Introduction to General and Generalized Linear Models

General Mixed effects models and automatic differentiation

Jan Kloppenborg Møller, Henrik Madsen, and Anders Nielsen

May 7, 2012

Today

- General mixed effect models
- Laplace approximation
- Automatic differentiation (in R)
- Course summary

General mixed effect models

Remember that the likelihood of a general mixed effect model can be written as

$$L_M(\theta; \mathbf{y}) = \int_{\mathbb{R}^q} L(\boldsymbol{\theta}; \mathbf{u}, \mathbf{y}) d\mathbf{u} \tag{1}$$

where $\theta = (\boldsymbol{\beta}, \psi)$ and

$$L(\boldsymbol{\theta}; \mathbf{y}) = f_{Y|u}(\mathbf{y}; \mathbf{u}, \boldsymbol{\beta}) f_U(\mathbf{u}; \boldsymbol{\psi}) \tag{2}$$

is the joint likelihood.

General mixed effect models - one level of grouping

If only one level of grouping is present

$$L(\boldsymbol{\theta}; \mathbf{y}) = \prod_{i=1}^{M} \int_{\mathbb{R}^{q_i}} f_{Y|u_i}(\mathbf{y}; \mathbf{u}_i, \boldsymbol{\beta}) f_{U_i}(\mathbf{u}_i; \boldsymbol{\psi}) \, d\mathbf{u}_i \tag{3}$$

Which simplify the integration greatly, but might stille be impossible to solve, even when $q_i = 1$.

Laplace approximation

Approximate the joint log-likelihood by the second order Taylor approximation

$$l_{LA}(\boldsymbol{\theta}; \mathbf{u}, \mathbf{y}) \approx l(\boldsymbol{\theta}; \tilde{\mathbf{u}}, \mathbf{y}) - \frac{1}{2}(\mathbf{u} - \tilde{\mathbf{u}})^T \mathbf{H}(\tilde{\mathbf{u}})(\mathbf{u} - \tilde{\mathbf{u}}) \tag{4}$$

with

$$\tilde{\mathbf{u}} = \arg\max_{\mathbf{u}} l(\mathbf{u}, \boldsymbol{\theta}, \mathbf{y}) \tag{5}$$

and

$$H(\tilde{\mathbf{u}}) = -l''_{uu}(\mathbf{u}, \boldsymbol{\theta}, \mathbf{y})|_{\mathbf{u}=\tilde{\mathbf{u}}} \tag{6}$$

The log likelihood is approximated by

$$l_{M,LA}(\boldsymbol{\theta}, \mathbf{y}) = \log f_{Y|u}(\mathbf{y}; \tilde{\mathbf{u}}, \boldsymbol{\beta}) + \log f_U(\tilde{\mathbf{u}}; \boldsymbol{\psi}) - \frac{1}{2} \log |H(\tilde{\mathbf{u}})| \tag{7}$$

Laplace approximation

As usual the maximum likelihood estimate $\hat{\boldsymbol{\theta}}$ is given by

$$\hat{\boldsymbol{\theta}} = \arg\max_{\theta}(l_{M,LA}(\boldsymbol{\theta}, \mathbf{y})) \tag{8}$$

In general we need to solve (8) by numerical methods.

Each step in the numerical procedure require the numerical solution of the second stage model *and* the (an approximation of) the hessian at the estimated random effects.

In general numerical optimisation wil speed up significantly if we can supply gradients of the objective function.

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 $\tilde{\boldsymbol{u}}_{\boldsymbol{\theta}}$ and corresponding Hessian $H(\tilde{\boldsymbol{u}}_{\boldsymbol{\theta}})$.
2. Use $\tilde{\boldsymbol{u}}_{\boldsymbol{\theta}}$ and $H(\tilde{\boldsymbol{u}}_{\boldsymbol{\theta}})$ 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.

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

- ADMB (`http://admb-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.

AD Model Builder

- AD 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.
- The central feature is automatic differentiation (AD), which is implemented in such a way that the user rarely has to think about it at all.
- AD Model Builder can be used for fixed effects models, but in addition it includes Laplace approximation and importance sampling for dealing with general mixed effects models.
- AD Model Builder is developed by Dr. Dave Fournier and was a commercial product for many years. Recently AD Model Builder has been placed in the public domain (see `http://www.admb-project.org`).

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

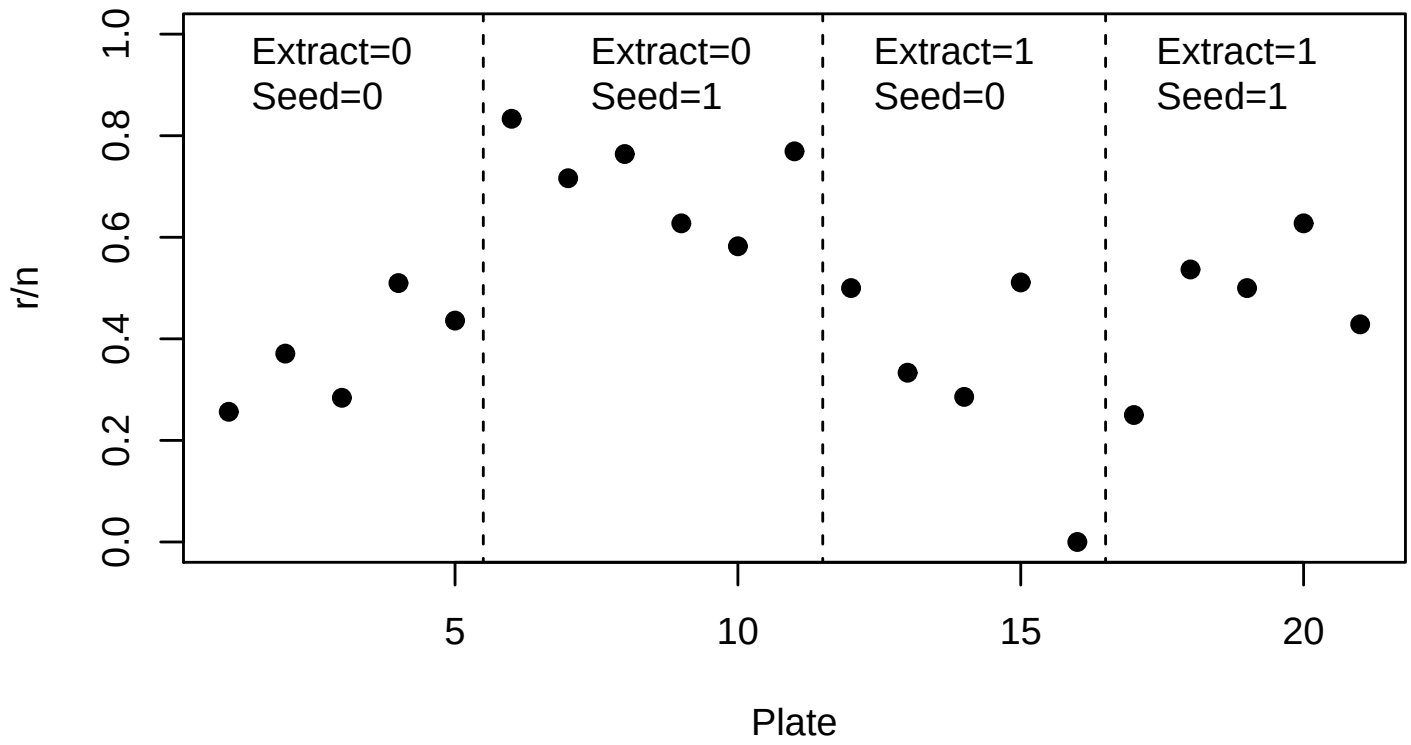
Example of use

```
> 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
}
```

Example (Example 5.7 in the book)

The book show the implementation i ADMB, here we will use R

- Proportion (r_i/n_i) of germinated seeds on N=21 plates
- Classified by seed (0=bean, 1=cucumber) and type of extract (0 or 1)



Example: Model

Logistic regression with overdispersion

$$r_i \sim Bin(n_i, p_i) \tag{9}$$

$$p_i = \frac{e^{\mathbf{X}\beta + B_i}}{1 + e^{\mathbf{X}\beta + B_i}} = \frac{e^{\mu + \beta_1 e_i + \beta_2 s_i + \beta_3 s_i : e_i + B_i}}{1 + e^{\mu + \beta_1 e_i + \beta_2 s_i + \beta_3 s_i : e_i + B_i}} \tag{10}$$

$$B_i \sim N(0, \sigma^2) \tag{11}$$

Random effects do seperate so

$$L_M = \prod_{i=1}^{21} \int_{\mathbb{R}} f_{Y|B_i}(\cdot) f_{B_i}(\cdot)\, dB_i \tag{12}$$

but the integral still does not allow an anlytical solution.

Example: Model

Log likelihood of $\mathbf{B}$

$$
\begin{aligned}
l(\boldsymbol{\theta}, \mathbf{B}, \mathbf{r}, \mathbf{n}) =& \log f_{Y|u}(\mathbf{r}, \mathbf{n}; \boldsymbol{\beta}, \mathbf{B}) + \log f_U(\mathbf{B}; \boldsymbol{\sigma}^2) && (13)\\
=& \sum_{i=1}^{21} \log \binom{n_i}{r_i} + r_i \log(p_i) + (n_i - r_i) \log(1 - p_i) - \\
& \frac{1}{2}\left(\log(2\pi) + \log(\sigma^2) + \frac{B_i}{\sigma^2}\right) && (14)
\end{aligned}
$$

and

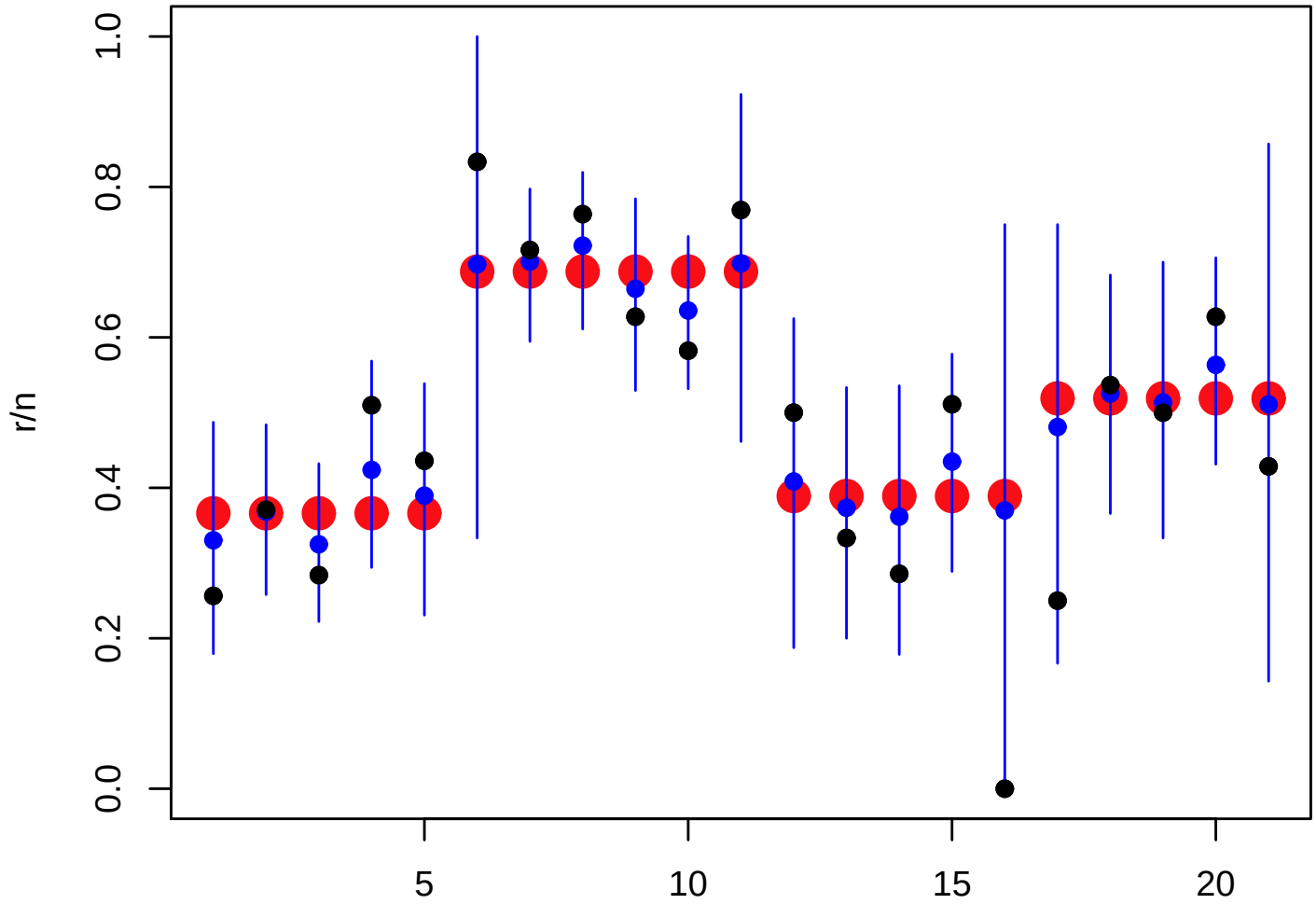
$$
l_{M,LA} = l(\boldsymbol{\theta}, \tilde{\mathbf{B}}, \mathbf{r}, \mathbf{n}) - \frac{1}{2} \sum_{i=1}^{21} \log l_{B_i, B_i}(\boldsymbol{\theta}, \tilde{B}_i, r_i, n_i) \qquad (15)
$$

We have used that $\mathbf{B} \sim (\mathbf{0}, \sigma^2 \mathbf{I})$.
It is straight forward to find l_{B_i} and $l_{B_i B_i}$ by `D` and `deriv` in R.

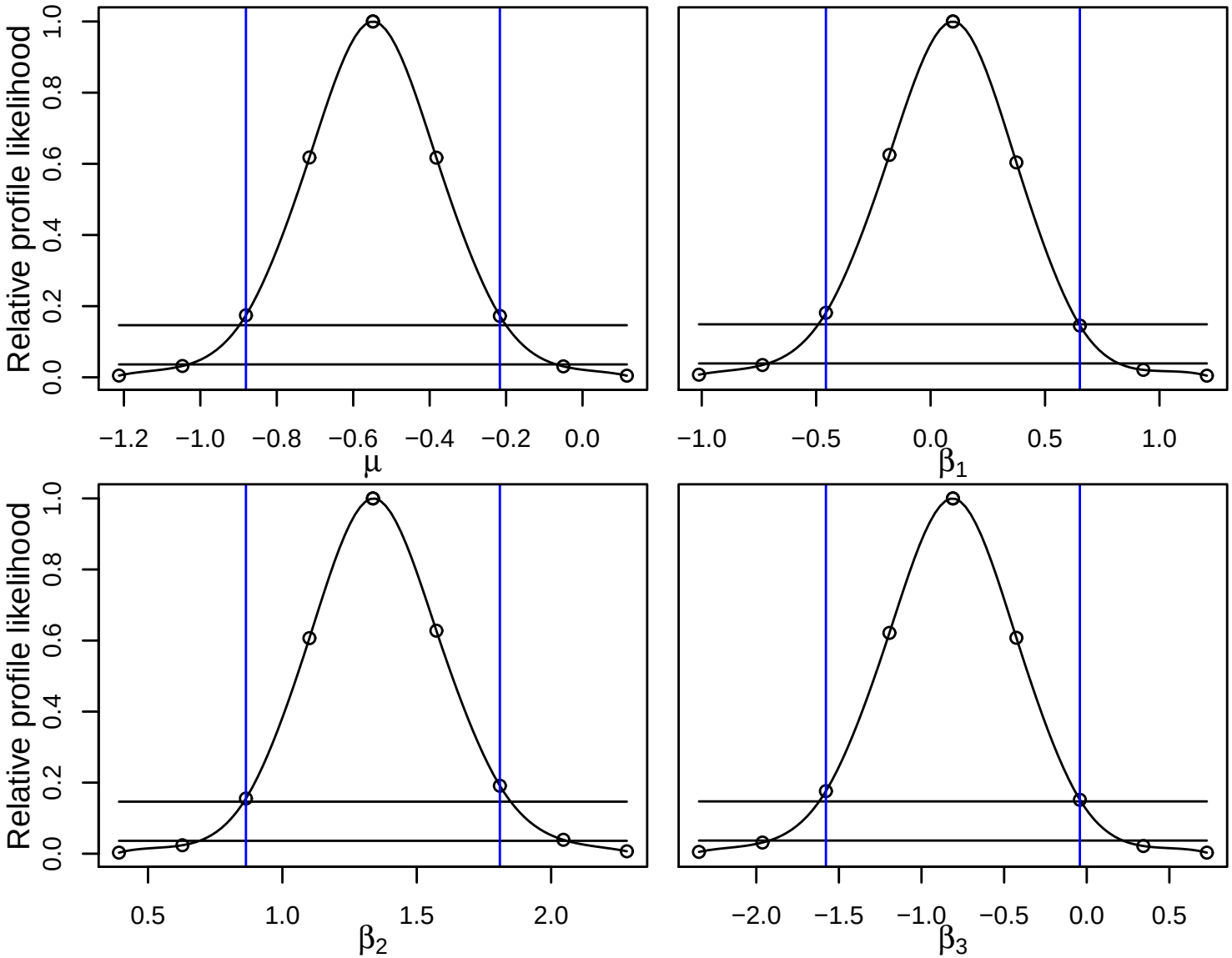
Example: Results

Variable	$\hat{\theta}$	S.E	95%C.I.
μ	-0.548	0.166	(-0.881,-0.216)
β_1	0.097	0.277	(-0.457,0.652)
β_2	1.337	0.236	(0.864,1.809)
β_3	-0.810	0.384	(-1.578,-0.042)
σ	0.235	0.110	(0.016,0.454)

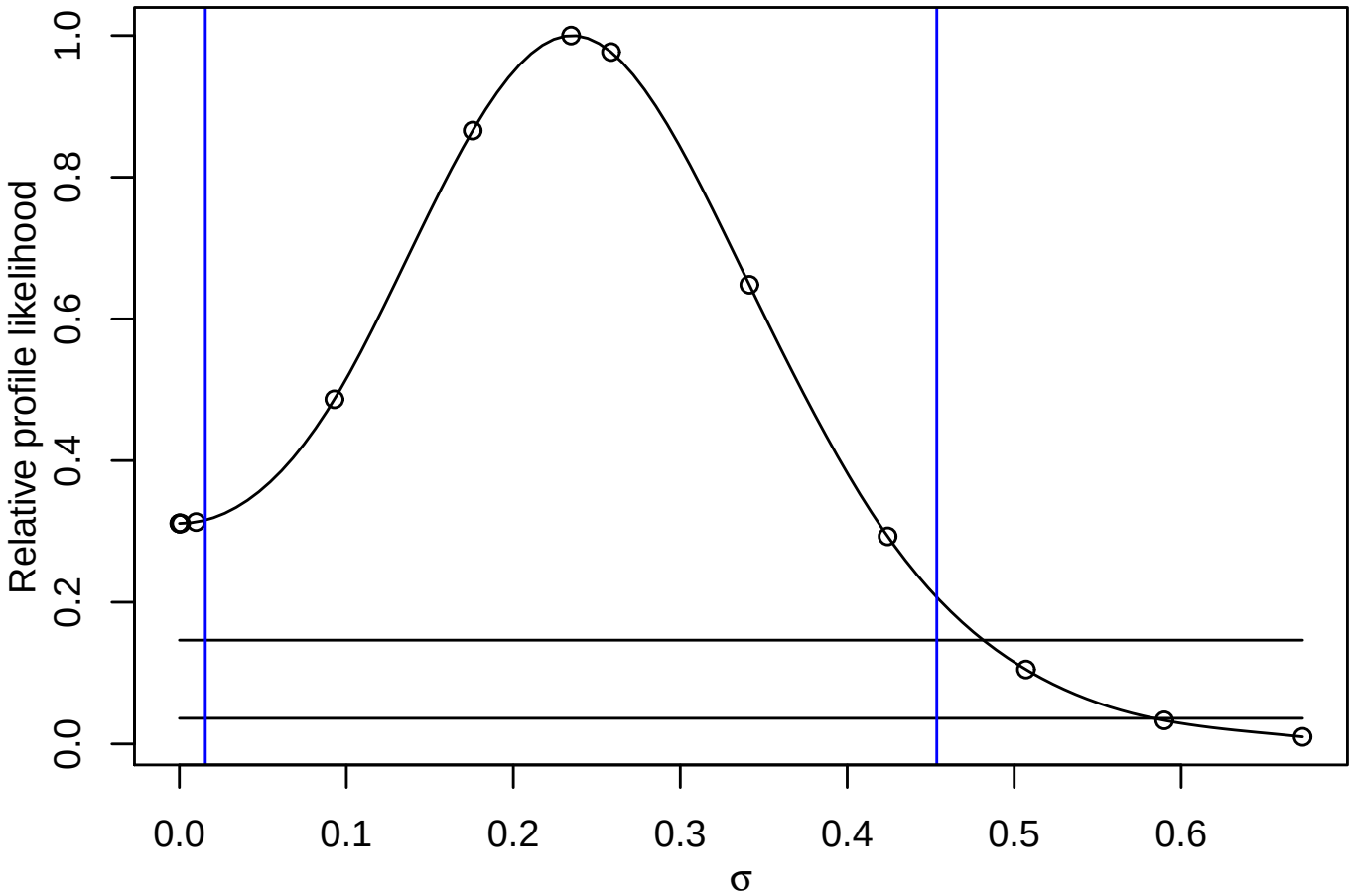
Example: Plots results



Example: Profile likelihood (Fixed effects)



Example: Profile likelihood (Random effects)



Conclusion

- We have obtained significant speed up using automatic differentiation in R
- More complicated structures might be difficult to implement in R
- The full use of autoamtic differentiation require $\frac{\partial \tilde{\boldsymbol{u}}}{\partial \boldsymbol{\theta}}$ to be calculated, which is difficult to implement, but is an integreated part of ADMB