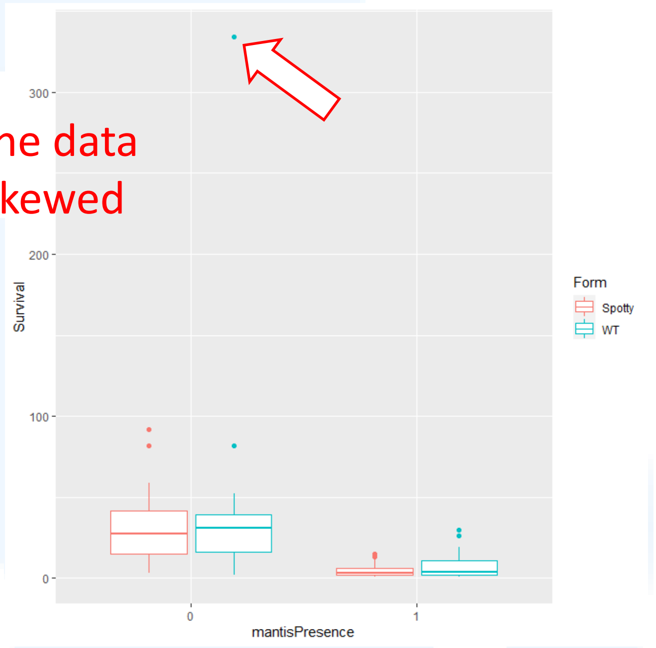


```
> table(butt$Censored)

  N   Y
357  32
```

Example – Exploring the data

Boxplots for quick comparison

```
ggplot(data=butt,aes(x=mantisPresence,y=Survival))+
geom_boxplot(aes(col=Form))
```

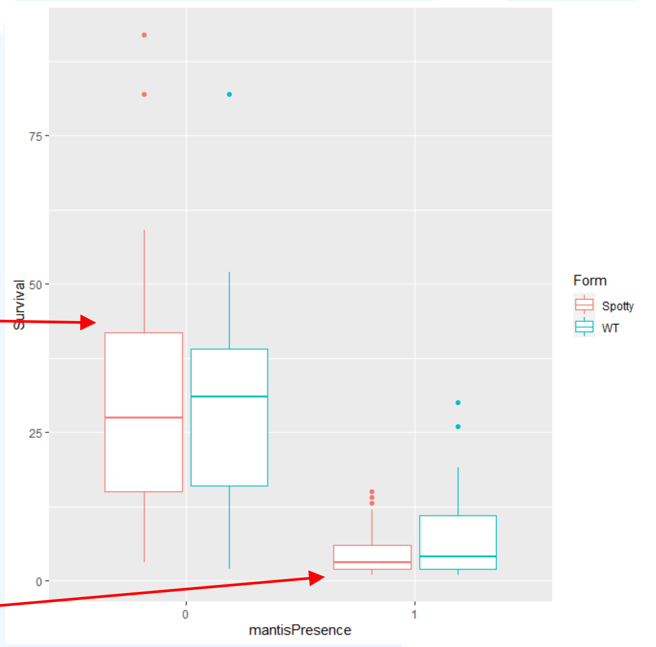


Notice how the data is very right skewed

Quickly remove the extreme outlier (optional, up to you to decide if this is necessary) and replot

```
butt=butt[-which(butt$Survival>300),]
```

```
ggplot(data=butt,aes(x=mantisPresence,y=Survival))+
geom_boxplot(aes(col=Form))
```



As expected, when there are no mantid predators, butterflies survive longer and it looks like there's no difference between Spotty and WT.

When there are predators, it looks like WT may survive longer than Spotty. However, this is not clear.

Example – Fitting a GLM with Gamma errors

Debugging Time!

Can you spot the error in this slide?

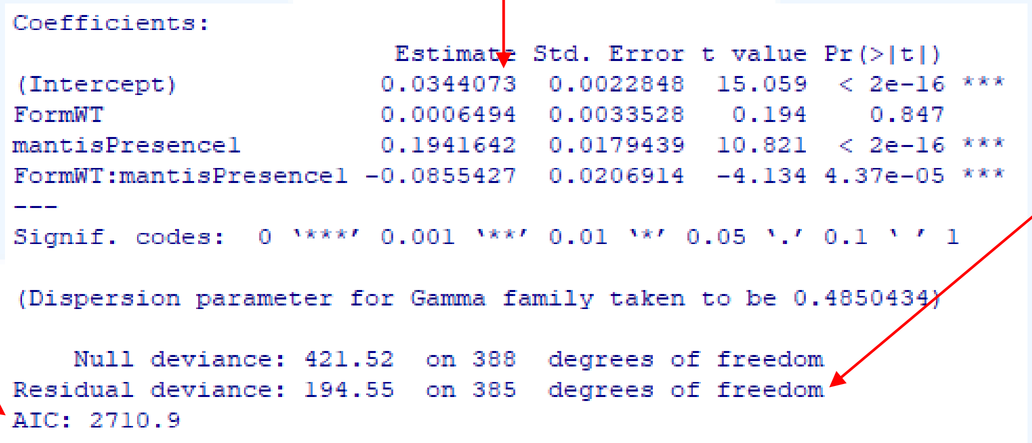
no censoring, no hazard function, no random effect

```
mod1.1=glm(Survival~Form*mantisPresence,family=gamma,data=butt)
#Diagnostics
require(DHARMa)
plot(simulateResiduals(mod1.1))
#View results
summary(mod1.1)
```

Gamma!! Capital!!

performance: check_model

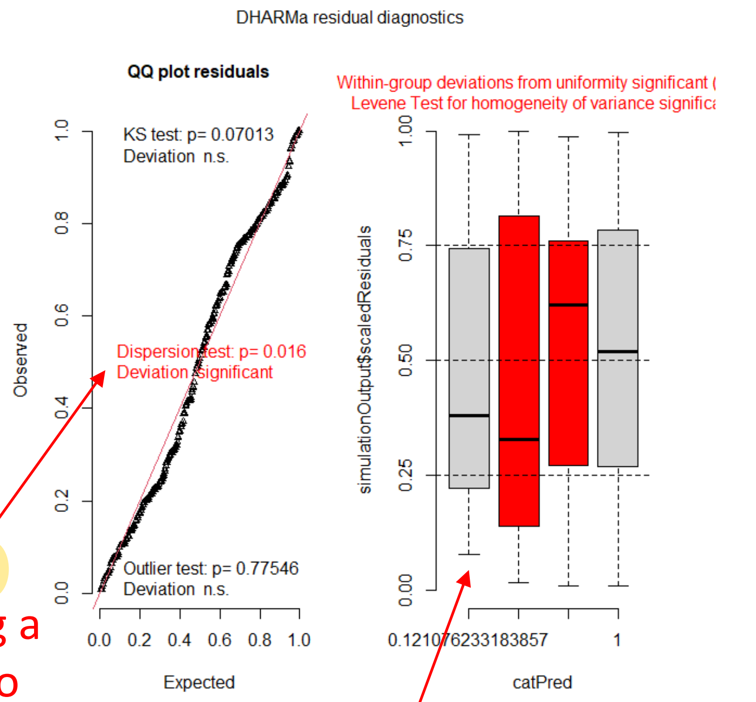
Link function is inverse (check ?family), so to convert back to time survived, take the inverse of this value:
- Survival of Spotty = inverse of intercept (1/0.034).
- Survival of WT = 1/(0.0344+0.00065).



```
Coefficients:
                         Estimate  Std. Error t value Pr(>|t|)
(Intercept)             0.0344073  0.0022848  15.059  < 2e-16 ***
FormWT                  0.0006494  0.0033528   0.194    0.847
mantisPresence1         0.1941642  0.0179439  10.821  < 2e-16 ***
FormWT:mantisPresence1 -0.0855427  0.0206914  -4.134 4.37e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Gamma family taken to be 0.4850434)

    Null deviance: 421.52  on 388  degrees of freedom
Residual deviance: 194.55  on 385  degrees of freedom
AIC: 2710.9
```

Can do model simplification using AIC()

Gamma accommodates for overdispersion using a dispersion parameter, so don't worry about this.



Looks like there is a problem with equality of variance. We cannot use these results and will need to do the survival analysis another way.

We could not include censoring or non-constant hazards, or a random effect because this is a GLM. These may be reasons for our problems.

Example – Fitting a Parametric model

Make sure to code the <Censored> variable correctly: Not censored (i.e. use the data) = 1; Censored (i.e. don’t use the data) = 0

```
butt$Censored[butt$Censored=="N"]=1
butt$Censored[butt$Censored=="Y"]=0
butt$Censored=as.numeric(butt$Censored) #Must be logical or numeric
```

Fit the model

```
require(survival)
mod1.3=survreg(Surv(Survival,Censored)~Form*mantisPresence,cluster=Cage,
data=butt,dist="weibull")
```

This is the main function

Your explanatory variables. Can also use “+” for non-interacting or “/” for nesting.

Dataset

Hazard function is specified using the “dist=” argument. If none is provided, Weibull is used by default.

This is the “y-variable”. You have to use the Surv() function to create an object with your two variables for survival duration and censorship status.

Your “random effect”. Can only have 1 in survreg() models. So if you have multiple random effects, you could create a variable that reflects all of them using paste() though this is not ideal.

Example – Fitting a Parametric model

Fit other hazard functions and compare using AIC

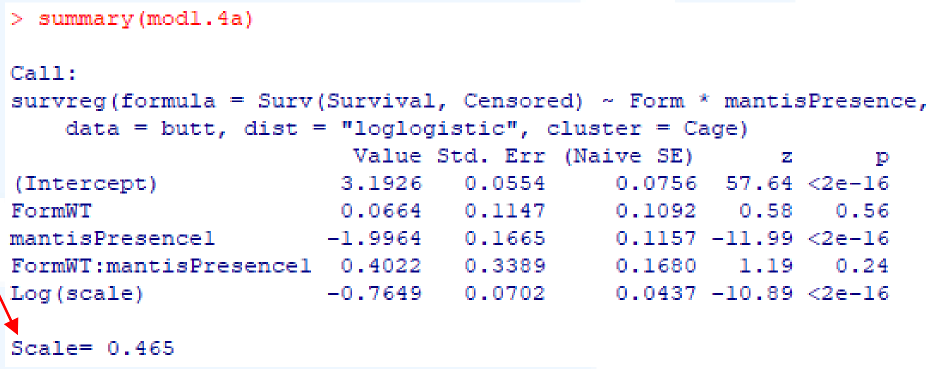
```
mod1.4a=survreg(Surv(Survival,Censored)~Form*mantisPresence,cluster=Cage,
data=butt,dist="loglogistic")
```

```
mod1.4b=survreg(Surv(Survival,Censored)~Form*mantisPresence,cluster=Cage,
data=butt,dist="exponential")
```

```
AIC(mod1.3,mod1.4a,mod1.4b) #mod1.4a best
```

Note: Distributions available in survreg(): "extreme", "gaussian", "logistic", "t", "loggaussian", "loglogistic", "rayleigh", "exponential" and "weibull" (the default)

Note that Scale < 1: this indicates that hazard decreases with time



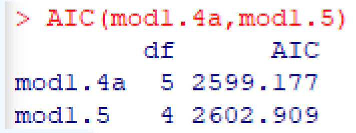
```
> summary(mod1.4a)

Call:
survreg(formula = Surv(Survival, Censored) ~ Form * mantisPresence,
    data = butt, dist = "loglogistic", cluster = Cage)
                         Value Std. Err (Naive SE)      z     p
(Intercept)             3.1926   0.0554     0.0756  57.64 <2e-16
FormWT                  0.0664   0.1147     0.1092   0.58   0.56
mantisPresence1        -1.9964   0.1665     0.1157 -11.99 <2e-16
FormWT:mantisPresence1  0.4022   0.3389     0.1680   1.19   0.24
Log(scale)             -0.7649   0.0702     0.0437 -10.89 <2e-16

Scale= 0.465
```

Proceed with <mod1.4a> for simplification

```
summary(mod1.4a) #interaction non-significant
```

```
mod1.5=update(mod1.4a,~.-Form:mantisPresence)
```

```
AIC(mod1.4a,mod1.5) #mod1.4a is better, so we keep the interaction
```

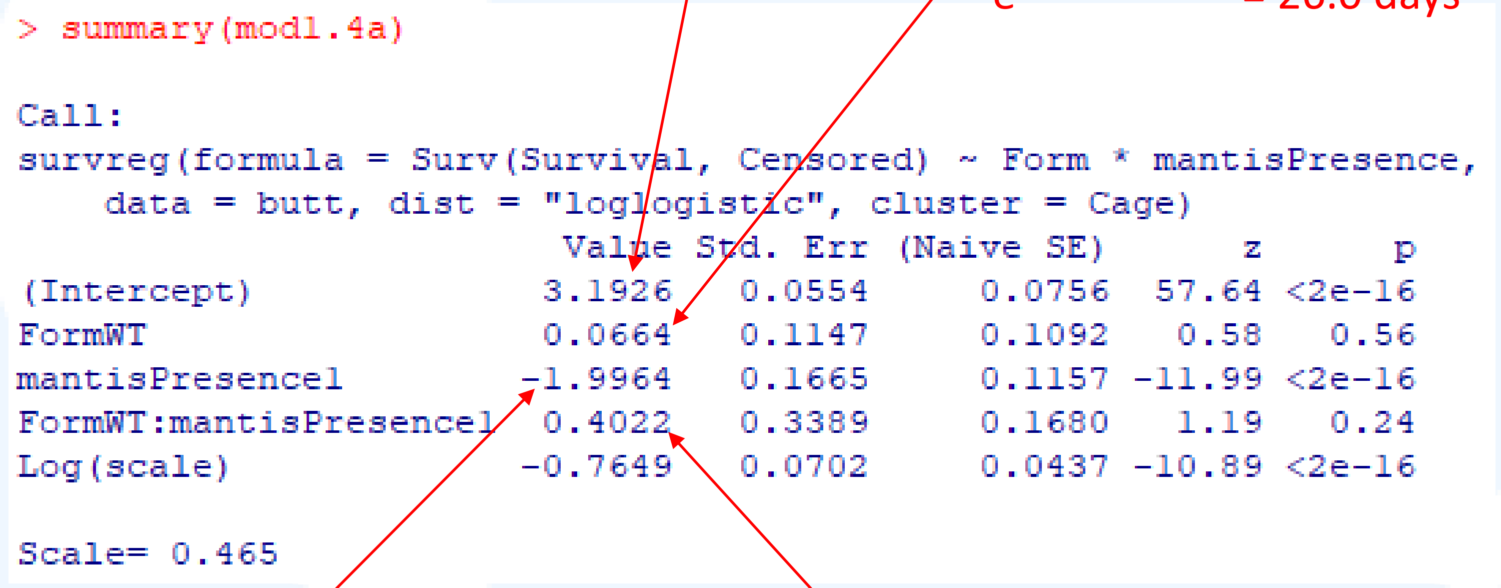


```
> AIC(mod1.4a,mod1.5)
        df      AIC
mod1.4a  5 2599.177
mod1.5   4 2602.909
```

Example – Fitting a Parametric model

Interpret results

`summary(mod1.4a)`



```
> summary(mod1.4a)

Call:
survreg(formula = Surv(Survival, Censored) ~ Form * mantisPresence,
    data = butt, dist = "loglogistic", cluster = Cage)
                        Value Std. Err (Naive SE)      z       p
(Intercept)            3.1926   0.0554     0.0756  57.64 <2e-16
FormWT                 0.0664   0.1147     0.1092   0.58   0.56
mantisPresence1       -1.9964   0.1665     0.1157 -11.99 <2e-16
FormWT:mantisPresence1  0.4022   0.3389     0.1680   1.19   0.24
Log(scale)            -0.7649   0.0702     0.0437 -10.89 <2e-16

Scale= 0.465
```

Average Spotty (the reference <Form>) survival = $e^{3.1926}$ = 24.4 days

Average WT survival = $e^{(3.1926+0.0664)}$ = 26.0 days

Average survival of Spotty in the presence of mantids = $e^{(3.1926-1.9964)}$ = 3.3 days

Average survival of WT in the presence of mantids = $e^{(3.1926-1.9964+0.4022)}$ = 4.9 days

given the presence of mantis, whats the survival of WT?

intercept + mantisPresece1 + FormWT:mantisPresence1

only look at the second value! cuz the value assumes you already corrected for mantis presence.

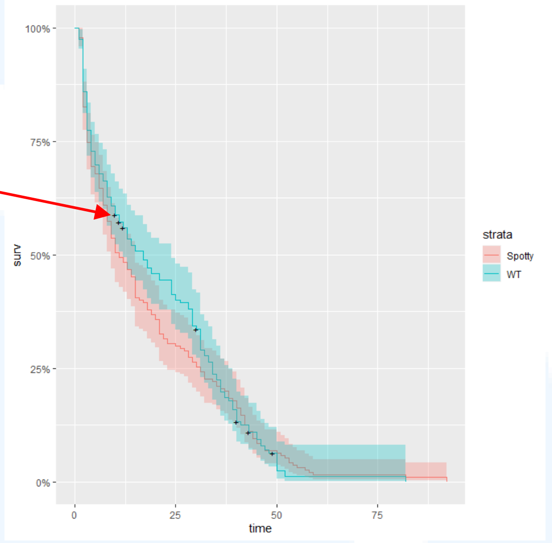
Example – Plotting a Kaplan-Meier survival curve

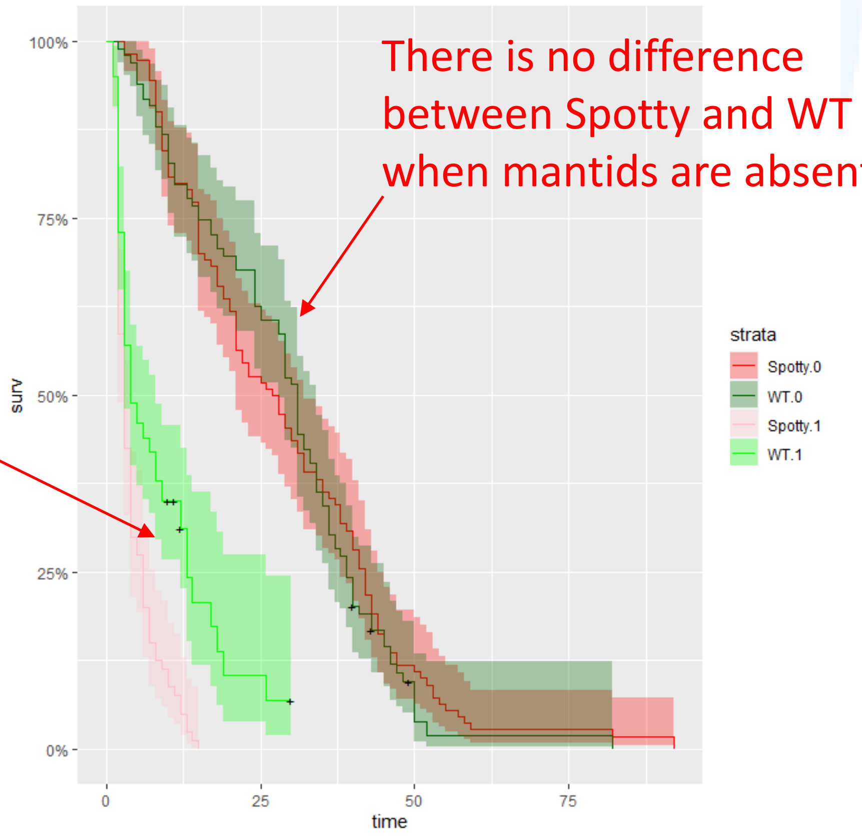
Plot the data

```
require(ggfortify) #for autoplot
#For 1 variable
splot1=survfit(Surv(Survival,Censored)~Form,data=butt)
autoplot(splot1)


#For 2 variables
splot2=survfit(Surv(Survival,Censored)~
interaction(Form,mantisPresence),data=butt)
autoplot(splot2)
```

Can add more variables if you want: "interaction(xv1,xv2,xv3)"

"+" denotes places where a datapoint was censored





There is no difference between Spotty and WT when mantids are absent

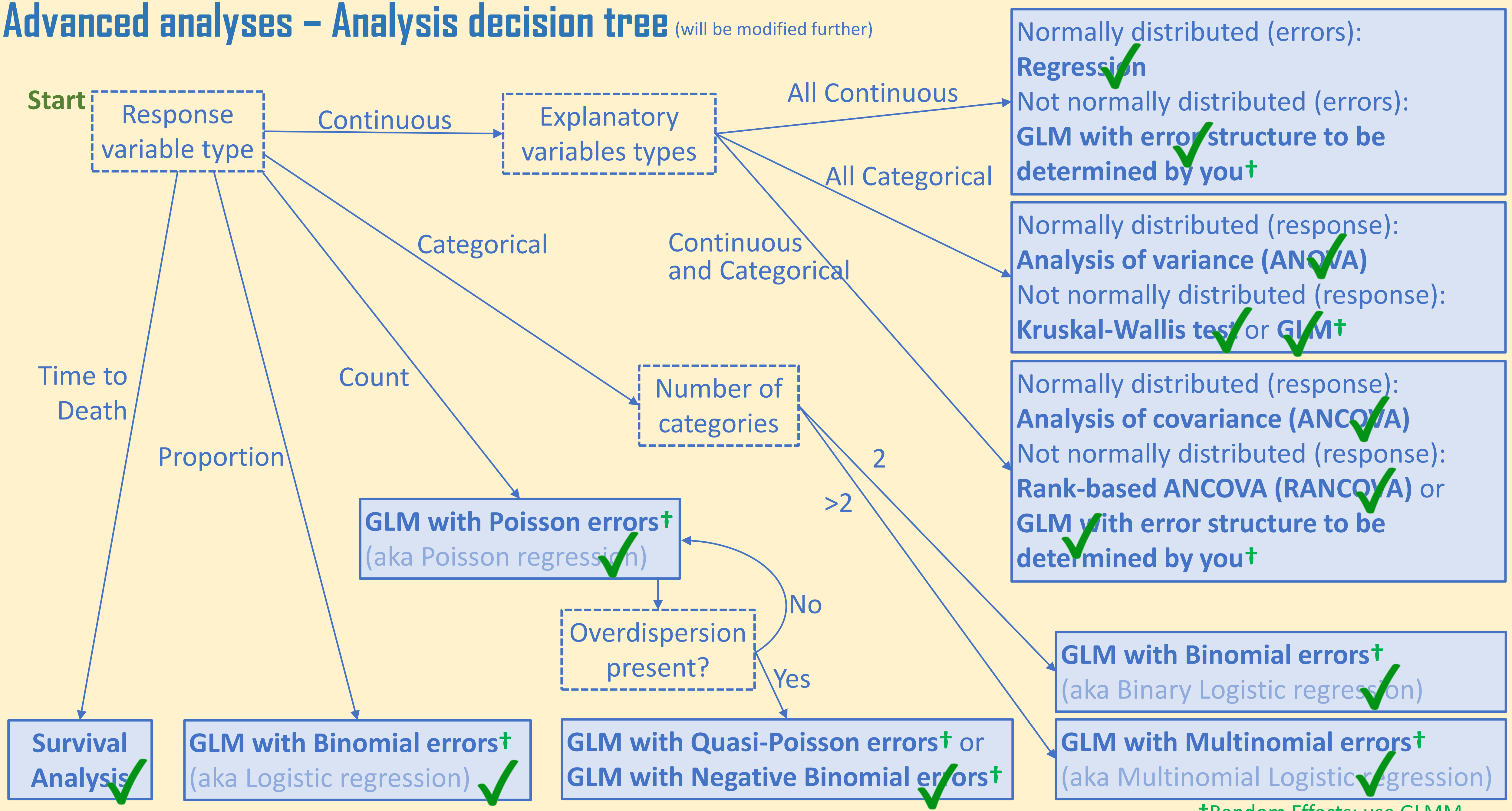
There's a clear difference between Spotty and WT when mantids are present

Even though the interaction was non-significant, from the plots we can see it has an important effect: be careful of following p-values blindly

GLMM

Generalised Mixed (effects) Linear Models

Advanced analyses – Analysis decision tree (will be modified further)


Start
Response variable type
Continuous
Explanatory variables types
All Continuous
Normally distributed (errors): **Regression** ✓
Not normally distributed (errors): **GLM with error structure to be determined by you**† ✓
All Categorical
Normally distributed (response): **Analysis of variance (ANOVA)** ✓
Not normally distributed (response): **Kruskal-Wallis test** ✓ or **GLM**† ✓
Continuous and Categorical
Normally distributed (response): **Analysis of covariance (ANCOVA)** ✓
Not normally distributed (response): **Rank-based ANCOVA (RANCOVA)** ✓ or **GLM with error structure to be determined by you**† ✓
Categorical
Number of categories
2
GLM with Binomial errors† (aka Binary Logistic regression) ✓
>2
GLM with Multinomial errors† (aka Multinomial Logistic regression) ✓
Count
GLM with Poisson errors† (aka Poisson regression) ✓
Overdispersion present?
No
Yes
GLM with Quasi-Poisson errors† or **GLM with Negative Binomial errors**† ✓
Proportion
GLM with Binomial errors† (aka Logistic regression) ✓
Time to Death
Survival Analysis ✓

†Random Effects: use GLMM.

What is it?

Generalised Liner Mixed-effect Models are exactly like GLMs (you just need to specify the correct error distribution for your variable type).

Recall:

Response Variable type	Error distribution	Canonical link function	Corresponding activation function
Continuous (normal)	Gaussian (aka Normal)	Identity: no conversion	Identity: no conversion
Count	Poisson	Log: $\ln(\text{count})$	e^x
Proportion (*p*)	Binomial	Logit: $\ln\left(\frac{p}{1-p}\right)$ (aka "log-odds")	$\frac{1}{1+\frac{1}{e^x}}$
Time (*T*) to event (e.g. survival)	Exponential, Gamma	Inverse: $\frac{1}{T}$	Inverse: $\frac{1}{\mathrm{x}}$
Continuous (non-normal)	Quasi	Nil	Nil

Except that GLMMs can also include random effects!!

Nice symmetry:
LM + Random Effects = LME
GLM + Random Effects = GLMM

Are they God's greatest gift to ecologists for doing stats?—Yes! Well, maybe.. Hmm, not quite.

Flexible: can model many different types of data. [Yes!]

- Continuous, counts, proportions, etc.

Powerful: they can also account for random effects. [Yes Yes Yes Yes Yes!!]

Problematic: "Convergence errors" are frequently encountered. [oh, well maybe]

- Try to keep your models simple and your sample size large.

They are new: many aspects are still being developed, there are many "rules" and best practices are still changing—making it very confusing. [hmm.. not quite]

- Examples: REML is not possible, quasipoisson and quasibinomial families were previously OK but are now not recommended.
- You will run into errors, BE BRAVE and experiment!

4 main Likelihood estimation methods

1) Penalised Quasi-likelihood (PQL). Estimates quasi-likelihood. This was the most widely used. However, it is sometimes unreliable, especially when the standard deviations of the random effects are large, with binary data, etc.

2) Laplace. Estimates true likelihood. More reliable than PQL but more computationally intensive.

3) Gauss-Hermite quadrature (GHQ). Even more reliable and computationally intensive, hence cannot handle more than 1 random effect.

4) Bayesian using MCMC. This is used in a Bayesian framework and can handle many random effects, but is the most computationally intensive.

MCMC (Markov chain Monte Carlo) is a way of getting values by repeated trial and error.

For more information: *Bolker et al. (2009). Generalized Linear Mixed Models: a practical guide for ecology and evolution. Trends in Ecology and Evolution 24(3):127-135.* (Ben Bolker is huge in the field, you see him in stats-exchange)

Fitting a GLMM – 3 different packages for different occasions

1) `glmmPQL()` from the "MASS" package—uses PQL.

- Pros: can fit variance structures (based on "nlme"), uses quasi-families automatically for overdispersion.

Cons: not good for multiple random effects, no AIC(), no diagnostics, less reliable in some cases.

General code:

Function to fit GLMM

Fixed effects. Can contain interacting, non-interacting and nested variables

Error distribution. Because this is glmmPQL(), this is automatically quasipoisson. "family=binomial" would be quasibinomial.

```
require(MASS)
modelObject=glmmPQL(Y~X1*X2+X3/X4,random=~X5|X6/X7,family=poisson,
data=dataset,weights=vs1)
```

Name of dataframe object containing all the variables to be used

Variance structure. Needs to be predefined before this, same as with gls()

Random effects. Can contain random slope and intercept, and nested variables, but not good for multiple independent random effects (aka "crossed random effects")

Fitting a GLMM – 3 different packages for different occasions

2) `glmer()` from the “lme4” package (same as `lmer()`)—uses Laplace or GHQ.

- Pros: more reliable, can fit multiple random effects and use AIC().

- Cons: cannot deal with overdispersion for binomial distributions.

Error distribution. Can take poisson, binomial, Gamma, gaussian, but NOT quasipoisson and quasibinomial. Note that negative binomial is run using a different function

General code:

Function to fit GLMM

Fixed effects. Can contain interacting, non-interacting and nested variables

```
require(lme4)
modelObject=glmer(Y~X1*X2+X3/X4+(1|X6/X7)+(X8|X9),family=poisson,nAGQ=1,
data=dataset)
```

Name of dataframe object containing all the variables to be used

For negative binomial, only the function is different, all the other commands are the same (but no need to specify “family=“)

Random effects. Can contain random slope and intercept, nested variables and multiple independent random effects

This controls whether Laplace (nAGQ=1) or GHQ (nAGQ=2) estimation is used.
- If you set nAGQ=0, you get something similar to PQL. Default is nAGQ=1, so you can omit this command if you want to use Laplace (which I normally do).
- Note that for GHQ, you can only have one random intercept (aka scalar) random effect (i.e. no random slope, no multiple random effects and no nesting).

Separate function for negative binomial error distribution:

```
modelObject=glmer.nb(Y~X1+(X2|X3),nAGQ=1,data=dataset)
```

Fitting a GLMM – 3 different packages for different occasions

3) `stan_glmer()` from the "rstanarm" package—uses MCMC.

- Pros: powerful—can take multiple explanatory variables and random effects with interactions, nesting, random slopes and intercepts, etc.

- Cons: no quasi-families, takes a long time, harder to become expert.

MCMCglmm used to be the default package/function for doing GLMMs using Bayesian estimation. However, its documentation is notoriously poor and you MUST define priors (something to do with Bayesian stats which I will briefly cover, this is complicated to learn). With stan_glmer(), the rstanarm package uses default priors for the error distribution you specify. In an ideal world, you should learn how to define your own priors (I will not teach this) but it is still OK to use the defaults for a start.

General code (similar to `glmer()`):

```
require(rstanarm)
modelObject=stan_glmer(Y~X1*X2+X3/X4+(1|X6/X7)+(X8|X9),family=poisson,
QR=T,data=dataset)
```

Function to fit the GLMM

Fixed effects. Can contain interacting, non-interacting and nested variables

Random effects. Can contain random slope and intercept, nested variables and multiple independent random effects

Error distribution. Can take poisson, binomial, Gamma, gaussian, but NOT quasipoisson and quasibinomial. Similar to glmer(), negative binomial is run using a different function

non-informative prier

"QR" defaults to F. You can choose to set to T if you have more than 1 explanatory variable. Does not change results; just improves the computation

Separate function for negative binomial error distribution:

```
modelObject=stan_glmer.nb(Y~X1+(X2|X3),data=dataset)
```

No need to specify error distribution here

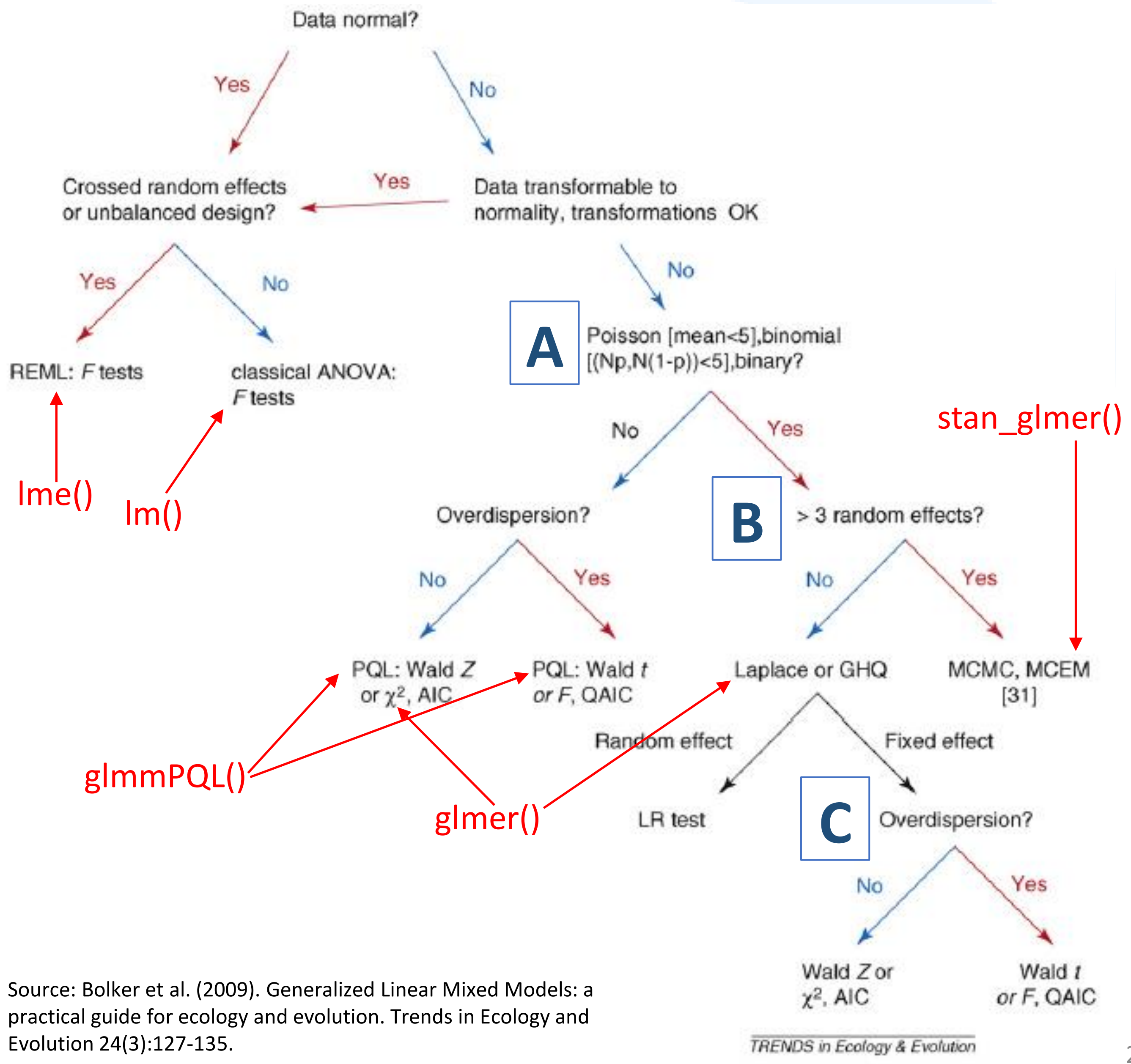
When to use what?: Bolker's decision tree

3 main decision points…

A: If you have: (i) Poisson where mean < 5; (ii) proportion where the number of successes or failures < 5; or (iii) binary data, do not use PQL.

B: If you have > 3 random effects, use MCMC.

C: When using Laplace/GHQ, if you have overdispersion, use Quasi-AIC.



Source: Bolker et al. (2009). Generalized Linear Mixed Models: a practical guide for ecology and evolution. Trends in Ecology and Evolution 24(3):127-135.

When to use what?: My personal procedure

1) Start with using `glmer()`.

2) Check for overdispersion. If there is overdispersion:

   For Gamma data: it is OK.

   For Poisson data: switch to `glmer.nb()` or `glmmPQL()`.

   For binomial data:

      If your data is binary: do both `glmer()` and `glmmPQL(family=binomial)` and compare the results (see boxed text).

      If your data is proportion:

         If either the number of successes or failures < 5: do both `glmer()` and `glmmPQL(family=binomial)` and compare the results.

         If both > 5: switch to `glmmPQL(family=binomial)`.

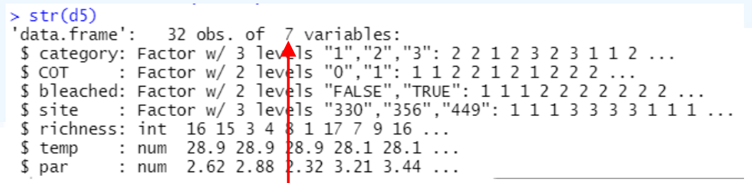
3) If you have variance structures (see GLS lecture), fit using `glmmPQL()`.

4) If you have very complex explanatory variables and error structures (and keep running into convergence errors): switch to `stan_glmer()`.

Binary data and overdispersion

- Recall that the binomial distribution is used to fit 2 kinds of data: proportions and binary.
- It has long been accepted that proportion data can be overdispersed but binary data cannot. Therefore, with binary data, even if residual deviance > d.f, people usually just ignore it.
- However, recently there have been arguments that both proportion and binary data CAN be overdispersed. If this is true, you should still switch to quasibinomial if your binary data shows overdispersion.
- My advice: try both.

Example – Preparing the data



```
> str(d5)
'data.frame':   32 obs. of  7 variables:
 $ category: Factor w/ 3 levels "1","2","3": 2 2 1 2 3 2 3 1 1 2 ...
 $ COT     : Factor w/ 2 levels "0","1": 1 1 2 2 1 2 1 2 2 2 ...
 $ bleached: Factor w/ 2 levels "FALSE","TRUE": 1 1 1 2 2 2 2 2 2 2 ...
 $ site    : Factor w/ 3 levels "330","356","449": 1 1 1 3 3 3 3 1 1 1 ...
 $ richness: int  16 15 3 4 8 1 17 7 9 16 ...
 $ temp    : num  28.9 28.9 28.9 28.1 28.1 ...
 $ par     : num  2.62 2.88 2.32 3.21 3.44 ...
```

32 observations for 7 variables is a little low: watch out for convergence errors!

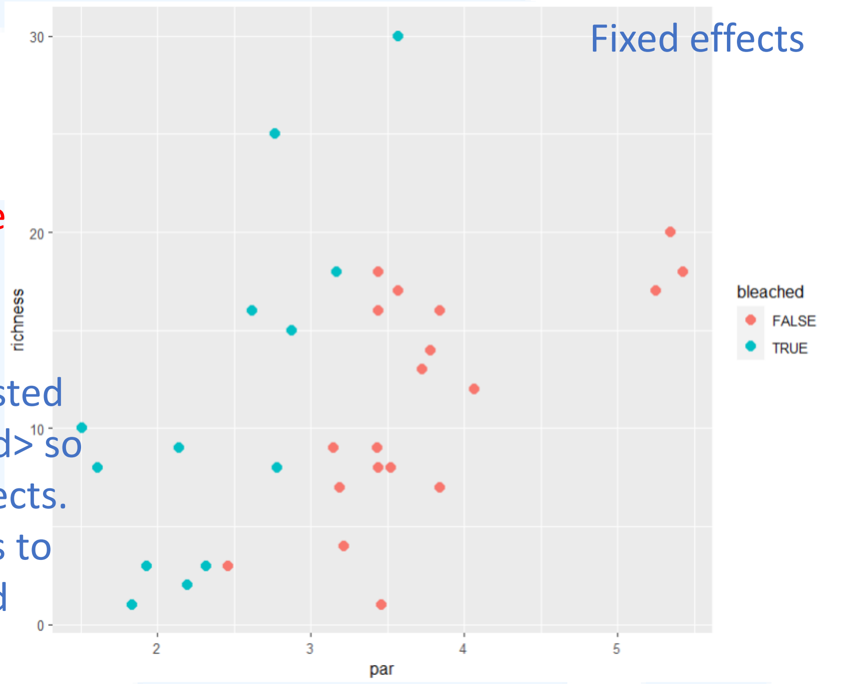
Reading in dataset:

```
d5=read.csv("ReefData.csv")
d5$category=as.factor(d5$category)
d5$bleached=as.factor(d5$bleached)
d5$site=as.factor(d5$site)
d5$COT=as.factor(d5$COT)
```

Response variable <richness> is a count

But we are only interested in <par> and <bleached> so these are our fixed effects. We use random effects to control for <temp> and <category>.
It looks like there is some pattern for all 4 variables: <par>, <bleached>, <temp> and <category>.

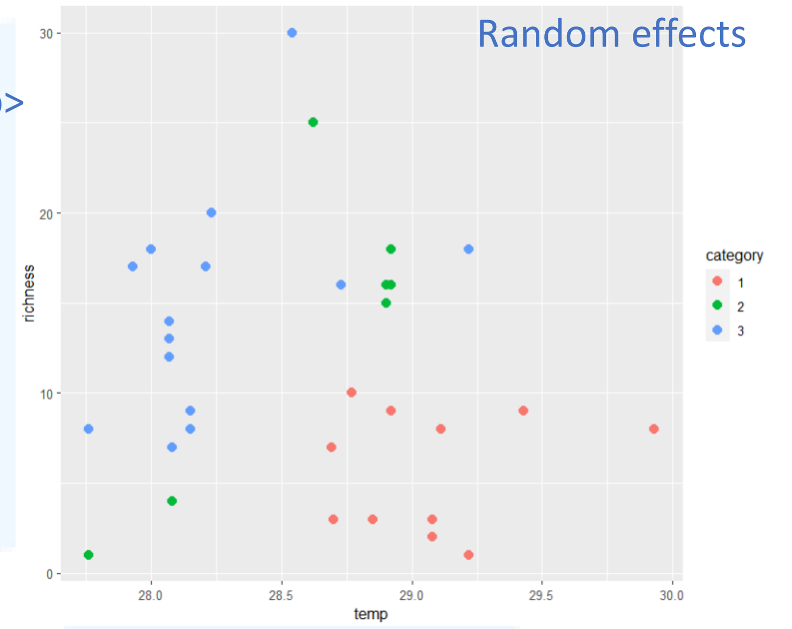
Fixed effects



Visualising fixed effects <par> and <bleached>:

```
ggplot(d5,aes(x=par,y=richness))+geom_point(aes(col=
bleached),size=3)
```

Random effects



Visualising random effects <temp> and <category>:

```
ggplot(d5,aes(x=temp,y=richness))+geom_point(aes(col
=category),size=3)
```

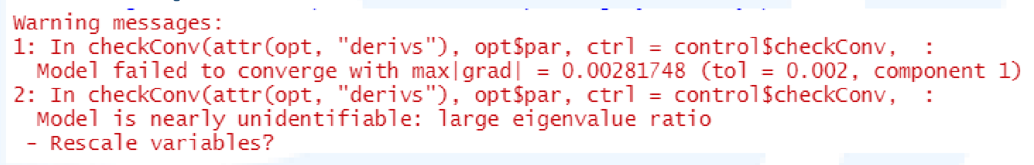
Example – Fitting the GLMM

Compared to LME
- Recall with LME, we used REML to simplify the random effects and then ML to simplify the fixed effects. This is not needed for GLMMs (there are arguments about whether REML makes sense here).
- It used to be doable in glmer()—you could previously specify "REML=T" or "REML=F"—but Ben Bolker has changed this.

Using glmer():

```
require(lme4)
mod2.1=glmer(richness~par*bleached+(temp|category),family=poisson,data=d5)
```

#Warnings you may see: "Model failed to converge", "Model is nearly unidentifiable". Let's try to simplify by removing <temp>.



```
Warning messages:
1: In checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv,  :
  Model failed to converge with max|grad| = 0.00281748 (tol = 0.002, component 1)
2: In checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv,  :
  Model is nearly unidentifiable: large eigenvalue ratio
 - Rescale variables?
```

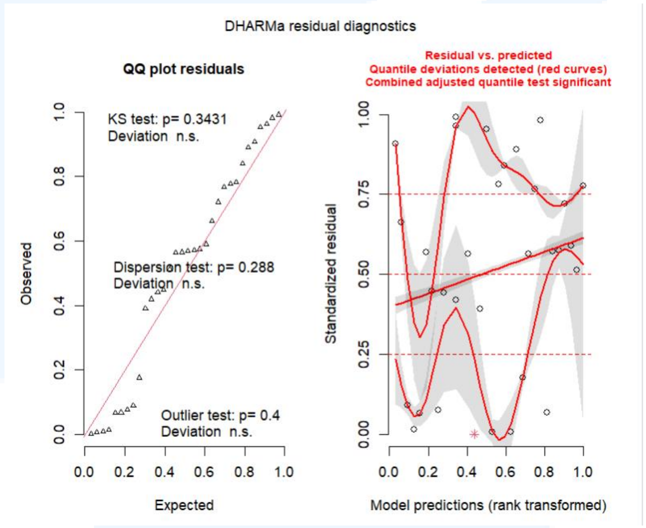
"Simplify" random effects first:

```
mod2.2=glmer(richness~par*bleached+(1|category),family=poisson,data=d5)
```

Test assumptions:

```
require(DHARMa)
plot(simulateResiduals(mod2.2))
```

No overdispersion but crazy problems with residuals.



Example – When there is overdispersion and/or other problems

Debugging Time!

Can you spot the error in this slide?

We would need to use either a quasipoisson or negative binomial distribution.

To do a quasipoisson we would use glmmPQL, but we first need to make sure the mean of <richness> is at least 5 (look at Bolker's decision tree).

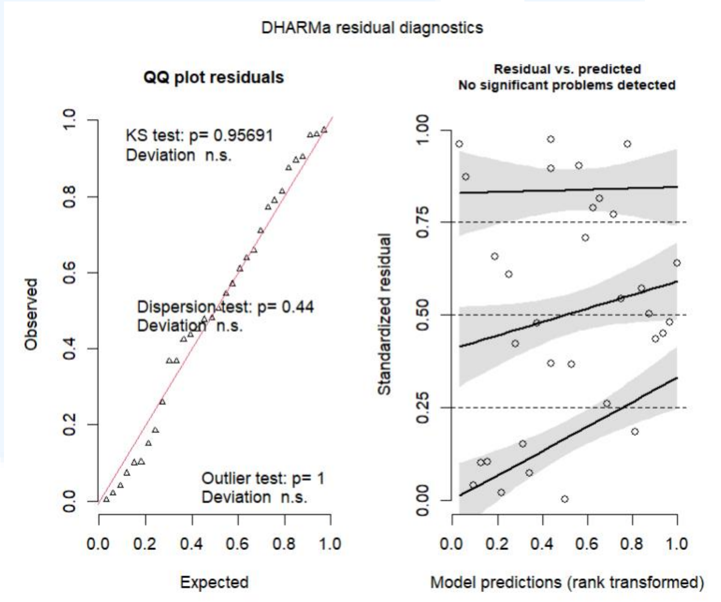
```
mean(d5$richness) #11.4, so either PQL or negative binomial are fine
```

Try a negative binomial first, we use glmer.nb():

```
mod2.2a=glmer.nb(richness~par+bleached+(1|category),data=d5)
plot(simulateResiduals(mod2.2a)) #solved!
```

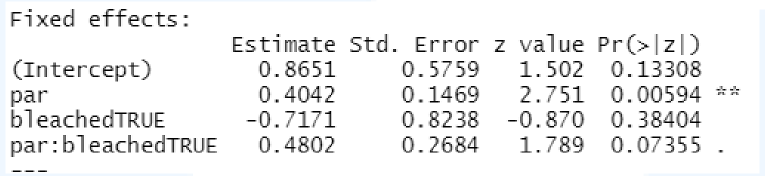
If not solved, we could switch to quasipoisson:

```
mod2.2b=glmmPQL(richness~par+bleached,
random=~1|category,family=quasipoisson, data=d5)
```



Example – Simplifying

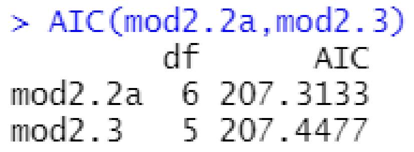
```
summary(mod2.2a) #interaction: P = 0.073
```



```
Fixed effects:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)        0.8651     0.5759   1.502  0.13308
par                0.4042     0.1469   2.751  0.00594 **
bleachedTRUE      -0.7171     0.8238  -0.870  0.38404
par:bleachedTRUE   0.4802     0.2684   1.789  0.07355 .
---
```

```
mod2.3=update(mod2.2a,~.-par:bleached)
```

```
AIC(mod2.2a,mod2.3)
```



```
> AIC(mod2.2a,mod2.3)
        df      AIC
mod2.2a  6 207.3133
mod2.3   5 207.4477
```

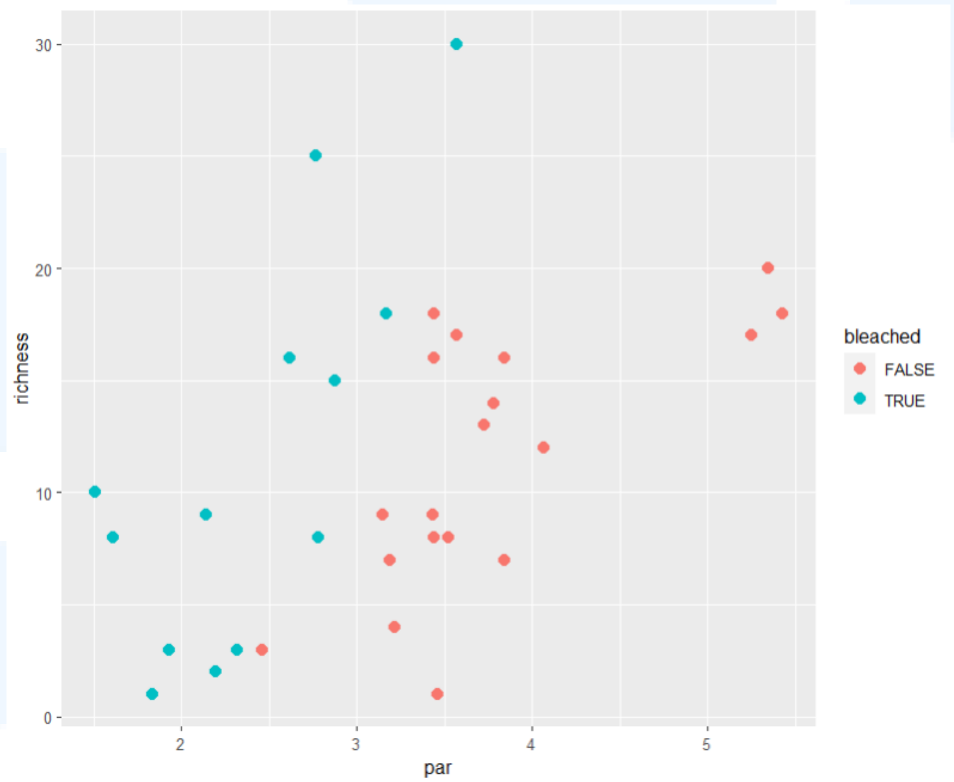
Always go back to the data and the biological intuition. Is there an interaction? It looks like there is no interaction, so I simplify:

```
summary(mod2.3)
```

```
#Both <par> and <bleached> are now
significant, this is my final model
```

Current practices for simplifying GLMMs

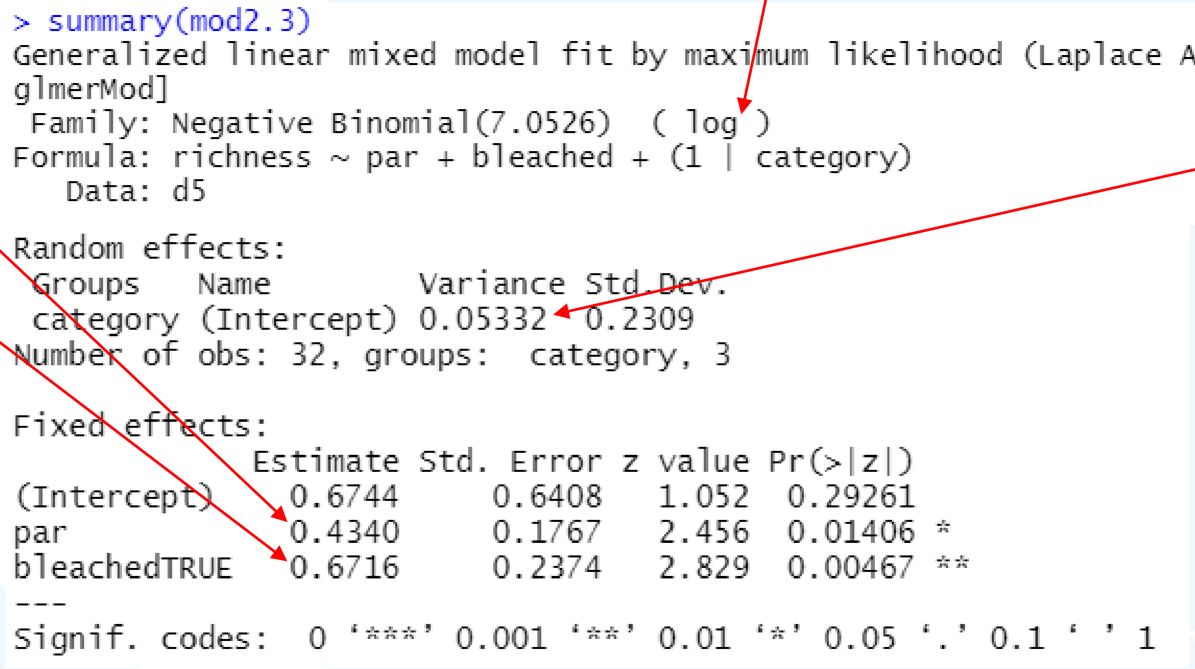
- We don't really simplify Random Effects. We choose them based on our experimental design. During analysis, we may choose to "simplify" the Random Effects to solve convergence problems.
- To compare fixed effects in models, we use…
  For non-quasi models: AIC or anova(test="chisq")
  For quasi models: there is currently no test to compare models fit using glmmPQL, we just have to rely on the p-values (read this).



Example – Interpreting

```
summary(mod2.3)

exp(summary(mod2.3)$coef[1,1]+summary(mod2.3)$coef[2,1]) #for <par>

exp(0.67844+0.6716)
```

Recall this is the link function, so to convert the effect sizes from the units of the linear predictor (reported as the "Estimate"), we need to take e^{Estimate}.



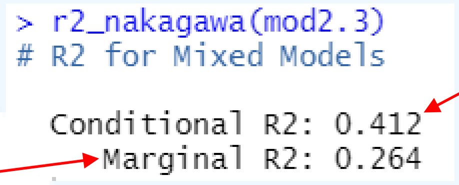
```
> summary(mod2.3)
Generalized linear mixed model fit by maximum likelihood (Laplace Approximation) [
glmerMod]
 Family: Negative Binomial(7.0526)  ( log )
Formula: richness ~ par + bleached + (1 | category)
   Data: d5

Random effects:
 Groups   Name        Variance Std.Dev.
 category (Intercept) 0.05332  0.2309
Number of obs: 32, groups:  category, 3

Fixed effects:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    0.6744     0.6408   1.052  0.29261
par            0.4340     0.1767   2.456  0.01406 *
bleachedTRUE   0.6716     0.2374   2.829  0.00467 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Amount of variance explained by the random effect <category>. Not easy to compare this to the amount explained by the fixed effects directly (need another function below).

Calculate proportion of variance explained by fixed vs. random effects

```
require(performance)

r2_nakagawa(mod2.3)
```

Proportion of total variance explained by fixed effects alone



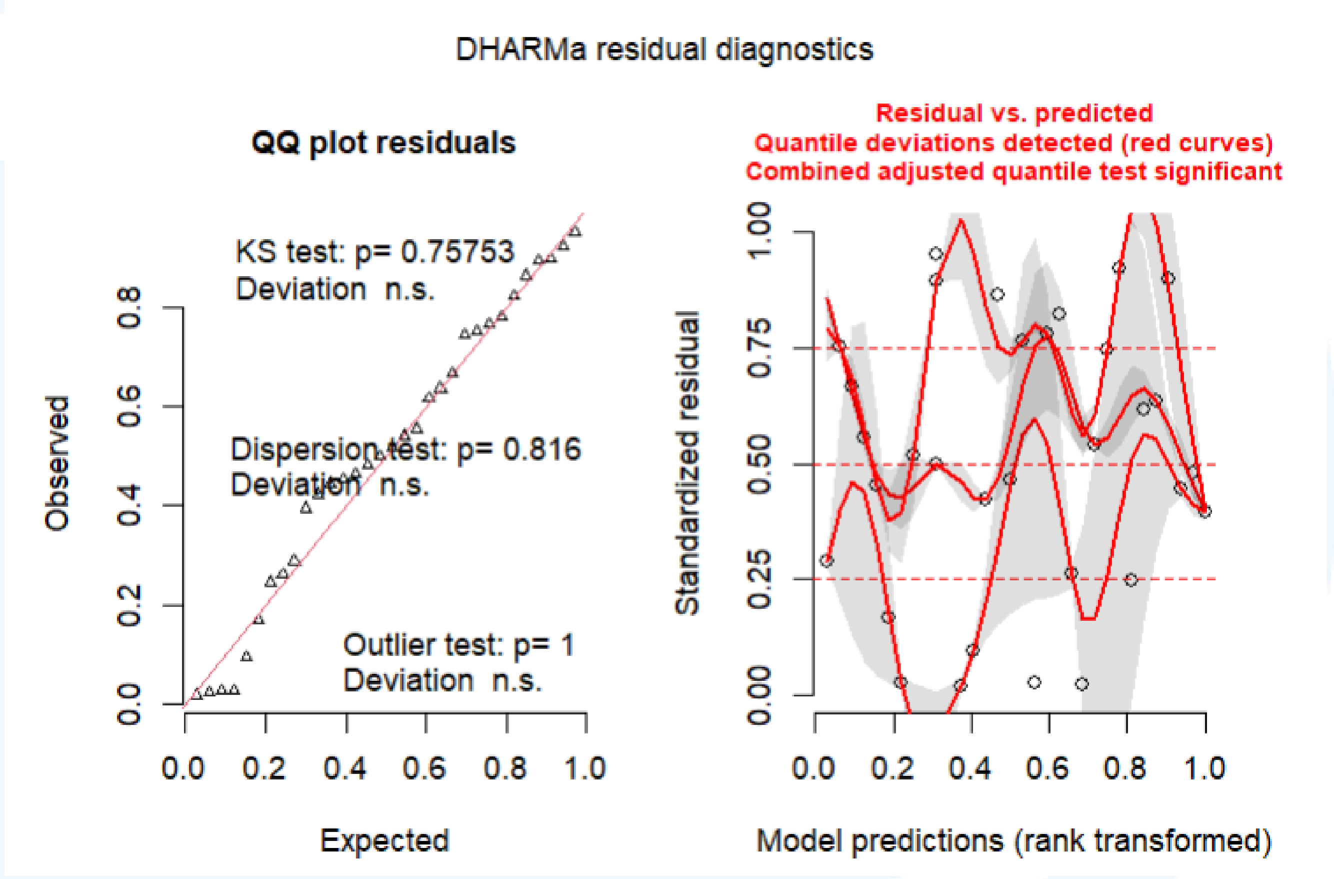
```
> r2_nakagawa(mod2.3)
# R2 for Mixed Models

  Conditional R2: 0.412
     Marginal R2: 0.264
```

Proportion of total variation explained by fixed and random effects combined

Example – model checking

Check the final model:

```
require(DHARMa)
plot(simulateResiduals(mod2.2b))
```



Looks like we still might need to switch to quasipoisson in the end:

```
mod2.3b=glmmPQL(richness~par+bleached,random=~1|category,family=poisson,
data=d5)

summary(mod2.3b)
```

```
Fixed effects:  richness ~ par + bleached
                 Value Std.Error DF  t-value p-value
(Intercept)  0.8523776 0.5175297 27 1.647012  0.1111
par          0.3724809 0.1275029 27 2.921353  0.0070
bleachedTRUE 0.7278017 0.2151191 27 3.383250  0.0022
```

Result is qualitatively the same (i.e. both variables are significant) but effect sizes are slightly different. This gives me more confidence in the results.

A note on multinomial GLMMs

There is no good and easy way to fit multinomial GLMMs as yet.

Option 1: Fit a multinomial GLM, i.e. with everything (including your desired random effects) as fixed effects. Use this to assess whether your “random effects” are important and decide whether or not to keep them in the model.

Option 2: If you really want to do a GLMM, you can try the `npmlt()` function from the “mixcat” package. But it is not yet well established (and slightly compliated).

Dealing with errors

Common errors: “Iteration limit reached”, “Failed to Converge”, “Singular fit”, “Maximum number of evaluations reached”, etc.

- Mostly because you have too many random (or even fixed) effects for your *N*
- Your model is too complex for your *N* (interactions, random slopes, etc.)

Strategies to fix the errors:

1) Gather more data to increase the *N*! =)

2) Simplify your random effects if possible (or fixed effects too, but this is often linked to your research question so less likely).

3) Use a different function from other packages: e.g. glmmADMB, glmmML, MCMCglmm, etc. (Bayesian methods are able to avoid these errors if your prior is meaningful).

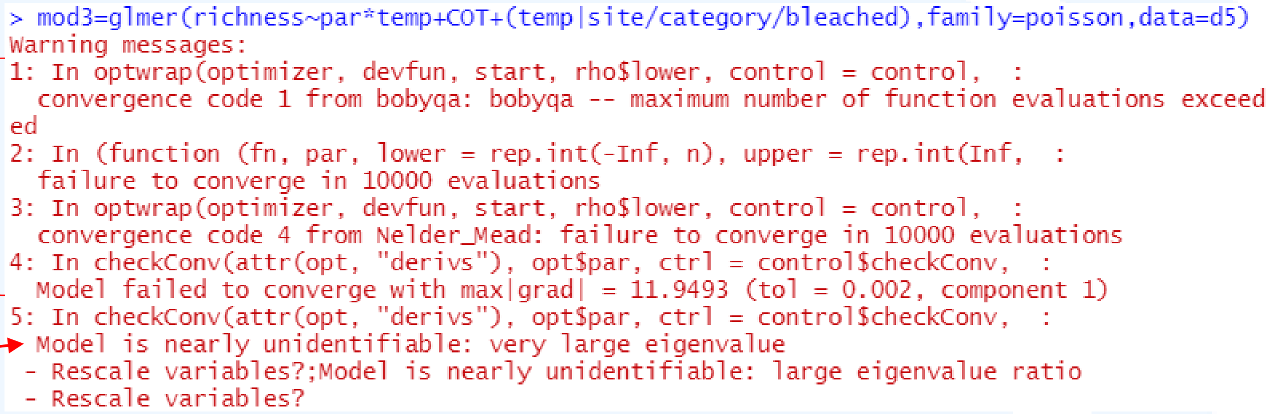
4) Change optimiser settings. These affect things like how many times R repeats its trial and error before stopping, its starting values, the algorithm used, etc.

Dealing with errors – Scaling variables

We are trying to fit this super complex model:

```
mod3=glmer(richness~par*temp+COT+(temp|site/category/bleached),family=poisson
,data=d5)
```

Convergence errors



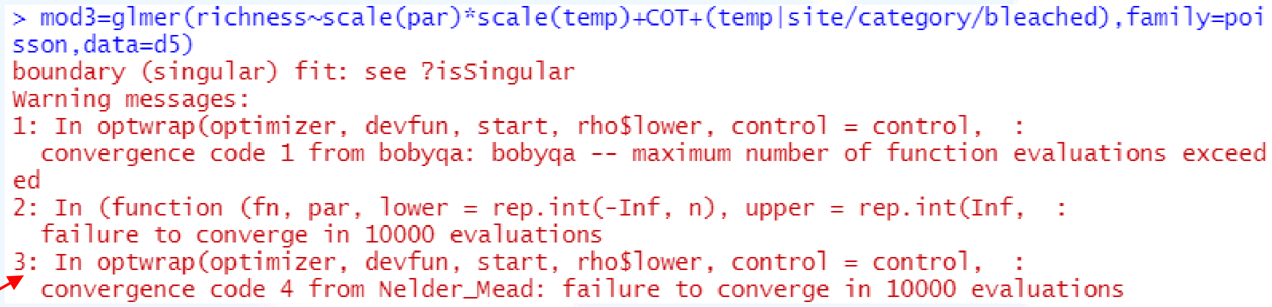
```
> mod3=glmer(richness~par*temp+COT+(temp|site/category/bleached),family=poisson,data=d5)
Warning messages:
1: In optwrap(optimizer, devfun, start, rho$lower, control = control,  :
  convergence code 1 from bobyqa: bobyqa -- maximum number of function evaluations exceed
ed
2: In (function (fn, par, lower = rep.int(-Inf, n), upper = rep.int(Inf,  :
  failure to converge in 10000 evaluations
3: In optwrap(optimizer, devfun, start, rho$lower, control = control,  :
  convergence code 4 from Nelder_Mead: failure to converge in 10000 evaluations
4: In checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv,  :
  Model failed to converge with max|grad| = 11.9493 (tol = 0.002, component 1)
5: In checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv,  :
  Model is nearly unidentifiable: very large eigenvalue
 - Rescale variables?;Model is nearly unidentifiable: large eigenvalue ratio
 - Rescale variables?
```

Calculation anomaly, caused by continuous variables that have very different ranges of values. Here mean <par> is 3.2 and mean <temp> is 28.6: a difference of one order of magnitude

Let's try to fix the (easier) warning about the need to scale variables:

```
mod3.1=glmer(richness~scale(par)*scale(temp)+COT+(temp|site/category/bleached
),family=poisson,data=d5)
```

Command to scale variables: converts variables to a mean of 0 and a S.D. of 1. But careful with interpretation if you do this. You could avoid this step by just simplifying your model.



```
> mod3=glmer(richness~scale(par)*scale(temp)+COT+(temp|site/category/bleached),family=poi
sson,data=d5)
boundary (singular) fit: see ?isSingular
Warning messages:
1: In optwrap(optimizer, devfun, start, rho$lower, control = control,  :
  convergence code 1 from bobyqa: bobyqa -- maximum number of function evaluations exceed
ed
2: In (function (fn, par, lower = rep.int(-Inf, n), upper = rep.int(Inf,  :
  failure to converge in 10000 evaluations
3: In optwrap(optimizer, devfun, start, rho$lower, control = control,  :
  convergence code 4 from Nelder_Mead: failure to converge in 10000 evaluations
```

Solved the Scale warning, but there are still convergence errors

Dealing with errors – Adjusting control settings

Try turning off “calc.derivs”:

extra steps they do when they calculate

```
mod3.2=update(mod3.1,control=glmerControl(calc.derivs=F))
```

Try decreasing tolerance levels (decreases number of trials needed):

```
mod3.3=update(mod3.1,control=glmerControl(optCtrl=list(FtolAbs=1e-10)))
```

the model is not so strictg about the process anymore

Try changing optimizers:

```
mod3.4=update(mod3.1,control=glmerControl(optimizer="bobyqa"))
```

Note: other optimizers include "Nelder_Mead", "optim", "optimx", "nlminbwrap", "nloptwrap", "nmkbw". Some require you to install other packages.

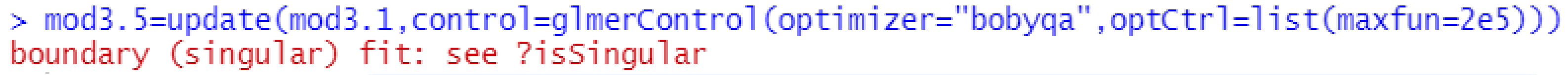
None of these work: there are still convergences errors!

Dealing with errors – Adjusting optimiser settings

This is the maximum number of tries R is allowed to fit a model: it might take some time to run. Some optimizers (e.g. optim and optimx) use “maxit” instead of “maxfun”.

Try allowing more trials:

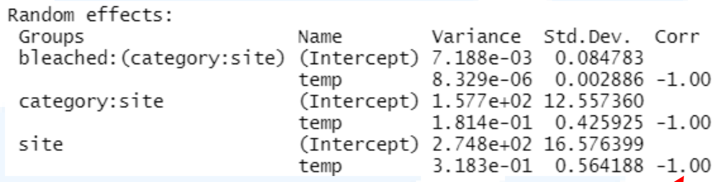
```
mod3.5=update(mod3.1,control=glmerControl(optimizer="bobyqa",
optCtrl=list(maxfun=2e5)))
```



```
> mod3.5=update(mod3.1,control=glmerControl(optimizer="bobyqa",optCtrl=list(maxfun=2e5)))
boundary (singular) fit: see ?isSingular
```

The only error left is “boundary fit”: this is not necessarily bad. It usually means one of your variables is not useful. In this case <temp> followed by bleached>.

mod3.5



```
Random effects:
 Groups                   Name        Variance  Std.Dev.  Corr
 bleached:(category:site) (Intercept) 7.188e-03  0.084783
                          temp        8.329e-06  0.002886 -1.00
 category:site            (Intercept) 1.577e+02 12.557360
                          temp        1.814e-01  0.425925 -1.00
 site                     (Intercept) 2.748e+02 16.576399
                          temp        3.183e-01  0.564188 -1.00
```

```
mod3.6=glmer(richness~scale(par)*scale(temp)+COT+
(1|site/category/bleached),family=poisson,data=d5,
control=glmerControl(optimizer="bobyqa",optCtrl=list(maxfun=2e5)))
```

<temp> removed

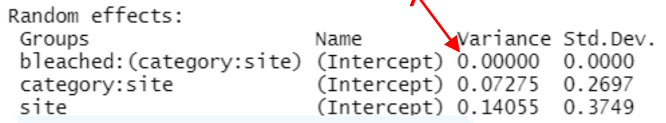
A perfect correlation may mean one of the variables is not needed. It also (normally) doesn’t make sense to have the same variable (<temp>) as both a fixed AND random effect.

```
mod3.7=glmer(richness~scale(par)*scale(temp)+
COT+(1|site/category),family=poisson,data=d5,
control=glmerControl(optimizer="bobyqa",optCtrl=list(maxfun=2e5)))
```

<bleached> removed

<bleached> is not helping to explain any variation

mod3.6



```
Random effects:
 Groups                   Name        Variance Std.Dev.
 bleached:(category:site) (Intercept) 0.00000  0.0000
 category:site            (Intercept) 0.07275  0.2697
 site                     (Intercept) 0.14055  0.3749
```

Fixed! No more errors! Hooray!

Dealing with errors – Adjusting optimiser settings

But note that running this simpler model using default settings also runs fine:

```
mod3.8=glmer(richness~scale(par)*scale(temp)+COT+(1|site/category),
family=poisson,data=d5)
```

So if you can, always try to simplify your random effects first! If this doesn't work THEN try to adjust settings.

For more information, read:

- https://bbolker.github.io/mixedmodels-misc/glmmFAQ.html#convergence-warnings
- Run "?convergence" in R.

Summary (Learning Objectives)

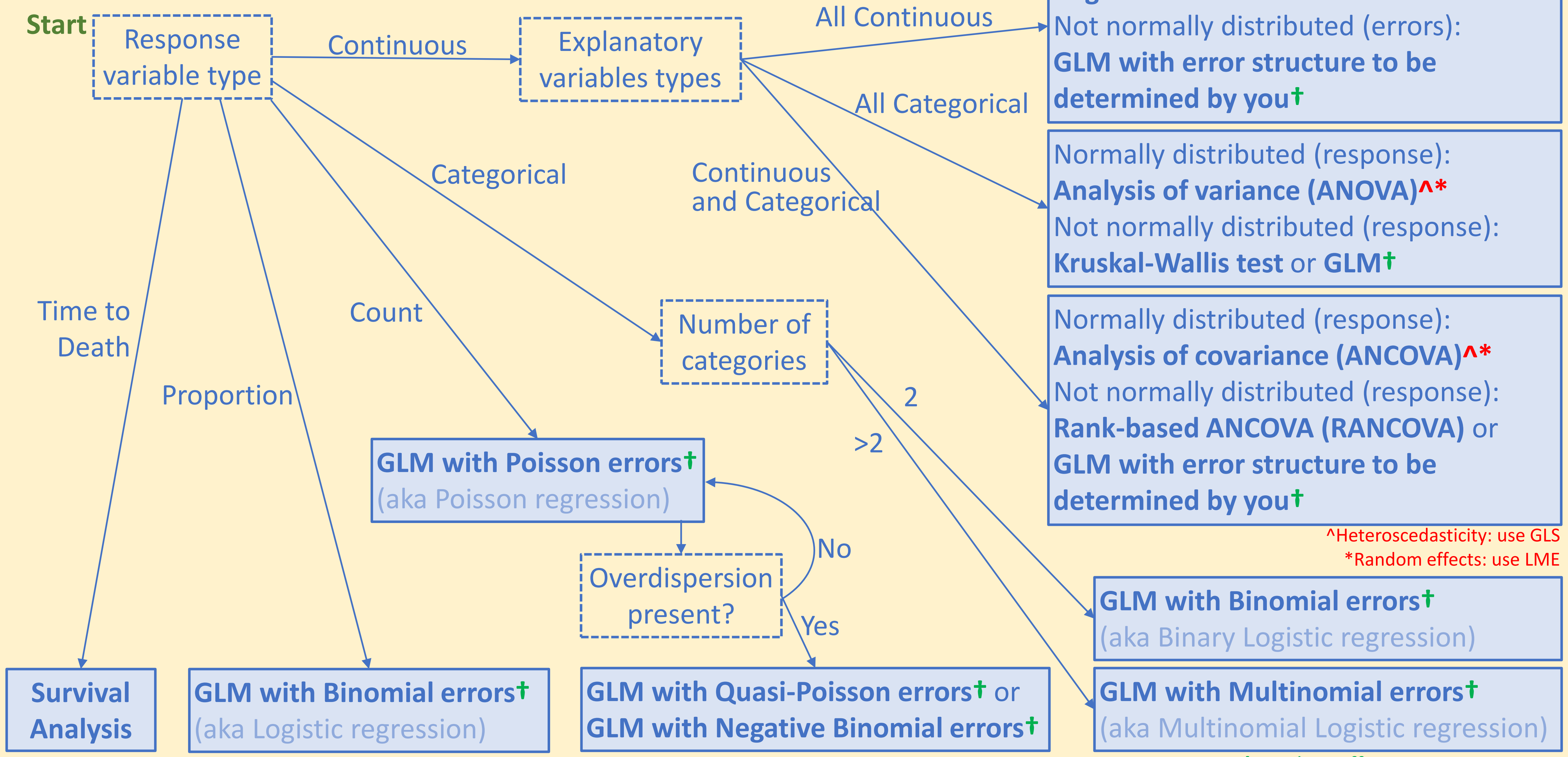
Survival Analysis

- What it does, censoring and hazard functions
- Fitting, simplifying and interpreting: Gamma glm() and **survreg()**
- Plotting Kaplan-Meier curves using survfit() and autoplot()

Generalised Linear Mixed Models

- What is a GLMM?: error families, random errors and estimation methods
- Functions: **glmer()**, glmmPQL() and stan_glmer()
- Fitting and simplifying
- Solving errors

Advanced analyses – Analysis decision tree (Final Version)


Start
Response variable type
Continuous
Explanatory variables types
All Continuous
Normally distributed (errors): Regression^*
Not normally distributed (errors): GLM with error structure to be determined by you†
All Categorical
Normally distributed (response): Analysis of variance (ANOVA)^*
Not normally distributed (response): Kruskal-Wallis test or GLM†
Continuous and Categorical
Normally distributed (response): Analysis of covariance (ANCOVA)^*
Not normally distributed (response): Rank-based ANCOVA (RANCOVA) or GLM with error structure to be determined by you†
Categorical
Number of categories
2
GLM with Binomial errors† (aka Binary Logistic regression)
>2
GLM with Multinomial errors† (aka Multinomial Logistic regression)
Count
GLM with Poisson errors† (aka Poisson regression)
Overdispersion present?
No
Yes
GLM with Quasi-Poisson errors† or GLM with Negative Binomial errors†
Proportion
GLM with Binomial errors† (aka Logistic regression)
Time to Death
Survival Analysis

^Heteroscedasticity: use GLS

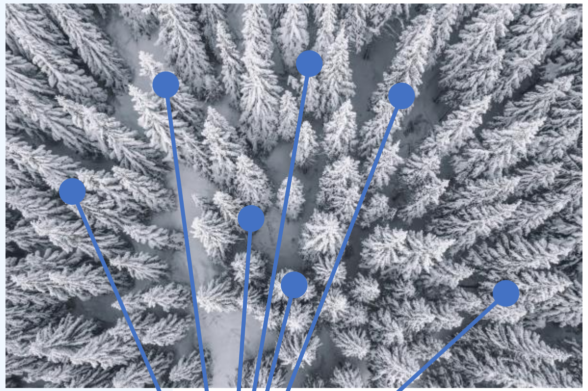
*Random effects: use LME

†Random Effects: use GLMM.

Teaser for next week…

To compare the biodiversity in Forest 1 vs. Forest 2,

Forest 1



we have 4 response variables…

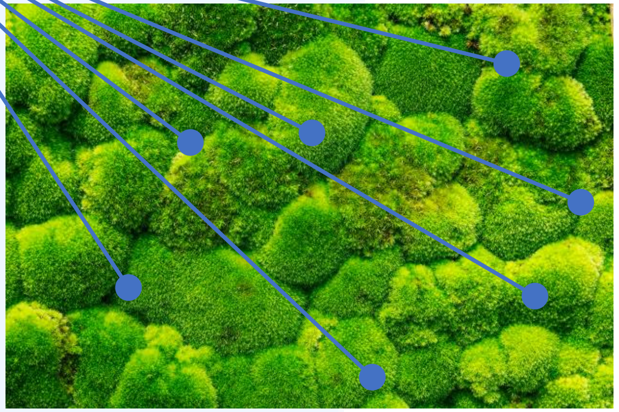
Biodiversity

	Sp. 1	Sp. 2	Sp. 3	Sp. 4
Site 1	0	0	22	4
Site 2	11	4	12	13
Site 3	1	1	0	14
Site 4	0	0	0	19
Site 5	2	5	7	2
Site 6	8	8	11	17

vs.

Biodiversity

	Sp. 1	Sp. 2	Sp. 3	Sp. 4
Site 1	7	8	10	0
Site 2	6	9	2	3
Site 3	0	0	11	1
Site 4	1	13	0	9
Site 5	3	5	6	1
Site 6	12	2	8	0



Forest 2

We're going to need **Multivariate Analyses**!