comparison_Poisson.r

max

2021-03-13

```
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
```

```
source("../Poisson/data_Poisson.r")

po.data <- list(
  N0 = N_0,
  y0 = y_0,
  alpha0 = alpha_0,
  beta0 = beta_0,
  a_0 = NA
)
###
get_l_a0_poisson <- function(y0, n0, alpha0, beta0, a_0){
  logPprime <- sum(lfactorial(y0))
  S <- sum(y0)
  ans <- -a_0 * logPprime + lgamma(a_0 * S + alpha0) - (a_0 * S + alpha0)* log(a_0 *n0 + beta0) + (alpha
  return(ans)
}
#############
l_a0 <- function(x) {
  get_l_a0_poisson(
    y0 = po.data$y0,
    n0 = po.data$N0,
    alpha0 = po.data$alpha0,
    beta0 = po.data$beta0,
```

```
    a_0 = x
  )
}
l_a0 <- Vectorize(l_a0)

########

maxA <- 1
prior <- stan_model("../Poisson/stan/simple_Poisson_prior.stan")
```

```
## Trying to compile a simple C file
```

```
## Running /usr/local/lib/R/bin/R CMD SHLIB foo.c
## gcc -I"/usr/local/lib/R/include" -DNDEBUG   -I"/home/max/R/x86_64-pc-linux-gnu-library/4.0/Rcpp/inclu
## In file included from /home/max/R/x86_64-pc-linux-gnu-library/4.0/RcppEigen/include/Eigen/Core:88,
##                  from /home/max/R/x86_64-pc-linux-gnu-library/4.0/RcppEigen/include/Eigen/Dense:1,
##                  from /home/max/R/x86_64-pc-linux-gnu-library/4.0/StanHeaders/include/stan/math/prim/
##                  from <command-line>:
## /home/max/R/x86_64-pc-linux-gnu-library/4.0/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: err
##   613 | namespace Eigen {
##       | ^~~~~~~~~
## /home/max/R/x86_64-pc-linux-gnu-library/4.0/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:17: er
##   613 | namespace Eigen {
##       |                 ^
## In file included from /home/max/R/x86_64-pc-linux-gnu-library/4.0/RcppEigen/include/Eigen/Dense:1,
##                  from /home/max/R/x86_64-pc-linux-gnu-library/4.0/StanHeaders/include/stan/math/prim/
##                  from <command-line>:
## /home/max/R/x86_64-pc-linux-gnu-library/4.0/RcppEigen/include/Eigen/Core:96:10: fatal error: complex
##    96 | #include <complex>
##       |          ^~~~~~~~~
## compilation terminated.
## make: *** [/usr/local/lib/R/etc/Makeconf:172: foo.o] Error 1
```

```
# 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 = po.data, pars = c("lambda"))
)
```

```
## 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.
```

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

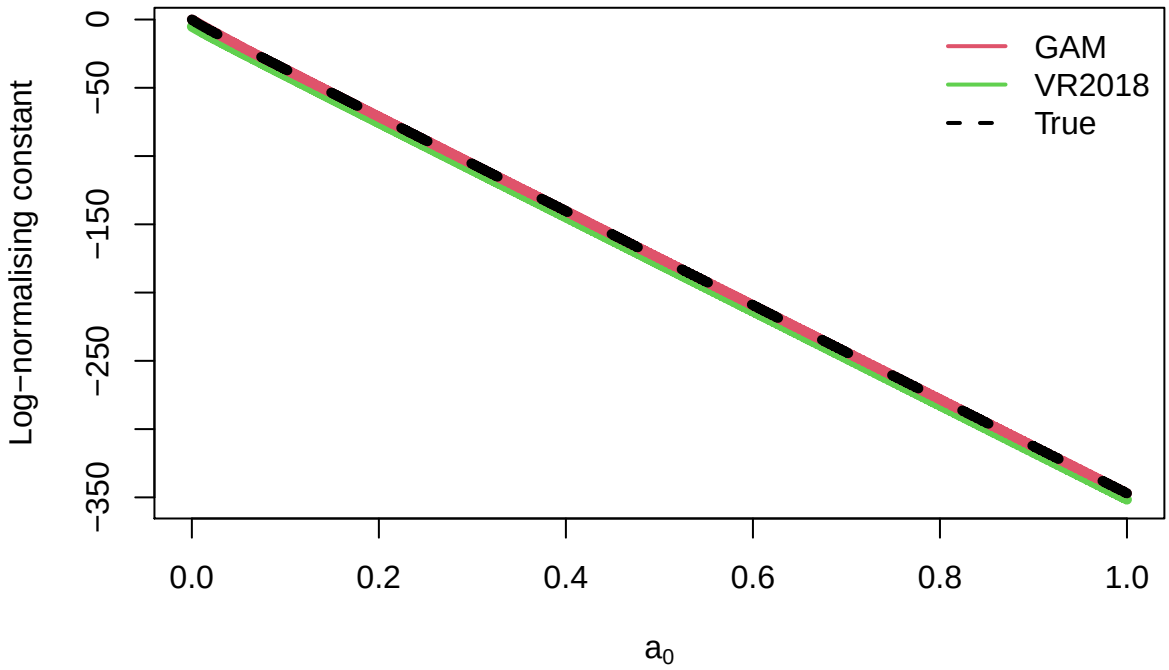
```
vr2018.time <- system.time(
  vr2018.estimates <-  create_lc_df_derivOnly(a0_grid = a0s.vr2018,
                               compiled.model.prior = prior,
                               stan.list = po.data, pars = c("lambda") )
)

write.csv(vr2018.estimates$result,
          file = "Poisson_VR2018.csv", row.names = FALSE)
# Run times
adaptive.time
```

```
##    user  system elapsed
##  21.229   0.059  21.426
```

```
vr2018.time
```

```
##    user  system elapsed
##  40.565   0.020  41.052
```

```
###
## 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)
legend(x = "topright", 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] 0.07360804
##
## $VR2018
## [1] 4.614112
```

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

```
## $adaptive
## [1] 0.01941045
##
## $VR2018
## [1] 4.613954
```