Lecture 5: Binomial models and prior strength

Zachary Marion

2/7/2018

We will need to load some packages and set some options prior to running any models:

```
library(rstan)
library(shinystan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
```

Recall from the past few lectures the motivating question has been to estimate the probability $P(p|D)$ of encountering water relative to land on a small model of the Earth (i.e., an inflatable globe).

- We can also interpret p as the relative proportion of water covering the Earth (or blue covering our model)

Last time we had gotten to the point of creating a Stan model:

```
data {
  int<lower=0> nObs;                 // Total number of observations
  int<lower=0, upper=1> obs[nObs];   // 1D array of observations
  real<lower=0> alpha;               // beta prior
  real<lower=0> beta;                // beta prior
}

parameters {
  real<lower=0, upper=1> p;     // prob. of water
}

model {
  p ~ beta(alpha, beta);                //prior on p
  for(n in 1:nObs) {
    obs[n] ~ bernoulli(p);      // bernoulli likelihood function
  }
}
```

Setting up the data as a list:

```
obs <- rep(c(1, 0), times = c(7, 3))  # our Bernoulli observations
nObs <- length(obs)  # number of observations
alpha <- 1  # Prior for alpha
beta <- 1  # Prior for beta
dat <- list(obs = obs, nObs = nObs, alpha = alpha, beta = beta)
```

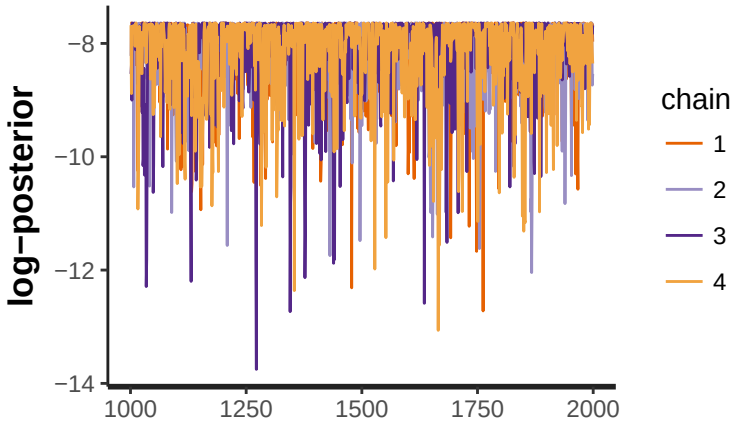
And then running the model using the `stan()` function:

```
mod1 <- stan(file="04.ex1Bernoulli.stan", #path to .stan file
             data=dat,
             iter=2000, # number of MCMC iterations
             chains=4,  # number of independent MCMC chains
             seed=3)    # set the seed so run is repeatable
```

```
recompiling to avoid crashing R session
```

We inspected chain mixing using the `traceplot` function of the `log-posterior`:

```
traceplot(mod1, par="lp__")
```



and looked at a summary of our results using the `print` function.

```
print(mod1)
```

```
Inference for Stan model: 04.
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% n_eff Rhat
p      0.66    0.00 0.13   0.39  0.58  0.67  0.76  0.89  1517    1
lp__ -8.15    0.02 0.72 -10.25 -8.33 -7.87 -7.69 -7.64  1541    1

Samples were drawn using NUTS(diag_e) at Mon Feb  5 13:40:32 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).
```

As with the `traceplot` output, a good overall diagnostic of model adequacy is to look at the log-posterior (`lp__`) results. For both the `lp__` and `p`, the ESS and $\hat{R}$ look good.

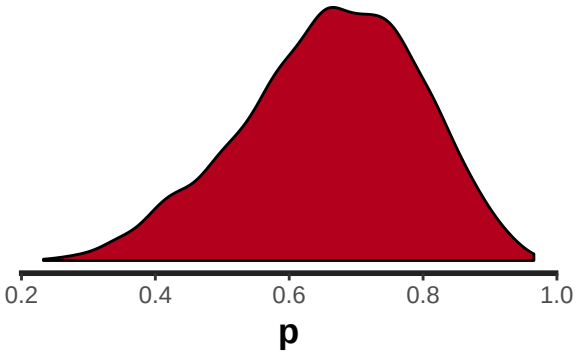
Accessing results

As with the diagnostics, we can get a numerical summary by using the `print` command. We will use the `par` argument to filter the results to return just the `p` parameter (useful when you have estimated hundreds or thousands of parameters).

```
print(mod1, par="p")
```

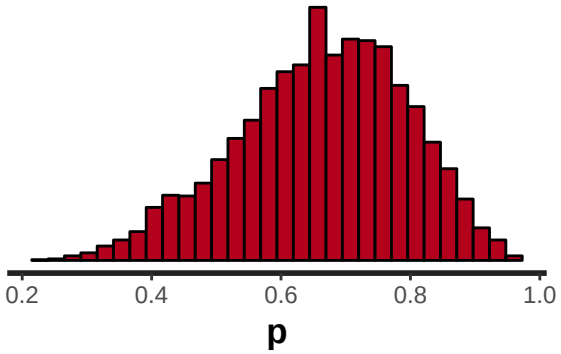
We can also plot the posterior as a density:

```
stan_dens(mod1, par="p")
```



or a histogram:

```
stan_hist(mod1, par="p")
```



From the plots we can see that the posterior is skewed. Thus one should be careful about using the mean as a point estimate. The median is probably more appropriate.

When reporting uncertainty intervals, most people use the 95% Highest Density Interval. However, Gelman recommends using the 50% interval for the following reasons:

1. Computational stability (it takes a lot fewer iterations to get good coverage around the 50% interval)
2. More intuitive evaluation (half the 50% intervals should contain the "true" value)
3. In applications, it is best to get a sense of where the parameter and predicted values will be, not to attempt unrealistic near-certainty.

shinystan

Another option for assessing the performance of models is to use the `shinystan` library. We can call the model using

```
launch_shinystan(mod1)
```

Shiny Stan basically gives us all of this output as an interactive GUI interface, which is convienent (at least until models get big!).

Bernoulli model as a binomial

The Bernoulli is a special case of the binomial distribution, the likelihood function of which—for the pedantic—is:

$$p(y|N,p) = \frac{N!}{y!(N-y)!}p^y(1-p)^{N-y} \tag{1}$$

where $y = \sum(W)$.

We can easily streamline and recode the model using a *Binomial* likelihood for more flexibility later:

```
data {
  int<lower=0> nObs;                // Total number of observations
  int<lower=0> obs;                 // scalar count of Ws
  real<lower=0> alpha;              // alpha and beta input as data
  real<lower=0> beta;               //  rather than hard-coded
}

parameters {
  real<lower=0, upper=1> p; // prob. of water
}

model {
  p ~ beta(alpha, beta);            // prior on theta
  obs ~ binomial(nObs, p);  // binomial likelihood function
}
```

We can recode the data and run the new model:

```
nObs <- length(obs)
obs <- sum(obs)
alpha <- 1
beta <- 1
dat2 <-  list(nObs=nObs, obs = obs, alpha=alpha, beta=beta)
```

```
modBin1 <- stan(file="05.ex1Binomial.stan", #path to .stan file
             data=dat2,
             iter=2000, # number of MCMC iterations
             chains=4,  # number of independent MCMC chains
             seed=3)    # set the seed so run is repeatable
```

```
recompiling to avoid crashing R session
```

The estimates from both the Bernoulli and Binomial models are identical (minus MCMC variation):

```
print(mod1)
```

```
      mean se_mean   sd   2.5%   25%   50%   75% 97.5%   n_eff Rhat
p     0.66    0.00 0.13   0.39  0.58  0.67  0.76  0.89 1516.76    1
lp__ -8.15    0.02 0.72 -10.25 -8.33 -7.87 -7.69 -7.64 1540.98    1
```

```
print(modBin1)
```

```
      mean se_mean   sd   2.5%   25%   50%   75% 97.5%   n_eff Rhat
p     0.66    0.00 0.13   0.39  0.57  0.67  0.76  0.89 1306.98 1.00
lp__ -8.16    0.02 0.75 -10.33 -8.32 -7.87 -7.69 -7.64 1333.15 1.01
```

Alternative parameterizations and prior strength

As mentioned two lectures ago, it is often convienient to specify *alpha* and *beta* of a beta prior in terms of the concentration (κ) and mean (μ):

$$a = \mu\kappa \text{ and } b = (1-\mu)\kappa. \tag{2}$$

or—for $\kappa > 2$—mode (ω):

$$a = \omega(\kappa-2)+1 \text{ and } b = (1-\omega)(\kappa-2)+1. \tag{3}$$

We can easily do this by using the `transformed parameters` block of Stan. I will use the mode but the strategy for using the mean is the same.

We will read the mode and concentration in as `data`. The `parameters` and `model` blocks doesn't change. After specifying the transformed parameters (`alpha` & `beta`), we write the equations as noted above.

```
data {
  int<lower=0> nObs;               // Total number of observations
  int<lower=0> obs;                // obs as scalar
  real<lower=0, upper=1> omega;  // mode as input data
  real<lower=2> kappa;           // concentration as input data
}

parameters {
  real<lower=0, upper=1> p;     // prob. of water
}

transformed parameters {
  real<lower=0> alpha;
  real<lower=0> beta;

  alpha = omega * (kappa - 2) + 1;
  beta = (1 - omega) * (kappa - 2) + 1;
```

```
}

model {
  p ~ beta(alpha, beta);                  // prior on p
  obs ~ binomial(nObs, p);       // likelihood
}
```

Suppose we think there is a 50/50 probability of water. We can see the effect of varying κ, our confidence in this assumption by running two models:

- Completely uninformed: $\omega = 0.5$, $\kappa = 2$
- Low confidence: $\omega = 0.5$, $\kappa = 4$
- High confidence: $\omega = 0.5$, $\kappa = 20$

To determine the influence of the prior, One option (for simple models), to exclude the likelihood and look at the posterior distribution of our parameters. We can do this by commenting out the last part of our model, and comparing the parameter estimates with and without the "data."

```
...

model {
  p ~ beta(alpha, beta);              // prior on p
 // obs ~ binomial(nObs, p);  // Exclude likelihood
}
```

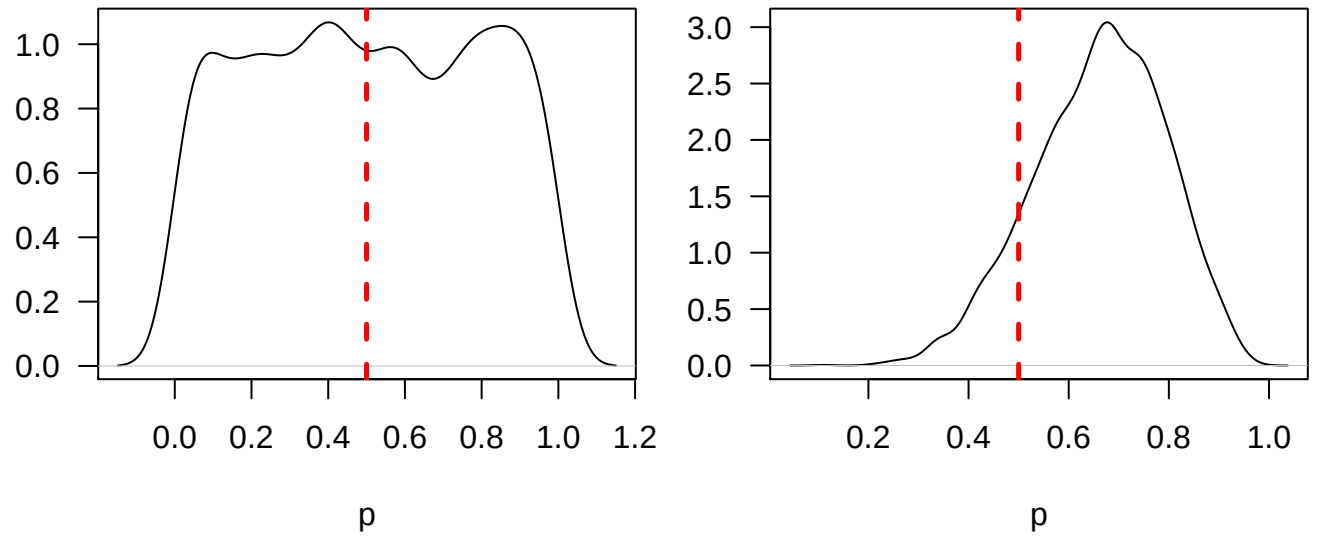
Completely uninformed: $\omega = 0.5$, $\kappa = 2$:

```
## Essentially Beta(1,1)
omega <- 0.5
kappa <- 2
dat11 <-  list(nObs=nObs, obs = obs, omega=omega, kappa=kappa)
```

```
## Beta(1,1) parameterized by mode
modBin11 <- stan(file="05.binomialMode.stan", #path to .stan file
             data=dat11,
             iter=2000, # number of MCMC iterations
             chains=4,  # number of independent MCMC chains
             seed=3)    # set the seed so run is repeatable
```

```
## No likelihood Beta(1,1)
modNoLik11 <- stan(file="05.noLikBinomial.stan", #path to .stan file
             data=dat11,
             iter=2000, # number of MCMC iterations
             chains=4,  # number of independent MCMC chains
             seed=3)    # set the seed so run is repeatable
```

```
p11 <- as.matrix(modBin11, par="p")
pNoLik11 <- as.matrix(modNoLik11, par="p")
par(mfrow=c(1,2))
par(mar=c(4,3,0.1,0.5))
plot(density(pNoLik11), xlab="p", las=1, main="", ylab="")
abline(v=omega, lty=2, lwd=2.5, col="red")
plot(density(p11), xlab="p", las=1, main="", ylab="Density")
abline(v=omega, lty=2, lwd=2.5, col="red")
```



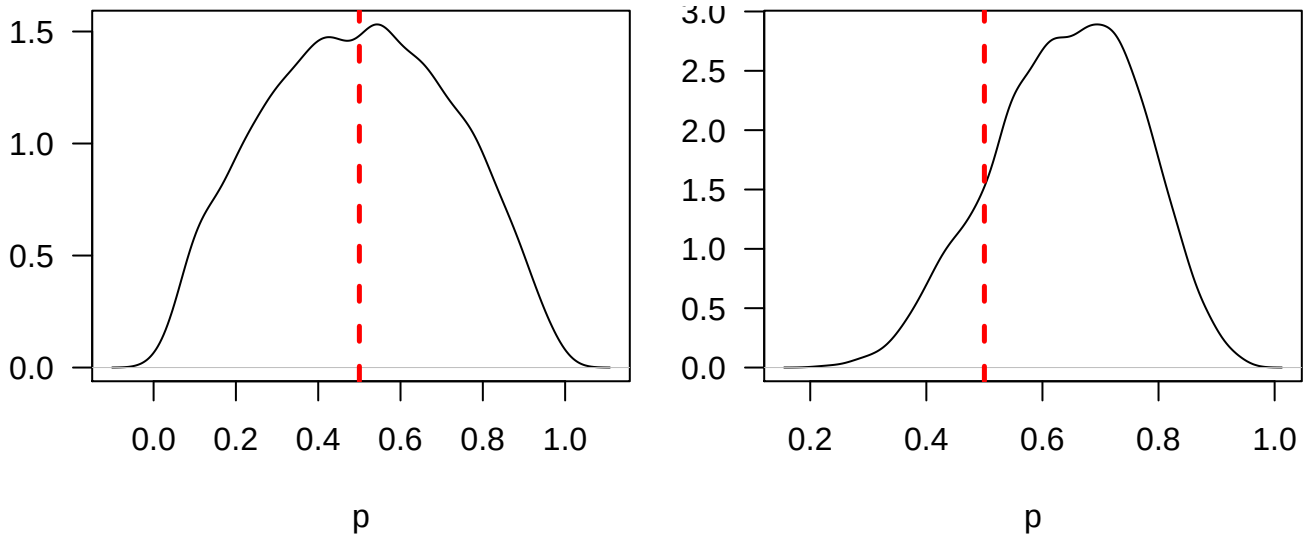

```
## Essentially Beta(2,2)
omega <- 0.5
kappa <- 4
dat22 <-  list(nObs=nObs, obs = obs, omega=omega, kappa=kappa)
```

```
## Beta(2,2) parameterized by mode
modBin22 <- stan(file="05.binomialMode.stan", #path to .stan file
             data=dat22,
             iter=2000, # number of MCMC iterations
             chains=4,  # number of independent MCMC chains
             seed=3)    # set the seed so run is repeatable
```

```
## No likelihood Beta(1,1)
modNoLik22 <- stan(file="05.noLikBinomial.stan", #path to .stan file
             data=dat22,
             iter=2000, # number of MCMC iterations
             chains=4,  # number of independent MCMC chains
             seed=3)    # set the seed so run is repeatable
```

```
p22 <- as.matrix(modBin22, par="p")
pNoLik22 <- as.matrix(modNoLik22, par="p")
par(mfrow=c(1,2))
par(mar=c(4,3,0.1,0.5))
plot(density(pNoLik22), xlab="p", las=1, main="", ylab="")
```

```
abline(v=omega, lty=2, lwd=2.5, col="red")
plot(density(p22), xlab="p", las=1, main="", ylab="Density")
abline(v=omega, lty=2, lwd=2.5, col="red")
```



Strongly informed: $\omega = 0.5$, $\kappa = 20$**:**

```
## Essentially Beta(10,10)
omega <- 0.5
kappa <- 20
dat1010 <-  list(nObs=nObs, obs = obs, omega=omega, kappa=kappa)
```

```
## Beta(2,2) parameterized by mode
modBin1010 <- stan(file="05.binomialMode.stan", #path to .stan file
             data=dat1010,
             iter=2000, # number of MCMC iterations
             chains=4,  # number of independent MCMC chains
             seed=3)    # set the seed so run is repeatable
```

```
## No likelihood Beta(1,1)
modNoLik1010 <- stan(file="05.noLikBinomial.stan", #path to .stan file
             data=dat1010,
             iter=2000, # number of MCMC iterations
             chains=4,  # number of independent MCMC chains
             seed=3)    # set the seed so run is repeatable
```

```
p1010 <- as.matrix(modBin1010, par="p")
pNoLik1010 <- as.matrix(modNoLik1010, par="p")
par(mfrow=c(1,2))
par(mar=c(4,3,0.1,0.5))
plot(density(pNoLik1010), xlab="p", las=1, main="", ylab="")
abline(v=omega, lty=2, lwd=2.5, col="red")
```

```r
plot(density(p1010), xlab="p", las=1, main="", ylab="Density")
abline(v=omega, lty=2, lwd=2.5, col="red")
```