Bayesian data analysis – Assignment 7

General information

- The recommended tool in this course is R (with the IDE R-Studio). You can download R **here** and R-Studio **here**. There are tons of tutorials, videos and introductions to R and R-Studio online. You can find some initial hints from **RStudio Education pages**.
- Instead of installing R and RStudio on you own computer, see **how to use R and RStudio remotely**.
- When working with R, we recommend writing the report using R markdown and the provided **R markdown template**. The remplate includes the formatting instructions and how to include code and figures.
- Instead of R markdown, you can use other software to make the PDF report, but the the same instructions for formatting should be used. These instructions are available also in **the PDF produced from the R markdown template**.
- Report all results in a single, **anonymous** *.pdf -file and return it to **peergrade.io**.
- The course has its own R package `aaltobda` with data and functionality to simplify coding. To install the package just run the following (upgrade=”never” skips question about updating other packages):
  1. `install.packages("remotes")`
  2. `remotes::install_github("avehtari/BDA_course_Aalto",`
     `subdir = "rpackage", upgrade="never")`
- Many of the exercises can be checked automatically using the R package `markmyassignment`. Information on how to install and use the package can be found **here**. 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.
- 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 trhe 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. 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 or almost empty PDFs as these are just harming the other students as they can’t do peergrading for the empty or almost empty

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!

Information on this assignment

This assignment is related to Chapter 5. The maximum amount of points from this assignment is 9.

Reading instructions: Chapter 5 in BDA3, see reading instructions **here**.

Grading instructions: The grading will be done in peergrade. All grading questions and evaluations for assignment 7 can be found **here**

Reporting accuracy: For posterior statistics of interest, only report digits for which the Monte Carlo standard error (MCSE) is zero. *Example:* If you estimate $E(\mu) = 1.234$ with $\text{MCSE}(E(\mu)) = 0.01$, you should report $E(\mu) = 1.2$.

Installing and using `rstan`: See the Stan demos on how to use Stan in R (or Python) . jupyter.cs.aalto.fi has working R and RStan environment and is probably the easiest way to use RStan. The Aalto Ubuntu desktops also have the necessary libraries installed. To install Stan on your laptop, see the instructions below and additional answers in **FAQ**. Recently there have been reports of installation problems with Windows and R 4.0 (see Stan discourse for more), so if you don't succeed in a short amount of time, it is probably easier to use jupyter.cs.aalto.fi.

If you use `jupyter.cs.aalto.fi`, all necessary packages have been pre-installed. In your laptop, install package `rstan`. Installation instructions on Linux, Mac and Windows can be found at `https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started`. Additional useful packages are `loo`, `bayesplot` and `shinystan` (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 `http://mc-stan.org/documentation/`. From this website, you can also find a lot of other useful material about Stan.

1. Linear model: drowning data with Stan (3p)

The provided data `drowning` in the `aaltobda` package contains the number of people who died from drowning each year in Finland 1980–2019. A statistician is going to fit a linear model with Gaussian residual model to these data using time as the predictor and number of drownings as the target variable (see the related linear model example for the Kilpisjärvi-temperature data in the example Stan codes). She has two objective questions:

i) What is the trend of the number of people drowning per year? (We would plot the histogram of the slope of the linear model.)

ii) What is the prediction for the year 2020? (We would plot the histogram of the posterior predictive distribution for the number of people drowning at $\tilde{x} = 2020$.)

To access the data, use:

```
> library(aaltobda)
> data("drowning")
```

Corresponding Stan code is provided in Listing 1. However, it is not entirely correct for the problem. First, there are *three mistakes.* Second, there are no priors defined for the parameters. In Stan, this corresponds to using uniform priors.

Your tasks are the following:

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.

   Hint: 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.

b) Determine a suitable weakly-informative prior normal$(0, \sigma_\beta)$ for the slope `beta`. It is very unlikely that the mean number of drownings changes more than 50 % in one year. The approximate historical mean yearly number of drownings is 138. Hence, set σ_β so that the following holds for the prior probability for `beta`: $\Pr(-69 <$ `beta` $< 69) = 0.99$. Determine suitable value for σ_β and report the approximate numerical value for it.

c) Using the obtained σ_β, add the desired prior in the Stan code. In the report, in a separate section, indicate clearly how you carried out your prior implementation, e.g. "Added line … in block …".

d) In a similar way, add a weakly informative prior for the intercept `alpha` and explain how you chose the prior.

Hint! Example resulting plots for the problem, with the fixes and the desired prior applied, are shown in Figure 1. If you want, you can use these plots as a reference for testing if your modified Stan code produces similar results. However, running the inference and comparing the plots is not required.

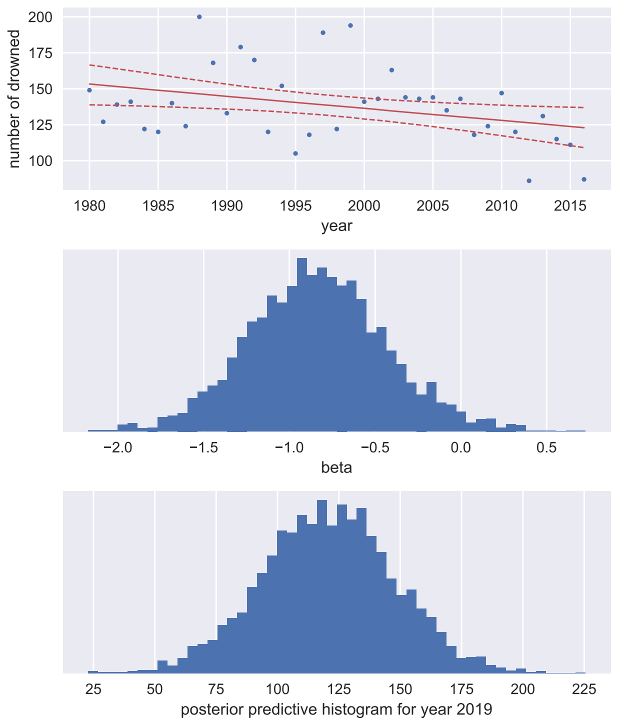

Figure 1: Example plots for the results obtained for the problem in the Question 1. In the first subplot, the red lines indicate the resulting 5 %, 50 %, and 95 % posterior quantiles for the transformed parameter `mu` at each year.

Listing 1: Broken Stan code for question 1

```
data {
    int<lower=0> N;  // number of data points
    vector[N] x;     // observation year
    vector[N] y;     // observation number of drowned
    real xpred;      // prediction year
}
parameters {
    real alpha;
    real beta;
    real<upper=0> sigma;
}
transformed parameters {
    vector[N] mu = alpha + beta*x;
}
model {
    y ~ normal(mu, sigma)
}
generated quantities {
    real ypred = normal_rng(mu, sigma);
}
```

2. Hierarchical model: factory data with Stan (3p)

Note! Both Assignment 8 and 9 build upon this part of the assignment, so it is important to get this assignment correct before you start with Assignment 8 and 9.

The `factory` data in the `aaltobda` package contains quality control measurements from 6 machines in a factory (units of the measurements are irrelevant here). In the data file, each column contains the measurements for a single machine. Quality control measurements are expensive and time-consuming, so only 5 measurements were done for each machine. In addition to the existing machines, we are interested in the quality of another machine (the seventh machine). To read in the data, just use:

```
> library(aaltobda)
> data("factory")
```

For this problem, you'll use the following Gaussian models:

- a separate model, in which each machine has its own model
- a pooled model, in which all measurements are combined and there is no distinction between machines
- a hierarchical model, which has a hierarchical structure as described in BDA3 Section 11.6

As in the model described in the book, use the same measurement standard deviation σ for all the groups in the hierarchical model. In the separate model, however, use separate measurement standard deviation σ_j for each group j. You should use weakly informative priors for all your models.

The provided Stan code in Listing 2 given on the next page is an example of the separate model (but with very strange results, why?). This separate model can be summarized mathematically as:

$$\begin{aligned} y_{ij} &\sim N(\mu_j, \sigma_j) \\ \mu_j &\sim N(0, 1) \\ \sigma_j &\sim \text{Inv-}\chi^2(10) \end{aligned}$$

To run Stan for that model, simply use:

```
> data("factory")
> sm <- rstan::stan_model(file = "[path to stan model code]")
> stan_data <- list(
    y = factory,
    N = nrow(factory),
    J = ncol(factory)
  )
> model <- rstan::sampling(sm, data = stan_data)
> model
```

```
Inference for Stan model: 5cbfa723dd8fb382e0b647b3943db079.
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%
mu[1]       0.11    0.01  0.98   -1.81    -0.56    0.12    0.77
mu[2]       0.10    0.01  1.00   -1.86    -0.56    0.10    0.79
...
```

Note! These are *not* the results you would expect to turn in your report. You will need to change the code for the separate model as well.

For *each of the three models* (separate, pooled, hierarchical), your tasks are the following:

a) Describe the model with mathematical notation (as is done for the separate model above). Also describe in words the difference between the three models.

b) Implement the model in Stan and include the code in the report. Use weakly informative priors for all your models.

c) Using the model (with weakly informative priors) report, comment on and, if applicable, plot histograms for the following distributions:

   i) the posterior distribution of the mean of the quality measurements of the sixth machine.

   ii) the predictive distribution for another quality measurement of the sixth machine.

   iii) the posterior distribution of the mean of the quality measurements of the seventh machine.

d) Report the posterior expectation for μ_1 with a 90% credible interval but using a normal$(0, 10)$ prior for the μ parameter(s) and a Gamma$(1, 1)$ prior for the σ parameter(s). For the hierarchical model, use the normal$(0, 10)$ and Gamma$(1, 1)$ as hyper-priors.

Hint! See the example Stan codes **here** for the comparison of k groups with and without the hierarchical structure.

Listing 2: Stan code for a bad separate model

```
data {
  int<lower=0> N;
  int<lower=0> J;
  vector[J] y[N];
}

parameters {
  vector[J] mu;
  vector<lower=0>[J] sigma;
}

model {
  // priors
  for (j in 1:J){
    mu[j] ~ normal(0, 1);
    sigma[j] ~ inv_chi_square(10);
  }

  // likelihood
  for (j in 1:J)
    y[,j] ~ normal(mu[j], sigma[j]);
}

generated quantities {
  real ypred;
  // Compute predictive distribution
  // for the first machine
  ypred = normal_rng(mu[1], sigma[1]);
}
```