Likelihood and Model Evaluation

Zack Treisman

Spring 2022

WESTERN COLORADO UNIVERSITY
MATHEMATICS & COMPUTER SCIENCE DEPARTMENT

Philosophy

The time has come for us to discuss in some detail how our models are being constructed and evaluated.

- The key idea is **likelihood**, the probability that a model would produce the data we observe.

When do we say *likelihood* instead of *probability*?

- The probability $P(\text{data}|\text{model})$ considers the model as fixed (it's the laws of nature or something), and the data as variable.
- The likelihood $\mathcal{L}(\text{model}|\text{data})$ considers the data to be fixed (they are what has been observed) and the model as variable.

This kind of model-space/ data-space parameter duality is all over the place in math, so it must be kind of important. But this is not a class for that sort of *abstract nonsense*[1].

[1] A technical term. No seriously.

Maximum Likelihood Estimation

To find a parametric model to fit data we need to choose values for the parameters according to some criteria.

- Minimizing mean squared error is ideal if the true error distribution is normal with constant variance.
    - Non-normal errors are modeled with a GLM. (e.g. Poisson, negative binomial or logistic regression)
    - Heterogenous variance can be accounted for by a mixed/ multilevel model. (Coming soon)
- Finding parameters by maximizing the likelihood agrees with least squares when errors are normal and homoscedastic gives good results when they are not.

Computing likelihood

An experiment described in Bolker (2008) considers tadpoles in an aquarium undergoing predation.

- Four tanks, each with 10 tadpoles to start. The experiment ends with 7, 5, 9 and 9 tadpoles in the tanks.

What is the likelihood of the data if the survival probability is 70%?

- `dbinom` computes binomial probabilities (likelihoods).
- The tanks are independent, so the individual likelihoods can be multiplied to compute the total likelihood.

```
x <- data.frame(survived=c(7,5,9,9), density=c(10,10,10,10))
(individual_likelihood <- dbinom(x$survived, x$density, p=0.7))
```

```
## [1] 0.2668279 0.1029193 0.1210608 0.1210608
```

```
(likelihood <- prod(individual_likelihood))
```

```
## [1] 0.0004024719
```

Likelihoods tend to be quite small.

Optimizing

R has taken Calc I and can find local maxima or minima using the `optim` function.

- What value of the parameter `p` gives the maximum likelihood of the observed data?

```
likhd_fn <- function(p){prod(dbinom(x$survived, x$density, p=p))}
opt0 <- optim(fn = likhd_fn, par = list(p=0.7),
              control=list(fnscale=-1), #maximize instead of minimize
              method="BFGS") #better for one variable
opt0$par
```

```
##         p
## 0.7499974
```

This is an example of process, not something we don't already know:

```
sum(x$survived)/sum(x$density)
```

```
## [1] 0.75
```

Negative log-likelihoods

For computational reasons, it is easier to deal with the logs of likelihood values.

- Products of more than a few numbers between 0 and 1 get really small. Like really really small.
- Logs turn products into sums.

For emotional reasons, it's easier to minimize than to maximize.

- Because gravity?

Replace `prod` with `-sum` and add in `log=TRUE`. Same result.

```r
nll_fn <- function(p){-sum(dbinom(x$survived, x$density, p=p, log = TRUE))}
opt1 <- optim(fn = nll_fn, par = list(p=0.7),
              method="BFGS") #better for one variable
opt1$par
```

```
##         p
## 0.7500001
```

The mle2 command

The bbmle package provides mle2 for maximum likelihood estimates.

```
#library(bbmle)
mle2(nll_fn, start = list(p=0.7), data=x)
```

```
##
## Call:
## mle2(minuslogl = nll_fn, start = list(p = 0.7), data = x)
##
## Coefficients:
##         p
## 0.7500001
##
## Log-likelihood: -7.57
```

MLE agrees with ordinary least squares

```r
x<-rnorm(50,5,1)
y<-rnorm(50, 3+2*x,1.5)
lm(y~x)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##       3.767        1.865
```

```r
nll_fn <- function(a,b,s) -sum(dnorm(y, a+b*x, s, log = TRUE))
mle2(nll_fn, start = list(a=2,b=3,s=1))
```

```
##
## Call:
## mle2(minuslogl = nll_fn, start = list(a = 2, b = 3, s = 1))
##
## Coefficients:
##        a        b        s
## 3.767375 1.864594 1.336506
##
## Log-likelihood: -85.45
```

MLE is used to fit a GLM

```r
n <- 50; x <- runif(n, min=0, max=5); y <- rpois(n, exp(0.5+0.3*x))
glm1 <- glm(y~x, family = poisson)
coef(glm1)
```

```
## (Intercept)           x
##   0.4098762   0.3149077
```

- In `mle2` both the negative log-likelihood function and the parameters can be expressed as formulas.

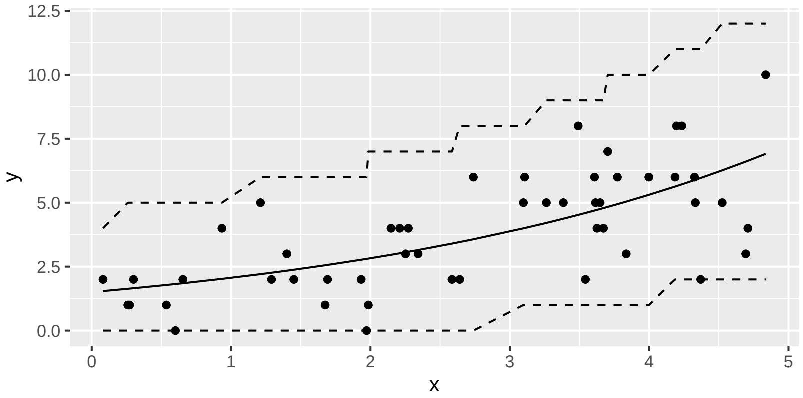
```r
mle2(y~dpois(exp(loglambda)),        ## use log link/exp inverse-link
     data=data.frame(y,x),           ## need to specify as data frame
     parameters=list(loglambda~x),   ## linear model for loglambda
     start=list(loglambda=0))        ## start values for *intercept*
```

```
##
## Call:
## mle2(minuslogl = y ~ dpois(exp(loglambda)), start = list(loglambda = 0),
##     data = data.frame(y, x), parameters = list(loglambda ~ x))
##
## Coefficients:
## loglambda.(Intercept)           loglambda.x
##             0.4098919             0.3149032
##
## Log-likelihood: -95.04
```

A plot of glm1

It is always helpful to plot your data and models.

```r
ggplot(data.frame(x=x, y=y), aes(x,y))+
  geom_point()+
  geom_line(aes(y=exp(predict(glm1))))+
  geom_line(aes(y=qpois(0.025,lambda = exp(predict(glm1)))), linetype=2)+
  geom_line(aes(y=qpois(0.975,lambda = exp(predict(glm1)))), linetype=2)
```



Analysis of a GLM

```r
summary(glm1)
```

```
##
## Call:
## glm(formula = y ~ x, family = poisson)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.40988    0.20357   2.013   0.0441 *
## x            0.31491    0.05861   5.373 7.73e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 72.102  on 49  degrees of freedom
## Residual deviance: 40.293  on 48  degrees of freedom
## AIC: 194.09
##
## Number of Fisher Scoring iterations: 4
```

Deviance

In general, the deviance compares any two models. Here, the fitted glm is compared to a model perfectly overfit to the sample.

- This perfect model is known as the saturated model, or the Bayes optimal model.
- The responses $f_{sat}(x_i)$ equal the observed responses y_i, or $f_{sat}(x_i)$ is the mean of the relevant y_i if there are multiple observations with the same x_i.

In code:

$$f_{sat}(x_i) = \texttt{mean}(\texttt{y}[\texttt{x} == \texttt{xi}])$$

Deviance is the difference between the log-likelihoods of the fitted and saturated models. Times -2.

$$D = -2(L(\text{model}) - L_{sat})$$

Deviance acts kind of like variance. Since likelihood is computed as a sum of contributions from each data point, it's easy to split total deviance into **residual deviances**.

Dispersion

When modeling the lily data we computed the ratio of residual deviance to degrees of freedom and compared this ratio to one. If this ratio was greater than one we called the model *overdispersed*.

In a Poisson distribution, the mean equals the variance. This is really restrictive.

- For a Poisson GLM,
  variance $=$ mean $\rightsquigarrow$ residual deviance $=$ degrees of freedom.
  Each degree of freedom contributes about 1 to the residual deviance.
- Negative binomial/ quasipoisson GLMs usually work better.

Bikeshare example: Poisson vs. Negative Binomial

How does temperature affect bike share use for people out late?

```r
# select only data between 10pm and 3am on non-work days.
riders <- subset(Bikeshare, hr %in% c(22,23,0,1,2) & workingday==0)
# build a Poisson and a negative binomial model
glmRidersP <- glm(bikers~temp, data=riders, family = poisson)
glmRidersNB <- glm.nb(bikers~temp, data=riders)
```

The deterministic parts of the models are quite similar:

```r
glmRidersP$coefficients
```

```
## (Intercept)        temp
##    3.011596    2.466751
```

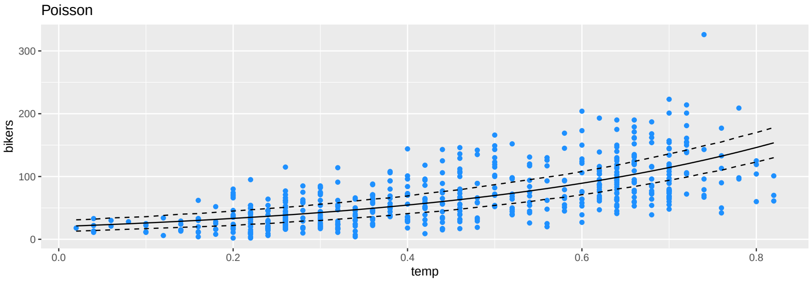
```r
glmRidersNB$coefficients
```

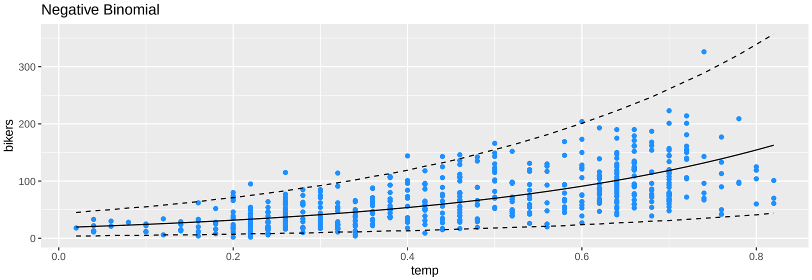
```
## (Intercept)        temp
##    2.927756    2.639448
```

Both of these models say that at these times $\mu_{bikers} \approx e^{3+2.5\cdot temp}$.

Bikeshare example: Poisson vs. Negative Binomial (cont.)

The models differ in their estimates of the variability.





Code for those plots

```r
ggplot(riders, aes(temp,bikers)) +
  geom_point(color="dodgerblue")+
  geom_line(aes(y=exp(predict(glmRidersP))))+
  geom_line(aes(y=qpois(0.025,
                        lambda=exp(predict(glmRidersP)))),
            linetype=2)+
  geom_line(aes(y=qpois(0.975,
                        lambda=exp(predict(glmRidersP)))),
            linetype=2)+
  labs(title="Poisson")

ggplot(riders, aes(temp,bikers))+
  geom_point(color="dodgerblue")+
  geom_line(aes(y=exp(predict(glmRidersNB))))+
  geom_line(aes(y=qnbinom(0.025,
                          mu=exp(predict(glmRidersNB)),
                          size=glmRidersNB$theta)),
            linetype=2)+
  geom_line(aes(y=qnbinom(0.975,
                          mu=exp(predict(glmRidersNB)),
                          size=glmRidersNB$theta)),
            linetype=2)+
  labs(title="Negative Binomial")
```

Null and residual deviance and how they connect to degrees of freedom

The null deviance is the deviance of the null model

`glm(y~1, family=poisson)`

and the residual deviance is the deviance of the fitted model

`glm(y~x, family=poisson)`.

Dividing by degrees of freedom puts these on the same scale in a “number of ways to alter the model” sense of same.

If residual deviance is a lot less than null deviance, the model is working in some sense.

Likelihood and AIC

Akaike's Information Criterion is the negative log-likelihood plus a penalty term for each coefficient that is estimated in the model. Times 2.

```r
(L <- logLik(glm1)[1])
```

```
## [1] -95.04429
```

```r
(k <- length(coef(glm1)))
```

```
## [1] 2
```

```r
-2*L+2*k
```

```
## [1] 194.0886
```

```r
AIC(glm1)
```

```
## [1] 194.0886
```

Interpreting AIC

Given a set of M different models $\text{mod}_1, \ldots, \text{mod}_M$ write AIC_i for the AIC of the i^{th} model and AIC_{min} for the smallest of these values.

- Define $\Delta_i = AIC_i - AIC_{min}$. These differences in AIC values are what is relevant, not the actual AICs.
- An estimate of the relative likelihood of model i is $\exp\left(-\frac{\Delta_i}{2}\right)$. These are sometimes called **evidence ratios**.
- Scaling the relative likelihoods by the sum of the relative likelihoods for all models in consideration give the AIC weight.

$$w_i = \frac{\exp\left(-\frac{\Delta_i}{2}\right)}{\sum_{j=i}^{M} \exp\left(-\frac{\Delta_j}{2}\right)}$$

This is roughly interpretable as the probability that the i^{th} model is the best model.

Customary meanings for differences in AIC

The following are somewhat customary interpretations. Use these rules of thumb similar to the arbitrary bright lines for *p*-values of 0.05, 0.01, etc.

- Models with $\Delta < 2$ are basically indistinguishable from the model with AIC_{min}.
- Models with $\Delta > 10$ are much less predictive or explanatory than the model with AIC_{min}.
- Models with $2 < \Delta < 10$ are in a bit of a grey area. User discretion is advised.

```r
model.sel(glmRidersNB, glmRidersP)
```

```
## Model selection table
##             (Intrc)  temp       family  class init.theta link df    logLik
## glmRidersNB   2.928 2.639 NB(4.0759,l) negbin       4.08  log  3 -2709.218
## glmRidersP    3.012 2.467         p(l)    glm                  2 -6056.141
##                AICc   delta weight
## glmRidersNB  5424.5    0.00      1
## glmRidersP  12116.3 6691.83      0
## Abbreviations:
##  family: NB(4.0759,l) = 'Negative Binomial(4.0759,log)', p(l) = 'poisson(log)'
## Models ranked by AICc(x)
```

Other information critera

Many other modifications of the likelihood are often used. Most behave and are interpreted similarly to AIC.

```r
AIC(glm1)+2*k*(k+1)/(n-k-1)
```

```
## [1] 194.3439
```

```r
AICc(glm1) #from MuMIn. bbmle's doesn't work on glm
```

```
## [1] 194.3439
```

```r
-2*L+log(n)*k
```

```
## [1] 197.9126
```

```r
BIC(glm1)
```

```
## [1] 197.9126
```

Fisher scoring iterations

Fisher's method of iteratively adjusting the parameters to get an increase in maximum likelihood is how the job is done. The number of iterations is how long it took to converge.

Other applications of likelihood

- Likelihood ratio tests

These can be implemented using the `anova` command on a `glm` and specifying `test="LRT"`. See `?anova.glm`. There is also the command `drop1` if you want to do the equivalent of marginal instead of sequential ANOVA.

- Bayesian analysis

The likelihood is a key element in Bayesian analysis, which takes the idea that model parameters are the real variables, and the data are the values that we really know and runs with it.

Likelihood ratio tests

```r
x1 <- runif(n, min=0, max=5); x2 <- runif(n, min=0, max=10)
y <- rpois(n, exp(2+2*x1+x2))
glm2interact <- glm(y~x1*x2, family = poisson)
glm2both <- glm(y~x1+x2, family = poisson)
glm2x1 <- glm(y~x1, family = poisson)
glm2x2 <- glm(y~x2, family=poisson)
glm2null <- glm(y~1, family = poisson)
anova(glm2null,glm2x1,glm2both, glm2interact, test="LRT")
```

```
## Analysis of Deviance Table
##
## Model 1: y ~ 1
## Model 2: y ~ x1
## Model 3: y ~ x1 + x2
## Model 4: y ~ x1 * x2
##   Resid. Df Resid. Dev Df   Deviance Pr(>Chi)
## 1        49 3919013108
## 2        48 2340401353  1 1578611755   <2e-16 ***
## 3        47         59  1 2340401294   <2e-16 ***
## 4        46         59  1          0   0.9837
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Could also use dredge

```
glm2interact <- glm(y~x1*x2, family = poisson, na.action = "na.fail")
dredge(glm2interact)
```

```
## Fixed term is "(Intercept)"

## Global model call: glm(formula = y ~ x1 * x2, family = poisson, na.action = "na.fail")
## ---
## Model selection table
##    (Int)    x1     x2     x1:x2 df        logLik         AICc        delta
## 4  1.999 2.000 1.0000             3 -3.768560e+02        760.2 0.000000e+00
## 8  1.999 2.000 1.0000 1.554e-06  4 -3.768560e+02        762.6 2.370000e+00
## 3 12.290       0.6419             2 -1.046310e+09 2092620067.4 2.092619e+09
## 2 11.310 1.497                    2 -1.170201e+09 2340402051.7 2.340401e+09
## 1 16.740                          1 -1.959507e+09 3919013804.8 3.919013e+09
##   weight
## 4  0.766
## 8  0.234
## 3  0.000
## 2  0.000
## 1  0.000
## Models ranked by AICc(x)
```

From the MuMIn manual: Users should keep in mind the hazards that a "thoughtless approach" of evaluating all possible models poses. Although this procedure is in certain cases useful and justified, it may result in selecting a spurious "best" model, due to the model selection bias.

"Let the computer find out" is a poor strategy and usually reflects the fact that the researcher did not bother to think clearly about the problem of interest and its scientific setting (Burnham and Anderson, 2002).

Other applications of `mle2`

This tool can do maximum likelihood estimation to find parameters for many sorts of models that `glm` can't. It's tricky to use, and can rely in unexpected ways on the `start` values, so often it's easier to find a prepackaged tool like `glm`.

But if you know exactly the model you want to build and have good guesses for the parameters that you hope to estimate, this could be the tool for you.

Things in `summary.lm` but not `summary.glm`

There are two things that we are missing.

- R^2

There's no consensus about what we can or should use to replace R^2. There's a good discussion of this by Bolker here: [https://bbolker.github.io/mixedmodels-misc/glmmFAQ.html#how-do-i-compute-a-coefficient-of-determination-r2-or-an-analogue-for-glmms]

- An overall p-value.

There's no obvious replacement for F, or clear reason why we can't keep using F. But since there are likelihood ratios (and other metrics of goodness-of-fit) it would be presumptuous to give an overall p-value based on an F test.

Bolker, Benjamin M. 2008. *Ecological Models and Data in R*. Princeton University Press.