Evaluating the sensitivity of informative priors based on historical data

The power prior (PP) and the robust meta-analytic prior (RMAP) are two very common choices for constructing informative priors by borrowing information from historical data. However, these modelling devices depend on tuning parameters that must be chosen in advance, and it is not clear which values should be picked. In this vignette we will use the data set in the AIDS progression to explore the choice of tuning parameter in a logistic regression setting. Please refer to that document for model, data and research question details.

```
#> 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
#> -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
#> v tibble  3.2.1     v dplyr   1.1.2
#> v tidyr   1.3.0     v stringr 1.5.1
#> v readr   2.1.4     v forcats 0.5.2
#> v purrr   1.0.1
#> -- Conflicts ------------------------------------------ tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
#>
#> Attaching package: 'scales'
#>
#>
#> The following object is masked from 'package:purrr':
#>
#>     discard
#>
#>
#> The following object is masked from 'package:readr':
#>
#>     col_factor
```

Let' setup

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

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

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

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
```

And obtain some maximum likelihood estimates (MLE):

```r
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)

mle.estimates.past <- data.frame(
  parameter = names(coef(fit.mle.hist)),
  value = coef(fit.mle.hist),
  MLE = "Historical"
)
mle.estimates.curr <- data.frame(
  parameter = names(coef(fit.mle.cur)),
  value = coef(fit.mle.cur),
  MLE = "Current"
)

mle.estimates <- rbind(mle.estimates.past,
                       mle.estimates.curr)

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

base.pars <- c("(Intercept)", "age", "race", "treatment", "cd4")

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

Robust Meta-analytic predictive prior (RMAP)

The RMAP (Schmidli et al. (2014)) is a (linear) mixture between the Bayesian hierarchical model (BHM) and a vague prior. Conceptually, if θ is the parameter of interest, we have

$$\pi_{\text{RMAP}}(\theta) = w\pi_{\text{BHM}}(\theta) + (1-w)\pi_{\text{Vague}}(\theta), \tag{1}$$

where $w \in (0,1)$ is a mixture weight that controls for the level of borrowing of the historical data. Note that when $w = 1$, the robust MAP prior should collapse to the BHM. The defaults are the same as in the BHM and the default value for w is 0.1. We will now fit the RMAP for a few values of w and see what effect (if any) this has on the regression coefficients (hence $\theta = \boldsymbol{\beta}$ here). First, let us wrap the function to fit the RMAP, `glm.robustmap()`, into a convenient function that takes a given w and returns the output:

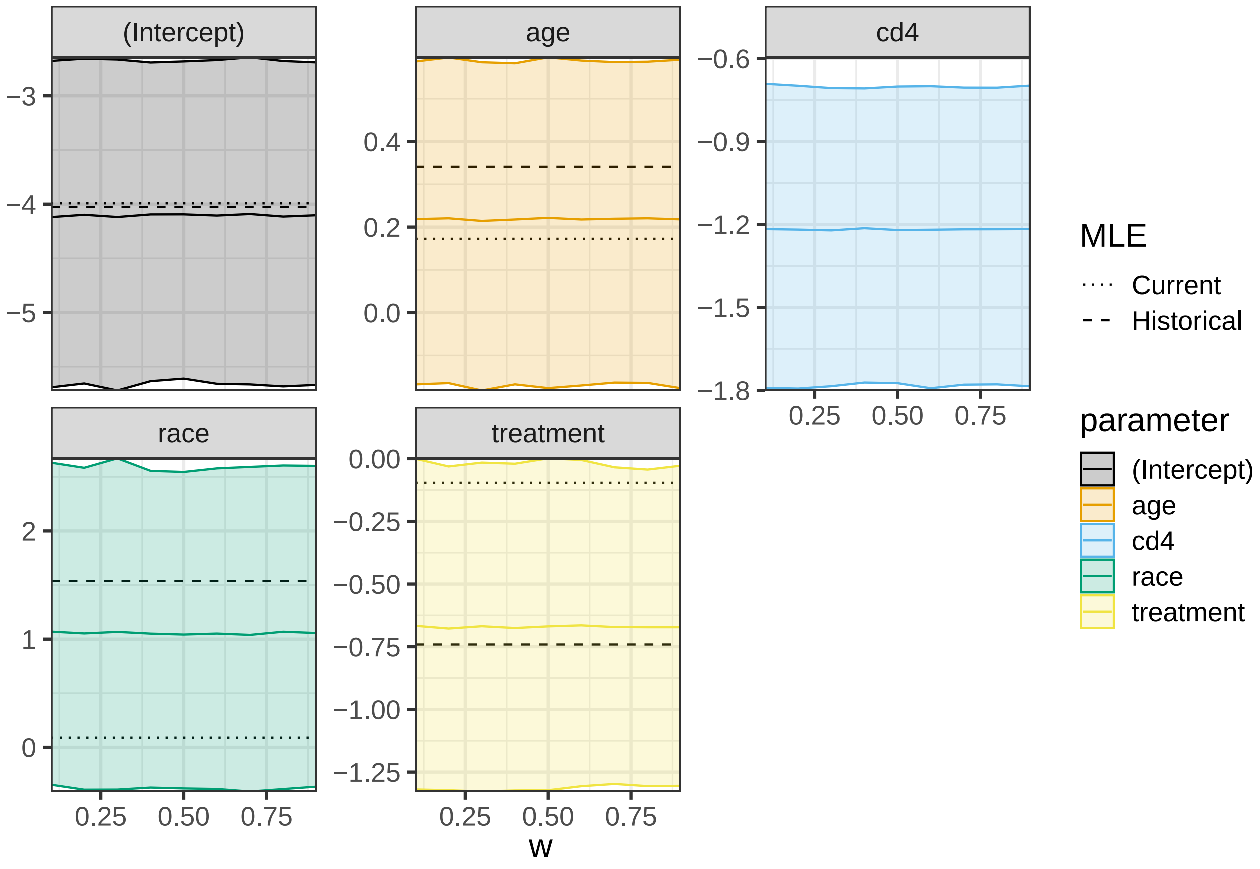
```
fit.bhm.hist <- glm.rmap.bhm(
    formula,
    family,
    hist.data.list = list(actg019),
    meta.mean.mean = 0,
    meta.mean.sd = 10,
    meta.sd.mean = 0,
    meta.sd.sd = .5,
    parallel_chains = 4,
    iter_warmup = warmup,
    iter_sampling = samples,
    thin = 10,
    refresh = 0
  )
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Running MCMC with 4 parallel chains...
#>
#> Chain 1 finished in 9.9 seconds.
#> Chain 2 finished in 10.4 seconds.
#> Chain 3 finished in 10.7 seconds.
#> Chain 4 finished in 10.7 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 10.4 seconds.
#> Total execution time: 10.9 seconds.
#> Warning: 2 of 1000 (0.0%) transitions ended with a divergence.
#> See https://mc-stan.org/misc/warnings for details.

samples_bhm <- fit.bhm.hist$beta_pred

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

Now, we will do this for a small set of weights, just to see what is going on:

```
rmap_fit_and_report <- function(weight){
  cat("Doing w=", weight, "\n")
  fit.robustmap <- glm.rmap(
    formula,
    family,
    curr.data = actg036,
    probs = res_approx$probs,
    means = res_approx$means,
    covs = res_approx$covs,
    w = weight,
    parallel_chains = 4,
    iter_warmup = warmup,
    iter_sampling = samples,
    show_messages = FALSE,
    show_exceptions = FALSE,
    refresh = 0
  )
  return(fit.robustmap)
}

ws.regular <- seq(0.1, 0.9, by = 0.1)

system.time(
  all.fits.regular <- lapply(ws.regular, rmap_fit_and_report)
)
#> Doing w= 0.1
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 0.2
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 0.3
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 0.4
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 0.5
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 0.6
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 0.7
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 0.8
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 0.9
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#>    user  system elapsed
#>  31.915   1.474  11.908

all.summaries.regular <- lapply(1:length(all.fits.regular),  function(i) {
```

```r
  out <- tibble(get_summaries(fit = all.fits.regular[[i]], base.pars),
                name = paste0("w=", ws.regular[i]))
  return(out)
})

results.regular <- do.call(rbind, all.summaries.regular)
results.regular$w <- as.numeric(gsub("w=", "", results.regular$name))
results.regular <- results.regular %>% rename(lwr = q5)
results.regular <- results.regular %>% rename(upr = q95)
results.regular <- results.regular %>% rename(parameter = variable)
```

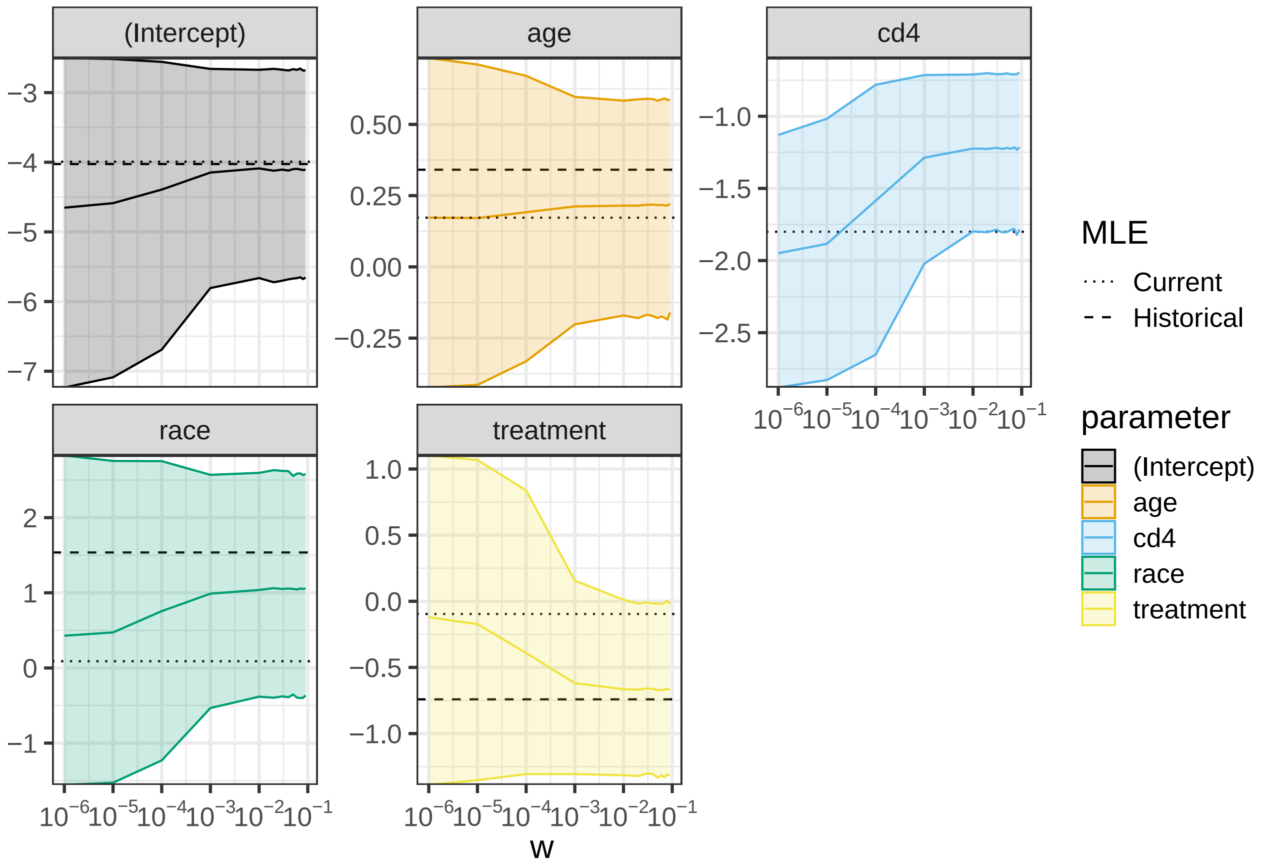
Next, we will produce a little plot to help us visualise things:



As we can see, the actual posterior estimates do not seem to be very sensitive to the value of $w \in [0.1, 0.9]$. This might prompt the curious to ask whether this would change for small values of w. Let's investigate:

```r
ws.small <- c(10^{-(6:2)}, 2:9/100)

system.time(
  all.fits.small <- lapply(ws.small, rmap_fit_and_report)
)
#> Doing w= 1e-06
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 1e-05
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 1e-04
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 0.001
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 0.01
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 0.02
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 0.03
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 0.04
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 0.05
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 0.06
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 0.07
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 0.08
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing w= 0.09
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#>    user  system elapsed
#>  52.170   2.428  18.853

all.summaries.small <- lapply(1:length(all.fits.small),
                              function(i) {
                                out <- tibble(get_summaries(fit = all.fits.small[[i]], base.pars),
                                              name = paste0("w=", ws.small[i]))
                                return(out)
                              })

results.small <- do.call(rbind, all.summaries.small)
results.small$w <- as.numeric(gsub("w=", "", results.small$name))
results.small <- results.small %>% rename(lwr = q5)
results.small <- results.small %>% rename(upr = q95)
results.small <- results.small %>% rename(parameter = variable)
```

Again, let's plot



and voilà! Now we do see that the posterior resulting from the RMAP is indeed sensitive to w, as long as that value is well below 0.1.

Power prior (PP)

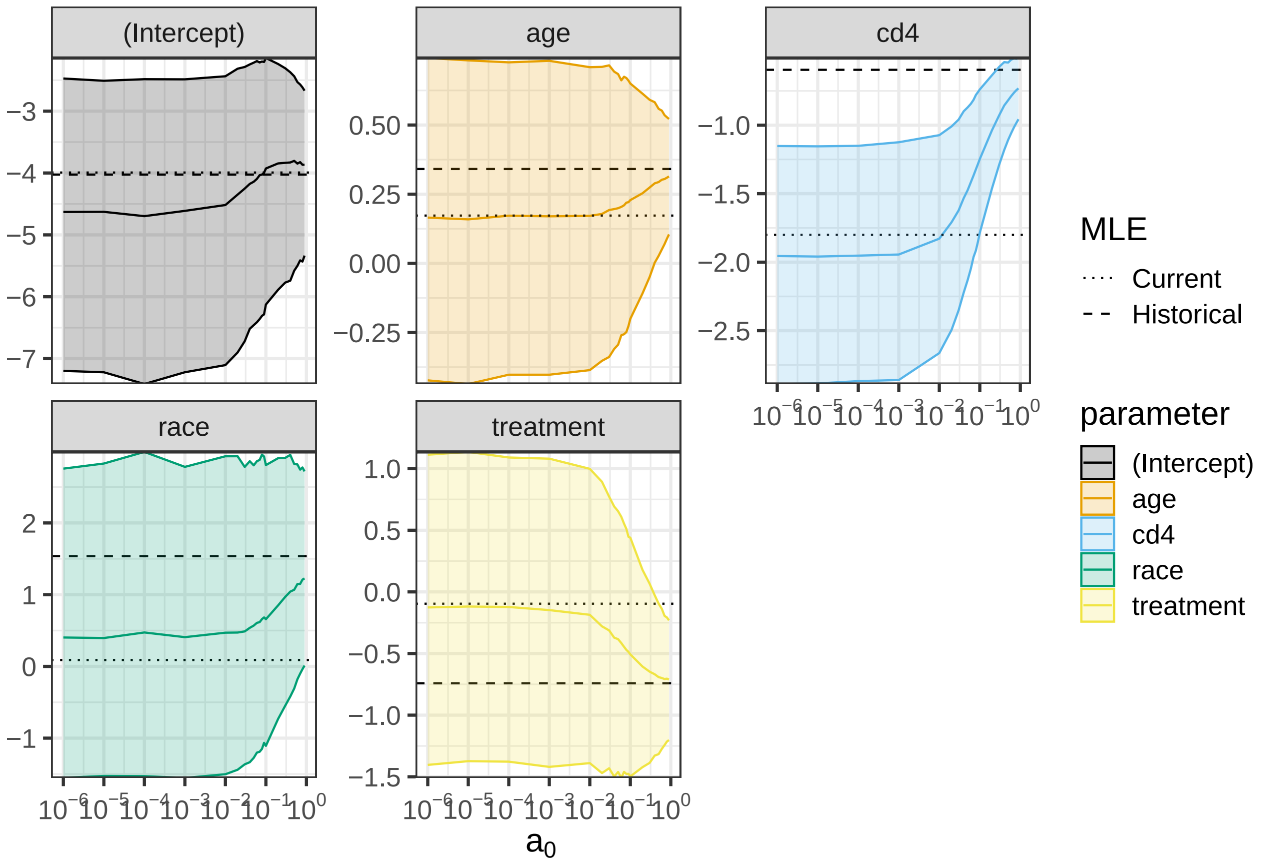
We now turn our attention to the power prior (PP, Ibrahim and Chen (2000)), which raises the likelihood of the historical data to a discounting factor, a_0:

$$\pi_{\text{PP}}(\theta) \propto L(D_0 \mid \theta)^{a_0}\pi_0(\theta). \tag{2}$$

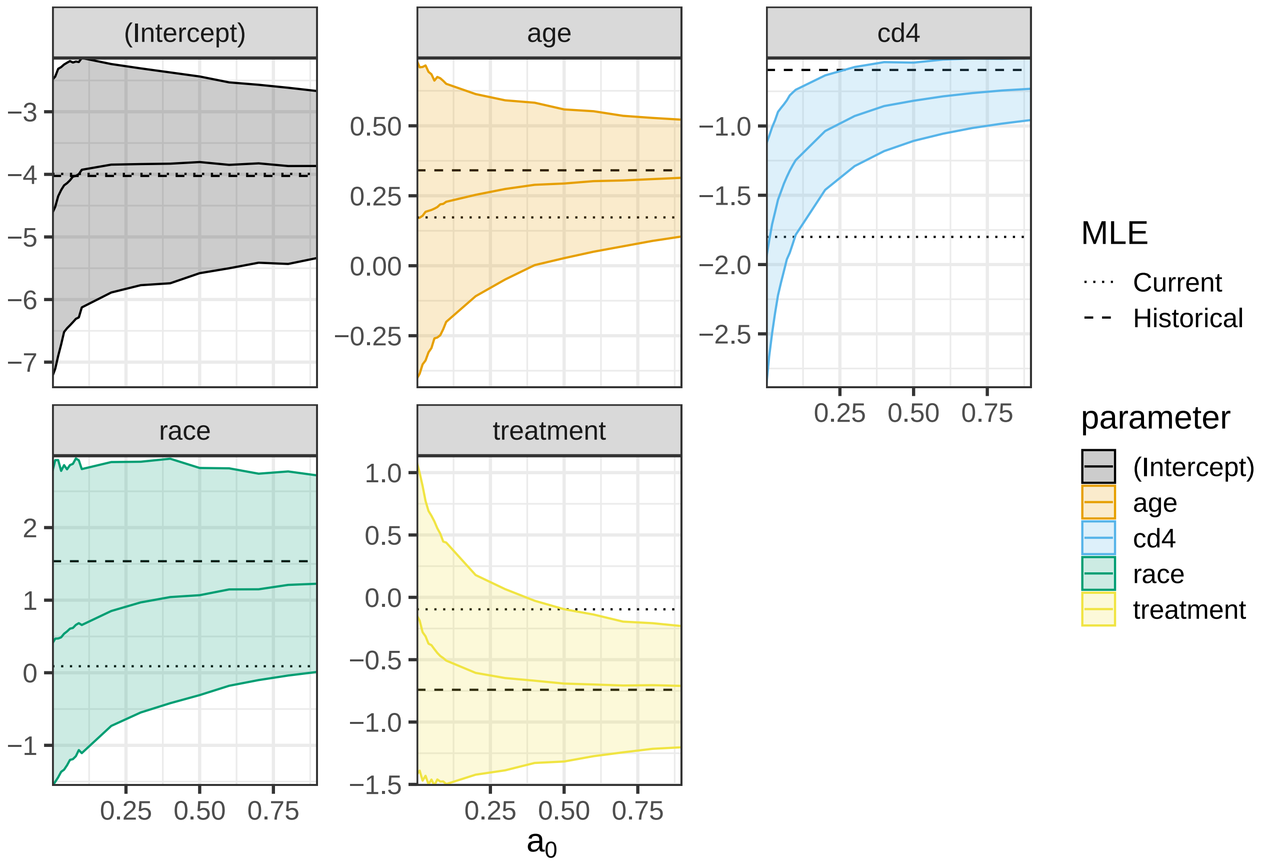
```
pp_fit_and_report <- function(a){
  cat("Doing a0=", a, "\n")
  fit.pp <- glm.pp(
    formula,
    family,
    data.list = the.data,
    a0 = a,
    parallel_chains = 4,
    iter_warmup = warmup,
    iter_sampling = samples,
    show_messages = FALSE,
    show_exceptions = FALSE,
    refresh = 0
  )
  return(fit.pp)
}
```

```
as <- sort(c(ws.small, ws.regular))

system.time(
  all.pp.fits <- lapply(as, pp_fit_and_report)
)
#> Doing a0= 1e-06
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 1e-05
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 1e-04
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.001
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.01
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.02
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.03
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.04
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.05
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.06
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.07
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.08
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.09
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.1
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.2
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.3
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.4
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.5
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.6
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.7
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.8
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.9
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#>    user  system elapsed
#> 424.683   4.288 119.387

all.summaries.pp <- lapply(1:length(all.pp.fits),  function(i) {
  out <- tibble(get_summaries(fit = all.pp.fits[[i]], base.pars),
                name = paste0("a_0=", as[i]))
  return(out)
})

results.pp <- do.call(rbind, all.summaries.pp)
results.pp$a0 <- as.numeric(gsub("a_0=", "", results.pp$name))
results.pp <- results.pp %>% rename(lwr = q5)
results.pp <- results.pp %>% rename(upr = q95)
results.pp <- results.pp %>% rename(parameter = variable)
```

We can now plot the results:



only to realise that the sensitivity is in the $(0.1, 1)$ interval. Let's slightly re-do the plot:



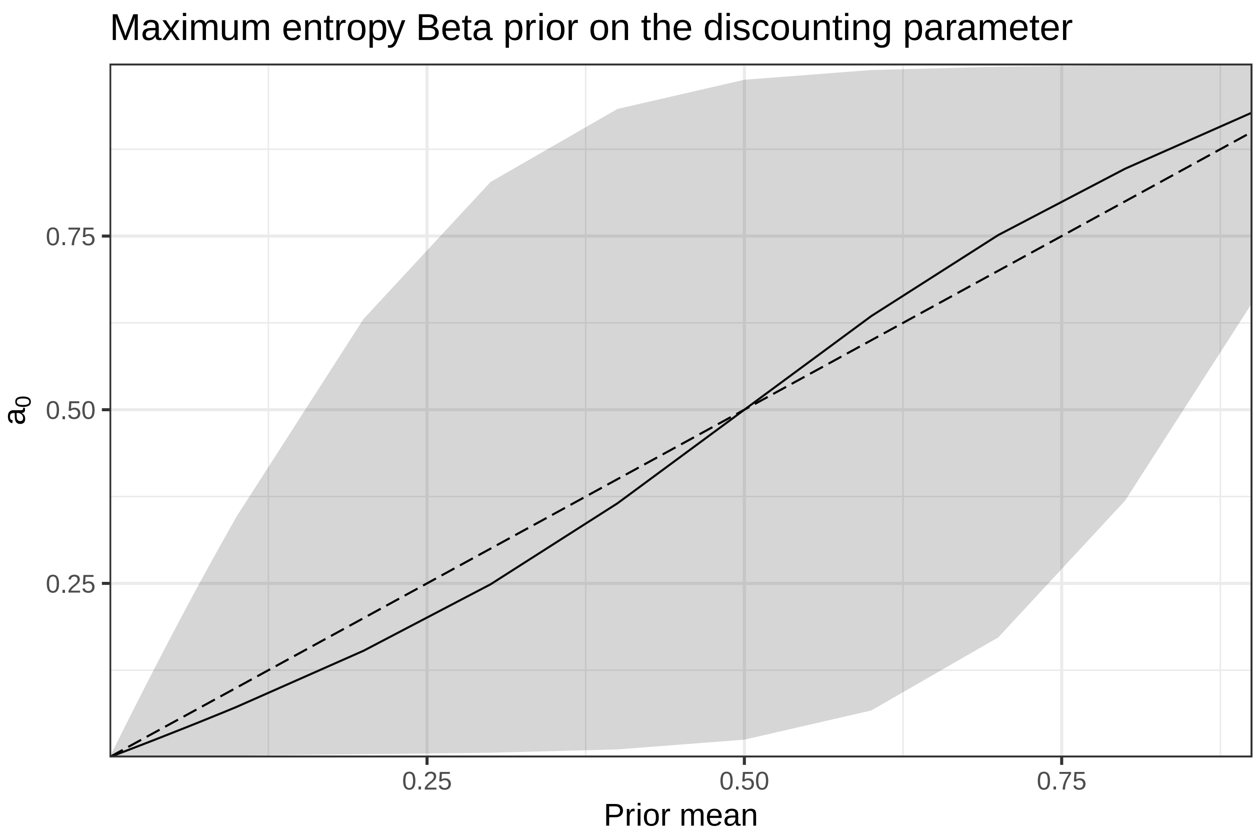
Now we can clearly see that the posteriors seem to start converging when $a_0 > 0.3$, although the picture is not as clear cut as for the RMAP.

Now, let's take a look at prior sensitivity in normalised versions of the power prior, the NPP and NAPP.

First, let's elicit Beta priors with a given prior mean and maximum entropy (subject to having the right mean)

```
# remotes::install_github("maxbiostat/logPoolR")
library(logPoolR)
#> Loading required package: matrixStats
#>
#> Attaching package: 'matrixStats'
#> The following object is masked from 'package:dplyr':
#>
#>     count
#> Loading required package: caTools
#> Loading required package: compiler
#> Loading required package: mvtnorm
#> Loading required package: parallel
hyperpars <- lapply(as, elicit_beta_mean_maxent)
K <- length(hyperpars)
prior.quantiles <- vector(K, mode = "list")
Alpha <- .95
for(k in 1:K) {
  prior.quantiles[[k]] <- tibble(median = qbeta(p = 1/2,
                                                  shape1 = hyperpars[[k]]$a,
                                                  shape2 = hyperpars[[k]]$b),
                                 lwr = qbeta(p = (1 - Alpha)/2,
                                             shape1 = hyperpars[[k]]$a,
                                             shape2 = hyperpars[[k]]$b),
                                 upr = qbeta(p = (1 + Alpha)/2,
                                             shape1 = hyperpars[[k]]$a,
                                             shape2 = hyperpars[[k]]$b))
  prior.quantiles[[k]]$a0 <- as[k]

  prior.quantiles.df <- do.call(rbind, prior.quantiles)
}
```

Now let's plot these (hyper-) priors on a_0:



These seem wide enough. Now, let's prepare what we need to fit the normalised power prior (NPP) under each of these priors.

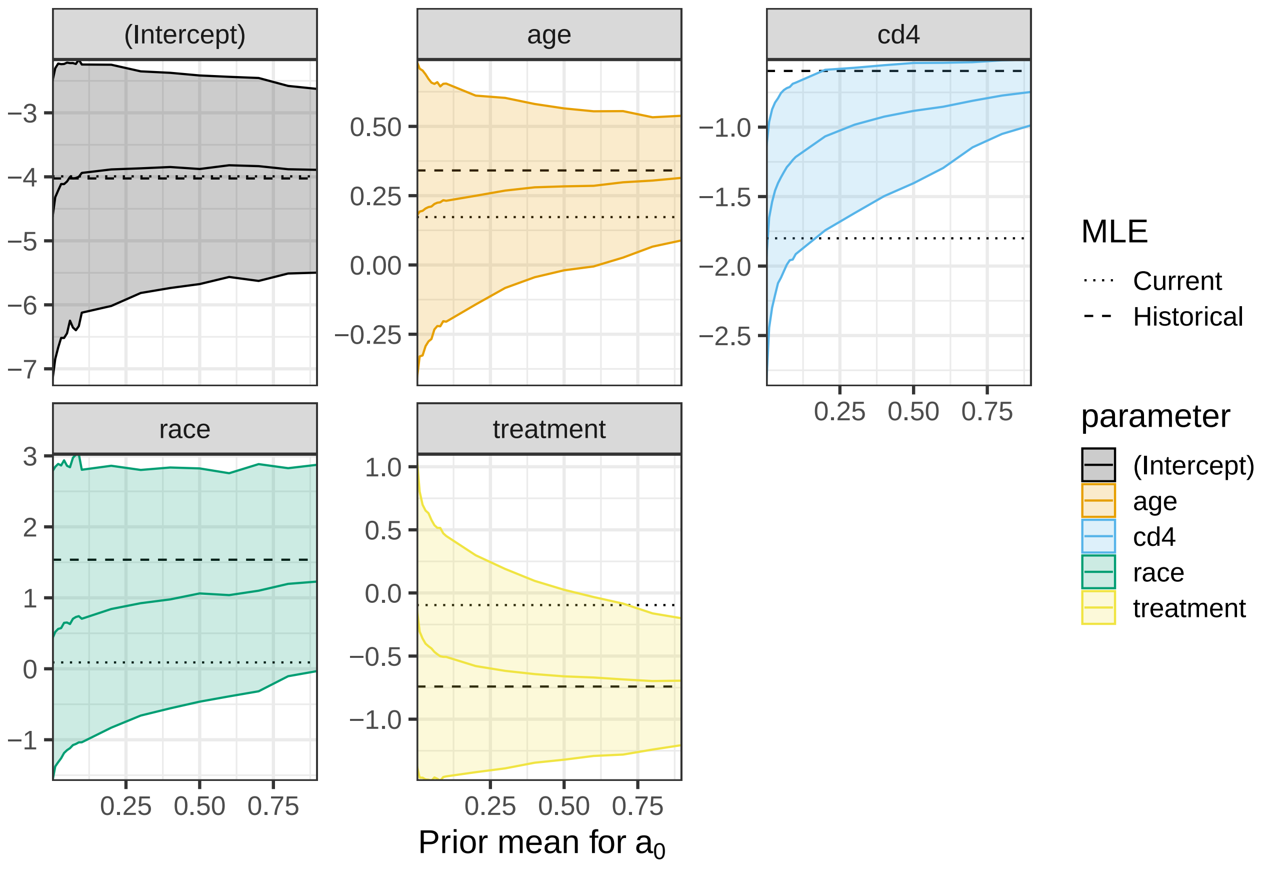
```
a0.lognc <- list()
a0.lognc.hdbayes <- data.frame(a0 = as)

glm_lp <- hdbayes:::glm_lp
get_lp2mean <- hdbayes:::get_lp2mean
bernoulli_glm_lp <- hdbayes:::bernoulli_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
```

```r
  logncfun <- function(a0, ...) {
    glm.npp.lognc(
      formula = formula,
      family = family,
      a0 = a0,
      histdata = histdata,
      show_messages = FALSE,
      show_exceptions = FALSE,
      ...
    )
  }
  cl <- parallel::makeCluster(10)
  parallel::clusterSetRNGStream(cl, 123)
  parallel::clusterExport(
    cl,
    varlist = c(
      'formula',
      'family',
      'histdata',
      'glm.npp.lognc',
      'glm_lp',
      'get_lp2mean',
      'bernoulli_glm_lp'
    )
  )
    a0.lognc[[i - 1]] <- parallel::parLapply(
      cl = cl,
      X = as,
      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])
```

Now we will actually fit the NPP under each of these priors.

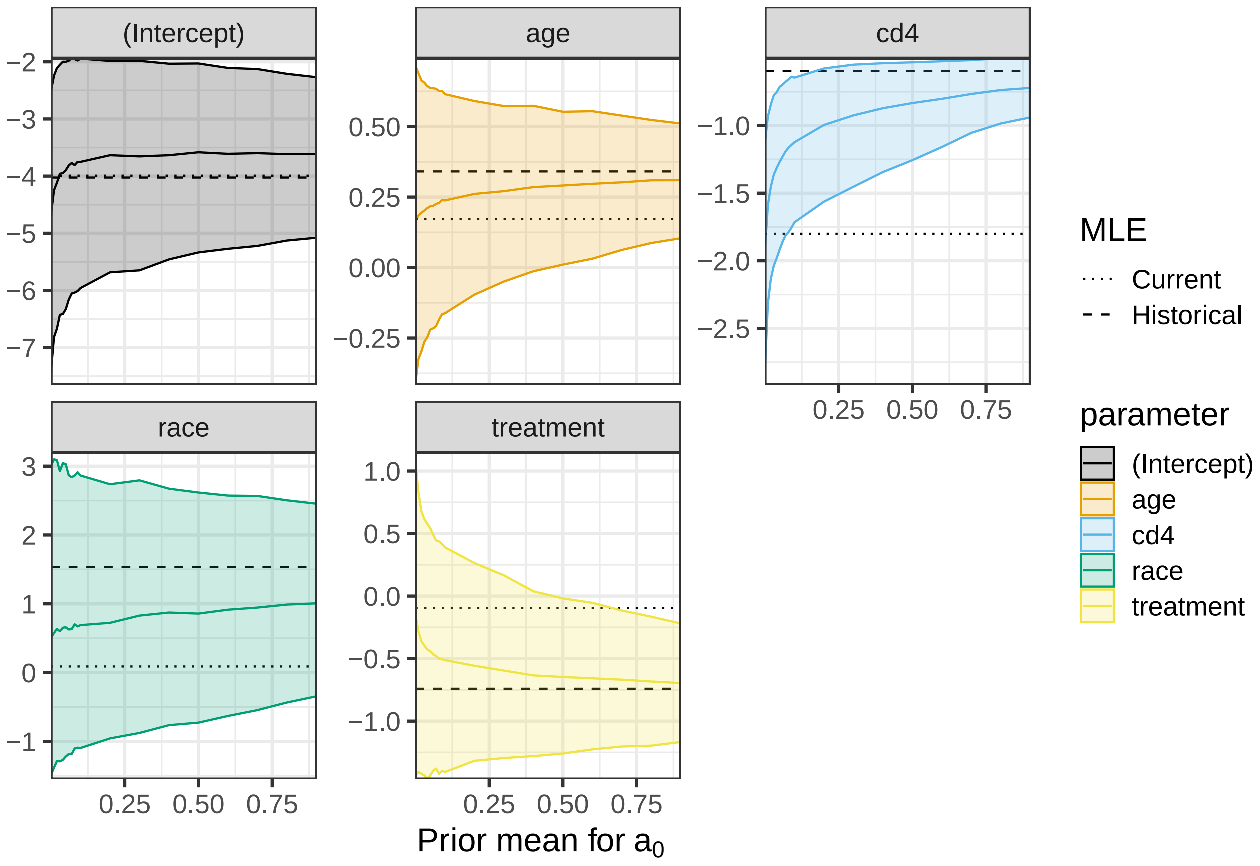
```r
npp_fit_and_report <- function(k){
  cat("Doing a0=", as[k], "\n")
  fit.npp <- glm.npp(
    formula,
    family,
    data.list = the.data,
    a0.lognc = a0.lognc,
    lognc = lognc,
    a0.shape1 =  hyperpars[[k]]$a, a0.shape2 = hyperpars[[k]]$b,
    parallel_chains = 4,
    iter_warmup = warmup,
    iter_sampling = samples,
    show_messages = FALSE,
    show_exceptions = FALSE,
    refresh = 0
  )
  return(fit.npp)
}

system.time(
  all.npp.fits <- lapply(seq_along(as), npp_fit_and_report)
)
#> Doing a0= 1e-06
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 1e-05
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 1e-04
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.001
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.01
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.02
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.03
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.04
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.05
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.06
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.07
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.08
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.09
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.1
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.2
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Warning: 20 of 10000 (0.0%) transitions ended with a divergence.
#> See https://mc-stan.org/misc/warnings for details.
#> Doing a0= 0.3
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Warning: 312 of 10000 (3.0%) transitions ended with a divergence.
#> See https://mc-stan.org/misc/warnings for details.
#> Doing a0= 0.4
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Warning: 1008 of 10000 (10.0%) transitions ended with a divergence.
#> See https://mc-stan.org/misc/warnings for details.
#> Doing a0= 0.5
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Warning: 1899 of 10000 (19.0%) transitions ended with a divergence.
#> See https://mc-stan.org/misc/warnings for details.
#> Doing a0= 0.6
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Warning: 2601 of 10000 (26.0%) transitions ended with a divergence.
#> See https://mc-stan.org/misc/warnings for details.
#> Doing a0= 0.7
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Warning: 3394 of 10000 (34.0%) transitions ended with a divergence.
#> See https://mc-stan.org/misc/warnings for details.
#> Doing a0= 0.8
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Warning: 4470 of 10000 (45.0%) transitions ended with a divergence.
#> See https://mc-stan.org/misc/warnings for details.
#> Doing a0= 0.9
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Warning: 6319 of 10000 (63.0%) transitions ended with a divergence.
#> See https://mc-stan.org/misc/warnings for details.
#>    user  system elapsed
#> 447.161   5.329 140.522

all.summaries.npp <- lapply(1:length(all.npp.fits),  function(i) {
  out <- tibble(get_summaries(fit = all.npp.fits[[i]], base.pars),
              name = paste0("a_0=", as[i]))
  return(out)
})

results.npp <- do.call(rbind, all.summaries.npp)
```

```r
results.npp$prior_mean_a0 <- as.numeric(gsub("a_0=", "", results.npp$name))
results.npp <- results.npp %>% rename(lwr = q5)
results.npp <- results.npp %>% rename(upr = q95)
results.npp <- results.npp %>% rename(parameter = variable)
```

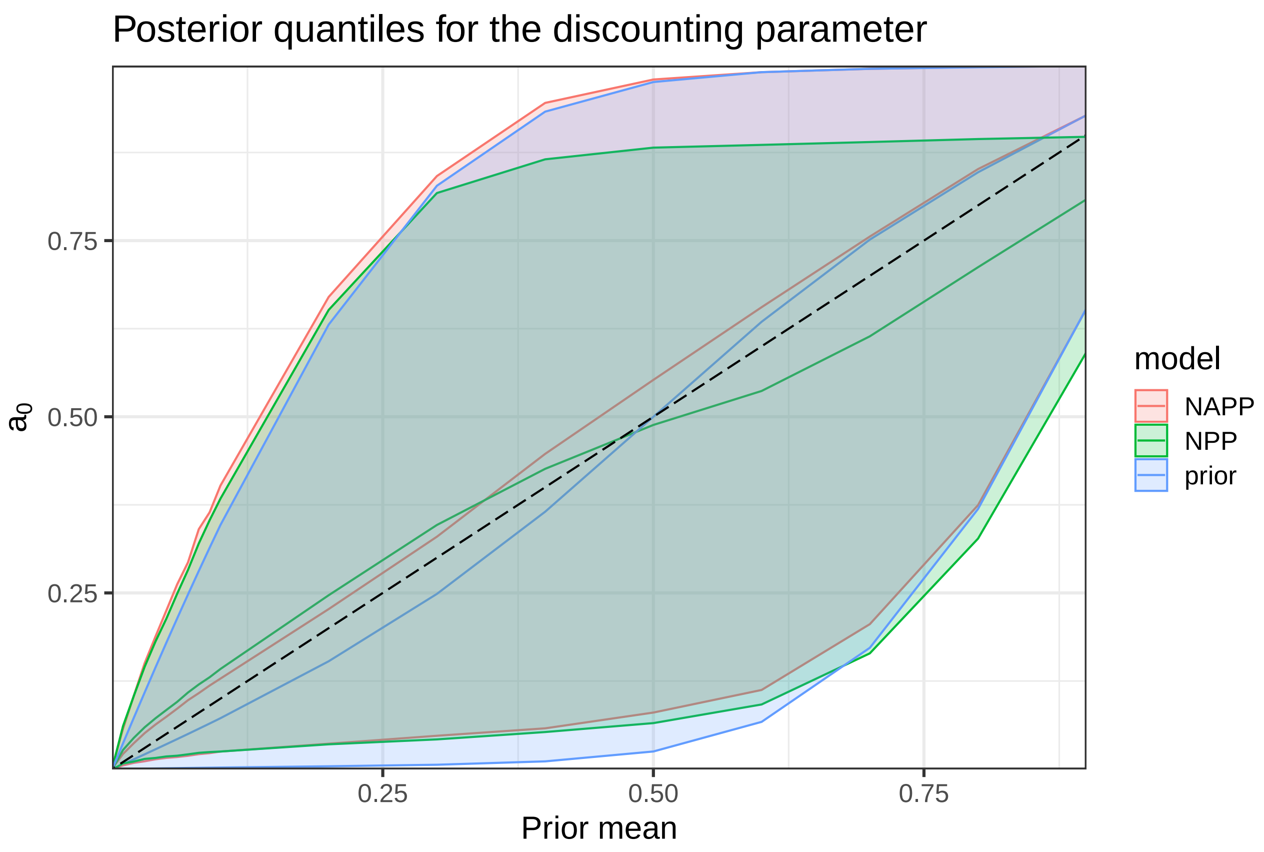


And now we will repeat the exercise with the normalised asymptotic power prior (NAPP).

```r
napp_fit_and_report <- function(k){
  cat("Doing a0=", as[k], "\n")
  fit.napp <- glm.napp(
    formula,
    family,
    data.list = the.data,
    a0.shape1 =  hyperpars[[k]]$a, a0.shape2 = hyperpars[[k]]$b,
    parallel_chains = 4,
    show_messages = FALSE,
    show_exceptions = FALSE,
    iter_warmup = warmup,
    iter_sampling = samples,
    refresh = 0
  )
  return(fit.napp)
}

system.time(
  all.napp.fits <- lapply(seq_along(as), napp_fit_and_report)
)
#> Doing a0= 1e-06
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 1e-05
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 1e-04
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.001
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.01
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.02
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.03
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.04
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.05
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.06
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.07
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.08
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.09
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.1
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.2
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.3
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.4
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.5
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.6
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.7
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.8
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#> Doing a0= 0.9
#> the first element in data.list is regarded as the current data
#> i Using cached Stan models
#>    user  system elapsed
#> 115.725   4.837  39.114

all.summaries.napp <- lapply(1:length(all.napp.fits),  function(i) {
  out <- tibble(get_summaries(fit = all.napp.fits[[i]], base.pars),
                name = paste0("a_0=", as[i]))
  return(out)
})

results.napp <- do.call(rbind, all.summaries.napp)
results.napp$prior_mean_a0 <- as.numeric(gsub("a_0=", "", results.napp$name))
results.napp <- results.napp %>% rename(lwr = q5)
results.napp <- results.napp %>% rename(upr = q95)
results.napp <- results.napp %>% rename(parameter = variable)
```



Finally, we will look at the posterior distribution for a_0:



from which we conclude that the posterior for a_0 does not change much relative to its prior.

References

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

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.

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  compiler  stats     graphics  grDevices utils     datasets
#> [8] methods   base
#>
#> other attached packages:
#>  [1] logPoolR_0.0.0.9000 mvtnorm_1.2-4       caTools_1.18.2
#>  [4] matrixStats_1.2.0   scales_1.3.0        forcats_0.5.2
#>  [7] stringr_1.5.1       dplyr_1.1.2         purrr_1.0.1
#> [10] readr_2.1.4         tidyr_1.3.0         tibble_3.2.1
#> [13] tidyverse_1.3.2     ggthemes_4.2.4      ggplot2_3.4.4
#> [16] posterior_1.5.0     hdbayes_0.0.0.9000
#>
#> loaded via a namespace (and not attached):
#>  [1] Brobdingnag_1.2-9    httr_1.4.4           jsonlite_1.8.7
#>  [4] modelr_0.1.10        assertthat_0.2.1     distributional_0.4.0
#>  [7] tensorA_0.36.2.1     googlesheets4_1.0.1  cellranger_1.1.0
#> [10] yaml_2.3.6           pillar_1.9.0         backports_1.4.1
#> [13] lattice_0.20-45      glue_1.7.0           digest_0.6.33
#> [16] checkmate_2.3.1      rvest_1.0.3          colorspace_2.1-0
#> [19] htmltools_0.5.4      Matrix_1.5-1         pkgconfig_2.0.3
#> [22] broom_1.0.2          haven_2.5.2          processx_3.8.3
#> [25] tzdb_0.4.0           timechange_0.2.0     googledrive_2.0.0
#> [28] generics_0.1.3       withr_2.5.2          enrichwith_0.3.1
#> [31] formula.tools_1.7.1  cli_3.6.2            crayon_1.5.2
#> [34] magrittr_2.0.3       readxl_1.4.2         mclust_6.0.1
#> [37] evaluate_0.23        ps_1.7.6             fs_1.6.3
#> [40] fansi_1.0.6          operator.tools_1.6.3 xml2_1.3.3
#> [43] tools_4.2.2          hms_1.1.3            gargle_1.2.1
#> [46] lifecycle_1.0.4      instantiate_0.2.1    munsell_0.5.0
#> [49] reprex_2.0.2         callr_3.7.3          rlang_1.1.3
#> [52] grid_4.2.2           rstudioapi_0.14      bitops_1.0-7
#> [55] rmarkdown_2.19       gtable_0.3.4         abind_1.4-5
#> [58] DBI_1.1.3            R6_2.5.1             lubridate_1.9.2
#> [61] knitr_1.41           bridgesampling_1.1-2 fastmap_1.1.1
#> [64] utf8_1.2.4           stringi_1.8.3        vctrs_0.6.5
#> [67] dbplyr_2.3.0         tidyselect_1.2.0     xfun_0.39
#> [70] coda_0.19-4.1
```