Lecture 3: MLE (logistic regression)

MLE for Bernoulli variables

Let's look at the example from class where we observe 3 diseased individuals out of a sample of 10.

```
pvec <- c(0,.1,.2,.29,.3,.33,.4,1)
# grouped
llvec1 <- dbinom(3,10,pvec,log=TRUE)
# ungrouped
llvec2 <- sapply(pvec,function(p)
sum(dbinom(c(rep(1,3),rep(0,7)),1,p,log=TRUE)))

data.frame(p=pvec,grouped.loglik=llvec1,ungrouped.loglik=llvec2)
```

```
##      p grouped.loglik ungrouped.loglik
## 1 0.00           -Inf             -Inf
## 2 0.10      -2.857787        -7.645279
## 3 0.20      -1.602827        -6.390319
## 4 0.29      -1.323563        -6.111055
## 5 0.30      -1.321151        -6.108643
## 6 0.33      -1.341839        -6.129331
## 7 0.40      -1.537160        -6.324652
## 8 1.00           -Inf             -Inf
```

The grouped and ungrouped log-likelihoods only differ from each other by a constant dependent on the data.

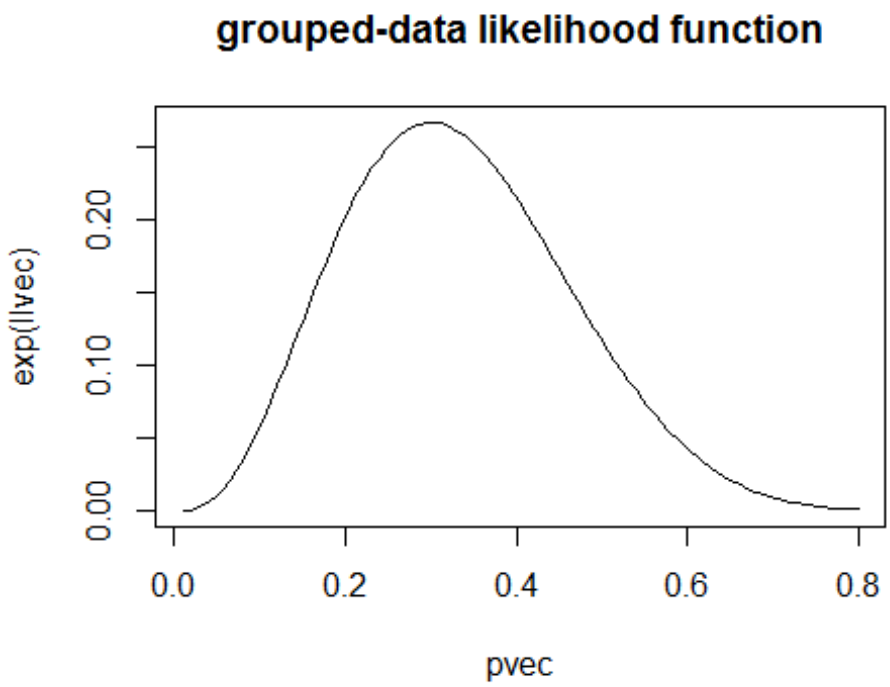
```
llvec1-llvec2
```

```
## [1]      NaN 4.787492 4.787492 4.787492 4.787492 4.787492 4.787492
NaN
```

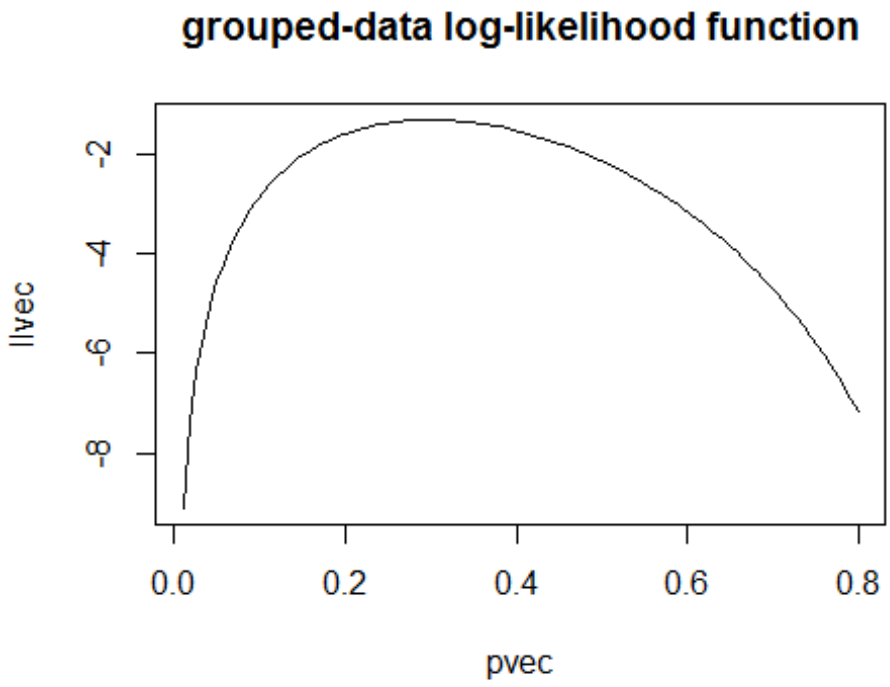
```
lchoose(10,3)
```

```
## [1] 4.787492
```

```
pvec <- seq(.01,.8,length.out=101)

llvec <- dbinom(3,10,pvec,log=TRUE)

plot(pvec,exp(llvec),type='l', main='grouped-data likelihood function')
```

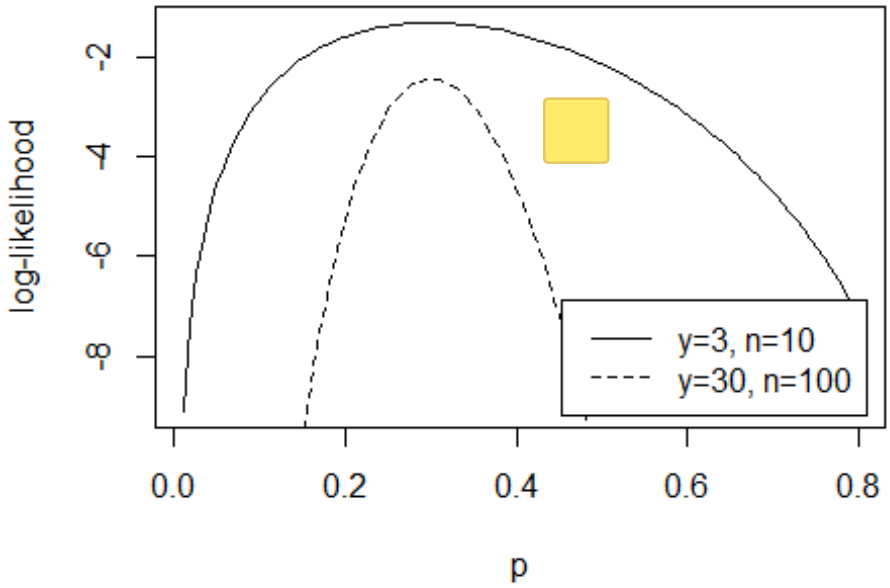



```
plot(pvec,llvec,type='l', main='grouped-data log-likelihood function')
```



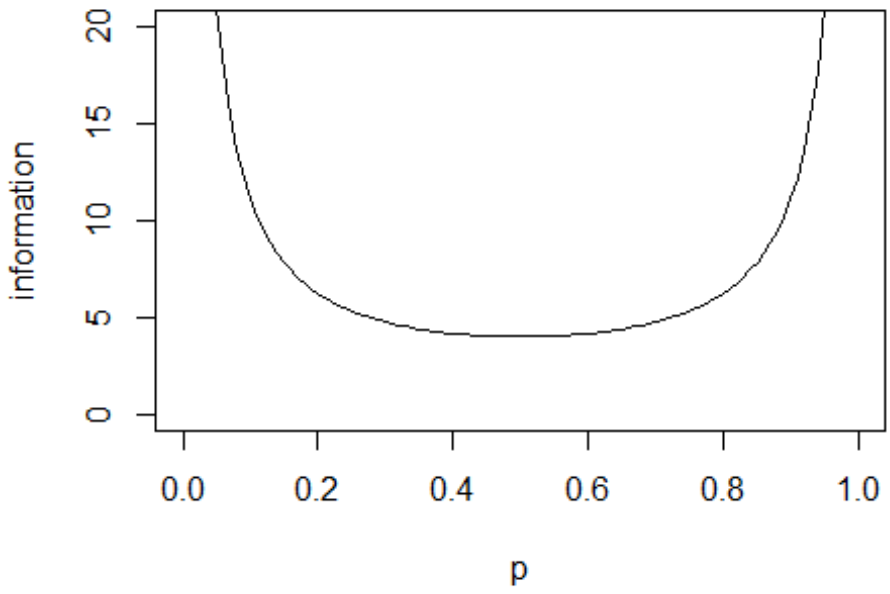
What if we had observed the same proportion in a larger sample? Say 30 diseased individuals out of 100.

```
llvec.smallsamp <- dbinom(3,10,pvec,log=TRUE)
llvec.largesamp <- dbinom(30,100,pvec,log=TRUE)

plot(pvec,llvec.smallsamp,type='l',xlab='p',ylab='log-likelihood')
# plot(pvec,llvec.largesamp,type='l')
lines(pvec,llvec.largesamp,lty=2)

legend(x='bottomright',inset=.025,lty=1:2,legend=c('y=3, n=10','y=30,
n=100'))
```



The log-likelihood function is becoming more peaked as sample size increases. This means that the variance of the MLE is decreasing for larger samples. The variance is the inverse of the information, a measure of the curvature of the log-likelihood function.

```
# the information in a single binary observation with probability p is
1/(p*(1-p))
curve(1/(x*(1-x)),ylim=c(0,20),xlab='p',ylab='information')
```



The information contained a single random variable is much higher when its mean is close to 0 or 1 than when it is close to 1/2.

Smoking and lung cancer

This example will again look at the smoking data set from the previous lecture.

```
##     y   n passive
## 1 281 491       1
## 2 228 507       0
```

R uses Fisher scoring to fit the logistic regression model, which is fairly specific to the generalized linear model framework. Now let's look at another way to fit this model that can be applied much more generally. It is based on directly coding the log-likelihood function for this model.

```
logit.llikfun <- function(theta,y,n,xmat) { # first argument has to be
parameter vector
  eta <- cbind(1,xmat) %*% theta # linear predictor
  prob <- plogis(eta) # here is where the logistic function appears
  sum(dbinom(x=y,size=n,prob=prob,log=TRUE)) # sum up the individual log
probability mass values
}

# check the value at the solution found by glm()
logLik(smoke.logitmod)
```

```
## 'log Lik.' -6.649433 (df=2)

# using our new function
logit.llikfun(coef(smoke.logitmod),
              y=smoke$y,
              n=smoke$n,
              xmat=as.matrix(smoke$passive,ncol=1))

## [1] -6.649433

# if we want to use this function to fit a model, we need to specify starting
values
start.beta <- c(0,0)

mod1 <- optim(start.beta, # starting values
              logit.llikfun, # function to optimize
              hessian=TRUE, # so we can get standard errors
              # arguments to the log-likelihood function
              y=smoke$y,
              n=smoke$n,
              xmat=as.matrix(smoke$passive,ncol=1),
              method='BFGS', # optimization method (quasi-Newton)
              control=list(fnscale=-1) # tells optim() to maximize instead of
minimize
              )
```

We can compare the estimates from this model with those from `glm()`.

```
# compare with glm()
# estimates
coef(smoke.logitmod)

## (Intercept)     passive
##  -0.2018662   0.4931133

mod1$par

## [1] -0.2018661  0.4931132

# covariance matrix
V.smoke

##              (Intercept)      passive
## (Intercept)  0.007970194 -0.007970194
## passive     -0.007970194  0.016290818

solve(-mod1$hessian)

##              [,1]         [,2]
## [1,]  0.007970196 -0.007970196
## [2,] -0.007970196  0.016290820
```

Being able to calculate the log-likelihood directly can be helpful. For example, we can visualize the likelihood surface this way.

```
beta0vec <- seq(-.5,.5,length.out=51)
beta1vec <- seq(0,1,length.out=51)
llsurf <- array(dim=c(length(beta0vec),length(beta1vec)))

for(i0 in seq_along(beta0vec)) {
  for(i1 in seq_along(beta1vec)) {
    llsurf[i0,i1] <- logit.llikfun(c(beta0vec[i0],beta1vec[i1]),
              y=smoke$y,
              n=smoke$n,
              xmat=as.matrix(smoke$passive,ncol=1))
  }
}

contour(beta0vec,beta1vec,llsurf)
```