Class 3: An introduction to Bayesian Statistics

Andrew Parnell
andrew.parnell@ucd.ie

Learning outcomes

- Understand the terms prior, likelihood and posterior
- Know what a posterior probability distribution is, and why we take samples from it
- Know how to formulate a linear regression model in a Bayesian format
- Be able to suggest appropriate prior distributions for different situations

Who was Bayes?

An essay towards solving a problem on the doctrine of chances (1763)

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

What is Bayesian statistics?

- Bayesian statistics is based on an interpretation of Bayes' theorem
- All quantities are divided up into *data* (i.e. things which have been observed) and *parameters* (i.e. things which haven't been observed)
- We use Bayes' interpretation of the theorem to get the *posterior probability distribution*, the probability of the unobserved given the observed
- Used now in almost all areas of statistical application (finance, medicine, environmetrics, gambling, etc, etc)

Why Bayes?

The Bayesian approach has numerous advantages:

- It's easier to build complex models and to analyse the parameters you want directly
- We automatically obtain the best parameter estimates and their uncertainty from the posterior samples
- It allows us to get away from (terrible) null hypothesis testing and *p*-values

Bayes theorem in english

Bayes' theorem can be written in words as:

posterior is proportional to likelihood times prior

... or ...

$$\text{posterior} \propto \text{likelihood} \times \text{prior}$$

Each of the three terms *posterior*, *likelihood*, and *prior* are *probability distributions* (pdfs).

In a Bayesian model, every item of interest is either data (which we will write as x) or parameters (which we will write as θ). Often the parameters are divided up into those of interest, and other *nuisance parameters*

Bayes theorem in maths

Bayes' equation is usually written mathematically as:

$$p(\theta|x) \propto p(x|\theta) \times p(\theta)$$

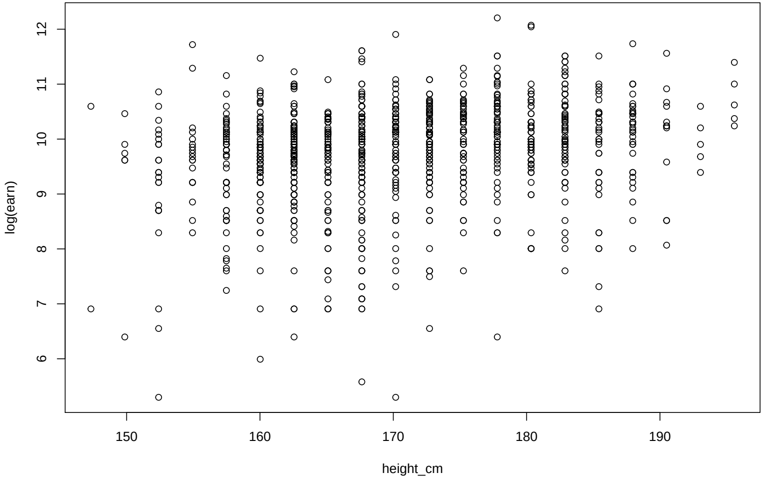
or, more fully:

$$p(\theta|x) = \frac{p(x|\theta) \times p(\theta)}{p(x)}$$

- The *posterior* is the probability of the parameters given the data
- The *likelihood* is the probability of observing the data given the parameters (unknowns)
- The *prior* represents external knowledge about the parameters

A very simple linear regression example

Suppose you had some data that looked like this:



What you are used to doing

```r
model = lm(log(earn) ~ height_cm, data = dat)
summary(model)
```

```
##
## Call:
## lm(formula = log(earn) ~ height_cm, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.4351 -0.3705  0.1615  0.5761  2.3302
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.895057   0.489107  12.053  < 2e-16 ***
## height_cm   0.022555   0.002866   7.869 8.84e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9066 on 1057 degrees of freedom
## Multiple R-squared:  0.05533,    Adjusted R-squared:  0.05444
## F-statistic: 61.91 on 1 and 1057 DF,  p-value: 8.836e-15
```

What you will now get instead

```r
print(stan_run)
```

```
## Inference for Stan model: a3e6f2fe6b7e8944db52faab015c0bfe.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                mean se_mean   sd    2.5%     25%     50%     75%   97.5%
## intercept      5.90    0.01 0.48    4.99    5.59    5.88    6.21    6.86
## slope          0.02    0.00 0.00    0.02    0.02    0.02    0.02    0.03
## residual_sd    0.91    0.00 0.02    0.87    0.89    0.91    0.92    0.95
## lp__        -426.20    0.03 1.21 -429.23 -426.79 -425.86 -425.30 -424.82
##             n_eff Rhat
## intercept    1319    1
## slope        1327    1
## residual_sd  1538    1
## lp__         1297    1
##
## Samples were drawn using NUTS(diag_e) at Sat Jan 27 22:39:21 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Using prior information

- The Bayesian model in the previous slide divided up everything into *parameters* (the intercept, slope and residual standard deviation), and data (the height and log(earnings) values)
- The software in the background created a posterior probability distribution of the parameters given the data
- The model I fitted used no *prior information*. However, if we had done a previous experiment that suggested the intercept should be around 9 with standard deviation 0.5 we can put this in the model

A model with prior information

```r
print(stan_run_2)
```

```
## Inference for Stan model: 187c0783dc10ea59139b887fc0a3bf53.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                 mean se_mean   sd    2.5%     25%     50%     75%   97.5%
## intercept       7.41    0.01 0.35    6.74    7.18    7.41    7.65    8.09
## slope           0.01    0.00 0.00    0.01    0.01    0.01    0.02    0.02
## residual_sd     0.91    0.00 0.02    0.87    0.90    0.91    0.92    0.95
## lp__         -436.02    0.04 1.22 -439.30 -436.58 -435.69 -435.13 -434.65
##             n_eff Rhat
## intercept    1494    1
## slope        1502    1
## residual_sd  1477    1
## lp__         1004    1
##
## Samples were drawn using NUTS(diag_e) at Sat Jan 27 22:39:52 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

An early example of a Bayesian model

- To create the Bayesian version of this model I used the following Stan code:

```
stan_code = '
data {
  int N;
  vector[N] x;
  vector[N] y;
}
parameters {
  real intercept;
  real slope;
  real<lower=0> residual_sd;
}
model {
  // Likelihood
  y ~ normal(intercept + slope * x, residual_sd);
  // Priors
  intercept ~ normal(0, 100);
  slope ~ normal(0, 100);
  residual_sd ~ uniform(0, 100);
}
'
```

Understanding the different parts of a Bayesian model

- The likelihood is the probability of observing the data given the parameters. It represents the *data generating process*
- The prior is the probability distribution of the parameters independent from the current data you have been generated. It often requires care (and philosophy) to choose. More on this later
- The posterior is the probability distribution of the parameters given the data. It is always the target of our inference
- The two software packages we will explore (Stan and JAGS), create the posterior distribution for us

Lots of probability distributions

Almost always, the likelihood and prior can be picked from the standard probability distributions:

Distribution	Range of parameter	Useful for
Normal, $N(\mu, \sigma^2)$	$(-\infty, \infty)$	A good default choice
Uniform, $U(a, b)$	(a, b)	Vague priors when we only know the range of the parameter
Binomial, $Bin(k, \theta)$	$[0, k]$	Count or binary data restricted to have an upper value
Poisson, $Po(\lambda)$	$[0, \infty)$	Count data with no upper limit
Gamma, $Ga(\alpha, \beta)$	$(0, \infty)$	Continuous data with a lower bound of zero
Multivariate Normal, $MVN(\mu, \Sigma)$	$(-\infty, \infty)$	Multivariate unbounded data with correlation between parameters/observations

The more probability distributions you know the better you will be at Bayes!

An example: linear regression

- Let's suppose we decide that a straight line will be a good fit for our earnings data
- A straight line has two parameters, the *slope* and the *intercept*
- For a full linear regression model, we also have a parameter for the residual standard deviation
- If we knew these three values we can simulate some data:

```
slope = 6
intercept = 0.03
res_sd = 1
log_earn_sim = rnorm(n = nrow(dat),
                     mean = intercept +
                       slope * dat$height_cm,
                     sd =  res_sd)
```

Example continued

- If the data we simulated looked exactly like the real data, we might believe that these are plausible values of the parameters
- We could do this lots of times with slightly different `slope`, `intercept` and `res_sd` values, and collect all the parameter values that are ‘close’ to the true data values
- This would provide us with a list of parameter values that approximately *match* the data
- To decide how close the match should be we use the normal distribution to give each set of parameter values a *score*. This score is the likelihood

Simulating from the prior and the likelihood

- But which values should we choose to simulate our parameters from?
- We could use vague priors

```r
slope = rnorm(1, mean = 0, sd = 100)
intercept = rnorm(1, mean = 0, sd = 100)
res_sd = runif(1, min = 0, max = 100)
log_earn_sim = rnorm(n = nrow(dat),
                     mean = intercept +
                       slope * dat$height_cm,
                     sd =  res_sd)
```

... but this would lead to crazily variable simulated data and we would have a very inefficient algorithm

- The computer software we will use avoids this random simulation and instead focusses on the values of the parameters which score highly on the likelihood and the prior

Practical differences between frequentist statistics and Bayes

- In frequentist statistics you tend to get a single best estimate of a parameter and a standard error, often assumed normally distributed, and a p-value
- In Bayesian statistics you get a large set of samples of the parameter values which match the data best. You get to choose what you do with these
- In frequentist statistics if the p-value is less than 0.05 you win. If not you cry and try a different model
- In Bayesian statistics you try to quantify the size of an effect from the posterior distribution, or find a particular posterior probability, e.g. P(slope $>$ 0 given the data).

More generally choosing a likelihood and a prior

- The key to choosing a likelihood is to pick a probability distribution which matches your data, e.g. if it's a continuous measurement and is unbounded then a normal distribution. If it's count data bounded above and below then a Binomial might be appropriate
- The key to choosing a prior distribution is to choose values which you believe represent the reasonable range that the parameter can take, or come from a related study in the literature
- Again, use an *informative prior* if possible

Note: the shape of the distribution of the response variable is usually completely unimportant when choosing the likelihood!

Stan vs JAGS

We will be using two different software tools to calculate posterior distributions. These represent the state of the art for user-friendly, research quality Bayesian statistics.

- Stan positives: very flexible, uses sensible distribution names, everything is declared, lots of documentation support, written by people at the top of the field
- Stan negatives: cannot have discrete parameters, some odd declaration choices, slower to run code, code tends to be longer
- JAGS positives: very quick for simple models, no declarations required, a bit older than Stan so more queries answered online
- JAGS negatives: harder to get complex models running, not as fancy an algorithm as Stan, crazy way of specifying normal distributions

Posterior computation in Stan

A Stan model looks like this:

```
stan_code = '
data {
  int N;
  vector[N] x;
  vector[N] y;
}
parameters {
  real intercept;
  real slope;
  real<lower=0> residual_sd;
}
model {
  // Likelihood
  y ~ normal(intercept + slope * x, residual_sd);
  // Priors
  intercept ~ normal(0, 100);
  slope ~ normal(0, 100);
  residual_sd ~ uniform(0, 100);
}
'
```

Running a model in Stan

To run the Stan model:

```
stan_run = stan(data = list(N = nrow(dat),
                            y = log(dat$earn),
                            x = dat$height_cm),
                model_code = stan_code)
print(stan_run)
plot(stan_run)
```

Posterior computation in JAGS

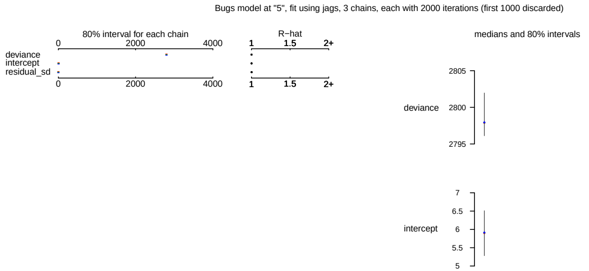
The same model in JAGS:

```
jags_code = '
model{
  # Likelihood
  for(i in 1:N) {
    y[i] ~ dnorm(intercept + slope * x[i],
                   residual_sd^-2)
  }
  # Priors
  intercept ~ dnorm(0, 100^-2)
  slope ~ dnorm(0, 100^-2)
  residual_sd ~ dunif(0, 100)
}
'
```

Note the ^−2 everywhere - JAGS uses precision rather than standard deviation in the normal distribution

Running the JAGS mdoel

```r
jags_run = jags(data = list(N = nrow(dat),
                            y = log(dat$earn),
                            x = dat$height_cm),
                parameters.to.save = c('intercept',
                                       'slope',
                                       'residual_sd'),
                model.file = textConnection(jags_code))
print(jags_run)
plot(jags_run)
```



Calculating the posterior vs sampling from it

- There are two ways to get at a posterior:
    1. Calculate it directly using hard maths
    2. Use a simulation method
- Number 1 is impractical once you move beyond a few parameters, so number 2 is used by almost everybody
- This means that we create *samples* from the posterior distribution. Here are three samples from the earnings example:

```
##   intercept      slope residual_sd      lp__
## 1  5.618430 0.02431245   0.8994509 -425.2909
## 2  6.637314 0.01840306   0.8982722 -426.6541
## 3  5.712600 0.02349494   0.8945060 -425.2656
```

- We often create thousands of posterior samples to represent the posterior distribution

Things you can do with posterior samples

- Create histograms or density plots:
- Individual summaries such as means, standard deviations, and quantiles (e.g. 95% confidence intervals)
- Joint summaries such as scatter plots or correlations
- Transformations such as logs/exponents, squares/square roots, etc

The posterior distribution will usually be stored in a matrix where each row is a sample, and each column is a different parameter. Having the posterior distribution enables you to get at exactly the quantities you are interested in

Summary so far: for and against Bayes

For:

- A Bayesian model can be simply displayed as a likelihood and a prior. Everything is explicit
- JAGS/Stan finds the posterior distribution for us so we don't need to do any maths
- We can get exactly the quantity we are interested in, the probability distribution of our unknowns given our knowns

Against:

- It can be hard to create a prior distribution (and a likelihood)
- Not having p-values can make papers harder to publish (but this is changing)

Checking assumptions (e.g. residuals)

- Sometimes people, because Bayesian modelling seems much richer and incorporates more information, think that their model is perfect
- In reality we need to check our assumptions. This may include:

1. Checking residuals in a linear regression model
2. Checking whether the parameter values actually match the data
3. Seeing whether a simpler or richer model might fit the data better

Some of this we will cover in later classes, but e.g. traditional residual analysis for linear regression is still important here

The danger of vague priors

- Suppose you use the prior distribution `intercept ~ normal(0, 100)` in Stan because you had very little information about the intercept. Is this a reasonable thing to do? Do you honestly believe that the intercept might be as small as -200 or as big as 200?
- If you fit a model and the parameters do not match your views about the data, there must be some information you have not encoded in the prior, go back and change it!
- In more complex models, you need the prior to constrain some of the parameters so that the model will fit. These are known as *regularisation* priors
- Use an *informative prior* when you can

Replication in Science and the horror of p-values

- We want our findings to be reproducible. We don't want to publish something which is only true for our own data
- Unfortunately p-values do not do this because:

1. A p-value is essentially a function of the sample size. If you want a smaller p-value simply collect more data
2. A p-value tells you nothing about the null hypothesis. It's a probability on the data assuming the null hypothesis is true!
3. The null hypothesis is never true. It's a straw man that is waiting to be knocked down by your data

Bayesian hierarchical modelling is a great way to stop using p-values and start doing reproducible science

General tips

- If you have lots of disparate data, try to build one model for all it. You'll be able to *borrow strength* from the data (e.g. in a hierarchical model) and reduce the uncertainty in your parameter estimates
- Try your hardest to use informative priors, and always justify the values you use (especially when trying to publish). In this course we're presenting generic examples so have almost always used vague priors
- Check your model. Many of the usual requirements from traditional statistics (e.g. residual checks) are still relevant in the Bayesian world. There are some extra Bayesian tricks we can also do; discussed in later classes

Summary

- Bayesian statistical models involve a *likelihood* and a *prior*. These both need to be carefully chosen. From these we create a posterior distribution
- The likelihood represents the information about the data generating process; the prior information about the unknown parameters
- We usually create and analyse samples from the posterior probability distribution of the unknowns (the parameters) given the knowns (the data)
- From the posterior distribution we can create means, medians, standard deviations, credible intervals, etc, from samples we take using e.g. JAGS or Stan