# SISG Bayes: Session 5 FTO/Stan example

Ken Rice

2026-05-27

*Before you start:* please check the course website for instructions: Windows users will need the RTools bundle, and the rstan package download is non-standard.

Once those steps are done, load the data:

```
# install.packages("rstan")
setwd("C:/Users/kenrice/OneDrive - UW/Desktop/SISGBayes/exercises")
load("yX_FTO.Rdata")
y <- yX$y
X <- yX$X
n <- nrow(X)
p <- ncol(X)
```

Write a file containing a description of the model in Stan's language, it has elements of C, R and BUGS/JAGS:

```
cat(file="FTOexample.stan", "
data {
  int n; //the number of observations
  int p; //the number of columns in the model matrix
  real y[n]; //the response
  matrix[n,p] X; //the model matrix
  real g; // Zellner scale factor
  vector[p] mu; // Zellner prior mean (all zeros)
  matrix[p,p] XtXinv; // information matrix
}
parameters {
  vector[p] beta; //the regression parameters
  real invsigma2; //the precision, a.k.a. inverse-variance
}
transformed parameters {
  vector[n] linpred;
  cov_matrix[p] Sigma;
  real sigma;
  linpred = X*beta;
  sigma = 1/sqrt(invsigma2);
  for (j in 1:p){
    for (k in 1:p){
      Sigma[j,k] = g*sigma^2*XtXinv[j,k];
      } } }
model {
  beta ~ multi_normal(mu, Sigma);
```

```
  y ~ normal(linpred, sigma);
  invsigma2 ~ gamma(0.5, 1.839); // we took nu0=1, sigma0=1.91
}
")
```

From within R, we set Stan going – first compiling the model code, to write code that does the MCMC sampling, then running that code and collecting its output in an R object:

```
library("rstan")
```

```
## Loading required package: StanHeaders
```

```
##
## rstan version 2.32.7 (Stan version 2.32.2)
```

```
## 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)
## For within-chain threading using ‘reduce_sum()‘ or ‘map_rect()‘ Stan functions,
## change ‘threads_per_chain‘ option:
## rstan_options(threads_per_chain = 1)
```

```
## Do not specify '-march=native' in 'LOCAL_CPPFLAGS' or a Makevars file
```

```
stan2 <- stan(file = "FTOexample.stan",
data = list(n=n,p=p, y=y, X=X, g=n, mu=rep(0,p), XtXinv=solve(crossprod(X)) ),
iter = 100000, chains = 1, pars=c("beta","sigma"))
```

```
## WARNING: Rtools is required to build R packages, but is not currently installed.
##
## Please download and install the appropriate version of Rtools for 4.6.0 from
## https://cran.r-project.org/bin/windows/Rtools/.
```

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

```
## Running "C:/PROGRA~1/R/R-46~1.0/bin/x64/Rcmd.exe" SHLIB foo.c
## using C compiler: 'gcc.exe (GCC) 14.3.0'
## gcc  -I"C:/PROGRA~1/R/R-46~1.0/include" -DNDEBUG   -I"C:/Users/kenrice/AppData/Local/R/win-library/4
## cc1.exe: warning: command-line option '-std=c++14' is valid for C++/ObjC++ but not for C
## In file included from C:/Users/kenrice/AppData/Local/R/win-library/4.6/RcppEigen/include/Eigen/Core:
##                  from C:/Users/kenrice/AppData/Local/R/win-library/4.6/RcppEigen/include/Eigen/Dense
##                  from C:/Users/kenrice/AppData/Local/R/win-library/4.6/StanHeaders/include/stan/math/
##                  from <command-line>:
## C:/Users/kenrice/AppData/Local/R/win-library/4.6/RcppEigen/include/Eigen/src/Core/util/Macros.h:679:
##   679 | #include <cmath>
##       |          ^~~~~~
## compilation terminated.
## make: *** [C:/PROGRA~1/R/R-46~1.0/etc/x64/Makeconf:297: foo.o] Error 1
```

```
## WARNING: Rtools is required to build R packages, but is not currently installed.
##
## Please download and install the appropriate version of Rtools for 4.6.0 from
## https://cran.r-project.org/bin/windows/Rtools/.

##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: model7f147e7d293e__namespace::log_prob: Sigma is not symmetric. Sigma[1,2] = nan
## Chain 1:
## Chain 1: Gradient evaluation took 6.1e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.61 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:     1 / 100000 [  0%]  (Warmup)
## Chain 1: Iteration: 10000 / 100000 [ 10%]  (Warmup)
## Chain 1: Iteration: 20000 / 100000 [ 20%]  (Warmup)
## Chain 1: Iteration: 30000 / 100000 [ 30%]  (Warmup)
## Chain 1: Iteration: 40000 / 100000 [ 40%]  (Warmup)
## Chain 1: Iteration: 50000 / 100000 [ 50%]  (Warmup)
## Chain 1: Iteration: 50001 / 100000 [ 50%]  (Sampling)
## Chain 1: Iteration: 60000 / 100000 [ 60%]  (Sampling)
## Chain 1: Iteration: 70000 / 100000 [ 70%]  (Sampling)
## Chain 1: Iteration: 80000 / 100000 [ 80%]  (Sampling)
## Chain 1: Iteration: 90000 / 100000 [ 90%]  (Sampling)
## Chain 1: Iteration: 100000 / 100000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 6.171 seconds (Warm-up)
## Chain 1:                6.003 seconds (Sampling)
## Chain 1:                12.174 seconds (Total)
## Chain 1:
```

Some summaries of the posterior samples – using packaged code instead of 'by hand'.

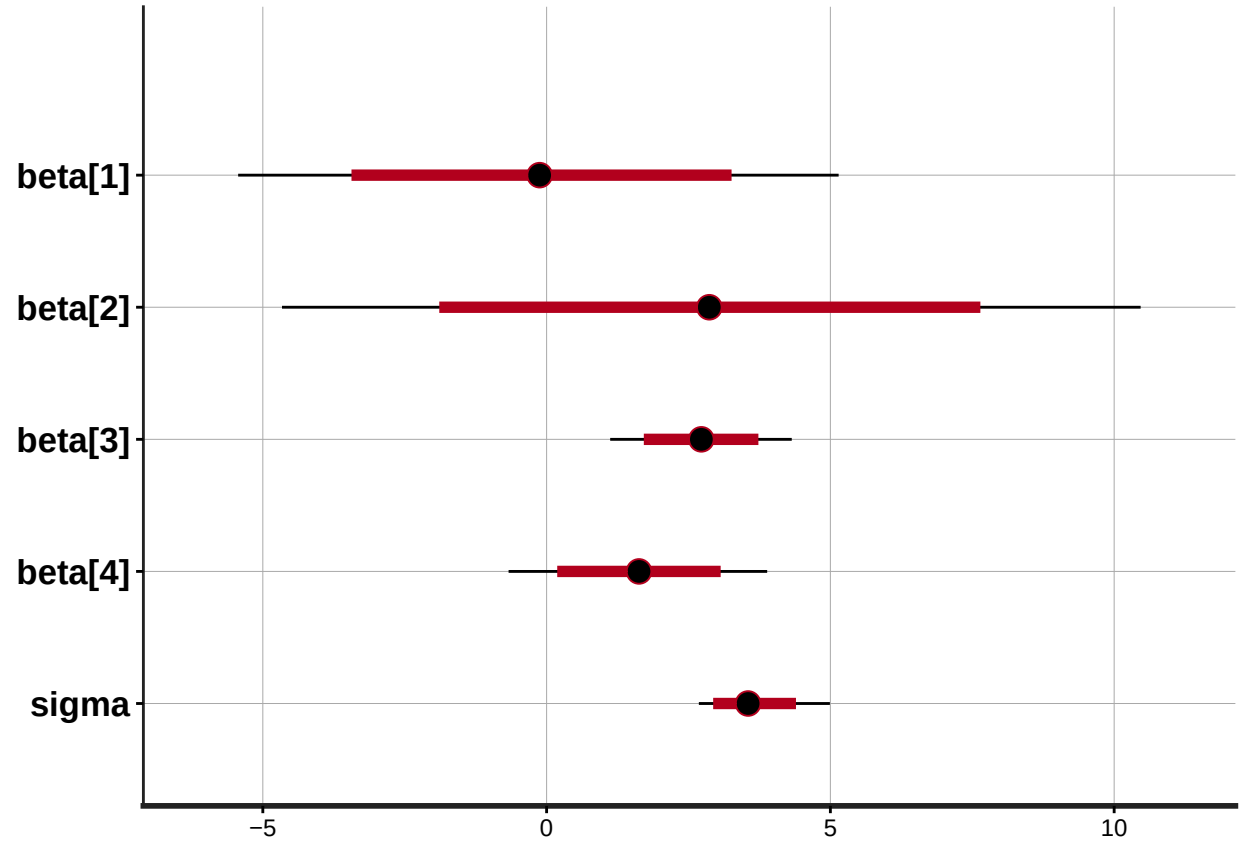
```
print(stan2)
```

```
## Inference for Stan model: anon_model.
## 1 chains, each with iter=1e+05; warmup=50000; thin=1;
## post-warmup draws per chain=50000, total post-warmup draws=50000.
##
##           mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## beta[1]  -0.11    0.02 2.66  -5.43  -1.83  -0.12   1.62   5.15 14957    1
## beta[2]   2.87    0.03 3.80  -4.66   0.38   2.87   5.31  10.47 14320    1
## beta[3]   2.72    0.01 0.80   1.12   2.20   2.73   3.24   4.32 14945    1
## beta[4]   1.63    0.01 1.15  -0.67   0.89   1.63   2.37   3.89 14471    1
## sigma     3.62    0.00 0.59   2.68   3.20   3.55   3.96   4.99 18227    1
## lp__    -42.49    0.01 1.66 -46.59 -43.36 -42.15 -41.28 -40.32 14971    1
##
## Samples were drawn using NUTS(diag_e) at Wed May 27 16:08:18 2026.
## 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).
```

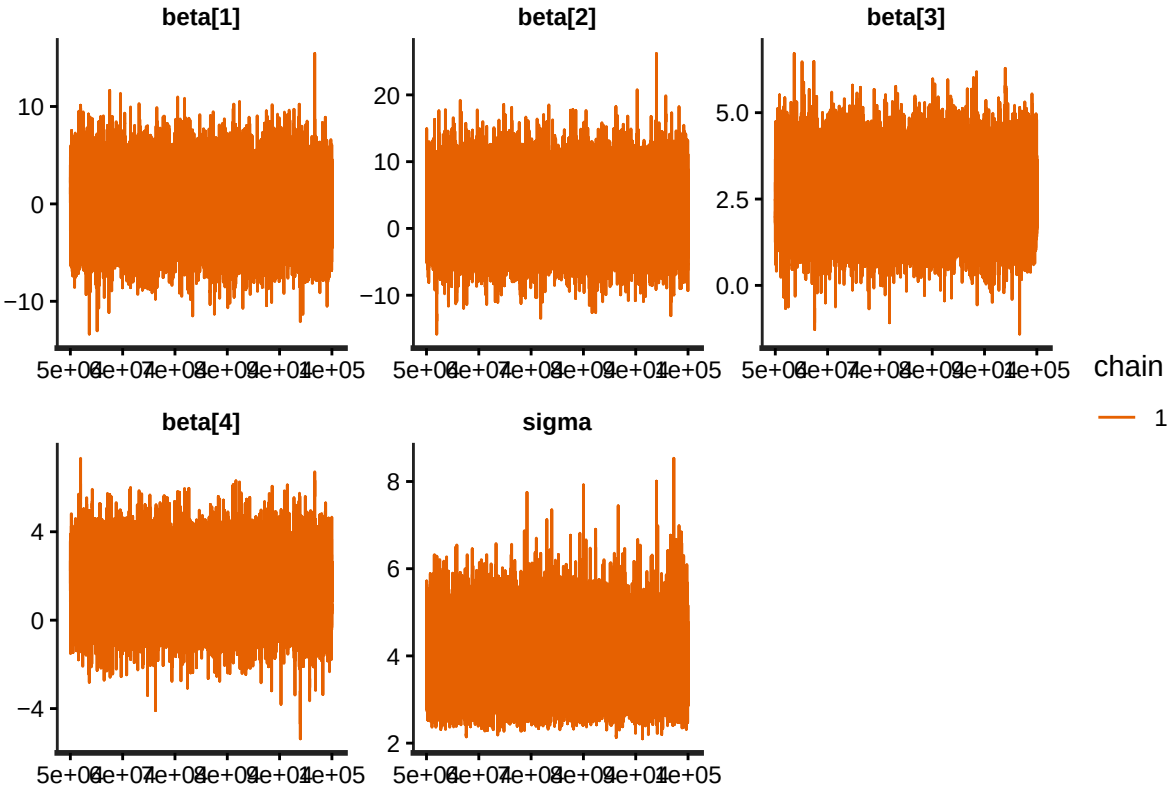
```
plot(stan2)
```

```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```




```
traceplot(stan2)
```


beta[1]
beta[2]
beta[3]
beta[4]
sigma
chain
1