CHL5223 - Applied Bayesian Methods - Assignment 2

Luis Correia - Student No. 1006508566

February 22nd 2020

Question 1

In the week 4 lecture, we looked at a regression example which predicted the log(brain weight) from the log(body weight) of different species of animals. In the lecture notes, an OpenBugs was used to create the sampling algorithm and then produce the sampling. In this homework, you are asked to create an MCMC simulations algorithm using basic R programming and the Gibbs sampler. The data are given in the file: `brnbdy.csv`. First, the probabilistic model and the conditional distributions are given, then the questions to be answered are given. Let $Y = log(\text{brain weight})$ and $X = log(\text{body weight})$ The probabilistic model for this problems is the following.

$$Y_i|\alpha, \beta, X_i, \tau \sim \mathcal{N}(Y_i|\alpha + \beta X_i, \tau)$$
$$\alpha|\mu_\alpha, \tau_\alpha \sim \mathcal{N}(\alpha|\mu_\alpha, \tau_\alpha)$$
$$\beta|\mu_\beta, \tau_\beta \sim \mathcal{N}(\beta|\mu_\beta, \tau_\beta)$$
$$\tau|a_\tau, b_\tau \sim Gamma(a_\tau, b_\tau)$$

To finish off the model, assign the following values for the parameter variables: $\mu_\alpha = 0 = \mu_\beta$,$\tau_\alpha = 0.0001 = \tau_\beta$ and $a_\tau = 0.0001 = b_\tau$.

To construct the Gibbs sampler, you need to know the conditional distribution given all the other parameters in the model. (Note: I will only list the parameters that are needed in the below.) The conditional distribution for α given everything else is as follows:

$$\alpha|X_{i=1:n}, Y_{i=1:n}, \beta, \tau, \mu_\alpha, \tau_\alpha \sim \mathcal{N}(\mu_\alpha^*, \tau_\alpha^*)$$

with

$$\mu_\alpha^* = \frac{\tau \sum_{i=1}^n (Y_i - \beta X_i) + \tau_\alpha \mu_\alpha}{n\tau + \tau_\alpha}$$
$$\tau_\alpha^* = n\tau + \tau_\alpha$$

The conditional distribution of β given everything else is as follows:

$$\beta|X_{i=1:n}, Y_{i=1:n}, \alpha, \tau, \mu_\beta, \tau_\beta \sim \mathcal{N}(\mu_\beta^*, \tau_\beta^*)$$

with

$$\mu_\beta^* = \frac{\tau \sum_{i=1}^n [X_i(Y_i - \alpha)] + \tau_\beta \mu_\beta}{\tau \sum_{i=1}^n X_i^2 + \tau_\beta}$$

$$\tau_\beta^* = \tau \sum_{i=1}^{n} X_i^2 + \tau_\beta$$

The conditional distribution of τ given everything else is as follows:

$$\tau | X_{i=1:n}, Y_{i=1:n}, \alpha, \beta, a_\tau, b_\tau \sim Gamma(a_\tau^*, b_\tau^*)$$

with

$$a_\tau^* = a_\tau + n/2$$

$$b_\tau^* = b_\tau + \frac{1}{2} \sum_{i=1}^{n} [Y_i - (\alpha + \beta X_i)]^2$$

Do the following:

item (a)

Use R to run a Gibbs sampler algorithm, a Markov chain Monte Carlo algorithm, on this data. (DO NOT USE WinBugs, Openbugs, Jags or any of their relatives. The purpose is for you to do some basic programming.) You only need to run the MCMC for about 20,000 iterations for this example. Note, that for full credit, you need to submit your code as part of the homework.

{*Solution.*}

For this question we will use the regression problem from week 4, where `log(Brain Weight)` (in grams) can be represented as a linear model from `log(Body Weight)` (in Kg).

From our data-set, it's easy to verify that the relationship we are about to model has linear characteristics as we can see from graph below:

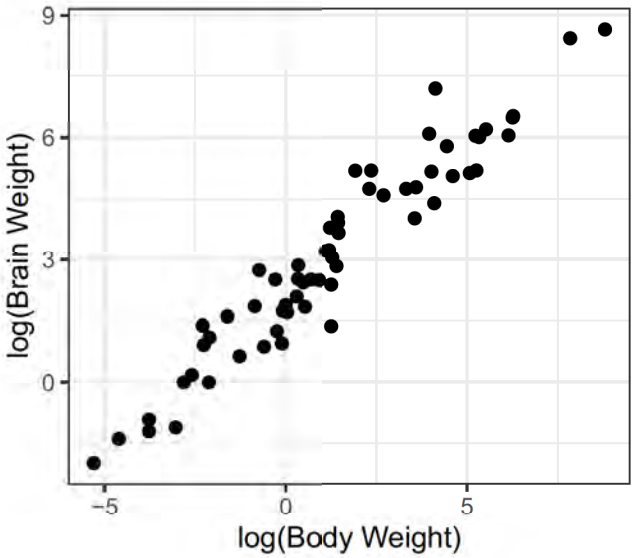

Figure 1: Scatter-plot - log(Brain Weight) vs. log(Body Weight)

The Gibbs Sampling Algorithm

The *pseudo-code* of Gibbs Sampling algorithm applied to our problem can be summarized as follows:

set $N = 20,000$;
set $a_\tau = b_\tau = 0.0001$;
set $\alpha^{(1)} = \beta^{(1)} = 0$, $\tau^{(1)} = a_r/b_r$;
for $i = 2$ *to* N **do**
  Generate $\alpha^{(i)}$;
  begin
    Calculate $\tau_\alpha^{(i)} = n\tau^{(i-1)} + \tau_\alpha$;
    Calculate $\mu_\alpha^{(i)} = \dfrac{\tau^{(i-1)} \sum_{j=1}^{n} (Y_j - \beta^{(i-1)} X_j) + \tau_\alpha \mu_\alpha}{\tau_\alpha^{(i)}}$;
    Sample $\alpha^{(i)}$ from $\mathcal{N}(\mu_\alpha^{(i)}, \tau_\alpha^{(i)})$;
  end
  Generate $\beta^{(i)}$;
  begin
    Calculate $\tau_\beta^{(i)} = \tau^{(i-1)} \sum_{j=1}^{n} X_j^2 + \tau_\beta$;
    Calculate $\mu_\beta^{(i)} = \dfrac{\tau^{(i-1)} \sum_{j=1}^{n} [X_j (Y_j - \alpha^{(i)})] + \tau_\beta \mu_\beta}{\tau_\beta^{(i)}}$;
    Sample $\beta^{(i)}$ from $\mathcal{N}(\mu_\beta^{(i)}, \tau_\beta^{(i)})$;
  end
  Generate $\tau^{(i)}$;
  begin
    Calculate $a_\tau^{(i)} = a_\tau + n/2$;
    Calculate $b_\tau^{(i)} = b_\tau + \frac{1}{2} \sum_{j=1}^{n} [Y_j - (\alpha^{(i)} + \beta^{(i)} X_j)]^2$;
    Sample $\tau^{(i)}$ from $\Gamma(a_\tau^{(i)}, b_\tau^{(i)})$;
  end
end

Algorithm 1: Gibbs Sampling Algorithm

The corresponding `R-Code` is as follows:

```
# Setup Initial Variables
Mu_alpha <- Mu_beta <- 0
Tau_alpha <- Tau_beta <- 0.0001
a_Tau<- b_Tau <- 0.0001  # v.16

# Setup Iteration Variables
N <- 20000

alpha <- beta <- tau <- rep(N,0)

# Setup \theta^{(0)}
alpha[1] <- beta[1] <- 0
tau[1] <- 1

# Calculates the value of parameters, given everything else
sample_Alpha <- function (X, Y, beta, tau, mu_alpha, tau_alpha) {
  tau_alphaStar <- length(Y)*tau+tau_alpha
  mu_alphaStar <- (tau*sum(Y-beta*X)+tau_alpha*mu_alpha)/tau_alphaStar
```

```
  alpha <- rnorm(1,mean=mu_alphaStar, sd=sqrt(1/tau_alphaStar))
  return(alpha)
}

sample_Beta <- function (X, Y, alpha, tau, mu_beta, tau_beta) {
  tau_betaStar<- tau*sum(X^2)+tau_beta
  mu_betaStar <- (tau*sum(X*(Y-alpha))+tau_beta*mu_beta)/tau_betaStar
  beta <- rnorm(1,mean=mu_betaStar, sd=sqrt(1/tau_betaStar))
  return(beta)
}

sample_Tau <- function (X, Y, alpha, beta, a_r, b_r) {
  a_rStar <- a_r+length(Y)/2
  b_rStar <- b_r+0.5*sum((Y-(alpha+beta*X))^2)
  tau <- rgamma(1,shape=a_rStar, rate=b_rStar)
  return(tau)
}

Gibbs <- function(X, Y, alpha, beta, tau, N) {
  for (i in 2:N) {
    alpha[i] <- sample_Alpha(X, Y, beta[i-1], tau[i-1], Mu_alpha, Tau_alpha)
    beta[i] <- sample_Beta(X, Y, alpha[i], tau[i-1], Mu_beta, Tau_beta)
    tau[i] <- sample_Tau(X, Y, alpha[i], beta[i], a_Tau, b_Tau)   # v.17
  }
  return(list(Alpha = alpha, Beta = beta, Tau = tau))
}

# Set Seed for reproducibility
set.seed(737)

#Calculates 3 chains
ThetaC1 <- Gibbs(X, Y, alpha, beta, tau, N)
ThetaC2 <- Gibbs(X, Y, alpha, beta, tau, N)
ThetaC3 <- Gibbs(X, Y, alpha, beta, tau, N)
```

After running the algorithm we will have obtained 03 chains of 20,000 samples for posterior distributions of the parameters α (`Alpha[1:20000]`), β (`Beta[1:20000]`) and τ (`Tau[1:20000]`) as requested.

For this question we also plotted the Auto-Correlation of estimates to verify its independence and set the *burn-in* parameter to discard part of samples which may have some degree of interdependence.

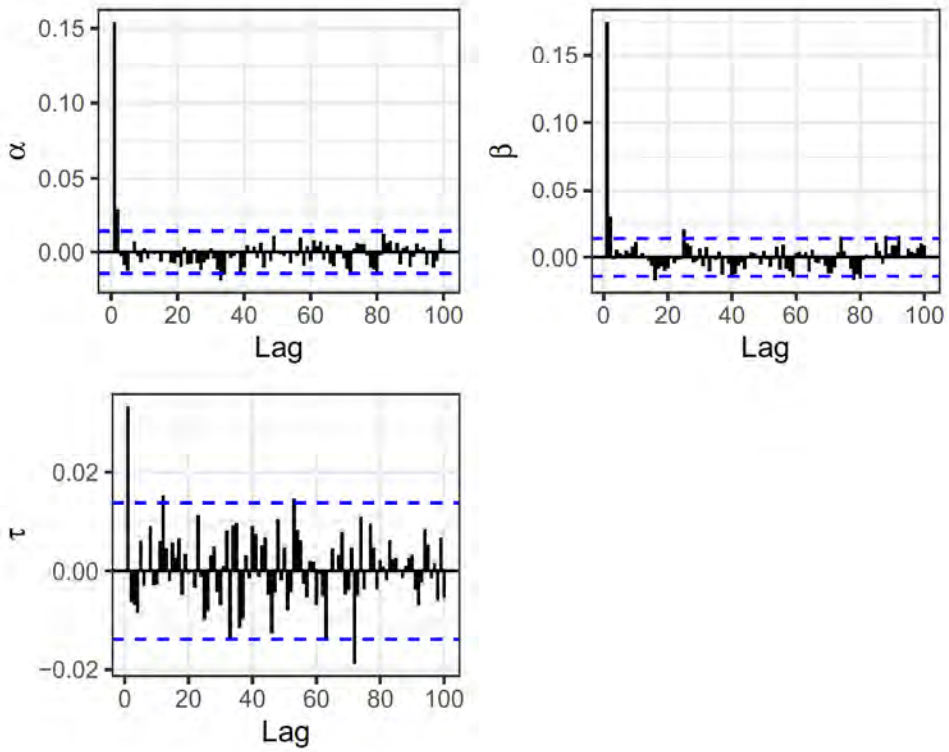

Figure 2: ACF plots - α, β and τ

The Auto-correlation plots (ACF) of the parameters indicate they are not correlated from the very initial lag's, so we can discard few estimates. For security, we will set the *burn-in* parameter to 100, which means we will discard the first 100 estimates.

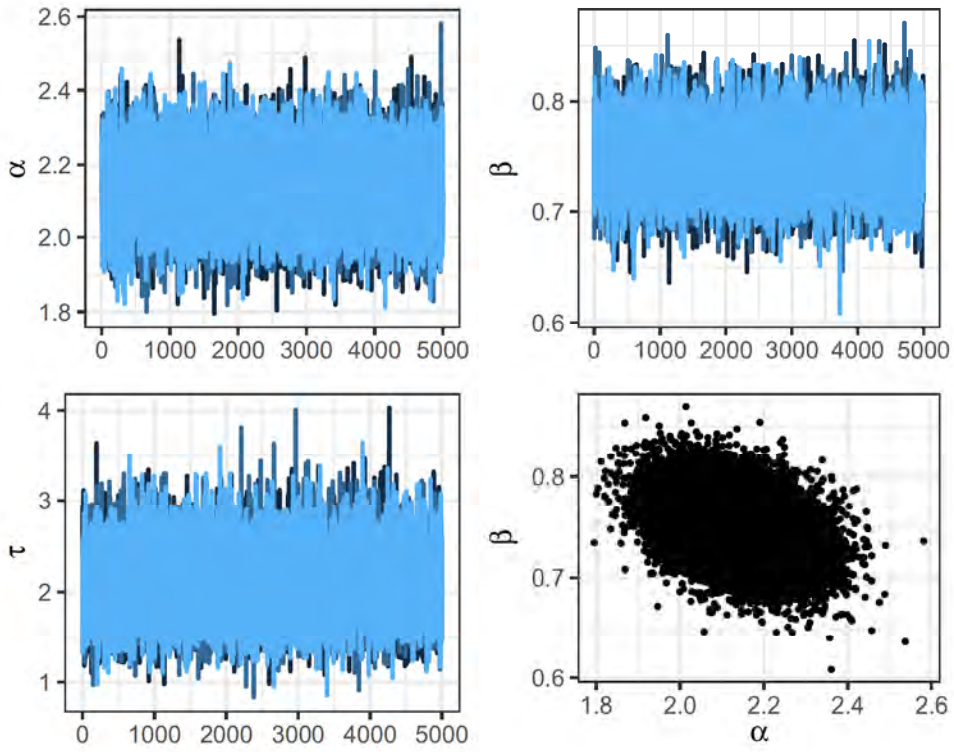

Figure 3: Traceplots of α, β and τ and Scatterplot α vs. β

After discarding the burn-in elements, we can see from trace-plots[1] that all chains are well mixed indicating the convergence of our estimates for our model's parameters.

We can also see from the joint distribution of α and β that there is no evidence that parameters are correlated, we can move forward with our estimation for the posterior distributions.

item (b)

Please give the posterior distribution (a plot and the summary statistics for the posterior distribution) for the parameters α, β, and τ. For the summary statistics of these posterior distributions it is sufficient to give the mean, standard deviation, and some type of 95% credible region.

{*Solution.*}

Using the samples generated by our algorithm for each parameter, we obtained the following graphs for posterior distributions of α, β and τ.

[1] In order to avoid too many points on each graph, affecting the display performance of PDF file, we limited the number of points to $5,000$ which is a good representation of parameter's convergence.

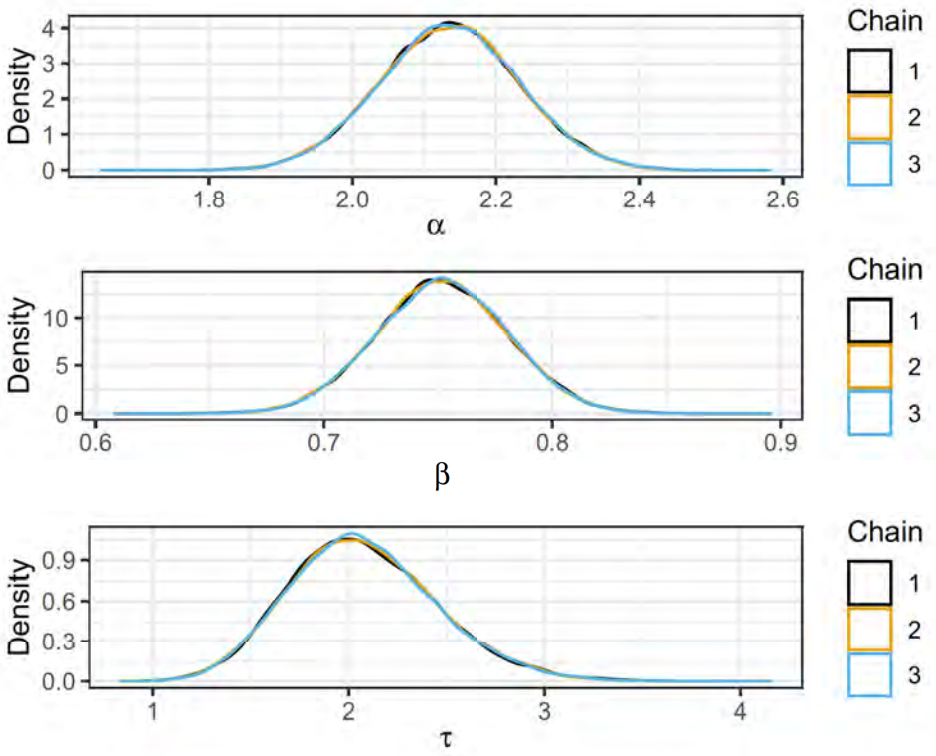

Figure 4: Posterior Distributions α, β and τ

... and the respective *Summary Statistics* from our posterior distributions are as follows:

```
##
## ---- Summary Statistics for  alpha  -----
## Mean alpha :  2.135
## Std.Dev. alpha :  0.0976
## 95% Credible Region for alpha ( 1.944 , 2.326 )

##
## ---- Summary Statistics for  beta  -----
## Mean beta :  0.752
## Std.Dev. beta :  0.0288
## 95% Credible Region for beta ( 0.695 , 0.8081 )

##
## ---- Summary Statistics for  tau  -----
## Mean tau :  2.076
## Std.Dev. tau :  0.3806
## 95% Credible Region for tau ( 1.330 , 2.822 )
```

item (c)

For an animal species with an average body weight of 55, provide the distribution which represents your belief of the brain weight. (Note, one might consider two different values here. Basically, I want the estimate of brain weight as a function of ($\{\alpha, \beta$, body weight $= 55\}$). Since this is a function of the random variables α and β, the estimate is also a random variable.) Provide the distribution by giving an estimate of the density

(a plot) and summary statistics of the distribution (mean, standard deviation, and credible interval).

{*Solution.*}

For an animal with body weight of 55 Kg, we will use our samples of α's and β's with $X = \texttt{log}(55)$ to generate samples of this animal.

By doing so, the resulting distribution that represents our belief of brain weight for such animal is as follows:

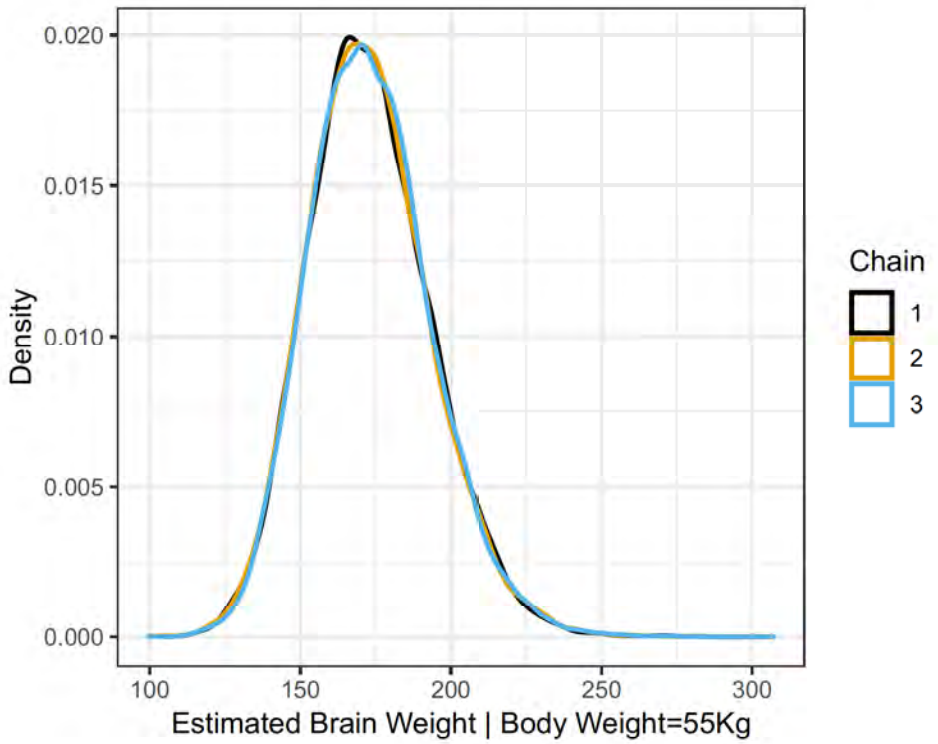

Figure 5: Posterior Distribution of Brain Weight of an animal with 55 Kg

```
##
## ---- Summary Statistics for Brain Weight | Body Weight = 55kg -----

##
## Mean :  173.122

##
## Std.Dev. :  20.5173

##
## 95% Credible Region ( 132.909 , 213.336 )

##
## >>> From this model, our belief of Brain Weight for an animal with

##     average Body Weight of 55Kg is 173.12 grams, with a 95% credible

##     region of ( 132.91 , 213.34 ) grams.
```

Question 2

The following table gives the number of lung cancers for different age groups and different histories of smoking. Also, given is the number of person years where a potential cases was exposed (was eligible to be identified by the study as having lung cancer) and was in a particular age group-smoker class. (Data is from page 495, Selvin, *Practical Biostatistical Methods*, 1995, (Belmont, CA: Wadsworth).) Do the following with this data.

item (a)

Do a Bayesian Poisson regression on this data. Model the logarithm of the rate of deaths (per person-years) as a linear function of age and smoking category. Model both the smoking category and the age groupings as discrete variables. Please give the `WinBugs` code for this data. Provide the uni-variate posterior distributions of each level of smoking and age and also give the posterior distribution of the precision used in your model. (If you wish, you may report the "standard deviations" instead of the precision.)

{*Solution.*}

The data-set used for this question is listed below.

Table 1: Deaths by Lung Cancer, per Age, Smoking Level and Person-Years

Smoking	Age	Deaths	PersonYears
1	1	11	114616
2	1	6	58259
3	1	4	19482
4	1	5	12947
1	2	20	101015
2	2	13	64836
3	2	2	11641
4	2	6	7450
1	3	31	78405
2	3	19	48952
3	3	4	8295
4	3	11	6823
1	4	40	49216
2	4	17	30296
3	4	8	5031
4	4	16	4024
1	5	93	58269
2	5	50	31854
3	5	20	6401
4	5	31	5032

For this problem, lets define $X_i :=$ Number of deaths by lung cancer and $p_i :=$ Person-Years, then our model can be represented as a Poisson Regression, as follows:

$$\begin{aligned}
\mathbb{E}(X_i) &= \big(\text{Death-rate}\big) \times \big(\text{Person-Years}\big) \\
\implies \log\big(X_i\big) &= \log\big(\text{Death-Rate}\big) + \log\big(p_i\big) \\
&= \Big(\beta_0 + \sum_{j=1}^{Q_a}\sum_{k=1}^{Q_s} \beta_{jk} X_{i,j,k}\Big) + \log\big(p_i\big)
\end{aligned}$$

with $Q_a = 4$ is the number of categories of *Age* and $Q_s = 5$ is the number of categories of *Smoking*

We can configure this model in `OpenBugs` with the following code:

```
model{
for(i in 1:NSmok)
{
#Age   Smoking   Deaths  PersonYears

Deaths[i]~dpois(lambda[i])
log(lambda[i]) <-  log(PersonYears[i]) + beta0 + beta.Age[Age[i]]+ beta.Smoking[Smoking[i]] + bias[i]
bias[i] ~dnorm(0,tau)
bias.adj[i] <- bias[i] - mean(bias[])
}

# Loop for Age
for(ia in 1:NCAge){
beta.Age[ia]~dnorm(0,tau.Age)
beta.Age.adj[ia] <- beta.Age[ia] - mean(beta.Age[])
}

# Loop for Smoking Level
for(is in 1:NCSmok){
beta.Smoking[is]~dnorm(0,tau.Smoking)
beta.Smoking.adj[is] <- beta.Smoking[is] - mean(beta.Smoking[])
}

# Initial Priors - Verify
beta0 ~ dnorm(0, 0.008)   # Overall rate of disease
beta0.adj <- beta0 + mean(bias[]) + mean(beta.Age[])+ mean(beta.Smoking[])

# Initial Priors - Verify
std ~ dunif(0, 4)
tau <- 1/pow(std,2)

# Initial Priors - Verify
std.Age ~dunif(0, 4)
tau.Age <- 1/pow(std.Age,2)
std.Smoking ~ dunif(0,4)
tau.Smoking <- 1/pow(std.Smoking,2)
}
```

After simulating $N = 30,000$ samples with *burn-in* of $10,000$ samples, with 5 chains the mean, standard-deviation and respective credible regions for model's parameters are listed in the following Report Summary.

Table 2: OpenBugs Summary - Deaths from Lung Cancer

Parameter	mean	sd	2.5%	50%	97.5%
beta.Age.adj[1]	-1.3400694	1.714784e-01	-1.6860000	-1.336500	-1.0139750
beta.Age.adj[2]	-0.7202229	1.458930e-01	-1.0150000	-0.717700	-0.4418000
beta.Age.adj[3]	-0.0239507	1.264731e-01	-0.2758000	-0.022585	0.2201000
beta.Age.adj[4]	0.6800478	1.199847e-01	0.4414000	0.681000	0.9130000
beta.Age.adj[5]	1.4041981	9.993130e-02	1.2070000	1.404000	1.6020000
std.Age	1.5573867	6.758871e-01	0.6996000	1.389000	3.3560000
beta.Smoking.adj[1]	-0.4585647	9.190310e-02	-0.6362025	-0.459650	-0.2754975
beta.Smoking.adj[2]	-0.5319693	1.028647e-01	-0.7338000	-0.532200	-0.3290000
beta.Smoking.adj[3]	0.0676280	1.375845e-01	-0.2137000	0.071110	0.3271000
beta.Smoking.adj[4]	0.9229055	1.168177e-01	0.6894000	0.924400	1.1480000
std.Smoking	1.1691879	6.918399e-01	0.4137975	0.958200	3.1530000
beta0.adj	-7.3734785	7.006750e-02	-7.5140000	-7.372000	-7.2390000
std	0.0863584	7.197330e-02	0.0044990	0.068675	0.2681000
tau	5110.5666380	2.439154e+04	13.9100000	212.049994	49410.0000000
tau.Age	0.6613994	5.262029e-01	0.0888097	0.518600	2.0430000
tau.Smoking	1.5748074	1.571033e+00	0.1006000	1.089000	5.8410250
deviance	102.1180091	4.304102e+00	95.6600000	101.500000	112.3000000

Posterior Distribution of β_0

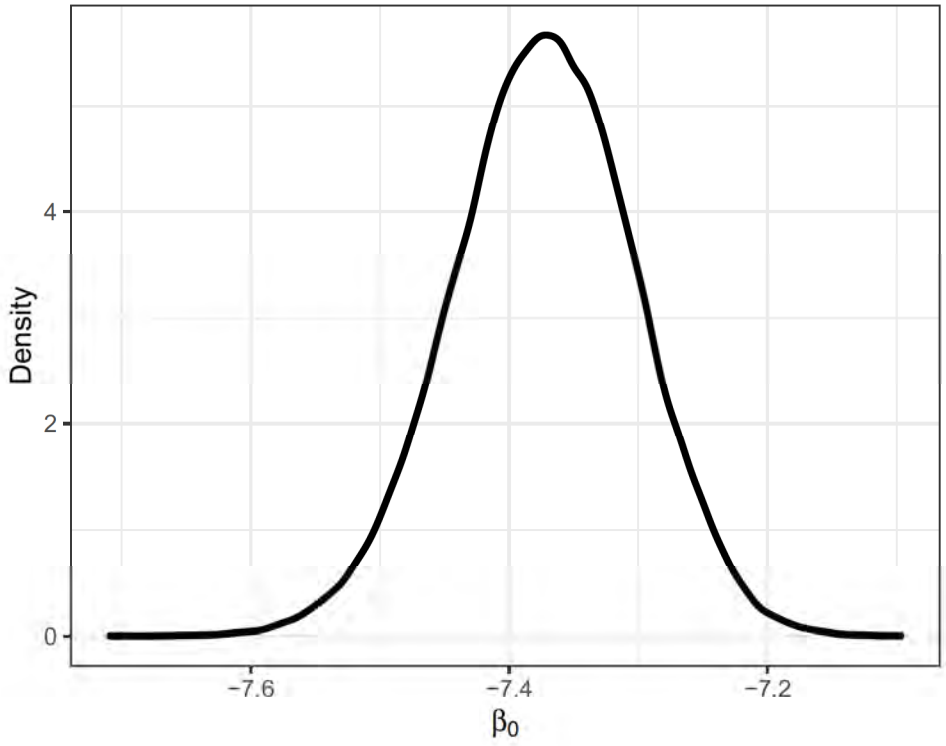

Figure 6: Posterior Distributions of β_0 (adjusted)

Comments:- The parameter β_0 on our model would represents the prediction of someone die from lung cancer with simultaneously both categories of *Age* and *Smoke* equals null. Intuitively, this parameter could then represent the inherent propensity of any individual dying from lung cancer which certainly would contribute for the death-rate of our model. On the other hand, *mathematically thinking* over our model, this individual would also be out of scope of our study since he/she doesn't fit any of the categories of Age and Smoking defined. For this reason we will not try to interpret the constant term as suggested by [2].

Posterior Distribution of each level of Age

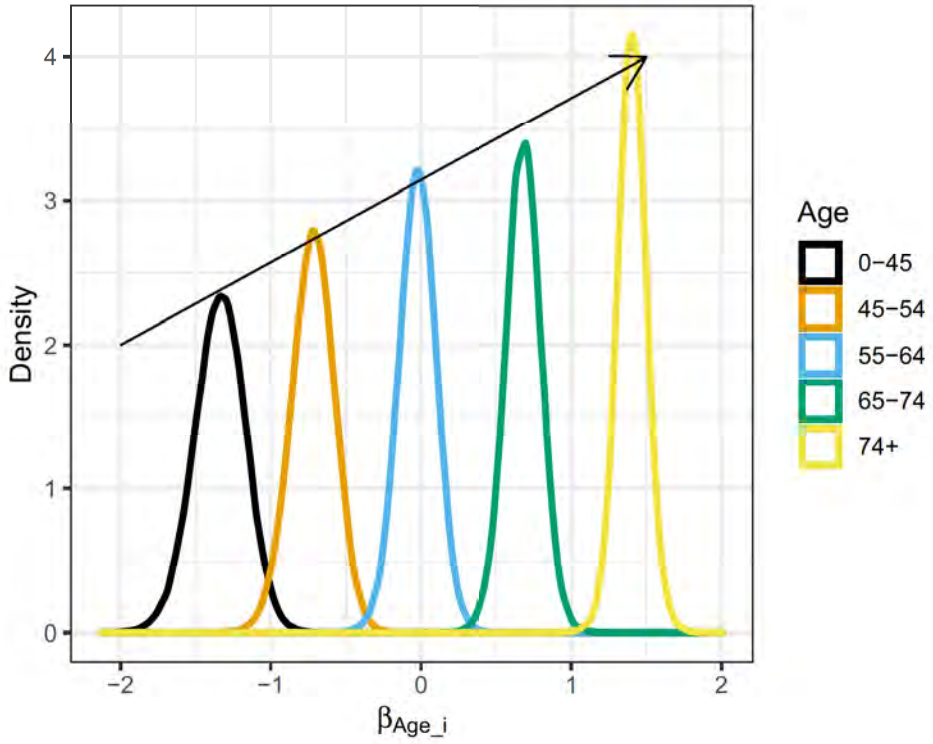

Figure 7: Posterior Distributions of β_{Age} for levels 1-5 (adjusted)

Comments:- We can see a *increasing pattern* on the posterior distributions of β_{Age} depending on age range. For younger people, the contribution of this parameter for death rate by lung cancer is less than 1 (i.e, for negative values for β's) and increasing from age range 55+ years old on. This may suggest that people who are in in smaller age-range have less probability of dying by lung cancer than people who are on greater age-ranges.

Posterior Distribution of each level of Smoking

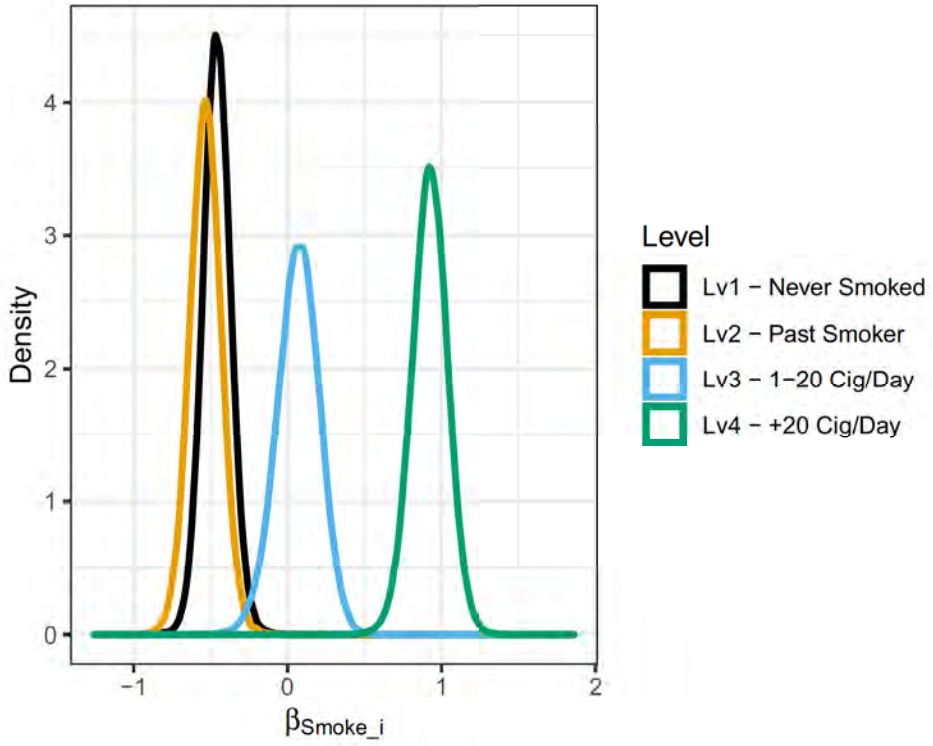

Figure 8: Posterior Distributions of $\beta_{Smoking}$ for levels 1-4 (adjusted)

Comments:- From this comparative graph, we can see a clear pattern on the posterior distributions depending of level of smoking. For Levels 1 and 2 (Never or Past Smoker) the estimations of β_{Smoke} are negative, implying that for those groups, the contribution of this parameter to death rate by lung cancer is less than 1. This may suggest that people who "never smoked" or are a "past smoker" have less probability of dying of lung cancer than people who are currently smoker. The reverse conclusion can also be drawn from people who are smokers, i.e. their smoking habit contributes to increase the probability of death by lung cancer.

Posterior Distribution of Standard Deviations / Precision of our model

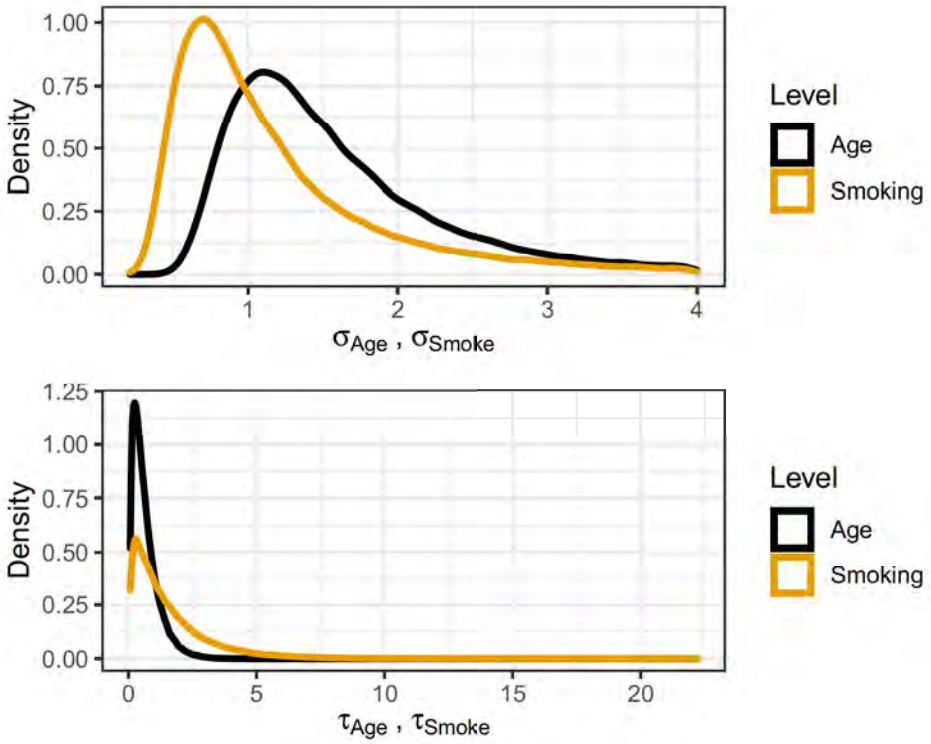

Figure 9: Posterior Distributions of Std. Deviation/Precision

item (b)

Provide a brief justification that your model has converged and that you have *burned-in* your simulation enough. (Note: you may give *thinned* plots of your simulations to avoid printing out plots which are suppose to represent several thousands or hundreds of thousand points. For the purpose of this question, you can provide simple diagnostics like trace/history plots and auto-correlation plots.)

{*Solution.*}

For this question we will present *thinned* trace and ACF plots of parameters estimates to demonstrate its *convergence* and that they are *uncorrelated*.

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 identical for all chains.

Please also note that I 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 $lag = 0$.

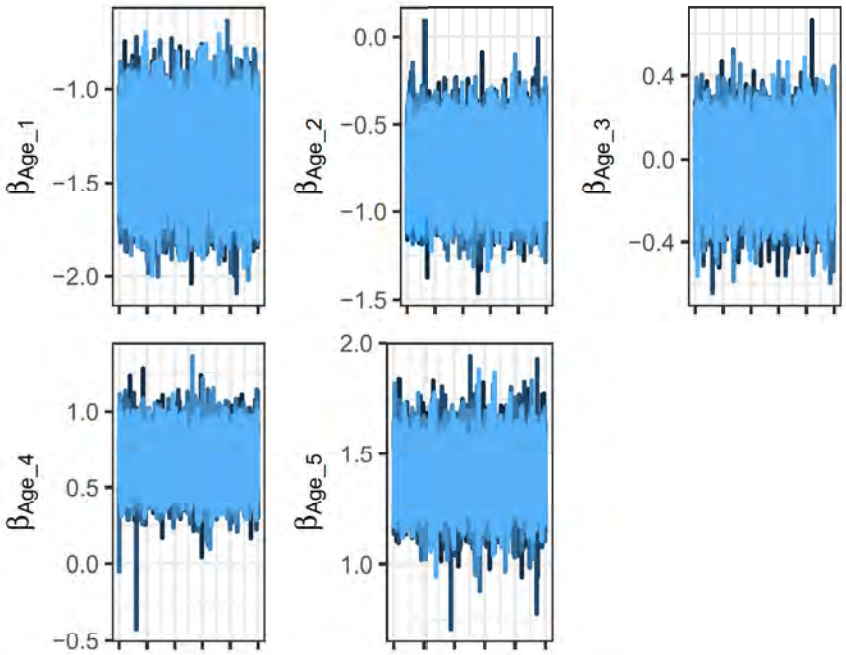

Figure 10: Trace-plots of β_{Age} (adjusted)

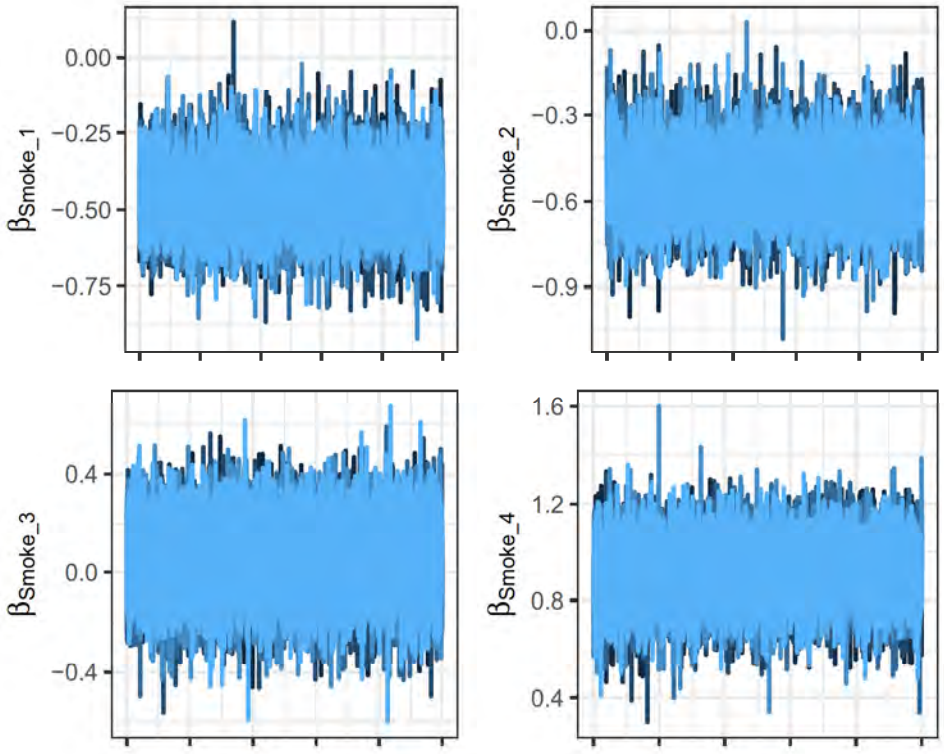

Figure 11: Trace-plots of $\beta_{Smoking}$ (adjusted)

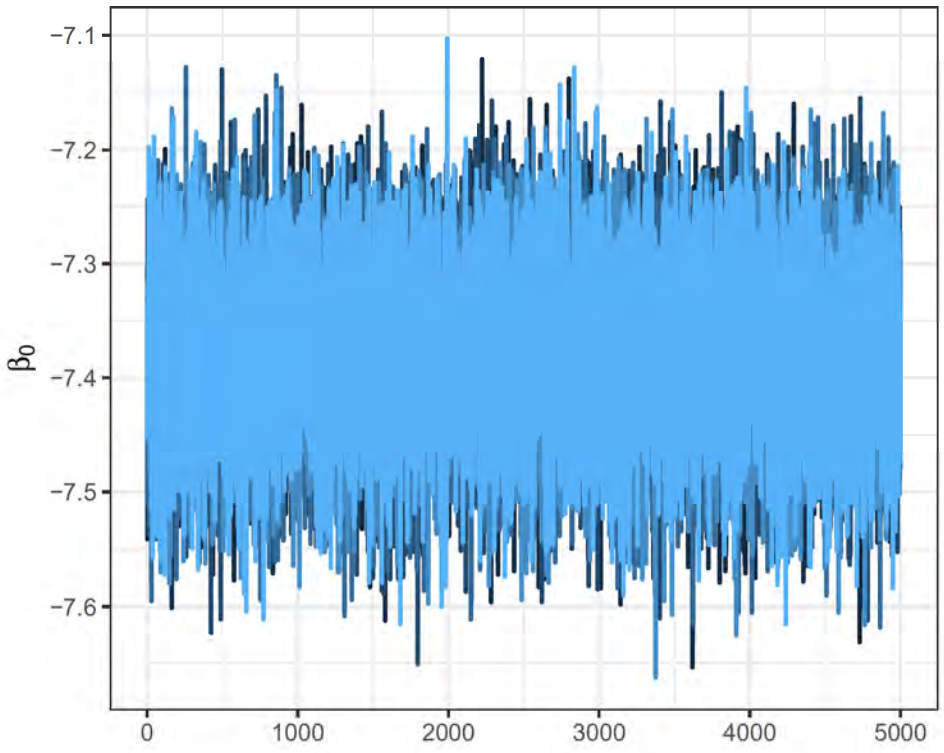

Figure 12: Trace-plots of β_0 (adjusted)

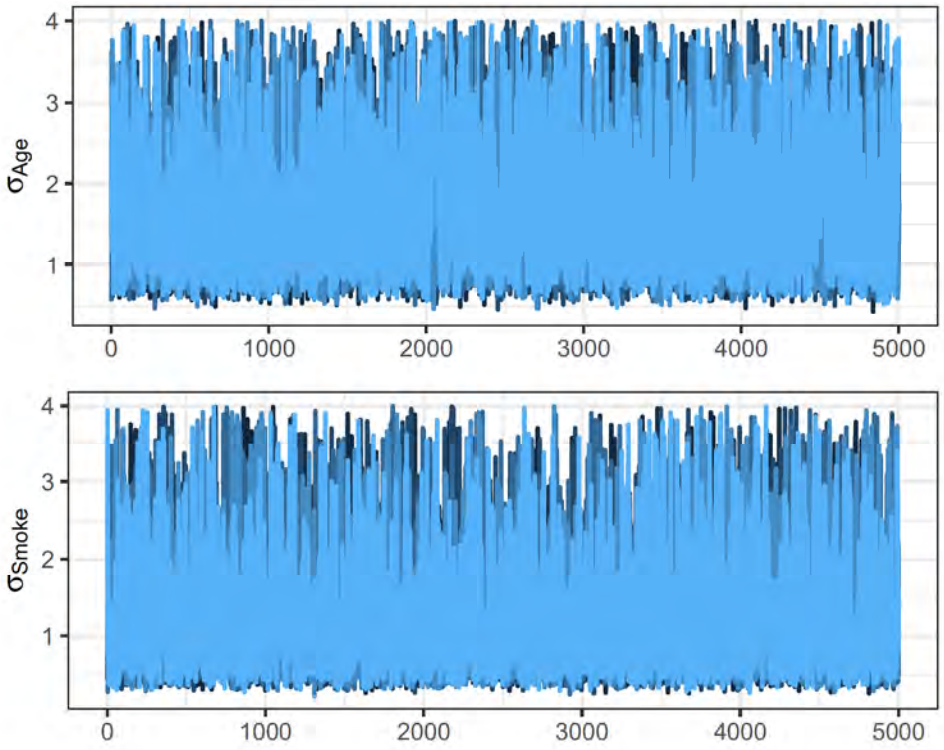

Figure 13: Trace-plots of σ_{Age} and $\sigma_{Smoking}$

Comment:- From Trace Plots we can verify that all parameters estimations generated by our simulations *converged.*

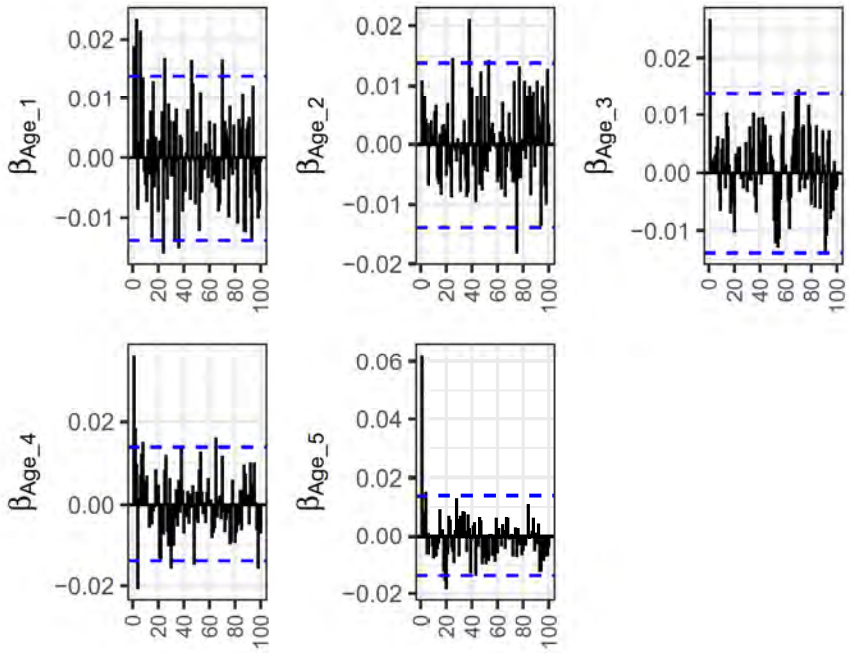

Figure 14: ACF-plots of β_{Age} (adjusted, max-Lag=100)

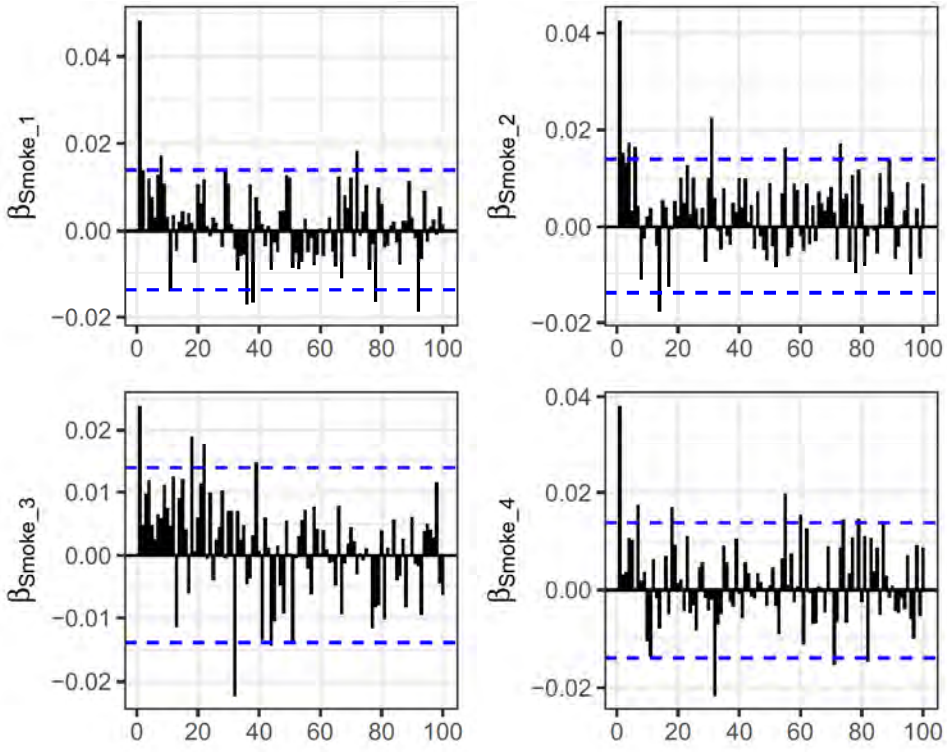

Figure 15: ACF-plots of $\beta_{Smoking}$ (adjusted, max-Lag=100)

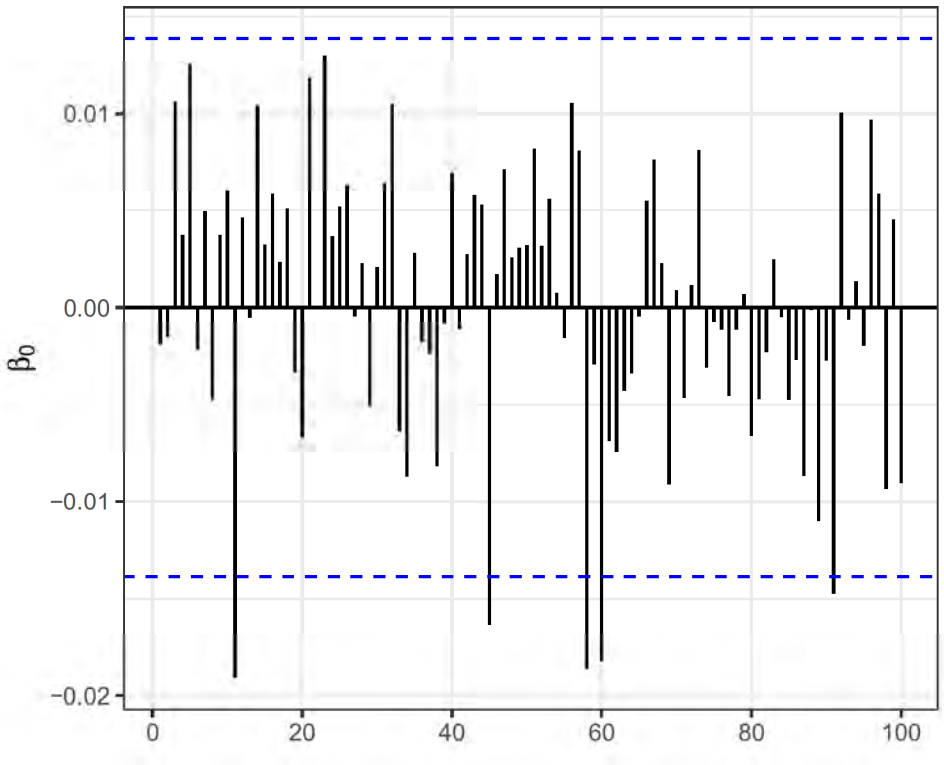

Figure 16: ACF-plots of β_0 (adjusted, max-Lag=100)

Comment:- From ACF Plots we can also verify the parameters estimates generated by our simulations are *uncorrelated*.

item (c)

Provide your belief as to the increase probability of death for someone who smokes > 20 cigarettes per day versus a nonsmoker. (That is, you would say that you believe that a smoker who smokes > 20 cigarettes per day has an increase risk of xxx times the risk of a nonsmoker.) Please, include both a "point" estimate of this increase risk and also state some interval estimate. Please identify the type of point and interval estimate that you are using.

{*Solution.*}

In order to calculate the increase in probability for someone who smokes > 20 cigarettes per day versus a nonsmoker we will analyze the posterior mean of parameters $\beta^{[1]}_{Smoking}$ and $\beta^{[4]}_{Smoking}$, both presented in table below.

Table 3: Smoking Categories - Never Smoked vs. >20 cigarettes per day

Parameter	mean	sd	2.5%	50%	97.5%
beta.Smoking.adj[1]	-0.4585647	0.0919031	-0.6362025	-0.45965	-0.2754975
beta.Smoking.adj[4]	0.9229055	0.1168177	0.6894000	0.92440	1.1480000

There are 02 ways to answer this question.

Using the **modeling results** and **posterior estimates of the parameters**:

- The coefficient *beta.Smoking.adj[1]* represents the expected difference in the number of deaths (on the logarithmic scale) if the individual "Never Smoked". Thus, the expected multiplicative increase is $e^{-0.4585647} = 0.6321904$.
- On the other side, individuals who smoke *more than 20 cigarettes per day*, represented here by *beta.Smoking.adj[4]*, have a multiplicative increase of $e^{0.9229055} = 2.516592$.
- Comparing one group (never-smoker) against the other (>20 cigarettes per day), we can say that the second group has an increased risk of lung cancer of approximately $e^{0.9229055-(-0.4585647)} = 3.98075$ times the risk of a individual who never smoked.

Another way of treat this question is by using **samples from** $\beta_{Smoking[1]}$ **and** $\beta_{Smoking[4]}$ and then generate a *sample of estimated increase in probability of death between both categories* by doing:

$$\Delta p(\text{Death Rate}) = \exp\left[\beta_{Smoking[4]} - \beta_{Smoking[1]}\right]$$

From this sample is possible estimate the *mean increase* and a corresponding *credible interval* for our estimate.

By using this method, we have the following results:

```
##
## ---- Summary Statistics for Increase in Probability -----

##
## Mean :  4.035

##
## Std.Dev. :  0.6655

##
## 95% Credible Region ( 2.730 , 5.339 )

##
## >>> From this model, our belief of increase probability of death by lung cancer

##     of someone who smokes >20 cigarettes per day vs. who never smoked is 4.03

##
##     with corresponding 95% credible region of ( 2.73 , 5.34 )
```

We can see that both methods provide similar results, which confirms our approach.

Question 3

(There is first preamble and background section. This is followed by the things to do for this problem.)

Preamble and Background: First, there is a description of a data analysis problem which includes a data file and a model for the data. You are asked to fit the model to the data with a Bayesian analysis. Use Openbugs or Jags (or a similar program) to fit the model. Please show the model file used by the software program and provide some preliminary evidence that that MCMC has converged. (That is, it is sufficient to use such methods as trace plot examination and looking at a plot of the auto-correlation function.) Afterwards, there are additional questions about the model that you will need to answer.

The data in the CSV file named `DiabetesDrugEffect.csv` which contains data from 12 different clinical trial. The purpose of each of the clinical trials was to see if drug A was better than drug B. (Aside, they were looking to lower blood glucose in diabetic patients.). The outcome results were measured on a continuous scale. The column labeled "diff" was the main result in the study and is the overall average difference between drug A and drug B. The column "Sediff" is the estimate of the standard error of the difference estimate. The column "StudyN" is the number of subjects that were in each of the studies. The studies are labeled with the variable "StudyId" and take on values from 1 to 12. Note that the values from each study are obtained from the individual study publications.

The goal is to combine the evidence in the 12 different studies. That is, the goal is to use the 12 different studies to obtain an overall belief in the average difference between the two drugs. In the medical literature, this is called a *meta-analysis*. Here, we make the assumption that for each study site there are unknown site specific conditions so that each site has a different effect and that these individual effects can be modeled as a normal distribution around an overall average different. That is, the following model is assumed.

- Define θ as the overall "true" effects of the difference between the two drugs. The main goal of this analysis is to learn about this parameter.
- Define δ_i be the true difference between treatments for the i-th study. These study effects are normally distributed around θ with a common variance of σ_0^2. The parameter σ_0^2 is of moderate interest.
- Let Y_i be the observed difference for the i-th study. This is the estimate for δ_i from the original paper from the i-th study.
- Let S_i be the observed standard error of the estimator Y_i of δ_i. It is assumed one can ignore the error in using the observed value S_i for the true value. That is, here one is assuming that the observed S_i is the exact value.

The implies of the following model which we can call model 1:

$$Y_i|\delta_i, S_i \sim \text{Normal}(\text{mean} = \delta_i, \text{variance} = S_i^2)$$

$$S_i|\theta, \sigma_0^2 \sim \text{Normal}(\text{mean} = \theta, \text{variance} = \sigma_0^2)$$

Sometimes one is not concerned with the latent variables δ_i. So, one can collapse the model into the following collapsed model which we call model 2:

$$Y_i|\theta, \sigma_0^2 \sim \text{Normal}(\text{mean} = \theta, \text{variance} = S_i^2 + \sigma_0^2)$$

(Note: in Openbugs/Jags, when one is using the normal distribution it is correct that the first parameter is still the mean but the second parameter is the precision which is 1/variance.)

Do the following for question 3

item (a)

Fit both model 1 and model 2 with Openbugs/Jags. Provide the model file. Provide evidence that the MCMC has converged. (You don't need to provide advance statistics. Trace plots and plot of the auto-correlation function with the appropriate discussion is sufficient for the purpose of this exam.) Also, provide summary statistics of the posterior distributions. (The statistics such as the posterior mean, posterior standard deviation, and credible region obtain from a print statement is sufficient.)

{*Solution.*}

For this question, we generated 02 `OpenBugs` configuration files, one for each model.

They are listed below[2].

Model 1 - Configuration file

```
model{
for(i in 1:NDiab)
{
#StudyID  StudyN  diff  Sediff

diff[i]~dnorm(delta[i], tau[i])
delta[i]~dnorm(theta, tau0)
tau[i] <- 1/pow(Sediff[i],2)

}

# Initial Priors - Verify
theta ~dnorm(-1.5, 30)  # v.19
sigma0 ~dgamma(2, 1)    # v.19
tau0 <- 1/pow(sigma0,2)

}
```

Model 2 - Configuration file

```
model{
for(i in 1:NDiab)
{
#StudyID  StudyN  diff  Sediff

diff[i]~dnorm(theta, tau[i])
tau[i] <- 1/(pow(sigma0,2)+pow(Sediff[i],2))
}

# Initial Priors - Verify
theta ~dnorm(-1.5, 30)  # v.19
sigma0 ~dgamma(2, 1)    # v.19

}
```

[2]The detailed R-Code can be verified at the Appendix of this document.

Simulation and Summary Statistics for Models 1 and 2

After simulating $N = 30,000$ samples with *burn-in* of $10,000$ samples, with 5 chains the mean, standard-deviation and respective credible regions for model's parameters are listed in the following Report Summary.

Table 4: Model 1 - Difference between Drug A and B

Parameter	mean	sd	2.5%	50%	97.5%
theta	-1.5073520	0.1158132	-1.738000	-1.5070	-1.2800000
tau0	9.2116823	25.1126545	1.688000	6.3460	29.1700000
sigma0	0.4179797	0.1505164	0.185200	0.3970	0.7697000
delta[1]	-1.0709825	0.1704370	-1.401000	-1.0730	-0.7351000
delta[2]	-1.5750672	0.3992128	-2.405000	-1.5650	-0.7952975
delta[3]	-1.5288841	0.2260357	-1.977000	-1.5290	-1.0830000
delta[4]	-1.5013430	0.2285132	-1.951000	-1.5020	-1.0500000
delta[5]	-1.6939467	0.2356354	-2.172000	-1.6880	-1.2440000
delta[6]	-1.8238888	0.2435459	-2.320000	-1.8170	-1.3660000
delta[7]	-1.7423319	0.1885267	-2.117000	-1.7400	-1.3820000
delta[8]	-1.7410468	0.1929021	-2.129000	-1.7380	-1.3709750
delta[9]	-1.7900834	0.2698410	-2.346000	-1.7810	-1.2850000
delta[10]	-1.1909227	0.2397234	-1.641000	-1.1980	-0.7040975
delta[11]	-1.0668475	0.2969962	-1.606000	-1.0800	-0.4464000
delta[12]	-1.4019725	0.3365700	-2.052000	-1.4110	-0.7045975
deviance	5.2510553	4.5485953	-1.844025	4.6295	15.8000000

Table 5: Model 2 - Difference between Drug A and B

Parameter	mean	sd	2.5%	50%	97.5%
theta	-1.5076295	0.1157006	-1.7340000	-1.5080	-1.2790
sigma0	0.4174542	0.1500608	0.1831975	0.3968	0.7694
deviance	16.6930346	1.9819034	14.7500000	16.0900	22.0300

Convergence of MCMC Simulation

For this section, we will show the Trace and ACF plots to demonstrate the posterior estimates of the parameters converged and are uncorrelated, as well.

The same restriction applied on Question 2 was also applied, i.e., trace and ACF plots with number of points limited to $5,000$ to preserve performance of PDF file.

Convergence and Uncorrelation of Parameters from Model 1

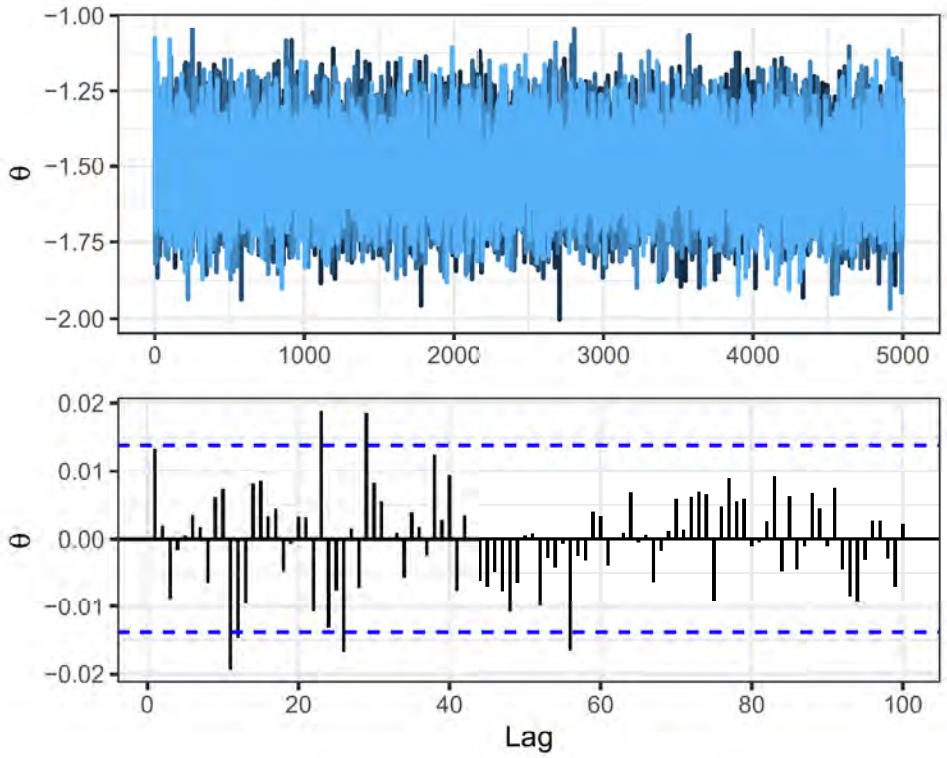

Figure 17: Model 1 - Trace and ACF of θ

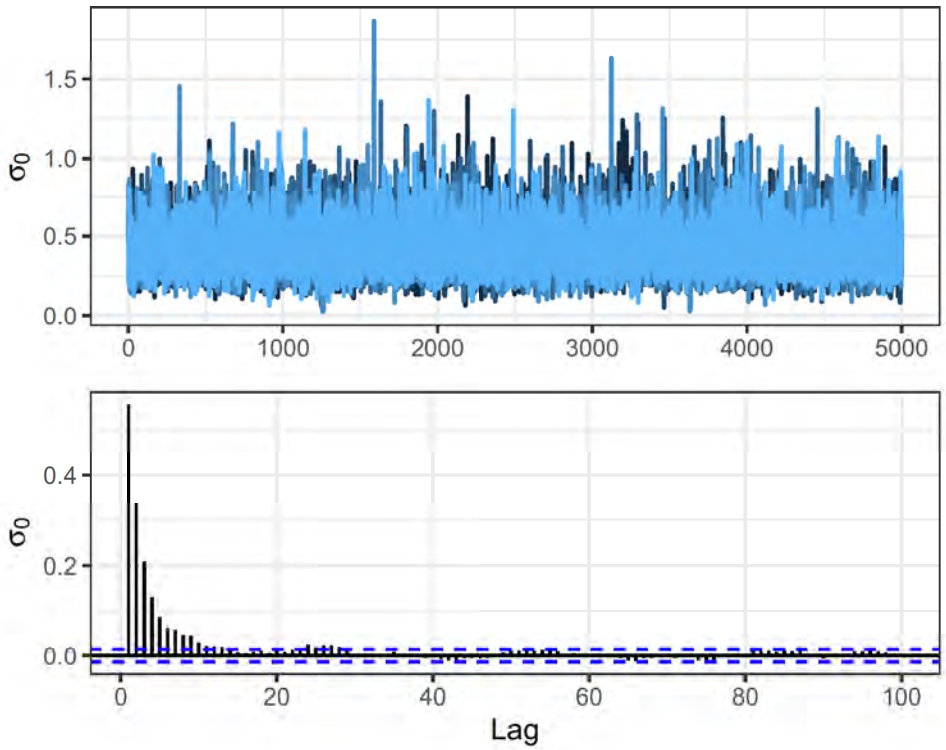

Figure 18: Model 1 - Trace and ACF of σ_0

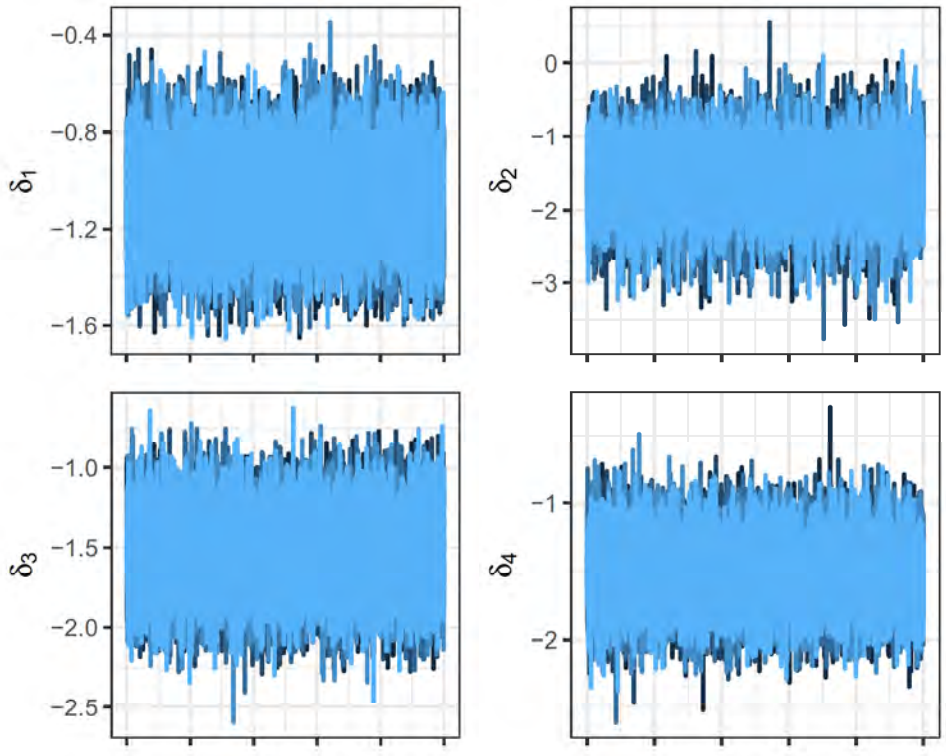

Figure 19: Model 1 - Trace-plots for $\hat{\delta}_{1:4}$

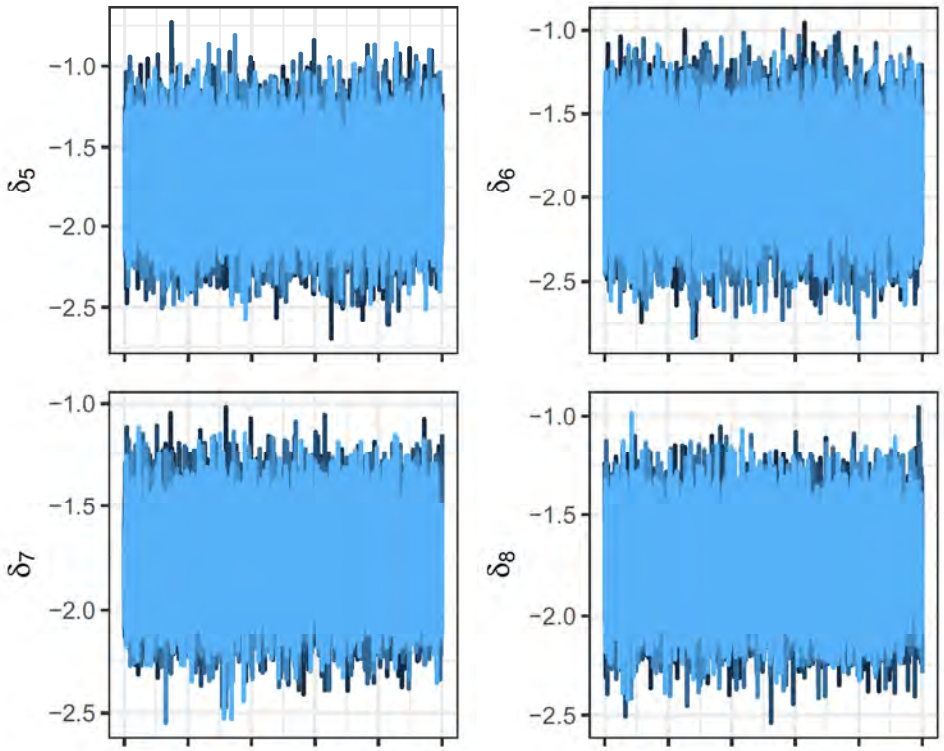

Figure 20: Model 1 - Trace-plots for $\hat{\delta}_{5:8}$

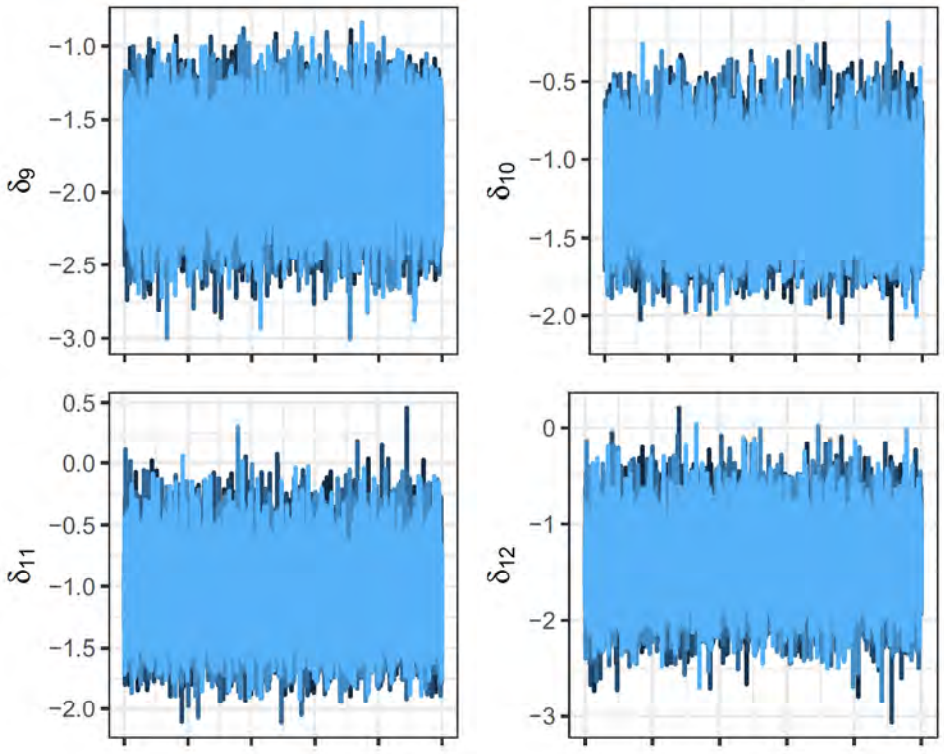

Figure 21: Model 1 - Trace-plots for $\hat{\delta}_{9:12}$

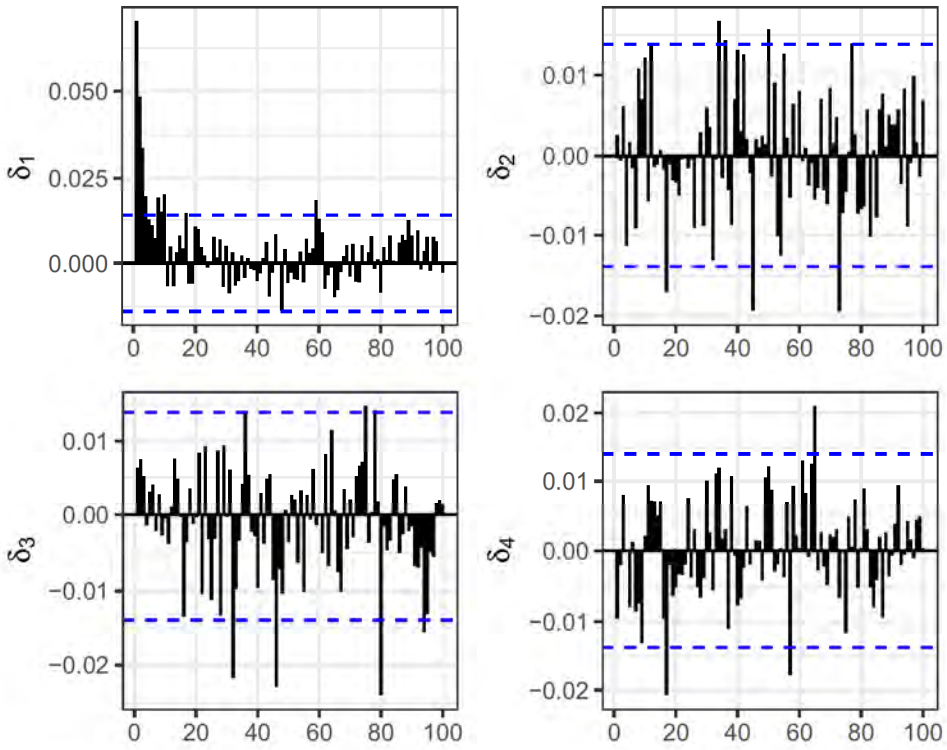

Figure 22: Model 1 - ACF-plots for $\hat{\delta}_{1:4}$

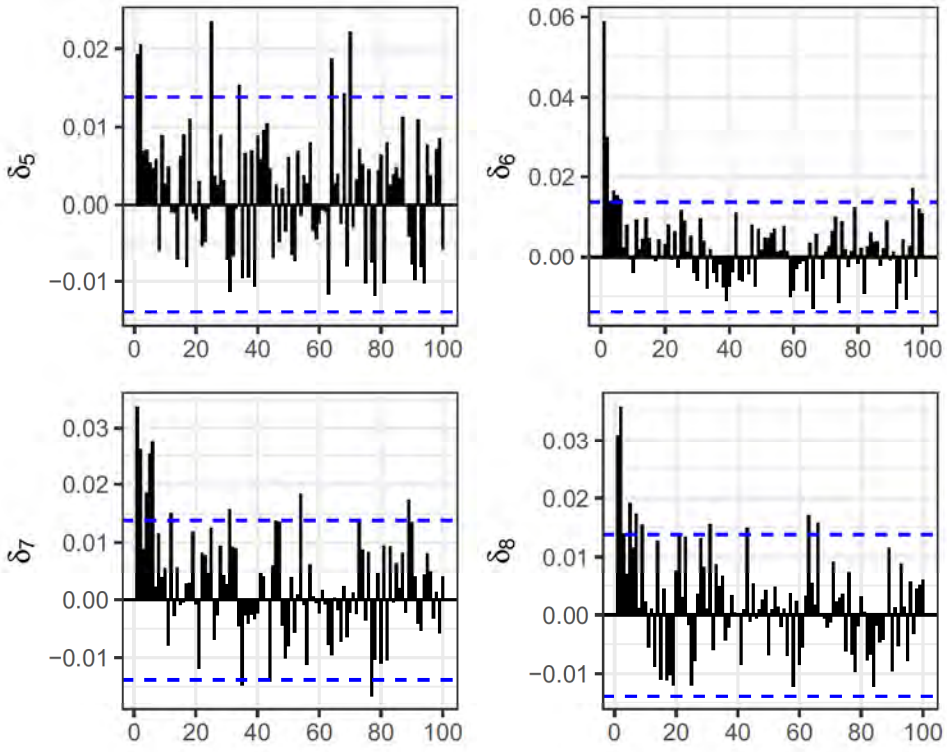

Figure 23: Model 1 - ACF-plots for $\hat{\delta}_{5:8}$

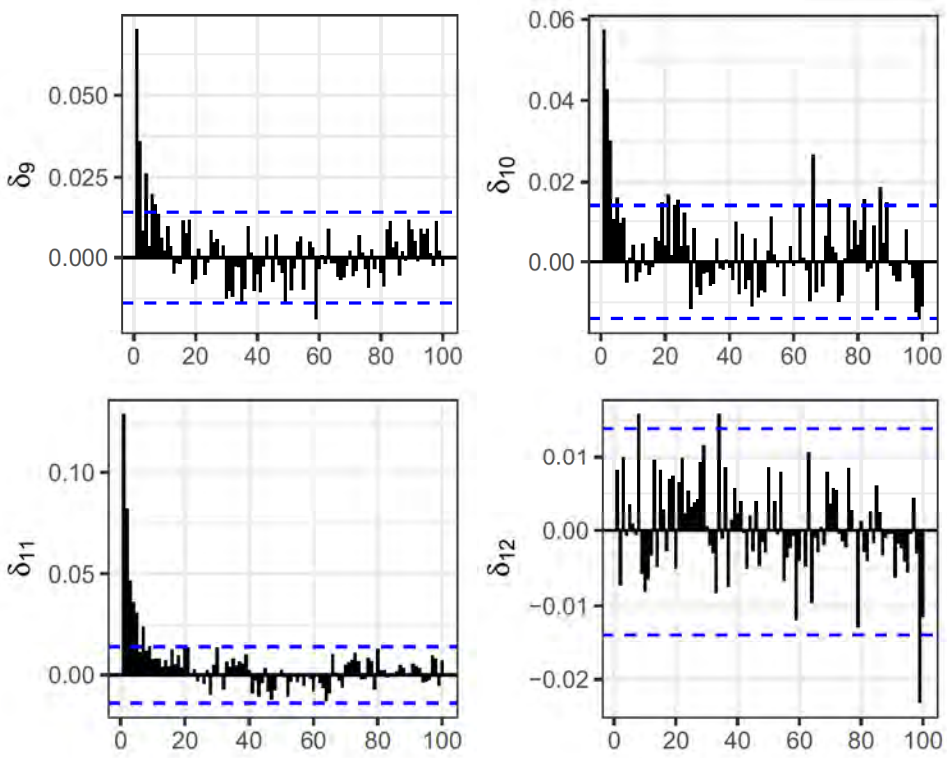

Figure 24: Model 1 - ACF-plots for $\hat{\delta}_{9:12}$

Convergence and Uncorrelation of Parameters from Model 2

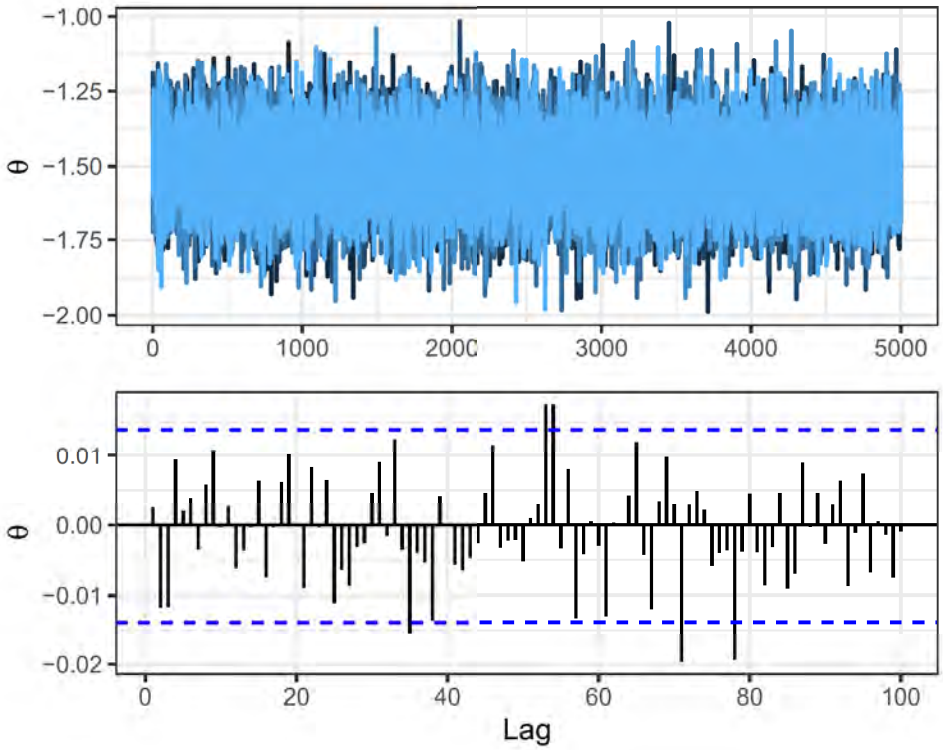

Figure 25: Model 2 - Trace and ACF of θ

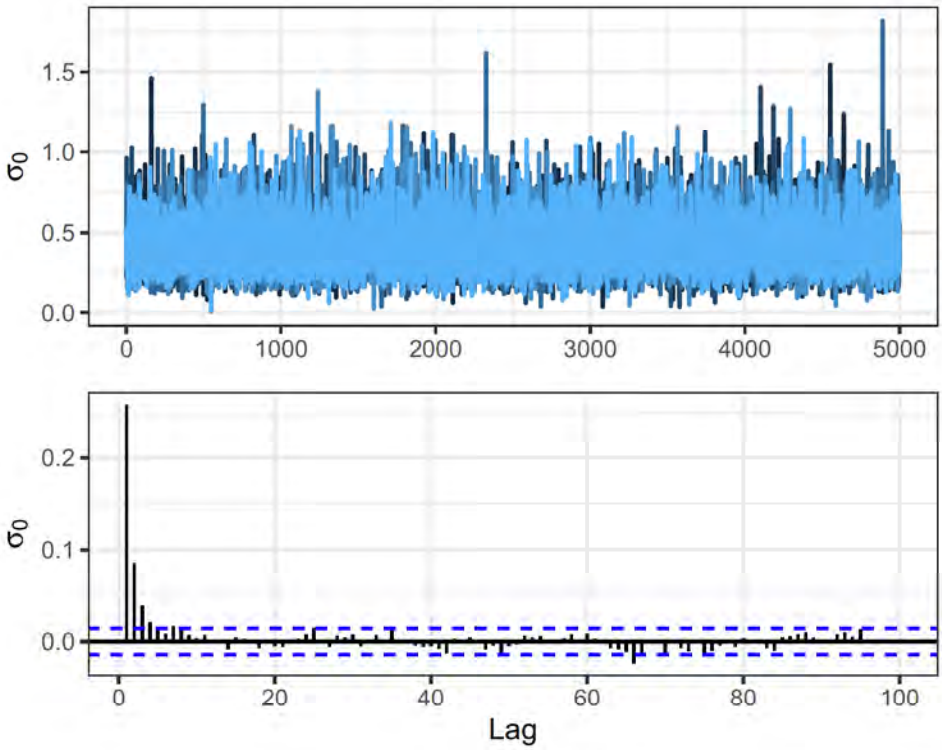

Figure 26: Model 2 - Trace and ACF of σ_0

Comment:- From Trace and ACF Plots we can verify the parameters estimates generated by our simulations for **Models 1 and 2** *converged* and are *uncorrelated.*

item (b)

For model 1 and model 2, compare the estimates of the posterior distribution of θ and σ_0 from the two models. Are they close?

{*Solution.*}

Posterior distributions of θ and σ_0 compared

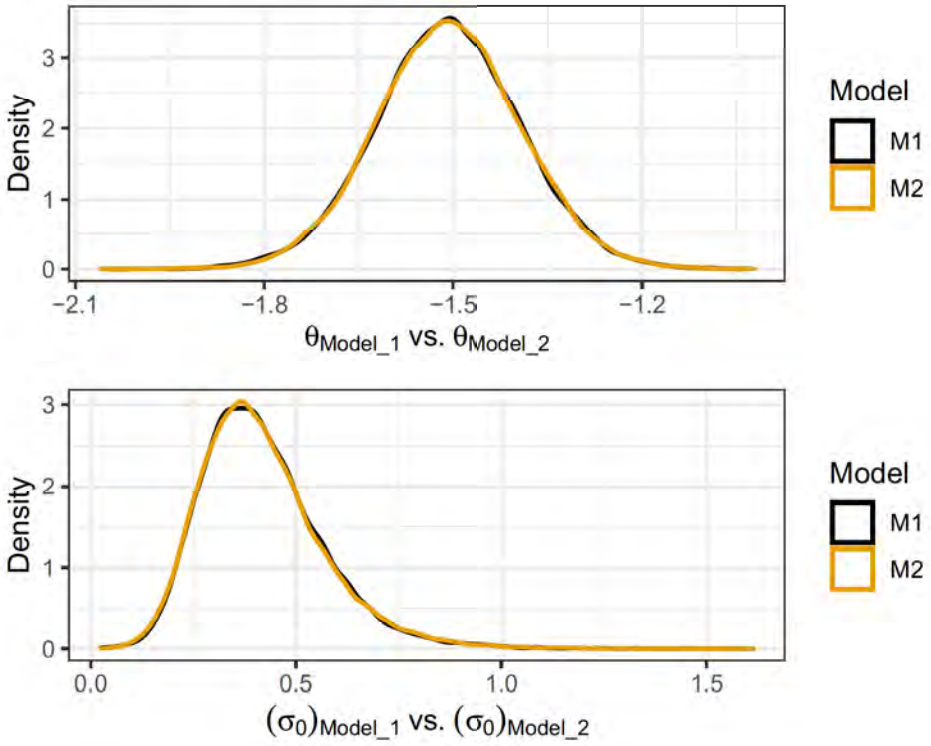

Figure 27: Posterior Distributions of θ and σ_0^2 for Models 1 and 2

- **Comparison of θ** - Both densities share almost the same location and scale parameters, which makes their respective densities to be close. As we started from common priors for both models, the hyper-parameter θ will present similar estimates, as well.
- **Comparison of σ_0** - The estimates for standard deviations of δ is almost the same for both models. This is explained because we started from same priors for σ_0 for both models. Despite of the fact they are used in different ways to model the precision of the difference between the drugs, they perform quite well in both models.

By this modelling exercise, we could verify that the posterior overall difference between drug A and drug B (modeled through θ) are almost the same, and the condensed Model-2 is as good as complete Model-1.

item (c)

Using model 1, one gets an estimate of the posterior distribution of the individual study effects, δ_i. That is, one can get the posterior mean, standard deviation and credible region. Also, from the original, individual study data, one can use the variables "diff" and "Sediff" to get an estimate of the study effects from the original studied publications. Therefore, one can use these values to obtained the estimated effect from the study and the confidence interval for the estimate. Compare the estimates from the individual studies to the estimates from the meta-analysis. For each study compare the values of "diff" and the associated confidence interval with the posterior mean of the δ_i estimate and the associated credible region.

(i) You will probably find that the estimate of the individual study effect using "diff" will be different than the estimate using the posterior mean of δ. When considering the movement of the estimate from the value "diff" to the posterior mean of δ, what general direction is this movement?

(ii) When considering the length of the confidence intervals (using the individual study publications) compared to the credible region (obtained from the Bayesian meta-analysis), what generally happens to the change in the length?

(iii) When looking at study 2, note that this study had the largest value of "Sediff" which is the standard error of the estimate in the original study. You might notice that there is a large difference between "diff" compared to the posterior mean of δ compared to most of the other studies. What might be causing this large shift. (Look at some other features of this study.)

{*Solution.*}

Comparison between "diff" / posterior means of δ_i and respective C.I. and C.R.'s

In order to answer this question, we plotted 02 comparative graphs composed by *Credible Regions* and *Confidence Intervals*, with its respective posterior means and estimates.

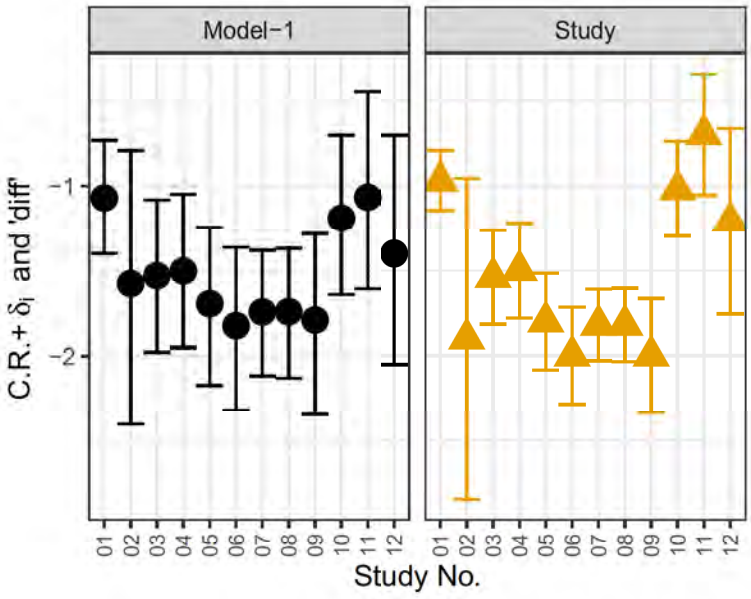

Figure 28: Comparative C.R. vs. C.I. for Model-1 and Studies

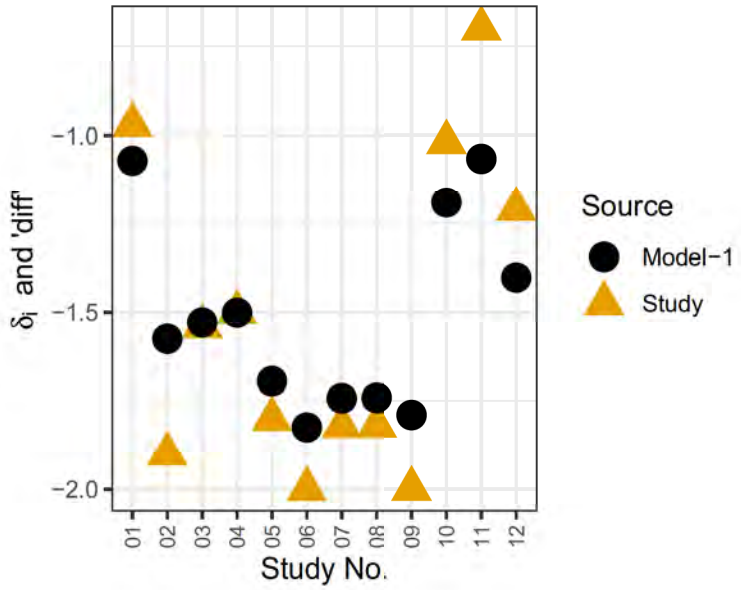

Figure 29: Comparative Posterior Means vs. δ's for Model-1 and Studies

(i) From these graphs we can see that for 1/3 (i.e., for studies $1, 10, 11$ and 12), the posterior mean for δ_i is smaller than the obtained in that study. For the other group (i.e, 2/3 or the remaining 8 studies), we have the reverse scenario - posterior mean is smaller than obtained in studies. If we consider that *smaller negative numbers* of δ_i and "diff" represent larger differences between the two drugs, we can say our overall scenario is more *conservative*, considering a more balanced view of the effect of both drugs.

(ii) When analyzing the C.R's and C.I.'s, we observed that, with exception of study No.2, all Credible Regions are wider than the corresponding Confidence Interval obtained on each study. The mean estimate of σ_0 is 0.418 for both models, which is higher than most of the standard-error in 10 studies (exceptions are studies #2 and #12), possibly generating the discrepancy between the length of intervals.

(iii) We noticed study No. 2 has one of the smallest samples from all studies, here represented by N. [1] justifies that with larger sample sizes (i.e., more than 50 persons in each treatment group in nearly all of the studies), the variances in each study are *more precisely estimated* so the initial assumption that S_2 was the true standard error for study No. 2 may not be completely true and this might have impacted the determination of the estimates (this study has the highest difference between the posterior mean of δ_2 and "delta[2]").

Another important aspect to notice is that Model-1 uses "Sediff[i]" to compose the variance of Y's which might have interfered in our estimation for δ of each study. Using larger differences as the true S_i imposes more variability of estimates on these studies, with consequent larger credible regions and larger differences between posterior mean of δ_i and "delta[i]".

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] Selvin, S. *Practical Biostatistical Methods* - Dusbury Press, 1995

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

[5] Hirst J. A., Farmer A. J., Dyar A., Lung T. W. C., Stevens, R. J. *Estimating the effect of sulfonylurea on HbA_{1c} in diabetes: a systematic review and meta-analysis*, Diabetologia 56:973–984 (2013)

[6] Sutton A.J., Abrams K.R., Jones D.R., *Methods for Meta-Analysis in Medical Research.* London: Wiley, 2000.

[7] Friendly, M., Meyer., D. *Discrete Data Analysis with R - Visualization and Modeling Techniques for Categorical and Count Data.* CRC, 2013.

Appendix - R-Code

```
knitr::opts_chunk$set(echo = FALSE, message = FALSE, fig.width=5, fig.height=4.0)
library(tidyverse)
library(R2OpenBUGS)
library(kableExtra)
library(ggplot2)
library(forecast)

# 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")

# Read database
bb_data <- read.csv("Data/brnbdy.csv")

# Introducing Scale Power transformation - Weisberg(2013) pp.189-190
scalePower <- function (Z,Lambda=0, Mod = FALSE) {
  # Introduced by Box, Cox (1964) for strictly positive Z
  if (Mod)
    gm = exp(sum(log(Z)/length(Z)))
  else
    gm = 1.0
  if (Lambda == 0)
    return(gm*log(Z))
  else
    return(gm^(1-Lambda)*(Z^Lambda-1)/Lambda)
}

# Setup Transformed Variables
Y  <-  scalePower(bb_data$brain)
X  <-  scalePower(bb_data$body)


# Prepare dataframe and plot brain vs. body weight
df <- data.frame (lBrW = Y, lBdW = X)

# Plot Graph for
df %>%
  ggplot(aes(x=X, y=Y))+
  geom_point(size=1.8)+
  labs(x = "log(Body Weight)", y = "log(Brain Weight)") +
  ggtitle(NULL)+
  theme_bw()

# Remove Temporary Variable
remove(df)

# Setup Initial Variables
```

```r
Mu_alpha <- Mu_beta <- 0
Tau_alpha <- Tau_beta <- 0.0001
a_Tau<- b_Tau <- 0.0001  # v.16

# Setup Iteration Variables
N <- 20000

alpha <- beta <- tau <- rep(N,0)

# Setup \theta^{(0)}
alpha[1] <- beta[1] <- 0
tau[1] <- 1

# Calculates the value of parameters, given everything else
sample_Alpha <- function (X, Y, beta, tau, mu_alpha, tau_alpha) {
  tau_alphaStar <- length(Y)*tau+tau_alpha
  mu_alphaStar <- (tau*sum(Y-beta*X)+tau_alpha*mu_alpha)/tau_alphaStar
  alpha <- rnorm(1,mean=mu_alphaStar, sd=sqrt(1/tau_alphaStar))
  return(alpha)
}

sample_Beta <- function (X, Y, alpha, tau, mu_beta, tau_beta) {
  tau_betaStar<- tau*sum(X^2)+tau_beta
  mu_betaStar <- (tau*sum(X*(Y-alpha))+tau_beta*mu_beta)/tau_betaStar
  beta <- rnorm(1,mean=mu_betaStar, sd=sqrt(1/tau_betaStar))
  return(beta)
}

sample_Tau <- function (X, Y, alpha, beta, a_r, b_r) {
  a_rStar <- a_r+length(Y)/2
  b_rStar <- b_r+0.5*sum((Y-(alpha+beta*X))^2)
  tau <- rgamma(1,shape=a_rStar, rate=b_rStar)
  return(tau)
}

Gibbs <- function(X, Y, alpha, beta, tau, N) {
  for (i in 2:N) {
    alpha[i] <- sample_Alpha(X, Y, beta[i-1], tau[i-1], Mu_alpha, Tau_alpha)
    beta[i] <- sample_Beta(X, Y, alpha[i], tau[i-1], Mu_beta, Tau_beta)
    tau[i] <- sample_Tau(X, Y, alpha[i], beta[i], a_Tau, b_Tau)   # v.17
  }
  return(list(Alpha = alpha, Beta = beta, Tau = tau))
}

# Set Seed for reproducibility
set.seed(737)

#Calculates 3 chains
ThetaC1 <- Gibbs(X, Y, alpha, beta, tau, N)
ThetaC2 <- Gibbs(X, Y, alpha, beta, tau, N)
ThetaC3 <- Gibbs(X, Y, alpha, beta, tau, N)
```

```r
# Plot ACF * ALPHA, BETA and TAU *

# Determine Burnin based on the convergence of ACF plot

# Build working dataframe
D_WorkAcfABT <- data.frame(ValCh1=ThetaC1$Alpha,
                           ValCh2=ThetaC1$Beta,
                           ValCh3=ThetaC1$Tau)


P1 <- ggAcf(D_WorkAcfABT$ValCh1, lag.max = 100)+
  labs(x = "Lag", y = expression(alpha)) +
  ggtitle(NULL)+
  theme_bw()
P2 <- ggAcf(D_WorkAcfABT$ValCh2, lag.max = 100)+
  labs(x = "Lag", y = expression(beta)) +
  ggtitle(NULL)+
  theme_bw()
P3 <- ggAcf(D_WorkAcfABT$ValCh3, lag.max = 100)+
  labs(x = "Lag", y = expression(tau)) +
  ggtitle(NULL)+
  theme_bw()

ggpubr::ggarrange(P1, P2, P3, ncol = 2, nrow = 2)

# Remove Working Variable to free memory
remove(D_WorkAcfABT)

# Set Burn-in parameter
Burnin <- 100

# Prepare Data-Frame for next steps
D1_Work <- rbind(data.frame(alpha=ThetaC1$Alpha[Burnin:N],
                            beta=ThetaC1$Beta[Burnin:N],
                            tau=ThetaC1$Tau[Burnin:N],Chain=factor(1)),
                 data.frame(alpha=ThetaC2$Alpha[Burnin:N],
                            beta=ThetaC2$Beta[Burnin:N],
                            tau=ThetaC2$Tau[Burnin:N],Chain=factor(2)),
                 data.frame(alpha=ThetaC3$Alpha[Burnin:N],
                            beta=ThetaC3$Beta[Burnin:N],
                            tau=ThetaC3$Tau[Burnin:N],Chain=factor(3)))


# Plot TracePlots * ALPHA, BETA and TAU *

set.seed(340)

Sz <- 5000

L <- N-Burnin+1

# Sampling 5000 points to generate "thinner" traceplots
S <- if (Sz <= L) sort(sample(Burnin:N, Sz, replace = FALSE)) else Burnin:N
```

```
# Build working dataframe
D_WorkTpAlpha <- data.frame(ValCh1=ThetaC1$Alpha[S],
                            ValCh2=ThetaC2$Alpha[S],
                            ValCh3=ThetaC3$Alpha[S])
P1 <- D_WorkTpAlpha %>%
  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(alpha)) +
  theme_bw()

# Build working dataframe
D_WorkTpBeta <- data.frame(ValCh1=ThetaC1$Beta[S],
                           ValCh2=ThetaC2$Beta[S],
                           ValCh3=ThetaC3$Beta[S])
P2 <- D_WorkTpBeta %>%
  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)) +
  theme_bw()

# Build working dataframe
D_WorkTpTau <- data.frame(ValCh1=ThetaC1$Tau[S],
                          ValCh2=ThetaC2$Tau[S],
                          ValCh3=ThetaC3$Tau[S])
P3 <- D_WorkTpTau %>%
  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()

# Build working dataframe
D_WorkTpAlphaBeta <- data.frame(ValAlpha=c(ThetaC1$Alpha[S],
                                           ThetaC2$Alpha[S],
                                           ThetaC3$Alpha[S]),
                                ValBeta=c(ThetaC1$Beta[S],
                                          ThetaC2$Beta[S],
                                          ThetaC3$Beta[S]))

P4 <- D_WorkTpAlphaBeta %>%
  ggplot(aes(x=ValAlpha, y=ValBeta))+
  geom_point(size=0.6)+
  labs(x = expression(alpha), y = expression(beta)) +
  theme_bw()

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(legend.position="none"), ncol = 2, nrow = 2)

# Remove Working Variable to free memory
remove(D_WorkTpAlpha, D_WorkTpBeta, D_WorkTpTau, D_WorkTpAlphaBeta)

# Plot the Graphs of Alpha
P1 <- D1_Work %>%
  ggplot(mapping = aes(x = alpha, group = Chain))+
  geom_density(aes(colour=Chain), size=0.5)+
  labs(x = expression(alpha), y = "Density") +
  scale_color_manual(values = cbp2)+
  theme_bw()

# Plot the Graphs of Beta
P2 <- D1_Work %>%
  ggplot(mapping = aes(x = beta, group = Chain))+
  geom_density(aes(colour=Chain), size=0.5)+
  labs(x = expression(beta), y = "Density") +
  scale_color_manual(values = cbp2)+
  theme_bw()

# Plot the Graphs of Tau
P3 <-D1_Work %>%
  ggplot(mapping = aes(x = tau, group = Chain))+
  geom_density(aes(colour=Chain), size=0.5)+
  labs(x = expression(tau), y = "Density") +
  scale_color_manual(values = cbp2)+
  theme_bw()

ggpubr::ggarrange(P1, P2, P3,ncol = 1, nrow = 3)


# Summary Statistics - alpha
prt_SummaryStat <- function (sample_, parameter) {
  m_ <- mean(sample_)
  sd_ <- sd(sample_)
  cat("\n---- Summary Statistics for ",parameter," -----")
  cat("\nMean",parameter,": ",format(m_, digits = 2, nsmall = 3))
  cat("\nStd.Dev.",parameter,": ",format(sd_, digits = 2, nsmall = 4))
  CI <- qnorm(p = c(0.025,0.975), mean = m_, sd = sd_)
  # cat("\nC.I. for",parameter,": [",CI[1],",",CI[2],"]\n")
  cat("\n95% Credible Region for",parameter,"(", format(CI[1], digits = 2, nsmall = 3),
      ",",format(CI[2], digits = 4, nsmall = 3),")\n")
}
prt_SummaryStat(D1_Work %>% pull(alpha), "alpha")
prt_SummaryStat(D1_Work %>% pull(beta), "beta")
prt_SummaryStat(D1_Work %>% pull(tau), "tau")


# Using the Alphas and Betas found on item (a) We will generate a sample of exp(Y_55)
```

```r
D2_Work <- D1_Work %>%
  mutate(Y_55 = alpha+beta*log(55))

D2_Work %>%
  ggplot(mapping = aes(x = exp(Y_55), group = Chain))+
  geom_density(aes(colour=Chain), size=0.8)+
  labs(x = expression("Estimated Brain Weight | Body Weight=55Kg"), y = "Density") +
  scale_color_manual(values = cbp2)+
  theme_bw()

# Summary Statistics for the Distribution of an animal with average bodyweight of 55
m_a <- mean(exp(D2_Work$Y_55))
sd_a <- sd(exp(D2_Work$Y_55))
cat("\n---- Summary Statistics for Brain Weight | Body Weight = 55kg -----")
cat("\nMean : ",format(m_a, digits = 2, nsmall = 3))
cat("\nStd.Dev. : ",format(sd_a, digits = 2, nsmall = 4))
CI_a <- qnorm(p = c(0.025,0.975), mean = m_a, sd = sd_a)
cat("\n95% Credible Region (", format(CI_a[1], digits = 2, nsmall = 3),
    ",",format(CI_a[2], digits = 2, nsmall = 3),")\n")
cat("\n>>> From this model, our belief of Brain Weight for an animal with \n")
cat("    average Body Weight of 55Kg is",format(m_a, digits = 2, nsmall = 2),
    "grams, with a 95% credible\n")
cat("    region of (",format(CI_a[1], digits = 2, nsmall = 2),",",
    format(CI_a[2], digits = 2, nsmall = 2),") grams.\n")

# Setup Data-set - Read Data
shist_data <- read.table(file="data/SmokeAgeDeath.csv",header=TRUE,sep=",")
colnames(shist_data) <- c("Smoking", "Age", "Deaths", "PersonYears")

# Setup Variables
NSmok=nrow(shist_data)  # Number of Items in data-set
NCSmok=4    # Number of classes of Smoking
NCAge=5     # Number of Classes of Age

# Setup OpenBugs running parameters
NSim <- 30000    # No. of simulations for productio
NChain <- 5      # No. of chains for production
NThin <- 5       # 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
shist_data %>%
  kbl(booktabs = TRUE,
      caption = "Deaths by Lung Cancer, per Age, Smoking Level and Person-Years") %>%
  kable_styling(latex_options = "striped")


cat("
model{
for(i in 1:NSmok)
{
```

```
#Age   Smoking   Deaths  PersonYears

Deaths[i]~dpois(lambda[i])
log(lambda[i]) <-  log(PersonYears[i]) + beta0 + beta.Age[Age[i]]+ beta.Smoking[Smoking[i]] + bias[i]
bias[i] ~dnorm(0,tau)
bias.adj[i] <- bias[i] - mean(bias[])
}

# Loop for Age
for(ia in 1:NCAge){
beta.Age[ia]~dnorm(0,tau.Age)
beta.Age.adj[ia] <- beta.Age[ia] - mean(beta.Age[])
}

# Loop for Smoking Level
for(is in 1:NCSmok){
beta.Smoking[is]~dnorm(0,tau.Smoking)
beta.Smoking.adj[is] <- beta.Smoking[is] - mean(beta.Smoking[])
}

# Initial Priors - Verify
beta0 ~ dnorm(0, 0.008)   # Overall rate of disease
beta0.adj <- beta0 + mean(bias[]) + mean(beta.Age[])+ mean(beta.Smoking[])

# Initial Priors - Verify
std ~ dunif(0, 4)
tau <- 1/pow(std,2)

# Initial Priors - Verify
std.Age ~dunif(0, 4)
tau.Age <- 1/pow(std.Age,2)
std.Smoking ~ dunif(0,4)
tau.Smoking <- 1/pow(std.Smoking,2)

}", file="LungCancerCentered.txt")

# Setup Parameters
params=c("beta.Age.adj",  "std.Age", "beta.Smoking.adj", "std.Smoking",
         "beta0.adj", "std", "tau", "tau.Age", "tau.Smoking")

# Setup Initial Values
init.fun=function(){list(
  beta.Age=rnorm(5), std.Age=runif(1,0,4),
  beta.Smoking=rnorm(4), std.Smoking=runif(1,0,4),
  std=runif(1,0,4), beta0=rnorm(1),
  bias=rnorm(20,0,.1))}


# Run Open Bugs
set.seed(2602)
model.data <- list("Age", "Smoking","Deaths","PersonYears","NSmok",
                   "NCSmok", "NCAge")
attach(shist_data)
```

```
LungCanc_v0=bugs(model.data, init.fun, params, model.file="LungCancerCentered.txt",
                 working.directory = ".", n.chains=NChain, n.iter=NSim,
                 n.burnin=Burnin, n.thin=NThin)
detach(shist_data)


# Get Simulation from OpenBugs
SArray= LungCanc_v0$sims.array    # Data Arrays
vname=attr(SArray,"dimnames")[3][[1]]  # Variable Names


# Print Summary statistics of parameters oif Interest
RNames <- rownames(LungCanc_v0[["summary"]][,c(1:3,5,7)]) # List of parameters
df_Prt <- data.frame(Parameter = RNames)
df_Prt <- cbind(df_Prt, as_tibble(LungCanc_v0[["summary"]][,c(1:3,5,7)]))
df_Prt %>%
  kbl(booktabs = TRUE,
      caption = "OpenBugs Summary - Deaths from Lung Cancer") %>%
  kable_styling(latex_options = "striped")

# Plot Sample densities to Compare Distributions over *BETA-0*
D_WorkBeta0 <- as_tibble(data.frame(LvlBeta=as.vector(SArray[,,"beta0.adj"])))

D_WorkBeta0 %>%
  ggplot(aes(x = LvlStd))+
  geom_density(aes(x=LvlBeta), size=1.2)+
  labs(x = expression(list(beta[0])), y = "Density") +
  scale_color_manual(values = cbp2)+
  theme_bw()

# Remove Working Variable to free memory
remove(D_WorkBeta0)


# Plot Sample densities to Compare Distributions over *AGE*
D_WorkAge <- as_tibble(rbind(data.frame(LvlAge=as.vector(SArray[,,paste0("beta.Age.adj[", 1, "]")]),Age=
                             data.frame(LvlAge=as.vector(SArray[,,paste0("beta.Age.adj[", 2, "]")]),Age=
                             data.frame(LvlAge=as.vector(SArray[,,paste0("beta.Age.adj[", 3, "]")]),Age=
                             data.frame(LvlAge=as.vector(SArray[,,paste0("beta.Age.adj[", 4, "]")]),Age=
                             data.frame(LvlAge=as.vector(SArray[,,paste0("beta.Age.adj[", 5, "]")]),Age=


P0 <- D_WorkAge %>%
  ggplot(aes(x = LvlAge))+
  geom_density(aes(x=LvlAge,group=Age, colour=Age), size=1.2)+
  labs(x = expression(list(beta[Age_][i])), y = "Density") +
  scale_color_manual(values = cbp2)+
  theme_bw()

P0 + geom_segment(aes(x = -2, y = 2, xend = 1.5, yend = 4), size = 0.3,
                  arrow = arrow(length = unit(0.5, "cm")))

# Remove Working Variable to free memory
```

```
remove(D_WorkAge)

# Plot Sample densities to Compare Distributions over *SMOKE LEVEL*
D_WorkSMoke <- as_tibble(rbind(data.frame(LvlSmoke=as.vector(SArray[,,paste0("beta.Smoking.adj[", 1, "]
                               data.frame(LvlSmoke=as.vector(SArray[,,paste0("beta.Smoking.adj[", 2, "]
                               data.frame(LvlSmoke=as.vector(SArray[,,paste0("beta.Smoking.adj[", 3, "]
                               data.frame(LvlSmoke=as.vector(SArray[,,paste0("beta.Smoking.adj[", 4, "]


D_WorkSMoke %>%
  ggplot(aes(x = LvlSmoke))+
  geom_density(aes(x=LvlSmoke,group=Level, colour=Level), size=1.2)+
  labs(x = expression(list(beta[Smoke_][i])), y = "Density") +
  scale_color_manual(values = cbp2)+
  theme_bw()

# Remove Working Variable to free memory
remove(D_WorkSMoke)

# Plot Sample densities to Compare Distributions over *STD DEVIATION* and *PRECISION*
D_WorkStd <- as_tibble(rbind(data.frame(LvlStd=as.vector(SArray[,,"std.Age"]),Level="Age"),
                             data.frame(LvlStd=as.vector(SArray[,,"std.Smoking"]),Level="Smoking")))

D_WorkPrc <- as_tibble(rbind(data.frame(LvlStd=as.vector(SArray[,,"tau.Age"]),Level="Age"),
                             data.frame(LvlStd=as.vector(SArray[,,"tau.Smoking"]),Level="Smoking")))


P1 <- D_WorkStd %>%
  ggplot(aes(x = LvlStd))+
  geom_density(aes(x=LvlStd,group=Level, colour=Level), size=1.2)+
  labs(x = expression(list(sigma[Age]~","~sigma[Smoke])), y = "Density") +
  scale_color_manual(values = cbp2)+
  theme_bw()

P2 <- D_WorkPrc %>%
  ggplot(aes(x = LvlStd))+
  geom_density(aes(x=LvlStd,group=Level, colour=Level), size=1.2)+
  labs(x = expression(list(tau[Age]~","~tau[Smoke])), y = "Density") +
  scale_color_manual(values = cbp2)+
  theme_bw()

ggpubr::ggarrange(P1, P2, ncol = 1, nrow = 2)

# Remove Working Variable to free memory
remove(D_WorkStd, D_WorkPrc)


# Plot TracePlots * BETA.AGE *

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

L <- NSim-Burnin
```

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

# Build working dataframe

P <- list()

for (i in 1:5) {
  # Build working dataframe
  D_WorkTpAge <- data.frame(ValCh1=SArray[S,1,paste0("beta.Age.adj[", i, "]")],
                            ValCh2=SArray[S,2,paste0("beta.Age.adj[", i, "]")],
                            ValCh3=SArray[S,3,paste0("beta.Age.adj[", i, "]")],
                            ValCh4=SArray[S,4,paste0("beta.Age.adj[", i, "]")],
                            ValCh5=SArray[S,5,paste0("beta.Age.adj[", i, "]")])

  p <- D_WorkTpAge %>%
    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)+
    geom_line(aes(y=ValCh4, colour=4), size=0.8)+
    geom_line(aes(y=ValCh5, colour=5), size=0.8)+
    labs(y = eval(bquote(expression(beta[Age_][.(i)])))) +
    theme_bw()

  P <- c(P, list(p))
}

ggpubr::ggarrange(P[[1]]+theme(axis.text.x=element_blank(),
                               axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[2]]+theme(axis.text.x=element_blank(),
                               axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[3]]+theme(axis.text.x=element_blank(),
                               axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[4]]+theme(axis.text.x=element_blank(),
                               axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[5]]+theme(axis.text.x=element_blank(),
                               axis.title.x=element_blank(),
                               legend.position="none"), ncol = 3, nrow = 2)

# Remove Working Variable to free memory
remove(p, P, D_WorkTpAge)

# Plot TracePlots * BETA.SMOKING *

# Sampling 5000 points to generate "thinner" traceplots
set.seed(351)

L <- NSim-Burnin
```

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

# Build working dataframe

P <- list()

for (i in 1:4) {
  # Build working dataframe
  D_WorkTpSmok <- data.frame(ValCh1=SArray[S,1,paste0("beta.Smoking.adj[", i, "]")],
                             ValCh2=SArray[S,2,paste0("beta.Smoking.adj[", i, "]")],
                             ValCh3=SArray[S,3,paste0("beta.Smoking.adj[", i, "]")],
                             ValCh4=SArray[S,4,paste0("beta.Smoking.adj[", i, "]")],
                             ValCh5=SArray[S,5,paste0("beta.Smoking.adj[", i, "]")])

  p <- D_WorkTpSmok %>%
    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)+
    geom_line(aes(y=ValCh4, colour=4), size=0.8)+
    geom_line(aes(y=ValCh5, colour=5), size=0.8)+
    labs(y = eval(bquote(expression(beta[Smoke_][.(i)])))) +
    theme_bw()
  P <- c(P, list(p))
}

ggpubr::ggarrange(P[[1]]+theme(axis.text.x=element_blank(),
                               axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[2]]+theme(axis.text.x=element_blank(),
                               axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[3]]+theme(axis.text.x=element_blank(),
                               axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[4]]+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(p, P, D_WorkTpSmok)

# Plot TracePlots * BETA.0 *

# Sampling 5000 points to generate "thinner" traceplots
set.seed(761)

L <- NSim-Burnin

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

# Build working dataframe
D_WorkTpBeta0 <- data.frame(ValCh1=SArray[S,1,paste0("beta0.adj")],
```

```
                          ValCh2=SArray[S,2,paste0("beta0.adj")],
                          ValCh3=SArray[S,3,paste0("beta0.adj")],
                          ValCh4=SArray[S,4,paste0("beta0.adj")],
                          ValCh5=SArray[S,5,paste0("beta0.adj")])

P0 <- D_WorkTpBeta0 %>%
  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)+
  geom_line(aes(y=ValCh4, colour=4), size=0.8)+
  geom_line(aes(y=ValCh5, colour=5), size=0.8)+
  labs(x = "Iteration", y = expression(beta[0])) +
  theme_bw()

P0 + theme(axis.title.x=element_blank(),
           legend.position="none")

# Remove Working Variable to free memory
remove(D_WorkTpBeta0)

# Plot TracePlots * STD.AGE and STD.SMOKING *

# Sampling 5000 points to generate "thinner" traceplots
set.seed(390)

L <- NSim-Burnin

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

# Build working dataframe
D_WorkTpStdAge <- data.frame(ValCh1=SArray[S,1,paste0("std.Age")],
                             ValCh2=SArray[S,2,paste0("std.Age")],
                             ValCh3=SArray[S,3,paste0("std.Age")],
                             ValCh4=SArray[S,4,paste0("std.Age")],
                             ValCh5=SArray[S,5,paste0("std.Age")])

P1 <- D_WorkTpStdAge %>%
  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)+
  geom_line(aes(y=ValCh4, colour=4), size=0.8)+
  geom_line(aes(y=ValCh5, colour=5), size=0.8)+
  labs(x = "Iteration", y = expression(sigma[Age])) +
  theme_bw()

# Build working dataframe
D_WorkTpStdSmok <- data.frame(ValCh1=SArray[S,1,paste0("std.Smoking")],
                              ValCh2=SArray[S,2,paste0("std.Smoking")],
                              ValCh3=SArray[S,3,paste0("std.Smoking")],
                              ValCh4=SArray[S,4,paste0("std.Smoking")],
                              ValCh5=SArray[S,5,paste0("std.Smoking")])
```

```
P2 <- D_WorkTpStdSmok %>%
  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)+
  geom_line(aes(y=ValCh4, colour=4), size=0.8)+
  geom_line(aes(y=ValCh5, colour=5), size=0.8)+
  labs(x = "Iteration", y = expression(sigma[Smoke])) +
  theme_bw()

ggpubr::ggarrange(P1+theme(axis.title.x=element_blank(),
                           legend.position="none"),
                  P2+theme(axis.title.x=element_blank(),
                           legend.position="none"), ncol = 1, nrow = 2)

# Remove Working Variable to free memory
remove(D_WorkTpStdAge, D_WorkTpStdSmok)

# Plot TracePlots * BETA.AGE *

P <- list()

for (i in 1:5) {
  # Build working dataframe
  D_WorkAcfAge <- data.frame(ValCh1=SArray[,1,paste0("beta.Age.adj[", i, "]")])

  p <- ggAcf(D_WorkAcfAge$ValCh1, lag.max = 100)+
    labs(x = "Lag", y = eval(bquote(expression(beta[Age_][.(i)])))) +
    ggtitle(NULL)+
    theme_bw()
  P <- c(P, list(p))
}

ggpubr::ggarrange(P[[1]]+theme(axis.title.x=element_blank(),
                               axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5, size = 8)
                               legend.position="none"),
                  P[[2]]+theme(axis.title.x=element_blank(),
                               axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5, size = 8)
                               legend.position="none"),
                  P[[3]]+theme(axis.title.x=element_blank(),
                               axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5, size = 8)
                               legend.position="none"),
                  P[[4]]+theme(axis.title.x=element_blank(),
                               axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5, size = 8)
                               legend.position="none"),
                  P[[5]]+theme(axis.title.x=element_blank(),
                               axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5, size = 8)
                               legend.position="none"), ncol = 3, nrow = 2)

# Remove Working Variable to free memory
remove(p, P, D_WorkAcfAge)

# Plot ACF * BETA.SMOKING *
```

```r
# Group 1-4 - Elaborate a List of Graphs

P <- list()

for (i in 1:4) {
  # Build working dataframe
  D_WorkAcfSmok <- data.frame(ValCh1=SArray[,1,paste0("beta.Smoking.adj[", i, "]")])

  p <- ggAcf(D_WorkAcfSmok$ValCh1, lag.max = 100)+
    labs(x = "Lag", y = eval(bquote(expression(beta[Smoke_][.(i)])))) +
    ggtitle(NULL)+
    theme_bw()
  P <- c(P, list(p))
}

ggpubr::ggarrange(P[[1]]+theme(axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[2]]+theme(axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[3]]+theme(axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[4]]+theme(axis.title.x=element_blank(),
                               legend.position="none"), ncol = 2, nrow = 2)

# Remove Working Variable to free memory
remove(p, P, D_WorkAcfSmok)


# Plot TracePlots * BETA.0 *

# Build working dataframe
D_WorkAcfBeta0 <- data.frame(ValCh1=SArray[,1,paste0("beta0.adj")])

P1 <- ggAcf(D_WorkAcfBeta0$ValCh1, lag.max = 100)+
  labs(x = "Lag", y = expression(beta[0])) +
  ggtitle(NULL)+
  theme_bw()

P1+theme(axis.title.x=element_blank(),
         legend.position="none")

# Remove Working Variable to free memory
remove(D_WorkAcfBeta0)

# Comparison of Smoking Categories
SMokCat <- rbind(LungCanc_v0[["summary"]]["beta.Smoking.adj[1]",c(1:3,5,7)],
                 LungCanc_v0[["summary"]]["beta.Smoking.adj[4]",c(1:3,5,7)])
RNames <- c("beta.Smoking.adj[1]", "beta.Smoking.adj[4]")
df_Prt <- data.frame(Parameter = RNames)
df_Prt <- cbind(df_Prt, as_tibble(SMokCat))
df_Prt %>%
  kbl(booktabs = TRUE,
      caption = "Smoking Categories - Never Smoked vs. >20 cigarettes per day") %>%
```

```
  kable_styling(latex_options = "striped")

# Calculate Probability Increase between categories
ProbInc <- exp(SArray[,,"beta.Smoking.adj[4]"]-SArray[,,"beta.Smoking.adj[1]"])
m_Smoke <- mean(ProbInc)
sd_Smoke <- sd(ProbInc)
cat("\n---- Summary Statistics for Increase in Probability -----")
cat("\nMean : ",format(m_Smoke, digits = 2, nsmall = 3))
cat("\nStd.Dev. : ",format(sd_Smoke, digits = 2, nsmall = 4))
CI_Smoking <- qnorm(c(0.025,0.975), mean=m_Smoke, sd=sd_Smoke)
cat("\n95% Credible Region (", format(CI_Smoking[1], digits = 2, nsmall = 3),
    ",",format(CI_Smoking[2], digits = 2, nsmall = 3),")\n")
cat("\n>>> From this model, our belief of increase probability of death by lung cancer\n")
cat("    of someone who smokes >20 cigarettes per day vs. who never smoked is",format(m_Smoke, digits =
cat("\n    with corresponding 95% credible region of (",    format(CI_Smoking[1], digits = 2, nsmall =
    format(CI_Smoking[2], digits = 2, nsmall = 2),")\n")

# Setup - Read Data Diabetics
diab_data <- read.table(file="data/DiabetesDrugEffect.csv",header=TRUE,sep=",")
NDiab <- nrow(diab_data)

# Setup OpenBugs running parameters

NSim <- 30000   # No. of simulations for productio
NChain <- 5     # No. of chains for production
NThin <- 5      # n.thin parameter for production
Burnin <- 10000 # Burn-In parameter for production
Sz <- 5000      # Size of samples for trace/acf plots


### *************** M O D E L  -  1 ***************** ###

### ---- SETUP MODELS ---- ###

# MODEL 1 - Splitted
cat("
model{
for(i in 1:NDiab)
{
#StudyID  StudyN  diff  Sediff

diff[i]~dnorm(delta[i], tau[i]) #v.19
delta[i]~dnorm(theta, tau0)
tau[i] <- 1/pow(Sediff[i],2)

}

# Initial Priors - Verify
theta ~dnorm(-1.5, 30)  # v.19
sigma0 ~dgamma(2, 1)    # v.19
tau0 <- 1/pow(sigma0,2)

}", file="DiabetDrug_M1.txt")
```

```
### ---- PARAM SETTINGS ---- ###

# Setup Parameters
paramsM1=c("theta", "tau0", "sigma0", "delta")

### ---- INITIALIZATION PROCEDURES ---- ###

# Setup Initial Values for Model 1
init.funM1=function(){list(
  theta = rnorm(1, mean=-1.5, sd=sqrt(.3)),
  sigma0=rgamma(1, 2, 1))}  # v.19

### ---- PROCESS MODELS ON OPENBUGS ---- ###

# Run Open Bugs - Model 1
set.seed(2602)
model1.data <- list("StudyID", "StudyN", "diff", "Sediff","NDiab")
attach(diab_data)
DiabetDrug_M1=bugs(model1.data, init.funM1, paramsM1, model.file="DiabetDrug_M1.txt",
                   working.directory = ".", n.chains=NChain, n.iter=NSim, n.burnin=Burnin, n.thin=NThin
detach(diab_data)

### ---- ORGANIZE OUTPUTS ---- ###

# Get Simulation from OpenBugs - Model 1
DArrayM1= DiabetDrug_M1$sims.array    # Data Arrays
vname=attr(DArrayM1,"dimnames")[3][[1]]  # Variable Names


### ********** M O D E L  -  2 ********** ###

### ---- SETUP MODELS ---- ###

cat("
model{
for(i in 1:NDiab)
{
#StudyID  StudyN  diff  Sediff

diff[i]~dnorm(theta, tau[i])
tau[i] <- 1/(pow(sigma0,2)+pow(Sediff[i],2))
}

# Initial Priors - Verify
theta ~dnorm(-1.5, 30)  # v.19
sigma0 ~dgamma(2, 1)    # v.19

}", file="DiabetDrug_M2.txt")

### ---- PARAM SETTINGS ---- ###

# Setup Parameters
paramsM2=c("theta", "sigma0")
```

```
### ---- INITIALIZATION PROCEDURES ---- ###

# Setup Initial Values for Model 2
init.funM2=function(){list(
  theta = rnorm(1, mean=-1.5, sd=sqrt(.3)),
  sigma0=rgamma(1, 2, 1))}  # v.19

### ---- PROCESS MODELS ON OPENBUGS ---- ###

# Run Open Bugs - Model 2
set.seed(8421)
model2.data <- list("StudyID", "StudyN", "diff", "Sediff","NDiab")
attach(diab_data)
DiabetDrug_M2=bugs(model2.data, init.funM2, paramsM2, model.file="DiabetDrug_M2.txt",
                   working.directory = ".", n.chains=NChain, n.iter=NSim, n.burnin=Burnin, n.thin=NThin
detach(diab_data)

### ---- ORGANIZE OUTPUTS ---- ###

# Get Simulation from OpenBugs - Model 2
DArrayM2= DiabetDrug_M2$sims.array    # Data Arrays
vname=attr(DArrayM2,"dimnames")[3][[1]]  # Variable Names


# Print Summary statistics of parameters oif Interest
RNames <- rownames(DiabetDrug_M1[["summary"]][,c(1:3,5,7)]) # List of parameters
df_Prt <- data.frame(Parameter = RNames)
df_Prt <- cbind(df_Prt, as_tibble(DiabetDrug_M1[["summary"]][,c(1:3,5,7)]))
df_Prt %>%
  kbl(booktabs = TRUE,
      caption = "Model 1 - Difference between Drug A and B") %>%
  kable_styling(latex_options = "striped")


# Print Summary statistics of parameters oif Interest
RNames <- rownames(DiabetDrug_M2[["summary"]][,c(1:3,5,7)]) # List of parameters
df_Prt <- data.frame(Parameter = RNames)
df_Prt <- cbind(df_Prt, as_tibble(DiabetDrug_M2[["summary"]][,c(1:3,5,7)]))
df_Prt %>%
  kbl(booktabs = TRUE,
      caption = "Model 2 - Difference between Drug A and B") %>%
  kable_styling(latex_options = "striped")

# Plot TracePlots and ACF of *THETA *

# Sampling 5000 points to generate "thinner" traceplots
set.seed(3961)

L <- NSim - Burnin

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

```r
# Build working dataframe
D_WorkTpTheta <- data.frame(ValCh1=DArrayM1[S,1,paste0("theta")],
                            ValCh2=DArrayM1[S,2,paste0("theta")],
                            ValCh3=DArrayM1[S,3,paste0("theta")],
                            ValCh4=DArrayM1[S,4,paste0("theta")],
                            ValCh5=DArrayM1[S,5,paste0("theta")])

P1 <- D_WorkTpTheta %>%
  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)+
  geom_line(aes(y=ValCh4, colour=4), size=0.8)+
  geom_line(aes(y=ValCh5, colour=5), size=0.8)+
  labs(y = expression(theta)) +
  theme_bw()

P2 <- ggAcf(DArrayM1[,1,paste0("theta")], lag.max = 100)+
  labs(x = "Lag", y = expression(theta)) +
  ggtitle(NULL)+
  theme_bw()

ggpubr::ggarrange(P1+theme(axis.title.x=element_blank(),
                           legend.position="none"), P2, ncol = 1, nrow = 2)

# Remove Working Variable to free memory
remove(D_WorkTpTheta)

# Plot TracePlots * SIGMA0*

# Sampling 5000 points to generate "thinner" traceplots
set.seed(2391)

L <- NSim - Burnin

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

# Build working dataframe
D_WorkTpSigma0 <- data.frame(ValCh1=DArrayM1[S,1,paste0("sigma0")],
                             ValCh2=DArrayM1[S,2,paste0("sigma0")],
                             ValCh3=DArrayM1[S,3,paste0("sigma0")],
                             ValCh4=DArrayM1[S,4,paste0("sigma0")],
                             ValCh5=DArrayM1[S,5,paste0("sigma0")])

P1 <- D_WorkTpSigma0 %>%
  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)+
  geom_line(aes(y=ValCh4, colour=4), size=0.8)+
  geom_line(aes(y=ValCh5, colour=5), size=0.8)+
  labs(y = expression(sigma[0])) +
  theme_bw()
```

```
P2 <- ggAcf(DArrayM1[,1,paste0("sigma0")], lag.max = 100)+
  labs(x = "Lag", y = expression(sigma[0])) +
  ggtitle(NULL)+
  theme_bw()

ggpubr::ggarrange(P1+theme(axis.title.x=element_blank(),
                           legend.position="none"), P2, ncol = 1, nrow = 2)

# Remove Working Variable to free memory
remove(D_WorkTpSigma0)

# Plot TracePlots for Deltas

# Sampling 5000 points to generate "thinner" traceplots
set.seed(746)

L <- NSim - Burnin

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

# Group 1-4 - Elaborate a List of Graphs

P <- list()

for (i in 1:4) {
  # Build working dataframe
  D_WorkTpDelta <- data.frame(ValCh1=DArrayM1[S,1,paste0("delta[", i, "]")],
                              ValCh2=DArrayM1[S,2,paste0("delta[", i, "]")],
                              ValCh3=DArrayM1[S,3,paste0("delta[", i, "]")],
                              ValCh4=DArrayM1[S,4,paste0("delta[", i, "]")],
                              ValCh5=DArrayM1[S,5,paste0("delta[", i, "]")])
  p <- D_WorkTpDelta %>%
    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)+
    geom_line(aes(y=ValCh4, colour=4), size=0.8)+
    geom_line(aes(y=ValCh5, colour=5), size=0.8)+
    labs(y = eval(bquote(expression(delta[.(i)])))) +
    theme_bw()
  P <- c(P, list(p))
}

ggpubr::ggarrange(P[[1]]+theme(axis.text.x=element_blank(),
                               axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[2]]+theme(axis.text.x=element_blank(),
                               axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[3]]+theme(axis.text.x=element_blank(),
                               axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[4]]+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(p, P, D_WorkTpDelta)

# Plot TracePlots for Deltas

# Sampling 5000 points to generate "thinner" traceplots
set.seed(456)

L <- NSim - Burnin

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

# Group 5-8 - Elaborate a List of Graphs

P <- list()

for (i in 5:8) {
  # Build working dataframe
  D_WorkTpDelta <- data.frame(ValCh1=DArrayM1[S,1,paste0("delta[", i, "]")],
                              ValCh2=DArrayM1[S,2,paste0("delta[", i, "]")],
                              ValCh3=DArrayM1[S,3,paste0("delta[", i, "]")],
                              ValCh4=DArrayM1[S,4,paste0("delta[", i, "]")],
                              ValCh5=DArrayM1[S,5,paste0("delta[", i, "]")])
  p <- D_WorkTpDelta %>%
    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)+
    geom_line(aes(y=ValCh4, colour=4), size=0.8)+
    geom_line(aes(y=ValCh5, colour=5), size=0.8)+
    labs(y = eval(bquote(expression(delta[.(i)])))) +
    theme_bw()
  P <- c(P, list(p))
}

ggpubr::ggarrange(P[[1]]+theme(axis.text.x=element_blank(),
                               axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[2]]+theme(axis.text.x=element_blank(),
                               axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[3]]+theme(axis.text.x=element_blank(),
                               axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[4]]+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(p, P, D_WorkTpDelta)
```

```
# Plot TracePlots for Deltas

# Sampling 5000 points to generate "thinner" traceplots
set.seed(198)

L <- NSim - Burnin

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

# Group 9-12 - Elaborate a List of Graphs

P <- list()

for (i in 9:12) {
  # Build working dataframe
  D_WorkTpDelta <- data.frame(ValCh1=DArrayM1[S,1,paste0("delta[", i, "]")],
                              ValCh2=DArrayM1[S,2,paste0("delta[", i, "]")],
                              ValCh3=DArrayM1[S,3,paste0("delta[", i, "]")],
                              ValCh4=DArrayM1[S,4,paste0("delta[", i, "]")],
                              ValCh5=DArrayM1[S,5,paste0("delta[", i, "]")])
  p <- D_WorkTpDelta %>%
    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)+
    geom_line(aes(y=ValCh4, colour=4), size=0.8)+
    geom_line(aes(y=ValCh5, colour=5), size=0.8)+
    labs(y = eval(bquote(expression(delta[.(i)])))) +
    theme_bw()
  P <- c(P, list(p))
}

ggpubr::ggarrange(P[[1]]+theme(axis.text.x=element_blank(),
                               axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[2]]+theme(axis.text.x=element_blank(),
                               axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[3]]+theme(axis.text.x=element_blank(),
                               axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[4]]+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(p, P, D_WorkTpDelta)

# Plot ACF plots for Deltas

# Sampling 5000 points to generate "thinner" traceplots

# Group 1-4 - Elaborate a List of Graphs
```

```r
P <- list()

for (i in 1:4) {
  # Build working dataframe
  D_WorkTpDelta <- data.frame(ValCh1=DArrayM1[,1,paste0("delta[", i, "]")])

  p <- ggAcf(D_WorkTpDelta$ValCh1, lag.max = 100)+
    labs(x = "Lag", y = eval(bquote(expression(delta[.(i)])))) +
    ggtitle(NULL)+
    theme_bw()
  P <- c(P, list(p))
}

ggpubr::ggarrange(P[[1]]+theme(axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[2]]+theme(axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[3]]+theme(axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[4]]+theme(axis.title.x=element_blank(),
                               legend.position="none"), ncol = 2, nrow = 2)

# Remove Working Variable to free memory
remove(p, P, D_WorkTpDelta)

# Plot ACF plots for Deltas

# Sampling 5000 points to generate "thinner" traceplots

# Group 5-8 - Elaborate a List of Graphs

P <- list()

for (i in 5:8) {
  # Build working dataframe
  D_WorkTpDelta <- data.frame(ValCh1=DArrayM1[,1,paste0("delta[", i, "]")])

  p <- ggAcf(D_WorkTpDelta$ValCh1, lag.max = 100)+
    labs(x = "Lag", y = eval(bquote(expression(delta[.(i)])))) +
    ggtitle(NULL)+
    theme_bw()
  P <- c(P, list(p))
}

ggpubr::ggarrange(P[[1]]+theme(axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[2]]+theme(axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[3]]+theme(axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[4]]+theme(axis.title.x=element_blank(),
                               legend.position="none"), ncol = 2, nrow = 2)
```

```r
# Remove Working Variable to free memory
remove(p, P, D_WorkTpDelta)

# Plot ACF plots for Deltas

# Group 9-12 - Elaborate a List of Graphs

P <- list()

for (i in 9:12) {
  # Build working dataframe
  D_WorkTpDelta <- data.frame(ValCh1=DArrayM1[,1,paste0("delta[", i, "]")])

  p <- ggAcf(D_WorkTpDelta$ValCh1, lag.max = 100)+
    labs(x = "Lag", y = eval(bquote(expression(delta[.(i)])))) +
    ggtitle(NULL)+
    theme_bw()
  P <- c(P, list(p))
}

ggpubr::ggarrange(P[[1]]+theme(axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[2]]+theme(axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[3]]+theme(axis.title.x=element_blank(),
                               legend.position="none"),
                  P[[4]]+theme(axis.title.x=element_blank(),
                               legend.position="none"), ncol = 2, nrow = 2)
# Remove Working Variable to free memory
remove(p, P, D_WorkTpDelta)

# Plot TracePlots and ACF of *THETA *

# Sampling 5000 points to generate "thinner" traceplots
set.seed(3871)

L <- NSim - Burnin

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

# Build working dataframe
D_WorkTpTheta <- data.frame(ValCh1=DArrayM2[S,1,paste0("theta")],
                            ValCh2=DArrayM2[S,2,paste0("theta")],
                            ValCh3=DArrayM2[S,3,paste0("theta")],
                            ValCh4=DArrayM2[S,4,paste0("theta")],
                            ValCh5=DArrayM2[S,5,paste0("theta")])

P1 <- D_WorkTpTheta %>%
  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)+
  geom_line(aes(y=ValCh4, colour=4), size=0.8)+
```

```
  geom_line(aes(y=ValCh5, colour=5), size=0.8)+
  labs(y = expression(theta)) +
  theme_bw()

P2 <- ggAcf(DArrayM2[,1,paste0("theta")], lag.max = 100)+
  labs(x = "Lag", y = expression(theta)) +
  ggtitle(NULL)+
  theme_bw()

ggpubr::ggarrange(P1+theme(axis.title.x=element_blank(),
                           legend.position="none"), P2, ncol = 1, nrow = 2)

# Remove Working Variable to free memory
remove(D_WorkTpTheta)

# Plot TracePlots * SIGMA0*

# Sampling 5000 points to generate "thinner" traceplots
set.seed(8463)

L <- NSim - Burnin

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

# Build working dataframe
D_WorkTpSigma0 <- data.frame(ValCh1=DArrayM2[S,1,paste0("sigma0")],
                             ValCh2=DArrayM2[S,2,paste0("sigma0")],
                             ValCh3=DArrayM2[S,3,paste0("sigma0")],
                             ValCh4=DArrayM2[S,4,paste0("sigma0")],
                             ValCh5=DArrayM2[S,5,paste0("sigma0")])

P1 <- D_WorkTpSigma0 %>%
  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)+
  geom_line(aes(y=ValCh4, colour=4), size=0.8)+
  geom_line(aes(y=ValCh5, colour=5), size=0.8)+
  labs(y = expression(sigma[0])) +
  theme_bw()

P2 <- ggAcf(DArrayM2[,1,paste0("sigma0")], lag.max = 100)+
  labs(x = "Lag", y = expression(sigma[0])) +
  ggtitle(NULL)+
  theme_bw()

ggpubr::ggarrange(P1+theme(axis.title.x=element_blank(),
                           legend.position="none"), P2, ncol = 1, nrow = 2)

# Remove Working Variable to free memory
remove(D_WorkTpSigma0)

# Plot Sample densities to Compare Distributions of *THETA* and *SIGMA0*
```

```r
set.seed(9023)

L <- NSim-Burnin

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

D_WorkThetas <- as_tibble(rbind(data.frame(LvlStd=as.vector(DArrayM1[S,,paste0("theta")]), Model = "M1"
                                data.frame(LvlStd=as.vector(DArrayM2[S,,paste0("theta")]), Model = "M2"
D_WorkSigmas <- as_tibble(rbind(data.frame(LvlStd=as.vector(DArrayM1[S,,paste0("sigma0")]), Model = "M1
                                data.frame(LvlStd=as.vector(DArrayM2[S,,paste0("sigma0")]), Model = "M2


P1 <- D_WorkThetas %>%
  ggplot(aes(x = LvlStd))+
  geom_density(aes(x=LvlStd, group=Model, colour=Model), size=0.8)+
  labs(x = expression(list(theta[Model_][1]~"vs."~theta[Model_][2])), y = "Density") +
  scale_color_manual(values = cbp2)+
  theme_bw()

P2 <- D_WorkSigmas %>%
  ggplot(aes(x = LvlStd))+
  geom_density(aes(x=LvlStd, group=Model, colour=Model), size=0.8)+
  labs(x = expression(list((sigma[0])[Model_][1]~"vs."~(sigma[0])[Model_][2])), y = "Density") +
  scale_color_manual(values = cbp2)+
  theme_bw()

ggpubr::ggarrange(P1, P2, ncol = 1, nrow = 2)

# Remove Working Variable to free memory
remove(D_WorkThetas, D_WorkSigmas)


# Prepare Data - Get delta[1-12]
D3_Work <- as.data.frame(DiabetDrug_M1[["summary"]][4:15,c(1:3,5,7)])

# Generates Studies labels
StudyLst <- rapply(list(diab_data$StudyID), sprintf, fmt = "%02d", how = "replace")

# Collects C.I's of observed values from studies
CI_Study <- data.frame(ID = StudyLst[[1]],
                       lower = diab_data$diff-diab_data$Sediff,
                       estim = diab_data$diff,
                       upper = diab_data$diff+diab_data$Sediff,
                       Source = "Study")

# Collects C.R's from Summary report
CI_Estim <- data.frame(ID = StudyLst[[1]],
                       lower = D3_Work$`2.5%`,
                       estim = D3_Work$mean,
                       upper = D3_Work$`97.5%`,
                       Source = "Model-1")

# Organize data to generate graphs & analysis
```

```
D4_Work <- rbind(CI_Study, CI_Estim)

#  (i) Identifying the direction of change of estimates
# (ii) Identifying differences o length of CR and CI for delta and "diff"
P1 <- D4_Work %>%
  ggplot() +
  geom_point(aes(x=ID, y=estim, color = Source, shape = Source, size = 0.5)) +
  geom_errorbar(aes(x=ID, ymin=lower, ymax=upper,
                    color = Source), width = 1) +
  xlab("Study No.") +
  ylab(expression("C.R.+"~delta[i]~" and 'diff'")) +
  scale_color_manual(values = cbp2)+
  facet_wrap(~Source)+
  theme_bw()

P1 + theme(legend.position="none",
           axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5, size = 8))

# (i) Identifying the direction of posterior mean of delta and "diff"
P2 <- D4_Work %>%
  ggplot() +
  geom_point(aes(x=ID, y=estim, colour = Source, shape = Source), size=5) +
  xlab(expression("Study No.")) +
  ylab(expression(delta[i]~" and 'diff'")) +
  scale_color_manual(values = cbp2)+
  theme_bw()

P2 + theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5, size = 8))

remove(D3_Work, D4_Work, StudyLst)
```