(Generalized) Linear Mixed Models

Simon Wood
Mathematical Sciences, University of Bath, U.K.

Generalized linear mixed model

- So far we have allowed very flexible models for the expected response and very simplistic models for its stochastic component. Let's fix that.
- A *Generalized linear mixed model* (GLMM) has the form

$$g(\mu_i) = \mathbf{X}_i\boldsymbol{\beta} + \mathbf{Z}_i\mathbf{b}, \quad \mathbf{b} \sim N(\mathbf{0}, \boldsymbol{\psi}_\theta), \quad y_i \sim \text{EF}(\mu_i, \phi)$$

- $\mathbf{Z}$ is a model matrix for the *random effects* $\mathbf{b}$.
- The parameters are $\boldsymbol{\beta}$, ϕ and $\boldsymbol{\theta}$, the latter parameterizing $\mathbf{b}$'s covariance matrix, $\boldsymbol{\psi}_\theta$.
- First consider finding estimates $\hat{\boldsymbol{\beta}}$ and posterior modes $\hat{\mathbf{b}}$, given ϕ and $\boldsymbol{\theta}$. $\hat{\phi}$ and $\hat{\boldsymbol{\theta}}$ will be covered subsequently.

$\hat{\boldsymbol{\beta}}$ and $\hat{\mathbf{b}}$ — Bayesian

- Treat $\boldsymbol{\beta}$ as random variables with improper uniform priors, and then find their posterior modes (MAP estimates).
- $f(\boldsymbol{\beta}, \mathbf{b}|\mathbf{y}) \propto f(\mathbf{y}|\boldsymbol{\beta}, \mathbf{b}) f(\boldsymbol{\beta}, \mathbf{b}) \propto f(\mathbf{y}|\boldsymbol{\beta}, \mathbf{b}) f(\mathbf{b})$ where $f(\mathbf{y}|\boldsymbol{\beta}, \mathbf{b})$ is determined by the EF used, and $f(\mathbf{b})$ is $N(\mathbf{0}, \boldsymbol{\psi}_\theta)$.
- Maximization of $f(\boldsymbol{\beta}, \mathbf{b}|\mathbf{y})$ is achievable by Penalized IRLS.
- Initialize $\hat{\eta}_i = g(y_i)$, then iterate the following steps.
  1. Form pseudodata z_i and weights w_i, exactly as for a GLM, except using linear predictor $\hat{\boldsymbol{\eta}} = \mathbf{X}\hat{\boldsymbol{\beta}} + \mathbf{Z}\hat{\mathbf{b}}$.
  2. Minimize the penalized weighted sum of squares $\sum_i w_i(z_i - \mathbf{X}_i\boldsymbol{\beta} - \mathbf{Z}_i\mathbf{b})^2/\phi + \mathbf{b}^\mathsf{T}\boldsymbol{\psi}_\theta^{-1}\mathbf{b}$ w.r.t. $\boldsymbol{\beta}, \mathbf{b}$ to obtain a new $\hat{\boldsymbol{\beta}}, \hat{\mathbf{b}}$, and hence new $\hat{\boldsymbol{\eta}}$ and $\hat{\boldsymbol{\mu}}$.
- In LMM case don't need to iterate, and $\hat{\mathbf{b}} = \widehat{E(\mathbf{b}|\mathbf{y})}$.

$\hat{\boldsymbol{\beta}}$ by MLE

- Integrate out $\mathbf{b}$ by Laplace approximation...

$$f_{\boldsymbol{\beta}}(\mathbf{y}) \simeq f_{\boldsymbol{\beta}}(\mathbf{y}, \hat{\mathbf{b}}) \frac{(2\pi)^{\dim(\mathbf{b})/2}}{|\mathbf{Z}^{\mathsf{T}}\mathbf{W}\mathbf{Z}/\phi + \boldsymbol{\psi}_{\theta}|^{-1/2}}$$

- This is exact for a LMM, in which case $\mathbf{W} = \mathbf{I}$ and $L(\boldsymbol{\beta}) = f_{\boldsymbol{\beta}}(\mathbf{y})$ is maximized by the MAP estimates.
- For other GLMMs MAP and MLE differ, but are close if $\mathbf{W}$ varies only 'slowly' with $\boldsymbol{\beta}$.
- Most users of GLMMs are Bayesians out of laziness, and use the MAP estimates. So will we ...

$\hat{\phi}$ and $\hat{\boldsymbol{\theta}}$ — Laplace approximation

- To estimate ϕ and $\boldsymbol{\theta}$ we need to integrate $\boldsymbol{\beta}$ and $\mathbf{b}$ out of $f_{\boldsymbol{\beta}}(\mathbf{y}, \mathbf{b})$, and optimize the result w.r.t. $\phi, \boldsymbol{\theta}$.
- The result of the integration is known as *marginal likelihood* or *restricted likelihood*, L_r.
- The integral can be obtained by Laplace approximation, with the resulting expression dependent on $\phi, \boldsymbol{\theta}$ via $\hat{\boldsymbol{\beta}}, \hat{\mathbf{b}}$ and $\boldsymbol{\psi}_\theta$ plus direct dependence on ϕ.
- $l_r = \log L_r$ can be optimized numerically w.r.t. $\phi, \boldsymbol{\theta}$, with each l_r evaluation requiring a PIRLS loop to find $\hat{\boldsymbol{\beta}}, \hat{\mathbf{b}}$.
- For a LMM the Laplace approximation is exact, and the PIRLS iteration is not needed.
- A cheaper, but less reliable method is PQL...

$\hat{\phi}$ and $\hat{\boldsymbol{\theta}}$ — PQL

- Consider the expression minimized at each PIRLS step:

$$Q = \sum_i w_i (z_i - \mathbf{X}_i \boldsymbol{\beta} - \mathbf{Z}_i \mathbf{b})^2 / \phi + \mathbf{b}^{\mathsf{T}} \boldsymbol{\psi}_\theta^{-1} \mathbf{b}$$

- $-Q/2$ is a (rough) quadratic approximation to the log of the part of $f_{\boldsymbol{\beta}}(\mathbf{y}, \mathbf{b})$ which needs to be integrated w.r.t. $\mathbf{b}$.
- It is also *exactly* the log of the part of $f_{\boldsymbol{\beta}}(\mathbf{y}, \mathbf{b})$ which would need to be integrated w.r.t. $\mathbf{b}$ for a weighted LMM.
- So, at each PIRLS step, why not estimate this weighted LMM to optimize Q w.r.t. $\boldsymbol{\beta}, \mathbf{b}$, **and** get estimates of ϕ and $\boldsymbol{\theta}$ into the bargain? No reason!
- This is 'PQL'. It works surprisingly well, except for binary data.

Distributional results

- Distributions for $\hat{\phi}, \hat{\boldsymbol{\theta}}$ are from large sample MLE theory applied to l_r or the PQL working model l_r.
- Distributions for $\hat{\boldsymbol{\beta}}$ (conditional on $\hat{\phi}, \hat{\boldsymbol{\theta}}$) are from MLE theory applied to $I(\hat{\boldsymbol{\beta}})$, if this is computed. (Conditionally exact for LMM).
- Alternatively use the large sample Bayesian result

$$\begin{bmatrix} \boldsymbol{\beta} \\ \mathbf{b} \end{bmatrix} \sim N\left(\begin{bmatrix} \hat{\boldsymbol{\beta}} \\ \hat{\mathbf{b}} \end{bmatrix}, \begin{bmatrix} \mathbf{X}^{\mathsf{T}}\mathbf{W}\mathbf{X} & \mathbf{X}^{\mathsf{T}}\mathbf{W}\mathbf{Z} \\ \mathbf{Z}^{\mathsf{T}}\mathbf{W}\mathbf{X} & \mathbf{Z}^{\mathsf{T}}\mathbf{W}\mathbf{Z} + \boldsymbol{\psi}_\theta/\phi \end{bmatrix}^{-1} \phi \right)$$

Model Comparison

- If fixed effects are the same in the alternative models, then l_r can be used for GLRT testing or AIC model comparison.
- If fixed effects differ then estimation has to be performed without integrating out $\boldsymbol{\beta}$: then GLRT and AIC can be used.
- When using the GLRT take care that the null model is not restricting alternative model parameters to the edge of the parameter space. If it is, the results are only a rough guide.
- PQL estimated model comparison is difficult. The working l_r is not really a valid basis for model comparison.

Residuals and fitted values

- We can produce fitted values at different *levels*, depending on which components of $\mathbf{b}$ are set to their unconditional mean 0, and which are set to their conditional mean $\mathbb{E}(b_i|\mathbf{y})$.
- At the highest level the fitted values are: $\hat{\boldsymbol{\mu}} = \mathbf{X}\hat{\boldsymbol{\beta}} + \mathbf{Z}\hat{\mathbf{b}}$
- The residuals $\hat{\epsilon} = \mathbf{y} - \hat{\boldsymbol{\mu}}$ are examined to check assumptions about $\boldsymbol{\epsilon}$.
- $\hat{\mathbf{b}}$ is examined to check the random effects assumptions. Note that if $\mathbf{b} \sim N(\mathbf{0}, \boldsymbol{\psi}_\theta)$ then $\sqrt{\boldsymbol{\psi}_\theta{}^{-1}}\mathbf{b} \sim N(\mathbf{0}, \mathbf{I})$. Standardizing $\hat{\mathbf{b}}$ in the same way is a help when model checking.

Mixed models in R

- Three R packages provide the mixed modelling methods described above.
- Recommended package `nlme` provides function `lme` for Linear Mixed Modelling. It is particularly useful when the random effects have a nested structure.
- Recommended package bundle `MASS` provides a function `glmmPQL` for fitting generalized linear mixed models using PQL, based on iterative calls to `lme`.
- Doug Bates' package `lme4` provides functions `lmer` (`glmer`) for fitting (generalized) linear mixed models. In the generalized case Laplace approximation is used.

lme

- `nlme` provides linear mixed model functions for R.
- The linear mixed model fitting function is `lme`.
- `lme` is used in a similar way to `lm`.
  - A model formula specifies the response (on the left) and the fixed effects model structure (on the right).
  - A `data` argument is used to pass a data frame containing the data to be modelled.
  - A fitted model object is returned

  But the random effects model must also be specified
  - A model formula, or list of model formulae, specifies the random effects model structure.

`lme` mixed model form

- `lme` assumes that your data are *grouped* and that you want the following model for the i^{th} group.

$$\mathbf{y}_i = \mathbf{X}_i\boldsymbol{\beta} + \mathbf{Z}_i\mathbf{b}_i + \boldsymbol{\epsilon}_i, \qquad \mathbf{b}_i \sim N(\mathbf{0}, \boldsymbol{\psi}_\theta), \qquad \boldsymbol{\epsilon}_i \sim N(\mathbf{0}, \boldsymbol{\Lambda}).$$

  where the $\mathbf{b}_i$ are independent between groups.
- Note what varies with group (e.g. $\mathbf{b}_i$), and what does not (i.e. $\boldsymbol{\theta}$, $\boldsymbol{\beta}$ and $\boldsymbol{\Lambda}$).
- $\boldsymbol{\Lambda}$ is often $\mathbf{I}\sigma^2$, but it may have a more complicated structure, to allow residual correlation within groups.
- The model is a special case of the general model considered thus far.
- If we have just one group then the general model is recovered.

Calling `lme`

- Because of the assumed model structure, two parts must be supplied for the random effects specification.
  1. A grouping factor (or factors) indicating how the data are divided into groups.
  2. A model formula (or formulae) specifying the random effects model matrix for each group.
- For example (assuming x and z are not factors)

  `lme(y~x+z,dat,~z|g)`

  fits the model, $\mathbf{b} \sim N(\mathbf{0}, \mathbf{I}\sigma_b^2)$ and

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 z_i + b_j z_i + \epsilon_i \text{ if } g_i = j.$$

Specifying random model

- An alternative to

  ```
  lme(y~x+z,dat,~z|g)
  ```

  is

  ```
  lme(y~x+z,dat,list(g=~z))
  ```

- In both cases the *formula*, `~z`, specifies the random effect model matrix used at each level of the *grouping factor*, `g`.
- Nested groups are supported. e.g. `~z+x/g/f` would repeat the same *random effects structure* at each level of `g` and each combination of levels of `g` and `f`.
- `list(g=~z+x,f=~z+x)` does the same, but also allows *different* random effects structures at each grouping level.

Simple example:`Rail`

- 6 Railway rails were each tested 3 times, by sending an ultrasonic pulse along the rail, and measuring the time it takes.
- Data are in data frame `Rail`, which contains variables `Rail` ID and `travel` time.
- A suitable model has a random effect for each rail and a fixed overall mean travel time, so that,

$$y_i = \beta + b_j + \epsilon_i \text{ if } y_i \text{ relates to rail } j. \quad \mathbf{b} \sim N(\mathbf{0}, \mathbf{I}_6\sigma^2).$$

- In `lme` terms `Rail` is the grouping factor. **Exercise**: write out the model for rail i in standard `lme` form.

Rail model: lme form

The model for the i^{th} rail is simply

$$\mathbf{y}_i = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} \beta \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} b_i \end{bmatrix} + \boldsymbol{\epsilon}_i$$

where $b_i \sim N(0, \sigma_b^2)$ and the components of $\boldsymbol{\epsilon}_i$ are i.i.d. $N(0, \sigma^2)$. β, σ_b^2 and σ^2 are the parameters to be estimated.

Fitting the Rail model

```
> library(nlme)
> rm <- lme(travel~1,Rail,list(Rail=~1))
> rm
Linear mixed-effects model fit by REML
  Data: Rail
  Log-restricted-likelihood: -61.0885
  Fixed: travel ~ 1
(Intercept)
       66.5
Random effects:
 Formula: ~1 | Rail
        (Intercept) Residual
StdDev:    24.80547 4.020779
Number of Observations: 18
Number of Groups: 6
```

summary(rm)

```
> summary(rm)
Linear mixed-effects model fit by REML
 Data: Rail
      AIC      BIC   logLik
  128.177 130.6766 -61.0885
Random effects:              # sigma_b and sigma
 Formula: ~1 | Rail
        (Intercept) Residual
StdDev:    24.80547 4.020779
Fixed effects: travel ~ 1    # beta
            Value Std.Error DF  t-value p-value
(Intercept)  66.5  10.17104 12 6.538173       0
...
```

Confidence intervals: `intervals`

```
> intervals(rm)
Approximate 95% confidence intervals
 Fixed effects:                    ## beta
               lower est.    upper
(Intercept) 44.33921 66.5 88.66079
 Random Effects:                   ## sigma_b
  Level: Rail
                   lower     est.    upper
sd((Intercept)) 13.27434 24.80547 46.35341
 Within-group standard error:      ## sigma
   lower     est.    upper
2.695007 4.020779 5.998747
```

predict method for lme

predict method similar to that for lm, but

- no se argument.
- can control the level at which to predict

```
> predict(rm) ## default highest level: use E(b|y) for all b
       1        1        1        2        2        2        3
54.10852 54.10852 54.10852 31.96909 31.96909 31.96909 84.50894
       3        3        4        4        4        5        5
84.50894 84.50894 95.74388 95.74388 95.74388 50.14325 50.14325
       5        6        6        6
50.14325 82.52631 82.52631 82.52631

> predict(rm,level=0) ## set level=0: E(b)=0 used for all b
   1    1    1    2    2    2    3    3    3    4    4    4    5
66.5 66.5 66.5 66.5 66.5 66.5 66.5 66.5 66.5 66.5 66.5 66.5 66.5
...
```

`Loblolly` pine example

- `Loblolly` data frame contains `height` of a number of Loblolly pine trees at different `ages`.
- We expect some tree-to-tree variability from a mean growth trajectory, plus auto-correlation for within tree measurements.
- A possible model is

$$\texttt{height}_{ji} = \beta_0 + \beta_1 \texttt{age}_{ji} + \beta_2 \texttt{age}_{ji}^2 + \beta_3 \texttt{age}_{ji}^3 + b_0 + b_{j1} \texttt{age}_{ji} + b_{j2} \texttt{age}_{ji}^2 + b_{j3} \texttt{age}_{ji}^3 + \epsilon_{ji}$$

the ϵ_{ji} are zero mean normal random variables, with within tree correlation given by $\rho(\epsilon_{j,i}, \epsilon_{j,i-1}) = \phi$. $\mathbf{b}_j \sim N(0, \boldsymbol{\psi})$

Fitting problems

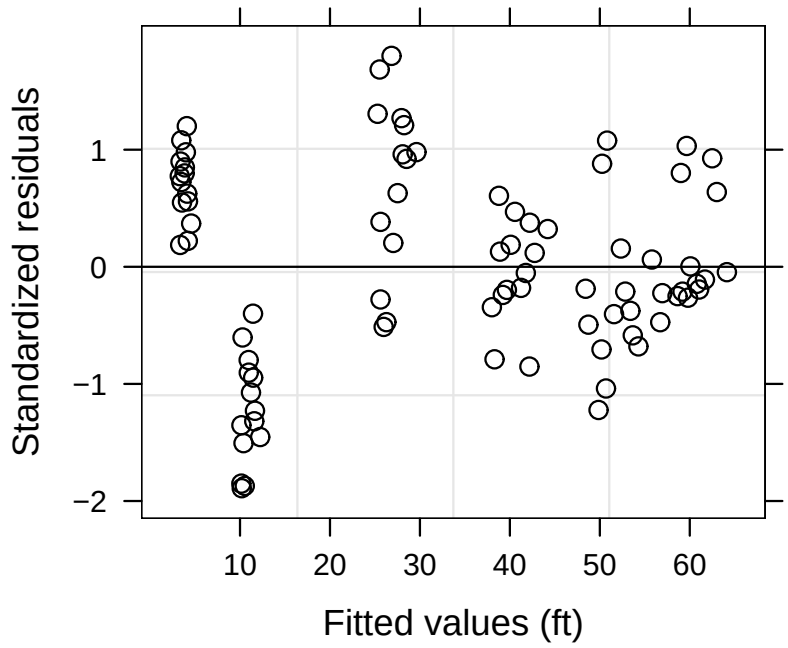
- The `lme` optimizer sometimes fails, but the situation can usually be rectified by changing some of the control parameters for optimization, via e.g.

```
lme(...,control=lmeControl(msMaxIter=100,niterEM=1000))
```

- `msMaxIter` controls the maximum number of Newton iterations used in MLE or REML.
- `niterEM` controls the number of EM algorithm steps used to find initial values for θ, before using Newton's method. Non-convergence can often be eliminated by increasing this, but other failures may require it to be reduced (especially errors mentioning `MEEM` or `NaNs`).
- For a full list of control constants see `?lmeControl`.
- It can also help to experiment with starting values for parameters.

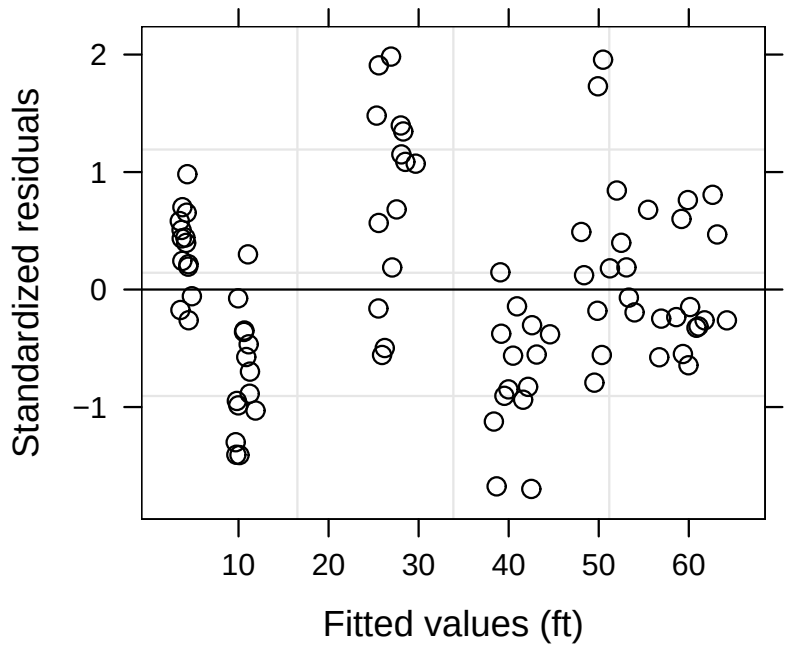
Fitting the Loblolly model

```
lmc <- lmeControl(niterEM=500,msMaxIter=100)
m0 <- lme(height ~ age + I(age^2) + I(age^3),Loblolly,
      random=list(Seed=~age+I(age^2)+I(age^3)),
      correlation=corAR1(-.5,form=~age|Seed),control=lmc)
plot(m0)
```



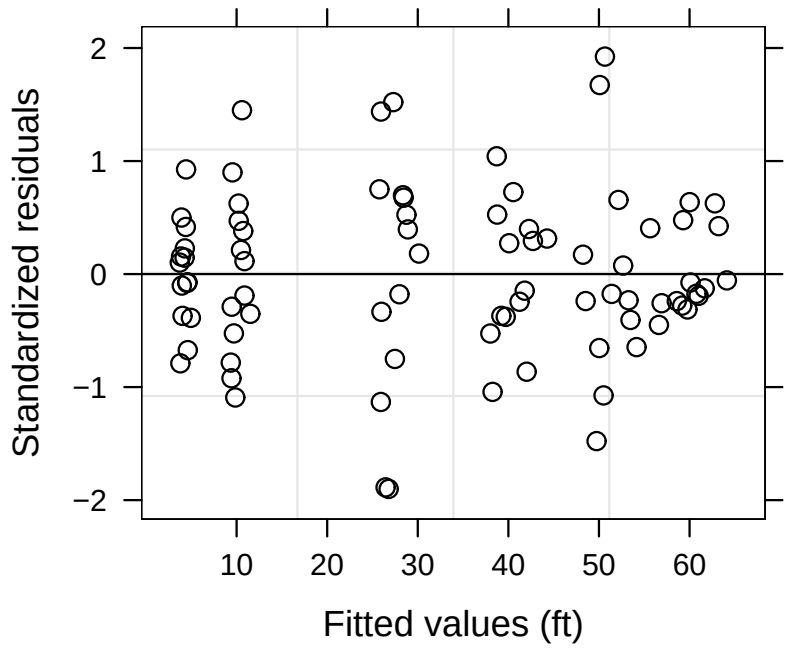
Try again...

```
m1<-lme(height ~ age + I(age^2) + I(age^3)+ I(age^4),
    Loblolly,random=list(Seed=~age+I(age^2)+I(age^3)),
    correlation=corAR1(-.5,form=~age|Seed),control=lmc)
plot(m1)
```



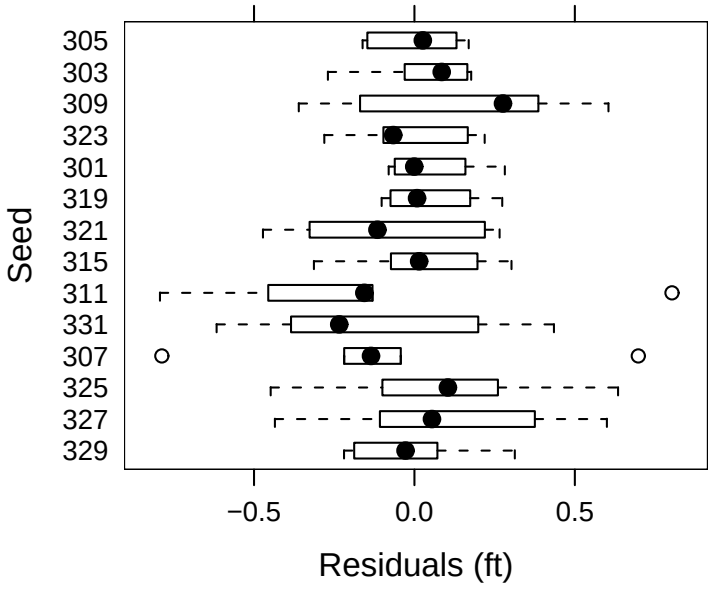
Improved Loblolly model

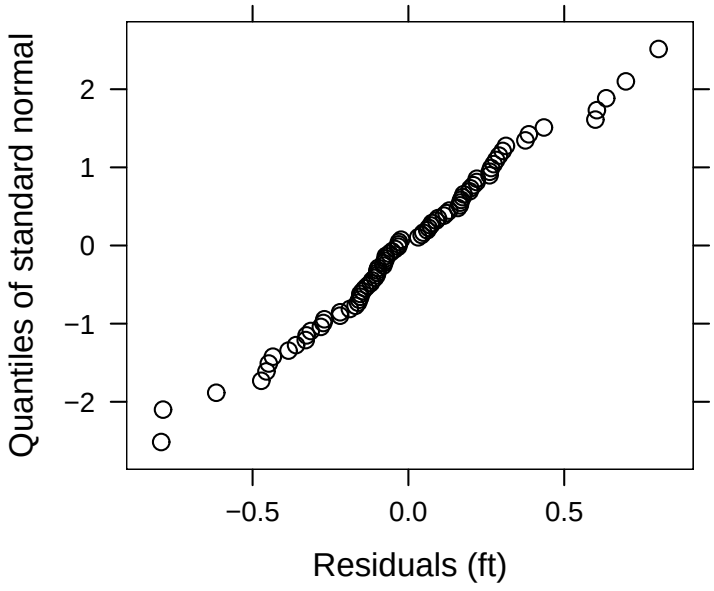
```
m2<-lme(height~age+I(age^2)+I(age^3)+I(age^4)+I(age^5),
    Loblolly,random=list(Seed=~age+I(age^2)+I(age^3)),
    correlation=corAR1(-.5,form=~age|Seed),control=lmc)
plot(m2)
```



More model checking

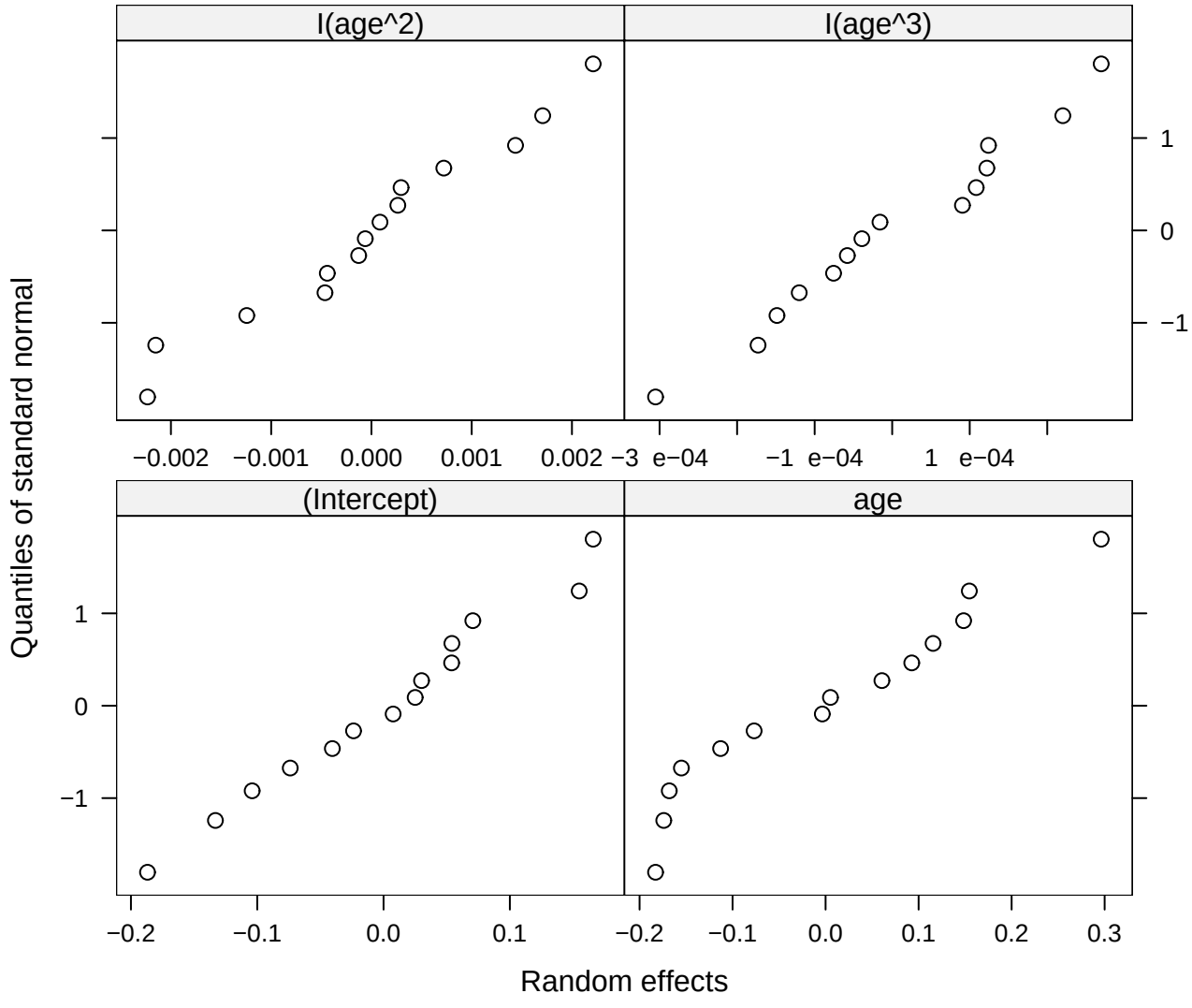
```
plot(m2,Seed~resid(.))# any pattern in resid vs. tree?
qqnorm(m2,~resid(.))  # are resids normal?
```






```
qqnorm(m2,~ranef(.))# now check random effects?
```

Checking random effects, **b**



Is autocorrelation needed?

```
> m3 <- lme(height ~ age + I(age^2) + I(age^3) +
+       I(age^4) + I(age^5),Loblolly,control=lmc,
+       random=list(Seed=~age+I(age^2)+I(age^3)))
> anova(m3,m2) ## GLRT
Model df   AIC   BIC logLik   Test L.Ratio p-value
m3  1 17 250.5 290.5 -108.2
m2  2 18 239.4 281.8 -101.7 1 vs 2 13.1041   3e-04
```

- AIC and hypothesis testing both strongly support retention of the autocorrelation model.
- **Question**: are the GLRT assumptions met for this test?

Simpler random effects structure?

```
> m4 <- lme(height ~ age + I(age^2) + I(age^3)+
+       I(age^4)+ I(age^5),Loblolly,control=lmc,
+       random=list(Seed=~age+I(age^2)),
+       correlation=corAR1(-.1,form=~age|Seed))
> anova(m4,m2)
Model df   AIC   BIC logLik   Test L.Ratio p-value
m4  1 14 253.8 286.8 -112.9
m2  2 18 239.4 281.8 -101.7 1 vs 2 22.4004   2e-04
```

- AIC and hypothesis testing both suggest that the cubic tree specific effect is needed in the model.
- **Question**: are the GLRT assumptions met in this case?

Simpler random effects correlation?

- We have assumed that ψ_θ can be any positive definite matrix. Let's try a simple diagonal structure, for ψ_θ...

```
> m5<-lme(height~age+I(age^2)+I(age^3)+
+         I(age^4)+I(age^5),Loblolly,random=
+         list(Seed=pdDiag(~age+I(age^2)+I(age^3))),
+      correlation=corAR1(-.5,form=~age|Seed),control=lmc)
> anova(m5,m2)
Model df      AIC    BIC  logLik Test L.Ratio p-value
m5    1 12 293.71 321.99 -134.85
m2    2 18 239.36 281.78 -101.68 1 vs 2 66.35  <.0001
```

- AIC and GLRT both imply that the diagonal covariance matrix is insufficient.
- **Question**: are the GLRT assumptions met here?

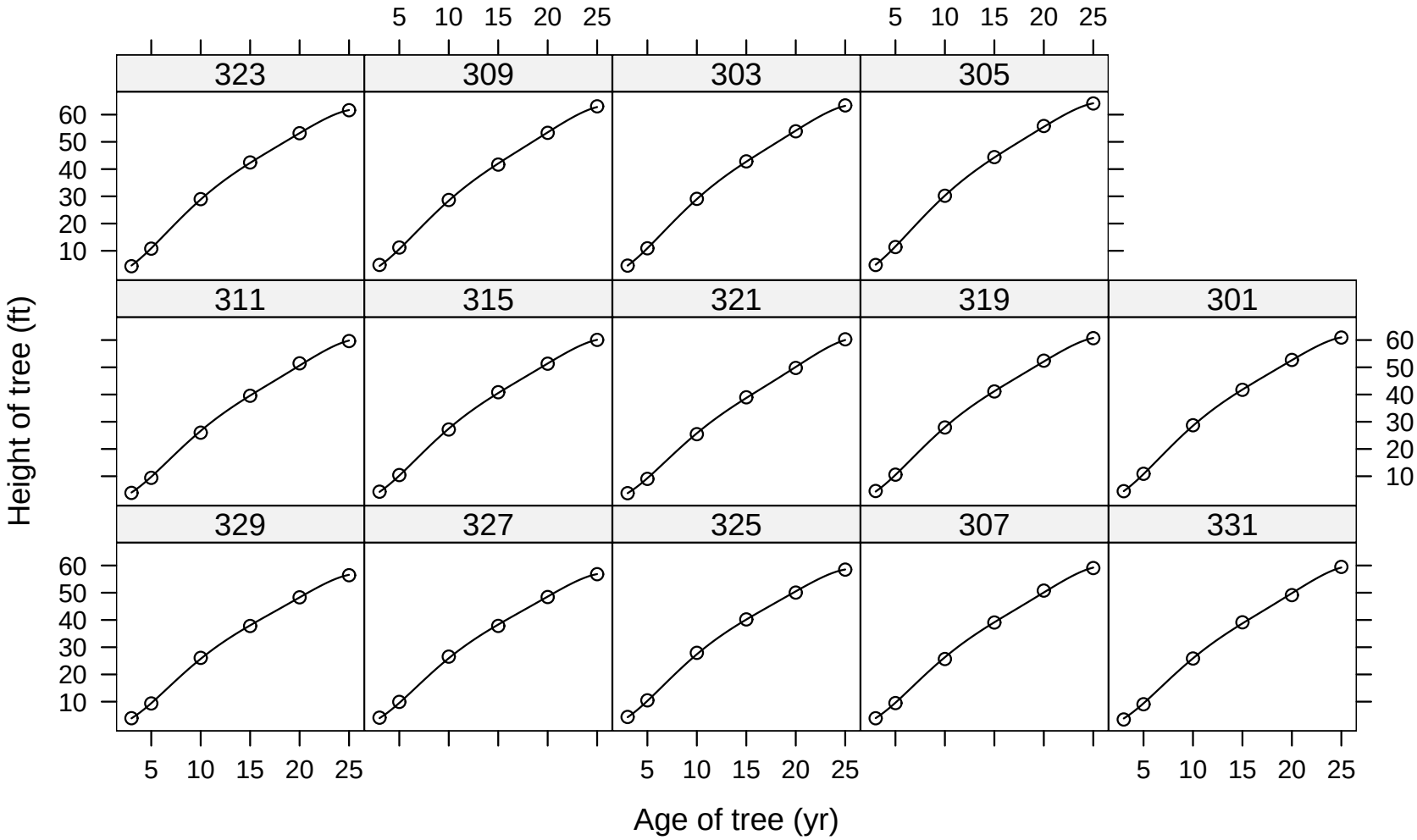
Follow up?

- We can test things about the fixed effects (conditional on $\hat{\boldsymbol{\theta}}$) as follows:

```
> anova(m2)
            numDF denDF  F-value p-value
(Intercept)     1    65   41.528  <.0001
age             1    65 9957.451  <.0001
I(age^2)        1    65  656.343  <.0001
I(age^3)        1    65   51.842  <.0001
I(age^4)        1    65  361.458  <.0001
I(age^5)        1    65   69.200  <.0001
```

- `intervals` and a `summary` of the selected model, `m2`, would also be examined. Some quite advanced plotting facilities are available in `nlme`, for example: `plot(augPred(m2))`

Loblolly fit



lme4:lmer

- Use of `lmer` is easier than `lme`.
- One formula specifies the model. Terms of the form `(x|g)` indicate random effects: `g` is a grouping factor; `x` determines the random effect model matrix for each level of `g`.
- Here's the `Rail` example again

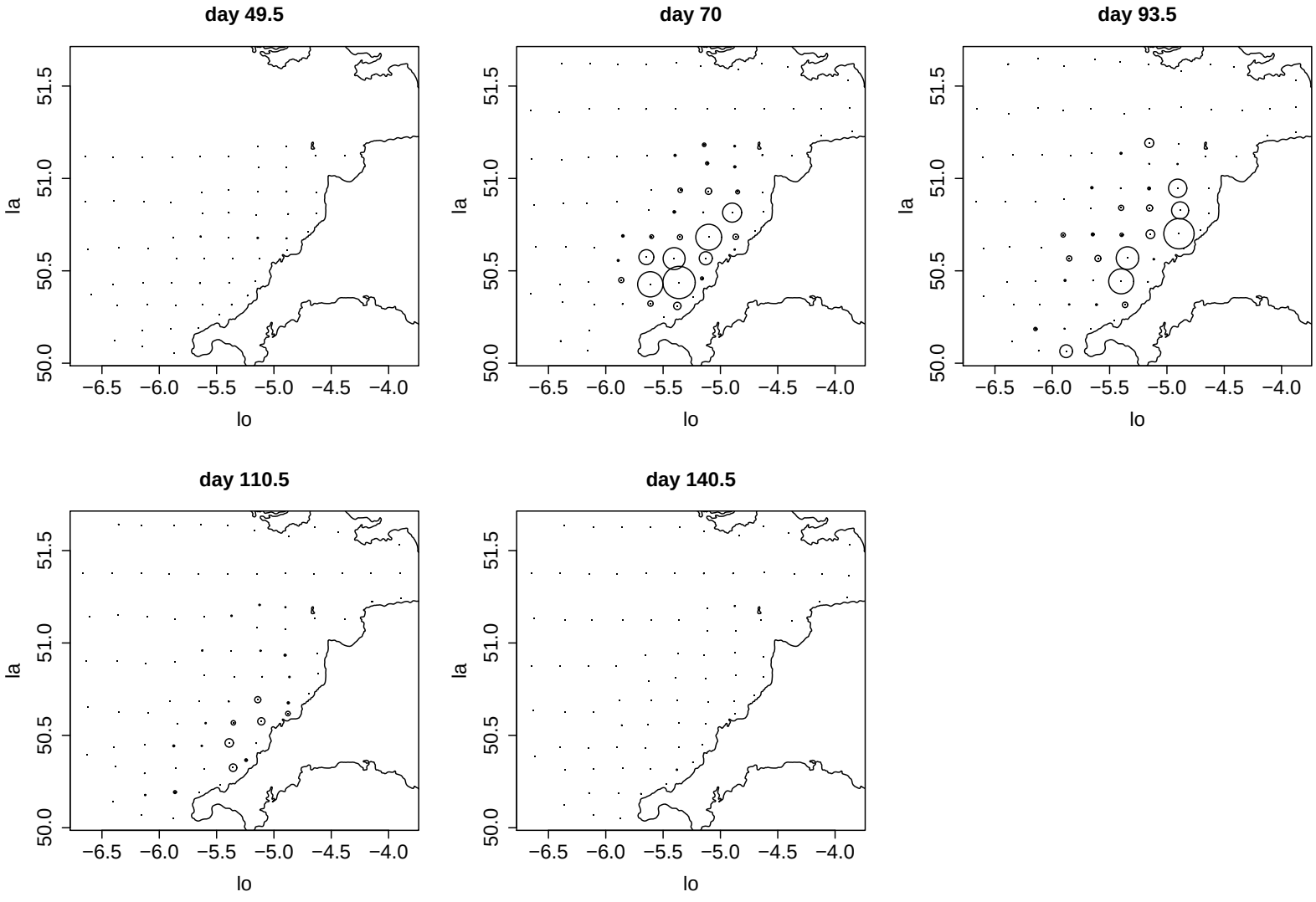
```
> lmer(travel~(1|Rail),data=Rail)
Linear mixed model fit by REML
Formula: travel ~ (1 | Rail)
   Data: Rail
   AIC   BIC logLik deviance REMLdev
 128.2 130.8 -61.09    128.6   122.2
Random effects:
 Groups   Name        Variance Std.Dev.
 Rail     (Intercept) 615.311  24.8055
 Residual              16.167   4.0208
Number of obs: 18, groups: Rail, 6

Fixed effects:
            Estimate Std. Error t value
(Intercept)    66.50      10.17   6.539
```

glmmPQL

- PQL estimation of GLMMs is available in the `glmmPQL` routine in the `MASS` library.
- Use of `glmmPQL` is very similar to use of `lme`, except that a `family` argument is now needed.
- `glmmPQL` operates by iteratively calling `lme`, and returns the final fitted model object returned by `lme`, at convergence.
- The PQL iterations are not guaranteed to converge, and the routine may fail.

Bristol Channel Sole Eggs



Sole Egg Model

- Observe egg densities, y_{ij} in 4 distinct age classes at each of several sampling stations at 5 times of year.
- Want to know rate of spawning, and total number of eggs spawned.
- y_{ij} is observed density for stage j at sampling station i, then

$$\mathbb{E}(y_{ij}|b_i) = \Delta_{ij} R(\mathrm{lo}_i, \mathrm{la}_i, t_i) e^{-\delta(t_i) a_{ij}} b_i.$$

- R is spawning rate at $(\mathrm{lo}_i, \mathrm{la}_i, t_i)$; δ is mortality rate; Δ_{ij} is the j^{th} age class duration, while a_{ij} is the mid-point age of the class. The $\log(b_i)$ are i.i.d. $N(0, \sigma_b^2)$ random effects for sampling station.

Linearized Sole Egg Model

- The model is linearized by a log link.

$$\log\{\mathbb{E}(y_{ij}|b_i)\} = \log(\Delta_{ij}) + r(\mathrm{lo}_i, \mathrm{la}_i, t_i) - \delta(t_i)a_{ij} + \log(b_i)$$

  where $r = \log R$.
- r can modelled using a cubic in lo_i, la_i and t_i, and $-\delta$ modelled using a quadratic in time.
- Assume $\mathrm{var}(y_{ij}|b_i) \propto \mathbb{E}(y_{ij}|b_i)$
- Then the model has the structure of a GLMM, and can be estimated by `glmmPQL`.

Sole data preparation

The data are available in data frame `sole`, in the `gamair` package. Some manipulation is needed first:

```
sole$off <- log(sole$a.1-sole$a.0) # offset term
sole$a<-(sole$a.1+sole$a.0)/2      # mean stage age
solr<-sole                  # make copy for rescaling
## rescale terms for better numerical behaviour...
solr$t<-solr$t-mean(sole$t)
solr$t<-solr$t/var(sole$t)^0.5
solr$la<-solr$la-mean(sole$la)
solr$lo<-solr$lo-mean(sole$lo)
## make a lable for sampling station...
solr$station <-
     factor(with(solr,paste(-la,-lo,-t,sep="")))
```

Sole model fitting

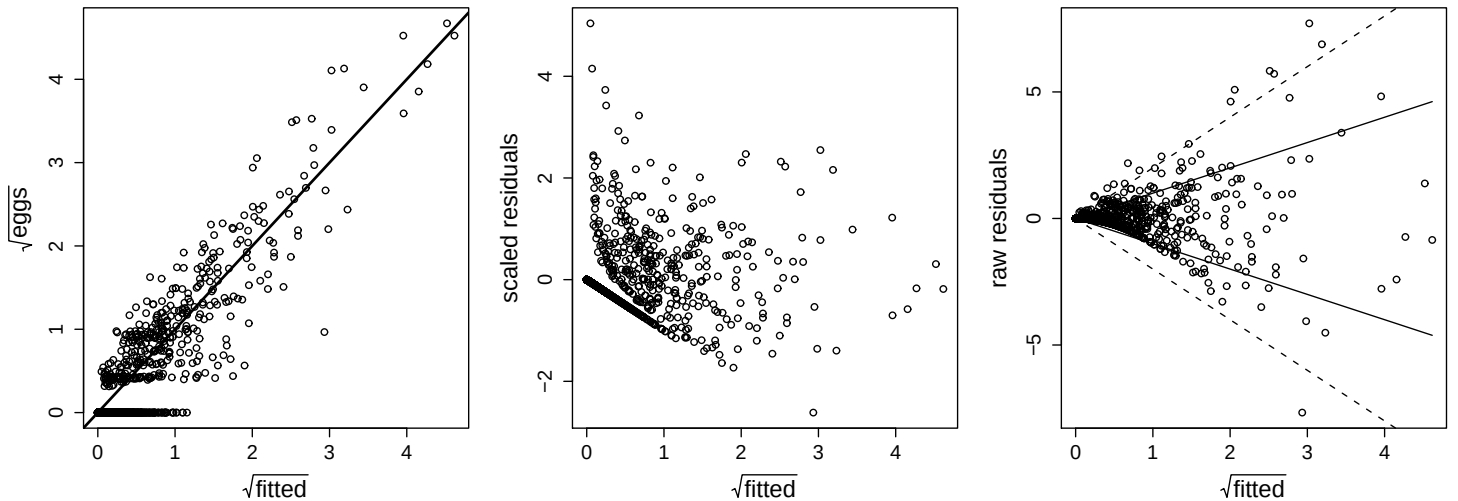
Here is the command to fit the model, in full horrible detail . . .

```
b<-glmmPQL(eggs~offset(off)+lo+la+t+I(lo*la)+I(lo^2)+
           I(la^2)+I(t^2)+I(lo*t)+I(la*t)+I(lo^3)+
           I(la^3)+I(t^3)+I(lo*la*t)+I(lo^2*la)+
           I(lo*la^2)+I(lo^2*t)+I(la^2*t)+I(la*t^2)+
           I(lo*t^2) # end log spawn
           + a +I(a*t)+I(t^2*a), # death term
           random=list(station=~1),data=solr,
           family=quasi(link=log,variance="mu"))
```

Now we need residual plots. . .

Sole residuals

```
fv <- exp(fitted(b4)+solr$off) # note need to add offset
resid <- solr$egg-fv           # raw residuals
plot(fv^.5,solr$eggs^.5);abline(0,1,lwd=2)
plot(fv^.5,resid/fv^.5)
plot(fv^.5,resid)
fl<-sort(fv^.5)
## add 1 s.d. and 2 s.d. reference lines
lines(fl,fl);lines(fl,-fl);lines(fl,2*fl,lty=2);lines(fl,-2*fl,lty=2)
```

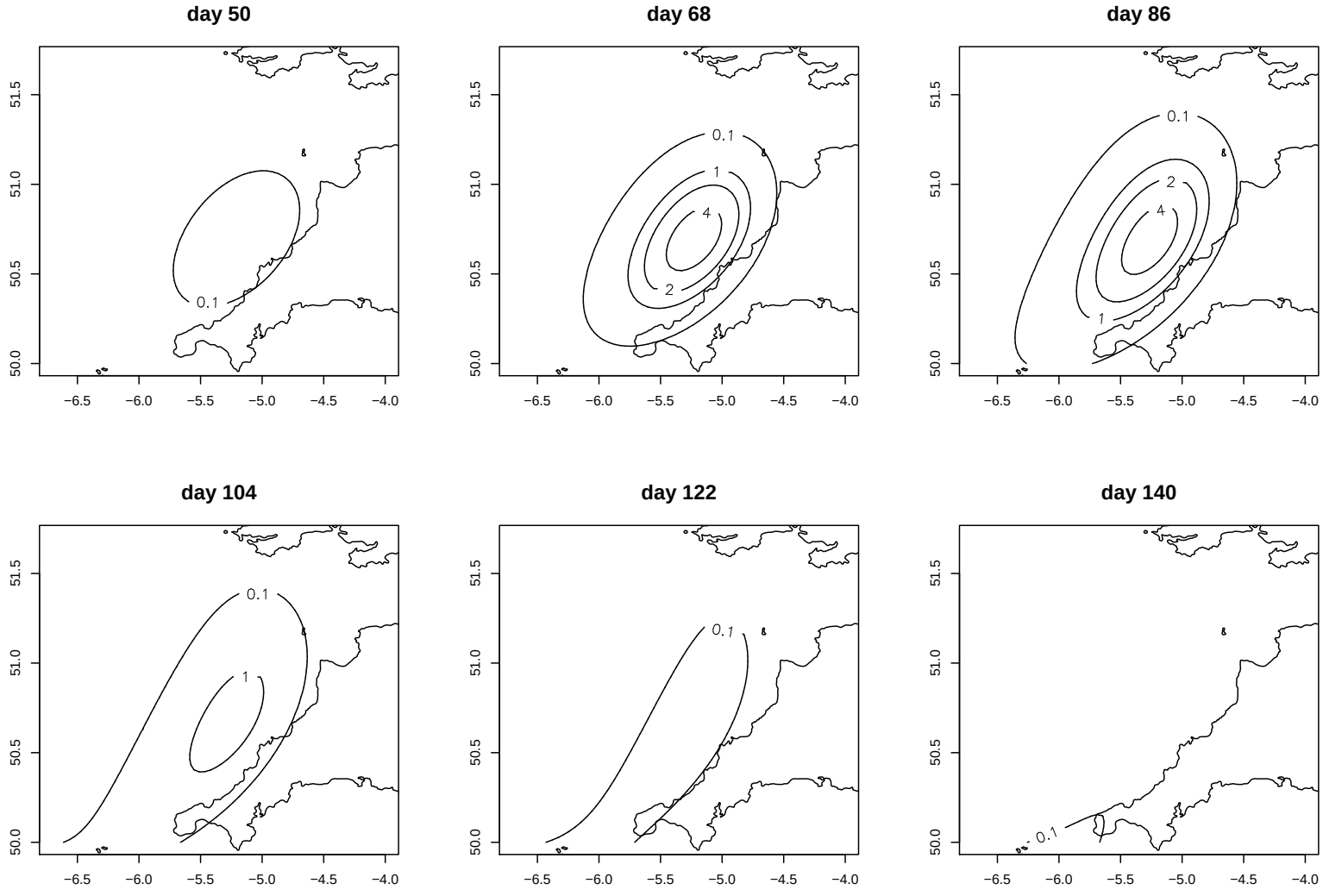


Model selection

- We can base model selection on backward selection using `summary(b)`.
- Can also use `anova(b,type="marginal")`, which is more useful if the model has factor variables, as it gives p-values for whole term removal.
- Actually only 4 terms get dropped from `b` in this way.
- Can also look at CI for σ_b^2

```
> intervals(b4,which="var-cov")
  Level: station
                     lower      est.    upper
sd((Intercept)) 0.8398715 0.9599066 1.097097
```

Spawning rate predictions



lme4: glmer

- glmer fits GLMMs by Laplace approximation.
- It is just like lmer, except that the exponential family and link function are supplied by a glm style family argument.
- The fit of the sole model would look like this

```
br <- glmer(eggs~offset(off)+lo+la+t+I(lo*la)+I(lo^2)+
           I(la^2)+I(t^2)+I(lo*t)+I(la*t)+I(lo^3)+
           I(la^3)+I(t^3)+I(lo*la*t)+I(lo^2*la)+
           I(lo*la^2)+I(lo^2*t)+I(la^2*t)+I(la*t^2)+
           I(lo*t^2) # end log spawn
           + a +I(a*t)+I(t^2*a) + # death term
           (1|station), ## station random effect
           data=solr,
           quasi(link=log,variance="mu"))
```

- . . . but it is not clear that quasi really works with Laplace approximate fitting!

Moving on

- Notice how cumbersome the specification of the Sole egg model was.
- Time for GAMs!