ICES-TCADSA

Likelihood based inference

Anders Nielsen & Arni Magnusson

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 $2(\ell_B(\widehat{\theta_B}, Y) - \ell_A(\widehat{\theta_A}, Y)) \sim \chi^2_{\dim(A)-\dim(B)}$

Fisher (1922) identified the likelihood function as the key inferential quantity conveying all inferential information in statistical modelling including the uncertainty

A motivating 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

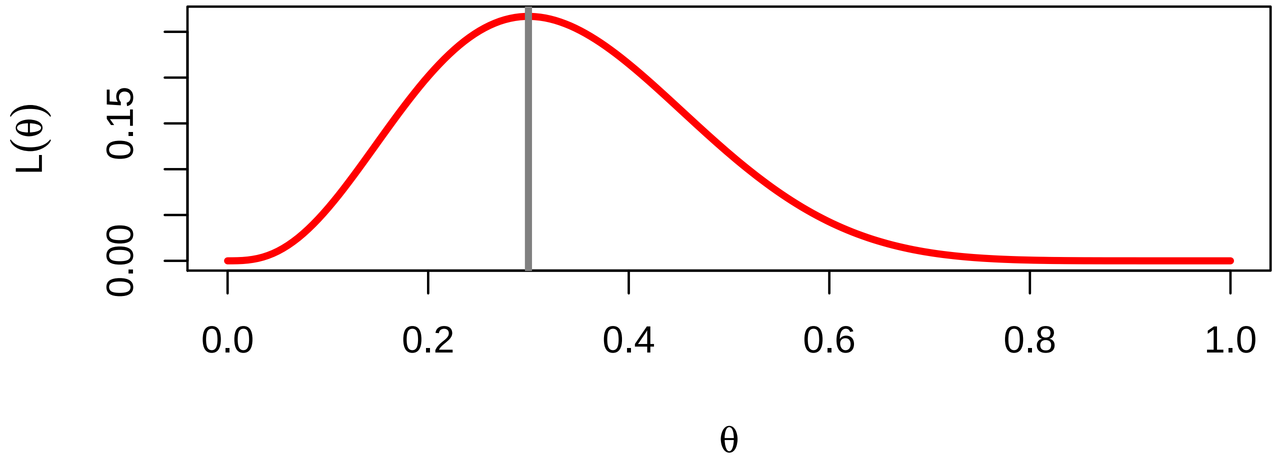
A motivating example

By considering $P_\theta(Y=3)$ to be a function of the unknown parameter we have the *likelihood function*:

$$L(\theta) = P_\theta(Y=3)$$

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

$$L(\theta) = P_\theta(Y=y) = \begin{pmatrix} n \\ y \end{pmatrix} \theta^y (1-\theta)^{n-y}$$



A motivating example

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

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

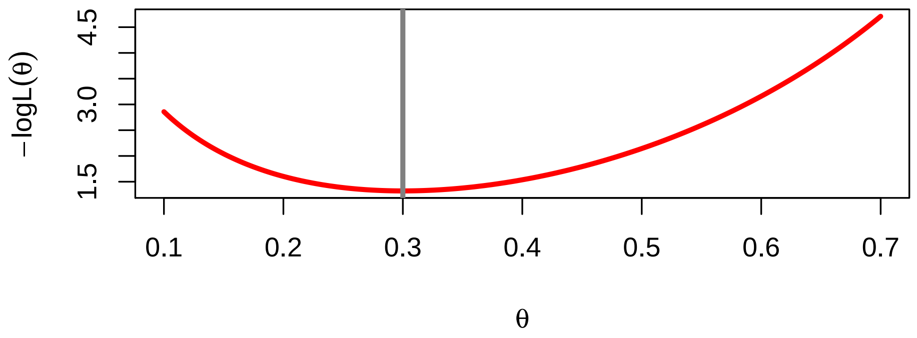
where *const* indicates a term that does not depend on θ.

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

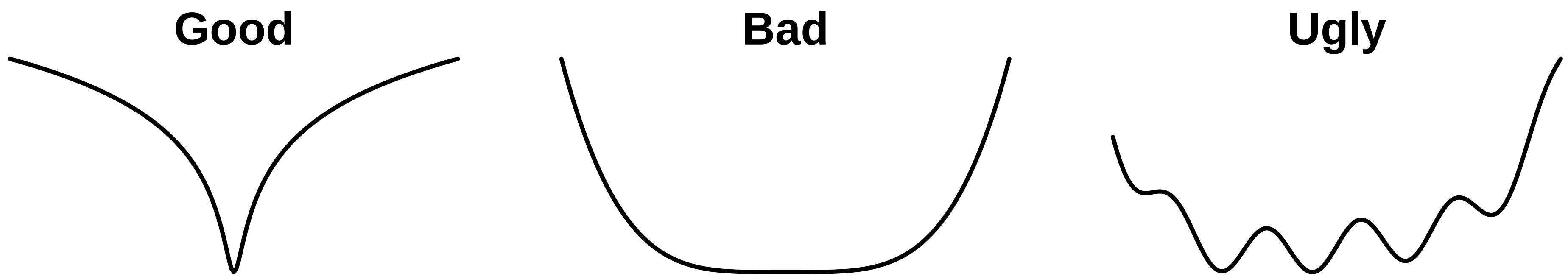
- A sensible estimate of the model parameters is to choose the values that maximize the likelihood for the actual observations.

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

- 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:

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

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

- 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:

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

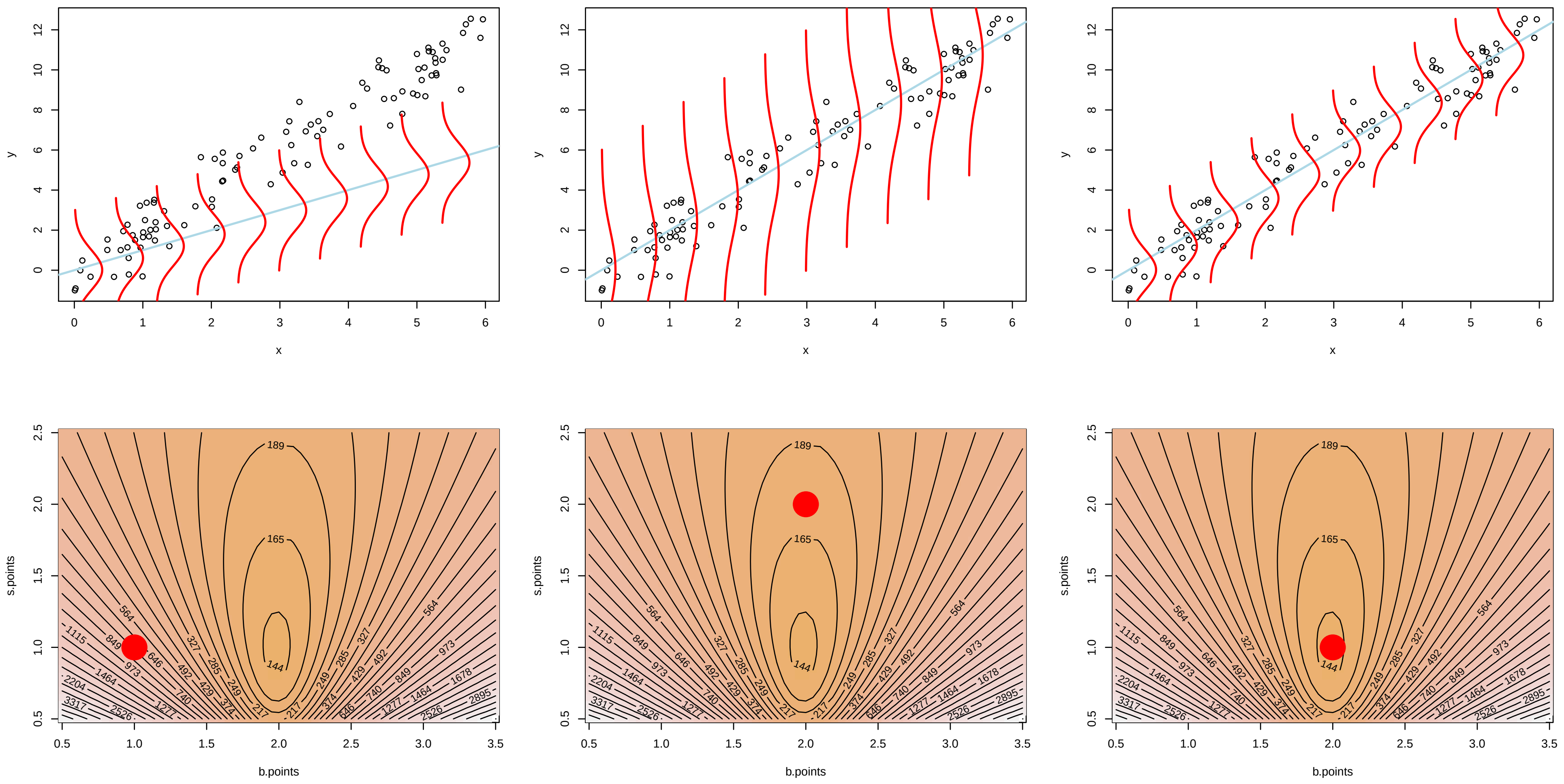
- Likelihood function: For given model parameters: $\theta = (\alpha, \beta, \sigma^2)$ we can via the model express the probability (likelihood) of seeing the actual observations y

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

- 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=lambda*N-log(lambda)*sum(X)+sum(gammln(X+1.0));
```

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

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

```
dvariable ss=square(sigma);
nll=0.5*(N*log(2.0*M_PI*ss)+sum(square(X-mu))/ss);
```

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(log_comb(N,X))-log(p)*sum(X)-log(1.0-p)*sum(N-X);
```

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 $\mathrm{AIC}_A = 2(\dim(A) + \ell_A(y|\widehat{\theta_A}))$

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 AD Model Builder program becomes

```
DATA_SECTION
  init_vector X(1,15);

PARAMETER_SECTION
  init_number logsize;
  init_bounded_number p(0,1);
  sdreport_number size;
  objective_function_value nll;

PROCEDURE_SECTION
  size=exp(logsize);
  int N=X.indexmax()-X.indexmin()+1;
  nll=-sum(gammln(X+size))+N*gammln(size)+
       sum(gammln(X+1.0))-N*size*log(p)-sum(X)*log(1.0-p);
```

```
 index    name     value      std dev
     1    logsize  1.3017e+00 4.7101e-01
     2    p        2.2218e-01 8.5571e-02
     3    size     3.6754e+00 1.7312e+00
```

Exercise: Linear regression via maximum likelihood

Implement the linear regression model $Y_i = \alpha x_i + \beta + \varepsilon_i$, where $\varepsilon_i \sim N(0, \sigma^2)$. Use the data below to estimate the three model parameters.

```
# number of observations
     10
# observed Y values
    1.4  4.7  5.1  8.3  9.0  14.5  14.0  13.4  19.2  18
# observed x values
    -1   0 1  2  3  4  5  6  7  8
```

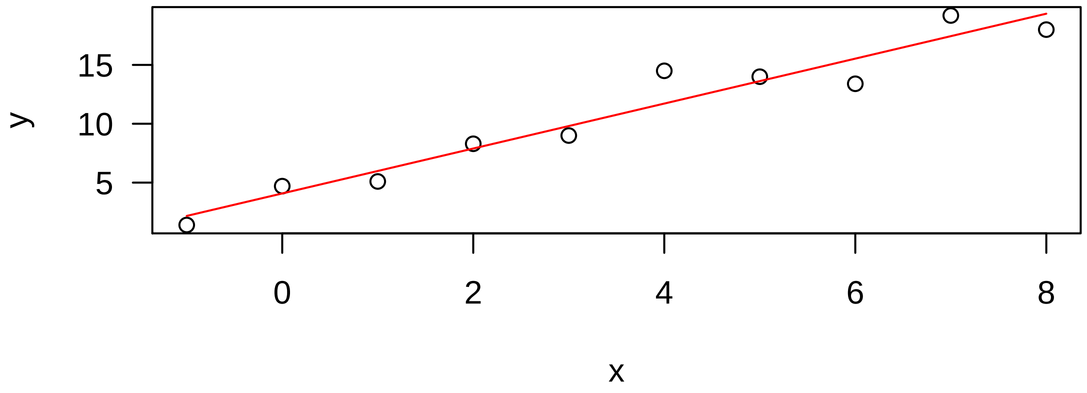
The more advanced can try to make a robust reggression, where the noise is distributed is a mixture between a normal and a t_1-distribution $p(x) = (1-p)\phi(x) + pt_1(x)$. Insert an artificial outlier and see if it works.

Solution: Linear regression via maximum likelihood

```
DATA_SECTION
  init_int N
  init_vector Y(1,N)
  init_vector x(1,N)

PARAMETER_SECTION
  init_number a
  init_number b
  init_number logSigma
  sdreport_number sigmasq
  objective_function_value nll

PROCEDURE_SECTION
  sigmasq=exp(2*logSigma);
  nll=0.5*(N*log(2*M_PI*sigmasq)+sum(square(Y-(a+b*x)))/sigmasq);
```



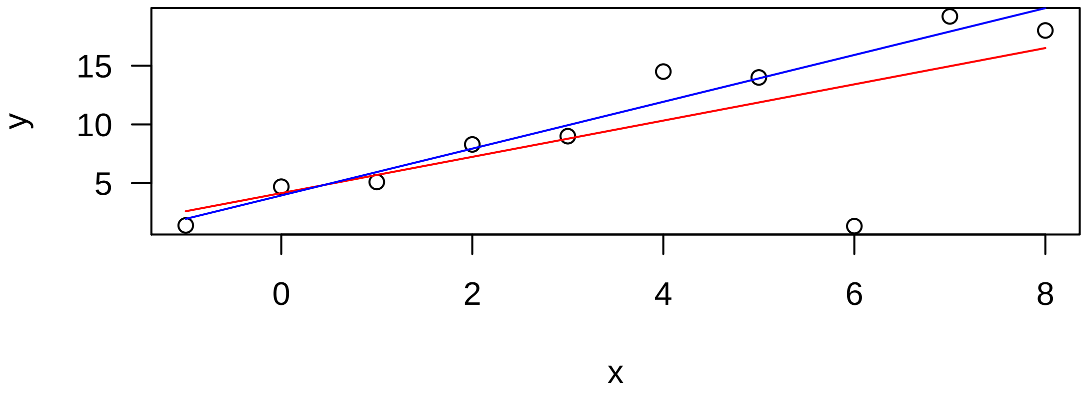
```
GLOBALS_SECTION
  #include <fvar.hpp>

  dvariable dnorm(const dvariable& x, const dvariable& mu=0.0, const dvariable& var=1.0){
    return 0.5*(log(2.0*M_PI*var)+square(x-mu)/var);
  }
  dvariable dt(const dvariable& x){
    return log(M_PI)+log(1.0+square(x));
  }
  dvariable dens(const dvariable& x, const dvariable& mu, const dvariable& v, const dvariable& p){
    dvariable z=(x-mu)/sqrt(v);
    return -log(1.0/sqrt(v)*((1.0-p)*exp(-dnorm(z))+p*exp(-dt(z))));
  }

DATA_SECTION
  init_int N
  init_vector Y(1,N)
  init_vector x(1,N)

PARAMETER_SECTION
  init_number a
  init_number b
  init_number logSigma
  init_bounded_number p(0,1)
  sdreport_number sigmasq
  objective_function_value nll

PROCEDURE_SECTION
  sigmasq=exp(2*logSigma);
  nll=0.0;
  for(int i=1; i<=N; ++i){
    nll+=dens(Y(i),a+b*x(i),sigmasq,p);
  }
```



A1: Complete program using the Poisson likelihood

```
DATA_SECTION
  !! random_number_generator rng(123456);
  vector X(1,1000);
  !! X.fill_randpoisson(5.0,rng);

PARAMETER_SECTION
  init_bounded_number lambda(0,100);
  objective_function_value nll;

PROCEDURE_SECTION
  int N=X.indexmax()-X.indexmin()+1;
  nll=lambda*N-log(lambda)*sum(X)+sum(gammln(X+1.0));
```

A2: Complete program using the normal likelihood

```
DATA_SECTION
  vector X(1,1000);
 LOC_CALCS
  random_number_generator rng(123456);
  X.fill_randn(rng);
  X*=5.0;
  X+=2.0;
 END_CALCS

PARAMETER_SECTION
  init_number logSigma;
  init_number mu;
  sdreport_number sigma;
  objective_function_value nll;

PROCEDURE_SECTION
  sigma=exp(logSigma);
  int N=X.indexmax()-X.indexmin()+1;
  dvariable ss=square(sigma);
  nll=0.5*(N*log(2*M_PI*ss)+sum(square(X-mu))/ss);
```

A3: Complete program using the binomial likelihood

```
DATA_SECTION
  !! random_number_generator rng(123456);
  int n;
  !! n=1000;
  vector N(1,n);
  vector X(1,n);
  !! N.fill_randpoisson(10,rng);
  !! for(int i=N.indexmin(); i<=N.indexmax(); ++i){
  !!   dvector tmp(1,(int)N(i));
  !!   tmp.fill_randbi(0.7,rng);
  !!   X(i)=sum(tmp);
  !! }

PARAMETER_SECTION
  init_bounded_number p(0,1);
  objective_function_value nll;

PROCEDURE_SECTION
  nll=-sum(log_comb(N,X))-log(p)*sum(X)-log(1-p)*sum(N-X);
```