ps9 solutions template

] # Question 1

prelims

```
# import data
rm(list=ls())
set.seed(123)

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)
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
library(loo)
```

```
## This is loo version 2.4.1
```

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

```
## - As of v2.0.0 loo defaults to 1 core but we recommend using as many as possible. Use the 'cores' a
```

```
##
## Attaching package: 'loo'
```

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

```
options(mc.cores = parallel::detectCores())

data = read.csv("~/Desktop/Wages1.csv")
ydat = data$wage
ydat = log(ydat)
x1dat = data$school
x1dat = x1dat #- mean(x1dat)
x2dat = data$exper
x2dat = x2dat #- mean(x2dat)
x3dat = data$sex
x3new = rep(0, length(x3dat))
```

```
for (i in 1:length(x3dat)){
  if (x3dat[i] == "female"){
    x3new[i] = 1
  }
}
x3dat = x3new
```

step 1: Compare models using WAIC

```
computeWAIC  = function(aModel, ydat, x1dat, x2dat, x3dat) {
  fit = sampling(aModel, iter = 1000, chains = 4, data = list(N = length(ydat), y = ydat, x1=x1dat, x2
  logLike = extract_log_lik(fit, 'logLike')
  WAIC = waic(logLike)
  return(WAIC)
}

modelLinearWAIC = stan_model("~/Desktop/linear.stan")
modelInteractionWAIC = stan_model("~/Desktop/interaction.stan")

normalWAIC  = computeWAIC(modelLinearWAIC, ydat, x1dat, x2dat, x3dat)
print(normalWAIC)
```

```
##
## Computed from 2000 by 3294 log-likelihood matrix
##
##           Estimate    SE
## elpd_waic  -2944.0  73.1
## p_waic         6.5   0.7
## waic        5888.0 146.1
```

```
interactionWAIC  = computeWAIC(modelInteractionWAIC, ydat, x1dat, x2dat, x3dat)
print(interactionWAIC)
```

```
##
## Computed from 2000 by 3294 log-likelihood matrix
##
##           Estimate    SE
## elpd_waic  -2877.5  74.1
## p_waic         9.8   0.9
## waic        5755.0 148.2
```

step 2: Compare models using CV estimates of the deviance

```
kFold  = function(aModel, testIndeces, y, x1, x2, x3) {
  nFolds = length(testIndeces)

  pointLogLikeTotal = vector()

  for (i in 1:nFolds) {
    print(paste("iteration #", i))
    yTest = y[testIndeces[[i]]]
    yTrain = y[-testIndeces[[i]]]
```

```
    x1Test = x1[testIndeces[[i]]]
    x1Train = x1[-testIndeces[[i]]]
    x2Test = x2[testIndeces[[i]]]
    x2Train = x2[-testIndeces[[i]]]
    x3Test = x3[testIndeces[[i]]]
    x3Train = x3[-testIndeces[[i]]]
    fit = sampling(aModel, iter = 1000, chains = 4,
                   data = list(NTest = length(yTest), NTrain = length(yTrain),
                               yTrain = yTrain, yTest = yTest, x1Train = x1Train,
                               x1Test = x1Test, x2Train = x2Train, x2Test = x2Test,
                               x3Train = x3Train, x3Test = x3Test), refresh=0)
    logLike = extract_log_lik(fit, 'logLike')
    pointLogLikeTemp = colMeans(logLike)
    pointLogLikeTotal = c(pointLogLikeTotal, pointLogLikeTemp)
  }

  return(pointLogLikeTotal)
}

modelLinearCV = stan_model("~/Desktop/linear_cv.stan")
modelInteractionCV = stan_model("~/Desktop/interaction_cv.stan")
```

```
## Warning in readLines(file, warn = TRUE): incomplete final line found on '/Users/
## alexandraklipfel/Desktop/interaction_cv.stan'
```

```
## create folds using caret package
testIndeces = createFolds(ydat, k = 9, list = TRUE, returnTrain = FALSE)

## compute ELLPD using cross validation
ellpdLinear  = sum(kFold(modelLinearCV, testIndeces, ydat, x1dat, x2dat, x3dat))
```

```
## [1] "iteration # 1"
## [1] "iteration # 2"
## [1] "iteration # 3"
## [1] "iteration # 4"
## [1] "iteration # 5"
## [1] "iteration # 6"
## [1] "iteration # 7"
## [1] "iteration # 8"
## [1] "iteration # 9"
```

```
ellpdInteraction = sum(kFold(modelInteractionCV, testIndeces, ydat, x1dat, x2dat, x3dat))
```

```
## [1] "iteration # 1"
## [1] "iteration # 2"
## [1] "iteration # 3"
## [1] "iteration # 4"
## [1] "iteration # 5"
## [1] "iteration # 6"
## [1] "iteration # 7"
## [1] "iteration # 8"
## [1] "iteration # 9"
```

```
# output
print(paste("Deviance-CV for Linear Model    = ", -2 * round(ellpdLinear,2)))
```

```
## [1] "Deviance-CV for Linear Model    =  5888.12"
```

```
print(paste("Devaince-CV for Interaction Model = ", -2 * round(ellpdInteraction,2)))
```

```
## [1] "Devaince-CV for Interaction Model =  5758.22"
```

step 3

WAIC results: Linear Model: 5887.8 Interaction Model: 5755.4 According to the WAIC estimate, the Interaction Model is superior to the Linear Model because the WAIC estimate of the deviance is lower in the case of the Interaction Model.

CV results: Linear Model: 5895.6 Interaction Model: 5765.6 Likewise, according to the CV estimate, the Interaction Model is superior to the Linear Model because the CV estimate of the deviance is lower in the case of the Interaction Model.

Thus, both methods of estimating the deviance agree and provide consistent answers. We conclude that the Interaction Model has a better predictive accuracy out of sample.

Question 2

prelims

```
rm(list=ls())
set.seed(123)
```

step 1

```
numStep1 = c()
numStep2 = c()
###repeat 1000 times:
for (j in 1:1000){
  nObs = 100
  nPreds = 50
  #generate 100 * 51 random values from N(0,1)
  dat50 = rnorm(nObs * (1 + nPreds))
  #create a matrix of the values
  dataset50 = matrix(data = dat50, nrow = nObs, ncol = nPreds + 1)
  dframe50 = as.data.frame(dataset50)
  strs50 = c("y50")
  for (k in 1:nPreds){
    strs50 = c(strs50, paste("x", toString(k), sep=""))
  }
  colnames(dframe50) = c(strs50)
  #linear model
  model50 = lm(y50 ~ ., data=dframe50)
  vals50 = summary(model50)$coefficients
  sig_vals50 = vals50[ ,4]
  #count number of predictors with val < 0.05
  counter1 = 0
  for (i in 1:length(sig_vals50)){
```

```
    if (sig_vals50[i] <= 0.05){
      counter1 = counter1 + 1
    }
  }
  #now we check for the significant predictors (those with a val < 0.25)
  keep50 = c(1)
  for (i in 1:length(sig_vals50)){
    if (sig_vals50[i] <= 0.25){
      keep50 = append(keep50, i)
    }
  }
  new_matrix50 = dataset50[ , keep50]
  new_dframe50 = as.data.frame(new_matrix50)
  new_names50 = c("ynew50")
  for (k in 1:(length(keep50)-1)){
    new_names50 = c(new_names50, paste("x", toString(k), sep=""))
  }
  colnames(new_dframe50) = c(new_names50)

  #new regression:
  new_model50 = lm(ynew50 ~ ., data=new_dframe50)
  new_vals50 = summary(new_model50)$coefficients
  new_sig_vals50 = new_vals50[ ,4]

  #count number of predictors with val < 0.05
  counter2 = 0
  for (i in 1:length(new_sig_vals50)){
    if (new_sig_vals50[i] <= 0.05){
      counter2 = counter2 + 1
    }
  }
  numStep1 = c(numStep1, counter1)
  numStep2 = c(numStep2, counter2)
}
```

```
## Warning in summary.lm(new_model50): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(new_model50): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(new_model50): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(new_model50): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(new_model50): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(new_model50): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(new_model50): essentially perfect fit: summary may be
```