AIDS progression

Here we will analyse the data in Section 4.2 of Chen, Ibrahim, and Yiannoutsos (1999) using a logistic regression model. The analysis consists of two AIDS clinical trials called ACTG036 and ACTG019, the data for which can be accessed by calling `data(actg036)` and `data(actg019)`, respectively (see below).

The historical data will come from the ACTG019 (Volberding et al. (1990)) study, which was a double-blind placebo-controlled clinical trial comparing zidovudine (AZT) with a placebo in people with CD4 cell counts less than 500. The sample size of complete observations for this study was `n0` = 822. The response variable (`y0`) for these data is binary, with 1 indicating death, development of AIDS or ARC and 0 otherwise. We will use the following covariates: `CD4` cell count, `age`, `treatment` and `race`. To facilitate computation, we will center and scale the continuous covariates (`age` and `CD4` count). In general, we recommend this centering procedure in order to keep coefficients on roughly the same scale and thus avoid difficult posterior geometries for the Stan dynamic Hamiltonian Monte Carlo (dHMC) to explore.

We will use the methods in **hdbayes** to construct informative priors using the ACTG019 data as historical data in order to analyse the data from ACTG036 study (Merigan et al. (1991)), for which we will use the same four covariates. For the ACTG036 study the sample size was `n`= 183.

Let's take a look at the data and take the opportunity to centre the continuous covariates (`CD4` and `age`):

```
library(parallel)
library(posterior)
#> This is posterior version 1.5.0
#>
#> Attaching package: 'posterior'
#> The following objects are masked from 'package:stats':
#>
#>     mad, sd, var
#> The following objects are masked from 'package:base':
#>
#>     %in%, match
library(ggplot2)
library(ggthemes)
library(tibble)
library(wacolors)
library(tidyverse)
#> -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
#> v tidyr   1.3.0     v dplyr   1.1.2
#> v readr   2.1.4     v stringr 1.5.1
#> v purrr   1.0.1     v forcats 0.5.2
#> -- Conflicts ------------------------------------------ tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
library(hdbayes)
```

```
data("actg019")
data("actg036")

summary(actg019)
```

```
#>     outcome             age          treatment          race
#>  Min.   :0.00000   Min.   :19.00   Min.   :0.0000   Min.   :0.0000
#>  1st Qu.:0.00000   1st Qu.:29.00   1st Qu.:0.0000   1st Qu.:1.0000
#>  Median :0.00000   Median :34.00   Median :1.0000   Median :1.0000
#>  Mean   :0.06569   Mean   :34.63   Mean   :0.5085   Mean   :0.9136
#>  3rd Qu.:0.00000   3rd Qu.:39.00   3rd Qu.:1.0000   3rd Qu.:1.0000
#>  Max.   :1.00000   Max.   :71.00   Max.   :1.0000   Max.   :1.0000
#>       cd4
#>  Min.   : 30.0
#>  1st Qu.:254.2
#>  Median :353.5
#>  Mean   :334.6
#>  3rd Qu.:420.8
#>  Max.   :704.0
summary(actg036)
#>     outcome             race          treatment          age
#>  Min.   :0.00000   Min.   :0.0000   Min.   :0.0000   Min.   :12.00
#>  1st Qu.:0.00000   1st Qu.:1.0000   1st Qu.:0.0000   1st Qu.:22.00
#>  Median :0.00000   Median :1.0000   Median :0.0000   Median :29.00
#>  Mean   :0.06011   Mean   :0.9071   Mean   :0.4863   Mean   :30.43
#>  3rd Qu.:0.00000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:37.00
#>  Max.   :1.00000   Max.   :1.0000   Max.   :1.0000   Max.   :66.00
#>       cd4
#>  Min.   : 22.0
#>  1st Qu.:221.8
#>  Median :293.5
#>  Mean   :297.7
#>  3rd Qu.:385.2
#>  Max.   :743.5

actg019$age <- scale(actg019$age)
actg019$cd4 <- scale(actg019$cd4)

actg036$age <- scale(actg036$age)
actg036$cd4 <- scale(actg036$cd4)
```

To establish a baseline, we will estimate the parameters via maximum likelihood, using `glm()` from **stats**:

```
formula <- outcome ~  age + race + treatment + cd4
p <- length(attr(terms(formula), "term.labels")) # number of predictors
family  <- binomial('logit')

fit.mle.cur  <- glm(formula, family, actg036)
fit.mle.hist <- glm(formula, family, actg019)

summary(fit.mle.hist)
#>
#> Call:
#> glm(formula = formula, family = family, data = actg019)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -1.0275  -0.3977  -0.2885  -0.2147   2.8011
#>
```

```
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  -4.0254     1.0262  -3.923 8.76e-05 ***
#> age           0.3410     0.1321   2.581  0.00984 **
#> race          1.5368     1.0244   1.500  0.13355
#> treatment    -0.7412     0.3043  -2.436  0.01485 *
#> cd4          -0.5958     0.1354  -4.402 1.07e-05 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 398.43  on 821  degrees of freedom
#> Residual deviance: 360.54  on 817  degrees of freedom
#> AIC: 370.54
#>
#> Number of Fisher Scoring iterations: 7
summary(fit.mle.cur)
#>
#> Call:
#> glm(formula = formula, family = family, data = actg036)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -1.0674  -0.2938  -0.1705  -0.0926   3.2055
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept) -3.99251    1.30107  -3.069 0.002150 **
#> age          0.17270    0.33265   0.519 0.603648
#> race         0.08954    1.18186   0.076 0.939607
#> treatment   -0.09600    0.71886  -0.134 0.893761
#> cd4         -1.80039    0.49731  -3.620 0.000294 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 83.180  on 182  degrees of freedom
#> Residual deviance: 61.527  on 178  degrees of freedom
#> AIC: 71.527
#>
#> Number of Fisher Scoring iterations: 7

confint(fit.mle.hist)
#> Waiting for profiling to be done...
#>                   2.5 %     97.5 %
#> (Intercept) -6.91489595 -2.4598790
#> age          0.07822343  0.5986140
#> race        -0.02626701  4.4242649
#> treatment   -1.35741690 -0.1576862
#> cd4         -0.86412831 -0.3317272
confint(fit.mle.cur)
```

```
#> Waiting for profiling to be done...
#>                  2.5 %     97.5 %
#> (Intercept) -7.2372472 -1.8398296
#> age         -0.5130633  0.8126854
#> race        -1.9320180  3.1410322
#> treatment   -1.5747131  1.3168469
#> cd4         -2.9132023 -0.9259674

the.data <-  list(actg036,
                  actg019)
```

from which we can see quite some discrepancy in the coefficient estimates. In particular, there is substantial uncertainty in the estimates of the treatment effect, as evidenced by a wide 95% confidence interval. We would thus like to incorporate information from the ACTG019 trial into a prior distribution for the regression coefficients. We will use the plethora of methods available in **hdbayes** to construct informative priors using the available historical data.

Bayesian Hierarchical model (BHM)

Our first approach will be to fit a Bayesian hierarchical model (BHM), where we model the coefficients for historical and current data jointly through a hierarchical (multilevel) structure: the coefficients for each study are drawn from the same multivariate normal distribution. The hyperparameters of this distribution are also assigned (inverse Wishart and inverse Gamma) hyperpriors.

In summary, we employ the model:

$$
\begin{aligned}
y_i | x_i, \beta &\sim \text{Bernoulli}\left(\text{logit}^{-1}(x_i'\beta)\right), \\
y_{0i} | x_{0i}, \beta_0 &\sim \text{Bernoulli}\left(\text{logit}^{-1}(x_{0i}'\beta_0)\right), \\
\beta, \beta_0 &\sim N_p(\mu, \Sigma), \\
\mu &\sim N_p(\mu_0, \Sigma_0), \\
\Sigma &\sim \text{IW}_p(\nu_0, \Psi_0).
\end{aligned}
$$

where by default we set

- $\mu_0 = 0$;
- $\Sigma_0 = I_p$;
- $\nu_0 = p + 10$;
- $\Psi_0 = I_p$,

where p is the number of predictors (including the intercept if applicable).

To fit this model, let us first set up the computational specs, which will also be used in the subsequent analyses.

```
ncores <- 4 # max(1, parallel::detectCores() - 2)
nchains <- ncores
warmup  <- 1000
total.samples <- 10000   ## number of samples post warmup
samples <- ceiling( total.samples / ncores)  ## outputs approx total.samples samples
base.pars <- c("(Intercept)", "age", "race", "treatment", "cd4")

get_summaries <- function(fit, pars.interest,
                          digits = 2) {
  ## A little function to pull out the summaries in a convenient form
```

```
  out <- subset(posterior::summarise_draws(fit,
                                           .num_args = list(
                                             sigfig = digits,
                                             notation = "dec"
                                           )),
                variable %in% pars.interest)

  return(out)
}
```

Now, a simple call to `glm.bhm()` will fit the model:

```
time.bhm <- system.time(
  fit.bhm <- glm.bhm(
    formula,
    family,
    data.list = the.data,
    meta.mean.mean = 0,
    meta.mean.sd = 10,
    meta.sd.mean = 0,
    meta.sd.sd = .5,
    parallel_chains = 4,
    iter_warmup = warmup,
    iter_sampling = samples,
    refresh = 0
  )
)
#> Running MCMC with 4 parallel chains...
#>
#> Chain 1 finished in 13.1 seconds.
#> Chain 3 finished in 13.4 seconds.
#> Chain 4 finished in 13.7 seconds.
#> Chain 2 finished in 13.9 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 13.5 seconds.
#> Total execution time: 14.1 seconds.
#> Warning: 4 of 10000 (0.0%) transitions ended with a divergence.
#> See https://mc-stan.org/misc/warnings for details.
get_summaries(fit.bhm, pars.interest = base.pars)
#> # A tibble: 5 x 10
#>   variable     mean median    sd   mad    q5    q95  rhat ess_bulk ess_tail
#>   <chr>       <dbl>  <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl>    <dbl>    <dbl>
#> 1 (Intercept) -4.2   -4.1   0.91  0.86 -5.8  -2.8     1.0    6484.    4645.
#> 2 age          0.23   0.24  0.24  0.22 -0.19  0.59    1.0   10589.    7927.
#> 3 race         1.0    0.98  0.91  0.85 -0.30  2.6     1.0    6627.    4840.
#> 4 treatment   -0.63  -0.64  0.44  0.40 -1.3   0.089   1.0   10357.    8277.
#> 5 cd4         -1.2   -1.2   0.39  0.39 -1.9  -0.67    1.0    6330.    7134.
```

As we can see, all MCMC (Stan) diagnostics (Rhat, divergences, etc) look good so we can continue.

Commensurate prior

Next, we consider the commensurate prior of Hobbs, Sargent, and Carlin (2012). This model has the following structure:

$$
\begin{aligned}
y_i|x_i, \beta &\sim \text{Bernoulli}\left(\text{logit}^{-1}(x_i'\beta)\right), \\
y_{0i}|x_{0i}, \beta_0 &\sim \text{Bernoulli}\left(\text{logit}^{-1}(x_{0i}'\beta_0)\right), \\
\beta_0 &\sim N_p(\mu_0, \Sigma_0), \\
\beta_j &\sim N_1\left(\beta_{0j}, \tau_j^{-1}\right), j = 1, \ldots, p,
\end{aligned}
$$

where the τ_j's are precisions elicited by the user. The defaults in **hdbayes** are

- $\mu_0 = 0$
- $\Sigma_0 = 100 \times I_p$

Again, a simple call to the appropriate function

```
time.commensurate <- system.time(
  fit.commensurate <- glm.commensurate(
    formula,
    family,
    tau = rep(5, p + 1),
    data.list = the.data,
    parallel_chains = 4,
    iter_warmup = warmup,
    iter_sampling = samples,
    refresh = 0
  )
)
#> Running MCMC with 4 parallel chains...
#>
#> Chain 1 finished in 6.7 seconds.
#> Chain 4 finished in 7.2 seconds.
#> Chain 3 finished in 7.3 seconds.
#> Chain 2 finished in 7.5 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 7.2 seconds.
#> Total execution time: 7.5 seconds.

get_summaries(fit.commensurate, pars.interest = base.pars)
#> # A tibble: 5 x 10
#>   variable     mean median    sd   mad    q5    q95  rhat ess_bulk ess_tail
#>   <chr>       <dbl>  <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl>    <dbl>    <dbl>
#> 1 (Intercept) -4.1   -4.1   0.89  0.86 -5.7  -2.8     1.0    2993.    2984.
#> 2 age          0.19   0.19  0.26  0.26 -0.25  0.60    1.0   11219.    6213.
#> 3 race         1.1    1.0   0.88  0.84 -0.17  2.6     1.0    3050.    2987.
#> 4 treatment   -0.64  -0.64  0.42  0.42 -1.3   0.045   1.0    9456.    7062.
#> 5 cd4         -1.2   -1.2   0.28  0.28 -1.6  -0.71    1.0   10638.    7105.
```

will fit the model. And again the diagnostics are fine, so we may proceed with our exploration of informative priors.

Robust Meta-Analytic Predictive (RMAP) Prior

The Robust MAP prior (Schmidli et al. (2014)) is a generalization of the Bayesian Hierarchical Model (BHM), and takes the form

$$
\begin{aligned}
y_i|x_i, \beta &\sim \text{Bernoulli}\left(\text{logit}^{-1}(x_i'\beta)\right), \\
y_{0i}|x_{0i}, \beta_0 &\sim \text{Bernoulli}\left(\text{logit}^{-1}(x_{0i}'\beta_0)\right), \\
\beta_0 &\sim N_p(\mu, \Sigma), \\
\beta &\sim w \times N_p(\mu, \Sigma) + (1-w)N_p(\mu_v, \Sigma_v), \\
\mu &\sim N_p(\mu_0, \Sigma_0), \\
\Sigma &\sim \text{IW}_p(\nu_0, \Psi_0),
\end{aligned}
$$

where $w \in (0, 1)$ controls for the level of borrowing of the historical data. Note that when $w = 1$, the robust MAP prior effectively becomes the BHM. The defaults are the same as in the BHM and the default value for w is 0.1. See the sensitivity analysis vignette for insight into how to set w. The RMAP can be called with `glm.robustmap()`:

```
rmap.t1 <- system.time(
  fit.bhm.hist <- glm.rmap.bhm(
    formula,
    family,
    hist.data.list = list(actg019),
    meta.mean.mean = 0,
    meta.mean.sd = 10,
    meta.sd.mean = 0,
    meta.sd.sd = .5,
    parallel_chains = 4,
    iter_warmup = warmup,
    iter_sampling = samples,
    thin = 10,
    refresh = 0
  )
)
#> Running MCMC with 4 parallel chains...
#>
#> Chain 1 finished in 11.3 seconds.
#> Chain 3 finished in 12.0 seconds.
#> Chain 2 finished in 12.4 seconds.
#> Chain 4 finished in 13.0 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 12.1 seconds.
#> Total execution time: 13.1 seconds.
samples_bhm <- fit.bhm.hist$beta_pred

rmap.t2 <- system.time(res_approx <- glm.rmap.bhm.approx(
  samples.bhm = samples_bhm,
  G = 1:9,
  verbose = FALSE
))

rmap.t3 <- system.time(
  fit.rmap.a <- glm.rmap(
    formula,
```

```
    family,
    curr.data = actg036,
    probs = res_approx$probs,
    means = res_approx$means,
    covs = res_approx$covs,
    w = 0.1,
    parallel_chains = 4,
    iter_warmup = warmup,
    iter_sampling = samples,
    refresh = 0
  )
)
#> Running MCMC with 4 parallel chains...
#>
#> Chain 1 finished in 0.9 seconds.
#> Chain 2 finished in 1.0 seconds.
#> Chain 3 finished in 1.0 seconds.
#> Chain 4 finished in 1.1 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 1.0 seconds.
#> Total execution time: 1.2 seconds.

rmap.t4 <- system.time(
  fit.rmap.b <- glm.rmap(
    formula,
    family,
    curr.data = actg036,
    probs = res_approx$probs,
    means = res_approx$means,
    covs = res_approx$covs,
    w = 0.5,
    parallel_chains = 4,
    iter_warmup = warmup,
    iter_sampling = samples,
    refresh = 0
  )
)
#> Running MCMC with 4 parallel chains...
#>
#> Chain 3 finished in 1.0 seconds.
#> Chain 1 finished in 1.1 seconds.
#> Chain 2 finished in 1.1 seconds.
#> Chain 4 finished in 1.1 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 1.1 seconds.
#> Total execution time: 1.2 seconds.

rmap.t5 <- system.time(
  fit.rmap.c <- glm.rmap(
    formula,
    family,
```

```
    curr.data = actg036,
    probs = res_approx$probs,
    means = res_approx$means,
    covs = res_approx$covs,
    w = 0.9,
    parallel_chains = 4,
    iter_warmup = warmup,
    iter_sampling = samples,
    refresh = 0
  )

)
#> Running MCMC with 4 parallel chains...
#>
#> Chain 3 finished in 1.0 seconds.
#> Chain 4 finished in 1.0 seconds.
#> Chain 1 finished in 1.1 seconds.
#> Chain 2 finished in 1.1 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 1.0 seconds.
#> Total execution time: 1.2 seconds.

time.rmap <- rmap.t1 + rmap.t2 + rmap.t4

get_summaries(fit.rmap.a,
              pars.interest = base.pars)
#> # A tibble: 5 x 10
#>   variable     mean median    sd   mad    q5   q95  rhat ess_bulk ess_tail
#>   <chr>       <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
#> 1 (Intercept) -4.1   -4.1   0.99  0.94 -5.8  -2.6    1.0    4751.    4031.
#> 2 age          0.22   0.24  0.23  0.19 -0.20  0.57   1.0    8360.    5399.
#> 3 race         0.93   0.87  0.96  0.93 -0.56  2.5    1.0    5175.    4284.
#> 4 treatment   -0.61  -0.62  0.46  0.44 -1.4   0.17   1.0    8204.    5548.
#> 5 cd4         -1.3   -1.3   0.41  0.42 -2.0  -0.71   1.0    6368.    5631.
get_summaries(fit.rmap.b,
              pars.interest = base.pars)
#> # A tibble: 5 x 10
#>   variable     mean median    sd   mad    q5   q95  rhat ess_bulk ess_tail
#>   <chr>       <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
#> 1 (Intercept) -4.1   -4.1   0.98  0.93 -5.8  -2.6    1.0    4300.    4398.
#> 2 age          0.22   0.24  0.24  0.20 -0.21  0.57   1.0    7187.    4670.
#> 3 race         0.92   0.89  0.95  0.91 -0.54  2.5    1.0    4746.    4355.
#> 4 treatment   -0.60  -0.61  0.46  0.44 -1.4   0.18   1.0    6958.    5147.
#> 5 cd4         -1.3   -1.3   0.41  0.42 -2.0  -0.70   1.0    6232.    5412.
get_summaries(fit.rmap.c,
              pars.interest = base.pars)
#> # A tibble: 5 x 10
#>   variable     mean median    sd   mad    q5   q95  rhat ess_bulk ess_tail
#>   <chr>       <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
#> 1 (Intercept) -4.1   -4.1   0.97  0.93 -5.8  -2.6    1.0    5195.    4913.
#> 2 age          0.22   0.24  0.23  0.20 -0.19  0.56   1.0    7086.    4977.
#> 3 race         0.93   0.89  0.94  0.90 -0.55  2.5    1.0    5530.    4883.
```

```
#> 4 treatment   -0.60  -0.61  0.45  0.43 -1.3   0.15   1.0    7876.    6234.
#> 5 cd4         -1.3   -1.3   0.41  0.42 -2.0  -0.70   1.0    7229.    5511.
```

The diagnostics look good, so we will proceed.

Power priors

In this section we will explore the power prior (PP, Ibrahim and Chen (2000)) and its variations: the normalised power prior (NPP, Duan, Smith, and Ye (2006), Neuenschwander, Branson, and Spiegelhalter (2009)) and the normalised asymptotic power prior (NAPP, Ibrahim et al. (2015)). The Power Prior takes the form

$$
\begin{aligned}
y_i|x_i,\beta &\sim \text{Bernoulli}\left(\text{logit}^{-1}(x_i'\beta)\right),\\
y_{0i}|x_{0i},\beta &\sim \text{Bernoulli}\left(\text{logit}^{-1}(x_{0i}'\beta)\right),\\
\pi(\beta|a_0) &\propto L(\beta|y_0)^{a_0}\pi_0(\beta),
\end{aligned}
$$

where $L(\beta|y_0)$ is the likelihood of the GLM based on the historical data, $a_0 \in (0,1)$ is a fixed hyperaparameter controlling the effective sample size contributed by the data (e.g., $a_0 = 1$ borrows the whole sample size), and $\pi_0(\beta)$ is an "initial prior" on β.

The default in **hdbayes** is a (noninformative) normal prior on β:

$$\beta \sim N_p(0, 100 \times I_p).$$

The question that then arises is how to choose a_0. While no definitive answer can be given without context of the specific data and model under analysis, we find it reasonable to set $a_0 = a^\star = \frac{1}{2}\frac{n}{n_0}$. This way, when the historical and current data sets are of the same size, we set the borrowing factor to $a_0 = 1/2$, reflecting no particular desire to either borrow or not borrow information. Let us do just that

```
n0 <- nrow(actg019)
n <- nrow(actg036)
a0.star <- (n/n0) * 1/2
```

and then proceed to call `glm.pp()` twice (once for $a_0 = 1/2$ and once for $a_0 = a^\star$):

```
time.pp <- system.time(
  fit.pp <- glm.pp(
    formula,
    family,
    data.list = the.data,
    a0 = 0.5,
    parallel_chains = 4,
    iter_warmup = warmup,
    iter_sampling = samples,
    refresh = 0
  )
)
#> Running MCMC with 4 parallel chains...
#>
#> Chain 3 finished in 5.5 seconds.
#> Chain 1 finished in 5.8 seconds.
#> Chain 2 finished in 6.1 seconds.
#> Chain 4 finished in 6.6 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 6.0 seconds.
```

```
#> Total execution time: 6.7 seconds.

time.pp.star <- system.time(
  fit.pp.star <- glm.pp(
    formula,
    family,
    data.list = the.data,
    a0 = a0.star,
    parallel_chains = 4,
    iter_warmup = warmup,
    iter_sampling = samples,
    refresh = 0
  )
)
#> Running MCMC with 4 parallel chains...
#>
#> Chain 1 finished in 5.2 seconds.
#> Chain 2 finished in 5.5 seconds.
#> Chain 3 finished in 5.6 seconds.
#> Chain 4 finished in 5.9 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 5.5 seconds.
#> Total execution time: 6.0 seconds.

get_summaries(fit.pp,
              pars.interest = base.pars)
#> # A tibble: 5 x 10
#>   variable     mean median    sd   mad     q5    q95  rhat ess_bulk ess_tail
#>   <chr>       <dbl>  <dbl> <dbl> <dbl>  <dbl>  <dbl> <dbl>    <dbl>    <dbl>
#> 1 (Intercept) -3.9   -3.7   0.99  0.94 -5.7   -2.5     1.0    4477.    3790.
#> 2 age          0.30   0.30  0.16  0.16  0.021  0.56    1.0    8423.    6572.
#> 3 race         1.1    1.0   0.99  0.95 -0.30   2.9     1.0    4697.    4042.
#> 4 treatment   -0.68  -0.68  0.37  0.36 -1.3   -0.086   1.0    8351.    6524.
#> 5 cd4         -0.82  -0.82  0.17  0.17 -1.1   -0.54    1.0    8440.    6560.
get_summaries(fit.pp.star,
              pars.interest = base.pars)
#> # A tibble: 5 x 10
#>   variable     mean median    sd   mad    q5   q95  rhat ess_bulk ess_tail
#>   <chr>       <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
#> 1 (Intercept) -4.0   -3.8   1.3   1.2  -6.3  -2.2    1.0    4446.    3699.
#> 2 age          0.23   0.23  0.26  0.25 -0.21  0.65   1.0    7513.    6203.
#> 3 race         0.75   0.60  1.2   1.2  -0.98  3.0    1.0    4548.    3636.
#> 4 treatment   -0.52  -0.52  0.57  0.55 -1.5   0.41   1.0    8177.    5974.
#> 5 cd4         -1.2   -1.2   0.31  0.31 -1.7  -0.73   1.0    7619.    5477.
```

from which we can see that (i) all of the diagnostics are fine and (ii) the choice of a_0 does seem to matter (look at the estimates for the `treatment` coefficient).

Normalised power prior (NPP) Now we are prepared to start our normalised power prior analysis. The NPP treats the hyperparameter a_0 as random, allowing the data to decide what is the best value. For most models, this requires estimating the normalising constant $Z(a_0) = \int L(\beta|y_0)^{a_0}\pi_0(\beta)d\beta$.

The NPP may be summarized as

$$
\begin{aligned}
y_i|x_i,\beta &\sim \text{Bernoulli}\left(\text{logit}^{-1}(x_i'\beta)\right),\\
y_{0i}|x_{0i},\beta &\sim \text{Bernoulli}\left(\text{logit}^{-1}(x_{0i}'\beta)\right),\\
\pi(\beta|a_0) &\propto \frac{1}{Z(a_0)}L(\beta|y_0)^{a_0}\pi_0(\beta),\\
\pi(a_0) &\propto a_0^{\alpha_0-1}(1-a_0)^{\gamma_0-1}.
\end{aligned}
$$

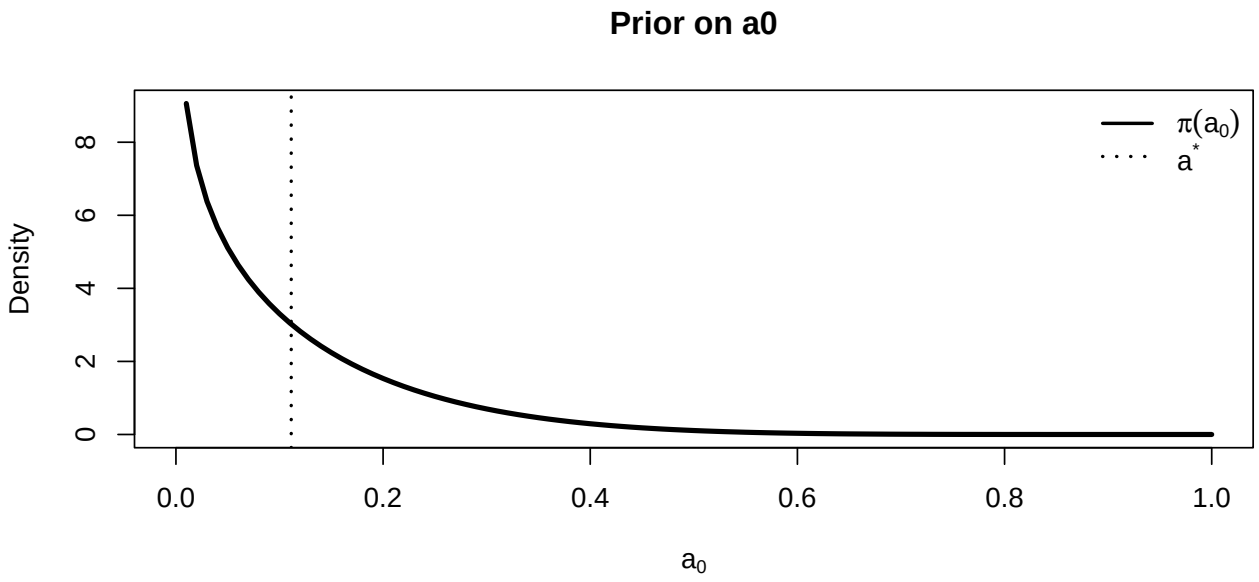
The defaults in **hdbayes** are

- $\pi_0(\beta) \propto N(\beta|0, 100 \times I_p)$;
- $\alpha_0 = 1$;
- $\gamma_0 = 1$,

when $\alpha_0 = 1$ and $\gamma_0 = 1$, the prior on a_0 is a $U(0, 1)$ prior.

We thus might decide to let a_0 vary and be estimated from data. This naturally necessitates placing a prior on a_0. We will now construct a Beta prior for a_0 centred on $a^\star = \frac{1}{2}\frac{n}{n_0}$ with coefficient of variation (cv) equal to 1:

```
devtools::source_url("https://raw.githubusercontent.com/maxbiostat/logPoolR/main/R/beta_elicitator.r")
#> i SHA-1 hash of file is "8124a73b8f4f592c5df173d1d6902c365cfe10da"
beta.pars <- elicit_beta_mean_cv(m0 = a0.star, cv = 1)
curve(dbeta(x, shape1 = beta.pars$a, shape2 = beta.pars$b),
      lwd = 3,
      main = "Prior on a0",
      ylab = "Density",
      xlab = expression(a[0]))
abline(v = a0.star, lwd = 2, lty = 3)
legend(x = "topright",
       legend = c(expression(pi(a[0])),
                  expression(a^'*')),
       lwd = 2, lty = c(1, 3),
       bty = 'n')
```



To conduct an NPP analysis we first need to estimate the normalising constant $c(a_0)$ for a grid of values of a_0. In **hdbayes**, there is one function to estimate the normalising constant across a grid of values for a_0 and

another to obtain posterior samples of the normalized power prior.

```
a0       <- seq(0, 1, length.out = 11)
a0.lognc <- list()
a0.lognc.hdbayes <- data.frame(a0 = a0)

glm_lp <- hdbayes:::glm_lp
get_lp2mean <- hdbayes:::get_lp2mean
normal_glm_lp <- hdbayes:::normal_glm_lp
bernoulli_glm_lp <- hdbayes:::bernoulli_glm_lp
poisson_glm_lp <- hdbayes:::poisson_glm_lp
gamma_glm_lp <- hdbayes:::gamma_glm_lp
invgauss_glm_lp <- hdbayes:::invgauss_glm_lp


## call created function
for (i in 2:length(the.data)) {
  histdata = the.data[[i]]
  ## wrapper to obtain log normalizing constant in parallel package
  logncfun <- function(a0, ...) {
    glm.npp.lognc(
      formula = formula,
      family = family,
      a0 = a0,
      histdata = histdata,
      ...
    )
  }
  cl <- parallel::makeCluster(10)
  parallel::clusterSetRNGStream(cl, 123)
  parallel::clusterExport(
    cl,
    varlist = c(
      'formula',
      'family',
      'histdata',
      'glm.npp.lognc',
      'glm_lp',
      'get_lp2mean',
      'normal_glm_lp',
      'bernoulli_glm_lp',
      'poisson_glm_lp',
      'gamma_glm_lp',
      'invgauss_glm_lp'
    )
  )
  time.npp.1 <- system.time(
    a0.lognc[[i - 1]] <- parallel::parLapply(
      cl = cl,
      X = a0,
      fun = logncfun,
      iter_warmup = warmup,
      iter_sampling = samples,
      chains = 1,
```

```
      refresh = 0
    )
  )

  stopCluster(cl)
  a0.lognc[[i - 1]] <- data.frame(do.call(rbind, a0.lognc[[i - 1]]))
  a0.lognc.hdbayes <- cbind(a0.lognc.hdbayes, a0.lognc[[i - 1]]$lognc)
  colnames(a0.lognc.hdbayes)[i] <- paste0("lognc_hist", i - 1)
}

a0.lognc <- a0.lognc.hdbayes$a0
lognc    <- as.matrix(a0.lognc.hdbayes[, -1, drop = F])
```

We will now fit the NPP with both a uniform (Beta(1, 1)) and the informative prior on a_0 devised above using the dictionary of a_0 and $c(a_0)$ we just created:

```
time.npp.2 <- system.time(
  fit.npp_unif <- glm.npp(
    formula,
    family,
    data.list = the.data,
    a0.lognc = a0.lognc,
    lognc = lognc,
    parallel_chains = 4,
    iter_warmup = warmup,
    iter_sampling = samples,
    refresh = 0
  )
)
#> Running MCMC with 4 parallel chains...
#>
#> Chain 3 finished in 5.6 seconds.
#> Chain 4 finished in 6.5 seconds.
#> Chain 1 finished in 7.3 seconds.
#> Chain 2 finished in 7.5 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 6.7 seconds.
#> Total execution time: 7.5 seconds.

devtools::source_url("https://raw.githubusercontent.com/maxbiostat/logPoolR/main/R/beta_elicitator.r")
#> i SHA-1 hash of file is "8124a73b8f4f592c5df173d1d6902c365cfe10da"
beta.pars <- elicit_beta_mean_cv(m0 = a0.star, cv = 1)


time.npp.2.star <- system.time(
  fit.npp.star <- glm.npp(
    formula,
    family,
    data.list = the.data,
    a0.lognc = a0.lognc,
    lognc = lognc,
    a0.shape1 = beta.pars$a, a0.shape2 = beta.pars$b,
    parallel_chains = 4,
```

```
    iter_warmup = warmup,
    iter_sampling = samples,
    refresh = 0
  )
)
#> Running MCMC with 4 parallel chains...
#>
#> Chain 4 finished in 5.5 seconds.
#> Chain 2 finished in 5.5 seconds.
#> Chain 3 finished in 5.8 seconds.
#> Chain 1 finished in 5.9 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 5.7 seconds.
#> Total execution time: 6.0 seconds.


get_summaries(fit.npp_unif,
              pars.interest = base.pars)
#> # A tibble: 5 x 10
#>   variable     mean median    sd   mad      q5    q95  rhat ess_bulk ess_tail
#>   <chr>       <dbl>  <dbl> <dbl> <dbl>   <dbl>  <dbl> <dbl>    <dbl>    <dbl>
#> 1 (Intercept) -3.8   -3.7   1.0   0.91 -5.6    -2.4     1.0    4183.    3098.
#> 2 age          0.29   0.30  0.18  0.17 -0.0010  0.57    1.0    6110.    4677.
#> 3 race         1.1    0.98  1.0   0.93 -0.40    2.8     1.0    4089.    3564.
#> 4 treatment   -0.68  -0.68  0.38  0.37 -1.3    -0.028   1.0    6370.    5462.
#> 5 cd4         -0.85  -0.82  0.23  0.20 -1.3    -0.54    1.0    4493.    3626.
get_summaries(fit.npp.star,
              pars.interest = base.pars)
#> # A tibble: 5 x 10
#>   variable     mean median    sd   mad    q5   q95  rhat ess_bulk ess_tail
#>   <chr>       <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
#> 1 (Intercept) -3.8   -3.7   1.1   1.0  -5.9  -2.2    1.0    4993.    4095.
#> 2 age          0.26   0.26  0.23  0.22 -0.12  0.62   1.0    8039.    6240.
#> 3 race         0.79   0.68  1.1   1.0  -0.81  2.8    1.0    4897.    4225.
#> 4 treatment   -0.60  -0.58  0.51  0.50 -1.5   0.26   1.0    6885.    5871.
#> 5 cd4         -1.1   -1.0   0.29  0.28 -1.6  -0.63   1.0    5999.    5420.
```

With all diagnostics failing to detect problems we move on.

Normalized asymptotic power prior (NAPP) The Normalized asymptotic power prior (NAPP) uses a large sample theory argument to formulate a normal approximation to the power prior, i.e., the prior is given by

$$\beta|a_0 \sim N(\hat{\beta}_0, a_0^{-1}[I_n(\beta)]^{-1}),$$

where $\hat{\beta}_0$ is the maximum likelihood estimate (MLE) of β based on the historical data and $I_n(\beta)$ is the associated information matrix (negative Hessian).

In this case, the normalising constant is known, so we do not need to estimate it before sampling.

The NAPP can be fitted with a call to `glm.napp()`:

```
time.napp <- system.time(
  fit.napp_unif <- glm.napp(
    formula,
```

```
    family,
    data.list = the.data,
    parallel_chains = 4,
    iter_warmup = warmup,
    iter_sampling = samples,
    refresh = 0
  )
)
#> Running MCMC with 4 parallel chains...
#> Chain 2 Informational Message: The current Metropolis proposal is about to be rejected because of th
#> Chain 2 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance matrix[1,2] = -
#> Chain 2 If this warning occurs sporadically, such as for highly constrained variable types like cova
#> Chain 2 but if this warning occurs often then your model may be either severely ill-conditioned or m
#> Chain 2
#> Chain 1 finished in 1.5 seconds.
#> Chain 3 finished in 1.6 seconds.
#> Chain 4 finished in 1.5 seconds.
#> Chain 2 finished in 1.7 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 1.6 seconds.
#> Total execution time: 1.7 seconds.

time.napp.star <- system.time(
  fit.napp.star <- glm.napp(
    formula,
    family,
    data.list = the.data,
    a0.shape1 = beta.pars$a, a0.shape2 = beta.pars$b,
    parallel_chains = 4,
    iter_warmup = warmup,
    iter_sampling = samples,
    refresh = 0
  )
)
#> Running MCMC with 4 parallel chains...
#> Chain 4 Informational Message: The current Metropolis proposal is about to be rejected because of th
#> Chain 4 Exception: multi_normal_lpdf: Covariance matrix is not symmetric. Covariance matrix[1,2] = -
#> Chain 4 If this warning occurs sporadically, such as for highly constrained variable types like cova
#> Chain 4 but if this warning occurs often then your model may be either severely ill-conditioned or m
#> Chain 4
#> Chain 1 finished in 1.2 seconds.
#> Chain 2 finished in 1.3 seconds.
#> Chain 3 finished in 1.4 seconds.
#> Chain 4 finished in 1.4 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 1.3 seconds.
#> Total execution time: 1.5 seconds.

get_summaries(fit.napp_unif,
              pars.interest = base.pars)
#> # A tibble: 5 x 10
```

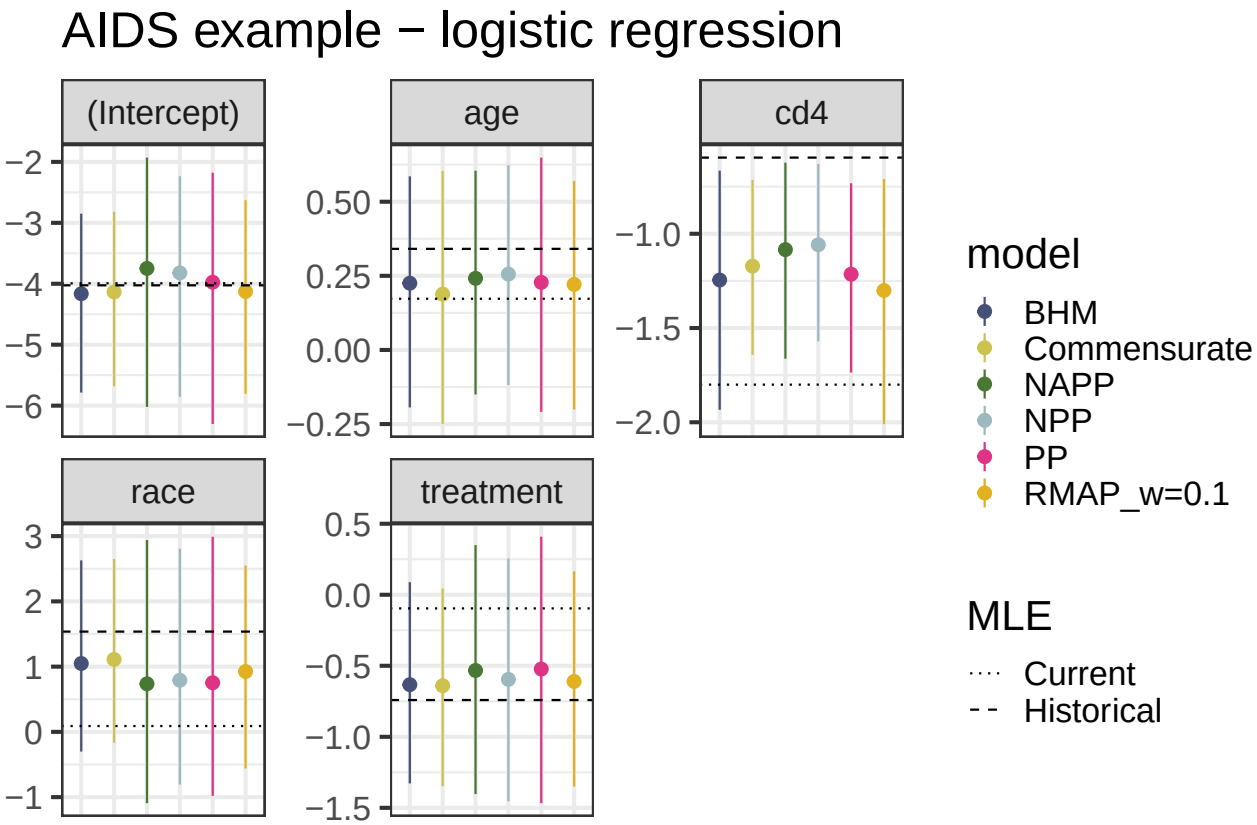
```
#>   variable     mean median    sd   mad      q5     q95  rhat ess_bulk ess_tail
#>   <chr>       <dbl>  <dbl> <dbl> <dbl>   <dbl>   <dbl> <dbl>    <dbl>    <dbl>
#> 1 (Intercept) -3.6   -3.5   1.0   0.99 -5.4    -2.1      1.0    4120.    4018.
#> 2 age          0.29   0.30  0.17  0.16  0.0070  0.55     1.0    7274.    4904.
#> 3 race         0.87   0.82  1.0   1.0  -0.71    2.6      1.0    4152.    4085.
#> 4 treatment   -0.65  -0.66  0.39  0.36 -1.2    -0.0086   1.0    7456.    5225.
#> 5 cd4         -0.83  -0.80  0.23  0.19 -1.2    -0.53     1.0    5009.    3660.
get_summaries(fit.napp.star,
              pars.interest = base.pars)
#> # A tibble: 5 x 10
#>   variable     mean median    sd   mad    q5   q95  rhat ess_bulk ess_tail
#>   <chr>       <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
#> 1 (Intercept) -3.7   -3.6   1.3   1.2  -6.0  -1.9    1.0    3813.    3711.
#> 2 age          0.24   0.25  0.23  0.23 -0.15  0.60   1.0    7759.    5872.
#> 3 race         0.74   0.62  1.3   1.2  -1.1   2.9    1.0    3952.    3533.
#> 4 treatment   -0.53  -0.53  0.53  0.51 -1.4   0.35   1.0    7026.    6189.
#> 5 cd4         -1.1   -1.1   0.32  0.30 -1.7  -0.62   1.0    5600.    5432.
```

Comparison of methods

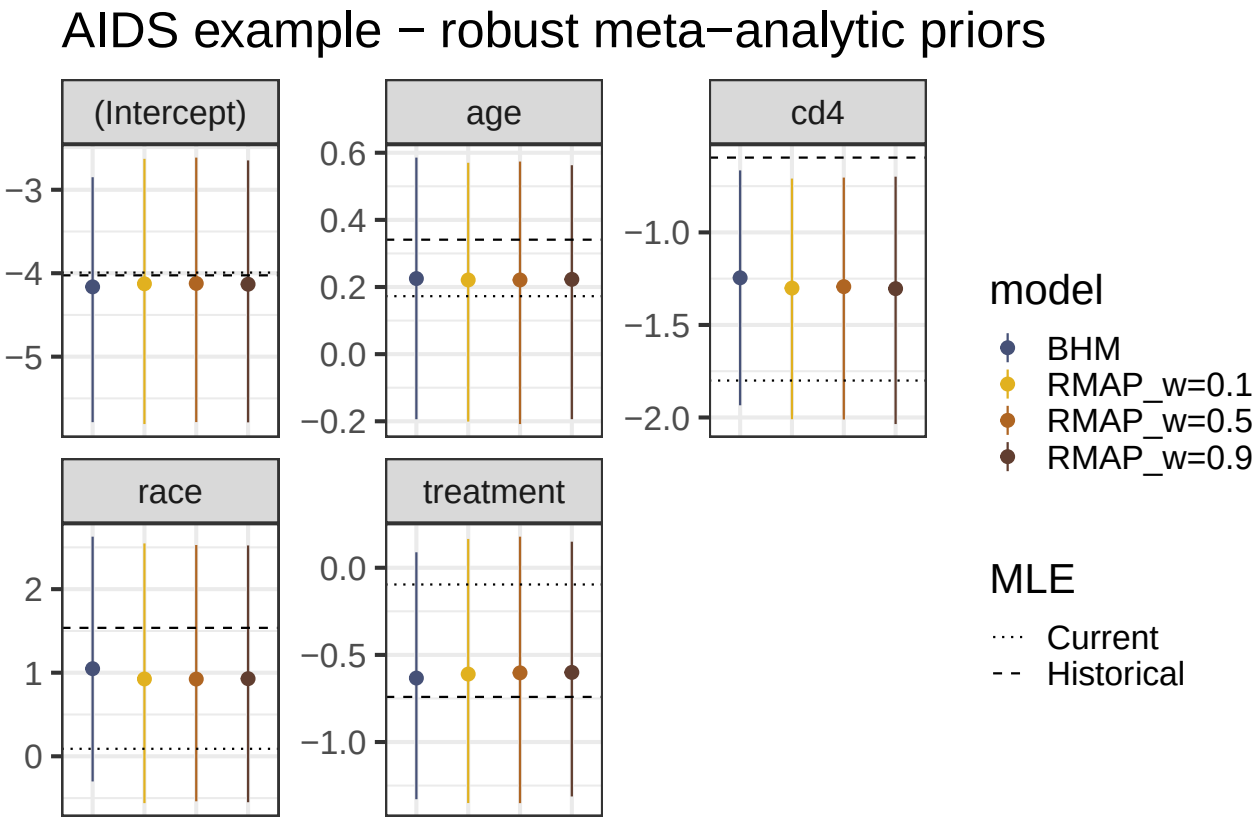
After all this work, we can now finally compare the point estimate (MLE / posterior mean) and uncertainty (SE / posterior standard deviation) of all the methods considered here.

```
fit.list <- list('BHM' = fit.bhm,
                 'Commensurate' = fit.commensurate,
                 'RMAP_w=0.1' = fit.rmap.a,
                 'RMAP_w=0.5' = fit.rmap.b,
                 'RMAP_w=0.9' = fit.rmap.c,
                 'NAPP_unif' = fit.napp_unif,
                 'NAPP' = fit.napp.star,
                 'NPP_unif' = fit.npp_unif,
                 'NPP' = fit.npp.star,
                 'PP_half' = fit.pp,
                 'PP' = fit.pp.star)
```

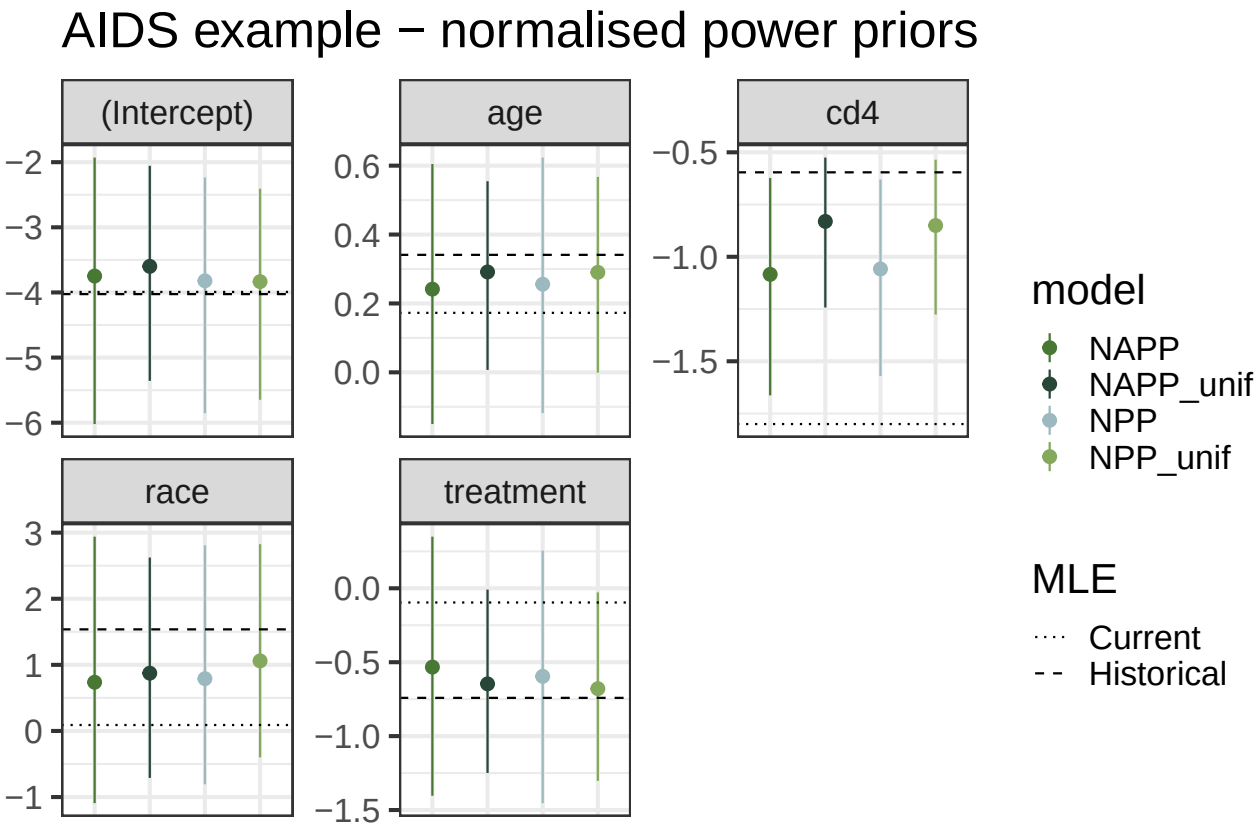
As we can see, there seems to be quite some variation in estimates across methods. To aid our understanding, we will now visualise the posterior distributions for each method:



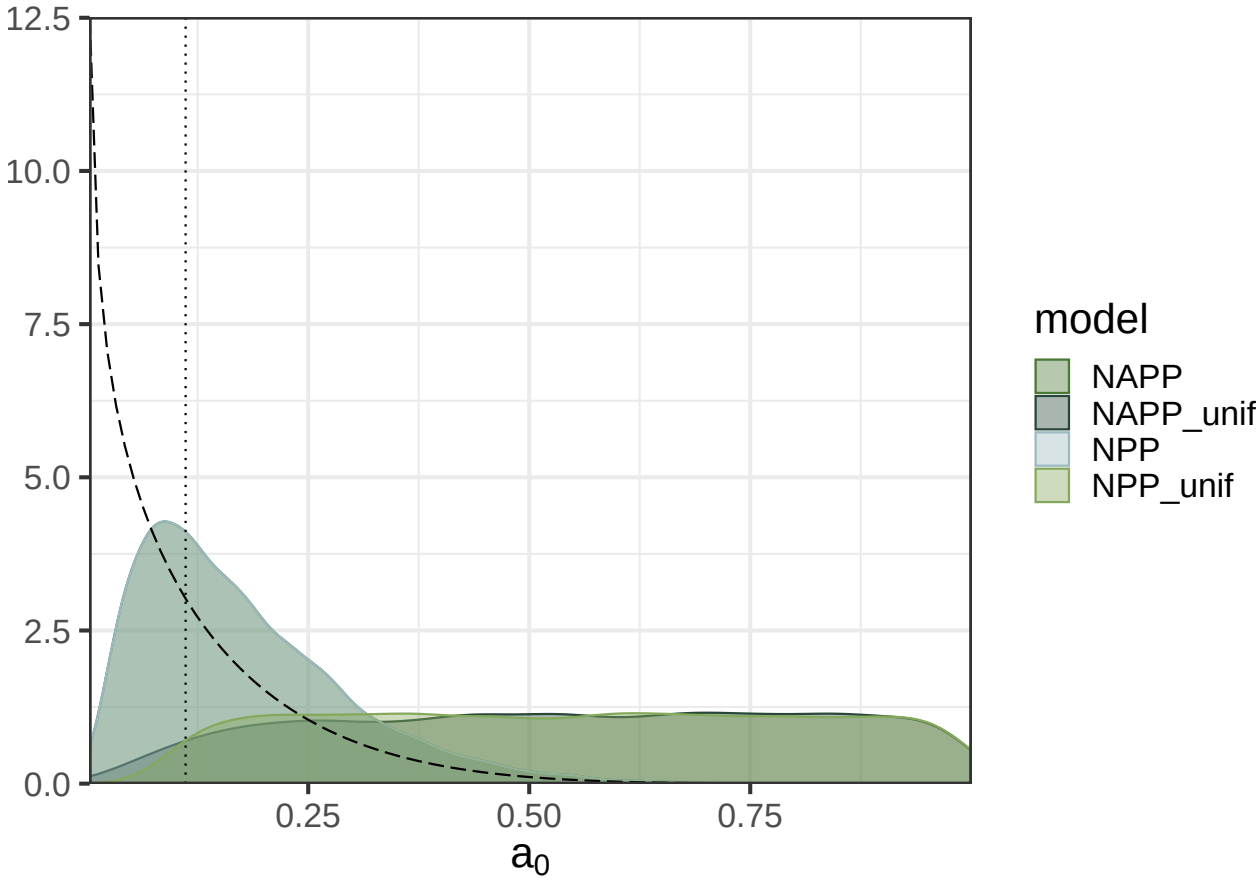
where we have omitted the NPP and NAPP with uniform priors and also the power prior with $a_0 = 1/2$ for clarity. The first observation to make is that the BHM and robust MAP shrink the most towards the historical MLE for the `treatment` effect, but the opposite behaviour occurs for the coefficient for `race`. Let's take a look at the coefficients under the rMAP with different values for w ($w = 0.1$, $w = 0.5$ and $w = 0.9$).



We can now plot the posterior summaries for the coefficients under the NPP and NAPP with uniform and informative priors on a_0 in order to visualise the effect these different prior choices have on the estimated coefficients.



Finally, we will take a look at the posterior distribution of a_0 for the NPP and NAPP models under uniform and informative priors:



where the vertical dotted line marks $a^\star = \frac{1}{2}\frac{n}{n_0}$ and the dashed curve depicts the informative beta prior. As we can see, the prior does make a big difference with regards to the posterior for a_0. This is not unexpected, since a_0 is a hierarchical parameter and learning it from data is not a simple task.

Now a comparison of running times

```
time.bhm
#>    user  system elapsed
#>  54.716   0.293  14.736
time.commensurate
#>    user  system elapsed
#>  29.119   0.219   7.906
time.rmap
#>    user  system elapsed
#>  59.393   0.393  20.833
time.pp.star
#>    user  system elapsed
#>  22.417   0.234   6.323
time.npp.1 + time.npp.2
#>    user  system elapsed
#>  27.214   0.233  29.792
time.npp.1 + time.npp.2.star
#>    user  system elapsed
#>  23.065   0.232  28.237
time.napp
#>    user  system elapsed
#>   6.495   0.223   2.063
```

```
time.napp.star
#>    user  system elapsed
#>   5.578   0.189   1.890
```

References

Chen, M-H, Joseph G Ibrahim, and Constantin Yiannoutsos. 1999. "Prior Elicitation, Variable Selection and Bayesian Computation for Logistic Regression Models." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61 (1): 223–42.

Duan, Yuyan, Eric P Smith, and Keying Ye. 2006. "Using Power Priors to Improve the Binomial Test of Water Quality." *Journal of Agricultural, Biological, and Environmental Statistics* 11 (2): 151–68.

Hobbs, Brian P, Daniel J Sargent, and Bradley P Carlin. 2012. "Commensurate Priors for Incorporating Historical Information in Clinical Trials Using General and Generalized Linear Models." *Bayesian Analysis (Online)* 7 (3): 639.

Ibrahim, Joseph G, and Ming-Hui Chen. 2000. "Power Prior Distributions for Regression Models." *Statistical Science*, 46–60.

Ibrahim, Joseph G, Ming-Hui Chen, Yeongjin Gwon, and Fang Chen. 2015. "The Power Prior: Theory and Applications." *Statistics in Medicine* 34 (28): 3724–49.

Merigan, Thomas C, David A Amato, James Balsley, Maureen Power, William A Price, Sharon Benoit, Angela Perez-Michael, Alan Brownstein, Amy Simon Kramer, and Doreen Brettler. 1991. "Placebo-Controlled Trial to Evaluate Zidovudine in Treatment of Human Immunodeficiency Virus Infection in Asymptomatic Patients with Hemophilia. NHF-ACTG 036 Study Group."

Neuenschwander, Beat, Michael Branson, and David J Spiegelhalter. 2009. "A Note on the Power Prior." *Statistics in Medicine* 28 (28): 3562–66.

Schmidli, Heinz, Sandro Gsteiger, Satrajit Roychoudhury, Anthony O'Hagan, David Spiegelhalter, and Beat Neuenschwander. 2014. "Robust Meta-Analytic-Predictive Priors in Clinical Trials with Historical Control Information." *Biometrics* 70 (4): 1023–32.

Volberding, Paul A, Stephen W Lagakos, Matthew A Koch, Carla Pettinelli, Maureen W Myers, David K Booth, Henry H Balfour Jr, et al. 1990. "Zidovudine in Asymptomatic Human Immunodeficiency Virus Infection: A Controlled Trial in Persons with Fewer Than 500 CD4-Positive Cells Per Cubic Millimeter." *New England Journal of Medicine* 322 (14): 941–49.

Computing environment

```
sessionInfo()
#> R version 4.2.2 (2022-10-31)
#> Platform: x86_64-pc-linux-gnu (64-bit)
#> Running under: Pop!_OS 22.04 LTS
#>
#> Matrix products: default
#> BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.10.0
#> LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0
#>
#> locale:
#>  [1] LC_CTYPE=en_GB.UTF-8       LC_NUMERIC=C
#>  [3] LC_TIME=en_GB.UTF-8        LC_COLLATE=en_GB.UTF-8
#>  [5] LC_MONETARY=en_GB.UTF-8    LC_MESSAGES=en_GB.UTF-8
#>  [7] LC_PAPER=en_GB.UTF-8       LC_NAME=C
#>  [9] LC_ADDRESS=C               LC_TELEPHONE=C
#> [11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C
#>
#> attached base packages:
```

```
#> [1] parallel  stats     graphics  grDevices utils     datasets  methods
#> [8] base
#>
#> other attached packages:
#>  [1] hdbayes_0.0.0.9000 forcats_0.5.2      stringr_1.5.1      dplyr_1.1.2
#>  [5] purrr_1.0.1        readr_2.1.4        tidyr_1.3.0        tidyverse_1.3.2
#>  [9] wacolors_0.3.1     tibble_3.2.1       ggthemes_4.2.4     ggplot2_3.4.4
#> [13] posterior_1.5.0
#>
#> loaded via a namespace (and not attached):
#>   [1] googledrive_2.0.0    colorspace_2.1-0     ellipsis_0.3.2
#>   [4] mclust_6.0.1         fs_1.6.3             rstudioapi_0.14
#>   [7] farver_2.1.1         remotes_2.4.2        fansi_1.0.6
#>  [10] mvtnorm_1.2-4        lubridate_1.9.2      xml2_1.3.3
#>  [13] bridgesampling_1.1-2 codetools_0.2-18     cachem_1.0.7
#>  [16] knitr_1.41           pkgload_1.3.3        jsonlite_1.8.7
#>  [19] broom_1.0.2          dbplyr_2.3.0         shiny_1.7.4
#>  [22] compiler_4.2.2       httr_1.4.4           backports_1.4.1
#>  [25] assertthat_0.2.1     Matrix_1.5-1         fastmap_1.1.1
#>  [28] gargle_1.2.1         cli_3.6.2            later_1.3.0
#>  [31] htmltools_0.5.4      prettyunits_1.2.0    tools_4.2.2
#>  [34] enrichwith_0.3.1     coda_0.19-4.1        gtable_0.3.4
#>  [37] glue_1.7.0           Rcpp_1.0.11          cellranger_1.1.0
#>  [40] vctrs_0.6.5          tensorA_0.36.2.1     xfun_0.39
#>  [43] ps_1.7.6             rvest_1.0.3          timechange_0.2.0
#>  [46] mime_0.12            miniUI_0.1.1.1       lifecycle_1.0.4
#>  [49] formula.tools_1.7.1  devtools_2.4.5       googlesheets4_1.0.1
#>  [52] instantiate_0.2.1    MASS_7.3-58.1        scales_1.3.0
#>  [55] hms_1.1.3            promises_1.2.0.1     Brobdingnag_1.2-9
#>  [58] curl_5.0.0           yaml_2.3.6           memoise_2.0.1
#>  [61] stringi_1.8.3        checkmate_2.3.1      pkgbuild_1.4.2
#>  [64] cmdstanr_0.7.0       operator.tools_1.6.3 rlang_1.1.3
#>  [67] pkgconfig_2.0.3      matrixStats_1.2.0    distributional_0.4.0
#>  [70] evaluate_0.23        lattice_0.20-45      labeling_0.4.3
#>  [73] htmlwidgets_1.6.1    processx_3.8.3       tidyselect_1.2.0
#>  [76] magrittr_2.0.3       R6_2.5.1             generics_0.1.3
#>  [79] profvis_0.3.7        DBI_1.1.3            pillar_1.9.0
#>  [82] haven_2.5.2          withr_2.5.2          abind_1.4-5
#>  [85] modelr_0.1.10        crayon_1.5.2         utf8_1.2.4
#>  [88] tzdb_0.4.0           rmarkdown_2.19       urlchecker_1.0.1
#>  [91] usethis_2.1.6        grid_4.2.2           readxl_1.4.2
#>  [94] data.table_1.14.8    callr_3.7.3          reprex_2.0.2
#>  [97] digest_0.6.33        xtable_1.8-4         httpuv_1.6.8
#> [100] munsell_0.5.0        sessioninfo_1.2.2
```