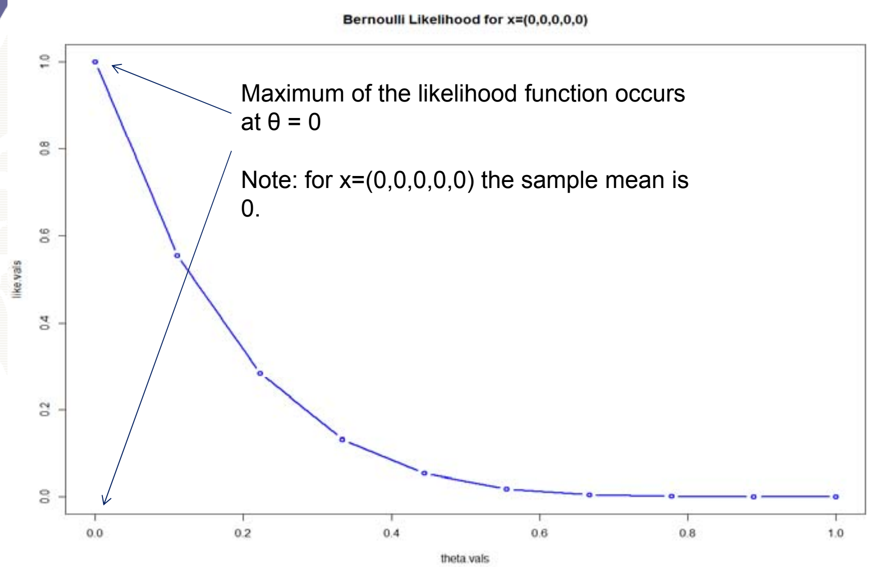
Maximum Likelihood Estimation

Econ 424/Amath 540
Eric Zivot
Summer 2012
Updated: July 26, 2012

Bernoulli Likelihood

```
likelihood.Bernoulli = function(theta, x) {
  # theta   success probability parameter
  # x       vector of data
  n = length(x)
  ans = theta^sum(x) * (1-theta)^(n-sum(x))
  return(ans)
}

# plot Bernoulli likelihood
> x = rep(0,5)
> theta.vals = seq(0,1, length.out=10)
> like.vals = likelihood.Bernoulli(theta.vals, x)
> plot(theta.vals, like.vals, type="b", col="blue", lwd=2,
+      main="Bernoulli Likelihood for x=(0,0,0,0,0)")
```

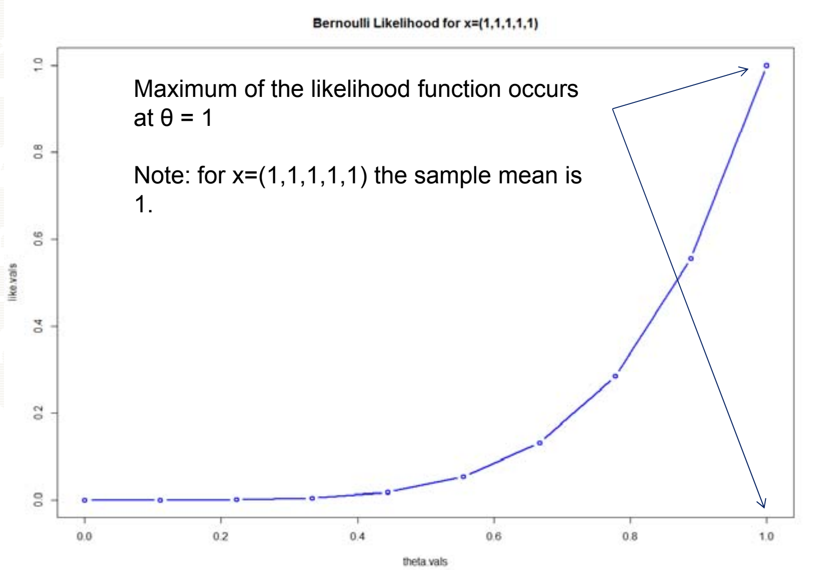
Bernoulli Likelihood



Maximum of the likelihood function occurs at θ = 0

Note: for x=(0,0,0,0,0) the sample mean is 0.

Bernoulli Likelihood



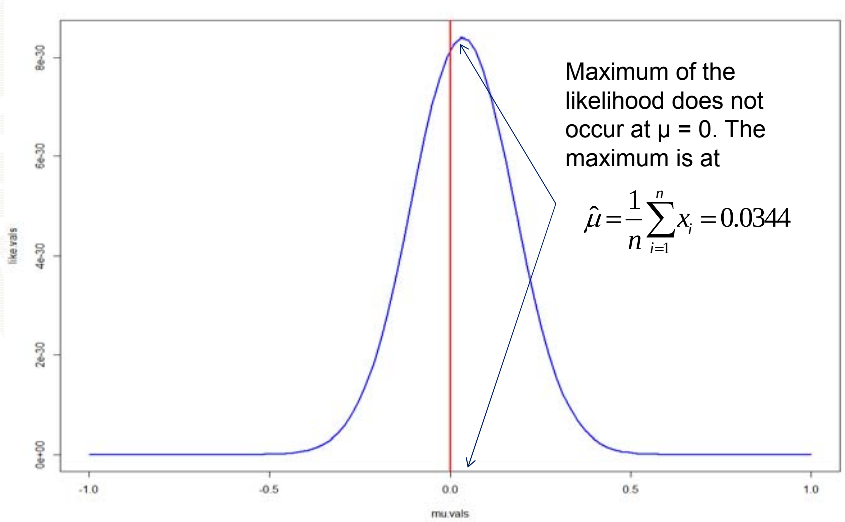
Maximum of the likelihood function occurs at θ = 1

Note: for x=(1,1,1,1,1) the sample mean is 1.

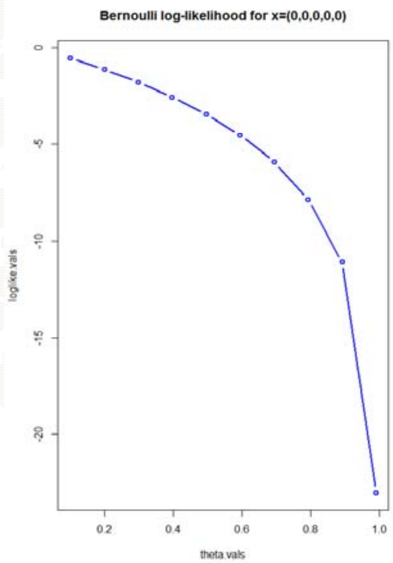
Normal Likelihood

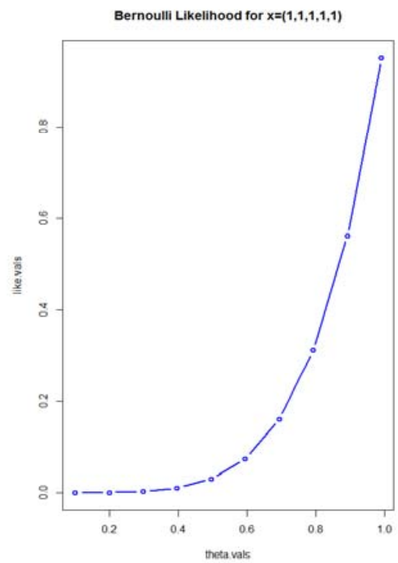
```
likelihood.normal.mu = function(mu, sig2=1, x) {
  # mu       mean of normal distribution for given sig2
  # x        vector of data
  n = length(x)
  a1 = (2*pi*sig2)^-(n/2)
  a2 = -1/(2*sig2)
  y = (x-mu)^2
  ans = a1*exp(a2*sum(y))
  return(ans)
}

# generate N(0,1) data
> n = 50
> x = rnorm(n, mean=0, sd=1)

# compute normal likelihood as function of mu
> mu.vals = seq(-1,1, length.out=100)
> like.vals = rep(0,length(mu.vals))
> for (i in 1:length(like.vals)) {
+   like.vals[i] = likelihood.normal.mu(mu.vals[i], sig2=1, x=x)
+ }
```


Normal Likelihood for μ

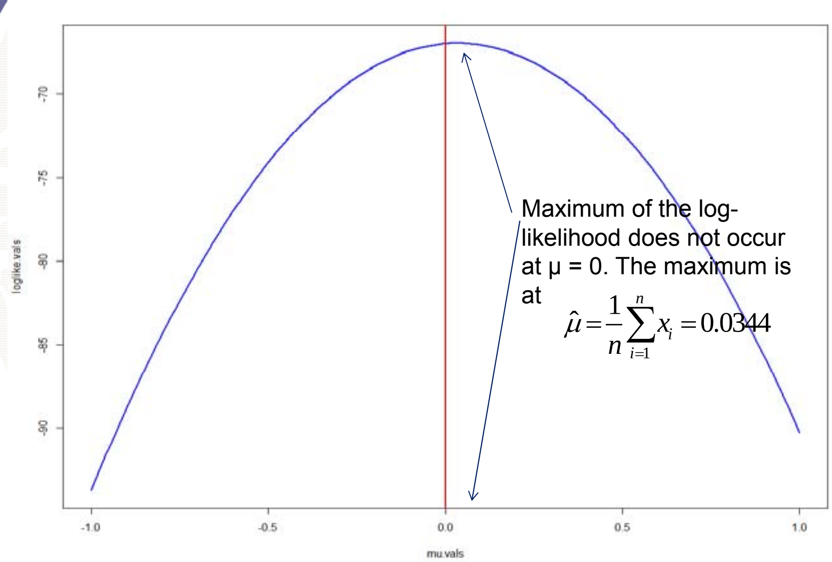


Bernoulli Log-Likelihood





Normal Log-Likelihood

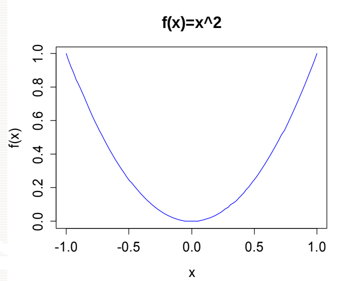


Maximum of the log-likelihood does not occur at μ = 0. The maximum is at

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i = 0.0344$$

R `optimize()` function

```
# use optimize() to maximize or minimize function of one variable

test.fun = function(x) {
  return(x^2)
 }

> ans = optimize(test.fun, lower=-1, upper=1, maximum=FALSE)
> class(ans)
[1] "list"
> names(ans)
[1] "minimum"   "objective"

> ans
$minimum
[1] -2.776e-17

$objective
[1] 7.704e-34
```



R `optim()` function

```
# use optim() to minimize functions of multiple variables
test.fun = function(theta) {
  ans = theta[1]^2 + theta[2]^2
  return(ans)
}

# set starting values for optimizer
> theta.start = c(1,1)
# optimize function
> ans = optim(par=theta.start, fn=test.fun,
+              method="BFGS")
> class(ans)
[1] "list"

> names(ans)
[1] "par"         "value"         "counts"        "convergence"
[5] "message"
```

R optim() function

```
> ans
$par
[1] -4.264e-16 -4.264e-16

$value
[1] 9.088e-30

$counts
function gradient
       8        3

$convergence
[1] 0

$message
NULL
```

Solution to minimization problem

Value of function at solution

Compute MLEs using optim()

```
# maximize normal log-likelihood using optim
# by minimizing -1*log-likelihood
loglike.normal = function(theta, x) {
  # theta   parameters c(mu,sig2)
  # x       vector of data
  mu = theta[1]
  sig2 = theta[2]
  n = length(x)
  a1 = -(n/2)*log(2*pi)-(n/2)*log(sig2)
  a2 = -1/(2*sig2)
  y = (x-mu)^2
  ans = a1+a2*sum(y)
  # return -1 * loglike
  return(-ans)
}
```

Compute MLEs using **optim()**

```
# generate N(0,1) data
> n = 50
> set.seed(123)
> x = rnorm(n, mean=0, sd=1)

# set starting values for optimizer
> theta.start = c(0,1)
> ans = optim(par=theta.start, fn=loglike.normal, x=x,
+             method="BFGS")
Warning messages:
1: In log(sig2) : NaNs produced
2: In log(sig2) : NaNs produced

> ans$par
[1] 0.03442 0.84011

# check result agains analytic formulas
> mean(x)
[1] 0.0344
> var(x)*(n-1)/n
[1] 0.8401
```


Computing MLEs using **optim()** with Estimated Standard Errors

```
> ans = optim(par=theta.start, fn=loglike.normal, x=x,
+             method="BFGS", hessian=TRUE)

> names(ans)
[1] "par"         "value"       "counts"
"convergence"
[5] "message"     "hessian"

> ans$hessian
           [,1]       [,2]
[1,] 59.5160231 -0.0009716
[2,] -0.0009716 35.4202372

> se.mle = sqrt(diag(solve(ans$hessian)))
> se.mle
[1] 0.1296 0.1680
```

Computing MLEs using **maxLik()** function

```
> library(maxLik)
# here function to compute log-likelihood returns
# log-likelihood values and not -1*log-likelihood
# values
loglike.normal = function(theta, x) {
  # theta   parameters c(mu,sig2)
  # x       vector of data
  mu = theta[1]
  sig2 = theta[2]
  n = length(x)
  a1 = -(n/2)*log(2*pi)-(n/2)*log(sig2)
  a2 = -1/(2*sig2)
  y = (x-mu)^2
  ans = a1+a2*sum(y)
  return(ans)
}
```


Computing MLEs using **maxLik()** function

```
> theta.start = c(0,1)
> names(theta.start) = c("mu","sig2")
> theta.mle = maxLik(loglike.normal, start=theta.start, x=x)
> class(theta.mle)
[1] "maxLik" "maxim"  "list"

> names(theta.mle)
 [1] "maximum"    "estimate"   "gradient"   "hessian"     "code"
 [6] "message"    "last.step"  "fixed"      "iterations" "type"

> theta.mle
Maximum Likelihood estimation
Newton-Raphson maximisation, 5 iterations
Return code 1: gradient close to zero
Log-Likelihood: -66.59 (2 free parameter(s))
Estimate(s): 0.0344 0.8401
```

Computing MLE using `maxLik()` function

```
> summary(theta.mle)
--------------------------------------------
Maximum Likelihood estimation
Newton-Raphson maximisation, 5 iterations
Return code 1: gradient close to zero
Log-Likelihood: -66.59
2  free parameters
Estimates:
     Estimate Std. error t value Pr(> t)
mu     0.0344     0.1296    0.27    0.79
sig2   0.8401     0.1680    5.00 5.8e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
--------------------------------------------
```