# Censored observations

Week3-ex2, solution

## Exercise instructions

This is the same exercise as 2.1 except that now the posterior inference is performed with MCMC using Stan. Hence, instead of the grid based approximation we use Monte Carlo approximation to do the same analyses as in exercise 2.1.

Suppose you have a Gamma($\alpha = 1, \beta = 1$) prior distribution on the parameter $\lambda$ which corresponds to the expected number of ship ice besetting events (=events where a ship gets stuck in ice) during 1000 nautical miles in ice infested waters. The number of besetting events, $y$ per distance $d$ (nm) is modeled with a Poisson distribution Poisson($\lambda \times d$). The hyper-parameter $\alpha$ is the shape and $\beta$ is the inverse scale parameter. You are told that during winters 2013-2017 category A ice breakers traveled in total 6560 nautical miles in the Kara Sea (a sea area in the Arctic Sea). Within this distance they experienced in total more than 2 but less than 7 ice besetting events.

1) Implement the model with Stan and sample from the posterior of $\lambda$.
    a) Check for convergence visually and by calculating the PSRF statistics.
    b) Calculate the autocorrelation of the samples.
2) Using the samples of $\lambda$
    a) draw the posterior density function of $\lambda$ and
    b) calculate the posterior probability that $\lambda < 1$ and the 5% and the 95% quantiles.
    c) calculate the posterior mean and variance of $\lambda$.
3) Draw samples from the posterior predictive distribution for new $\tilde{y}$ for a ship traveling 1500 nm distance and
    a) draw histogram of samples from the posterior predictive distribution for $\tilde{y}$
    b) Calculate the posterior predictive mean and variance of $\tilde{y}$

## Model answer

### 1-3)

The posterior probability density function in case of censored observation is

$$p(\lambda|2 < y < 7, d = 6.56) \propto (\text{Poisson}(3|\lambda \times d) + \text{Poisson}(4|\lambda \times d) + \text{Poisson}(5|\lambda \times d) + \text{Poisson}(6|\lambda \times d))\,\text{Gamma}(\lambda|1, 1)$$

When using Stan, we need to first load the needed libraries into R and define a Stan model

```
library(ggplot2)
library(StanHeaders)
library(rstan)
```

```
## rstan (Version 2.21.2, 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)
```

```
library(coda)
```

```
##
## Attaching package: 'coda'

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

```
set.seed(123)

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

censored_observations_model="
data{
  real<lower = 0> d;
  int<lower = 0> y1;
  int<lower = 0> y2;
}
parameters{
  real<lower = 0> lambda;
}
model{
  lambda ~ gamma(1,1);  //prior
  target += log( poisson_cdf(y2,d*lambda)-poisson_cdf(y1,d*lambda) );
}
"
```

Now we can define the data list and run Markov chain with Stan

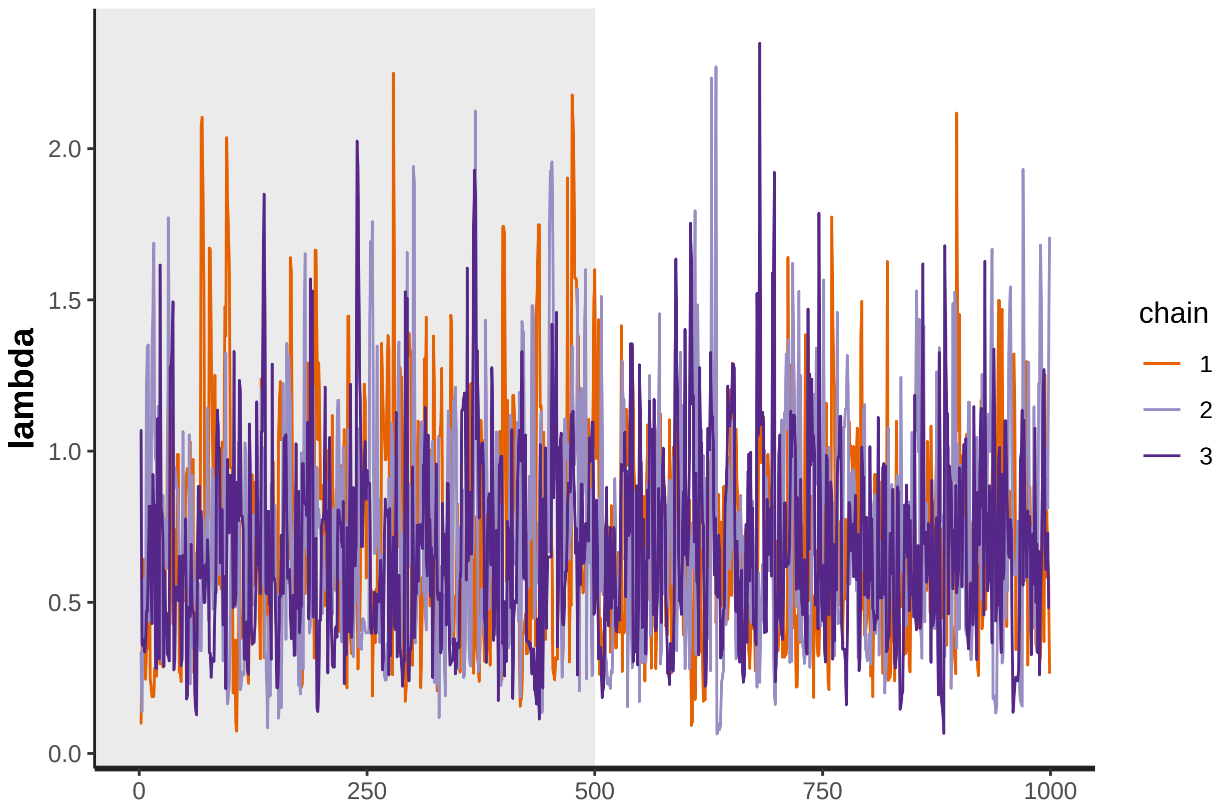
```
dataset <- list ("d"=6.56, "y1"=2, "y2"=6)

#give initial values for all chains for parameter theta
init1 <- list (lambda = 0.1)
init2 <- list (lambda = 2.5)
init3 <- list (lambda = 1)
inits <- list(init1, init2, init3)

# stan function does all of the work of fitting a Stan model and
# returning the results as an instance of stanfit = post in our exercises.
post=stan(model_code=censored_observations_model,data=dataset,init=inits,
          warmup=500,iter=1000,chains=3,thin=1)
```

Next we can examine the posterior samples in various ways.

```
# visual inspection
plot(post, plotfun= "trace", pars ="lambda", inc_warmup = TRUE)
```



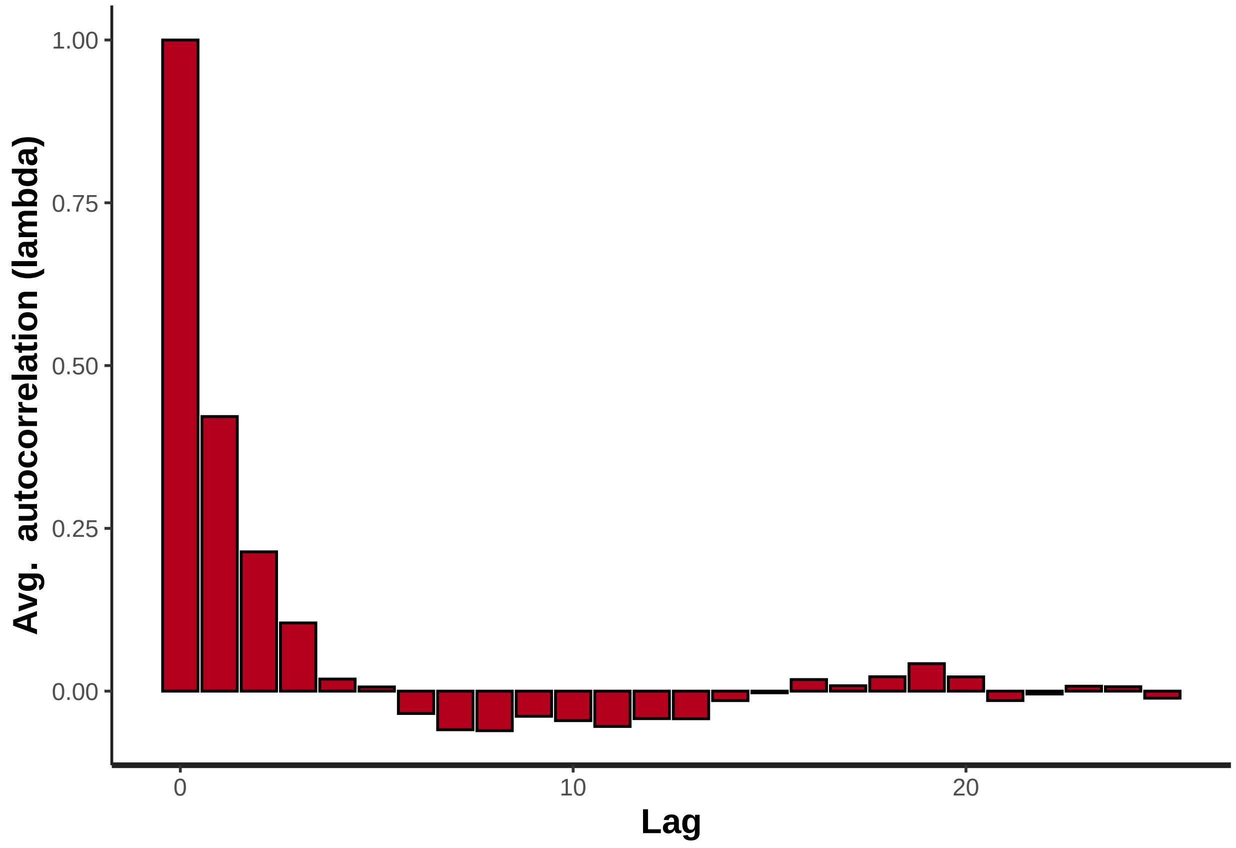
```r
# Inspection with PSRF=Rhat
print(post,pars="lambda")
```

```
## Inference for Stan model: a12cebf7fb8ec87bf4fafb6fe4783fc5.
## 3 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=1500.
##
##        mean se_mean   sd 2.5%  25%  50% 75% 97.5% n_eff Rhat
## lambda  0.7    0.01 0.33 0.21 0.45 0.66 0.9  1.47   592 1.01
##
## Samples were drawn using NUTS(diag_e) at Fri Nov 19 09:32:19 2021.
## 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).
```

```r
# Compared summary statistics for different chains
summary(post,pars="lambda")
```

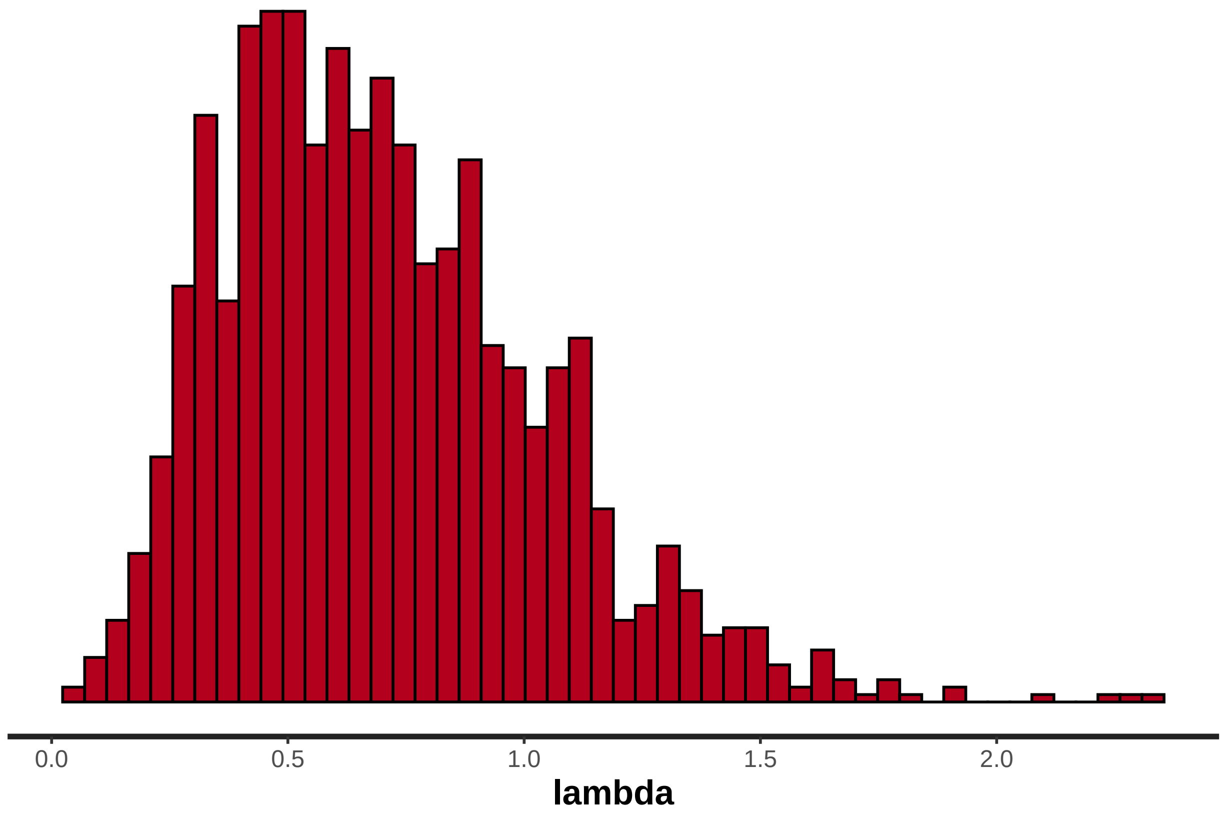
```
## $summary
##            mean    se_mean        sd      2.5%       25%       50%       75%
## lambda 0.704197 0.01374909 0.3344102 0.2075673 0.4538574 0.6588977 0.9023536
##           97.5%    n_eff     Rhat
## lambda 1.469214 591.5771 1.005417
##
## $c_summary
## , , chains = chain:1
##
##          stats
## parameter      mean        sd      2.5%       25%       50%       75%    97.5%
##    lambda 0.6855171 0.3099121 0.2396534 0.4531753 0.6420893 0.8747131 1.401723
##
## , , chains = chain:2
##
##          stats
## parameter      mean        sd      2.5%       25%       50%       75%    97.5%
##    lambda 0.7263724 0.3603203 0.1757574 0.4492082 0.6793014 0.9294298 1.519243
##
## , , chains = chain:3
##
##          stats
## parameter      mean        sd      2.5%       25%       50%       75%    97.5%
##    lambda 0.7007014 0.3304663 0.2075673 0.4587008 0.6488258 0.9090938 1.437936
```

```
#Calculate the autocorrelation of the samples after removing burn-in.  Is autocorrelation
#a problem here?
stan_ac(post,"lambda",inc_warmup = FALSE, lags = 25)
```



**2)**

```
#plot histogram of the posterior of lambda (approximation for density function)
plot(post, plotfun = "hist", pars = "lambda",bins=50)
```



```r
# take samples of lambda into a vector
lambda = as.matrix(post, pars="lambda")

# Calculate the probability that lambda<1
pr.1 = sum(lambda<1)/length(lambda)
print(pr.1)
```

```
## [1] 0.816
```

```r
post_summary2 <- quantile(lambda,probs = c(0.05, 0.95))
print(post_summary2)
```

```
##        5%       95%
## 0.2597716 1.3215794
```

```r
#calculate the posterior mean and variance
print(mean(lambda))
```

```
## [1] 0.704197
```

```r
print(var(lambda))
```

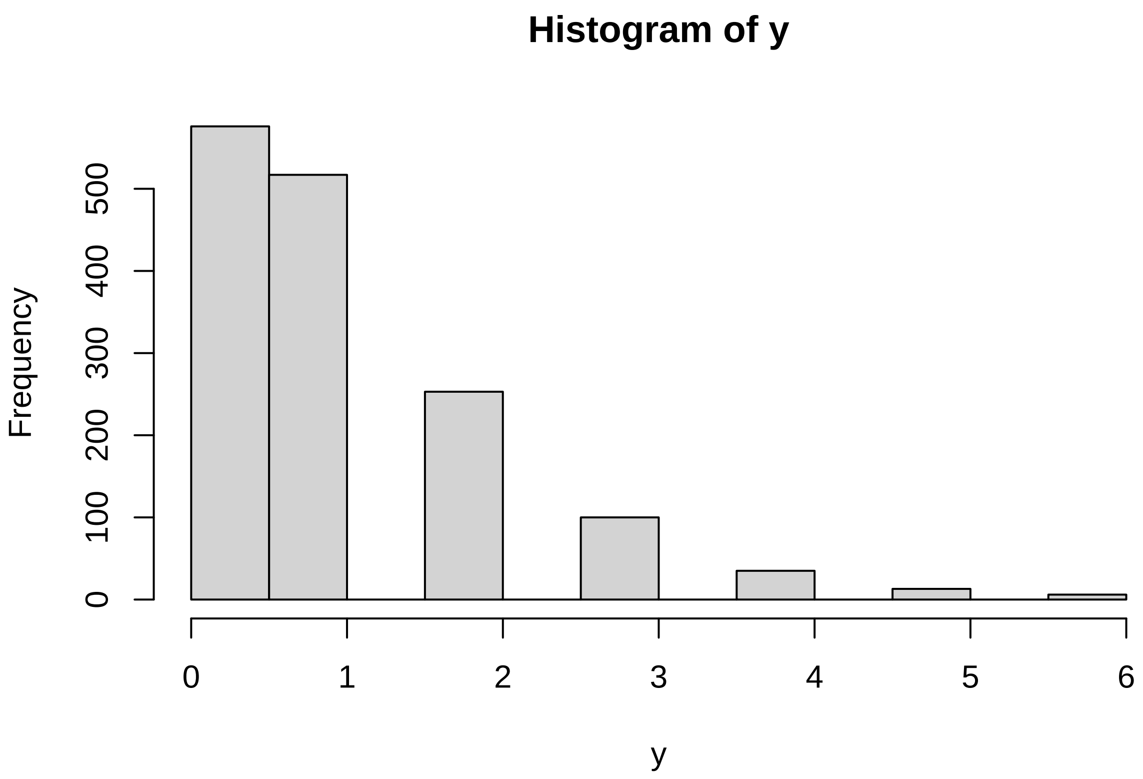
```
##           lambda
## lambda 0.1118302
```

## 3

Next we draw samples from the posterior predictive distribution for new $\tilde{y}$

```r
y = matrix(,nrow=length(lambda),ncol=1)
for (i1 in 1:length(lambda)){
  y[i1] = rpois(1,1.5*lambda[i1])
}
# alternatively you can draw samples as follows
# y = rpois(lambda,1.5*lambda)

hist(y)
```



```
(mean.y = mean(y))
```

```
## [1] 1.042667
```

```
(var.y = var(y))
```

```
##          [,1]
## [1,] 1.267024
```

## Grading

**Total 10 points:** 4 points for correclty doing step 1. 3 points for correctly doing step 2. 3 points for correctly doing step 3.