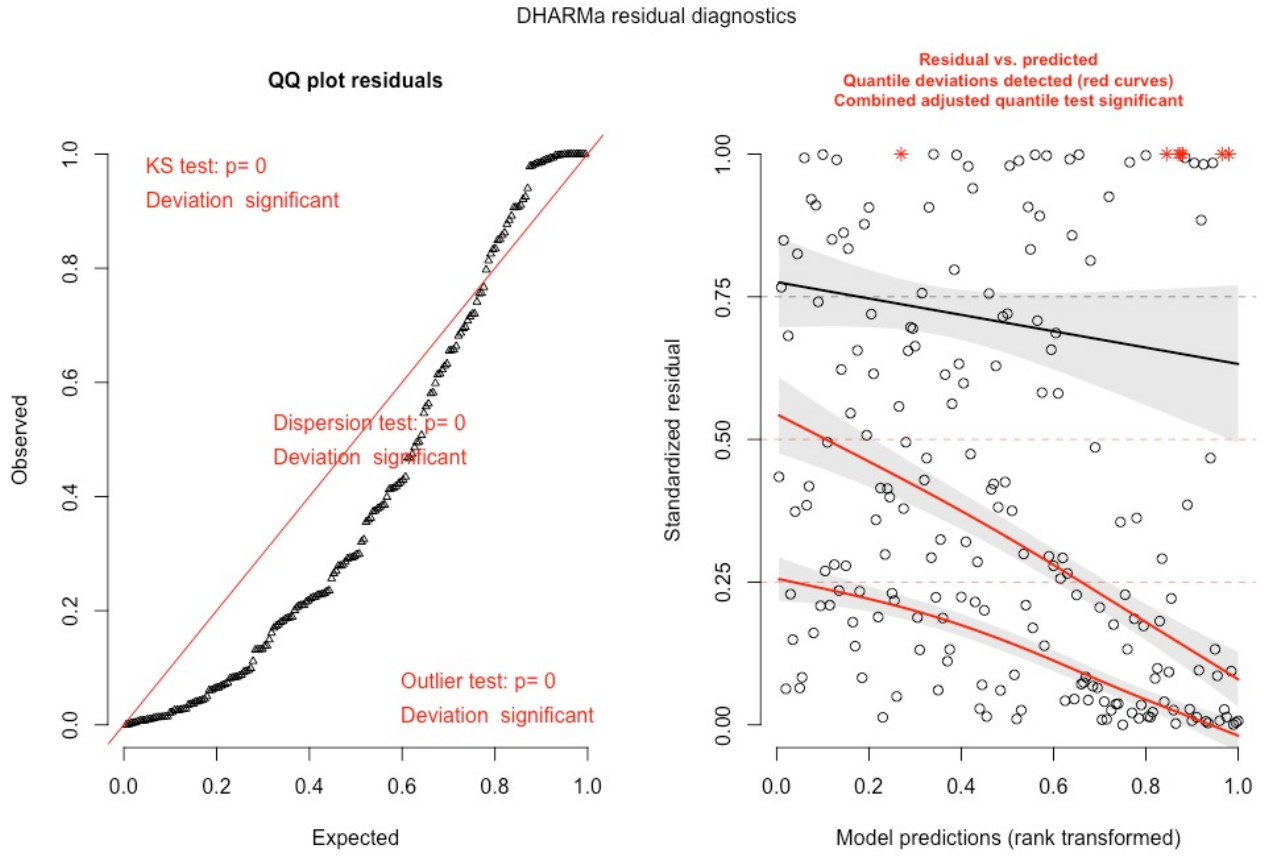
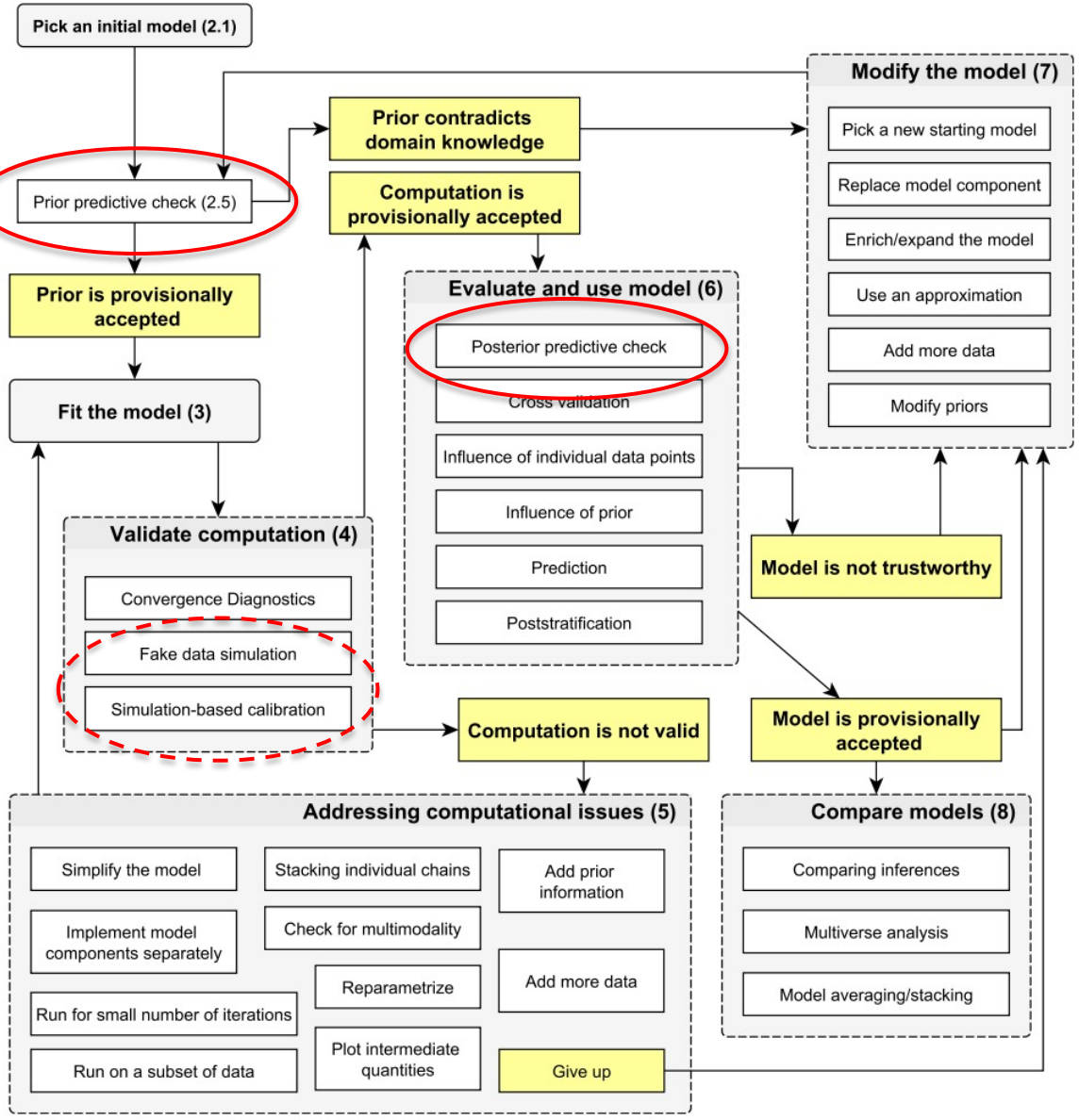


FLORIAN HARTIG, U. REGENSBURG

BAYESIAN MODEL CRITICISM

Twitter: @florianhartig

Bayesian Workflow following Gelman et al.



Purpose of these two checks: evaluate the validity of the assumptions (model structure + prior) on which we condition our (Bayesian) inference

From: Gelman, Andrew, et al. "Bayesian workflow." *arXiv preprint arXiv:2011.01808* (2020).

Why should we check our assumptions?

Bayesian inference calculates the posterior ***conditionally*** on prior + likelihood being being true (which, of course, is not guaranteed).

If assumptions hold

- We have certain mathematical guarantees for the quality of the inference
- E.g. about frequency of errors, coverage, consistency of estimators etc.

If assumptions != ”truth”

- No guarantees that the inference is correct
- However, also no guarantee that the inference is incorrect!
- → risky, but not necessarily wrong, worth checking on a case-by-case basis!

Model checking and its "position" in the Bayesian philosophy

Prior predictive check = is the prior correct / reasonable?
Posterior predictive check = is the model structure correct?

Bayesian "purist"

- The prior is your current knowledge, how can you "reject" your knowledge?
- Rejecting is a frequentist technique, we should speak about posterior probabilities
- Instead of conditioning on assuptions, should have defined all possible models *a priori* and calculate posterior weights

Bayesian "pragmatist"

- People often make mistakes when specifying priors
- Should we just keep on using a (wrong)linear model, just because we didn't think about a quadratic effect when we started?
- Considering all possible models is impractical

→ We can pick up these arguments later when in our discussion, let's look how these checks work first

General technique for prior / posterior predictive checks

Repeat:

1. Simulate a parameter combination from the prior / posterior
2. Simulate along the model structure (= likelihood) to create new (synthetic) data or predictions

Based on the simulated predictions / data, explore the plausibility of your prior / posterior model assumptions

Note: in principle, we could also check analytically, but in practical applications (hierarchical models), it's usually not possible to forward pdfs analytically, thus in practice, we usually always work with simulations!

Prior predictive checks

Prior predictive checks

- Generate data simulations from prior + model (likelihood)
  - Keep in mind: although issues in prior predictive checks are usually attributed to the prior, it could also be the model

- **Example A:** we have sharp informative priors for all parameters, but model predictions are implausible
  - If we trust the priors → structural / model error
  - If we trust the model → priors must be wrong

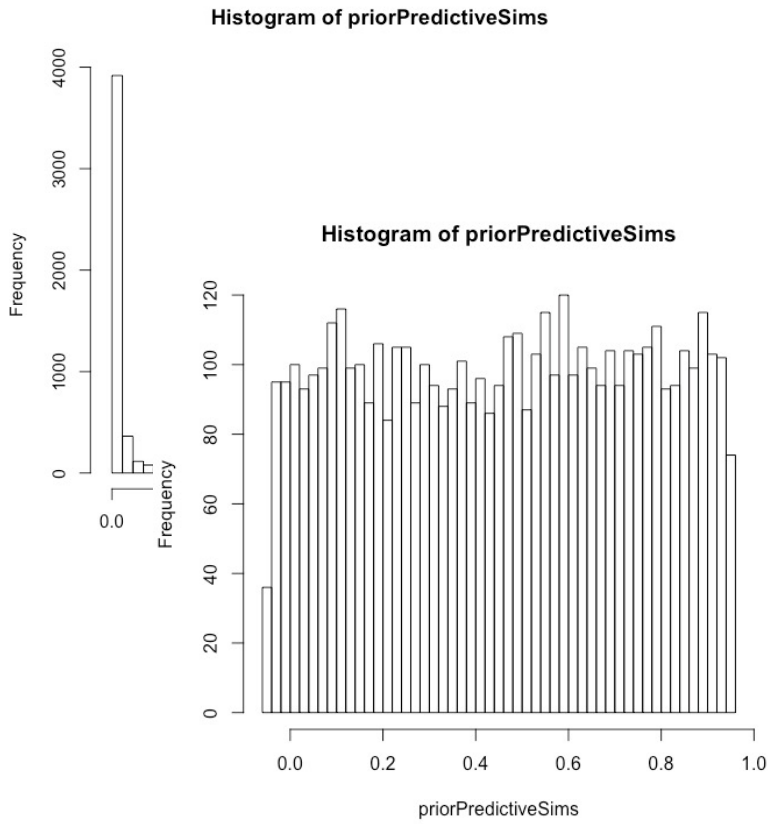
- **Example B:** we thought we had specified vague / uninformative priors, but predictions are sharply peaked at (maybe implausible) values
  - We may not have specified a truly uninformative prior for the respective model

Specifying uninformative priors in complex models is hard!

```r
# Strongly nonlinear model
model = function(x) exp(-abs(x)) + rnorm(1,sd =
0.05)

# wide uniform prior leads to a very "biased"
prior predictive distribution
priorDraws = rnorm(5000, sd = 20)
priorPredictiveSims = model(priorDraws)
hist(priorPredictiveSims, breaks = 50)

# an exponential prior, which concentrates mass
around 0 (transformed by the model to large
values) is arguably far more "neutral"
priorDraws = rexp(5000) * sample(c(-1,1), 5000,
T)
priorPredictiveSims = model(priorDraws)
hist(priorPredictiveSims, breaks = 50)
```



What do we learn? a ”good” uninformative prior is not necessarily flat! → old debate, see Jeffrey’s prior

Two more import checks based on prior simulations

→Fit posteriors to all prior-simulated data

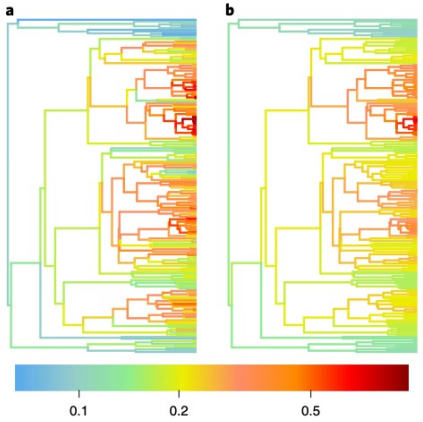
Calibration checks

- Theorem: mixture of all posteriors should be equal to the prior
  - To check if the MCMC implementation is exact
  - Implementation and references BayesianTools::calibrationTest( )

Bias / estimator checks

- Check if you can retrieve the parameters from which you simulated (bias, error coverage)
  - Note: estimators based on Bayesian inference are in general not unbiased, even with uninformative priors.

Example



nature ecology & evolution

ARTICLES

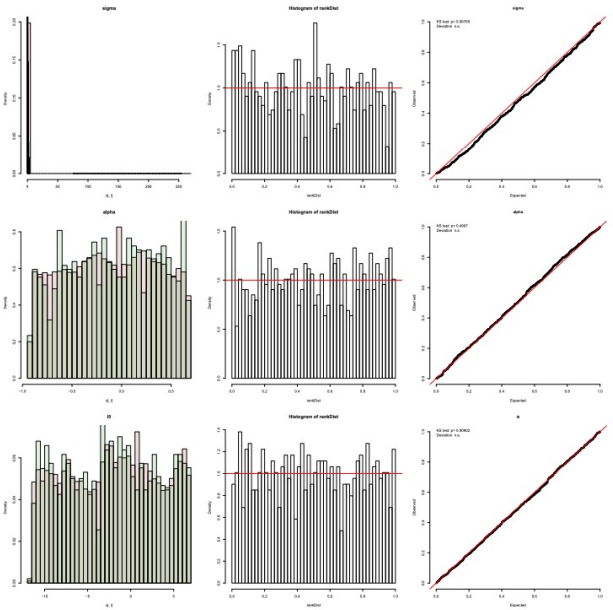
https://doi.org/10.1038/s41559-019-0908-0

A model with many small shifts for estimating species-specific diversification rates

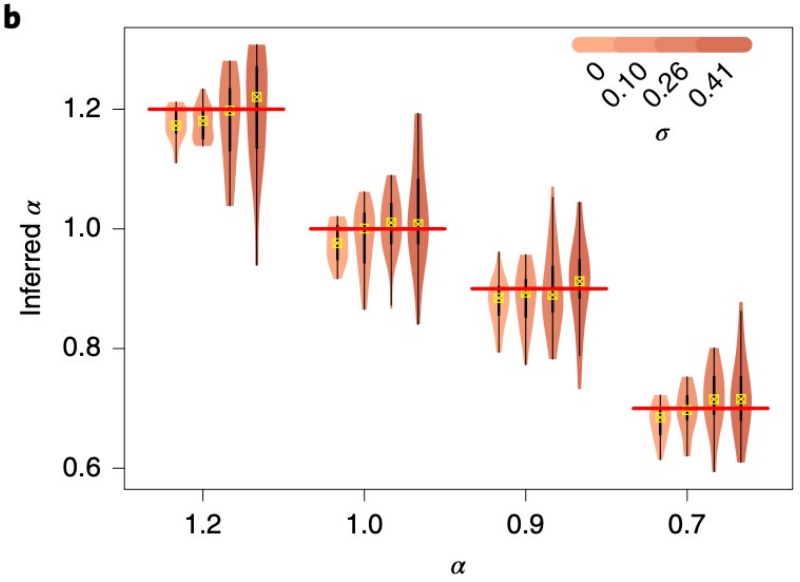
Odile Maliet[1]*, Florian Hartig[2] and Hélène Morlon[1]

Understanding how and why diversification rates vary through time and space and across species groups is key to understanding the emergence of today's biodiversity. Phylogenetic approaches aimed at identifying variations in diversification rates during the evolutionary history of clades have focused on exceptional shifts subtending evolutionary radiations. While such shifts

Calibration tests (Fig. S9)



Test for bias (Fig. 2)



Posterior predictive checks

Goal: see if the estimated model is compatible with the data

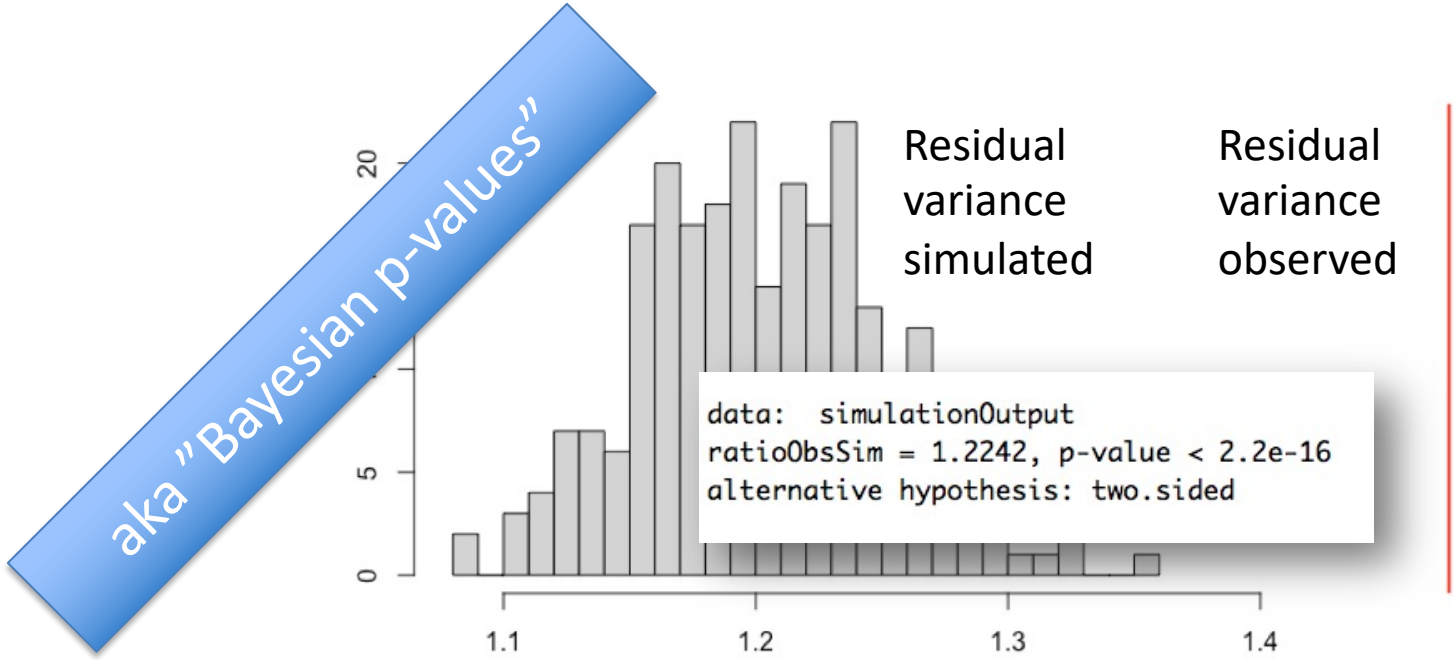
- **Bayesians:**
  - **Posterior predictive checks** = check if simulated data from the fitted model (**posterior**) is similar to the observed data

- **Frequentists:**
  - The same is done in frequentist stats (simulations **around the MLE** aka the **parametric bootstrap**)

- For either Bayesian / Frequentists, several approaches to compare observed / simulated data, which are implemented in the **DHARMa R package**
  - **Global p-values** based on **omnibus GOF tests** – test if observed / simulated are "similar" in some general sense (distribution) or tests for **specific problems** (e.g. zero-inflation)
  - **Generalized residuals** = so-called (randomized) quantile residuals based on simulated data

"global" p-values

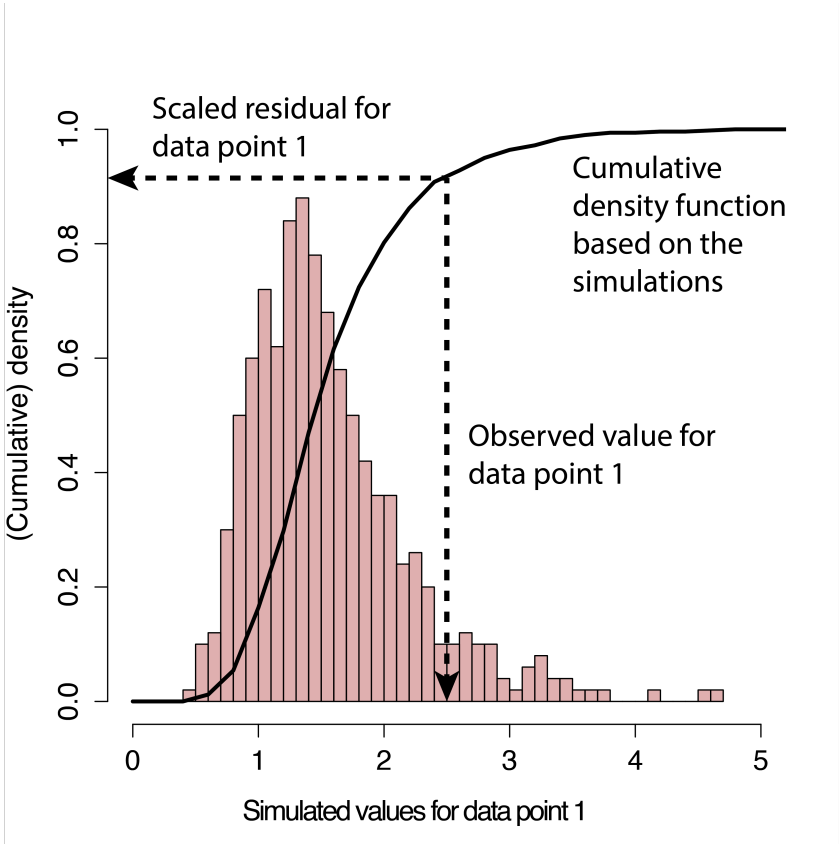
- Calculate p-values for "global" summary statistics of the observed / simulated data. Examples:
  - **Zero-inflation test** - calculate simulated number of zeros vs. observed number of zeros
  - **Dispersion test** - calculate simulated vs. observed variance around model predictions



Option 2: calculate p-values per observation (=quantile residuals)

- Goal: measure how far each data point deviates from the expected distribution

- Idea: express this in terms of the **cumulative distribution of the simulated data / residuals** → standardizes residual to [0,1]



Translation: residual = p(x >= X0), X0 = null distribution from the fitted model

Key property for quantile residuals

- Each residual [0,1] is essentially a **p-value: p(x >= X0)**
  - Side note: for discrete distribution, it is essential to add some additional noise on x and X to make the ecdf smooth → randomized quantile residuals (Dunn & Smyth, 1996)

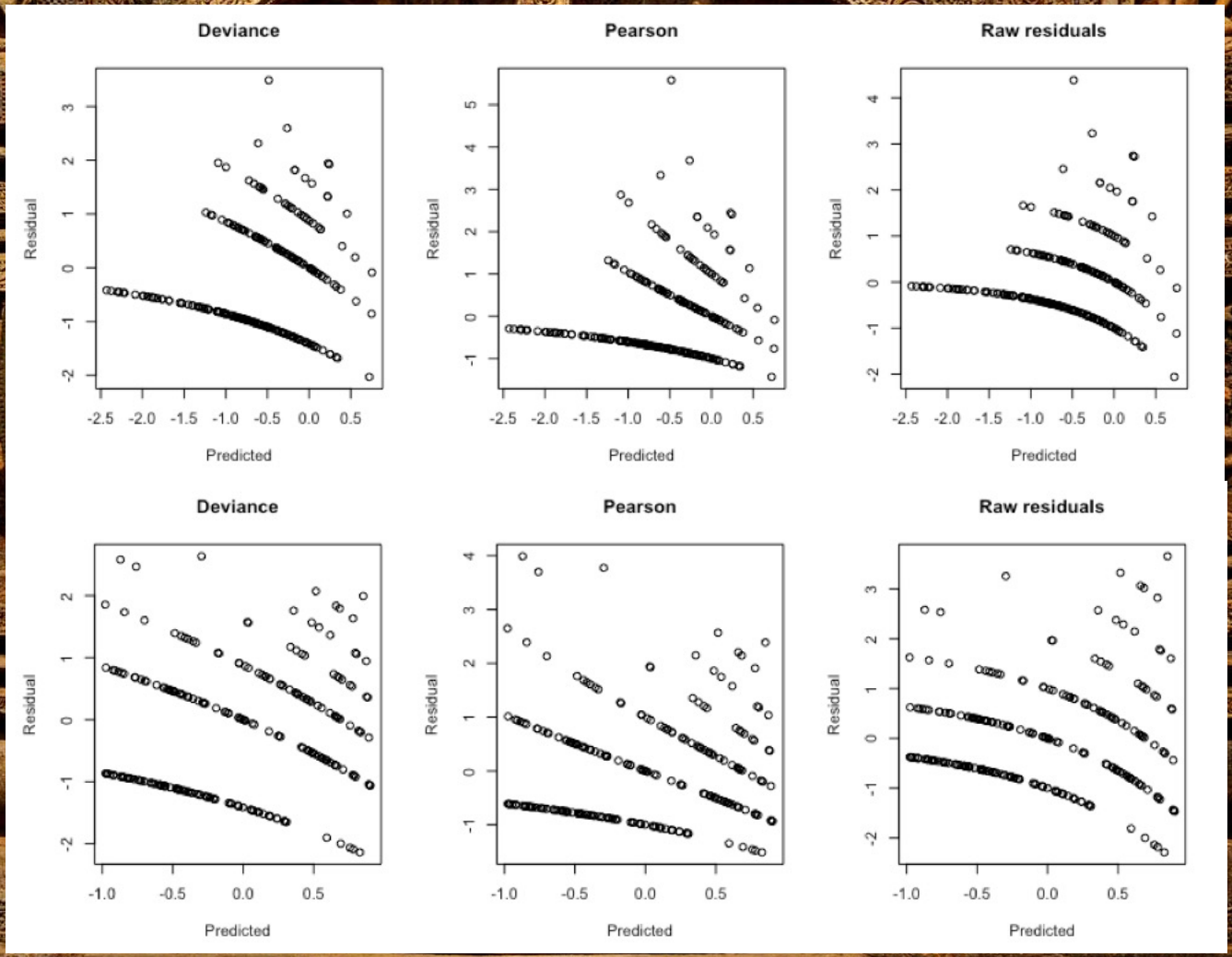
- Thus: if the fitted model is correct (H0), the residual distribution **p(x >= X0) should be uniform**

- Consequence: for **ANY** hierarchical model structure, **if the model is correct** in structure and parameters, **residuals should be uniform**!

Why do we not just calculate “normal” residuals?

Model 1

Model 2



Issues in interpreting residuals for GLMMs + beyond

- GLMM distributions are typically **asymmetric** and **change their shape with the mean** – won't be transformed away by simply dividing through expected sd (Pearson)

- Problems get worse for more complicated GLMMs and hierarchical models, where the **effective distribution of the residuals arises from a mix of distributions / random effects**

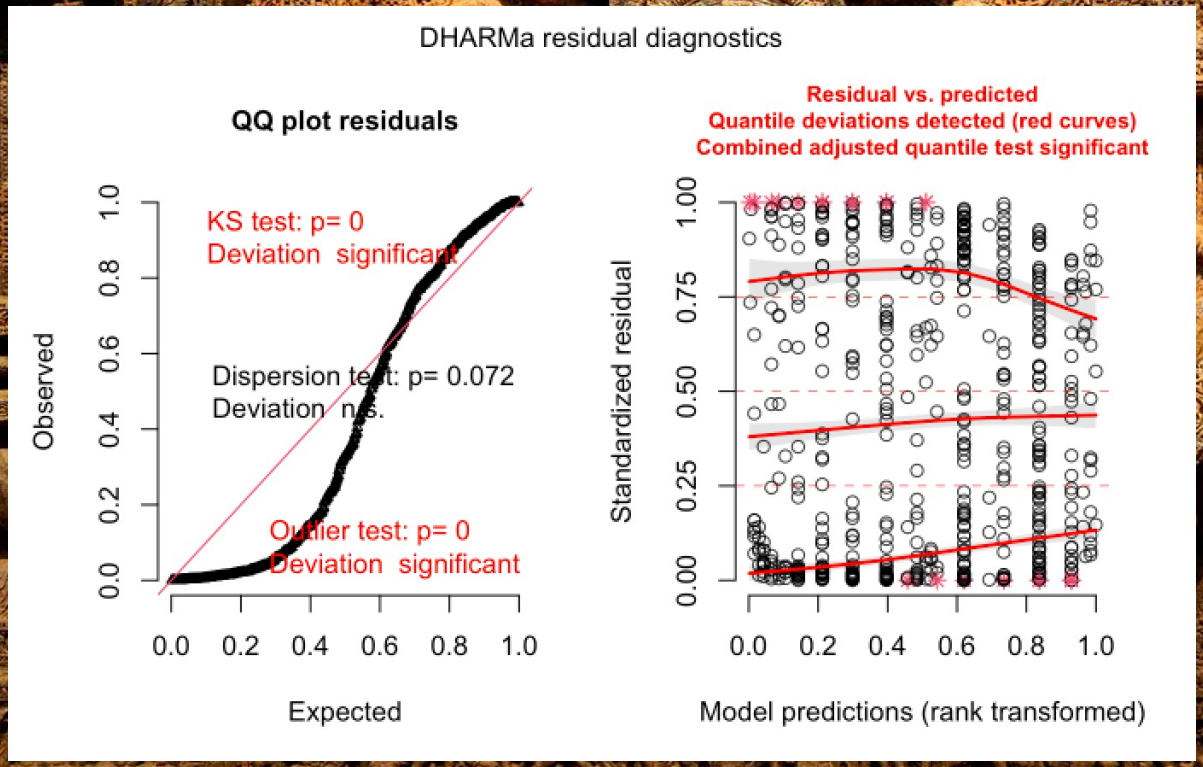
- Consequence: GLMMs(+) used to be rarely checked in practice, although they can have all the same problems we teach for LMs (e.g. misfit, heteroskedasticity, outliers, …)

- → Therefore we use simulation-based checks! And the **DHARMa package** simplifies this Problem for Bayesians and Frequentists

Teaser: example workflow in DHARMa

```
library(lme4)
m2 <- glmer(SiblingNegotiation ~ FoodTreatment*SexParent +
offset(log(BroodSize)) + (1|Nest), data=Owls , family = poisson)
```

```
library(DHARMa)
res <- simulateResiduals(m2)
plot(res)
```



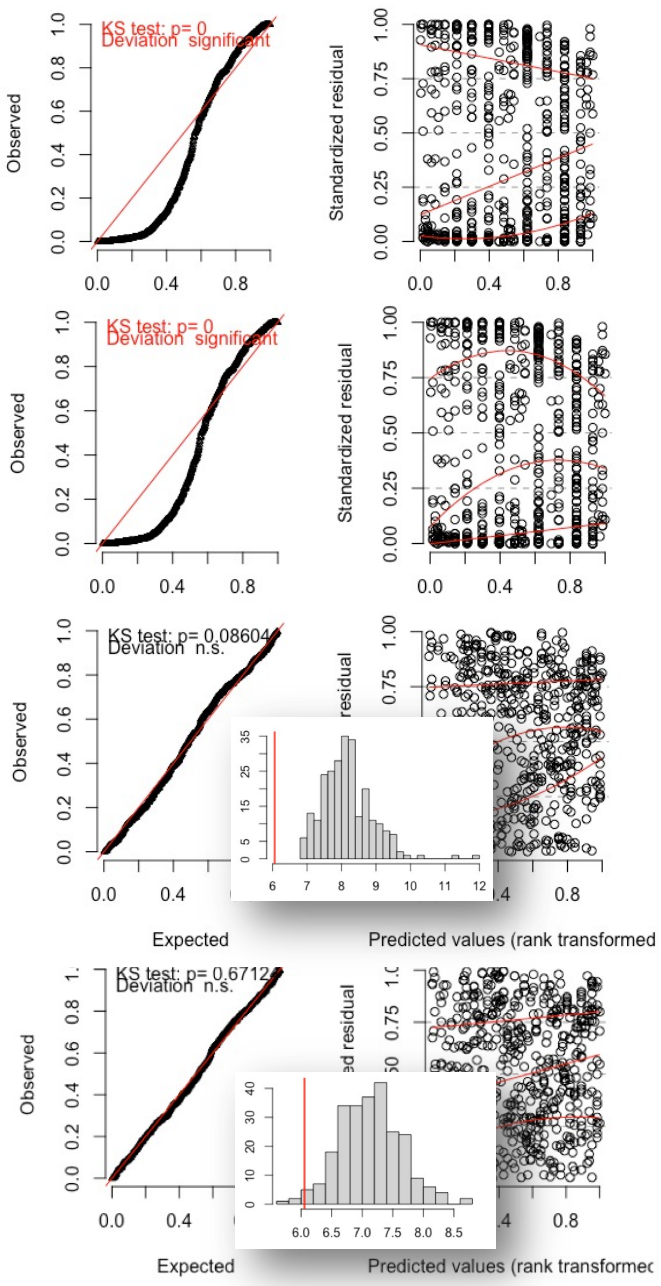
DHARMa directly implements this idea for a wide range of frequentist models

```
m1 <- glm(SiblingNegotiation ~ FoodTreatment*SexParent +
offset(log(BroodSize)), data=Owls , family = poisson)
```

```
m2 <- glmer(SiblingNegotiation ~ FoodTreatment*SexParent +
offset(log(BroodSize)) + (1|Nest), data=Owls , family =
poisson)
```

```
m3 <- glmmTMB(SiblingNegotiation ~ FoodTreatment*SexParent +
offset(log(BroodSize)) + (1|Nest), data=Owls , family =
nbinom1)
```

```
m4 <- glmmTMB(SiblingNegotiation ~ FoodTreatment*SexParent +
offset(log(BroodSize)) + (1|Nest), ziformula = ~
FoodTreatment + SexParent,  data=Owls , family = nbinom1)
```



For Bayesians working with DHARMa, plots and tests are identical, just that we have to create the simulations by hand

How to do this in Jags – in principle

```
# Likelihood
for (i in 1:nobs) {
  lambda[i] <- exp(intercept + alt * altitude[i] + alt2 * altitude[i] * altitude[i])
  beetles[i]~dpois(lambda[i])
}

# Fixed effect priors
intercept ~ dnorm(0,0.0001)
alt ~ dnorm(0,0.0001)
alt2 ~ dnorm(0,0.0001)

# Posterior predictive simulations
for (i in 1:nobs) {
  beetlesPred[i]~dpois(lambda[i])
}
```

- Copy the likelihood, but don't connect to data – response will be simulated
  - More details later
- Read these simulations into DHARMa (recommended), or analyze by hand!

Interpreting quantile residuals in DHARMa (here: good fit)

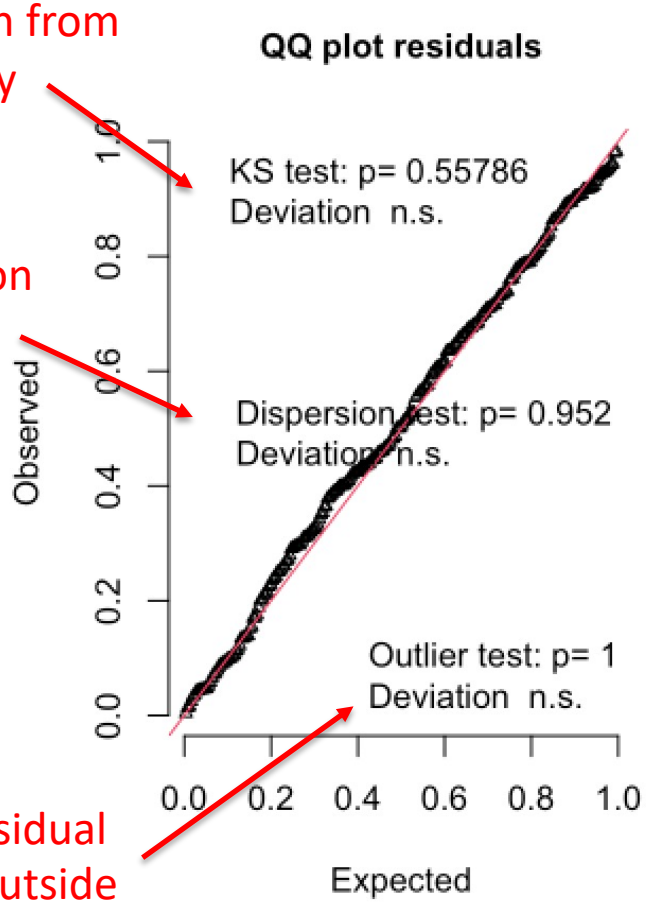
"global" Distribution

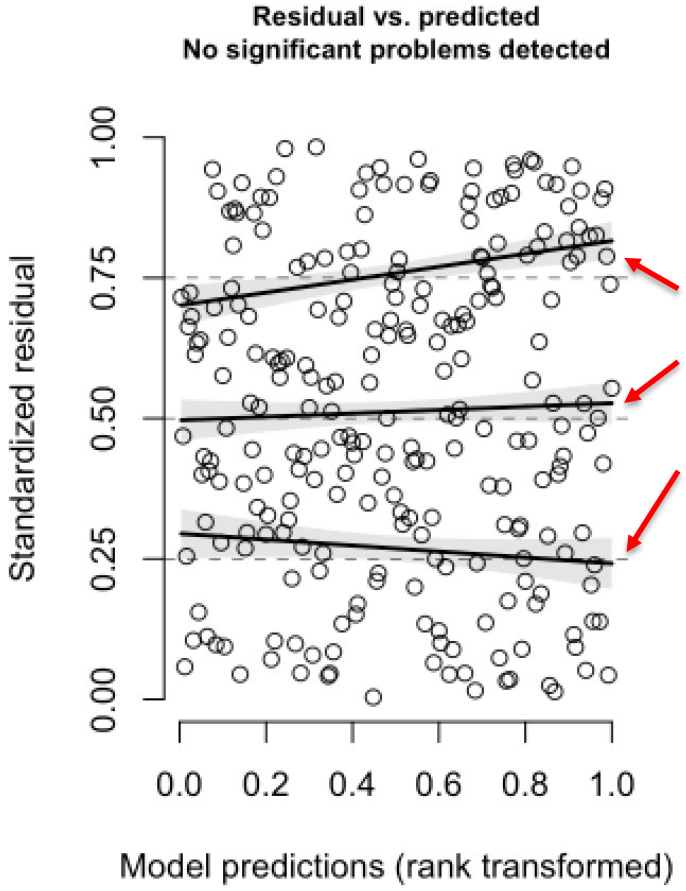
Residuals ~ predicted value

KS test for Deviation from normality

Dispersion test

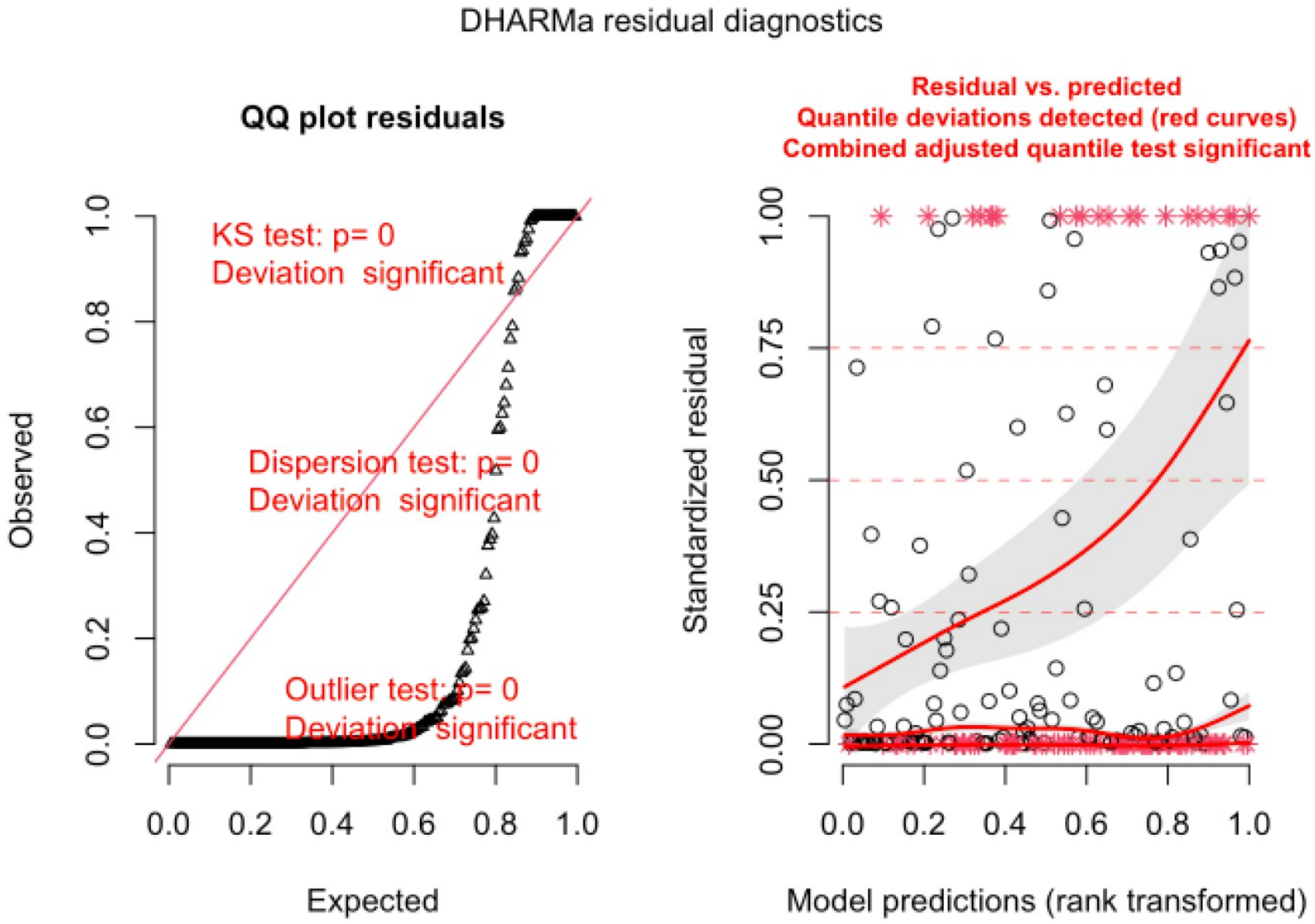
Prop. residual values outside simulation envelope



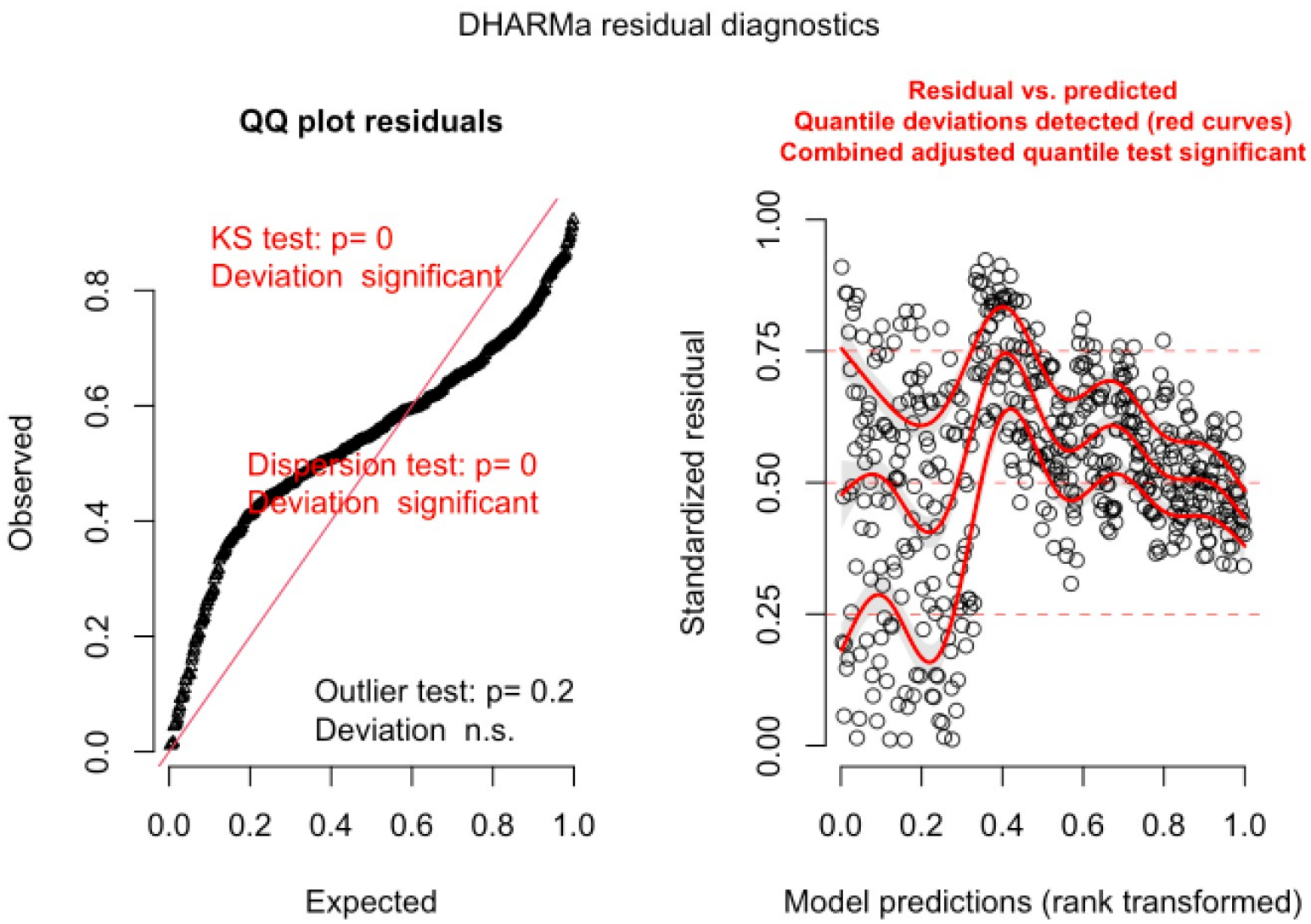


Quantile Regressions For 0.25, 0.5, 0.75

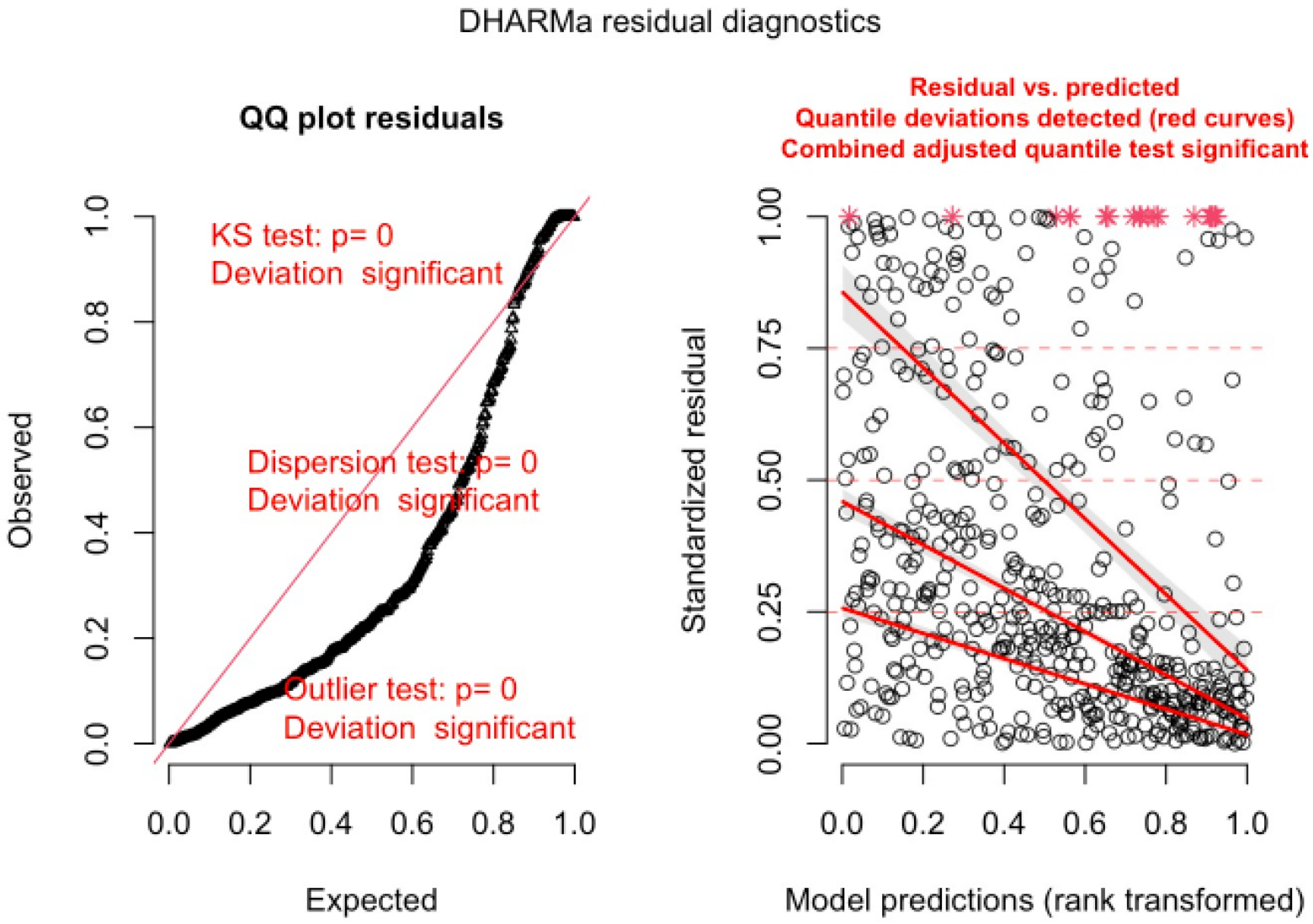
Overdispersion



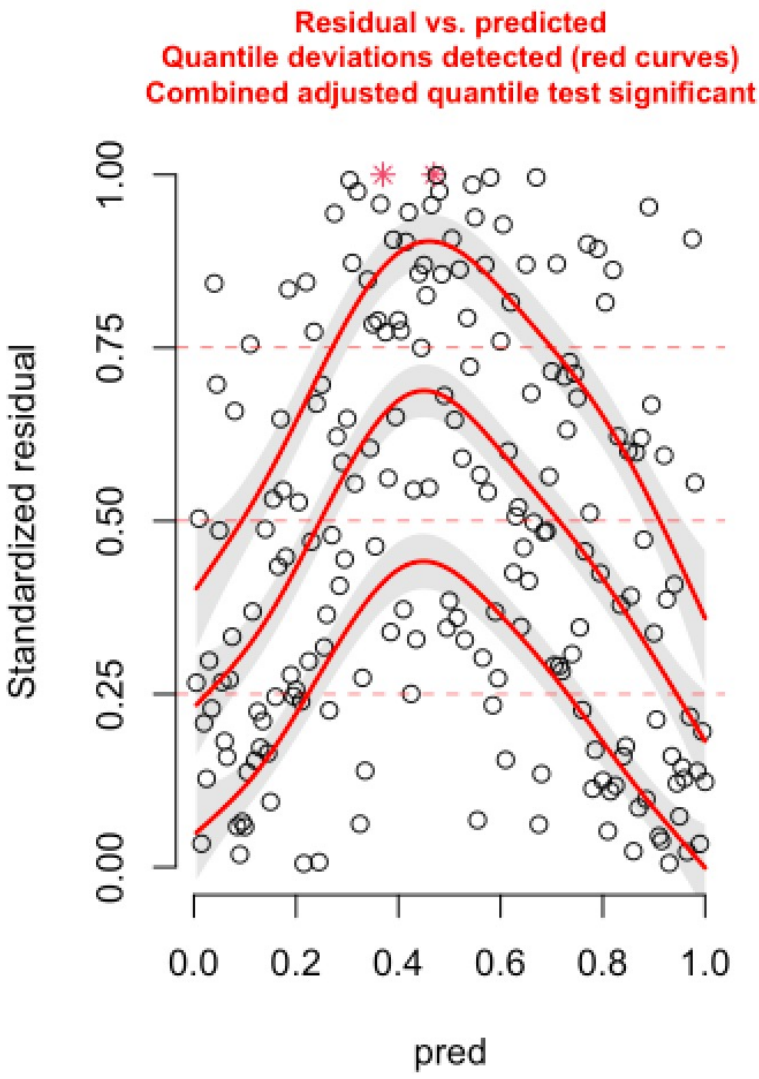
Underdispersion

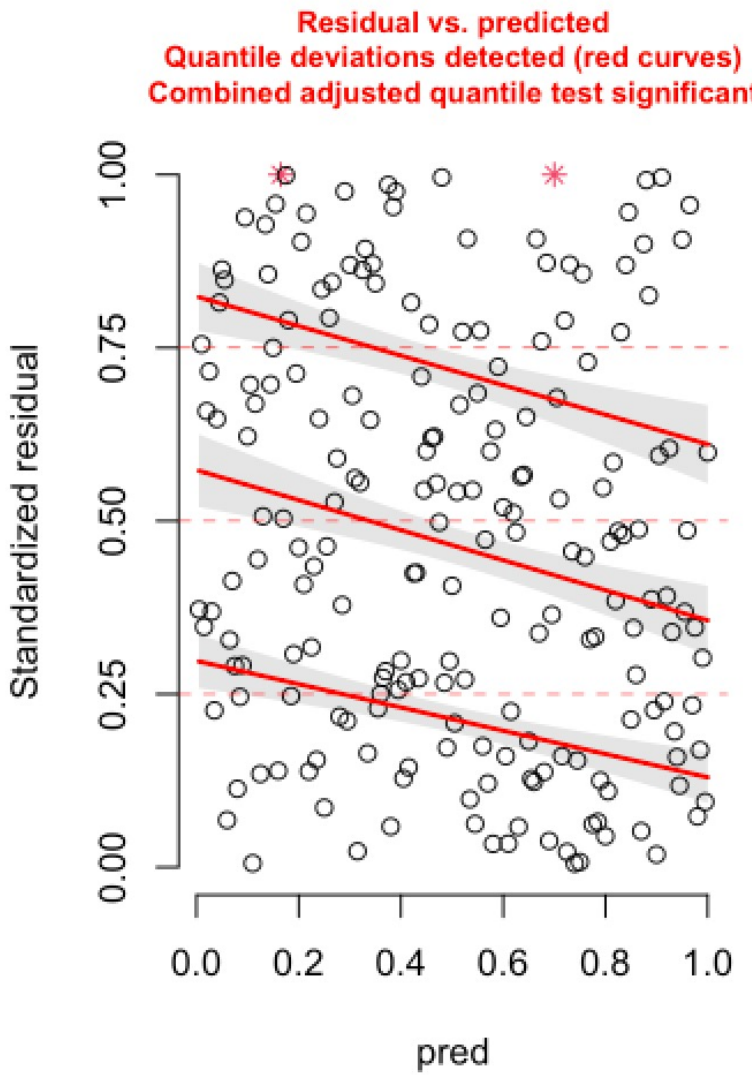


Heteroskedasticity

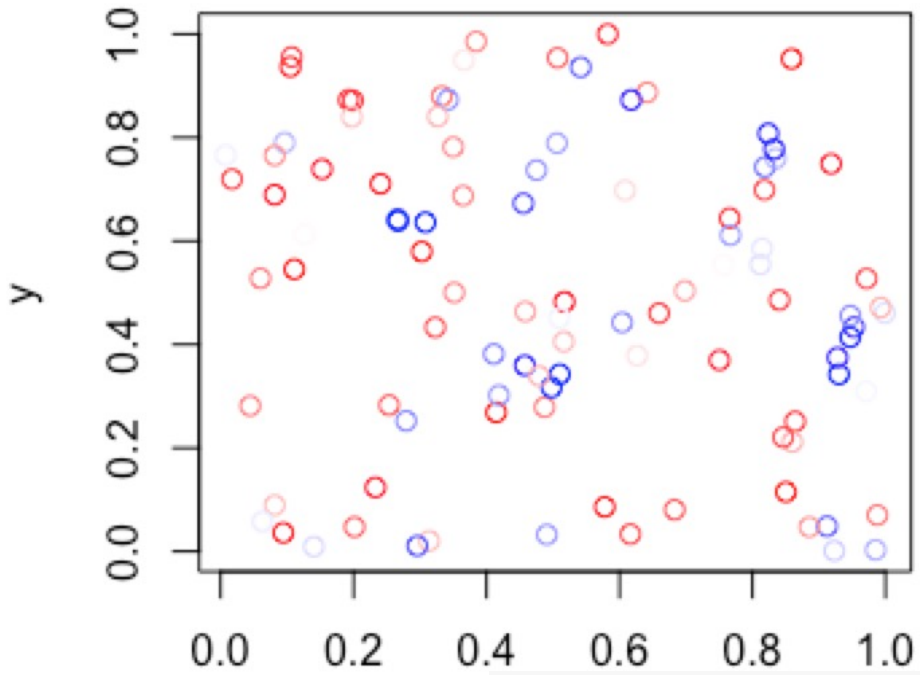


Missing predictor





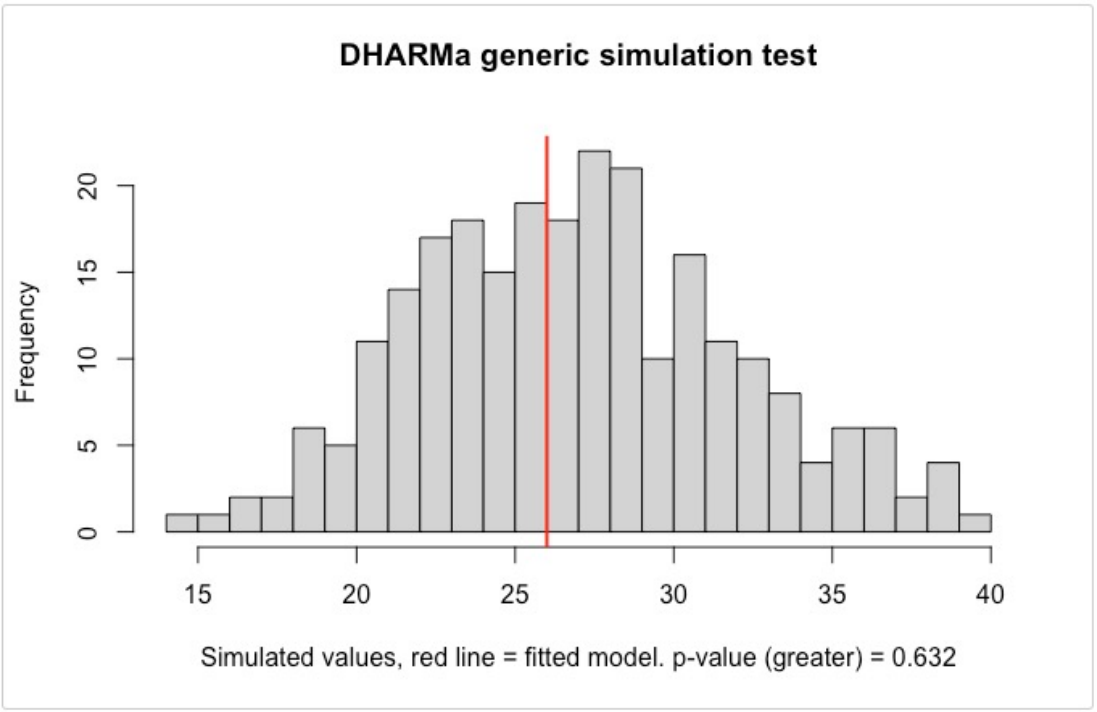
Residual spatial / temporal autocorrelation




```
##
##  DHARMa Moran's I test for spatial autocorrelation
##
## data:  simulationOutput
## observed = 0.073698, expected = -0.010101, sd = 0.022155, p-value
## = 0.0001553
## alternative hypothesis: Spatial autocorrelation
```

DHARMa also supports comparisons of generic summary statistics

```
countOnes <- function(x) sum(x == 1)  # testing for number of 1s
testGeneric(simulationOutput, summary = countOnes, alternative = "greater") # 1-inflation
```




```
##
##  DHARMa generic simulation test
##
## data:  simulationOutput
## ratioObsSim = 0.94807, p-value = 0.632
## alternative hypothesis: greater
```

- In this example, we compare the number of ”1” values simulated / observed
- Again, in Bayesian analysis, these kind of calculations are known as Bayesian p-values

Demo?

More examples / comments on residual interpretation in https://cran.r-project.org/web/packages/DHARMa/vignettes/DHARMa.html

Practical details for Bayesians

```
modelCode = "model{

  for(i in 1:nobs){
    observedResponse[i] ~ dpois(lambda[i])  # poisson error distribution
    lambda[i] <- exp(eta[i]) # inverse link function
    eta[i] <- intercept + env*Environment1[i]  # linear predictor
  }

  intercept ~ dnorm(0,0.0001)
  env ~ dnorm(0,0.0001)

  # Posterior predictive simulations
  for (i in 1:nobs) {
    observedResponsePred[i]~dpois(lambda[i])
    lambda2[i] <- exp(intercept + env*Environment1[i])
  }

}"

jagsModel <- jags.model(file= textConnection(modelCode), data=Data, n.chains = 3)
para.names <- c("intercept","env", "lambda2", "observedResponsePred")
Samples <- coda.samples(jagsModel, variable.names = para.names, n.iter = 5000)

x = BayesianTools::getSample(Samples)

colnames(x) # problem: all the variables are in one array - this is better in STAN, where
this is a list - have to extract the right columns by hand
posteriorPredDistr = x[,3:202] # this is the uncertainty of the mean prediction (lambda)
posteriorPredSim = x[,203:402] # these are the simulations

sim = createDHARMa(simulatedResponse = t(posteriorPredSim),
                   observedResponse = dat$observedResponse,
                   fittedPredictedResponse = apply(posteriorPredDistr, 2, median),
                   integerResponse = T)
plot(sim)
```

Likelihood

Prior

Posterior simulations

Reformating MCMC output

Covert to DHARMa object

Mixed model → conditional or unconditional simulations

```
  # Posterior predictive simulations
  for(i in 1:nobs){
    # conditional simulations, use lambda from the model
    observedResponsePred[i]~dpois(lambda[i])

    # unconditional simulations, use new lambda with new RE
    observedResponsePred2[i] ~ dpois(lambda2[i])  # poisson
error distribution
    lambda2[i] <- exp(eta2[i])
    eta2[i] <- intercept + env*Environment1[i] + RE2[group[i]]


    # unconditional predictions (conditional predictions use
normal lambda)
    lambda3[i] <- exp(intercept + env*Environment1[i])
  }
```

Conditional simulations, what we had before, only re-simulate Poisson conditional on fitted REs

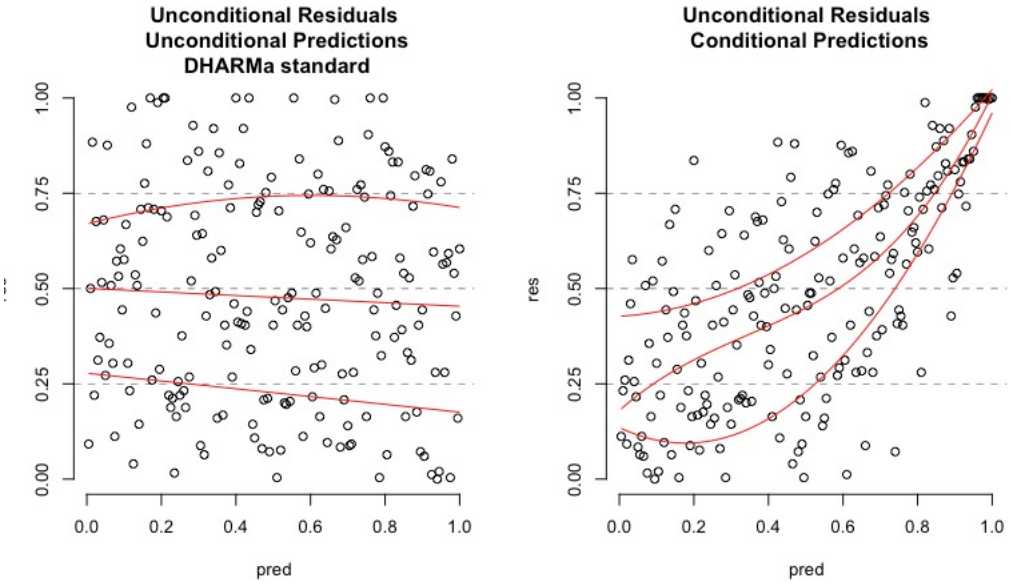
Unconditional simulations, re-simulate also REs

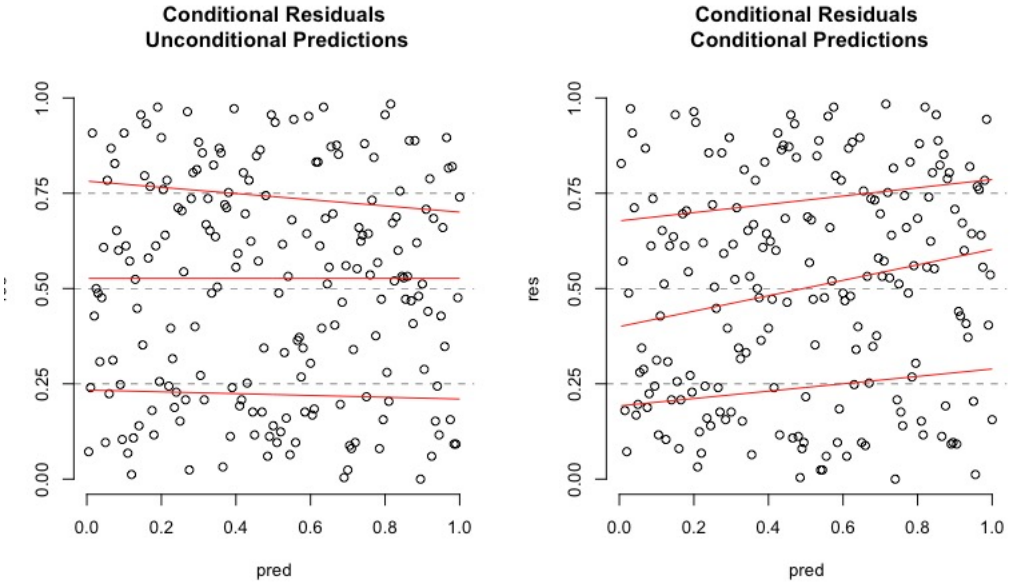
Uncoditional predictions (without RE)

Obvious question - which option is more appropriate?

Unfortunately, as often in statistics, the answer is: it depends on what you want to know

Easy: when plotting res ~ pred, use unconditional for pred





- Especially when using uncoditional simulations (next slide), plotting residuals against predictions with RE creates a spurious pattern.
- Detailed explanation in https://github.com/florianhartig/DHARMa/issues/43
- The bottomline: do not plot simulated residuals against predictions made with random effects

Conditional vs. unconditional simulations?

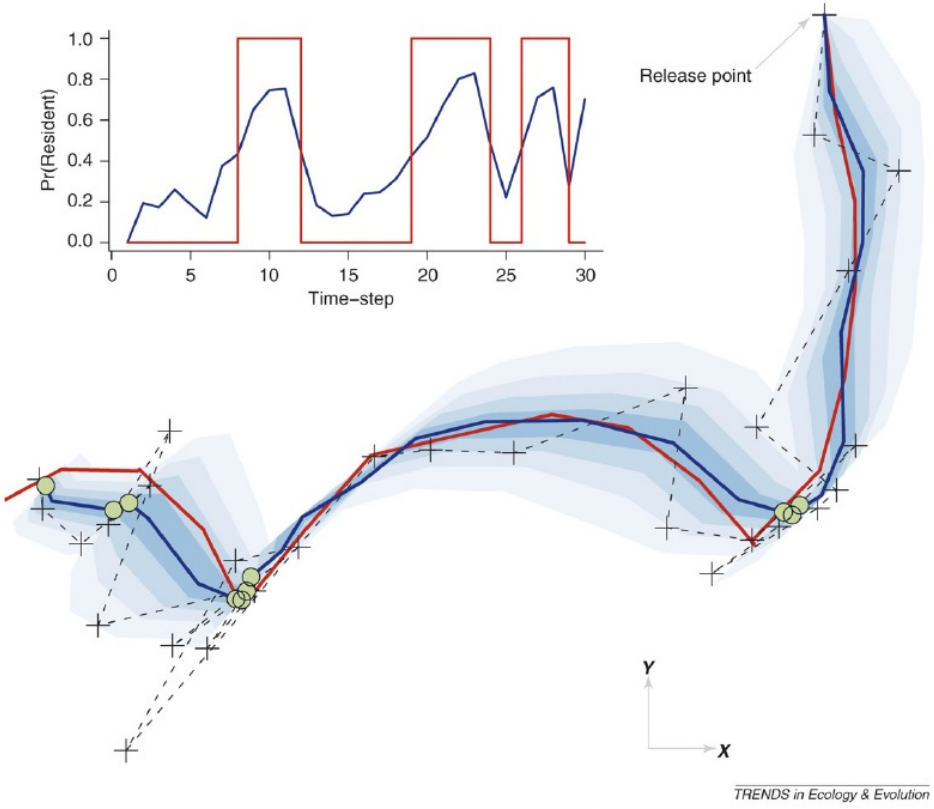
- Imagine you fit a GLMM with spatial autocorrelation and a missing predictor, which are both absorved by a random effect (1 | space)

- Because the RE will absorb these issues, conditional simulations
  - Will not show spatial autocorrelation
  - Will possibly not show a correlation with the missing predictor

- While unconditional simulations
  - Will show spatial autocorrelation (despite the RE potentially absorbing it already)
  - Will show a correlation with the missing predictor

→ It depends if you you are happy that, conditional on the fitted RE, residuals are fine, or if you want to be alerted to the fact that the data would be unlikely to occur like that given the assumed model (because there is spatial / covariance structure absorved in the RE)

Issue intensifies in models with many hierarchical levels



- Residuals in published state-space (HMM) always look great - because they are conditional residuals
- If you simulate without conditioning on the process-error, they can be horrible
- → How to derive good residuals checks for these models is a topic on its own

Patterson, Toby A.; Thomas, Len; Wilcox, Chris; Ovaskainen, Otso & Matthiopoulos, Jason (2008) State-space models of individual animal movement. *Trends in Ecology & Evolution,* 23, 87-94.

Special considerations for a Bayesian analysis

- Quantile residuals are flat if we simulate from the true model (likelihood + exact true parameters)

- Bayesian posteriors, however, are wide and possibly biased towards the prior
  - Bayesian quantile residuals might not be entirely flat, although the model is correct, especially if the likelihood is weak
  - In practice, however I have found this rarely to be a problem

- More detailed comments in the vignette DHARMa for Bayesians https://github.com/florianhartig/DHARMa/blob/master/DHARMa/vignettes/DHARMaForBayesians.Rmd

Conclusions

- Maybe a bit overwhelming at first, but randomized quantile residuals + DHARMa offer a completely general framework to check ANY (however complicated / hierarchical) model
  - Therefore, worth learning how this works

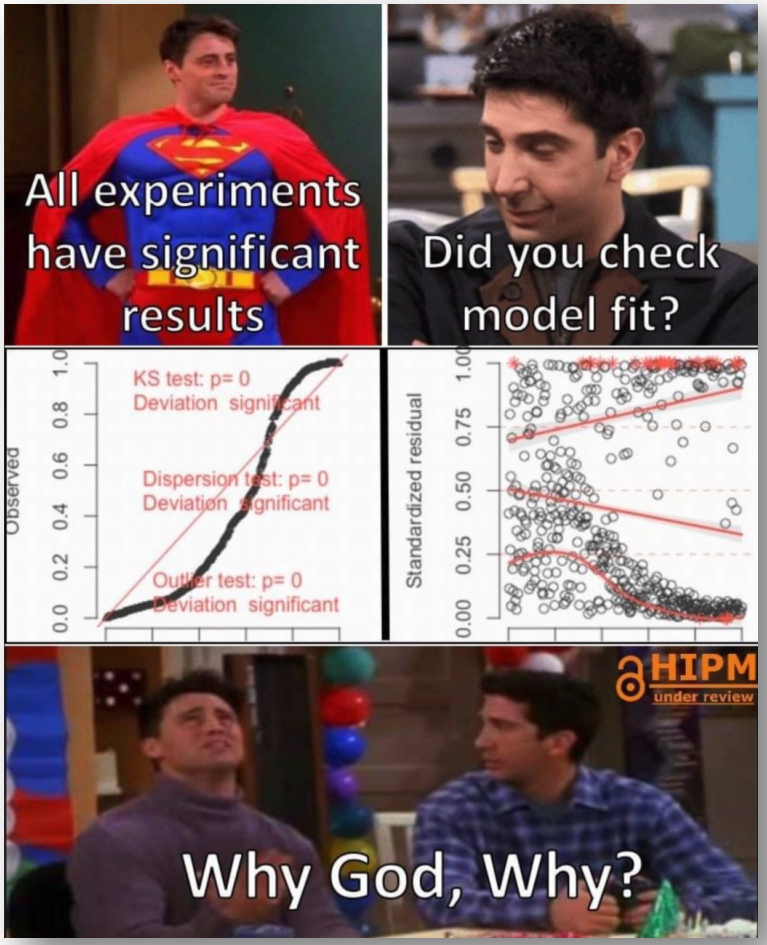
- The basic idea:
  - We simulate new data from the fitted model, and then compare simulated / observed, either
    - Regarding specific properties, e.g. dispersion (Bayesian p-values)
    - For each observation (randomized quantile residuals)

A bit of wisdom from countless interactions with DHARMa users

- People spend too much time on posterior / fitted residual checks and too little on
  - Does the model make sense? Causality?
  - Is the model fittable / unbiased in the first place?

- If a residual pattern is detected, the question is if it is large enough to be worrysome
  - p-value is **NOT a measure of effect size**, it just tells you that the detected pattern is non-random
  - You have to estimate effect size by eyeballing or by the provided statistics (e.g. dispersion ratio)
  - Many residual patterns (but not dispersion problems) only have minor consequences for the inference
  - Don't trust rules in the books, you have to simulate!

Thank you!


All experiments have significant results
Did you check model fit?
KS test: p= 0
Deviation significant
Dispersion test: p= 0
Deviation significant
Outlier test: p= 0
Deviation significant
Observed
Standardized residual
HIPM
under review
Why God, Why?