BDA - Assignment 8

Anonymous

```
library(tidyr)
library(rstan)
```

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

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

```
## rstan (Version 2.19.2, 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)
```

```
## For improved execution time, we recommend calling
## Sys.setenv(LOCAL_CPPFLAGS = '-march=native')
## although this causes Stan to throw an error on a few processors.
```

```
##
## Attaching package: 'rstan'
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
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 tl
```

```
## **NOTE for Windows 10 users: loo may be very slow if 'mc.cores' is set in your .Rprofile file (see ht
```

```
##
## Attaching package: 'loo'
```

```
## The following object is masked from 'package:rstan':
##
##     loo
```

```
library(ggplot2)
library(gridExtra)
library(bayesplot)
```

```
## This is bayesplot version 1.7.0

## - Online documentation and vignettes at mc-stan.org/bayesplot

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

##    * Does _not_ affect other ggplot2 plots

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

```
theme_set(bayesplot::theme_default(base_family = "sans"))
library(shinystan)
```

```
## Loading required package: shiny

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

##
## This is shinystan version 2.5.0
```

```
source('stan_utility.R')
library(aaltobda)
SEED <- 48927 # set random seed for reproducability
```

Problem 1: Fit the models with Stan as instructed in Assignment 7.

A) Separate model

```
data("factory")
```

First I considered the separated model. For this model, each machine j is assumed to have unrelated means μ_j and σ_j. The priors are also non-informative (uniform). The Stan implementation is as follows:

```
writeLines(readLines("separat.stan"))
```

```
##
## data {
##   int<lower=0> N; // number of data points
##   int<lower=0> K; // number of groups
##   int<lower=1,upper=K> x[N]; // group indicator
##   vector[N] y; //
## }
## parameters {
##   vector[K] mu; // group means
##   vector<lower=0>[K] sigma; // group stds
## }
```

```
## model {
##   y ~ normal(mu[x], sigma[x]);
## }
## generated quantities {
##   real ypred;
##   vector[N] log_lik;
##   ypred = normal_rng(mu[6], sigma[6]);
##   for (i in 1:N){
##     log_lik[i] = normal_lpdf(y[i] | mu[x[i]], sigma[x[i]]);
##   }
## }
```

The data related to this model is :

```
data_separate <-list(N = ncol(factory) * nrow(factory),
                     K = ncol(factory),
                     x = rep(1:ncol(factory), nrow(factory)),
                     y = c(t(factory)))
```

We fit the separate model in stan as follow:

```
fit_separate <- stan(file="separat.stan", data = data_separate, seed = SEED)
# monitor(fit_separate, probs = c(0.1, 0.5, 0.9))
```

B) Pooled model

For the pooled model I assumed that μ and σ are the same for all machines. I also assumed uniform (non-informatice) priror for these parameters. The Stan implementation for pooled model is as:

```
writeLines(readLines("pooled.stan"))
```

```
##
## data {
##   int<lower=0> N; // number of data points
##   vector[N] y; //
## }
## parameters {
##   real mu; // group means
##   real<lower=0> sigma; // common std
## }
## model {
##   y ~ normal(mu, sigma);
## }
## generated quantities {
##   real ypred;
##   vector[N] log_lik;
##   ypred = normal_rng(mu, sigma);
##   for (i in 1:N){
##     log_lik[i] = normal_lpdf(y[i] | mu, sigma);
##   }
## }
```

The data related to this model is :

```
data_pooled <- list(N = ncol(factory) * nrow(factory),
                    y = c(t(factory)))
```

We fit the pooled model in stan as follows:

```
fit_pooled <- stan(file = "pooled.stan", data = data_pooled, seed = SEED)
```

C) Hierarchical model

In this model the means of the different machines are assumed to have a common standard deviation σ and means μ that are drawn from a normal distribution with μ_0 and σ_0. I assumed weekly informative priors for μ_0 and σ_0 and σ. The Stan implementation of the model is as follows:

```
writeLines(readLines("hierarchical.stan"))
```

```
## Warning in readLines("hierarchical.stan"): incomplete final line found on
## 'hierarchical.stan'
```

```
## data {
##   int<lower=0> N; // number of data points
##   int<lower=0> K; // number of groups
##   int<lower=1,upper=K> x[N]; // group indicator
##   vector[N] y; //
## }
## parameters {
##   real mu0; // prior mean
##   real<lower=0> sigma0; // prior std
##   vector[K] mu; // group means
##   real<lower=0> sigma; // group stds
## }
## model {
##   mu0 ~ normal(90, 15); // weakly informative prior
##   sigma0 ~ cauchy(0,4); // weakly informative prior
##   mu ~ normal(mu0, sigma0); // population prior with unknown parameters
##   sigma ~ cauchy(0,4); // weakly informative prior
##   y ~ normal(mu[x] , sigma);
## }
## generated quantities {
##   real ypred;
##   real mu_7;
##   vector[N] log_lik;
##   ypred = normal_rng(mu[6], sigma);
##   mu_7 = normal_rng(mu0, sigma0);
##   for (i in 1:N){
##     log_lik[i] = normal_lpdf(y[i] | mu[x[i]], sigma);
##   }
## }
```

We fit the separate model in stan as follow:

```
fit_hierarchical <- stan(file="hierarchical.stan", data = data_separate, seed = SEED)
```

```
## Warning in readLines(file, warn = TRUE): incomplete final line
## found on 'C:\Users\Z RY\Dropbox (Aalto)\PhD\Courses\BDA\2019\E8 -
## Copy\hierarchical.stan'
```

```
## Warning: There were 53 divergent transitions after warmup. Increasing adapt_delta above 0.8 may help
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

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

```
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#bulk-ess
```

```
## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quanti
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#tail-ess
```

Problem 2: Compute the PSIS-LOO elpd values and the $\hat{k}$-values for each of the three models.

The PSIS-LOO elpd and p_loo values are computed using the following codes for each model. The $\hat{k}$-values are visualized in Problem 4 section to estimate the reliability of each model.

```
log_lik_separate <- extract_log_lik(fit_separate, merge_chains = FALSE)
r_eff_separate <- relative_eff(exp(log_lik_separate))
loo_separate <- loo(log_lik_separate, r_eff = r_eff_separate)
```

```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.
```

```
print(loo_separate)
```

```
##
## Computed from 4000 by 30 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo   -132.1 3.0
## p_loo         9.5 0.9
## looic       264.1 6.0
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     24    80.0%   851
##  (0.5, 0.7]   (ok)        3    10.0%   422
##    (0.7, 1]   (bad)       3    10.0%   207
##    (1, Inf)   (very bad)  0     0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
```

```
log_lik_pooled <- extract_log_lik(fit_pooled, merge_chains = FALSE)
r_eff_pooled <- relative_eff(exp(log_lik_pooled))
loo_pooled <- loo(log_lik_pooled, r_eff = r_eff_pooled)
print(loo_pooled)
```

```
##
## Computed from 4000 by 30 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo   -130.9 4.3
## p_loo         2.0 0.8
## looic       261.8 8.5
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
log_lik_hierarchical <- extract_log_lik(fit_hierarchical, merge_chains = FALSE)
r_eff_hierarchical <- relative_eff(exp(log_lik_hierarchical))
loo_hierarchical <- loo(log_lik_hierarchical, r_eff = r_eff_hierarchical)
```

```
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for deta
```

```
print(loo_hierarchical)
```

```
##
## Computed from 4000 by 30 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo   -127.4 4.5
## p_loo         5.6 1.5
## looic       254.7 9.1
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     28    93.3%   624
##  (0.5, 0.7]   (ok)        2     6.7%   276
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

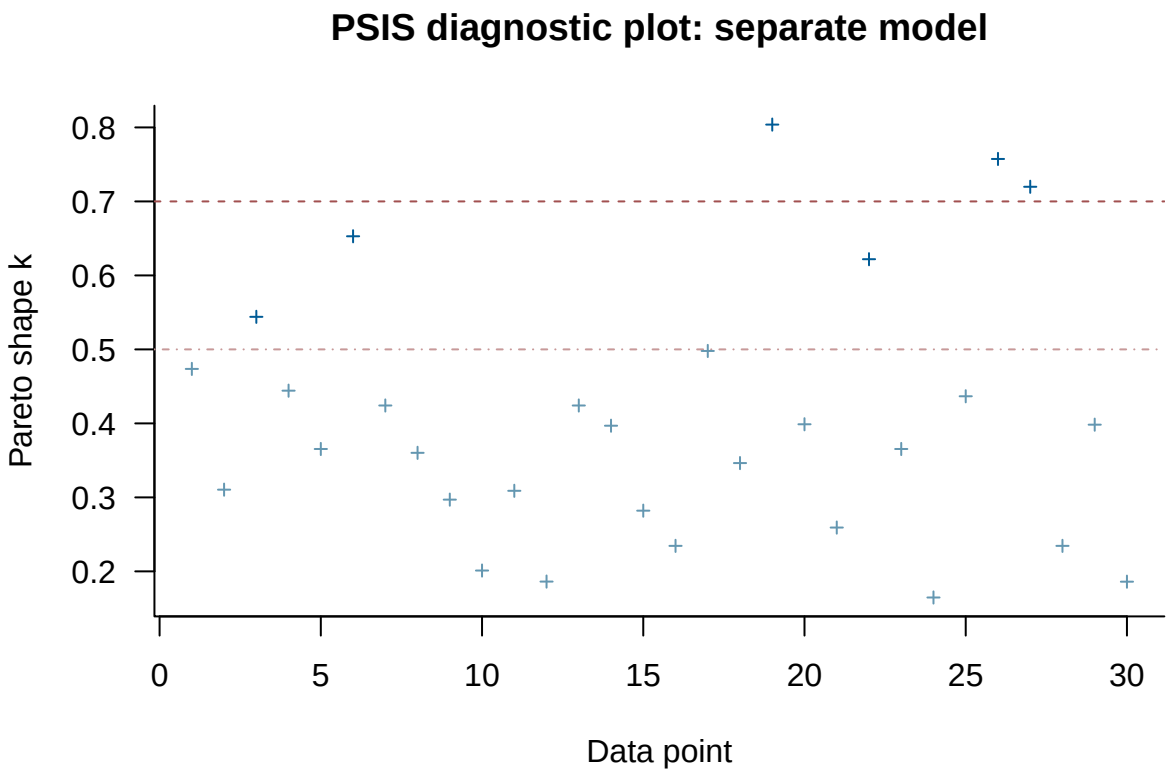
Problem 3: Compute the effective number of parameters p_{eff} for each of the three models.

The effective number of parameters can be calculated usinf $p_{eff} = lpd - elpd_{loo}$ which is already calculated in loo() function as p_loo. Therefore, p_{eff} is approximately 9.5 , 2 and 5.6 for separate, pooled and hierarchical model, respectively.

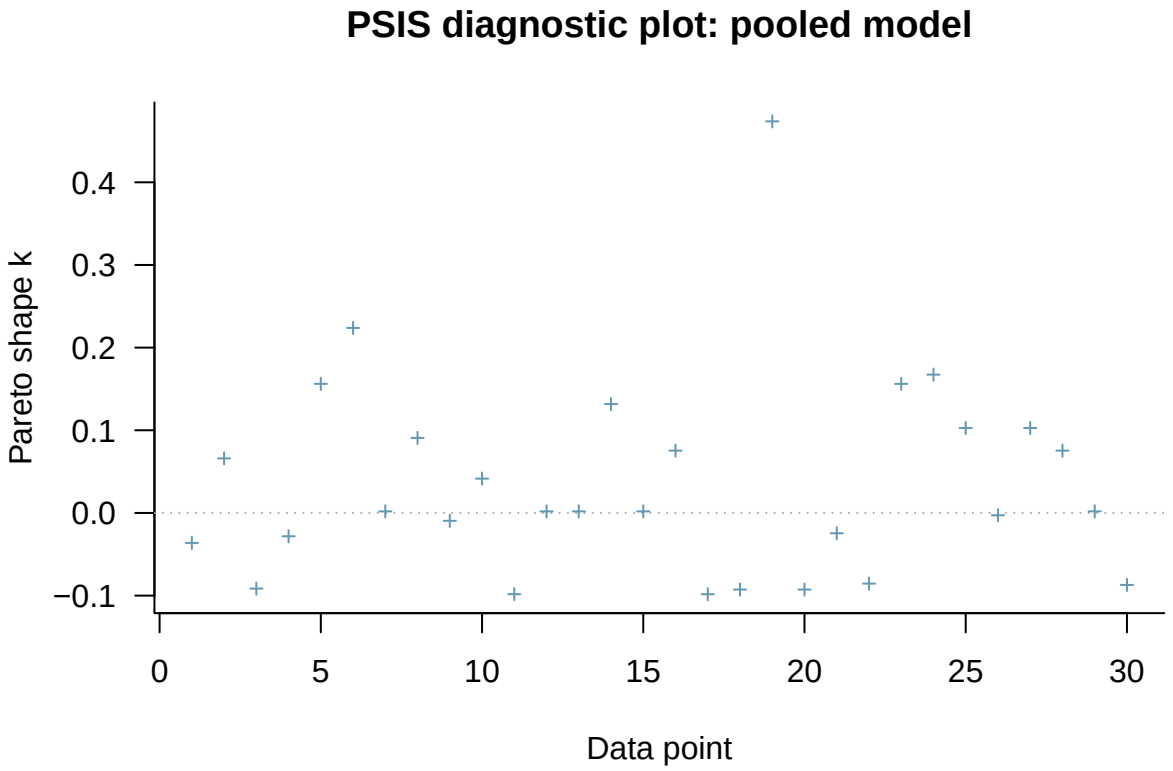
Problem 4: Assess how reliable the PSIS-LOO estimates are for the three models based on the $\hat{k}$-values.

In order to assess the reliability of PSIS-LOO estimates, I plot the PSIS diagnostic plot for three models that shows the $\hat{k}$-values for each observation. Pareto $\hat{k}$ estimates the tail shape and determines the convergence rate of PSIS. If these values are less than 0.7 they would be ok and the model would be reliable.

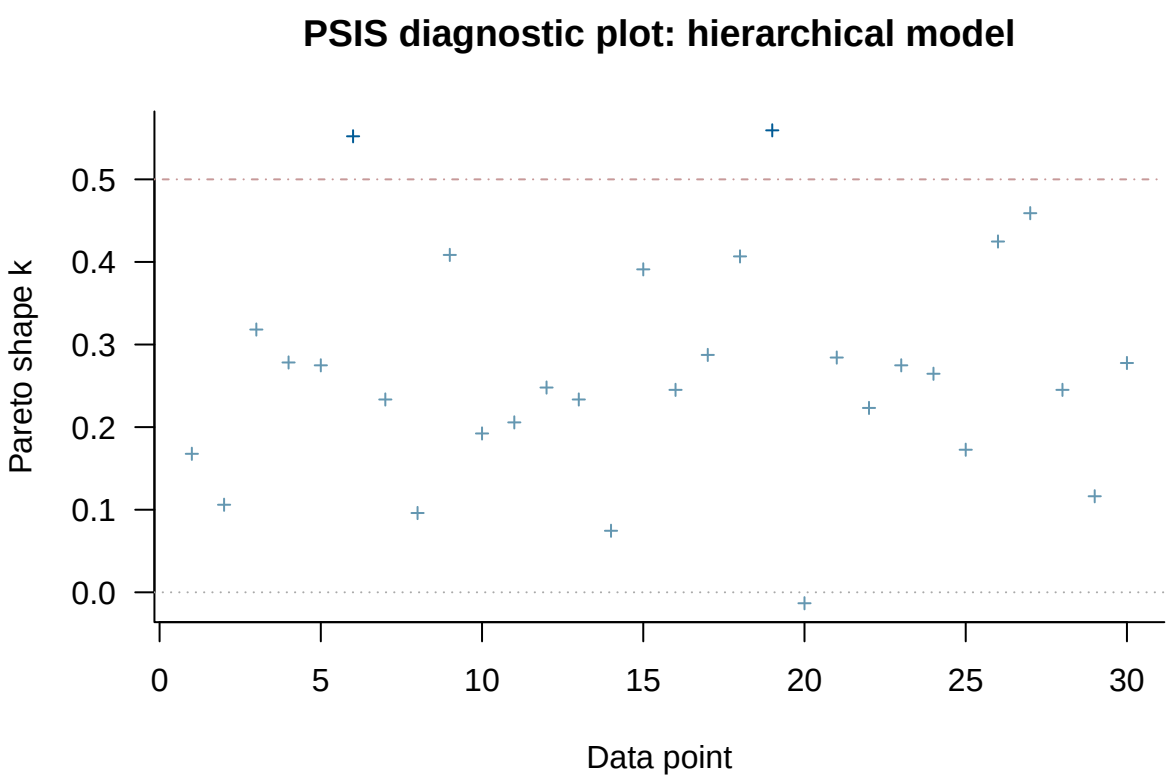
```
plot(loo_separate, diagnostic = c("k", "n_eff"),
  label_points = FALSE, main = "PSIS diagnostic plot: separate model")
```




```
plot(loo_pooled, diagnostic = c("k", "n_eff"),
  label_points = FALSE, main = "PSIS diagnostic plot: pooled model")
```




```
plot(loo_hierarchical, diagnostic = c("k", "n_eff"),
  label_points = FALSE, main = "PSIS diagnostic plot: hierarchical model")
```



As it can be seen from the plots, in the separate model there are two observations with pareto $\hat{k}$ values more than 0.7 which indicates that this model is not reliable. However, in the pooled model all the $\hat{k}$-values are less that or equal to 0.5 which means this model is reliable. In the hirerachical mdoel there are 3 $\hat{k}$-values which are $0.5 < \hat{k} < 0.7$ and therefore the hierachical model is also reliable.

Problem 5: An assessment of whether there are differences between the models with regard to the elpd loo-cv , and if so, which model should be selected according to PSIS-LOO.

We can now compare the models on LOO using the compare function. This object contains the estimated difference of expected leave-one-out prediction errors between the two models, along with the standard error:

```
compare_sep_pooled <- compare(loo_separate, loo_pooled)
print(compare_sep_pooled)
```

```
## elpd_diff        se
##       1.1       3.9
```

The positive difference in elpd (and its scale relative to the standard error) indicates a preference for the second model.

```
compare_sep_hier <- compare(loo_separate, loo_hierarchical)
print(compare_sep_hier)
```

```
## elpd_diff        se
##       4.7       3.1
```

```r
compare_pooled_hier <- compare(loo_pooled, loo_hierarchical)
print(compare_pooled_hier)
```

```
## elpd_diff        se
##       3.5       1.7
```

According to the above comparisons, the hierarchical model is a clear winner in the predictive performance and its PSIS-LOO has the highest value among the three models.