Maximum likelihood estimation

AD Model Builder foundation

anders@nielsensweb.org

Strategy

- Most statistical software packages have a lot of built-in standard models, and a number of options
- Think about R (`lm`, `glm`, `aov`, `lme`, `nlme`, ...)
- Once in a while we need a model that is not standard
- In many cases we can get "close" with one of the standard models, but in AD Model Builder it is often feasible to do it right
- Here we will focus on
  - setting up your own likelihood function
  - minimize the negative log likelihood to get estimates of model parameters
  - finding uncertainties on estimated parameters
- Illustrated by a few examples

Reminder: Statistical model and likelihood function

- 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:

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

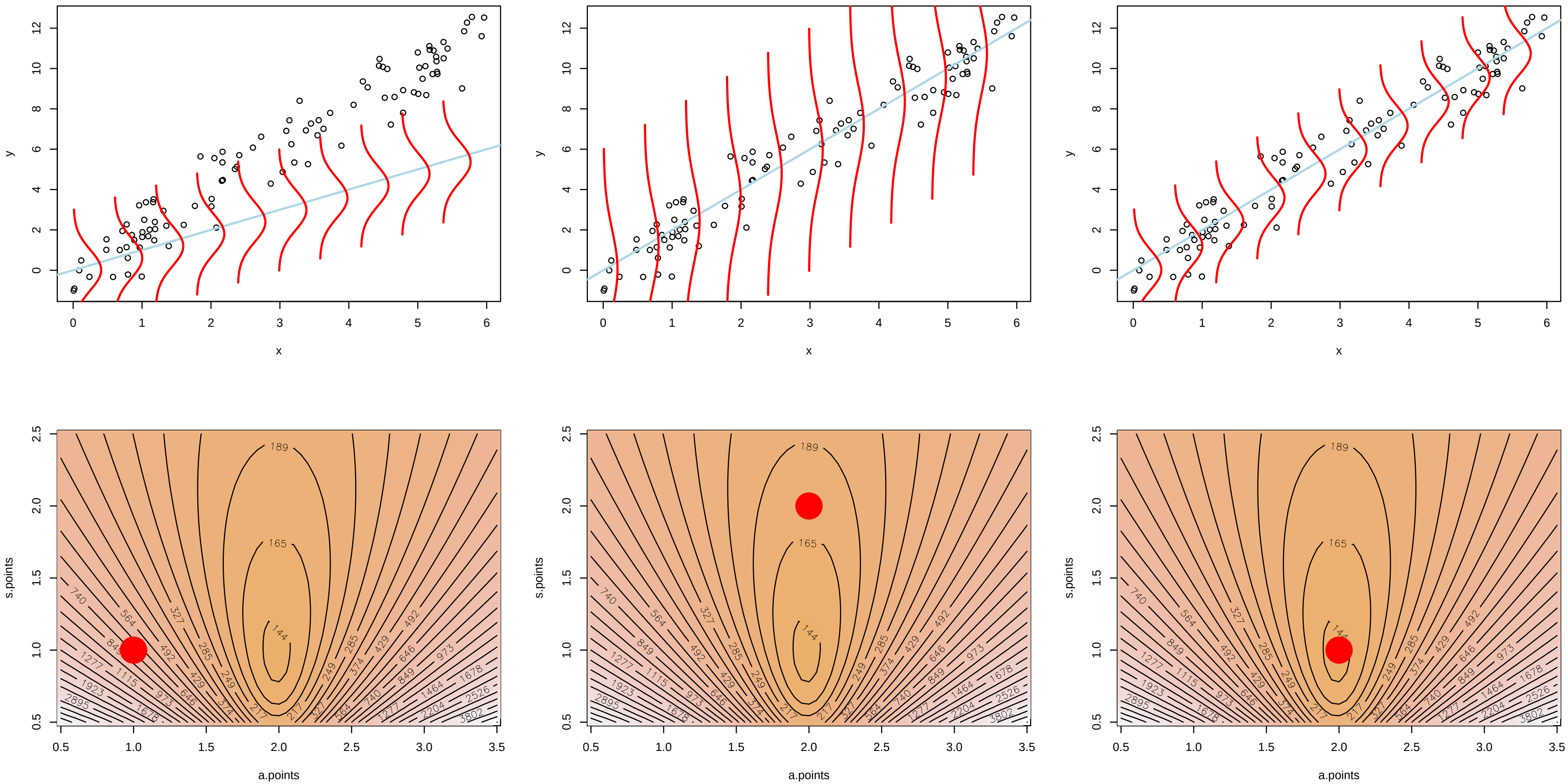
AD Model Builder program for linear regression

```
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 sigma;
  objective_function_value nll

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


How to choose (estimate) the model parmeters



Maximum likelihood estimator

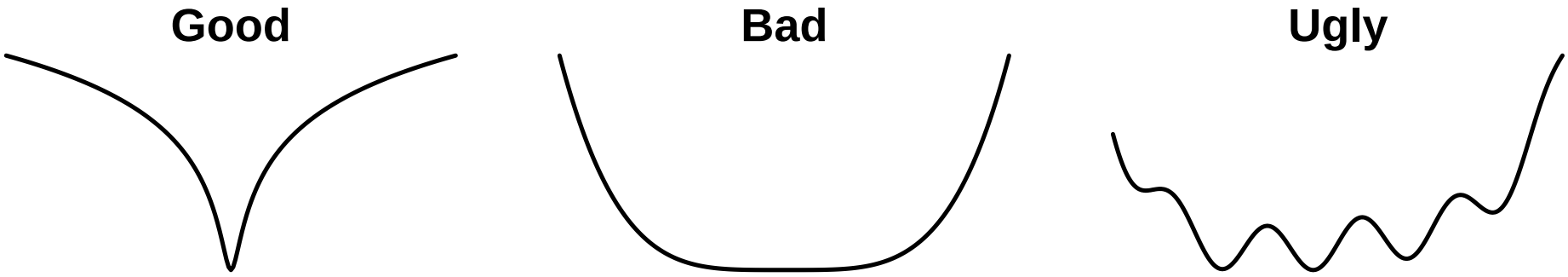
- A sensible estimate of the model parameters is to choose the values that maximize the likelihood for the actual observations.

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

- 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.



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:

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

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

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)

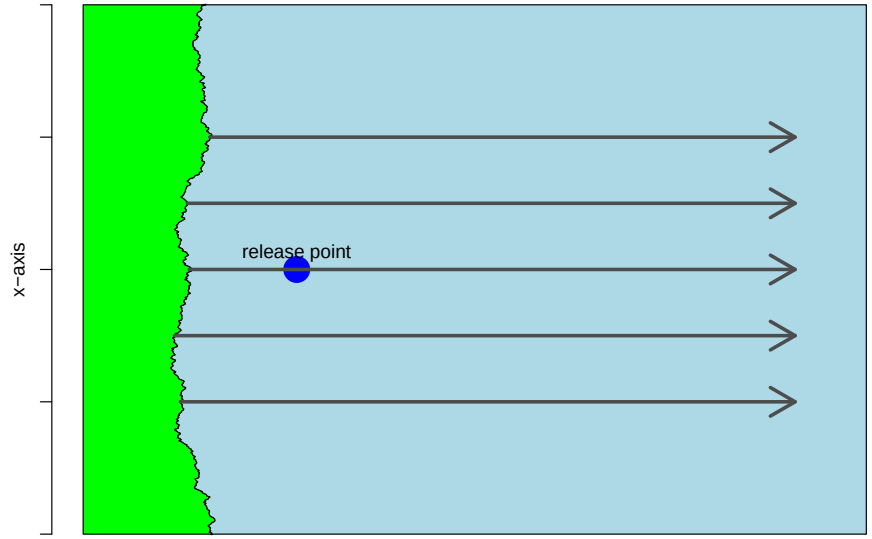
$$\ell(x|p) = -\sum \log \binom{N_i}{x_i} - \log(p)\sum x_i - \log(1-p)\sum(N_i - x_i)$$

```
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.

Example: Diffusion of fish from central release

- Part of a larger study shown here:
- Consider the following experiment:
  - Day 0 release of N=3529 fish at a central location (position=0)
  - Day 1–9 a number of fish are caught at different locations
  - Observations are the number caught in each trawl
- The table show the catch at different positions on day 2



	day	position	catch
1	2	80	14
2	2	40	31
3	2	-40	8
4	2	-80	4
5	2	-1	16
6	2	20	20
7	2	120	4
8	2	-120	4

- Assume all fish are independent and following:

$$dx_t^{(i)} = \alpha dt + \sigma dB_t^{(i)}, \quad \& \quad x_0^{(i)} = 0$$

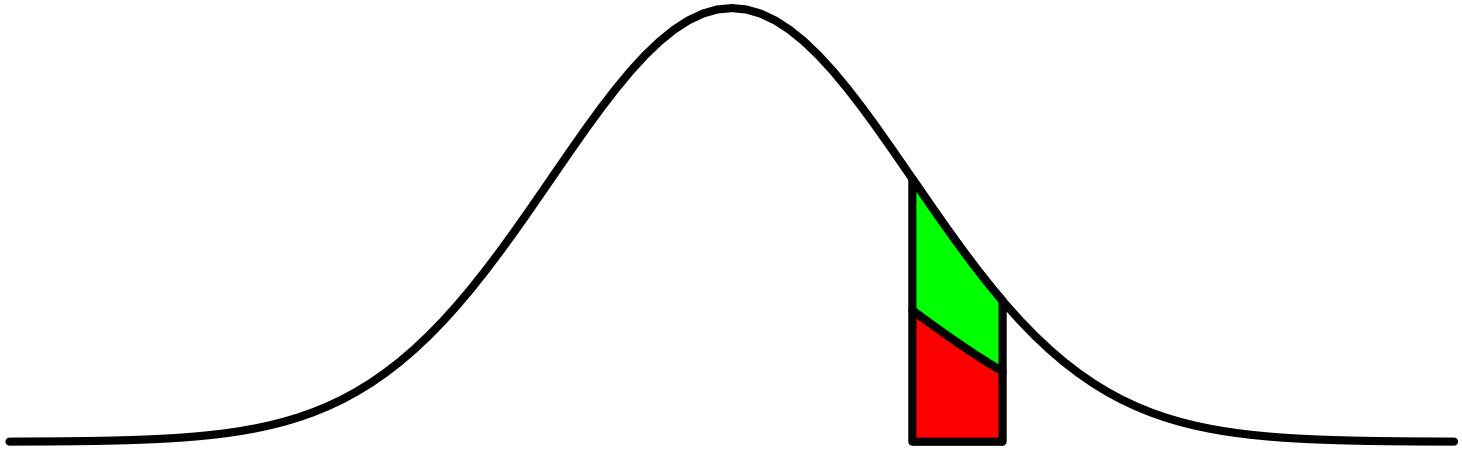
- The population at time t will follow a $\mathcal{N}(\alpha t, \sigma^2 t)$
- The expected number below a trawl with width w, at position p at time t will be:

$$\overline{N}_{w,p,t} = N_{0,0}\left(\Phi\left(\frac{p + w/2 - \alpha t}{\sqrt{\sigma^2 t}}\right) - \Phi\left(\frac{p - w/2 - \alpha t}{\sqrt{\sigma^2 t}}\right)\right)$$

- From this number a fraction q is caught. q reflects the efficiency of the trawl, the expected catch is:

$$\overline{C}_{w,p,t} = q\overline{N}_{w,p,t}$$

- If $w << \sqrt{\sigma^2 t}$ it is a fair to assume: $C_{w,p,t} \sim Pois(q\overline{N}_{w,p,t})$



- Now we can implement the likelihood:

```
sigma=exp(logSigma);
int from=cat.indexmin();
int to=cat.indexmax();
dvar_vector Nbar(from,to);
for(int i=from; i<=to; ++i){
  Nbar(i)=cumd_norm((pos(i) + w/2.0 - alpha * day(i))/sigma/sqrt(day(i)))-
          cumd_norm((pos(i) - w/2.0 - alpha * day(i))/sigma/sqrt(day(i)));
}
Nbar*=N00;
dvar_vector Cbar=q*Nbar;
nll=sum(Cbar-elem_prod(log(Cbar),cat)+gammln(cat+1.0));
}
```


- The entire ADMB program is:

```
DATA_SECTION
  init_matrix turbot(1,8,1,3);
  vector day(1,8);
  vector pos(1,8);
  vector cat(1,8);
  !! day=column(turbot,1); pos=column(turbot,2); cat=column(turbot,3);
  int N00;
  !! N00=3529;
  number w;
  !! w=4.5;
PARAMETER_SECTION
  init_number alpha;
  init_number logSigma;
  init_bounded_number q(0,1);
  sdreport_number sigma;
  objective_function_value nll;

PRELIMINARY_CALCS_SECTION
  logSigma=log(1000);

PROCEDURE_SECTION
  sigma=exp(logSigma);
  int from=cat.indexmin();
  int to=cat.indexmax();
  dvar_vector Nbar(from,to);
  for(int i=from; i<=to; ++i){
    Nbar(i)=cumd_norm((pos(i) + w/2.0 - alpha * day(i))/sigma/sqrt(day(i)))-
            cumd_norm((pos(i) - w/2.0 - alpha * day(i))/sigma/sqrt(day(i)));
  }
  Nbar*=N00;
  dvar_vector Cbar=q*Nbar;
  nll=sum(Cbar-elem_prod(log(Cbar),cat)+gammln(cat+1.0));
```


- The output from running the model is:

turbot.par

```
#Number of parameters = 3 Objective function value = 22.2888 Maximum gradient component = 1.93694e-07
# alpha:
12.5060367109
# logSigma:
3.78254702089
# q:
0.208329297828
```

turbot.cor

```
 The logarithm of the determinant of the hessian = 9.60987
 index   name        value       std dev        1        2        3        4
     1   alpha     1.2506e+01 3.9232e+00   1.0000
     2   logSigma  3.7825e+00 1.0374e-01   0.2404   1.0000
     3   q         2.0833e-01 2.2178e-02   0.1586   0.3469   1.0000
     4   sigma     4.3928e+01 4.5570e+00   0.2404   1.0000   0.3469   1.0000
```

turbot.std

```
 index   name       value       std dev
     1   alpha     1.2506e+01 3.9232e+00
     2   logSigma  3.7825e+00 1.0374e-01
     3   q         2.0833e-01 2.2178e-02
     4   sigma     4.3928e+01 4.5570e+00
```


Exercises

Exercise 1: 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

Estimate the two unknown parameters.

Exercise 2: In the 1D diffusion fish model we assumed:

$$C_{w,p,t} \sim Pois(\overline{C}_{w,p,t})$$

In the Poisson distribution the variance is equal to the mean, which is an assumption that is not always valid. In this exercise we will expand the model and test the assumption.

- Consider the model:

$$C \sim \text{Pois}(\lambda), \quad \text{where} \quad \lambda \sim \Gamma\left(n, \frac{1-\phi}{\phi}\right) \quad 0 < \phi < 1$$

- It can be shown that:

$$C \sim \text{Nbinom}(n, \phi)$$

- This negative binomial has mean $E(C) = n\frac{1-\phi}{\phi}$ and the variance is $E(C)/\phi$ (so greater than the mean)
- To extend the 1D diffusion fish model we use:

$$C_{w,p,t} \sim \text{Nbinom}\left(\overline{C}_{w,p,t} \cdot \left(\frac{\phi}{1-\phi}\right), \phi\right) \quad (\overline{C} \text{ is calculated as before.})$$

Modify the AD Model Builder program to estimate the four model parameters $\alpha, \sigma, q,$ and ϕ and compare the two models

Exercise 3: Assume the following is a sample from a multinomial distribution:

```
10  16  31  13  14  16
```

Write a program to estimate the unknown model parameters.

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
  !! random_number_generator rng(123456);
  vector X(1,1000);
  !! X.fill_randn(rng);
  !! X*=5.0;
  !! X+=2.0;

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);
```