experiments_Yao.r

max

2020-04-07

```r
## Scenario 1: data is Normal(2, 10)
## Scenario 2: data is Normal(5/2, 10)
## Scenario 3: data is Cauchy(1, 1)
## Priors are Normal(k, 1) for k = 1, ...,K  for all scenarios

source("pooling_aux.r")
```

```
## Loading required package: ggplot2
```

```r
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
## rstan (Version 2.19.3, 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)
```

```r
rstan_options(auto_write = TRUE)

# This is the Stan code
writeLines(readLines("stan/optimise_normal_pool.stan"))
```

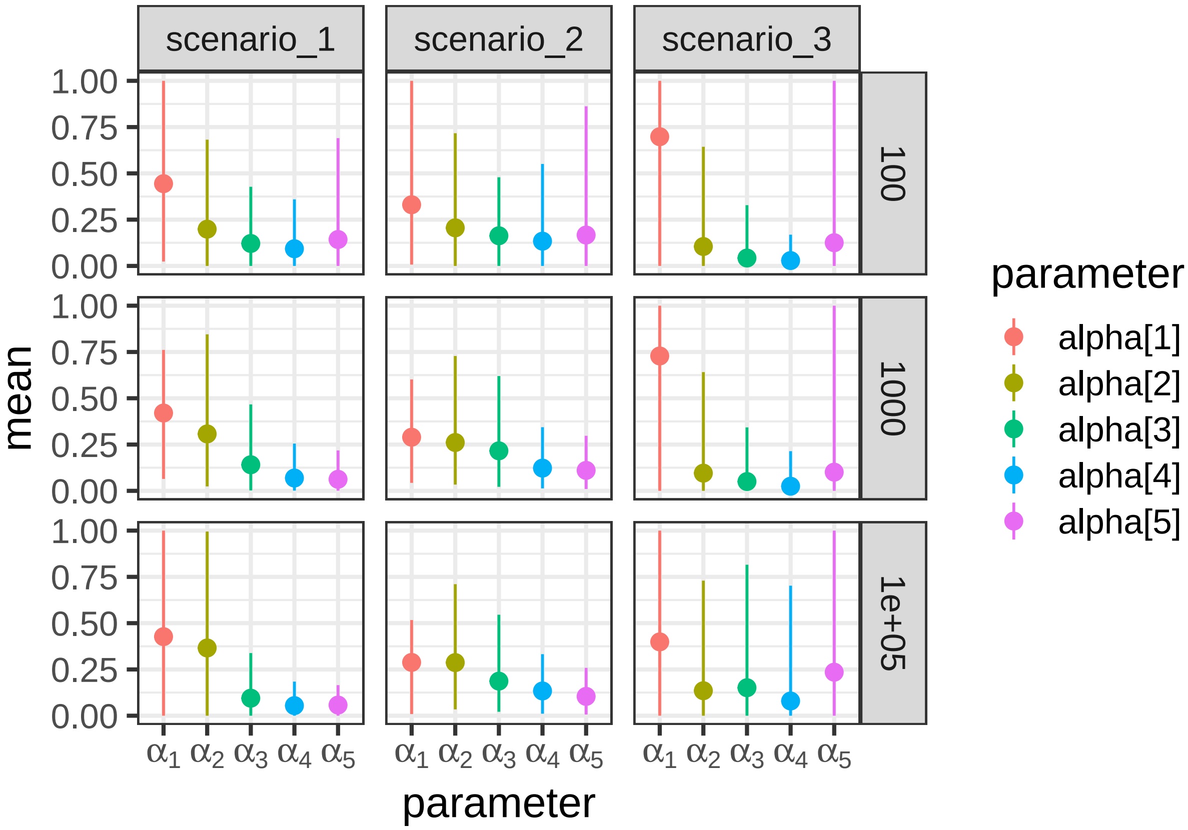
```
## data {
##   int<lower=0> N; // number of observations
##   real Y[N]; // successes
##   int<lower=1> K; // number of priors to combine
##   vector[K] mu; // parameters of the K priors
##   vector[K] sigma_sq;
## }
## parameters {
##   simplex[K] alpha;
## }
## transformed parameters{
##   real<lower=0> vstar;
##   real mustar;
##   vector[K]  w;
##   for(k in 1:K) w[k] =  alpha[k] / sigma_sq[k];
##   vstar = 1/sum(w);// ommitting <lower=0>
##   mustar =  sum(w .* mu) * vstar;
## }
## model {
##   Y ~ normal(mustar, vstar);
```

```
## }
```

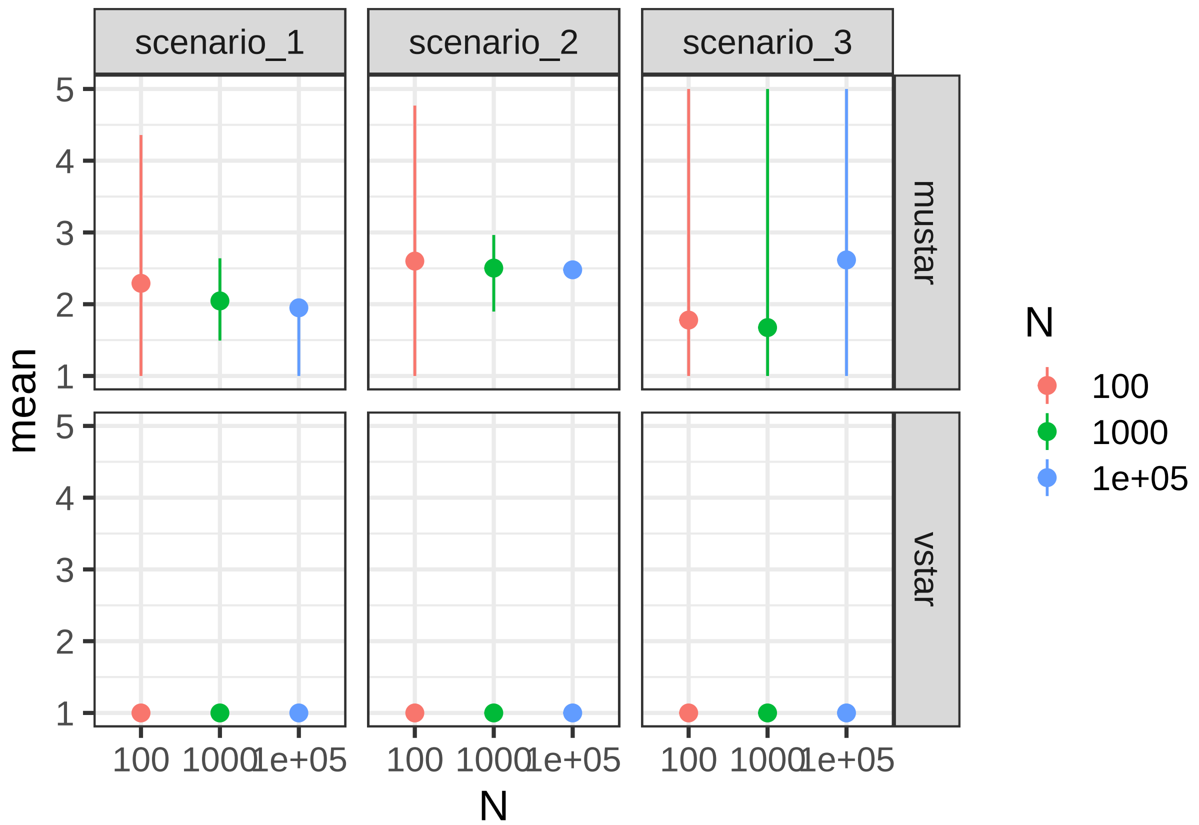
```
normal.model <- stan_model("stan/optimise_normal_pool.stan")

###############
## Auxiliary Functions
estimate_once <- function(parms, data){
  N <- length(data)
  K <- length(parms$m)
  if(length(parms$v) != K) stop("mu and v need to be the same size!")
  data.list <- list(
    N = N,
    Y = data,
    K = K,
    mu = parms$m,
    sigma_sq = parms$v
  )
  get_estimate <- function(){
    opt <- optimizing(normal.model, data = data.list)
    ans <- opt$par[-grep(c("w"), names(opt$par)) ]
    return(ans)
  }
  result <- tryCatch(get_estimate(), ## Had to add this because vstar would be Nan sometimes. Might be
                error = function(err){
                  ans <- rep(NA, K + 2)
                  names(ans) <-  c("alpha[1]", "alpha[2]", "alpha[3]", "alpha[4]", "alpha[5]", "vstar
                  return(ans)
                } )
  return(result)
}

generate_data <- function(n, scenario){
  data <- switch (scenario,
                  "1" = rnorm(n = n, mean = 2, sd = 10),
                  "2" = rnorm(n = n, mean = 5/2, sd = 10),
                  "3" = rcauchy(n = n, location = 1, scale = 1)
  )
  return(data)
}

rowLowers <- function(x, na.rm = FALSE) apply(x, 1, quantile, probs =  .025, na.rm = na.rm)

rowUppers <- function(x, na.rm = FALSE) apply(x, 1, quantile, probs =  .975, na.rm = na.rm)

get_summaries <- function(x){
  ans <- data.frame(
    lwr = rowLowers(x, na.rm = TRUE),
    mean = rowMeans(x, na.rm = TRUE),
    upr = rowUppers(x, na.rm = TRUE),
    parameter =  c("alpha[1]", "alpha[2]", "alpha[3]", "alpha[4]", "alpha[5]", "vstar", "mustar")
  )
  return(ans)
}
##
```

```r
pick_samples <- function(result, k = 10){
  result <- result[1:5, ]
  M <- ncol(result)
  if(k > M) stop("Want more samples than there are columns!")
  pos <- which(!is.na(result[1, ]))
  samp.pos <- sample(pos, k, replace = FALSE)
  ans <- data.frame(result[, samp.pos], expert = paste("f_", 1:5, sep = ""))
  return(ans)
}
###############
parameters <- list(
  m = c(1, 2, 3, 4, 5),
  v = c(1, 1, 1, 1, 1)
)
Ns <- c(100, 1000, 1E5)
M <- 100

## Experiment 1
data.1 <- lapply(Ns, function(n) matrix(generate_data(n = M * n, scenario = "1"), nrow = n, ncol = M) )
estimates.1 <- lapply(data.1, function(y) apply(y, 2, function(x) estimate_once(parms = parameters, data

## Experiment 2
data.2 <- lapply(Ns, function(n) matrix(generate_data(n = M * n, scenario = "2"), nrow = n, ncol = M) )
estimates.2 <- lapply(data.2, function(y) apply(y, 2, function(x) estimate_once(parms = parameters, data

## Experiment 3
data.3 <- lapply(Ns, function(n) matrix(generate_data(n = M * n, scenario = "3"), nrow = n, ncol = M) )
estimates.3 <- lapply(data.3, function(y) apply(y, 2, function(x) estimate_once(parms = parameters, data


raw.summaries.1 <- lapply(estimates.1, get_summaries)
raw.summaries.2 <- lapply(estimates.2, get_summaries)
raw.summaries.3 <- lapply(estimates.3, get_summaries)

for(i in 1:length(Ns)){
  raw.summaries.1[[i]]$N <- Ns[i]
  raw.summaries.1[[i]]$scenario <- 'scenario_1'
  raw.summaries.2[[i]]$N <- Ns[i]
  raw.summaries.2[[i]]$scenario <- 'scenario_2'
  raw.summaries.3[[i]]$N <- Ns[i]
  raw.summaries.3[[i]]$scenario <- 'scenario_3'
}

results.dt <- rbind(do.call(rbind, raw.summaries.1),
                    do.call(rbind, raw.summaries.2),
                    do.call(rbind, raw.summaries.3)
)
row.names(results.dt) <- NULL
results.dt$N <- as.factor(results.dt$N)

## First, let's look at the 'estimated' alphas and their distribution.
# Here I am showing mean and 95% quantiles over M replicates
```

```r
only.alphas <- subset(results.dt, !(parameter %in% c("mustar", "vstar")) )

p0 <- ggplot(only.alphas,
             aes(x = parameter, y = mean, ymin = lwr, ymax = upr, colour = parameter, fill = parameter)
  geom_pointrange() +
  scale_x_discrete(labels = parse(text = levels(only.alphas$parameter))) +
  facet_grid(N~scenario) +
  theme_bw(base_size = 16)

p0
```




```r
# Now let's look at mu* and v*
# since all v_i = v = 1, v* is always fixed. Showing here for consistency only

only.pars <- subset(results.dt, parameter %in% c("mustar", "vstar") )

p1 <- ggplot(only.pars,
             aes(x = N, y = mean, ymin = lwr, ymax = upr, colour = N, fill = N)) +
  geom_pointrange() +
  facet_grid(parameter~scenario) +
  theme_bw(base_size = 16)

p1
```




```
subsampled.1 <- lapply(estimates.1, pick_samples, k = 10)
subsampled.2 <- lapply(estimates.2, pick_samples, k = 10)
subsampled.3 <- lapply(estimates.3, pick_samples, k = 10)

## Now let's do a bit of housekeeping/cleaning to get data in 'plottable' format
library(reshape2)

subsamp.dt.1 <- melt(data = subsampled.1, id.vars = "expert",  value.name = "alpha")
names(subsamp.dt.1) <- c("expert", "replicate", "alpha", "N")
subsamp.dt.1$replicate <- as.factor(gsub("X", "", subsamp.dt.1$replicate))
subsamp.dt.1$N <- as.factor(Ns[subsamp.dt.1$N])
subsamp.dt.1$scenario <- "scenario_1"

subsamp.dt.2 <- melt(data = subsampled.2, id.vars = "expert",  value.name = "alpha")
names(subsamp.dt.2) <- c("expert", "replicate", "alpha", "N")
subsamp.dt.2$replicate <- as.factor(gsub("X", "", subsamp.dt.2$replicate))
subsamp.dt.2$N <- as.factor(Ns[subsamp.dt.2$N])
subsamp.dt.2$scenario <- "scenario_2"

subsamp.dt.3 <- melt(data = subsampled.3, id.vars = "expert",  value.name = "alpha")
names(subsamp.dt.3) <- c("expert", "replicate", "alpha", "N")
subsamp.dt.3$replicate <- as.factor(gsub("X", "", subsamp.dt.3$replicate))
subsamp.dt.3$N <- as.factor(Ns[subsamp.dt.3$N])
subsamp.dt.3$scenario <- "scenario_3"
#
all.subsamp.alpha.dt <- rbind(subsamp.dt.1, subsamp.dt.2, subsamp.dt.3)
```

```r
# Now we have a radar plot showing 10 sample results for each scenario and sample size
radar_alphas <- ggplot(data = all.subsamp.alpha.dt,
                       aes(x = expert, y = alpha, group = replicate, colour = replicate, fill = replica
  geom_point() +
  geom_polygon(alpha = 0.4) +
  facet_grid(N ~ scenario) +
  theme_bw(base_size = 16) +
  scale_y_continuous(expand = c(0, 0), limits = c(0, 1),
                     breaks = number_ticks(10)) +
  coord_radar() +
  theme(axis.title.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.text.x = element_text(face = "bold"),
        axis.title.y = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank()
  )
radar_alphas
```