Bayesian Inference for Linear and Logistic Regression Parameters

Bayesian inference for simple linear and logistic regression parameters follows the usual pattern for all Bayesian analyses:

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

- We have applied this generic formulation so far to problems with binomial distributions, normal means (with variance known), multinomial parameters, and to the Poisson mean parameter. All of these problems involved only one parameter at a time (strictly speaking, more than one in the multinomial case, but only a single Dirichlet distribution was used, so all parameters were treated together, as if it were just one parameter).
- What makes regression different is that we have three unknown parameters, since the intercept and slope of the line, α and β are unknown, and the residual standard deviation (for linear regression), σ is also unknown.
- Hence our Bayesian problem becomes slightly more complicated, since we are in a multi-parameter situation. Thus, we will use the Gibbs sampler, as automatically implemented in WinBUGS software.

Introduction to WinBUGS

WinBUGS is a free program available from the Biostatistics Unit of the Medical Research Council in the UK (see link on course web page). Installing WinBUGS is straightforward, one downloads the single file required, typically called winbugs14.exe, and runs that file to install in typical windows fashion. However, this allows only a limited use of WinBUGS, and one also need to install a key (again, this is free) for unlimited use. Instructions are on the WinBUGS home page, and please let me know if you experience any difficulty

installing the program or the key. All computers in the basement of Purvis Hall have WinBUGS installed on them for your use.

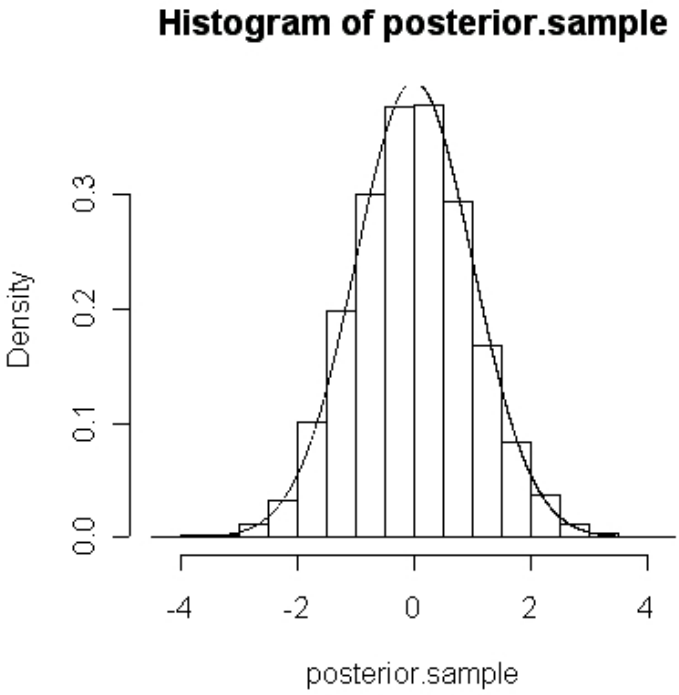
Once installed, a WinBUGS program consists of three parts, all of which can be placed into a single file, or as three separate files (or two files):

Main program: A set of lines that let WinBUGS know what the prior and likelihood of the model are.

Data set: The data that will be used, either formatted as an R like list, or in rectangular format. Note that you can have as many data sets as you wish, and can mix formatting types when entering data.

Initial Values: These are optional (but generally a good idea to include), and are used by WinBUGS to start its algorithm.

WinBUGS solves problems in Bayesian analysis by multiplying the prior by the likelihood, and then taking samples from the posterior distributions via an iterative algorithm called the Gibbs sampler. Thus, rather than get an exact formula for the posterior distribution, WinBUGS returns samples from it, as shown below:



WinBUGS can be used to solve a *very* wide class of problems, much wider than standard software packages. It must be used carefully, as the Gibbs sampler algorithm does not always converge. Good practice dictates to run WinBUGS several times (say, three times) to ensure that the answer is always the same (or extremely close). For problems in this course, convergence will seldom, if ever, be a problem.

Running WinBUGS

Follow these steps to produce analyses in WinBUGS:

1. Open WinBUGS by clicking on the WinBUGS icon on desktop (after installation).
2. Either open an existing WinBUGS file (typical extension is *.odc) if continuing from an already created file, or open a new window from the file menu.

3. Type in the program (or cut and paste if a program already exists, typically will be the case in this course).

4. Below the program (or if you prefer, in another window inside of WinBUGS), type in or cut and paste in the data set, remembering that the format must be either rectangular or list-like (see examples on course web page).

5. In list format, type in initial values (see examples on web page).

   For our normal problem with unknown variance, the WinBUGS window will typically look like this:

```
model
{                          #  Program must start with model
                           #  statement and open {
  for (i in 1:27)          #  Loop over number of data points
   {
    x[i] ~ dnorm(mu,tau) #  Enter the 27 likelihood terms
   }
  mu  ~ dnorm(70,0.04)     #  Prior for the normal mean, mu, 0.04=1/25
  sigma ~ dunif(0,50)      #  Prior for the normal standard deviation, sigma
  tau<-1/(sigma*sigma)     #  WinBUGS normal form needs 1/sigma^2
      }                    #  End of program part

#  Data                    #  Data section

list(x=c(76, 71, 82, 63, 76, 64, 64, 74, 70, 64, 75, 81, 75, 78, 66,
62, 79, 82, 78, 62, 72, 83, 79, 41, 80, 77, 67))

#  Initial Vlaues          #  Initial Values section

list(mu=60, sigma = 20)
```

6. Once all input is ready, WinBUGS must be run to produce the output, following these steps:

   (a) Open the Model $\rightarrow$ Specification menu item, highlight the word "model" in the program, and then click on "check model". This checks the model for syntax errors. Watch for (often quite cryptic) error messages on bottom status line, if any errors occur.

(b) Highlight the word "list" in the data section of the program, and click on "load data". This loads in your data. Repeat this step as many times as needed to load in all your data. If in rectangular array style, highlight the first line (variable names) rather than the word "list".

(c) Click on compile (assuming all is fine so far...if not, find error(s) and repeat first two steps).

(d) Highlight the word "list" in the initial values section of the program, and click on "load inits". This loads in your initial values. If you did not provide all initial values, click on "gen inits" to have WinBUGS do this automatically for you (not always a good idea).

(e) WinBUGS can now be run to produce posterior distributions. First, run some "burn-in" or "throw away" values, to ensure convergence. Typically, 2000 should be (more than) sufficient for models in this course. To create the burn-in, open the Model → Update menu item, change the default 1000 value to 2000, and click on the "update" button.

(f) To run further iterations "to keep", open the Inference → Samples menu item. Enter all unknown variables you wish inferences on into the window (in this case, mu and sigma), clicking "set" after each one. Then go back to the update box, which should still be open, and run the number of iterations you want. We will use 10,000, so enter 10000 and click again on the update button.

(g) Finally, we can look at summaries of the posterior distributions. Going back to the Samples dialogue box, type a " * " in the window (meaning look at all variables tracked), and then click on the various items to get the desired summaries. Most useful for us will be "Stats" and "density", to get basic posterior inferences such as means, medians and 95% credible intervals, and density plots (somewhat crude). The results are:

```
node  mean  sd    MC error 2.5%  median 97.5% start sample
mu    71.63 1.727 0.01698  68.24 71.62  74.97 2001  10000
sigma 9.69  1.43  0.01567  7.358 9.532  12.92 2001  10000
```

For the mean, note that our posterior has mean of about 71.6 (compare to 71.69 found previously), and the standard deviation for this parameter is 1.73, compared to 1.63 found previously. As we have an extra parameter to estimate, this decrease in the accuracy in

estimating μ is expected. Note that we have a new parameter to estimate, σ, which was assumed known before.

Having been introduced to WinBUGS, we can now look at Bayesian linear regression, first in general terms, then in terms of WinBUGS programming and inferences. After that, we will look at logistic regression.

Brief Sketch of Bayesian linear regression

Recall the three steps: prior $\rightarrow$ likelihood $\rightarrow$ posterior.

1. We need a joint prior distribution over α, β, and σ. We will specify these as three independent priors [which when multiplied together will produce a joint prior]. One choice is:

- $\alpha \sim \text{uniform}[-\infty, +\infty]$
- $\beta \sim \text{uniform}[-\infty, +\infty]$
- $log(\sigma) \sim \text{uniform}[-\infty, +\infty]$

With this choice, we can approximate the results from a frequentist regression analysis. Another possible choice is (more practical for WinBUGS):

- $\alpha \sim \text{Normal}(0, 0.0001)$
- $\beta \sim \text{Normal}(0, 0.0001)$
- $\sigma \sim \text{uniform}[0, 100]$

These are very diffuse (typically, depends on scale of data), so approximate the first set of priors. Of course, if real prior information is available, it can be incorporated.

Notes:

- The need for the log in first set of priors comes from the fact the the variance must be positive. The prior on σ is equivalent to a density that is proportional to $\frac{1}{\sigma^2}$.
- We specify a non-informative prior distribution of these three parameters. Of course, we can also include prior information when available, but this is beyond the scope of this course.
- First set of priors is in fact "improper" because their densities do not integrate to one, since the area under these curves in infinite! In general this is to be avoided since sometimes it can cause problems with posterior distributions. This is *not* one of those problem cases, however, and it is sometimes convenient to use a "flat" prior everywhere, so it is mentioned here (even if we use proper priors for the rest of this lecture).

2. Likelihood function in regression:

- As is often the case, the likelihood function used in a Bayesian analysis is the same as the one used for the frequentist analysis.
- Recall that we have normally distributed residuals, $\epsilon \sim N(0, \sigma^2)$
- Recall that the mean of the regression line, given that we know α and β is $y = \alpha + \beta \times x$.
- Putting this together, we have $y \sim N(\alpha + \beta \times x, \sigma^2)$.
- So for a single patient with observed value x_i, we have $y \sim N(\alpha + \beta \times x_i, \sigma^2)$
- So for a single patient, the likelihood function is:

$$f(y_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{\frac{(y_i - (\alpha + \beta \times x_i))^2}{\sigma^2}\right\}$$

- So for a group of n patients each contributing data (x_i, y_i), the likelihood function is given by

$$\prod_{i=1}^{n} f(y_i) = f(y_1) \times f(y_2) \times f(y_3) \times \ldots \times f(y_n)$$

- So the likelihood function is simply a bunch of normal densities multiplied together ... a multivariate normal likelihood of dimension n.

3. Posterior densities in regression

- Bayes theorem now says to multiply the likelihood function (multivariate normal) with the prior. For the first prior set, the joint prior simply is $1 \times 1 \times \frac{1}{\sigma^2}$. For the second, it is two normal distributions multiplied by a uniform. Since the uniform is equal to 1 everywhere, this reduced to a product of two normal distributions.
- For the first prior set, the posterior distribution simply is:

$$\prod_{i=1}^{n} f(y_i) \times \frac{1}{\sigma^2}.$$

- This is a three dimensional posterior involving α, β, and σ^2.
- By integrating this posterior density, we can obtain the marginal densities for each of α, β, and σ^2.
- After integration (tedious details omitted):
  - $\alpha \sim t_{n-2}$
  - $\beta \sim t_{n-2}$
  - $\sigma^2 \sim$ Inverse Chi-Square (so $1/\sigma^2 \sim$ Chi-Square)
- Note the similar results given by Bayesian and frequentist approaches for α and β, and, in fact, the means and variances are the same as well, assuming the "infinite priors" listed above are used.
- When the second prior set is used, the resulting formulae are more complex, and WinBUGS will be used. However, as both sets of priors are very "diffuse" or "non-informative", numerically, results will be similar to each other, and both will be similar to the inferences given by the frequentist approach (but interpretations will be very different).
- Although the t distributions listed above can be used, Bayesian computations usually carried out via computer programs. We will use WinBUGS to compute Bayesian posterior distributions for regression and logistic problems.

- Bayes approach also suggests different ways to assess goodness of fit and model selection (we will see this later).

Example

Consider the following data:

Community Number	DMF per 100 children	Fluoride Concentration in ppm
1	236	1.9
2	246	2.6
3	252	1.8
4	258	1.2
5	281	1.2
6	303	1.2
7	323	1.3
8	343	0.9
9	412	0.6
10	444	0.5
11	556	0.4
12	652	0.3
13	673	0.0
14	703	0.2
15	706	0.1
16	722	0.0
17	733	0.2
18	772	0.1
19	810	0.0
20	823	0.1
21	1027	0.1

We will now analyze these data from a Bayesian viewpoint, using WinBUGS. The program we need is:

```
model
{
 for (i in 1:21)                     # loop over cities
    {
     mu.dmf[i] <- alpha + beta*fl[i] # regression equation
     dmf[i] ~ dnorm(mu.dmf[i],tau)   # distribution individual values
    }
    alpha ~ dnorm(0.0,0.000001)      # prior for intercept
    beta  ~ dnorm(0.0,0.000001)      # prior for slope
    sigma ~ dunif(0,400)             # prior for residual SD
    tau     <- 1/(sigma*sigma)       # precision required by WinBUGS
 for (i in 1:21)
    {                                # calculate residuals
     residual[i] <- dmf[i] - mu.dmf[i]
    }
pred.mean.1.7 <- alpha + beta*1.7         # mean prediction for fl=1.7
pred.ind.1.7 ~ dnorm(pred.mean.1.7, tau)  # individual pred for fl=1.7
}
```

Data are entered in one of the following two formats (both equivalent, note blank line required after "END" statement):

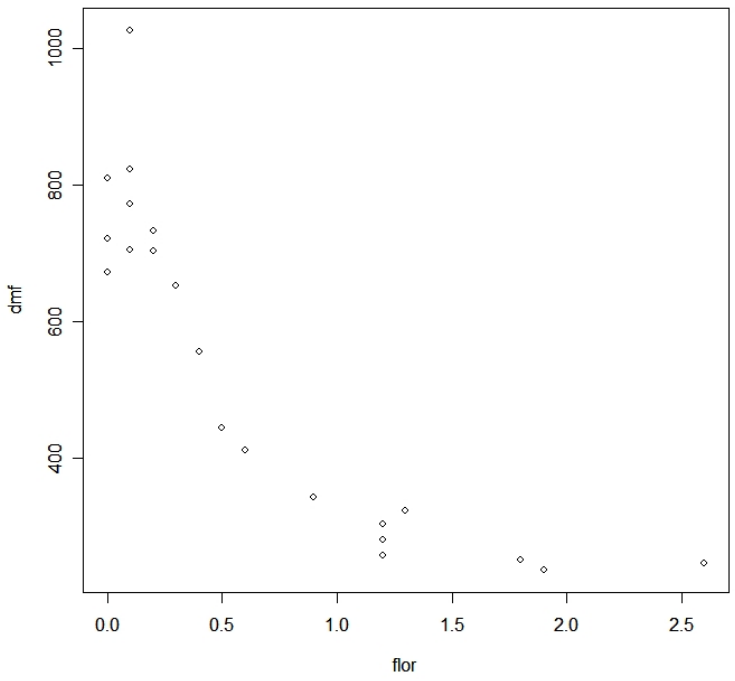
```
 dmf[]   fl[]
 236     1.9
 246     2.6
 252     1.8
 258     1.2
 281     1.2
 303     1.2
 323     1.3
 343     0.9
 412     0.6
 444     0.5
 556     0.4
 652     0.3
 673     0.0
 703     0.2
 706     0.1
 722     0.0
```

```
 733      0.2
 772      0.1
 810      0.0
 823      0.1
 1027     0.1
END
```

or

```
list(dmf=c( 236, 246, 252, 258, 281, 303, 323, 343, 412, 444, 556, 652,
673, 703, 706, 722, 733, 772, 810, 823, 1027), fl=c( 1.9, 2.6, 1.8, 1.2,
1.2, 1.2, 1.3, 0.9, 0.6, 0.5, 0.4, 0.3, 0.0, 0.2, 0.1, 0.0, 0.2, 0.1,
0.0, 0.1, 0.1))
```

Let's look at a graph of these data:



We will use these initial values (rather arbitrary, just to get into right general area to start):

```
list(alpha=100, beta = 0, sigma=100)
```

Running 2000 burn-in values and then 10,000 further iterations, produces the following results:

node	mean	sd	2.5%	median	97.5%
alpha	730.1	41.45	646.8	730.6	809.7
beta	-277.4	40.86	-358.8	-277.4	-196.8
mu.dmf[1]	203.0	57.16	89.49	203.0	314.1
mu.dmf[2]	8.869	82.93	-152.8	8.885	171.1
mu.dmf[3]	230.8	53.72	124.3	230.6	335.6
mu.dmf[4]	397.2	35.98	325.8	397.5	467.1
mu.dmf[5]	397.2	35.98	325.8	397.5	467.1
mu.dmf[6]	397.2	35.98	325.8	397.5	467.1
mu.dmf[7]	369.5	38.42	293.3	369.6	444.2
mu.dmf[8]	480.4	30.81	418.9	480.4	540.8
mu.dmf[9]	563.7	30.09	503.0	563.4	621.8
mu.dmf[10]	591.4	30.94	529.2	591.4	651.0
mu.dmf[11]	619.1	32.29	554.6	619.3	680.9
mu.dmf[12]	646.9	34.08	578.9	647.1	712.3
mu.dmf[13]	730.1	41.45	646.8	730.6	809.7
mu.dmf[14]	674.6	36.24	602.0	675.1	744.8
mu.dmf[15]	702.4	38.72	624.6	702.7	777.0
mu.dmf[16]	730.1	41.45	646.8	730.6	809.7
mu.dmf[17]	674.6	36.24	602.0	675.1	744.8
mu.dmf[18]	702.4	38.72	624.6	702.7	777.0
mu.dmf[19]	730.1	41.45	646.8	730.6	809.7
mu.dmf[20]	702.4	38.72	624.6	702.7	777.0
mu.dmf[21]	702.4	38.72	624.6	702.7	777.0
pred.ind.1.7	257.3	145.8	-33.89	255.3	546.8
pred.mean.1.7	258.5	50.37	158.7	258.5	356.8
residual[1]	32.96	57.16	-78.05	32.97	146.5
residual[2]	237.1	82.93	75.25	237.1	398.8
residual[3]	21.22	53.72	-83.55	21.39	127.9
residual[4]	-139.2	35.98	-209.0	-139.5	-67.76

residual[5]	-116.2	35.98	-186.0	-116.5	-44.76
residual[6]	-94.22	35.98	-164.0	-94.46	-22.76
residual[7]	-46.48	38.42	-121.0	-46.57	29.72
residual[8]	-137.4	30.81	-197.8	-137.4	-75.79
residual[9]	-151.7	30.09	-209.8	-151.4	-90.83
residual[10]	-147.4	30.94	-206.9	-147.4	-85.25
residual[11]	-63.13	32.29	-124.9	-63.27	1.432
residual[12]	5.126	34.08	-60.23	4.903	73.22
residual[13]	-57.09	41.45	-136.7	-57.55	26.23
residual[14]	28.39	36.24	-41.83	27.93	101.0
residual[15]	3.647	38.72	-71.02	3.295	81.4
residual[16]	-8.092	41.45	-87.68	-8.55	75.23
residual[17]	58.39	36.24	-11.83	57.93	131.0
residual[18]	69.65	38.72	-5.02	69.3	147.4
residual[19]	79.91	41.45	0.324	79.45	163.2
residual[20]	120.6	38.72	45.98	120.3	198.4
residual[21]	324.6	38.72	250.0	324.3	402.4
sigma	135.0	21.83	98.52	132.5	183.2

A quadratic model may fit better. Here is a program for this more complex model, which also illustrates Bayesian regression for more than one variable.

```
model {
 for (i in 1:21)                  # loop over cities
    {
     mu.dmf[i] <- alpha + beta1*fl[i] + beta2*fl[i]*fl[i]
                                  # regression equation
     dmf[i] ~ dnorm(mu.dmf[i],tau)
                                  # distribution individual values
    }
    alpha ~ dnorm(0.0, 0.000001)  # prior for intercept
    beta1  ~ dnorm(0.0, 0.000001) # prior for slope for fl
    beta2  ~ dnorm(0.0, 0.000001) # prior for slope for fl squared
    sigma ~ dunif(0,200)          # prior for residual SD
   tau     <- 1/(sigma*sigma)     # precision required by WinBUGS
 for (i in 1:21)
   {                              # calculate residuals
    residual[i] <- dmf[i] - mu.dmf[i]
   }
```

```
pred.mean.1.7 <- alpha + beta1*1.7 + beta2*1.7*1.7
                                    # mean prediction for fl=1.7
pred.ind.1.7 ~ dnorm(pred.mean.1.7, tau)
                                    # individual pred for fl=1.7
}
```

Running this model, the results are:

node	mean	sd	2.5%	median	97.5%
alpha	808.6	34.31	739.8	808.4	875.5
beta1	-624.2	87.07	-792.6	-624.9	-448.1
beta2	161.5	38.16	85.78	162.1	236.3
pred.ind.1.7	214.7	105.0	3.166	213.7	424.9
pred.mean.1.7	214.2	37.34	140.5	213.8	287.7
sigma	94.4	17.54	67.35	91.79	135.8

Comparing Bayesian to Frequentist Regression

- If no prior information is used (i.e., if we use a Bayesian approach with "noninformative" or "flat" or "diffuse"or "reference" priors), then the inferences from Bayesian and frequentist are numerically similar. For example, 95% confidence intervals will be very similar to 95% credible intervals. We have illustrated this in the above examples.
- However, the *interpretations* of these intervals are completely different: Bayesian intervals are directly interpreted as the probability the parameter is in the credible interval, given the data and any prior information. Frequentist confidence intervals cannot be interpreted this way, one can only say that if the confidence interval procedure were to be used repeatedly, then 95% of all intervals will contain the true value.
- If there is prior information, then only the Bayesian approach can formally include this information in the model.
- Both Bayesian and frequentist approaches can be extended to much more complex models. However Bayesian methods tend to be more flexible, in general.

- Using WinBUGS, it is trivially easy to obtain inferences for any function of any estimated parameter. For example, having estimated, say, β_1 and β_2, it is trivial to obtain the posterior distribution for, say, $\beta_1 \times \sqrt{\beta_2}$.
- As we will soon see, Bayesian methods for model selection tend to work better than standard frequentist methods (although there is no single perfect method for model selection developed so far).

Bayesian Inference for Logistic Regression Parameters

Bayesian inference for logistic analyses follows the usual pattern for all Bayesian analyses, likelihood, prior, then multiple to derive the posterior density, from which all inferences follow.

For logistic regression, the 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, ..., 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 , 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.