comparison_Bernoulli.r

max

2021-03-13

```r
library(npowerPrioR)
```

```
## Loading required package: parallel

## Loading required package: mgcv

## Loading required package: nlme

## This is mgcv 1.8-33. For overview type 'help("mgcv-package")'.

## Loading required package: rstan

## Loading required package: StanHeaders

## Loading required package: ggplot2

## rstan (Version 2.21.2, GitRev: 2e1f913d3ca3)

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

## Loading required package: bridgesampling
```

```r
source("../Bernoulli/data_Bernoulli_scenario_4.r")

bb.data <- list(
  N0 = N_0,
  y0 = y_0,
  c = cc,
  d = dd,
  a_0 = NA
)

get_l_a0_bernoulli <- function(y0, n0, cc, dd, a_0){
  ans <- lbeta(a_0 * y0 + cc, a_0 *(n0 -y0) + dd)-lbeta(cc, dd)
  return(ans)
}

l_a0 <- function(x) {
  get_l_a0_bernoulli(
    y0 = bb.data$y0,
    n0 = bb.data$N0,
    cc = bb.data$c,
    dd = bb.data$d,
    a_0 = x
  )
}
########

maxA <- 1
prior <- stan_model("../Bernoulli/stan/simple_Bernoulli_prior.stan")


# direct method
J <- 20
epsilon <- 0.05

adaptive.time <- system.time(
  adaptive.ca0.estimates <- build_grid(compiled.model.prior = prior, eps = epsilon,
                                       M = maxA, J = J, v1 = 10, v2 = 10,
                                       stan.list = bb.data, pars = c("theta"))
)
```

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## Warning: effective sample size cannot be calculated, has been replaced by number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

## Warning: effective sample size cannot be calculated, has been replaced by number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

## Warning: effective sample size cannot be calculated, has been replaced by number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

## Warning: effective sample size cannot be calculated, has been replaced by number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

## Warning: effective sample size cannot be calculated, has been replaced by number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

## Warning: effective sample size cannot be calculated, has been replaced by number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

## Warning: effective sample size cannot be calculated, has been replaced by number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

## Warning: effective sample size cannot be calculated, has been replaced by number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

## Warning: effective sample size cannot be calculated, has been replaced by number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

## Warning: effective sample size cannot be calculated, has been replaced by number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

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

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

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

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

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

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

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

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

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

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

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

```
# VR2018
Delta.a <- 0.01
a0s.vr2018 <- seq(0, maxA, by = Delta.a)

vr2018.time <- system.time(
  vr2018.estimates <-  create_lc_df_derivOnly(a0_grid = a0s.vr2018,
                               compiled.model.prior = prior,
                               stan.list = bb.data, pars = c("theta") )
)

write.csv(vr2018.estimates$result,
          file = "Gaussian_VR2018.csv", row.names = FALSE)
adaptive.time
```

```
##    user  system elapsed
##  12.766   0.092  12.970
```

```
vr2018.time
```

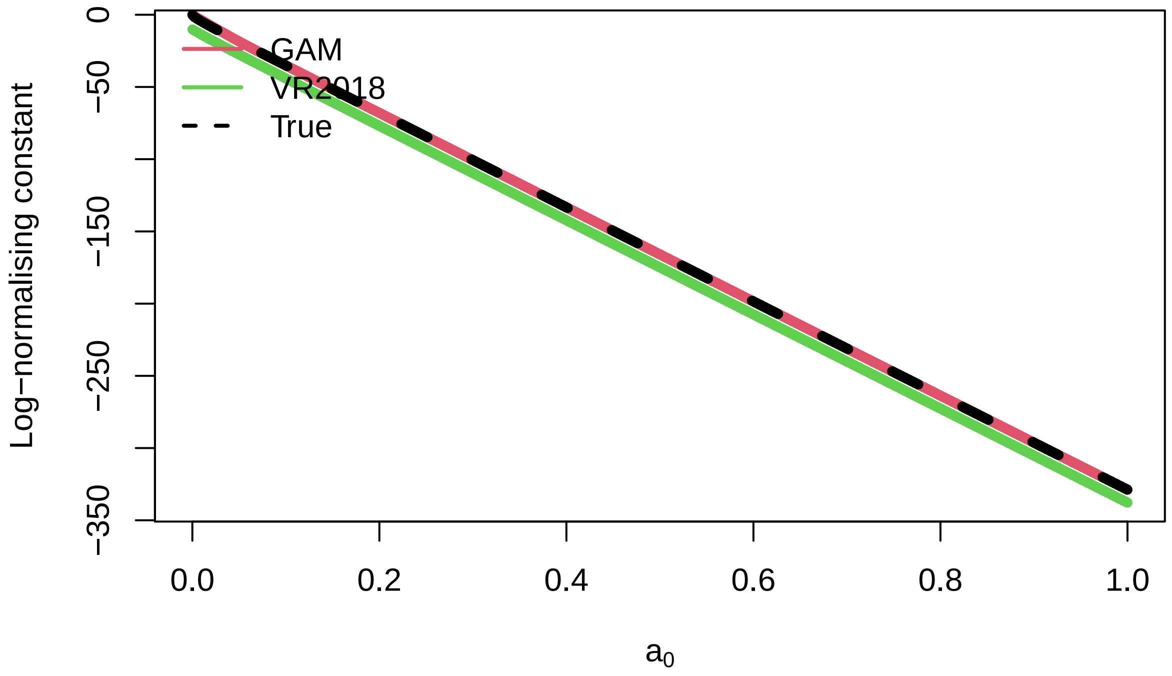
```
##    user  system elapsed
##  10.218   0.028  10.416
```

```
###
## Now the approximations
adapt.gam <-  mgcv::gam(lc_a0 ~ s(a0, k = J), data = adaptive.ca0.estimates$result)
vr2018.estimates$result$la0_est <- cumsum(vr2018.estimates$result$deriv_lc) * Delta.a



## Finally, comparisons
```

```
K <- 20000
pred.a0s <- seq(0, maxA, length.out = K)

true.la0s <- l_a0(pred.a0s)

adaptive.preds <- predict(adapt.gam, newdata = data.frame(a0 = pred.a0s))

vr2018.preds <- approx(x = vr2018.estimates$result$a0,
                          y =  vr2018.estimates$result$la0_est,
                          xout =  pred.a0s)

plot(vr2018.preds, type = "l", lwd = 5,
     col = 3,
     xlab = expression(a[0]), ylab = "Log-normalising constant")
lines(pred.a0s, adaptive.preds, col = 2, lwd = 5)
lines(pred.a0s, true.la0s, lwd = 5, lty = 2, add = TRUE)
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "add" is not a graphical
## parameter
```

```
legend(x = "topleft", legend =  c("GAM", "VR2018", "True"),
       col = c(2, 3, 1), lwd  = 2, lty = c(1, 1, 2), bty = 'n')
```



```
preds.list <- list(
  adaptive = adaptive.preds,
  VR2018 = vr2018.preds$y
)

ntrue.la0s <- true.la0s

lapply(preds.list, function(pred) sqrt(mean( ( pred- ntrue.la0s)^2 )) )
```

```
## $adaptive
## [1] 0.1348463
##
## $VR2018
## [1] 8.966377
```

```
lapply(preds.list, function(pred) mean( abs( pred- ntrue.la0s) ))
```

```
## $adaptive
## [1] 0.03383187
##
## $VR2018
## [1] 8.966237
```