Assignment 6

Aki Vehtari et al.

1 General information

The maximum amount of points from this assignment is 6.

We have prepared a **quarto template specific to this assignment (html, qmd, pdf)** to help you get started.

Setup

We recommend Aalto students use jupyter.cs.aalto.fi, for all others we also provide a docker container.

Tip

Reading instructions:

- **The reading instructions for BDA3 Chapter 10**.
- **The reading instructions for BDA3 Chapter 11**.

Grading instructions:
The grading will be done in peergrade. All grading questions and evaluations for this assignment are contained within this document in the collapsible **Rubric** blocks.
Installing and using `CmdStanR`:
See the Stan demos on how to use Stan in R (or Python). Aalto JupyterHub has working R and CmdStanR/RStan environment and is probably the easiest way to use Stan. * To use CmdStanR in Aalto JupyterHub: `library(cmdstanr) set_cmdstan_path('/coursedata/cmdstan')`
The Aalto Ubuntu desktops also have the necessary libraries installed.
To install Stan on your laptop, run 'install.packages("cmdstanr", repos = c("https://mc-stan.org/r-packages/", getOption("repos")))' in R. If you encounter problems, see additional answers in **FAQ**. For Aalto students, if you don't succeed in short amount of time, it is probably easier to use Aalto JupyterHub.
If you use `Aalto JupyterHub`, all necessary packages have been pre-installed. In your laptop, install package `cmdstanr`. Installation instructions on Linux, Mac and Windows can be found at https://mc-stan.org/cmdstanr/. Additional useful packages are `loo`, `bayesplot` and `posterior` (but you don't need these in this assignment). For Python users, `PyStan`, `CmdStanPy`, and `ArviZ` packages are useful.
Stan manual can be found at https://mc-stan.org/users/documentation/. From this website, you can also find a lot of other useful material about Stan.
If you edit files ending `.stan` in RStudio, you can click "Check" in the

editor toolbar to make syntax check. This can significantly speed-up writing a working Stan model.

! Reporting accuracy

For posterior statistics of interest, only report digits that are not completely random based on the Monte Carlo standard error (MCSE).
Example: If you estimate $E(\mu) \approx 1.234$ with $\text{MCSE}(E(\mu)) = 0.01$, then the true expectation is likely to be between 1.204 and 1.264, it makes sense to report $E(\mu) \approx 1.2$.
See Lecture video 4.1, the chapter notes, and a case study for more information.

Further information

- The recommended tool in this course is R (with the IDE RStudio).
- Instead of installing R and RStudio on you own computer, see **how to use R and RStudio remotely**.
- If you want to install R and RStudio locally, download R and RStudio.
- There are tons of tutorials, videos and introductions to R and RStudio online. You can find some initial hints from **RStudio Education pages**.
- When working with R, we recommend writing the report using `quarto` and the provided template. The template includes the formatting instructions and how to include code and figures.
- Instead of `quarto`, you can use other software to make the PDF report, but the the same instructions for formatting should be used.
- Report all results in a single, **anonymous** *.pdf -file and submit it in **peergrade.io**.
- The course has its own R package `aaltobda` with data and functionality to simplify coding. The package is pre-installed in JupyterHub. To install the package on your own system, run the following code (upgrade="never" skips question about updating other packages):

```
install.packages("aaltobda", repos = c("https://avehtari.github.io/BDA_course_Aalto/", getOption("re
```

- Many of the exercises can be checked automatically using the R package `markmyassignment` (pre-installed in JupyterHub). Information on how to install and use the package can be found in the `markmyassignment` documentation. There is no need to include `markmyassignment` results in the report.
- Recommended additional self study exercises for each chapter in BDA3 are listed in the course web page. These will help to gain deeper understanding of the topic.
- Common questions and answers regarding installation and technical problems can be found in Frequently Asked Questions (FAQ).
- Deadlines for all assignments can be found on the course web page and in Peergrade. You can set email alerts for the deadlines in Peergrade settings.
- You are allowed to discuss assignments with your friends, but it is not allowed to copy solutions directly from other students or from internet.

- You can copy, e.g., plotting code from the course demos, but really try to solve the actual assignment problems with your own code and explanations.
- Do not share your answers publicly.
- Do not copy answers from the internet or from previous years. We compare the answers to the answers from previous years and to the answers from other students this year.
- Use of AI is allowed on the course, but the most of the work needs to by the student, and you need to report whether you used AI and in which way you used them (See points 5 and 6 in Aalto guidelines for use of AI in teaching).
- All suspected plagiarism will be reported and investigated. See more about the **Aalto University Code of Academic Integrity and Handling Violations Thereof**.
- Do not submit empty PDFs, almost empty PDFs, copy of the questions, nonsense generated by yourself or AI, as these are just harming the other students as they can't do peergrading for the empty or nonsense submissions. Violations of this rule will be reported and investigated in the same way was plagiarism.
- If you have any suggestions or improvements to the course material, please post in the course chat feedback channel, create an issue, or submit a pull request to the public repository!

Rubric

- Can you open the PDF and it's not blank nor nonsense? If the pdf is blank, nonsense, or something like only a copy of the questions, 1) report it as problematic in Peergrade-interface to get another report to review, and 2) send a message to TAs.
- Is the report anonymous?

This is the template for assignment 6. You can download the broken stan-file and the qmd-file or copy the code from this rendered document after clicking on `</> Code` in the top right corner.

Please replace the instructions in this template by your own text, explaining what you are doing in each exercise.

Setup

JupyterHub has all the needed packages pre-installed.
The following installs and loads the `aaltobda` package:

```
if(!require(aaltobda)){
    install.packages("aaltobda", repos = c("https://avehtari.github.io/BDA_course_Aalto/", getOption
    library(aaltobda)
}
```

```
Loading required package: aaltobda
```

The following installs and loads the `latex2exp` package, which allows us to use LaTeX in plots:

```
if(!require(latex2exp)){
    install.packages("latex2exp")
    library(latex2exp)
}
```

```
Loading required package: latex2exp
```

The following installs and loads the `posterior` package which imports the `rhat_basic()` function:

```
if(!require(posterior)){
    install.packages("posterior")
    library(posterior)
}
```

```
Loading required package: posterior

This is posterior version 1.6.0

Attaching package: 'posterior'

The following object is masked from 'package:aaltobda':

    mcse_quantile

The following objects are masked from 'package:stats':

    mad, sd, var

The following objects are masked from 'package:base':

    %in%, match
```

The following installs and loads the `ggplot2` package, the `bayesplot` package and the `dplyr` package

```
if(!require(ggplot2)){
    install.packages("ggplot2")
    library(ggplot2)
}
```

```
Loading required package: ggplot2
```

```
if(!require(bayesplot)){
    install.packages("bayesplot")
    library(bayesplot)
}
```

```
Loading required package: bayesplot

This is bayesplot version 1.11.1

- 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

Attaching package: 'bayesplot'

The following object is masked from 'package:posterior':

    rhat
```

```
if(!require(dplyr)){
    install.packages("dplyr")
    library(dplyr)
}
```

```
Loading required package: dplyr

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
if(!require(tidyr)){
    install.packages("tidyr")
    library(tidyr)
}
```

```
Loading required package: tidyr
```

```
# Some additional set-up to make plots legible
ggplot2::theme_set(theme_minimal(base_size = 14))
bayesplot::bayesplot_theme_set(theme_minimal(base_size = 14))
# register_knitr_engine()
```

The following installs and loads the `cmdstanr` package and tries to install `cmdstan`.

```
if(!require(cmdstanr)){
    install.packages("cmdstanr", repos = c("https://mc-stan.org/r-packages/", getOption("repos")))
    library(cmdstanr)
}
```

```
Loading required package: cmdstanr

This is cmdstanr version 0.8.1

- CmdStanR documentation and vignettes: mc-stan.org/cmdstanr
```

```
- CmdStan path: /home/runner/.cmdstan/cmdstan-2.35.0

- CmdStan version: 2.35.0
```

```r
cmdstan_installed <- function(){
  res <- try(out <- cmdstanr::cmdstan_path(), silent = TRUE)
  !inherits(res, "try-error")
}
if(!cmdstan_installed()){
    install_cmdstan()
}
```

2 Stan warm-up: linear model of BDA retention with Stan (2 points)

From 2018 to 2022, we have been keeping track of assignment submissions for the BDA course given the number of submissions for the 1st assignment. We will fit a simple linear model to answer two questions of interest:

- What is the trend of student retention as measured by assignment submissions?
- Given the submission rates for assignments 1–8, how many students will complete the final 9th assignment (and potentially pass the course)?

The author has given you the broken Stan code below, which they intend to encode the following linear model:

$$\begin{aligned} p(\alpha, \beta, \sigma) &= \text{const.} && \text{(improper flat prior) and} \\ p(y|x, \alpha, \beta, \sigma) &= p_{\text{normal}}(y|\alpha + \beta x, \sigma) && \text{(normal likelihood).} \end{aligned}$$

In both the statistical model above and in the Stan model below, $x \in \mathbb{R}^N$ and $y \in \mathbb{R}^N$ are vectors of the covariates / predictors (the assignment number) and vectors of the observation (proportions of students who have handed in the respective assignment). $\alpha \in \mathbb{R}$ is the unknown scalar intercept, $\beta \in \mathbb{R}$ is the unknown scalar slope and $\sigma \in \mathbb{R}_{>0}$ is the unknown scalar observation standard deviation. The statistical model further implies

$$p(y_{\text{pred.}}|x_{\text{pred.}}, \alpha, \beta, \sigma) = p_{\text{normal}}(y_{\text{pred.}}|\alpha + \beta x_{\text{pred.}}, \sigma)$$

as the predictive distribution for a new observation $y_{\text{pred.}}$ at a given new covariate value $x_{\text{pred.}}$.

You can download the broken stan file from github.

```
data {
    // number of data points
    int<lower=0> N;
    // covariate / predictor
    vector[N] x;
    // observations
    vector[N] y;
    // number of covariate values to make predictions at
    int<lower=0> no_predictions;
```

```
    // covariate values to make predictions at
    vector[no_predictions] x_predictions;
}
parameters {
    // intercept
    real alpha;
    // slope
    real beta;
    // the standard deviation should be constrained to be positive
    real<upper=0> sigma;
}
transformed parameters {
    // deterministic transformation of parameters and data
    vector[N] mu = alpha + beta * x // linear model
}
model {
    // observation model / likelihood
    y ~ normal(mu, sigma);
}
generated quantities {
    // compute the means for the covariate values at which to make predictions
    vector[no_predictions] mu_pred = alpha + beta * x_predictions;
    // sample from the predictive distribution, a normal(mu_pred, sigma).
    array[no_predictions] real y_pred = normal_rng(mu, sigma);
}
```

Tip

A normal linear model is actually **not** the best model to use for this type of data, but we will use it here to illustrate the first step in building up to more appropriate, complicated models.

Subtask 2.a

Find the ***three mistakes*** in the code and fix them. Report the original mistakes and your fixes clearly in your report. Include the **full** corrected Stan code in your report. Verify that sampling was successful.

Tip

You may find some of the mistakes in the code using Stan syntax checker. If you copy the Stan code to a file ending `.stan` and open it in RStudio (you can also choose from RStudio menu File→New File→Stan file to create a new Stan file), the editor will show you some syntax errors. More syntax errors might be detected by clicking 'Check' in the bar just above the Stan file in the RStudio editor. Note that some of the errors in the presented Stan code may not be syntax errors.

Write your answers/code here!

The author runs the corrected Stan file using the following R code and plots the returned MCMC sample. Read through the code below to understand what is being plotted.

Data preparation and sampling from the posterior

Data assembly happens here:

```
# These are our observations y: the proportion of students handing in each assignment (1-8),
# sorted by year (row-wise) and assignment (column-wise).
# While the code suggest a matrix structure,
# the result will actually be a vector of length N = no_years * no_assignments
propstudents<-c(c(176, 174, 158, 135, 138, 129, 126, 123)/176,
                c(242, 212, 184, 177, 174, 172, 163, 156)/242,
                c(332, 310, 278, 258, 243, 242, 226, 224)/332,
                c(301, 269, 231, 232, 217, 208, 193, 191)/301,
                c(245, 240, 228, 217, 206, 199, 191, 182)/245)
# These are our predictors x: for each observation, the corresponding assignment number.
assignment <- rep(1:8, 5)
# These are in some sense our test data: the proportion of students handing in the last assignment (
# sorted by year.
# Usually, we would not want to split our data like that and instead
# use e.g. Leave-One-Out Cross-Validation (LOO-CV, see e.g. http://mc-stan.org/loo/index.html)
# to evaluate model performance.
propstudents9 = c(121/176, 153/242, 218/332, 190/301, 175/245)
# The total number of assignments
no_assignments = 9
# The assignment numbers for which we want to generate predictions
x_predictions = 1:no_assignments
# (Cmd)Stan(R) expects the data to be passed in the below format:
model_data = list(N=length(assignment),
                  x=assignment,
                  y=propstudents,
                  no_predictions=no_assignments,
                  x_predictions=x_predictions)
```

Sampling from the posterior distribution happens here:

```
# This reads the file at the specified path and tries to compile it.
# If it fails, an error is thrown.
retention_model = cmdstan_model("./additional_files/assignment6_linear_model.stan")
```

```
Error in initialize(...): Assertion on 'stan_file' failed: File does not exist: './additional_files/
```

```
# This "out <- capture.output(...)" construction suppresses output from cmdstanr
# See also https://github.com/stan-dev/cmdstanr/issues/646
out <- capture.output(
    # Sampling from the posterior distribution happens here:
    fit <- retention_model$sample(data=model_data, refresh=0, show_messages=FALSE)
)
```

```
Error in eval(expr, envir, enclos): object 'retention_model' not found
```

Draws postprocessing happens here:

```
# This extracts the draws from the sampling result as a data.frame.
draws_df = fit$draws(format="draws_df")
```

```
Error in eval(expr, envir, enclos): object 'fit' not found
```

```
# This does some data/draws wrangling to compute the 5, 50 and 95 percentiles of
# the mean at the specified covariate values (x_predictions).
# It can be instructive to play around with each of the data processing steps
# to find out what each step does, e.g. by removing parts from the back like "|>  gather(pct,y,-x)"
# and printing the resulting data.frame.
mu_quantiles_df = draws_df |>
      subset_draws(variable = c("mu_pred")) |>
      summarise_draws(~quantile2(.x, probs = c(0.05, .5, 0.95))) |>
      mutate(x = 1:9) |>
      pivot_longer(c(q5, q50, q95), names_to = c("pct"))
```

```
Error in UseMethod("subset_draws"): no applicable method for 'subset_draws' applied to an object of
```

```
# Same as above, but for the predictions.
y_quantiles_df = draws_df |>
      subset_draws(variable = c("y_pred")) |>
      summarise_draws(~quantile2(.x, probs = c(0.05, .5, 0.95))) |>
      mutate(x = 1:9) |>
      pivot_longer(c(q5, q50, q95), names_to = c("pct"))
```

```
Error in UseMethod("subset_draws"): no applicable method for 'subset_draws' applied to an object of
```

Plotting happens here:

```
ggplot() +
  # scatter plot of the training data:
  geom_point(
    aes(x, y, color=assignment),
    data=data.frame(x=assignment, y=propstudents, assignment="1-8")
) +
  # scatter plot of the test data:
  geom_point(
    aes(x, y, color=assignment),
    data=data.frame(x=no_assignments, y=propstudents9, assignment="9")
) +
  # you have to tell us what this plots:
  geom_line(aes(x,y=value,linetype=pct), data=mu_quantiles_df, color='grey', linewidth=1.5) +
  # you have to tell us what this plots:
  geom_line(aes(x,y=value,linetype=pct), data=y_quantiles_df, color='red') +
  # adding xticks for each assignment:
  scale_x_continuous(breaks=1:no_assignments) +
  # adding labels to the plot:
  labs(y="assignment submission %", x="assignment number") +
  # specifying that line types repeat:
  scale_linetype_manual(values=c(2,1,2)) +
  # Specify colours of the observations:
  scale_colour_manual(values = c("1-8"="black", "9"="blue")) +
  # remove the legend for the linetypes:
  guides(linetype="none")
```

```
Error in eval(expr, envir, enclos): object 'mu_quantiles_df' not found
```

Quick check for sampling convergence

If your model is correctly implemented, sampling from the posterior distribution should have been successful. You can check whether Stan thinks that sampling succeeded by inspecting the output of the below command, which you should be able to interpret with a little help from the CmdStan User's Guide.

```
fit$cmdstan_diagnose()
```

```
Error in eval(expr, envir, enclos): object 'fit' not found
```

Based on the above plot, answer the following questions:

Subtask 2.b

- What is the solid red line plotting? What are the dashed red lines? How and why are these different from the corresponding grey lines?
- What is the general trend of student retention as measured by assignment submissions?
- Given a model fitted to the submission data for assignments 1-8, does it do a good job predicting the proportion of students who submit the final 9th assignment?
- Name one different modeling choice you could make to improve the prediction.

Write your answers/code here!

Rubric

- Is the source code included?
  - No
  - Yes
- Is the full resulting modified Stan model code presented in the report?
  - No
  - Yes, but partially
  - Yes, with a few mistakes
  - Yes, and it is correct
- Has the sampling success been verified/summarized (e.g. by inspecting and summarizing the output of `CmdStan`'s `diagnose` method).
  - No
  - Yes, but partially
  - Yes, and it is correctly verified
- Fix #1: Is there a fix for line .
  - It has not been discussed, that this line should be fixed.
  - It has been discussed, that this line should be fixed, but there is no fix presented for it or the fix is clearly wrong.
  - There is a fix presented for this line, that clearly solves the problem.
- Fix #2: Is there a fix for line .

  - It has not been discussed, that this line should be fixed.
  - It has been discussed, that this line should be fixed, but there is no fix presented for it or the fix is clearly wrong.
  - There is a fix presented for this line, that clearly solves the problem.

- Fix #3: Is there a fix for line .
  - It has not been discussed, that this line should be fixed.
  - It has been discussed, that this line should be fixed, but there is no fix presented for it or the fix is clearly wrong.
  - There is a fix presented for this line, that clearly solves the problem.
- Have the red lines been correctly described ()?
- Have the grey lines been correctly described ()?
- Has the difference between the red and grey lines been explained ()?
- Has the student retention trend been described ()
- Has the predictive performance for the held out data been discussed and assessed satisfactorily ()?
  - No
  - Somewhat
  - Yes
- Has at least one way to improve the model been mentioned (E.g.)?

3 Generalized linear model: Bioassay with Stan (4 points)

Replicate the computations for the bioassay example of section 3.7 (BDA3) using Stan.

Subtask 3.a

Write down the model for the bioassay data in Stan syntax. For instructions in reporting your implementation, you can refer to parts 2c.3 - 2c.6 in Assignment 5. Use the Gaussian prior as in Assignment 4 and 5, that is

$$\begin{bmatrix}\alpha\\ \beta\end{bmatrix} \sim \mathrm{N}\left(\mu_0, \Sigma_0\right), \qquad \text{where} \quad \mu_0 = \begin{bmatrix}0\\ 10\end{bmatrix} \quad \text{and} \quad \Sigma_0 = \begin{bmatrix}2^2 & 12\\ 12 & 10^2\end{bmatrix}.$$

Tip

You will need Stan functions `multi_normal` and `binomial_logit` for implementing the prior and observation model, respectively. In Stan code, it is easiest to declare a variable (say `theta`) which is a two-element vector so that the first value denotes α and latter one β. This is because the `multi_normal` function that you need for implementing the prior requires a vector as an input.

Write your answers/code here!

```
data("bioassay")
```

Subtask 3.b

Use $\widehat{R}$ for convergence analysis. You can either use Eq. (11.4) in BDA3 or the later version that can be found in **a recent article**. You should specify which $\widehat{R}$ you used. In R the best choice is to use function `rhat_basic()` or `rhat()` from the `posterior` package (see `?posterior::rhat_basic`). To check $\widehat{R}$ and other diagnostics, you can also call `fit$summary()`, where `fit` is the fit object returned by Stan's sampling function. Report the $\widehat{R}$ values both for α and β and discuss the convergence of the chains. **Briefly explain in your own words how to interpret the obtained $\widehat{R}$ values**.

Write your answers/code here!

Subtask 3.c

Plot the draws for α and β (scatter plot) and include this plot in your report. You can compare the results to Figure 3.3b in BDA3 to verify that your code gives sensible results. Notice though that the results in Figure 3.3b are generated from posterior with a uniform prior, so even when your algorithm works perfectly, the results will look slightly different (although fairly similar).

Write your answers/code here!

Subtask 3.d

To develop the course and provide feedback to Stan developers, we collect information on which Stan setup you used and whether you had any problems in setting it up or using it. Please report,

- Operating system (Linux, Mac, Windows) or jupyter.cs.aalto.fi?
- Programming environment used: R or Python?
- Interface used: RStan, CmdStanR, PyStan, or CmdStanPy?
- Did you have installation or compilation problems? Did you try first installing locally, but switched to jupyter.cs.aalto.fi?
- In addition of these you can write what other things you found out difficult (or even frustrating) when making this assignment with Stan.

Write your answers/code here!

Rubric

- Is the Stan model code included?
  - No
  - Yes
- Does the implemented Stan-model seem to be working?
  - No implementation
  - Model implemented but results not visualized/reported
  - Model implemented, but the results seem weird
  - Model seems to work correctly
- Are the R_hat-values reported (potential scale reduction factor, Eq.

(11.4) in the BDA3)?

  - No
  - Yes, but only for alpha or beta
  - Yes, single values both for alpha and beta

- Is the interpretation of R_hat values correct ()?

  - No interpretation or discussion about the R_hat-values, or conclusions clearly wrong
  - Somewhat correct
  - Interpretation correct

- Does the report contain a scatter plot about the draws? Do the results look reasonable, that is, roughly like in the Figure below ?

  - No plot included
  - Plot included, but the results do not look like in the figure above
  - Plot included, and the results look roughly like in the figure above

- Does the report contain description of Stan setup used and whether there were any problems in setting it up or using it?

  - No
  - Yes

- Even if the Stan model code is correct, there might be ways to give improve the layout or write the model in more elegant ways. This optional feedback box can be used to give additional suggestions for better Stan code.

4 Overall quality of the report

Rubric

- Does the report include comment on whether AI was used, and if AI was used, explanation on how it was used?

  - No
  - Yes

- Does the report follow the formatting instructions?

  - Not at all
  - Little
  - Mostly
  - Yes

- In case the report doesn't fully follow the general and formatting instructions, specify the instructions that have not been followed. If applicable, specify the page of the report, where this difference is visible. This will help the other student to improve their reports so that they are easier to read and review. If applicable, specify the page of the report, where this difference in formatting is visible.
- Please also provide feedback on the presentation (e.g. text, layout, flow of the responses, figures, figure captions). Part of the course is practicing making data analysis reports. By providing feedback on the report

presentation, other students can learn what they can improve or what they already did well. You should be able to provide constructive or positive feedback for all non-empty and non-nonsense reports. If you think the report is perfect, and you can't come up with any suggestions how to improve, you can provide feedback on what you liked and why you think some part of the report is better than yours.