Advanced dataanalysis and statistical modelling, Week 11

General mixed effects models - III

Jan Kloppenborg Møller

April 22, 2018

Outline

Oversigt

General mixed effects models

The general mixed effects model can be represented by its likelihood function:

$$L_M(\boldsymbol{\theta}; \boldsymbol{y}) = \int_{\mathbb{R}^q} L(\boldsymbol{\theta}; \boldsymbol{u}, \boldsymbol{y}) d\boldsymbol{u}$$

- $\boldsymbol{y}$ is the observed random variables
- $\boldsymbol{u}$ is the q unobserved random variables
- $\boldsymbol{\theta}$ is the model parameters to be estimated

The likelihood function L is the joint likelihood of both the observed and the unobserved random variables.

The likelihood function for estimating $\boldsymbol{\theta}$ is the marginal likelihood L_M obtained by integrating out the unobserved random variables.

Hierarchical models

As for the Gaussian linear mixed models it is useful to formulate the model as a *hierarchical model* containing a *first stage model*

$$f_{Y|u}(\boldsymbol{y};\boldsymbol{u},\boldsymbol{\beta})$$

which is a model for the data given the random effects, and a *second stage model*

$$f_U(\boldsymbol{u};\boldsymbol{\Psi})$$

which is a model for the random effects. The total set of parameters is $\boldsymbol{\theta}=(\boldsymbol{\beta},\boldsymbol{\Psi})$. Hence the joint likelihood is given as

$$L(\boldsymbol{\beta},\boldsymbol{\Psi};\boldsymbol{u},\boldsymbol{y})=f_{Y|u}(\boldsymbol{y};\boldsymbol{u},\boldsymbol{\beta})f_U(\boldsymbol{u};\boldsymbol{\Psi})$$

Hierarchical models

To obtain the likelihood for the model parameters $\boldsymbol{\theta}$ the unobserved random effects are again integrated out.

The likelihood function for estimating $\boldsymbol{\theta}$ is as before the marginal likelihood

$$L_M(\boldsymbol{\theta}; \boldsymbol{y}) = \int_{\mathbb{R}^q} L(\boldsymbol{\theta}; \boldsymbol{u}, \boldsymbol{y}) d\boldsymbol{u}$$

where q is the number of random effects, and $\boldsymbol{\theta}$ contains all parameters to be estimated.

Oversigt

The Laplace approximation

- We need to calculate the difficult integral

$$L_M(\boldsymbol{\theta}, \boldsymbol{y}) = \int_{\mathbb{R}^q} L(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{y}) d\boldsymbol{u}$$

- So we set up an approximation of $\ell(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{y}) = \log L(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{y})$

$$\ell(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{y}) \approx \ell(\boldsymbol{\theta}, \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}, \boldsymbol{y}) - \frac{1}{2}(\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}})^T \left(-\ell''_{uu}(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{y})|_{\boldsymbol{u}=\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}}\right) (\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}})$$

- Which (for given $\boldsymbol{\theta}$) is the 2. order Taylor approximation around:

$$\hat{\boldsymbol{u}}_{\boldsymbol{\theta}} = \underset{\boldsymbol{u}}{\text{argmax}}\ L(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{y})$$

- With this approximation we can calculate:

$$\begin{aligned} L_M(\boldsymbol{\theta}, \boldsymbol{y}) &= \int_{\mathbb{R}^q} L(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{y}) d\boldsymbol{u} \\ &\approx \int_{\mathbb{R}^q} e^{\ell(\boldsymbol{\theta}, \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}, \boldsymbol{y}) - \frac{1}{2}(\boldsymbol{u}-\hat{\boldsymbol{u}}_{\boldsymbol{\theta}})^T \left(-\ell''_{uu}(\boldsymbol{\theta},\boldsymbol{u},\boldsymbol{y})|_{\boldsymbol{u}=\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}}\right)(\boldsymbol{u}-\hat{\boldsymbol{u}}_{\boldsymbol{\theta}})} d\boldsymbol{u} \\ &= L(\boldsymbol{\theta}, \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}, \boldsymbol{y}) \int_{\mathbb{R}^q} e^{-\frac{1}{2}(\boldsymbol{u}-\hat{\boldsymbol{u}}_{\boldsymbol{\theta}})^t \left(-\ell''_{uu}(\boldsymbol{\theta},\boldsymbol{u},\boldsymbol{y})|_{\boldsymbol{u}=\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}}\right)(\boldsymbol{u}-\hat{\boldsymbol{u}}_{\boldsymbol{\theta}})} d\boldsymbol{u} \\ &= L(\boldsymbol{\theta}, \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}, \boldsymbol{y}) \sqrt{\frac{(2\pi)^q}{|\left(-\ell''_{uu}(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{y})|_{\boldsymbol{u}=\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}}\right)|}} \end{aligned}$$

- In the last step we remember the normalizing constant for a multivariate normal, and that $|A^{-1}| = 1/|A|$.
- Taking the logarithm we get:

$$\ell_M(\boldsymbol{\theta}, \boldsymbol{y}) \approx \ell(\boldsymbol{\theta}, \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}, \boldsymbol{y}) - \frac{1}{2}\log(|\left(-\ell''_{uu}(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{y})|_{\boldsymbol{u}=\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}}\right)|) + \frac{q}{2}\log(2\pi)$$

Laplace approximation work flow

0. Initialize $\boldsymbol{\theta}$ to some arbitrary value $\boldsymbol{\theta}_0$
1. With current value for $\boldsymbol{\theta}$ optimize joint likelihood w.r.t. $\boldsymbol{u}$ to get $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}$ and corresponding Hessian $H(\hat{\boldsymbol{u}}_{\boldsymbol{\theta}})$.
2. Use $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}$ and $H(\hat{\boldsymbol{u}}_{\boldsymbol{\theta}})$ to approximate $\ell_M(\boldsymbol{\theta})$
3. Compute value and gradient of $\ell_M(\boldsymbol{\theta})$
4. If the gradient is "$> \epsilon$" set $\boldsymbol{\theta}$ to a different value and go to 1.

Notice the huge number of — possibly high dimensional — optimizations that are required.

Example - beetles

Look at the data and implement a “naive” Laplace approximation.

Grouping structures and nested effects

- For nonlinear mixed models where no closed form solution to the likelihood function is available it is necessary to invoke some form of numerical approximation to be able to estimate the model parameters.
- The complexity of this problem is mainly dependent on the dimensionality of the integration problem which in turn is dependent on the dimension of $\boldsymbol{U}$ and in particular the *grouping structure* in the data for the random effects.
- For problems with only one level of grouping the marginal likelihood can be simplified as

$$L_M(\boldsymbol{\beta}, \boldsymbol{\Psi}; \boldsymbol{y}) = \prod_{i=1}^{M} \int_{\mathbb{R}^{q_i}} f_{Y|u_i}(\boldsymbol{y}; \boldsymbol{u}_i, \boldsymbol{\beta}) f_{U_i}(\boldsymbol{u_i}; \boldsymbol{\Psi})\, d\boldsymbol{u}_i$$

where q_i is the number of random effects for group i and M is the number of groups.

Example - beetles

Use the structure of the random effects to speed up likelihood calculation.

Including derivatives wrt. the random effect

Significant speed up can be obtained if the first and/or the second derivative of the objective function is known, e.g. `nlminb` can take the gradient and the hessian as input functions.

Example - beetles

Find the derivative and the Hessian of the random effects likelihood and use this to speed up likelihood calculations further.

glmmTMB

glmmTMB (Generalized Linear Mixed Models with Template Model Builder) is a wrapper for TMB

- Build on the class of exponential family
- The family needs to be specified
- The link need to be specified
- The second stages model is assumed Gaussian

i.e.

$$\begin{aligned}\boldsymbol{\eta} =& \boldsymbol{X}\theta + \boldsymbol{U} \\ \boldsymbol{U} \sim& N(\mathbf{0}, \boldsymbol{\Psi}) \\ E[\boldsymbol{Y}|\boldsymbol{U}] =& g^{-1}(\boldsymbol{\eta})\end{aligned}$$

and the conditional distribution of $\boldsymbol{Y}$ belonging the the exponential family (see documentation for included families).

glmmTMB syntax

The syntax of `glmmTMB` follow the syntax in `lme4`, e.g.

```
(m1 <- glmmTMB(count~ mined + (1|site), family=poisson,
               data=Salamanders))
```

also different correlation structures (part of $\boldsymbol{\Psi}$) can be implemented in `glmmTMB`, e.g.

```
(m1 <- glmmTMB(count~ x + ar1(as.factor(time)-1|site),
               family=poisson, data=data))
```

Again consult the documentation for details on the type of possible correlation structures.

Importance sampling

- Importance sampling is a re-weighting technique for approximating integrals w.r.t. a density f by simulation in cases where it is not feasible to simulate from the distribution with density f.
- Instead it uses samples from a different distribution with density g, where the support of g includes the support of f.
- For general mixed effects models it is possible to simulate from the distribution with density proportional to the second order Taylor approximation

$$\widetilde{L}(\boldsymbol{\theta}, \hat{\boldsymbol{u}}_\theta, \boldsymbol{Y}) = e^{\ell(\boldsymbol{\theta}, \hat{\boldsymbol{u}}_\theta, \boldsymbol{Y}) - \frac{1}{2}(\boldsymbol{u}-\hat{\boldsymbol{u}}_\theta)^T(-\ell''_{uu}(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{Y})|_{\boldsymbol{u}=\hat{\boldsymbol{u}}_\theta})(\boldsymbol{u}-\hat{\boldsymbol{u}}_\theta)}$$

which, apart from a normalization constant, it is the density $\phi_{\hat{u}_\theta, \hat{V}_\theta}(\boldsymbol{u})$ of a multivariate normal with mean $\hat{\boldsymbol{u}}_\theta$ and covariance

$$\hat{\boldsymbol{V}}_\theta = \boldsymbol{H}^{-1}(\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}) = (-\ell''_{uu}(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{Y})|_{\boldsymbol{u}=\hat{\boldsymbol{u}}_\theta})^{-1}.$$

Importance sampling

The integral to be approximated can be rewritten as:

$$L_M(\boldsymbol{\theta}, \boldsymbol{Y}) = \int L(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{Y}) d\boldsymbol{u} = \int \frac{L(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{Y})}{\phi_{\hat{u}_\theta, \hat{V}_\theta}(u)} \phi_{\hat{u}_\theta, \hat{V}_\theta}(\boldsymbol{u}) d\boldsymbol{u}.$$

So if $\boldsymbol{u}^{(i)}$, $i = 1, \dots, N$ is simulated from the multivariate normal distribution with mean $\hat{\boldsymbol{u}}_\theta$ and covariance $\hat{\boldsymbol{V}}_\theta$, then the integral can be approximated by the mean of the importance weights

$$L_M(\boldsymbol{\theta}, \boldsymbol{Y}) = \frac{1}{N} \sum \frac{L(\boldsymbol{\theta}, \boldsymbol{u}^{(i)}, \boldsymbol{Y})}{\phi_{\hat{u}_\theta, \hat{V}_\theta}(\boldsymbol{u}^{(i)})}$$

Example - beetles

Check the results by importance sampling, also fit the model with direct use of numerical integration.

Two-level hierarchical model

- For the two-level or hierarchical model it is readily seen that the joint log-likelihood is

$$\ell(\boldsymbol{\theta}, \boldsymbol{u}, \boldsymbol{y}) = \ell(\boldsymbol{\beta}, \boldsymbol{\Psi}, \boldsymbol{u}, \boldsymbol{y}) = \log f_{Y|u}(\boldsymbol{y}; \boldsymbol{u}, \boldsymbol{\beta}) + \log f_U(\boldsymbol{u}; \boldsymbol{\Psi})$$

  which implies that the Laplace approximation becomes

$$\ell_{M,LA}(\boldsymbol{\theta}, \boldsymbol{y}) = \log f_{Y|u}(\boldsymbol{y}; \tilde{\boldsymbol{u}}, \boldsymbol{\beta}) + \log f_U(\tilde{\boldsymbol{u}}; \boldsymbol{\Psi}) - \frac{1}{2} \log \left| \frac{\boldsymbol{H}(\tilde{\boldsymbol{u}})}{2\pi} \right|$$

- It is clear that as long as a likelihood function of the random effects and model parameters can be defined it is possible to use the Laplace likelihood for estimation in a mixed model framework.

Gaussian second stage model

- Let us assume that the second stage model is zero mean Gaussian, i.e.

$$\boldsymbol{u} \sim N(\mathbf{0}, \boldsymbol{\Psi})$$

which means that the random effect distribution is completely described by its covariance matrix $\boldsymbol{\Psi}$.

- In this case the Laplace likelihood in becomes

$$\begin{aligned}\ell_{M,LA}(\boldsymbol{\theta}, \boldsymbol{y}) = \log f_{Y|u}(\boldsymbol{y}; \tilde{\boldsymbol{u}}, \boldsymbol{\beta}) - \frac{1}{2} \log |\boldsymbol{\Psi}| \\ - \frac{1}{2} \tilde{\boldsymbol{u}}^T \boldsymbol{\Psi}^{-1} \tilde{\boldsymbol{u}} - \frac{1}{2} \log |\boldsymbol{H}(\tilde{\boldsymbol{u}})|\end{aligned}$$

where it is seen that we still have no assumptions on the first stage model $f_{Y|u}(\boldsymbol{y}; \boldsymbol{u}, \boldsymbol{\beta})$.

Oversigt

Automatic Differentiation

Automatic differentiation is based on

- Derivatives of simple functions (like `'+'`, `'-'`, `'*'`, `'/'`, `'exp'`, `'sin'`, ...)
- The chain rule ($(f \circ g)(x) = f'(g(x))g'(x)$)

Applying the chain rule to complicated functions is intractable to do with pen and paper, but well suited for computer programs.
Tools

- TMB (`http://tmb-project.org`)
- `deriv` and `D` in R (Partly automatic differentiation)

Automatic differentiation

- Automatic differentiation is a technique where the chain rule is used by the computer program itself.
- When the program evaluates the log-likelihood it keeps track of all the operations used along the way, and then runs the program backwards (reverse mode automatic differentiation) and uses the chain rule to update the derivatives one simple operation at a time.
- Automatic differentiation is accurate, and the computational cost of evaluating the gradient is surprisingly low.

Automatic differentiation

Theorem

The computational cost of evaluating the gradient of the log-likelihood $\nabla\ell$ with reverse mode automatic differentiation is less than four times the computational cost of evaluating the log-likelihood function ℓ itself. This holds no matter how many parameters the model contain.

- It is surprising that computational cost does not depend on how many parameters the model contain.
- There is however a practical concern. The computational cost mentioned above is measured in the number of operations, but reverse mode automatic differentiation requires all the intermediate variables in the calculation of the negative log-likelihood to be stored in the computer's memory, so if the calculation is lengthy, for instance consisting of a long iterative procedure, then the memory requirements can be enormous.

Automatic differentiation combined with the Laplace approximation

- Finding the gradient of the Laplace approximation of the marginal log-likelihood is challenging, because the approximation itself includes the result of a function minimization, and not just a straightforward sequence of simple operations.
- It is however possible, but requires up to third order derivatives to be computed internally by clever successive application of automatic differentiation.

Automatic Differentiation in R

- `deriv` Takes an expression as input and return an expression or a function
- `D` Takes an expression as input and return an expression and is well suited for recursive differentiation

Example of use

```
> Ex<-expression(log((sin(phi*x+a))^2+x^2))
> D(Ex,"x")
(2 * (cos(phi * x + a) * phi * (sin(phi * x + a))) + 2 * x)/
    ((sin(phi * x + a))^2 + x^2)
```

Automatic Differentiation in R

Example of use

```
> Ex<-expression(log((sin(phi*x+a))^2+x^2))
> deriv(Ex,"x",func=function(x,a,phi,sigma){})
function (x, a, phi, sigma)
{
    .expr2 <- phi * x + a
    .expr3 <- sin(.expr2)
    .expr6 <- .expr3^2 + x^2
    .value <- log(.expr6)
    .grad <- array(0, c(length(.value), 1L), list(NULL, c("x")))
    .grad[, "x"] <- (2 * (cos(.expr2) * phi * .expr3) + 2 * x)/
                    .expr6
    attr(.value, "gradient") <- .grad
    .value
}
```

TMB Model Builder

- TMB Model Builder is a programming language that builds on C++.
- It includes helper functions for reading in data, defining model parameters, and implementing and optimizing the negative log-likelihood function.
- One central feature is automatic differentiation (AD), which is implemented in such a way that the user rarely has to think about it at all.
- TMB use a number of different software packages to perform algebraric differentiation and efficient matrix manipulation

From TMB-wiki page

TMB (Template Model Builder) is an R package for fitting statistical latent variable models to data. It is strongly inspired by ADMB. Unlike most other R packages the model is formulated in C++. This provides great flexibility, but requires some familiarity with the C/C++ programming language.

- TMB can calculate first and second order derivatives of the likelihood function by AD, or any objective function written in C++.
- The objective function (and its derivatives) can be called from R. Hence, parameter estimation via e.g. nlminb() is easy.
- The user can specify that the Laplace approximation should be applied to any subset of the function arguments.
- Yields marginal likelihood in latent variable model.
- Standard deviations of any parameter, or derived parameter, obtained by the 'delta method'.
- Pre and post-processing of data done in R.
- TMB is based on state-of-the art software: CppAD, Eigen, ...

Objective function in C++

Every model require a C++ file that defines the likelihood, e.g. (`tutorial.cpp` from TMB-homepage)

```
#include <TMB.hpp>                  // Links in the TMB libraries

template<class Type>
Type objective_function<Type>::operator() ()
{
  DATA_VECTOR(x);                   // Data vector transmitted from R
  PARAMETER(mu);                    // Parameter value transmitted
  PARAMETER(sigma);                 // from R

  Type f;                           // Declare the "objective function"
                                    // (neg. log. likelihood)
  f = -sum(dnorm(x,mu,sigma,true));     // Use R-style call
                                        // to normal density
  return f;
}}
```

Call from R

The optimization is done from R using `nlminb()`, e.g.

```
## simulate data
n = 100
x = rnorm(n=n,mean=0,sd=1)

library(TMB)                          ## Load library
compile("tutorial.cpp")              # Compile the C++ file
dyn.load(dynlib("tutorial"))         # Dynamically link the C++ code

## Function and derivative
f = MakeADFun(data=list(x=x),parameters=list(mu=0,sigma=1))
f$fn(list(mu=1,sigma=1))
f$gr(list(mu=1,sigma=1))                # First order derivatives (w.r
f$he(list(mu=1,sigma=1))                # Second order derivatives, i.

## Fit model
fit = nlminb(f$par,f$fn,f$gr,lower=c(-10,0.0),
             upper=c(10.0,10.0))
print(fit) ## print result
```

Example - beetles

Fit the model using TMB.

Beetles exposed to ethylene oxide

Ten groups beetles were exposed to different concentrations of ethylene oxide and it was recorded how many died.

```
> conc <- c(24.8, 24.6, 23, 21, 20.6, 18.2, 16.8, 15.8, 14.7, 10.8)
> n <- c(30, 30, 31, 30, 26, 27, 31, 30, 31, 24)
> y <- c(23, 30, 29, 22, 23, 7, 12, 17, 10, 0)
```

The natural model is a binomial, and we wish to setup a logit-linear model as a function of the logarithm of the concentrations

$$y_i \sim \text{Bin}(n_i, p_i) \text{ , where}$$
$$\text{logit}(p_i) = \mu + \beta \log(\text{conc}_i)$$

```
> resp <- cbind(y, n - y)
> fit <- glm(resp ~ I(log(conc)), family = binomial())
```

beetle.cpp

```
   .
   .
   .
 int nobs = y.size();
 Type mean_ran = Type(0);
 Type p = Type(0);

 Type f = 0;  // Declare the "objective function" (neg. log. li

 for(int i =0; i < nobs; i++){
   f -= dnorm(B[i], mean_ran , sigma_b, true);
   p = 1/(1+exp(-beta[0]-beta[1]*logc[i]-B[i]));
   f -= dbinom(y[i],n[i],p, true);
 }

 return f;
```

Example: Orange tree

- First we wish to run the model:

$$\mathsf{cir}_i = \frac{\beta_1 + U(\mathsf{tree}_i)}{1 + \exp(-(\mathsf{age}_i - \beta_2)/\beta_3)} + \epsilon_i$$

- where $\epsilon_i \sim N(0, \sigma^2)$ and $U(\mathsf{tree}_i) \sim N(0, \sigma_U^2)$ independent.

orange.cpp (part 1.)

```
#include <TMB.hpp>          // Links in the TMB libraries

template<class Type>
Type objective_function<Type>::operator() ()
{
  DATA_VECTOR(y);              // Data vector transmitted from R
  DATA_VECTOR(t);              // Data vector transmitted from R
  DATA_FACTOR(tree);           // Data vector transmitted from R

  PARAMETER_VECTOR(u);         // Random effects
  // Parameters
  PARAMETER_VECTOR(beta);  // Parameter value transmitted from R
  PARAMETER(sigma_u);       // Parameter value transmitted from R
  PARAMETER(sigma);         // Parameter value transmitted from R

  int nobs = y.size();
  int ntrees = u.size();
  Type mean_ran = Type(0);
  int j;
```

orange.cpp (part 2.)

```
  Type f = 0;  // Declare the "objective function" (neg. log. li

  for(int j=0; j < ntrees; j++){
    f -= dnorm(u[j], mean_ran , sigma_u, true);
  }

  for(int i =0; i < nobs; i++){
    j = tree[i];
    f -= dnorm(y[i],(beta[0]+u[j])/(1+exp(-(t[i]-beta[1])/
                    beta[2])), sigma, true);
  }

  return f;
}
```

Call from R (part 1)

```
compile("orange.cpp")
dyn.load(dynlib("orange"))

parameters <- list(u = rep(1, nlevels(Orange$Tree)),
                   beta = c(mean(data$y),mean(data$t),
                            diff(range(data$t))),
                   sigma_u= 1,
                   sigma = 1
                   )

obj <- MakeADFun(data = data,
                 parameters = parameters,
                 random = "u",
                 DLL = "orange"
                 )
```

Call from R (part 2)

```
## Fit model
opt <- nlminb(obj$par, obj$fn, obj$gr,lower=0.01)
opt$par
opt$objective

## report on result
sdreport(obj)

rap <- sdreport(obj,getJointPrecision = TRUE)
summary(rap,"random")
rap$par.random
rap$diag.cov.random
```

Other models..

- (5.111) (plus random component for sampling occasion (j))

$$y_{ij} = \frac{\beta_1 + u_{1i} + u_{2j}}{1 + \exp[-(t_j - \beta_2)/\beta_3]} + \epsilon_{ij} \tag{1}$$

- (5.112) (seasonal variation):

$$y_{ij} = \frac{\beta_1 + u_{1i}}{1 + \exp[-((t_j - \beta_2)/\beta_3 + s_j\beta_4)]} + \epsilon_{ij} \tag{2}$$

- (5.113) (First order AR)

$$\mathrm{cov}(\epsilon_{ij}, \epsilon_{ij'}) = \sigma^2 \exp(-\phi|t_{j'} - t_j|/(365/2)), \quad \phi \geq 0 \tag{3}$$

ie. the full covariance matrix is block diagonal.

orange2.cpp

Including one extra random component

```
  .
  .
  .
for(int j=0; j < ntrees; j++){
  f -= dnorm(u1[j], mean_ran, sigma_u1, true);
}

for(int k=0; k < ntimes; k++){
  f -= dnorm(u2[k], mean_ran, sigma_u2, true);
}

for(int i =0; i < nobs; i++){
  j = tree[i];
  k = times[i];
  f -= dnorm(y[i],(beta[0]+u1[j]+u2[k])/(1+exp(-(t[i]-beta[1])
}
```

orange5.cpp Including covariance structure (part 1)

```
    .
    .
    .
for(int j=0; j < ntrees; j++){
   f -= dnorm(u[j], beta[0] , sigma_u, true);
}

for(int i =0; i < nobs; i++){
  j = tree[i];
  pred[i] = u[j]/(1+exp(-((t[i]-beta[1])/beta[2] +
            season[i]*beta[3])));
  res[i] = y[i]-pred[i];
}
```

orange5.cpp Including covariance structure (part 2)

```
matrix<Type> cov(nobs,nobs);
for (int i=0;i<ntrees;i++)
{
  for (int j=0;j<ntimes;j++)
    {
      cov(i*ntimes+j,i*ntimes+j) = sigma * sigma;
      for (int k=0;k<j;k++)
        {
          cov(i*ntimes+k,i*ntimes+j) = sigma * sigma *
                                exp(- phi * Type(2)/Type(365) *
                                      (t[j]-t[k]));
          cov(i*ntimes+j,i*ntimes+k) =
                                  cov(i*ntimes+k,i*ntimes+j);
        }
    }
}

MVNORM_t<Type> neg_log_density(cov);
f += neg_log_density(res);
```

Table: Parameter estimates (and standard errors) and log-likelihoods for models estimated for the orange tree data.

Model	β_1	β_2	β_3	β_4	σ	σ_{u1}	σ_{u2}	ρ	$\log(L)$
(**??**)	192.1	727.9	348.1		7.84	31.6			-131.57
	(15.7)	(35.3)	(27.1)						
(1)	196.2	748.4	352.9		5.30	32.6	10.5		-125.45
	(19.4)	(62.3)	(33.3)						
(2)	217.1	857.5	436.8	0.322	4.79	36.0			-116.79
	(18.1)	(42.0)	(24.5)	(0.038)					
(1) + (3)	192.4	730.1	348.1		6.12	32.7	12.0	0.773	-118.44
	(19.6)	(63.8)	(34.2)						
(2) + (3)	216.2	859.1	437.8	0.330	5.76	36.7		0.811	-106.18
	(17.6)	(30.5)	(21.6)	(0.022)					

Overview

1. General mixed effects models
2. Laplace approximation
3. Template Model Builder