HW05 Solutions

BJ

2022-03-08

Problem 1.

Install and verify the software package Stan and the library rstan, on your personal computer. Instructions for doing this are appended at the end of this hw assignment. Then run the following short example to verify that rstan is intalled and working correctly (note that the text string assigned to the variable "model' ' runs over several lines; that's perfectly OK):

```
set.seed(1234567890) ## to make a reproducible simulation result

library(rstan)
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)
library(ggplot2); theme_set(theme_bw())
library(bayesplot)
```

```
text_model <- "
   data {
     real y_mean;
   }
   parameters {
     real y;
   }
   model {
     y ~ normal(y_mean,1);
   }"
```

This Stan program is not doing anything Bayesian, it's just making draws from a normal distribution with mean=0 and sd=1 (fit) or mean 5 and sd=1 (fit2).

Problem 1(a).

Turn in the output for both `print(fit)` (or just `fit`), and `summary(fit)`. Write a sentence or two explaining the difference between the two outputs (besides a different number of decimal places printed!).

```
compiled_model <- stan(model_code=text_model,data=list(y_mean=0),chains=0)
```

```
## the number of chains is less than 1; sampling not done
```

```
fit <- stan(fit=compiled_model,data=list(y_mean=0))
fit2 <- stan(fit=compiled_model,data=list(y_mean=5))
```

```
## fit or print(fit)
fit
```

```
## Inference for Stan model: bbeee10d20bb93495a9498b5824cef40.
## 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
## y    -0.04    0.02 0.98 -1.93 -0.71 -0.04  0.63  1.86  1536    1
## lp__ -0.48    0.02 0.69 -2.35 -0.62 -0.22 -0.05  0.00  1693    1
##
## Samples were drawn using NUTS(diag_e) at Tue Mar 08 14:01:53 2022.
## 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).
```

```
## summary(fit)
summary(fit)
```

```
## $summary
##             mean    se_mean        sd      2.5%        25%         50%
## y    -0.04261205 0.02493659 0.9774397 -1.928903 -0.7110062 -0.03694378
## lp__ -0.47848265 0.01684413 0.6930562 -2.351296 -0.6210165 -0.21902096
##              75%         97.5%    n_eff     Rhat
## y     0.63078879  1.8567974589 1536.406 1.001139
## lp__ -0.04948021 -0.0004965297 1692.933 1.000111
##
## $c_summary
## , , chains = chain:1
##
##          stats
## parameter         mean        sd      2.5%        25%         50%         75%
##      y    -0.006966561 0.9876578 -1.973039 -0.6825558  0.01208248  0.69318011
##      lp__ -0.487270530 0.7146902 -2.355388 -0.6185470 -0.23546790 -0.05524952
##          stats
## parameter         97.5%
##      y     1.8030643465
##      lp__ -0.0003349395
##
## , , chains = chain:2
##
##          stats
## parameter       mean        sd      2.5%        25%        50%         75%
##      y    -0.1336909 0.9921667 -1.949433 -0.8304396 -0.1457799  0.52342893
##      lp__ -0.5006418 0.7150150 -2.236020 -0.6848631 -0.2333959 -0.05505554
##          stats
## parameter         97.5%
##      y     1.7369456923
##      lp__ -0.0006828667
##
## , , chains = chain:3
##
##          stats
## parameter        mean        sd      2.5%        25%         50%         75%
##      y     0.02038783 0.9547573 -1.671558 -0.6651583 -0.02263061  0.68167440
##      lp__ -0.45553282 0.6295748 -2.195822 -0.6293893 -0.22483936 -0.05404659
##          stats
## parameter         97.5%
##      y     1.9468808498
##      lp__ -0.0008556265
##
## , , chains = chain:4
##
##          stats
## parameter        mean        sd      2.5%        25%         50%         75%
##      y    -0.05017861 0.9692226 -2.016786 -0.6216174 -0.02536212  0.59628723
##      lp__ -0.47048545 0.7093640 -2.479137 -0.5769040 -0.18858462 -0.03668036
##          stats
## parameter        97.5%
##      y     1.807814594
##      lp__ -0.000319429
```

Here's a comparison of the two outputs:

- `print(fit)` *just gives a summary of the estimated parameters,* `se_mean` *from the Monte Carlo sumualtion, the posterior* `sd`*, quantiles or the posterior distribution, and* `n_eff` *&* `Rhat` *values, for the "kept" part of all the chains, merged together.*
- `summary(fit)` *gives the same overall summary, as a two-dimensional array, in the* `$summary` *component, and then in* `$c_summary` *it provides a 3-dimensional array containing summaries for each individual Markov chain that was run (but without the* `n_eff` *&* `Rhat` *values).*

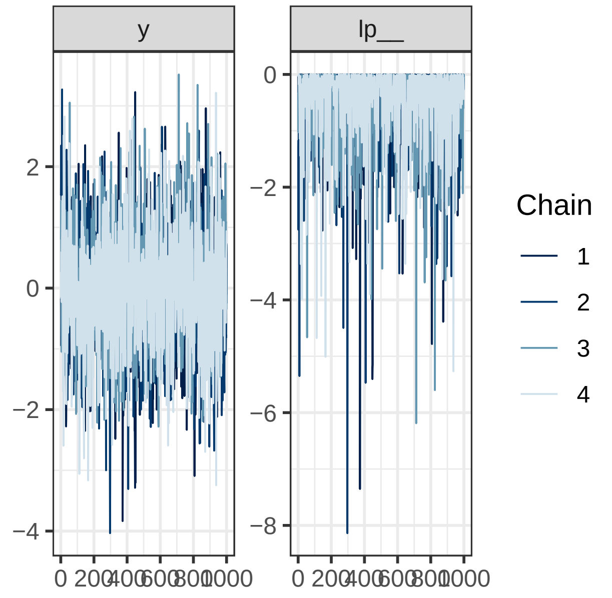
  (The two-dimensional array in `summary(fit)$summary` *is convenient for extracting mean, sd, quantiles, etc. to make CI's. etc.)*

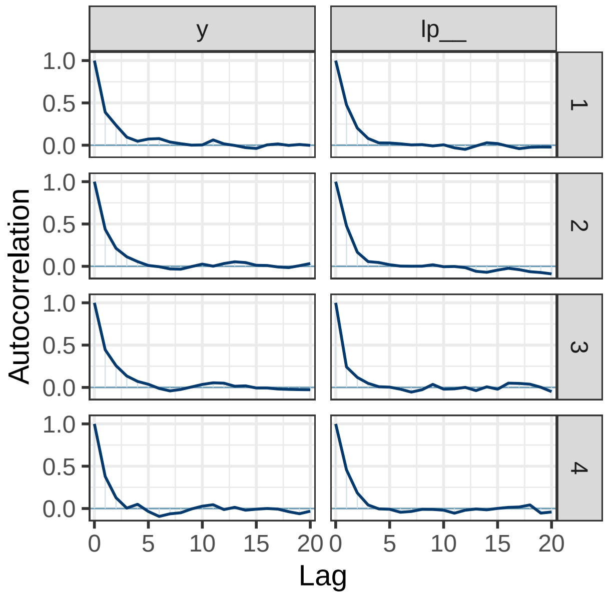
Problem 1(b).

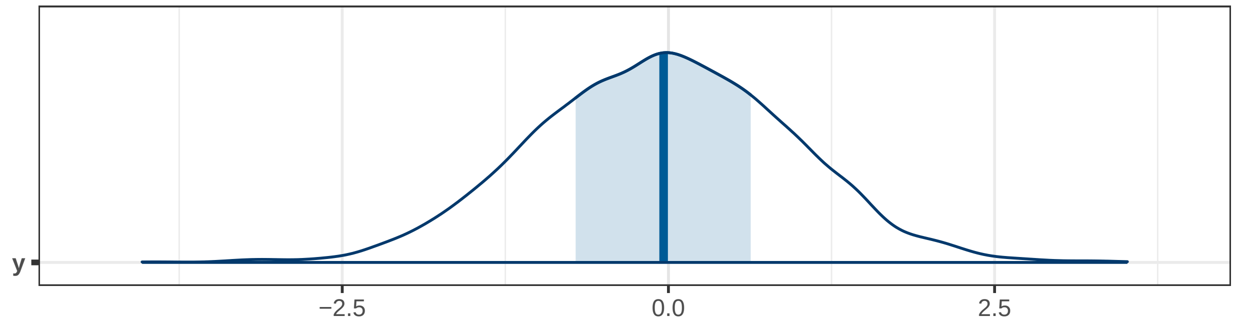
Use commands like those in the appendix (from `library(bayesplot)`) of lecture 13 to produce and turn in, for the fitted model `fit`:

- Trace plots of `y` (the variable being simulated) and `lp__` (the value of the log-posterior (in this case, not a posterior but just the normal density) at each simulated value of `y`).
- Autocorrelation plots of `y` and `lp__`.
- A histogram of just `y`.

```
library(gridExtra)
g1 <- mcmc_trace(fit)
g2 <- mcmc_acf(fit)
g3 <- mcmc_areas(fit,pars="y")
grid.arrange(arrangeGrob(g1,g2,ncol=2),g3,heights=c(2.5/4,1.4/4),ncol=1)
```





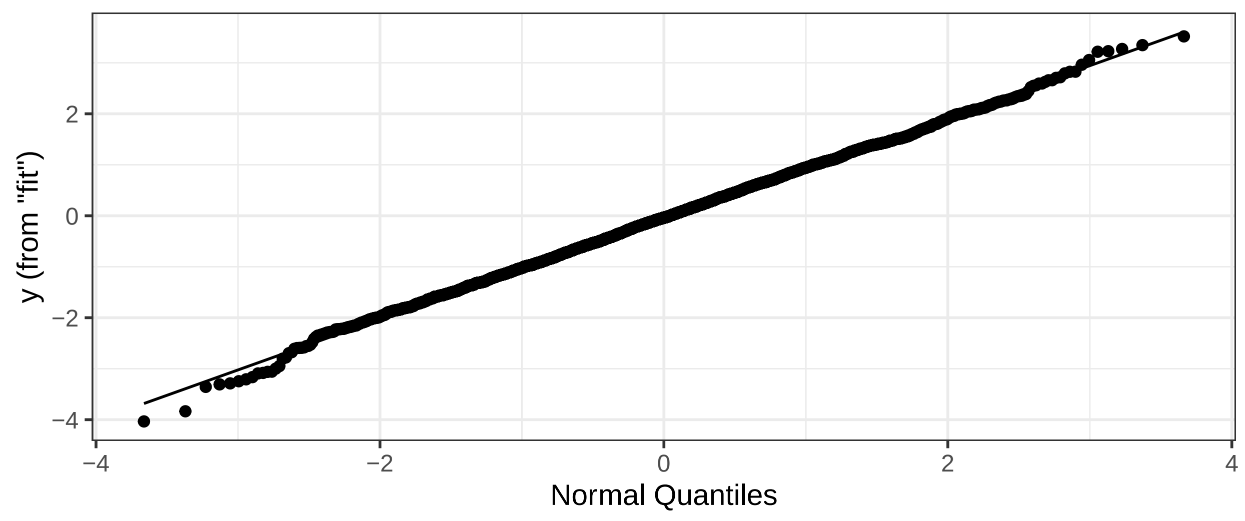


Problem 1(c).

Use the `extract` function to extract simulated values for `y` from `fit` and make a qqplot of the values against the Normal distribution. Is it plausible that the simulated values for `y` are normally distributed?

```r
sim <- extract(fit)

ggplot(as.data.frame(sim),aes(sample=y)) +
  geom_qq() +
  geom_qq_line() +
  xlab("Normal Quantiles") +
  ylab("y (from \"fit\")")
```



Yes, given this plot, it's pretty plausibly that the y's are normally distributed. There's just a hint of a deviation from the normal in the tails (where there's very little sample size, so it's hard to say these are "significant" deviations), but overall the data follow the normal distribution well.

Problem 1(d).

Print out and turn in output from `print(fit)` and `print(fit2)`. Write a sentence noting any similarities or differences between them.

```
fit
```

```
## Inference for Stan model: bbeee10d20bb93495a9498b5824cef40.
## 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
## y    -0.04    0.02 0.98 -1.93 -0.71 -0.04  0.63  1.86  1536    1
## lp__ -0.48    0.02 0.69 -2.35 -0.62 -0.22 -0.05  0.00  1693    1
##
## Samples were drawn using NUTS(diag_e) at Tue Mar 08 14:01:53 2022.
## 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).
```

```
fit2
```

```
## Inference for Stan model: bbeee10d20bb93495a9498b5824cef40.
## 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
## y     4.96    0.03 0.99  2.96  4.31  4.95  5.64  6.82  1274    1
## lp__ -0.49    0.02 0.69 -2.48 -0.65 -0.22 -0.05  0.00  1849    1
##
## Samples were drawn using NUTS(diag_e) at Tue Mar 08 14:02:03 2022.
## 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).
```

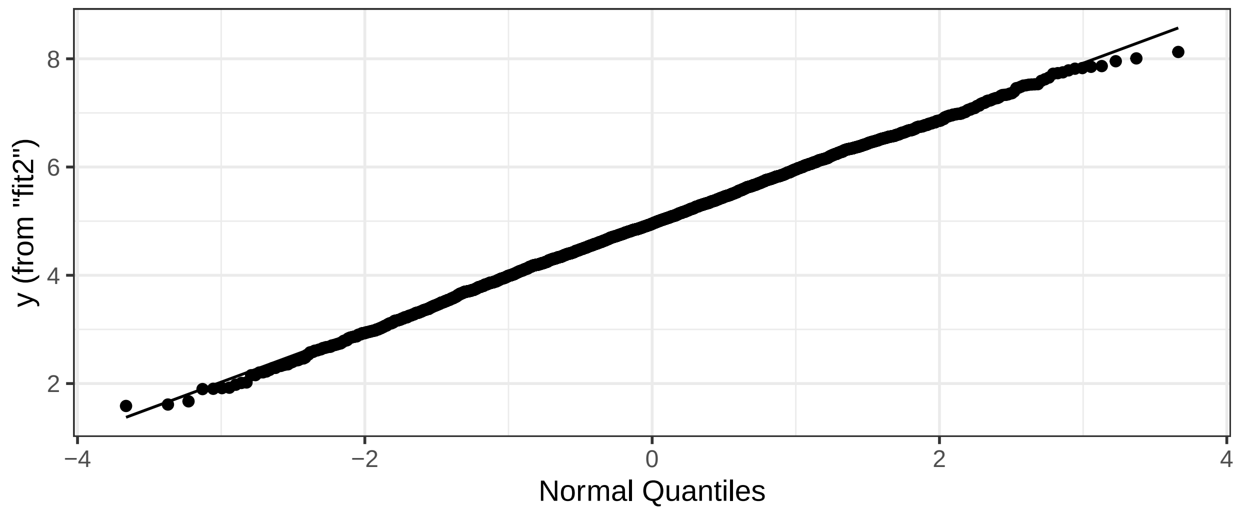
Everything is pretty similar for the two outputs, except of course the estimated mean (as either a posterior mean or a posterior median) is ≈ 0 for `fit` *and it is ≈ 5 for* `fit2`*. The posterior quantiles are similarly shifted by about 5 units.*

Problem 1(e).

Use the `extract` function to extract simulated values for `y` from `fit2` and make a qqplot of the values against the Normal distribution. Is it plausible that the simulated values for `y` are normally distributed?

```
sim <- extract(fit2)

ggplot(as.data.frame(sim),aes(sample=y)) +
  geom_qq() +
  geom_qq_line() +
  xlab("Normal Quantiles") +
  ylab("y (from \"fit2\")")
```



Again this qq plot is pretty convincing of normality. There's just a hint of a deviation from the normal in the tails (where there's very little sample size, so it's hard to say these are "significant" deviations). Note that the y=intercept for the fitted line is now about 5, corresponding to the fact that we are simulating from a $N(5, 1)$ instead of a $N(0, 1)$ this time.

Problem 2.

Problem 2(c)(iv) on hw #04 asked you to generate a point estimate and 95% CI for the model

$$\begin{aligned} y_i &\sim Poiss(\lambda),\ i = 1, 2, \ldots, n \\ \lambda &\sim Gamma(\alpha, \beta) \end{aligned}$$

with $\alpha = 1$ and $\beta = 1$, for the $n = 16$ data values

$$3, 7, 1, 7, 6, 1, 5, 4, 6, 2, 11, 3, 1, 6, 4, 4$$

Problem 2(a).

Write a stan program that specifies the likelihood and prior for this model, and use it to simulate draws from the posterior. You can use the default number of chains (4) and samples in each chain (2000). Include in your solutions:

- Your stan program
- your R code
- the output from `print()`'ing fitted stan simulation.

Hint: You will find the documentation at

https://mc-stan.org/docs/2_29/functions-reference/index.html

useful, especially the sections on Discrete and Continuous Distributions. You may also find this documentation useful

https://mc-stan.org/docs/2_29/stan-users-guide/index.html

https://mc-stan.org/docs/2_29/reference-manual/index.html

and, of course, you should imitate examples given in class to save yourself time, wherever possible.

```
## Either of the following Stan programs works. The only difference is whether you
## let stan "vectorize" the part of the code that establishes the poisson distribution
## for each observation y[i], or you do it yourself with a "for" loop...

text_model <- "
data {
  int n;
  int y[n];
}
parameters {
  real lambda;
}
model {
  y ~ poisson(lambda);   // likelihood
  lambda ~ gamma(1,1);   // prior
}"

text_model <- "
data {
  int n;
  int y[n];
}
parameters {
  real lambda;
}
model {
  for (i in 1:n) {       // likelihood
    y[i] ~ poisson(lambda);
  }
  lambda ~ gamma(1,1);   // prior
}"

## Then we compile and run the model as follows:

compiled_model <- stan(model_code=text_model,
                       data=list(n=2,y=c(1,2)),
                       chains=0)
```

```
## the number of chains is less than 1; sampling not done
```

```r
fit <- stan(fit=compiled_model,
            data=list(y=c(3,7,1,7,6,1,5,4,6,2,11,3,1,6,4,4),
                      n=16))

print(fit)
```

```
## Inference for Stan model: 20a1071517807ce6d551bf29a7d7efae.
## 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
## lambda  4.23    0.01 0.49  3.32  3.88  4.21  4.55  5.25  1404    1
## lp__   30.00    0.02 0.69 28.02 29.85 30.26 30.44 30.49  1688    1
##
## Samples were drawn using NUTS(diag_e) at Tue Mar 08 14:02:45 2022.
## 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).
```

Problem 2(b).

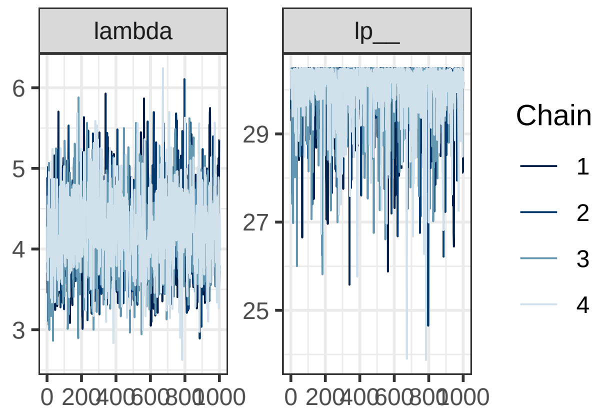
Use functions in `library(bayesplot)` to produce and turn in graphs of

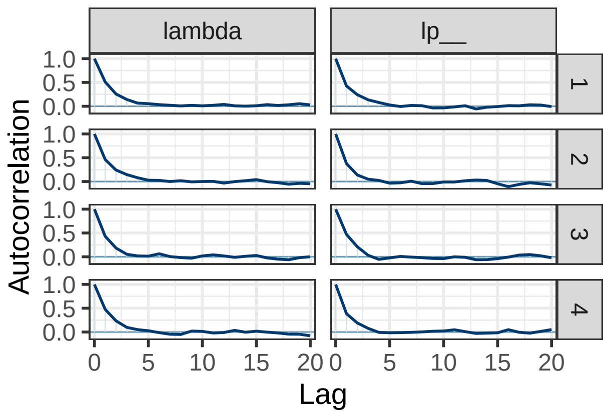
- trace plots
- acf plots
- rhat plots
- neff (ratio) plots

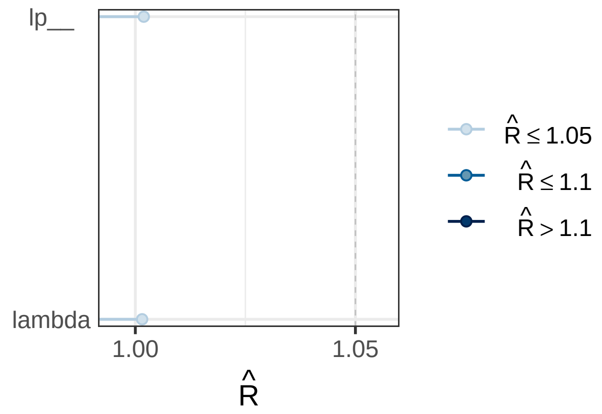
for your fitted simulation.

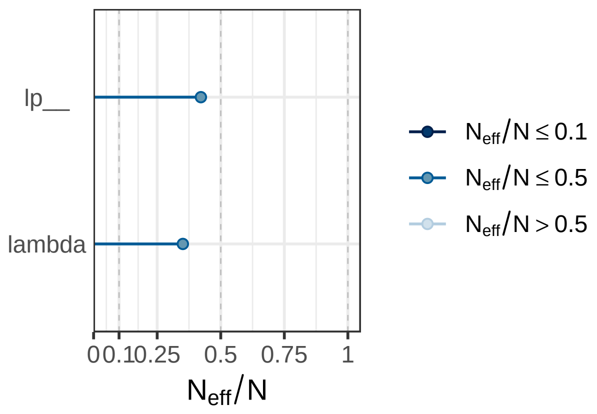
```r
g1 <- mcmc_trace(fit)
g2 <- mcmc_acf(fit)
g3 <- mcmc_rhat(rhat(fit)) + yaxis_text()
g4 <- mcmc_neff(neff_ratio(fit)) + yaxis_text()

grid.arrange(g1,g2,g3,g4,ncol=2)
```









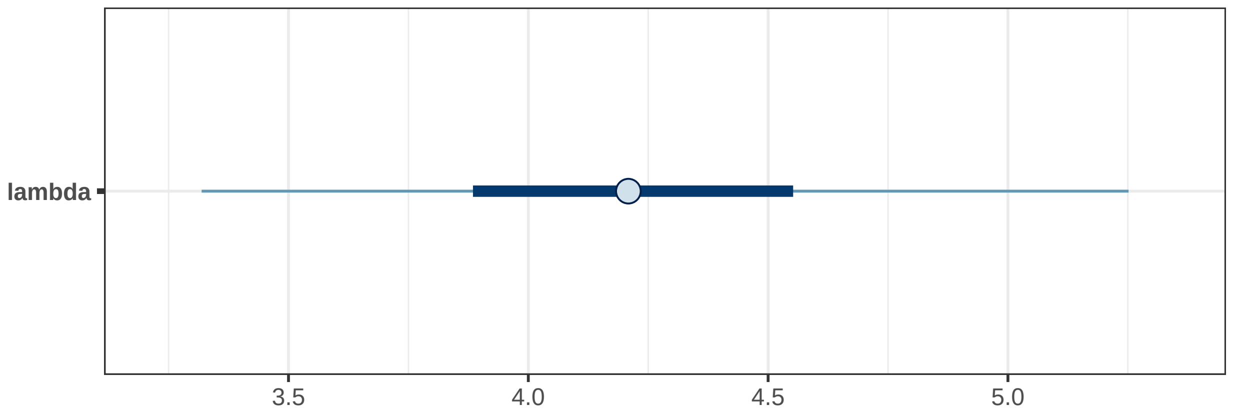
Problem 2(c).

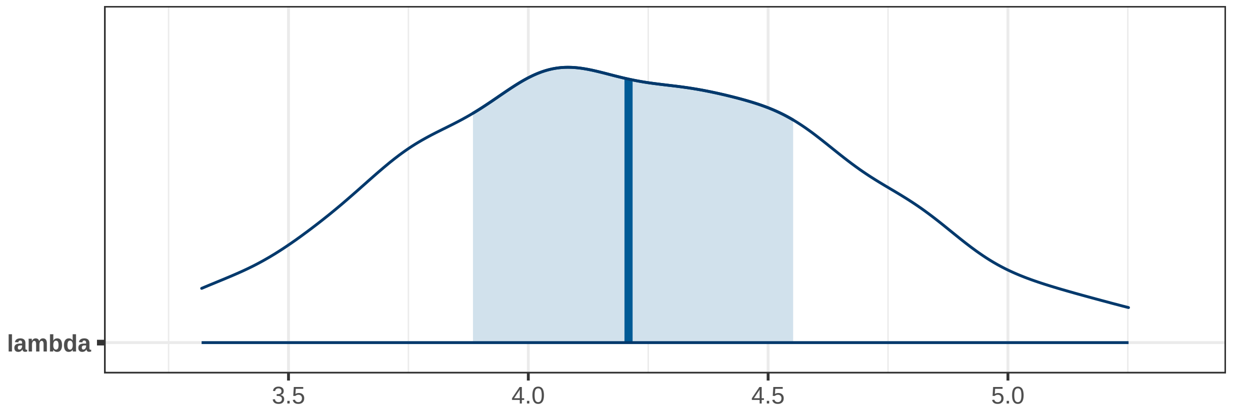
Use functions in `library(bayesplot)` to produce and turn in graphs of

- A 95% CI for λ
- A density plot for λ, with the 95% CI shaded in underneath it.

Compare your answer here with the answers you got on hw04 problem #2.

```
g1 <- mcmc_intervals(fit,pars="lambda",prob_outer=0.95)
g2 <- mcmc_areas(fit,pars="lambda",prob_outer=0.95)
grid.arrange(g1,g2,ncol=1)
```





Here are the intervals and point estimates from HW04:

```
## sample avg and sd:
round(c(xbar + c(-2,0,2)*stdev/sqrt(n)),2)
## [1] 3.08 4.44 5.79
## MLE:
round(c(lambda_hat + c(-2,0,2)*se_lambda_hat),2)
## [1] 3.38 4.44 5.49
## glm:
round(exp(c(beta_hat + c(-2,0,2)*se_beta_hat)),2)
## [1] 3.50 4.44 5.63
## gamma posterior, equal-tailed:
round(qgamma(c(0.025,0.5,0.975),astar,bstar),2)
## [1] 3.31 4.22 5.27
## gamma posterior, wald interval:
round(c(gamma_mean + c(-2,0,2)*gamma_sd),2)
## [1] 3.24 4.24 5.23
```

The interval from our simulation,

```
round(summary(fit)$summary[1,c(4,6,8)],2)
```

```
##  2.5%   50% 97.5%
##  3.32  4.21  5.25
```

is very similar to the interval we calculated using quantiles of the Gamma distribution. (This isn't at all suprising, since the posterior distribution we are simulating from with stan is exactly the same Gamma distribution we calculated for hw04).

Problem 3.

In the files area of Canvas there is a folder "presentation project ideas' '.

Problem 3(a).

Read the "overview.pdf' ' file in that folder, and browse through the subfolders there[1]. If you might have a project of your own to present, think about that too. Turn in your first, second and third choices for projects (these are tentative, not final, choices!). Also, indicate tentative team members if you are organizing a team.

If you list 3 ideas for presentation projects, and list team members (if you are part of a team), you will get a full 10 pts for this part.

Problem 3(b).

Sign up for a time to meet me in the week after spring break to talk about your presentation project. (You don't have to turn anything in for this, but it will be counted as part of your project grade.)

You get 10 automatic free points on the hw for this part; you don't have to turn anything in for these 10 pts!

I will separately give you a score in the "presentation projects" part of the gradebook for our meeting to discuss which presentation project you will actually do.

[1]If there is only a readme.txt file there now, it means I am still organizing and uploading content. It should all be there by Wednesday evening.

Installing Stan for R

1. Go to https://mc-stan.org/users/interfaces/rstan
   (can also find by googling "rstan").
2. Click on the "RStan Quick Start Guide" under "Download and Get Started"

   (*Note:* We will be installing Stan to run in R. There are also installations for python, matlab, julia, stata, mathematica, scala, etc... If you are interested, see https://mc-stan.org/ for details.)
3. Follow the instructions for configuring the `C++` Toolchain on your computer
   - This is the trickiest part of installing Stan in R on your computer.
   - There are separate instructions for Windows, Mac, & Linux
   - Make sure you are using R version 4.0 or higher. You may want to update Rstudio to the current version as well (you need at least version 1.2.5042).
   - **MAKE SURE YOU ARE** following the instructions for your version of R (version 4.0 or higher) not an earlier version of R such as version 3.6, etc.
4. Follow the instructions under "Installation of RStan" and "Verify Installation"
   - The most important parts are the "install.packages" command, to actually do the installation, and the "example" command to verify that your installation is good. If you can get these two commands to run successfully, you are golden.
     - Note: the "example" command will take from several seconds to up to a minute or so to run. It will also generate a ton of output that may not make much sense right now... Be patient...
   - Please note the other instructions under these two headings, which handle some common special cases that you may need to deal with on your laptop.
5. After you have verified that your installation is good, feel free to browse through the longer "How to Use RStan" section. As you get more accustomed to Stan, you'll see that the Stan modeling language is a kind of extension of R's usual programming language – much more general and flexibe than `lmer()`'s modeling language.
6. Check out https://mc-stan.org/ for background info on stan.