Woods Hole

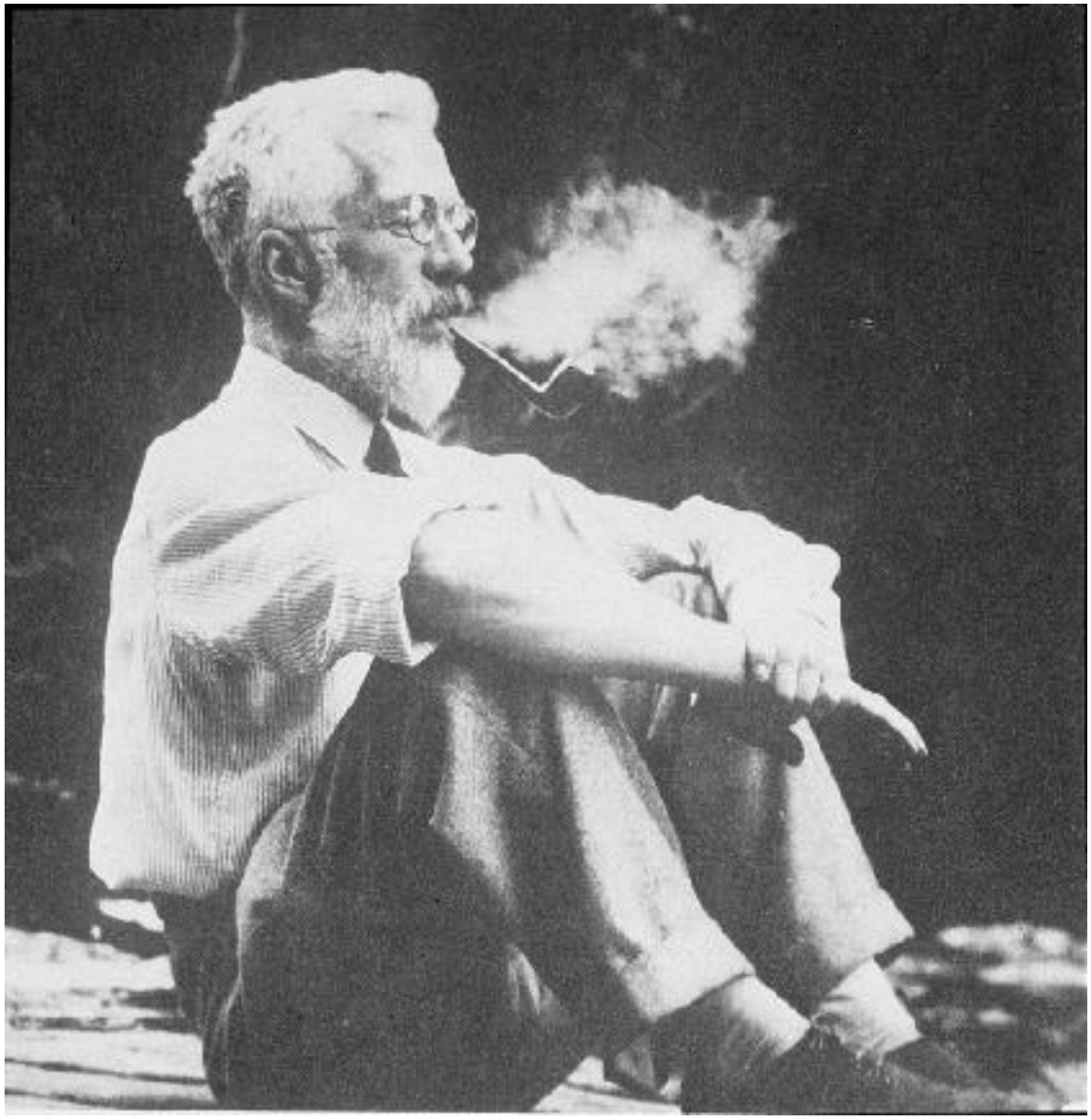
Likelihood based inference in TMB

Anders Nielsen

Outline

Don't worry this will not turn into a statistics course, but just a gentle reminder of

- Likelihood function $L(\theta) = P_\theta(Y = y)$
- Negative log likelihood function $\ell(\theta) = -\log(L(\theta))$
- Maximum likelihood estimate $\widehat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}}\ \ell(\theta)$
- Distribution of the ML estimator $\widehat{\theta} \sim \mathrm{N}(\theta, (\ell''(\widehat{\theta}))^{-1})$
- Likelihood ratio test $2(\ell_B(\widehat{\theta_B}, Y) - \ell_A(\widehat{\theta_A}, Y)) \sim \chi^2_{\dim(A)-\dim(B)}$



Sir Ronald Aylmer Fisher (1890 - 1962) identified the likelihood function as the key inferential quantity conveying all inferential information in statistical modelling including the uncertainty.

Observations with noise

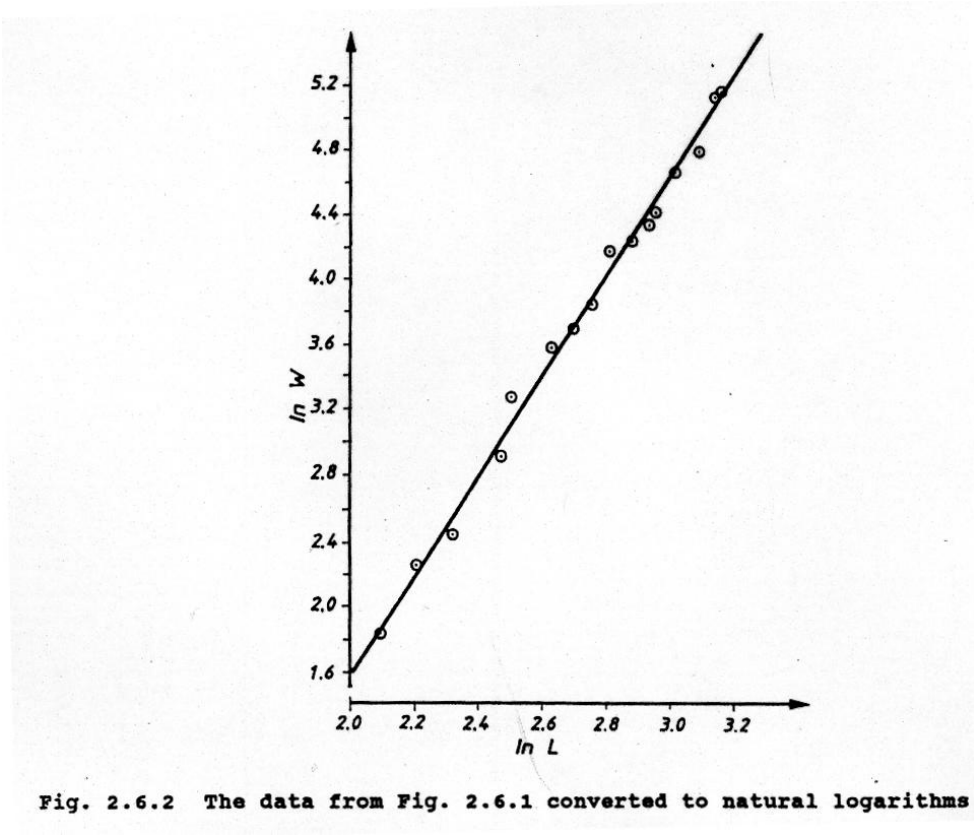


Fig. 2.6.2 The data from Fig. 2.6.1 converted to natural logarithms

- "Noise" is slang for unexplained variation in our observations
- Here the model $\log W_i = \alpha + \beta \log L_i$ is a very good description
- Still something is missing, as all the points are not exactly on the line

Statistical models

- Statistical models are explicit about the noise term.

$$\log W_i = \alpha + \beta \log L_i + \varepsilon_i, \text{ where } \varepsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ independent}$$

- Because:
  - We want to explain the entire system
  - We can better explain how good our model is
  - It help us to estimate the model parameters
  - It help us to quantify uncertainties on model parameters
  - It gives us an objective criteria for comparing models
  - ...
- Example question: Let's say I have a fish with log-length of 3.0 what can we say about its log-weight?
- Example question: How certain are we about our slope estimate?

Biological data uncertain?

- Catch at age data for instance:
  - Weights of (officially) landed fish
  - Samples of lengths
  - Samples of ages
  - Estimates of amount discarded at sea
- What do you think?

A simple example

Suppose we toss a thumbtack (used to fasten up documents to a background) 10 times and observe that 3 times it lands point up. Assuming we know nothing prior to the experiment, what is the probability of landing point up, θ?

- Binomial experiment with $y = 3$ and $n = 10$.
- P(Y=3; n=10, θ=0.1) = 0.0574
- P(Y=3; n=10, θ=0.2) = 0.2013
- P(Y=3; n=10, θ=0.3) = 0.2668
- P(Y=3; n=10, θ=0.4) = 0.2150
- P(Y=3; n=10, θ=0.5) = 0.1172

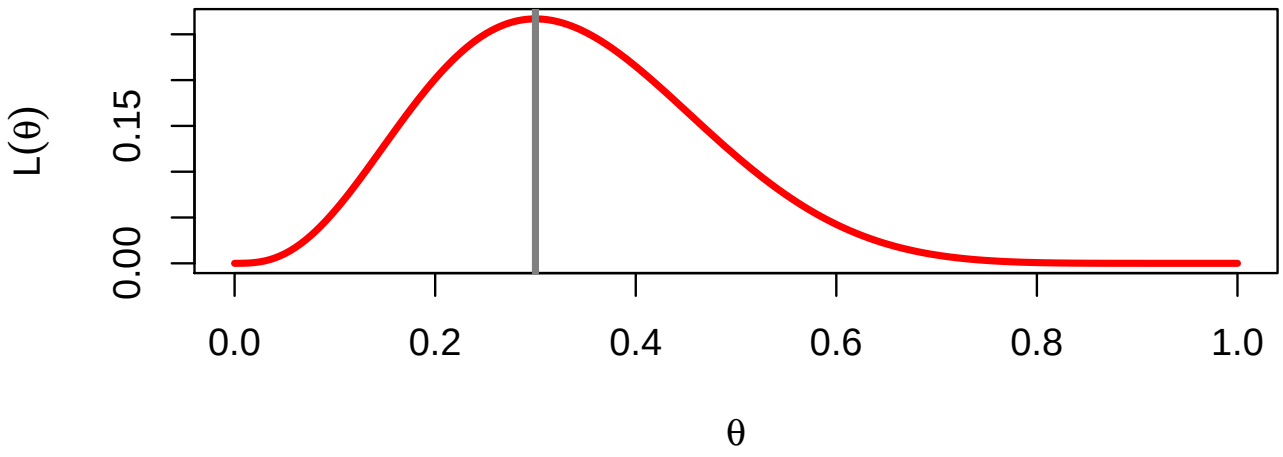
A simple example

By considering $\mathrm{P}_\theta(Y = 3)$ to be a function of the unknown parameter we have the *likelihood function*:

$$L(\theta) = \mathrm{P}_\theta(Y = 3)$$

In general, in a Binomial experiment with n trials and y successes, the likelihood function is:

$$L(\theta) = \mathrm{P}_\theta(Y = y) = \binom{n}{y} \theta^y (1-\theta)^{n-y}$$



A simple example

It is often more convenient to consider the negative log-likelihood function. The negative log-likelihood function is:

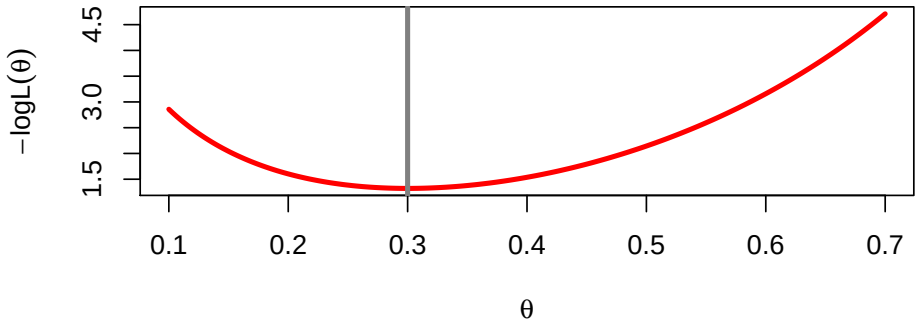
$$\ell(\theta) = -\log L(\theta) = -y \log \theta - (n-y) \log(1-\theta) + const$$

where *const* indicates a term that does not depend on θ.
By solving

$$\ell'(\theta) = 0$$

it is readily seen that the maximum likelihood *estimate* (MLE) for θ is

$$\widehat{\theta}(y) = \frac{y}{n} = \frac{3}{10} = 0.3$$



The likelihood principle

- Not just a method for obtaining a point estimate of parameters.
- It is the entire likelihood function that captures all the information in the data about a certain parameter.
- Likelihood based methods are inherently computational. In general numerical methods are needed to find the MLE.
- Today the likelihood principles play a central role in statistical modelling and inference.

Maximum likelihood estimator

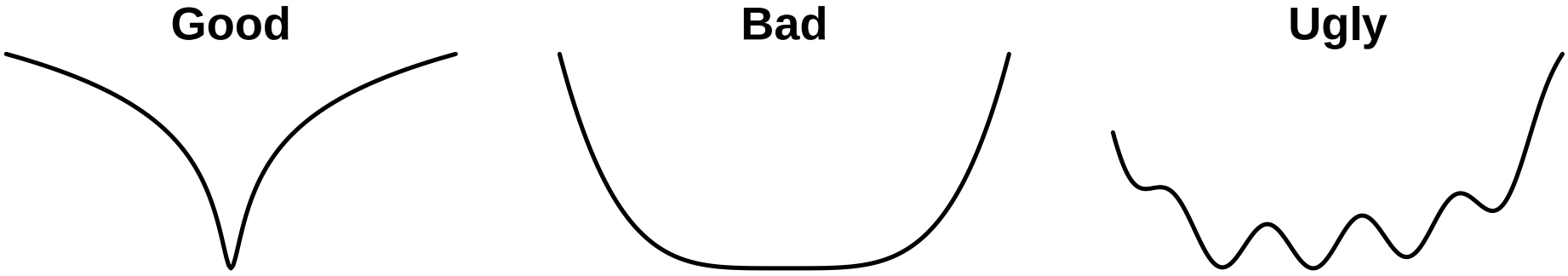
- A sensible estimate of the model parameters is to choose the values that maximize the likelihood for the actual observations.

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

- The curvature of the negative log likelihood function gives an estimate of the maximum likelihood estimator:

$$\widehat{\operatorname{var}(\widehat{\theta})} = \left(\frac{\partial^2 \ell(y|\theta)}{\partial \theta^2} \bigg|_{\theta=\widehat{\theta}} \right)^{-1}$$

- The matrix $\mathcal{H}(\widehat{\theta}) = \left(\frac{\partial^2 \ell(y|\theta)}{\partial \theta^2} \big|_{\theta=\widehat{\theta}} \right)$ is often referred to as "the hessian matrix"
- Both the estimator and the hessian matrix are often found by numerical methods.



Example: Likelihood function for mean of normal distribution

An automatic production of a bottled liquid is considered to be stable. A sample of 10 bottles were selected at random from the production and the volume of the content volume was measured. The deviation from the nominal volume of 700.0 ml was recorded.

The deviations (in ml) were:

1.67 -0.55 -0.71 0.17 4.5 1.67 2.62 5.01 2.34 -0.85

Example: Likelihood function for mean of normal distribution

First a *model* is formulated

i Model: C+E (center plus error) model, $Y = \mu + \epsilon$

ii Data: $Y_i = \mu + \epsilon_i$

iii Assumptions:

- $Y_1, Y_2, ..., Y_{10}$ are independent
- $Y_i \sim \mathrm{N}(\mu, \sigma^2)$

Thus, there are two unknown model parameter, μ and σ.

Example: Likelihood function for mean of normal distribution

The joint probability density function for $Y_1, Y_2, \ldots, Y_{10}$ is given by

$$\begin{aligned} L(y_1, \ldots, y_{10}; \mu, \sigma) = & \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(y_1-\mu)^2}{2\sigma^2}\right] \\ & \times \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(y_2-\mu)^2}{2\sigma^2}\right] \\ & \times \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(y_3-\mu)^2}{2\sigma^2}\right] \\ & \times \cdots \\ & \times \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(y_{10}-\mu)^2}{2\sigma^2}\right] \end{aligned}$$

Example: Likelihood function for mean of normal distribution

We have observed the ten values, so:

$$\begin{aligned} L(y_1, \ldots, y_{10}; \mu, \sigma) = & \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(1.67-\mu)^2}{2\sigma^2}\right] \\ & \times \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(-0.55-\mu)^2}{2\sigma^2}\right] \\ & \times \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(-0.71-\mu)^2}{2\sigma^2}\right] \\ & \times \cdots \\ & \times \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(-0.85-\mu)^2}{2\sigma^2}\right] \end{aligned}$$

Now this function actually only depends on μ and σ.

Example: Likelihood function for mean of normal distribution

Taking the log we get:

$$\begin{aligned}\log L(y_1, \ldots, y_{10}; \mu, \sigma) = &-\frac{1}{2}\log(2\pi\sigma^2) - \frac{(1.67-\mu)^2}{2\sigma^2} \\ &-\frac{1}{2}\log(2\pi\sigma^2) - \frac{(-0.55-\mu)^2}{2\sigma^2} \\ &-\frac{1}{2}\log(2\pi\sigma^2) - \frac{(-0.71-\mu)^2}{2\sigma^2} \\ &-\ldots \\ &-\frac{1}{2}\log(2\pi\sigma^2) - \frac{(-0.85-\mu)^2}{2\sigma^2}\end{aligned}$$

If we collect the terms the *negative* log likelihood becomes:

$$\ell(\mu, \sigma) = \frac{10}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\sum(y_i - \mu)^2$$

Linear regression as ML

- Consider the frequently used model:

$$y_i = \alpha + \beta \cdot x_i + \varepsilon_i \ , \quad \text{where } \varepsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ independent}$$

- Or put slightly different:

$$y_i \sim \mathcal{N}(\alpha + \beta \cdot x_i, \sigma^2) \text{ independent}$$

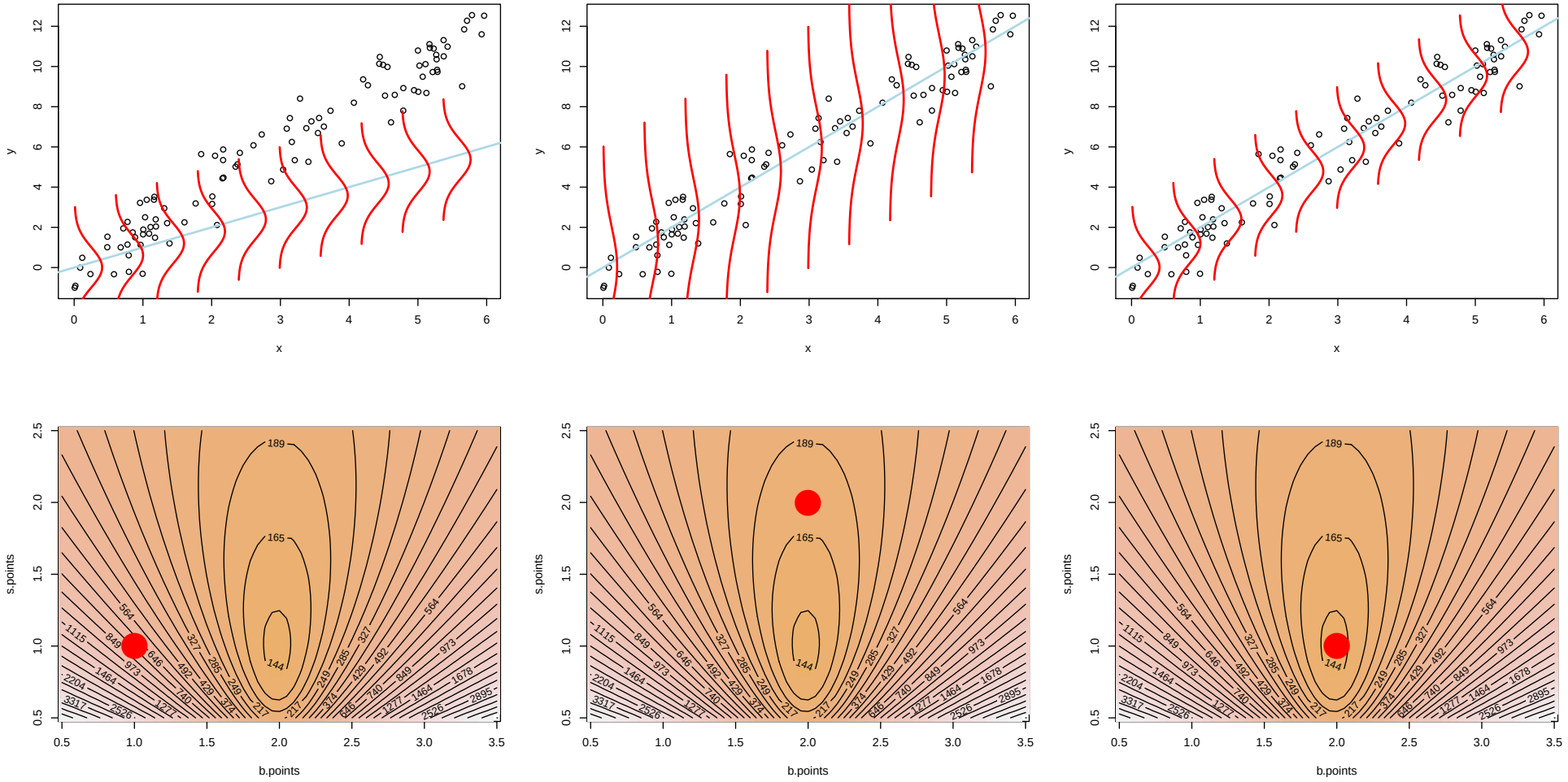
- Likelihood function: For given model parameters: $\theta = (\alpha, \beta, \sigma^2)$ we can via the model express the probability (likelihood) of seeing the actual observations y

$$L(y|\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y_i - (\alpha + \beta \cdot x_i))^2\right)$$

- Negative log likelihood: Often it is preferred to use $\ell(y|\theta) = -\log(L(y|\theta))$ instead:

$$\ell(y|\theta) = \frac{n}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - (\alpha + \beta \cdot x_i))^2$$

How to choose (estimate) the model parmeters



(in this example the intercept is considered fixed)

Likelihood functions from a few known models

Poisson: $x_i \sim Pois(\lambda)$ independent

$$\ell(x|\lambda) = \lambda n - \log(\lambda)\sum x_i + \sum \log(x_i!)$$

```
nll = -sum(dpois(X,lambda,true));
```

Normal: $x_i \sim \mathcal{N}(\mu, \sigma^2)$ independent

$$\ell(x|\mu,\sigma^2) = \frac{n}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\sum(x_i-\mu)^2$$

```
nll= -sum(dnorm(X,mu,sigma,true));
```

Binomial: $x_i \sim Bin(N_i, p)$ independent (assume N_i known)

$$\ell(x|p) = -\sum \log \binom{N_i}{x_i} - \log(p)\sum x_i - \log(1-p)\sum(N_i - x_i)$$

```
nll = -sum(dbinom(X,N,p,true));
```

Notation: In the above `lambda`, `mu`, `sigma`, and `p` are model parameters, `X` is the observation vector, and `N` is the number of observations, except for the binomial where `N` is a vector of the number of trials.

Likelihood ratio test

- Assume model B is a sub model of model A (this is for instance the case if a free model parameter in A is set to a fixed value in B)
- We can calculate the test statistic $G_{A\to B}$ for reducing model A to model B by:

$$G_{A\to B} = 2(\ell_B(y|\widehat{\theta_B}) - \ell_A(y|\widehat{\theta_A}))$$

- If the two optimal fits are "almost equal" the model reduction is accepted, if the fits are very different the model reduction is rejected
- Asymptotically G followers a χ^2–distribution, so the P-value is given by:

$$P_{A\to B} = P\left(\chi^2_{\dim(A)-\dim(B)} \geq G_{A\to B}\right)$$

- If this is small (often defined as $< 5\%$) the actual observations matches B poorly and the model reduction is rejected.
- Mention $\text{AIC}_A = 2(\dim(A) + \ell_A(y|\widehat{\theta_A}))$

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

  Wish to estimate the two unknown parameters.

- For one observation x the negative log likelihood is:

$$\ell(x, n, p) = -\log(\Gamma(x+n)) + \log(\Gamma(n)) + \log(\Gamma(x+1)) - n\log(p) - x\log(1-p)$$

- The TMB code becomes

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

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

param <- list()
param$logsize <- 0
param$p <- 0.5

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

```
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator() ()
{
  DATA_VECTOR(Y);
  PARAMETER(logsize);
  PARAMETER(p);
  Type size = exp(logsize);
  Type nll = -sum(dnbinom(Y, size, p, true));
  ADREPORT(size);
  return nll;
}
```

Running it

```
> library(TMB)
> compile("nbin.cpp")
Note: Using Makevars in /home/andni/.R/Makevars
make: Nothing to be done for `all'.
[1] 0
> dyn.load(dynlib("nbin"))
>
> data <- list()
> data$Y <- c(13, 5, 28, 28, 15, 4, 13, 4,
+             10, 17, 11, 13, 12, 17, 3)
>
> param <- list()
> param$logsize <- 0
> param$p <- 0.5
>
> obj <- MakeADFun(data, param, DLL="nbin")
Constructing atomic D_lgamma
> opt <- nlminb(obj$par, obj$fn, obj$gr)
outer mgc:  356
outer mgc:  284.0977
outer mgc:  627.8282
outer mgc:  76.31997
... <cut>
outer mgc:  0.002353296
outer mgc:  0.0001551027
Warning message:
In nlminb(obj$par, obj$fn, obj$gr) : NA/NaN function evaluation
> summary(sdreport(obj))
outer mgc:  0.0001551027
... <cut>
outer mgc:  3.675389
         Estimate Std. Error
logsize 1.3016590 0.47099344
p       0.2221847 0.08556683
size    3.6753893 1.73108423
>
```

Exercise: Fitting Poisson regression

- Consider the model:

$$y_i \sim \text{Pois}(e^{\theta_1 x_i + \theta_2}),\ i = 1, \ldots, 10$$

- With observations given in the following R snip:

```
> x <- 1:10
> y <- c(3, 0, 4, 5, 6, 4, 9, 7, 4, 10)
```

- Write a program to estimate the two model parameters
- Test the hypothesis H_0: $\theta_2 = 0$

Solution: Poisson regression

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

data <- list()
data$x <- 1:10
data$y <- c(3, 0, 4, 5, 6, 4, 9, 7, 4, 10)

param <- list()
param$theta <- c(0,0)

obj <- MakeADFun(data, param, DLL="poisreg")
opt <- nlminb(obj$par, obj$fn, obj$gr)
summary(sdreport(obj))
#
#       Estimate Std. Error
#theta 0.8019308 0.36261063
#theta 0.1395726 0.05064619

obj$fn()
#
# 21.41204

obj <- MakeADFun(data, param, DLL="poisreg",
                 map=list(theta=factor(c(NA,1))))
opt <- nlminb(obj$par, obj$fn, obj$gr)
obj$fn()
#
# 23.5391

1-pchisq(2*(23.5391-21.41204),1)
#
#  0.03915523
```

```
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator() ()
{
  DATA_VECTOR(x);
  DATA_VECTOR(y);
  PARAMETER_VECTOR(theta);
  vector<Type> logLam = theta(0)+theta(1)*x;
  Type nll = -sum(dpois(y, exp(logLam), true));
  return nll;
}
```

Exercise: Robust linear regression

- Make a robust regression, where the noise is distributed as a mixture between a normal and a t_1-distribution (p.d.f: $p(x) = ((1-p)\phi((x-\mu)/s) + pt_1((x-\mu)/s))/s$). Insert an artificial outlier and see if it works.
- The following data set can be used:

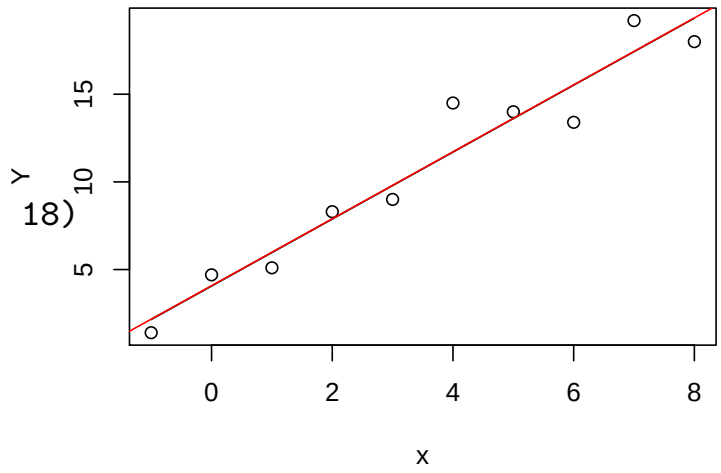
```
> data <- list()
> data$x <- c(-1, 0, 1, 2, 3, 4, 5, 6, 7, 8)
> data$Y <- c(1.4, 4.7, 5.1, 8.3, 9.0, 14.5, 14.0, 13.4, 19.2, 18)
```

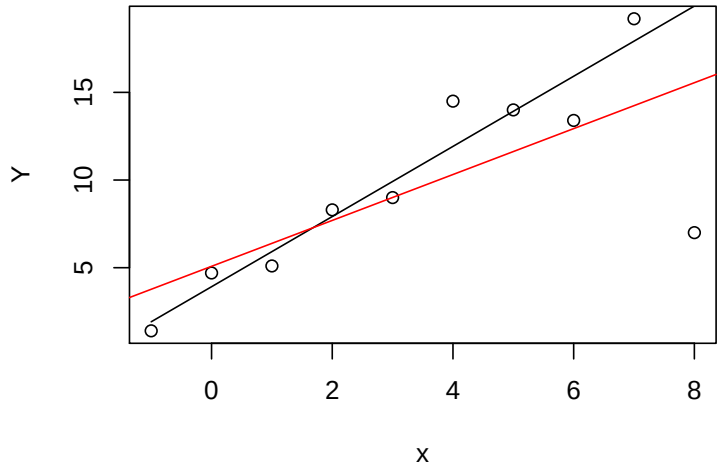
Solution: Robust linear regression (cpp)

```
#include <TMB.hpp>

template<class Type>
Type drobust(Type x, Type m, Type s, Type p, int give_log = 0){
  Type z=(x-m)/s;
  Type logres=log(1.0/s*((1.0-p)*dnorm(z,Type(0.0),Type(1.0),false)+p*dt(z,Type(1),false)));
  if(!give_log){
    return exp(logres);
  }else{
    return logres;
  }
}


template<class Type>
Type objective_function<Type>::operator() ()
{
  DATA_VECTOR(x);
  DATA_VECTOR(Y);
  PARAMETER(alpha);
  PARAMETER(beta);
  PARAMETER(logsigma);
  Type sigma = exp(logsigma);
  Type nll=0;
  for(int i=0; i<x.size(); ++i){
    nll += -drobust(Y(i), alpha*x(i)+beta, sigma, Type(0.1), true);
  }
  vector<Type> pred = alpha*x+beta;
  ADREPORT(sigma);
  ADREPORT(pred);
  return nll;
}
```

Solution: Robust linear regression (R)

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

data <- list()
data$x <- c(-1, 0, 1, 2, 3, 4, 5, 6, 7, 8)
data$Y <- c(1.4, 4.7, 5.1, 8.3, 9.0, 14.5, 14.0, 13.4, 19.2, 18)

param <- list(alpha = 0, beta = 0, logsigma = 0)

par(mfrow=c(2,1))
obj <- MakeADFun(data, param, DLL="lmr")
opt <- nlminb(obj$par, obj$fn, obj$gr)
sds<-summary(sdreport(obj))
plot(data$x,data$Y, xlab="x", ylab="Y")
lines(data$x, sds[rownames(sds)=="pred",1])
obj <- MakeADFun(data, param, DLL="lm")
opt <- nlminb(obj$par, obj$fn, obj$gr)
abline(b=opt$par[1], a=opt$par[2], col="red")

data$Y[10]<-7
obj <- MakeADFun(data, param, DLL="lmr")
opt <- nlminb(obj$par, obj$fn, obj$gr)
sds<-summary(sdreport(obj))
plot(data$x,data$Y, xlab="x", ylab="Y")
lines(data$x, sds[rownames(sds)=="pred",1])

obj <- MakeADFun(data, param, DLL="lm")
opt <- nlminb(obj$par, obj$fn, obj$gr)
abline(b=opt$par[1], a=opt$par[2], col="red")
```





A1: Complete program using Poisson likelihood

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

data <- list()
data$Y <- rpois(1000,3)

param <- list()
param$loglambda <- 0

obj <- MakeADFun(data, param, DLL="pois")
opt <- nlminb(obj$par, obj$fn, obj$gr)
summary(sdreport(obj))
```

```
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator() ()
{
  DATA_VECTOR(Y);
  PARAMETER(loglambda);
  Type lambda = exp(loglambda);
  Type nll = -sum(dpois(Y, lambda, true));
  ADREPORT(lambda);
  return nll;
}
```

A2: Complete program using normal likelihood

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

data <- list()
data$Y <- rnorm(1000,2,.3)

param <- list()
param$mu <- 0
param$logsigma <- 0

obj <- MakeADFun(data, param, DLL="norm")
opt <- nlminb(obj$par, obj$fn, obj$gr)
summary(sdreport(obj))
```

```
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator() ()
{
  DATA_VECTOR(Y);
  PARAMETER(mu);
  PARAMETER(logsigma);
  Type sigma = exp(logsigma);
  Type nll = -sum(dnorm(Y, mu, sigma, true));
  ADREPORT(sigma);
  return nll;
}
```

A3: Complete program using binomial likelihood

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

data <- list()
data$N <- seq(10,100,by=10)
data$Y <- rbinom(10,size=data$N, prob=.3)

param <- list()
param$p <- 0.5

obj <- MakeADFun(data, param, DLL="bin")
opt <- nlminb(obj$par, obj$fn, obj$gr,
              lower=0, upper=1)
summary(sdreport(obj))
```

```
#include <TMB.hpp>
template<class Type>
Type objective_function<Type>::operator() ()
{
  DATA_VECTOR(N);
  DATA_VECTOR(Y);
  PARAMETER(p);
  Type nll = -sum(dbinom(Y, N, p, true));
  return nll;
}
```