Probability of Success Using RBesT

Methodology and Code Walkthrough

James Normington

July 8, 2019

In this document, I outline the underlying methodology in the "probability of success" functionality of Sebastian Weber's RBesT package.

Throughout, we assume a likelihood for the current trial data y_c: $y_c \sim f(y_c|\theta)$, where θ is our main quantity of interest and $f()$ is the binomial, Gaussian, or Poisson probability density function depending on the endpoint type. We'll further assume the historical trial data y_h follows the hierarchical model

$$\begin{aligned} &y_h|\theta_h \sim f(y_h|\theta_h) \\ &g(\theta_h|\beta) = \beta + \epsilon_h, \epsilon_h \sim N(0, \tau^2), \text{where} \end{aligned} \tag{1}$$

y_h is the summary statistic (sample proportion or sample mean) yielded by the control set in trial h, θ_h is the treatment effect in trial h, $g()$ is the link function, β is the common mean of $\theta_1, ..., \theta_H$, and ϵ_h is a normally distributed mean 0 error with variance τ^2. The prior distribution for β is Gaussian, with a default mean of 0 and a standard deviation specified by the user. The prior distribution for τ is specified by the user.

Below, I list the steps necessary to incorporate historical trial data in estimating the probability of success at interim, at each step elaborating on the underlying mathematics while completing an example in R. The example uses historical meta-data from placebo arms in relevant studies for the treatment of Crohn's disease. The primary outcome y here is the change from baseline in Crohn's Disease Activity Index over 6 weeks, and is assumed to be normally distributed (so, $g(\theta_h) = \theta_h$). The standard deviation of y is estimated to be 88. Suppose the hypothesis set is

$H_0 : \theta = -50$

$H_1 : \theta < -50$

Some preliminaries in R:

```
#install.packages("RBesT")
library(RBesT)
options(RBesT.MC.control=list(adapt_delta=0.999))
crohn$y.se = 88 / sqrt(crohn$n) # "y" is a column of means; this line computes SE
```

1 Compute MAP prior.

The MAP prior $\pi(\theta_*|y_1, ..., y_h)$ is obtained through MCMC sampling using the hierarchical model in (1). In R,

```
set.seed(2019)
base.MAP.mc = gMAP(cbind(y, y.se) ~ 1| study, family = gaussian,
                   data = crohn, weights = n, beta.prior = 100*sd(crohn$y),
                   tau.dist = "HalfNormal", tau.prior = 1)
```

2 Compute parametric approximation of MAP MCMC samples.

The MAP prior as computed by the gMAP() function consists of thousands of posterior draws. To allow for the computation to proceed quickly upon introducing data at interim, one would want to approximate these raw posterior draws with parametric densities that are conjugate to the likelihood.

The software fits $K = 1$ component, $K = 2$ two component, $K = 3$ three component, and $K = 4$ four component mixture distributions, computing each mixture's AIC:

$$AIC_K = p\lambda - 2ln(\hat{L}_K), \text{ where}$$

p is the number of parameters estimated by the model, λ is the penalty parameter (AIC is by definition $\lambda = 2$ but here by default, $\lambda = 6$ is chosen to favor mixtures with fewer components) and $\hat{L}_K$ is the maximum value of the likelihood where K components are used. To obtain $\hat{L}_K$, the software uses an E-M algorithm to approximate the MAP MCMC samples as the mixture of K parametric distributions:

$$log(\pi(\theta_*|y_1, ..., y_h, a, b)) = \sum_{h=1}^{H} log\left[\sum_{k=1}^{K} w_k \pi_k(\theta_h|a_k, b_k)\right], \text{ where}$$

$k = 1, ..., K$ indexes the mixture components, w_k is the relative weight of mixture component k, and a_k and b_k are the parameters of mixture component k.

The E-M algorithm then extends the MAP prior to

$$log(\pi(\theta_*|y_1, ..., y_h, a, b)) = \int log(\pi(\theta_*, Z|y_1, ..., y_h, a, b))dZ$$

where Z represent the component indicators. The E-M algorithm then iteratively maximizes

$$\mathbb{E}_{Z|\theta_*, \boldsymbol{a}, \boldsymbol{b}}\left[log(\pi(\theta_*|y_1, ..., y_h, a, b, Z))\right]$$

until convergence. The mixture prior with the lowest AIC_K is then automatically selected.

In R,

```
base.MAP = automixfit(base.MAP.mc)
```

3 Robustify the mixture prior by adding non-informative component.

This step adds a non-informative component with a user-specified weight to the mixture prior in case of data-prior disagreement. In the "Binary" endpoint case, this is a $U(0,1)$ prior. In the "Normal" endpoint case, this is a diffuse Gaussian prior with a user-specified mean. In the "Poisson" endpoint case, this is a diffuse Gamma prior. In the latter two cases, the default mean is the mean of the current prior mixture, so the user is *strongly* recommended to specify a mean (the null hypothesized mean would make the most sense). In R,

```
MAP.robust = robustify(base.MAP, weight = 0.2, mean = -50)
```

4 Specify non-informative prior for treatment effect.

In the current RShiny app, this is automatically selected as a $U(0,1)$ prior for binary endpoints, a $N(0,1000)$ prior for Normal endpoints, and a Gamma$(0.001, 0.001)$ prior for Poisson endpoints, with no option for the user to change this.

RBesT expects these to be specified with mixbeta(), mixnorm() or mixgamma() as a triplet of numbers (w, a, b), where w is the relative weight of the mixture component. In the binary case, a and b are the shape parameters of the Beta distribution. In the Normal case, a and b are the mean and standard deviation. In the Poisson case, a and b are the shape and rate parameters of the Gamma distribution. In our example, the server will run

```
treat.prior = mixnorm(c(1, 0, 1000), sigma = 88)
```

5 Specify a decision rule defining trial success.

The decision rules take the form of "the probability of the quantity being less than some hypothesized constant is greater than ...". Use the following table, where columns indicate sided-ness of the alternative hypothesis and rows indicate one or two samples, to create a decision rule:

	one-sided	two-sided
one-sample	$P(\theta \leq \theta_0) > 1 - \alpha/2$	$P(\theta \leq \theta_0) > 1 - \alpha$
two-sample	$P(\theta_1 - \theta_2 \leq \Delta_0) > 1 - \alpha/2$	$P(\theta_1 - \theta_2 \leq \Delta_0) > 1 - \alpha$

where θ_0 and Δ_0 are the null hypothesized values. In our example,

```
decision = decision1S(0.975, qc = -50)
```

resolves to $P(\theta \leq -50) > 0.975$

6 Update the prior with interim data.

Using observed data from the interim analysis, update the prior to reflect the new information.

- In the binary case, this will yield a mixture of Beta distributions: $p_k(\theta|r, n, a_k, b_k) = \text{Beta}(r + a_k, n - r + b_k)$
- In the Normal case, this will yield a mixture of Normal distibutions: $p_k(\theta|\bar{y}, s^2, \mu_{0k}, \sigma_{0k}) = N\left(\frac{\sigma_{0k}^2\bar{y}+s^2\mu_{0k}}{n\sigma_{0k}^2+s^2}, \frac{n\sigma_{0k}^2 s^2}{n\sigma_{0k}^2+s^2}\right)$, where μ_{0k} and σ_{0k}^2 are the prior mean and variance (respectively) of mixture component k
- In the Poisson case. this will yield a mixture of Gamma distributions: $p_k(\theta|\bar{y}, a_k, b_k) = \text{Gamma}(a_k + \bar{y}, b_k + n)$

In our example, assume we observed $n = 50$ patients with a sample mean of -60 at interim:

```
interim = postmix(treat.prior, n = 50, m = -60)
```

7 Define probability of success function.

Assuming the trial's critical value y_{crit}, the conditional power of a trial at interim analysis is

$$CP_{N-n_I}(\theta|y_{n_I}) = \int I(y_N > y_{crit}|y_{n_I})p(y_{N-n_I}|\theta)dy_{N-n_I}, \text{ where}$$

y_{n_I} are the observed outcome measurements at interim, y_{N-n_I} are the yet to be observed post-interim observed outcome measurements, and $y_N = [y_{n_I}, y_{N-n_I}]$.

The probability of success then marginalizes the conditional power over the posterior distribution of θ:

$$PoS = \int CP_{N-n_I}(\theta|y_{n_I})p(\theta|y_{n_I}, y_1, ..., y_H)d\theta$$

In our example, assume we would enroll 25 more patients post-interim, for a total of $N = 75$ patients:

```
interim.PoS = pos1S(interim, n=75-50, decision=decision)
```

8 Update MAP prior with interim information.

```
interim.combined = postmix(MAP.robust, m = -60, n = 50)
```

9 Compute the probability of success given interim and historical data.

```
interim.PoS(interim.combined)
```

10 Use of concurrent trial

Often, there will be a concurrently-running trial whose information can be leveraged. Call the trial we've already looked at Trial A, and suppose there is a concurrently-running Trial B, which at interim observes 75 patients with a sample mean of -65. In R, we can combine the current trials with the historical trials:

```
crohn2 = rbind(crohn, data.frame(study = c("PhIIIA", "PhIIIB"), n = c(50, 75),
   y = c(-60,-65), y.se = 88 / sqrt(c(50, 75))))
```

Then, we'll update the MAP prior with the concurrent data

```
interim.MAP.mc = update(base.MAP.mc, data=crohn2)
```

and then extract the posterior distributions of $\theta_1, ..., \theta_H, \theta_A, \theta_B$

```
interim.MAP.post = as.matrix(interim.MAP.mc)[,1:nrow(crohn2)]
```

Just like before, we can approximate the MCMC samples with a mixture of parametric distributions and then calculate the probability of Trial A's success

```
interimA.allcombined = automixfit(interim.MAP.post[,nrow(crohn2)-1])
interim.PoS(interimA.allcombined)
```

For Trial B, the code is

```
interim.B = postmix(treat.prior, m = -65, n = 75)
interim.PoS.B = pos1S(interim.B, 100-75, decision)
interim.B.allcombined = automixfit(interim.MAP.post[,nrow(crohn2)])
interim.PoS.B(interim.B.allcombined)
```

11 Differential discounting

The gMAP() function allows data sources to be weighted differently. For example, Trial A and Trial B may be more similar than the historical trials, naturally defining two strata:

```
crohn2 = cbind(crohn2, data.frame(stratum = c(rep(2, 6), 1, 1)))
interim.diff.MAP.mc = gMAP(cbind(y, y.se) ~ 1| study, family = gaussian,
                           tau.strata = stratum,
                           data = crohn2, weights = n, beta.prior = 100*sd(crohn$y),
                           tau.dist = "HalfNormal", tau.prior = c(0.5, 1))
```

and again, extract the posterior distributions, approximate them with mixtures of parametric distributions, and compute the probability of trial success

```
interim.diff.MAP.post = as.matrix(interim.diff.MAP.mc)[,1:nrow(crohn2)]
interim.A.diff.allcombined = automixfit(interim.diff.MAP.post[,nrow(crohn2) - 1])
interim.B.diff.allcombined = automixfit(interim.diff.MAP.post[,nrow(crohn2)])
interim.PoS(interim.A.diff.allcombined)
interim.PoS.B(interim.B.diff.allcombined)
```

12 Probability of both trials being successful

Perhaps the probability that *both* trials are successful is more interesting to a trialist. Similar to before, the PoS marginalizes the conditional posterior over the joint density of θ_A and θ_B:

$$PoS = \int\int CP_{N-n_{I_A}}(\theta_A|y_{n_{I_A}})CP_{N-n_{I_B}}(\theta_B|y_{n_{I_B}})p(\theta_A,\theta_B|y_{n_{I_A}},y_{n_{I_B}})d\theta_A d\theta_B,$$

which can be approximated by the conditional power of Trial A and Trial B, averaged over MCMC draws:

$$PoS \approx \sum_{t=1}^{T} CP_{N-n_{I_A}}(\theta_{A,t})CP_{N-n_{I_B}}(\theta_{B,t})$$

which in R is performed by

```
interim.oc.A = oc1S(interim, 75-50, decision)
interim.oc.B = oc1S(interim.B, 100-75, decision)
mean(interim.oc.A(interim.diff.MAP.post[,nrow(crohn2) - 1])
    *interim.oc.B(interim.diff.MAP.post[,nrow(crohn2)]))
```