ISYE 6420 Bayesian Statistics

Homework 5

November 8th, 2021

1.

After running the provided python code we get the following output for a 20 train and test datasets for the x1, x2 and y variables:

```
x1_train = x1[0:20]
x1_test = x1[20:40]
x2_train = x2[0:20]
x2_test = x2[20:40]
y_train = y[0:20]
y_test = y[20:40]
```

```
x1_train
```

```
array([0.99279513, 0.94322305, 0.38720441, 0.5887295 , 0.63816661,
       0.6159705 , 0.34452812, 0.4529029 , 0.44338295, 0.96514942,
       0.15677267, 0.91908408, 0.72519724, 0.77870175, 0.09982977,
       0.30492805, 0.97771271, 0.99865307, 0.1800832 , 0.64258689])
```

```
x2_train
```

```
array([ 4.,  1., 10.,  6.,  7.,  8.,  5.,  5.,  5.,  7.,  6.,  4.,  5.,
        3.,  2.,  2.,  3.,  4.,  2.,  5.])
```

```
y_train
```

```
array([5.42943534, 8.18432937, 0.52303879, 2.56944025, 2.89974   ,
       0.85021538, 3.00229687, 3.34767405, 2.59278822, 3.48642516,
       0.16332507, 4.85961734, 3.31460175, 5.27560075, 1.33438488,
       1.7932343 , 6.01874266, 6.30550303, 1.7901943 , 3.20721695])
```

```
x1_test
```

```
array([0.37800106, 0.67726028, 0.79880161, 0.59200836, 0.75977296,
       0.87335856, 0.94805373, 0.43693321, 0.38912988, 0.8407542 ,
       0.42581398, 0.15843499, 0.68589242, 0.17248048, 0.33790474,
       0.3593793 , 0.85741877, 0.39973591, 0.50890528, 0.19253303])
```

```
x2_test
```

```
array([ 8.,  2.,  2.,  2.,  6.,  3.,  5.,  6.,  5.,  3.,  6., 10.,  9.,
        7.,  9.,  1.,  8.,  2.,  3.,  3.])
```

```
y_test
```

```
array([ 1.72843681,  5.74619857,  5.93401725,  3.45236797,  3.30630499,
        7.46063246,  5.48421555,  2.44700953,  2.3479734 ,  5.76065719,
        0.88238019, -2.40972398,  1.69409053,  0.45375947, -0.74101511,
        2.98610167,  2.65401765,  2.66206988,  2.21747678,  1.14111676])
```

Provided is the implementation in WinBugs:

```
model{
for (i in 1:n){
  y[i] ~ dnorm(mu[i], tau)
  mu[i] <- beta0 + beta1 * x1[i] + beta2 * x2[i]
  }
# Priors
beta0 ~ dnorm(0, 0.001)
beta1 ~ dnorm(0, 0.001)
beta2 ~ dnorm(0, 0.001)
tau ~ dgamma(0.001, 0.001)

for (i in 1:n){
  mu2[i] <- beta0 + beta1 * x1test[i] + beta2 * x2test[i]
```

```
  y_pred[i] ~ dnorm(mu2[i], tau)
  sq_error[i] <- (y_pred[i] - ytest[i]) * (y_pred[i] - ytest[i])
  }
  MSE <- mean(sq_error[])
}

list(n=20, y =c(5.42943534, 8.18432937, 0.52303879, 2.56944025,
2.89974, 0.85021538, 3.00229687, 3.34767405, 2.59278822,
3.48642516, 0.16332507, 4.85961734, 3.31460175, 5.27560075,
1.33438488,
1.7932343 , 6.01874266, 6.30550303, 1.7901943 , 3.20721695),
x1 = c(0.99279513, 0.94322305, 0.38720441, 0.5887295 , 0.63816661,
 0.6159705 , 0.34452812, 0.4529029 , 0.44338295, 0.96514942,
0.15677267, 0.91908408, 0.72519724, 0.77870175, 0.09982977,
0.30492805, 0.97771271, 0.99865307, 0.1800832 , 0.64258689),
x2 = c(4, 1, 10, 6, 7, 8, 5, 5, 5, 7, 6, 4, 5, 3, 2, 2, 3, 4, 2,
5),
ytest =c(1.72843681, 5.74619857,  5.93401725,  3.45236797,
3.30630499,  7.46063246, 5.48421555,  2.44700953,  2.3479734 ,
5.76065719,  0.88238019, -2.40972398,  1.69409053,  0.45375947,
-0.74101511,  2.98610167, 2.65401765,  2.66206988,  2.21747678,
1.14111676),
x1test = c(0.37800106, 0.67726028, 0.79880161, 0.59200836,
0.75977296, 0.87335856, 0.94805373, 0.43693321, 0.38912988,
0.8407542 , 0.42581398, 0.15843499, 0.68589242, 0.17248048,
0.33790474, 0.3593793 , 0.85741877, 0.39973591, 0.50890528,
0.19253303),
x2test = c(8, 2,  2,  2,  6,  3,  5,  6,  5,  3,  6, 10,  9,  7,
9,  1,  8,  2,  3,  3))


list(beta0=2, beta1=6, beta2=-0.5,tau=0.8)
```

After burning the first 100,000 observations, we get the following summary output for our variables:

Node statistics

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
MSE	1.407	0.5014	0.001645	0.6941	1.321	2.622	100001	100000
beta0	2.066	0.5702	0.001813	0.9345	2.066	3.197	100001	100000
beta1	5.547	0.6075	0.001949	4.342	5.55	6.744	100001	100000
beta2	-0.4447	0.08116	2.495E-4	-0.6057	-0.4443	-0.285	100001	100000
tau	1.789	0.6145	0.002155	0.7933	1.721	3.174	100001	100000

$\beta_0 : 2.066$
$\beta_1 : 5.547$
$\beta_2 : 0.4447$
$\sigma : 1.789$

We can see that these are somewhat close to the true observations of 2, 6, -0.5 and 0.8, respectively.

Our MSE yielded a value of 1.407.

2.
i)
a)
We put together the following additional code on top of BFReg in order to accomodate for a net new person.

q2_part1a

```
model{
for(i in 1:N){
BF[i] ~ dnorm(mu[i], tau)
BB[i] <- BAI[i] * BMI[i]
mu[i] <- b0 + b1 * Age[i] + b2*BAI[i] + b3*BMI[i] + b4*BB[i] + b5* Gender[i]
}

b0 ~ dnorm(0, 0.001)
b1 ~ dnorm(0, 0.001)
b2 ~ dnorm(0, 0.001)
b3 ~ dnorm(0, 0.001)
b4 ~ dnorm(0, 0.001)
b5 ~ dnorm(0, 0.001)
tau ~ dgamma(0.001, 0.001)

PersonAge <- 35
PersonBAI <- 26
PersonBMI <- 20
PersonGender <- 0
PersonBB <- 520
PersonBF <- b0 + b1 * PersonAge + b2*PersonBAI + b3*PersonBMI + b4*PersonBB +
b5* PersonGender
PersonBFPredict ~ dnorm(PersonBF, tau)
}

DATA
list(N=3200)

➔ BFData 🡐

INITS
list(b0=1, b1=0, b2=0, b3=0, b4=0, b5=0, tau=1)
```

After doing so we burn the first 1000 samples and generate our statistics from observation 1001 to 11000, in order to generate stats for the remaining 10000 observations. In this case we're running our model using all coefficients and displaying the yielded results:

Node statistics

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
PersonBFPredict	15.04	4.044	0.04472	7.034	15.09	22.95	1001	10000
b0	-33.07	2.211	0.2197	-36.96	-33.19	-27.69	1001	10000
b1	0.07394	0.007642	4.5E-4	0.05899	0.07396	0.0889	1001	10000
b2	0.7792	0.07194	0.00708	0.6396	0.7799	0.9217	1001	10000
b3	1.894	0.09394	0.009307	1.633	1.901	2.044	1001	10000
b4	-0.0241	0.00258	2.563E-4	-0.02872	-0.02421	-0.01811	1001	10000
b5	10.58	0.2372	0.01737	10.11	10.59	11.02	1001	10000
tau	0.06045	0.00151	1.701E-5	0.05751	0.06043	0.06345	1001	10000

From the following figure we can see our updated formula with the following coefficients whenever we include all predictors:

$$BF = -33.07 + 0.07394 Age + 0.7792 BAI + 1.894 BMI - 0.0241 BB + 10.58 Gender$$

Determing the single best predictor. For this we can reconstruct the problem by using one coefficient at a time (including the intercept), and determine the impact on the r2 value. We will add a portion to our code in order to calculate the R^2 calculation which is just $1 - \frac{RSS}{TSS}$, seen below:

```
model{
for(i in 1:N){
BF[i] ~ dnorm(mu[i], tau)
BB[i] <- BAI[i] * BMI[i]
# mu[i] <- b0 + b1 * Age[i] + b2*BAI[i] + b3*BMI[i] + b4*BB[i] +
b5* Gender[i]
mu[i] <- b0 + 0* Age[i] + 0*BAI[i] + b3*BMI[i] + 0*BB[i] + 0*
Gender[i]
}

b0 ~ dnorm(0, 0.001)
b1 ~ dnorm(0, 0.001)
b2 ~ dnorm(0, 0.001)
b3 ~ dnorm(0, 0.001)
b4 ~ dnorm(0, 0.001)
b5 ~ dnorm(0, 0.001)
```

```
tau ~ dgamma(0.001, 0.001)

difference <- N -2
sigmasquared <- 1/tau
sse <- difference*sigmasquared
for (i in 1:N) {
cBF[i] <- BF[i] - mean(BF[])
}
sst <- inprod(cBF[], cBF[])
Rsquared <- 1 - sse/sst
}

DATA
list(N=3200)

 BFData

INITS
list(b0=1, b1=0, b2=0, b3=0, b4=0, b5=0, tau=1)
```

In addition for each case will need to modify the formula to cancel certain coefficients with 0 depending on the scenario. For example:

$$mu[i] \ < - \ b0 \ + \ 0* \ Age[i] \ + \ 0*BAI[i] \ + \ b3*BMI[i] \ + \ 0*BB[i] \ + \ 0* \ Gender[i]$$

This is the updated formula in order to calculate b3 coefficient only.

$Case\ 1: \ Age\ used\ (b1),\ all\ other\ set\ to\ 0$

Node statistics

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
Rsquared	0.08362	0.02301	2.004E-4	0.03737	0.08382	0.1275	1001	13000
b0	19.35	0.5255	0.02265	18.33	19.34	20.36	1001	13000
b1	0.2196	0.01292	5.591E-4	0.1947	0.2196	0.2444	1001	13000

$Case\ 2: \ BAI\ used\ (b2),\ all\ other\ set\ to\ 0$

Node statistics

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
Rsquared	0.5467	0.01136	1.113E-4	0.5239	0.5468	0.5684	1001	10000
b0	-5.912	0.5512	0.03762	-6.975	-5.92	-4.829	1001	10000
b2	1.181	0.01893	0.001291	1.145	1.182	1.218	1001	10000

$Case\ 3:\ BMI\ used\ (b3),\ all\ other\ set\ to\ 0$

Node statistics

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
Rsquared	0.2955	0.01766	1.73E-4	0.2601	0.2957	0.3293	1001	10000
b0	3.68	0.6705	0.04506	2.378	3.67	4.996	1001	10000
b3	0.9592	0.02606	0.00175	0.9087	0.9597	1.01	1001	10000

$Case\ 4:\ BB\ used\ (b4),\ all\ other\ set\ to\ 0$

Node statistics

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
Rsquared	0.4755	0.01315	1.297E-4	0.4491	0.4757	0.5007	1001	10000
b0	11.96	0.3136	0.01264	11.36	11.97	12.58	1001	10000
b4	0.02159	3.989E-4	1.609E-5	0.02082	0.02159	0.02237	1001	10000

$Case\ 5:\ Gender\ used\ (b5),\ all\ other\ set\ to\ 0$

Node statistics

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
Rsquared	0.2309	0.01928	1.912E-4	0.1924	0.231	0.2678	1001	10000
b0	23.7	0.1856	0.003385	23.34	23.7	24.07	1001	10000
b5	7.878	0.2521	0.004639	7.382	7.878	8.385	1001	10000

From these outputs, we can see that the single best predictor for ***BF*** is BAI. Therefore, we will use a model with the BAI coefficient only and all predictors for part b.

b)

With the new formula from part a for all predictors we can see that for a new person with the provided values for each coefficient, we yield a BF prediction of 15.04.

For that same new person using only the BAI coefficient, we get the following result:

Node statistics

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
PersonBFPredict	24.84	5.535	0.05649	13.95	24.84	35.65	1001	10000
b0	-5.956	0.5771	0.03741	-7.028	-5.956	-4.819	1001	10000
b2	1.183	0.01981	0.001285	1.144	1.183	1.22	1001	10000

Meaning that 24.84 is the value of the BF prediction only using BAI, where this is the formula used:

$$BF \;=\; -5.956 \;+\; 1.183BAI$$

3.
Provided is the following OpenBugs code to complete the problem:

```
model{
  for( i in 1 : N ) {
    #Get y response for the observed data
    y[i] ~ dbin(p[i],n[i])
    #Run logistic regression on pass params
    logit(p[i]) <- alpha.init + beta * (x[i] - mean(x[]))
    #Get y prediction
  ypred[i] <- n[i] * p[i]
  }
    alpha <- alpha.init - beta * mean(x[])
    beta ~ dnorm(0.0,0.001)
    alpha.init ~ dnorm(0.0,0.001)

    #Initialize initial value for 2.5 milliamps
    temp.init <- 2.5
    l_aftershock <- alpha + beta * temp.init
    #Calculate proportion of responses after the shock
    prop_aftershock <- exp(l_aftershock)/(1+exp(l_aftershock))
  }

DATA
  list(n = c(70, 70, 70, 70, 70, 70),
  x = c(0, 1, 2, 3, 4, 5),
  y = c(0, 9, 21, 47, 60, 63), N = 6)
```

```
INITS
    list(alpha.init=0, beta=0)
```

Upon execution we can see that we get the following results for the proportion of responses after the shock, and the 95% credible set:

Node statistics

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
alpha	-3.34	0.3278	0.001466	-4.017	-3.331	-2.722	100001	100000
alpha.init	-0.1891	0.1382	5.428E-4	-0.4622	-0.1883	0.08013	100001	100000
beta	1.261	0.1136	5.203E-4	1.046	1.257	1.493	100001	100000
prop_aftershock	0.4531	0.03407	1.336E-4	0.3865	0.4531	0.52	100001	100000

We can see that the proportion of of responsble aftershock is $propaftershock = 0.4531$ and the 95% equitable set is $[0.3865,\ 0.52]$