Reproduce Biased coin factory example from DBDA using bridgesampling

Gertjan S Verhoeven (gertjan.verhoeven@gmail.com)

18 juli, 2018

Summary

To learn and gain confidence in the R package `bridgesampling`, we compare its estimated marginal likelihoods with an analytically tractable example from Kruschke's book "Doing Bayesian Data Analysis". In particular, we use the biased coin factory example from paragraph 10.2.

We have two competing models, that differ in their prior distribution for the probability of heads in a binomial likelihood. For the prior probability distribution, a beta distribution is used. One model ("the tail biased factory") generates coins that are distributed around mode ω 0.25, with a "concentration" of $\kappa = 12$. The other model ("the heads biased factory") generates coins that are distributed around a mode ω 0.75, als with a concentration of $\kappa = 12$. This translates to particular a/b parameters of the beta distribution.

We observe 6 heads after 9 flips. We are interested in the relative plausibility of which model has generated the data (which factory has produced the coin that generated the data).

Load packages

```
rm(list = ls())
library("bridgesampling")
```

```
## Warning: package 'bridgesampling' was built under R version 3.4.4
```

```
library("rstan")
```

```
## Warning: package 'rstan' was built under R version 3.4.3
```

```
## Loading required package: ggplot2
```

```
## Loading required package: StanHeaders
```

```
## Warning: package 'StanHeaders' was built under R version 3.4.3
```

```
## rstan (Version 2.17.3, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

Display beta-binomial model

```
data {
  int<lower = 1> T;
  int<lower = 0> y;
  real<lower = 0> beta_a;
  real<lower = 0> beta_b;
}
parameters {
  real<lower = 0, upper = 1> theta;
}
model {
  target += beta_lpdf(theta | beta_a, beta_b);  // prior
  target += binomial_lpmf(y | T, theta);  // likelihood
}
```

fit models

```
set.seed(1)
stanfit_H0 <- stan(file = "dbda_biased_coin_factory.stan",
                   data = list(y = 6,
                               T = 9, beta_a = 3.5, beta_b = 8.5),
                   iter = 15500, warmup = 500,
                   chains = 4, seed = 1)
```

```
##
## SAMPLING FOR MODEL 'dbda_biased_coin_factory' NOW (CHAIN 1).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:     1 / 15500 [  0%]  (Warmup)
## Iteration:   501 / 15500 [  3%]  (Sampling)
## Iteration:  2050 / 15500 [ 13%]  (Sampling)
## Iteration:  3600 / 15500 [ 23%]  (Sampling)
## Iteration:  5150 / 15500 [ 33%]  (Sampling)
## Iteration:  6700 / 15500 [ 43%]  (Sampling)
## Iteration:  8250 / 15500 [ 53%]  (Sampling)
## Iteration:  9800 / 15500 [ 63%]  (Sampling)
## Iteration: 11350 / 15500 [ 73%]  (Sampling)
## Iteration: 12900 / 15500 [ 83%]  (Sampling)
## Iteration: 14450 / 15500 [ 93%]  (Sampling)
## Iteration: 15500 / 15500 [100%]  (Sampling)
##
```

```
##  Elapsed Time: 0.016 seconds (Warm-up)
##                0.171 seconds (Sampling)
##                0.187 seconds (Total)
##
##
## SAMPLING FOR MODEL 'dbda_biased_coin_factory' NOW (CHAIN 2).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:     1 / 15500 [  0%]  (Warmup)
## Iteration:   501 / 15500 [  3%]  (Sampling)
## Iteration:  2050 / 15500 [ 13%]  (Sampling)
## Iteration:  3600 / 15500 [ 23%]  (Sampling)
## Iteration:  5150 / 15500 [ 33%]  (Sampling)
## Iteration:  6700 / 15500 [ 43%]  (Sampling)
## Iteration:  8250 / 15500 [ 53%]  (Sampling)
## Iteration:  9800 / 15500 [ 63%]  (Sampling)
## Iteration: 11350 / 15500 [ 73%]  (Sampling)
## Iteration: 12900 / 15500 [ 83%]  (Sampling)
## Iteration: 14450 / 15500 [ 93%]  (Sampling)
## Iteration: 15500 / 15500 [100%]  (Sampling)
##
##  Elapsed Time: 0.016 seconds (Warm-up)
##                0.203 seconds (Sampling)
##                0.219 seconds (Total)
##
##
## SAMPLING FOR MODEL 'dbda_biased_coin_factory' NOW (CHAIN 3).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:     1 / 15500 [  0%]  (Warmup)
## Iteration:   501 / 15500 [  3%]  (Sampling)
## Iteration:  2050 / 15500 [ 13%]  (Sampling)
## Iteration:  3600 / 15500 [ 23%]  (Sampling)
## Iteration:  5150 / 15500 [ 33%]  (Sampling)
## Iteration:  6700 / 15500 [ 43%]  (Sampling)
## Iteration:  8250 / 15500 [ 53%]  (Sampling)
## Iteration:  9800 / 15500 [ 63%]  (Sampling)
## Iteration: 11350 / 15500 [ 73%]  (Sampling)
## Iteration: 12900 / 15500 [ 83%]  (Sampling)
## Iteration: 14450 / 15500 [ 93%]  (Sampling)
```

```
## Iteration: 15500 / 15500 [100%]  (Sampling)
##
##  Elapsed Time: 0 seconds (Warm-up)
##                0.171 seconds (Sampling)
##                0.171 seconds (Total)
##
##
## SAMPLING FOR MODEL 'dbda_biased_coin_factory' NOW (CHAIN 4).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:     1 / 15500 [  0%]  (Warmup)
## Iteration:   501 / 15500 [  3%]  (Sampling)
## Iteration:  2050 / 15500 [ 13%]  (Sampling)
## Iteration:  3600 / 15500 [ 23%]  (Sampling)
## Iteration:  5150 / 15500 [ 33%]  (Sampling)
## Iteration:  6700 / 15500 [ 43%]  (Sampling)
## Iteration:  8250 / 15500 [ 53%]  (Sampling)
## Iteration:  9800 / 15500 [ 63%]  (Sampling)
## Iteration: 11350 / 15500 [ 73%]  (Sampling)
## Iteration: 12900 / 15500 [ 83%]  (Sampling)
## Iteration: 14450 / 15500 [ 93%]  (Sampling)
## Iteration: 15500 / 15500 [100%]  (Sampling)
##
##  Elapsed Time: 0.016 seconds (Warm-up)
##                0.173 seconds (Sampling)
##                0.189 seconds (Total)
```

```
stanfit_H1 <- stan(file = "dbda_biased_coin_factory.stan",
                   data = list(y = 6,
                               T = 9, beta_a = 8.5, beta_b = 3.5),
                   iter = 15500, warmup = 500,
                   chains = 4, seed = 1)
```

```
##
## SAMPLING FOR MODEL 'dbda_biased_coin_factory' NOW (CHAIN 1).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:     1 / 15500 [  0%]  (Warmup)
## Iteration:   501 / 15500 [  3%]  (Sampling)
## Iteration:  2050 / 15500 [ 13%]  (Sampling)
```

```
## Iteration:  3600 / 15500 [ 23%]  (Sampling)
## Iteration:  5150 / 15500 [ 33%]  (Sampling)
## Iteration:  6700 / 15500 [ 43%]  (Sampling)
## Iteration:  8250 / 15500 [ 53%]  (Sampling)
## Iteration:  9800 / 15500 [ 63%]  (Sampling)
## Iteration: 11350 / 15500 [ 73%]  (Sampling)
## Iteration: 12900 / 15500 [ 83%]  (Sampling)
## Iteration: 14450 / 15500 [ 93%]  (Sampling)
## Iteration: 15500 / 15500 [100%]  (Sampling)
##
##  Elapsed Time: 0.016 seconds (Warm-up)
##                0.171 seconds (Sampling)
##                0.187 seconds (Total)
##
##
## SAMPLING FOR MODEL 'dbda_biased_coin_factory' NOW (CHAIN 2).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:     1 / 15500 [  0%]  (Warmup)
## Iteration:   501 / 15500 [  3%]  (Sampling)
## Iteration:  2050 / 15500 [ 13%]  (Sampling)
## Iteration:  3600 / 15500 [ 23%]  (Sampling)
## Iteration:  5150 / 15500 [ 33%]  (Sampling)
## Iteration:  6700 / 15500 [ 43%]  (Sampling)
## Iteration:  8250 / 15500 [ 53%]  (Sampling)
## Iteration:  9800 / 15500 [ 63%]  (Sampling)
## Iteration: 11350 / 15500 [ 73%]  (Sampling)
## Iteration: 12900 / 15500 [ 83%]  (Sampling)
## Iteration: 14450 / 15500 [ 93%]  (Sampling)
## Iteration: 15500 / 15500 [100%]  (Sampling)
##
##  Elapsed Time: 0.016 seconds (Warm-up)
##                0.203 seconds (Sampling)
##                0.219 seconds (Total)
##
##
## SAMPLING FOR MODEL 'dbda_biased_coin_factory' NOW (CHAIN 3).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:     1 / 15500 [  0%]  (Warmup)
```

```
## Iteration:   501 / 15500 [  3%]  (Sampling)
## Iteration:  2050 / 15500 [ 13%]  (Sampling)
## Iteration:  3600 / 15500 [ 23%]  (Sampling)
## Iteration:  5150 / 15500 [ 33%]  (Sampling)
## Iteration:  6700 / 15500 [ 43%]  (Sampling)
## Iteration:  8250 / 15500 [ 53%]  (Sampling)
## Iteration:  9800 / 15500 [ 63%]  (Sampling)
## Iteration: 11350 / 15500 [ 73%]  (Sampling)
## Iteration: 12900 / 15500 [ 83%]  (Sampling)
## Iteration: 14450 / 15500 [ 93%]  (Sampling)
## Iteration: 15500 / 15500 [100%]  (Sampling)
##
##  Elapsed Time: 0.015 seconds (Warm-up)
##                0.203 seconds (Sampling)
##                0.218 seconds (Total)
##
##
## SAMPLING FOR MODEL 'dbda_biased_coin_factory' NOW (CHAIN 4).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:     1 / 15500 [  0%]  (Warmup)
## Iteration:   501 / 15500 [  3%]  (Sampling)
## Iteration:  2050 / 15500 [ 13%]  (Sampling)
## Iteration:  3600 / 15500 [ 23%]  (Sampling)
## Iteration:  5150 / 15500 [ 33%]  (Sampling)
## Iteration:  6700 / 15500 [ 43%]  (Sampling)
## Iteration:  8250 / 15500 [ 53%]  (Sampling)
## Iteration:  9800 / 15500 [ 63%]  (Sampling)
## Iteration: 11350 / 15500 [ 73%]  (Sampling)
## Iteration: 12900 / 15500 [ 83%]  (Sampling)
## Iteration: 14450 / 15500 [ 93%]  (Sampling)
## Iteration: 15500 / 15500 [100%]  (Sampling)
##
##  Elapsed Time: 0.016 seconds (Warm-up)
##                0.171 seconds (Sampling)
##                0.187 seconds (Total)
```

Check Stan fit output

```
print(stanfit_H0)
```

```
## Inference for Stan model: dbda_biased_coin_factory.
```

```
## 4 chains, each with iter=15500; warmup=500; thin=1;
## post-warmup draws per chain=15000, total post-warmup draws=60000.
##
##        mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## theta  0.45       0 0.11  0.25  0.38  0.45  0.53  0.66 23374    1
## lp__  -3.78       0 0.71 -5.82 -3.95 -3.51 -3.33 -3.28 23894    1
##
## Samples were drawn using NUTS(diag_e) at Wed Jul 18 12:04:41 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).
```

```
print(stanfit_H1)
```

```
## Inference for Stan model: dbda_biased_coin_factory.
## 4 chains, each with iter=15500; warmup=500; thin=1;
## post-warmup draws per chain=15000, total post-warmup draws=60000.
##
##        mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## theta  0.69       0 0.10  0.48  0.63  0.70  0.76  0.86 23065    1
## lp__  -2.32       0 0.72 -4.37 -2.49 -2.04 -1.86 -1.81 22329    1
##
## Samples were drawn using NUTS(diag_e) at Wed Jul 18 12:04:42 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).
```

compute (log) marginal likelihoods

```
set.seed(1)
bridge_H0 <- bridge_sampler(stanfit_H0)
```

```
## Warning: effective sample size cannot be calculated, has been replaced by
## number of samples.
```

```
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
```

```
bridge_H0
```

```
## Bridge sampling estimate of the log marginal likelihood: -3.17141
## Estimate obtained in 4 iteration(s) via method "normal".
```

```
bridge_H1 <- bridge_sampler(stanfit_H1)
```

```
## Warning: effective sample size cannot be calculated, has been replaced by
## number of samples.
```

```
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
```

```
bridge_H1
```

```
## Bridge sampling estimate of the log marginal likelihood: -1.62721
## Estimate obtained in 4 iteration(s) via method "normal".
```

Compute approximate percentage errors

```
error_measures(bridge_H0)$percentage
```

```
## [1] "0.00673%"
```

```
error_measures(bridge_H1)$percentage
```

```
## [1] "0.0201%"
```

The marginal likelihoods differ from Kruschke's probabilities. This is because he does not use the binomial distribution, but uses a bernoulli likelihood for the outcome of a set of coin flips.

```
exp(bridge_H0$logml)
```

```
## [1] 0.04194459
```

```
exp(bridge_H1$logml)
```

```
## [1] 0.1964778
```

```
exp(bridge_H0$logml)/exp(bridge_H1$logml)
```

```
## [1] 0.2134826
```

compute Bayes factor

```
bf(bridge_H0, bridge_H1)
```

```
## The estimated Bayes factor in favor of x1 over x2 is equal to: 0.21348
```

This compares nicely with the analytically calculated value of 0.213

Calculated posterior model probabilities given equal prior probability

```r
post1 <- post_prob(bridge_H0, bridge_H1)
print(post1)
```

```
## bridge_H0 bridge_H1
## 0.1759255 0.8240745
```