Dia N°

Overdispersion (in R)

→ question
⇢ action
Package::fonction
Possibilité non explorée

Données :

Binomiales

Les données sont en format 2

- court
- long

long ⇢ court

court ⇢ GLM binomial

long → Format court impossible => les données sont Bernoulli => GLM binomial, Pas de sur-dispersion possible (cf blabla 2).

GLM binomial ⇢ Test sur-dispersion
DHARMa::simulateResiduals(m)
Φ = DHARMa::testDispersion(m)$statistic # est-ce > 1.05 ?
Si oui, il y a de la surdispersion

- $\Phi < 1{,}05$
- $\Phi > 1{,}05$

$\Phi > 1{,}05$ → le nombre d'essais n est constant pour chaque observation ?

- oui → GLM bêta betareg::betareg()
- non → LM

LM
- Sur les proportions brutes
- Sur les proportions transformées (boot::logit ; Si besoin, back-transformation avec boot::inv.logit)

GLM
- Beta-binomiale VGAM::vglm(… , family = betabinomial)
- Quasi-binomiale glm(…, family = quasibinomial(link = 'logit')), but : 3
- Binomial-OLRE lme4::glmer(…+(1|…), family = binomial(link = 'logit'))

Respect des conditions d'applications ?

Aucun modèle n'est acceptable ⇢ Test zero-inflation

Si un GLM est acceptable, il est toujours préférable au LM ⇢

Comparaison des modèles parmi ceux qui respectent leurs conditions d'applications
- En validation croisée
- En terme d'adéquation entre la relation moyenne-variance empirique et relation supposée par le modèle (uniquement Beta-binomiale *vs* Quasi-binomiale)
- En AICc (uniquement Beta-binomiale *vs* Binomial-OLRE)

Poisson

Poisson ⇢ GLM poisson ⇢ Test sur-dispersion
DHARMa::simulateResiduals(m)
Φ = DHARMa::testDispersion(m)$statistic # est-ce > 1.05 ?
Si oui, il y a de la surdispersion

- $\Phi < 1{,}05$
- $\Phi > 1{,}05$

$\Phi > 1{,}05$ ⇢

LM
- Sur les comptages brutes
- Sur les comptages transformés (transf. **log** ou box-cox, seuls ou transf.(comptes+1) ou transf.(comptes+constante)) /!\ constante choisie pour obtenir le respect des conditions d'applications, PAS pour obtenir la p.value qui nous plait bien…

GLM
- Negative-binomiale lme4::glmer.nb(…)
- Quasi-Poisson glm(…, family = quasipoisson(link = 'log')) , but : 3
- Poisson-OLRE lme4::glmer(…+(1|…), family = poisson(link = 'log'))

Respect des conditions d'applications ?

Aucun modèle n'est acceptable → Test zero-inflation

Si un GLM est acceptable, il est toujours préférable au LM ⇢

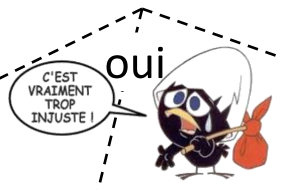
Comparaison des modèles parmi ceux qui respectent leurs conditions d'applications
- En validation croisée
- En terme d'adéquation entre la relation moyenne-variance empirique et relation supposée par le modèle (uniquement Negative-binomiale *vs* Quasi-Poisson)
- En AICc (uniquement Negative-binomiale *vs* Poisson-OLRE)

Test zero-inflation

Test zero-inflation
DHARMa::testZeroInflation (m)
C'est le problème ?

- oui ⇢ Utilisez des Zero-Inflated Models Liens 1, 2 et 3, mais aussi : 5
- non → Vous êtes vraiment sûr qu'aucun modèle n'est acceptable ?

Vous êtes vraiment sûr qu'aucun modèle n'est acceptable ?
- non ⇢ Comparaison des modèles
- oui



- Utilisez des random forests
- Bootstrapper le modèle : 4
- Utilisez de la « Randomized residual permutation procedure » (RRPP) sur le moins pire des modèles linéaire. Liens 1, 2,
Collyer, M. L., & Adams, D. C. (2007). Analysis of Two-State Multivariate Phenotypic Change in Ecological Studies. Ecology, 88(3), 683–692.
Paragraph "Statistically Comparing Phenotypic Change with a Permutation Procedure Vectors"
The methods works as follow: If two models M0 and M1 differs only for the effect to be tested, (M0 does not contain the effect) then the residuals of M0 contain the variability that would be explained by the considered effect.
Thus we can use M0 to build the null distribution of the coefficient(s) of the effect: The fitted values of M0 + The residuals of M0 = the original data. The fitted values of M0 + The *randomized* residuals of M0 = the data from which the variability that would be explained by the considered effect have been randomized. Refitting M1 on this randomized dataset allows to build the null distribution of the considered effect.

J'avais des données Poisson !

2

Estimating overdispersion on data Bernoulli-formatted is meaningless

Format long :

Success	X1	X2
Yes	Yes	10.7
Yes	Yes	10.7
No	Yes	10.7
Yes	No	11.3
Yes	No	11.3
Yes	No	11.3
No	No	11.3
Yes	Yes	9.9
Yes	Yes	9.9
Yes	Yes	9.9
No	Yes	9.9
No	Yes	9.9

=

Format court : **Yes**

Successes	Failures	X1	X2
2	1	Yes	10.7
3	1	No	11.3
3	2	Yes	9.9

If, and only if you cannot do this transformation, then, your data are truly Bernoulli data. You don't have to worry about over/under-dispersion. See *Skrondal and Rabe-Hesketh 2007* *Redundant Overdispersion Parameters in Multilevel Models for Categorical Responses* for the demonstration (https://app.box.com/s/nj4osunwzwfadml3f997gkzxuoq28gmo). Otherwise, you have to do format your data in that way before fitting the model on which you will check for over-dispersion.

3

A weakness of the 'quasi' approach is that it does not model the overdispersion in the data, but merely adjusts the resulting parameter estimates with a single correction factor. The assumption that all standard errors are biased to the same degree is an obvious problem, which may not be appropriate.

Harrison, X. A. (2015). A comparison of observation-level random effect and Beta-Binomial models for modelling overdispersion in Binomial data in ecology & evolution.

4

Pour Bootstrapper le modèle :

Pour des GLM (pas d'effet aléatoire) :

→ car::Boot(model, …)

Pour des GLMM (au moins un effet aléatoire) :

→ afex::mixed(…, method = "PB")

5

Since zip (zero inflated poisson) has both a count model and a logit model, each of the two models should have good predictors. The two models do not necessarily need to use the same predictors.

Problems of perfect prediction, separation or partial separation can occur in the logistic part of the zero-inflated model.

Count data often use exposure variables to indicate the number of times the event could have happened. You can incorporate a logged version of the exposure variable into your model by using the offset() option.

It is not recommended that zero-inflated Poisson models be applied to small samples.

pscl::zeroinfl

Many packages implement some models suited for count data, but `glmmADMB` and `MCMCglmm` are (or were) the only one that implements all those that we are considering, and that work, both with and without random effects.

Exemple for count data in `glmmADMB` :

```
library( glmmADMB )
glmmadmb(BLABLA             ,family="poisson" ,zeroInflation=FALSE)        # Poisson
glmmadmb(BLABLA             ,family="nbinom"  ,zeroInflation=FALSE)        # Negative binomial
glmmadmb(BLABLA             ,family="poisson" ,zeroInflation=TRUE)         # Zero-inflated poisson
glmmadmb(BLABLA             ,family="nbinom"  ,zeroInflation=TRUE)         # Zero-inflated negative binomial
glmmadmb(BLABLA + (1|obs)   ,family="poisson" ,zeroInflation=FALSE)        # Random-effects count models (OLRE)
```

(If need, Zero-truncated and Hurdle are also available in this package)

Getting started with the glmmADMB package: http://glmmadmb.r-forge.r-project.org/glmmADMB.html

An alternative: MCMCglmm::MCMCglmm heavy, but very flexible

see the relevant **course notes** (one of the most interesting documents on statistics I've ever seen).

Avec ce type de modèle, vous aurez parfois des problèmes de convergence …

?lme4::convergence: short of exhaustive/detailed evaluation of the optimization procedure, your best best is to compare results from different optimizers.

Hurdle *vs.* Zero-inflated

http://stats.stackexchange.com/questions/81457/what-is-the-difference-between-zero-inflated-and-hurdle-distributions-models

Hurdle models can be motivated by sequential decision-making processes confronted by individuals. You first decide if you need to buy something, and then you decide on the quantity of that something (which must be positive). When you are allowed to (or can potentially) buy nothing after your decision to buy something is an example of a situation where zero-inflated model is appropriate. Zeros may come from two sources: a) no decision to buy; b) wanted to buy but ended up buying nothing (e.g. out of stock).

Hurdle models in glmmADMB

In contrast to zero-inflated models, hurdle models treat zero-count and non-zero outcomes as two completely separate categories, rather than treating the zero-count outcomes as a mixture of structural and sampling zeros.

As of version 0.6.7.1, glmmADMB includes truncated Poisson and negative binomial familes and hence can fit hurdle models. The two parts of the model have to be fitted separately, however. First we fit a truncated distribution to the non-zero outcomes:

```
> fit_hnbinom1 <- glmmadmb(NCalls~(FoodTreatment+ArrivalTime)*SexParent+
                BroodSize+(1|Nest),
                data=subset(Owls,NCalls>0),
                family="truncnbinom1")
```

Then we fit a model to the binary part of the data (zero vs. non-zero). In this case, I started by fitting a simple (intercept-only) model with intercept-level random effects only. This comes a bit closer to matching the previous (zero-inflation) models, which treated zero-inflation as a single constant level across the entire data set (in fact, leaving out the random effects and just using glmmADMB(nz~1,data=Owls,family="binomial"), or glm(nz~1,data=Owls,family="binomial"), would be an even closer match). I then fitted a more complex binary model — this is all a matter of judgment about how complex a model it's worth trying to fit to a given data set — but it does look as though the zero-inflation varies with arrival time and satiation.

```
> Owls$nz <- as.numeric(Owls$NCalls>0)
> fit_count <- glmmadmb(nz~1+(1|Nest),
                data=Owls,
                family="binomial")
> fit_ccount <- glmmadmb(nz~(FoodTreatment+ArrivalTime)*SexParent+(1|Nest),
               data=Owls,
               family="binomial")
> AICtab(fit_count,fit_ccount)
> summary(fit_ccount)
```