stan model

Barbara E. Mottey

1/25/2021

```
dat <- readRDS('C:/Users/sympl/Documents/UMass/msthesis/Data/completedata.rds')

to_factors <- c("fuel_bin", "gender", "residence", "wealth", "education", "marital_s", "region")
dat %<>% mutate_at(to_factors, funs(factor(.)))
```

```
## Warning: `funs()` is deprecated as of dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
head(dat)
```

```
##   c_weight fuel_bin w_age   bmi gender residence wealth education marital_s
## 1     1800        1    29 27.30      2         2      5         2         1
## 2     2800        0    30 26.43      1         2      5         3         1
## 3     2500        0    26 36.13      2         2      5         2         1
## 4     3000        1    43 35.33      2         2      5         3         1
## 5     2900        1    25 19.21      2         2      2         1         1
## 6     3300        1    37 36.46      2         2      4         1         1
##   region
## 1      1
## 2      1
## 3      1
## 4      1
## 5      5
## 6      5
```

Define the model

We have multiple measurements from each region - independence assumption might be violated. So we add a random/varying intercept

$$Y_{ij} = \beta_0 + \beta_1 x_{1,i} + ... + \beta_p x_{p,i} + u_{0j} + \varepsilon_{ij}$$

$p = 8$

Take

$$u_{0j} \sim N(0, \sigma_u^2)$$

$$\varepsilon_{ij} \sim N(0, \sigma_\varepsilon^2)$$

Option A (not using a .stan file)

```
stancode <- "
  data {
    // number of observations
    int<lower=1> Nobs;

    // number of predictors
    int<lower=1> Npreds;

    // number of regions
    int<lower=1> J;

    // response list of length Nobs
    real weight[Nobs];

    // response matrix
    matrix[Nobs, Npreds] X;

   // provide the id for each region
   // this will be a list of length Nobs
   int<lower=1, upper=J> region[Nobs];

  }

  parameters {
    // matrix of regression coefficients ...
    // matrix[Npreds, J] beta; // this gives random slopes model
    vector[Npreds] beta; // this gives fixed slopes


    // region intercept
    vector[J] u;

    // specify the error terms
    real<lower=0> sigma_model;
    real<lower=0> sigma_region;
  }
```

```
  model {
    // declare a local variable
    real mu;
    // draw value from it's theoretical formulation
    u ~ normal(0, sigma_region);
    // likelihood
    for (i in 1:Nobs){
      // remember mu = XB + u  ... you could move this to transformed parameters
      mu = X[i, ]*beta + u[region[i]];
      weight[i] ~ normal(mu, sigma_model);
    }
  }

"
```

```
### instead of this, subset (fuel_bin to marital_s +intercept term) and use that as your matrix and com

sub2 <- dat[, 2:9]
#sub2$
intercept <- rep(1, nrow(sub2))

sub2<-cbind(intercept, sub2)
X.matrix2 <- sub2
```

```
# data for stan
stanData <- list(
  # number of observations
  Nobs = nrow(dat),
  # number of predictors
  Npreds = dim(X.matrix2)[2],
  # number of regions
  J = nlevels(dat$region),
  # region indicators -- should be integers
  region = as.integer(dat$region),
  # response vector
  weight = dat$c_weight,
  # design matrix
  X = X.matrix2
  )
```

```
library(rstan)
```

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

```
## 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)
```

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

##
## Attaching package: 'rstan'

## The following object is masked from 'package:magrittr':
##
##     extract

## The following object is masked from 'package:tidyr':
##
##     extract
```

```
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
fit <- stan(model_code = stancode, data=stanData, iter=1000, chains=4)
```

```
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#bulk-ess

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quanti
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#tail-ess
```

```
print(fit)
```

```
## Inference for Stan model: ef712a36bf4952dd38775f3236b08693.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##                  mean se_mean     sd      2.5%       25%       50%       75%
## beta[1]        2851.66    4.15 146.31   2568.58   2751.34   2853.82   2950.59
## beta[2]          22.65    0.83  36.09    -46.27     -1.22     22.18     47.82
## beta[3]           6.41    0.04   1.76      3.03      5.25      6.39      7.63
## beta[4]           1.81    0.02   0.96      0.01      1.17      1.79      2.46
## beta[5]        -132.98    0.60  23.40   -177.87   -147.75   -132.80   -117.45
## beta[6]          82.32    0.74  30.71     21.70     61.46     83.07    103.58
## beta[7]          27.82    0.35  12.92      3.40     19.25     27.65     36.59
## beta[8]          -9.63    0.36  15.68    -40.10    -19.88    -10.01      1.36
## beta[9]          42.86    0.86  36.28    -25.80     18.21     41.78     67.54
## u[1]             -2.13    1.14  36.02    -71.16    -25.81     -1.79     20.37
## u[2]            -12.82    0.98  35.40    -89.01    -34.14    -11.21      9.90
## u[3]            -16.17    1.12  32.36    -84.56    -36.17    -14.03      4.61
## u[4]             30.37    1.10  36.41    -34.16      5.07     26.57     53.77
## u[5]             75.25    1.71  41.89      1.10     45.13     73.53    102.10
## u[6]            -47.27    1.23  31.97   -112.72    -67.81    -45.55    -24.72
## u[7]              7.86    0.97  32.36    -53.85    -13.68      6.66     28.24
## u[8]             11.96    1.16  35.78    -55.82    -10.82     10.31     34.56
## u[9]            -34.26    1.09  36.26   -106.14    -58.55    -32.23     -8.81
## u[10]           -12.84    0.96  35.66    -85.55    -33.67    -12.30      9.57
```

```
## sigma_model     636.73    0.18   8.16    621.03    631.08    636.84    642.19
## sigma_region     53.29    1.16  23.85     15.52     37.00     49.56     65.70
## lp__         -20623.49    0.23   3.95 -20631.10 -20626.00 -20623.35 -20621.04
##                  97.5% n_eff Rhat
## beta[1]        3130.65  1244 1.00
## beta[2]          93.17  1880 1.00
## beta[3]           9.80  2294 1.00
## beta[4]           3.70  2066 1.00
## beta[5]         -86.40  1523 1.00
## beta[6]         141.33  1732 1.00
## beta[7]          53.47  1386 1.00
## beta[8]          20.41  1882 1.00
## beta[9]         116.96  1791 1.00
## u[1]             72.41  1005 1.00
## u[2]             58.33  1301 1.00
## u[3]             43.95   832 1.00
## u[4]            108.67  1103 1.00
## u[5]            166.60   597 1.00
## u[6]              9.06   679 1.00
## u[7]             74.16  1106 1.00
## u[8]             86.42   951 1.00
## u[9]             30.46  1112 1.00
## u[10]            57.70  1378 1.00
## sigma_model     653.13  2140 1.00
## sigma_region    107.81   425 1.01
## lp__         -20616.19   299 1.01
##
## Samples were drawn using NUTS(diag_e) at Wed Jan 27 12:47:59 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).
```

```
betas <- (extract(fit))$beta
apply(betas, 2, mean)
```

```
## [1] 2851.660795   22.649607    6.410741    1.812659 -132.979066   82.318784
## [7]   27.823612   -9.631668   42.862193
```

```
##mean and CI of posterior estimate of cooking fuel
mean(betas[, 1])
```

```
## [1] 2851.661
```

```
quantile(betas[, 1], c(0.025,0.975))
```

```
##     2.5%    97.5%
## 2568.582 3130.654
```

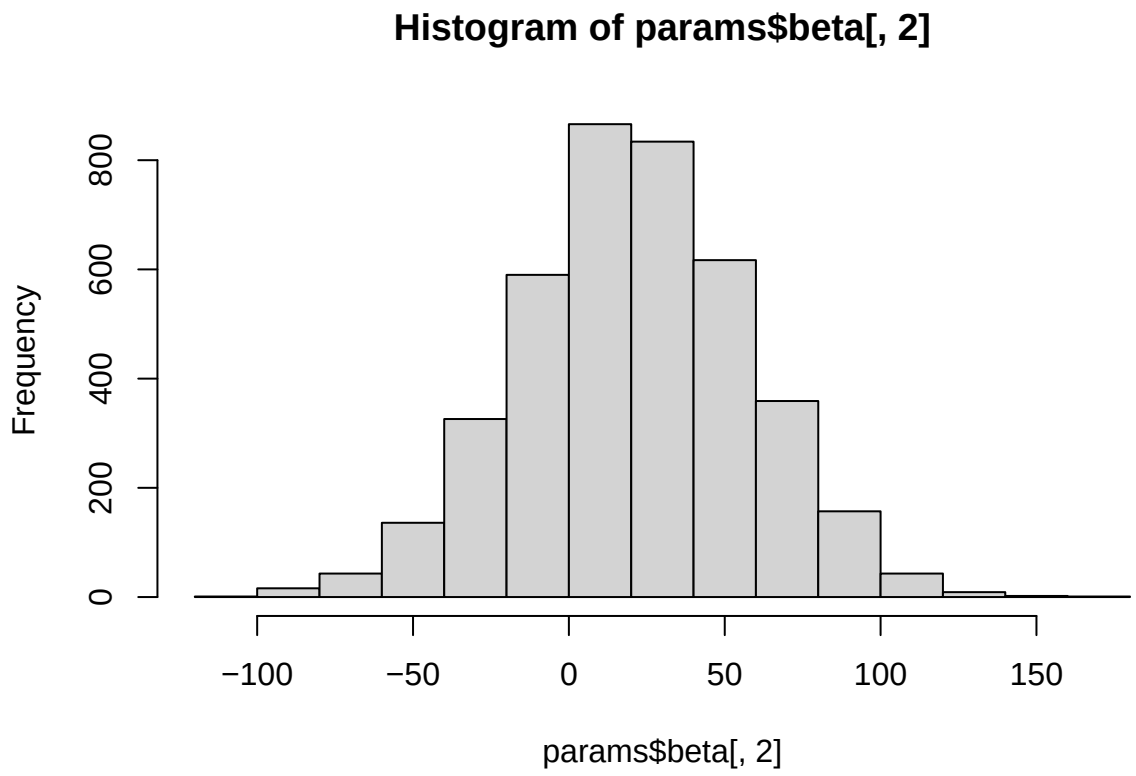
##Option 2

```
model<- stan_model("stanbw.stan")
```

```
## recompiling to avoid crashing R session
```

```
fit1<- sampling(model, stanData,  iter=2000, chains=4)
params<- extract(fit1) #extract the parameters for graphing

## histogram of average birthweight from cooking fuel
hist(params$beta[,2])
```




```
Betas <- (extract(fit1))$beta
apply(Betas, 2, mean)
```

```
## [1] 2858.895933   20.816340    6.351673    1.817759 -133.594194   82.407086
## [7]   27.629748  -10.100833   42.485188
```

```
##mean and CI of posterior estimate of cooking fuel
mean(Betas[, 2])
```

```
## [1] 20.81634
```

```r
quantile(Betas[, 2], c(0.025,0.975))
```

```
##      2.5%     97.5%
## -52.49386  91.74646
```

```r
library(shinystan)
```

```
## Loading required package: shiny
```

```
##
## Attaching package: 'shiny'
```

```
## The following object is masked from 'package:pander':
##
##     p
```

```
##
## This is shinystan version 2.5.0
```

```r
launch_shinystan(fit)
```

```
##
## Launching ShinyStan interface... for large models this  may take some time.
```

```
##
## Listening on http://127.0.0.1:6429
```