sample(ECOLOGY)

Thoughts on ecology, biodiversity, and science in general

R^2 for linear mixed effects models

Linear mixed effects models are a powerful technique for the analysis of ecological data, especially in the presence of nested or hierarchical variables. But unlike their purely fixed-effects cousins, they lack an obvious criterion to assess model fit. In the fixed-effects world, the coefficient of determination, better known as R^2, is a useful and intuitive tool for ascertaining whether a model describes the data well: it's simply the "variance explained" by the model. Its worth the time to take a moment to represent this mathematically: R^2 is simply 1 – the ratio of the unexplained variance by the model (residual variance, difference between observed and predicted values) over the total observed variance.

R^2 is also dimensionless, which makes it ideal for comparing fits across different datasets, although I should note that is a poor tool for model selection, since it almost always favors the most complex models. If the goal is to select among the best models, an information criterion approach (such as AIC or BIC) is preferred, since these indicators penalize for the number of predictors. However (and this is a common pitfall), the best model as determined by AIC is not synonymous with a good model. In other words, AIC provides a great test of relative model fit, but not absolute model fit. AIC values are also not comparable across models using different data.

Often, researchers using mixed models report an R^2 from a linear mixed model as simply the squared correlation between the fitted and observed values (see here), but this is technically a pseudo-R^2 and thus incorrect. Why? Because a mixed effects model yields a variance associated with each random factor *and* the residual variance, so its not entirely clear which to use when calculating the R^2 (the approach above ignores this issue by choosing to calculate R^2 relative to only the residual variance, which omits any structure in the data imposed by the random factor). When hierarchical models with >1 levels are considered, this problem is exacerbated since now we are dealing with variances at multiple levels, plus the residual error. Some people have discussed calculating an R^2 value for each level of the random factor, but this can lead to negative R^2 values when addition of predictors reduces the residual error while increasing the variance of the random component (or vice versa), even though the sum of the variance components remains unchanged.

So what have we done? We've compared AIC values for the full and null models, or calculated all manner of pseudo-R^2s, and quietly (and perhaps for some, ashamedly) went about our business. Which is why I was so happy to see a paper with the title "A general and simple method for obtaining R^2 from generalized linear mixed-effects models" by Shinichi Nakagawa and Holger Schielzeth appear in my newsfeed. Basically, they have derived two easily interpretable values of R^2 that address the above issues, specifically that of negative pseudo-R^2s with more predictors, while still honoring the random structure of the

data.

The first is called the ***marginal R^2*** and describes the proportion of variance explained by the fixed factor(s) alone. The second is the ***conditional R^2***, which describes the proportion of variance explained by both the fixed and random factors. They've also gone on to extend the calculations to non-normal distributions (although here, I deal only with non-generalized linear mixed models). While this may not be the holy grail of fit statistics (and the authors admit this point as well), in my opinion, it does address many of the problems of pseudo-R^2s and so is going to be my preferred statistic, at least until someone proposes something better. The authors also note (and I echo) that looking at R^2 and AIC values is not a replacement for testing basic model assumptions, such as normality of predictors and non-constant variance, and should be used in tandem.

I've written a function in R called *rsquared.lme* that calculates the marginal and condition R^2s. The input is simply a list of model objects of either class *mer* or *lme* (from the "lme4" and "nlme" packages, respectively), and the output is a matrix where each row is a model from the list with two columns for the marginal and conditional R^2s. Happy modeling!

```
#require(lme4)
#require(nlme)

rsquared.lme=function(modlist) {
  modclass=unlist(lapply(modlist,class))
  ifelse(
    all(modclass[1]==modclass)==FALSE,return("Error: Objects in list need to all be of
  if(modclass[1]=="mer") { #For models fit using lmer
    #Get variance of fixed effects by multiplying coefficients by design matrix
    VarF.list=lapply(modlist,function(i) varF=var(as.vector(fixef(i) %*% t(i@X))) )
    #Get variance of random effects by extracting variance components
    VarRand.list=lapply(modlist,function(i) do.call(rbind,lapply(VarCorr(i),function(j)
    #Get residual variance
    VarResid.list=lapply(modlist,function(i) attr(VarCorr(i), "sc")^2 )
    #Calculate marginal R-squared (fixed effects/total variance)
    Rm.list=lapply(seq_along(modlist),function(i) VarF.list[[i]]/(VarF.list[[i]]+colSum
    #Calculate conditional R-squared (fixed effects+random effects/total variance)
    Rc.list=lapply(seq_along(modlist),function(i) (VarF.list[[i]]+colSums(VarRand.list[
    #Bind R^2s into a matrix and return
    Rsquared.mat=do.call(cbind,list(Rm.list,Rc.list)); colnames(Rsquared.mat)=c("Margir
    return(Rsquared.mat)
  }
  if(modclass[1]=="lme") {#For models fit using lme
    #Get design matrix of fixed effects from model
    Fmat.list=lapply(modlist,function(i) model.matrix(eval(i$call$fixed)[-2],i$data) )
    #Get variance of fixed effects by multiplying coefficients by design matrix
    VarF.list=lapply(seq_along(modlist),function(i) var(as.vector(fixef(modlist[[i]]) %
```

```
    #Get variance of random effects by extracting variance components
    VarComp.list=lapply(modlist,function(i) VarCorr(i) )
    VarRand.list=lapply(VarComp.list,function(i) as.numeric(i[rownames(i)!="Residual" &
    #Get residual variance
    VarResid.list=lapply(VarComp.list,function(i) as.numeric(i[rownames(i)=="Residual",
    #Calculate marginal R-squared (fixed effects/total variance)
    Rm.list=lapply(seq_along(modlist),function(i) VarF.list[[i]]/(VarF.list[[i]]+sum(Va
    Rc.list=lapply(seq_along(modlist),function(i) (VarF.list[[i]]+ifelse(length(VarRanc
    #Bind R^2s into a matrix and return
    Rsquared.mat=do.call(cbind,list(Rm.list,Rc.list)); colnames(Rsquared.mat)=c("Margir
    return(Rsquared.mat)
  #Or return error if objects are not linear mixed effects models
  else { print("Error: Function requires list of objects of class 'mer' or 'lme'") }
}

#Example
#mod1=lmer(rnorm(100,5,10)~rnorm(100,20,100)+(1|rep(c("A","B"),50)))
#mod2=lmer(rnorm(100,5,10)~rnorm(100,20,100)+rnorm(100,0.5,2)+(1|rep(c("A","B"),50)))
#rsquared.lme(list(mod1,mod2))
```

References:

Shinichia, N., and H. Schielzeth. 2013. A general and simple method for obtaining R^2 from generalized linear mixed-effects models. Methods in Ecology and Evolution 4(2): 133-142.

Created by Pretty R at inside-R.org

This entry was posted in R, Uncategorized and tagged lme, mixed models, R on March 13, 2013 [https://jslefche.wordpress.com/2013/03/13/r2-for-linear-mixed-effects-models/] .

5 thoughts on "R^2 for linear mixed effects models"

Lee Gutowsky
March 20, 2013 at 1:38 am

Awesome paper, I'm looking forward to reading it! By the way, any chance you'll write such a code for PQL objects from the MASS package? Or, are marginal and conditional R^2 for PQL estimated GLMMs still unavailable?

aghaynes
March 20, 2013 at 2:59 pm

Nice looking function!
Just a heads-up – the development version of lme4 no longer uses the "mer" class (I believe its changed to "merMod"), nor @ for accessing slots (although a quick look suggests that this doesnt matter – you dont use it). I'm not sure if attr names have changed or not though…

aghaynes
March 20, 2013 at 3:48 pm

Hi again,
I just tried your function out on a list of lme objects and it broke for 3 reasons
1) you did'nt do the Rc.list part for lme models;
I copied the line from the mer part to fix this, but…
2) colSums only works if there are >1 random effect;
I added an ifelse statement to sort this, and…
3) you include AIC.list when you bind the results together but you never make it so I removed that from the call.

Here's my edited version of the lme section for you. Hope you dont mind my tinkering with your code!

if(modclass[1]=="lme") {#For models fit using lme
#Get design matrix of fixed effects from model
Fmat.list=lapply(modlist,function(i) model.matrix(eval(i$call$fixed)[-2],i$data))
#Get variance of fixed effects by multiplying coefficients by design matrix
VarF.list=lapply(seq_along(modlist),function(i) var(as.vector(fixef(modlist[[i]]) %*% t(Fmat.list[[i]]))))
#Get variance of random effects by extracting variance components
VarComp.list=lapply(modlist,function(i) VarCorr(i))
VarRand.list=lapply(VarComp.list,function(i) as.numeric(i[rownames(i)!="Residual" & rownames(i)=="(Intercept)","Variance"]))
#Get residual variance
VarResid.list=lapply(VarComp.list,function(i) as.numeric(i[rownames(i)=="Residual","Variance"]))
#Calculate marginal R-squared (fixed effects/total variance)

```
Rm.list=lapply(seq_along(modlist),function(i)
VarF.list[[i]]/(VarF.list[[i]]+sum(VarRand.list[[i]])+VarResid.list[[i]]) )
Rc.list=lapply(seq_along(modlist),function(i) (VarF.list[[i]]+ifelse(length(VarRand.list[[i]]
[1])==1,VarRand.list[[i]][1],colSums(VarRand.list[[i]])))/(VarF.list[[i]]+ifelse(length(VarRand.list[[i]]
[1])==1,VarRand.list[[i]][1],colSums(VarRand.list[[i]]))+VarResid.list[[i]]) )
#Bind R^2s into a matrix and return
Rsquared.mat=do.call(cbind,list(Rm.list,Rc.list)); colnames(Rsquared.mat)=c("Marginal","Conditional")
return(Rsquared.mat)
}
```

jslefche Post author
March 21, 2013 at 2:42 am

Great, thanks so much for catching these errors! I think you can see where I was going with expanding the function to return AIC scores but since it requires ML vs REML estimation, the implementation was a little too lengthy for what I wanted to do here. Also, nice catch with the colSums: I went through this function with the example provided in the Nakagawa & Schielzeth supplements, which of course used multiple random effects.

aghaynes
March 21, 2013 at 7:11 am

Not a problem…your function actually came along at an ideal time for me! For the AIC scores you could hijack aictab in the AICcmodavg package. It produces a table of AIC (or AICc, QAIC, QAICc) values. It only provides the AIC for REML or ML, whichever was used in the fitting process, but using lapply and update you could get around this (for lme, lmer should be very similar though):

```
REML.AIC <- aictab(mods, 1:8, second.ord=FALSE)[,"AIC"] # for AICc second.ord=TRUE and [,"AICc"]
ML.AIC <- aictab(lapply(mods, function(i) update(i, method="ML")), 1:8, second.ord=FALSE)[,"AIC"]
```

Although it might be sensible to do a logical test on whether the model was fit with ML or REML first.
HTH
