Homework Help

```r
# load libraries
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.3     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.4     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.0
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
library(mvtnorm)
library(coda)
library(rstanarm)
```

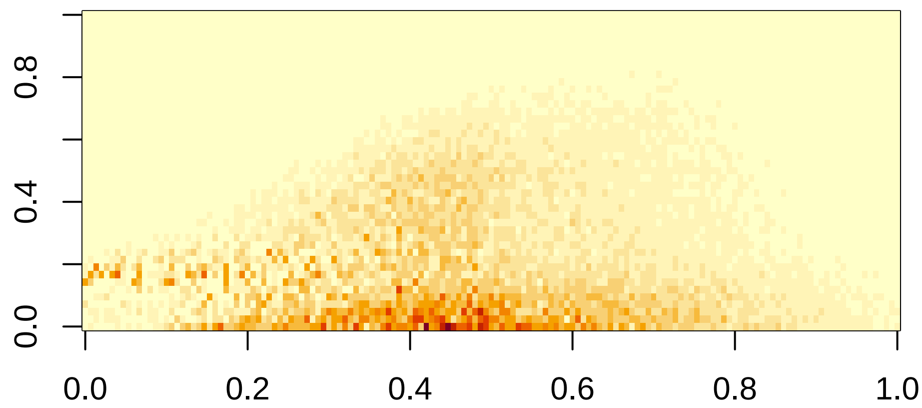
```
Loading required package: Rcpp
This is rstanarm version 2.26.1
- See https://mc-stan.org/rstanarm/articles/priors for changes to default priors!
- Default priors may change, so it's safest to specify priors, even if equivalent to the defa
- For execution on a local, multicore CPU with excess RAM we recommend calling
  options(mc.cores = parallel::detectCores())
```

```r
yX = readRDS(
  url("http://www2.stat.duke.edu/~pdh10/Teaching/360/Materials/yXSS.rds"))

y<-yX[,1]
```

```r
X<-yX[,-1]
```

```r
# image(matrix(y,151,43))
image(matrix(y,151,43))
```



$$y \sim N(X\beta, \sigma^2 I)$$

- y: 6493×1
- X: 6493×9
- β: 9×1
- σ^2: 1×1
- I: 6493×6493

$$y_i = \beta_1 x_1 + \beta_2 x_2 + ... \beta_p x_p + \epsilon_i$$

```r
set.seed(360)
# prior hyperparameters
p = 9 # number of covariates
Sigma0 = 1 * diag(rep(1, p)) # p dimension since no intercept.
b0 = rep(1/9, p)
nu0 = 2
sigma02 = 1
n = length(y)

# starting values
## note: gamma = 1 / sigma^2
beta = t(rep(1/9, 9))
```

```r
# values we should compute just once
SigmaInv = solve(Sigma0)
X2 = t(X) %*% X
Xy = t(X) %*% y
SIB0 = SigmaInv %*% b0
a = (nu0 + n) / 2
nu0s02 = nu0 * sigma02

## empty objects to fill
BETA = NULL
GAMMA = NULL

S = 2000
for (s in 1:S) {

  ### UPDATE SIGMA
  SSR1 = (y - (X %*% t(beta)))
  SSRB = t(SSR1) %*% SSR1
  gamma = rgamma(1, a, ((nu0s02 + SSRB) / 2))

  ### UPDATE BETA
  V = solve(SigmaInv + (gamma * X2))
  m = V %*% (Xy * gamma) # simplified since b0 = 0
  beta = rmvnorm(1, mean = m, sigma = V)


  ### SAVE STATES
  GAMMA = c(GAMMA, gamma)
  BETA = rbind(BETA, beta)
}
```

```r
effectiveSize(BETA)
```

```
    var1     var2     var3     var4     var5     var6     var7     var8 
2000.000 2000.000 2000.000 2000.000 2000.000 2040.626 2000.000 2000.000 
    var9 
2000.000 
```

```r
effectiveSize(GAMMA)
```

```
    var1
1810.087
```

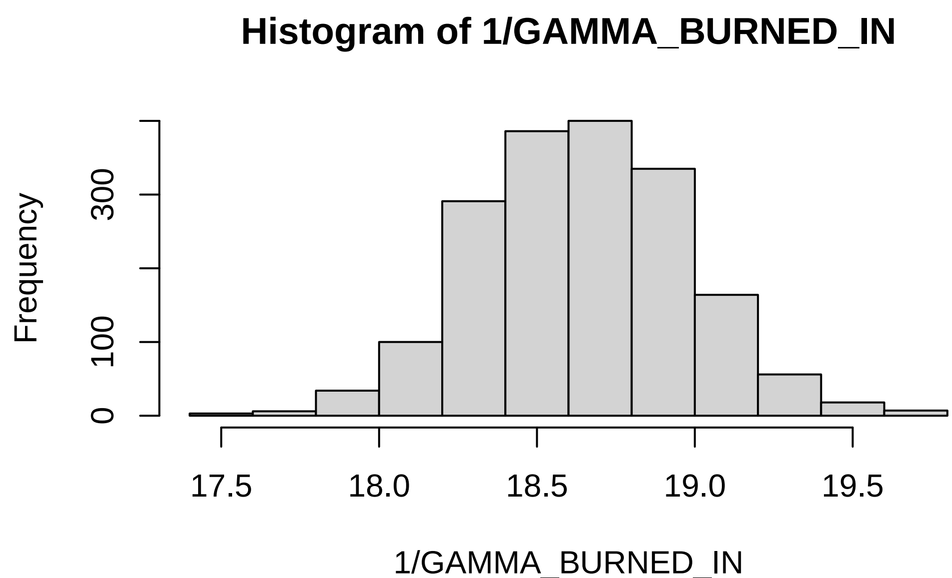
```r
# hist(1 / GAMMA)
s2 = 1 / GAMMA
sum(s2 > 100)
```

```
[1] 1
```

```r
s2[1:5]
```

```
[1] 3086.86079   19.68145   18.75163   18.19946   18.33743
```

```r
GAMMA_BURNED_IN = GAMMA[-c(1:S/10)]
hist(1 / GAMMA_BURNED_IN)
```




```r
BETA_BURNED = BETA[-c(1:S/10),]

posteriorMeanBeta = apply(BETA_BURNED, 2, mean)
posteriorCIBeta = apply(BETA_BURNED, 2, quantile, probs = c(0.025, 0.975))
rbind(posteriorMeanBeta, posteriorCIBeta)
```

```
                         [,1]        [,2]          [,3]        [,4]          [,5]
posteriorMeanBeta 0.5466149 -0.01868557  0.0002884716 0.049511233 -0.0122083365
2.5%              0.4300168 -0.05692517 -0.0163582932 0.004122091 -0.0255515678
97.5%             0.6608110  0.02140099  0.0168541549 0.096441935  0.0009236346
                         [,6]      [,7]        [,8]        [,9]
posteriorMeanBeta 0.2741874 0.1606993 0.004890124 -0.08407046
2.5%              0.1969602 0.1133077 0.002942036 -0.22878357
97.5%             0.3453285 0.2089253 0.006761198  0.06937184
```

Predictors 1, 6, 7.

```r
colnames(X)[c(1, 6, 7)]
```

```
[1] "Effluent" "Soil"     "Street"
```

Alternate method

- `rstanarm` uses software `stan`
- stan lets us sample from posteriors without writing a sampler ourselves each time.
- stan uses "HMC": Hamiltonian Monte Carlo to sample from the posterior.
- one of the drawbacks of stan is that the specific prior specification we want may be difficult or impossible to obtain in the package.

```r
yX_df = data.frame(yX)
post1 = stan_glm(V1 ~ 0 + ., data = yX_df, # remove the intercept
                 family = gaussian(link = "identity"),
                 seed = 360,
                 prior = normal(1/9, 1), # sets the beta prior
                 prior_aux = NULL) # set a flat prior on sigma
```

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 6.6e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.66 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
```

```
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.75 seconds (Warm-up)
Chain 1:                0.993 seconds (Sampling)
Chain 1:                1.743 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 1.3e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.13 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 1.109 seconds (Warm-up)
Chain 2:                1.072 seconds (Sampling)
Chain 2:                2.181 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 1.4e-05 seconds
```

```
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.862 seconds (Warm-up)
Chain 3:                1.101 seconds (Sampling)
Chain 3:                1.963 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 3.8e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.38 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 1.309 seconds (Warm-up)
```

```
Chain 4:                1.013 seconds (Sampling)
Chain 4:                2.322 seconds (Total)
Chain 4:
```

```r
cbind(post1$coefficients, posteriorMeanBeta)
```

```
                        posteriorMeanBeta
Effluent   0.5489818902      0.5466148619
Influent  -0.0193440302     -0.0186855712
Poultry    0.0004494069      0.0002884716
Reference  0.0511734579      0.0495112329
Septic    -0.0117853543     -0.0122083365
Soil       0.2742320104      0.2741873716
Street     0.1588910894      0.1606993061
Swine      0.0048560389      0.0048901240
Wetland   -0.0855007899     -0.0840704585
```

```r
mean(sqrt(1 / GAMMA_BURNED_IN))
```

```
[1] 4.317466
```