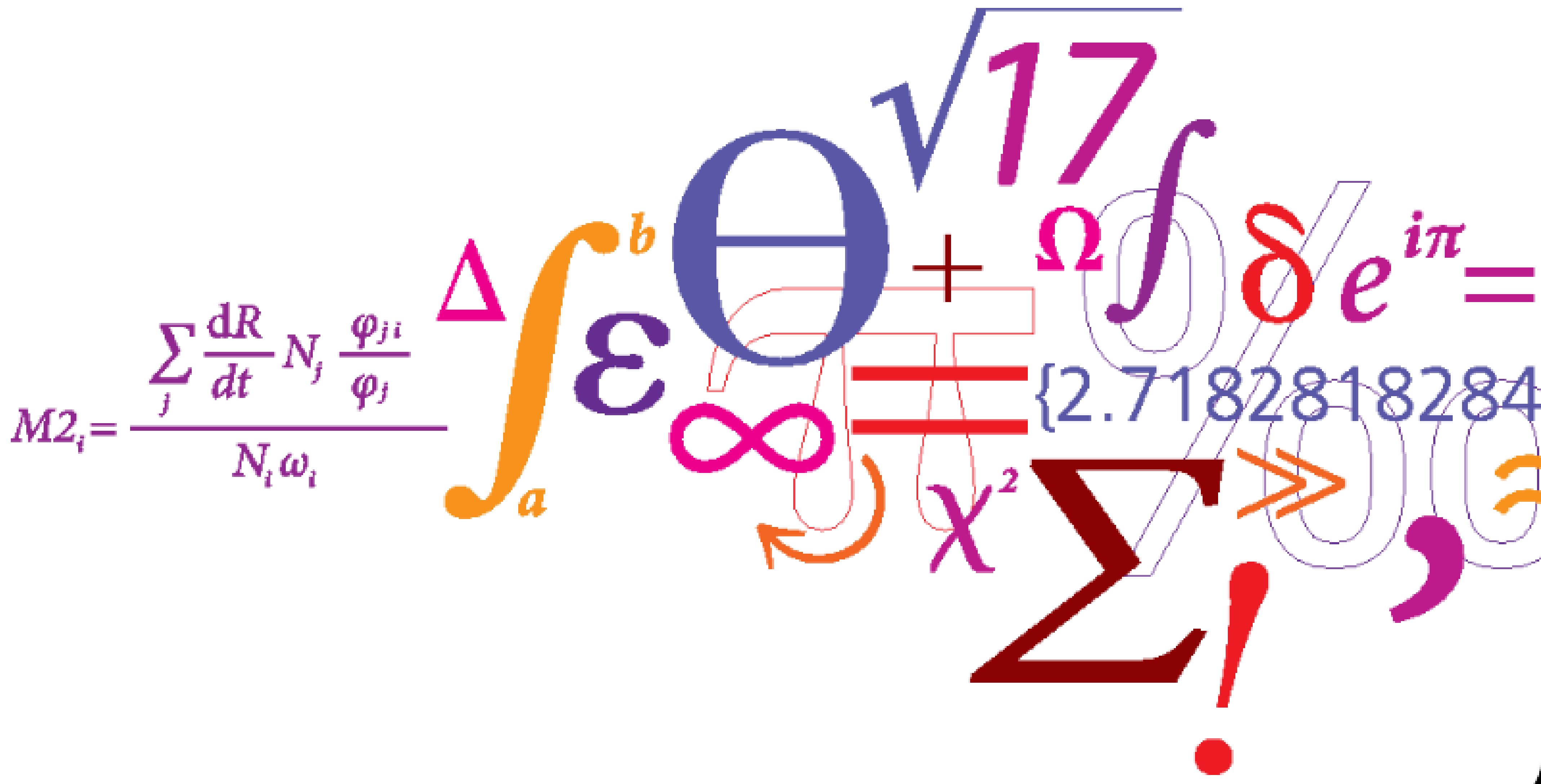
DTU

Introduction to TMB

Christoffer Moesgaard Albertsen

cmoe@aqua.dtu.dk

July 28, 2014



DTU Aqua
National Institute of Aquatic Resources

History of ADMB/TMB

Year	Event
Late 1980s:	David Fournier develops AUTODIFF and ADMB
1988:	Otter Research Ltd. founded by David Fournier and John Sibert
1991:	First release of AUTODIFF
1993:	First release of ADMB
2004:	Random-effects in ADMB developed by Hans Skaug & David Fournier
2007:	ADMB Foundation founded
2008:	Software was bought from Otter Research Ltd. and made available
2009:	The source code of ADMB is available
2009:	Kasper Kristensen begins development of TMB (at that time RcppAD)
2013: 10 Sep	TMB is made available at GitHub
2013: 18-22 Sep	TMB is presented at ADMB Developers Workshop, Reykjavik
2014: Jul	TMB paper submitted

Wikipedia (2014), Bolker et al. (2013), Kristensen (2013), A. Magnusson (2013)

What is TMB?

- R package for fitting statistical models to data
- The user implements the model in C++ in a *user template*
- TMB uses C++ *template classes/functions* to reduce user code
- R is used for pre- and post-processing of data
- And for estimation (optim, nlminb,...)
- Free and open source

https://github.com/kaskr/adcomp/

Why do we need it?

- Why not just use a standard package or function (lm, glm, lmer,ar,glm.nb,...)?
- Why can't we just write the likelihood function in R?

I will show you that:

- TMB can handle a wide variety of standard and non-standard models.
- TMB is fast.
- It is easy to:
  - switch between models,
  - fix parameters,
  - calculate derived quantities (and their uncertanties).

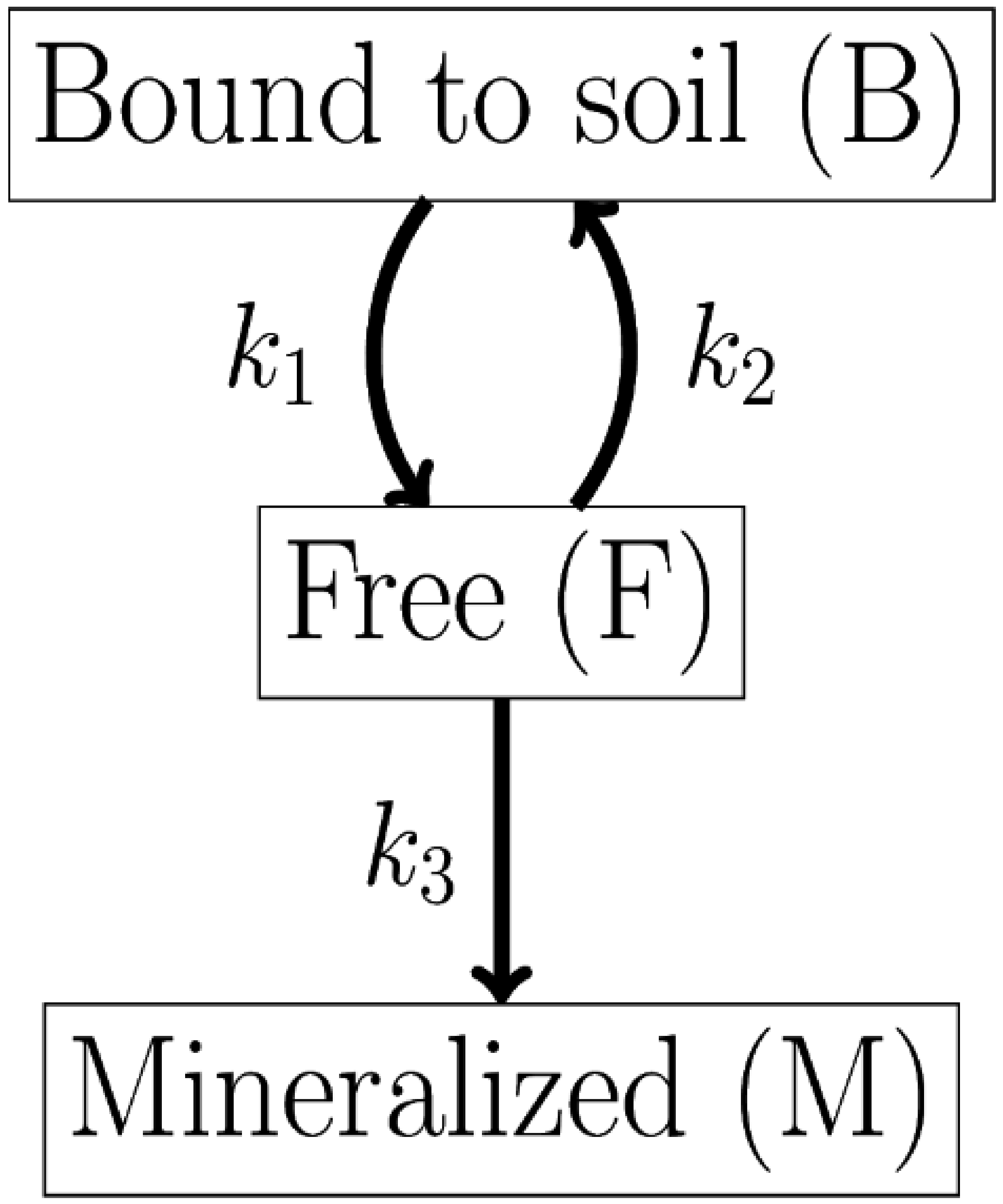
Mineralization of herbicide

- $M_t = 100 - B_t - F_t$
- Define $X_t = \begin{pmatrix} B_t \\ F_t \end{pmatrix}$, $X_0 = \begin{pmatrix} 0 \\ 100 \end{pmatrix}$
- Then the system is:

$$\frac{dX_t}{dt} = \begin{pmatrix} -k_1 & k_2 \\ k_1 & -(k_2 + k_3) \end{pmatrix} X_t = AX_t$$

- This is solved by $X_t = e^{At} X_0$
- Model:

$$M_{t_i} \overset{iid}{\sim} \mathcal{N}(100 - \sum X_{t_i}, \sigma^2)$$



Bolker et al. (2013)

Mineralization - R code

```
library(Matrix)
nlogL<-function(theta){
  k<-exp(theta[1:3])
  sigma<-exp(theta[4])
  A<-rbind(
       c(-k[1], k[2]),
       c( k[1], -(k[2]+k[3]))
     )
  x0<-c(0,100)
  sol<-function(t)100-sum(expm(A*t)%*%x0)
  pred<-sapply(dat[,1],sol)
  -sum(dnorm(dat[,2],mean=pred,sd=sigma, log=TRUE))
}
c(system.time(fit <- optim(rep(-2,4),nlogL))['elapsed'],
"value"=fit$value,
"convergence"=fit$convergence)
```

```
##     elapsed       value convergence
##       9.169      19.269       0.000
```

Mineralization - optimizers

```r
library(optimx)
mthds <- c("bobyqa","nlm","newuoa","Nelder-Mead","ucminf","spg",
           "L-BFGS-B","nlminb")
fit<-optimx(rep(-2,4),nlogL,method=mthds)
```

method	value	fevals	gevals	convcode	kkt1	kkt2	xtimes
bobyqa	153.305623	139		0	TRUE	FALSE	5.757
nlm	102.766038			0	TRUE	FALSE	7.230
newuoa	91.174679	1461		0	TRUE	FALSE	58.283
Nelder-Mead	19.269052	223		0	FALSE	FALSE	8.857
ucminf	0.939218	40	40	0	FALSE	TRUE	8.892
spg	0.939214	221		0	FALSE	TRUE	36.401
L-BFGS-B	0.939214	79	79	0	FALSE	TRUE	28.683
nlminb	0.939214	33	120	0	TRUE	TRUE	6.200

Mineralization - using TMB

method	value	fevals	gevals	convcode	kkt1	kkt2	xtimes
bobyqa	153.305623	137		0	TRUE	FALSE	0.012
newuoa	153.305622	270		0	TRUE		0.022
nlm	102.766038			0	TRUE		0.013
Nelder-Mead	19.269052	223		0	FALSE	FALSE	0.015
spg	0.939214	229		0	TRUE	TRUE	0.186
L-BFGS-B	0.939214	72	72	0	TRUE	TRUE	0.013
ucminf	0.939214	53	53	0	TRUE	TRUE	0.009
nlminb	0.939214	33	30	0	TRUE	TRUE	0.006

Calculating derivatives

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Analytical:

- Difficult (and a lot of work) in many cases

Numerically:

- $f'(x) \approx \frac{f(x+h)-f(x)}{h}$ for small (arbitrary) h.
- Simple, but slow and inaccurate.
- There are more advanced methods to increase accuracy

Automatic differentiation

- Given a computer program that defines a function,

  AD computes the derivatives.

- A computer program is a long list of operations ("+","*","exp","sqrt",...).

- The chain rule tells us how to combine them:

$$(f(g(x)))' = f(g(x))'g(x)'$$

- The computer just need to:

  - Keep track of all the simple operations used

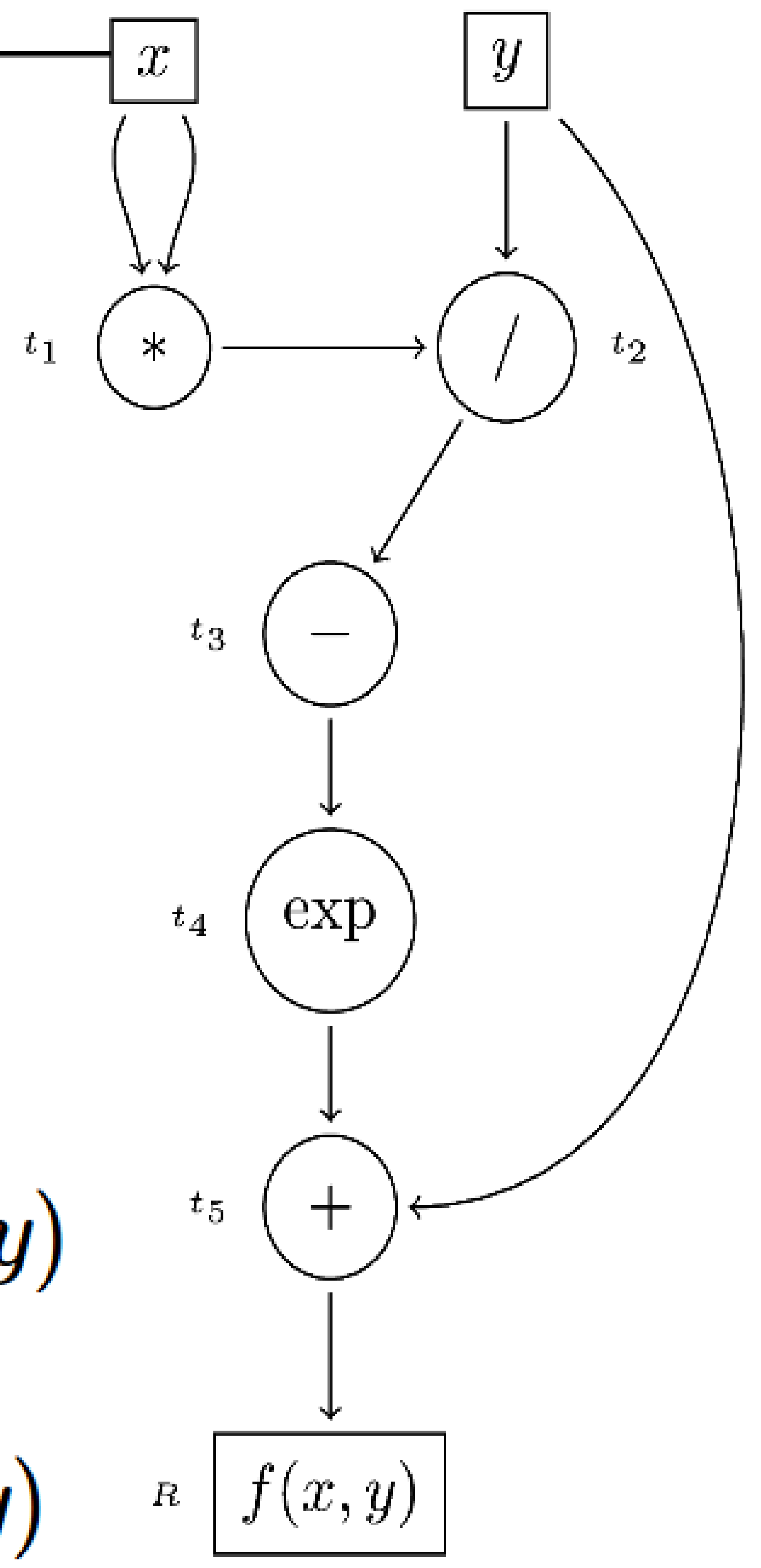
  - Use the simple derivative formulas and the chain rule

Forward-mode AD

Calculating $\frac{\partial f(x,y)}{\partial x}$:

Operation:	Value:	Derivative:
$t_1 = x \cdot x$	x^2	$\frac{\partial t_1}{\partial x} = 2x$
$t_2 = t_1/y$	x^2/y	$\frac{\partial t_2}{\partial t_1}\frac{\partial t_1}{\partial x} = 1/y \cdot 2x$
$t_3 = -t_2$	$-x^2/y$	$\frac{\partial t_3}{\partial t_2}\frac{\partial t_2}{\partial x} = -1 \cdot 2x/y$
$t_4 = \exp(t_3)$	$\exp(-x^2/y)$	$\frac{\partial t_4}{\partial t_3}\frac{\partial t_3}{\partial x} = \exp(t_3) \cdot (-2x/y)$
$t_5 = t_4 + y$	$\exp(-x^2/y) + y$	$\frac{\partial t_5}{\partial t_4}\frac{\partial t_4}{\partial x} = 1 \cdot (-\exp(t_3) \cdot 2x/y)$
$R = t_5$	$\exp(-x^2/y) + y$	$\frac{\partial R}{\partial t_5}\frac{\partial t_5}{\partial x} = -2x/y \cdot \exp(-x^2/y)$

$f(x,y) = \exp\left(\frac{-x^2}{y}\right) + y$



Reverse-mode AD

Step 1: Run through graph to calculate $t_1, \cdot, t_5, R$

Step 2: Run backwards to calculate derivatives of R w.r.t. notes:

$$\frac{\partial R}{\partial t_5} = 1$$

$$\frac{\partial R}{\partial t_4} = \frac{\partial R}{\partial t_5}\frac{\partial t_5}{\partial t_4} = 1 \cdot 1$$

$$\frac{\partial R}{\partial t_3} = \frac{\partial R}{\partial t_4}\frac{\partial t_4}{\partial t_3} = 1 \cdot \exp(t_3)$$

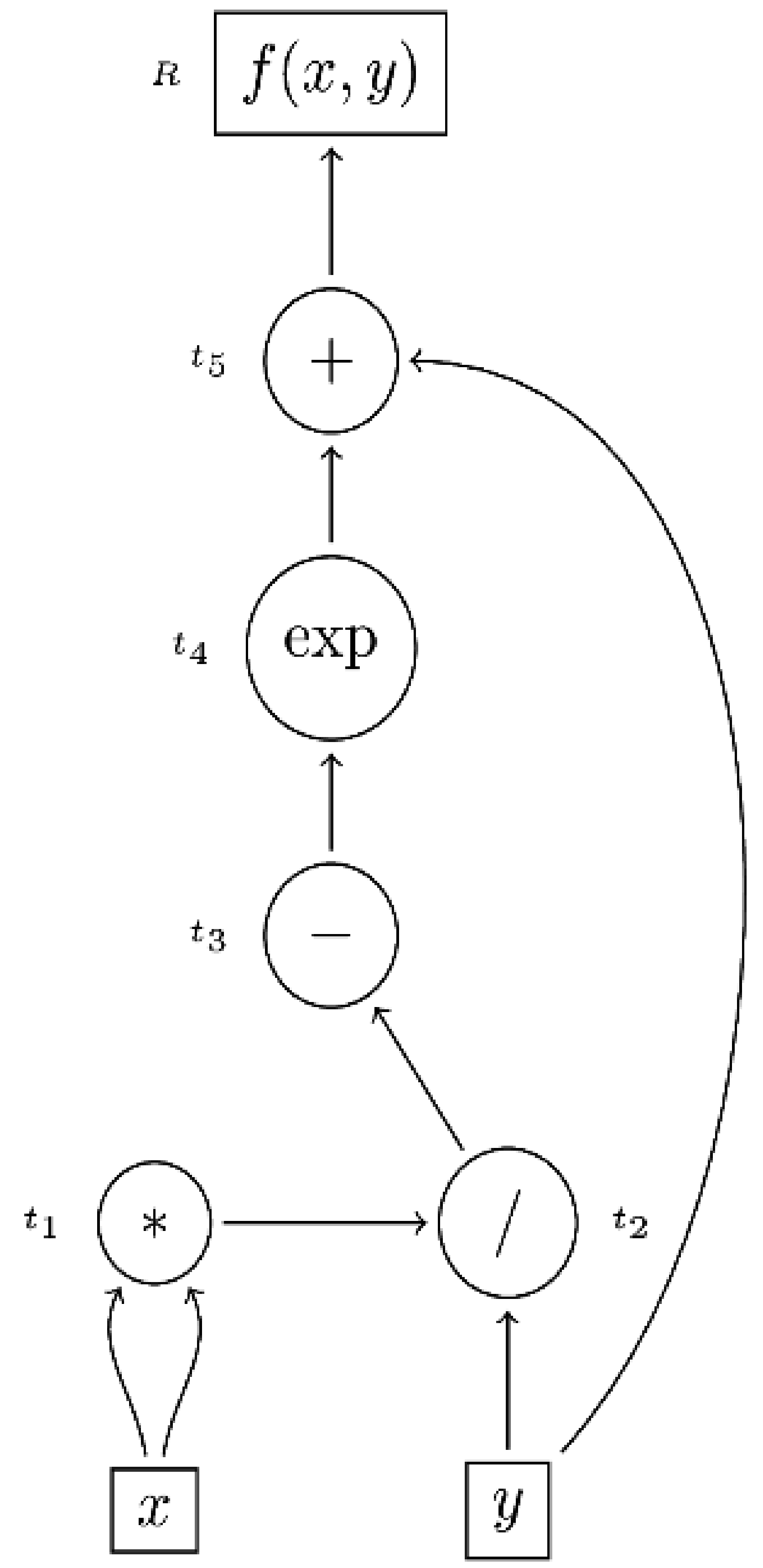
$$\frac{\partial R}{\partial t_2} = \frac{\partial R}{\partial t_3}\frac{\partial t_3}{\partial t_2} = \exp(t_3) \cdot (-1)$$

$$\frac{\partial R}{\partial t_1} = \frac{\partial R}{\partial t_2}\frac{\partial t_2}{\partial t_1} = -\exp(t_3) \cdot 1/y$$

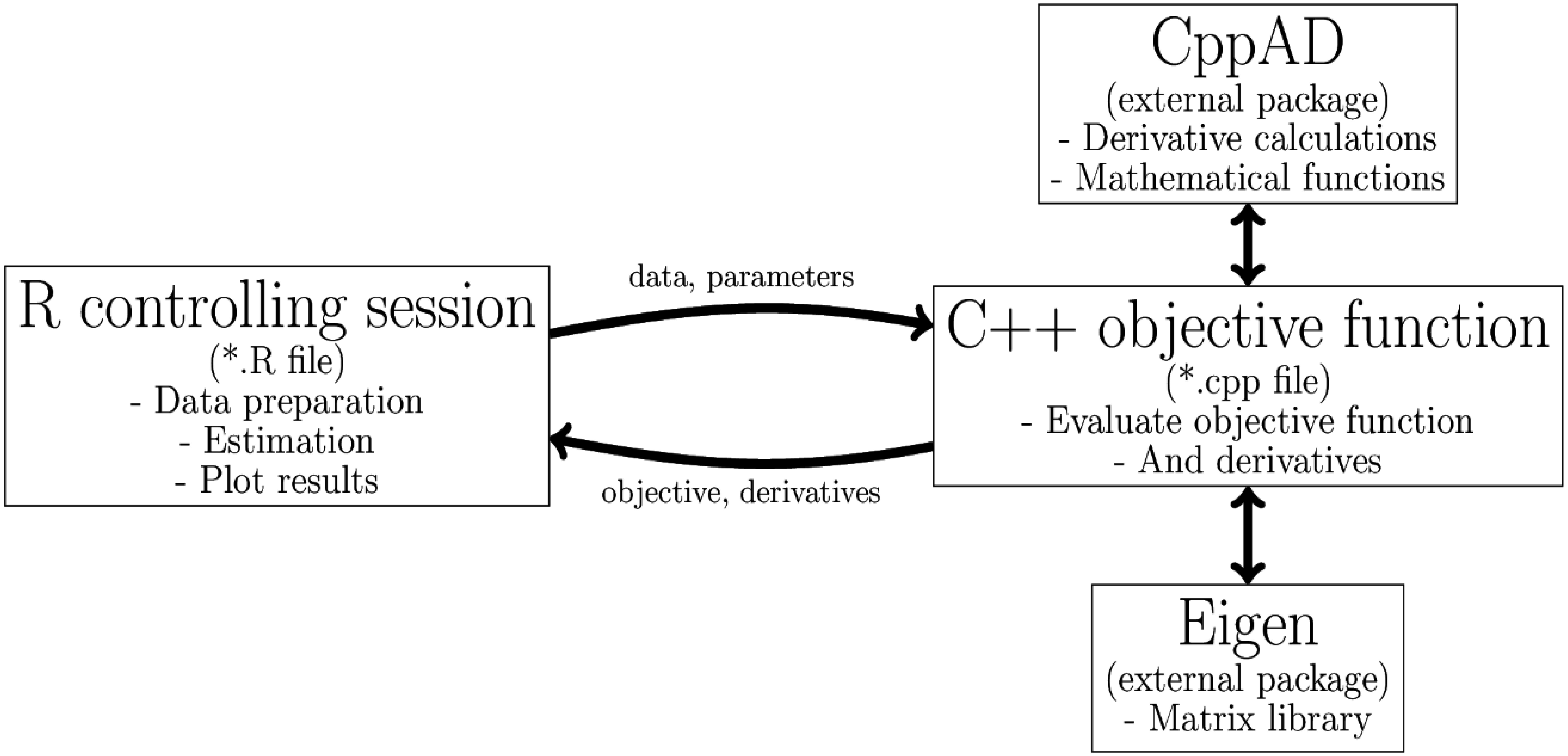
$$\frac{\partial R}{\partial y} = \frac{\partial R}{\partial t_2}\frac{\partial t_2}{\partial y} + \frac{\partial R}{\partial t_5}\frac{\partial t_5}{\partial y} = -\exp(t_3) \cdot (-1/y^2) + 1 \cdot 1$$

$$\frac{\partial R}{\partial x} = \frac{\partial R}{\partial t_1}\frac{\partial t_1}{\partial x} = -\exp(t_3)/y \cdot 2x = -2x/y \cdot \exp(-x^2/y)$$

$$f(x,y) = \exp\left(\frac{-x^2}{y}\right) + y$$



Building models in TMB



http://folk.uib.no/hsk021/tmbdoc/group__Structure__TMB.html

Models in TMB - C++

- Include TMB specific macros and functions

```
#include <TMB.hpp>
```

- Start the definition of the objective function

```
template<class Type>
Type objective_function<Type>::operator() ()
{
```

Models in TMB - C++

- Define data and parameters

Data	Parameters	R equivalent
DATA_SCALAR	PARAMETER	numeric(1)
DATA_VECTOR	PARAMETER_VECTOR	vector
DATA_MATRIX	PARAMETER_MATRIX	matrix
DATA_ARRAY	PARAMETER_ARRAY	array
DATA_SPARSE_MATRIX	-	dgTMatrix
DATA_INTEGER	-	integer(1)
DATA_FACTOR	-	factor

Models in TMB - C++

- Code the objective function, e.g.,

```
  Type f;
  f = -sum(dnorm(x,mu,sigma,true));
```

- Report calculations back to R

```
REPORT(x-mu);
ADREPORT(pow(sigma,2));
```

- Return the objective function value

```
  return f;
```

- End the function definition

```
}
```

See http://folk.uib.no/hsk021/tmbdoc/modules.html for available distributions

Models in TMB - R

- load the package

```r
library(TMB)
```

- Compile the C++ code (when changed)

```r
compile("mycode.cpp")
```

- Load the C++ part into R

```r
dyn.load(dynlib("mycode"))
```

Models in TMB - R

- Define a data, a parameter, and a map list, e.g. dat, param, map.
  - dat is a list corresponding to the DATA_ variables
  - param is a list corresponding to the PARAMETER_ variables
  - map is a list of factors fixing parameter values
- Create the TMB object

```r
obj <- MakeADFun(data=dat,parameters=param,map=map,DLL="mycode")
```

Models in TMB - R

- Use the many functions and variables in the object for inference and calculations

Function/variable	Description
obj$par	Vector of initial parameters to estimate
obj$fn()	Objective function
obj$gr()	Objective gradient
obj$he()	Objective hessian (only fixed effects)
obj$report()	List of values reported by REPORT(...)
obj$env$parList()	List of all parameters
obj$env$last.par	Last evaluated parameters
obj$env$last.par.best	Best evaluated parameters
obj$env$value.best	Lowest function value encountered
sdreport(obj)	Estimates and st.dev. for parameters and ADREPORT values

Normal distribution - C++

```cpp
#include <TMB.hpp>

template<class Type>
Type objective_function<Type>::operator() ()
{
  DATA_VECTOR(x);
  PARAMETER(mu);
  PARAMETER(sigma);

  Type f;
  f = -sum(dnorm(x,mu,sigma,true));

  return f;
}
```

Normal distribution - R

- Compiling

```
library(TMB)
compile("norm.cpp")
dyn.load(dynlib("norm"))
x<-rnorm(100,3,2)
```

- Creating the TMB object

```
obj <- MakeADFun(data=list(x=x),parameters=list(mu=0,sigma=1))
```

- Estimating

```
fit <- nlminb(obj$par,obj$fn,obj$gr)
```

- Fix mu at the initial value

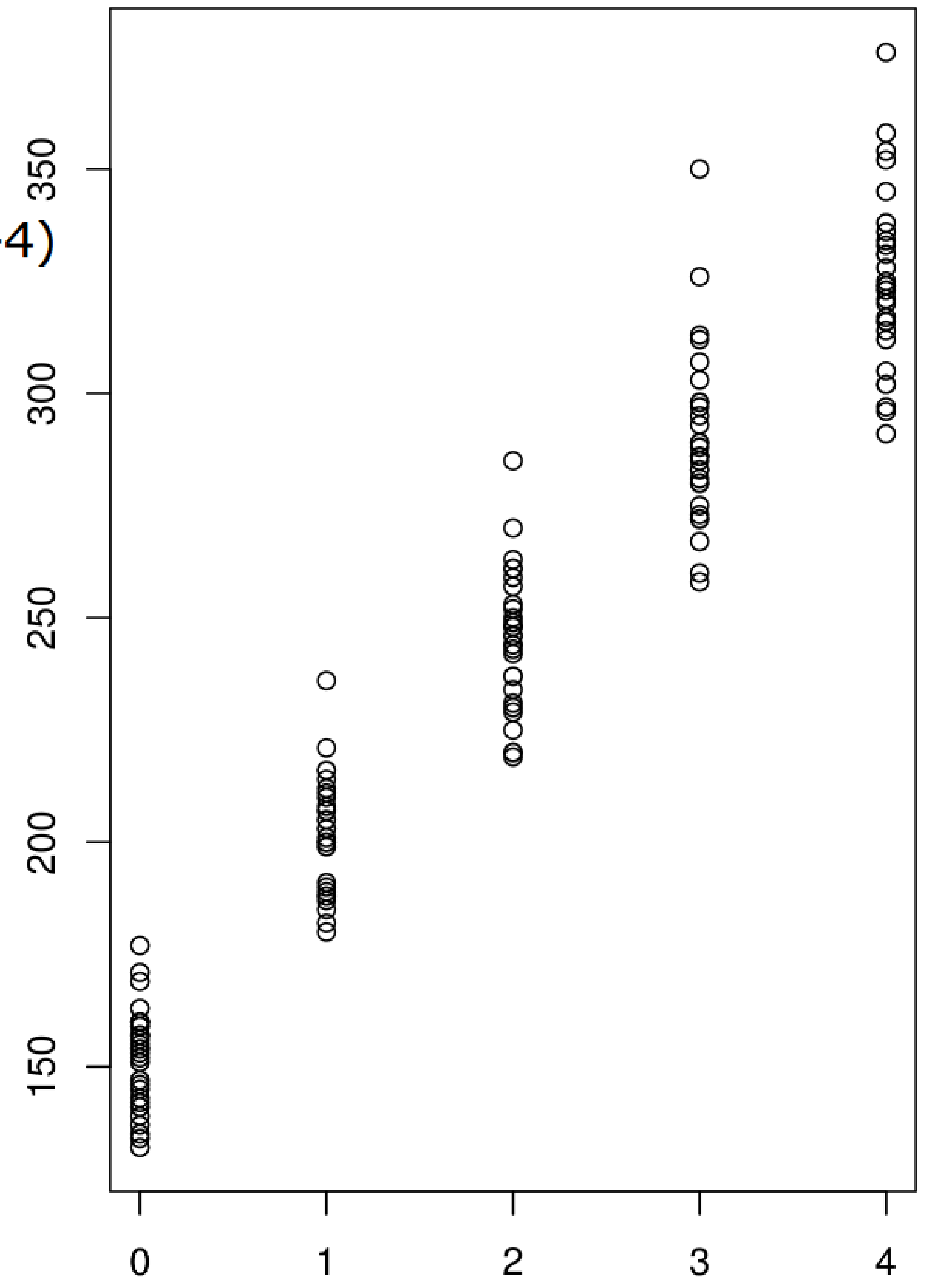
```
obj <- MakeADFun(data=list(x=x), parameters=list(mu=0,sigma=1),
                 map=list(mu=factor(NA)))
```

Half-normal distribution

```cpp
#include <TMB.hpp>
template<class Type>
Type ldhalfnorm(Type x, Type var){
  return 0.5*log(2)-0.5*log(var*M_PI)+pow(x,2)/(2*var);
}
template<class Type>
Type objective_function<Type>::operator() ()
{
  DATA_VECTOR(x);
  PARAMETER(sigma);
  Type f;
  for(int i = 0; i<x.size(); ++i){
    f = -ldhalfnorm(x(i),pow(sigma,2));
  }
  return f;
}
```

Rats' weight

- Setup:
  - 30 newborn rats weighed weekly (week 0-4)
- Model:
  - $X_i \sim \mathcal{N}(\alpha + \beta \cdot t_i, \sigma^2)$
  - X_i: weight of observation $i = 1, \ldots, 150$
  - t_i: week at observation $i = 1, \ldots, 150$
- Estimation in R:
  - lm(weight ~ week, data = rat_dat)



Gelfland and Hills (1990)

Rats' weight

```cpp
#include <TMB.hpp>

template<class Type>
Type objective_function<Type>::operator() ()
{
  DATA_VECTOR(weight);
  DATA_MATRIX(modMat);
  PARAMETER_VECTOR(beta);
  PARAMETER(logSd);

  Type sd = exp(logSd);
  ADREPORT(sd);
  vector<Type> mu = modMat * beta;
  Type nll = 0.0;

  nll -= sum(dnorm(weight, mu,sd,true));
  return nll;
}
```

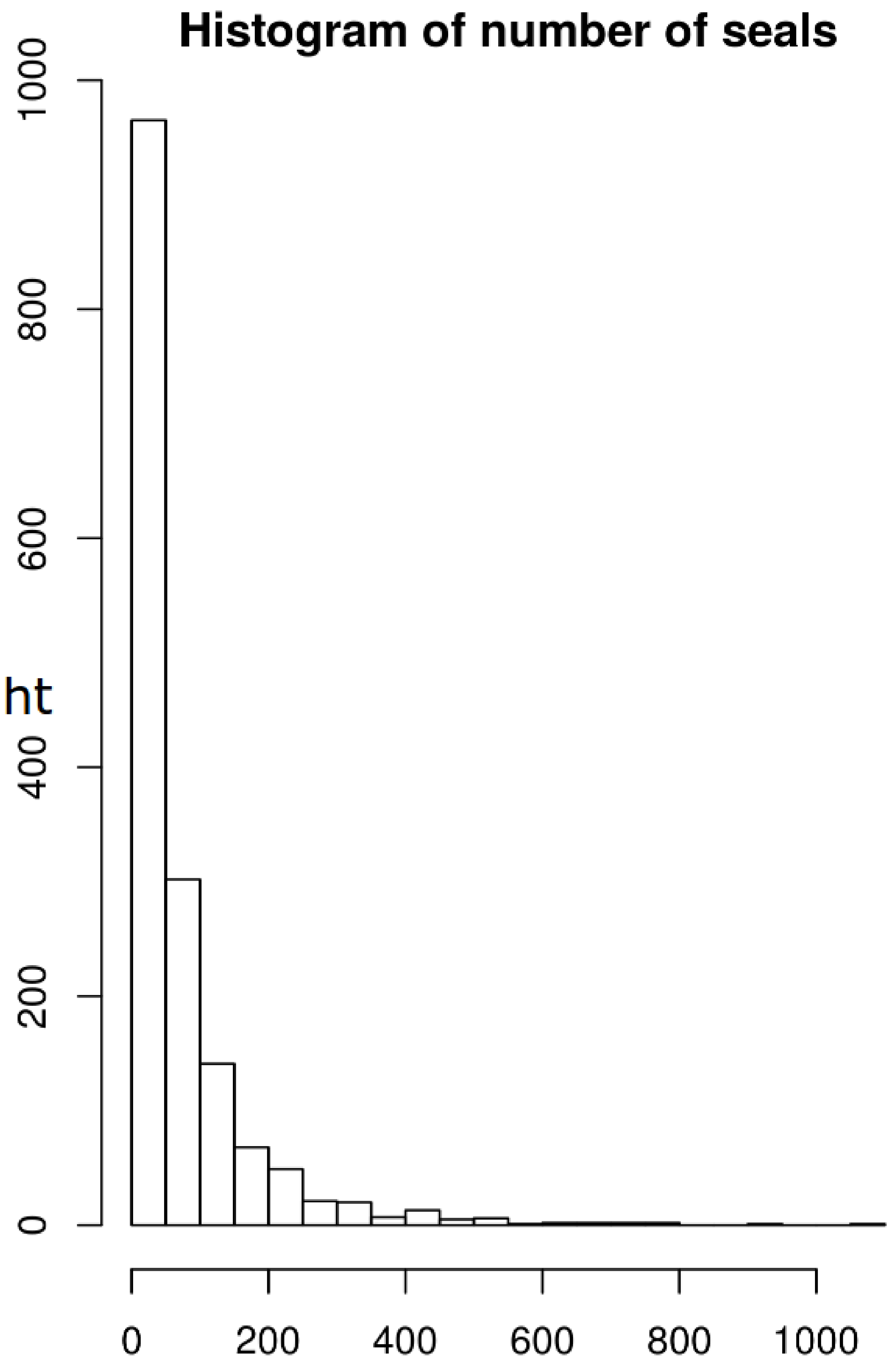
Rats' weight

- Estimates, standard errors and estimation time for TMB compared with lm.

	lm Estimate	lm Std. Error	tmb Estimate	tmb Std. Error
(Intercept)	156.05333	2.28145	156.05333	2.26619
week	43.30000	0.93140	43.30000	0.92517
sigma	16.13226		16.02435	0.92517
time	0.00400		0.00800	

Seal countings

- Setup:
  - Survey of harbor seals in Alaska 1998
- Model:
  - X_i follows negative binomial distribution
  - $E(X_i) = \exp(S(t_i) + \beta \cdot d_i)$
  - $S(t_i)$: spline, fractional hours since midnight
  - d_i: tide height relative to the low tide
  - $V(X_i) = E(X_i) \cdot (1 + \nu), \nu > 0$
- Estimation in R
  - glm.nb from MASS and splines with Hmisc



Hoef and Boveng (2007)

Seal countings

```
#include <TMB.hpp>
using namespace tmbutils;

template<class Type>
Type objective_function<Type>::operator() ()
{
  DATA_VECTOR(count);
  DATA_VECTOR(time);
  DATA_VECTOR(tide);
  DATA_VECTOR(knots_tm);
  DATA_VECTOR(xx);
  PARAMETER_VECTOR(beta_tm);
  PARAMETER(beta_td);
  PARAMETER(logvar);

  Type nll = 0.0;
  Type var = exp(logvar);
```

Seal countings

```
splinefun<Type> spl_tm (knots_tm,beta_tm);
vector<Type> fs_tm(xx.rows());
for(int i=0;i<xx.rows();i++){
  fs_tm(i) = spl_tm(xx(i));
}
ADREPORT(fs_tm);
```

Seal countings

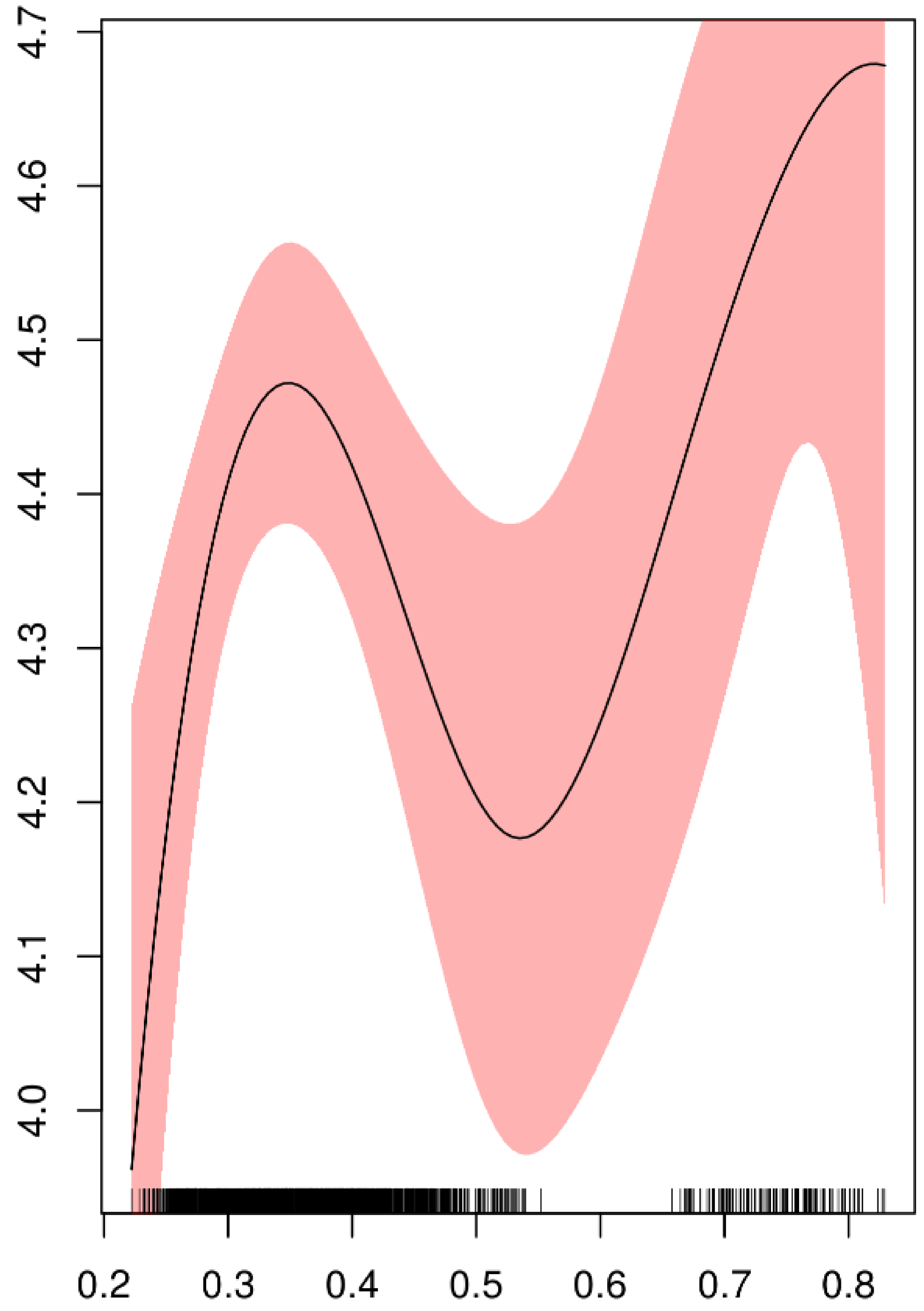
```
  vector<Type> mu(count.size());

  for(int i = 0; i < count.size(); ++i){
    mu(i) = exp(spl_tm(time(i))+beta_td*tide(i));
    nll -= dnbinom2(count(i),mu(i),mu(i)*(1.0+var),true);
  }
  REPORT(mu);

  return nll;
}
```

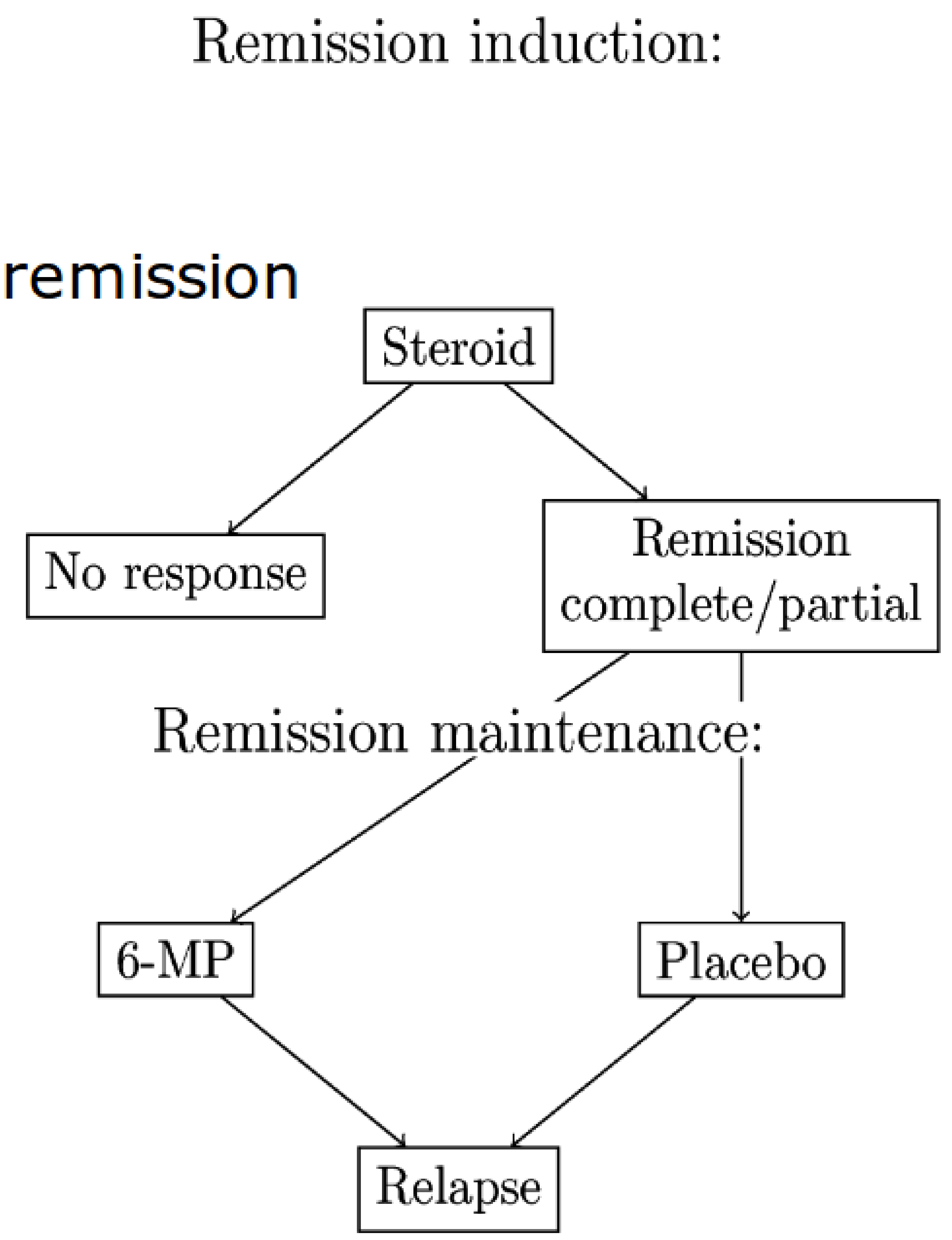
Seal countings

- Estimate of β: -0.3435 (sd: 0.0524)
- Estimate of $\log(\nu)$: -0.3435 (sd: 0.0524)
- Time to estimate: 0.145



Acute leukemia

- Setup:
  - 21 pairs of children with leukemia, monitor length of remission
- Model:
  - T_i follows Weibull distribution
  - $f(x;k,\theta) = \frac{k}{\theta}\left(\frac{x}{\theta}\right)^{x-1}\exp(-(x/\theta)^{k-1})$
  - shape: k; scale: $\theta = \exp(\boldsymbol{\beta} * \boldsymbol{X})^{-1/k}$
- Estimation in R
  - survreg from survival package (note different parameterization)



Freireich et al. (1963)

Acute leukemia

```
#include <TMB.hpp>
using namespace tmbutils;

template<class Type>
Type objective_function<Type>::operator() ()
{
  DATA_VECTOR(time);
  DATA_VECTOR(censored);
  DATA_MATRIX(modMat);
  PARAMETER_VECTOR(beta);
  PARAMETER(logshape);
```

Acute leukemia

```
  Type nll = 0.0;
  Type shape = exp(logshape);
  vector<Type> rate = modMat * beta;
  rate = rate.exp();
  for(int i = 0; i < time.size(); ++i){
    Type scale = pow(rate(i),-1.0/shape);
    if(censored(i) == 1){
      nll -= log(1.0-pweibull(time(i),shape,scale));
    }else{
      nll -= dweibull(time(i),shape,scale,true);
    }
  }
  return nll;
}
```

Other distributions

- Add a code for distribution

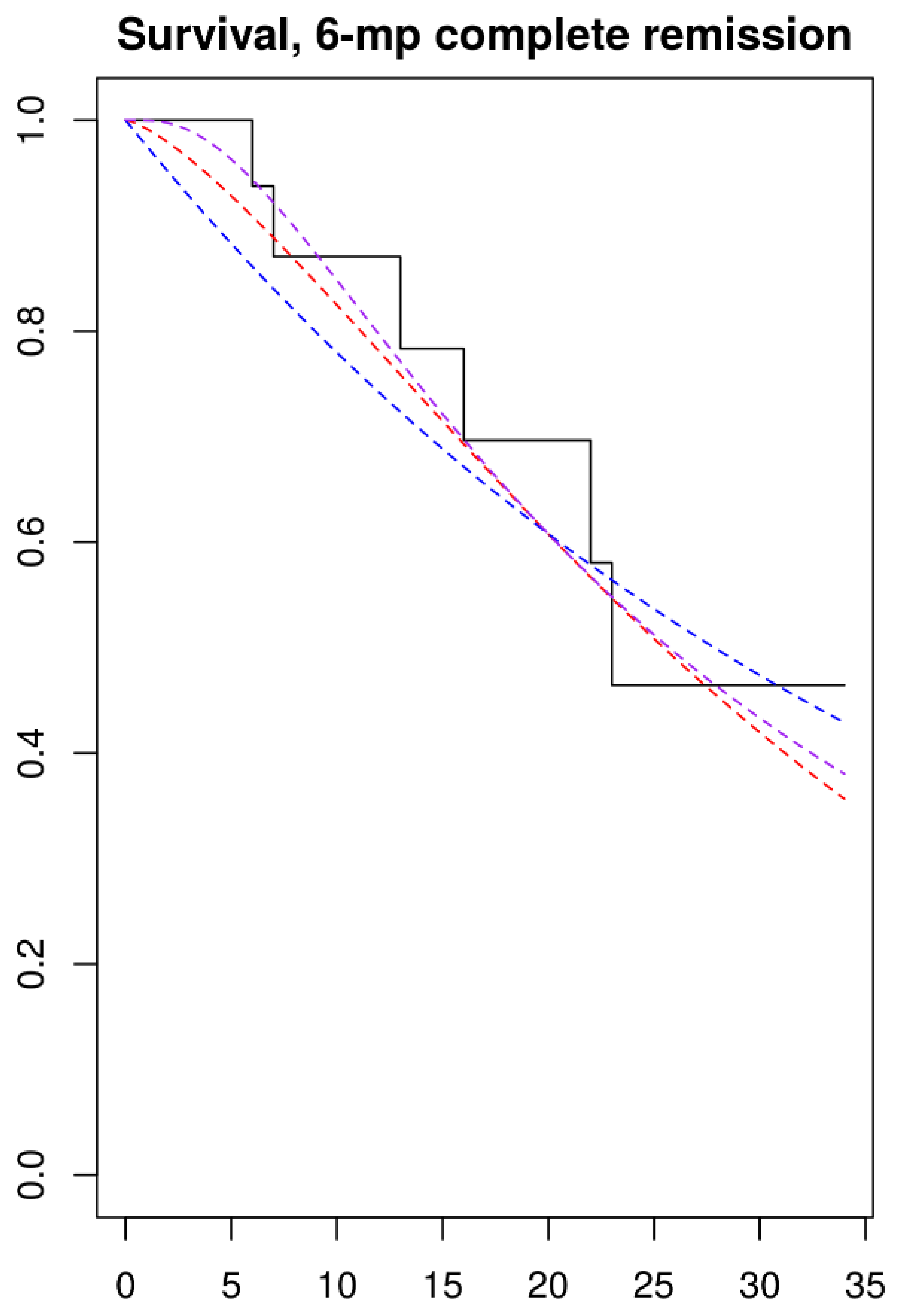
```
DATA_INTEGER(modCode);
```

- Switch between models (similar for uncensored)

```
switch(modCode){
  case 1: //Weibul (exponential for shape=1)
    nll -= log(1.0-pweibull(time(i),shape,scale));
    break;
  case 2: //Log normal
    nll -= log(1.0-pnorm_approx((log(time(i))-log(rate(i)))/shape));
    break;
}
```

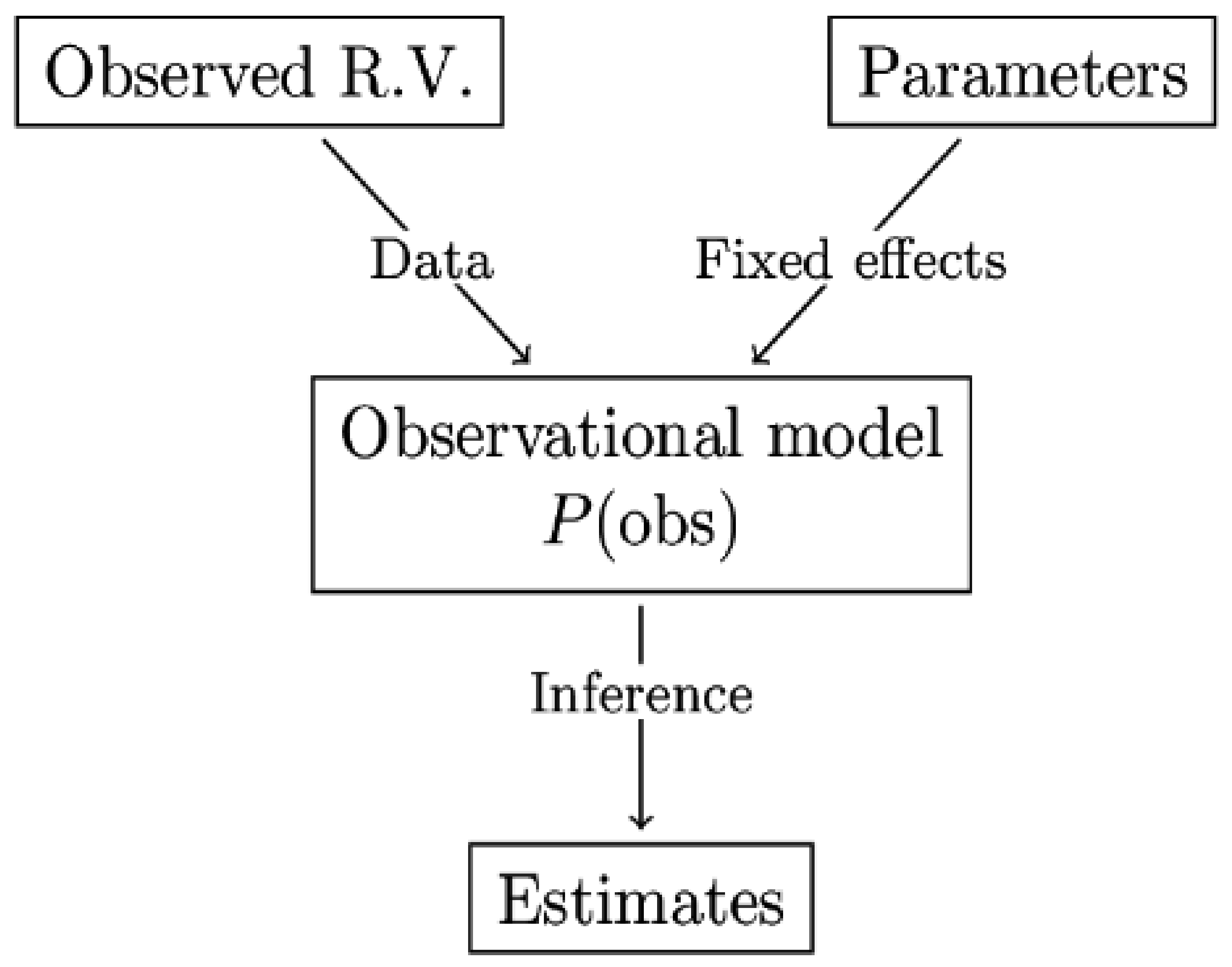
Acute leukemia

- Black: Kaplan-Meier
- Red: Weibull
- Blue: Exponential
- Purple: log-normal

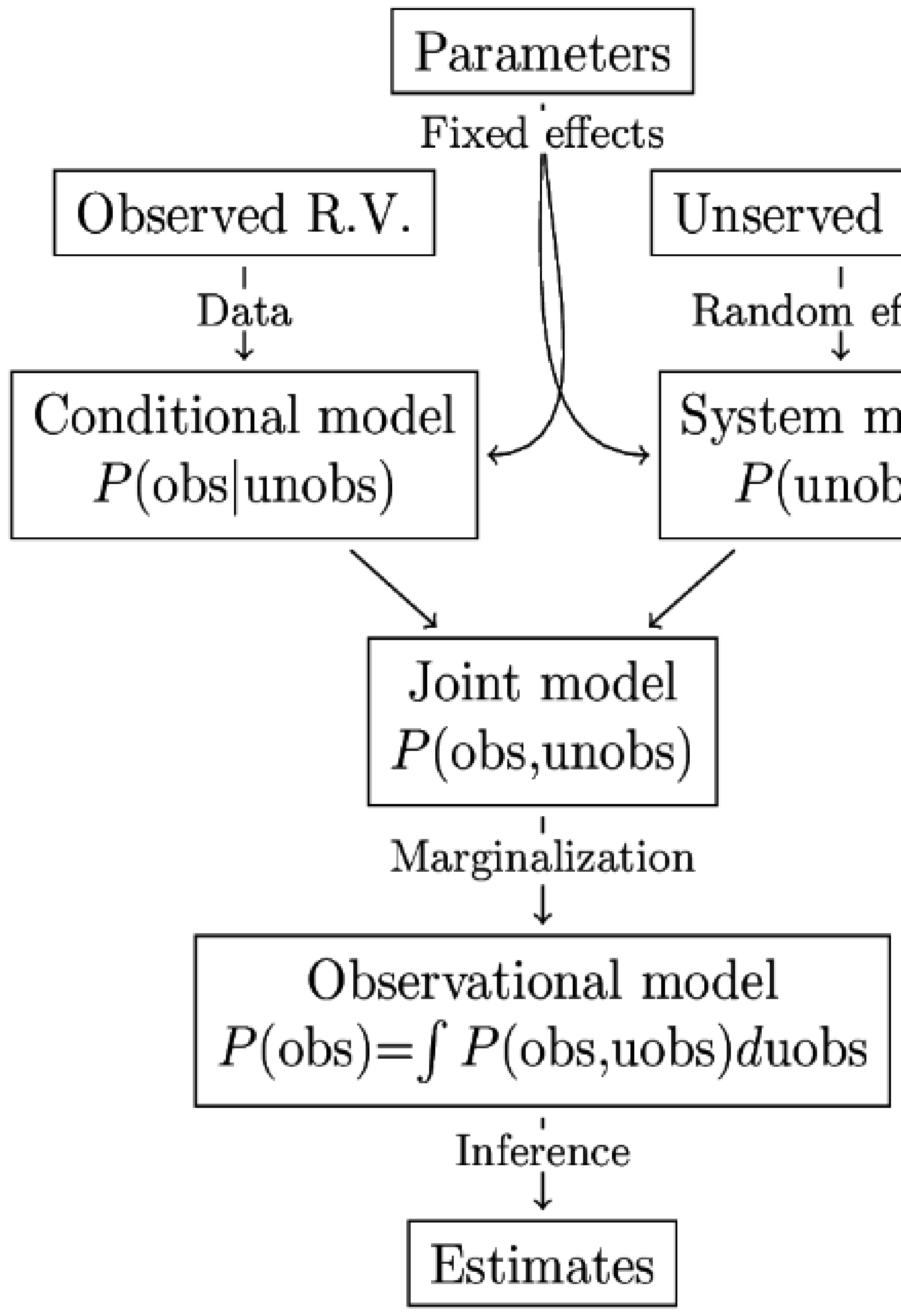


Hierachical models

Fixed effects model

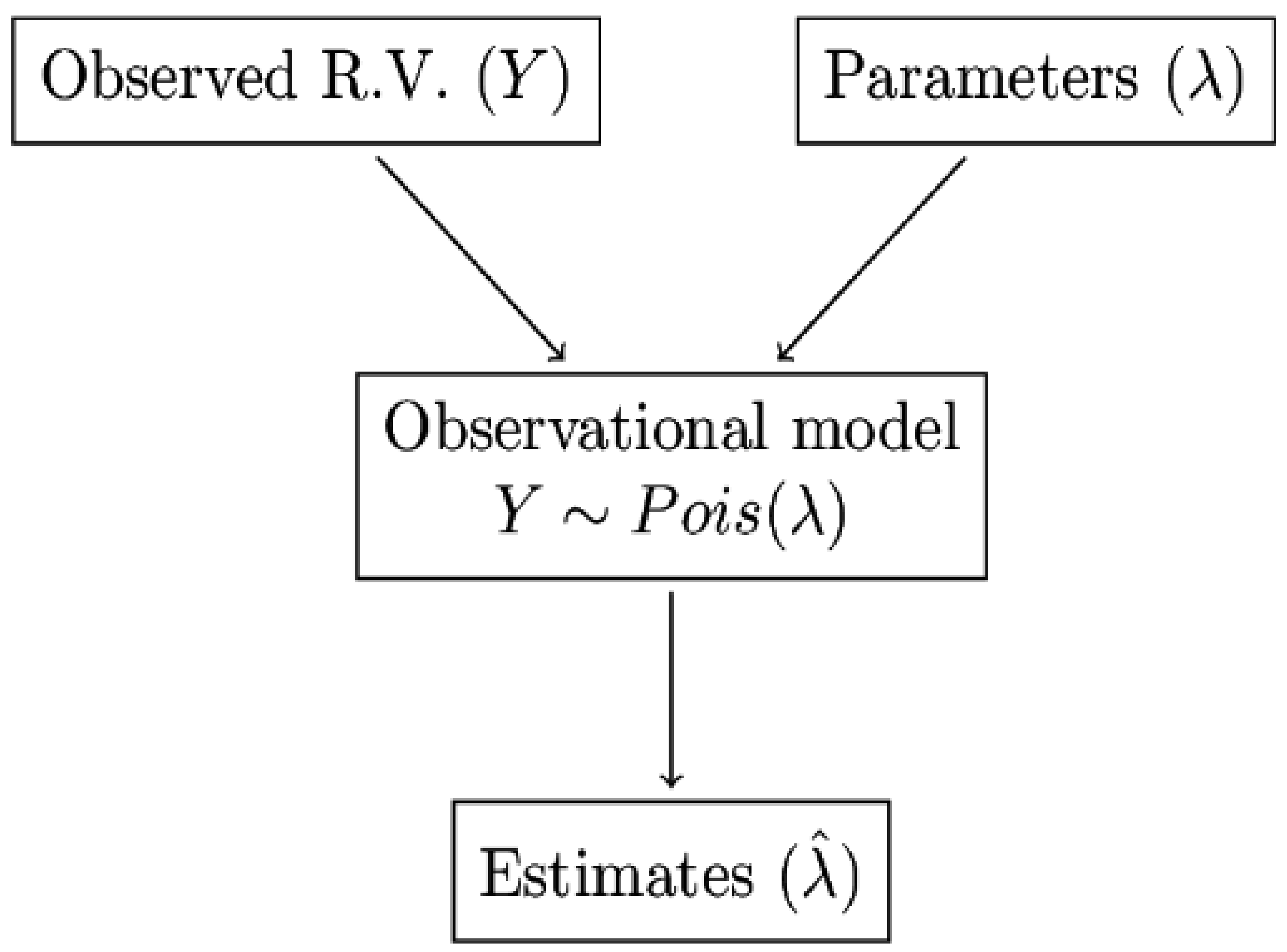


Random effects model



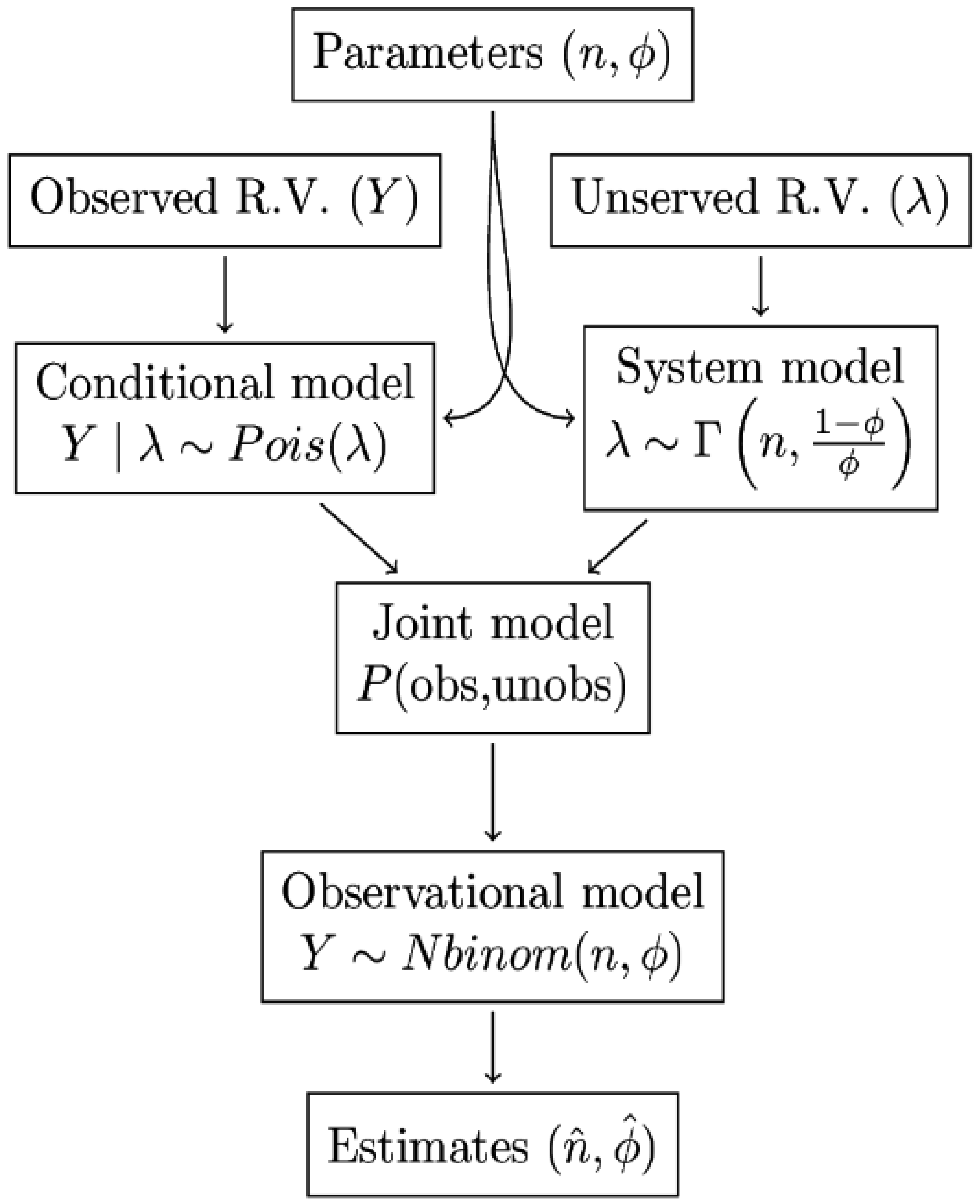
Example: Poisson model

Poisson model



Observed R.V. (Y)
Parameters (λ)
Observational model $Y \sim Pois(\lambda)$
Estimates $(\hat{\lambda})$

Poisson model with overdispersion



Parameters (n, ϕ)
Observed R.V. (Y)
Unserved R.V. (λ)
Conditional model $Y \mid \lambda \sim Pois(\lambda)$
System model $\lambda \sim \Gamma\left(n, \frac{1-\phi}{\phi}\right)$
Joint model P(obs,unobs)
Observational model $Y \sim Nbinom(n, \phi)$
Estimates $(\hat{n}, \hat{\phi})$

Laplace approximation

- We need to calculate

$$L_M(\theta; y) = \int L(\theta; u, y) du$$

θ: Parameters; y: Observed; u: unobserved

- Approximate $\ell(\theta; u, y) = \log L(\theta; u, y)$ by
- 2nd order Taylor around $\hat{u}_\theta = \operatorname{argmax}_u \ell(\theta; u, y)$

$$\ell(\theta; u, y) \approx \ell(\theta; u, y) - \frac{1}{2}(u - \hat{u}_\theta)^t (-\ell''_{uu}(\theta; u, y) \mid_{u=\hat{u}_\theta})(u - \hat{u}_\theta)$$

Laplace approximation

Now

$$\begin{aligned} L_M(\theta;y) &= \int L(\theta;u,y)du \\ &\approx \int \exp\left(\ell(\theta;u,y) - \frac{1}{2}(u-\hat{u}_\theta)^t(-\ell''_{uu}(\theta;u,Y)\mid_{u=\hat{u}_\theta})(u-\hat{u}_\theta)\right)du \\ &= L(\theta;u,y)\int \exp\left(-\frac{1}{2}(u-\hat{u}_\theta)^t(-\ell''_{uu}(\theta;u,y)\mid_{u=\hat{u}_\theta})(u-\hat{u}_\theta)\right)du \\ &= L(\theta;u,y)\cdot(2\pi)^{n/2}\cdot det(-\ell''_{uu}(\theta;u,Y)\mid_{u=\hat{u}_\theta})^{-1/2} \end{aligned}$$

Taking the logaritm:

$$\ell_M(\theta;y) = \ell(\theta;u,y) - \frac{1}{2}\log(det(-\ell''_{uu}(\theta;u,y)\mid_{u=\hat{u}_\theta})) + \frac{n}{2}\log(2\pi)$$

Rats cont'd

- We add a random intercept to the model

$$\begin{aligned} U_i &\sim \mathcal{N}(0, \nu^2) \\ X_{ij} &\sim \mathcal{N}(\alpha + U_i + \beta \cdot t_{ij}, \sigma^2) \end{aligned}$$

- i: Rat $1, \dots, 30$
- j: Observation $1, \dots, 5$ for rat i.
- Joint log-likelihood: $\ell(\theta, u, x) = \ell_{x|u}(x, \alpha, \beta, \sigma) + \ell_U(u, \nu)$

Rats cont'd

- $\ell_U(u, \nu)$ is implemented in R by:

```r
l.u <- function(u, s.u){
    sum(dnorm(u,mean=0,sd=s.u,log=TRUE))
}
```

- $\ell_{x|u}(x, \alpha, \beta, \sigma)$ is implemented by:

```r
l.x <- function(x,u,a,b,s){
    mu <- a+u[rat_data$id]+b*rat_data$week
    sum(dnorm(x,mean=mu,sd=s,log=TRUE))
}
```

- And the joint negative log-likelihood by:

```r
nl <- function(th,u,x){
    -l.x(x,u,th[1],th[2],exp(th[3]))-l.u(u,exp(th[4]))
}
```

Rats cont'd

- Now we can set up the Laplace approximation

```
library(numDeriv)
l.LA <- function(th){
    u.init <- rep(0,max(rat_data$id))
    obj <- function(u)nl(th,u,rat_data$weight)
    est <- nlminb(u.init,obj)
    lval <- est$objective
    u <- est$par
    H <- hessian(obj,u)
    lval+0.5*log(det(H))-0.5*length(u)*log(2*pi)
}
```

```
sp <- c(150,50,log(8),log(15))
system.time(fit <- optim(sp,l.LA,method="L-BFGS-B"))
```

```
##    user  system elapsed
## 109.032   0.402 109.460
```

Rats cont'd

```r
fit
```

```
## $par
## [1] 156.053  43.300   2.100   2.624
##
## $value
## [1] 568.8
##
## $counts
## function gradient
##       31       31
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

Laplace approximation

0. Initialize θ to some value θ_0

1. With curren**c**t value for θ optimize joint likelihood w.r.t u to get $\hat{u}_\theta$ and Hessian $H(\hat{u}_\theta)$

2. Use $\hat{u}_\theta$ and $H(\hat{u}_\theta)$ to approximate $\ell_M(\theta)$

3. Compute value and gradient of $\ell_M(\theta)$

4. If gradient is not (close to) 0 set θ to different value and go to 1.

- TMB handles step 1-4 internally and calculates gradients of the marginal likelihood
- We only supply the joint negative log-likelihood, starting values, and the names of latent variables.
- Unlike ADMB, TMB can automatically detect seperability of random effects to reduce computations

Laplace approximation

- If random effects are non-Gaussian (cdf F) the Laplace approximation can be inaccurate.
- Solution:
  - Use Gaussian random effects (U)
  - Transform the variables by inverse transform

$$X = F^{-1}(\Phi(U))$$

REML estimation

- Remember: Maximum likelihood estimates of variance parameters can be downwards biased.
- Let $L(\beta, \alpha)$ be the likelihood function
- Define: $\bar{L}(\beta) = \int L(\beta, \alpha) d\alpha$
- The REML (REstricted/REduced/REsidual Maximum Likelihood) method is:
  - Estimate β by: $\bar{\beta} = \text{argmax}_\beta \bar{L}(\beta)$
  - Estimate α by: $\bar{\alpha} = \text{argmax}_\alpha L(\bar{\beta}, \alpha)$
- Exactly the same work flow as for the Laplace approximation

REML estimation

- We compile and load the normal distribution from before and simulate data

```
library(TMB)
compile("norm.cpp")
dyn.load(dyn.lib("norm"))
x<-rnorm(100,3,2)
```

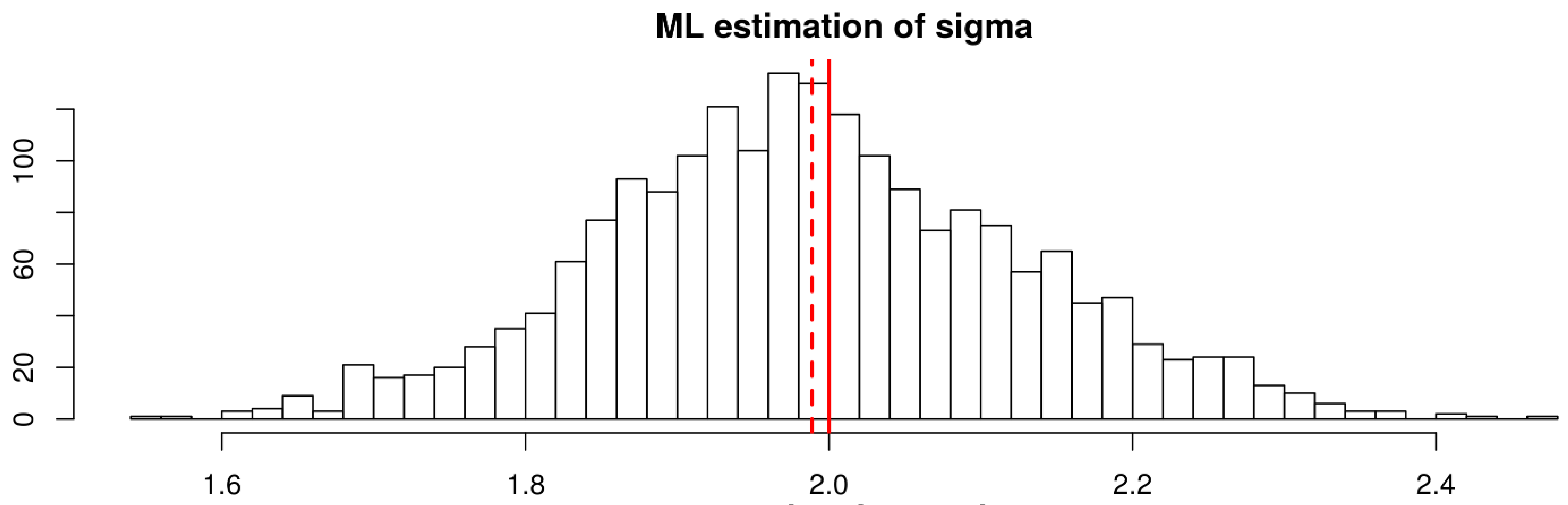
- Maximum likelihood estimation

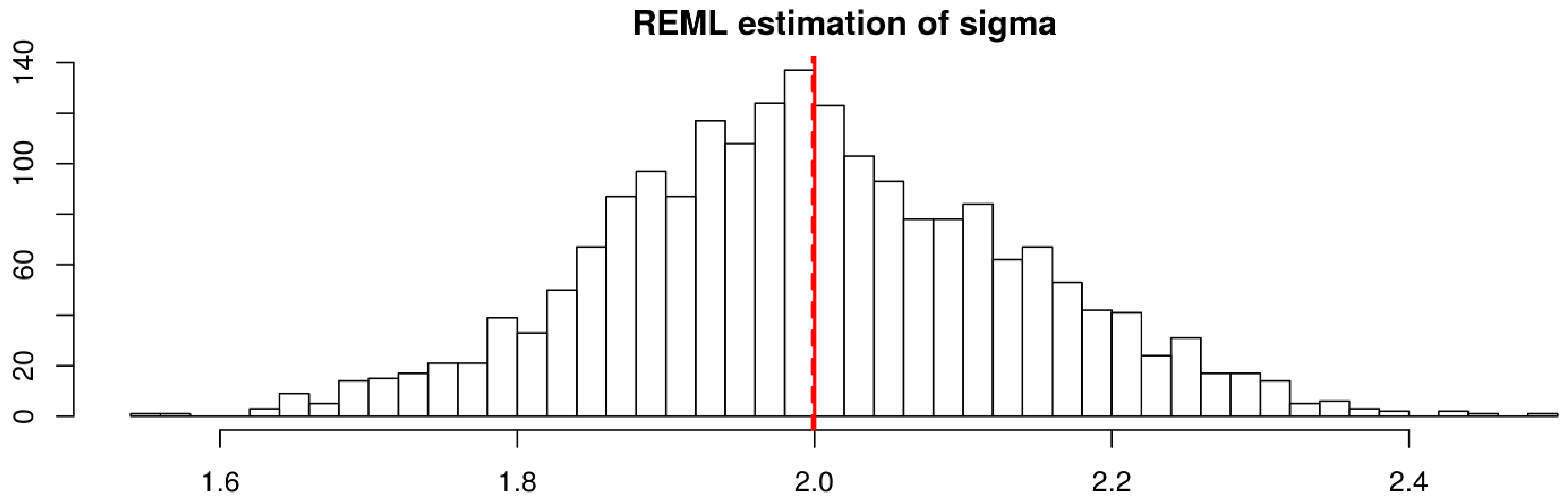
```
f_ml <- MakeADFun(data=list(x=x),parameters=list(mu=0,sigma=1))
```

- REML estimation

```
f_re <- MakeADFun(data=list(x=x),parameters=list(mu=0,sigma=1),
                  random="mu")
```

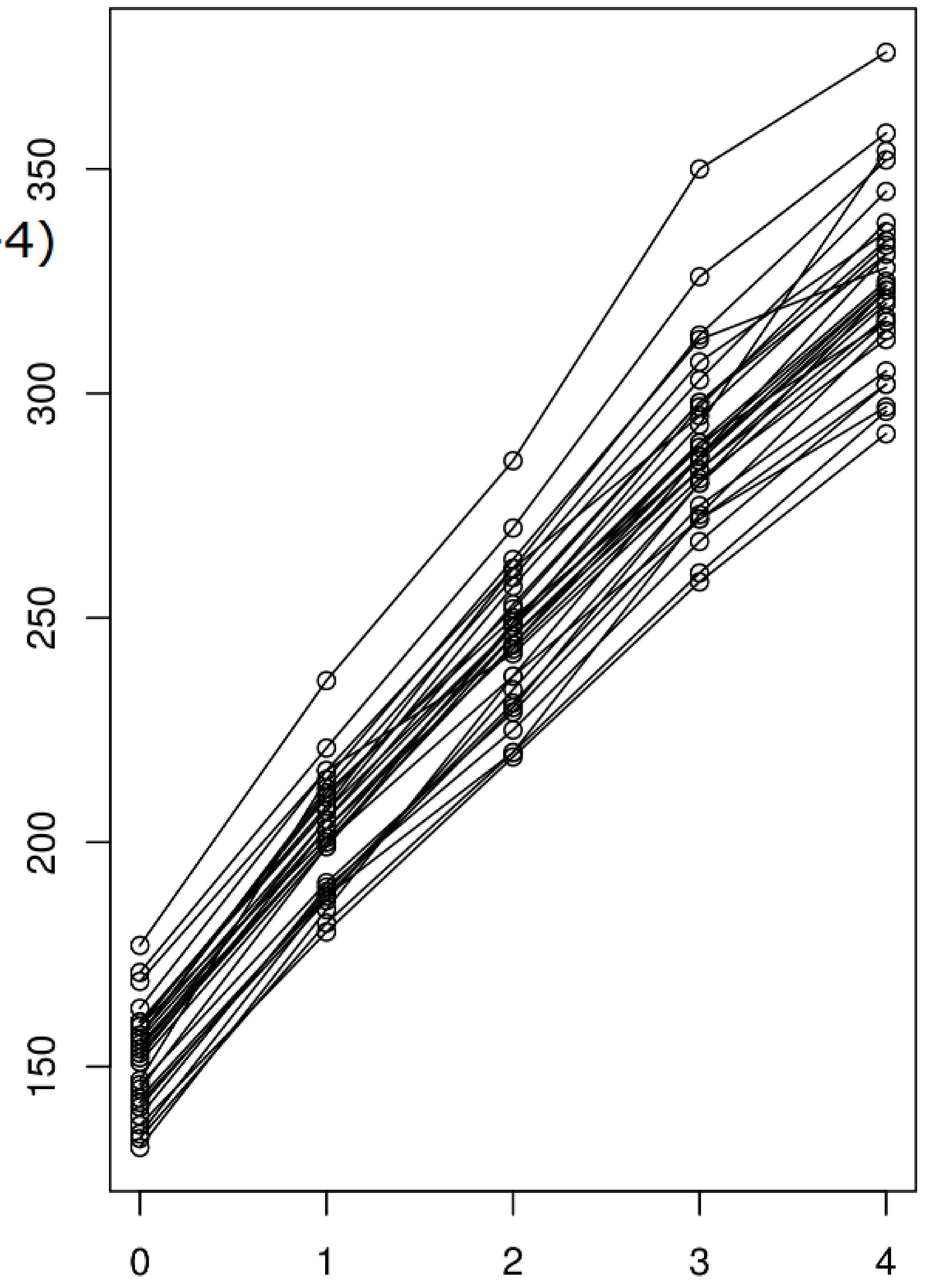
REML estimation





Rats cont'd

- Setup:
  - 30 newborn rats weighed weekly (week 0-4)
- Model:
  - $X_i \sim \mathcal{N}(\alpha + Y_i + (\beta + Z_i) \cdot t_i, \sigma^2)$
  - $(Y_i, Z_i) \sim \mathcal{N}_2(0, \Sigma)$
  - X_i: weight of observation $i = 1, \ldots, 150$
  - t_i: week at observation $i = 1, \ldots, 150$
- Estimation in R:
  - lmer(weight ~ week+(week|id), data = rat_dat)



Rats cont'd

```
#include <TMB.hpp>
using namespace density;

template<class Type>
Type objective_function<Type>::operator() ()
{
  DATA_VECTOR(weight);
  DATA_MATRIX(modMat);
  DATA_MATRIX(reInclMat);
  PARAMETER_VECTOR(beta);
  PARAMETER(logSd);
  PARAMETER(logSdY);
  PARAMETER(logSdZ);
  PARAMETER(logitRho);
  PARAMETER_MATRIX(YZ);
```

Rats cont'd

```
Type nll = 0.0;
Type rho = 2.0/(1.0+exp(-logitRho))-1;

matrix<Type> cov(2,2);
cov(0,0) = exp(2.0*logSdY);
cov(1,1) = exp(2.0*logSdZ);
cov(1,0) = rho*exp(logSdY+logSdZ);
cov(0,1) = cov(1,0);

MVNORM_t<Type> renll(cov);
vector<Type> tmp(YZ.cols());

for(int i = 0; i < YZ.rows(); ++i){
  tmp = YZ.row(i);
  nll += renll(tmp);
}
```

Rats cont'd

```
  matrix<Type> muretmp = reInclMat * YZ;
  muretmp = (muretmp.array() * modMat.array()).matrix();
  vector<Type> mure = muretmp.rowwise().sum();

  Type sd = exp(logSd);
  vector<Type> mu = modMat * beta;

  nll -= sum(dnorm(weight, mu+mure,sd,true));

  return nll;
}
```

Rats cont'd

- We create the R object with random="YZ" to invoke the Laplace approximation

```
obj1<-MakeADFun(data,parameters,random="YZ",DLL="rats_random")
```

```
tmbtime <- system.time(opt1<-nlminb(obj1$par,obj1$fn,obj1$gr))
```

```
tmbtime
```

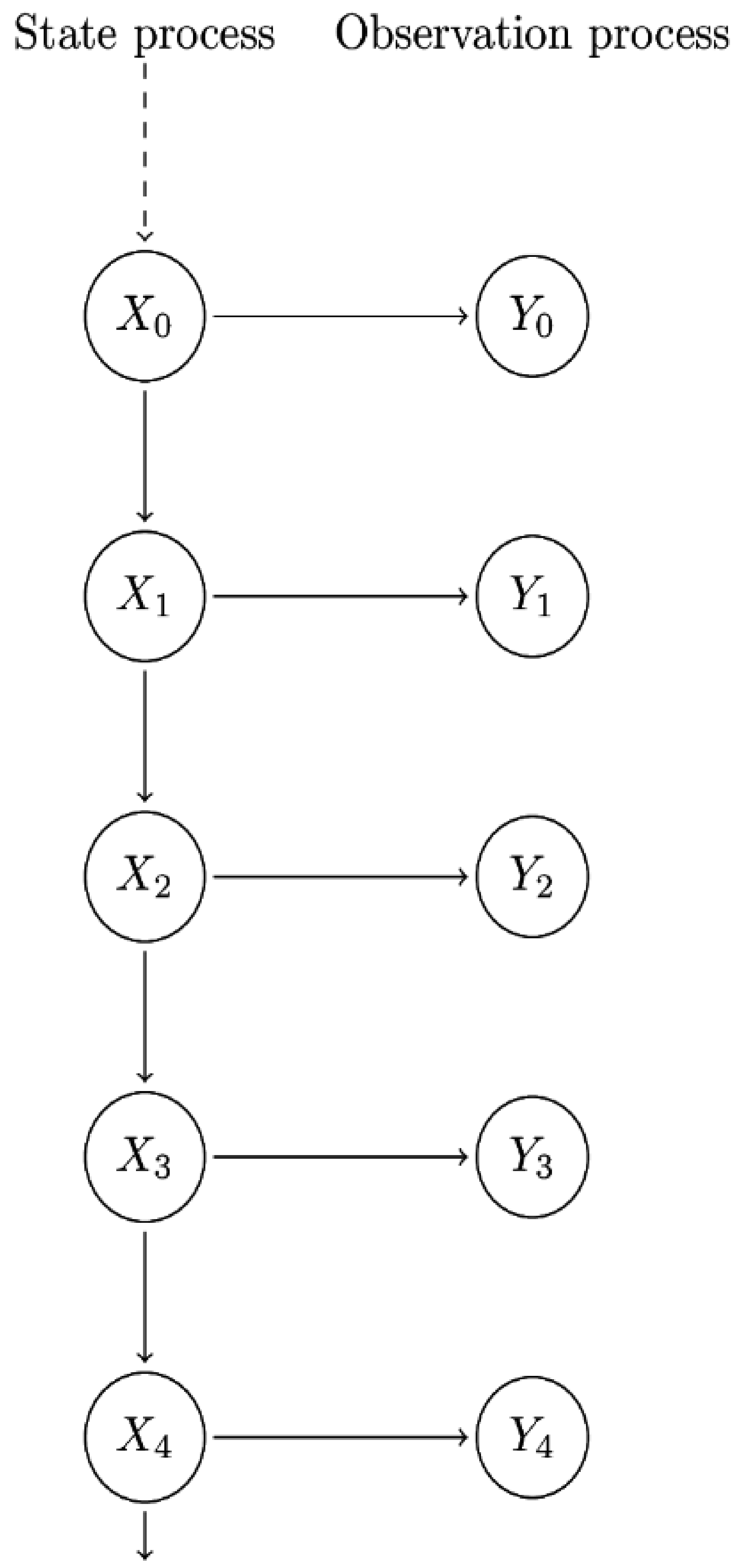
```
##    user  system elapsed
##   0.304   0.002   0.307
```

```
opt1$par
```

```
##     beta     beta    logSd   logSdY   logSdZ logitRho
## 156.0533  43.3000   1.7942   2.3658   1.2518   0.4614
```

State-space models

- Hierachical time series model
- System (or State) model for $P(X_t \mid X_0, \dots, X_{t-1})$
- Typically $= P(X_t \mid X_{t-1})$
- $P(X_t \mid X_1, \dots X_{t-1})$ is a continuous distribution
- Conditional model for observations: $P(Y_t | X_t)$
- The Laplace approximation can be used.



Geolocation

- Argos data for subadult ringed seal
- Extension of Jonsen, Flemming, and Myers (2005) (BSAM) and Johnson et al. (2008) (CRAWL)
- Need fast and robust estimation
- States - velocity:

$$\nu_c(t_{i+1}) - \gamma_c = e^{-\beta_c \Delta_i} (\nu_c(t_i) - \gamma_c) + \eta_{ci},$$

- States - position

$$\mu_c(t_{i+1}) = \mu_c(t_i) + \nu_c(t_i) \left(1 - e^{-\beta_c \Delta_i}\right) / \beta_c + \zeta_{ci}.$$

Geolocation

- Variance and covariance of error

$$
\begin{aligned}
V_c^{\eta}(t_i) &= \sigma_c^2(1 - e^{-2\beta_c\Delta_i})/(2\beta_c),\\
V_c^{\zeta}(t_i) &= \frac{\sigma_c^2}{\beta_c^2}\left(\Delta_i - 2\left(1 - e^{-\beta_c\Delta_i}\right)/\beta_c + \left(1 - e^{-2\beta_c\Delta_i}\right)/(2\beta_c)\right),\\
C_c(t_i) &= \frac{\sigma_c^2}{2\beta_c^2}\left(1 - 2e^{-\beta_c\Delta_i} + e^{-2\beta_c\Delta_i}\right),
\end{aligned}
$$

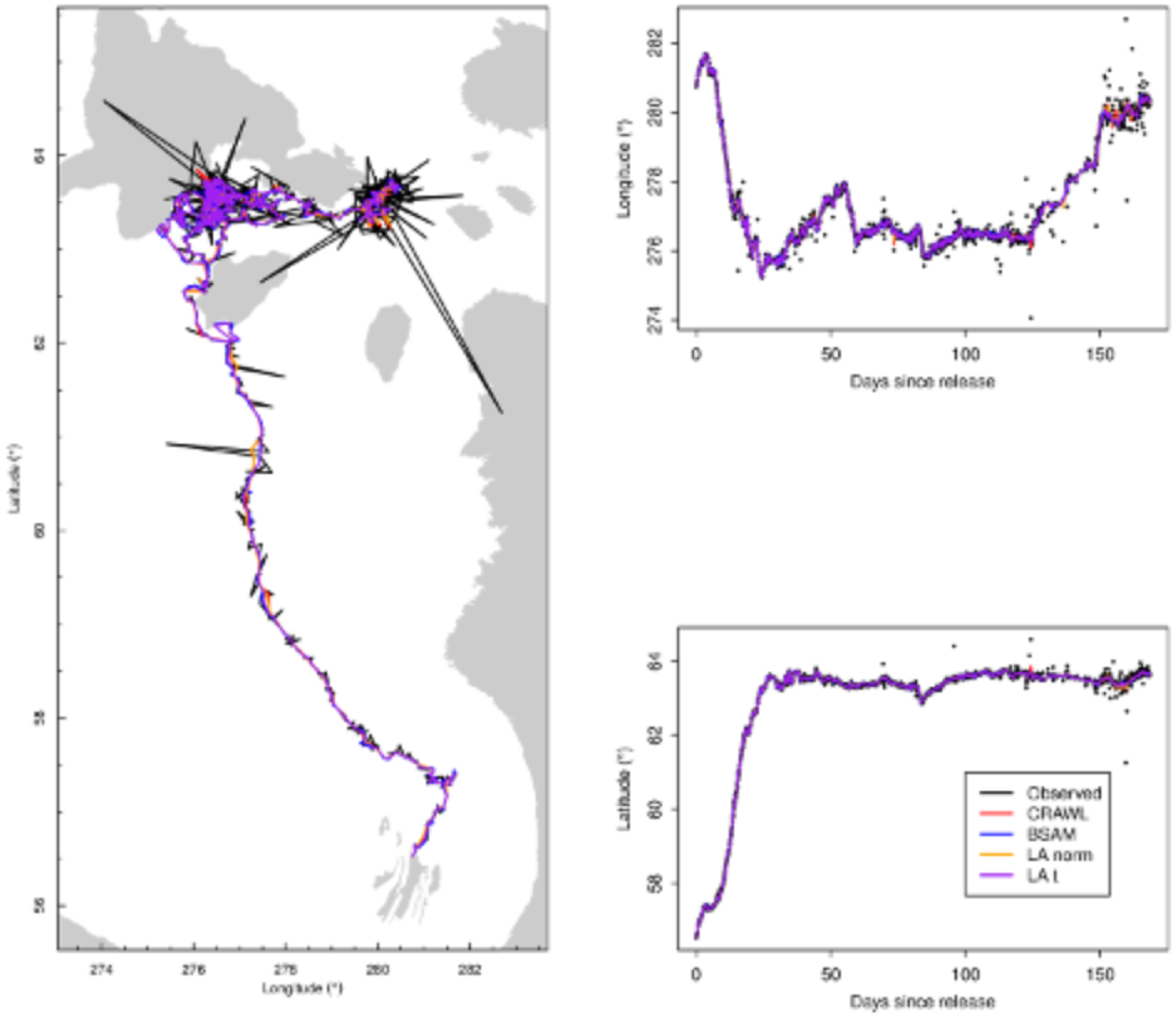
Geolocation

- Measurement equation

$$y_c(t_i) = \mu_c(t_i) + \epsilon_{ci}.$$

- ϵ_{ci} is modelled as bivariate t-distribution or bivariate normal. Scale/covariance matrix depends on Argos class

Geolocation



Geolocation

- Execution times in seconds for estimation procedures for the two tracks. All methods are run on a standard laptop.

Method	Subadult	Adult	Parameters	Latent variables (subadult;adult)
CRAWL	12.687	8.270	4	13680;10404
BSAM	11109.762	11745.828	7	2698;3500
LA-n	30.339	24.895	17	13680;10404
LA-t	47.866	37.972	17	13680;10404

Stock Assessment Models

- Example from the TMB github page
- Age-based state-space fish stock assessment model
- By Nielsen and Berg (2014)
- States: Numbers-at-age and fishing mortality
- Observations: Catch-at-age (commercial and survey)
- Originally in ADMB (http://www.stockassessment.org) - 3.1 min to estimate
- In TMB: 3.1 sec to estimate

Summary

- TMB is a flexible tool that can be used for a lot of models
- TMB is very powerfull for difficult models
- Especially with random effects
- Easy integration with R
- Parameters must be continuous
- Examples and slides are available at http://staff.dtu.dk/cmoe/tmb-dal-2014

References

Bolker, B.M., B. Gardner, M. Maunder, Casper Willestofte Berg, M. Brooks, L. Comita, E. Crone, et al. 2013. “Strategies for Fitting Nonlinear Ecological Models in R, AD Model Builder, and BUGS.” *Methods in Ecology and Evolution* 4 (6): 501–512. doi:10.1111/2041-210X.12044.

Freireich, Emil J, Edmund Gehan, Emil Frei, Leslie R. Schroeder, Irving J. Wolman, Rachad Anbari, E. Omar Burgert, et al. 1963. “The Effect of 6-Mercaptopurine on the Duration of Steroid-Induced Remissions in Acute Leukemia: A Model for Evaluation of Other Potentially Useful Therapy.” Edited by *Blood* 21 (6): 699–716.

Gelfland, Alan E., and Susan E. Hills. 1990. “Illustration of Bayesian Inference in Normal Data Models Using Gibbs Sampling.” *Journal of the American Statistical Association* 85 (412): 972.

Hoef, Jay M. Ver, and Peter L. Boveng. 2007. “Quasi-Poisson Vs. Negative Binomial Regression: How Should We Model Overdispersed Count Data?” *ECOLOGY* 88 (11): 2766–2772. doi:10.1890/07-0043.1.

Johnson, Devin S., Joshua M. London, Mary-Anne Lea, and John W. Durban. 2008. “Continuous-Time Correlated Random Walk Model for Animal Telemetry Data.” *Ecology* 89 (5): 1208–1215. doi:10.1890/07-1032.1.

Jonsen, ID, JM Flemming, and RA Myers. 2005. “Robust State-Space Modeling of Animal Movement Data.” *Ecology* 86 (11): 2874–2880.

Kristensen, Kasper. 2013. “New AD Software.” http://www.admb-project.org/developers/workshop/reykjavik-2013/TMB.pdf/view.

Magnusson, Arni. 2013. “AD Model Builder - Brief Introduction.” http://www.admb-project.org/documentation/intro/slides-arni/.

Wikipedia. 2014. “ADMB.” http://en.wikipedia.org/wiki/ADMB.