Simple regression with Stan

In this program, we assume that we have a vector Y of N observations and a vector X of continuous predictors.

Our objective is to fit a simple regression model:

$$Y = \beta_0 + \beta_1 X + e$$

```
library(rstan)

## Loading required package:  ggplot2
## Loading required package:  StanHeaders
## rstan (Version 2.16.2, packaged:  2017-07-03 09:24:58 UTC, GitRev:
2e1f913d3ca3)
## For execution on a local, multicore CPU with excess RAM we recommend
calling
## rstan_options(auto_write = TRUE)
## options(mc.cores = parallel::detectCores())

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

As with the frequentist version, we'll use the highway mpg values from the epa mileage dataset.

```
df    =  read.table('body.dat.txt')      #read the body measurements data

y     = df$V23

x     = df$V24

N     = length(y)
```

```
stanfit = stan("simple_regression.stan")         #call stan to fit the model

print(stanfit)                              #print a summary of the results

## Inference for Stan model: simple_regression.
## 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%
## beta[1]  -104.61    0.22 7.48  -119.54  -109.69  -104.78   -99.51   -89.81
```

```
## beta[2]     1.02    0.00 0.04     0.93     0.99     1.02     1.04     1.10
## sigma       9.33    0.01 0.29     8.79     9.14     9.32     9.52     9.91
## lp__    -1383.99    0.04 1.24 -1387.27 -1384.49 -1383.67 -1383.10 -1382.61
##         n_eff Rhat
## beta[1]  1171    1
## beta[2]  1171    1
## sigma    1561    1
## lp__     1238    1
##
## Samples were drawn using NUTS(diag_e) at Sun Oct 29 21:55:38 2017.
## 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).
```

```
print(get_stanmodel(stanfit))
```

```
## S4 class stanmodel 'simple_regression' coded as follows:
## //Simple regression model
## data {
##   int N;               //sample size
##   vector[N] y;         //y data values
##   vector[N] x;         //x data values
## }
## parameters {
##   real beta[2];           //intercept and slope
##   real<lower=0> sigma;    //standard error
## }
## model {
##   beta ~ normal(0,100);   //normal priors for intercept and slope
##   sigma ~ cauchy(0,10);   //half-cauchy prior for sigma
##
##   y ~ normal(beta[1]+beta[2]*x,sigma);   //model normal with parameters (mu,sigma)
## }
##
```

```
pd=extract(stanfit)       #extract the posterior draw values

str(pd)                                  #show the structure of the posterior draw
```

```
## List of 3
##  $ beta : num [1:4000, 1:2] -109.9 -97.2 -103.9 -109.9 -84.4 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ iterations: NULL
##   .. ..$           : NULL
##  $ sigma: num [1:4000(1d)] 9.41 9.39 8.96 10.44 9.28 ...
```

```
##   ..- attr(*, "dimnames")=List of 1
##   .. ..$ iterations: NULL
##  $ lp__ : num [1:4000(1d)] -1383 -1386 -1383 -1390 -1386 ...
##   ..- attr(*, "dimnames")=List of 1
##   .. ..$ iterations: NULL
```

```
Sys.info()[["user"]]

## [1] "gquinn"
```