ML estimation - Introduction & estimation uncertainty

FPA2020 Lecture 4 and 5

Toshihide Kitakado

May 27, 2020

Contents

1 Brief introdcution of optimization

Just in case, we shall begin with an introduction of optimization method. Here, we we will use "optim" function in R.

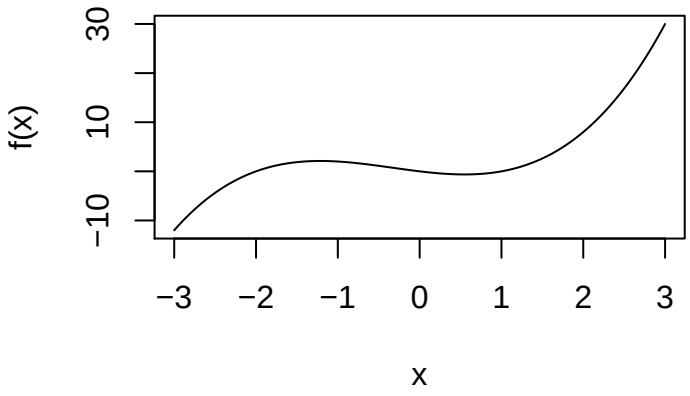
1.1 Definition of convenient functions in advance

```
logit <- function(p) log(p/(1-p))
logitinv <- function(x) 1/(1+exp(-x))
```

1.2 Optimization of a simple function

Procedure: - First you need to define a function for which you want to optimize (minimize)
- If you can draw a rough shape, that would be great - Use "optim" with an initial value or vector (method="BFGS" is for the quasi-Newton method)

```
f <- function(x){x^3 + x^2 - 2*x }
x <- seq(-3,3,0.01)
plot(x, f(x), type="l")
```




```
#Minimization
#Syntax: optim(Initial value, function name, optimization method)
optim(-2, f, method="BFGS")
```

```
$par
[1] -2.612088e+21

$value
[1] -1.782228e+64

$counts
function gradient
       6        6

$convergence
[1] 0

$message
NULL
```

```
optim(10, f, method="BFGS")
```

```
$par
[1] -1.775499e+23

$value
[1] -5.597078e+69
```

```
$counts
function gradient
       5        5

$convergence
[1] 0

$message
NULL
```

```
#optim(1, f, method="BFGS")
```

The function is to find a local minima, so the answer that you will get depends on the initial value.

1.3 Optimization of a simple function in a limited domain

If you want to give a constraint for the range of optimization, you can use an option of 'method="L-BFGS-B"' with upper and lower bounds.

```
#Optimization in a domain (0,1)
optim(-2, f, method="L-BFGS-B", lower=0, upper=1)
```

```
$par
[1] 0.5485836

$value
[1] -0.6311303

$counts
function gradient
       7        7

$convergence
[1] 0

$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

Alternatively, you can transform your original parameter to that takes any real number as follows:

```
#Optimization in a domain (0,1)
f <- function(logitx){
x <- 1/(1+exp(-logitx))
x^3 + x^2 - 2*x
}
res <- optim(-5, f, method="BFGS")
res$par
```

```
[1] 0.1949502
```

```
exp(res$par) /(1+exp(res$par))
```

```
[1] 0.5485838
```

2 ML estimation for simple binomial case

2.1 Just in case.... definition of binomial distribution

2.1.1 Definition

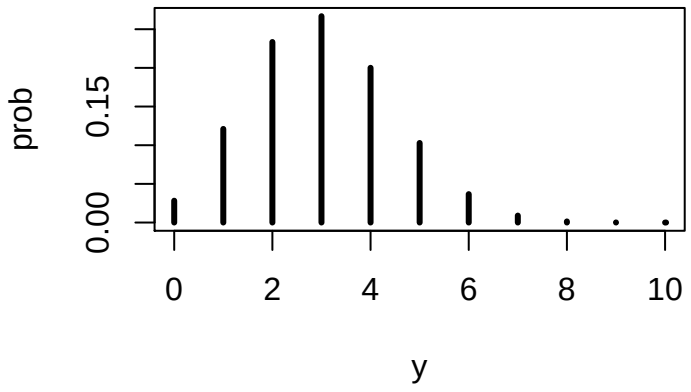
$$P(Y=y) = \binom{N}{y} p^y (1-p)^{N-y}, \quad y = 0, 1, \ldots, N \quad (0 \leq p \leq 1)$$

$$\begin{aligned} E[Y] &= Np \\ V[Y] &= Np(1-p) \end{aligned}$$

```
N <- 10
p <- 0.3
y <- seq(0, N, 1)
prob <- dbinom(y, N, p); prob
```

```
 [1] 0.0282475249 0.1210608210 0.2334744405 0.2668279320 0.2001209490
 [6] 0.1029193452 0.0367569090 0.0090016920 0.0014467005 0.0001377810
[11] 0.0000059049
```

```
plot(y, prob, type="h", lwd=3)
```



2.1.2 Generation of simulatio data

We suppose we will obtain 20 samples as

$$Y_1, Y_2, \ldots, Y_n \sim (iid) Bin(N, p).$$

```
set.seed(2020)
ns <- 20
yobs <- rbinom(ns, N, p); yobs
```

```
 [1] 3 3 3 3 1 1 1 3 0 3 4 4 4 3 3 3 6 4 3 2
```

```
mean(yobs)
```

```
[1] 2.85
```

```
var(yobs)
```

```
[1] 1.818421
```

```
sd(yobs)
```

```
[1] 1.348488
```

```
median(yobs)
```

```
[1] 3
```

2.2 Definition of the likelihood function and an analytical solution

The likelihood function in this case is defined as

$$L(p) = P(Y_1 = y_1, \dots, Y_n = y_n) = \prod_{i=1}^{n} \binom{N}{y_i} p^{y_i} (1-p)^{N-y_i},$$

and therefore the log-likelihood can be expressed as follows:

$$l(p) = \log L(p) = \log \prod_{i=1}^{n} \binom{N}{y_i} + \left(\sum_{i=1}^{n} y_i \right) \log p + \left(\sum_{i=1}^{n} (N - y_i) \right) \log(1-p).$$

In this simple example, we can easily obtain an analytical solution.

$$\frac{\partial}{\partial p} l(p) = \left(\sum_{i=1}^{n} y_i \right) \frac{1}{p} - \left(\sum_{i=1}^{n} (N - y_i) \right) \frac{1}{1-p} = 0$$

$$\hat{p}(Y) = \frac{1}{Nn} \sum_{i=1}^{n} y_i$$

2.3 Quick overview of properties of ML estimator $\hat{p}(Y)$

You can see that the estimator $\hat{p}(Y)$ is unbiased. Also, the variance is disprportional to the sample size N and n.

$$\begin{aligned} E[\hat{p}(Y)] &= E\left[\frac{1}{Nn} \sum_{i=1}^{n} Y_i \right] = \frac{1}{Nn} E\left[\sum_{i=1}^{n} Y_i \right] = \frac{n}{Nn} E[Y_1] = \frac{1}{N} Np = p, \\ V[\hat{p}(Y)] &= V\left[\frac{1}{Nn} \sum_{i=1}^{n} Y_i \right] = \left(\frac{1}{Nn} \right)^2 V\left[\sum_{i=1}^{n} Y_i \right] \\ &= \frac{n}{N^2 n^2} n V[Y_1] = \frac{1}{nN^2} Np(1-p) = \frac{1}{Nn} p(1-p). \end{aligned}$$

```
Nit <- 10^4
Nvec <- c(10,100,1000)
pvec <- c(0.1, 0.3, 0.5)
pest <- array(NA,c(Nit,9))
col.list <- c("blue","orange","green")
```

```
par(mfrow=c(3,3), mar=c(3,3,4,2))
for(i in 1:3){
for(j in 1:3){
   tmp <- rbinom(Nit,Nvec[i],pvec[j])/Nvec[i]
   hist(tmp, xlab="Estimate", xlim=c(0,1), col=col.list[j],
        main=paste("N=",Nvec[i]," p=",pvec[j]))
}}
```

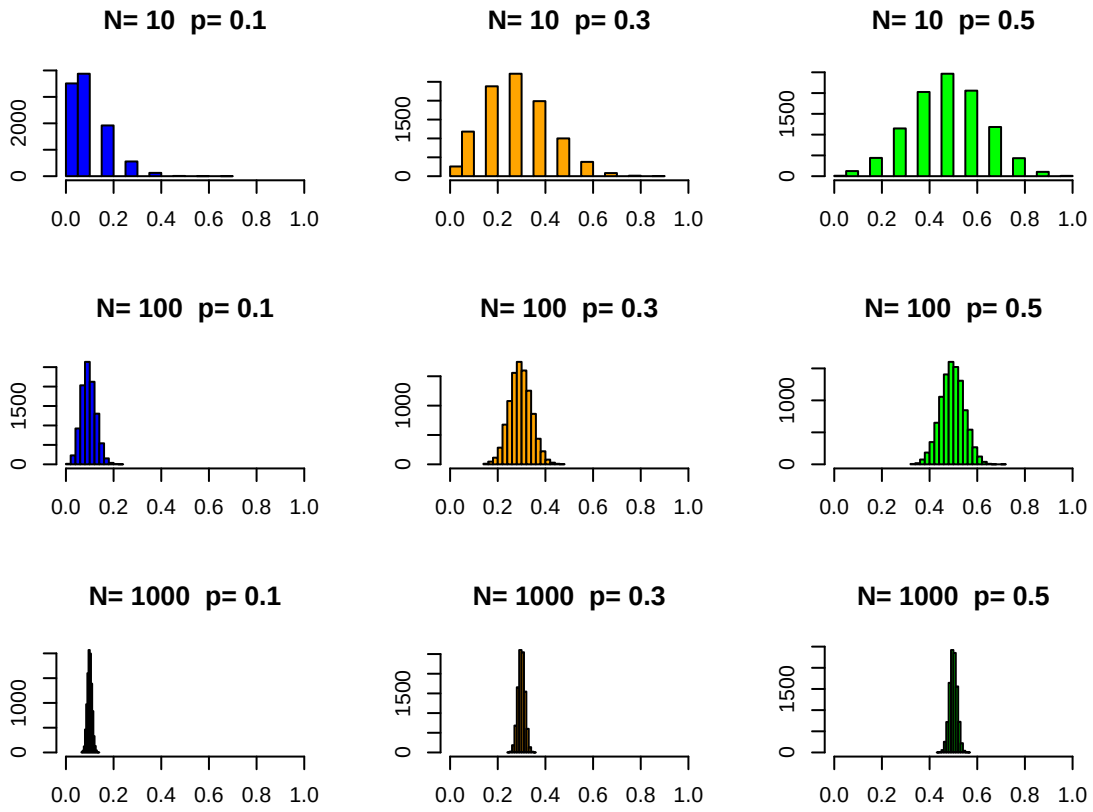
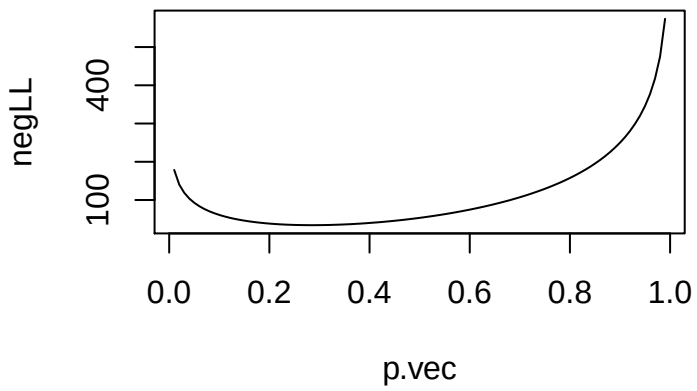


Figure 1: Simulated probability distributions for combinations of N and p for $n = 1$.

2.4 Numerical optimization in R

2.4.1 Definition of the negative log-likelihood function

```
negloglike.bin <- function(p){
 prob <- dbinom(yobs, N, p, log=T)
 loglike <- sum(prob)
 negloglike <- -loglike
 negloglike
}

p.vec <- seq(0.01, 0.99, 0.01)
negLL <- sapply(p.vec, negloglike.bin)
plot(p.vec, negLL, type="l")
```



2.4.2 Optimization (1) with bounds

```
res.bin <- optim(0.5, negloglike.bin, method="L-BFGS-B", lower=0.001, upper=0.999)
res.bin
```

```
$par
[1] 0.2850007

$value
[1] 34.1979

$counts
function gradient
       7        7

$convergence
[1] 0

$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

```
p.est <- res.bin$par
p.est
```

```
[1] 0.2850007
```

```
mean(yobs)/N
```

```
[1] 0.285
```

2.4.3 Optimization (2) with a parameter transformation

```
negloglike.bin <- function(par){
 p <- 1/(1+exp(-par))
 prob <- dbinom(yobs, N, p, log=T)
 loglike <- sum(prob)
```

```
 negloglike <- -loglike
 negloglike
}

res.bin <- optim(0, negloglike.bin, method="BFGS")
p.est <- 1/(1+exp(-res.bin$par))
p.est
```

```
[1] 0.2850001
```

---

3 ML estimation for occupancy model (1) -simple poisson model-

3.1 Description

- Case 1: Suppose that you are studying occupation of a species in a study area. At first, let us assume a quite ideal situation that you can observe counts of animals without missing.
- Notation: $Y_1, Y_2, \ldots, Y_k \sim (iid)Pois(\lambda)$
- Task: to estimate λ
- simulation data

```
Ns <- 20
lambda <- 3
yy <- rpois(Ns, lambda)
yy
```

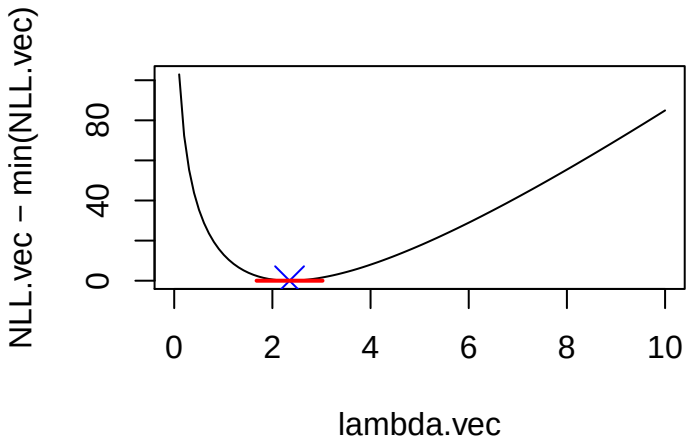
```
 [1] 1 1 4 2 3 1 3 1 9 1 1 0 2 2 4 2 2 3 3 2
```

3.2 Definition of likelihood function and implementation of optimization-1

```
#Def of negative loglikelihood
NLL.pois <- function(lambda){
 logPF <- dpois(yy, lambda, log=T)
 NLL <- (-1.0)*sum(logPF)
 return(NLL)
}

lambda.vec <- seq(0, 10, length.out=100)
NLL.vec <- sapply(lambda.vec, NLL.pois)
plot(lambda.vec, NLL.vec-min(NLL.vec), type="l")

Res.pois <- optim(5, NLL.pois, method="BFGS", hessian=T)
lambda.est <- Res.pois$par
lambda.se <- sqrt(1/Res.pois$hessian)
data.frame(lambda.est,lambda.se)
```

```
  lambda.est lambda.se
1   2.350024 0.3427861
```

```
points(Res.pois$par, 0, pch=4, col="blue", cex=2)
lines(lambda.est+c(-1.96,1.96)*lambda.se, c(0,0), col="red", lwd=2)
```

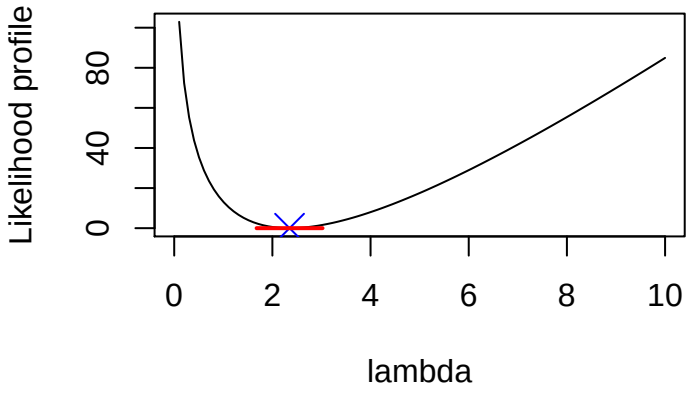


3.3 Definition of likelihood function and implementation of optimization-2

```
#Def of negative loglikelihood-2
NLL.pois <- function(loglambda){
 lambda <- exp(loglambda)
 logPF <- dpois(yy, lambda, log=T)
 NLL <- (-1.0)*sum(logPF)
 return(NLL)
}

lambda.vec <- seq(0, 10, length.out=100)
NLL.vec <- sapply(log(lambda.vec), NLL.pois)
plot(lambda.vec, NLL.vec-min(NLL.vec), type="l", xlab="lambda", ylab="Likelihood profile")

Res.pois <- optim(0, NLL.pois, method="BFGS", hessian=T)
lambda.est <- exp(Res.pois$par)
lambda.se <- lambda.est*sqrt(1/Res.pois$hessian)
data.frame(lambda.est,lambda.se)
```

```
  lambda.est lambda.se
1   2.350001 0.3427827
```

```
points(lambda.est, 0, pch=4, col="blue", cex=2)
lines(lambda.est+c(-1.96,1.96)*lambda.se, c(0,0), col="red", lwd=2)
```



---

4 ML estimation for occupancy model (2) -PA model-

4.1 Description

- Case 2: Like in Case 1, suppose that you are studying occupation of a species in a study area. Here, let us assume that you can observe only "presence/absence" without missing (0 means absence). (In this case, you can increase the number of sample locations (k))
- Notation: $Z_1, Z_2, \ldots, Z_k \sim (iid) Bin(1, p(\lambda))$, where $p(\lambda) = what?$.
- Task: to estimate λ
- simulation data

```
Ns <- 50
lambda <- 3
yy <- rpois(Ns, lambda)
zz <- as.numeric(yy>0)
zz
```

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1
[39] 1 1 1 1 1 1 1 1 1 1 0 1
```

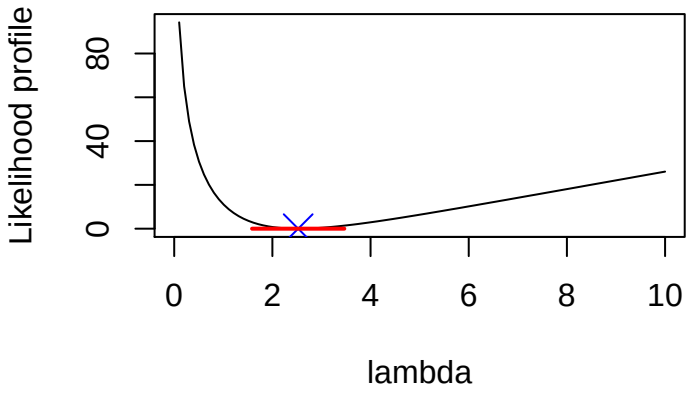
4.2 Definition of likelihood function and implementation of optimization

```
#Def of negative loglikelihood-2
NLL.PA <- function(loglambda){
 lambda <- exp(loglambda)
 pp <- 1-exp(-lambda)
 logPF <- dbinom(sum(zz), Ns, pp, log=T)
 NLL <- (-1.0)*sum(logPF)
 return(NLL)
}

lambda.vec <- seq(0, 10, length.out=100)
```

```
NLL.vec <- sapply(log(lambda.vec), NLL.PA)
plot(lambda.vec, NLL.vec-min(NLL.vec), type="l", xlab="lambda", ylab="Likelihood profile")

Res.PA <- optim(0, NLL.PA, method="BFGS", hessian=T)
lambda.est <- exp(Res.PA$par)
lambda.se <- lambda.est*sqrt(1/Res.PA$hessian)
data.frame(lambda.est,lambda.se)
```

```
  lambda.est lambda.se
1   2.525729 0.4795832
```

```
points(lambda.est, 0, pch=4, col="blue", cex=2)
lines(lambda.est+c(-1.96,1.96)*lambda.se, c(0,0), col="red", lwd=2)
```



5 ML estimation for occupancy model (3) -realistic case-

5.1 Description

- Case 3: Like in the previous cases, suppose that you are studying occupation of a species in a study area. Again, suppose that you visit each sampling location several day and observe if at least one individual exist of not ("presence/absence) every visit", but your detection process is subject to missing. The probability of detecting presence depends on the number of individuals in a sampling unit and detection probability per individual. Note that the number of individuals in each location does not change during the study period.
- Notation:

$$N_1, N_2, \ldots, N_k \sim (iid) Pois(\lambda)$$
$$Z_{it}|N_i \sim (iid) Bin(1, p(N_i)), \quad \text{where } p(N_i) = 1-(1-p)^{N_i} \quad (t = 1, 2, ..., T_i; i = 1, 2, \ldots, k)$$
$$Y_i|N_i \sim Bin(T_i, p(N_i)) \quad (i = 1, 2, \ldots, k)$$

- Task: to estimate λ and p (λ is of your interest)
- simulation data

```
lambda <- 3
pp <- 0.2
Ns <- 50
```

```
Tmax <- 10
Tvec <- sample(1:Tmax, Ns, replace=T)
zz <- array(NA, c(Ns, Tmax))
yy <- numeric(Ns)

for(i in 1:Ns){
NN <- rpois(Ns, lambda)
zz[i,1:Tvec[i]] <- rbinom(Tvec[i], 1, 1-(1-pp)^NN[i])
yy[i] <- sum(zz[i,], na.rm=T)
}

head(zz, 10)
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    1    1    1    1   NA   NA   NA   NA    NA
 [2,]    1   NA   NA   NA   NA   NA   NA   NA   NA    NA
 [3,]    1    1    0    1    0    0    0    1   NA    NA
 [4,]    1    1    0    0    1    1    0    1   NA    NA
 [5,]    1    1    0    1    1    0   NA   NA   NA    NA
 [6,]    0    1    0    1    1    1    0   NA   NA    NA
 [7,]    1    1    1    0    0    0   NA   NA   NA    NA
 [8,]    0    0   NA   NA   NA   NA   NA   NA   NA    NA
 [9,]    0    0    0    0    0    1    0    0    1     0
 [ reached getOption("max.print") -- omitted 1 row ]
```

```
yy
```

```
 [1] 5 1 4 5 4 4 3 0 2 3 3 1 0 1 1 1 7 0 2 2 2 5 2 5 1 0 3 5 3 4 5 0 4 2 1 3 1 2
[39] 1 7 3 2 3 6 4 1 4 3 5 0
```

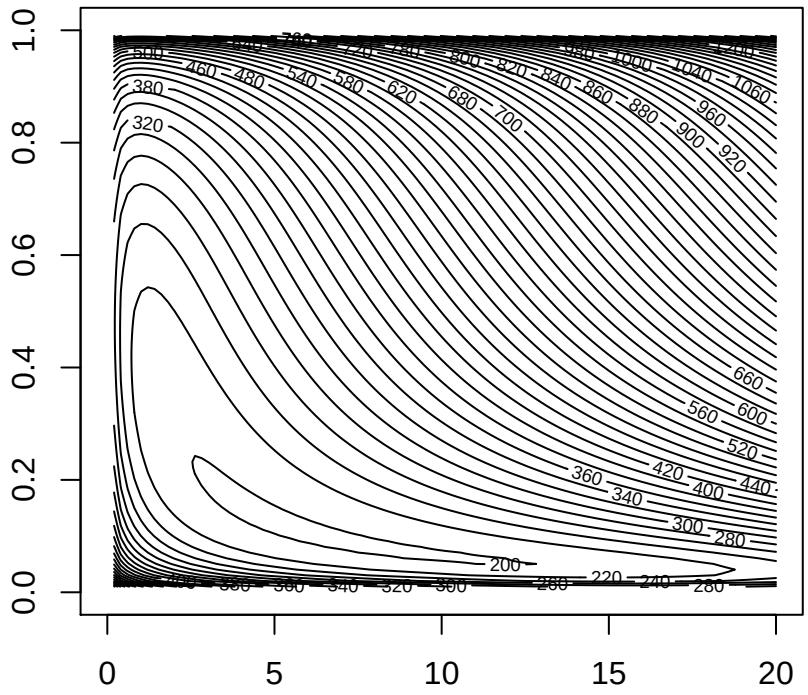
5.2 Defining likelihood with respect to $\log\lambda$ and logit$p$

```
NLL <- function(par, Kmax=20){
 lambda <- exp(par[1])
 pp <- 1/(1+exp(-par[2]))
 Detectvec <<- 1-(1-pp)^(0:Kmax)

 Tmp1 <- dpois(0:Kmax, lambda)
 Likevec <- numeric(Ns)

 for(i in 1:Ns){
  Tmp2 <- Detectvec^yy[i]*(1-Detectvec)^(Tvec[i]-yy[i])
  Likevec[i] <- sum(Tmp1*Tmp2)
 }
 obj <- (-1.0)*sum(log(Likevec))
}
```

5.3 Contour of likelihood with respect to λ and p

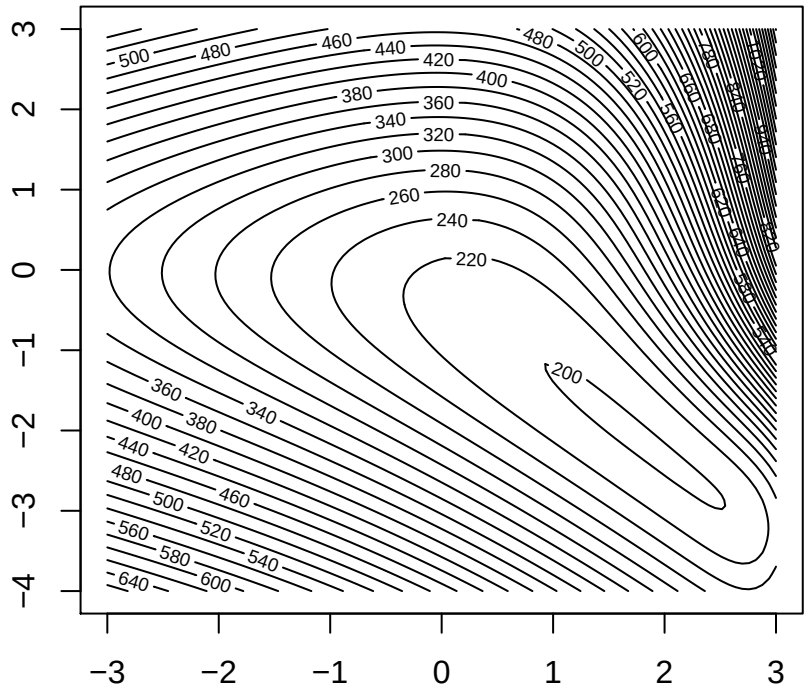
```
Len <- 100
LL <- array(0, c(Len,Len))
```

```
lvec <- seq(0,20,length.out=Len)
pvec <- seq(0,1,length.out=Len)

for(i in 1:Len){
  for(j in 1:Len){
    LL[i,j] <- NLL(c(log(lvec[i]), logit(pvec[j])), Kmax=20)
  }
}
contour(lvec,pvec,LL,nlevels=50)
```



5.4 Contour of likelihood with respect to $log\lambda$ and logit$p$

```
Len <- 100
LL <- array(0, c(Len,Len))
loglvec <- seq(-3,3,length.out=Len)
logitpvec <- seq(-4,3,length.out=Len)
for(i in 1:Len){
  for(j in 1:Len){
    LL[i,j] <- NLL(c(loglvec[i], logitpvec[j]), Kmax=20)
  }
}
contour(loglvec,logitpvec,LL,nlevels=50)
```



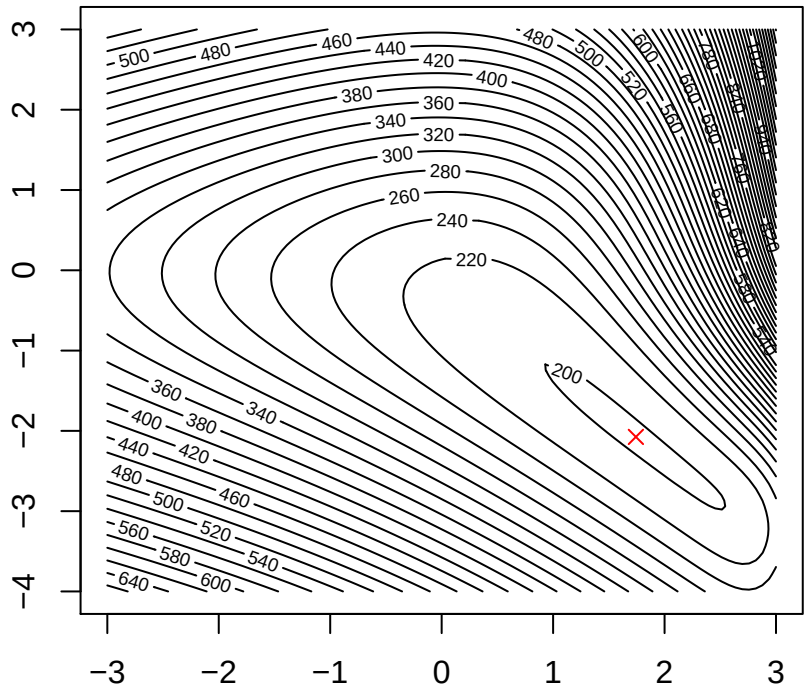
5.5 Optimization with respect to $\log \lambda$ and logit$p$

```
init <- c(0,0)
Res <- optim(init, NLL, method="BFGS", hessian=T, Kmax=20)
Res
```

```
$par
[1]  1.742692 -2.074976

$value
[1] 198.2165

$counts
function gradient
      22       11

$convergence
[1] 0

$message
NULL

$hessian
```

```
         [,1]     [,2]
[1,] 89.41554 78.85059
[2,] 78.85059 72.00360
```

```
contour(loglvec,logitpvec,LL,nlevels=50)
points(Res$par[1],Res$par[2], col="red", pch=4)
```



5.6 Estimates and their standard errors

```
lambda.est <- exp(Res$par[1])
p.est <- 1/(1+exp(-Res$par[2]))
Avar <- solve(Res$hessian)
lambda.se <- lambda.est*sqrt(Avar[1,1])
p.se <- p.est*(1-p.est)*sqrt(Avar[2,2])
data.frame(lambda.est, lambda.se, p.est, p.se)
```

```
  lambda.est lambda.se    p.est       p.se
1   5.712699  3.262073 0.111553 0.06306588
```

More accurate CIs can be given by profile likelihood methods, bootstrapping and Bayesian methods.