

Introduction

Anders Nielsen, Ethan Lawler, Sean Anderson

an@aqua.dtu.dk

Welcome!

- Please introduce yourselves
 - Name and Organization
 - Experience (TMB, ADMB, R/C coding)
 - What do you hope to learn
 - Special requests

Non-standard models

- Models where you need to write your own likelihood
- Models you cannot write in one line in R
- non-trivial non-linearities
- complex covariance structures
- complicated couplings between fixed and random effects
- different sources of observations needing different likelihood types
- Standard models are very useful, but should not limit us

Formula interfaces are sometimes frustrating ...

- A useful model for longitudinal data:

$$\begin{aligned} \mathbf{lnc} &\sim N(\boldsymbol{\mu}, \mathbf{V}), \text{ where} \\ \mu_i &= \mu + \alpha(\text{treat}_i) + \beta(\text{month}_i) + \gamma(\text{treat}_i, \text{month}_i), \text{ and} \\ V_{i_1, i_2} &= \begin{cases} 0 & , \text{ if } \text{cage}_{i_1} \neq \text{cage}_{i_2} \text{ and } i_1 \neq i_2 \\ \nu^2 + \tau^2 \exp \left\{ \frac{-(\text{month}_{i_1} - \text{month}_{i_2})^2}{\rho^2} \right\} & , \text{ if } \text{cage}_{i_1} = \text{cage}_{i_2} \text{ and } i_1 \neq i_2 \\ \nu^2 + \tau^2 + \sigma^2 & , \text{ if } i_1 = i_2 \end{cases} \end{aligned}$$

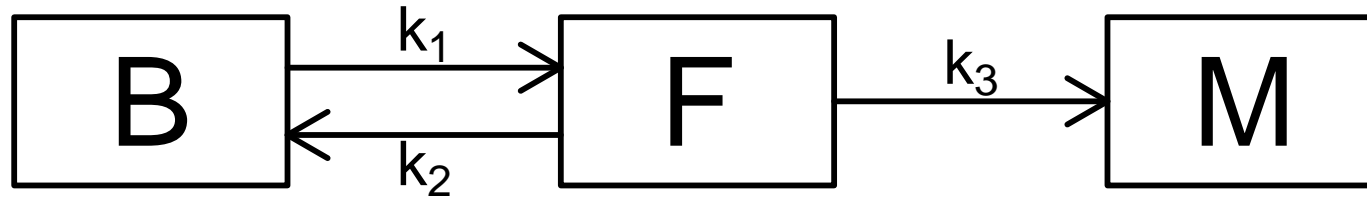
- This model is implemented by:

```
fit.gau <- lme(lnc~month+treatm+month:treatm,  
             random=~1|cage,  
             correlation=corGaus(form=~as.numeric(month)|cage,nugget=TRUE),  
             data=rats)
```

- So many pitfalls and much is hidden. Even difficult to recover model parameters.
- What is τ ? Some hours with manual, but re-implement to be sure...
- Restricted by what someone else has put in there. Giant task to move beyond.

Terbuthylazine

- It is a herbicide
- Free terbuthylazine can be washed into the drinking water
- It can be bound to the soil
- Certain bacteria can mineralize it



$$\frac{dB_t}{dt} = -k_1 B_t + k_2 F_t,$$

$$B_0 = 0$$

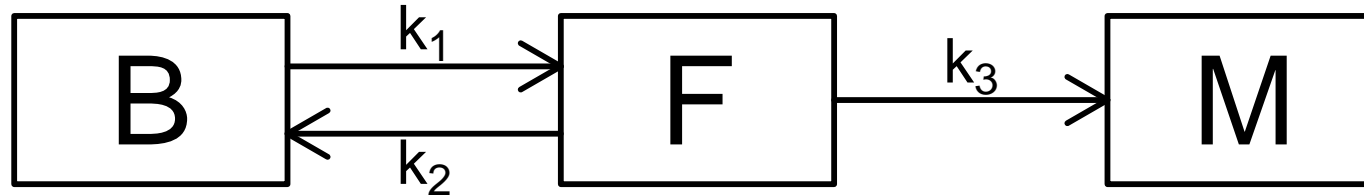
$$\frac{dF_t}{dt} = k_1 B_t - (k_2 + k_3) F_t,$$

$$F_0 = 100$$

$$\frac{dM_t}{dt} = k_3 F_t,$$

$$M_0 = 0$$

Simplifying



- The system is closed, so $M_t = 100 - B_t - F_t$
- Define $X_t = \begin{pmatrix} B_t \\ F_t \end{pmatrix}$
- The simplified system is:

$$\frac{dX_t}{dt} = \underbrace{\begin{pmatrix} -k_1 & k_2 \\ k_1 & -(k_2 + k_3) \end{pmatrix}}_A X_t, \quad X_0 = \begin{pmatrix} 0 \\ 100 \end{pmatrix}$$

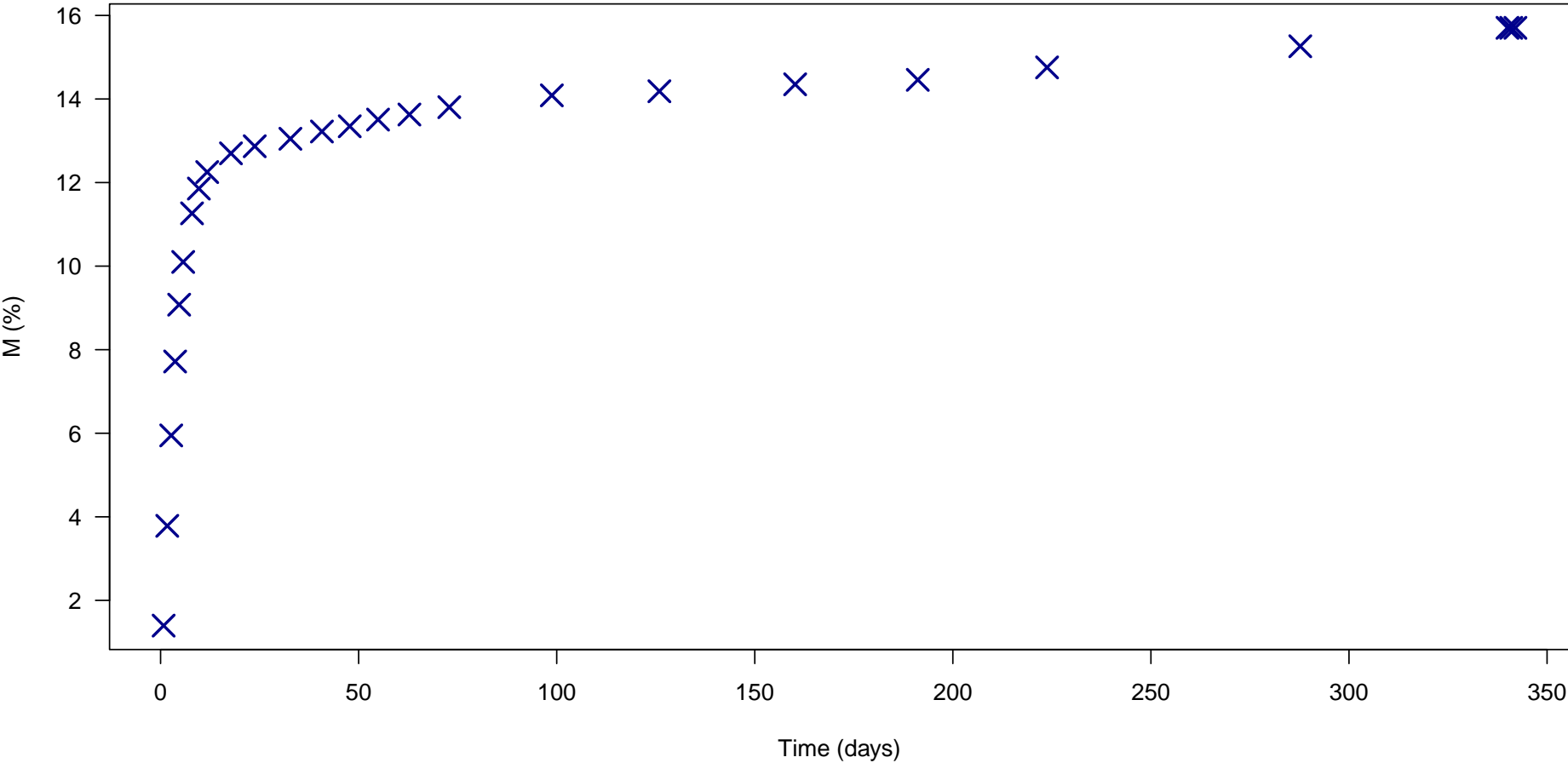
- The system is linear, so it can be solved for instance via the matrix exponential

$$X_t = e^{At} X_0$$

Observations

- The amount of mineralized terbuthylazine was measured 26 times throughout a year

Time	M
0.77	1.396
1.69	3.784
2.69	5.948
3.67	7.717
4.69	9.077
5.71	10.100
7.94	11.263
9.67	11.856
11.77	12.251
17.77	12.699
23.77	12.869
32.77	13.048
40.73	13.222
47.75	13.347
54.90	13.507
62.81	13.628
72.88	13.804
98.77	14.087
125.92	14.185
160.19	14.351
191.15	14.458
223.78	14.756
287.70	15.262
340.01	15.703
340.95	15.703
342.01	15.703



Simplest statistical model

- The simplest model we can think of would be:

$$M_{t_i} \sim \mathcal{N}\left(100 - \sum X_{t_i}, \sigma^2\right), \quad \text{independent, and with } X_{t_i} = e^{At_i} X_0.$$

```
> 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))
+ }
> system.time(fit <- optim(c(-2, -2, -2, -2), nlogL, hessian = TRUE))

   user  system elapsed
19.409   0.004  19.428

> fit$value

[1] 19.26905

> fit$convergence

[1] 0
```


- Try some of the different minimizers in R

```
> library(optimx)
> fit<-optimx(c(-2,-2,-2,-2),nlogL,hessian=TRUE,control=list(all.methods=TRUE))
> fit
```

fvalues	method	fns	grs	conv	KKT1	KKT2	xtimes
153.3056	bobyqa	144	NA	0	TRUE	FALSE	9.629
102.7661	Rcgmin	85	50	0	TRUE	FALSE	22.314
102.7660	nlm	NA	NA	0	TRUE	FALSE	11.865
102.7660	BFGS	79	18	0	TRUE	FALSE	15.005
102.7660	Rvmmin	81	15	0	TRUE	FALSE	10.517
102.7660	CG	567	101	1	TRUE	FALSE	91.569
91.17466	newuoa	696	NA	0	TRUE	FALSE	46.063
19.26905	Nelder-Mead	223	NA	0	FALSE	FALSE	14.837
0.9392184	ucminf	40	40	0	FALSE	TRUE	14.953
0.9392142	spg	198	NA	0	FALSE	TRUE	55.732
0.9392142	L-BFGS-B	85	85	0	FALSE	TRUE	50.807
0.9392142	nlminb	33	128	0	TRUE	TRUE	10.729

- Difficult because it is non-linear
- Would possibly be helped by accurate gradient info
- Runs in a fraction of a second in RTMB (later)
- Notice this is a miniature example with only 4 parameters

What is needed to handle a non-standard problem

- A purely parametric assessment model has more than 100 model parameters and it is non-linear
- Code up the negative log likelihood function
- A good function minimizer

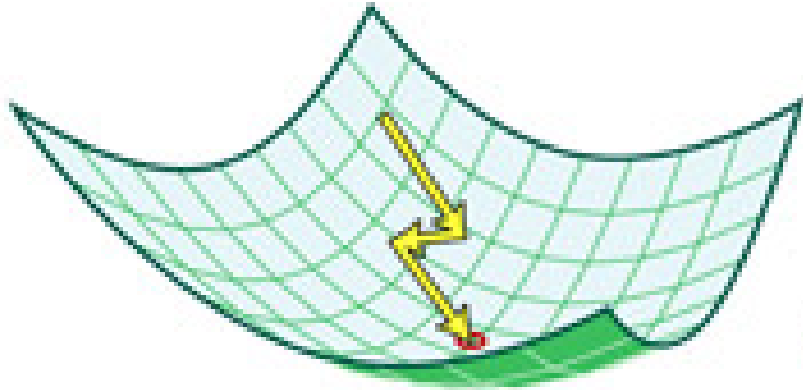
AD aided minimizer

- Want to minimize the negative log likelihood w.r.t. $\theta = (\theta_1, \dots, \theta_n)$

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \ell(y|\theta)$$

- If the dimension of θ is low (say n less than 5) any method can be used (grid search, random search, finite difference approximations, ...)
- We would like to be able to handle **much larger** problems
- Important for fixed effects models, and even more for random effects models
- A **quasi-Newton minimizer** aided by **automatic differentiation**

Quasi-Newton minimizer



ADMB

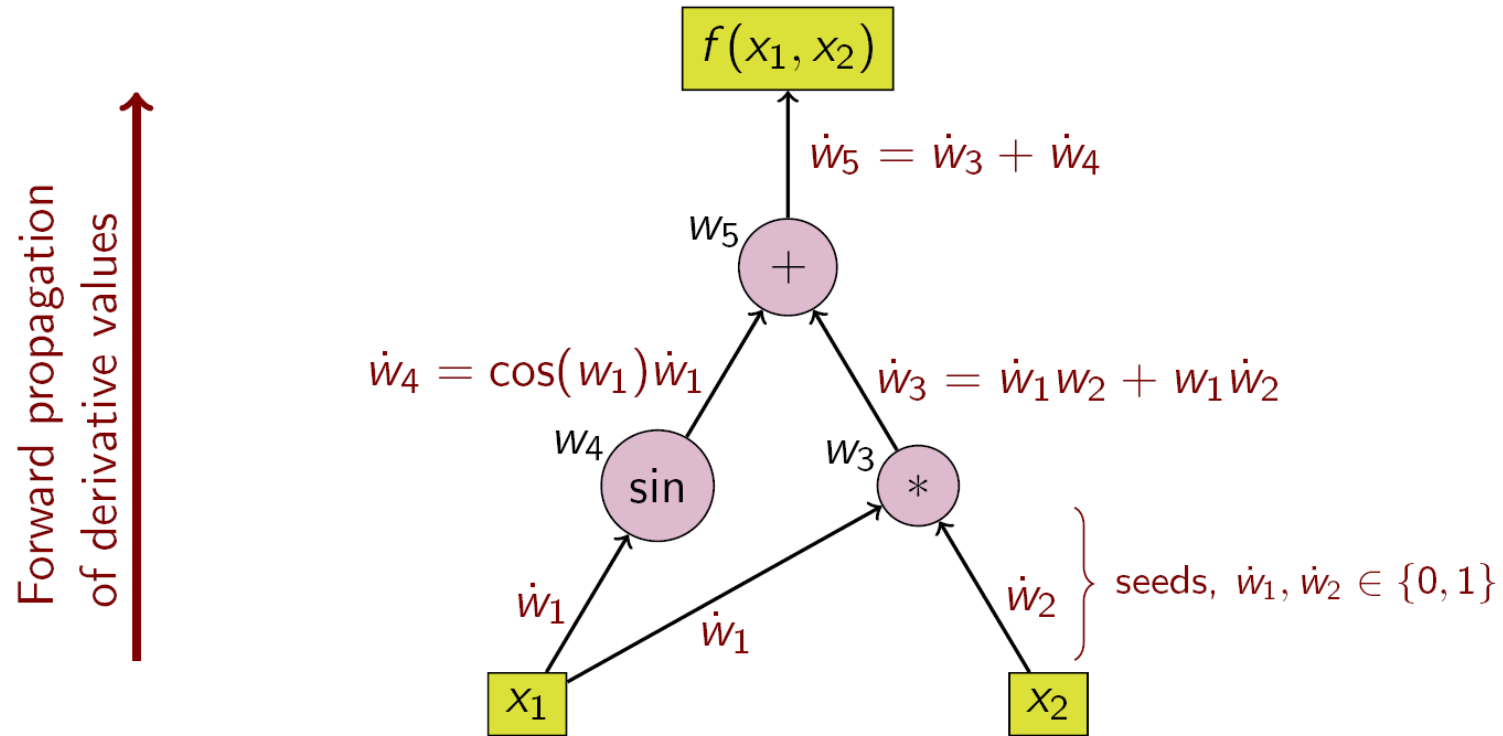
Automatic Differentiation Model Builder

- A **Newton** minimizer is an iterative algorithm
- Each step assumes that the function $\ell(x, \theta)$ can be approximated locally by a quadratic function
- It uses the first ℓ'_θ and second ℓ''_θ derivatives to find the minimum
- Instead of calculating ℓ''_θ at every step, a **quasi**-Newton minimizer uses successive first derivatives ℓ'_θ to approximate ℓ''_θ
- So a fast and accurate way to calculate ℓ'_θ is needed

Automatic Differentiation

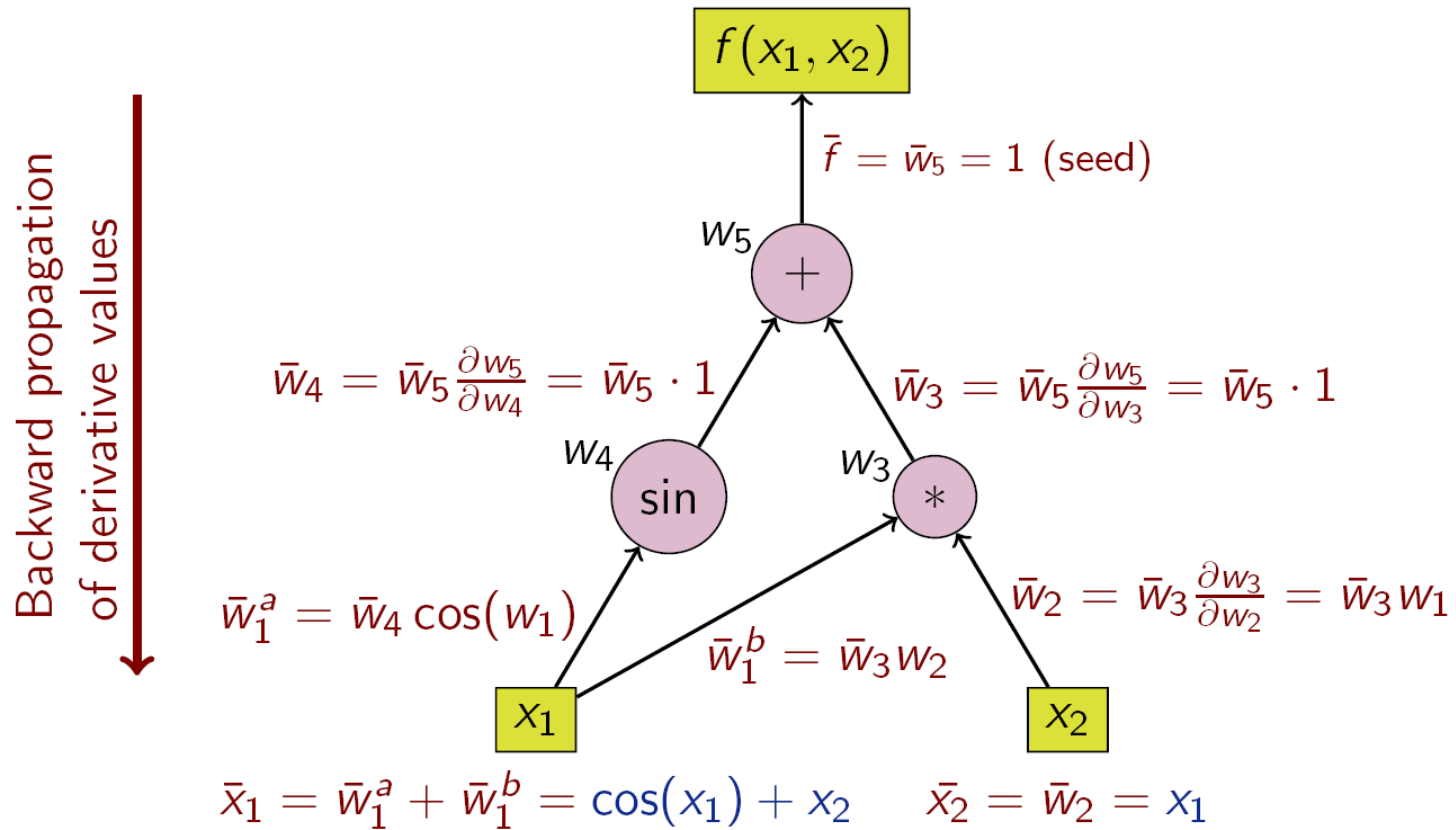
- We need to write a program to compute $\ell(\theta, x)$ anyway
- A computer program is a long list of simple operations:
'+', '-', '*', '/', 'exp', 'log', 'sin', 'cos', 'tan', 'sqrt', and so on
- We know how to derive each of these operations
- The chain rule tells us how to combine: $(f(g(x)))' = f'(g(x))g'(x)$
- So if the computer is instructed to:
 - keep track of all the simple operations used when calculating $\ell(\theta, x)$
 - use the simple derivative formulas and the chain rule
- Then once $\ell(\theta, x)$ is computed, we also have ℓ'_θ with a minimum of extra calculations
- This is fast and accurate, and the difficult part is built into TMB(!)
- Alternatives:
 - Finite difference: $(\ell'_\theta)_i \approx \frac{\ell(\theta_i + \Delta\theta_i, x) - \ell(\theta, x)}{\Delta\theta_i}$ Simple, but slow and inaccurate
 - Analytical: Excellent option, but difficult in larger models

Forward and reverse mode



(Image from Wikipedia)

- Forward mode is easy to understand and implement
- Not efficient when θ is high dimensional



(Image from Wikipedia)

- Requires recording a stack of all operations
- Efficient in number of operations ($C(\ell'_\theta) < 4C(\ell)$ ^a)
- TMB uses reverse mode
- Except for random effects models where a combo of forward and reverse mode is used

^aGriewank, A., 2000. Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. SIAM, Philadelphia, PA.

Template Model Builder (TMB):

- Developed by Kasper Kristensen (DTU-Aqua)
- ADMB inspired R-package
- Combines external libraries: CppAD, Eigen, CHOLMOD
- Continuously developed since 2009
- Implements Laplace approximation for random effects
- C++ Template based
- Automatic sparseness detection
- Parallelism through BLAS
- Parallel user templates
- Parallelism through `parallel` package

Example

- Assume that these 15 numbers follow a negative binomial distribution:

13 5 28 28 15 4 13 4 10 17 11 13 12 17 3

- The TMB code becomes

```
library(TMB)
compile("nbin.cpp")
dyn.load(dynlib("nbin"))

dat <- list()
dat$Y <- c(13, 5, 28, 28, 15, 4, 13, 4, 10, 17, 11, 13, 12, 17, 3)

par <- list()
par$logsize <- 0
par$logitp <- 0

obj <- MakeADFun(dat, par, DLL="nbin")
opt <- nlminb(obj$par, obj$fn, obj$gr)
summary(sdreport(obj))
```

nbin.R

```
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator() ()
{
    DATA_VECTOR(Y);
    PARAMETER(logsize);

    PARAMETER(logitp);
    Type size = exp(logsize);
    Type p=invlogit(logitp);
    Type nll = -sum(dnbinom(Y, size, p, true));
    ADREPORT(size);
    return nll;
}
```

nbin.cpp

Template Model Builder in R (RTMB):

- Developed by Kasper Kristensen (DTU-Aqua)
- ADMB inspired R-package
- Combines external libraries: CppAD, Eigen, CHOLMOD
- Continuously developed since 2009
- Implements Laplace approximation for random effects
- **Mode code written completely in R**
- Automatic sparseness detection
- Parallelism through BLAS
- Parallel user templates
- Parallelism through `parallel` package

Same example via RTMB

- Assume that these 15 numbers follow a negative binomial distribution:

13 5 28 28 15 4 13 4 10 17 11 13 12 17 3

- The RTMB code becomes

```
library(RTMB)
dat <- list()
dat$Y <- c(13, 5, 28, 28, 15, 4, 13, 4, 10, 17, 11, 13, 12, 17, 3)

par <- list()
par$logsize <- 0
par$logitp <- 0

nLogL <- function(par){
  -sum(dnbinom(dat$Y,exp(par$logsize),plogis(par$logitp),log=TRUE))
}

obj <- MakeADFun(nLogL, par)
opt <- nlminb(obj$par, obj$fn, obj$gr)
summary(sdreport(obj))
```



nbinR.R

Exercise 1:

- Installing RTMB on your computer via the instructions on (if you have not already):
<https://github.com/kaskr/RTMB#install>
- Try replicating the negative binomial example
- What are the derivatives w.r.t. the two model parameters at the initial parameter values?

Access to AD on custom functions

- To see and use the AD-engine we can do:

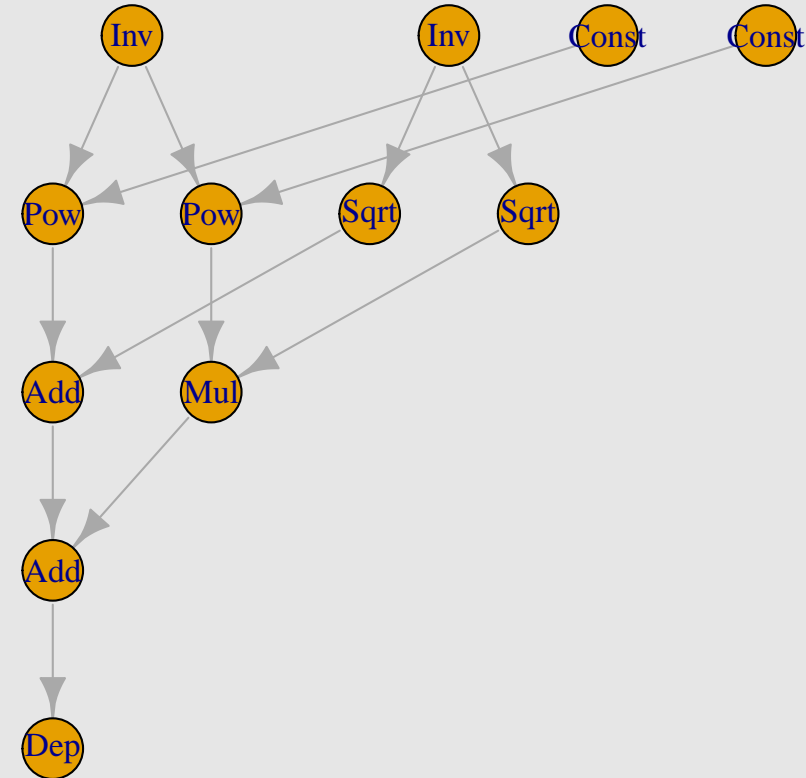
```
library(RTMB)

f<-function(x){
  x[1]^2+sqrt(x[2])+x[1]^2*sqrt(x[2])
}

F<-MakeTape(f, c(2,3))

F
F(c(1,2))
F$jacobian(c(1,2))
DF<-F$jacfun()
DF
DF$jacobian(c(1,2))
F$graph()

library(igraph)
G <- graph_from_adjacency_matrix(F$graph())
plot(G, vertex.size=17, layout=layout_as_tree)
```



adfun.R

Final comments:

- Fisheries research has inspired some tools which are useful for statisticians
- These tools have made the jump from standard to non-standard models smaller
- Writing own models give greater insights
- Read more in:
 - Fournier DA, HJ Skaug, J Ancheta, J Ianelli, A Magnusson, MN Maunder, A Nielsen, J Sibert 2012. AD Model Builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models. Optimization Methods and Software 27 (2), 233-249
 - Kristensen, K, A. Nielsen, C.W. Berg, H.J. Skaug, B. Bell 2016. TMB: Automatic differentiation and laplace approximation. Journal of Statistical Software 70 (5), 1-21
 - Nielsen, A. and C.W. Berg 2014. Estimation of time-varying selectivity in stock assessments using state-space models. Fisheries Research 158, 96-101
 - Thygesen, U.H., C.M. Albertsen, C.W. Berg, K. Kristensen, and A. Nielsen 2017. Validation of state space models fitted as mixed effects models. (Subm. EES).
 - <http://tmb-project.org>
 - <http://admb-project.org>
 - <https://github.com/fishfollower/SAM>

Appendix: RTMB code for the terbuthylazine example

```
library(RTMB)
par <- list(logk=c(0,0,0), logSigma=0)
dat <- read.table('min.dat', skip=3, head=FALSE)
nLogL <- function(par) {
  k <- exp(par$logk[1:3])
  sigma <- exp(par$logSigma)
  A <- matrix(c(-k[1], k[1], k[2], -(k[2] + k[3])),2,2);
  x0 <- c(0, 100)
  sol <- function(t) 100 - sum(expm(A*t)%*%x0)
  pred <- numeric(nrow(dat))
  for(i in 1:nrow(dat))pred[i] <- sol(dat[i,1])
  -sum(dnorm(dat[, 2], mean = pred, sd = sigma, log = TRUE))
}

obj <- MakeADFun(nLogL,par, silent=T)

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

min/min.R