Practical 1: Statistical modeling and likelihood

Benjamin Rosenbaum

Table of contents

We learn about the likelihood function and do some didactic examples of maximum likelihood estimation (MLE) with the `optim()` function.

```
rm(list=ls())
library("sfsmisc") # mathematical integration through data points
try(dev.off())
```

Example 1: Survival rate

Data from 3 habitats and their deer population.

We generate population size before and after the winter.

```
data = data.frame(total = c(18, 25, 20),
                  survived = c(9, 10, 8))
```

Question: What is the average survival rate?

Deterministic part: θ (just fitting a mean, or the intercept if you will)
Stochastic part: $survived \sim \text{Binomial}(total, \theta)$

Naively, we could just calculate rate for each habitat and compute their mean. But we can do better. Use the statistical model.

```
data$survived/data$total
```

```
[1] 0.5 0.4 0.4
```

```
mean(data$survived/data$total)
```

```
[1] 0.4333333
```

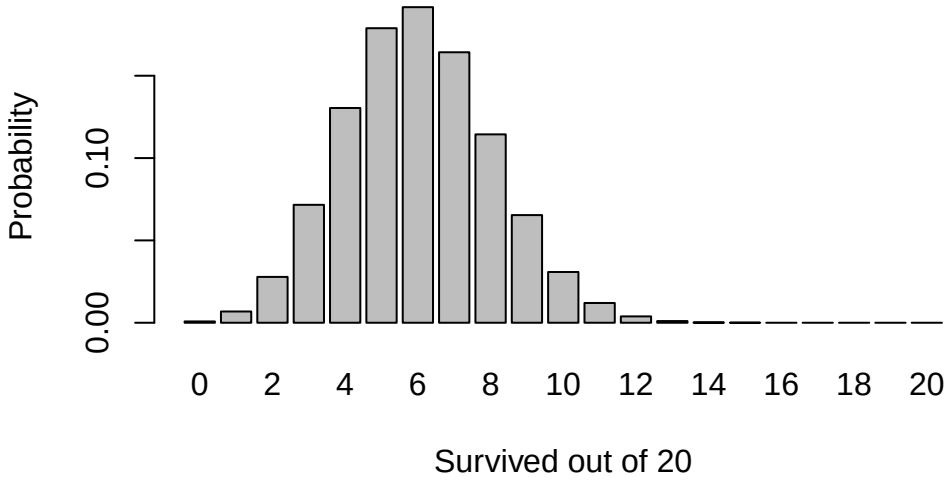
Probability density function $p(y|\theta)$

If we know the survival rate, we can compute probability for each outcome

```
x = 0:20
y = dbinom(x=x, size=20, prob=0.3)
sum(y)
```

```
[1] 1
```

```
barplot(y~x, xlab="Survived out of 20", ylab="Probability")
```

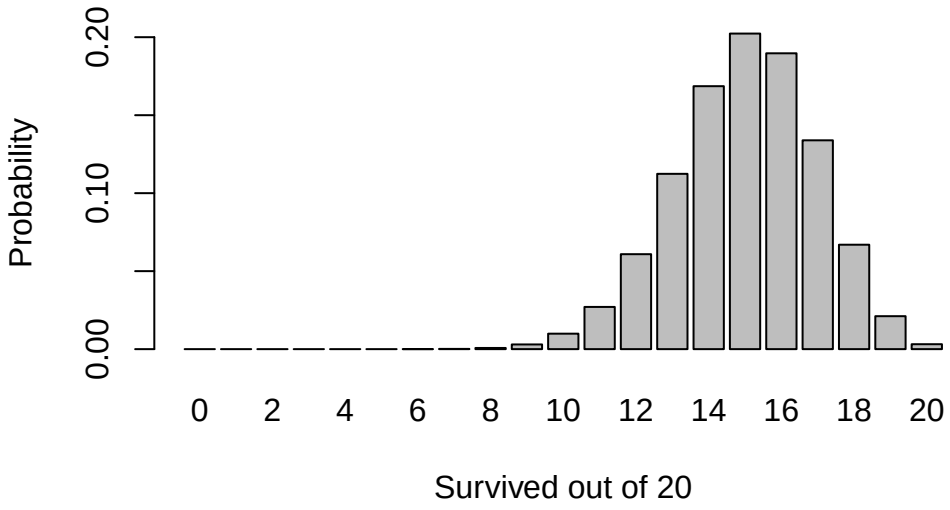


Same for a different value of survival rate

```
x = 0:20
y = dbinom(x=x, size=20, prob=0.75)
sum(y)
```

```
[1] 1
```

```
barplot(y~x, xlab="Survived out of 20", ylab="Probability")
```



Likelihood $L(\theta) = p(\theta|y)$

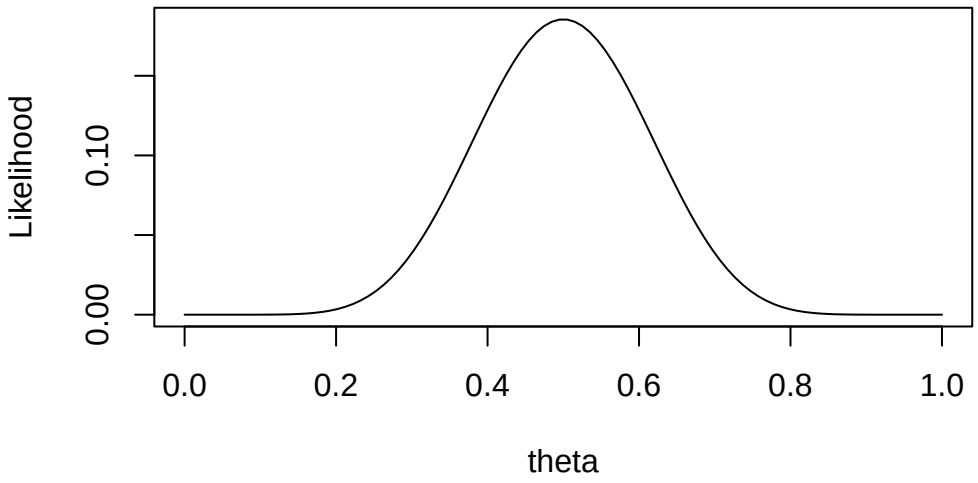
Likelihood is probability in reverse. For a given datapoint, likelihood is the probability of that observation as a function of parameter value (survival probability θ).

```
theta = seq(0,1, length.out=100)
head(theta)
```

```
[1] 0.00000000 0.01010101 0.02020202 0.03030303 0.04040404 0.05050505
```

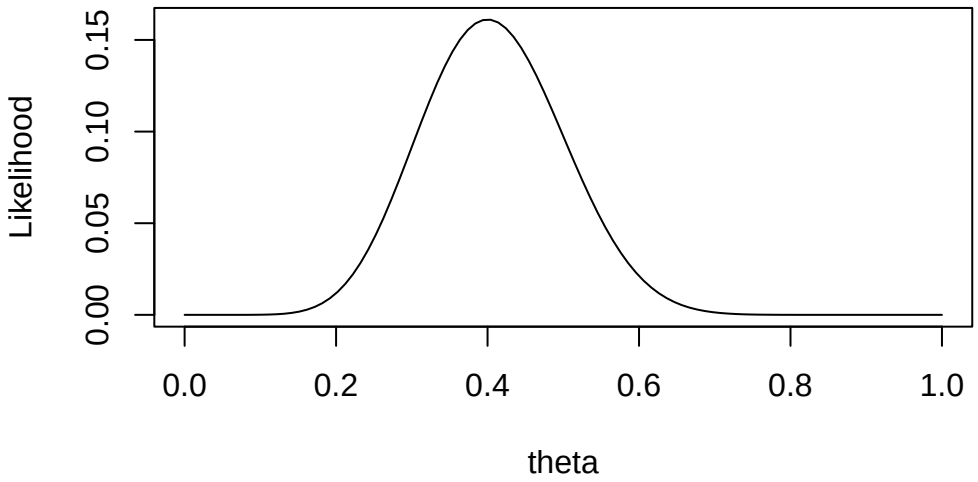
Likelihood function for parameter theta and 1st datapoint

```
lik = dbinom(x=9, size=18, prob=theta)
plot(theta, lik, type="l", xlab="theta", ylab="Likelihood")
```



Likelihood function for parameter theta and 2nd datapoint

```
lik = dbinom(x=10, size=25, prob=theta)
plot(theta, lik, type="l", xlab="theta", ylab="Likelihood")
```



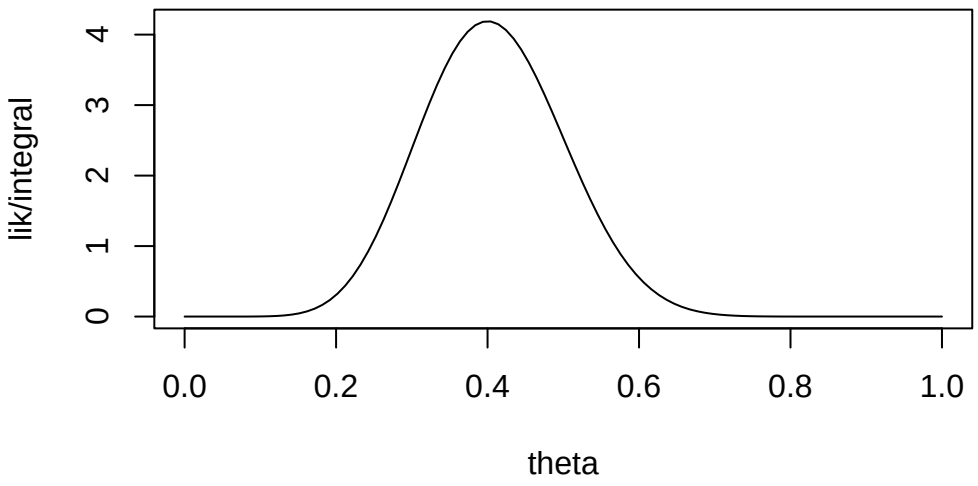
Attention: Likelihood $L(\theta)$ is not a probability density function for theta. It does not integrate to 1. This means we can use it for maximum likelihood, but not for statistical inference without knowing the full integral.

```
integral = integrate.xy(theta, lik)
integral
```

```
[1] 0.03846154
```

However, if we scale (divide) $L(\theta)$ by its integral, this new function integrates to 1. But this is unfeasible in most applications (>1 parameter).

```
plot(theta, lik/integral, type="l", xlab="theta")
```

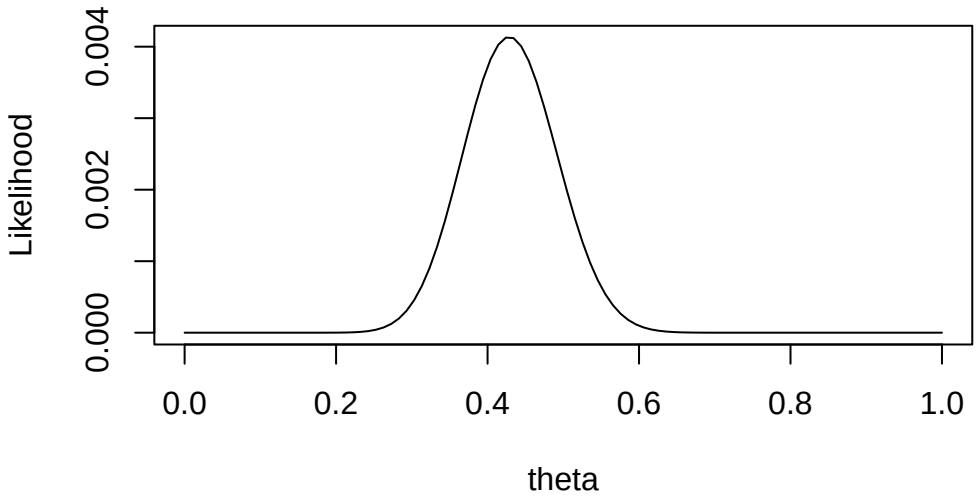



```
integrate.xy(theta, lik/integral)
```

```
[1] 1
```

OK, so far we only calculated likelihood of single datapoints. Likelihood function for the whole dataset is the product of these likelihoods.

```
lik = dbinom(x=data$survived[1], size=data$total[1], prob=theta)*
  dbinom(x=data$survived[2], size=data$total[2], prob=theta)*
  dbinom(x=data$survived[3], size=data$total[3], prob=theta)
plot(theta, lik, type="l", xlab="theta", ylab="Likelihood")
```



Maximum likelihood means finding the parameter value for which the observed data is most likely to have occurred. Here we use GLM to find that value.

```
model1 = glm( cbind(survived, total-survived) ~ 1, data=data,
              family=binomial)
summary(model1)
```

```
Call:
glm(formula = cbind(survived, total - survived) ~ 1, family = binomial,
    data = data)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.2877     0.2546   -1.13    0.258

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 0.52207  on 2  degrees of freedom
Residual deviance: 0.52207  on 2  degrees of freedom
AIC: 12.975

Number of Fisher Scoring iterations: 3
```

What is happening? GLM estimates a negative survival probability? No, the binomial family uses a log-link as default. We can override the default by specifying an identity link function (parameter is estimated on its original scale). More on that in Lesson 5.

```r
model1 = glm( cbind(survived, total-survived) ~ 1, data=data,
              family=binomial(link="identity"))
summary(model1)
```

```
Call:
glm(formula = cbind(survived, total - survived) ~ 1, family = binomial(link = "identity"),
    data = data)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.42857    0.06235   6.874 6.25e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 0.52207  on 2  degrees of freedom
Residual deviance: 0.52207  on 2  degrees of freedom
AIC: 12.975

Number of Fisher Scoring iterations: 3
```

```r
confint(model1)
```

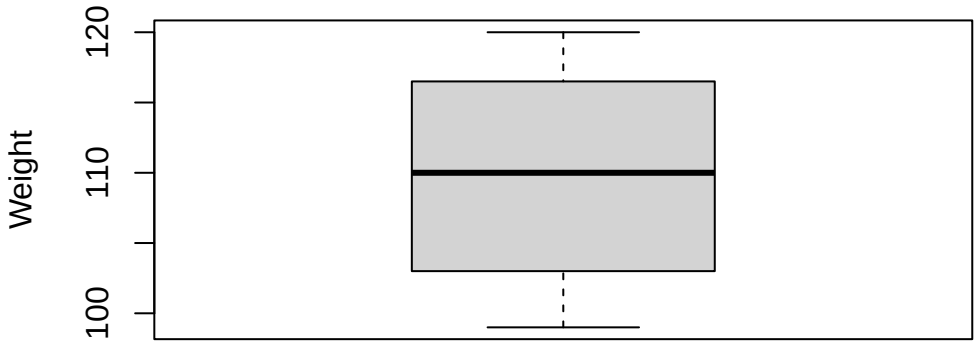
```
    2.5 %    97.5 %
0.3110495 0.5517955
```

Here, we get a mean survival prob of 0.43 with 95%-CI [0.31, 0.55]. CIs are not computed from the full likelihood, but rely on a local approximation.

Example 2: Mean body size of a mammal population

Now, bodysize is a continuous response. Generate the weight of 7 individuals.

```r
data = data.frame(weight = c(104, 120, 118, 115, 99, 110, 102))
boxplot(data$weight, ylab="Weight")
```



Question: What's the mean bodysize?

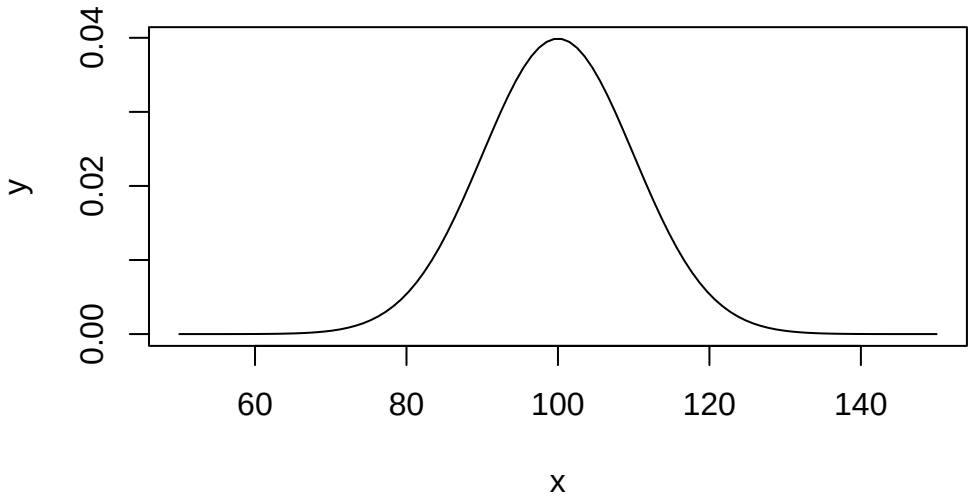
Deterministic part: μ (just fitting a mean, or the intercept if you will)
Stochastic part: $weight \sim \text{Normal}(\mu, \sigma)$

First with known σ. Estimate just 1 parameter μ

```
sigma=10
```

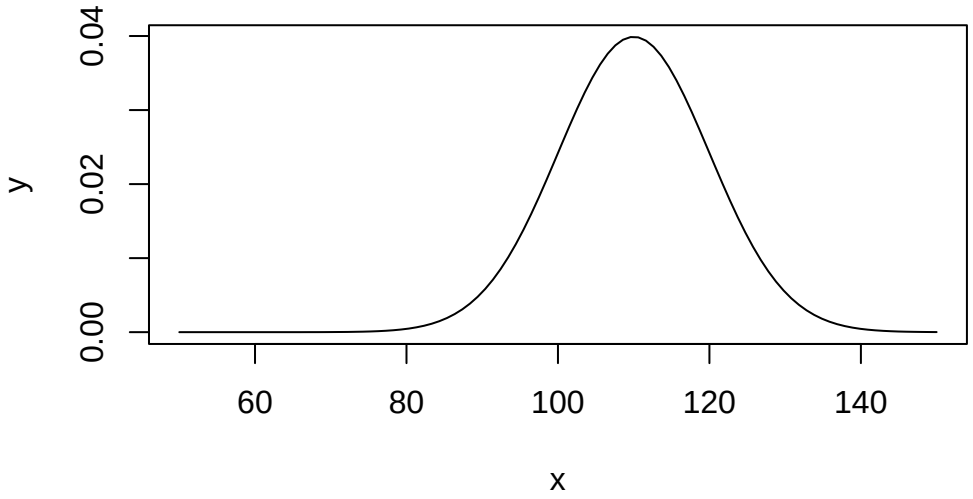
Probability for given μ and σ

```
x = seq(50, 150, length.out=100)
y = dnorm(x=x, mean=100, sd=sigma)
plot(x,y, type="l")
```



another mean

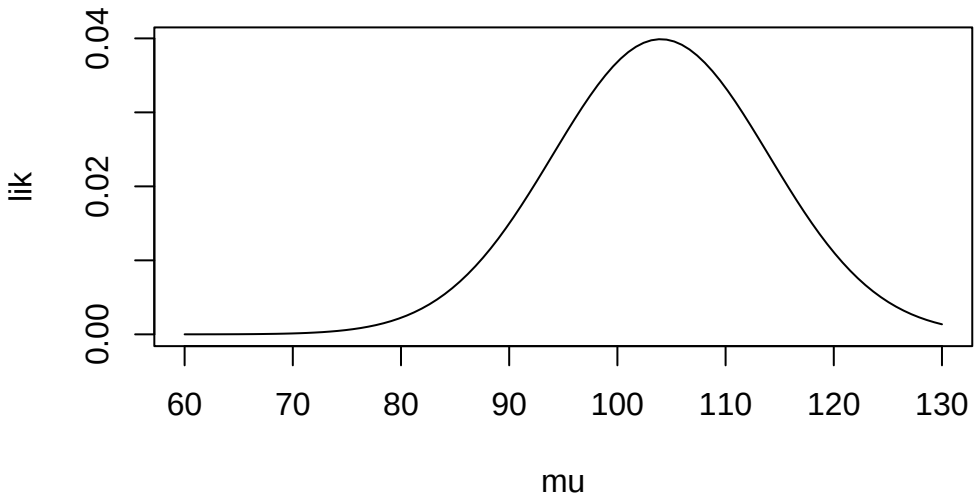
```
y = dnorm(x=x, mean=110, sd=sigma)
plot(x,y, type="l")
```



likelihood = probability density of data for a parameter

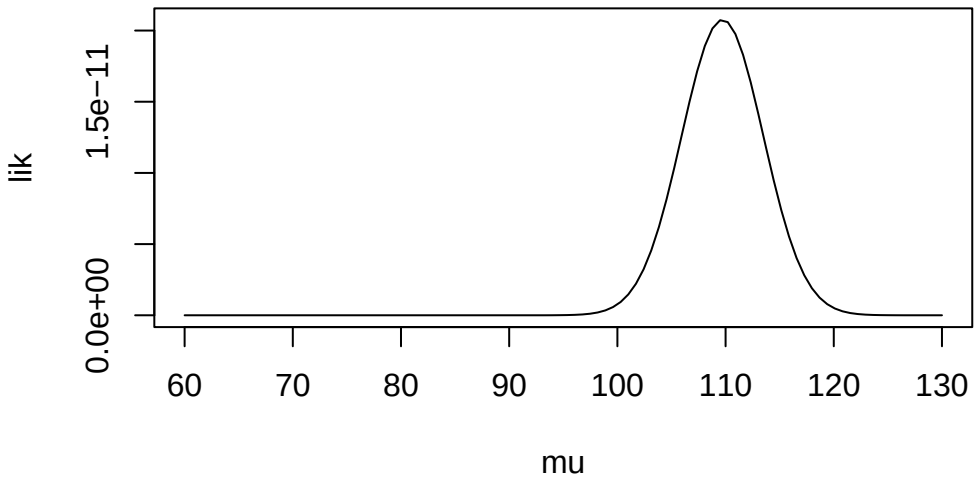
likelihood function for 1 datapoint

```
mu = seq(60, 130, length.out=100)
lik = dnorm(data$weight[1], mean=mu, sd=sigma)
plot(mu, lik, type="l")
```



likelihood function for all datapoints

```
lik = dnorm(data$weight[1], mean=mu, sd=sigma)*
  dnorm(data$weight[2], mean=mu, sd=sigma)*
  dnorm(data$weight[3], mean=mu, sd=sigma)*
  dnorm(data$weight[4], mean=mu, sd=sigma)*
  dnorm(data$weight[5], mean=mu, sd=sigma)*
  dnorm(data$weight[6], mean=mu, sd=sigma)*
  dnorm(data$weight[7], mean=mu, sd=sigma)
plot(mu, lik, type="l")
```



In our statistical model, there is a second parameter σ. Likelihood is a function of both parameters $L(\mu, \sigma)$

Write likelihood as a function but use negative **log**-likelihood (NLL). This makes things easier since L can be VERY small and numerically unstable

```
likelihood = function(parameters, weights){
  lik = dnorm(weights, mean=parameters[1], sd=parameters[2], log=TRUE)
  return(-sum(lik))
}
```

Try to minimize the NLL by guessing

```
likelihood(c(110,10), data$weight)
```

```
[1] 24.60067
```

```r
likelihood(c(111,10), data$weight)
```

```
[1] 24.65567
```

```r
likelihood(c(109,10), data$weight)
```

```
[1] 24.61567
```

```r
likelihood(c(108,10), data$weight)
```

```
[1] 24.70067
```

```r
likelihood(c(109,10), data$weight)
```

```
[1] 24.61567
```

```r
likelihood(c(109,11), data$weight)
```

```
[1] 24.92445
```

```r
likelihood(c(109,9), data$weight)
```

```
[1] 24.36252
```

```r
likelihood(c(109,8), data$weight)
```

```
[1] 24.21522
```

```r
likelihood(c(109,7), data$weight)
```

```
[1] 24.26823
```

```r
likelihood(c(109,8), data$weight)
```

```
[1] 24.21522
```

```
likelihood(c(110,8), data$weight)
```

```
[1] 24.19179
```

```
likelihood(c(108,8), data$weight)
```

```
[1] 24.34804
```

```
likelihood(c(109,8), data$weight)
```

```
[1] 24.21522
```

We can do better, with an iterative algorithm to search for optimum (μ, σ) with initial guess (100,10) using `optim()`

```
ml = optim(fn = likelihood,
           par = c(100,10),
           weights = data$weight) # the data
ml
```

```
$par
[1] 109.715724   7.647299

$value
[1] 24.17355

$counts
function gradient
      53       NA

$convergence
[1] 0

$message
NULL
```

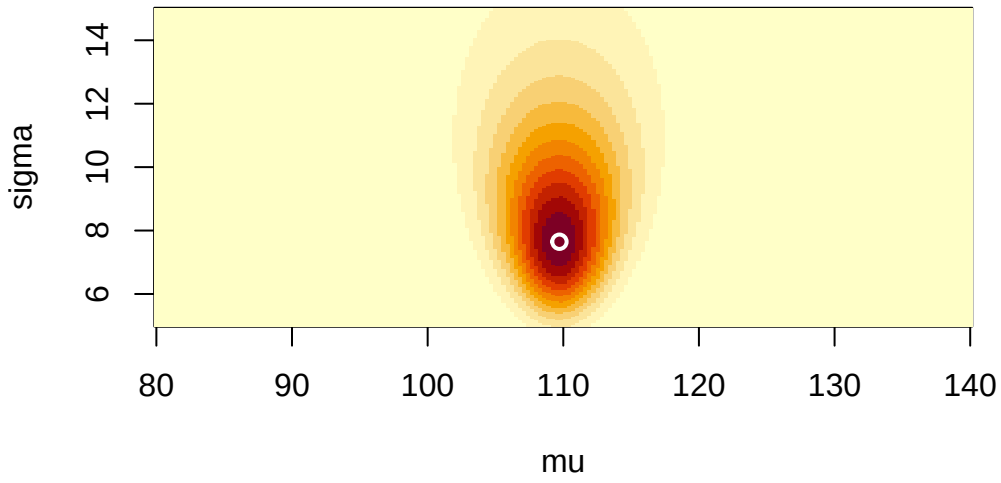
lm-solution to compare

```r
model2 = lm(weight ~ 1, data=data)
summary(model2)
```

```
Call:
lm(formula = weight ~ 1, data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-10.7143  -6.7143   0.2857   6.7857  10.2857

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  109.714      3.122   35.14 3.54e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.261 on 6 degrees of freedom
```

visualize 2-dimensional likelihood and plot ML solution

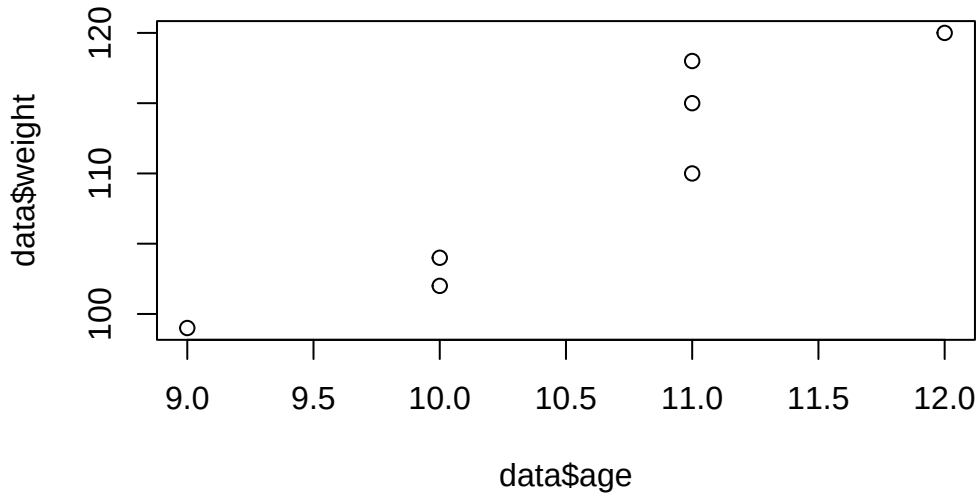
```r
mu.plot = seq(from=80, to=140, length.out=200)
sigma.plot = seq(from=5, to=15, length.out=200)

test = expand.grid(mu=mu.plot, sigma=sigma.plot)
test$lik = NA
for(i in 1:nrow(test)){
  test$lik[i] = likelihood(c(test$mu[i], test$sigma[i]), data$weight)
}
lik.plot = matrix(test$lik, nrow=length(mu.plot), ncol=length(sigma.plot))

image(mu.plot, sigma.plot, exp(-lik.plot),
      xlab="mu", ylab="sigma")# , col = hcl.colors(10, "terrain"))
points(ml$par[1], ml$par[2], col="white", lwd=2)
```



Example 3: body size vs age

Now we have a continuous predictor age. Generate data for 7 individuals.

```
data = data.frame(weight = c(104, 120, 118, 115, 99, 110, 102),
                  age    = c(10, 12, 11, 11, 9, 11, 10))
plot(data$age, data$weight)
```



Question: What's the average growth per year? (Slope in age)

Deterministic part: $\mu = a + b \cdot age$
Stochastic part: $weight \sim \text{Normal}(\mu, \sigma)$

We could easily fit this model with LM

```r
model3 = lm(weight ~ age, data=data)
summary(model3)
```

```
Call:
lm(formula = weight ~ age, data = data)

Residuals:
   1    2    3    4    5    6    7
-1.2 -1.0  4.9  1.9  1.7 -3.1 -3.2

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   26.200     14.423   1.817  0.12899
age            7.900      1.359   5.811  0.00213 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.25 on 5 degrees of freedom
Multiple R-squared:  0.871, Adjusted R-squared:  0.8452
F-statistic: 33.77 on 1 and 5 DF,  p-value: 0.002129
```

```r
coef(model3)
```

```
(Intercept)         age
       26.2         7.9
```

```r
intercept = coef(model3)[1]
slope = coef(model3)[2]
```

But for didactic purposes, we're doing it the hard way. Maximum likelihood. We code the likelihood function (probability density of all datapoints for a given set of parameters a, b, σ)

```r
likelihood=function(parameters, weights, ages){
  lik = dnorm(weights,
              mean=parameters[1] + parameters[2]*ages,
              sd=rep(parameters[3],length(weights)),
              log=TRUE)
  return(-sum(lik)) # negative log likelihood NLL
}
```

This is the lm-solution and its likelihood. Other solutions fit the data worse and have a higher NLL = lower likelihood than the lm-solution

```
likelihood(c(intercept, slope, 3.25), data$weight, data$age)
```

```
[1] 17.18256
```

```
likelihood(c(26, 7.5, 3.25), data$weight, data$age)
```

```
[1] 23.72457
```

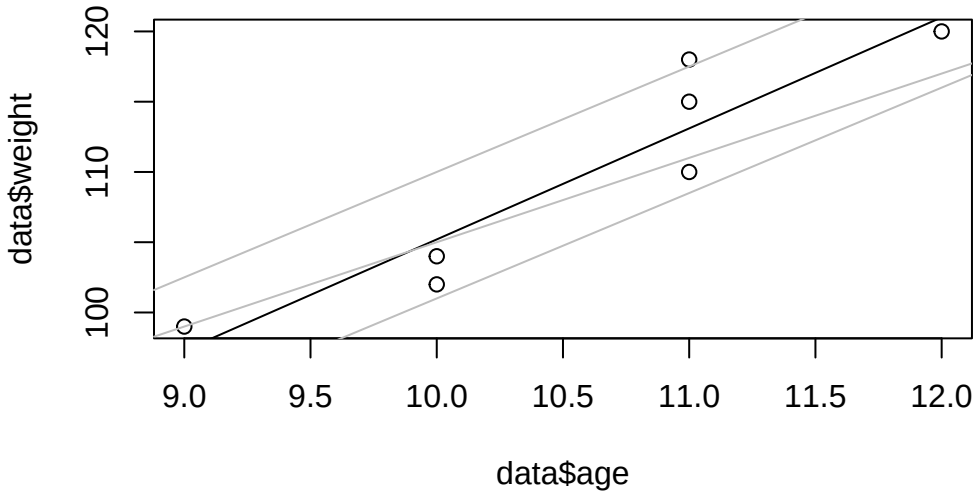
```
likelihood(c(35, 7.5, 3.25), data$weight, data$age)
```

```
[1] 24.15061
```

```
likelihood(c(45, 6.0, 3.25), data$weight, data$age)
```

```
[1] 18.70682
```

```
plot(data$age, data$weight)
abline(intercept, slope)
abline(26, 7.5, col="grey")
abline(35, 7.5, col="grey")
abline(45, 6.0, col="grey")
```



Max-likelihood: iterative algorithm to search for optimum a, b, σ with initial guess (100,5,8)

```r
ml = optim(fn = likelihood,
           par = c(100,5,8),
           weights = data$weight, # data
           ages = data$age)       # data
ml
```

```
$par
[1] 26.193382  7.900498  2.745227

$value
[1] 17.00468

$counts
function gradient
     172       NA

$convergence
[1] 0

$message
NULL
```

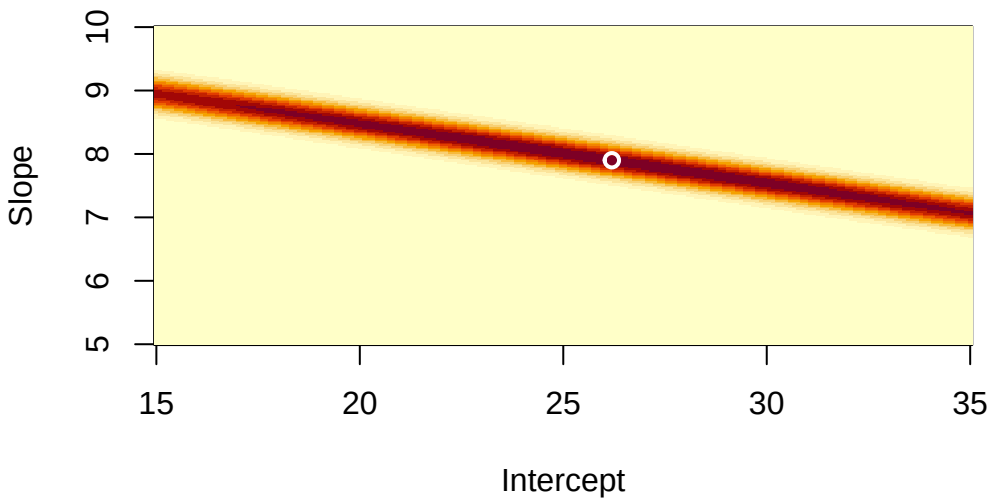
That's VERY close to the lm-solution

Visualize 2-dimensional likelihood (intercept & slope), σ fixed. Plot ML solution

```r
int.plot = seq(from=15, to=35, length.out=200)
slope.plot = seq(from=5, to=10, length.out=200)
test = expand.grid(int=int.plot, slope=slope.plot)
test$lik = NA
head(test)
```

```
       int slope lik
1 15.00000     5  NA
2 15.10050     5  NA
3 15.20101     5  NA
4 15.30151     5  NA
5 15.40201     5  NA
6 15.50251     5  NA
```

```r
for(i in 1:nrow(test)){
  test$lik[i] = likelihood(c(test$int[i], test$slope[i], 5.0), data$weight, data$age)
}
lik.plot = matrix(test$lik, nrow=length(int.plot), ncol=length(slope.plot))
image(int.plot, slope.plot, exp(-lik.plot),
      xlab="Intercept", ylab="Slope")# , col = hcl.colors(10, "terrain"))
points(ml$par[1], ml$par[2], col="white", lwd=2)
```



There is a strong negative correlation between Intercept and Slope. This is common if the predictor is not centered (mean=0). In extreme cases, that can cause numerical problems and wrong ML solutions.

Example 4: body size vs age (centered)

We repeat the same analysis, but this time the predictor is centered

```r
data = data.frame(weight = c(104, 120, 118, 115, 99, 110, 102),
                  age    = c(10, 12, 11, 11, 9, 11, 10))
data$agecenter = data$age-mean(data$age)
plot(data$agecenter, data$weight)

model4 = lm(weight ~ agecenter, data=data)
summary(model4)
```

```
Call:
```

```
lm(formula = weight ~ agecenter, data = data)

Residuals:
   1    2    3    4    5    6    7
-1.2 -1.0  4.9  1.9  1.7 -3.1 -3.2

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  109.714      1.228  89.326 3.33e-09 ***
agecenter      7.900      1.359   5.811  0.00213 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.25 on 5 degrees of freedom
Multiple R-squared:  0.871, Adjusted R-squared:  0.8452
F-statistic: 33.77 on 1 and 5 DF,  p-value: 0.002129
```

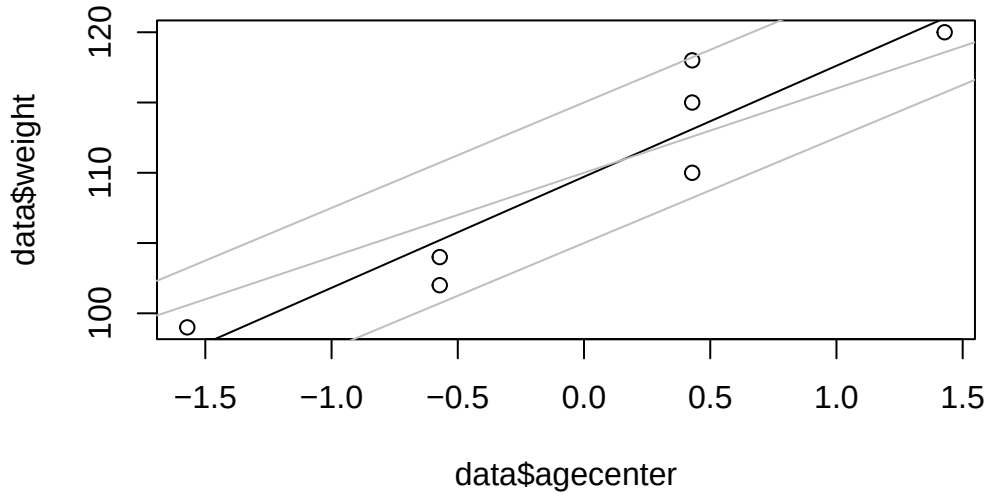
```
coef(model4)
```

```
(Intercept)   agecenter
   109.7143      7.9000
```

```
intercept = coef(model4)[1]
slope = coef(model4)[2]

likelihood=function(parameters, weights, ages){
  lik = dnorm(weights,
              mean=parameters[1] + parameters[2]*ages,
              sd=rep(parameters[3],length(weights)),
              log=TRUE)
  return(-sum(lik))
}

likelihood(c(intercept, slope, 3.25), data$weight, data$agecenter)
```

```
[1] 17.18256
```

```
plot(data$agecenter, data$weight)
abline(intercept, slope)
abline(105, 7.5, col="grey")
abline(115, 7.5, col="grey")
abline(110, 6.0, col="grey")
```




```
ml = optim(fn = likelihood,
           par = c(100,5,8),
           weights = data$weight, # data
           ages = data$agecenter) # data
ml
```

```
$par
[1] 109.712504   7.900602   2.746666

$value
[1] 17.00468

$counts
function gradient
     116       NA

$convergence
[1] 0

$message
NULL
```

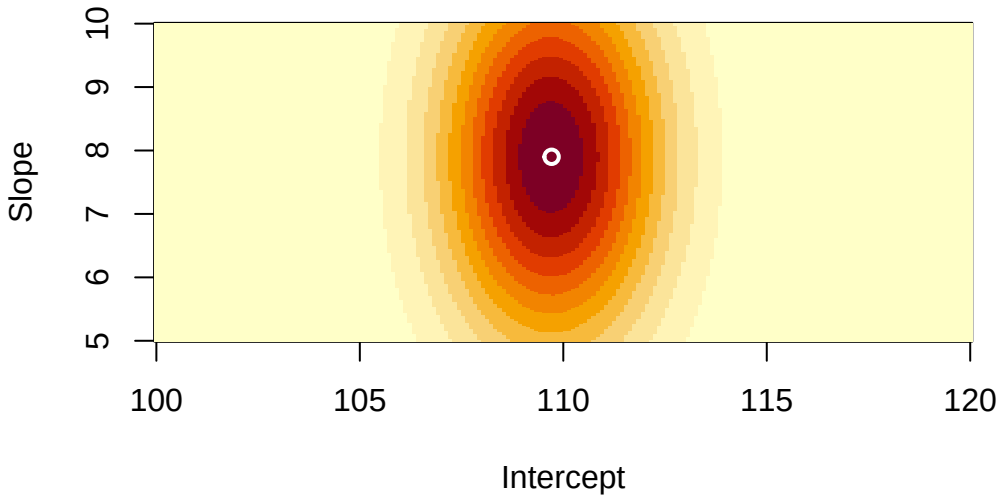
Visualize 2-dimensional likelihood (intercept & slope), σ fixed

```
int.plot = seq(from=100, to=120, length.out=200)
slope.plot = seq(from=5, to=10, length.out=200)
test = expand.grid(int=int.plot, slope=slope.plot)
```

```
test$lik = NA
head(test)
```

```
       int slope lik
1 100.0000     5  NA
2 100.1005     5  NA
3 100.2010     5  NA
4 100.3015     5  NA
5 100.4020     5  NA
6 100.5025     5  NA
```

```
for(i in 1:nrow(test)){
  test$lik[i] = likelihood(c(test$int[i], test$slope[i], 5.0), data$weight, data$agecenter)
}
lik.plot = matrix(test$lik, nrow=length(int.plot), ncol=length(slope.plot))

image(int.plot, slope.plot, exp(-lik.plot),
      xlab="Intercept", ylab="Slope")# , col = hcl.colors(10, "terrain"))
points(ml$par[1], ml$par[2], col="white", lwd=2)
```



By using a centered predictor, the correlation between intercept & slope is resolved.

Exercise 5: two predictors

Now we have more data and a second predictor for avg annual temperature. Write a model that includes both predictors for LM and ML. Test different initial guesses for the optim function and see if they all converge to the same result.

```r
data = data.frame(weight = c(65,80,129,41,77,133,75,75,87,109,136,134,91,49,127,120,108,126,8
                  age    = c(7,7,14,6,10,15,9,8,10,13,14,15,9,7,15,13,13,13,7,12),
                  temperature = c(5.9,6.9,1.5,9.6,9.0,6.9,8.0,10.2,4.8,7.6,6.2,11.2,7.3,11.4

model5 = lm(weight ~ age+temperature, data=data)
summary(model5)
```

```
Call:
lm(formula = weight ~ age + temperature, data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-13.1864  -7.3443  -0.4815   6.2809  15.6250

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  10.5811    12.3765   0.855    0.404
age           8.5746     0.7572  11.325 2.43e-09 ***
temperature  -0.8169     0.9275  -0.881    0.391
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.854 on 17 degrees of freedom
Multiple R-squared:  0.8998,    Adjusted R-squared:  0.888
F-statistic: 76.32 on 2 and 17 DF,  p-value: 3.219e-09
```

```r
likelihood=function(pars, df){
  lik = dnorm(df$weight,
              mean=pars[1] + pars[2]*df$age + pars[3]*df$temperature,
              sd=rep(pars[4],nrow(df)),
              log=TRUE)
  return(-sum(lik))
}

ml = optim(fn=likelihood, par=c(100,5,0,8), df=data)
ml$par
```

```
[1] 10.5594067  8.5795082 -0.8206534  9.0902816
```

```
ml = optim(fn=likelihood, par=c(20,1,1,10), df=data)
ml$par
```

```
[1] 10.5764548  8.5744169 -0.8169554  9.0831255
```

```
ml = optim(fn=likelihood, par=c(200,10,10,10), df=data)
ml$par
```

```
[1] 10.5576618  8.5724384 -0.8132537  9.0897939
```

Here, all initial guesses converge to the same solution (more or less) with the maximum likelihood approach.