GLMMs in action: gene-by-environment interaction in total fruit production of wild populations of *Arabidopsis thaliana*
Revised version, part 1

Benjamin M. Bolker* Mollie E. Brooks[2] Connie J. Clark[3]
Shane W. Geange[4] John R. Poulsen[3] M. Henry H. Stevens[5]
Jada-Simone S. White[6]

September 16, 2011

1 Introduction

Spatial variation in nutrient availability and herbivory is likely to cause population differentiation and maintain genetic diversity in plant populations. Here we measure the extent to which mouse-ear cress (*Arabidopsis thaliana*) exhibits population and genotypic variation in their responses to these important environmental factors. We are particularly interested in whether these populations exhibit nutrient mediated *compensation*, where higher nutrient levels allow individuals to better tolerate herbivory (Banta et al., 2010). We use GLMMs to estimate the effect of nutrient levels, simulated hebivory, and their interaction on fruit production in *Arabidopsis thaliana* (fixed effects), and the extent to which populations vary in their responses (random effects, or variance components)[1].

*[1]Departments of Mathematics & Statistics and Biology, McMaster University, Hamilton, Ontario, Canada L8S 4K1; [2] Department of Biology, University of Florida, PO Box 118525, Gainesville FL 32611-8525; [3] Nicholas School of the Environment, Duke University, PO Box 90328, Durham, NC 27708; [4]School of Biological Sciences, Victoria University of Wellington PO Box 600, Wellington 6140, New Zealand; [5]Department of Botany, Miami University, Oxford OH, 45056; [6]Department of Biological Sciences, California State University, Chico, Chico, CA 9529-515. *Corresponding author*: Bolker, B. M. (bolker@mcmaster.ca)

[1]Data and context are provided courtesy of J. Banta and M. Pigliucci, State University of New York (Stony Brook).

Below we follow guidelines from (Bolker et al., 2009) to step deliberately through the process of model building and analysis. In this example, we use the `lme4` package (Bates and Maechler, 2010) in the **R** language and environment (R Development Core Team, 2009); other approaches to fitting GLMMs are illustrated in part 2.

We will use some other packages for plotting, manipulating data, and interpreting results:

```
> library(lme4)
> library(coefplot2)  ## for coefplot2
> library(reshape)
> library(plyr)
> library(ggplot2)
> library(gridExtra)
> library(emdbook)   ## for qchibarsq
> source("glmm_funs.R")
```

[2]

2 The data set

In this data set, the response variable is the number of fruits (i.e. seed capsules) per plant. The number of fruits produced by an individual plant (the experimental unit) was hypothesized to be a function of fixed effects, including nutrient levels (low *vs.* high), simulated herbivory (none *vs.* apical meristem damage), region (Sweden, Netherlands, Spain), and interactions among these. Fruit number was also a function of random effects including both the population and individual genotype. Because *Arabidopsis* is highly selfing, seeds of a single individual served as replicates of that individual. There were also nuisance variables, including the placement of the plant in the greenhouse, and the method used to germinate seeds. These were estimated as fixed effects but interactions were excluded.

[2] We used R version 2.13.1 (2011-07-08) and package versions:

```
coefplot2        doBy     emdbook     ggplot2   gridExtra        lme4
    0.1.1       4.4.0       1.3.1       0.8.9         0.8 0.999375-41
     plyr     reshape      RLRsim
      1.6       0.8.4      2.0-10
```

The `coefplot2` package must be installed from `http://r-forge.r-project.org`; all others are on CRAN.

3 Specifying fixed and random effects

Here we need to select a realistic full model, based on the scientific questions and the data actually at hand. We first load the data set and make sure that each variable is appropriately designated as numeric or factor (i.e. categorical variable).

```
> dat.tf <- read.csv("Banta_TotalFruits.csv")
> str(dat.tf)

'data.frame':        625 obs. of  9 variables:
 $ X           : int  1 2 3 4 5 6 7 8 9 10 ...
 $ reg         : Factor w/ 3 levels "NL","SP","SW": 1 1 1 1 1 1 1 1 1 1 ...
 $ popu        : Factor w/ 9 levels "1.SP","1.SW",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ gen         : int  4 4 4 4 4 4 4 4 4 5 ...
 $ rack        : int  2 1 1 2 2 2 2 1 2 1 ...
 $ nutrient    : int  1 1 1 1 8 1 1 1 8 1 ...
 $ amd         : Factor w/ 2 levels "clipped","unclipped": 1 1 1 1 1 2 1 1 2 1 ...
 $ status      : Factor w/ 3 levels "Normal","Petri.Plate",..: 3 2 1 1 3 2 1 1 1 2 ..
 $ total.fruits: int  0 0 0 0 0 0 0 3 2 0 ...
```

The `X`, `gen`, `rack` and `nutrient` variables are coded as integers, but we want them to be factors: we could either redo the `read.csv` command with `colClasses` set, or set the variables to factors by hand.

- We use `transform()`, which operates within the data set, to avoid typing lots of commands like `dat.tf$rack <- factor(dat.tf$rack)`
- At the same time, we reorder the clipping variable so that `"unclipped"` is the reference level: we could also have used `relevel(amd,"unclipped")`

```
> dat.tf <- transform(dat.tf,
                      X=factor(X),
                      gen=factor(gen),
                      rack=factor(rack),
                      amd=factor(amd,levels=c("unclipped","clipped")),
                      nutrient=factor(nutrient,label=c("Low","High")))
```

[3]

The above output shows that the data include,

[3] There is sometimes a tradeoff between clarity and comprehension. More laboriously but perhaps more clearly, we could have made the same changes to the integer-coded

`X` observation number (we will use this observation number later, when we are accounting for overdispersion)

`reg` a factor for region (Netherlands, Spain, Sweden).

`popu` a factor with a level for each population.

`gen` a factor with a level for each genotype.

`rack` a nuisance factor for one of two greenhouse racks.

`nutrient` a factor with levels for minimal or additional nutrients.

`amd` a factor with levels for no damage or simulated herbivory (apical meristem damage; we will sometimes refer to this as "clipping")

`status` a nuisance factor for germination method.

`total.fruits` the response; an integer count of the number of fruits per plant.

Now we check replication for each genotype (columns) within each population (rows).

```
> (reptab <- with(dat.tf, table(popu, gen)))
```

```
      gen
popu    4  5  6 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 27 28 30 34 35 36
  1.SP  0  0  0  0  0 39 26 35  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  1.SW  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 28 20  0  0  0  0  0
  2.SW  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 18 14  0  0  0
  3.NL 31 11 13  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  5.NL  0  0  0 35 26  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  5.SP  0  0  0  0  0  0  0  0 43 22 12  0  0  0  0  0  0  0  0  0  0  0  0  0
  6.SP  0  0  0  0  0  0  0  0  0  0  0 13 24 14  0  0  0  0  0  0  0  0  0  0
  7.SW  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 45 47 45
  8.SP  0  0  0  0  0  0  0  0  0  0  0  0  0  0 13 16 35  0  0  0  0  0  0  0
```

[4] This reveals that we have only 2–4 populations per region and 2–3 genotypes per population. However, we also have 2–13 replicates per genotype for each treatment combination (four unique treatment combinations: 2 levels of nutrients by 2 levels of simulated herbivory). Thus, even though this was a reasonably large experiment (625 plants), there were a very small

---

factors as follows:

```
> dat.tf$X <- factor(dat.tf$X)
> dat.tf$gen <- factor(dat.tf$gen)
> dat.tf$rack <- factor(dat.tf$rack)
> dat.tf$nutrient <- factor(dat.tf$nutrient)
```

We find the code with `transform` clearer, but you should use whatever form makes more sense to you (and is easier to understand when you come back to the code later).

[4] you could also inspect this graphically: `with(dat.tf,plot(table(popu,gen)))` produces a *mosaic plot* of the two-way table ...

number of replicates with which to estimate variance components, and many more potential interactions than our data can support. Therefore, judicious selection of model terms, based on both biology and the data, is warranted. We note that we don't really have enough levels per random effect, nor enough replication per unique treatment combination. Therefore, we decide to omit the fixed effect of "region", although we recognize that populations in different regions are widely geographically separated[5].

We have only two random effects (population, individual), and so Laplace or Gauss-Hermite Quadrature (GHQ) should suffice, rather than requiring more complex methods. However, as in all GLMMs where the scale parameter is treated as fixed and deviations from the fixed scale parameter would be identifiable (i.e. Poisson and binomial ($N > 1$), but not binary, models) we may have to deal with overdispersion.

4 Choose an error distribution; graphical checks

The data are non-normal in principle (i.e., count data, so our first guess would be a Poisson distribution). If we transform total fruits with the canonical link function (log), we hope to see relatively homogeneous variances across categories and groups.

First we define a new factor that represents every combination of genotype and treatment (nutrient × clipping) treatment, and sort it in order of increasing mean fruit set.

```
> ## use within() to make multiple changes within a data frame
> dat.tf <- within(dat.tf,
                   {
                     gna <- interaction(gen,nutrient,amd)
                     gna <- reorder(gna, total.fruits, mean)
                   })
```

Now we use the `bwplot` function in the `lattice` package to generate a box-and-whisker plot of log(fruits + 1).

```
> print(bwplot(log(total.fruits + 1) ~  gna,
               data=dat.tf,
               scales=list(x=list(rot=90))  ## rotate x-axis labels
               ))
```

[5] In hindsight, we should probably have included region as a *fixed* effect; re-doing the analysis with this change is left as an exercise for the reader.

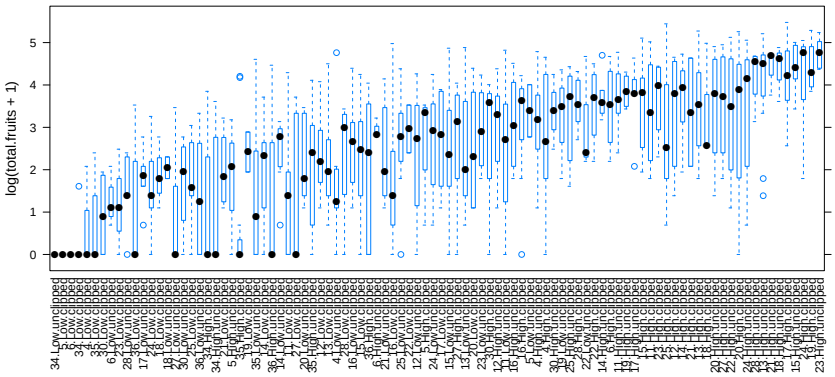

Figure 1: Box-and-whisker plots of log(total fruits+1) by treatment combinations and genotypes reveal heterogeneity of variances across groups.

6 7

Taking the logarithm of total fruit (or in this case $\log(1 + \text{fruit})$ to avoid taking the log of zero) should stabilize the variance (McCullagh and Nelder, 1989), but this does not seem to be the case (Figure 1). The variance actually seems to decline slightly in the largest groups.

We could also calculate the variance for each genotype $\times$ treatment combination and provide a statistical summary of these variances.

```
> grpVarL <- with(dat.tf, tapply(log(1+total.fruits),
                                 list(gna), var) )
> summary(grpVarL)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.774   1.460   1.514   2.025   4.847
```

This reveals substantial variation among the sample variances on the transformed data. In addition to heterogeneous variances across groups, Figure 1

[6]The `print()` function is not necessary in interactive use, only when you want to send `lattice` or `ggplot` graphs to a file during an R batch run.

[7]Alternatively, one can also use the `ggplot2` package to draw these graphs. In this case, we rotate the graph 90 degrees (using `coord_flip()`) rather than rotating the x-axis labels ...

```
> ggplot(dat.tf,aes(x=gna,y=log(1+total.fruits)))+geom_boxplot()+coord_flip()+
    theme_bw()
```

reveals many zeroes in groups, and some groups with a mean and variance of zero, further suggesting we need a non-normal error distribution, and perhaps something other than a Poisson distribution.

Figure 1 also indicates that the mean (λ) of the Poisson distribution is well above 5 (i.e. in general the center of each group is greater than $\log(6) = 1.8$), suggesting that PQL may be appropriate, if it is the only option available. We could calculate λ for each genotype $\times$ treatment combination and provide a statistical summary of each group's λ.

```
> grpMeans <- with(dat.tf, tapply(total.fruits, list(gna), mean))
> summary(grpMeans)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00   11.35   23.16   31.86   49.74  122.40
```

The average by-group $\bar{\lambda}$ is 32; the median is ≈ 23; and at least one group has $\lambda = 0$ (i.e., complete absence of fruit set for that genotype $\times$ treatment combination).

A core property of the Poisson distribution is that the variance is equal to the mean. A simple diagnostic is a plot of the group variances against the group means (Figure 2):

- Poisson-distributed data will result in a linear pattern with slope=1
- as long as the variance is generally greater than the mean, we call the data *overdispersed*. Overdispersion comes in various forms:
  - a linear mean-variance relationship with Var $= \phi\mu$ (a line through the origin) with $\phi > 1$ is called a *quasi-Poisson* pattern (this term describes the mean-variance relationship, not any particular probability distribution); we can implement it statistically via quasilikelihood (Venables and Ripley, 2002) or by using a particular parameterization of the negative binomial distribution ("NB1" in the terminology of Hardin and Hilbe (2007)).
  - a semi-quadratic pattern, Var $= \mu(1 + \alpha\mu)$ or $\mu(1 + \mu/k)$, is characteristic of overdispersed data that is driven by underlying heterogeneity among samples, either the negative binomial (gamma-Poisson) or the lognormal-Poisson (Elston et al., 2001); we will see below how to fit data following these distributions.

We calculated the (genotype $\times$ treatment) variances in the same way as the means:

```
> grpVars <- with(dat.tf,
                  tapply(total.fruits,
                         list(gna), var) )  ## variance of UNTRANSFORMED data
```

[8]

We can get approximate estimates of the quasi-Poisson (linear) pattern using `lm`, as follows: (The `-1` in the formulas below specifies a model with the intercept set to zero.)

```
> lm1 <- lm(grpVars~grpMeans-1)  ## `quasi-Poisson' fit
> phi.fit <- coef(lm1)
```

We can also use `lm` to estimate the negative binomial (linear/quadratic) pattern $V = \mu + \mu^2/k$ by adding an offset (`offset(grpMeans)`) to the right-hand side to specify that we want the group means added as a term with its coefficient fixed to 1 (we use `I()` to specify that we really want to square the group means):

```
> lm2 <- lm(grpVars~I(grpMeans^2)+offset(grpMeans)-1)
> k.fit <- 1/coef(lm2)
```

We add a non-parametric `loess` fit to the plot of the means vs. variances and the fitted relationships;

```
> plot( grpVars ~ grpMeans, xlab="group means", ylab="group variances" )
> abline(c(0,1), lty=2)
> text(105,500,"Poisson")
> curve(phi.fit*x, col=2,add=TRUE)
> ## bquote() is used to substitute numeric values
> ##   in equations with symbols
> text(110,3900,
      bquote(paste("QP: ",sigma^2==.(round(phi.fit,1))*mu)),
      col=2)
```

[8] Here are two alternative ways to compute the mean and variance of total fruits by group, one using `plyr::ddply` and the other `aggregate` from base R:

```
> ddply(dat.tf,"gna",
      summarise,
      mean=mean(total.fruits),var=var(total.fruits))
> with(dat.tf,aggregate(total.fruits,
                        list(gna),
                        FUN=function(x) c(mean=mean(x),var=var(x))))
```

```
> curve(x*(1+x/k.fit),col=4,add=TRUE)
> text(104,7200,paste("NB: k=",round(k.fit,1),sep=""),col=4)
> ## loess fit
> Lfit <- loess(grpVars~grpMeans)
> mvec <- 0:120
> lines(mvec,predict(Lfit,mvec),col=5)
> text(118,2000,"loess",col=5)
```

[9]

These fits are not rigorous statistical tests — they violate a variety of assumptions of linear regression (e.g. constant variance, independence), but they are good enough to give us an initial guess about what distributions we should use.

We find that the group variances increase with the mean much more rapidly than expected under the Poisson distribution. This indicates that we need to account for overdispersion in the model. The linear (quasi-likelihood/NB1) fit may be better than the semi-quadratic (NB/lognormal-Poisson) fit, but it's hard to tell without additional model-fitting and diagnostics.

4.1 Plotting the responses *vs.* treatments

We would like to plot the response by treatment effects, to make sure there are no surprises.

```
> print(stripplot(log(total.fruits+1) ~ amd|popu,
                  data=dat.tf,
                  groups=nutrient,
                  auto.key=list(lines=TRUE, columns=2),
                  strip=strip.custom(strip.names=c(TRUE,TRUE)),
                  type=c('p','a'), layout=c(5,2,1),
                  scales=list(x=list(rot=90)), main="Panel: Population"))
```

[9] `ggplot` solution:

```
> ggplot(data.frame(grpMeans,grpVars),
       aes(x=grpMeans,y=grpVars))+geom_point()+
   geom_smooth(colour="blue",fill="blue")+
   geom_smooth(method="lm",formula=y~x-1,colour="red",fill="red")+
   geom_smooth(method="lm",formula=y~I(x^2)+offset(x)-1,
               colour="purple",fill="purple")
```

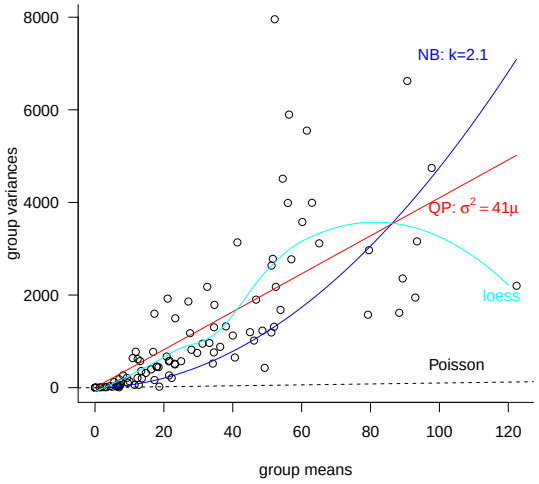

Figure 2: The variance-mean relation of Poisson random variables should exhibit a slope of approximately one (dashed line). In contrast, these data reveal a slope of much greater than one (the wedge or cornupia shape is expected for small Poisson samples, but the slope here is much, much greater than 1). The colored lines show estimated mean-variance relationships for quasi-Poisson (or negative-binomial "type I", Hardin and Hilbe (2007)) [red]: $V = \phi\mu$; negative binomial or lognormal-Poisson (Elston et al., 2001) [blue]: $V = \mu(1 + \mu/k)$ or $V = \mu(1 + \mu \cdot (\exp(\sigma^2) - 1))$; or a loess fit [cyan].

There seems to be a consistent nutrient effect (different intercepts), and perhaps a weaker clipping effect (negative slopes) — maybe even some compensation (steeper slope for zero nutrients, Fig. 3).

Because we have replicates at the genotype level, we are also able to examine the responses of genotypes (Fig. 4).

```
> print(stripplot(log(total.fruits+1) ~ amd|nutrient, dat.tf,
                  groups=gen,
                  strip=strip.custom(strip.names=c(TRUE,TRUE)),
                  type=c('p','a'), ## points and panel-average value --
                                   ## see ?panel.xyplot
                  scales=list(x=list(rot=90)),
                  main="Panel: nutrient, Color: genotype"))
```

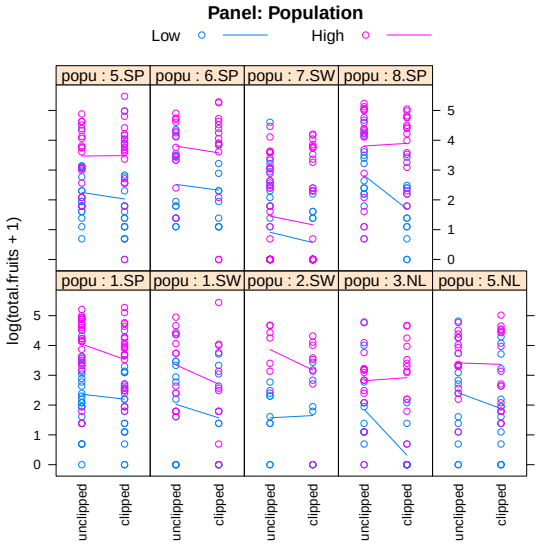

Figure 3: Interaction plots of total fruit response to nutrients and clipping, by population. Different lines connect population means of log-transformed data for different nutrient levels; colors represent nutrient levels (low=1=blue, high=8=pink).

10

There appears to be a consistent nutrient effect among genotypes (means in right panel higher than left panel), but the variation increases dramatically, obscuring the clipping effects and any possible interaction (Fig. 4).

[10] `ggplot` versions:

```
> ggplot(dat.tf,aes(x=amd,y=log(total.fruits+1),colour=nutrient))+
   geom_point()+
   ## need to use as.numeric(amd) to get lines
   stat_summary(aes(x=as.numeric(amd)),fun.y=mean,geom="line")+
   theme_bw()+opts(panel.margin=unit(0,"lines"))+
   facet_wrap(~popu)
> ggplot(dat.tf,aes(x=amd,y=log(total.fruits+1),colour=gen))+
   geom_point()+
   stat_summary(aes(x=as.numeric(amd)),fun.y=mean,geom="line")+
   theme_bw()+
   ## label_both adds variable name ('nutrient') to facet labels
   facet_grid(.~nutrient,labeller=label_both)
```

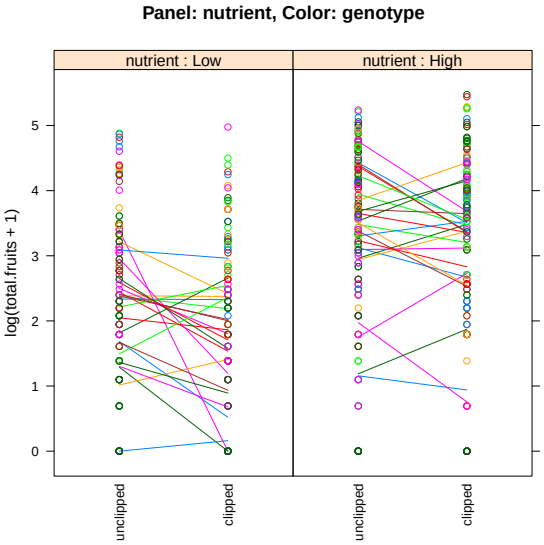

Figure 4: Interaction plots of total fruit response to clipping, for each genotype (lines and colors). Different panels show different nutrient levels.

5 Fitting group-wise GLMs

Another useful starting point is to fit GLMs to each group of data separately, equivalent to treating the grouping variables as fixed effects. Odd patterns in the parameters could call into question our assumption of a single underlying normal distribution in the parameters. In particular, groups with extreme parameter values or bimodal patterns in the parameter values could indicate a problem, although extreme estimated parameter values for poorly sampled groups are expected.

We first fit the models, and then examine the coefficients. We would normally use `glm` to fit a GLM, but we use `lmList` here because can fit a single GLM to many separate groups conveniently. We fit Poisson models rather than negative binomial models because `lmList` does not support negative binomial fits: we could use `family="quasipoisson"`, but it would not affect the parameter estimates (only the standard errors or confidence intervals), which we are not examining here. [11]

[11] If we insisted on fitting a negative binomial GLM we could most conveniently use `ldply` from the `plyr` package to split the data set by genotype and apply `glm.nb` from the `MASS` package to each chunk:

```
> library(plyr)
```

```
> glm.lis <- lmList(total.fruits~nutrient*amd|gen,data=dat.tf,
                    family="poisson")
```

12

There are a variety of ways to examine the different coefficients. We have written a function to save all the coefficients in a data frame, order the data frame by rank order of one of the coefficients, and then plot all the cofficients together (see Pinheiro and Bates (2000)); it was loaded when we ran `source("glmm_funs.R")`.

Plot the list of models created above:

```
> print(plot(glm.lis,scale=list(x=list(relation="free"))))
```

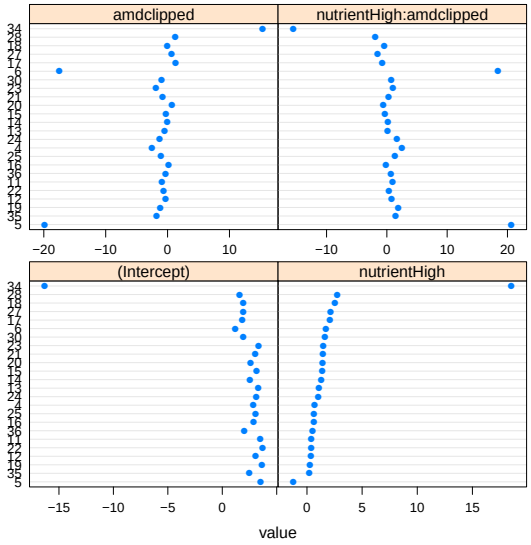

Figure 5: Model coefficients for GLM fits on each genotype.

Three genotypes (5, 6, 34) have extreme coefficients (Fig. 5).

```
> glmnbfitlist <- dlply(dat.tf,"gen",MASS::glm.nb,formula=total.fruits~nutrient*amd)
> coefmat <- t(sapply(glmnbfitlist,coef))
```

[12]The versions of `lmList` in the `lme4` and `nlme` packages are different (only the one in `lme4` has a `family` argument for running GLMs), and the `RLRsim` package loads the `nlme` package. You should `detach("package:RLRsim"); detach("package:nlme")` before running this command if you have loaded `RLRsim` in the meantime ...

A mixed model assumes that the underlying random effects are normally distributed, although we note that we are not actually plotting the random effects, or even estimates of random effects (which are not themselves guaranteed to be normally distributed), but rather separate estimates for each group.

The `glmm_funs.R` file contains a plotting function (specifically, a `qqmath` method) that draws Q-Q plots based on these fits to visualize the departure from normality (Figure 6):

```
> print(qqmath(glm.lis))
```

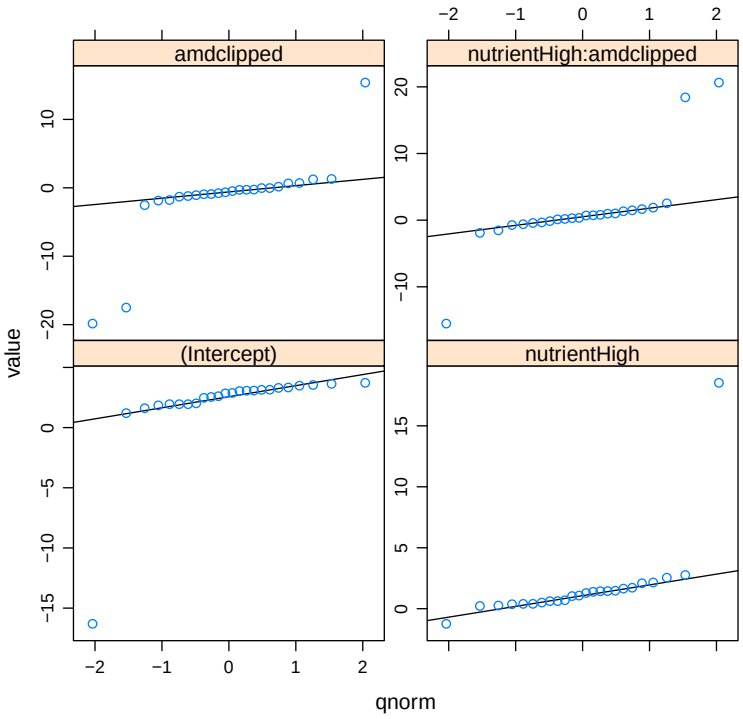

Figure 6: Q-Q plots of model coefficients for GLM fits on each genotype.

We see that these extreme coefficients fall far outside a normal error distribution (Fig. 6). As mentioned above, these are not estimates of random effects, however, but rather separate estimates for each group. Especially if these groups have relatively small sample sizes, the estimates may eventually be "shrunk" closer to the mean when we do the mixed model. Shrinkage, the

tendency of extreme (and poorly estimated) fixed effect parameters to be estimated as less extreme in a mixed model, is a general property of mixed models (Pinheiro and Bates, 2000, p. 152).

It turns out that two out of three of the outlier genotypes have small sample sizes, although the third is near the maximum sample size:

```
> gentab <- with(dat.tf,table(gen))
> summary(as.numeric(gentab))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  11.00   14.00   25.00   26.04   35.00   47.00

> gentab[c("5","6","34")]

gen
 5  6 34
11 13 45
```

We should nonetheless take care to see if the coefficients for these genotypes from the GLMM are still outliers, and take the same precautions as we usually do for outliers. For example, we can look back at the original data to see if there is something weird about the way those genotypes were collected, or try re-running the analysis without those genotypes to see if the results are robust.[13]

6 Fitting and evaluating the full GLMM

Now we (try to) build and fit a full model, using `glmer` in the `lme4` package[14]. This model has random effects for all genotype and population × treatment random effects, and for the nuisance variables for the rack and germination method (`status`). (Given the mean-variance relationship we saw it's pretty clear that we are going to have to proceed eventually to a model with overdispersion, but we fit the Poisson model first for illustration.)

[13] Other options such as weakening the distributional assumptions on the random effects (making them completely nonparametric or using a fatter-tailed distribution such as the Student t) are beyond the scope of this note, although they can be achieved via general-purpose tools such as WinBUGS or AD Model Builder.

[14] As of this writing, `glmer` and `lmer` are nearly synonymous: calling `lmer` with a `family` argument automatically hands off to `glmer` to run a GLMM. This may change in the future, and using `glmer` makes it clearer that one is really trying to run a GLMM rather than a LMM.

```
> mp1 <- glmer(total.fruits ~ nutrient*amd +
              rack + status +
              (amd*nutrient|popu)+
              (amd*nutrient|gen),
              data=dat.tf, family="poisson")
```

This fits without any (apparent) problems — the model converges without warnings.

We can use the Pearson residuals (i.e., $r_P = (y_i - u_i)/\sqrt{Var_i}$, where the variance is λ and is therefore the fitted value for each observation) to assess overdispersion quantitatively, although it's really unnecessary in this case because the pattern (Figure 2) is so obvious: if r_{Pi} are the Pearson residuals and d is the number of residual degrees of freedom, we want $\sum r_{Pi}^2 \approx d$ — or $\sum r_{Pi}^2/d \approx 1$, or more precisely, $\sum r_{Pi}^2 \sim \chi_d^2$ (Venables and Ripley, 2002, p. 208: but note warnings about applying this criterion for data with small sample means).

We use a convenience function, `overdisp_fun`, defined in `glmm_funs.R`:

```
> overdisp_fun(mp1)

      chisq       ratio           p
15077.17371    25.21267     0.00000
```

This shows that the data are (extremely) over-dispersed, given the model (the p-value is so low — less than 10^{-308} — that R just reports 0).

The *deviance* of the model (in this case, a penalized version of -2 times the log-likelihood) gives a similar value (also approximate for the purposes of inference):

```
> deviance(mp1)

[1] 14353.71
```

Now we add the observation-level random effect to the model to account for overdispersion, which gives us a lognormal-Poisson model for the individual samples (Elston et al., 2001):

```
> mp2 <- update(mp1,.~.+(1|X))
```

We get a warning that the `Number of levels of a grouping factor for the random effects is *equal* to n, the number of observations`, but we know this — we did it on purpose. The model also takes much longer to fit.

We start by looking at the variance components. In particular, if we look at the correlation matrix among the genotype random effects, we see a perfect correlation:

```
> attr(VarCorr(mp2)$gen,"correlation")
```

```
                     (Intercept) amdclipped nutrient8 amdclipped:nutrient8
(Intercept)                    1         -1        -1                     1
amdclipped                    -1          1         1                    -1
nutrient8                     -1          1         1                    -1
amdclipped:nutrient8           1         -1        -1                     1
```

The `printvc` is a utility function that prints the variance-covariance matrices along with an equals sign (=) appended to any covariance component that corresponds to a perfect (or nearly perfect ± 1 correlation):

```
> printvc(mp2)
```

```
X:
            (Intercept)
(Intercept) 2

gen:
                     (Intercept) amdclipped nutrient8 amdclipped:nutrient8
(Intercept)           0.263
amdclipped           -0.060=      0.014
nutrient8            -0.198=      0.045=     0.150
amdclipped:nutrient8  0.081=     -0.019=    -0.062=    0.025

popu:
                     (Intercept) amdclipped nutrient8 amdclipped:nutrient8
(Intercept)           0.4857
amdclipped            0.0886=     0.0162
nutrient8             0.0880=     0.0160=    0.0159
amdclipped:nutrient8 -0.0267=    -0.0049=   -0.0048=   0.0015
```

We'll try getting rid of the correlations between clipping (`amd`) and nutrients, using `amd+nutrient` instead of `amd*nutrient` in the random effects specification (here it seems easier to re-do the model rather than using `update` to add and subtract terms):

```
> mp3 <- glmer(total.fruits ~ nutrient*amd +
              rack + status +
              (amd+nutrient|popu)+
              (amd+nutrient|gen)+(1|X),
              data=dat.tf, family="poisson")
```

Unfortunately, we still have a problem with perfect correlations among the random effects terms:

```
> printvc(mp3)
```

```
X:
             (Intercept)
(Intercept) 2

gen:
             (Intercept) amdclipped nutrient8
(Intercept)  0.22485
amdclipped  -0.00739=    0.00024
nutrient8   -0.15141=    0.00498=   0.10195

popu:
             (Intercept) amdclipped nutrient8
(Intercept) 0.495
amdclipped  0.075=       0.011
nutrient8   0.079=       0.012=     0.013
```

For some models (e.g. random-slope models), it is possible to fit random effects models in such a way that the correlation between the different parameters (intercept and slope in the case of random-slope models) is constrained to be zero, by fitting a model like `(1|f)+(0+x|f)`; unfortunately, because of the way `lme4` is set up, this is considerably more difficult with categorical predictors (factors).

We have to reduce the model further in some way in order not to overfit (i.e., in order to avoid perfect ± 1 correlations among random effects). It looks like we can't allow both nutrients and clipping in the random effect model at either the population or the genotype level. However, it's hard to know whether we should proceed with `amd` or `nutrient` in the model.

A convenient way to proceed if we are going to try fitting several different combinations of random effects is to fit the model with all the fixed effects but only observation-level random effects, and then to use `update` to add various components to it:

```
> mp_obs <- glmer(total.fruits ~ nutrient*amd +
                  rack + status +
                  (1|X),
                  data=dat.tf, family="poisson")
```

Now, for example, `update(mp_obs,.~.+(1|gen)+(amd|popu))` fits the model with intercept random effects at the genotype level and variation in clipping effects across populations. [15]

It is not too hard to discover by fitting various models such as

```
> amd_gen_amd_popu  <- update(mp_obs,. ~ . +(amd|gen)+(amd|popu))
```

that *every* model more complicated than a simple random intercept at the genotype and population levels (`(1|gen)+(1|popu)`) is subject to problems with zero variances or perfect correlations among variance components. Thus we used this genotype-intercept+population-intercept model for the rest of our analysis. [16] In other words, while it's biologically plausible that there is some variation in the nutrient or clipping effect at the genotype or population levels, and while this variation may be what we are most interested in, it seems that we really don't have enough data to speak confidently about these effects.

Fit the genotype+population-intercept model:

[15] One can also model the interaction of categorical predictors (such as clipping) with a random effect in a slightly more restricted way, for example coding the `(amd|popu)` term in the model above as `(1|popu/amd)` instead. In effect, we are treating the different `amd` levels as nested groups within populations. This approach is slightly less flexible (for example, one cannot have negative correlations between treatment levels) but slightly easier computationally. In this case, trying to fit models with `amd` or `nutrient` nested within population or genotype leads to zero variances for the lowest level — in other words, just as we concluded before, we don't have enough power to detect variation in treatment effects among groups.

[16] In our original analysis of this data, we (slightly desperately) decided to fit a whole series of models with `amd`, `nutrient`, just an intercept, or no random effect in either the population or the genotype model, leading to a total of 24 models. We ran all 24 models in a batch file, and saved the whole set as a list (`mp_fits`) with appropriate names; we did some magic to generate the set of 24 models automatically, but we could also have done it by brute force enumeration, for example:

```
> amd_gen_amd_popu  <- update(mp_obs,. ~ . +(amd|gen)+(amd|popu))
> amd_gen_nut_popu  <- update(mp_obs,. ~ . +(amd|gen)+(nutrient|popu))
```

(*et cetera* ...)

We then used AIC (although there are some issues with applying AIC to mixed models (Greven, 2008; Greven and Kneib, 2010), it is a reasonable rough-and-ready indication of the relative goodness of fit of different models) to keep models with $\Delta\text{AIC} > 10$ and without fitting problems ... which ended up being only a few of the simplest ones.

```
> mp4 <- update(mp_obs,. ~ . + (1|gen) + (1|popu))
```

All three levels (observation, genotype, population) have non-zero variance, and there is no possibility of correlation since we are fitting only a single random effect at each level:

```
> printvc(mp4)

X:
             (Intercept)
(Intercept) 2.1

gen:
             (Intercept)
(Intercept) 0.082

popu:
             (Intercept)
(Intercept) 0.65
```

Testing overdispersion:

```
> overdisp_fun(mp4)

     chisq      ratio          p
97.7999099  0.1590242  1.0000000
```

The reduced model (`mp4`) now meets the residual deviance criterion (in fact, the residual deviance is now *smaller* than expected).

Next we apply a function from `glmm_funs.R` that defines a location-scale plot (Figure 7: this is plot #3 in the standard `plot.lm` sequence). The variance pattern is slightly alarming, with higher variance for small fitted values. Some but not all of this pattern is generated by zeros in the original data, so later on (in Part 2) we'll try fitting a zero-inflated model.

```
> locscaleplot(mp4,col=ifelse(dat.tf$total.fruits==0,"blue","black"))
```

7 Inference

Now that we have a full model, we want to make inferences about the parameters.

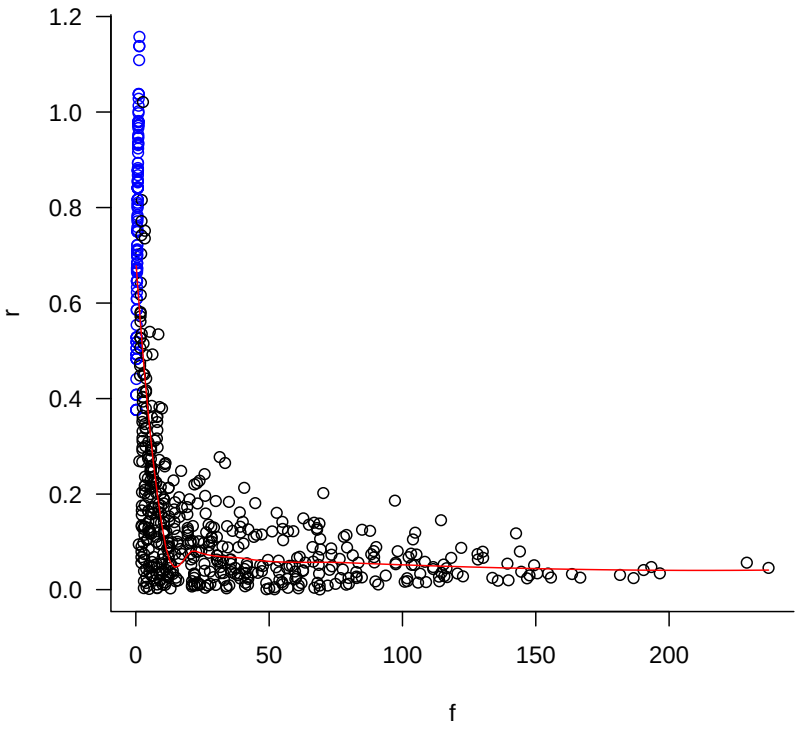

Figure 7: Location-scale plot, with superimposed loess fit. Zeros in the data are marked in blue.

7.1 Random effects

We could use `coefplot2(mp4,ptype="vcov",intercept=TRUE)` to look at a graphical representation of the variance terms (without error bars, because `glmer` doesn't give us information about the uncertainty of the variance terms), but in this model with only three variance components it's not much more informative than printing the values, as we did above with `printvc(mp4)`.

If we want to test the significance of the random effects we can fit reduced models and run likelihood ratio tests via `anova`, keeping in mind that in this case (testing a null hypothesis of zero variance, where the parameter is on the boundary of its feasible region) the reported p value is approximately

twice what it should be (Pinheiro and Bates, 2000):[17]

```
> mp4v1 <- update(mp_obs,.~.+(1|popu)) ## popu only (drop gen)
> mp4v2 <- update(mp_obs,.~.+(1|gen))  ## gen only (drop popu)

> anova(mp4,mp4v1)

Data: dat.tf
Models:
mp4v1: total.fruits ~ nutrient + amd + rack + status + (1 | X) + (1 |
mp4v1:     popu) + nutrient:amd
mp4: total.fruits ~ nutrient + amd + rack + status + (1 | X) + (1 |
mp4:     gen) + (1 | popu) + nutrient:amd
      Df    AIC    BIC  logLik  Chisq Chi Df Pr(>Chisq)
mp4v1  9 2654.4 2694.3 -1318.2
mp4   10 2652.3 2696.7 -1316.2 4.0637      1    0.04381 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> anova(mp4,mp4v2)

Data: dat.tf
Models:
mp4v2: total.fruits ~ nutrient + amd + rack + status + (1 | X) + (1 |
mp4v2:     gen) + nutrient:amd
mp4: total.fruits ~ nutrient + amd + rack + status + (1 | X) + (1 |
mp4:     gen) + (1 | popu) + nutrient:amd
      Df    AIC    BIC  logLik  Chisq Chi Df Pr(>Chisq)
mp4v2  9 2668.5 2708.4 -1325.2
mp4   10 2652.3 2696.7 -1316.2 18.174      1  2.016e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

For various forms of linear mixed models, the `RLRsim` package can do efficient simulation-based hypothesis testing of variance components — unfortunately, that doesn't include GLMMs. If we are sufficiently patient we can do hypothesis testing via brute-force parametric bootstrapping where

[17] The `pchibarsq` function in the `emdbook` package can be used for these kinds of tests, although in this case it's overkill (we get the same answer by simply dividing the p values by 2)

we repeatedly simulate data from the reduced (null) model, fit both the reduced and full models to the simulated data, and compute the distribution of the deviance (change in -2 log likelihood). The code below took about half an hour on a reasonably modern desktop computer:

```
> simdev <- function() {
   newdat <- simulate(mp4v1)
   reduced <- refit(mp4v1,newdat)
   full <- refit(mp4,newdat)
   2*(c(logLik(full)-logLik(reduced)))
 }
> library(plyr)
> set.seed(101)
> ## raply is a convenient wrapper for repeating the simulation many times
> nulldist <- raply(200,simdev(),.progress="text")
> ## zero spurious (small) negative values
> nulldist[nulldist<0 & abs(nulldist)<1e-5] <- 0
```

Calculate the observed statistic and compare it to the null distribution (including the observed value in the null distribution, because under the null hypothesis it would be in the null distribution):

```
> obsdev <- 2*c(logLik(mp4)-logLik(mp4v1))
> mean(c(nulldist,obsdev)>=obsdev)

[1] 0.04975124
```

The true p-value is actually closer to 0.05 than 0.02. In other words, here the deviations from the original statistical model from that for which the original "p value is inflated by 2" rule of thumb was derived — fitting a GLMM instead of a LMM, and using a moderate-sized rather than an arbitrarily large (asymptotic) data set — have made the likelihood ratio test liberal (increased type I error) rather than conservative (decreased type I error).

We can visualize the distribution by drawing a Q-Q plot, comparing the simulated distribution to the the theoretically expected $\bar{\chi}^2_1$ distribution rather than the normal distribution (Figure 8):

```
> library(emdbook)
> print(qqmath(c(nulldist,obsdev),
              distribution=qchibarsq,
```

```
aspect="iso",
xlab="theoretical",
ylab="simulated",
panel = function(x, ...) {
  panel.abline(a=0,b=1,col="gray")
  panel.qqmath(x, ...)
  panel.abline(v=obsdev,lty=2)
  panel.abline(h=obsdev,lty=2)
}))
```

We can see that the null distribution has more large values than expected under the theoretical distribution.

We can also inspect the random effects estimates themselves (in proper statistical jargon, these might be considered "predictions" rather than "estimates" (Robinson, 1991): Figure 9). We use the built-in `dotplot` method for the random effects extracted from `glmer` fits (i.e. `ranef(model,postVar=TRUE)`), which returns a list of plots, one for each random effect level in the model. (We used a bit of trickery from the `gridExtra` package to arrange these plots on the page, and we omit the observation-level random effects, which are returned as the `$obs` component of the list of plots.)

```
> ## squash margins a bit ...
> pp <- list(layout.widths=list(left.padding=0, right.padding=0),
            layout.heights=list(top.padding=0, bottom.padding=0))
> r2 <- ranef(mp4,postVar=TRUE)
> d2 <- dotplot(r2,  par.settings=pp)
> library(gridExtra)
> grid.arrange(d2$gen,d2$popu,nrow=1)
```

As expected from the similarity of the variance estimates, the population-level estimates (the only shared component) do not differ much between the two models. There is a hint of regional differentiation — the Spanish populations have higher fruit sets than the Swedish and Dutch populations. Genotype 34 again looks a little bit unusual.

7.2 Fixed effects

Now we want to do inference on the fixed effects. We use the `drop1` function to assess both the AIC difference and the likelihood ratio test between models. (In `glmm_funs.R` we define a convenience function `dfun` to convert the AIC tables returned by `drop1` (which we will create momentarily)

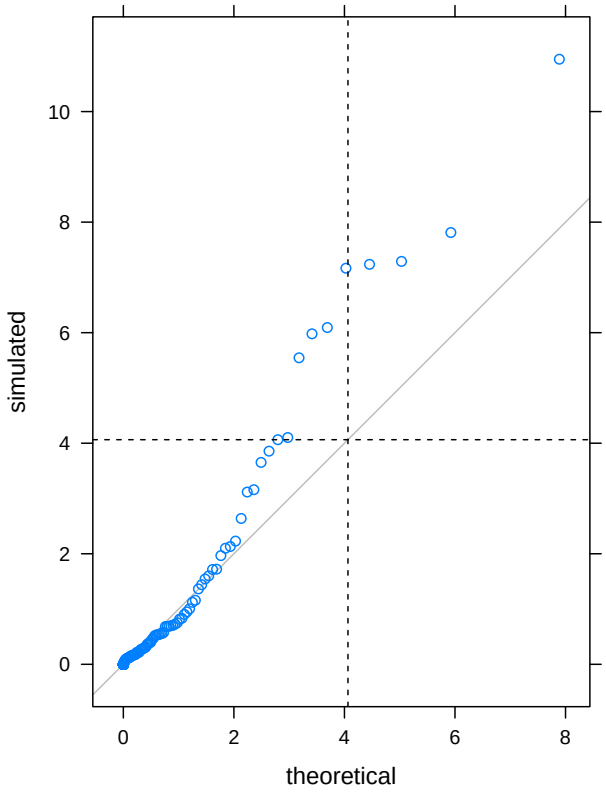

Figure 8: Q-Q plot of the simulated null deviance between the models with and without a variance component at the genotype level, compared against the $\bar{\chi}^2_1$ distribution (Goldman and Whelan, 2000)

into ΔAIC tables.) Although the likelihood ratio test (and the AIC) are asymptotic tests, comparing fits between full and reduced models is still more accurate than the Wald (curvature-based) tests shown in the `summary` tables for `glmer` fits.[18]

```
> (dd_AIC <- dfun(drop1(mp4)))
```

[18]In general you should *not* explore both AIC- and LRT-based comparisons when analyzing a models; they represent different inferential approaches, and you should decide before analyzing the model which you prefer, to avoid the temptation of data snooping or "cherry-picking" (i.e., comparing the results of the two approaches and choosing the ones you prefer). Here we show both analyses for illustration.

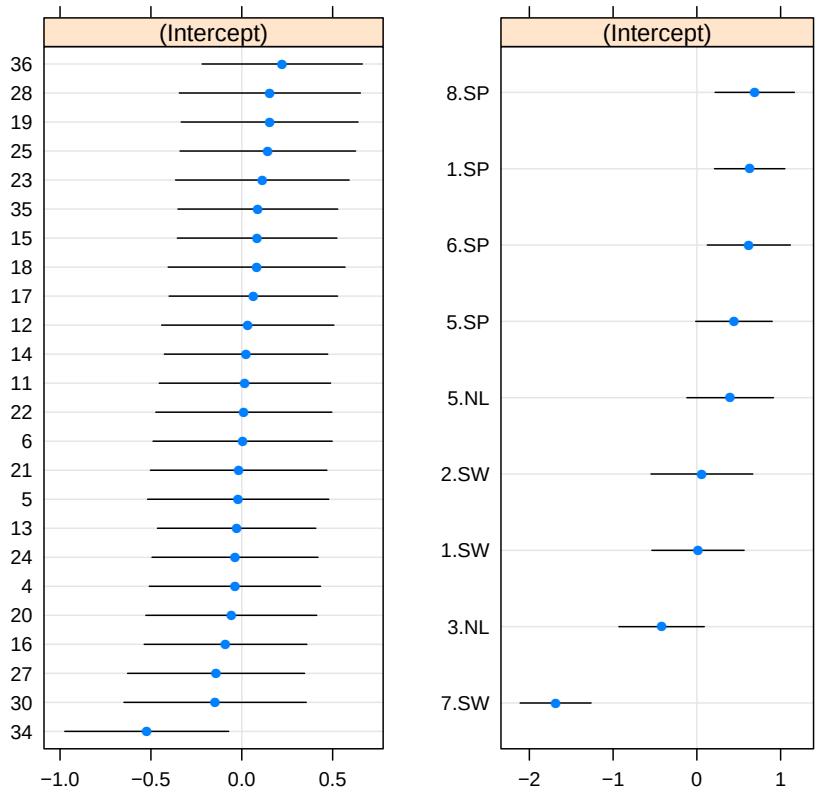

Figure 9: Random effects estimates/predictions for the final model

```
Single term deletions

Model:
total.fruits ~ nutrient + amd + rack + status + (1 | X) + (1 |
    gen) + (1 | popu) + nutrient:amd
             Df   dAIC
<none>           0.000
rack          1 55.087
status        2  1.613
nutrient:amd  1  1.446

> (dd_LRT <- drop1(mp4,test="Chisq"))
```

```
Single term deletions

Model:
total.fruits ~ nutrient + amd + rack + status + (1 | X) + (1 |
    gen) + (1 | popu) + nutrient:amd
             Df    AIC    LRT   Pr(Chi)
<none>          2652.3
rack          1 2707.4 57.087 4.169e-14 ***
status        2 2653.9  5.613   0.06040 .
nutrient:amd  1 2653.8  3.446   0.06339 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

On the basis of these comparisons, there appears to be a very strong effect of rack and weak effects of status and of the interaction term. Dropping the `nutrient:amd` interaction gives a (slightly) increased AIC (ΔAIC = 1.4), so the full model has the best expected predictive capability (by a small margin). On the other hand, the p-value is slightly above 0.05 ($p = 0.06$).

At this point we remove the non-significant interaction term so we can test the main effects. [19] (We don't worry about removing `status` because it measures an aspect of experimental design that we want to leave in the model whether it is significant or not.)

Once we have fitted the reduced model, we can run the LRT via `anova`:

```
> mp5 <- update(mp4, . ~ . - amd:nutrient)

> anova(mp5,mp4)

Data: dat.tf
Models:
mp5: total.fruits ~ nutrient + amd + rack + status + (1 | X) + (1 |
mp5:     gen) + (1 | popu)
mp4: total.fruits ~ nutrient + amd + rack + status + (1 | X) + (1 |
mp4:     gen) + (1 | popu) + nutrient:amd
```

[19] There is significant controversy over this point. Some researchers state that centering the input variables (in this case using `contr.sum` to change to "sum to zero" contrasts) will make the main effects interpretable even in the presence of the interaction term Schielzeth (2010), while others say one should almost never violate the "principle of marginality" in this way (Venables, 1998). Dropping the interactions may be an appropriate way forward (Pinheiro and Bates, 2000), although others say that dropping non-significant terms from models is almost never justified (Harrell, 2001; Whittingham et al., 2006) ...

```
    Df    AIC    BIC  logLik  Chisq Chi Df Pr(>Chisq)
mp5  9 2653.8 2693.7 -1317.9
mp4 10 2652.3 2696.7 -1316.2 3.4465      1    0.06339 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

We can also test all the reduced models:

```
> dd_AIC2 <- dfun(drop1(mp5))

Single term deletions

Model:
total.fruits ~ nutrient + amd + rack + status + (1 | X) + (1 |
    gen) + (1 | popu)
         Df    dAIC
<none>        0.000
nutrient  1 135.700
amd       1  10.221
rack      1  54.236
status    2   1.287

> dd_LRT2 <- drop1(mp5,test="Chisq")

Single term deletions

Model:
total.fruits ~ nutrient + amd + rack + status + (1 | X) + (1 |
    gen) + (1 | popu)
         Df    AIC     LRT   Pr(Chi)
<none>      2653.8
nutrient  1 2789.5 137.700 < 2.2e-16 ***
amd       1 2664.0  12.221 0.0004727 ***
rack      1 2708.0  56.236 6.426e-14 ***
status    2 2655.1   5.287 0.0711227 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

In the reduced model, we find that both nutrients and clipping have strong effects, whether measured by AIC or LRT.

If we wanted to be still more careful about our interpretation, we would try to relax the asymptotic assumption. In classical linear models, we

would do this by doing F tests with the appropriate denominator degrees of freedom. In "modern" mixed model approaches, we might try to use denominator-degree-of-freedom approximations such as the Kenward-Roger (despite the controversy over these approximations, they are actually available in the `doBy` package (see `www.warwick.ac.uk/statsdept/useR-2011/abstracts/290311-halekohulrich.pdf`)), but they do not apply to GLMMs.

We can use a parametric bootstrap comparison between nested models to test fixed effects, as we did above for random effects, with the caveat that is computationally slow. (Here we illustrate how to use functions in the `doBy` package rather than the home-grown functions we used above.)

```
> library(doBy)
```

The following command took about 35 minutes:

```
> pb1 <- PBrefdist(mp4,mp5)
```

Once we have the null distribution, doing inference with it is quick.

```
> PBmodcomp(mp4,mp5,ref=pb1)
```

```
large : total.fruits ~ nutrient + amd + rack + status + (1 | X) + (1 |
    gen) + (1 | popu) + nutrient:amd
<environment: 0xc72fc58>
small : total.fruits ~ nutrient + amd + rack + status + (1 | X) + (1 |
    gen) + (1 | popu)
<environment: 0xb53ce90>
           tobs df         p
LRT    3.446454  1 0.0633876
PBtest 3.446454 NA 0.0950000
```

This compares the likelihood ratio and parametric bootstrap tests: the parametric bootstrap test is slightly more conservative, but both tests have $0.05 < p < 0.1$.

At this point, we could re-check our assumptions.

```
> overdisp_fun(mp5)
```

```
     chisq      ratio          p
97.4751757  0.1582389  1.0000000
```

If we did `locscaleplot(mp5)` we would see very much the same pattern as in Figure 7.

In addition, we can check the normality of the random effects and find they are reasonable (Fig. 10). We use `ldply` from the `reshape` package to collapse the list of random effect values into a data frame (we might have to do something different if there were more than one random effect within each level, e.g. a model including `(nutrient|gen)`). The fancy `panel` code in the figure adds a reference line to the Q-Q plot: see `?qqmath`.

```
> reStack <- ldply(ranef(mp5))
> print( qqmath( ~`(Intercept)`|.id, data=reStack, scales=list(relation="free"),
                 prepanel = prepanel.qqmathline,
                 panel = function(x, ...) {
                   panel.qqmathline(x, ...)
                   panel.qqmath(x, ...)
                 },
                   layout=c(3,1)))
```

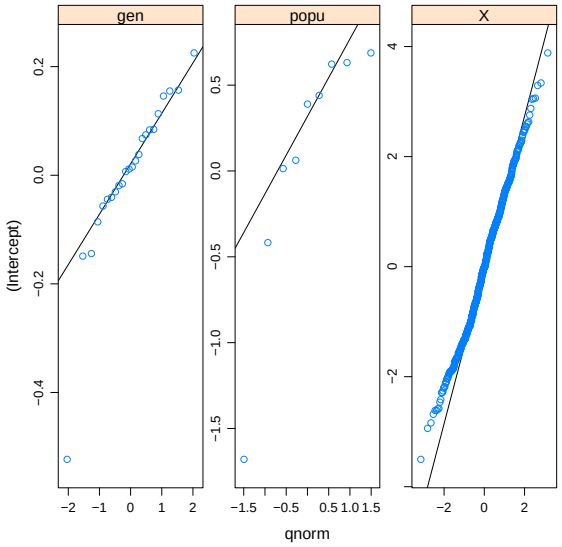

Figure 10: Quantile-quantile plot of the genotype random effects of the final model.

8 Conclusions

Our final model includes fixed effects of nutrients and clipping, as well as the nuisance variables `rack` and `status`; observation-level random effects to account for overdispersion; and variation in overall fruit set at the population and genotype levels. However, we don't (apparently) have quite enough information to estimate the variation in clipping and nutrient effects, or their interaction, at the genotype or population levels. There is a strong overall positive effect of nutrients and a slightly weaker negative effect of clipping. The interaction between clipping and nutrients is only weakly supported (i.e. the p-value is not very small), but it is positive and about the same magnitude as the clipping effect, which is consistent with the statement that "nutrients cancel out the effect of herbivory". To test this equivalence formally, we could set the analysis up as a one-way layout (with levels [`Nutrient1:Unclipped`, `Nutrient1:Clipped`, `Nutrient8:Clipped`, `Nutrient8:Unclipped`]); the contrast between `Nutrient1:Unclipped` and `Nutrient8:Clipped` would then test the hypothesis (see `http://glmm.wikidot.com/local--files/examples/culcita_glmm.pdf` for a similar example).

This concludes part 1. To see the comparisons of the same analysis with a variety of different estimation approaches using different R packages, see part 2.

Save results to use in part 2:

```
> save("dat.tf","mp4",file="Banta_part1.RData")
```

References

Banta, J. A., M. H. H. Stevens, and M. Pigliucci. 2010. A comprehensive test of the 'limiting resources' framework applied to plant tolerance to apical meristem damage. Oikos **119**:359–369. URL `http://onlinelibrary.wiley.com/doi/10.1111/j.1600-0706.2009.17726.x/abstract`.

Bates, D. and M. Maechler. 2010. lme4: Linear mixed-effects models using S4 classes. URL `http://CRAN.R-project.org/package=lme4`. R package version 0.999375-33.

Bolker, B. M., M. E. Brooks, C. J. Clark, S. W. Geange, J. R. Poulsen, M. H. H. Stevens, and J.-S. S. White. 2009. Generalized linear mixed models: a practical guide for ecology and evolution. Trends in Ecology and Evolution **24**:127–135.

Elston, D. A., R. Moss, T. Boulinier, C. Arrowsmith, and X. Lambin. 2001. Analysis of aggregation, a worked example: numbers of ticks on red grouse chicks. Parasitology **122**:563–569.

Goldman, N. and S. Whelan. 2000. Statistical tests of gamma-distributed rate heterogeneity in models of sequence evolution in phylogenetics. Molecular Biology and Evolution **17**:975–978.

Greven, S. 2008. Non-Standard Problems in Inference for Additive and Linear Mixed Models. Cuvillier Verlag, Göttingen, Germany. URL `http://www.cuvillier.de/flycms/en/html/30/-UickI3zKPS,3cEY=/Buchdetails.html?SID=wVZnpL8f0fbc`.

Greven, S. and T. Kneib. 2010. On the behaviour of marginal and conditional Akaike information criteria in linear mixed models. Biometrika **97**:773–789. URL `http://www.bepress.com/jhubiostat/paper202/`.

Hardin, J. W. and J. Hilbe. 2007. Generalized linear models and extensions. Stata Press.

Harrell, F. 2001. Regression Modeling Strategies. Springer.

McCullagh, P. and J. A. Nelder. 1989. Generalized Linear Models. Chapman and Hall, London.

Pinheiro, J. C. and D. M. Bates. 2000. Mixed-effects models in S and S-PLUS. Springer, New York.

R Development Core Team. 2009. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

Robinson, G. K. 1991. That BLUP is a good thing: The estimation of random effects. Statistical Science **6**:15–32. URL `http://www.jstor.org/stable/2245695`.

Schielzeth, H. 2010. Simple means to improve the interpretability of regression coefficients. Methods in Ecology and Evolution **1**:103–113. URL `http://dx.doi.org/10.1111/j.2041-210X.2010.00012.x`.

Venables, W. and B. D. Ripley. 2002. Modern Applied Statistics with S. Springer, New York. 4th edition.

Venables, W. N. 1998. Exegeses on linear models. 1998 International S-PLUS User Conference. Washington, DC. URL `http://www.stats.ox.ac.uk/pub/MASS3/Exegeses.pdf`.

Whittingham, M. J., P. A. Stephens, R. B. Bradbury, and R. P. Freckleton. 2006. Why do we still use stepwise modelling in ecology and behaviour? Journal of Animal Ecology **75**:1182–1189.