Speed of light data analysis

Instructions

Here we redo the analysis from page 66 in BDA3. The data are available from ex_speedOfLight.dat.

Simon Newcomb conducted experiments on speed of light in 1882. He measured the time required for light to travel a certain distance and here we will analyze a data recorded as deviations from 24,800 nanoseconds. The model used in BDA3 is %

$$
\begin{aligned}
y_i &\sim N(\mu, \sigma^2) \\
p(\mu, \sigma^2) &\propto \sigma^{-2}.
\end{aligned}
$$

% where y_i is the i'th measurement, μ is the mean of the measurement and σ^2 the variance of the measurements. Notice that this prior is improper ("uninformative"). This corresponds to widely used uniform prior for μ in the range $(-\infty, \infty)$, and uniform prior for $\log(\sigma)$ (BDA3 pp. 66, 52, and 21). Both priors are improper and cannot be found from Stan. You can use instead %

$$
\begin{aligned}
p(\mu) &\sim N(0, (10^3)^2) \\
p(\sigma^2) &\sim \text{Inv-}\chi^2(\nu = 4, s^2 = 1)
\end{aligned}
\tag{1}
$$

In this exercise your tasks are the following:

1. Write a Stan model for the above model and sample from the posterior of the parameters. Report the posterior mean, variance and 95% central credible interval for μ and σ^2.
2. Additionally draw samples from the posterior predictive distribution of hypothetical new measurement $p(\tilde{y}|y)$. Calculate the mean, variance and 95% quantile of the posterior predictive distribution.
3. How does the posterior predictive distribution differ from the posterior of μ and Why?
4. Which parts of the model could be interpreted to correspond to aleatory and epistemic uncertainty? Discuss whether this distinction is useful here.
5. Instead of Inverse-χ^2 distribution the variance parameter prior has traditionally been defined using Gamma distribution for the precision parameter $\tau = 1/\sigma^2$. By using the results in Appendix A of BDA3 derive the analytic form of a Gamma prior for the precision corresponding to the prior (1). This should be of the form Gamma(α, β), where α and β are functions of ν and s^2.

Note! Many common distributions have multiple parameterizations, for which reason you need to be careful when interpreting others' works. The variance/precision parameter and their priors are notorious for this. The reason is mainly historical since different parameterizations correspond to different analytical solutions.

Grading: 2 points from correct answer for each of the above steps.

Model answers

Load the needed libraries into R and set options for multicore computer.

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

```
## rstan (Version 2.19.3, 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)
```

```r
set.seed(123)

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

Part 1.

write the model description, set up initial values for 4 chains and sample from the posterior

```r
y = read.table("ex_speedOfLight.dat" , header=TRUE)
y = as.vector(y$y)
y
```

```
##  [1]  28  26  33  24  34 -44  27  16  40  -2  29  22  24  21  25  30  23  29  31
## [20]  19  24  20  36  32  36  28  25  21  28  29  37  25  28  26  30  32  36  26
## [39]  30  22  36  23  27  27  28  27  31  27  26  33  26  32  32  24  39  28  24
## [58]  25  32  25  29  27  28  29  16  23
```

```r
model="
data{
vector[66] y;
}

parameters{
real mu;
real<lower = 0> sigma2;
}

model{
mu ~ normal(0,1000);
sigma2 ~ scaled_inv_chi_square(4,1);
y ~ normal(mu,sqrt(sigma2));
}

generated quantities{
  real y_tilde=normal_rng(mu, sqrt(sigma2));
}
"
```

Let's then examine the convergence and autocorrelation of the chains.

```r
dataset <- list("y"=y)

#give initial values for all chains for parameter theta
init1 <- list (y = -25)
init2 <- list (y = 50)
init3 <- list (y = 25)

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=model,data=dataset,init=inits,
          warmup=500,iter=1000,chains=3,thin=1)
```

```
## Trying to compile a simple C file
```

```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## gcc -std=gnu99 -I"/usr/share/R/include" -DNDEBUG   -I"/usr/lib/R/site-library/Rcpp/include/"  -I"/us
## In file included from /usr/lib/R/site-library/RcppEigen/include/Eigen/Core:88,
##                  from /usr/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp:1
##                  from <command-line>:
## /usr/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: error: unknown type na
##   613 | namespace Eigen {
##       | ^~~~~~~~~
## /usr/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:17: error: expected ‘=’,
##   613 | namespace Eigen {
##       |                 ^
## In file included from /usr/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp:1
##                  from <command-line>:
## /usr/lib/R/site-library/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such file or di
##    96 | #include <complex>
##       |          ^~~~~~~~~
## compilation terminated.
## make: *** [/usr/lib/R/etc/Makeconf:167: foo.o] Error 1
```

```r
#  Report the posterior mean, variance and 95% central credible interval for

paste("Data mean:", mean(y))
```

```
## [1] "Data mean: 26.2121212121212"
```

```r
paste("Data variance:", var(y))
```

```
## [1] "Data variance: 115.462004662005"
```

```r
summary(post,pars="sigma2", probs = c(0.025, 0.975))
```

```
## $summary
```

```
##             mean   se_mean      sd     2.5%    97.5%   n_eff     Rhat
## sigma2 112.2316 0.6233482 20.4615 79.93693 156.4305 1077.49 1.000165
##
## $c_summary
## , , chains = chain:1
##
##           stats
## parameter     mean       sd     2.5%    97.5%
##    sigma2 111.7419 21.67514 78.56476 161.0449
##
## , , chains = chain:2
##
##           stats
## parameter     mean       sd     2.5%    97.5%
##    sigma2 112.0543 19.07191 81.18651 153.7133
##
## , , chains = chain:3
##
##           stats
## parameter     mean      sd     2.5%    97.5%
##    sigma2 112.8985 20.5774 79.88617 160.4329
```

```r
summary(post,pars="mu", probs = c(0.025, 0.975))
```

```
## $summary
##        mean    se_mean       sd    2.5%    97.5%    n_eff     Rhat
## mu 26.21446 0.03863509 1.285328 23.7259 28.76306 1106.787 1.000278
##
## $c_summary
## , , chains = chain:1
##
##           stats
## parameter     mean       sd     2.5%   97.5%
##        mu 26.22055 1.174531 24.02197 28.8345
##
## , , chains = chain:2
##
##           stats
## parameter     mean       sd     2.5%    97.5%
##        mu 26.15474 1.383464 23.12944 28.68575
##
## , , chains = chain:3
##
##           stats
## parameter     mean      sd     2.5%    97.5%
##        mu 26.26809 1.28951 23.80744 28.81245
```

```r
#Using samples method
muSamp = as.matrix(post, pars ="mu")
mean(muSamp)
```

```
## [1] 26.21446
```

```r
var(muSamp)
```

```
##           mu
## mu 1.652067
```

```r
sigma2Samp = as.matrix(post, pars ="sigma2")
mean(sigma2Samp)
```

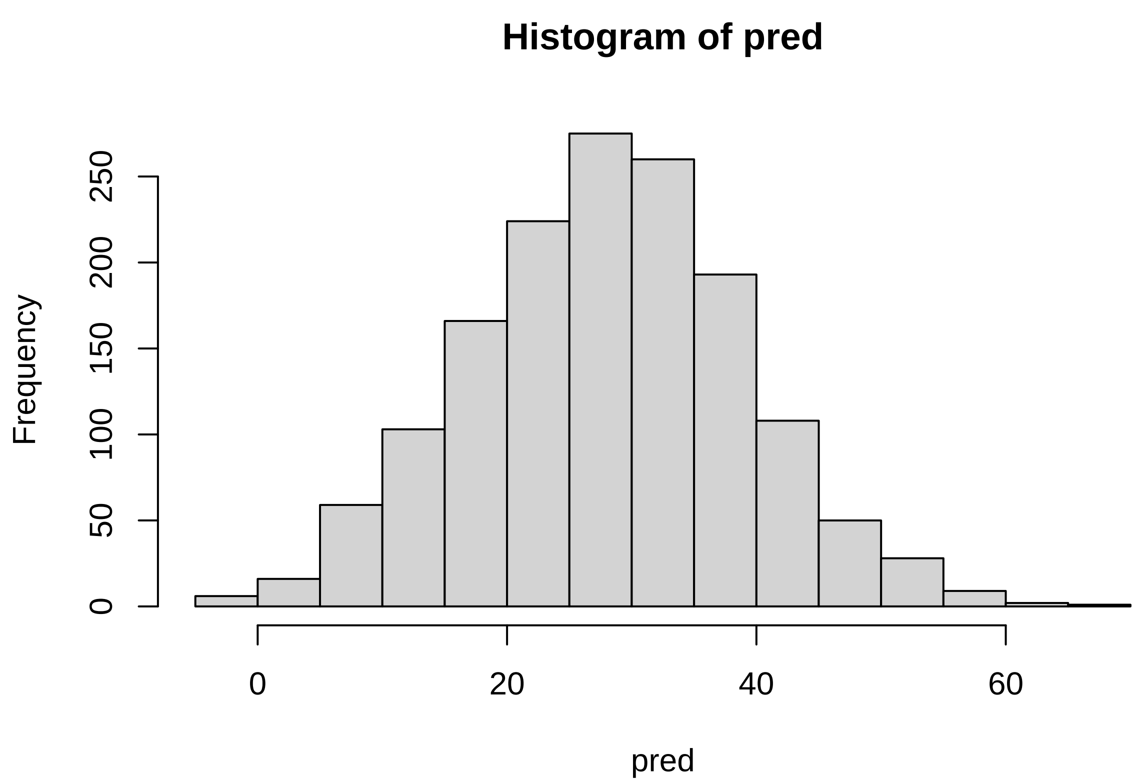
```
## [1] 112.2316
```

```r
var(sigma2Samp)
```

```
##          sigma2
## sigma2 418.6728
```

Part 2.

```r
pred = rep(0,length(muSamp))
for (i  in 1:length(muSamp) ){
  pred[i] = rnorm(1,muSamp,sqrt(sigma2Samp) )
}
hist(pred)
```



```r
mean(pred)
```

```
## [1] 28.07676
```

```r
var(pred)
```

```
## [1] 120.7651
```

```r
paste(quantile(pred, c(0.025, 0.975)))
```

```
## [1] "6.31456270067137" "50.094427836774"
```

Another method

Get predictions from the model directly

```r
# Get params from model
params = extract(post)

# Get and print y_tilde mean
paste("Y_tilde mean:", mean(params$y_tilde))
```

```
## [1] "Y_tilde mean: 26.4037968477721"
```

```r
# Get and print y_tilde variance
paste("Y_tilde variance:", var(params$y_tilde))
```

```
## [1] "Y_tilde variance: 112.103274358662"
```

```r
# Get and print interval [0.025, 0.975]
paste("95% central interval (0.025 to 0.975):")
```

```
## [1] "95% central interval (0.025 to 0.975):"
```

```r
paste(quantile(params$y_tilde, c(0.025, 0.975)))
```

```
## [1] "5.79853330642911" "47.485345796184"
```

```
**Part 3**

‘‘‘r
params = extract(post)
# Print mu mean and variance
paste("Mu mean:", mean(muSamp))
```

```
## [1] "Mu mean: 26.2144593744971"
```

```r
paste("Mu variance:", var(muSamp))
```

```
## [1] "Mu variance: 1.65206746825021"
```

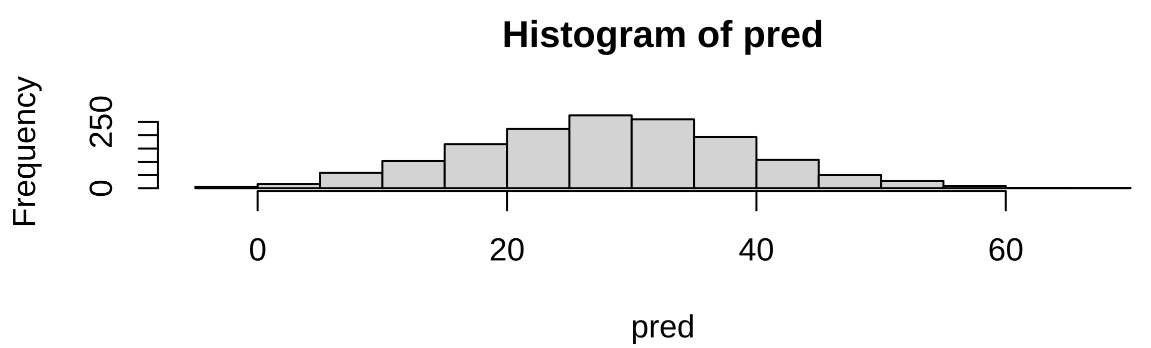
```r
# Print y_tilde mean and variance
paste("Y_tilde mean:", mean(pred))
```

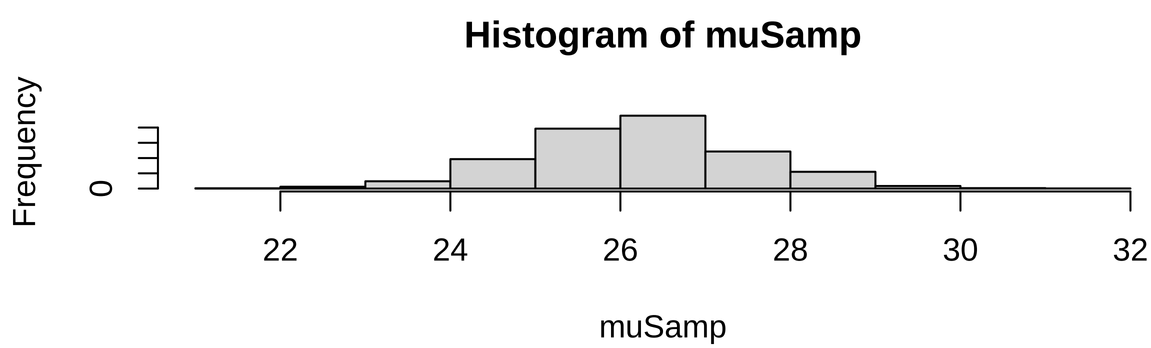
```
## [1] "Y_tilde mean: 28.076763215537"
```

```r
paste("Y_tilde variance:", var(pred))
```

```
## [1] "Y_tilde variance: 120.765065842372"
```

```r
# Create histogram of y_tilde and mu
par(mfrow=c(2,1))
hist(pred)
hist(muSamp)
```





As we can see, the two vectors have a close mean but a different variance. the width of the distribution is much bigger for the predictions ($\tilde{y}$) this is caused by the formula of the predictions $P(\tilde{y}|y)$. In this formula we have a factor $P(\theta|y)$ which is causing the difference in the variance. If we take an infinit number of samples, this problem will be resolved.

Part 4

Values μ and $\sigma 2$ are epistemic uncertainty. They will come more precice with larger amount of observations/measurements. At the same time, values of $\tilde{y}$ come from normal distribution and therefore it can be considered to belong to aleatory uncertainty.