Bayesian Inference for Logistic Regression Parameters

Bayesian inference for logistic analyses follows the usual pattern for all Bayesian analyses:

1. Write down the likelihood function of the data.
2. Form a prior distribution over all unknown parameters.
3. Use Bayes theorem to find the posterior distribution over all parameters.

We have applied this generic formulation so far to univariate problems with binomial distributions, normal means (with variance known and unknown), multinomial parameters, Poisson mean parameters, and, in the multivariate case, to linear regression parameters with normally distributed residuals.

The latter case is most similar to Bayesian inference in logistic regression, but in some ways logistic regression is even simpler, because there is no variance term to estimate, only the regression parameters.

For logistic regression, the above three steps are summarized as follows:

Likelihood function: As usual, the likelihood function used by Bayesians matches that from frequentist inference.

In particular, recall that once we have the probability of success (which in logistic regression varies from one subject to another, depending on their covariates), the likelihood contribution from the i^{th} subject is binomial:

$$\text{likelihood}_i = \pi(x_i)^{y_i}(1-\pi(x_i))^{(1-y_i)}$$

where $\pi(x_i)$ represents the probability of the event for subject i who has covariate vector x_i, and y_i indicates the presence, $y_i = 1$, or absence $y_i = 0$ of the event for that subject.

Of course, in logistic regression, we know that

$$\pi(x) = \frac{e^{\beta_0+\beta_1 X_1+\ldots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\ldots+\beta_p X_p}}$$

so that the likelihood contribution from the i^{th} subject is

$$\text{likelihood}_i = \left(\frac{e^{\beta_0+\beta_1 X_{i1}+\ldots+\beta_p X_{ip}}}{1+e^{\beta_0+\beta_1 X_{i1}+\ldots+\beta_p X_{ip}}}\right)^{y_i} \left(1-\frac{e^{\beta_0+\beta_1 X_{i1}+\ldots+\beta_p X_{ip}}}{1+e^{\beta_0+\beta_1 X_{i1}+\ldots+\beta_p X_{ip}}}\right)^{(1-y_i)}$$

Since individual subjects are assumed independent from each other, the likelihood function over a data set of n subjects is then

$$\text{likelihood} = \prod_{i=1}^{n}\left[\left(\frac{e^{\beta_0+\beta_1 X_{i1}+\ldots+\beta_p X_{ip}}}{1+e^{\beta_0+\beta_1 X_{i1}+\ldots+\beta_p X_{ip}}}\right)^{y_i} \left(1-\frac{e^{\beta_0+\beta_1 X_{i1}+\ldots+\beta_p X_{ip}}}{1+e^{\beta_0+\beta_1 X_{i1}+\ldots+\beta_p X_{ip}}}\right)^{(1-y_i)}\right]$$

Prior distribution:

The set of unknown parameters consists of β_0, β_1, $\ldots$, β_p.

In general, any prior distribution can be used, depending on the available prior information. The choice can include informative prior distributions if something is known about the likely values of the unknown parameters, or "diffuse" or "non-informative" priors if either little is known about the coefficient values or if one wishes to see what the data themselves provide as inferences.

If informative prior distributions are desired, it is often difficult to give such information on the logit scale, i.e., on the β parameters directly.

One may prefer to provide prior information on the $OR = exp(\beta)$ scale, and mathematically transform back to the logit scale. Alternatively, one can take various situations (e.g., male, 73 years old, on drug A, etc.) and derive prior distributions on the probability scale. If sufficient elicitations of this type are made, one can mathematically transform back to the coefficient scale. One can find free software (e.g., a program called ELICITOR) that facilitates such prior derivations.

For this course, however, we will use the most common priors for logistic regression parameters, which are of the form:

$$\beta_j \sim N(\mu_j, \sigma_j^2)$$

The most common choice for μ is zero, and σ is usually chosen to be large enough to be considered as non-informative, common choices being in the range from $\sigma = 10$ to $\sigma = 100$.

Posterior distribution via Bayes Theorem:

The posterior distribution is derived by multiplying the prior distribution over all parameters by the full likelihood function, so that

$$\begin{aligned} \text{posterior} &= \prod_{i=1}^{n} \left[\left(\frac{e^{\beta_0+\beta_1 X_{i1}+\ldots+\beta_p X_{ip}}}{1+e^{\beta_0+\beta_1 X_{i1}+\ldots+\beta_p X_{ip}}} \right)^{y_i} \left(1 - \frac{e^{\beta_0+\beta_1 X_{i1}+\ldots+\beta_p X_{ip}}}{1+e^{\beta_0+\beta_1 X_{i1}+\ldots+\beta_p X_{ip}}} \right)^{(1-y_i)} \right] \\ &\times \prod_{j=0}^{p} \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left\{ -\frac{1}{2} \left(\frac{\beta_j - \mu_j}{\sigma_j} \right)^2 \right\} \end{aligned}$$

the latter part of the above expression being recognized as normal distributions for the β parameters.

Of course, the above expression has no closed form expression, and even if it did, we would have to perform multiple integration to obtain the marginal distribution for each coefficient. So, as is usual for Bayesian analysis, we will use the Gibbs sampler as implemented by WinBUGS to solve approximate the properties of the marginal posterior distributions for each parameter.

As was the case for frequentist inference, taking $\exp(\beta)$ provides the odds ratio for a one unit change of that parameter. We will see how easy it is to carry out such inferences from a Bayesian viewpoint using WinBUGS.

Let's see some examples. We will begin with a simple simulated model, and move on to multivariate examples. We will program each in WinBUGS.

Simple Logistic Regression Program Using WinBUGS

We will investigate a simulated logistic regression model of bone fractures with independent variables age and sex. The true model had: alpha = -25, b.sex = 0.5, b.age = 0.4. With so few data points and three parameters to estimate, do not expect posterior means/medians to equal the correct values exactly, but all would most likely be in the 95% intervals.

Model

```
model
{
    for (i in 1:n) {

                                        #  Linear regression on logit
  logit(p[i]) <- alpha + b.sex*sex[i] + b.age*age[i]

                                        #  Likelihood function for each data point
    frac[i]   ~ dbern(p[i])
    }
    alpha     ~ dnorm(0.0,1.0E-4)   #  Prior for intercept
    b.sex     ~ dnorm(0.0,1.0E-4)   #  Prior for slope of sex
    b.age     ~ dnorm(0.0,1.0E-4)   #  Prior for slope of age
}
```

Data

list(sex=c(1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1,
1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1,
0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0,
1, 1, 1, 1, 1),
age= c(69, 57, 61, 60, 69, 74, 63, 68, 64, 53, 60, 58, 79, 56, 53, 74, 56, 76, 72, 56, 66,
52, 77, 70, 69, 76, 72, 53, 69, 59, 73, 77, 55, 77, 68, 62, 56, 68, 70, 60, 65, 55, 64, 75,
60, 67, 61, 69, 75, 68, 72, 71, 54, 52, 54, 50, 75, 59, 65, 60, 60, 57, 51, 51, 63, 57, 80,
52, 65, 72, 80, 73, 76, 79, 66, 51, 76, 75, 66, 75, 78, 70, 67, 51, 70, 71, 71, 74, 74, 60,
58, 55, 61, 65, 52, 68, 75, 52, 53, 70),
frac=c(1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1,
1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0,
1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1,
0, 0, 1),
n=100)

Initial Values

list(alpha=0, b.sex=1, b.age=1)

Results

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
alpha	-22.55	5.013	0.6062	-34.33	-21.64	-14.29	1001	4000
b.age	0.3559	0.07771	0.009395	0.227	0.3418	0.5338	1001	4000
b.sex	1.405	0.7719	0.05094	-0.0387	1.374	3.031	1001	4000
p[1]	0.9575	0.03153	0.002943	0.879	0.9647	0.9952	1001	4000
p[2]	0.307	0.09828	0.004853	0.13	0.3012	0.5082	1001	4000
p[3]	0.6308	0.1041	0.003344	0.4166	0.6356	0.8178	1001	4000
p[4]	0.2477	0.103	0.007281	0.07738	0.2379	0.4728	1001	4000

(etc...)

So, as expected, CrIs are wide, but all contain the true values.

Can compare the results to a frequentist analysis of the same data:

```
> logistic.regression.or.ci( glm( frac ~ age + sex, family = binomial)   )

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -21.85041    4.42511  -4.938 7.90e-07 ***
age           0.34467    0.06857   5.027 4.99e-07 ***
sex           1.36110    0.73336   1.856   0.0635 .
```

Note the very similar results. Small differences arise from two different places: Frequentist results are not particularly accurate in small sample sizes, and Bayesian results are not that accurate (check the MC error column) because only 4000 iterations were run.

Bayesian analysis accuracy here can be improved by running more iterations. Nothing much can be done for the frequentist analysis except to change to a more "exact" method (beyond scope of this course).

We will next look at another simple example, but include predictions and odds ratios.

CHD and age example

We have already seen this data set:

Age	CHD	Age	CHD	Age	CHD	Age	CHD
20	0	35	0	44	1	55	1
23	0	35	0	44	1	56	1
24	0	36	0	45	0	56	1
25	0	36	1	45	1	56	1
25	1	36	0	46	0	57	0
26	0	37	0	46	1	57	0
26	0	37	1	47	0	57	1
28	0	37	0	47	0	57	1
28	0	38	0	47	1	57	1
29	0	38	0	48	0	57	1
30	0	39	0	48	1	58	0
30	0	39	1	48	1	58	1
30	0	40	0	49	0	58	1
30	0	40	1	49	0	59	1
30	0	41	0	49	1	59	1
30	1	41	0	50	0	60	0
32	0	42	0	50	1	60	1
32	0	42	0	51	0	61	1
33	0	42	0	52	0	62	1
33	0	42	1	52	1	62	1
34	0	43	0	53	1	63	1
34	0	43	0	53	1	64	0
34	1	43	1	54	1	64	1
34	0	44	0	55	0	65	1
34	0	44	0	55	1	69	1

A WinBUGS program for these data could be:

```
model {
    for (i in 1:n) {


  logit(p[i]) <- alpha + b.age*age[i]   #  Linear regression on logit for age

                                        #  Likelihood function for each data point
    CHD[i]    ~ dbern(p[i])
    }
    alpha      ~ dnorm(0.0,1.0E-4)   #  Prior for intercept
    b.age      ~ dnorm(0.0,1.0E-4)   #  Prior for slope of age

#  Now to calculate the odds ratios for various functions of age
```

```
#  OR per unit change in age

   or.age <- exp(b.age)

#  OR per decade change in age

   or.age10 <- exp(10*b.age)

#  OR per five year change in age

   or.age5 <- exp(5*b.age)

#  We can also make various predictions

#  Predict fracture rate for 20 year old

   pred.age20 <- exp(alpha + b.age*20)/(1+ exp(alpha + b.age*20))

#  Predict fracture rate for 50 year old

   pred.age50 <- exp(alpha + b.age*50)/(1+ exp(alpha + b.age*50))

#  Predict fracture rate for 70 year old

   pred.age70 <- exp(alpha + b.age*70)/(1+ exp(alpha + b.age*70))

}

#  Data

list(CHD = c(0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0,
1 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0, 1 , 0 , 0 , 1, 0 ,
0  , 0 , 0 , 1 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 1 , 0, 0 ,
1 , 1  , 0 , 1 , 0 , 1 , 0 , 0 , 1 , 0 , 1 , 1 , 0 , 0 , 1 , 0, 1 ,
0 , 0 , 1  , 1 , 1 , 1 , 0 , 1 , 1 , 1 , 1 , 1 , 0 , 0 , 1 , 1,
1 , 1 , 0 , 1 , 1  , 1 , 1 , 0 , 1 , 1 , 1 , 1 , 1 , 0 , 1 , 1 , 1),
n=100,
age = c(20, 23, 24, 25, 25, 26, 26, 28, 28, 29, 30, 30, 30, 30, 30,
30, 32, 32, 33, 33, 34, 34, 34, 34, 34, 35, 35, 36, 36, 36, 37, 37,
37, 38, 38, 39, 39, 40, 40, 41, 41, 42, 42, 42, 42, 43, 43, 43, 44,
44, 44, 44, 45, 45, 46, 46, 47, 47, 47, 48, 48, 48, 49, 49, 49, 50,
50, 51, 52, 52, 53, 53, 54, 55, 55, 55, 56, 56, 56, 57, 57, 57, 57,
57, 57, 58, 58, 58, 59, 59, 60, 60, 61, 62, 62, 63, 64, 64, 65, 69))
```

```
#  Inits

list(alpha = 0, b.age=0)

# Results

node        mean    sd      MC error    2.5%     median  97.5%   start   sample
alpha       -5.465  1.166   0.05372    -7.833   -5.426  -3.302   1001    20000
b.age       0.1142  0.02476 0.001142    0.06816  0.1134  0.1646  1001    20000
or.age      1.121   0.02783 0.001285    1.071    1.12    1.179   1001    20000
or.age10    3.232   0.8271  0.03851     1.977    3.108   5.184   1001    20000
or.age5     1.784   0.2238  0.01039     1.406    1.763   2.277   1001    20000
pred.age20  0.0484  0.03161 0.001302    0.01013  0.04084 0.1294  1001    20000
pred.age50  0.5599  0.06235 8.727E-4    0.4388   0.5599  0.6821  1001    20000
pred.age70  0.9143  0.04954 0.001894    0.7902   0.9252  0.9785  1001    20000

#  Note ease in getting ORs for any value of age change, and predictions.

#  Compare to frequentist results we saw earlier

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.30945    1.13365  -4.683 2.82e-06 ***
age          0.11092    0.02406   4.610 4.02e-06 ***

$intercept.ci
[1] -7.531374 -3.087533

$slopes.ci
[1] 0.06376477 0.15807752

$OR
     age
1.117307

$OR.ci
[1] 1.065842 1.171257

# Note that results are extremely similar, because we ran
# more iterations, and because sample size is larger.

# Can get MC errors smaller by running more iterations...

node     mean     sd      MC error  2.5%     median  97.5%   start   sample
b.age    0.1145  0.02475  5.187E-4  0.06807  0.1137  0.1655  1001    100000
```

```
or.age   1.122   0.02782  5.838E-4  1.07     1.12    1.18    1001    100000

#  ...but nothing much really changes.  Differences due to priors
#  and/or frequentist normal approximations, sample size not that
#  large still.
```

One more example

Finally, consider again the multivariate model we saw before on low birth weights.

Recall the data description:

Variable	Coding
Low Birth Weight (0 = Birth Weight $\geq$ 2500g, 1 = Birth Weight $<$ 2500g)	low
Age of the Mother in Years	age
Weight in Pounds at the Last Menstrual Period	lwt
Race (1 = White, 2 = Black, 3 = Other)	race
Smoking Status During Pregnancy (1 = Yes, 0 = No)	smoke
History of Premature Labor (0 = None 1 = One, etc.)	ptl
History of Hypertension (1 = Yes, 0 = No)	ht
Presence of Uterine Irritability (1 = Yes, 0 = No)	ui
Number of Physician Visits During the First Trimester (0 = None, 1 = One, 2 = Two, etc.)	ftv
Birth Weight in Grams	bwt

Unlike frequentist analyses, there is no problem in declaring variables as factors or not in WinBUGS. This simplifies matters somewhat, but also means that dummy variables need to be created "manually" before the analysis is run.

The only such variable here is race, which we will need to recode as two separate dummy variables (as is done internally by R when the variable is declared to be a factor).

```
model {
    for (i in 1:189) {

  logit(p[i]) <-  alpha + b.age*age[i]   + b.lwt*lwt[i]
                  + b.race2*race2[i] + b.race3*race3[i]
                  +b.smoke*smoke[i]  + b.ptl*ptl[i]
                  + b.ht*ht[i]  + b.ui*ui[i]   + b.ftv*ftv[i]

                                    #  Likelihood function for each data point
    low[i]   ~ dbern(p[i])
    }
    alpha     ~ dnorm(0.0,1.0E-2)   #  Prior for intercept
    b.age     ~ dnorm(0.0,1.0E-2)   #  Priors for slopes
    b.lwt     ~ dnorm(0.0,1.0E-2)
    b.race2     ~ dnorm(0.0,1.0E-2)
    b.race3     ~ dnorm(0.0,1.0E-2)
    b.smoke     ~ dnorm(0.0,1.0E-2)
    b.ptl     ~ dnorm(0.0,1.0E-2)
    b.ht     ~ dnorm(0.0,1.0E-2)
    b.ui     ~ dnorm(0.0,1.0E-2)
    b.ftv     ~ dnorm(0.0,1.0E-2)

#  Now to calculate the odds ratios for various functions of age

#  OR per decade change in age

   or.age10 <- exp(10*b.age)

#  OR for smoke

  or.smoke <- exp(b.smoke)

#  Predict rate for 40 year old, with lwt, smoker and ht

   pred.age20 <- exp(alpha + b.age*40 + b.lwt + b.smoke + b.ht)
              /(1+ exp(alpha + b.age*40 + b.lwt + b.smoke + b.ht))
}


#  Inits

list(alpha = 0, b.age=0, b.lwt=0, b.race2=0, b.race3=0,
             b.smoke=0, b.ptl=0, b.ht=0)
```

```
#  Data

low[]   age[]  lwt[]   race2[]  race3[] smoke[]  ptl[]  ht[]  ui[]  ftv[]
   0      19    182      1         0      0       0      0     1     0
   0      33    155      0         1      0       0      0     0     1
   0      20    105      0         0      1       0      0     0     1
   0      21    108      0         0      1       0      0     1     1
   0      18    107      0         0      1       0      0     1     0
   0      21    124      0         1      0       0      0     0     0
   0      22    118      0         0      0       0      0     0     1
   0      17    103      0         1      0       0      0     0     1
   0      29    123      0         0      1       0      0     0     1
   0      26    113      0         0      1       0      0     0     0
..............etc.......................
END

#  Note:  Compulsory blank line above after END statement.

#  Results

node      mean     sd      MC error    2.5%     median   97.5%    start   sample
alpha      0.7015   1.272   0.06921    -1.659    0.6785   3.368    1001    20000
b.age     -0.02903  0.04015 0.001827   -0.1112  -0.02705  0.0462   1001    20000
b.ftv     -0.07864  0.3802  0.00588    -0.8317  -0.0774   0.6678   1001    20000
b.ht       1.964    0.7269  0.01085     0.5814   1.951    3.447    1001    20000
b.lwt     -0.01705  0.00689 3.053E-4   -0.03086 -0.017   -0.00379  1001    20000
b.ptl      0.5933   0.3651  0.00421    -0.1112   0.5897   1.327    1001    20000
b.race2    1.313    0.5474  0.006877    0.2388   1.308    2.402    1001    20000
b.race3    0.8732   0.4706  0.01131    -0.04335  0.8692   1.822    1001    20000
b.smoke    0.9452   0.4286  0.008723    0.1257   0.942    1.798    1001    20000
b.ui       0.7688   0.4802  0.005109   -0.173    0.7688   1.714    1001    20000
or.age10   0.8096   0.3299  0.01425     0.3288   0.763    1.587    1001    20000
or.smoke   2.825    1.307   0.02651     1.134    2.565    6.035    1001    20000
pred       0.8559   0.163   0.005065    0.3824   0.9182   0.9956   1001    20000
```

Results similar to frequentist results previously seen, differences mostly owing to some coding changes.

Conclusion

In these sorts of simple examples, there is not really much difference between Bayesian and frequentist inferences.

More generally, we have these comparisons:

- Not repeated above since we have seen this example before, but all preliminary steps apply equally to Bayesian analysis as to frequentist analysis.
- Maybe frequentist analysis of standard models easier (maybe depends on taste/habit as well).
- Bayesian models more flexible, handles more complex models.
- Bayesian model selection probably superior (BIC/AIC).
- Bayesian hierarchical models easier to extend to many levels.
- Philosophical differences.
- Bayesian analysis more accurate in small samples (but then may depend on priors).
- Bayesian models can incorporate prior information.