CHL5223 - Applied Bayesian Methods - Assignment 3

Luis Correia - Student No. 1006508566

March 20th 2020

Question 1

In the file "SmokeHyperHwkQuesR.txt", there is a preliminary analysis of a data-set. The analysis looks at the relationship between smoking and hypertension. For the population under study, there are a series of binary variables that are recorded. They record the following traits: smoker, obesity, snoring, gender as well as the outcome of interest which is hypertension. A table is created where the number of people with hypertension is recorded as well as the total number in that category.

(Aside: there were was no one who was a a smoker and obese who did not snore. So that group is not included in the table.) Also, note in that file, there is a basic analysis of this data. Starting with the analysis in this file, run the MCMC with more iterations and see if the chain has "converged". For the purpose of this exercise, just look at three parameters. Use either "sd.b" or "tau.b" for one of the parameters and use one of the "beta" parameters for the other two. Fit this model with either WinBugs, OpenBugs, or Jags (and provide the model in your answer.) Run the iteration with at least three chains in this model. Do the following:

item (a)

Run between $10,000$ to $30,000$ iterations of the MCMC. For the three parameters, provide a copy of the trace plot for a portion of the iteration, the auto-correlation plot, and the statistics.

{*Solution.*}

For this item, we will run the model with *no-burnin* and *no-thin*, i.e.:

- Run-0: `n.iter`$= 30,000$ with `n.burnin`$= 0$ and `n.thin`$= 1$ (every sample will be stored);

Table 1: Data - Relationship between Hypertension and Smoking

smoke	obese	snore	male	hypoten	n
0	0	0	1	5	60
0	0	0	0	10	149
1	0	0	1	2	17
1	0	0	0	6	16
0	1	0	1	1	12
0	1	0	0	2	9
0	0	1	1	36	187
0	0	1	0	28	138
1	0	1	1	13	85
1	0	1	0	4	39
0	1	1	1	15	51
0	1	1	0	11	28
1	1	1	1	8	23
1	1	1	0	4	12

Table 2: OpenBugs Summary - Hypertension and Smoke Study (Run-0)

Parameter	mean	sd	2.5%	50%	97.5%
b0	-2.5243	0.3348	-3.1740	-2.5260	-1.8690
b.smok	1.3913	0.5568	0.2763	1.4010	2.4620
b.ob	0.8539	0.2932	0.2834	0.8521	1.4380
b.sn	1.1490	0.3678	0.4219	1.1510	1.8700
b.male	-0.1630	0.2719	-0.7307	-0.1533	0.3432
b.smsn	-1.6682	0.6538	-2.9200	-1.6720	-0.3777
sd.b	0.2239	0.2118	0.0081	0.1707	0.7273
deviance	61.2941	3.9453	55.2600	60.7000	70.5600

For this question we will generate *no-thinned* trace and ACF plots of parameter estimates for σ, β_0, β_{Smoke} and $\beta_{Obesity}$ to assess its *convergence* and possible existence of auto-correlation in sampled data contains.

For the sake of simplicity, ACF plots will contain only chain #1 in order to avoid deteriorating the PDF's performance with uninformative data, since the results are quite similar for all chains.

Please also note that it was used the `ggAcf()` function from package `forecast` due to its flexibility and configuration capabilities. The main difference of it from the standard `acf` is that `ggAcf()` starts plotting from $lag = 1$, and `acf()` starts at $lag = 0$.

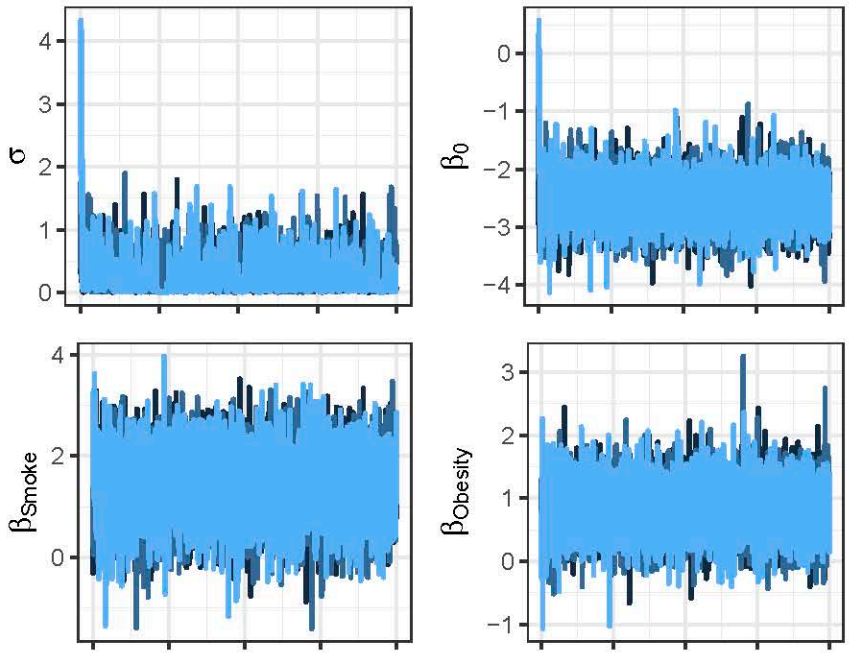


Figure 1: Trace-plots of σ, β_0, β_{Smoke} and $\beta_{Obesity}$ (Run-0)

Comment:- From Trace Plots we can verify that parameters estimates for σ, β_0, β_{Smoke} and $\beta_{Obesity}$ generated by our simulations *converged* after a few iterations. We can also observe a peak in the early simulations due to stabilization of Markov Chain as we are not using *burnin* parameters for **Run-0**.

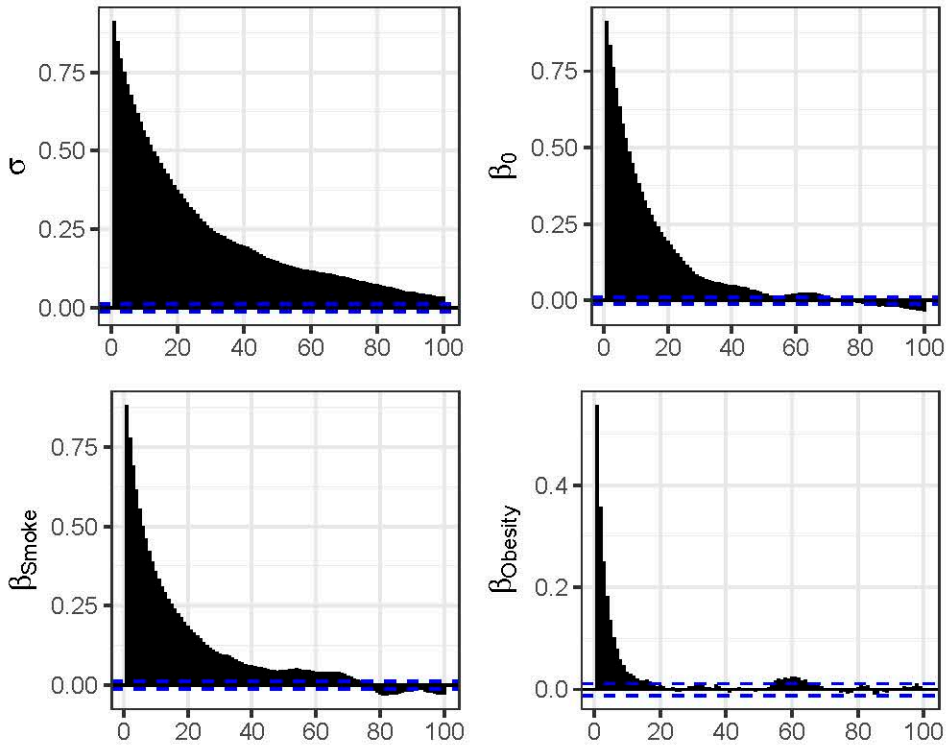


Figure 2: ACF-plots of σ, β_0, β_{Smoke} and $\beta_{Obesity}$ (max-Lag=100) (Run-0)

Comment:- From ACF Plots we can also verify the parameters estimates generated by our simulations are *highly correlated* for all parameters in the early stages of the simulation, approximately up to `lag`=100 (e.g., parameter σ). This is due to MCMC simulations carry some auto-correlation between samples by construction of Gibbs Sampler and also because we are not using the `n.thin` parameter to discard consecutive samples.

item (b)

Input the MCMC values into R (but not yet into coda). Plot the trace plot in R. Remove some of the early values of the chain (throwing away a part that is "burned-in") and then plot the estimate of the densities for the parameters with a different density estimate for each of the three chains. What effect does "burning in" the chain have?

{*Solution.*}

In this item we will use the results of **Run-01**, i.e., simulations generated with `n.burnin`= 10,000 and `n.thin`= 50 (every 50-th value sampled will be maintained).

Table 3: OpenBugs Summary - Hypertension and Smoke Study (Run-1)

Parameter	mean	sd	2.5%	50%	97.5%
b0	-2.5240	0.3238	-3.1650	-2.5190	-1.8970
b.smok	1.3827	0.5556	0.2569	1.3930	2.4420
b.ob	0.8574	0.2930	0.2892	0.8555	1.4390
b.sn	1.1484	0.3625	0.4376	1.1470	1.8680
b.male	-0.1664	0.2678	-0.7196	-0.1586	0.3356
b.smsn	-1.6531	0.6411	-2.8900	-1.6615	-0.3710
sd.b	0.2237	0.1949	0.0080	0.1750	0.7161
deviance	61.2446	3.9073	55.1800	60.6800	70.4700

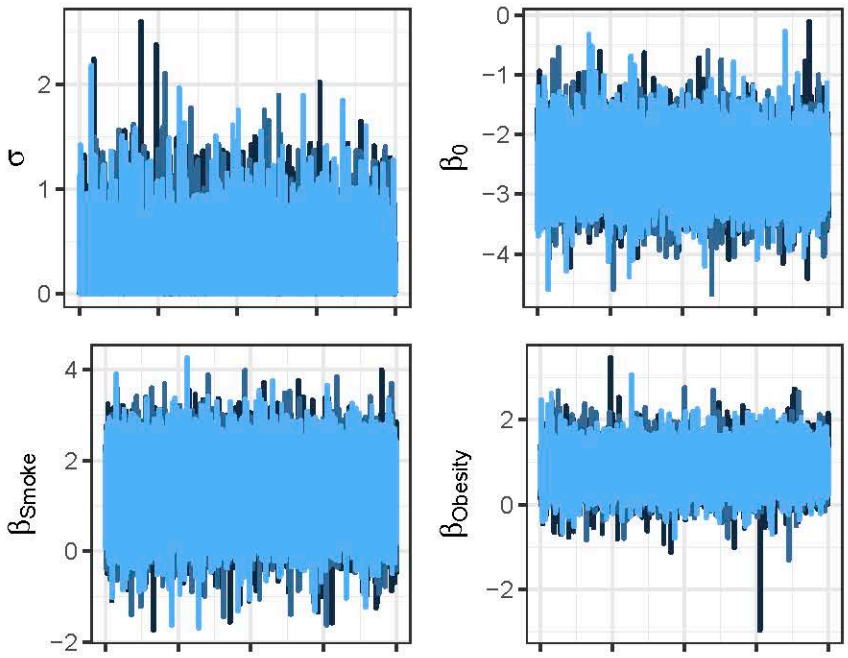


Figure 3: Trace-plots of σ, β_0, β_{Smoke} and $\beta_{Obesity}$

Comment:- From Trace Plots for **Run-1** we can verify that parameters estimates for σ, β_0, β_{Smoke} and $\beta_{Obesity}$ generated by our simulations have chains well mixed and all *converged*.

We can notice the effect of *burnin* when we compare the above trace-plots with the ones obtained on Run-0: the stabilization of chains were in the very early iterations in Run-1 while in Run-0 we had to run a few iterations before the samples stabilizes.

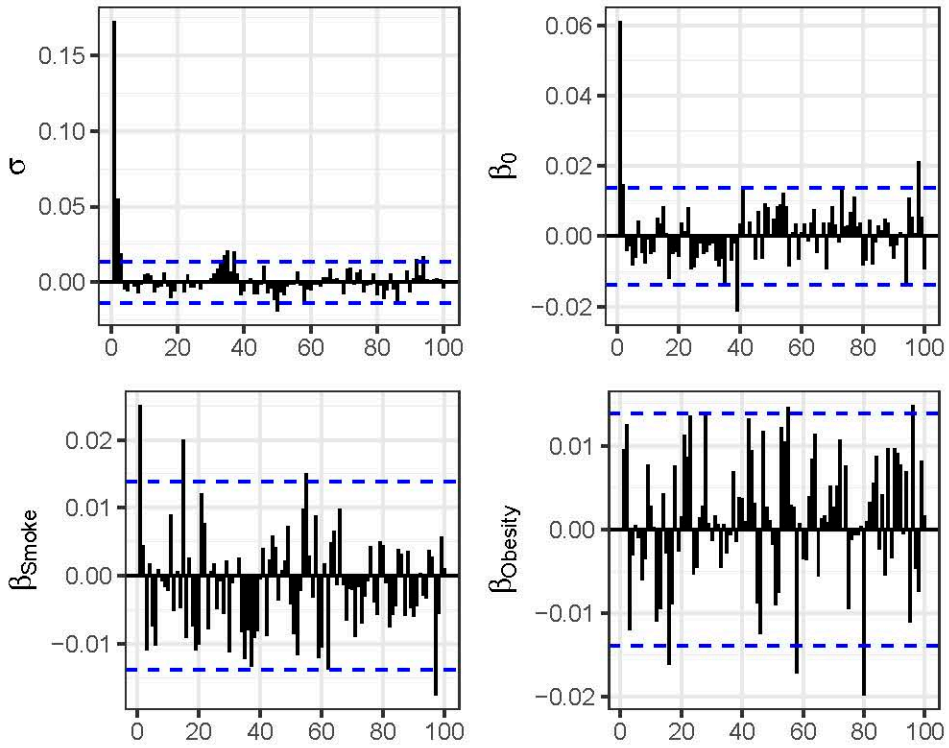


Figure 4: ACF-plots of σ, β_0, β_{Smoke} and $\beta_{Obesity}$ (max-Lag=100)

Comment:- Differently from **Run-0**, ACF Plots from **Run-1** have small auto-correlation between estimates concentrated in initial lags and all apparently become uncorrelated after `lag=5`. Thinning the MCMC simulations contributed to generate less auto-correlated samples.

item (c)

If you thinned the chain, what would be the advantages? Is it necessary to thin a chain?

{*Solution.*}

In practice, when we generate samples through MCMC (Markov Chain Monte Carlo) algorithms, we need to make all efforts possible to have the convergence achieved. In this sense, when we stop a Markov chain at a finite number of iterations, we don't obtain independent realizations as each estimate is generated from the previous ones. On the other hand, as the chain estimates advances, such dependence will decrease and, in the limit, we will have independent realizations from the posterior distribution.

Thinning a Markov Chain consists in picking separated points from the sample, at each *k-th* step. The major advantage of this process is to separate the points sampled by the Markov Chain to reduce auto-correlation between observations. By doing this we will obtain some level of independent samples at the end-game.

In our case, it was used `n.thin=50` for *Run-1* and we have obtained quite good results in terms of convergence and *less*-auto-correlated samples.

Another technical advantage of specifying a thinning parameter is because when the number of parameters is large, thinning helps obtain the saved output of our model to a reasonably-sized R object.

item (d)

Provide the estimate of the posterior mean of the three parameters for each chain and also give the Monte Carlo accuracy of your estimate. For the Monte Carlo accuracy, compute by batch means and by using the auto-correlation function.

{*Solution.*}

We calculated posterior means for the parameters using *Batch-Means* procedure, including *Batch-Standard Error* and *ACF Standard Error*. We also plotted the posterior densities of each parameter.

The results are shown below.

Parameter σ

```
##
## ---- Summary Statistics for SIGMA (Chain  1 ) -----
## B-Mean :  0.2235
## B-S.E.:  0.001573
## 95% Credible Region ( 0.2204 , 0.2267 )
##
## (ACF) S.E.:  0.001703
## (ACF) 95% Credible Region ( 0.2201 , 0.2270 )
## -------------------------------------------------

##
## ---- Summary Statistics for SIGMA (Chain  2 ) -----
## B-Mean :  0.2229
## B-S.E.:  0.001782
## 95% Credible Region ( 0.2193 , 0.2265 )
##
## (ACF) S.E.:  0.001517
## (ACF) 95% Credible Region ( 0.2199 , 0.2259 )
## -------------------------------------------------

##
## ---- Summary Statistics for SIGMA (Chain  3 ) -----
## B-Mean :  0.2248
## B-S.E.:  0.001704
## 95% Credible Region ( 0.2214 , 0.2282 )
##
## (ACF) S.E.:  0.001892
## (ACF) 95% Credible Region ( 0.2210 , 0.2286 )
## -------------------------------------------------
```

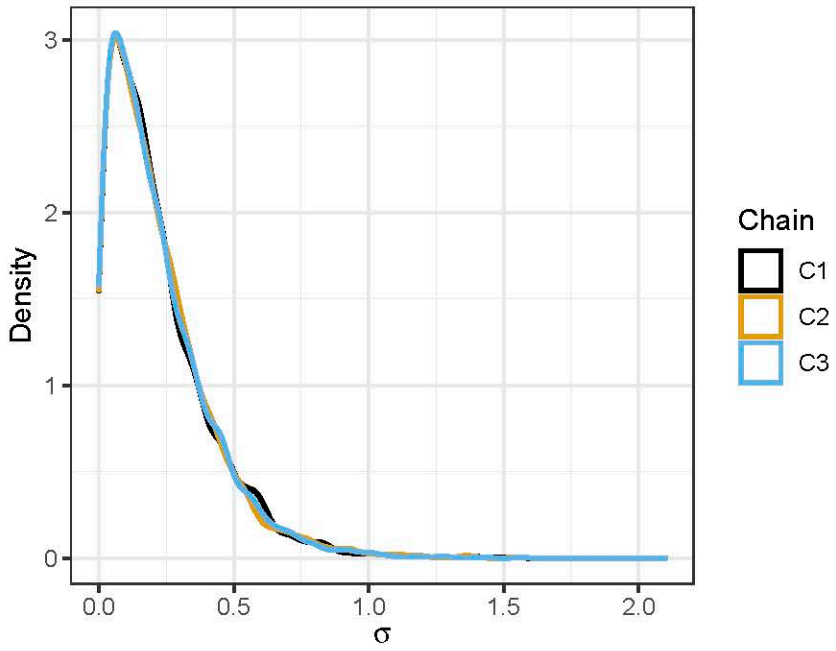


Figure 5: Posterior Distributions of σ

Parameter β_0

```
##
## ---- Summary Statistics for BETA0 (Chain  1 ) -----
## B-Mean :  -2.5220
## B-S.E.:  0.002489
## 95% Credible Region ( -2.5270 , -2.5171 )
##
## (ACF) S.E.:  0.002415
## (ACF) 95% Credible Region ( -2.5269 , -2.5172 )
## -------------------------------------------------

##
## ---- Summary Statistics for BETA0 (Chain  2 ) -----
## B-Mean :  -2.5239
## B-S.E.:  0.002205
## 95% Credible Region ( -2.5283 , -2.5195 )
##
## (ACF) S.E.:  0.002139
## (ACF) 95% Credible Region ( -2.5282 , -2.5196 )
## -------------------------------------------------

##
## ---- Summary Statistics for BETA0 (Chain  3 ) -----
## B-Mean :  -2.5261
## B-S.E.:  0.002269
## 95% Credible Region ( -2.5306 , -2.5216 )
##
## (ACF) S.E.:  0.002337
```

```
## (ACF) 95% Credible Region ( -2.5308 , -2.5214 )
## -----------------------------------------------
```

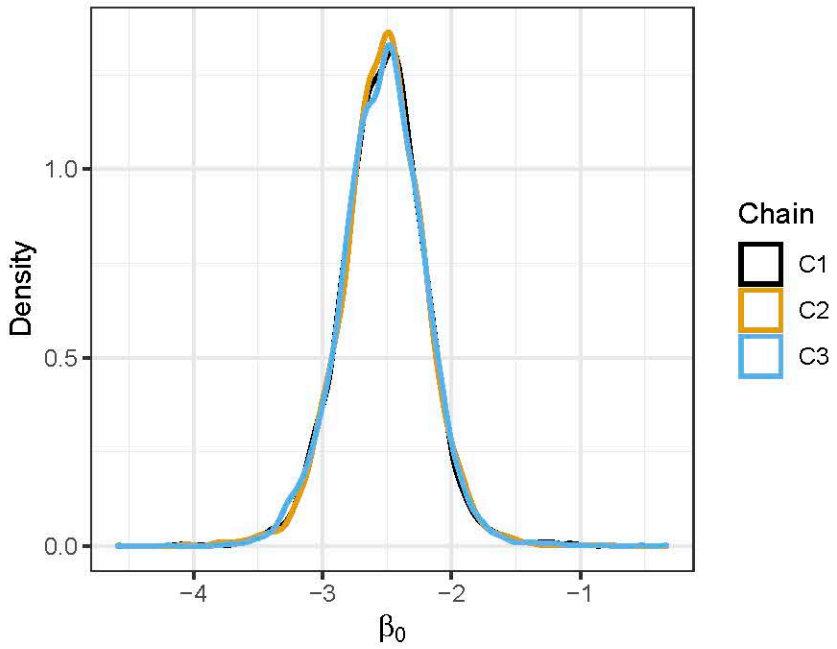


Figure 6: Posterior Distributions of β_0

Parameter β_{Smoke}

```
##
## ---- Summary Statistics for BETA-SMOKE (Chain  1 ) -----
## B-Mean :  1.3778
## B-S.E.:  0.002836
## 95% Credible Region ( 1.3722 , 1.3835 )
##
## (ACF) S.E.:  0.003366
## (ACF) 95% Credible Region ( 1.3711 , 1.3846 )
## -----------------------------------------------

##
## ---- Summary Statistics for BETA-SMOKE (Chain  2 ) -----
## B-Mean :  1.3842
## B-S.E.:  0.003524
## 95% Credible Region ( 1.3771 , 1.3912 )
##
## (ACF) S.E.:  0.003750
## (ACF) 95% Credible Region ( 1.3767 , 1.3917 )
## -----------------------------------------------

##
## ---- Summary Statistics for BETA-SMOKE (Chain  3 ) -----
## B-Mean :  1.3861
## B-S.E.:  0.004204
## 95% Credible Region ( 1.3776 , 1.3945 )
```

```
##
## (ACF) S.E.:  0.004041
## (ACF) 95% Credible Region ( 1.3780 , 1.3941 )
## ------------------------------------------------
```

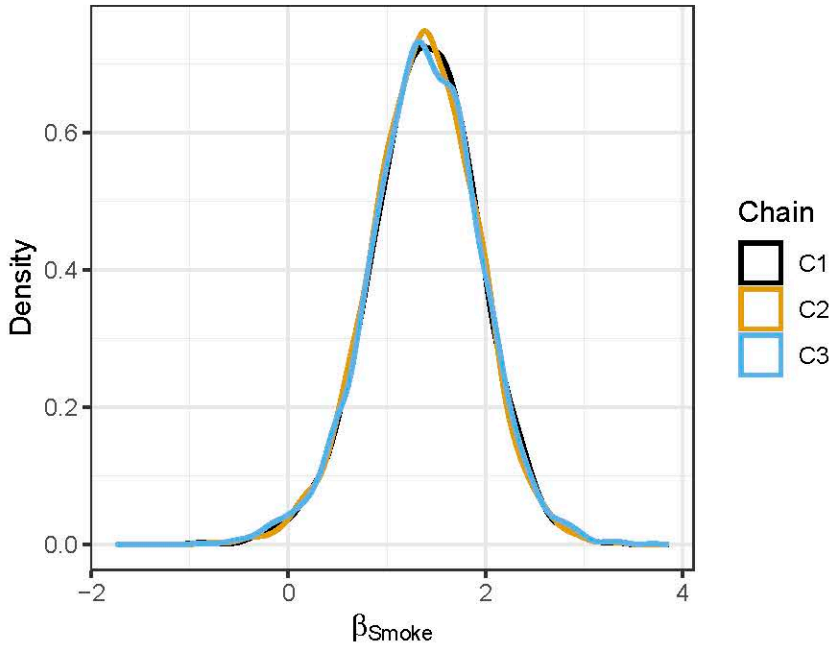


Figure 7: Posterior Distributions of β_{Smoke}

Parameter $\beta_{Obesity}$

```
##
## ---- Summary Statistics for BETA-OBESITY (Chain  1 ) -----
## B-Mean :  0.8555
## B-S.E.:  0.002206
## 95% Credible Region ( 0.8511 , 0.8599 )
##
## (ACF) S.E.:  0.002337
## (ACF) 95% Credible Region ( 0.8509 , 0.8602 )
## ------------------------------------------------

##
## ---- Summary Statistics for BETA-OBESITY (Chain  2 ) -----
## B-Mean :  0.8600
## B-S.E.:  0.002243
## 95% Credible Region ( 0.8555 , 0.8645 )
##
## (ACF) S.E.:  0.002213
## (ACF) 95% Credible Region ( 0.8556 , 0.8644 )
## ------------------------------------------------

##
## ---- Summary Statistics for BETA-OBESITY (Chain  3 ) -----
## B-Mean :  0.8568
```

```
## B-S.E.:  0.002053
## 95% Credible Region ( 0.8527 , 0.8609 )
##
## (ACF) S.E.:  0.002035
## (ACF) 95% Credible Region ( 0.8527 , 0.8609 )
## ------------------------------------------------
```

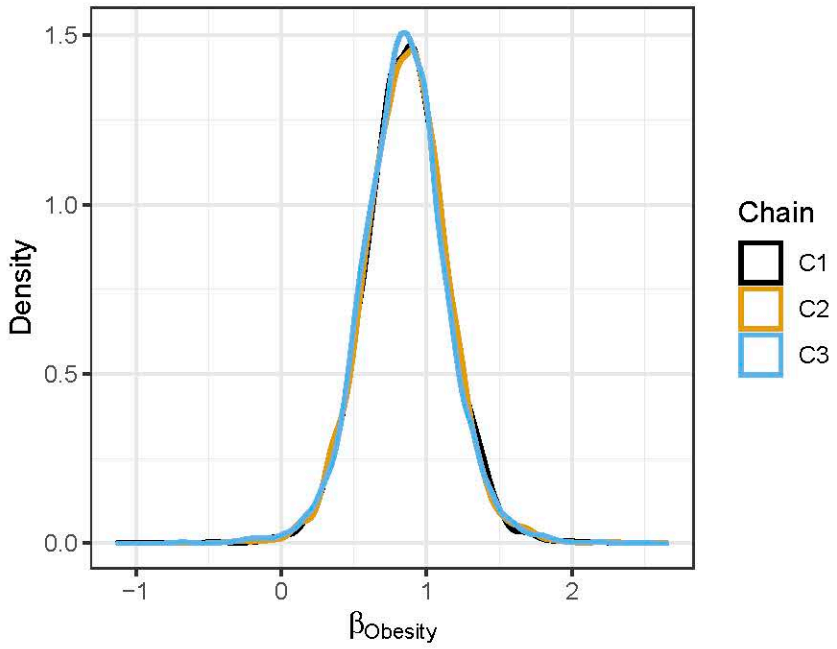


Figure 8: Posterior Distributions of $\beta_{Obesity}$

Comment:- Batch-means statistics are close in all chains of its own parameter. We also can see the Batch and ACF-Standard Errors and are small and close to each other, also reinforcing our belief of convergence, which can also be seen through the 95% Credible Regions obtained and the respective posterior densities, as well.

item (e)

Using the coda (or boa) package, use the *Geweke* and *Brooks-Gelman-Rubin* diagnostic procedures to assess how well the MCMC algorithm has converged.

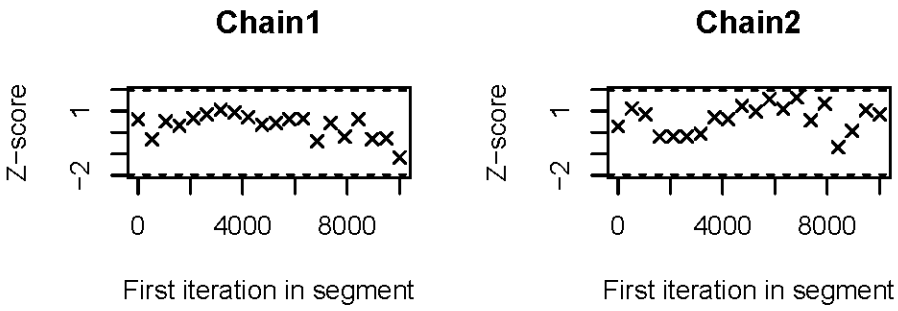
{*Solution.*}

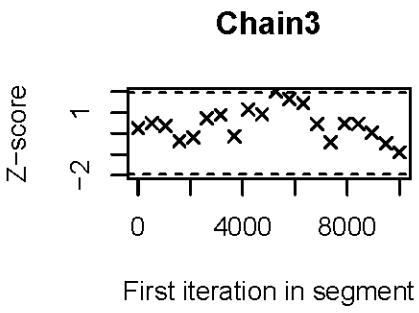
Using the `coda` package we obtained the Geweke and Brooks-Gelman-Rubin diagnostics for our estimates. We also plotted the useful `geweke.plot()` and `gelman.plot()` to verify convergence and stabilization of simulated chains for our parameters. The results are shown below:

Parameter σ

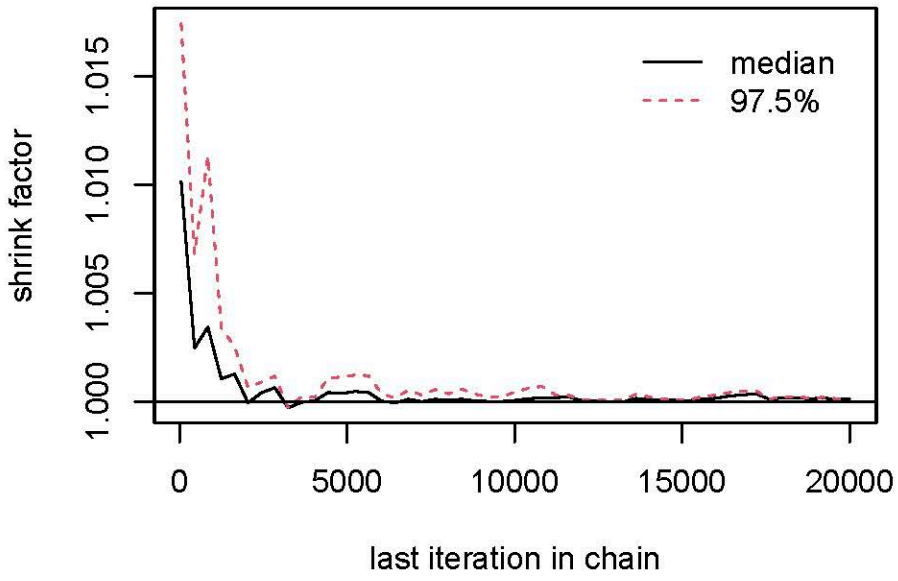
```
##
## --------- CODA Summary Statistics for SIGMA ---------
##
## Iterations = 1:20000
```

```
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 20000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean     SD Naive SE Time-series SE
## Chain1 0.2235 0.1966 0.001390       0.001699
## Chain2 0.2229 0.1925 0.001361       0.001666
## Chain3 0.2248 0.1957 0.001384       0.001712
##
## 2. Quantiles for each variable:
##
##            2.5%     25%    50%    75%  97.5%
## Chain1 0.007091 0.07932 0.1740 0.3124 0.7156
## Chain2 0.008322 0.08187 0.1754 0.3089 0.7102
## Chain3 0.008477 0.08066 0.1755 0.3134 0.7200
##
## >>> Effective Size:
##   Chain1   Chain2   Chain3
## 13389.41 13352.94 13070.65
##
## >>> GEWEKE Diagnostics:
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
## Chain1 Chain2 Chain3
## 0.6180 0.2853 0.2508
```





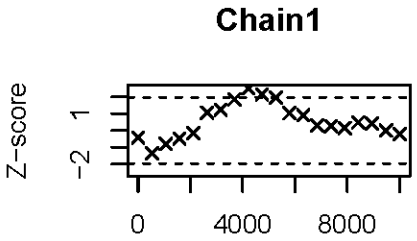
```
## >>> GELMAN Diagnostics:
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
```

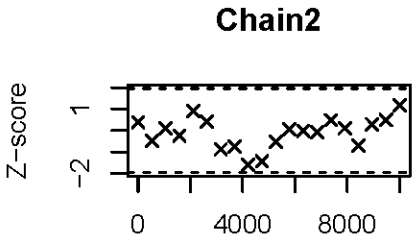


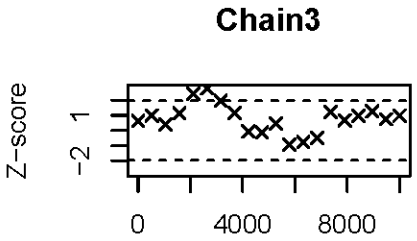
Parameter β_0

```
##
## --------- CODA Summary Statistics for BETA-0 ---------
##
## Iterations = 1:20000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 20000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean     SD Naive SE Time-series SE
## Chain1 -2.522 0.3249 0.002297       0.002470
## Chain2 -2.524 0.3213 0.002272       0.002332
## Chain3 -2.526 0.3252 0.002300       0.002367
##
## 2. Quantiles for each variable:
##
##          2.5%    25%    50%    75%  97.5%
## Chain1 -3.162 -2.724 -2.518 -2.319 -1.889
## Chain2 -3.158 -2.725 -2.518 -2.319 -1.903
```

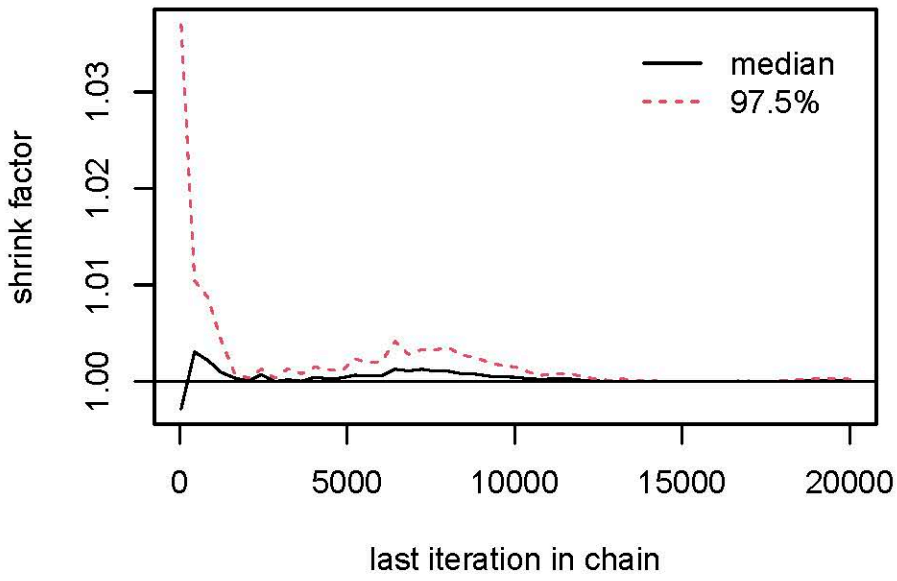
```
## Chain3 -3.173 -2.730 -2.522 -2.320 -1.899
##
## >>> Effective Size:
##   Chain1   Chain2   Chain3
## 17302.44 18979.19 18875.77
##
## >>> GEWEKE Diagnostics:
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##  Chain1  Chain2  Chain3
## -0.4126  0.3897  0.6365
```







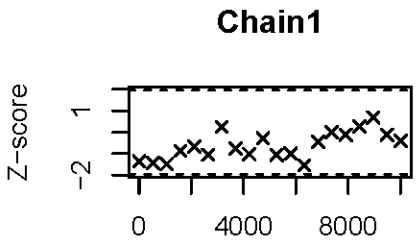

```
## >>> GELMAN Diagnostics:
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
```

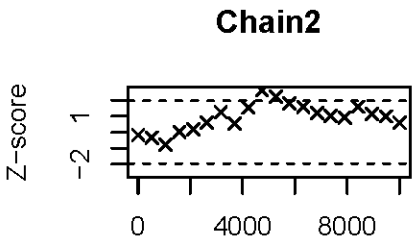


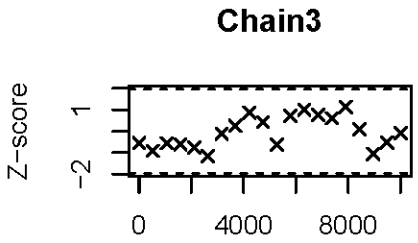
Parameter β_{Smoke}

```
##
## --------- CODA Summary Statistics for BETA-SMOKE ---------
##
## Iterations = 1:20000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 20000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##         Mean     SD Naive SE Time-series SE
## Chain1 1.378 0.5556 0.003928       0.004029
## Chain2 1.384 0.5528 0.003909       0.004004
## Chain3 1.386 0.5585 0.003949       0.004168
##
## 2. Quantiles for each variable:
##
##          2.5%   25%   50%   75% 97.5%
## Chain1 0.2531 1.023 1.391 1.747 2.436
## Chain2 0.2712 1.024 1.395 1.754 2.446
## Chain3 0.2492 1.026 1.393 1.755 2.448
##
## >>> Effective Size:
##   Chain1   Chain2   Chain3
## 19018.77 19061.80 17954.61
##
```

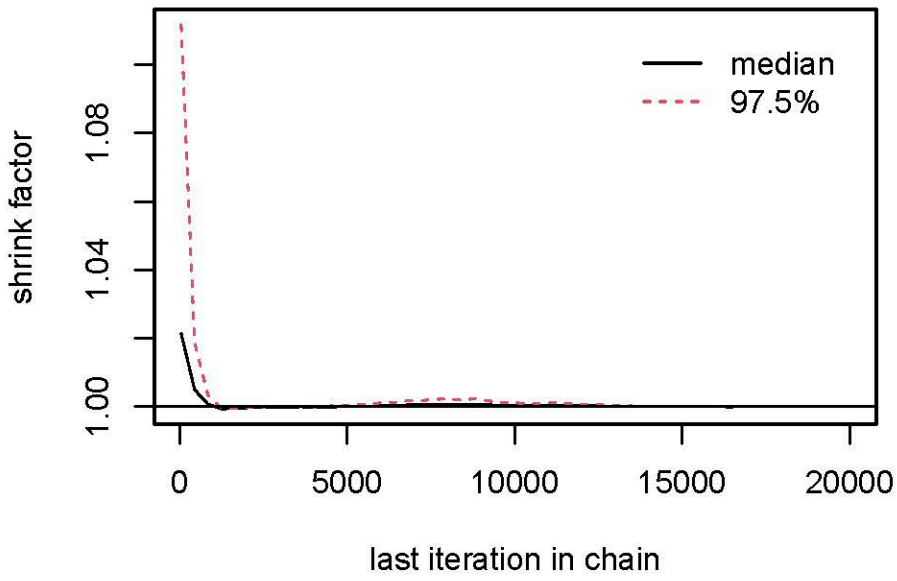
```
## >>> GEWEKE Diagnostics:
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##  Chain1  Chain2  Chain3
## -1.3536 -0.1762 -0.5546
```







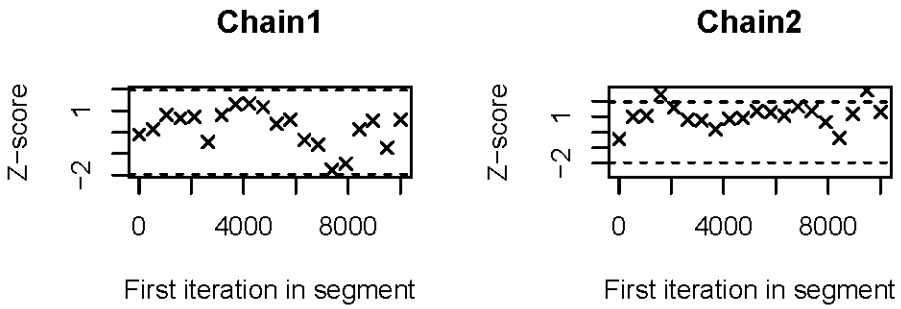

```
## >>> GELMAN Diagnostics:
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
```

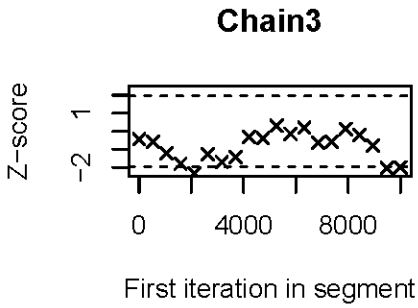


Parameter $\beta_{Obesity}$

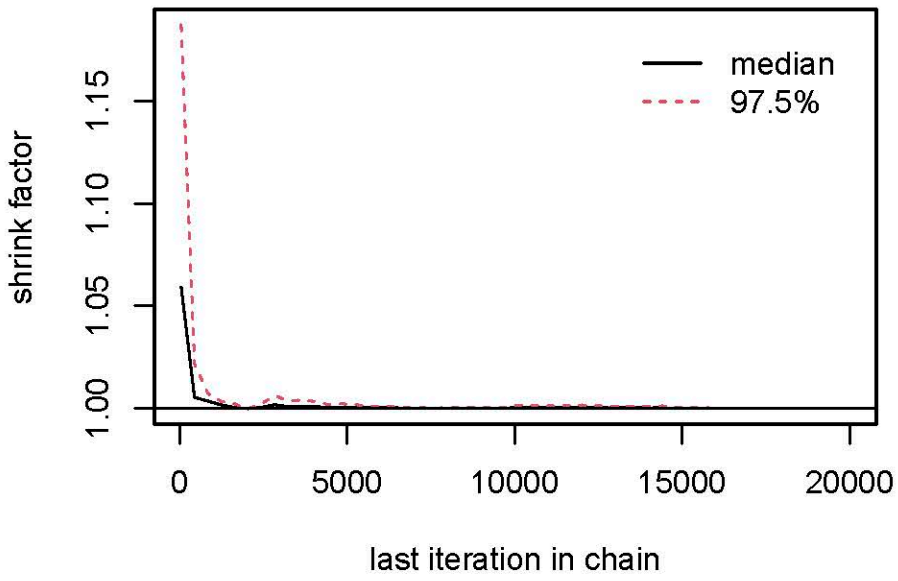
```
##
## --------- CODA Summary Statistics for BETA-OBESITY ---------
##
## Iterations = 1:20000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 20000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean     SD Naive SE Time-series SE
## Chain1 0.8555 0.2945 0.002083       0.002103
## Chain2 0.8600 0.2918 0.002064       0.002064
## Chain3 0.8568 0.2926 0.002069       0.002069
##
## 2. Quantiles for each variable:
##
##          2.5%    25%    50%   75% 97.5%
## Chain1 0.2915 0.6752 0.8534 1.036 1.438
## Chain2 0.2938 0.6772 0.8584 1.039 1.436
## Chain3 0.2832 0.6728 0.8555 1.035 1.443
##
## >>> Effective Size:
##   Chain1   Chain2   Chain3
## 19609.36 20000.00 20000.00
##
```

```
## >>> GEWEKE Diagnostics:
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##  Chain1  Chain2  Chain3
## -0.1084 -0.4499 -0.4444
```






```
## >>> GELMAN Diagnostics:
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
```



item (f)

Using the information from this question, state if you feel that the MCMC algorithm has converged. Justify your answer.

{*Solution.*}

Geweke diagnostics applies a simple Z-test to check whether the means estimated from two different subsamples of the total MCMC output are equal, in particular, the default parameters are the initial 10% and the last 50% of the total iterations. It can be demonstrated[1] that the statistic Z provided by the diagnostic asymptotically follows the standardized normal distribution. Values that lie in its tails provide an indication of non-convergence.

In our case, all Z-scores provided to all chains of the parameters of interest have fallen into the interval ± 1.96 in Geweke Diagnostics. This result provides no evidence of lack of convergence of MCMC Algorithm.

This can be also seen in *Geweke Plots* where every batch lie inside the 95% confidence interval, i.e.,$\pm 2 * SE$ which reinforces the no evidence of lack of convergence.

In same way, we obtained values near or equal to 1.0 in *Gelman Diagnostics*, which means there is no indication that the chains have not converged in *Run-1* model.

Gelman Plots shows in what iteration the chains approximately stabilizes.

We can conclude the parameter `n.burnin` $= 10,000$ reduced the fluctuation of all chains to an acceptable scale-reduction over time[2].

[1] See [6].

[2] We noticed some fluctuation at the beginning of each chain but there is no evidence they have not converged.

Question 2

Consider the following data which contains information on harvesting dates versus crop yields. The **x** variable is data on date of harvesting (which is the number of days after flowering) and yield **y** (kg/ha) of paddy, a grain farmed in India. (Data from Devore pg 518 and originally from *J. Agricultural Eng. Research*, 1975, pp 353-363.)

Here is the data:

```
list(x=c(16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46), y=c(2508, 2518,
3304, 3423, 3057, 3190, 3500, 3883, 3823, 3646, 3708, 3333, 3517, 3241, 3103, 2776))
```

Consider two models for this data. The first model predicts the population as a linear function of the date and the second predicts with a quadratic function. The following WinBugs/OpenBugs code will fit these models (This information is in a txt file that is sent with this file. You can get the the data and model codes from that file and you won't have to type them in):

Consider two models for this data. The first model predicts the population as a linear function of the date and the second predicts with a quadratic function. The following WinBugs code will fit these models (This information is in a txt file that is sent with this file. You can get the the data and model codes from that file and you won't have to type them in):

Model 1:

```
model{
for(i in 1:16){
  y[i]~dnorm(mu[i],tau)
  mu[i]<- b[1] + b[2]*(x[i]-31)
}
b[1]~dnorm(0,.000001)
b[2]~dnorm(0,.000001)
tau~dgamma(.0001,.0001)
}
```

Model 2:

```
model{
for(i in 1:16){
  y[i]~dnorm(mu[i],tau)
  mu[i]<- b[1] + b[2]*(x[i]-31)+b[3]*pow((x[i]-31),2)
}
b[1]~dnorm(0,.000001)
b[2]~dnorm(0,.000001)
b[3]~dnorm(0,.01)
tau~dgamma(.0001,.0001)
}
```

Do the following two parts:

item (a)

Compare these two models. First, compare these two models by looking at the "deviance" measures and the DIC. Calculate these values for each model and comment on them. Then, compare these two models by calculating the Bayes Factor. To calculate the Bayes factor, run an MCMC algorithm which switches between the two models using a method similar (in which you might have to somewhat change the model code as well as the model) to the method proposed by Kuo and Mallick. Comment on your belief between

the two models. (That is, which model do you prefer and justify your answer.)

{*Solution.*}

The data of our problem is presented in table below.

Table 4: (x):Harvesting Date (in days) vs. (y):Yield (in Kg/Ha)

x	y
16	2508
18	2518
20	3304
22	3423
24	3057
26	3190
28	3500
30	3883
32	3823
34	3646
36	3708
38	3333
40	3517
42	3241
44	3103
46	2776

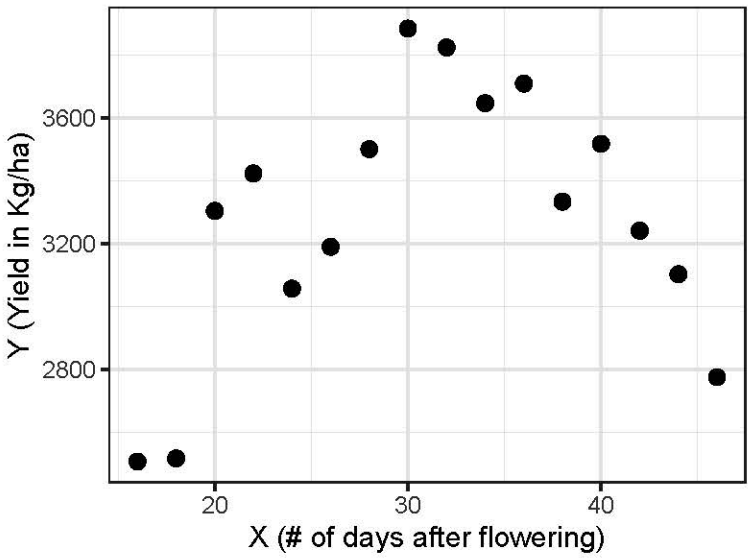


Figure 9: Scatterplot Harvesting Data - X vs. Y

For each Model, we simulated MCMC samples using *OpenBugs* with parameters `n.iter`= 30,000 iterations, `n.burnin`= 10,000 and `n.thin`= 10. The summary of results are as follows:

Table 5: OpenBugs Summary - Harvesting Date vs. Yield (Model-1)

Parameter	mean	sd	2.5%	50%	97.5%
b[1]	3241.2470	113.1482	3008.0	3244.00	3457.0000
b[2]	12.2725	12.3122	-12.3	12.29	36.6503
tau	0.0000	0.0000	0.0	0.00	0.0000

Checking Convergence and Auto-Correlation for Model-1

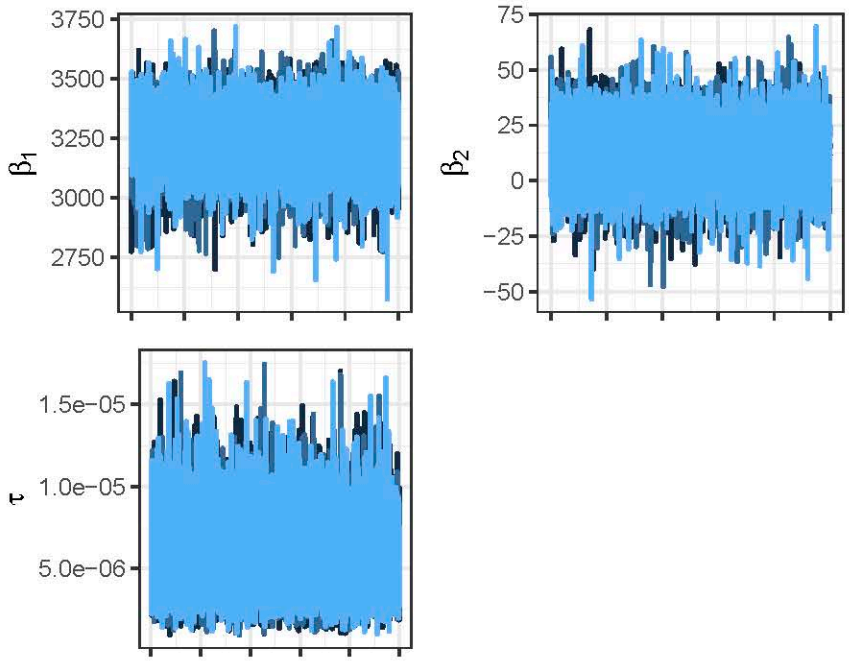


Figure 10: Trace-plots of β_1, β_2 and τ for Model-1

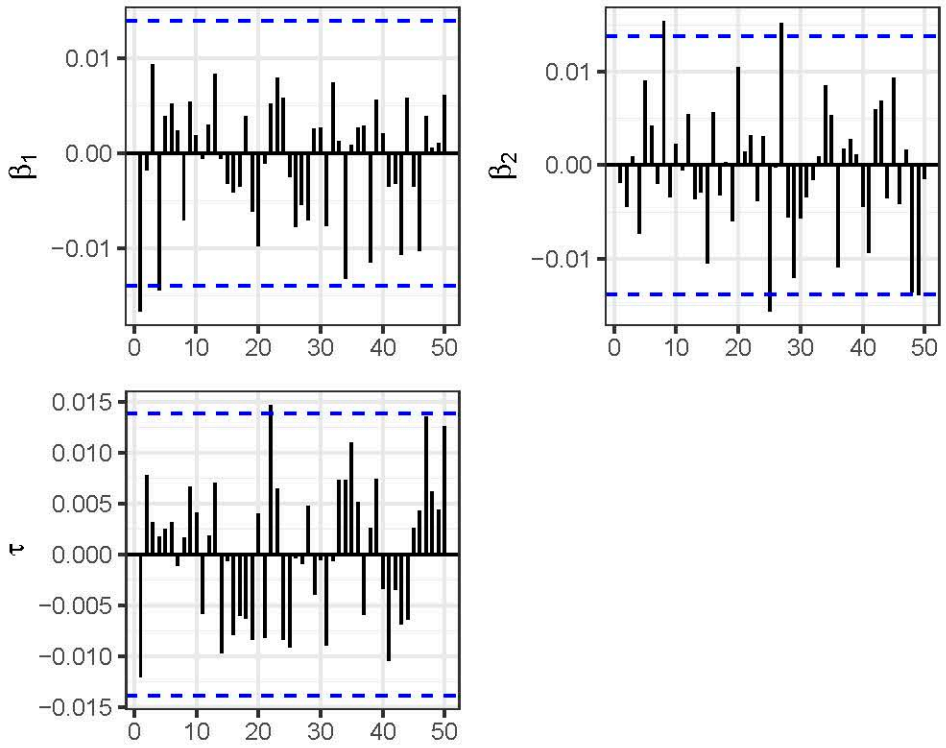


Figure 11: ACF-plots of β_1, β_2 and τ for Model-1 (max-Lag=50)

Comment:- We can see from trace and ACF plots that we have no evidence that our estimates have not converged or are auto-correlated for **Model-1**.

Table 6: OpenBugs Summary - Harvesting Date vs. Yield (Model-2)

Parameter	mean	sd	2.5%	50%	97.5%
b[1]	3640.2481	84.5355	3465.0000	3643.000	3801.000
b[2]	12.2888	6.0445	0.2991	12.310	24.240
b[3]	-4.3367	0.7391	-5.7540	-4.356	-2.813
tau	0.0000	0.0000	0.0000	0.000	0.000

Checking Convergence and Auto-Correlation for Model-2

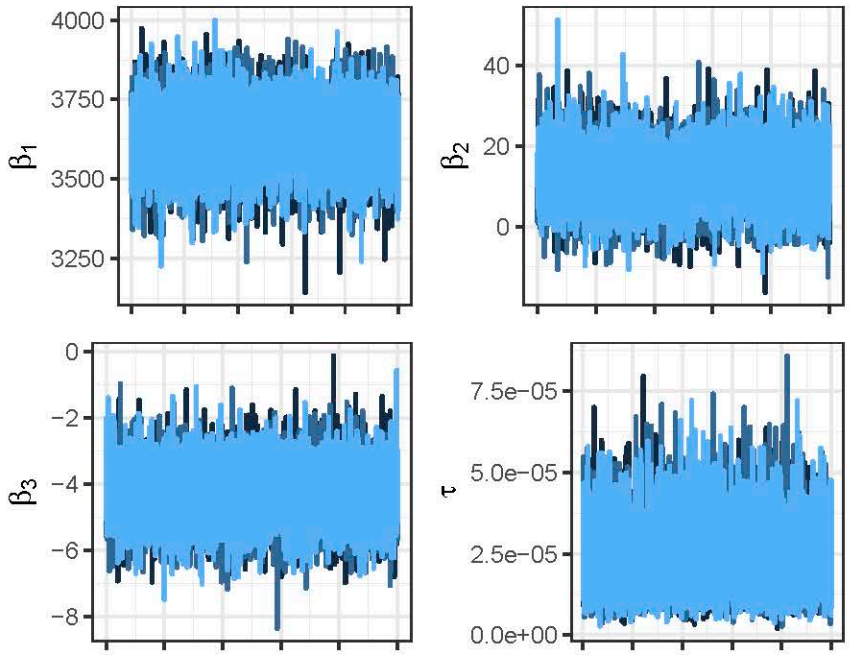


Figure 12: Trace-plots of β_1, β_2, β_3 and τ for Model-2

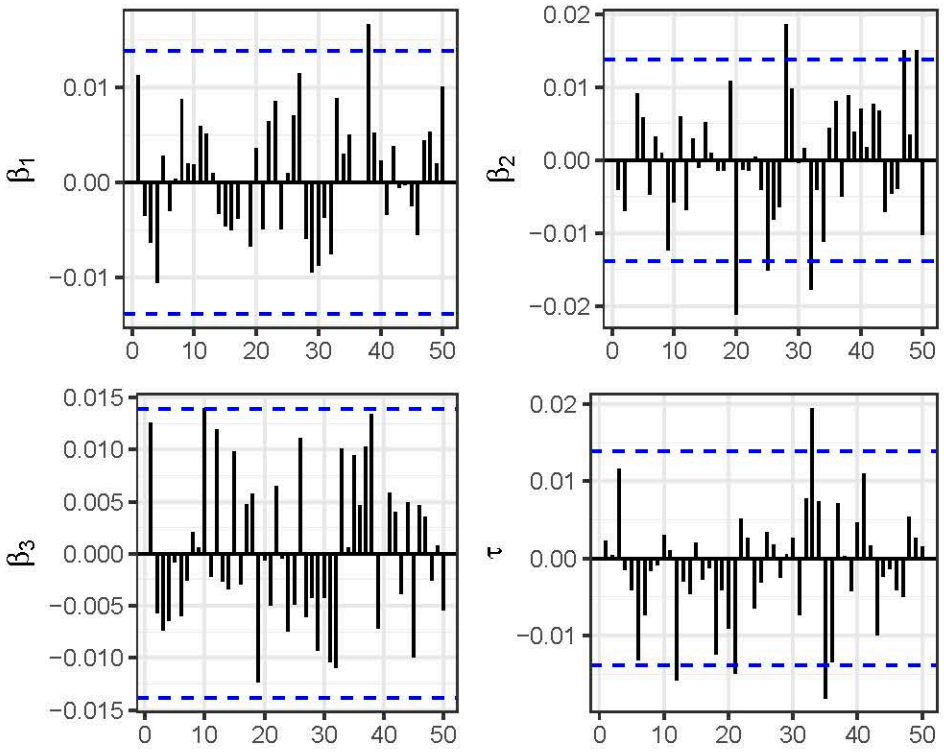


Figure 13: ACF-plots of β_1, β_2, β_3 and τ for Model-2 (max-Lag=50)

Comment:- We can see from trace and ACF plots that we have no evidence of our estimates have not converged or are auto-correlated for **Model-2**.

In this sense, we can proceed with the comparative analysis between the two models.

Comparing Model-1 vs. Model-2

In this subsection, we will compare *DIC*[3] Statistics in order to identify the best between the two models.

```
##
## ------- Goodness of Fit Statistics for Model-1  --------
## Deviance:
##                  2.5%   25% 50% 75% 97.5%
## SelfPrograMed: 236.5 237.6 239 241 247.2
## Openbugs Made: 236.5 237.6 239 241 247.2
##
## Dbar:  239.7148
## pD:   4.1139
## DIC:  243.8288
## ------------------------------------------------------

##
## ------- Goodness of Fit Statistics for Model-2  --------
## Deviance:
##                  2.5%   25%   50%   75% 97.5%
## SelfPrograMed: 212.8 214.5 216.2 218.6 225.6
## Openbugs Made: 212.8 214.5 216.2 218.6 225.6
##
## Dbar:  216.9903
## pD:   5.7376
## DIC:  222.7279
## ------------------------------------------------------
```

Comment:- After calculating the DIC and Deviance of both models, we obtained the smallest deviance and DIC for **Model-2** which would indicate in this case, it is the best model.

Bayes Factor by method Kuo-Mallick

In order to calculate Bayes Factor by running the MCMC with the method Kuo-Mallick, we will use the following model:

Openbugs Model

```
### Kuo-Mallick Model Selection
model{
  for (i in 1:N) {
    sy[i]~dnorm(mu[i],tau)
    mu[i]<- delta[1]*beta[1]*sx[i] + delta[2]*beta[2]*pow(sx[i],2)
  }

  for (ix in 1:2) {
    beta[ix]~dnorm(0,tau0)
```

[3]Instead of using the Deviance and DIC obtained from OpenBugs report, we calculated these statistics from our model. The complete model used as well as the detailed R-Code can be found at the Appendix of this document.

```
  }

  tau~dgamma(.5,.01)

  # Prior Distribution of delta[]
  for(k in 1:2){
    delta[k]~dbern(pp)
  }

  # Model Deltas
  for(i1 in 1:2){
    for(i2 in 1:2){
      mod[i1,i2]<-equals((2-i1),delta[1])*equals((2-i2),delta[2])
     }
  }

}
```

Table 7: OpenBugs Summary - Kuo-Mallik Model Selection Statistics

Parameter	mean	sd	2.5%	50%	97.5%
beta[1]	0.0187	3.7046	-7.3550	0.1937	7.5566
beta[2]	-0.2300	2.7580	-6.5563	-0.4163	6.3476
tau	1.3587	0.5316	0.5556	1.2770	2.6021
delta[1]	0.1231	0.3286	0.0000	0.0000	1.0000
delta[2]	0.5173	0.4997	0.0000	1.0000	1.0000
mod[1,1]	0.0628	0.2426	0.0000	0.0000	1.0000
mod[1,2]	0.0603	0.2381	0.0000	0.0000	1.0000
mod[2,1]	0.4546	0.4980	0.0000	0.0000	1.0000
mod[2,2]	0.4223	0.4940	0.0000	0.0000	1.0000
deviance	42.6662	3.1678	38.2900	43.6850	48.5602

IN the present analysis, Model-1 is represented by parameter *mod[1,2]* and Model-2 by the parameter *mod[1,1]*.

```
##
## -------- Kuo-Mallik Bayes Factor Analysis ( M2 / M1 ) ---------
##
## Posterior Probability M2 | Data:  0.0628
##
## Posterior Probability M1 | Data:  0.0603
##
## >>> Bayes Factor :  1.0405
##
## >>> Jeffrey's Evidence of M2 over M1 :  Not worth more than a bare mention
##
## >>> Kass/Raftery's Evidence of M2 over M1 :  Not worth more than a bare mention
## ------------------------------------------------------------
```

Comment:- From *Kuo-Mallik* Method we obtained a Bayes Factor(BF) of 1.0405. Using Jeffrey's and Kass/Raftery's Criteria to compare the models, both provided the same result, i.e., *there is no sufficient evidence that Model-2 is better than Model-1* and *not worth more than a bare mention*

Considering this, we will choose the *highest posterior probability model* which is **Model-2**.

item (b)

For model 1, look at the residuals for the model using functions 1, 2, and 3. That is, calculate 1) the residuals, 2) the standardized residuals, and 3) the chance of getting a more extreme observation. For the residual and the standardize residual, please calculate the distribution of these statistics under the predictive distribution. Comment on the results of these statistics for the observation. Also, comment on how well you think the model fits the data.

{*Solution.*}

The OpenBugs simulation of *Model-1* generated the following results:

Residuals

The Residuals for **Model-1** is as follows:

Table 8: OpenBugs Summary - Residuals for Model-1

Parameter	mean	sd	2.5%	50%	97.5%
res[1]	-549.1603	215.9246	-968.7000	-552.3000	-111.1000
res[2]	-563.7058	195.3784	-942.8025	-566.9000	-167.0975
res[3]	197.7493	175.8822	-143.7025	195.0000	555.5000
res[4]	292.2046	157.8269	-14.3512	289.4000	613.3025
res[5]	-98.3406	141.7626	-371.2050	-101.3000	191.2000
res[6]	10.1145	128.4396	-236.3025	7.2240	272.2000
res[7]	295.5696	118.7839	68.1097	292.8000	538.1025
res[8]	654.0245	113.7326	436.2975	651.4000	887.7000
res[9]	569.4795	113.9015	352.3975	566.4000	805.0000
res[10]	367.9344	119.2667	141.0000	365.0000	613.9000
res[11]	405.3893	129.1833	159.3000	402.1500	670.5025
res[12]	5.8443	142.7053	-266.2000	2.7475	299.9000
res[13]	165.2994	158.9154	-137.1025	162.6000	493.0025
res[14]	-135.2458	177.0765	-472.8000	-138.0000	228.8025
res[15]	-297.7905	196.6487	-674.1000	-300.9000	105.0000
res[16]	-649.3358	217.2515	-1067.0000	-652.7000	-206.6975

Standardized Residuals

The Standardized Residuals for **Model-1** is as follows:

Table 9: OpenBugs Summary - Standardized Residuals for Model-1

Parameter	mean	sd	2.5%	50%	97.5%
stdres[1]	-1.2977	0.5535	-2.3840	-1.2955	-0.2104
stdres[2]	-1.3319	0.5186	-2.3500	-1.3290	-0.3192
stdres[3]	0.4576	0.3913	-0.3109	0.4572	1.2280
stdres[4]	0.6796	0.3609	-0.0302	0.6782	1.3900
stdres[5]	-0.2383	0.3214	-0.8672	-0.2381	0.3902
stdres[6]	0.0165	0.2835	-0.5407	0.0167	0.5717
stdres[7]	0.6874	0.2786	0.1436	0.6871	1.2370
stdres[8]	1.5298	0.3610	0.8400	1.5230	2.2570
stdres[9]	1.3311	0.3344	0.6917	1.3260	2.0000
stdres[10]	0.8574	0.2924	0.2898	0.8561	1.4350
stdres[11]	0.9453	0.3196	0.3270	0.9430	1.5740
stdres[12]	0.0063	0.3149	-0.6057	0.0065	0.6234
stdres[13]	0.3810	0.3513	-0.2997	0.3810	1.0730
stdres[14]	-0.3254	0.4013	-1.1050	-0.3251	0.4690
stdres[15]	-0.7075	0.4653	-1.6160	-0.7085	0.2106
stdres[16]	-1.5337	0.5814	-2.6660	-1.5340	-0.3949

Chance of Getting Extreme Observations

The Chance of Getting Extreme Observations for **Model-1** is as follows:

Table 10: OpenBugs Summary - Chance of Extreme Observations for Model-1

Parameter	mean	sd
p.smaller[1]	0.1287	0.3349
p.smaller[2]	0.1198	0.3247
p.smaller[3]	0.6676	0.4711
p.smaller[4]	0.7351	0.4413
p.smaller[5]	0.4120	0.4922
p.smaller[6]	0.5024	0.5000
p.smaller[7]	0.7473	0.4346
p.smaller[8]	0.9243	0.2645
p.smaller[9]	0.8990	0.3014
p.smaller[10]	0.7931	0.4051
p.smaller[11]	0.8151	0.3882
p.smaller[12]	0.5051	0.5000
p.smaller[13]	0.6450	0.4785
p.smaller[14]	0.3796	0.4853
p.smaller[15]	0.2598	0.4385
p.smaller[16]	0.0936	0.2913

Distribution of Residuals and Standardized Residuals under the Predictive Distribution

The Distribution of Residuals and Standardized Residuals under the Predictive Distribution can be represented by the following posterior distributions:

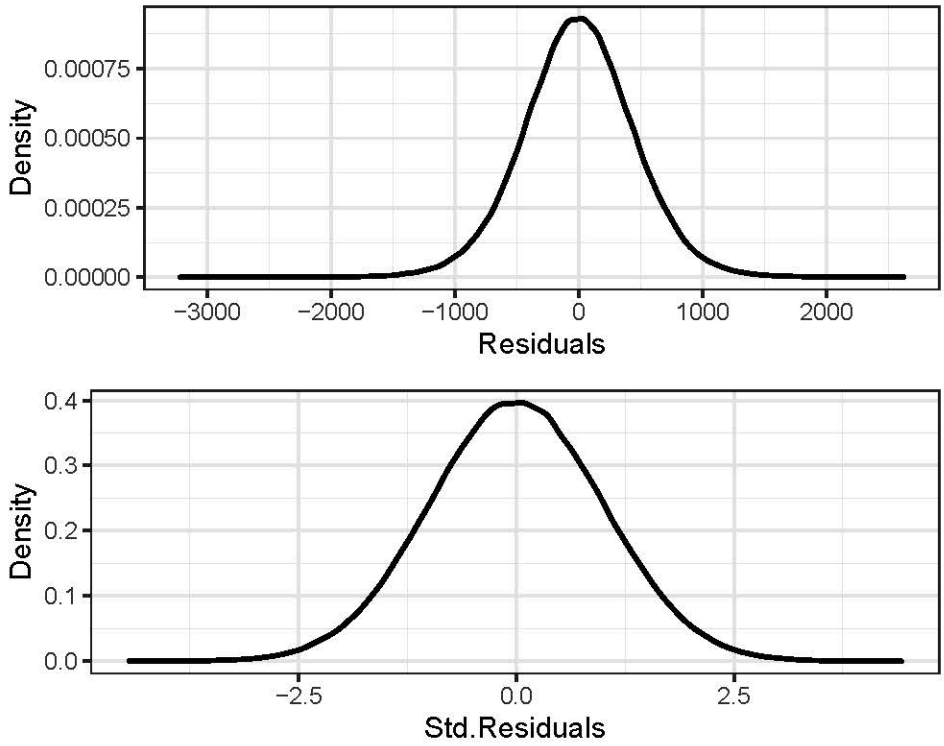


Figure 14: Model-1 - Residuals under Predictive Distribution

Overview of Model-1

Overall, the goodness of fit of **Model-1** to our data is not good.

There are a few reasons for it:

- The probability of Model-1, given the data, is around 6%, which is low if we compare it among the remaining alternative models;
- The DIC and Deviance of this model are both higher when compared, for instance, with alternative Model-2;
- The shape of scatter-plot of the source data suggests a quadratic relation between X and Y and this may impose limitations when we try to fit a purely linear model to the data (this is the case in Model-2).

When we compare the probability of presence of each variable in the final model we see that X^2 has higher probability reinforcing the relation between the parameters of our model and the data.

Table 11: Probability of Variables in model

Var	Prob
X	0.1231
X**2	0.5173

Models which have a quadratic term on it are considerably more probable than others that considers only the linear component.

One possible reason why Model-2 has also low probability might be due to the presence of the *linear component* on X. If we consider a *purely quadratic* model - i.e., with no linear term - and recalculate the Bayes-Factor we will obtain a quite different diagnostic.

Bayes Factor between *Model-1* and *Model-2Q* (Quadratic Model)

```
##
## -------- Kuo-Mallik Bayes Factor Analysis ( M2-Q / M1 ) ---------
##
## Posterior Probability M2-Q | Data:  0.4546
##
## Posterior Probability M1 | Data:  0.0603
##
## >>> Bayes Factor :  7.5341
##
## >>> Jeffrey's Evidence of M2-Q over M1 :  Substantial
##
## >>> Kass/Raftery's Evidence of M2-Q over M1 :  Positive
## ------------------------------------------------------------
```

In this case, there are evidences that **Model-2Q** outperforms *Model-1*. We could explore further other alternative models but this is out of the scope of the current question.

Final Remark:- This question were very useful to understand the Bayesian Process of Model Analysis and Diagnostics.

References

[1] Gelman, A., Carlin, J.B., Stern, H.S., Dunson, D.B., Vehtari, A., Rubin, D.B. *Bayesian Data Analysis.* pp.34 - CRC Press, 2014.

[2] Gelman, A., Hill, J. *Data Analysis using Regression and Multilevel/Hierarchical Models*, Cambridge Press, 2007.

[3] Marin, J. M., Robert, C. *Bayesian Essentials with R*, 2nd. Ed. pp.49 - Springer, 2014.

[4] Geweke J. *Bayesian Contemporary Econometrics and Statistics.* London: Wiley, 2005.

[5] Geweke J. *Evaluating the accuracy of sampling-based approaches to calculating posterior moments*, Bayesian Statistics 4, Clarendon Press, Oxford, UK, (1992)

[6] Ntzoufras I. *Bayesian Modeling Using WinBUGS*, Wiley, 2009

Appendix - R-Code

```
library(tidyverse)
library(R2OpenBUGS)
library(kableExtra)
library(ggplot2)
library(forecast)
library(coda)
# library(boa)

# The palette with black - Used in Graphs with :
cbp1 <- c("#999999", "#E69F00", "#56B4E9", "#009E73",
          "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
cbp2 <- c("#000000", "#E69F00", "#56B4E9", "#009E73",
          "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
cbp3 <- c("#FFDB6D", "#C4961A", "#F4EDCA", "#D16103",
          "#C3D7A4", "#52854C", "#4E84C4", "#293352")

#
#data from Healy, page 90.
#  (MJR Healy, 1988, Glim: An Introduction, Clarendon Press: Oxford.)
#  Looking to see if Smoking is a risk factor for hypertension, controlling for obesity, snoring, and g
#  Note 1: there was no males or females who were smokers and obese and who did not snore (so 1 1 0 had
#  Note 2: here we are simply looking at the effect of smoking given the other factors.  We are ignorin
#  obesity might be related to smoking or that snoring might be strongly effected by smoking and obesit
#  In modern epi, these factors might be consider to be in the <<causal path>> and perhaps you might no
cat(
"smoke  obese  snore male hypoten n
0 0 0 1 5 60
0 0 0 0 10 149
1 0 0 1 2 17
1 0 0 0 6 16
0 1 0 1 1 12
0 1 0 0 2 9
0 0 1 1 36 187
0 0 1 0 28 138
1 0 1 1 13 85
1 0 1 0 4 39
0 1 1 1 15 51
0 1 1 0 11 28
1 1 1 1 8 23
1 1 1 0 4 12
", file= "Data/SmokeHyperData.txt")

# Setup Data-set - Read Data
SmokeHyper=read.table("Data/SmokeHyperData.txt",header=TRUE,sep = "")

# Setup Variables
N=nrow(SmokeHyper)  # Number of Items in data-set

# Setup OpenBugs running parameters
NSim <- 30000     # No. of simulations for productio
NChain <- 3       # No. of chains for production
```

```
NThin <- 50       # n.thin parameter for production
Burnin <- 10000   # Burn-In parameter for production
Sz <- 5000        # Size of samples for trace/acf plots


# Printing the Data-Frame
SmokeHyper %>%
  kbl(booktabs = TRUE, digits = 4,
      caption = "Data - Relationship between Hypertension and Smoking") %>%
  kable_styling(latex_options = "striped")


cat("
model{
  for(i in 1:N){
   hypoten[i] ~ dbin(mu[i], n[i])
   logit(mu[i]) <- b0 + b.smok*smoke[i]+ b.ob*obese[i]+ b.sn*snore[i] +
     b.male*male[i] + b.smsn*smoke[i]*snore[i] + b[i]
    b[i] ~dnorm(0, tau.b)
   }
  b.smok ~ dnorm(0, .04) # so, sd =5.  exp(5) ~ 148 which is huge
  b.ob ~ dnorm(0, .04)
  b.sn ~ dnorm(0, .04)
  b.male ~ dnorm(0, .04)
  b0 ~ dnorm(0, .04)
  b.smsn ~dnorm(0, .04)
  sd.b ~ dunif(0, 5)
  tau.b <- 1/pow(sd.b,2)
  }
  ", file="SmokeHyperMod3.txt")

# Setup Parameters
paramsM0=c("b0", "b.smok", "b.ob", "b.sn", "b.male", "b.smsn" , "sd.b")

bugM0.dat=list("hypoten", "n", "smoke", "obese", "snore", "male", "N")  # what variable you need in the


# Setup Initial Values
initM0.fun=function(){ list(  b=runif(N,-.8,-.2),
    b0=runif(1,-.8,-.2),
    b.smok=runif(1,-.8,-.2),b.ob=runif(1,-.8,-.2), b.sn=runif(1,-.8,-.2),
    b.male=runif(1,-.8,-.2), b.smsn=runif(1, -8,-.2), sd.b=runif(1,.2,.8)
    ) }

#### Could change the code below...
# Run Open Bugs - NO BURNIN / NO THINN
set.seed(2602)
attach(SmokeHyper)
SmokeHypeBaseM0=bugs(bugM0.dat, initM0.fun, paramsM0, model.file="SmokeHyperMod3.txt",
    n.chains=NChain, n.iter=NSim, n.burnin=0, n.thin=1, debug=FALSE)
detach(SmokeHyper)
```

```r
# Get Simulation from OpenBugs
SArrayM0= SmokeHypeBaseM0$sims.array    # Data Arrays
vname=attr(SArrayM0,"dimnames")[3][[1]]  # Variable Names


# Print Summary statistics of parameters oif Interest
RNames <- rownames(SmokeHypeBaseM0[["summary"]][,c(1:3,5,7)]) # List of parameters
df_Prt <- data.frame(Parameter = RNames)
df_Prt <- cbind(df_Prt, as_tibble(SmokeHypeBaseM0[["summary"]][,c(1:3,5,7)]))
df_Prt %>%
  kbl(booktabs = TRUE, digits = 4,
      caption = "OpenBugs Summary - Hypertension and Smoke Study (Run-0)") %>%
  kable_styling(latex_options = "striped")

# Plot TracePlots * SIGMA, BETA0, BETA.SMOKE and BETA.OBESITY *

# Sampling 5000 points to generate "thinner" traceplots
set.seed(312)

L <- NSim-Burnin

OldSz <- Sz
Sz <- L       # Override Original value

S <- if (Sz < L) sort(sample(1:L, Sz, replace = FALSE)) else 1:Sz

# Build working dataframe
D_WorkSigma <- data.frame(ValCh1=SArrayM0[S,1,paste0("sd.b")],
                          ValCh2=SArrayM0[S,2,paste0("sd.b")],
                          ValCh3=SArrayM0[S,3,paste0("sd.b")])

D_WorkBeta0 <- data.frame(ValCh1=SArrayM0[S,1,paste0("b0")],
                          ValCh2=SArrayM0[S,2,paste0("b0")],
                          ValCh3=SArrayM0[S,3,paste0("b0")])

D_WorkBetaSmok <- data.frame(ValCh1=SArrayM0[S,1,paste0("b.smok")],
                             ValCh2=SArrayM0[S,2,paste0("b.smok")],
                             ValCh3=SArrayM0[S,3,paste0("b.smok")])

D_WorkBetaObes <- data.frame(ValCh1=SArrayM0[S,1,paste0("b.ob")],
                             ValCh2=SArrayM0[S,2,paste0("b.ob")],
                             ValCh3=SArrayM0[S,3,paste0("b.ob")])

# Plot Traceplots for selected data-frames
P1 <- D_WorkSigma %>%
        ggplot(aes(seq(from=1,to=Sz)))+
        geom_line(aes(y=ValCh1, colour=1), size=0.8)+
        geom_line(aes(y=ValCh2, colour=2), size=0.8)+
        geom_line(aes(y=ValCh3, colour=3), size=0.8)+
        labs(y = expression(sigma)) +
        theme_bw()
P2 <- D_WorkBeta0 %>%
        ggplot(aes(seq(from=1,to=Sz)))+
```

```
        geom_line(aes(y=ValCh1, colour=1), size=0.8)+
        geom_line(aes(y=ValCh2, colour=2), size=0.8)+
        geom_line(aes(y=ValCh3, colour=3), size=0.8)+
        labs(y = expression(beta[0])) +
        theme_bw()
P3 <- D_WorkBetaSmok %>%
        ggplot(aes(seq(from=1,to=Sz)))+
        geom_line(aes(y=ValCh1, colour=1), size=0.8)+
        geom_line(aes(y=ValCh2, colour=2), size=0.8)+
        geom_line(aes(y=ValCh3, colour=3), size=0.8)+
        labs(y = expression(beta[Smoke])) +
        theme_bw()
P4 <- D_WorkBetaObes %>%
        ggplot(aes(seq(from=1,to=Sz)))+
        geom_line(aes(y=ValCh1, colour=1), size=0.8)+
        geom_line(aes(y=ValCh2, colour=2), size=0.8)+
        geom_line(aes(y=ValCh3, colour=3), size=0.8)+
        labs(y = expression(beta[Obesity])) +
        theme_bw()

# Plot all Graphs in a same frame
ggpubr::ggarrange(P1+theme(axis.text.x=element_blank(),
                           axis.title.x=element_blank(),
                           legend.position="none"),
                  P2+theme(axis.text.x=element_blank(),
                           axis.title.x=element_blank(),
                           legend.position="none"),
                  P3+theme(axis.text.x=element_blank(),
                           axis.title.x=element_blank(),
                           legend.position="none"),
                  P4+theme(axis.text.x=element_blank(),
                           axis.title.x=element_blank(),
                           legend.position="none"), ncol = 2, nrow = 2)

# Remove Working Variable to free memory
Sz <- OldSz # Restore original value
remove(OldSz, D_WorkSigma, D_WorkBeta0, D_WorkBetaSmok, D_WorkBetaObes )

# Plot ACF Plots * SIGMA, BETA0, BETA.SMOKE and BETA.OBESITY *

# Build working dataframe
D_WorkSigma <- data.frame(ValCh1=SArrayM0[,1,paste0("sd.b")])
D_WorkBeta0 <- data.frame(ValCh1=SArrayM0[,1,paste0("b0")])
D_WorkBetaSmok <- data.frame(ValCh1=SArrayM0[,1,paste0("b.smok")])
D_WorkBetaObes <- data.frame(ValCh1=SArrayM0[,1,paste0("b.ob")])

# Plot ACF Plots for selected data-frames
P1 <- ggAcf(D_WorkSigma$ValCh1, lag.max = 100)+
  labs(x = "Lag", y = expression(sigma)) +
  ggtitle(NULL)+
  theme_bw()
P2 <- ggAcf(D_WorkBeta0$ValCh1, lag.max = 100)+
  labs(x = "Lag", y = expression(beta[0])) +
```

```
  ggtitle(NULL)+
  theme_bw()
P3 <- ggAcf(D_WorkBetaSmok$ValCh1, lag.max = 100)+
  labs(x = "Lag", y = expression(beta[Smoke])) +
  ggtitle(NULL)+
  theme_bw()
P4 <- ggAcf(D_WorkBetaObes$ValCh1, lag.max = 100)+
  labs(x = "Lag", y = expression(beta[Obesity])) +
  ggtitle(NULL)+
  theme_bw()

# Plot all Graphs in a same frame
ggpubr::ggarrange(P1+theme(axis.title.x=element_blank(),
                           legend.position="none"),
                  P2+theme(axis.title.x=element_blank(),
                           legend.position="none"),
                  P3+theme(axis.title.x=element_blank(),
                           legend.position="none"),
                  P4+theme(axis.title.x=element_blank(),
                           legend.position="none"), ncol = 2, nrow = 2)
# Remove Working Variable to free memory
remove(D_WorkSigma, D_WorkBeta0, D_WorkBetaSmok, D_WorkBetaObes )

# Setup Parameters for Run-01

paramsM1 <- paramsM0

bugM1.dat <- bugM0.dat

initM1.fun <- initM0.fun

# Run Open Bugs - BURNIN=10,000 / THINN=50
set.seed(2212)
attach(SmokeHyper)
SmokeHypeBaseM1=bugs(bugM1.dat, initM1.fun, paramsM1, model.file="SmokeHyperMod3.txt",
    n.chains=NChain, n.iter=NSim, n.burnin=Burnin, n.thin=NThin , debug=FALSE)
detach(SmokeHyper)


# Get Simulation from OpenBugs
SArrayM1= SmokeHypeBaseM1$sims.array   # Data Arrays
vname=attr(SArrayM1,"dimnames")[3][[1]]  # Variable Names


# Print Summary statistics of parameters oif Interest
RNames <- rownames(SmokeHypeBaseM1[["summary"]][,c(1:3,5,7)]) # List of parameters
df_Prt <- data.frame(Parameter = RNames)
df_Prt <- cbind(df_Prt, as_tibble(SmokeHypeBaseM1[["summary"]][,c(1:3,5,7)]))
df_Prt %>%
  kbl(booktabs = TRUE, digits = 4,
      caption = "OpenBugs Summary - Hypertension and Smoke Study (Run-1)") %>%
  kable_styling(latex_options = "striped")
```

```r
# Plot TracePlots * SIGMA, BETA0, BETA.SMOKE and BETA.OBESITY *

# Sampling 5000 points to generate "thinner" traceplots
set.seed(312)

L <- NSim-Burnin

OldSz <- Sz
Sz <- L       # Override Original value

S <- if (Sz < L) sort(sample(1:L, Sz, replace = FALSE)) else 1:Sz

# Build working dataframe
D_WorkSigma <- data.frame(ValCh1=SArrayM1[S,1,paste0("sd.b")],
                          ValCh2=SArrayM1[S,2,paste0("sd.b")],
                          ValCh3=SArrayM1[S,3,paste0("sd.b")])

D_WorkBeta0 <- data.frame(ValCh1=SArrayM1[S,1,paste0("b0")],
                          ValCh2=SArrayM1[S,2,paste0("b0")],
                          ValCh3=SArrayM1[S,3,paste0("b0")])

D_WorkBetaSmok <- data.frame(ValCh1=SArrayM1[S,1,paste0("b.smok")],
                             ValCh2=SArrayM1[S,2,paste0("b.smok")],
                             ValCh3=SArrayM1[S,3,paste0("b.smok")])

D_WorkBetaObes <- data.frame(ValCh1=SArrayM1[S,1,paste0("b.ob")],
                             ValCh2=SArrayM1[S,2,paste0("b.ob")],
                             ValCh3=SArrayM1[S,3,paste0("b.ob")])

# Plot Traceplots for selected data-frames
P1 <- D_WorkSigma %>%
        ggplot(aes(seq(from=1,to=Sz)))+
        geom_line(aes(y=ValCh1, colour=1), size=0.8)+
        geom_line(aes(y=ValCh2, colour=2), size=0.8)+
        geom_line(aes(y=ValCh3, colour=3), size=0.8)+
        labs(y = expression(sigma)) +
        theme_bw()
P2 <- D_WorkBeta0 %>%
        ggplot(aes(seq(from=1,to=Sz)))+
        geom_line(aes(y=ValCh1, colour=1), size=0.8)+
        geom_line(aes(y=ValCh2, colour=2), size=0.8)+
        geom_line(aes(y=ValCh3, colour=3), size=0.8)+
        labs(y = expression(beta[0])) +
        theme_bw()
P3 <- D_WorkBetaSmok %>%
        ggplot(aes(seq(from=1,to=Sz)))+
        geom_line(aes(y=ValCh1, colour=1), size=0.8)+
        geom_line(aes(y=ValCh2, colour=2), size=0.8)+
        geom_line(aes(y=ValCh3, colour=3), size=0.8)+
        labs(y = expression(beta[Smoke])) +
        theme_bw()
P4 <- D_WorkBetaObes %>%
        ggplot(aes(seq(from=1,to=Sz)))+
```

```
        geom_line(aes(y=ValCh1, colour=1), size=0.8)+
        geom_line(aes(y=ValCh2, colour=2), size=0.8)+
        geom_line(aes(y=ValCh3, colour=3), size=0.8)+
        labs(y = expression(beta[Obesity])) +
        theme_bw()

# Plot all Graphs in a same frame
ggpubr::ggarrange(P1+theme(axis.text.x=element_blank(),
                           axis.title.x=element_blank(),
                           legend.position="none"),
                  P2+theme(axis.text.x=element_blank(),
                           axis.title.x=element_blank(),
                           legend.position="none"),
                  P3+theme(axis.text.x=element_blank(),
                           axis.title.x=element_blank(),
                           legend.position="none"),
                  P4+theme(axis.text.x=element_blank(),
                           axis.title.x=element_blank(),
                           legend.position="none"), ncol = 2, nrow = 2)

# Remove Working Variable to free memory
Sz <- OldSz # Restore original value
remove(OldSz, D_WorkSigma, D_WorkBeta0, D_WorkBetaSmok, D_WorkBetaObes )

# Plot ACF Plots * SIGMA, BETA0, BETA.SMOKE and BETA.OBESITY *

# Build working dataframe
D_WorkSigma <- data.frame(ValCh1=SArrayM1[,1,paste0("sd.b")])
D_WorkBeta0 <- data.frame(ValCh1=SArrayM1[,1,paste0("b0")])
D_WorkBetaSmok <- data.frame(ValCh1=SArrayM1[,1,paste0("b.smok")])
D_WorkBetaObes <- data.frame(ValCh1=SArrayM1[,1,paste0("b.ob")])

# Plot ACF Plots for selected data-frames
P1 <- ggAcf(D_WorkSigma$ValCh1, lag.max = 100)+
  labs(x = "Lag", y = expression(sigma)) +
  ggtitle(NULL)+
  theme_bw()
P2 <- ggAcf(D_WorkBeta0$ValCh1, lag.max = 100)+
  labs(x = "Lag", y = expression(beta[0])) +
  ggtitle(NULL)+
  theme_bw()
P3 <- ggAcf(D_WorkBetaSmok$ValCh1, lag.max = 100)+
  labs(x = "Lag", y = expression(beta[Smoke])) +
  ggtitle(NULL)+
  theme_bw()
P4 <- ggAcf(D_WorkBetaObes$ValCh1, lag.max = 100)+
  labs(x = "Lag", y = expression(beta[Obesity])) +
  ggtitle(NULL)+
  theme_bw()

# Plot all Graphs in a same frame
ggpubr::ggarrange(P1+theme(axis.title.x=element_blank(),
                           legend.position="none"),
```

```
                  P2+theme(axis.title.x=element_blank(),
                           legend.position="none"),
                  P3+theme(axis.title.x=element_blank(),
                           legend.position="none"),
                  P4+theme(axis.title.x=element_blank(),
                           legend.position="none"), ncol = 2, nrow = 2)
# Remove Working Variable to free memory
remove(D_WorkSigma, D_WorkBeta0, D_WorkBetaSmok, D_WorkBetaObes )

# Prepare data to plot densities to Compare Distributions of
# * SIGMA, BETA0, BETA.SMOKE and BETA.OBESITY *

set.seed(9023)

L <- NSim-Burnin

S <- if (Sz < L) sort(sample(1:L, Sz, replace = FALSE)) else 1:Sz

# Build working dataframe
D_WorkSigma <- as_tibble(rbind(data.frame(LvlStd=as.vector(SArrayM1[S,1,paste0("sd.b")]), Chain = "C1")
                               data.frame(LvlStd=as.vector(SArrayM1[S,2,paste0("sd.b")]), Chain = "C2")
                               data.frame(LvlStd=as.vector(SArrayM1[S,3,paste0("sd.b")]), Chain = "C3"))
D_WorkBeta0 <- as_tibble(rbind(data.frame(LvlStd=as.vector(SArrayM1[S,1,paste0("b0")]), Chain = "C1"),
                               data.frame(LvlStd=as.vector(SArrayM1[S,2,paste0("b0")]), Chain = "C2"),
                               data.frame(LvlStd=as.vector(SArrayM1[S,3,paste0("b0")]), Chain = "C3")))
D_WorkBetaSmok <- as_tibble(rbind(data.frame(LvlStd=as.vector(SArrayM1[S,1,paste0("b.smok")]), Chain =
                                  data.frame(LvlStd=as.vector(SArrayM1[S,2,paste0("b.smok")]), Chain =
                                  data.frame(LvlStd=as.vector(SArrayM1[S,3,paste0("b.smok")]), Chain =
D_WorkBetaObes <- as_tibble(rbind(data.frame(LvlStd=as.vector(SArrayM1[S,1,paste0("b.ob")]), Chain = "C
                                  data.frame(LvlStd=as.vector(SArrayM1[S,2,paste0("b.ob")]), Chain = "C
                                  data.frame(LvlStd=as.vector(SArrayM1[S,3,paste0("b.ob")]), Chain = "C

# Calculates Batch Means and respective SE - Author: Michael Escobar (thank you!)
CalcBatchMeans <- function (x, Batn = 50) {
  BigN=length(x)
  BatInc=ceiling((1:BigN)/(BigN/Batn) )
  BM=tapply(x,BatInc,mean)
  return(list(MCE=(sd(BM)/sqrt(length(BM))), BM=BM))
}

# Calculates the Standard Error via Auto-Correlation Function - Author: Michael Escobar (thank you!)
CalcAcSe=function(x,lag.max=50){
  autoc=(acf(x,lag.max=lag.max,plot=FALSE))$acf
  sd(x)/sqrt(length(x))*sqrt(-1+2*sum(autoc))
}

CalcChainStats <- function (parm, chain, TxtParm) {
  L <- CalcBatchMeans(SArrayM1[,chain,paste0(parm)])
  S <- CalcAcSe(SArrayM1[,chain,paste0(parm)],120)
  cat("\n---- Summary Statistics for",TxtParm,"(Chain ",chain,") -----")
  cat("\nB-Mean : ",format(mean(L$BM), digits = 2, nsmall = 4))
  cat("\nB-S.E.: ",format(L$MCE, digits = 2, nsmall = 6))
  CI_up <- mean(L$BM)+2*L$MCE; CI_lo <- mean(L$BM)-2*L$MCE;
```

```
  cat("\n95% Credible Region (",
      format(CI_lo, digits = 2, nsmall = 4), ",",
      format(CI_up, digits = 2, nsmall = 4),")\n")
  cat("\n(ACF) S.E.: ",format(S, digits = 2, nsmall = 6))
  CI_up <- mean(L$BM)+2*S; CI_lo <- mean(L$BM)-2*S;
  cat("\n(ACF) 95% Credible Region (",
      format(CI_lo, digits = 2, nsmall = 4), ",",
      format(CI_up, digits = 2, nsmall = 4),")\n")
  cat("-------------------------------------------------\n")
}

# Calculates Posterior Means using Batch-Means and ACF - SIGMA
CalcChainStats("sd.b",1,"SIGMA")
CalcChainStats("sd.b",2,"SIGMA")
CalcChainStats("sd.b",3,"SIGMA")

D_WorkSigma %>%
  ggplot(aes(x = LvlStd))+
  geom_density(aes(x=LvlStd, group=Chain, colour=Chain), size=0.8)+
  labs(x = expression(sigma), y = "Density") +
  scale_color_manual(values = cbp2)+
  theme_bw()

# Calculates Posterior Means using Batch-Means and ACF - BETA0
CalcChainStats("b0",1,"BETA0")
CalcChainStats("b0",2,"BETA0")
CalcChainStats("b0",3,"BETA0")


D_WorkBeta0 %>%
  ggplot(aes(x = LvlStd))+
  geom_density(aes(x=LvlStd, group=Chain, colour=Chain), size=0.8)+
  labs(x = expression(beta[0]), y = "Density") +
  scale_color_manual(values = cbp2)+
  theme_bw()

# Calculates Posterior Means using Batch-Means and ACF - BETA-SMOKE
CalcChainStats("b.smok",1,"BETA-SMOKE")
CalcChainStats("b.smok",2,"BETA-SMOKE")
CalcChainStats("b.smok",3,"BETA-SMOKE")



D_WorkBetaSmok %>%
  ggplot(aes(x = LvlStd))+
  geom_density(aes(x=LvlStd, group=Chain, colour=Chain), size=0.8)+
  labs(x = expression(beta[Smoke]), y = "Density") +
  scale_color_manual(values = cbp2)+
  theme_bw()

# Calculates Posterior Means using Batch-Means and ACF - BETA-OBESITY
CalcChainStats("b.ob",1,"BETA-OBESITY")
CalcChainStats("b.ob",2,"BETA-OBESITY")
```

```
CalcChainStats("b.ob",3,"BETA-OBESITY")


D_WorkBetaObes %>%
  ggplot(aes(x = LvlStd))+
  geom_density(aes(x=LvlStd, group=Chain, colour=Chain), size=0.8)+
  labs(x = expression(beta[Obesity]), y = "Density") +
  scale_color_manual(values = cbp2)+
  theme_bw()

# Remove Working Variables and workdataframes used in this question to free memory
remove(D_WorkSigma, D_WorkBeta0, D_WorkBetaSmok, D_WorkBetaObes )

# Prepare Structures for CODA
Chn.sigma <- list( C1=mcmc(SArrayM1[,1,paste0("sd.b")]),
                   C2=mcmc(SArrayM1[,2,paste0("sd.b")]),
                   C3=mcmc(SArrayM1[,3,paste0("sd.b")]))
Chn.b0 <- list( C1=mcmc(SArrayM1[,1,paste0("b0")]),
                C2=mcmc(SArrayM1[,2,paste0("b0")]),
                C3=mcmc(SArrayM1[,3,paste0("b0")]))
Chn.bsmok <- list( C1=mcmc(SArrayM1[,1,paste0("b.smok")]),
                   C2=mcmc(SArrayM1[,2,paste0("b.smok")]),
                   C3=mcmc(SArrayM1[,3,paste0("b.smok")]))
Chn.bob <- list( C1=mcmc(SArrayM1[,1,paste0("b.ob")]),
                 C2=mcmc(SArrayM1[,2,paste0("b.ob")]),
                 C3=mcmc(SArrayM1[,3,paste0("b.ob")]))

# Calculate Summary MCMC of estimates
CalcDiags <- function (MC, TxtParm) {
  MMC <- cbind(MC$C1,MC$C2,MC$C3)
  colnames(MMC) <- c("Chain1","Chain2","Chain3")
  cat("\n--------- CODA Summary Statistics for",TxtParm,"---------\n")
  print(summary(mcmc(MMC)))
  cat(">>> Effective Size:\n")
  print(effectiveSize(MMC))
  cat("\n>>> GEWEKE Diagnostics:")
  print(geweke.diag(MMC))
  geweke.plot(mcmc(MMC))
  cat(">>> GELMAN Diagnostics:\n")
  print(gelman.diag(MC))
  gelman.plot(MC)
}


CalcDiags(Chn.sigma, "SIGMA")

CalcDiags(Chn.b0, "BETA-0")

CalcDiags(Chn.bsmok, "BETA-SMOKE")

CalcDiags(Chn.bob, "BETA-OBESITY")

# Setup Data-set - Read Data
```

```
HarvestingData <- data.frame( x=c(16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46),
y=c(2508,2518,3304,3423,3057,3190,3500,3883,3823,3646,3708,
3333,3517,3241,3103,2776))

# Setup Variables
N=nrow(HarvestingData)  # Number of Items in data-set

# Setup OpenBugs running parameters
NSim <- 30000     # No. of simulations for productio
NChain <- 3       # No. of chains for production
NThin <- 10       # n.thin parameter for production
Burnin <- 10000   # Burn-In parameter for production
Sz <- 5000        # Size of samples for trace/acf plots


# Printing the Data-Frame
HarvestingData %>%
  kbl(booktabs = TRUE, digits = 1,
      caption = "(x):Harvesting Date (in days) vs. (y):Yield (in Kg/Ha)") %>%
  kable_styling(latex_options = "striped")


HarvestingData %>%
  ggplot(aes(x=x, y=y))+
  geom_point(size=2.5)+
  labs(x = "X (# of days after flowering)", y = "Y (Yield in Kg/ha)") +
  theme_bw()

# MODEL Q2_1
cat("### Model 1 -
model{
  for (i in 1:N) {
    y[i]~dnorm(mu[i],tau)
    mu[i]<- b[1] + b[2]*(x[i]-31)

    # Get the residuals for the observed value
    res[i] <- (y[i]-mu[i])           # Estimate the residual for this model  - Item (1)
    stdres[i] <- res[i]*sqrt(tau)    # Calculate the standard residuals - Item (2)

    dev1.obs[i] <- pow(res[i],2)
    dev2.obs[i] <- pow(stdres[i],2)

    # Get a replicated Sample - sample of the predictive distribution
    y.rep[i]~dnorm(mu[i],tau)
    p.smaller[i] <- step(y[i]-y.rep[i])  # Check to see the probability of getting extreme value - Item

    # Residual and Moments replicated - this gives the predicted distribution for these values
    res.rep[i] <- y.rep[i]-mu[i]
    stdres.rep[i] <- res.rep[i]*sqrt(tau)

    dev1.rep[i] <- pow(res.rep[i],2)
    dev2.rep[i] <- pow(stdres.rep[i],2)
```

```
    # Likelihood for each observed and replicated data
    # note: Need to know the density function of the probability model

    loglike[i] <- (0.5)*log(tau/6.283185) + (-0.5)*tau*pow((y[i]-mu[i]),2)
    loglike.rep[i]<-  (0.5)*log(tau/6.283185) + (-0.5)*tau*pow((y.rep[i]-mu[i]),2)

    p.inv[i]<- 1/exp(loglike[i])      #  This is to find the predictive ordinate of the observations
  }

  # Prior definitions
  b[1]~dnorm(0,.000001)
  b[2]~dnorm(0,.000001)
  tau~dgamma(.0001,.0001)

  # Summing Diagnostics Values
  chidev1.obs <- sum(dev1.obs[])
  chidev2.obs <- sum(dev2.obs[])

  chidev1.rep <- sum(dev1.rep[] )
  chidev2.rep <- sum(dev2.rep[] )

  chidev1.pval<-step(chidev1.obs-chidev1.rep)
  chidev2.pval<-step(chidev2.obs-chidev2.rep)

  #   Deviance statistic
  dev<-   -2*sum(loglike[])
  dev.rep <-  -2*sum(loglike.rep[])
  dev.pval<-step(dev-dev.rep)

}", file="HarvestModQ21.txt")


# Setup Parameters
bugMQ21.dat<-list("x", "y", "N")

initMQ21.fun<-function(){ list(b=runif(2,-.8,-.2), tau=runif(1,.2,.8), y.rep=rnorm(N))}

paramsMQ21<-c("b", "tau",          # the rest are for the model checking
              "mu", "res", "stdres", "res.rep", "stdres.rep", "p.smaller",
              "p.inv", "chidev1.pval", "chidev2.pval", "chidev1.obs", "chidev2.obs",
              "chidev1.rep", "chidev2.rep", "dev", "dev.rep", "dev.pval")


#### Could change the code below...
# Run Open Bugs -
set.seed(2157)
attach(HarvestingData)
HarvestingMQ21=bugs(bugMQ21.dat, initMQ21.fun, paramsMQ21, model.file="HarvestModQ21.txt",
    n.chains=NChain, n.iter=NSim, n.burnin=Burnin, n.thin=NThin)
detach(HarvestingData)


# Get Simulation from OpenBugs
```

```
SArrayMQ21= HarvestingMQ21$sims.array   # Data Arrays
vname=attr(SArrayMQ21,"dimnames")[3][[1]]  # Variable Names
MQ21.coda <- as.mcmc.list(HarvestingMQ21)


# Print Summary statistics of parameters oif Interest
RNames <- rownames(HarvestingMQ21[["summary"]][,c(1:3,5,7)]) # List of parameters
df_Prt <- data.frame(Parameter = RNames)
df_Prt <- cbind(df_Prt, as_tibble(HarvestingMQ21[["summary"]][,c(1:3,5,7)]))
df_Prt %>%
  filter(substr(Parameter,1,2)=="b[" |
           substr(Parameter,1,3)=="tau") %>%
  kbl(booktabs = TRUE, digits = 4,
      caption = "OpenBugs Summary - Harvesting Date vs. Yield (Model-1)") %>%
  kable_styling(latex_options = "striped")

# Plot TracePlots * b[1], b[2] and tau *

# Sampling 5000 points to generate "thinner" traceplots
set.seed(312)

L <- NSim-Burnin

#OldSz <- Sz
#Sz <- L       # Override Original value

S <- if (Sz < L) sort(sample(1:L, Sz, replace = FALSE)) else 1:Sz

# Build working dataframe
D_Workb1<- data.frame(ValCh1=SArrayMQ21[S,1,paste0("b[1]")],
                      ValCh2=SArrayMQ21[S,2,paste0("b[1]")],
                      ValCh3=SArrayMQ21[S,3,paste0("b[1]")])

D_Workb2 <- data.frame(ValCh1=SArrayMQ21[S,1,paste0("b[2]")],
                       ValCh2=SArrayMQ21[S,2,paste0("b[2]")],
                       ValCh3=SArrayMQ21[S,3,paste0("b[2]")])

D_Worktau <- data.frame(ValCh1=SArrayMQ21[S,1,paste0("tau")],
                        ValCh2=SArrayMQ21[S,2,paste0("tau")],
                        ValCh3=SArrayMQ21[S,3,paste0("tau")])

# Plot Traceplots for selected data-frames
P1 <- D_Workb1 %>%
        ggplot(aes(seq(from=1,to=Sz)))+
        geom_line(aes(y=ValCh1, colour=1), size=0.8)+
        geom_line(aes(y=ValCh2, colour=2), size=0.8)+
        geom_line(aes(y=ValCh3, colour=3), size=0.8)+
        labs(y = expression(beta[1])) +
        theme_bw()
P2 <- D_Workb2 %>%
        ggplot(aes(seq(from=1,to=Sz)))+
        geom_line(aes(y=ValCh1, colour=1), size=0.8)+
        geom_line(aes(y=ValCh2, colour=2), size=0.8)+
```

```
        geom_line(aes(y=ValCh3, colour=3), size=0.8)+
        labs(y = expression(beta[2])) +
        theme_bw()
P3 <- D_Worktau %>%
        ggplot(aes(seq(from=1,to=Sz)))+
        geom_line(aes(y=ValCh1, colour=1), size=0.8)+
        geom_line(aes(y=ValCh2, colour=2), size=0.8)+
        geom_line(aes(y=ValCh3, colour=3), size=0.8)+
        labs(y = expression(tau)) +
        theme_bw()

# Plot all Graphs in a same frame
ggpubr::ggarrange(P1+theme(axis.text.x=element_blank(),
                           axis.title.x=element_blank(),
                           legend.position="none"),
                  P2+theme(axis.text.x=element_blank(),
                           axis.title.x=element_blank(),
                           legend.position="none"),
                  P3+theme(axis.text.x=element_blank(),
                           axis.title.x=element_blank(),
                           legend.position="none"), ncol = 2, nrow = 2)

# Remove Working Variable to free memory
# Sz <- OldSz # Restore original value
remove(D_Workb1, D_Workb2, D_Worktau )

# Plot ACF Plots * b[1], b[2] and tau *

# Build working dataframe
D_Workb1 <- data.frame(ValCh1=SArrayMQ21[,1,paste0("b[1]")])
D_Workb2 <- data.frame(ValCh1=SArrayMQ21[,1,paste0("b[2]")])
D_Worktau <- data.frame(ValCh1=SArrayMQ21[,1,paste0("tau")])

# Plot ACF Plots for selected data-frames
P1 <- ggAcf(D_Workb1$ValCh1, lag.max = 50)+
  labs(x = "Lag", y = expression(beta[1])) +
  ggtitle(NULL)+
  theme_bw()
P2 <- ggAcf(D_Workb2$ValCh1, lag.max = 50)+
  labs(x = "Lag", y = expression(beta[2])) +
  ggtitle(NULL)+
  theme_bw()
P3 <- ggAcf(D_Worktau$ValCh1, lag.max = 50)+
  labs(x = "Lag", y = expression(tau)) +
  ggtitle(NULL)+
  theme_bw()

# Plot all Graphs in a same frame
ggpubr::ggarrange(P1+theme(axis.title.x=element_blank(),
                           legend.position="none"),
                  P2+theme(axis.title.x=element_blank(),
                           legend.position="none"),
                  P3+theme(axis.title.x=element_blank(),
```

```
                            legend.position="none"), ncol = 2, nrow = 2)
# Remove Working Variable to free memory
remove(D_Workb1, D_Workb2, D_Worktau )

# MODEL Q2_2
cat("### Model 2 -
model{
  for (i in 1:N) {
    y[i]~dnorm(mu[i],tau)
    mu[i]<- b[1] + b[2]*(x[i]-31)+ b[3]*pow((x[i]-31),2)

    # Get the residuals for the observed value
    res[i] <- (y[i]-mu[i])           # Estimate the residual for this model  - Item (1)
    stdres[i] <- res[i]*sqrt(tau)    # Calculate the standard residuals - Item (2)

    dev1.obs[i] <- pow(res[i],2)
    dev2.obs[i] <- pow(stdres[i],2)

    # Get a replicated Sample - sample of the predictive distribution
    y.rep[i]~dnorm(mu[i],tau)
    p.smaller[i] <- step(y[i]-y.rep[i])  # Check to see the probability of getting extreme value - Item

    # Residual and Moments replicated - this gives the predicted distribution for these values
    res.rep[i] <- y.rep[i]-mu[i]
    stdres.rep[i] <- res.rep[i]*sqrt(tau)

    dev1.rep[i] <- pow(res.rep[i],2)
    dev2.rep[i] <- pow(stdres.rep[i],2)

    # Likelihood for each observed and replicated data
    # note: Need to know the density function of the probability model

    loglike[i] <- (0.5)*log(tau/6.283185) + (-0.5)*tau*pow((y[i]-mu[i]),2)
    loglike.rep[i]<-  (0.5)*log(tau/6.283185) + (-0.5)*tau*pow((y.rep[i]-mu[i]),2)

    p.inv[i]<- 1/exp(loglike[i])      #  This is to find the predictive ordinate of the observations
  }

  # Prior definitions
  b[1]~dnorm(0,.000001)
  b[2]~dnorm(0,.000001)
  b[3]~dnorm(0,.01)
  tau~dgamma(.0001,.0001)

  # Summing Diagnostics Values
  chidev1.obs <- sum(dev1.obs[])
  chidev2.obs <- sum(dev2.obs[])

  chidev1.rep <- sum(dev1.rep[] )
  chidev2.rep <- sum(dev2.rep[] )

  chidev1.pval<-step(chidev1.obs-chidev1.rep)
  chidev2.pval<-step(chidev2.obs-chidev2.rep)
```

```
  #   Deviance statistic
  dev<-   -2*sum(loglike[])
  dev.rep <-  -2*sum(loglike.rep[])
  dev.pval<-step(dev-dev.rep)

}", file="HarvestModQ22.txt")

# Setup Parameters
paramsMQ22 <- c("b", "tau",          # the rest are for the model checking
              "mu", "res", "stdres", "res.rep", "stdres.rep", "p.smaller",
              "p.inv", "chidev1.pval", "chidev2.pval", "chidev1.obs", "chidev2.obs",
              "chidev1.rep", "chidev2.rep", "dev", "dev.rep", "dev.pval")

bugMQ22.dat=list("x", "y", "N")  # what variable you need in the model

# Setup Initial Values
initMQ22.fun=function(){
  list(b=runif(3,-.8,-.2), tau=runif(1,.2,.8), y.rep=rnorm(N))
}

#### Run Openbugs Simulation
set.seed(3508)
attach(HarvestingData)
HarvestingMQ22=bugs(bugMQ22.dat, initMQ22.fun, paramsMQ22, model.file="HarvestModQ22.txt",
    n.chains=NChain, n.iter=NSim, n.burnin=Burnin, n.thin=NThin , debug=FALSE)
detach(HarvestingData)


# Get Simulation from OpenBugs
SArrayMQ22= HarvestingMQ22$sims.array   # Data Arrays
vname=attr(SArrayMQ22,"dimnames")[3][[1]]  # Variable Names
MQ22.coda <- as.mcmc.list(HarvestingMQ22)


# Print Summary statistics of parameters oif Interest
RNames <- rownames(HarvestingMQ22[["summary"]][,c(1:3,5,7)]) # List of parameters
df_Prt <- data.frame(Parameter = RNames)
df_Prt <- cbind(df_Prt, as_tibble(HarvestingMQ22[["summary"]][,c(1:3,5,7)]))
df_Prt %>%
  filter(substr(Parameter,1,2)=="b[" |
           substr(Parameter,1,3)=="tau") %>%
  kbl(booktabs = TRUE, digits = 4,
      caption = "OpenBugs Summary - Harvesting Date vs. Yield (Model-2)") %>%
  kable_styling(latex_options = "striped")

# Plot TracePlots * b[1], b[2], b[3] and tau *

# Sampling 5000 points to generate "thinner" traceplots
set.seed(792)

L <- NSim-Burnin

S <- if (Sz < L) sort(sample(1:L, Sz, replace = FALSE)) else 1:Sz
```

```
# Build working dataframe
D_Workb1<- data.frame(ValCh1=SArrayMQ22[S,1,paste0("b[1]")],
                      ValCh2=SArrayMQ22[S,2,paste0("b[1]")],
                      ValCh3=SArrayMQ22[S,3,paste0("b[1]")])

D_Workb2 <- data.frame(ValCh1=SArrayMQ22[S,1,paste0("b[2]")],
                       ValCh2=SArrayMQ22[S,2,paste0("b[2]")],
                       ValCh3=SArrayMQ22[S,3,paste0("b[2]")])

D_Workb3 <- data.frame(ValCh1=SArrayMQ22[S,1,paste0("b[3]")],
                       ValCh2=SArrayMQ22[S,2,paste0("b[3]")],
                       ValCh3=SArrayMQ22[S,3,paste0("b[3]")])

D_Worktau <- data.frame(ValCh1=SArrayMQ22[S,1,paste0("tau")],
                        ValCh2=SArrayMQ22[S,2,paste0("tau")],
                        ValCh3=SArrayMQ22[S,3,paste0("tau")])

# Plot Traceplots for selected data-frames
P1 <- D_Workb1 %>%
        ggplot(aes(seq(from=1,to=Sz)))+
        geom_line(aes(y=ValCh1, colour=1), size=0.8)+
        geom_line(aes(y=ValCh2, colour=2), size=0.8)+
        geom_line(aes(y=ValCh3, colour=3), size=0.8)+
        labs(y = expression(beta[1])) +
        theme_bw()
P2 <- D_Workb2 %>%
        ggplot(aes(seq(from=1,to=Sz)))+
        geom_line(aes(y=ValCh1, colour=1), size=0.8)+
        geom_line(aes(y=ValCh2, colour=2), size=0.8)+
        geom_line(aes(y=ValCh3, colour=3), size=0.8)+
        labs(y = expression(beta[2])) +
        theme_bw()
P3 <- D_Workb3 %>%
        ggplot(aes(seq(from=1,to=Sz)))+
        geom_line(aes(y=ValCh1, colour=1), size=0.8)+
        geom_line(aes(y=ValCh2, colour=2), size=0.8)+
        geom_line(aes(y=ValCh3, colour=3), size=0.8)+
        labs(y = expression(beta[3])) +
        theme_bw()
P4 <- D_Worktau %>%
        ggplot(aes(seq(from=1,to=Sz)))+
        geom_line(aes(y=ValCh1, colour=1), size=0.8)+
        geom_line(aes(y=ValCh2, colour=2), size=0.8)+
        geom_line(aes(y=ValCh3, colour=3), size=0.8)+
        labs(y = expression(tau)) +
        theme_bw()

# Plot all Graphs in a same frame
ggpubr::ggarrange(P1+theme(axis.text.x=element_blank(),
                           axis.title.x=element_blank(),
                           legend.position="none"),
                  P2+theme(axis.text.x=element_blank(),
                           axis.title.x=element_blank(),
```

```
                        legend.position="none"),
                  P3+theme(axis.text.x=element_blank(),
                        axis.title.x=element_blank(),
                        legend.position="none"),
                  P4+theme(axis.text.x=element_blank(),
                        axis.title.x=element_blank(),
                        legend.position="none"), ncol = 2, nrow = 2)

# Remove Working Variable to free memory
remove(D_Workb1, D_Workb2, D_Workb3, D_Worktau )

# Plot ACF Plots * b[1], b[2], b[3] and tau *

# Build working dataframe
D_Workb1 <- data.frame(ValCh1=SArrayMQ22[,1,paste0("b[1]")])
D_Workb2 <- data.frame(ValCh1=SArrayMQ22[,1,paste0("b[2]")])
D_Workb3 <- data.frame(ValCh1=SArrayMQ22[,1,paste0("b[3]")])
D_Worktau <- data.frame(ValCh1=SArrayMQ22[,1,paste0("tau")])

# Plot ACF Plots for selected data-frames
P1 <- ggAcf(D_Workb1$ValCh1, lag.max = 50)+
  labs(x = "Lag", y = expression(beta[1])) +
  ggtitle(NULL)+
  theme_bw()
P2 <- ggAcf(D_Workb2$ValCh1, lag.max = 50)+
  labs(x = "Lag", y = expression(beta[2])) +
  ggtitle(NULL)+
  theme_bw()
P3 <- ggAcf(D_Workb3$ValCh1, lag.max = 50)+
  labs(x = "Lag", y = expression(beta[3])) +
  ggtitle(NULL)+
  theme_bw()
P4 <- ggAcf(D_Worktau$ValCh1, lag.max = 50)+
  labs(x = "Lag", y = expression(tau)) +
  ggtitle(NULL)+
  theme_bw()

# Plot all Graphs in a same frame
ggpubr::ggarrange(P1+theme(axis.title.x=element_blank(),
                        legend.position="none"),
                  P2+theme(axis.title.x=element_blank(),
                        legend.position="none"),
                  P3+theme(axis.title.x=element_blank(),
                        legend.position="none"),
                  P4+theme(axis.title.x=element_blank(),
                        legend.position="none"), ncol = 2, nrow = 2)
# Remove Working Variable to free memory
remove(D_Workb1, D_Workb2, D_Workb3, D_Worktau )

# Routines to Calculate DIC, Deviance of Model - Author: Michael Escobar (Thanks!)

# OBS>:- Flow remodeled to document each step
```

```
# Calculates the Deviance by hand
devNormFunc <- function(beta0, beta1, beta2, tau, x, y) {
  mu <- beta0 + beta1*x + beta2*x^2
  return(-2*sum(log(dnorm(y,mu,1/sqrt(tau)))))
}

CalcModelStats <- function (Model, TxtModel) {

  # STEP 1 - Calculate the Likelihood and respective Inverse
  M1.pinv <- Model$mean$p.inv       # Get 1/p(x)
  PLogLI <- -1*log(M1.pinv)                  # Get p(x)

  # Collects Inv.p(x) Likelihood, P[Log(Likelihood)]
  TbWork1 <- cbind(M1.pinv,1/M1.pinv,PLogLI)
  colnames(TbWork1) <- c("p.inv","p(x)","PLogLI");#TbWork1
  Pseudom2logL= -2*sum(PLogLI);#Pseudom2logL
  #cat("\n-------------------\n\n")

  # Collect Residuals (TbWork2)
  TbWork2<-cbind(Model$mean$res,
                 t(apply(Model$sims.list$res.rep,2,
                         function(x){c(quantile(x,probs=c(0.025,.975)),mean(x),sd(x))})))
  colnames(TbWork2)=c("res","2.5%","97.5%","mean","SD");#TbWork2
  #cat("\n-------------------\n\n")

  # Collect Std.Residuals (TbWork3)
  TbWork3<-cbind(Model$mean$stdres,
                 t(apply(Model$sims.list$stdres.rep,2,
                         function(x){c(quantile(x,probs=c(0.025,.975)),mean(x),sd(x))})))
  colnames(TbWork3)=c("stdres","2.5%","97.5%","mean","SD");#TbWork3
  #cat("\n-------------------\n\n")

  # Collect Deviance (TbWork4)
  M1.devrep<-Model$sims.list$dev.rep
  TbWork4<-c(Model$mean$dev,
             quantile(M1.devrep,probs=c(0.025,.975)),
             mean(M1.devrep),sd(M1.devrep))
  names(TbWork4)=c("Deviance","2.5%","97.5%","mean","SD");#TbWork4
  #cat("\n-------------------\n\n")

  # Collect Chi-Deviance (TbWork5)
  M1.chidev2rep<-Model$sims.list$chidev2.rep
  TbWork5<-c(Model$mean$chidev2.obs,
             quantile(M1.chidev2rep,probs=c(0.025,.975)),
             mean(M1.chidev2rep),sd(M1.chidev2rep))
  names(TbWork5)=c("Chi-Dev2","2.5%","97.5%","mean","SD");#TbWork5
  #cat("\n-------------------\n\n")

  # Calculates Non-Calibrated ("pval-stats") - REVIEW -
  #apply(Model$sims.list$p.smaller,2,mean)
  #Model$mean$chidev2.pval
  #Model$mean$dev.pval
  #cat("\n-------------------\n\n")
```

```
  # Comparing intrinsic and self calculated value for Deviance:
  M1.dev <-Model$sims.list$dev
  M1.deviance <-Model$sims.list$deviance
  TbWork6 <-rbind(
    quantile(M1.dev,probs=c(0.025,.25,.5,.75,.975)),
    quantile(M1.deviance,probs=c(0.025,.25,.5,.75,.975)) )
  rownames(TbWork6)=c("SelfPrograMed:","Openbugs Made:");# TbWork6
  # cat("\n--------------------\n\n")

  # c(Model$mean$deviance,Model$mean$dev)
  Dbar<-mean(M1.dev);#Dbar   # Calculated - Deviance
  pd2<-0.5*var(M1.dev);#pd2


  # Setup Parameters for Model
  beta0Bar<- Model$mean$b[1]
  beta1Bar<- Model$mean$b[2]
  beta2Bar<- ifelse (dim(Model$sims.list$b)[2]==3,Model$mean$b[3],0)
  tauBar <- Model$mean$tau
  Dhat <- devNormFunc(beta0Bar, beta1Bar, beta2Bar, tauBar,
                      HarvestingData$x,HarvestingData$y);#Dhat
  pd1<-Dbar-Dhat;#pd1

  DIC1<-Dbar+pd1; #DIC1
  DIC2<-Dbar+pd2; #DIC2


  cat("\n------- Goodness of Fit Statistics for",TxtModel," --------")
  cat("\nDeviance: \n");
  print(TbWork6, dig=4)
  cat("\nDbar: ",format(Dbar, digits = 2, nsmall = 4))
  # cat("\nDhat: ",format(Dhat, digits = 2, nsmall = 4))
  cat("\npD:  ",format(pd2, digits = 2, nsmall = 4))
  cat("\nDIC: ",format(DIC2, digits = 2, nsmall = 4))
  cat("\n-----------------------------------------------------------\n")
}


CalcModelStats(HarvestingMQ21,"Model-1")
CalcModelStats(HarvestingMQ22,"Model-2")

# Setup for Kuo-Mallick Model Variables - LAST VERSION (v12)
mu_y <- mean(HarvestingData$y)
sd_y <- sd(HarvestingData$y)
mu_x <- mean(HarvestingData$x)
sd_x <- sd(HarvestingData$x)
#sd_x2 <- sd(HarvestingData$x^2)
#mu_x2 <- mean(HarvestingData$x^2)
pp <- 0.5
tau0 <- 0.0625

# Standardizing Variables X and Y
sx <- (HarvestingData$x-mu_x)/sd_x
```

```
sy <- (HarvestingData$y-mu_y)/sd_y

cat("### Kuo-Mallick Model Selection
model{
  for (i in 1:N) {
    sy[i]~dnorm(mu[i],tau)
    mu[i]<- delta[1]*beta[1]*sx[i] + delta[2]*beta[2]*pow(sx[i],2)
  }

  for (ix in 1:2) {
    beta[ix]~dnorm(0,tau0)
  }

  tau~dgamma(.5,.01)

  # Prior Distribution of delta[]
  for(k in 1:2){
    delta[k]~dbern(pp)
  }

  # Model Deltas
  for(i1 in 1:2){
    for(i2 in 1:2){
      mod[i1,i2]<-equals((2-i1),delta[1])*equals((2-i2),delta[2])
     }
  }

}", file="Harvest_KM.txt")


dataDiag_KM<-list("sx", "sy", "mu_y", "sd_y", "N",
                  "mu_x", "sd_x", "pp", "tau0")

initsDiag_KM<-function(){ list(beta=rnorm(2), tau=runif(1,.5,1))}

paramDiag_KM<-c("beta", "tau", "delta", "mod")


set.seed(9212)
attach(HarvestingData)

HarvestDiag_KM<-bugs(dataDiag_KM,initsDiag_KM, paramDiag_KM, model.file="Harvest_KM.txt",
                        n.chains=1, n.iter=10000, n.burnin=1000,
                        n.thin=1)
detach(HarvestingData)


# Calculus of abeta[]'s
abeta1 <- HarvestDiag_KM$sims.array[,,paste0("beta[1]")]*sd_x/sd_y
abeta2 <- HarvestDiag_KM$sims.array[,,paste0("beta[2]")]*sd_x/sd_y

t_abeta1 <- c(mean(abeta1),sd(abeta1),quantile(abeta1,c(.025,.5,.975)))
t_abeta2 <- c(mean(abeta2),sd(abeta2),quantile(abeta2,c(.025,.5,.975)))
```

```
abeta <- rbind(t_abeta1,t_abeta2)
colnames(abeta) <- c("mean","sd","2.5%","50%","97.5%")
rownames(abeta) <- c("abeta[1]","abeta[2]")

# Print Summary statistics of parameters oif Interest
RNames <- rownames(HarvestDiag_KM[["summary"]][,c(1:3,5,7)]) # List of parameters
df_KM <- data.frame(Parameter = RNames)
df_KM <- cbind(df_KM, as_tibble(HarvestDiag_KM[["summary"]][,c(1:3,5,7)]))
df_KM %>%
  kbl(booktabs = TRUE, digits = 4,
      caption = "OpenBugs Summary - Kuo-Mallik Model Selection Statistics") %>%
  kable_styling(latex_options = "striped")

# Jeffrey's Criteria for Bayes Factor
JeffCrit <- function(BF) {
  if (BF>=0.0 && BF<=.5) return("Not worth more than a bare mention")
  else{
    if (BF>.5 && BF<=1.0) return("Substantial")
    else {
      if (BF>1.0 && BF<=2.0) return("Strong")
      else return("Decisive")
    }
  }
}

# Kass, Raftery's Criteria for Bayes Factor
KassRafteryCrit <- function(BF) {
  if (BF>=0.0 && BF<=2.0) return("Not worth more than a bare mention")
  else{
    if (BF>2.0 && BF<=6.0) return("Positive")
    else {
      if (BF>6.0 && BF<=10.0) return("Strong")
      else return("Decisive")
    }
  }
}

Kuo_MallickCrit <- function(PModA, PModB, TxtModA, TxtModB) {
  BF_KM <- PModA/PModB
  # Print Kuo-Mallik Bayes Factor Analysis for Model-A over Model-B
  cat("\n-------- Kuo-Mallik Bayes Factor Analysis (",TxtModA,"/",TxtModB,") ---------")
  cat("\n\nPosterior Probability",TxtModA,"| Data: ",format(PModA, digits = 2, nsmall = 4))
  cat("\n\nPosterior Probability",TxtModB,"| Data: ",format(PModB, digits = 2, nsmall = 4))
  cat("\n\n>>> Bayes Factor : ",format(BF_KM, digits = 2, nsmall = 4))
  cat("\n\n>>> Jeffrey's Evidence of",TxtModA,"over",TxtModB,": ",JeffCrit(log10(BF_KM)))
  cat("\n\n>>> Kass/Raftery's Evidence of",TxtModA,"over",TxtModB,": ",KassRafteryCrit(2*log(BF_KM)))
  cat("\n-----------------------------------------------------------------\n")
}


# Calculation of BF - It is equivalent to compare
#    mod[1,1] (Model-2) agains mod[1,2] (Model-1)
PM1 <- df_KM[which(df_KM$Parameter=="mod[1,2]"),"mean"]#;PM1  # This is Probability of Model-1
```

```
PM2 <- df_KM[which(df_KM$Parameter=="mod[1,1]"),"mean"]#;PM2  # This is Probability of Model-2

Kuo_MallickCrit (PM2, PM1, "M2", "M1")

# Item (1) - Print Residuals
RNames <- rownames(HarvestingMQ21[["summary"]][,c(1:3,5,7)]) # List of parameters
df_Prt <- data.frame(Parameter = RNames)
df_Prt <- cbind(df_Prt, as_tibble(HarvestingMQ21[["summary"]][,c(1:3,5,7)]))
df_Prt %>%
  filter(substr(Parameter,1,4)=="res[") %>%
  kbl(booktabs = TRUE, digits = 4,
      caption = "OpenBugs Summary - Residuals for Model-1") %>%
  kable_styling(latex_options = "striped")

# Item (2) - Print Standardized Residuals
RNames <- rownames(HarvestingMQ21[["summary"]][,c(1:3,5,7)]) # List of parameters
df_Prt <- data.frame(Parameter = RNames)
df_Prt <- cbind(df_Prt, as_tibble(HarvestingMQ21[["summary"]][,c(1:3,5,7)]))
df_Prt %>%
  filter(substr(Parameter,1,7)=="stdres[") %>%
  kbl(booktabs = TRUE, digits = 4,
      caption = "OpenBugs Summary - Standardized Residuals for Model-1") %>%
  kable_styling(latex_options = "striped")

# Item (3) - Print Chance of Getting Extreme Observations
RNames <- rownames(HarvestingMQ21[["summary"]][,c(1:2)]) # List of parameters
df_Prt <- data.frame(Parameter = RNames)
df_Prt <- cbind(df_Prt, as_tibble(HarvestingMQ21[["summary"]][,c(1:2)]))
df_Prt %>%
  filter(substr(Parameter,1,10)=="p.smaller[") %>%
  kbl(booktabs = TRUE, digits = 4,
      caption = "OpenBugs Summary - Chance of Extreme Observations for Model-1") %>%
  kable_styling(latex_options = "striped")

# Get Distribution of Residuals/Std Residuals under Predictive Distribution
SArrayDiagM1= HarvestingMQ21$sims.array      # Data Arrays
vname=attr(SArrayDiagM1,"dimnames")[3][[1]]  # Variable Names


# Plot Sample densities to Compare Distributions of * SIGMA, BETA0, BETA.SMOKE and BETA.OBESITY *

# Build working dataframe
D_WorkRes <- as_tibble(data.frame(LvlStd=as.vector(
  SArrayDiagM1[,1,c(paste0("res.rep[", 1:16, "]"))]), Chain = "C1"))
D_WorkstdRes <- as_tibble(data.frame(LvlStd=as.vector(
  SArrayDiagM1[,1,c(paste0("stdres.rep[", 1:16, "]"))]), Chain = "C1"))

# Ploting the Sensities
P1 <- D_WorkRes %>%
  ggplot(aes(x = LvlStd))+
  geom_density(aes(x=LvlStd, group=Chain, colour=Chain), size=0.8)+
  labs(x = "Residuals", y = "Density") +
  scale_color_manual(values = cbp2)+
```

```
  theme_bw()
P2 <- D_WorkstdRes %>%
  ggplot(aes(x = LvlStd))+
  geom_density(aes(x=LvlStd, group=Chain, colour=Chain), size=0.8)+
  labs(x = "Std.Residuals", y = "Density") +
  scale_color_manual(values = cbp2)+
  theme_bw()

ggpubr::ggarrange(P1+theme(legend.position="none"),
                  P2+theme(legend.position="none"), ncol = 1, nrow = 2)
remove(D_WorkRes,D_WorkstdRes)

# Print Probabilities of each variable present in the final model
df_Prt <- data.frame(Var = c("X", "X**2"),
                     Prob = c( df_KM[which(df_KM$Parameter=="delta[1]"),"mean"],
                               df_KM[which(df_KM$Parameter=="delta[2]"),"mean"]))
df_Prt %>%
  kbl(booktabs = TRUE, digits = 4,
      caption = "Probability of Variables in model") %>%
  kable_styling(latex_options = "striped")

# Calculation of BF - It is equivalent to compare
#    mod[2,1] (Model-2Q) agains mod[1,2] (Model-1)
PM2Q <- df_KM[which(df_KM$Parameter=="mod[2,1]"),"mean"]#;PM2  # This is Probability of Model-2

Kuo_MallickCrit (PM2Q, PM1, "M2-Q", "M1")
```