Marginal Likelihood Computation with the `margLikArrogance` package

By Benedict Escoto

Abstract

The purpose of the `margLikArrogance` package is to compute marginal likelihoods from the posterior parameter distributions of Bayesian models using "arrogance sampling". These marginal likelihoods can then be used to compare how strongly the evidence supports competing theories. This vignette treats a simple Bayesian model comparison problem in detail from start to finish and shows how to apply the `margLikArrogance` package.

1 Introduction

Model choice is theoretically simple in Bayesian statistics. Given two competing models or theories, T_1 and T_2, and a vector of observations $\boldsymbol{x}$, a Bayesian conditionalizes on $\boldsymbol{x}$ and finds via Bayes's Theorem that

$$\frac{p(T_1|\boldsymbol{x})}{p(T_2|\boldsymbol{x})} = \frac{p(\boldsymbol{x}|T_1)}{p(\boldsymbol{x}|T_2)}\frac{p(T_1)}{p(T_2)}. \tag{1}$$

The quantity $\frac{p(\boldsymbol{x}|T_1)}{p(\boldsymbol{x}|T_2)}$ is called a *Bayes factor* and the quantities $p(\boldsymbol{x}|T_1)$ and $p(\boldsymbol{x}|T_2)$ are called the theories' *marginal likelihoods*. Once we compute either of these quantities we know the strength of evidence $\boldsymbol{x}$ in support of theory T_1 vs T_2.

The problem is that the marginal likelihoods are hard to compute. They are called marginal likelihoods because typically Bayesian models have several parameters necessary to compute the models' likelihood on any evidence. If $\boldsymbol{\theta}_i$ are parameters for T_i, then

$$p(\boldsymbol{x}|T_i) = \int p(\boldsymbol{x}|\boldsymbol{\theta}_i, T_i)p(\boldsymbol{\theta}_i|T_i)\, d\boldsymbol{\theta}_i \tag{2}$$

and the marginal likelihood computation requires evaluating an integral. Typically this integral has no analytic solution and must be solved numerically. Even numerical approximation is often difficult because $p(\boldsymbol{x}|\boldsymbol{\theta}_i, T_i)$ is often very "spiky"—close to 0 except in a small region where it is very large.

1.1 MCMC and Arrogance Sampling

The purpose of the `margLikArrogance` package is to help compute integral (2). The first step is to sample from the posterior parameter distributions $\boldsymbol{\theta}_i|\boldsymbol{x}, T_i$. This can be done using a markov chain monte carlo (MCMC) technique such as Gibbs sampling. Sampling from the

posterior parameter distribution is a common move in Bayesian statistics, frequently done even if theory choice is not an issue.

Second, the likelihoods $p(x|\boldsymbol{\theta}_i, T_i)$ are computed for each $\boldsymbol{\theta}_i$ in the posterior parameter sample.

Third, the `margLikArrogance` package processes the posterior parameter samples and the likelihoods and returns $p(\boldsymbol{x}|\boldsymbol{\theta}_i, T_i)$. These can then be plugged into formulas like (1) to update probabilities. The two basic inputs required for the package to estimate the marginal likelihood of a theory T_i are:

1. Samples from the posterior distribution of parameters $\boldsymbol{\theta}_i|\boldsymbol{x}, T_i$, denoted as $\boldsymbol{\theta}_{j,i}$.
2. At each point $\boldsymbol{\theta}_{j,i}$, the log-likelihood of the prior probability times the likelihood of the evidence: $\log(p(\boldsymbol{x}|\boldsymbol{\theta}_{j,i}, T_i)p(\boldsymbol{\theta}_{j,i}|T_i)) = \log p(\boldsymbol{x} \wedge \boldsymbol{\theta}_{j,i}|T_i)$.

The `margLikArrogance` packge uses a monte carlo technique known as non-parametric importance sampling, or arrogance sampling. Basically a kind of histogram is built from the posterior parameter samples and used for importance sampling. The rest of this vignette considers three simple theories in detail and shows how to use the `margLikArrogance` package to decide between them in light of some data. Technical details can be found in XXXX.

That paper also explains the advantages and disadvantages of this technique versus other methods such as simple monte carlo integration, parametrized importance sampling, path integration, the standard harmonic mean estimator, etc. To summarize, you may find this package useful if

- you already have the posterior parameter samples $\boldsymbol{\theta}_{j,i}$ available, probably through an MCMC method;
- the quantity $p(\boldsymbol{x}|\boldsymbol{\theta}_{j,i}, T_i)p(\boldsymbol{\theta}_{j,i}|T_i)$ is easy to compute;
- $p(\boldsymbol{\theta}_{j,i}|\boldsymbol{x}, T_i) > 0$ everywhere, or at least near the samples $\boldsymbol{\theta}_{j,i}$;
- and the dimensionality of the space $\boldsymbol{\theta}_i$ is not too large, say around 10 or less (or perhaps more depending on the desired accuracy).

1.2 Compared to the Harmonic Mean Estimator

The harmonic mean estimator (HME) is similar to what the `margLikArrogance` package does: they both approximate a theory's marginal likelihood given samples from that theory's posterior parameter distribution. There are two main differences:

1. This package is supposed to actually give the correct answer.
2. The HME requires the likelihood $p(\boldsymbol{x}|\boldsymbol{\theta}_{j,i}, T_i)$ at each point, while this package requires the value $p(\boldsymbol{x} \wedge \boldsymbol{\theta}_{j,i}|T_i)$.

See `http://http://radfordneal.wordpress.com/2008/08/17/the-harmonic-mean-of-the-likelihood-worst-monte-carlo-method-ever` for a nice explanation of why HME doesn't work. Because of the similarity in requirements to the HME, in many cases the arrogance sampling described here may be used as a convenient but superior replacement for the HME.

2 Example Theories and Evidence

Our simpleminded task is to measure people's heights and choose between three theories of how tall people are. Each theory is a Bayesian hierarchical model:

Theory 1: People's heights are normally distributed with mean μ and standard deviation 0.5.

- μ is normally distributed with mean 5 and standard deviation 0.1.

Theory 2: Heights are lognormally distributed with mean log μ and standard deviation log of 0.1.

- μ is normally distributed with mean 1.6 and standard deviation 0.02.

Theory 3: Heights are normally distributed with mean μ and precision τ (precision is the reciprocal of variance).

- μ is normally distributed with mean 5 and standard deviation 0.1.
- τ is gamma distributed with mean 4 and standard deviation 1.
- μ and τ are independent.

For each, the hyperparameters are shared by all people. For instance, in theory 1, if μ is 6.1, then the height distribution of everyone has a mean of 6.1.

```
> t1.mean.mean <- 5
> t1.mean.sd <- 0.1
> t1.sd <- 0.5
> t2.mu.mean <- 1.6
> t2.mu.sd <- 0.02
> t2.sigma <- 0.1
> t3.mean.mean <- 5
> t3.mean.sd <- 0.1
> t3.prec.rate <- 4/1^2
> t3.prec.shape <- 4 * t3.prec.rate
> set.seed(1)
> t1.prior.samples <- rnorm(1000, mean = rnorm(1000, mean = t1.mean.mean,
```

```
+     sd = t1.mean.sd), sd = t1.sd)
> t2.prior.samples <- exp(rnorm(1000, rnorm(1000, mean = t2.mu.mean,
+     sd = t2.mu.sd), t2.sigma))
> t3.prior.mu.samples <- rnorm(1000, mean = t3.mean.mean, sd = t3.mean.sd)
> t3.prior.prec.samples <- rgamma(1000, shape = t3.prec.shape,
+     rate = t3.prec.rate)
> t3.prior.samples <- rnorm(1000, mean = t3.prior.mu.samples, sd = 1/sqrt(t3.prior.pre
> p1.df <- data.frame(x = c(t1.prior.samples, t2.prior.samples,
+     t3.prior.samples), t = c(rep("Theory 1", 1000), rep("Theory 2",
+     1000), rep("Theory 3", 1000)))
> p1 <- (ggplot(data = p1.df) + geom_histogram(aes(x = x), binwidth = 0.2) +
+     facet_wrap(~t, ncol = 1) + labs(x = "Height", y = "Count"))
```

The R code above defined the initial parameters and samples 1000 heights from each prior marginal distribution. These samples are then used to plot a histogram shown in figure 1. As you can see, the marginal distributions look somewhat similar.

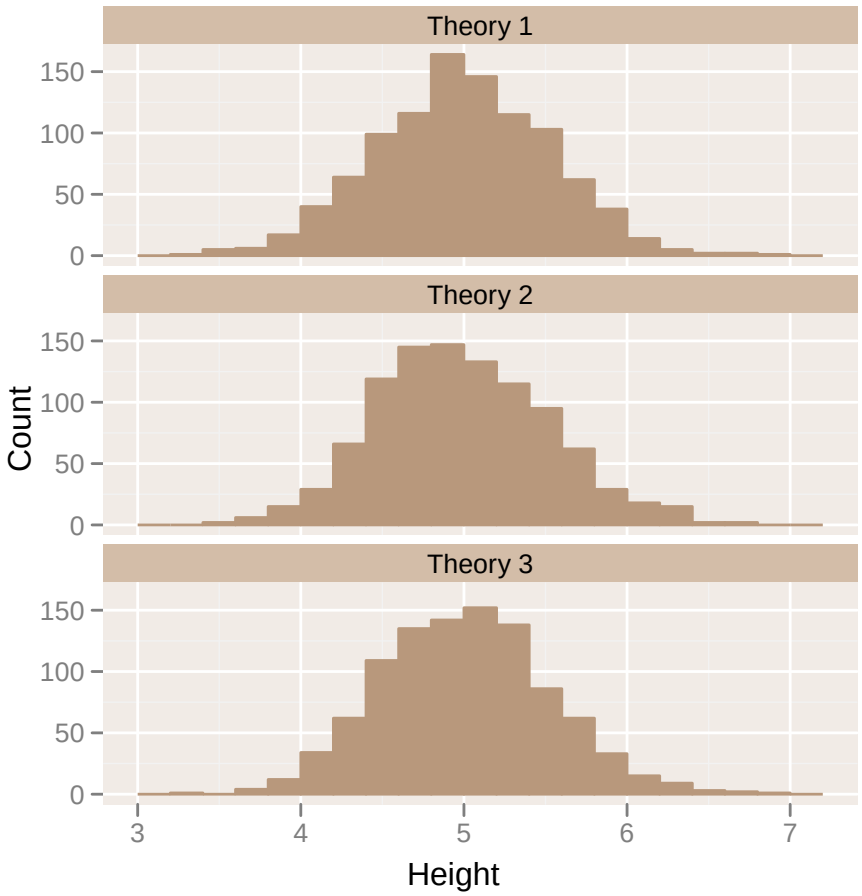

Figure 1: Prior Marginal Distributions

2.1 Example Data

To continue the sample, we now sample the heights of 100 people. The results are shown in figure 2. This is the evidence we will use to compare the three theories.

Person #	Height	Person #	Height	Person #	Height	Person #	Height
1	4.66	16	5.07	31	6.05	46	4.60
2	5.23	17	5.09	32	5.03	47	5.36
3	4.52	18	5.76	33	5.37	48	5.64
4	6.22	19	5.67	34	5.06	49	5.02
5	5.33	20	5.52	35	4.14	50	5.72
6	4.53	21	5.74	36	4.81	51	5.38
7	5.44	22	5.65	37	4.82	52	4.67
8	5.62	23	5.15	38	5.06	53	5.34
9	5.50	24	3.71	39	5.87	54	4.31
10	4.89	25	5.53	40	5.63	55	6.10
11	6.16	26	5.06	41	4.98	56	6.49
12	5.37	27	4.99	42	4.92	57	4.84
13	4.67	28	4.07	43	5.59	58	4.37
14	3.55	29	4.77	44	5.49	59	5.50
15	5.89	30	5.39	45	4.62	60	5.01

Figure 2: Sampled Heights

3 Sampling From the Posterior

The next step is to sample from the posterior of each distribution's parameters. 1000 samples will be taken from each posterior distribution.

This is somewhat trivial for theories 1 and 2 because they are Bayesian conjugates. In fact, the marginal likelihood is analytically soluable so there is no need to use the `margLikArrogance` package. They were chosen as an example so the output of the package can be compared to the exact answer. The code that computes them is shown below.

```
> UpdateMean <- function(mean.mean, mean.sd, sd, x) {
+     post.sd <- sqrt(1/(1/mean.sd^2 + length(x)/sd^2))
+     return(c(mean = (mean.mean/mean.sd^2 + sum(x)/sd^2) * post.sd^2,
+         sd = post.sd))
+ }
> set.seed(1)
```

```
> t1.post.param <- UpdateMean(t1.mean.mean, t1.mean.sd, t1.sd,
+     samples)
> t1.post.samples <- rnorm(1000, mean = t1.post.param["mean"],
+     sd = t1.post.param["sd"])
> t2.post.param <- UpdateMean(t2.mu.mean, t2.mu.sd, t2.sigma, log(samples))
> t2.post.samples <- rnorm(1000, mean = t2.post.param["mean"],
+     sd = t2.post.param["sd"])
```

For theory 3 we will sample from the posterior by taking advantage of the semi-conjugacy of our model and coding a Gibbs sampler. The code is shown below.

```
> UpdatePrecision <- function(prec.shape, prec.rate, mean, x) {
+     return(c(shape = prec.shape + length(x)/2, rate = prec.rate +
+         sum((x - mean)^2)/2))
+ }
> t3.mean.post.samples <- rep(NA, 1010)
> t3.sd.post.samples <- rep(NA, 1010)
> t3.mean.post.samples[1] <- 5
> t3.sd.post.samples[1] <- 0.5
> set.seed(1)
> for (i in 2:1010) {
+     mu.params <- UpdateMean(t3.mean.mean, t3.mean.sd, t3.sd.post.samples[i -
+         1], samples)
+     t3.mean.post.samples[i] <- rnorm(1, mu.params["mean"], mu.params["sd"])
+     prec.params <- UpdatePrecision(t3.prec.shape, t3.prec.rate,
+         t3.mean.post.samples[i - 1], samples)
+     t3.sd.post.samples[i] <- sqrt(1/rgamma(1, shape = prec.params["shape"],
+         rate = prec.params["rate"]))
+ }
> t3.post.samples <- cbind(t3.mean.post.samples, t3.sd.post.samples)[11:1010,
+     ]
> colnames(t3.post.samples) <- c("mean", "sd")
```

This chain should be checked for tuning, convergence, etc., but here we will just assume we have obtained the posterior samples we wanted. The prior and posterior parameter distributions are shown in figure 3.

3.1 Sampled Likelihoods

We have the sampled posterio parameter distributions for each theory; the next step is compute the value $\log p(\boldsymbol{x} \wedge \boldsymbol{\theta}_{j,i}|T_i) = \log(p(\boldsymbol{x}|\boldsymbol{\theta}_{j,i}, T_i)p(\boldsymbol{\theta}_{j,i}|T_i))$ for each point $\boldsymbol{\theta}_{j,i}$. For theory 1 we have:

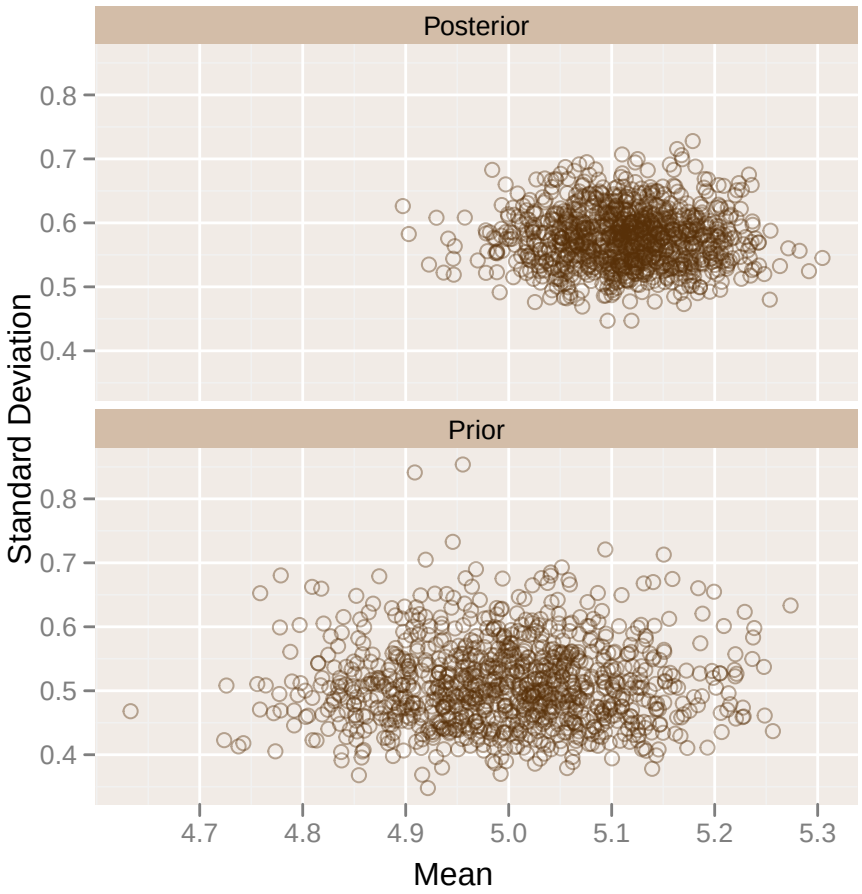

Figure 3: Theory 3 Prior vs Posterior Parameters

```
> T1OneLik <- function(theta) {
+     ll <- sum(dnorm(samples, mean = theta, sd = t1.sd, log = TRUE))
+     return(ll + dnorm(theta, mean = t1.mean.mean, sd = t1.mean.sd,
+         log = TRUE))
+ }
> t1.ll <- sapply(t1.post.samples, T1OneLik)
```

For theory 2:

```
> T2OneLik <- function(theta) {
+     ll <- sum(dlnorm(samples, meanlog = theta, sdlog = t2.sigma,
+         log = TRUE))
+     return(ll + dnorm(theta, mean = t2.mu.mean, sd = t2.mu.sd,
+         log = TRUE))
+ }
> t2.ll <- sapply(t2.post.samples, T2OneLik)
```

Finally, for theory 3 we have:

```
> T3OneLik <- function(theta) {
+     ll <- sum(dnorm(samples, mean = theta[1], sd = theta[2],
+         log = TRUE))
+     param.ll <- (dnorm(theta[1], mean = t3.mean.mean, sd = t3.mean.sd,
+         log = TRUE) + dgamma(1/theta[2]^2, shape = t3.prec.shape,
+         rate = t3.prec.rate, log = TRUE))
+     return(ll + param.ll)
+ }
> t3.ll <- apply(t3.post.samples, 1, T3OneLik)
```

4 Using the Package and Comparing the Theories

We can now apply the `margLikArrogance` package to compute each theory's marginal likelihood. The following are the only three lines in this example that actually use the package!

```
> library(margLikArrogance)
> t1.margll <- MarginalLikelihood(matrix(t1.post.samples, ncol = 1),
+     t1.ll)
> t2.margll <- MarginalLikelihood(matrix(t2.post.samples, ncol = 1),
+     t2.ll)
> t3.margll <- MarginalLikelihood(t3.post.samples, t3.ll, bounds = matrix(c(-Inf,
+     Inf, 0.1, Inf), nrow = 2))
```

The results are summarized in figure 4. The posterior probability shown assumes that each theory had probability one-third before the data was observed. Note that the `bounds` argument was used for theory 3, specifying that the standard deviation cannot be negative.

Theory	Estimated Marg LL	Confidence Interval	Actual MLL (Analytic)	Posterior Probability
1	-57.6	-57.8 to -57.5	-57.5	0.70
2	-59.9	-60.0 to -59.8	-59.9	0.08
3	-58.8	-58.9 to -58.8		0.22

Figure 4: Arrogance Sampling Results

4.1 Qualitative Comparisons

Instead of computing posterior probabilities directly, we can reference the table from Kass and Raftery (figure 5) when interpreting the results in figure 4. According to that, the sample data is positive evidence for theory 1 versus theory 3. The data does not help us much to decide between theories 1 and 3 or theories 2 and 3.

LL Difference	Strength of Evidence
0 to 1	Inconsequential
1 to 3	Positive
3 to 5	Strong
> 5	Very Strong

Figure 5: Kass and Raftery's Guidelines

5 A More Challenging Example

The `margLikArrogance` package requires samples $\boldsymbol{\theta}_{j,i}$ from the posterior parameter distribution $\boldsymbol{\theta}_i|\boldsymbol{x}, T_i$ and the value of $\log p(\boldsymbol{x} \wedge \boldsymbol{\theta}_{j,i}|T_i)$ at each point. Because

$$p(\boldsymbol{\theta}_i|\boldsymbol{x}, T_i) = \frac{p(\boldsymbol{x} \wedge \boldsymbol{\theta}_i|T_i)}{p(\boldsymbol{x}|T_i)},$$

computing the marginal likelihood $p(\boldsymbol{x}, T_i)$ is equivalent to computing the normalizing constant for an unknown density $p(\boldsymbol{\theta}_i|\boldsymbol{x}, T_i)$, given samples from $\boldsymbol{\theta}_i|\boldsymbol{x}, T_i$ and the unnormalized density $p(\boldsymbol{x} \wedge \boldsymbol{\theta}_i|T_i)$ at each point.

Thus we can test the algorithm by suppling samples from an arbitrary density along with densities multiplied by an arbitrary constant. In this section, we try the algorithm on a 50/50 mixture of two 15-dimensional normal distributions. The first has mean at

$$\mu_1 = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0).$$

The second has mean at $\mu_2 = 10^6\mu_1$. Both will have the identity covariance matrix. The code below generates $n = 100000$ samples from this distribution are generated below.

```
> library(mvtnorm)
> mvn.n <- 10^5
> mvn.mu1 <- c(1, rep(0, 14))
> mvn.mu2 <- 10^6 * mvn.mu1
> set.seed(1)
> mvn.theta <- rbind(rmvnorm(mvn.n/2, mean = mvn.mu1), rmvnorm(mvn.n/2,
+     mean = mvn.mu2))
```

The log likelihood of the samples are computed below. An arbitrary constant 1000 is added to each log-likelihood (equivalent to scaling each likelihood by a factor of e^{1000}). This means that the correct marginal log-likelihood is 1000.

```
> mvn.ll <- (1000 + log(dmvnorm(mvn.theta, mean = mvn.mu1)/2 +
+     dmvnorm(mvn.theta, mean = mvn.mu2)/2))
```

We can then apply the algorithm the same way as in section 4.

```
> mvn.mll <- MarginalLikelihood(mvn.theta, mvn.ll)
> mvn.mll$mll
[1] 1000.626
> mvn.mll$conf.interval
[1] 1000.380 1000.953
```

The algorithm returns a reasonable approximation of the correct log-likelihood 1000, and has a reasonably small confidence interval.

6 Conclusion

In this vignette we have applied the `margLikArrogance` package to two example problems. The first was detailed and involved choosing between three models of people's height distributions. The second was abstract but quick, and involved a bimodal 15-dimensional posterior parameter distribution. In applications such as these, I hope the `margLikArrogance` package can provide an easy and useful way to approximate marginal likelihoods.