Savchuk_beta_example.r

luiz

2019-06-25

```
# Leo Bastos & Luiz Max Carvalho (2019)
# This example was taken from Savchuk and Martz (1994)

source("pooling_aux.r")
```

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

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
a0 <- 18.1 ; b0 <- .995
a1 <- 3.44 ; b1 <- .860
a2 <- 8.32 ; b2 <- .924
a3 <- 1.98 ; b3 <- .848

av <- c(a0, a1, a2, a3)
bv <- c(b0, b1, b2, b3)
K <- length(av)

# Individual entropies
entropies <- rep(NA, K)
for(k in 1:K) entropies[k] <- entropy_beta(av[k], bv[k])

entropies
```

```
## [1] -1.9557768 -0.6301164 -1.3069373 -0.2668240
```

```
## Entropy surface (for a future dominance analysis)

library(fields)
```

```
## Loading required package: spam
```

```
## Loading required package: dotCall64
```

```
## Loading required package: grid
```

```
## Spam version 2.2-2 (2019-03-07) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
```

```
##
## Attaching package: 'spam'
```

```
## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve
```

```
## Loading required package: maps
```

```
## See https://github.com/NCAR/Fields for
##  an extensive vignette, other supplements and source code
```

```
ES <- entropy_surface_beta(av, bv)
export <- TRUE
if(export){
  pdf("../plots/entropy_surface_failureProbExample.pdf")
}
image.plot(ES$as, ES$bs, ES$M,
           xlab = expression(a), ylab = expression(b), horizontal = TRUE,
           cex.lab = 1.5, cex.axis = 1.5, axis.args = list(font = 2),
           legend.cex = 1.5,
           legend.lab = expression(H[pi]), main = "Entropy Beta distribution", font = 2)
if(export){
  dev.off()
}
```

```
## pdf
##   2
```

```
## Observed data
y <- 9
n <- 10

## Marginal likelihoods

marginal.likelihoods <- rep(NA, K)
for (k in 1:K){ marginal.likelihoods[k] <- ml_beta(yi = y, ni = n, a = av[k], b = bv[k]) }

marginal.likelihoods
```

```
## [1] 0.2370065 0.2114937 0.2566844 0.1636389
```

```
round( normalised.marginal.likelihoods<- marginal.likelihoods/sum(marginal.likelihoods), 2 )
```

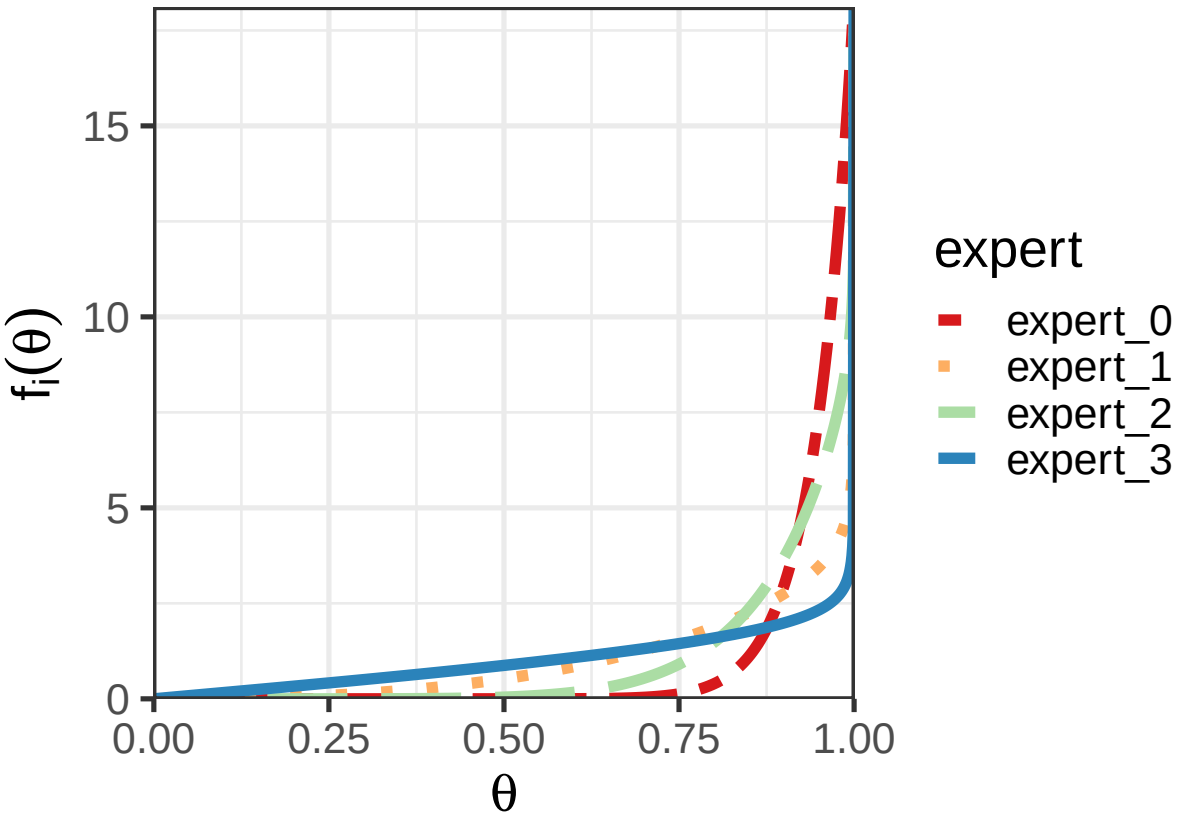
```
## [1] 0.27 0.24 0.30 0.19
```

```
MLS <- marginal_likelihood_surface_beta(y = y, n = n, av = av, bv = bv)
export <- TRUE
if(export){
  pdf("../plots/marginalLikelihood_surface_failureProbExample.pdf")
}
image.plot(MLS$as, MLS$bs, MLS$M,
           xlab = expression(a), ylab = expression(b), horizontal = TRUE,
           cex.lab = 1.5, cex.axis = 1.5, axis.args = list(font = 2),
           legend.cex = 1.5,
           legend.lab = expression(l(y, n)), main = "Marginal likelihood Beta distribution", font = 2)
if(export){
  dev.off()
}
```

```
## pdf
##   2
```

```
#############
PaperBeta.tbl <- data.frame(mean.prior = rep(NA, 6), lower.prior = NA,
```

```
                              upper.prior = NA, mean.post = NA, lower.post = NA,
                              upper.post = NA)
rownames(PaperBeta.tbl) <- c("equal_weights", "maximum_entropy", "minimum_KL",
                              "hierarchical_Dirichlet", "hierarchical_LogisticNormal", "Rufo_2012")

AlphasBeta.tbl <- data.frame(matrix(NA, nrow = 5, ncol = length(av)))
rownames(AlphasBeta.tbl) <- c("maximum_entropy", "minimum_KL",
                               "hierarchical_Dirichlet", "hierarchical_LogisticNormal", "Rufo_2012")
colnames(AlphasBeta.tbl) <- paste("alpha_", 0:(K-1), sep = "")


library(ggplot2)

theta.grid <- seq(0, 1, length.out = 1000)
expert.densities <- vector(K, mode = "list")
for(k in 1:K){
  expert.densities[[k]] <- data.frame(theta = theta.grid,
                                      dens = dbeta(theta.grid, shape1 = av[k], shape2 = bv[k]),
                                      expert = paste("expert_", k-1, sep = ""))

}
expert.densities.df <- do.call(rbind, expert.densities)

expert_priors <- ggplot(expert.densities.df, aes(x = theta, y = dens,
                                                 linetype = expert, colour = expert)) +
  geom_line(size = 2) +
  scale_linetype_manual(values = c("twodash", "dotted", "longdash", "solid"))+
  scale_colour_brewer(palette = "Spectral") +
  scale_x_continuous(expression(theta), expand = c(0, 0)) +
  scale_y_continuous(expression(f[i](theta)), expand = c(0, 0)) +
  theme_bw(base_size = 20)

expert_priors
```




```
ggsave(expert_priors, filename = "../plots/expert_densities_Savchuk.pdf")
```

```
## Saving 6.5 x 4.5 in image
```

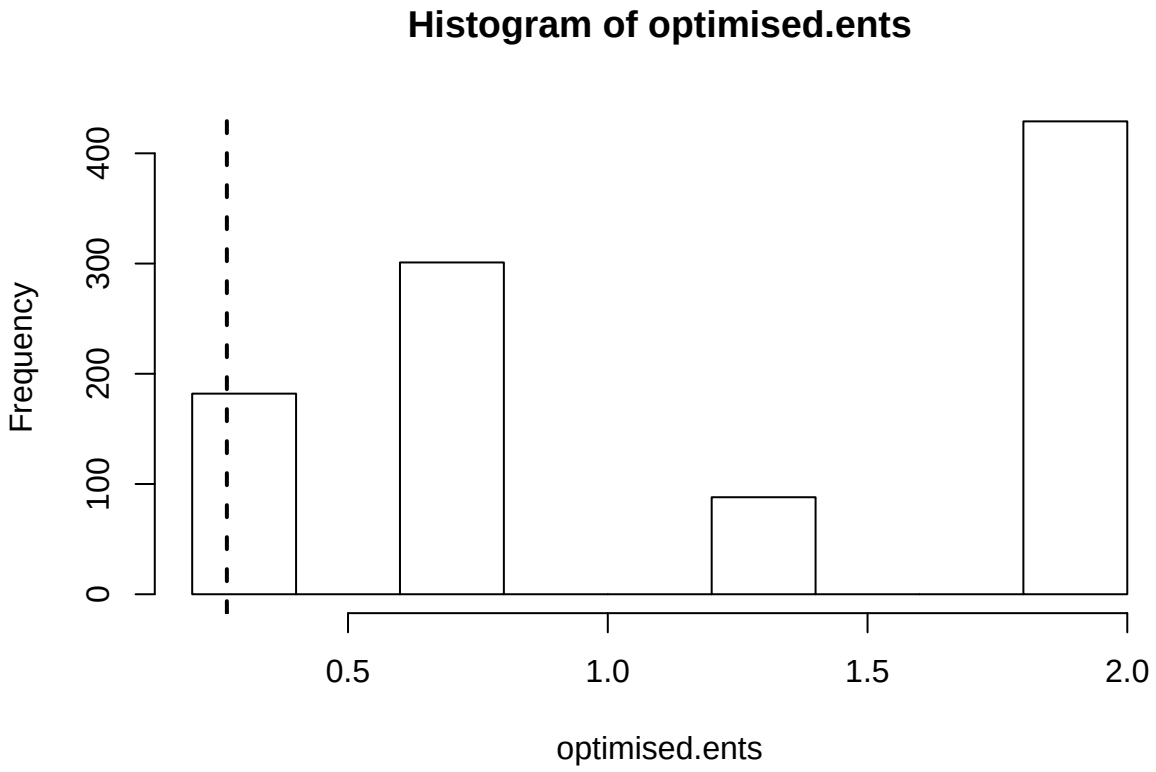
```
###### Equal weights

alphaEqual <- rep(1/K, K)

ab.Equal.star <- pool_par(alphaEqual, av, bv)
# Prior
(PaperBeta.tbl[1, 1:3] <- stat_beta(ab.Equal.star))
```

```
## [1] 0.8977359 0.6435035 0.9979079
```

```
# Posterior
(PaperBeta.tbl[1, 4:6] <- stat_beta(ab.Equal.star + c(y, n - y)))
```

```
## [1] 0.8989360 0.7327552 0.9877894
```

```
####### Maximum entropy

N <- 1000 ## could increase to, say, 10000 in order to make sure, but it's fine
ent.many.startingPoints <- matrix(rnorm(n = (K-1)*N, mean = 0, sd = 100), ncol = K-1, nrow = N)
many.ents <- lapply(1:N, function(i) {
  optim(ent.many.startingPoints[i, ], optentbeta_inv, ap = av, bp = bv)
})
optimised.ents <- unlist(lapply(many.ents, function(x) x$value))

hist(optimised.ents)
abline(v = optimised.ents[which.min(optimised.ents)], lty = 2, lwd = 2)
```




```
alphaMaxEnt.opt <- alpha_01(many.ents[[which.min(optimised.ents)]]$par)

round(alphaMaxEnt.opt, 2)
```

```
## [1] 0 0 0 1
```

```
( AlphasBeta.tbl[1, ] <- alphaMaxEnt.opt )
```

```
## [1] 2.798351e-18 5.791174e-16 4.501265e-44 1.000000e+00
```

```
ab.MaxEnt.star <- pool_par(alphaMaxEnt.opt, av, bv)

# Prior
(PaperBeta.tbl[2, 1:3] <- stat_beta(ab.MaxEnt.star))
```

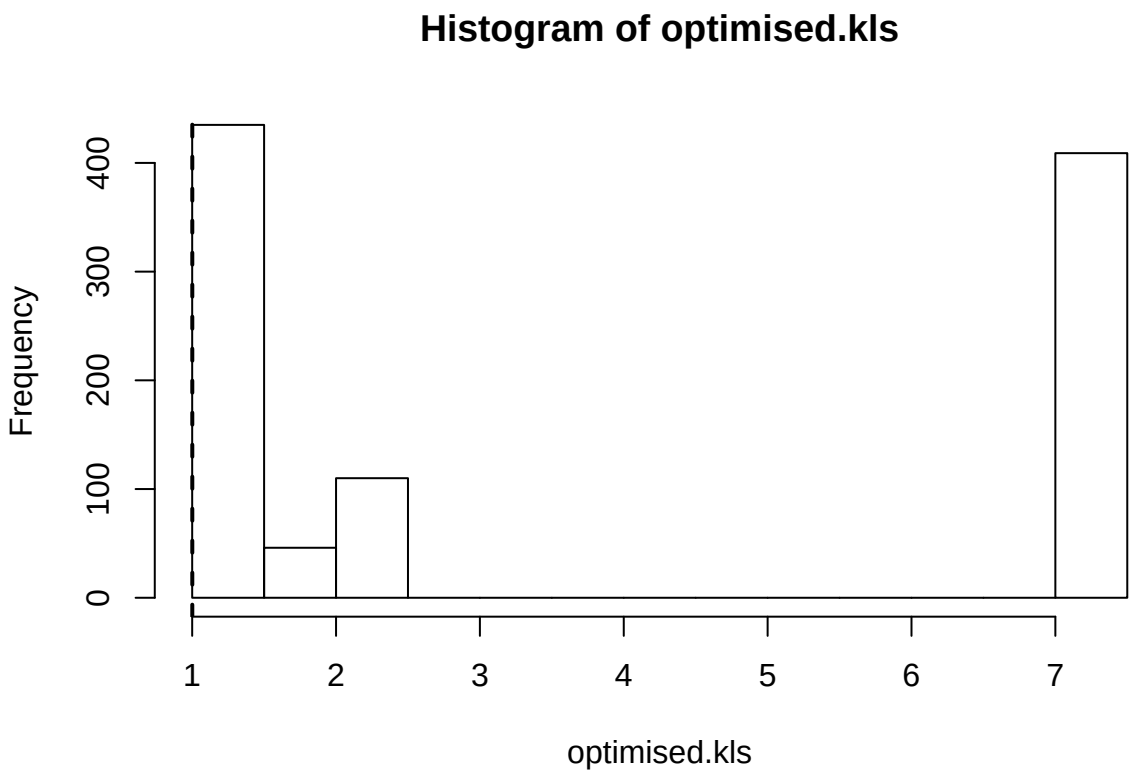
```
## [1] 0.7001414 0.1737705 0.9936577
```

```
# Posterior
(PaperBeta.tbl[2, 4:6] <- stat_beta(ab.MaxEnt.star +  c(y, n - y)))
```

```
## [1] 0.8559401 0.6273426 0.9828843
```

```
####### Minimum KL

N <- 1000 ## could increase to, say, 10000 in order to make sure, but it's fine
kl.many.startingPoints <- matrix(rnorm(n = (K-1)*N, mean = 0, sd = 100), ncol = K-1, nrow = N)
many.kls <- lapply(1:N, function(i) {
  optim(kl.many.startingPoints[i, ], optklbeta_inv, ap = av, bp = bv, type = "fp")
})
optimised.kls <- unlist(lapply(many.kls, function(x) x$value))

hist(optimised.kls)
```

```
abline(v = optimised.kls[which.min(optimised.kls)], lty = 2, lwd = 2)
```




```
alphaKL.opt <- alpha_01(many.kls[[which.min(optimised.kls)]]$par)

(AlphasBeta.tbl[2, ] <- alphaKL.opt)
```

```
## [1] 0.03987016 0.96012984 0.00000000 0.00000000
```

```
ab.KL.star <- pool_par(alphaKL.opt, av, bv)

# Prior
(PaperBeta.tbl[3, 1:3] <- stat_beta(ab.KL.star))
```

```
## [1] 0.8230258 0.4242831 0.9966280
```

```
# Posterior
(PaperBeta.tbl[3, 4:6] <- stat_beta(ab.KL.star + c(y, n-y)))
```

```
## [1] 0.8747214 0.6722137 0.9851126
```

```
####### Hierarchical priors
require("LearnBayes")
```

```
## Loading required package: LearnBayes
```

```
M <- 100000
X <- c(1, 1, 1, 1)/10
alpha.MC.dirichlet <- rdirichlet(M, X)
alpha.MC.logisticNormal <- rlogisticnorm(N = M,
                                m = digamma(X)-digamma(X[K]),
                                Sigma = constructSigma(X))
```

```r
apply(alpha.MC.dirichlet, 2, mean)
```

```
## [1] 0.2490319 0.2501797 0.2509338 0.2498546
```

```r
apply(alpha.MC.logisticNormal, 2, mean)
```

```
## [1] 0.2504521 0.2500445 0.2501548 0.2493486
```

```r
apply(alpha.MC.dirichlet, 2, sd)
```

```
## [1] 0.3649604 0.3661259 0.3662550 0.3663581
```

```r
apply(alpha.MC.logisticNormal, 2, sd)
```

```
## [1] 0.4026649 0.4025549 0.4027612 0.4021794
```

```r
beta.par.dirichlet <- alpha.MC.dirichlet %*% cbind(av, bv)
beta.par.logisticNormal <- alpha.MC.logisticNormal %*% cbind(av, bv)

theta.par.dirichlet <- apply(beta.par.dirichlet, 1, function(x) rbeta(1, x[1], x[2]))
theta.par.logisticNormal <- apply(beta.par.logisticNormal, 1, function(x) rbeta(1, x[1], x[2]))
# Prior
PaperBeta.tbl[4, 1] <- mean(theta.par.dirichlet)
PaperBeta.tbl[4, 2:3] <- quantile(theta.par.dirichlet, c(.025, .975))

PaperBeta.tbl[5, 1] <- mean(theta.par.logisticNormal)
PaperBeta.tbl[5, 2:3] <- quantile(theta.par.logisticNormal, c(.025, .975))

####### Hierarchical posteriors

library(rstan)
```

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

```
## rstan (Version 2.19.1, 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)
options(mc.cores = 4)

betadata.stan <-  list(Y = y, X = X, N = n, K = K, a = av, b = bv)

## Dirichlet
compiled.dirichlet <- stan_model("stan/posterior_beta_Dirichlet_pooled.stan")
dirichlet.posterior <- sampling(compiled.dirichlet, data = betadata.stan,
                                control = list(adapt_delta = .99, max_treedepth = 15))
```

```
## Warning: There were 105 divergent transitions after warmup. Increasing adapt_delta above 0.99 may hel
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```r
print(dirichlet.posterior, pars = c("theta", "alpha"))
```

```
## Inference for Stan model: posterior_beta_Dirichlet_pooled.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
```

```
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
## theta    0.89    0.00 0.08  0.7 0.85 0.91 0.95  0.99  2333    1
## alpha[1] 0.26    0.01 0.37  0.0 0.00 0.02 0.53  1.00  3057    1
## alpha[2] 0.24    0.01 0.36  0.0 0.00 0.01 0.44  1.00  3360    1
## alpha[3] 0.28    0.01 0.38  0.0 0.00 0.02 0.59  1.00  2902    1
## alpha[4] 0.21    0.01 0.34  0.0 0.00 0.01 0.32  1.00  2974    1
##
## Samples were drawn using NUTS(diag_e) at Tue Jun 25 10:09:50 2019.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
check_hmc_diagnostics(dirichlet.posterior)
```

```
##
## Divergences:
```

```
## 105 of 4000 iterations ended with a divergence (2.625%).
## Try increasing 'adapt_delta' to remove the divergences.
```

```
##
## Tree depth:
```

```
## 0 of 4000 iterations saturated the maximum tree depth of 15.
```

```
##
## Energy:
```

```
## E-BFMI indicated no pathological behavior.
```

```
theta.dirichlet <- extract(dirichlet.posterior, 'theta')$theta
alphas.dirichlet <- extract(dirichlet.posterior, 'alpha')$alpha
post.alpha.cred.dirichlet <- apply(alphas.dirichlet, 2, quantile, probs = c(.025, .975))
betaPars.dirichlet <- extract(dirichlet.posterior, c('astar', 'bstar'))
ab.dirichlet <- unlist( lapply(betaPars.dirichlet, mean) )

(PaperBeta.tbl[4, 4:6] <- mean_ci(theta.dirichlet) )
```

```
##        mean       lwr       upr
## 1 0.8930894 0.6967217 0.9886246
```

```
(AlphasBeta.tbl[3, ] <- colMeans(alphas.dirichlet))
```

```
## [1] 0.2636935 0.2437520 0.2781582 0.2143962
```

```
## Logistic normal
compiled.logisticNormal <- stan_model("stan/posterior_beta_logisticNormal_pooled.stan")

betadata.stan$means <- digamma(X)-digamma(X[K])
betadata.stan$Sigma <- constructSigma(X)

logisticNormal.posterior <- sampling(compiled.logisticNormal, data = betadata.stan,
                                     control = list(adapt_delta = .99, max_treedepth = 15))
print(logisticNormal.posterior, pars = c("theta", "alpha"))
```

```
## Inference for Stan model: posterior_beta_logisticNormal_pooled.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
```

```
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
## theta    0.89    0.00 0.08 0.68 0.85 0.91 0.95  0.99  2578    1
## alpha[1] 0.27    0.01 0.41 0.00 0.00 0.00 0.67  1.00  2111    1
## alpha[2] 0.25    0.01 0.40 0.00 0.00 0.00 0.48  1.00  2552    1
## alpha[3] 0.27    0.01 0.41 0.00 0.00 0.00 0.69  1.00  2927    1
## alpha[4] 0.21    0.01 0.38 0.00 0.00 0.00 0.17  1.00  2504    1
##
## Samples were drawn using NUTS(diag_e) at Tue Jun 25 10:09:51 2019.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
check_hmc_diagnostics(logisticNormal.posterior)
```

```
##
## Divergences:

## 0 of 4000 iterations ended with a divergence.

##
## Tree depth:

## 0 of 4000 iterations saturated the maximum tree depth of 15.

##
## Energy:

## E-BFMI indicated no pathological behavior.
```

```
theta.logisticNormal <- extract(logisticNormal.posterior, 'theta')$theta
alphas.logisticNormal <- extract(logisticNormal.posterior, 'alpha')$alpha
post.alpha.cred.logisticNormal <- apply(alphas.logisticNormal, 2, quantile, probs = c(.025, .975))
betaPars.logisticNormal <- extract(logisticNormal.posterior, c('astar', 'bstar'))
ab.logisticNormal <- unlist( lapply(betaPars.logisticNormal, mean) )

( PaperBeta.tbl[5, 4:6] <- mean_ci(theta.logisticNormal) )
```

```
##        mean       lwr       upr
## 1 0.8918794 0.6807232 0.9886171
```

```
( AlphasBeta.tbl[4, ] <- colMeans(alphas.logisticNormal) )
```

```
## [1] 0.2697354 0.2475329 0.2711305 0.2116012
```

```
####### KL prior from Rufo et al 2012

alphas.rufo <- c(0, 0, 0, 1)

(AlphasBeta.tbl[5, ] <- alphas.rufo)
```

```
## [1] 0 0 0 1
```

```
ab.rufo <- pool_par(alphas.rufo, av, bv)

# Prior
( PaperBeta.tbl[6, 1:3] <- stat_beta(ab.rufo) )
```

```
## [1] 0.7001414 0.1737705 0.9936577
```

```r
# Posterior
( PaperBeta.tbl[6, 4:6] <- stat_beta(ab.rufo +  c(y, n - y)) )
```

```
## [1] 0.8559401 0.6273426 0.9828843
```

```r
#### Finally, tables!
round(PaperBeta.tbl, 3)
```

```
##                              mean.prior lower.prior upper.prior mean.post
## equal_weights                     0.898       0.644       0.998     0.899
## maximum_entropy                   0.700       0.174       0.994     0.856
## minimum_KL                        0.823       0.424       0.997     0.875
## hierarchical_Dirichlet            0.855       0.391       0.998     0.893
## hierarchical_LogisticNormal       0.847       0.356       0.998     0.892
## Rufo_2012                         0.700       0.174       0.994     0.856
##                              lower.post upper.post
## equal_weights                     0.733      0.988
## maximum_entropy                   0.627      0.983
## minimum_KL                        0.672      0.985
## hierarchical_Dirichlet            0.697      0.989
## hierarchical_LogisticNormal       0.681      0.989
## Rufo_2012                         0.627      0.983
```

```r
round(AlphasBeta.tbl, 3)
```

```
##                              alpha_0 alpha_1 alpha_2 alpha_3
## maximum_entropy                0.000   0.000   0.000   1.000
## minimum_KL                     0.040   0.960   0.000   0.000
## hierarchical_Dirichlet         0.264   0.244   0.278   0.214
## hierarchical_LogisticNormal    0.270   0.248   0.271   0.212
## Rufo_2012                      0.000   0.000   0.000   1.000
```

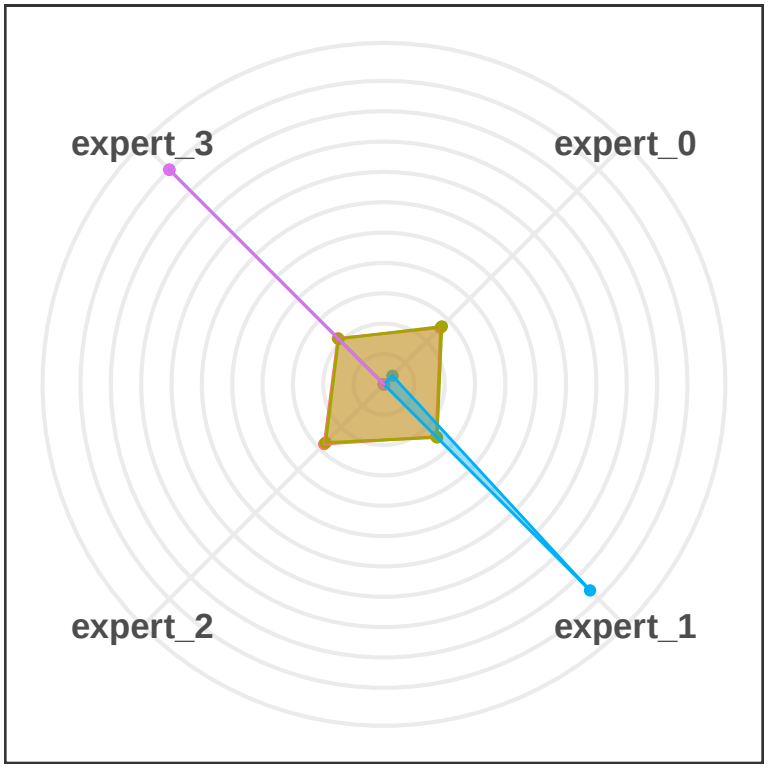
```r
round(PaperBeta.tbl, 2)
```

```
##                              mean.prior lower.prior upper.prior mean.post
## equal_weights                      0.90        0.64        1.00      0.90
## maximum_entropy                    0.70        0.17        0.99      0.86
## minimum_KL                         0.82        0.42        1.00      0.87
## hierarchical_Dirichlet             0.86        0.39        1.00      0.89
## hierarchical_LogisticNormal        0.85        0.36        1.00      0.89
## Rufo_2012                          0.70        0.17        0.99      0.86
##                              lower.post upper.post
## equal_weights                      0.73       0.99
## maximum_entropy                    0.63       0.98
## minimum_KL                         0.67       0.99
## hierarchical_Dirichlet             0.70       0.99
## hierarchical_LogisticNormal        0.68       0.99
## Rufo_2012                          0.63       0.98
```

```r
round(AlphasBeta.tbl, 2)
```

```
##                              alpha_0 alpha_1 alpha_2 alpha_3
## maximum_entropy                 0.00    0.00    0.00    1.00
## minimum_KL                      0.04    0.96    0.00    0.00
## hierarchical_Dirichlet          0.26    0.24    0.28    0.21
## hierarchical_LogisticNormal     0.27    0.25    0.27    0.21
## Rufo_2012                       0.00    0.00    0.00    1.00
```

```
####### Plotting

posterior_experts <- data.frame(
  alpha = as.numeric(c(AlphasBeta.tbl[1, ], AlphasBeta.tbl[2, ], AlphasBeta.tbl[5, ],
            AlphasBeta.tbl[3, ], AlphasBeta.tbl[4, ])),
  lwr = c(rep(NA, 12), post.alpha.cred.dirichlet[1, ], post.alpha.cred.logisticNormal[1, ]),
  upr = c(rep(NA, 12), post.alpha.cred.dirichlet[2, ], post.alpha.cred.logisticNormal[2, ]),
  expert = rep(paste("expert_", 0:(K-1), sep = ""), 5),
  method = rep(c("maximum_entropy", "minimum_KL", "Rufo_2012",  "Dirichlet", "logistic_normal"), each =
)

####
radar_alphas <- ggplot(data = posterior_experts,
       aes(x = expert, y = alpha, group = method, colour = method, fill = method)) +
  geom_point() +
  geom_polygon(alpha = 0.4) +
  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
```



```
ggsave(plot = radar_alphas, filename = "../plots/alphas_radar_Savchuk.pdf")
```

```
## Saving 6.5 x 4.5 in image
```

```
##############
# Now  let's look at marginal likelihoods for the pooled priors

pars <- list(equal_weights = ab.Equal.star,
             maximum_entropy = ab.MaxEnt.star,
             minimum_KL = ab.KL.star,
             hierarchical_Dirichlet = ab.dirichlet ,
             hierarchical_LogisticNormal = ab.logisticNormal,
             Rufo_2012 = ab.rufo)
lapply(pars, function(p) ml_beta(yi = y, ni = n, a = p[1], b = p[2]))
```

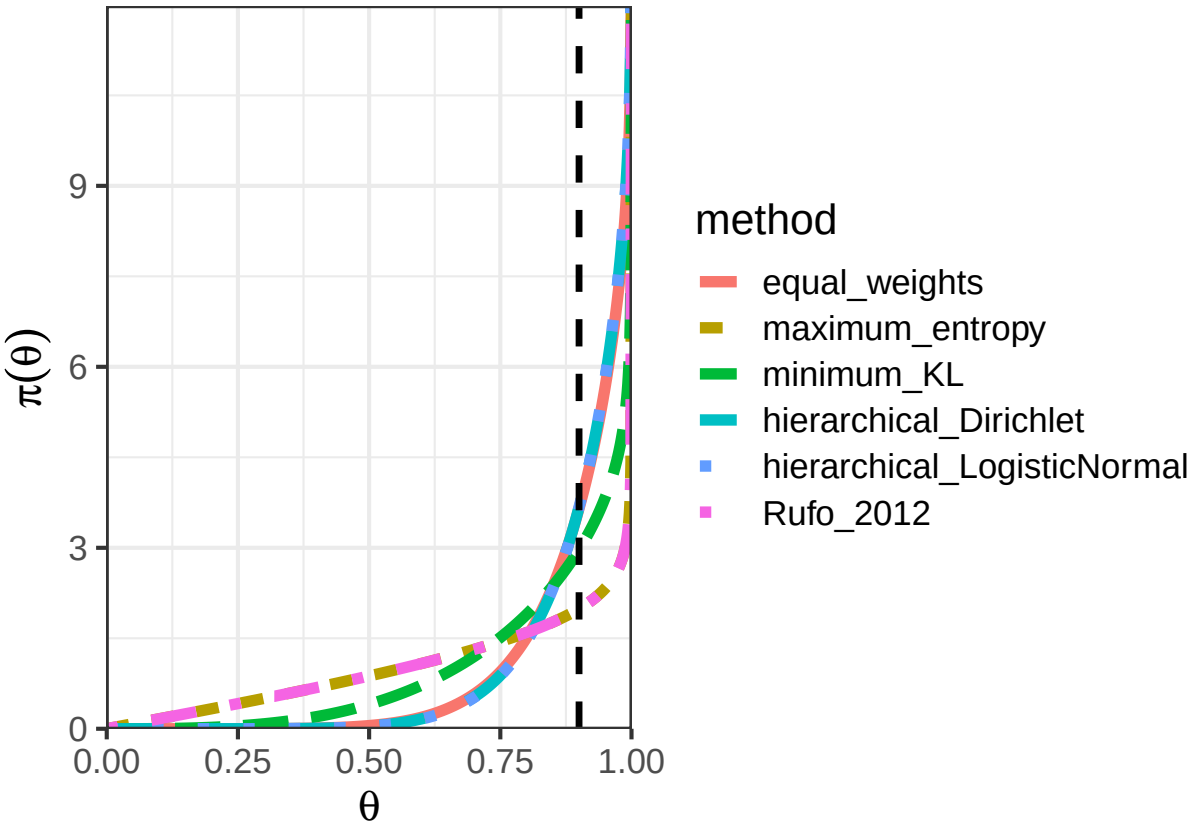
```
## $equal_weights
## [1] 0.2549001
##
## $maximum_entropy
## [1] 0.1636389
##
## $minimum_KL
## [1] 0.2233347
##
## $hierarchical_Dirichlet
##     astar
## 0.2555974
##
## $hierarchical_LogisticNormal
##     astar
## 0.2556627
##
## $Rufo_2012
## [1] 0.1636389
```

```
apply(AlphasBeta.tbl, 1, get_ratio)
```

```
##             maximum_entropy                  minimum_KL
##               1.726766e+15               2.408141e+01
##      hierarchical_Dirichlet hierarchical_LogisticNormal
##               1.054855e+00               1.005172e+00
##                  Rufo_2012
##                        Inf
```

```
J <- length(pars)
method.densities.list <- vector(J, mode ="list")
for (j in 1:J){
  method.densities.list[[j]] <- data.frame(
    theta = theta.grid,
    dens = dbeta(theta.grid, shape1 = pars[[j]][1], shape2 = pars[[j]][2]),
    method = names(pars)[j]
  )
}

method.densities.df <- do.call(rbind, method.densities.list)
```

```
method_priors <- ggplot(method.densities.df, aes(x = theta, y = dens,
                                                  linetype = method, colour = method)) +
  geom_line(size = 2) +
  scale_x_continuous(expression(theta), expand = c(0, 0)) +
  scale_y_continuous(expression(pi(theta)), expand = c(0, 0)) +
  geom_vline(xintercept = y/n, linetype = "dashed", size = 1.2) +
  theme_bw(base_size = 16)

method_priors
```




```
ggsave(method_priors, filename = "../plots/method_prior_densities_Savchuk.pdf")
```

```
## Saving 6.5 x 4.5 in image
```