# Maximum Likelihood Analysis in R

Chad E. Brassil – June 29, 2007

## *Load special libraries*

For the maximum likelihood analysis you will need download and load a library. I recommend bbmle which was developed be Ben Bolker.

```
library(bbmle)
```

Alternatively, you can use the older stats4 library. Note stats4 uses the function mle() whereas bbmle uses mle2(), and stats4 does not have some of the supporting functions.

```
library(stats4)
```

A few special functions are included in the file mleFunctions.R. It can be loaded using the following code if a copy of the file is placed in your working directory.

```
source("mleFunctions.R")
```

## *Input data*

First, use "File/Change Dir…" form the menu to set the correct working directory. Then you can load the data from a csv file. The top row should contain the names of each column. In this example the file is called comboR.csv and I store the data set into the variable bristol.

```
bristol <- read.csv("comboR.csv")
```

---

*Optional*

---

Look at the names of the columns

```
names(bristol)
```

Edit individual values

```
edit(bristol)
```

Attach the dataset so that you can refer to column name directly

```
attach(bristol)
```

Examine plots of columns

```
plot(bristol$clone,bristol$infection)
```

or, if dataset has been attached, then you can just use

```
plot(clone,infection)
```

Look at subset of data, in this case when species equals a or b

```
subdata <- subset(data,species == c("a","b") )
plot(subdata$clone, subdata$infection)
```

For an interactive 3-D examination of data try the feature-reach scatter3D(), which requires the Rcmdr library

```
scatter3d(data$mInfl,data$visits,data$height)
```

---

## *Create support function*

To create the support function use the density form of a probability distribution (see the reference list of "Basic Distributions" ). Setting log = TRUE will give in the log version and –sum() will take the negative sum across the data set. The resulting function should be the negative log likelihood.

```
ll <- function (m1,s)
-sum(dnorm(bristol$infection,m1,s,log = TRUE))
```

---

*Optional*

---

The function with() can be used to temporarily attach a data set

```
ll <- function (m1,s) with(bristol,
-sum(dnorm(infection,m1,s,log = TRUE)))
```

Categorical independent variables, i.e. factors

```
ll <- function (m.a,m.b,s) with(bristol,{
    m = c(m.a,m.b);
    -sum(dnorm(infection,mean=m[species], sd=s,
     log = TRUE))
    })
```

However, be sure the vector, in this case m, is in the same order as

```
levels(bristol$species)
```

Interactions are done the same way, use ":" operator for interaction

```
with(bristol,levels(species:sex))
ll <- function (a.f,a.r,b.f,b.r,s) with(data,{
    m = c(a.f,a.r,b.f,b.r);
    -sum(dnorm(consumed,mean=m[sex:status],sd=s,
     log = TRUE))
    })
```

---

## *Pick starting guess and find MLE*

The Nelder-Mead method, or simplex method, is a generally robust method for finding the minimum of the support function.

```
guess <- list(m1 = 0.3, s = 0.1)
fit <- mle2(ll, start=guess, method="Nelder-Mead")
```

---

*Optional*

---

Parameter constraints can be given when using a the L-BFGS-B minimization method

```
fit <- mle2(ll, start=guess, method="L-BFGS-B", lower
= c(-1000,0), upper = c(1000,1000))
```

General summary of results

```
summary(fit)
```

Find confidence intervals for a model

```
confint(fit)
```

Or for a single parameter

```
confint(fit,"m1")
```

Examine likelihood profile plots. The profiles in display the transformation $\left|\sqrt{-\mathbf{L}}\right|$ on the vertical axis. The second dashed, pink line from the top is the 95% confidence interval. The 0.99999 in conf forces it to plot a wider range of the profile; 0.95 plots the 95% confidence interval. Use par to show multiple plots at once.

```
par(mfrow=c(length(guess),1))
plot(profile(fit),conf=c(0.95,0.99999),absVal=TRUE)
par(mfrow=c(1,1))
```

To profile a single parameter

```
profile(fit,"m1")
```

To display the untransformed negative log likelihood, use the following function which is included in mleFunctions.R

```
prof <- profile(fit.saturating)
profpic(prof,"h",best=fit.saturating)
```

Visually examine model fit against data to ensure what you're doing makes sense

```
plot(mInfl,visits, xlab = "flowers", ylab = "visits")
```

Use curve() and coef() to show a line of your data onto of your data points

```
curve(coef(fit)["m"]*x+coef(fit)["b"], add=TRUE)
```

Use dev.copy2eps to save a copy of a figure

```
dev.copy2eps(file="fig.eps",width=8,height=6)
```

Randomly pick starting values and repeat optimization multiple times using this custom function called mle.random(). Output is the best fit. The support function is *ll* and the *guess* is the same as in mle(). Starting conditions are randomly selected *n* times from within *lower.guess* and *upper.guess*. Note, the code for mle.random() can be found in the file mleFunctions.R.

```
mle.random (ll, start, method, lower.start,
upper.start, n = 100 )
```

Example of use

```
ll <- function (m1,s)
-sum(dnorm(bristol$infection,m1,s,log = TRUE))
guess <- list(m1 = 0.3, s = 0.1)
fit <- mle2(ll, start=guess, method="Nelder-Mead")
summary(fit)
fit <- mle.random(ll,start=guess,lower.start=
     c(-3,0),upper.start=c(3,10),n=100)
summary(fit)
```

---

## *Compare models*

Extract AIC value for an individual model

```
AIC(fit)
```

To compare across a number of models, store each model in a separate fit value with an informative name. For example

```
AICtab(fit.constant,fit.linear,fit.saturating,
     weights=TRUE,delta=TRUE,sort=TRUE)
```

To calculate AICc, use nobs to specify the number of observations. In the code below I extract that from data.

```
AICctab(fit.constant,fit.linear,fit.saturating,
     nobs=dim(data)[1],weights=TRUE,delta=TRUE,sort=T)
```

The negative log-likelihood values for each function can be displayed using this code.

```
-logLik(fit.linear)
```

For nested models, you can use the Likelihood Ratio test. The user must ensure that models compared in this way are nested.

```
anova(fit.constant,fit.linear)
```