Lab6

Jun Rao

10/1/2019

# Model selection in R and Stan

I have provided you with a data set in which you have measured outcome and you have measured two input variables and . Your full, most complex linear model that we will consider is:

Therefore, we will ignore any multiplicative interactions between the inputs. Importantly, remember from your previous statistics classes that we can condense linear regression into matrix notation: , where is a row-vector that holds the slopes and is an input matrix of dimensions (), where is the number of data points and is the number of input variables.

## Task 1 (15 points)

#--------------------------------------  
#--------------------------------------  
# You may need to install some of these packages  
library(tidyverse)  
library(rstan)  
library(loo)  
rstan\_options(auto\_write = TRUE)  
options(mc.cores = parallel::detectCores())  
#--------------------------------------  
#--------------------------------------

*1.Import your data (code provided below), and then center and scale your input variables. Center by the mean and scale by 2standard deviation. This assignment will not work unless you do this step properly. Hint: You must center and scale the two variable columns separately (i.e., each variable will have its own mean and st. deviation).*

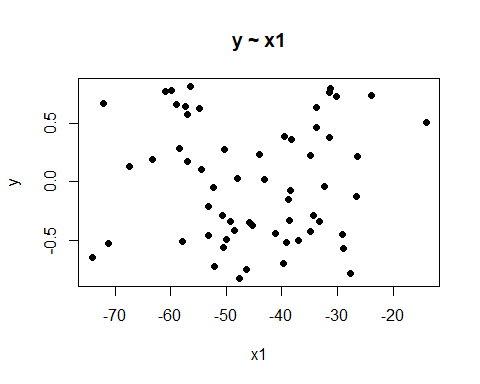
# Read in the .CSV file:  
data\_raw = read\_csv("lab6\_data.csv")  
#observations  
n\_sample = nrow(data\_raw)  
n\_sample

## [1] 60

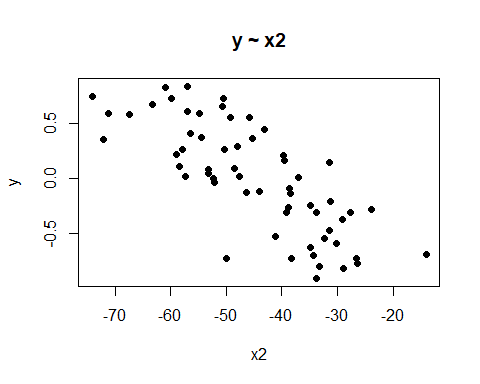
#y  
y = data\_raw$y  
  
#x1  
x1 = data\_raw$x1  
x1\_mean = mean(x1)  
x1\_sd = sd(x1)  
#X2  
x2 = data\_raw$x2  
x2\_mean = mean(x2)  
x2\_sd = sd(x2)  
  
#center and standardize:  
x1\_scaled = (x1 - x1\_mean)/(2\*x1\_sd)  
x2\_scaled = (x2 - x2\_mean)/(2\*x2\_sd)

*2.Create two scatterplots of your outcome versus your two inputs.*

plot(y, x1\_scaled, main="y ~ x1",xlab="x1", ylab="y", pch=19)



plot(y, x2\_scaled, main="y ~ x2",xlab="x2", ylab="y", pch=19)



*3.Look at the provided Stan script, come to an understanding of the data inputs required to run this Stan model. Use the centered and scaled inputs to set up and run the Stan model, by adapting previously provided code. Hint: You will have to monitor the parameter.*

stan\_data = list(n\_input = 2,  
 n\_sample = n\_sample,  
 x\_mat = cbind(x1\_scaled,x2\_scaled),  
 y\_vec = y)  
  
params\_monitor = c("beta", "alpha", "sigma", "log\_lik")  
  
test\_fit = stan(file = "Multi\_LinReg.stan",  
 data = stan\_data,  
 pars = params\_monitor,  
 chains = 1, # How many chains to run  
 iter = 10, # How many iterations per chain  
 algorithm="NUTS")

##   
## SAMPLING FOR MODEL 'Multi\_LinReg' 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: WARNING: No variance estimation is  
## Chain 1: performed for num\_warmup < 20  
## Chain 1:   
## Chain 1: Iteration: 1 / 10 [ 10%] (Warmup)  
## Chain 1: Iteration: 2 / 10 [ 20%] (Warmup)  
## Chain 1: Iteration: 3 / 10 [ 30%] (Warmup)  
## Chain 1: Iteration: 4 / 10 [ 40%] (Warmup)  
## Chain 1: Iteration: 5 / 10 [ 50%] (Warmup)  
## Chain 1: Iteration: 6 / 10 [ 60%] (Sampling)  
## Chain 1: Iteration: 7 / 10 [ 70%] (Sampling)  
## Chain 1: Iteration: 8 / 10 [ 80%] (Sampling)  
## Chain 1: Iteration: 9 / 10 [ 90%] (Sampling)  
## Chain 1: Iteration: 10 / 10 [100%] (Sampling)  
## Chain 1:   
## Chain 1: Elapsed Time: 0.001 seconds (Warm-up)  
## Chain 1: 0 seconds (Sampling)  
## Chain 1: 0.001 seconds (Total)  
## Chain 1:

## Now we will run our full model:  
# How many samples do we want of each parameter, from each chain?  
n\_mc\_samples = 1000  
# How much burn-in?  
n\_burn = 500  
# How much thinning? (take the ith value of the chain)  
n\_thin = 3  
# Total iterations needed:  
n\_iter\_total = (n\_mc\_samples \* n\_thin) + n\_burn  
model\_fit =  
 stan(fit = test\_fit, # So it knows we're already compiled  
 file = "Multi\_LinReg.stan",  
 data = stan\_data,  
 pars = params\_monitor,  
 chains = 3,  
 warmup = n\_burn,  
 thin = n\_thin,  
 iter = n\_iter\_total,  
 algorithm="NUTS")  
model\_out = rstan::extract(model\_fit)  
str(model\_out)

## List of 5  
## $ beta : num [1:3000, 1:2] 7.454 0.995 3.022 3.008 0.847 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ iterations: NULL  
## .. ..$ : NULL  
## $ alpha : num [1:3000(1d)] -44.3 -44.8 -44.7 -44.4 -45.2 ...  
## ..- attr(\*, "dimnames")=List of 1  
## .. ..$ iterations: NULL  
## $ sigma : num [1:3000(1d)] 10.26 6.96 8.57 7.74 10.23 ...  
## ..- attr(\*, "dimnames")=List of 1  
## .. ..$ iterations: NULL  
## $ log\_lik: num [1:3000, 1:60] -3.3 -2.87 -3.13 -3.05 -3.28 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ iterations: NULL  
## .. ..$ : NULL  
## $ lp\_\_ : num [1:3000(1d)] -163 -161 -158 -159 -161 ...  
## ..- attr(\*, "dimnames")=List of 1  
## .. ..$ iterations: NULL

*4.Show me that your model has converged.*

# Check Convergence:  
tail(summary(model\_fit)$summary[, "Rhat"],1)

## lp\_\_   
## 1.000437

***Since the value at here less than 1.1, hence we can conclude that the model is converged***

1. What are the median and 95% credible intervals for each model parameter?

#the median  
summary(model\_fit)$summary[1:4, "50%"]

## beta[1] beta[2] alpha sigma   
## 1.364938 -20.242058 -44.960697 8.488155

#the 95% credible intervals  
summary(model\_fit)$summary[1:4, "2.5%"]

## beta[1] beta[2] alpha sigma   
## -3.079679 -24.527465 -47.117754 7.145730

summary(model\_fit)$summary[1:4, "97.5%"]

## beta[1] beta[2] alpha sigma   
## 5.805474 -15.815826 -42.934438 10.313266

## Task 2 (30 points)

*1.Re-run your model two more times to store the nested model versions (e.g., with only input 1, and then with only input 2). Hint: You must store these models in objects with different names, however, you will not have to re-write the Stan model at all. Just think about how the inputs might change.*

#Only input1  
stan\_data\_1 = list(n\_input = 1,  
 n\_sample = n\_sample,  
 x\_mat = cbind(x1\_scaled),  
 y\_vec = y)  
#Only input2  
stan\_data\_2 = list(n\_input = 1,  
 n\_sample = n\_sample,  
 x\_mat = cbind(x2\_scaled),  
 y\_vec = y)  
  
params\_monitor = c("beta", "alpha", "sigma", "log\_lik")  
  
test\_fit\_1 = stan(file = "Multi\_LinReg.stan",  
 data = stan\_data\_1,  
 pars = params\_monitor,  
 chains = 1, # How many chains to run  
 iter = 10, # How many iterations per chain  
 algorithm="NUTS")

##   
## SAMPLING FOR MODEL 'Multi\_LinReg' 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: WARNING: No variance estimation is  
## Chain 1: performed for num\_warmup < 20  
## Chain 1:   
## Chain 1: Iteration: 1 / 10 [ 10%] (Warmup)  
## Chain 1: Iteration: 2 / 10 [ 20%] (Warmup)  
## Chain 1: Iteration: 3 / 10 [ 30%] (Warmup)  
## Chain 1: Iteration: 4 / 10 [ 40%] (Warmup)  
## Chain 1: Iteration: 5 / 10 [ 50%] (Warmup)  
## Chain 1: Iteration: 6 / 10 [ 60%] (Sampling)  
## Chain 1: Iteration: 7 / 10 [ 70%] (Sampling)  
## Chain 1: Iteration: 8 / 10 [ 80%] (Sampling)  
## Chain 1: Iteration: 9 / 10 [ 90%] (Sampling)  
## Chain 1: Iteration: 10 / 10 [100%] (Sampling)  
## Chain 1:   
## Chain 1: Elapsed Time: 0.001 seconds (Warm-up)  
## Chain 1: 0.003 seconds (Sampling)  
## Chain 1: 0.004 seconds (Total)  
## Chain 1:

test\_fit\_2 = stan(file = "Multi\_LinReg.stan",  
 data = stan\_data\_2,  
 pars = params\_monitor,  
 chains = 1, # How many chains to run  
 iter = 10, # How many iterations per chain  
 algorithm="NUTS")

##   
## SAMPLING FOR MODEL 'Multi\_LinReg' 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: WARNING: No variance estimation is  
## Chain 1: performed for num\_warmup < 20  
## Chain 1:   
## Chain 1: Iteration: 1 / 10 [ 10%] (Warmup)  
## Chain 1: Iteration: 2 / 10 [ 20%] (Warmup)  
## Chain 1: Iteration: 3 / 10 [ 30%] (Warmup)  
## Chain 1: Iteration: 4 / 10 [ 40%] (Warmup)  
## Chain 1: Iteration: 5 / 10 [ 50%] (Warmup)  
## Chain 1: Iteration: 6 / 10 [ 60%] (Sampling)  
## Chain 1: Iteration: 7 / 10 [ 70%] (Sampling)  
## Chain 1: Iteration: 8 / 10 [ 80%] (Sampling)  
## Chain 1: Iteration: 9 / 10 [ 90%] (Sampling)  
## Chain 1: Iteration: 10 / 10 [100%] (Sampling)  
## Chain 1:   
## Chain 1: Elapsed Time: 0 seconds (Warm-up)  
## Chain 1: 0 seconds (Sampling)  
## Chain 1: 0 seconds (Total)  
## Chain 1:

## Now we will run our full model:  
# How many samples do we want of each parameter, from each chain?  
n\_mc\_samples = 1000  
# How much burn-in?  
n\_burn = 500  
# How much thinning? (take the ith value of the chain)  
n\_thin = 3  
# Total iterations needed:  
n\_iter\_total = (n\_mc\_samples \* n\_thin) + n\_burn  
  
model\_fit\_1 =  
 stan(fit = test\_fit\_1, # So it knows we're already compiled  
 file = "Multi\_LinReg.stan",  
 data = stan\_data\_1,  
 pars = params\_monitor,  
 chains = 3,  
 warmup = n\_burn,  
 thin = n\_thin,  
 iter = n\_iter\_total,  
 algorithm="NUTS")  
  
model\_fit\_2 =  
 stan(fit = test\_fit\_2, # So it knows we're already compiled  
 file = "Multi\_LinReg.stan",  
 data = stan\_data\_2,  
 pars = params\_monitor,  
 chains = 3,  
 warmup = n\_burn,  
 thin = n\_thin,  
 iter = n\_iter\_total,  
 algorithm="NUTS")

tail(summary(model\_fit\_1)$summary[, "Rhat"],1)

## lp\_\_   
## 0.999492

tail(summary(model\_fit\_2)$summary[, "Rhat"],1)

## lp\_\_   
## 1.001422

#the median  
summary(model\_fit\_1)$summary[1:3, "50%"]

## beta[1] alpha sigma   
## -0.01201261 -44.95451231 13.18712198

summary(model\_fit\_2)$summary[1:3, "50%"]

## beta[1] alpha sigma   
## -20.23118 -45.01087 8.40686

#the 95% credible intervals  
summary(model\_fit\_1)$summary[1:3, "2.5%"]

## beta[1] alpha sigma   
## -6.890235 -48.378096 11.128468

summary(model\_fit\_1)$summary[1:3, "97.5%"]

## beta[1] alpha sigma   
## 6.917912 -41.619817 15.897878

#the 95% credible intervals  
summary(model\_fit\_2)$summary[1:3, "2.5%"]

## beta[1] alpha sigma   
## -24.407160 -47.182270 7.070992

summary(model\_fit\_2)$summary[1:3, "97.5%"]

## beta[1] alpha sigma   
## -15.88004 -42.86738 10.19127

*2.Use the () function to calculate the LOO-IC for each model. Follow my example code below, and make sure to store these into separate objects for each model run.*

# Because eval = FALSE this code chunk will not run.   
# You will have to input your correct objects into this function.  
  
# Calculate the LOO and WAIC for a single model:  
# I'm doing this for the 'full' or most complex model  
log\_lik\_full = extract\_log\_lik(model\_fit)  
log\_lik\_1 = extract\_log\_lik(model\_fit\_1)  
log\_lik\_2 = extract\_log\_lik(model\_fit\_2)  
  
loo\_full = loo(log\_lik\_full)

## Warning: Relative effective sample sizes ('r\_eff' argument) not specified.  
## For models fit with MCMC, the reported PSIS effective sample sizes and   
## MCSE estimates will be over-optimistic.

loo\_sub1 = loo(log\_lik\_1)

## Warning: Relative effective sample sizes ('r\_eff' argument) not specified.  
## For models fit with MCMC, the reported PSIS effective sample sizes and   
## MCSE estimates will be over-optimistic.

loo\_sub2 = loo(log\_lik\_2)

## Warning: Relative effective sample sizes ('r\_eff' argument) not specified.  
## For models fit with MCMC, the reported PSIS effective sample sizes and   
## MCSE estimates will be over-optimistic.

waic\_full = waic(log\_lik\_full)

## Warning: 2 (3.3%) p\_waic estimates greater than 0.4. We recommend trying  
## loo instead.

waic\_sub1 = waic(log\_lik\_1)  
waic\_sub2 = waic(log\_lik\_2)

## Warning: 1 (1.7%) p\_waic estimates greater than 0.4. We recommend trying  
## loo instead.

loo\_full

##   
## Computed from 3000 by 60 log-likelihood matrix  
##   
## Estimate SE  
## elpd\_loo -216.1 5.3  
## p\_loo 4.0 0.8  
## looic 432.3 10.7  
## ------  
## Monte Carlo SE of elpd\_loo is 0.0.  
##   
## All Pareto k estimates are good (k < 0.5).  
## See help('pareto-k-diagnostic') for details.

loo\_sub1

##   
## Computed from 3000 by 60 log-likelihood matrix  
##   
## Estimate SE  
## elpd\_loo -242.2 5.0  
## p\_loo 3.0 0.6  
## looic 484.5 10.0  
## ------  
## Monte Carlo SE of elpd\_loo is 0.0.  
##   
## All Pareto k estimates are good (k < 0.5).  
## See help('pareto-k-diagnostic') for details.

loo\_sub2

##   
## Computed from 3000 by 60 log-likelihood matrix  
##   
## Estimate SE  
## elpd\_loo -215.2 5.4  
## p\_loo 2.9 0.7  
## looic 430.3 10.7  
## ------  
## Monte Carlo SE of elpd\_loo is 0.0.  
##   
## All Pareto k estimates are good (k < 0.5).  
## See help('pareto-k-diagnostic') for details.

waic\_full

##   
## Computed from 3000 by 60 log-likelihood matrix  
##   
## Estimate SE  
## elpd\_waic -216.1 5.3  
## p\_waic 3.9 0.8  
## waic 432.2 10.6

## Warning: 2 (3.3%) p\_waic estimates greater than 0.4. We recommend trying  
## loo instead.

waic\_sub1

##   
## Computed from 3000 by 60 log-likelihood matrix  
##   
## Estimate SE  
## elpd\_waic -242.2 5.0  
## p\_waic 3.0 0.6  
## waic 484.4 10.0

waic\_sub2

##   
## Computed from 3000 by 60 log-likelihood matrix  
##   
## Estimate SE  
## elpd\_waic -215.2 5.4  
## p\_waic 2.9 0.7  
## waic 430.3 10.7

## Warning: 1 (1.7%) p\_waic estimates greater than 0.4. We recommend trying  
## loo instead.

*3.Use the () function to compare all three of your nested models using the LOO-CV. Your code will look something like:*

# Because eval = FALSE this code chunk will not run.   
# You will have to input your correct objects into this function.  
  
loo::compare(loo\_full, loo\_sub1, loo\_sub2)

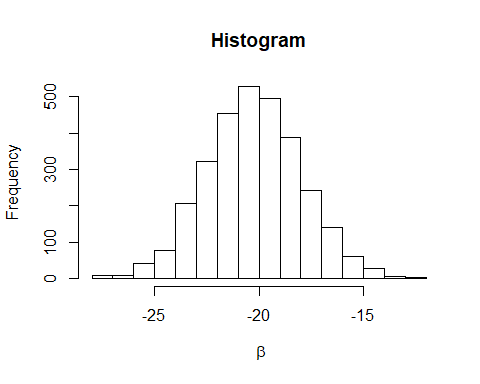
## elpd\_diff se\_diff elpd\_loo p\_loo looic   
## loo\_sub2 0.0 0.0 -215.2 2.9 430.3  
## loo\_full -1.0 0.7 -216.1 4.0 432.3  
## loo\_sub1 -27.1 4.9 -242.2 3.0 484.5

*4.Give a brief interpretation of your results of this statistical analysis and model comparison, citing the quantitative metrics as support for your conclusions. Remember your goal is to choose the most parsimonious model.*

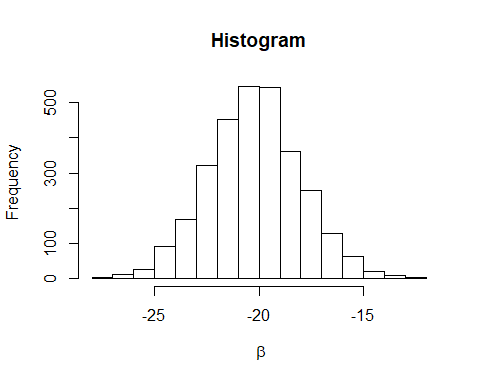
***According to the result shown in the question 3, we may could conclude that the best model is model\_fit\_2, which only have the 2nd input. The value of elpd\_loo is -215.1. Compare to the full model, it is also simpler (because we only have one input value) even the has the close elpd\_loo value.***

*5.Show me visually and quantitatively that the variance of the posterior samples of from the most complex model is than the variance of the same estimate from the simpler model. The difference will be subtle, but you can imagine that if you had many input variables the effect would become compounded.*

hist(extract(model\_fit)$beta[,2], main = paste("Histogram"),xlab=expression(~beta))



hist(extract(model\_fit\_2)$beta, main = paste("Histogram"),xlab=expression(~beta))



var(extract(model\_fit)$beta[,2])

## [1] 5.113633

var(extract(model\_fit\_2)$beta)

## [,1]  
## [1,] 4.843807