Homework 6

Garrett Carr

Welcome to homework 6!

For this homework, we want you to use the VO2.dat dataset to predict MaxVO2ML (maximum squared milliliters of air in lungs) based on the other covariates in the dataset. The idea is simple here. Find the "best" model.

Finding the "best" model is obviously subjective. We define the "best" model here to excel in both information criterion and interpretability. In identifying this model, use one of the probabilistic programming languages (PPLs), such as JAGS or Stan, that we have been using in class to build the model.

When you have decided on a model that you feel is justified, report all useful convergence diagnostics and information criterion. After deciding on a model, REPLICATE this model in both JAGS and Stan. It is expected that you will have attempted multiple different models, but only report statistics for a single model (using both languages).

For the grading of this assignment, we expect you to explain your reasoning and provide interpretation for the model you choose.

There is added incentive to this assignment. The top 2 students with lowest reported WAIC plus a complexity penalty will each be awarded 2 bonus points to their assignment. The catch is that the WAIC will be penalized for complexity (number of terms). For the penalization, each individual covariate in the model will receive 1 point of penalty, and interactions will receive 2 penalty points. A model has to be simple and effective.

```
library(cmdstanr)
```

```
## This is cmdstanr version 0.4.0

## - Online documentation and vignettes at mc-stan.org/cmdstanr

## - CmdStan path set to: C:/Users/fyref/Documents/.cmdstanr/cmdstan-2.29.0

## - Use set_cmdstan_path() to change the path
```

```
library(rstan)
```

```
## Loading required package: StanHeaders

## Loading required package: ggplot2

## rstan (Version 2.21.3, 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)
```

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

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::extract() masks rstan::extract()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
```

```
library(posterior)
```

```
## This is posterior version 1.2.0

##
## Attaching package: 'posterior'

## The following objects are masked from 'package:rstan':
##
##     ess_bulk, ess_tail

## The following objects are masked from 'package:stats':
##
##     mad, sd, var
```

```
library(bayesplot)
```

```
## This is bayesplot version 1.8.1

## - Online documentation and vignettes at mc-stan.org/bayesplot

## - bayesplot theme set to bayesplot::theme_default()

##    * Does _not_ affect other ggplot2 plots

##    * See ?bayesplot_theme_set for details on theme setting

##
## Attaching package: 'bayesplot'

## The following object is masked from 'package:posterior':
##
##     rhat
```

```
library(R2jags)
```

```
## Loading required package: rjags
```

```
## Loading required package: coda
```

```
##
## Attaching package: 'coda'
```

```
## The following object is masked from 'package:rstan':
##
##     traceplot
```

```
## Linked to JAGS 4.3.0
```

```
## Loaded modules: basemod,bugs
```

```
##
## Attaching package: 'R2jags'
```

```
## The following object is masked from 'package:coda':
##
##     traceplot
```

```
## The following object is masked from 'package:rstan':
##
##     traceplot
```

```
# Read in Data
dat <- read.table('vo2.dat', header = TRUE) %>%
  as_tibble()

# Create data list
data_list <- list(
  N = count(dat)[[1]],
  y = dat$MaxVO2ML,
  gender = dat$Gender,
  age = dat$Age1,
  bmi = dat$BMI,
  hr = dat$HR,
  rpe = dat$RPE
)

rstan_options(auto_write = TRUE)
```

```
##
## SAMPLING FOR MODEL 'vo2' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
```

```
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 9000 [  0%]  (Warmup)
## Chain 1: Iteration:  900 / 9000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1001 / 9000 [ 11%]  (Sampling)
## Chain 1: Iteration: 1900 / 9000 [ 21%]  (Sampling)
## Chain 1: Iteration: 2800 / 9000 [ 31%]  (Sampling)
## Chain 1: Iteration: 3700 / 9000 [ 41%]  (Sampling)
## Chain 1: Iteration: 4600 / 9000 [ 51%]  (Sampling)
## Chain 1: Iteration: 5500 / 9000 [ 61%]  (Sampling)
## Chain 1: Iteration: 6400 / 9000 [ 71%]  (Sampling)
## Chain 1: Iteration: 7300 / 9000 [ 81%]  (Sampling)
## Chain 1: Iteration: 8200 / 9000 [ 91%]  (Sampling)
## Chain 1: Iteration: 9000 / 9000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 8.65 seconds (Warm-up)
## Chain 1:                46.881 seconds (Sampling)
## Chain 1:                55.531 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'vo2' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 9000 [  0%]  (Warmup)
## Chain 2: Iteration:  900 / 9000 [ 10%]  (Warmup)
## Chain 2: Iteration: 1001 / 9000 [ 11%]  (Sampling)
## Chain 2: Iteration: 1900 / 9000 [ 21%]  (Sampling)
## Chain 2: Iteration: 2800 / 9000 [ 31%]  (Sampling)
## Chain 2: Iteration: 3700 / 9000 [ 41%]  (Sampling)
## Chain 2: Iteration: 4600 / 9000 [ 51%]  (Sampling)
## Chain 2: Iteration: 5500 / 9000 [ 61%]  (Sampling)
## Chain 2: Iteration: 6400 / 9000 [ 71%]  (Sampling)
## Chain 2: Iteration: 7300 / 9000 [ 81%]  (Sampling)
## Chain 2: Iteration: 8200 / 9000 [ 91%]  (Sampling)
## Chain 2: Iteration: 9000 / 9000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 7.126 seconds (Warm-up)
## Chain 2:                45.776 seconds (Sampling)
## Chain 2:                52.902 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'vo2' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 9000 [  0%]  (Warmup)
```

```
## Chain 3: Iteration:  900 / 9000 [ 10%]  (Warmup)
## Chain 3: Iteration: 1001 / 9000 [ 11%]  (Sampling)
## Chain 3: Iteration: 1900 / 9000 [ 21%]  (Sampling)
## Chain 3: Iteration: 2800 / 9000 [ 31%]  (Sampling)
## Chain 3: Iteration: 3700 / 9000 [ 41%]  (Sampling)
## Chain 3: Iteration: 4600 / 9000 [ 51%]  (Sampling)
## Chain 3: Iteration: 5500 / 9000 [ 61%]  (Sampling)
## Chain 3: Iteration: 6400 / 9000 [ 71%]  (Sampling)
## Chain 3: Iteration: 7300 / 9000 [ 81%]  (Sampling)
## Chain 3: Iteration: 8200 / 9000 [ 91%]  (Sampling)
## Chain 3: Iteration: 9000 / 9000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 6.654 seconds (Warm-up)
## Chain 3:                42.84 seconds (Sampling)
## Chain 3:                49.494 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'vo2' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 9000 [  0%]  (Warmup)
## Chain 4: Iteration:  900 / 9000 [ 10%]  (Warmup)
## Chain 4: Iteration: 1001 / 9000 [ 11%]  (Sampling)
## Chain 4: Iteration: 1900 / 9000 [ 21%]  (Sampling)
## Chain 4: Iteration: 2800 / 9000 [ 31%]  (Sampling)
## Chain 4: Iteration: 3700 / 9000 [ 41%]  (Sampling)
## Chain 4: Iteration: 4600 / 9000 [ 51%]  (Sampling)
## Chain 4: Iteration: 5500 / 9000 [ 61%]  (Sampling)
## Chain 4: Iteration: 6400 / 9000 [ 71%]  (Sampling)
## Chain 4: Iteration: 7300 / 9000 [ 81%]  (Sampling)
## Chain 4: Iteration: 8200 / 9000 [ 91%]  (Sampling)
## Chain 4: Iteration: 9000 / 9000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 6.791 seconds (Warm-up)
## Chain 4:                43.971 seconds (Sampling)
## Chain 4:                50.762 seconds (Total)
## Chain 4:


##
## SAMPLING FOR MODEL 'vo2' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:     1 / 15000 [  0%]  (Warmup)
## Chain 1: Iteration:  1500 / 15000 [ 10%]  (Warmup)
## Chain 1: Iteration:  3000 / 15000 [ 20%]  (Warmup)
## Chain 1: Iteration:  3001 / 15000 [ 20%]  (Sampling)
```

```
## Chain 1: Iteration:  4500 / 15000 [ 30%]  (Sampling)
## Chain 1: Iteration:  6000 / 15000 [ 40%]  (Sampling)
## Chain 1: Iteration:  7500 / 15000 [ 50%]  (Sampling)
## Chain 1: Iteration:  9000 / 15000 [ 60%]  (Sampling)
## Chain 1: Iteration: 10500 / 15000 [ 70%]  (Sampling)
## Chain 1: Iteration: 12000 / 15000 [ 80%]  (Sampling)
## Chain 1: Iteration: 13500 / 15000 [ 90%]  (Sampling)
## Chain 1: Iteration: 15000 / 15000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 16.722 seconds (Warm-up)
## Chain 1:                81.044 seconds (Sampling)
## Chain 1:                97.766 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'vo2' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:     1 / 15000 [  0%]  (Warmup)
## Chain 2: Iteration:  1500 / 15000 [ 10%]  (Warmup)
## Chain 2: Iteration:  3000 / 15000 [ 20%]  (Warmup)
## Chain 2: Iteration:  3001 / 15000 [ 20%]  (Sampling)
## Chain 2: Iteration:  4500 / 15000 [ 30%]  (Sampling)
## Chain 2: Iteration:  6000 / 15000 [ 40%]  (Sampling)
## Chain 2: Iteration:  7500 / 15000 [ 50%]  (Sampling)
## Chain 2: Iteration:  9000 / 15000 [ 60%]  (Sampling)
## Chain 2: Iteration: 10500 / 15000 [ 70%]  (Sampling)
## Chain 2: Iteration: 12000 / 15000 [ 80%]  (Sampling)
## Chain 2: Iteration: 13500 / 15000 [ 90%]  (Sampling)
## Chain 2: Iteration: 15000 / 15000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 21.039 seconds (Warm-up)
## Chain 2:                88.975 seconds (Sampling)
## Chain 2:                110.014 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'vo2' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:     1 / 15000 [  0%]  (Warmup)
## Chain 3: Iteration:  1500 / 15000 [ 10%]  (Warmup)
## Chain 3: Iteration:  3000 / 15000 [ 20%]  (Warmup)
## Chain 3: Iteration:  3001 / 15000 [ 20%]  (Sampling)
## Chain 3: Iteration:  4500 / 15000 [ 30%]  (Sampling)
## Chain 3: Iteration:  6000 / 15000 [ 40%]  (Sampling)
## Chain 3: Iteration:  7500 / 15000 [ 50%]  (Sampling)
## Chain 3: Iteration:  9000 / 15000 [ 60%]  (Sampling)
```

```
## Chain 3: Iteration: 10500 / 15000 [ 70%]  (Sampling)
## Chain 3: Iteration: 12000 / 15000 [ 80%]  (Sampling)
## Chain 3: Iteration: 13500 / 15000 [ 90%]  (Sampling)
## Chain 3: Iteration: 15000 / 15000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 14.983 seconds (Warm-up)
## Chain 3:                76.389 seconds (Sampling)
## Chain 3:                91.372 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'vo2' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:     1 / 15000 [  0%]  (Warmup)
## Chain 4: Iteration:  1500 / 15000 [ 10%]  (Warmup)
## Chain 4: Iteration:  3000 / 15000 [ 20%]  (Warmup)
## Chain 4: Iteration:  3001 / 15000 [ 20%]  (Sampling)
## Chain 4: Iteration:  4500 / 15000 [ 30%]  (Sampling)
## Chain 4: Iteration:  6000 / 15000 [ 40%]  (Sampling)
## Chain 4: Iteration:  7500 / 15000 [ 50%]  (Sampling)
## Chain 4: Iteration:  9000 / 15000 [ 60%]  (Sampling)
## Chain 4: Iteration: 10500 / 15000 [ 70%]  (Sampling)
## Chain 4: Iteration: 12000 / 15000 [ 80%]  (Sampling)
## Chain 4: Iteration: 13500 / 15000 [ 90%]  (Sampling)
## Chain 4: Iteration: 15000 / 15000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 16.152 seconds (Warm-up)
## Chain 4:                66.824 seconds (Sampling)
## Chain 4:                82.976 seconds (Total)
## Chain 4:
```

```
mdl <- "
  model {

    for (i in 1:31) {
      y[i] ~ dnorm(mu[i], 1/vr)
      mu[i] <- b0 + bage*age[i] + bgen*gen[i] + bbmi*bmi[i] + bhr*hr[i] + brpe*rpe[i]
    }
    b0 ~ dnorm(20,10)
    bage ~ dnorm(0,10)
    bgen ~ dnorm(0,10)
    bbmi ~ dnorm(0,10)
    bhr ~ dnorm(0,10)
    brpe ~ dnorm(0,10)
    vr ~ dgamma(4,.25)
  }
"

writeLines(mdl, 'vo2reg.txt')
```

```r
y <- dat$MaxVO2ML
age <- dat$Age1
gen <- dat$Gender
bmi <- dat$BMI
hr <- dat$HR
rpe <- dat$RPE

data.jags <- c('y', 'age', 'gen', 'bmi', 'hr', 'rpe')
parms <- c('b0', 'bage', 'bgen', 'bbmi', 'bhr', 'brpe', 'vr')

vo2reg.sim <- jags(data=data.jags, inits = NULL, parameters.to.save = parms,
                   model.file = 'vo2reg.txt', n.iter = 4000, n.burnin = 1000,
                   n.chains = 4, n.thin = 3)
```

```
## module glm loaded

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 31
##    Unobserved stochastic nodes: 7
##    Total graph size: 834
##
## Initializing model
```

```r
vo2reg.sim
```

```
## Inference for Bugs model at "vo2reg.txt", fit using jags,
##  4 chains, each with 4000 iterations (first 1000 discarded), n.thin = 3
##  n.sims = 4000 iterations saved
##          mu.vect sd.vect    2.5%     25%     50%     75%   97.5%  Rhat n.eff
## b0        20.028   0.316  19.416  19.812  20.021  20.247  20.634 1.001  4000
## bage       0.333   0.293  -0.235   0.133   0.332   0.531   0.903 1.001  4000
## bbmi      -0.541   0.192  -0.906  -0.670  -0.545  -0.417  -0.147 1.001  4000
## bgen      -0.003   0.317  -0.619  -0.218  -0.004   0.211   0.609 1.001  4000
## bhr        0.170   0.036   0.099   0.146   0.169   0.194   0.242 1.001  4000
## brpe       0.050   0.283  -0.491  -0.143   0.052   0.238   0.605 1.001  4000
## vr        33.999   6.857  22.337  29.197  33.251  38.036  49.379 1.001  4000
## deviance 206.097   3.661 200.160 203.461 205.644 208.258 214.325 1.001  4000
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 6.7 and DIC = 212.8
## DIC is an estimate of expected predictive error (lower deviance is better).
```