stacking.r

luiz

2019-11-25

```
library(loo)
```

```
## This is loo version 2.1.0.
## **NOTE: As of version 2.0.0 loo defaults to 1 core but we recommend using as many as possible. Use t
```

```
library(rstanarm)
```

```
## Loading required package: Rcpp

## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo

## rstanarm (Version 2.19.2, packaged: 2019-10-01 20:20:33 UTC)

## - Do not expect the default priors to remain the same in future rstanarm versions.

## Thus, R scripts should specify priors explicitly, even if they are just the defaults.

## - For execution on a local, multicore CPU with excess RAM we recommend calling

## options(mc.cores = parallel::detectCores())

## - bayesplot theme set to bayesplot::theme_default()

##    * Does _not_ affect other ggplot2 plots

##    * See ?bayesplot_theme_set for details on theme setting
```

```
library(ggplot2)
library(fitdistrplus)
```

```
## Loading required package: MASS

## Loading required package: survival

## Loading required package: npsurv

## Loading required package: lsei
```

```
source("pooling_aux.r")
##################

data(Kline)
d <- Kline
d$log_pop <- log(d$population)
d$contact_high <- ifelse(d$contact=="high", 1, 0)
str(d)
```

```
## 'data.frame':    10 obs. of  7 variables:
##  $ culture     : Factor w/ 10 levels "Chuuk","Hawaii",..: 4 7 6 10 3 9 1 5 8 2
##  $ population  : int  1100 1500 3600 4791 7400 8000 9200 13000 17500 275000
##  $ contact     : Factor w/ 2 levels "high","low": 2 2 2 1 1 1 1 2 1 2
##  $ total_tools : int  13 22 24 43 33 19 40 28 55 71
##  $ mean_TU     : num  3.2 4.7 4 5 5 4 3.8 6.6 5.4 6.6
##  $ log_pop     : num  7 7.31 8.19 8.47 8.91 ...
##  $ contact_high: num  0 0 0 1 1 1 1 0 1 0
```

```
N <- nrow(d) ## nobs

fit10 <-
  stan_glm(
    total_tools ~ log_pop + contact_high + log_pop * contact_high,
    family = poisson(link = "log"),
    data = d,
    prior = normal(0, 1, autoscale = FALSE),
    prior_intercept = normal(0, 100, autoscale = FALSE),
    seed = 2030
    # seed = 666
)
```

```
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.7e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.27 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.092289 seconds (Warm-up)
## Chain 1:                0.113179 seconds (Sampling)
## Chain 1:                0.205468 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 6e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.090599 seconds (Warm-up)
## Chain 2:                0.088617 seconds (Sampling)
## Chain 2:                0.179216 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 6e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.097708 seconds (Warm-up)
## Chain 3:                0.093126 seconds (Sampling)
## Chain 3:                0.190834 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 6e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.091593 seconds (Warm-up)
## Chain 4:                0.086967 seconds (Sampling)
## Chain 4:                0.17856 seconds (Total)
## Chain 4:
```

```
loo10 <- loo(fit10, save_psis = TRUE)
```

```
## Warning: Found 2 observation(s) with a pareto_k > 0.7. We recommend calling 'loo' again with argumen
```

```
loo10 <- loo(fit10, k_threshold=0.7, save_psis = TRUE)
```

```
## 2 problematic observation(s) found.
## Model will be refit 2 times.
```

```
##
## Fitting model 1 out of 2 (leaving out observation 4)
```

```
##
## Fitting model 2 out of 2 (leaving out observation 10)
```

```
waic10 <- waic(fit10)
```

```
## Warning: 4 (40.0%) p_waic estimates greater than 0.4. We recommend trying
## loo instead.
```

```
fit11 <- update(fit10, formula = total_tools ~ log_pop + contact_high)
```

```
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.1 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.014467 seconds (Warm-up)
## Chain 1:                0.015295 seconds (Sampling)
## Chain 1:                0.029762 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 2).
## Chain 2:
```

```
## Chain 2: Gradient evaluation took 6e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.014957 seconds (Warm-up)
## Chain 2:                0.017054 seconds (Sampling)
## Chain 2:                0.032011 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2.3e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.23 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.016614 seconds (Warm-up)
## Chain 3:                0.018128 seconds (Sampling)
## Chain 3:                0.034742 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 6e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
```

```
## Chain 4: 
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4: 
## Chain 4:  Elapsed Time: 0.015127 seconds (Warm-up)
## Chain 4:                0.016247 seconds (Sampling)
## Chain 4:                0.031374 seconds (Total)
## Chain 4:
```

```
fit12 <- update(fit10, formula = total_tools ~ log_pop)
```

```
## 
## SAMPLING FOR MODEL 'count' NOW (CHAIN 1).
## Chain 1: 
## Chain 1: Gradient evaluation took 2.6e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.26 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1: 
## Chain 1: 
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1: 
## Chain 1:  Elapsed Time: 0.013542 seconds (Warm-up)
## Chain 1:                0.014326 seconds (Sampling)
## Chain 1:                0.027868 seconds (Total)
## Chain 1: 
## 
## SAMPLING FOR MODEL 'count' NOW (CHAIN 2).
## Chain 2: 
## Chain 2: Gradient evaluation took 6e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2: 
## Chain 2: 
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.01429 seconds (Warm-up)
## Chain 2:                0.014157 seconds (Sampling)
## Chain 2:                0.028447 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 6e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.012453 seconds (Warm-up)
## Chain 3:                0.013971 seconds (Sampling)
## Chain 3:                0.026424 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 7e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.014094 seconds (Warm-up)
## Chain 4:                0.01486 seconds (Sampling)
## Chain 4:                0.028954 seconds (Total)
## Chain 4:
```

```r
loo11 <- loo(fit11, k_threshold = 0.7, save_psis = TRUE)
```

```
## 1 problematic observation(s) found.
## Model will be refit 1 times.
```

```
##
## Fitting model 1 out of 1 (leaving out observation 10)
```

```r
loo12 <- loo(fit12, k_threshold = 0.7, save_psis = TRUE)
```

```
## 1 problematic observation(s) found.
## Model will be refit 1 times.
##
## Fitting model 1 out of 1 (leaving out observation 10)
```

```r
lpd_point <- cbind(
  loo10$pointwise[, "elpd_loo"],
  loo11$pointwise[, "elpd_loo"],
  loo12$pointwise[, "elpd_loo"]
)

waic11 <- waic(fit11)
```

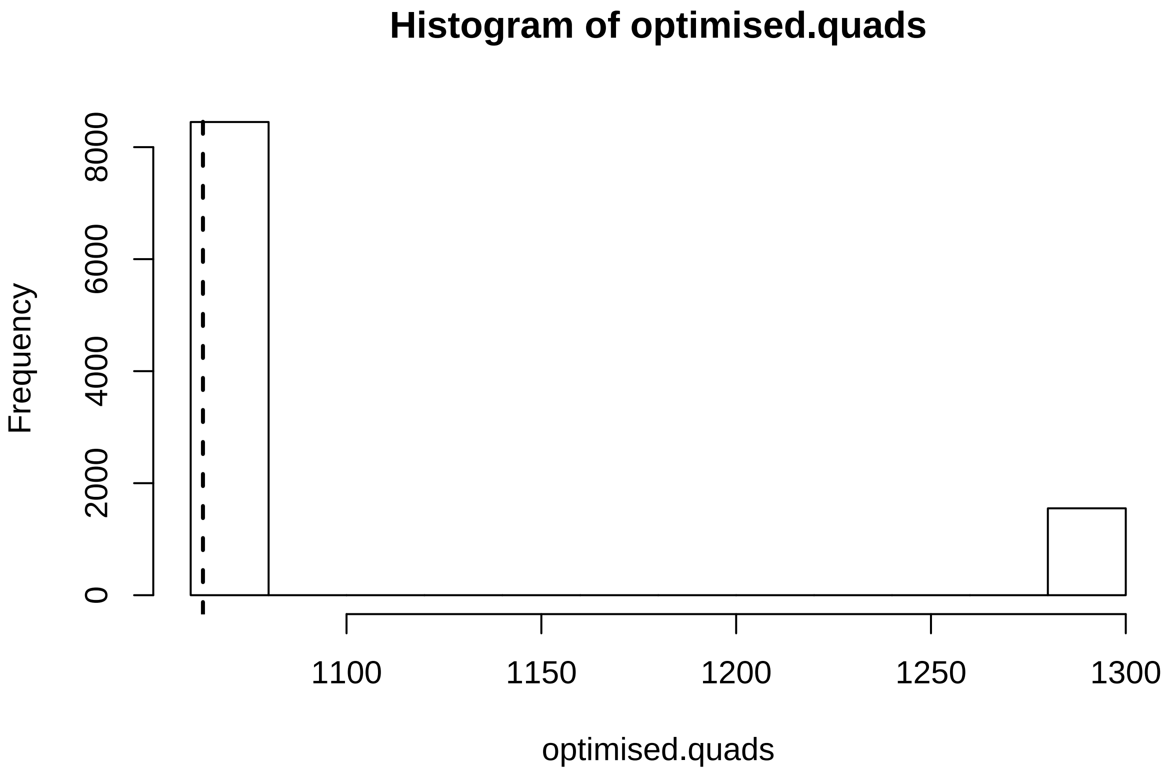
```
## Warning: 4 (40.0%) p_waic estimates greater than 0.4. We recommend trying
## loo instead.
```

```r
waic12 <- waic(fit12)
```

```
## Warning: 5 (50.0%) p_waic estimates greater than 0.4. We recommend trying
## loo instead.
```

```r
waics <- c(
  waic10$estimates["elpd_waic", 1],
  waic11$estimates["elpd_waic", 1],
  waic12$estimates["elpd_waic", 1]
)

waic_wts <- exp(waics) / sum(exp(waics))
pbma_wts <- pseudobma_weights(lpd_point, BB = FALSE)
pbma_BB_wts <- pseudobma_weights(lpd_point) # default is BB=TRUE
stacking_wts <- stacking_weights(lpd_point)

fits <- list(
  fit1 = fit10,
  fit2 = fit11,
  fit3 = fit12
)
K <- length(fits)

########
#### Optimisation
## Functions
loss <- function(y_obs, mus_i, vs_i, alpha){
  ## Expectation of squared error under the pooled predictive
 pars <- pool_par_gauss(alpha, mus_i, vs_i)
 loss <- y_obs^2 - 2*y_obs*pars[1] + pars[1]^2 + pars[2]^2
   return(loss)
}
#
compute_overall_loss <- function(dt, alpha, pars){
  N <- nrow(dt)
  if(length(pars) != N) stop("list of Gaussian parameters needs to be of same size as number of obs")
  Ls <- rep(NA, N)
  for(i in 1:N){
    Ls[i] <- loss(y_obs = dt$y[i], mus_i = pars[[i]]$mu, vs_i = pars[[i]]$v, alpha = alpha)
  }
  # cat(Ls, "\n")
  return(sum(Ls))
}
#
loss_alpha_unconstrained <- function(alpha_unc){
  compute_overall_loss(dt = d, alpha = alpha_01(alpha_unc), pars = pars.list)
}
#
parse_pars <- function(dist.fit){
  K <- length(dist.fit)
  mus <- vs <- rep(NA, K)
  for(k in 1:K){
    mus[k] <- dist.fit[[k]]$estimate[1]
    vs[k] <- dist.fit[[k]]$estimate[2]^2
  }
  return(list(
    mu = mus,
    v = vs
  ))
}
##################
#### Getting parameters
full.postpred <- lapply(fits,
                        posterior_predict)

pars.list <- vector(N, mode = "list")
for(i in 1:N){
  pars.list[[i]] <- parse_pars(
    lapply(full.postpred, function(pp) fitdist(pp[, i], distr = dnorm, method = "mle"))
  )
}
```

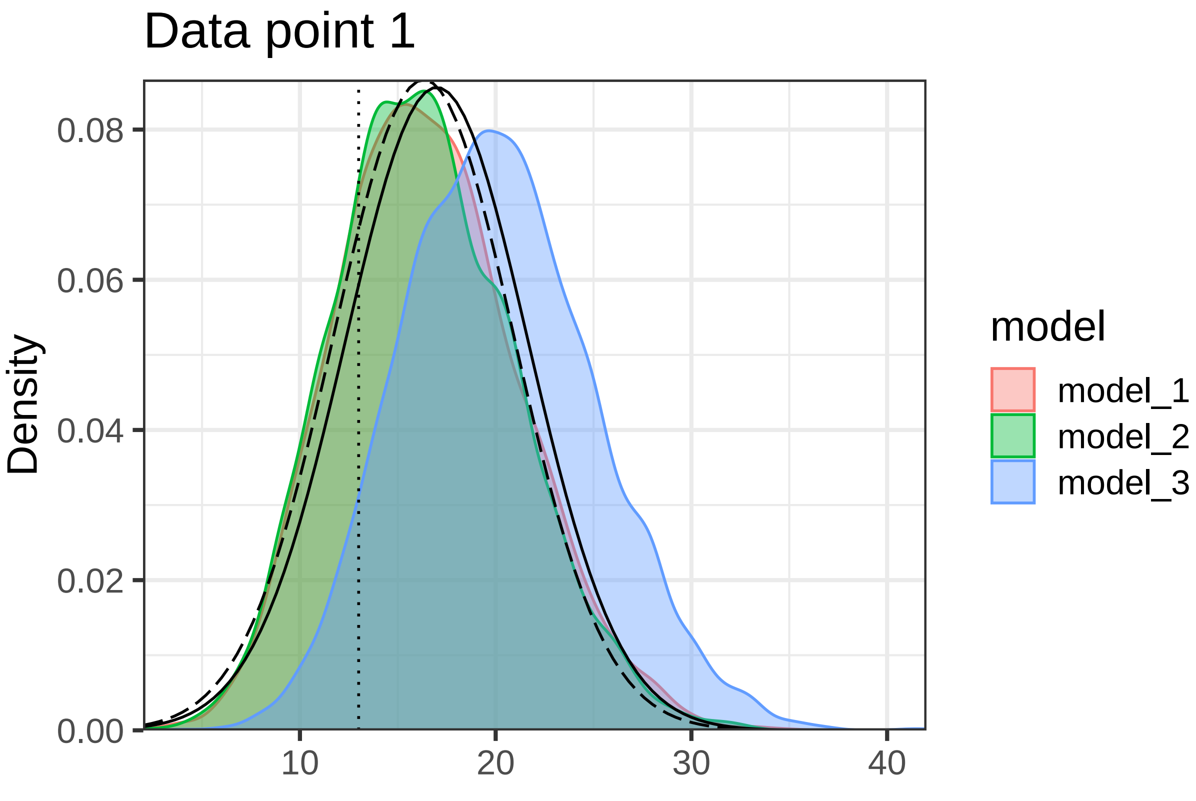
```
d$y <- d$total_tools

M <- 10000
quad.many.startingPoints <- matrix(rnorm(n = (K-1)*M, mean = 0, sd = 100), ncol = K-1, nrow = M)
many.quads <- lapply(1:M, function(i) {
  optim(quad.many.startingPoints[i, ], loss_alpha_unconstrained)
})
optimised.quads <- unlist(lapply(many.quads, function(x) x$value))

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

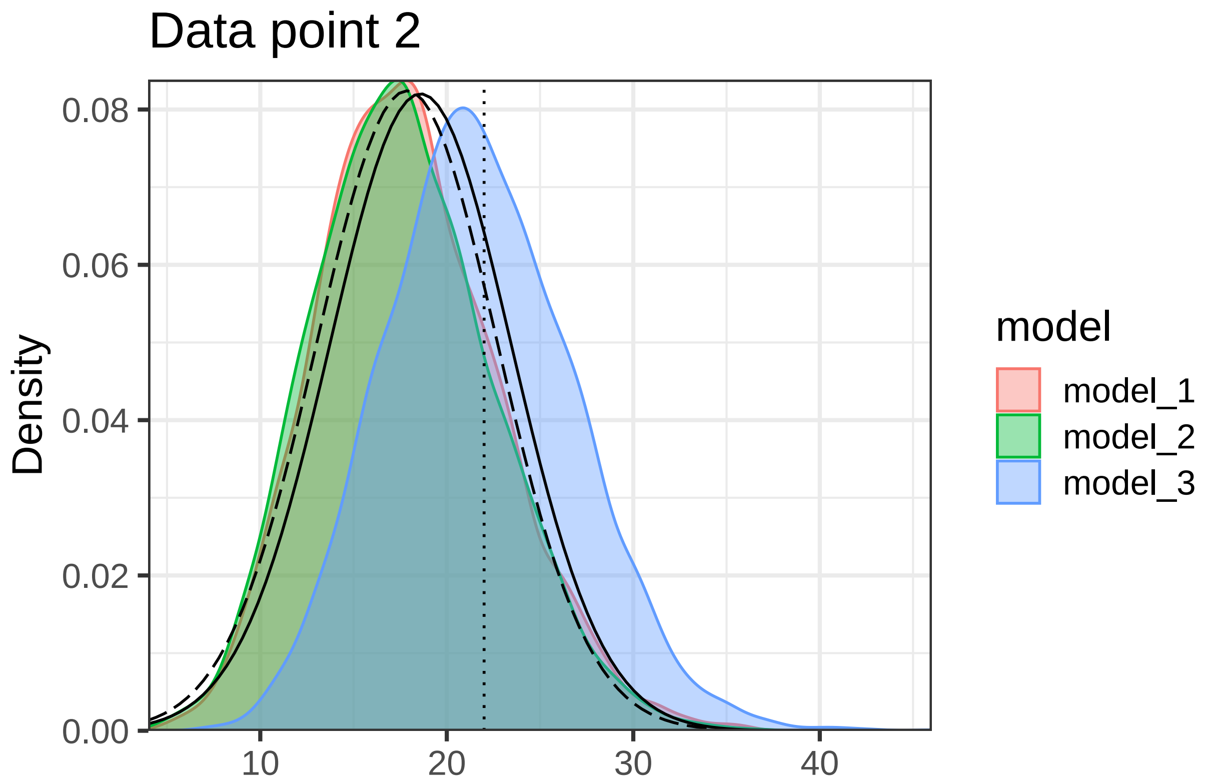


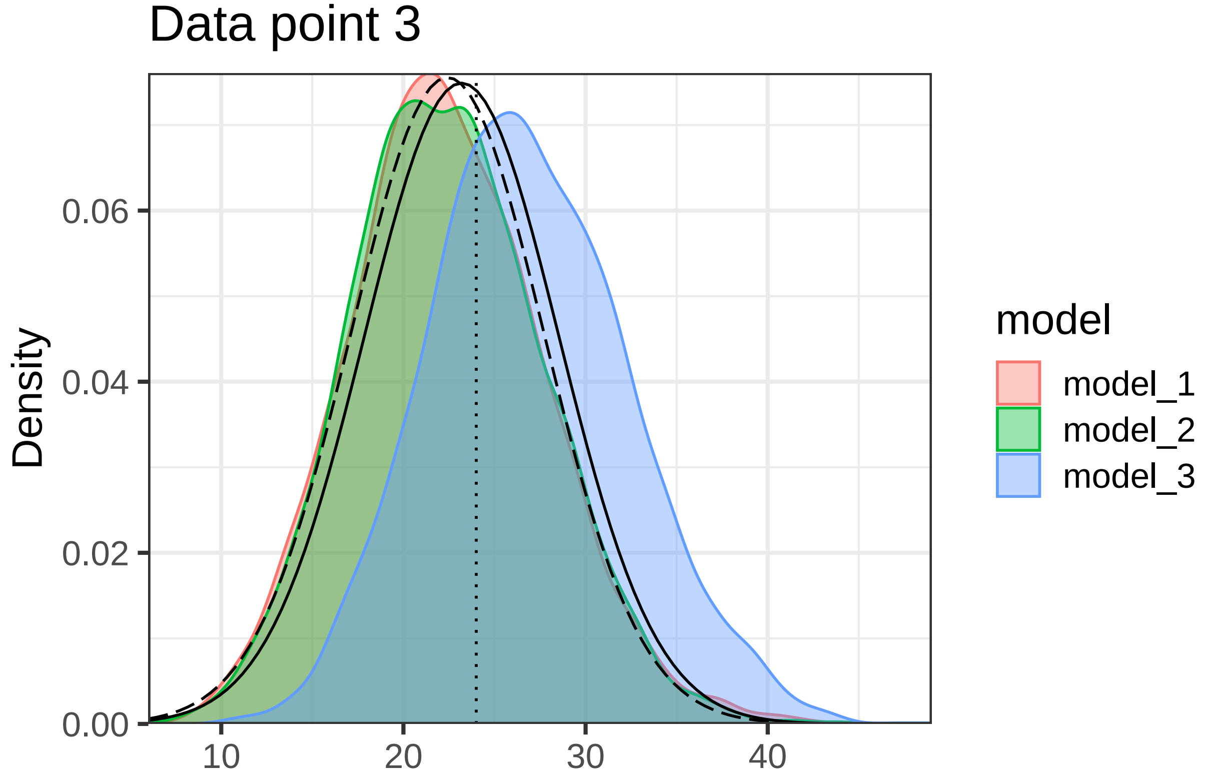
```
min_quad_loss_wts <- alpha_01(many.quads[[which.min(optimised.quads)]]$par)

round(cbind(waic_wts, pbma_wts, pbma_BB_wts, stacking_wts, min_quad_loss_wts), 2)
```

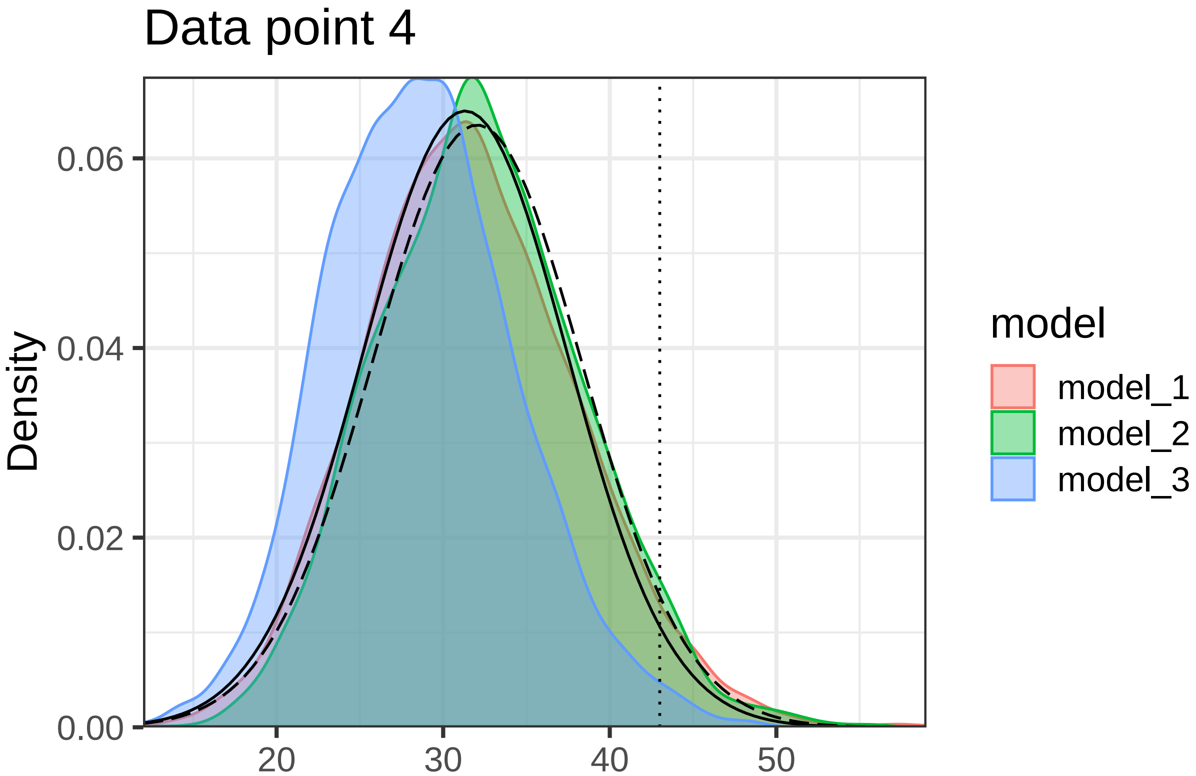
```
##        waic_wts pbma_wts pbma_BB_wts stacking_wts min_quad_loss_wts
## model1     0.39     0.33        0.27         0.00              0.00
## model2     0.57     0.65        0.53         0.78              0.96
## model3     0.04     0.03        0.20         0.22              0.04
```

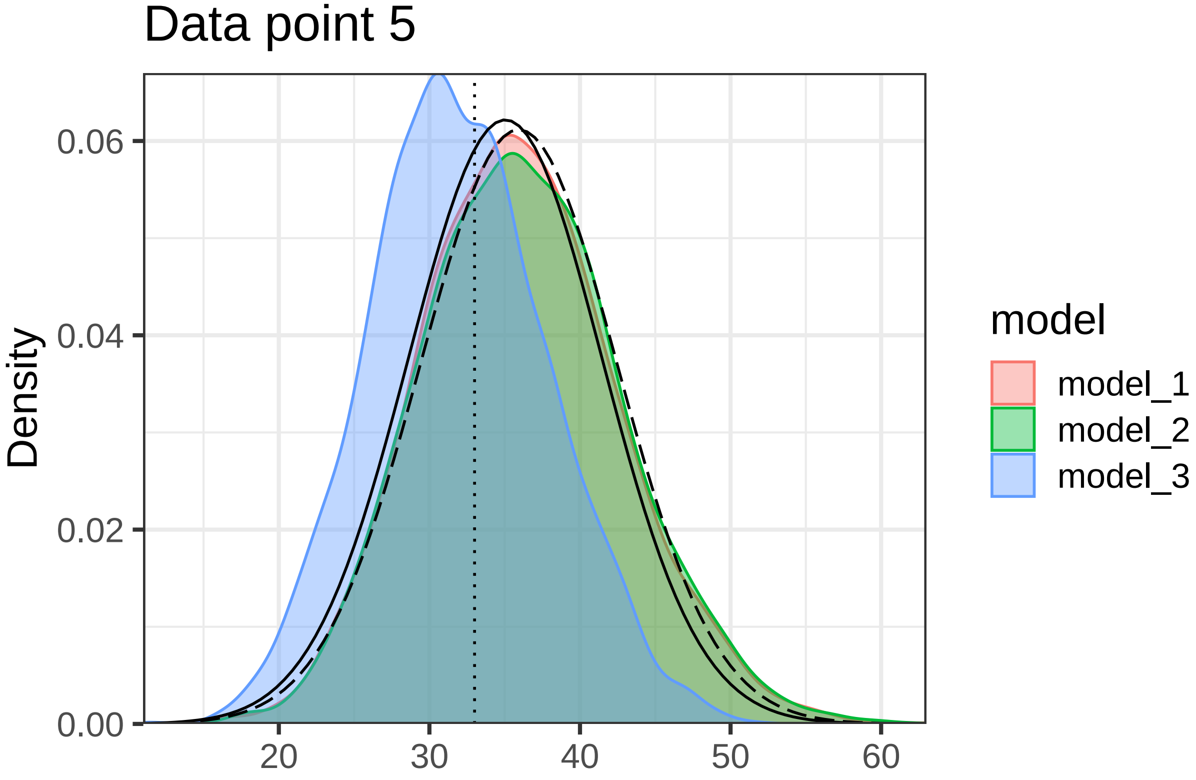
```
### Pooling and plotting for each observation


for(pos in 1:nrow(d)){

  new.data <- d[pos, ]

  mus <- pars.list[[pos]]$mu
  vs <- pars.list[[pos]]$v
  pars.EqualWeights <- pool_par_gauss(rep(1/K, K), mus, vs)
  pars.WAIC <- pool_par_gauss(waic_wts, mus, vs)
  pars.pBMA <- pool_par_gauss(pbma_BB_wts, mus, vs)
  pars.stacking <- pool_par_gauss(stacking_wts, mus, vs)
  pars.minQuadPool <- pool_par_gauss(min_quad_loss_wts, mus, vs)

  pred.dfs <- vector(length(fits), mode = "list")
  for(k in 1:K){
    pred.dfs[[k]] <- data.frame(
      y.pred.new = as.vector(posterior_predict(fits[[k]], newdata = new.data)),
      model = paste("model_", k, sep = "")
    )
  }
  all.preds <- do.call(rbind, pred.dfs)

  pplot <- ggplot(all.preds, aes(x = y.pred.new, colour = model, fill = model)) +
    geom_density(alpha = .4) +
    scale_x_continuous("", expand = c(0, 0)) +
    scale_y_continuous("Density", expand = c(0, 0)) +
    geom_vline(xintercept = new.data$total_tools, linetype = "dotted") +
    stat_function(fun = dnorm, args = list(mean = pars.minQuadPool[1],
                                           sd = pars.minQuadPool[2]), inherit.aes = FALSE, linetype = "
    stat_function(fun = dnorm, args = list(mean = pars.stacking[1],
                                           sd = pars.stacking[2]), inherit.aes = FALSE) +
    theme_bw(base_size = 16) +
    ggtitle(paste("Data point", pos))
  print(pplot)
}
```

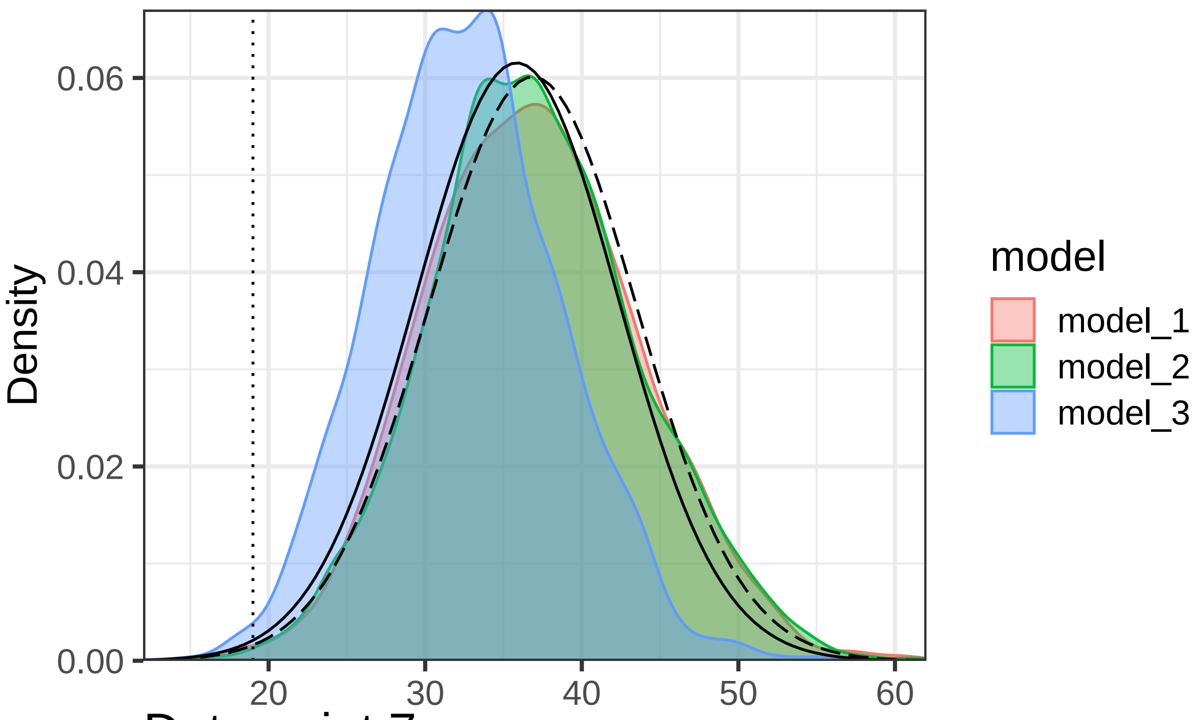



Data point 2
Density
0.08
0.06
0.04
0.02
0.00
10
20
30
40
model
model_1
model_2
model_3

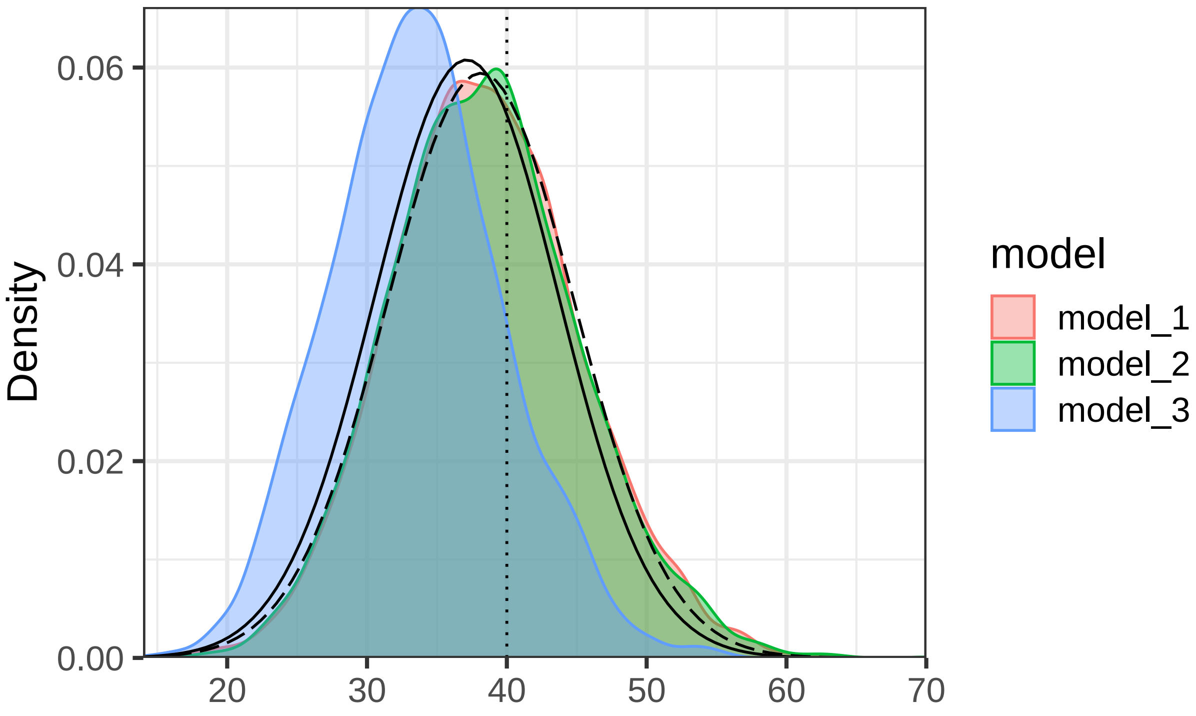

Data point 3
Density
0.06
0.04
0.02
0.00
10
20
30
40
model
model_1
model_2
model_3


Data point 4
Density
0.06
0.04
0.02
0.00
20
30
40
50
model
model_1
model_2
model_3


Data point 5
Density
0.06
0.04
0.02
0.00
20
30
40
50
60
model
model_1
model_2
model_3

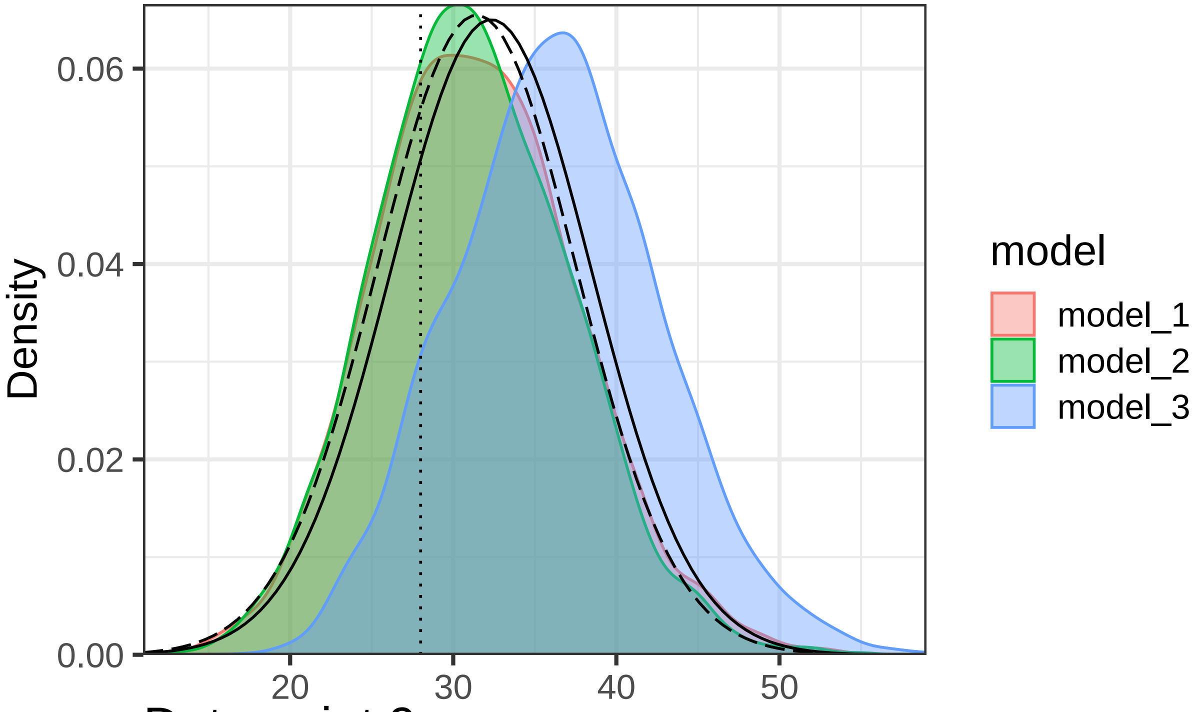


Data point 6

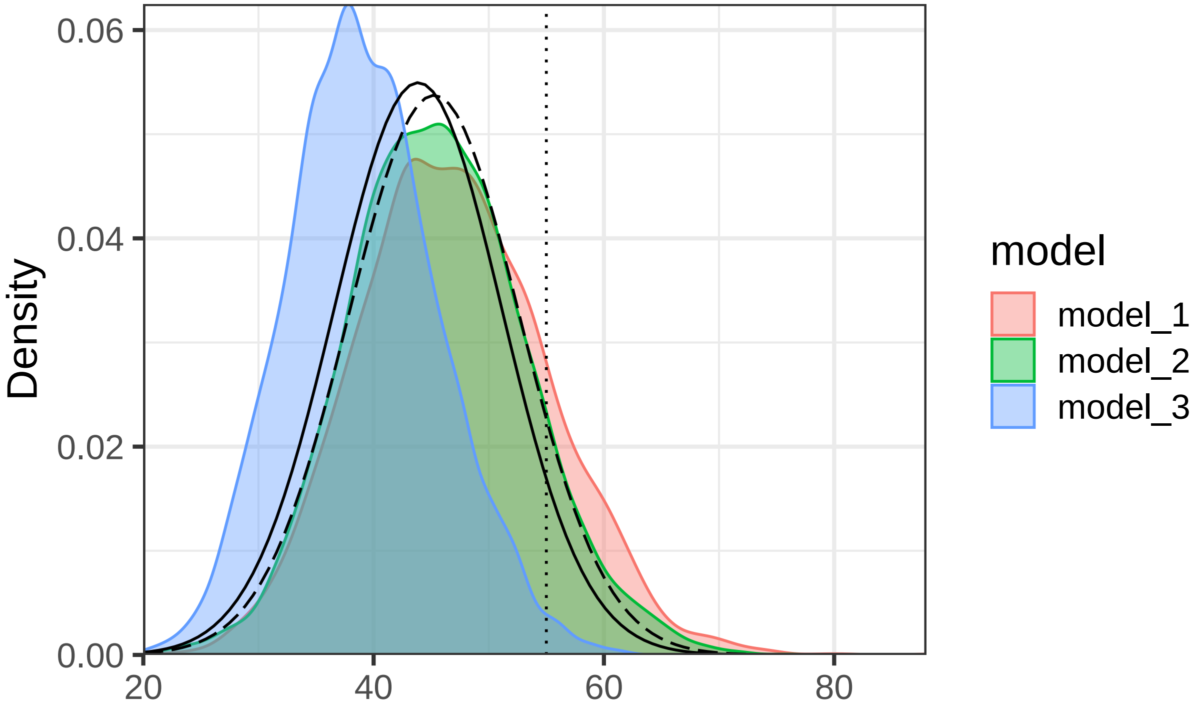


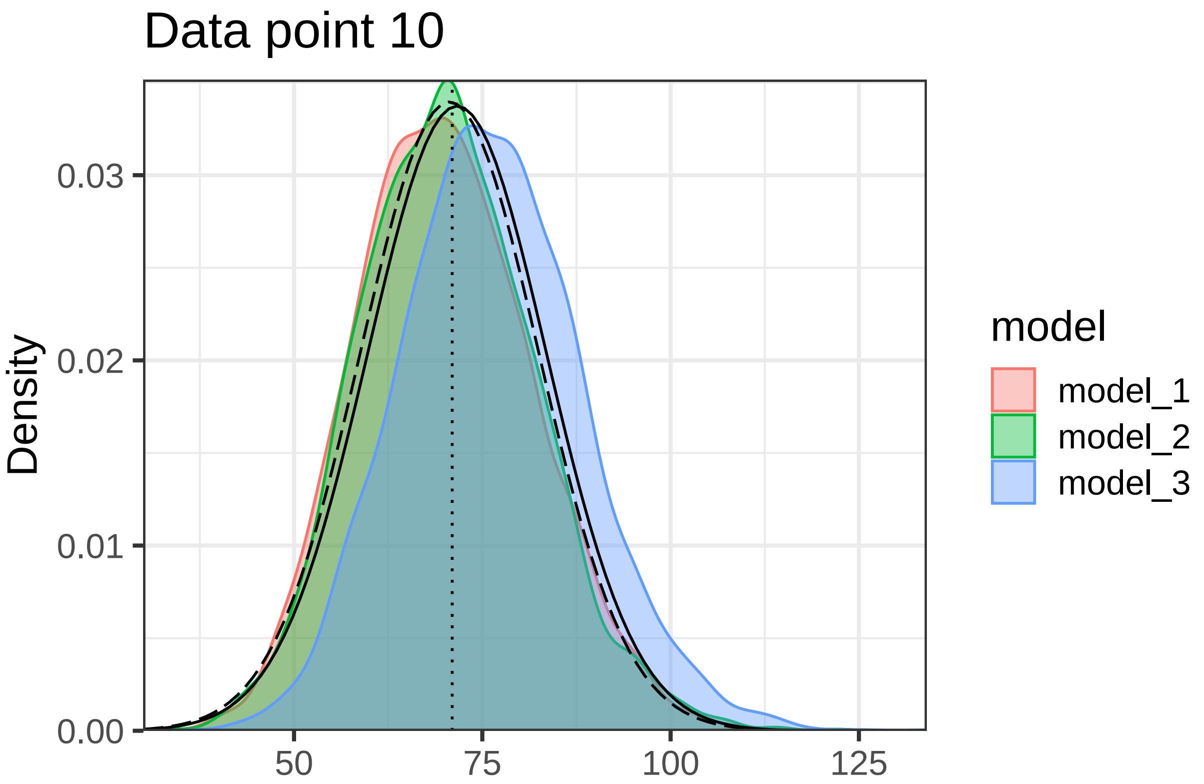
Data point 7

Data point 8



Data point 9




Data point 10
Density
0.03
0.02
0.01
0.00
50
75
100
125
model
model_1
model_2
model_3