

1.

Used the below python code for generating 40 triplets. The first 20 will be used for training and next 20 will be used for testing.

```
import numpy as np
```

```
x1 = np.random.uniform(0,1, 40)
x2 = np.floor(10 * np.random.uniform(0,1,40)) + 1
y = 2 + 6 * x1 - 0.5 * x2 + 0.8*np.random.normal(0,1,len(x1))
print (x1)
print (x2)
print (y)
```

Below data was generated for x1:

```
[ 0.30776644  0.14688978  0.94407773  0.2121937  0.14100712  0.53051039
 0.52719547  0.59594426  0.2338234  0.83737082  0.02865486  0.57221981
 0.91032036  0.13443993  0.07194394  0.64478776  0.04066892  0.81104009
 0.30519012  0.18559251  0.22389714  0.80307216  0.09963081  0.01041746
 0.3198728  0.81225209  0.76675059  0.47534412  0.22863295  0.67413502
 0.57031504  0.64708392  0.24195506  0.09851122  0.83330863  0.72088874
 0.84059394  0.65406967  0.25757441  0.86808854]
```

Below data was generated for x2:

```
[ 3.  4.  7.  3.  1.  5. 10.  3.  1.  3.  9.  8.  4.  4.  4.
 2.  7.  5.  9.  5.  4.  9.  5.  7.  4.  4.  3.  3.  7.  1.
 2.  4.  4. 10.  6.  9.  3.  8.  9.  8.]
```

Below data was generated for corresponding y:

```
[ 2.58481756  1.59894676  4.42928523  1.40161133  3.90314646  0.15263601
-0.21382859  4.03975109  3.05529608  5.9338465 -2.90315394  2.14714488
 5.31096849  0.75901141  0.11955305  4.43028162 -1.15643895  3.47590595
 0.48378093  1.01478288  0.47605451  2.15765844  0.767433 -0.60246567
 2.40334584  4.65644967  5.06278152  3.41725078 -1.15329498  4.46191386
 4.45182766  2.78491998  2.08970069 -3.61750259  3.99393386  0.6525259
 6.94657316  0.6530718  0.28327083  1.60945144]
```

Used first 20 of above triplets as my training set for regression in OpenBugs, and the last 20 of above triplets will be used for testing.

Init values set for beta0 is 2, beta1 is 6, beta2 = -0.5, tau/sigma = 0.8

Ran the below code in Openbugs:

```

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 * x1Testing[i] + beta2 * x2Testing[i]
  y_pred[i] ~ dnorm(mu2[i], tau)
  sq_error[i] <- (y_pred[i] - yTesting[i]) * (y_pred[i] - yTesting[i])
}
MSE <- mean(sq_error[])
}

list(n=20, y =c(2.58481756, 1.59894676, 4.42928523, 1.40161133, 3.90314646, 0.15263601, -0.21382859,
4.03975109, 3.05529608, 5.9338465, -2.90315394, 2.14714488, 5.31096849, 0.75901141, 0.11955305,
4.43028162, -1.15643895, 3.47590595, 0.48378093, 1.01478288),
x1 = c(0.30776644, 0.14688978, 0.94407773, 0.2121937, 0.14100712, 0.53051039, 0.52719547, 0.59594426,
0.2338234, 0.83737082, 0.02865486, 0.57221981, 0.91032036, 0.13443993, 0.07194394, 0.64478776, 0.4066892,
0.81104009, 0.30519012, 0.18559251),
x2 = c(3, 4, 7, 3, 1, 5, 10, 3, 1, 3, 9, 8, 4, 4, 4, 2, 7, 5, 9, 5),
yTesting =c(0.47605451, 2.15765844, 0.767433, -0.60246567, 2.40334584, 4.65644967, 5.06278152, 3.41725078,
-1.15329498, 4.46191386, 4.45182766, 2.78491998, 2.08970069, -3.61750259, 3.99393386, 0.6525259,
6.94657316, 0.6530718, 0.28327083, 1.60945144),
x1Testing = c(0.22389714, 0.80307216, 0.09963081, 0.01041746, 0.3198728, 0.81225209, 0.76675059,
0.47534412, 0.22863295, 0.67413502, 0.57031504, 0.64708392, 0.24195506, 0.09851122, 0.83330863,
0.72088874, 0.84059394, 0.65406967, 0.25757441, 0.86808854),
x2Testing = c(4, 9, 5, 7, 4, 4, 3, 3, 7, 1, 2, 4, 4, 10, 6, 9, 3, 8, 9, 8))

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

```

Burned first 100000 samples. Got below statistics for next 100000:

Node statistics								
	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
MSE	1.983	0.7928	0.003186	0.8975	1.837	3.943	100001	100000
beta0	2.362	0.5937	0.005942	1.184	2.365	3.529	100001	100000
beta1	5.502	0.8242	0.005895	3.883	5.501	7.147	100001	100000
beta2	-0.5537	0.09119	7.929E-4	-0.7351	-0.5538	-0.3736	100001	100000
tau	1.018	0.3488	0.001459	0.4521	0.9784	1.811	100001	100000

Therefore, from above

The Bayesian estimator of β_0 is: 2.362 which is not very close but still close to 2.

The Bayesian estimator of β_1 is: 5.502 which is not very close but still close to 6.

The Bayesian estimator of β_2 is: -0.5537 which is not very close but still close to -0.5.

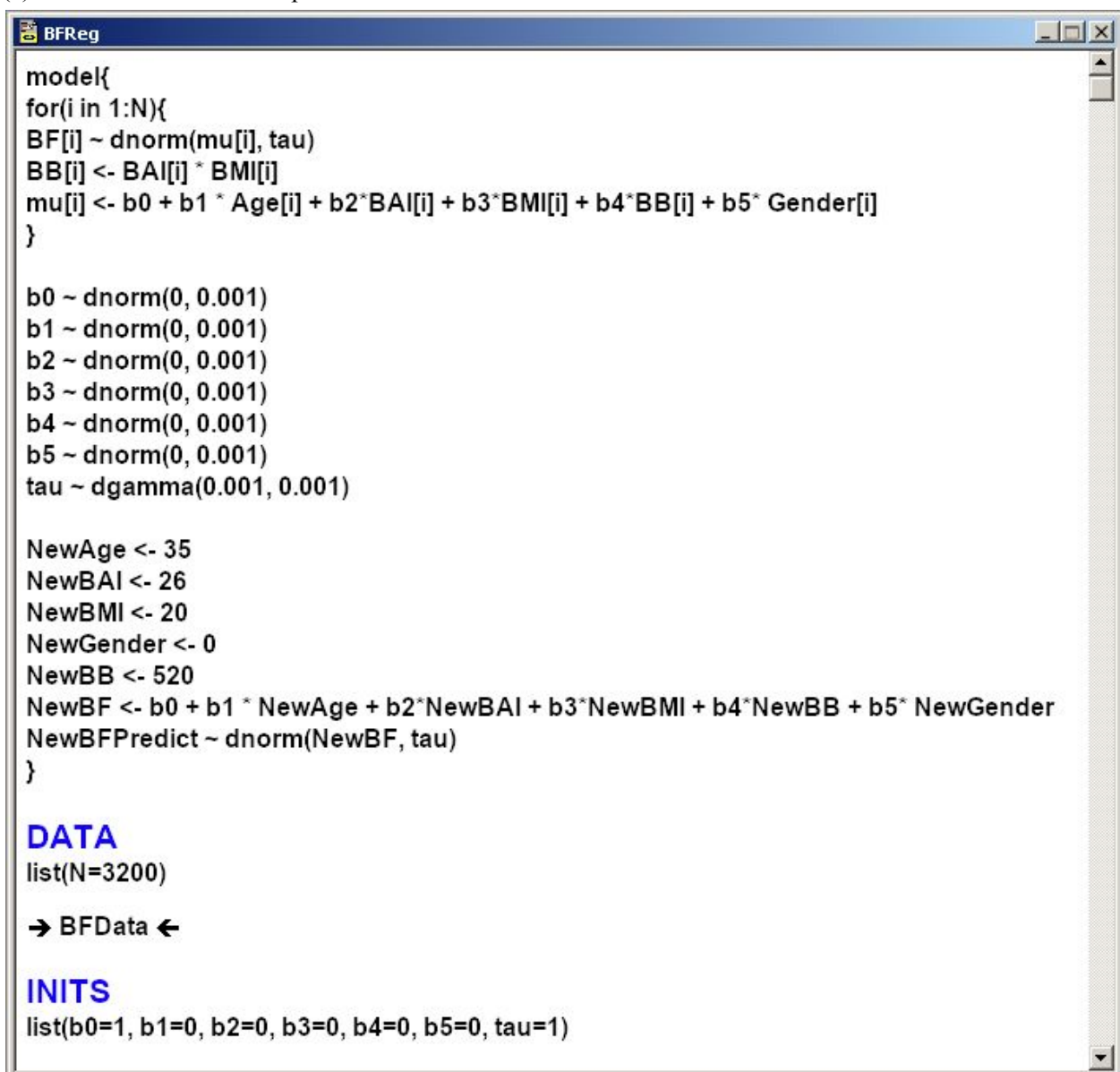
The Bayesian estimator of σ/τ is: 1.018 which is not very close but still close to 0.8.

For the cross-validation, I got MSE 1.983.

2.

Part I: Model with all the predictors:

(a) I have used the below OpenBUGS code:



```
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)

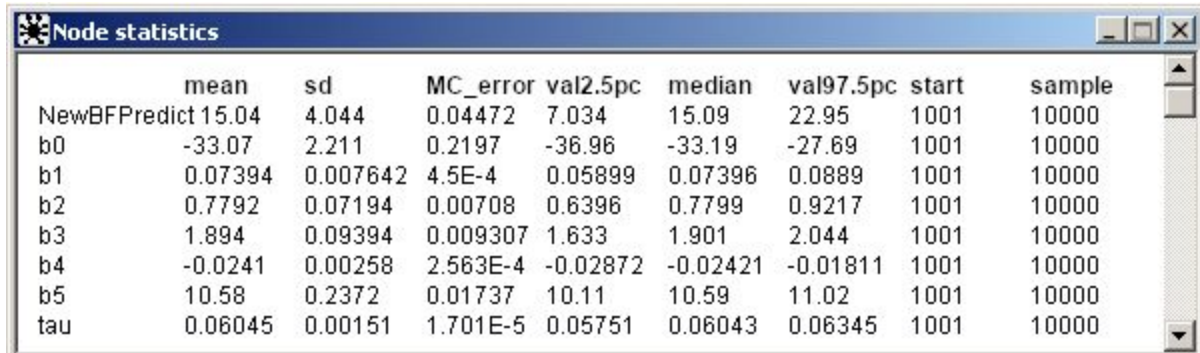
  NewAge <- 35
  NewBAI <- 26
  NewBMI <- 20
  NewGender <- 0
  NewBB <- 520
  NewBF <- b0 + b1 * NewAge + b2*NewBAI + b3*NewBMI + b4*NewBB + b5* NewGender
  NewBFPredict ~ dnorm(NewBF, tau)
}

DATA
list(N=3200)

→ BFData ←

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

First, I burned 1000 samples. And then ran 10000 updates.



	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
NewBFPredict	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

Therefore, if we include all the predictors, my model would be below:

$$BF = -33.07 + 0.07394 * \text{Age} + 0.7792 * \text{BAI} + 1.894 * \text{BMI} - 0.0241 * \text{BB} + 10.58 * \text{Gender}$$

(b) Evaluation of a new person with Age = 35, BAI = 26, BMI = 20, Gender = 0, BB = 520 is calculated using the same OpenBUGS code in part (a). In the Node statistics, the field NewBFPredict represents the statistics of the actual BF prediction for this new person. Therefore, the Bayesian estimate of BF for this person is 15.04.

Part II: Model with the single best predictor:

There are multiple ways to do this. One way is to compare R^2 for each of the model with only one predictor at a time and compare. The model with the highest R^2 value can be called the single best predictor. In the below cases, I'll set all b1, b2, b3, b4 and b5 as 0, except one of them and will run a simple linear regression for each of these cases and will look at R^2 .

Case I:

Setting $b2 = b3 = b4 = b5 = 0$:

```

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 + b1 * Age[i] + 0*BAI[i] + 0*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)

# NewAge <- 35
# NewBAI <- 26
# NewBMI <- 20
# NewGender <- 0
# NewBB <- 520
# NewBF <- b0 + b1 * NewAge + b2*NewBAI + b3*NewBMI + b4*NewBB + b5*NewGender
# NewBFPredict ~ dnorm(NewBF, tau)

nminusp <- N -2
sigma2 <- 1/tau
sse <- nminusp*sigma2
for (i in 1:N) {
  cBF[i] <- BF[i] - mean(BF[])
}
sst <- inprod(cBF[], cBF[])
BR2 <- 1 - sse/sst
}

DATA
list(N=3200)

→ BFData ←

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

```

Node statistics								
	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
BR2	0.08359	0.02297	2.26E-4	0.03752	0.08384	0.1275	1001	10000
b0	19.34	0.5193	0.02582	18.34	19.34	20.35	1001	10000
b1	0.2197	0.01276	6.345E-4	0.1952	0.2197	0.2443	1001	10000

We got R^2 value of 0.08359 which is quite low.

Case II:

Setting $b1 = b3 = b4 = b5 = 0$:

```

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] + b2*BAI[i] + 0*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)

  # NewAge <- 35
  # NewBAI <- 26
  # NewBMI <- 20
  # NewGender <- 0
  # NewBB <- 520
  # NewBF <- b0 + b1 * NewAge + b2*NewBAI + b3*NewBMI + b4*NewBB + b5*NewGender
  # NewBFPredict ~ dnorm(NewBF, tau)

  nminusp <- N -2
  sigma2 <- 1/tau
  sse <- nminusp*sigma2
  for (i in 1:N) {
    cBF[i] <- BF[i] - mean(BF[])
  }
  sst <- inprod(cBF[], cBF[])
  BR2 <- 1 - sse/sst
}

DATA
list(N=3200)
→ BFData ←

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

```

Node statistics								
	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
BR2	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

We got R^2 value of 0.5467 which is much better than Case I.

Case III:

Setting $b1 = b2 = b4 = b5 = 0$:


```

BFRreg
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)

  # NewAge <- 35
  # NewBAI <- 26
  # NewBMI <- 20
  # NewGender <- 0
  # NewBB <- 520
  # NewBF <- b0 + b1 * NewAge + b2*NewBAI + b3*NewBMI + b4*NewBB + b5*NewGender
  # NewBFpredict ~ dnorm(NewBF, tau)

  nminusp <- N -2
  sigma2 <- 1/tau
  sse <- nminusp*sigma2
  for (i in 1:N) {
    cBF[i] <- BF[i] - mean(BF[])
  }
  sst <- inprod(cBF[], cBF[])
  BR2 <- 1 - sse/sst
}

DATA
list(N=3200)

→ BFData ←

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

```

Node statistics								
	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
BR2	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
tau	0.0213	5.338E-4	5.242E-6	0.02027	0.02129	0.02236	1001	10000

We got R^2 value of 0.2955 which is worse than case II but better than case I.

Case IV:

Setting $b1 = b2 = b3 = b5 = 0$:

```

BFRag

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] + 0*BMI[i] + b4*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)

  # NewAge <- 35
  # NewBAI <- 26
  # NewBMI <- 20
  # NewGender <- 0
  # NewBB <- 520
  # NewBF <- b0 + b1 * NewAge + b2*NewBAI + b3*NewBMI + b4*NewBB + b5*NewGender
  # NewBFPredict ~ dnorm(NewBF, tau)

  nminusp <- N -2
  sigma2 <- 1/tau
  sse <- nminusp*sigma2
  for (i in 1:N) {
    cBF[i] <- BF[i] - mean(BF[])
  }
  sst <- inprod(cBF[], cBF[])
  BR2 <- 1 - sse/sst
}

DATA
list(N=3200)

→ BFData ←

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

```

Node statistics								
	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
BR2	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
tau	0.02861	7.171E-4	7.092E-6	0.02722	0.0286	0.03003	1001	10000

We got R^2 value of 0.4755 which is worse than case II but better than Case I and Case III.

Case V:

Setting $b1 = b2 = b3 = b4 = 0$:


```

BFReg
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] + 0*BMI[i] + 0*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)

  # NewAge <- 35
  # NewBAI <- 26
  # NewBMI <- 20
  # NewGender <- 0
  # NewBB <- 520
  # NewBF <- b0 + b1 * NewAge + b2*NewBAI + b3*NewBMI + b4*NewBB + b5*NewGender
  # NewBFPredict ~ dnorm(NewBF, tau)

  nminusp <- N -2
  sigma2 <- 1/tau
  sse <- nminusp*sigma2
  for (i in 1:N) {
    cBF[i] <- BF[i] - mean(BF[])
  }
  sst <- inprod(cBF[], cBF[])
  BR2 <- 1 - sse/sst
}

DATA
list(N=3200)
→ BFData ←

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

```

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
BR2	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
tau	0.01951	4.891E-4	4.862E-6	0.01857	0.0195	0.02048	1001	10000

We got R^2 value of 0.2309 which is worse than Case IV, Case III, and Case II but better than Case I.

Therefore, doing the above cases comparison, we now know that Case II is the best single predictor as it has the largest value of R^2 . The entire comparison of single best predictor models is done on

the basis of R^2 value comparison along. There might be above methods too but I couldn't get time to research those.

The Case II model is given by:

$$BF = -5.912 + 1.181 * BAI.$$

(b) BF evaluation of a new person with Age = 35, BAI = 26, BMI = 20, Gender = 0, BB = 520 is calculated using the below code.

```

BFReg
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] + b2*BAI[i] + 0*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)

  # NewAge <- 35
  NewBAI <- 26
  # NewBMI <- 20
  # NewGender <- 0
  # NewBB <- 520
  NewBF <- b0 + b2*NewBAI
  NewBFPredict ~ dnorm(NewBF, tau)

  nminusp <- N -2
  sigma2 <- 1/tau
  sse <- nminusp*sigma2
  for (i in 1:N) {
    cBF[i] <- BF[i] - mean(BF[])
  }
  sst <- inprod(cBF[], cBF[])
  BR2 <- 1 - sse/sst
}

DATA
list(N=3200)
→ BFData ←

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

```

Because, I ran the above code again, some of the statistics given below are minutely different from the one in case II in part (a) which is:

The b0 value changed from -5.912 to -5.956.

The value of b2 changed from 1.181 to 1.183.

Node statistics								
	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
BR2	0.5469	0.01132	1.168E-4	0.5241	0.547	0.5685	1001	10000
NewBFPredict	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
tau	0.03311	8.261E-4	8.485E-6	0.03151	0.03311	0.03475	1001	10000

From the above node statistic, the field NewBFPredict represents the statistics of the actual BF prediction for this new person using the single predictor model. Therefore, the Bayesian regression estimate using single predictor model of BF for this person is 24.84.

3.

The proportion of response after a shock of 2.5 milliamps can be calculated using the below OpenBUGS code:

```

model{
  for( i in 1 : N ) {
    y[i] ~ dbin(p[i],n[i])
    logit(p[i]) <- alpha.star + beta * (x[i] - mean(x[]))

    yhat[i] <- n[i] * p[i]
  }
  alpha <- alpha.star - beta * mean(x[])
  beta ~ dnorm(0.0,0.001)
  alpha.star ~ dnorm(0.0,0.001)

  temp.star <- 2.5
  lstar <- alpha + beta * temp.star
  pstar <- exp(lstar)/(1+exp(lstar))
}

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

INITS
list(alpha.star=0, beta=0)

```

With 100,000 simulations, I got the following statistics:

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.star	-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
pstar	0.4531	0.03407	1.336E-4	0.3865	0.4531	0.52	100001	100000

pstar here represents the estimated proportion of responses after a shock of 2.5 milliamps.

Therefore, from the above results, the Bayesian estimate of proportion of responses after a shock of 2.5 milliamps is 0.4531.

Also, from above, the 95% credible set for the population proportion of responses is [0.3865, 0.52].