

# Software tools for Maximum Likelihood Estimation:MB

## Lesson 2 - first RTMB & Derivatives

[Click here to view presentation online](#)

Jim Bence

1 December 2023

# Outline:

- Demos of using nlminb and RTMB to estimate parameters
- Explanation of what happened in RTMB
- Simple exercise adapting RTMB example
- All that derivative stuff
  - derivatives, partial derivatives, second derivatives, cross derivatives (aka mixed second derivatives), gradient vector and Hessian
- Methods to calculate/approximate derivatives
- Exercise: finite difference derivatives
- How the Hessian and gradient vector are used
- RTMB Nonlinear regression vonb example

# Demos of estimating the mean the hard way

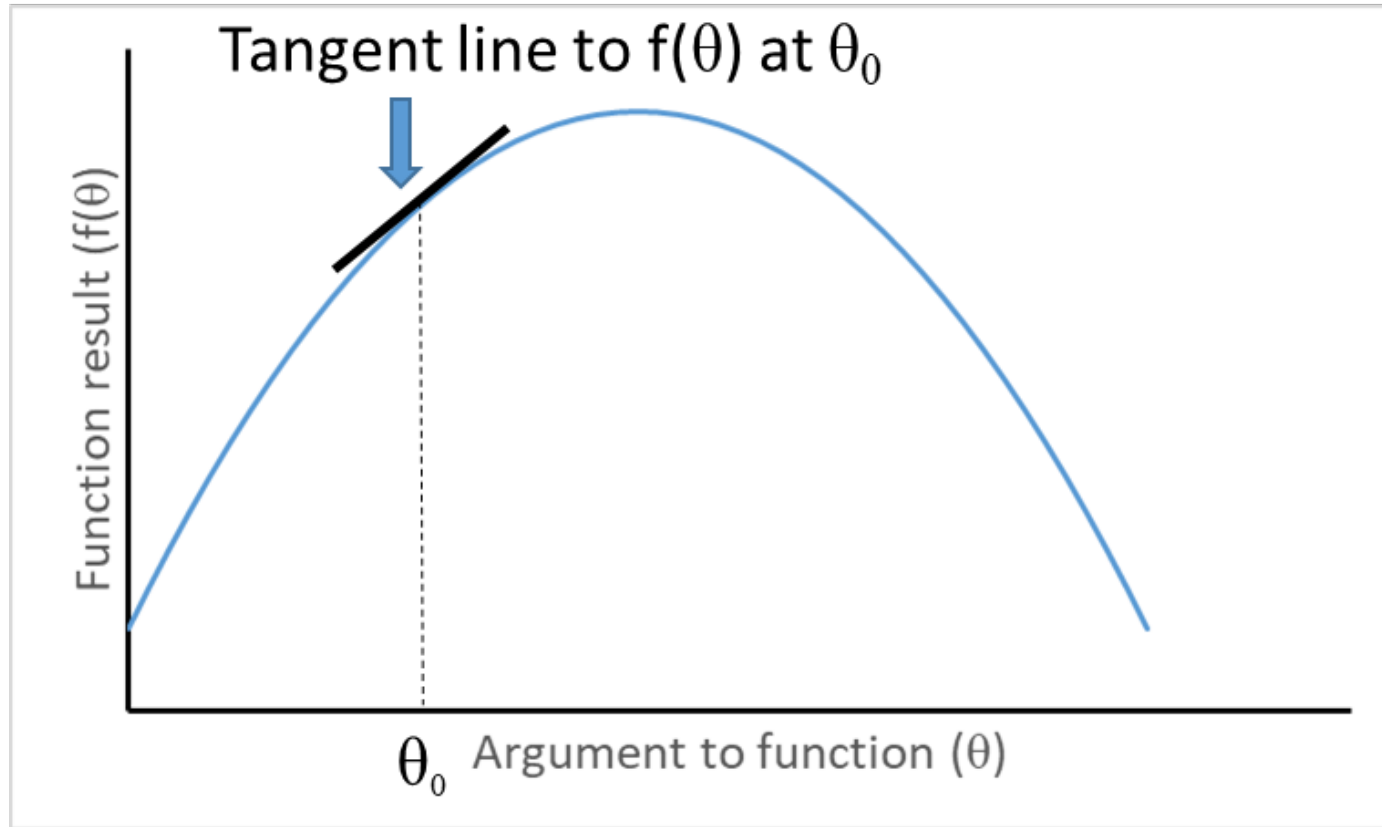
- Just using nlminb
- Using RTMB

# “Magic” when we used MakeADFun in RTMB

- Converts your parameter list and NLL function into new inputs for nlminb (and lots of hidden stuff)
  - obj\$par: your list of starting values as vector
  - obj\$fn: function pointing to memory location where NLL function result is stored
  - obj\$gr: function pointing to memory location where gradient stored
- IMPORTANT! obj\$fn and obj\$gr use hidden copy (created by MakeADFun) of any variables used in your NLL function

**Exercise - change model to assume gamma rather than normal (in breakout groups)**

# What is a derivative



$$\frac{df(\theta)}{d\theta} = \lim_{h \rightarrow 0} \frac{f(\theta + h) - f(\theta)}{h}$$

# Partial derivative

Function with multiple arguments but we treat all but one of them as constants, and calculate derivative with respect to just one! E.g.,

*[Math Processing Error]*

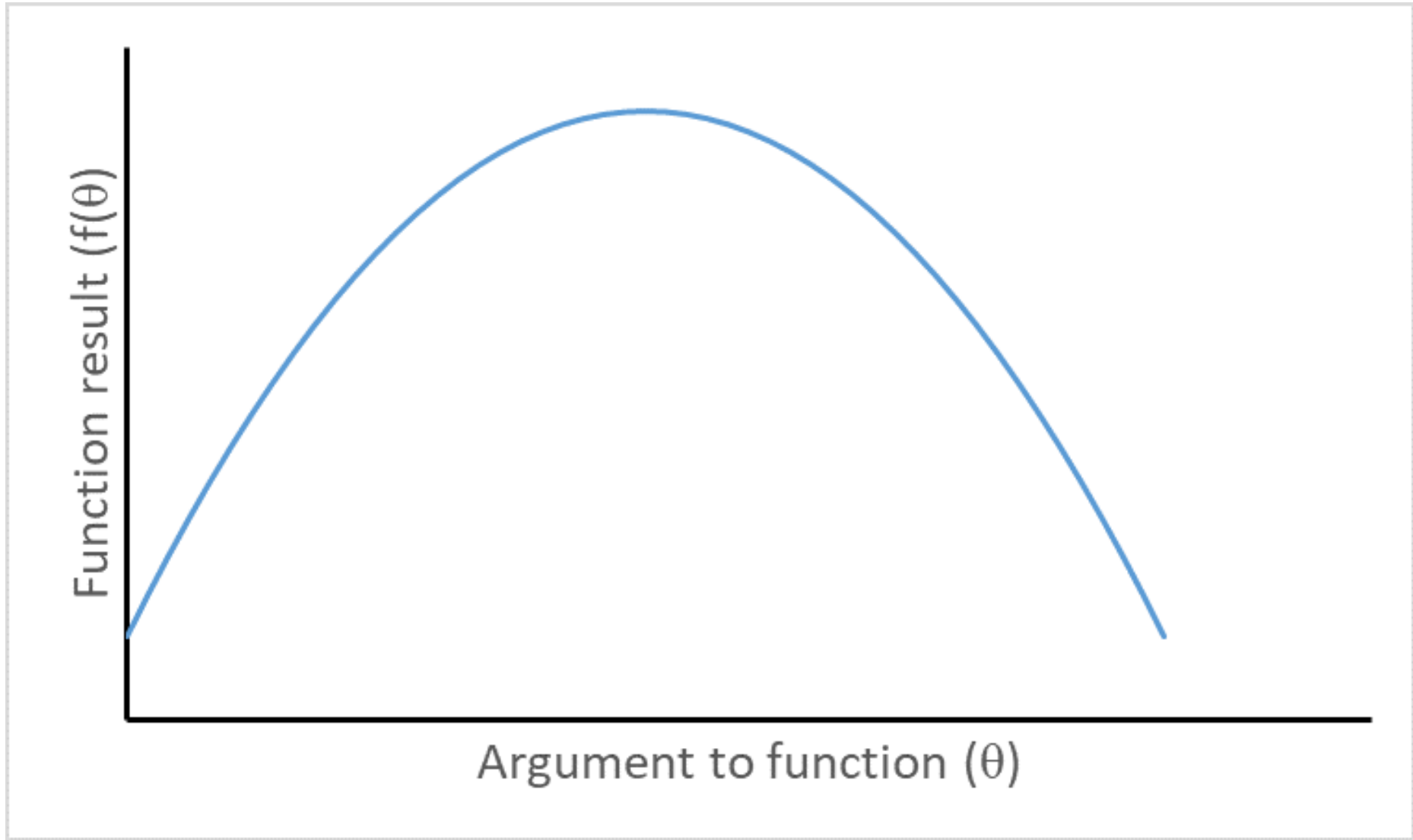
# Second derivative

Just a derivative of a derivative

$$\frac{\partial^2 f}{\partial \theta^2} = \frac{\partial \frac{\partial f}{\partial \theta}}{\partial \theta}$$

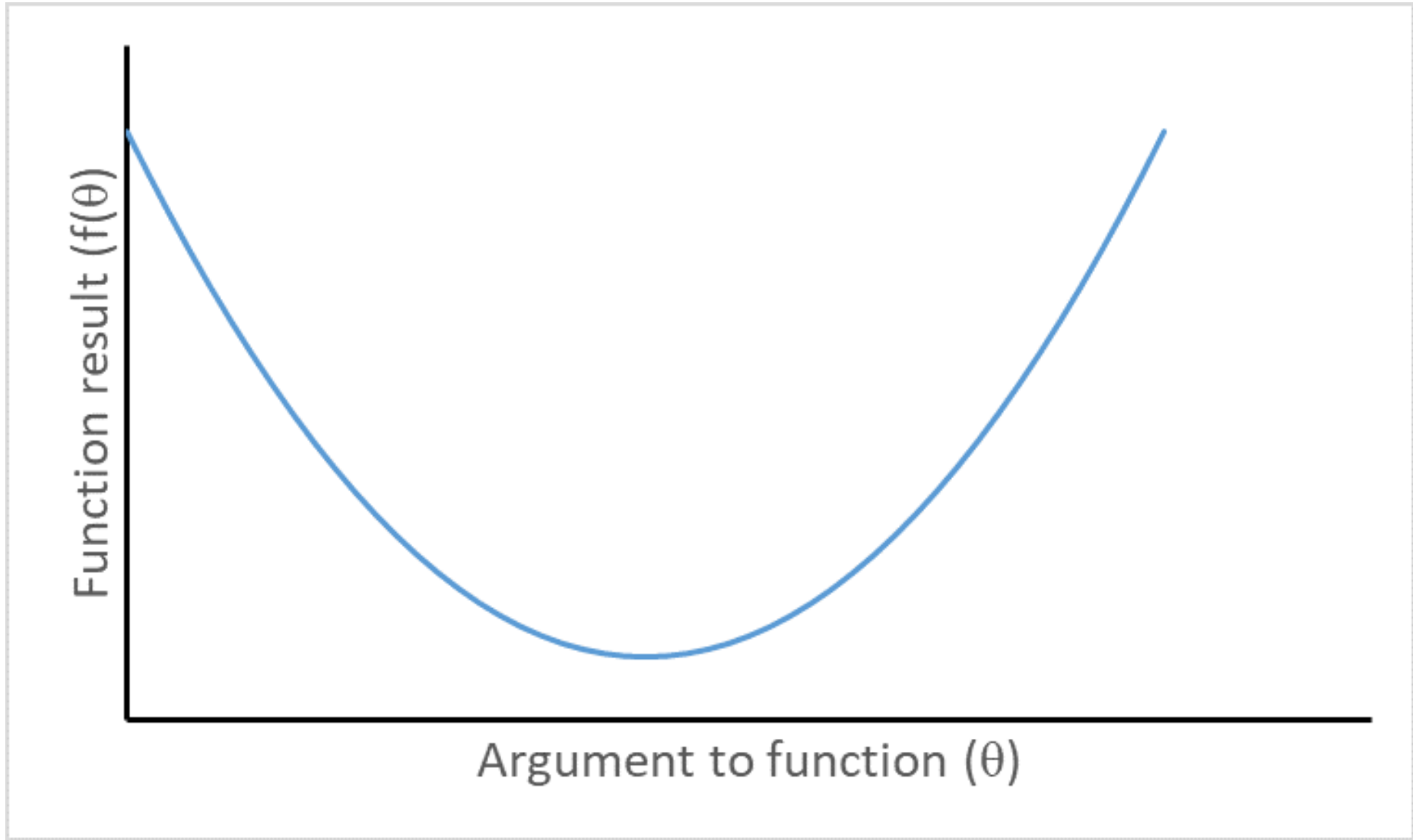


# Visualizing second derivatives



Concave function with negative second derivative

# Visualizing second derivatives



Convex function with positive second derivative

# Cross derivative (mixed second derivs)

$$\frac{\partial^2 f}{\partial \theta_1 \partial \theta_2} = \frac{\partial \frac{\partial f}{\partial \theta_1}}{\partial \theta_2}$$

An important and convenient fact:

$$\frac{\partial^2 f}{\partial \theta_1 \partial \theta_2} = \frac{\partial^2 f}{\partial \theta_2 \partial \theta_1}$$

# Methods for calculating derivatives

- Analytical derivatives. Gold standard but not available for many complex models.
- Finite difference methods. Intuitive but slow and propagate errors.
- Automatic differentiation. Fast and accurate but requires specialized software.

# Finite difference derivatives

Widely used, e.g., default of nlminb and Excel solver

Forward difference

*[Math Processing Error]*

h is semi-arbitrary but small relative to  $\theta_i$ .

Central differences

*[Math Processing Error]*

# Exercise - finite difference derivatives

- For  $g(x) = a + bX + \sin X$ , use finite difference methods to calculate the derivative of  $g(X)$  with respect to (wrt)  $X$ 
  - for  $x=1$ , with  $a=2$  and  $b=0.5$ . (answer approximately 1.040)
  - Repeat for  $x=2$ ,  $a=1$ , and  $b=1$ . Answer approximately 0.5839.
- For  $a=2$ ,  $b=0.5$ ,  $x=1$ , and same function, use finite differences to find the second derivative wrt  $x$  (answer approximately -0.8415)

# Automatic differentiation

- Uses repeated applications of chain rule:  
 $\partial z / \partial \theta = [\partial z / \partial y][\partial y / \partial \theta]$
- Simplest case.  $y = f(\theta)$ ,  $z = g(y)$ , i.e.,  $z = g(f(\theta))$
- General case we care about:

$$NLL = f_1(f_2(f_3(\dots f_k(\theta)\dots)))$$

# Gradient

Just a fancy term to mean the vector of derivatives of the NLL function with respect to each parameters (so if  $k$  parameters, then  $k$  elements)

*[Math Processing Error]*



# Hessian - a square symmetric matrix

*[Math Processing Error]*

$$h_{i,j} = h_{j,i} = \frac{\partial^2 f}{\partial \theta_i \partial \theta_j} = \frac{\partial^2 f}{\partial \theta_j \partial \theta_i}$$

If the NLL were a quadratic function as it would be for linear normal model...

$$\theta_{\min} = \theta_{\text{start}} + H^{-1}g$$

where  $H^{-1}$  is the matrix inverse of  $H$  and  $H^{-1}g$  is the product of the inverse of the Hessian and the gradient

# Because our models generally not normal and linear, iterative searches...

1. specify starting values for parameters,  $\underline{\theta}_0$
2. Replace  $\underline{\theta}_0$  by  $\underline{\theta}_1 = \underline{\theta}_0 + \delta_0$
3. Check gradient and Hessian and if at a minimum stop otherwise...
4. Return to step 2 but each time  $\underline{\theta}_{i+1} = \underline{\theta}_i + \delta_i$

Newton step:  $\underline{\delta}_i = H^{-1} \underline{g}$  evaluated at current params

Quasi-Newton method uses  $\underline{\delta}_i = \lambda H^{-1} \underline{g}$  with Hessian approximated using search path, and  $\lambda$  a number less than 1



# Using the Hessian to calculate asymptotic standard errors

- First some reminders
  - Parameter estimates are random variates that result from estimators (random variables)
  - The variance describes the variability of results from applying the estimation method, namely the expected squared deviation between an estimator and its expected value
  - What we report as a standard error for a parameter is the square-root of this variance.

# The variance-covariance matrix

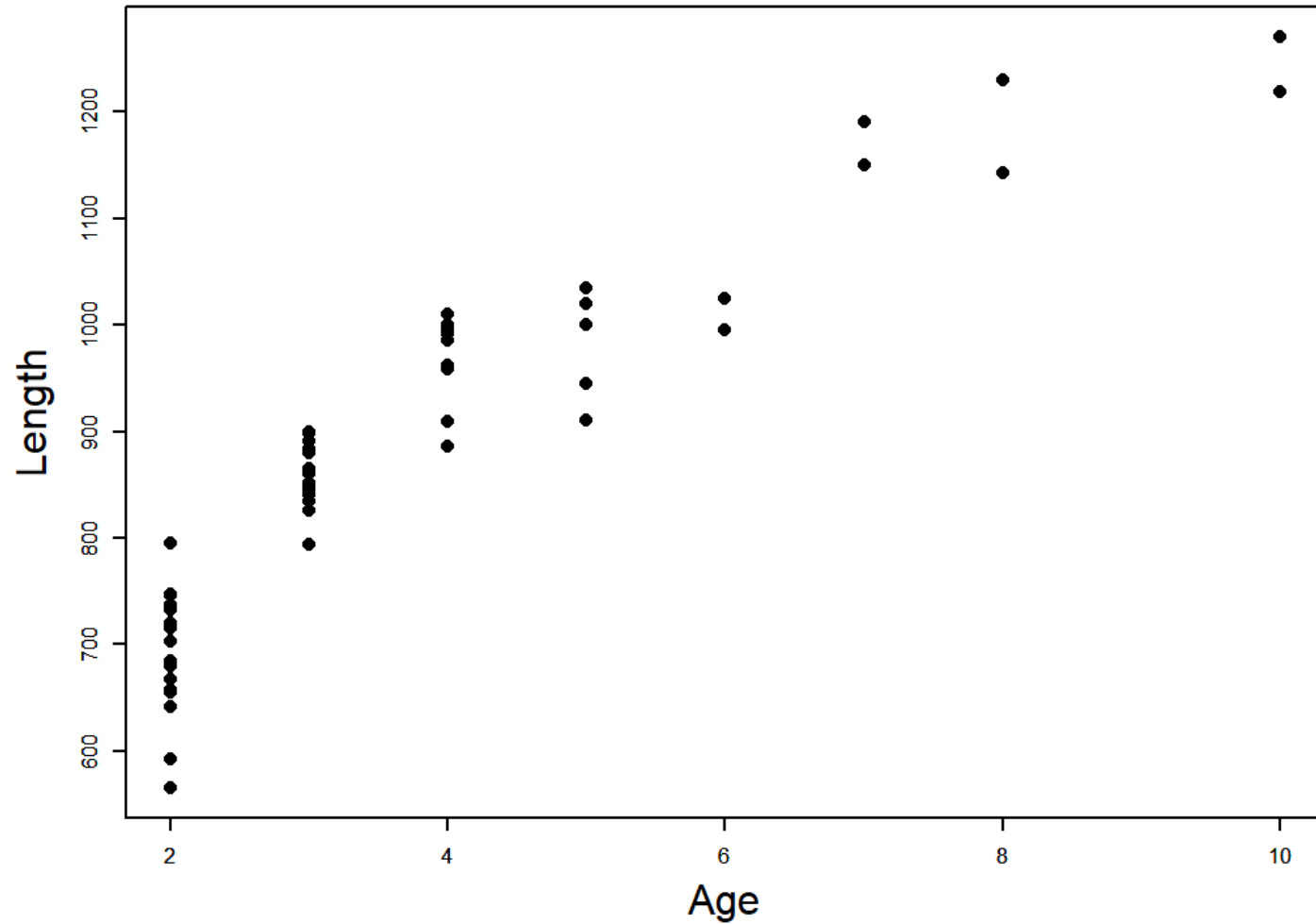
*[Math Processing Error]*

# The asymptotic variance-covariance matrix

$$\hat{\Sigma} = H^{-1}$$

- Square-root of diagonal gives standard errors
- Off-diagonals are covariances
- The Hessian needs to be positive definite for the calculation
- If the Hessian is not positive definite its a problem!
- Delta method used to obtain SEs for derived quantities (using  $\hat{\Sigma}$ )

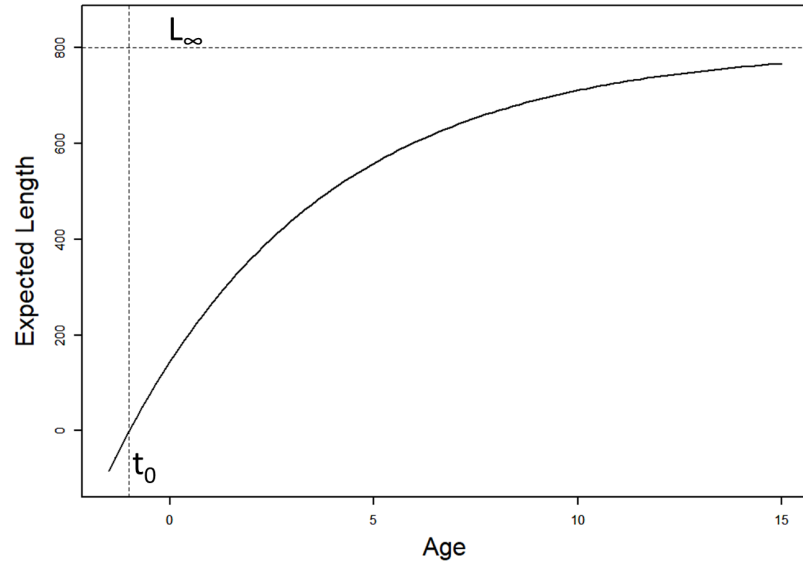
# Musky vonB example



musky\_vonb.dat



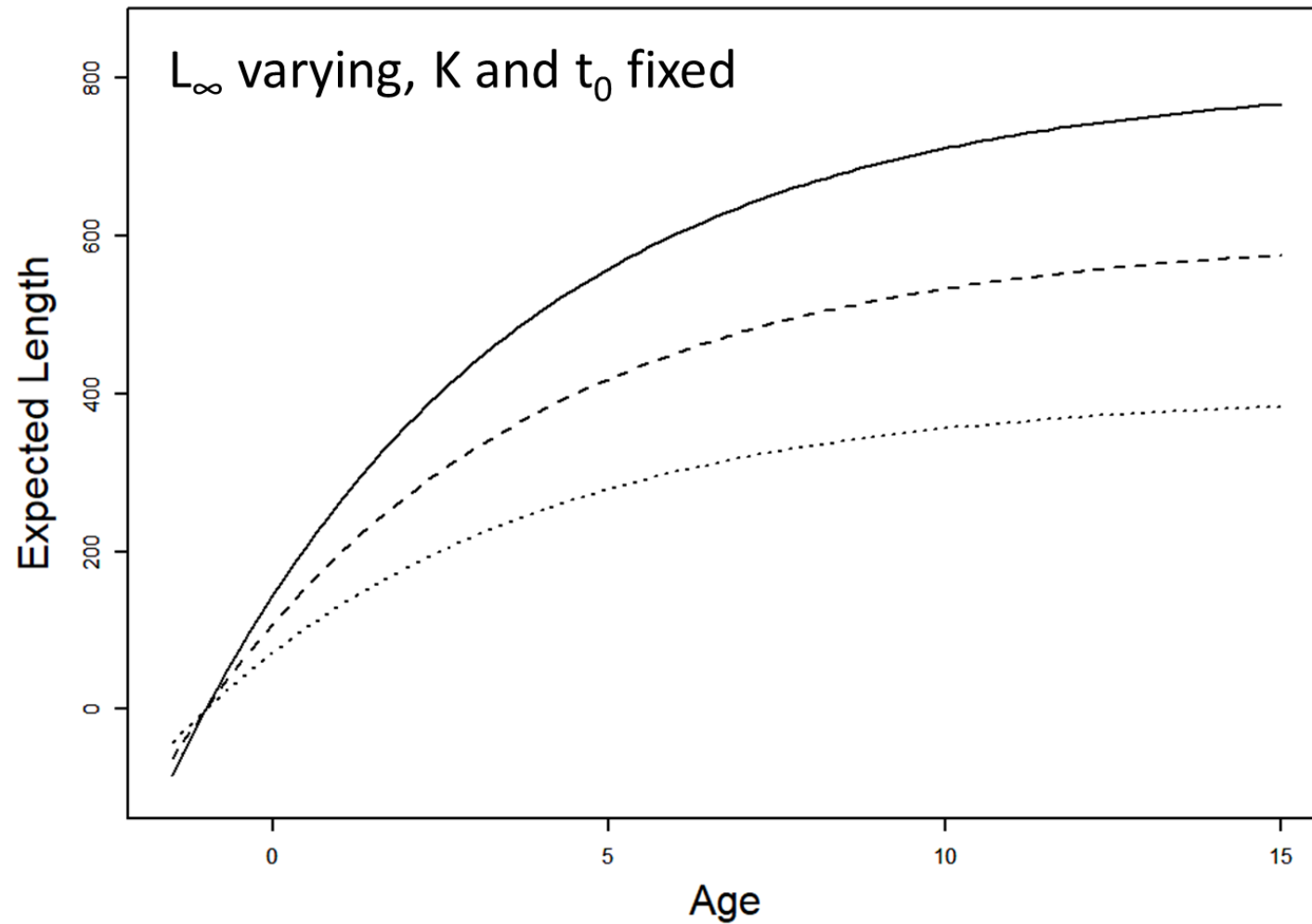
# von Bertalanffy model



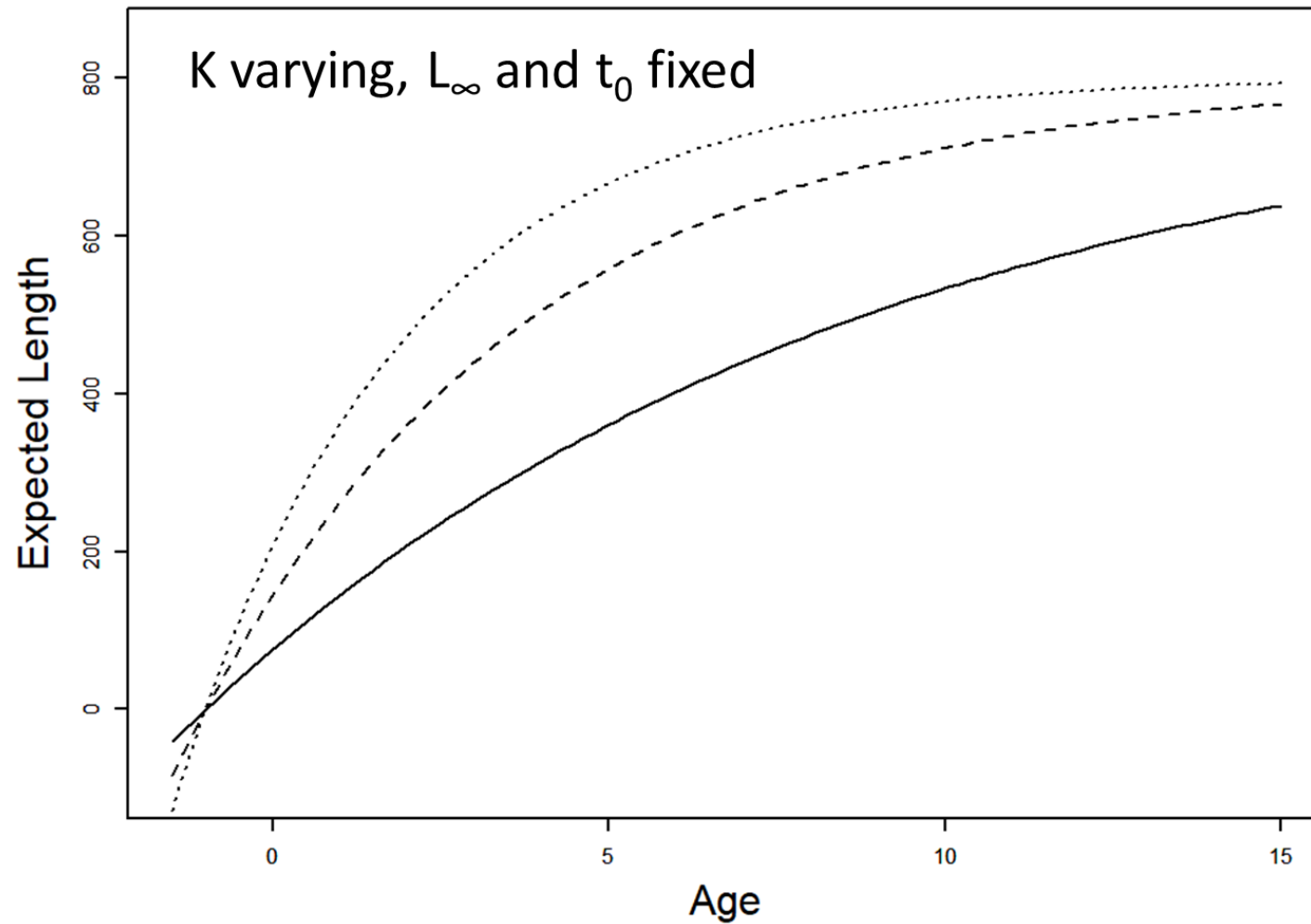
*[Math Processing Error]*

*[Math Processing Error]*

# Influence of $L_{\infty}$



# Influence of K



# Musky vonB setup code

```
1 library(RTMB);  
2  
3 gmRdat = read.table("lesson2/data/musky_vonb.dat", head=T);  
4  
5 #Set up the data and starting value of parameters for RTMB  
6 gmdat = list(len_obs=gmRdat[, "Length"], age=gmRdat[, "Age"]);  
7 gmpar = list(log_linf=7, log_vbk=-1.6, t0=0, log_sd=4);
```

# code for NLL for Musky vonb example

```
1  NLL_fun = function(par_lst){  
2    getAll(gmdat,par_lst);  
3    linf = exp(log_linf);  
4    vbk = exp(log_vbk);  
5    sd = exp(log_sd);  
6    len_pred = linf * (1 - exp(-vbk * (age - t0)));  
7    nll = -sum(dnorm(len_obs, len_pred, sd, TRUE));  
8    atage_pred = linf * (1 - exp(-vbk * ((1:11) - t0)))  
9    REPORT(atage_pred);  
10   nll  
11 }
```

# Create model object and print predicted lengths before fitting model

```
1  obj <- MakeADFun(NLL_fun, gmpar);  
2  
3  GMreport=obj$report();  
4  GMreport
```

\$atage\_pred

```
[1] 200.4870 364.3209 498.2026 607.6080 697.0118 770.0708 829.7731 878.5606  
[9] 918.4287 951.0082 977.6314
```

# fit the model

```
1 fit = nlminb(obj$par, obj$fn, obj$gr);
```

```
outer mgc: 3572.344
outer mgc: 115.0372
outer mgc: 306.8376
outer mgc: 101.0471
outer mgc: 30.69436
outer mgc: 135.1418
outer mgc: 120.5931
outer mgc: 25.99566
outer mgc: 197.1401
outer mgc: 122.5854
outer mgc: 66.52428
outer mgc: 36.54847
outer mgc: 3.778352
outer mgc: 19.59957
outer mgc: 1.539833
.
```

# Get parameter uncertainties and convergence diagnostics

```
1 sdr = sdreport(obj);
```

```
outer mgc: 0.0001970265
outer mgc: 19.77567
outer mgc: 19.71682
outer mgc: 9.178336
outer mgc: 9.171702
outer mgc: 2.35869
outer mgc: 2.358392
outer mgc: 0.1198859
outer mgc: 0.1201142
```

```
1 sdr #summary(sdr)
```

```
sdreport(.) result
      Estimate Std. Error
log_linf  7.1488714 0.04083255
log_vbk  -1.2456407 0.16950408
t0        -0.7710237 0.35092437
log_sd     3.8876390 0.09128707
Maximum gradient component: 0.0001970265
```



# Predicted lengths at age after model fitting

```
1 GMreport = obj$report();  
2 GMreport;
```

\$atage\_pred

```
[1] 508.1491 699.3217 842.6904 950.2090 1030.8420 1091.3122 1136.6614  
[8] 1170.6709 1196.1760 1215.3035 1229.6480
```

# vonB Exercises

- Change REPORT(atage\_pred) to ADREPORT(atage\_pred) and look at sdreport and summary of the sdreport
- Calculate a new variable equal to  $vb_k * l_{inf}$  as ADREPORT
- Change the model so data are assumed gamma distributed (with expected value given by vonB equation and constant variance)