Bayesian Data Analysis For The Slovenian Plebiscite

By
Budhinath Padhy

A Project
Submitted to the Faculty
Of
Worcester Polytechnic Institute
In partial fulfillment of the requirement for the
Degree of Master of Science
In
Applied Statistics

May 2011

Approved

Prof. Balgobin Nandram, Project Advisor

Bayesian Data Analysis for the Slovenian Plebiscite

ABSTRACT

Slovenia became an independent republic with its own constitution passed on December 23, 1991. The important step that led to the independence of Slovenia was the December 1990 plebiscite. It was at this plebiscite that the citizens of Slovenia voted for a sovereign and independent state. A public survey called Slovenian Public Opinion (SPO) survey was taken by the government of Slovenia for the plebiscite. The plebiscite counted 'YES voters' only those voters who attended and who voted for independence. Non-voters were counted as 'NO voters' and 'Don't Know' survey responses that could be thought of as missing data that was treated as 'YES' or 'NO'. Analysis of survey data is done using non-parametric fitting procedure, Bayesian ignorable nonresponse model and Bayesian nonignorable nonresponse model. Finally, a sensitivity analysis is conducted with respect to the different values of a prior parameter. The amazing estimates of the eventual plebiscite outcome show the validity our underlying models.

1 Introduction

Handling 'Don't Know' in the analysis of survey data is an an extremely important topic in many areas of Social Sciences, Medicine, Epidemiology, Engineering and other Natural Sciences. In this project we focus on a case of Slovenian Plebiscite. The Slovenian Public Opinion (SPO) Survey of November/December 1990 was used by the Government of Slovenia to prepare for the plebiscite. The plebiscite counted 'YES voters' only those voters who attended and who voted for independence. Non-voters were counted as 'NO voters' and 'Don't Know' survey responses that could be thought of as missing data that must be treated as 'YES' or 'NO'.

The entire Slovenian population is 2.00 million with 1.46 million eligible voters; more than 1.28 million voted for independence from Yugoslavia. To prepare for the results of the plebiscite, the Slovenian government collected data on its possible outcome by inserting questions into an administration of the SPO survey. The SPO survey which was carried out 4 weeks prior to the plebiscite, is a face-to-face survey with a standard three-stage (Yes, No, 'Don't Know') item. The number of participants were 2085 Slovenians in 139 randomly chosen primary sampling units of similar size with 3 secondary sampling units per primary sampling unit, and 5 individuals within each secondary sampling units.

Although the SPO survey asked questions about many aspects of Slovenian life as well as a number of questions about independence and constitution, we will focus on the three questions concerning the plebiscite.

1. Are you in favor of Slovenian's secession from Yugoslavia?
2. Will you attend plebiscite?
3. Are you in favor of Slovenian independence?

The answers were recorded as Yes, No, or Don't Know (DK). The results of 2074 responses on these three questions are given in Table 1.

A common parameter of interest is the proportion θ of the population planning to attend and vote in favor of independence. The true value of θ is .885.

The overview of the sections are as follows. Section 2 will use non-parametric fitting procedure to count these 'Don't Know Cases' as 'Yes' or 'No'. In sections 3 and 4, we will analyze the estimate using ignorable nonresponse model and nonignorable nonresponse model respectively. Sensitivity analysis of the estimate with respect to prior parameter will be discussed in section 5. Finally, overall conclusion, references and the R-code will follow

Table 1: The available Case

		Independence		
Secession	Attendance	Yes	No	Don't Know
Yes	Yes	1191	8	21
	No	8	0	4
	Don't Know	107	3	9
No	Yes	158	68	29
	No	7	14	3
	Don't Know	18	43	31
Don't Know	Yes	90	2	109
	No	1	2	25
	Don't Know	19	8	96

the analysis sections.

2 Non-Parametric Estimates

In this section we will use the sample data, from 2074 responses, to estimate the proportion θ, taking into account of the Don't Know (DK) responses. This is a Missing at Random (MAR) non-parametric model.

We take the whole available case of Secession, Attendance and Independence that have value of 'Yes', 'No' and 'DK'. For the sake of analysis, we will denote these cells occupied by this model as Y/N/D. Thus, we will have 3*3*3=27 cells in total. The ordering is always as SAI where S stands for Succession, A for Attendance, and I for Independence. The following table shows the places of these cells.

Table 2: Cells for available case in terms of Y/N/D

		Independence		
Secession	Attendance	Yes	No	DK
Yes	Yes	YYY(a_1)	YYN(a_2)	YYD
	No	YNY(a_3)	YNN(a_4)	YND
	DK	YDY	YDN	YDD
No	Yes	NYY(a_5)	NYN(a_6)	NYD
	No	NNY(a_7)	NNN(a_8)	NND
	DK	NDY	NDN	NDD
DK	Yes	DYY	DYN	DYD
	No	DNY	DNN	DND
	DK	DDY	DDN	DDD

The approach we will take to estimate θ is by accommodating the DK responses in a proportionate manner to the complete case where all (Secession, Attendance and Independence) responses are either Yes or No.

We categorize the the whole available array into the complete array (that contains Y/N only responses) denoted by $a_1..a_8$, and other arrays that contain DK elements. These arrays are conveniently written in the following subsequent tables.

2.1 The Complete Array

As mentioned before, we denote the complete array, i.e. the YYY, YYN, YNY, YNN, NYY, NYN, NNY, NNN values of the available array by $a_1..a_8$ respectively. Thus the complete array can be written in a simplified form as given in Table 3.

Table 3: The Complete Array Formula

		Independence		
Secession	Attendance	Yes	No	
Yes	Yes	a_1	a_2	
	No	a_3	a_4	
No	Yes	a_5	a_6	
	No	a_7	a_8	
				1454

Using the values of $(a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8)$ = (1191, 8, 8, 0, 158, 68, 7, 14) respectively, the complete array can be written in terms of the values as :

Table 4: The Complete Array values

		Independence		
Secession	Attendance	Yes	No	
Yes	Yes	1191	8	
	No	8	0	
No	Yes	158	68	
	No	7	14	
				1454

2.2 Allocating the Independence DKs

Our first other array is the Don't Know array associated with the Independence variable. The cells associated in the available array are (YYD, YND, NYD, NND), and has the values of (21, 4, 29, 3) respectively. Our goal is to take each of these cells and allocate in terms of cells that contain only Y/N. For instance the value of the cell in YYD need to be filled in proportion to their weights in YYY and YYN. Thus value of YYD can be allocated as $(\frac{YYY}{(YYY+YYN)}, \frac{YYN}{(YYY+YYN)}) * YYD.$

In other words, using the non-parametric fitting procedure, we need to impute the elements given the Independence DK values (21, 4, 29, 3) as shown in the table below.

The formula for imputing is given below:

Table 5: Formula for allocating the Independence DKs

		Independence		
Secession	Attendance	Yes	No	DK
Yes	Yes	$\frac{a_1}{(a_1+a_2)} * 21$	$\frac{a_2}{(a_1+a_2)} * 21$	21
	No	$\frac{a_3}{(a_3+a_4)} * 4$	$\frac{a_4}{(a_3+a_4)} * 4$	4
No	Yes	$\frac{a_5}{(a_5+a_6)} * 29$	$\frac{a_6}{(a_5+a_6)} * 29$	29
	No	$\frac{a_7}{(a_7+a_8)} * 3$	$\frac{a_8}{(a_7+a_8)} * 3$	3
				57

Using the values of $a_1..a_8$, we get after simplification the following table:

Table 6: Values of the independence DKs

		Independence		
Secession	Attendance	Yes	No	DK
Yes	Yes	21	0	21
	No	4	0	4
No	Yes	20	9	29
	No	1	2	3
				57

We could use similar techniques to populate other tables that follow.

2.3 Allocating the Attendance DKs

Using the non-parametric fitting procedure, we impute these unknown values as given in the following table.

Table 7: Formula for allocating the Attendance DKs

		Independence		
Secession	Attendance	Yes	No	DK
Yes	Yes	$\frac{a_1}{(a_1+a_3)} * 107$	$\frac{a_2}{(a_2+a_4)} * 3$	
	No	$\frac{a_3}{(a_1+a_3)} * 107$	$\frac{a_4}{(a_2+a_4)} * 3$	
	DK	107	3	110
No	Yes	$\frac{a_5}{(a_5+a_7)} * 18$	$\frac{a_6}{(a_6+a_8)} * 43$	
	No	$\frac{a_7}{(a_5+a_7)} * 18$	$\frac{a_8}{(a_6+a_8)} * 43$	
	DK	18	43	61
				171

Using the values of $a_1..a_8$, we get after simplification the following table:

Table 8: Values of the Attendance DKs

		Independence		
Secession	Attendance	Yes	No	DK
Yes	Yes	106	3	
	No	1	0	
	DK	107	3	110
No	Yes	17	36	
	No	1	7	
	DK	18	43	61
				171

2.4 Allocating the Secession DKs

Using the non-parametric fitting procedure, we impute these unknown values as given in the following table.

Table 9: Formula for allocating the Secession DKs

		Independence		
Secession	Attendance	Yes	No	DK
Yes	Yes	$\frac{a_1}{(a_1+a_5)} * 90$	$\frac{a_2}{(a_2+a_6)} * 2$	
	No	$\frac{a_3}{(a_3+a_7)} * 1$	$\frac{a_4}{(a_4+a_8)} * 2$	
No	Yes	$\frac{a_5}{(a_1+a_5)} * 90$	$\frac{a_6}{(a_2+a_6)} * 2$	
	No	$\frac{a_7}{(a_3+a_7)} * 1$	$\frac{a_8}{(a_4+a_8)} * 2$	
DK	Yes	90	2	
	No	1	2	
				95

Using the values of $a_1..a_8$, we get after simplification the following table:

Table 10: Values of the Secession DKs

		Independence		
Secession	Attendance	Yes	No	DK
Yes	Yes	79	0	
	No	1	0	
No	Yes	11	2	
	No	0	2	
DK	Yes	90	2	
	No	1	2	
				95

2.5 Allocating the Attendance-Independence DKs

Using the non-parametric fitting procedure, we impute these unknown values as given in the following table.

Table 11: Formula for allocating the Attendance-Independence DKs

		Independence		
Secession	Attendance	Yes	No	DK
Yes	Yes	$\frac{a_1}{(a_1+a_2+a_3+a_4)} * 9$	$\frac{a_2}{(a_1+a_2+a_3+a_4)} * 9$	
	No	$\frac{a_3}{(a_1+a_2+a_3+a_4)} * 9$	$\frac{a_4}{(a_1+a_2+a_3+a_4)} * 9$	
	DK			9
No	Yes	$\frac{a_5}{(a_5+a_6+a_7+a_8)} * 31$	$\frac{a_6}{(a_5+a_6+a_7+a_8)} * 31$	
	No	$\frac{a_7}{(a_5+a_6+a_7+a_8)} * 31$	$\frac{a_8}{(a_5+a_6+a_7+a_8)} * 31$	
	DK			31
				40

Using the values of $a_1..a_8$, we get after simplification the following table:

Table 12: Values of the Attendance-Independence DKs

		Independence		
Secession	Attendance	Yes	No	DK
Yes	Yes	9	0	
	No	0	0	
	DK			9
No	Yes	20	8	
	No	1	2	
	DK			31
				40

2.6 Allocating the Secession-Independence DKs

Using the non-parametric fitting procedure, we impute these unknown values as given in the following table.

Table 13: Formula for allocating the Secession-Independence DKs

		Independence		
Secession	Attendance	Yes	No	DK
Yes	Yes	$\frac{a_1}{(a_1+a_2+a_5+a_6)} * 109$	$\frac{a_2}{(a_1+a_2+a_5+a_6)} * 109$	
	No	$\frac{a_3}{(a_3+a_4+a_7+a_8)} * 25$	$\frac{a_4}{(a_3+a_4+a_7+a_8)} * 25$	
No	Yes	$\frac{a_5}{(a_1+a_2+a_5+a_6)} * 109$	$\frac{a_6}{(a_1+a_2+a_5+a_6)} * 109$	
	No	$\frac{a_7}{(a_3+a_4+a_7+a_8)} * 25$	$\frac{a_8}{(a_3+a_4+a_7+a_8)} * 25$	
DK	Yes			109
	No			25
				134

Using the values of $a_1..a_8$, we get after simplification the following table:

Table 14: Values of the Secession-Independence DKs

		Independence		
Secession	Attendance	Yes	No	DK
Yes	Yes	91	1	
	No	7	0	
No	Yes	12	5	
	No	6	12	
DK	Yes			109
	No			25
				134

2.7 Allocating the Secession-Attendance DKs

Using the non-parametric fitting procedure, we impute these unknown values as given in the following table.

Table 15: Formula for allocating the Secession-Attendance DKs

		Independence		
Secession	Attendance	Yes	No	DK
Yes	Yes	$\frac{a_1}{(a_1+a_3+a_5+a_7)} * 19$	$\frac{a_2}{(a_2+a_4+a_6+a_8)} * 8$	
	No	$\frac{a_3}{(a_1+a_3+a_5+a_7)} * 19$	$\frac{a_4}{(a_2+a_4+a_6+a_8)} * 8$	
No	Yes	$\frac{a_5}{(a_1+a_3+a_5+a_7)} * 19$	$\frac{a_6}{(a_2+a_4+a_6+a_8)} * 8$	
	No	$\frac{a_7}{(a_1+a_3+a_5+a_7)} * 19$	$\frac{a_8}{(a_2+a_4+a_6+a_8)} * 8$	
DK	DK	19	8	
				27

Using the values of $a_1..a_8$, we get after simplification the following table:

Table 16: Values of the Secession-Attendance DKs

		Independence		
Secession	Attendance	Yes	No	DK
Yes	Yes	17	1	
	No	0	0	
No	Yes	2	6	
	No	0	1	
DK	Yes	19	8	
				27

2.8 Allocating the Secession-Attendance-Independence DKs

Using the non-parametric fitting procedure, we impute these unknown values as given in the following table.

Table 17: Formula for allocating the Secession-Attendance-Independence DKs

		Independence		
Secession	Attendance	Yes	No	DK
Yes	Yes	$\frac{a_1}{\sum_{i=1}^{n} a_i} * 96$	$\frac{a_2}{\sum_{i=1}^{n} a_i} * 96$	
	No	$\frac{a_3}{\sum_{i=1}^{n} a_i} * 96$	$\frac{a_4}{\sum_{i=1}^{n} a_i} * 96$	
No	Yes	$\frac{a_5}{\sum_{i=1}^{n} a_i} * 96$	$\frac{a_6}{\sum_{i=1}^{n} a_i} * 96$	
	No	$\frac{a_7}{\sum_{i=1}^{n} a_i} * 96$	$\frac{a_8}{\sum_{i=1}^{n} a_i} * 96$	
DK	DK			96
				96

Using the values of $a_1..a_8$, we get after simplification the following table:

Table 18: Values of the Secession-Attendance-Independence DKs

		Independence		
Secession	Attendance	Yes	No	DK
Yes	Yes	79	1	
	No	1	0	
No	Yes	10	4	
	No	0	1	
DK	DK			96
				96

2.9 Computing θ

To compute the proportion, (θ), of voters who are planning to attend and vote in favor of independence, first we need to add all individual tables 1-8 as computed in the preceding section. We eventually get the following table, after all DKs has been absorbed in the all Y/N cells.

Table 19: Computation of Theta

		Independence		
Secession	Attendance	Yes	No	
Yes	Yes	1191+21+...+79=1593	8+0+...+1=13	
	No	8+4+...+1=21	0+0+...+0=0	
No	Yes	158+20+...+10=251	68+9+...+4=138	
	No	7+1+...+0=17	14+2+...+1=41	
				2074

From the table, the number of voters who attended and voted in favor of independence is given by: $\theta = (1593 + 251)/2074 = 0.889$

2.10 Results and conclusion

The results show that the proportion of voters who planning to attend and vote in favor of independence is theta = 0.889. Also it shows that the percent explained by complete cases is 70.11% and the percent explained by missing data is 29.89%.

3 Ignorable Nonresponse Model

In the previous chapter we have used non-parametric fitting procedure to compute the missing data and to estimate θ. In this section, we will use an ignorable nonresponse model that in turn uses Gibbs sampling techniques to compute the missing data and θ, the proportion of voters who are planning to attend and vote in favor of independence.

3.1 Notations

As in nonparametric fitting procedure case, we will deal with the complete (observed) data and missing(unobserved) data in terms of 4×2 categorical tables. Let the 8 cells of each of these tables be denoted by a vector with elements j=1..8 read row-wise from left to right and then top to down. Let $I_{jl} = 1$ if the l^{th} voter falls in the j^{th} cell and 0 otherwise. Also, let $J_{sl} = 1$ if l^{th} voter falls in the table s(s=1: complete case; s=2..8 missing data tables whose rows or columns or combination of rows/columns follow some distributions), and 0 otherwise.

Let p_j be the probability that a voter belongs to the cell (j) a table, and let π_{sj} be the probability that a voter belongs to the s^{th} table, given the cell status (j). The ignorable model assumes that $\pi_{sj} = \pi_s$, i.e. the probability that an individual belongs to a s^{th} table is independent of the cell status (j). Then,

$$I_l|p \overset{iid}{\sim} \text{Multinomial}(1,\ p),$$

where $\sum_{j=1}^{8} p_j = 1,\ p_j \geq 0$.

We take p as the prior density from the Dirichlet distributions:

$$p \sim Dirichlet(1, ..., 1),$$

where $\sum_{j=1}^{8} p_j = 1,\ p_j \geq 0$.

3.2 Finding the posterior density

For the ignorable nonresponse model, we take

$$J_l|\pi \overset{iid}{\sim} \text{Multinomial}(1, \pi)$$

Then the augmented likelihood for $p, \pi, y_{mis}|y_{obs}$ is proportional to

$$\frac{n!}{\prod_{s=1}^{8}\prod_{j=1}^{8} {y_j}^{(s)}!} \prod_{j=1}^{8} (\pi_s p_j)^{y_j(s)}$$

where ${y_j}^{(s)}$ is the data in the j^{th} cell of the s^{th} table.

3.3 Inference about π

For the ignorable model a posterior π and p are independent.

Hence inference about the parameter π can be obtained easily from the above expression:

$$\pi|p, y \sim Dirichlet(\sum_{j=1}^{8} {y_j}^{(1)} + 1, ..., \sum_{j=1}^{8} {y_j}^{(8)} + 1)$$

which simplifies to

$$\pi|p, y \sim Dirichlet(1455, 58, 172, 96, 41, 135, 28, 97)$$

This is independent of p, since we know the sum of data for each individual table.

3.4 Inference about p

Similarly, inference about the parameter p can be obtained as a conditional probability:

$$p|\pi, y \sim Dirichlet(\sum_{s=1}^{8} {y_1}^{(s)} + 1, ..., \sum_{s=1}^{8} {y_8}^{(s)} + 1)$$

3.5 Inference about Missing Data in Table 2(with Independence Don't Know data array as constraint)

Since the missing data follow some kind of constraints, we can easily infer about these missing data using the appropriate Binomial or Multinomial model.

For example, let us analyze the constraints for the missing data for the table 2, the table containing the Independence Don't Know data. The missing data ${y_1}^{(2)}$ and ${y_2}^{(2)}$ add up to 21, ${y_3}^{(2)}$ and ${y_4}^{(2)}$ add up to 4, ${y_5}^{(2)}$ and ${y_6}^{(2)}$ add up to 29 and ${y_7}^{(2)}$ and ${y_8}^{(2)}$ add up to 3. Now

$$y^{(2)}|p, y_1, \pi \propto \frac{1}{\prod_{j=1}^{8} {y_j}^{(2)}!} \prod_{j=1}^{8} (\pi_2 p_j)^{y_j(2)}$$

$$\propto \frac{(\pi_2 p_1)^{y_1(2)} (\pi_2 p_2)^{y_2(2)}}{{y_1}^{(2)}! {y_2}^{(2)}!}$$

$$\propto \frac{(p_1)^{y_1(2)} (p_2)^{21 - y_1(2)}}{{y_1}^{(2)}! (21 - y_1^{(2)})!}$$

$$\sim Binomial(21, \frac{p_1}{p_1 + p_2})$$

Hence,

$$y_1^{(2)}, y_2^{(2)} | p, y, \pi \sim Multinomial(21, \frac{p_1}{p_1 + p_2}, \frac{p_2}{p_1 + p_2})$$

Other missing data in this table 2, follow a Multinomial distribution as given below:

$$y_3^{(2)}, y_4^{(2)} | p, y, \pi \sim Multinomial(4, \frac{p_3}{p_3 + p_4}, \frac{p_4}{p_3 + p_4})$$

$$y_5^{(2)}, y_6^{(2)} | p, y, \pi \sim Multinomial(29, \frac{p_5}{p_5 + p_6}, \frac{p_6}{p_5 + p_6})$$

$$y_7^{(2)}, y_8^{(2)} | p, y, \pi \sim Multinomial(3, \frac{p_7}{p_7 + p_8}, \frac{p_8}{p_7 + p_8})$$

3.6 Inference about Missing Data in Table 3(with Attendance Don't Know data array as constraint)

In a similar manner, we could see that the distribution of missing data in other tables 3 to 8, i.e. the tables containing missing data for Attendance Don't Know data, Sccession Don't Know data, Don't Know data taken as a pair and Don't Know data taken as a triplet follow a Multinomial distribution too. The missing data for table 3 have the following distributions:

$$y_1^{(3)}, y_3^{(3)} | p, y, \pi \sim Multinomial(107, \frac{p_1}{p_1 + p_3}, \frac{p_3}{p_1 + p_3})$$

$$y_2^{(3)}, y_4^{(3)} | p, y, \pi \sim Multinomial(3, \frac{p_2}{p_2 + p_4}, \frac{p_4}{p_2 + p_4})$$

$$y_5^{(3)}, y_7^{(3)} | p, y, \pi \sim Multinomial(18, \frac{p_5}{p_5 + p_7}, \frac{p_7}{p_5 + p_7})$$

$$y_6^{(3)}, y_8^{(3)} | p, y, \pi \sim Multinomial(43, \frac{p_6}{p_6 + p_8}, \frac{p_8}{p_6 + p_8})$$

3.7 Inference about Missing Data in Table 4(with Secession Don't Know data array as constraint)

$$y_1^{(4)}, y_5^{(4)} | p, y, \pi \sim Multinomial(90, \frac{p_1}{p_1 + p_5}, \frac{p_5}{p_1 + p_5})$$

$$y_2^{(4)}, y_6^{(4)} | p, y, \pi \sim Multinomial(2, \frac{p_2}{p_2 + p_6}, \frac{p_6}{p_2 + p_6})$$

$$y_3^{(4)}, y_7^{(4)} | p, y, \pi \sim Multinomial(1, \frac{p_3}{p_3 + p_7}, \frac{p_7}{p_3 + p_7})$$

$$y_4^{(4)}, y_8^{(4)} | p, y, \pi \sim Multinomial(2, \frac{p_4}{p_4 + p_8}, \frac{p_8}{p_4 + p_8})$$

3.8 Inference about Missing Data in Table 5(with Attendance-Independence Don't Know data array as constraint)

$$y_1^{(5)}, y_2^{(5)}, y_3^{(5)}, y_4^{(5)} | p, y, \pi \sim Multinomial(9, \frac{p_1}{\sum_{j=1}^{4} p_j}, \frac{p_2}{\sum_{j=1}^{4} p_j}, \frac{p_3}{\sum_{j=1}^{4} p_j}, \frac{p_4}{\sum_{j=1}^{4} p_j})$$

$$y_5^{(5)}, y_6^{(5)}, y_7^{(5)}, y_8^{(5)} | p, y, \pi \sim Multinomial(31, \frac{p_5}{\sum_{j=5}^{8} p_j}, \frac{p_6}{\sum_{j=5}^{8} p_j}, \frac{p_7}{\sum_{j=5}^{8} p_j}, \frac{p_8}{\sum_{j=5}^{8} p_j})$$

3.9 Inference about Missing Data in Table 6(with Secession-Independence Don't Know data array as constraint)

$$y_1^{(6)}, y_2^{(6)}, y_5^{(6)}, y_6^{(6)} | p, y, \pi \sim Mult(109, \frac{p_1}{\sum_{j=1,2,5,6} p_j}, \frac{p_2}{\sum_{j=1,2,5,6} p_j}, \frac{p_5}{\sum_{j=1,2,5,6} p_j}, \frac{p_6}{\sum_{j=1,2,5,6} p_j}$$

$$y_3^{(6)}, y_4^{(6)}, y_7^{(6)}, y_8^{(6)} | p, y, \pi \sim Mult(25, \frac{p_3}{\sum_{j=3,4,7,8} p_j}, \frac{p_4}{\sum_{j=3,4,7,8} p_j}, \frac{p_7}{\sum_{j=3,4,7,8} p_j}, \frac{p_8}{\sum_{j=3,4,7,8} p_j})$$

3.10 Inference about Missing Data in Table 7(with Secession-Attendance Don't Know data array as constraint)

$$y_1^{(7)}, y_3^{(7)}, y_5^{(7)}, y_7^{(7)} | p, y, \pi \sim Mult(19, \frac{p_1}{\sum_{j=1,3,5,7} p_j}, \frac{p_3}{\sum_{j=1,3,5,7} p_j}, \frac{p_5}{\sum_{j=1,3,5,7} p_j}, \frac{p_7}{\sum_{j=1,3,5,7} p_j})$$

$$y_2^{(7)}, y_4^{(7)}, y_6^{(7)}, y_8^{(7)} | p, y, \pi \sim Mult(8, \frac{p_2}{\sum_{j=2,4,6,8} p_j}, \frac{p_4}{\sum_{j=2,4,6,8} p_j}, \frac{p_6}{\sum_{j=2,4,6,8} p_j}, \frac{p_8}{\sum_{j=2,4,6,8} p_j})$$

3.11 Inference about Missing Data in Table 8(with Secession-Attendance-Independence Don't Know data array as constraint)

$$y_1^{(8)}, y_2^{(8)}, y_3^{(8)}, y_4^{(8)}, y_5^{(8)}, y_6^{(8)}, y_7^{(8)}, y_8^{(8)} | p, y, \pi$$

$$\sim Multinomial(96, \frac{p_1}{\sum_{j=1}^{8} p_j}, \frac{p_2}{\sum_{j=1}^{8} p_j}, \frac{p_3}{\sum_{j=1}^{8} p_j}, \frac{p_4}{\sum_{j=1}^{8} p_j}, \frac{p_5}{\sum_{j=1}^{8} p_j}, \frac{p_6}{\sum_{j=1}^{8} p_j}, \frac{p_7}{\sum_{j=1}^{8} p_j}, \frac{p_8}{\sum_{j=1}^{8} p_j})$$

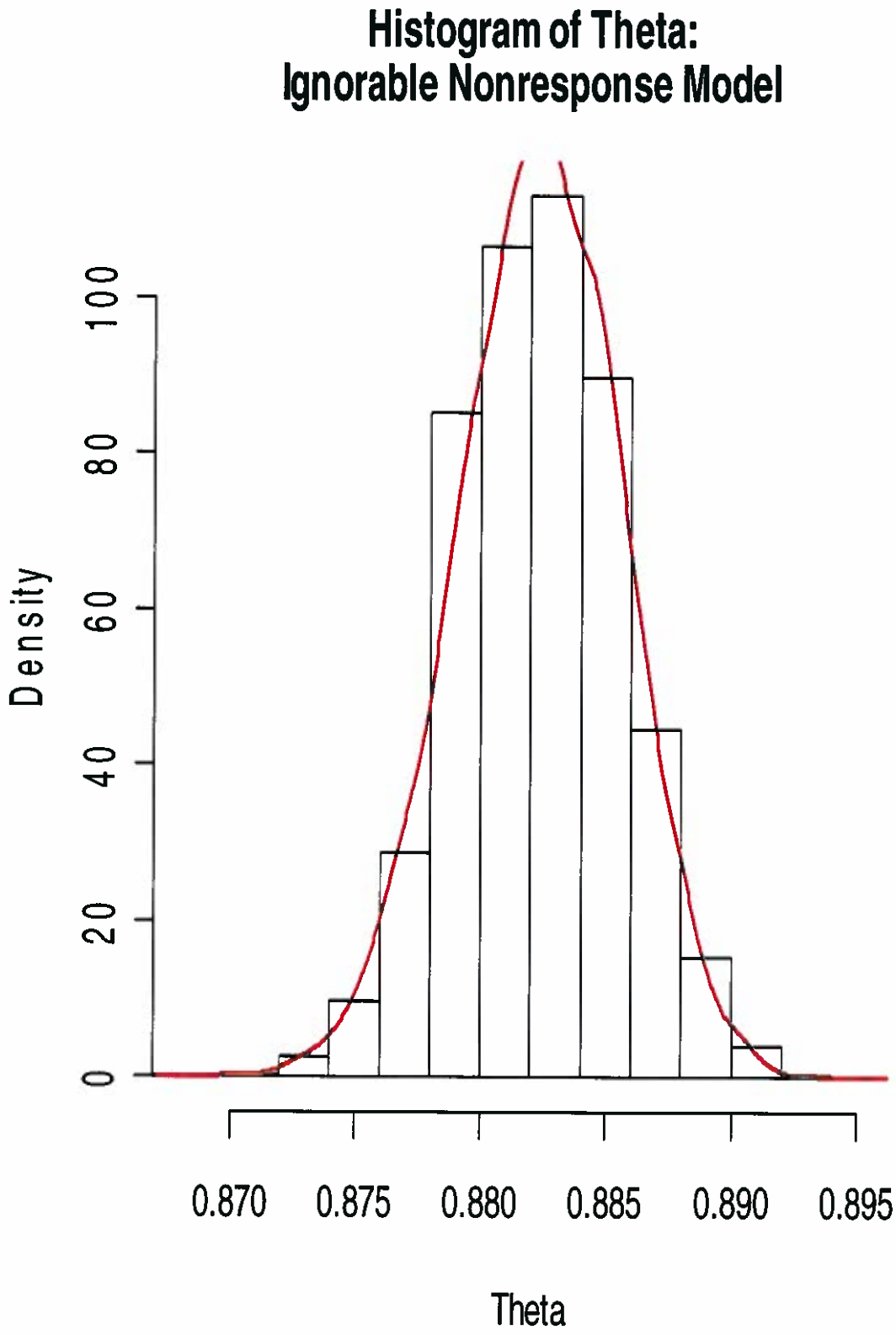
3.12 Simulation and Gibbs Sampling

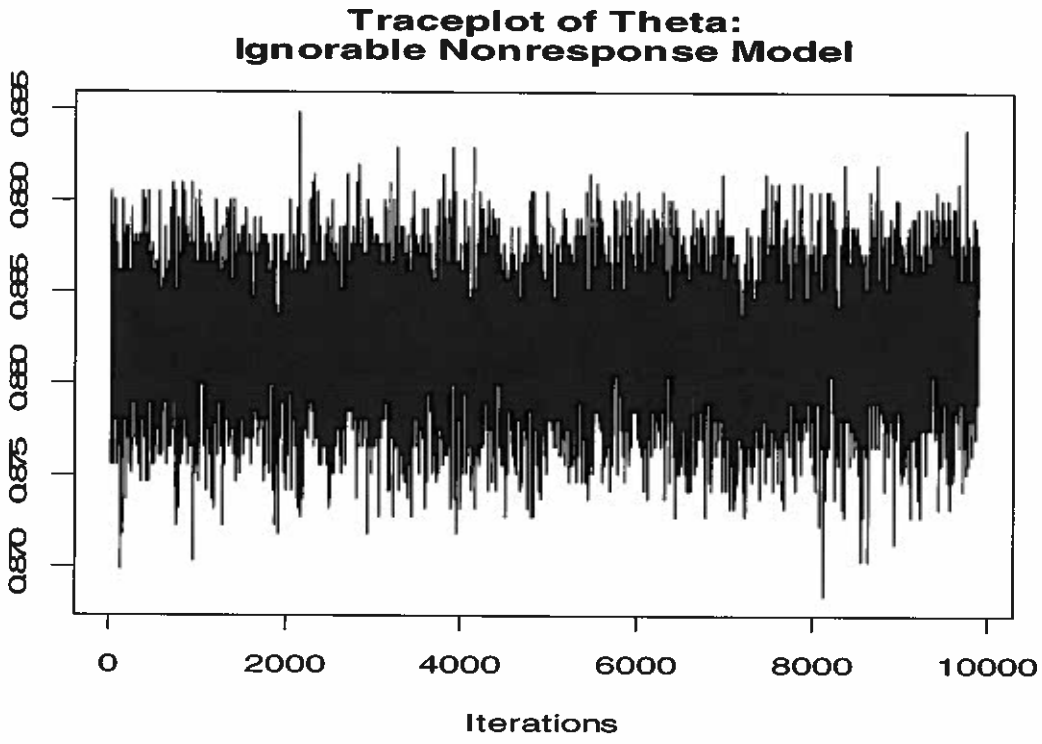
We use three-stage Gibbs sampler that creates a Markov Chain from a joint distribution. Initially the parameters π (probability for each table) and p (Probability for each cell) were populated using Dirichlet distributions. Once inside the simulation loop, these probabilities were used to compute the missing data inside the tables with their individual constraints. These missing data in conjunction with drawing from posterior Dirichlet distributions gives the next level of cell probabilities. This process is continued until the specified number of simulation ends. The whole simulation results were stored in a vector for further processing.

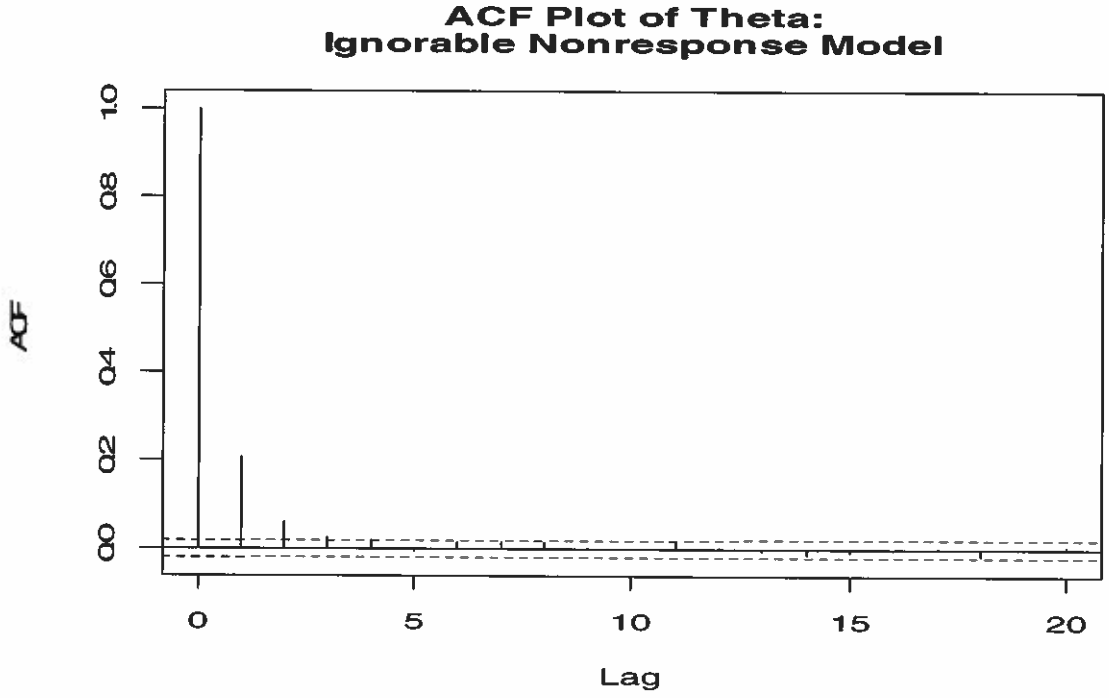
3.13 Computation of θ

Once we get the vector containing the simulation results, we compute θ, the proportion of voters who are planning to attend and vote in favor of independence. After 10000 simulations we get the value as 0.882 (which of course changes from simulations to simulations). But this is very close (0.889) to that obtained using the non-parametric approach of imputing the missing data inside the tables.

A histogram of theta along with its kernel density is shown. Also shown is the trace plot of MCMC object for theta. Finally, the autocorrelation diagram of theta shows a rapid convergence during the lags 1 to 20.. The **90%** credible interval is found to be **(0.877, 0.888).**


Histogram of Theta:
Ignorable Nonresponse Model
Density
0
20
40
60
80
100
0.870
0.875
0.880
0.885
0.890
0.895
Theta


Traceplot of Theta:
Ignorable Nonresponse Model
0.870
0.875
0.880
0.885
0.890
0.895
0
2000
4000
6000
8000
10000
Iterations


ACF Plot of Theta:
Ignorable Nonresponse Model
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
Lag

4 Nonignorable Nonresponse Model

4.1 Notations

As in the ignorable nonresponse model, for the nonignorable nonresponse model we will deal with the complete (observed) data and missing(unobserved) data in terms of $\mathbf{4 \times 2}$ categorical tables. Let the 8 cells of each of these tables be denoted by a vector with elements j=1..8 read row-wise from left to right and then top to down. Let $\boldsymbol{I_{jl} = 1}$ if the $\boldsymbol{l^{th}}$ voter falls in the $\boldsymbol{j^{th}}$ cell and 0 otherwise. Also, let $\boldsymbol{J_{sl} = 1}$ if $\boldsymbol{l^{th}}$ voter falls in the table s(s=1: complete case; s=2..8 missing data tables whose rows or columns or combination of rows/columns follow some distributions), and 0 otherwise.

Let $\boldsymbol{p_j}$ be the probability that a voter belongs to the cell $\boldsymbol{(j)}$ a table, and let $\boldsymbol{\pi_{sj}}$ be the probability that a voter belongs to the $\boldsymbol{s^{th}}$ table, given the cell status $\boldsymbol{(j)}$. In the nonignorable model $\boldsymbol{\pi_{sj}}$, i.e. the probability that an individual belongs to a $\boldsymbol{s^{th}}$ table is dependent of the cell status $\boldsymbol{(j)}$. Then,

$$\boldsymbol{I_l|p \overset{iid}{\sim} \text{Multinomial}(1,\ p)},$$

where $\boldsymbol{\sum_{j=1}^{8} p_j = 1,\ p_j \geq 0}$.

We take $\boldsymbol{p}$ as the prior density from the Dirichlet distributions:

$$\boldsymbol{p \sim Dirichlet(1, ..., 1)},$$

where $\boldsymbol{\sum_{j=1}^{8} p_j = 1,\ p_j \geq 0}$.

4.2 Finding the posterior density

For the nonignorable nonresponse model, we take

$$\boldsymbol{J_l|\{I_{jl} = 1, I_{j'l} = 0, j \neq j', \pi_j\} \overset{iid}{\sim} \text{Multinomial}(1, \pi_j)}$$

Then the augmented likelihood for $\boldsymbol{p, \pi, y_{mis}|y_{obs}}$ is proportional to

$$\boldsymbol{\prod_{s=1}^{8} \prod_{j=1}^{8} \frac{(\pi_j{}^{(s)})^{y_j{}^{(s)}}}{y_j{}^{(s)}!} \prod_{j=1}^{8} (p_j)^{\sum_{s=1}^{8} y_j{}^{(s)}}},$$

where $\boldsymbol{y_j{}^{(s)}}$ is the data in the $\boldsymbol{j^{th}}$ cell of the $\boldsymbol{s^{th}}$ table.

Note that for the nonignorable model, the parameters $\boldsymbol{\pi_j}$ and $\boldsymbol{p_j}$ are not identifiable.

If we assume π_j comes from a common distribution with known parameters, we could be able to estimate them. We assume that:

$$\pi_j|\mu,\tau \sim Dirichlet(\mu_1\tau, \mu_2\tau, ..., \mu_8\tau)$$

where $\sum_{s=1}^{8} {\pi_j}^{(s)} = 1,\ {\pi_j}^{(s)} \geq 0.$

As the parameters μ and τ are not identifiable, a natural way to proceed is to attempt to use some of the data already observed, like the data from the ignorable nonresponse model.

4.3 Estimation of μ and τ

We estimate μ as the mean of the probability table obtained in ignorable model:

$$\mu^{(s)} = mean(P[, s]),$$

where $\sum_{1}^{8} \mu^{(s)} = 1.$

To estimate τ, first we find the variance-covariance matrix of P and set it to the formula for variance and covariances. If

$$({\pi_j}^{(1)}, {\pi_j}^{(2)}, ..., {\pi_j}^{(8)}) \sim Dirichlet(\mu^{(1)}\tau, \mu^{(2)}\tau, ..., \mu^{(8)}\tau)$$

then,

$$E[{\pi_j}^{(s)}] = \mu^{(s)}/\tau$$

$$Var[{\pi_j}^{(s)}] = \mu^{(s)}(1 - \mu^{(s)})/(\tau_{ss} + 1)$$

$$Cov[{\pi_j}^{(s)}, {\pi_j}^{(s')}] = -\mu^{(s)}\mu^{(s')}/(\tau_{ss'} + 1), s \neq s'$$

Next we estimate τ by taking the geometric mean of all the solved τs from above:

$$\tau = [\prod_{s=1}^{8} \prod_{s'=1}^{8} \tau_{ss'}]^{(1/64)}$$

Then the joint posterior density of the parameters π, p, μ and τ and the latent variable y_{mis} given y_{obs} is proportional to:

$$[\prod_{s=1}^{8} \prod_{j=1}^{8} \frac{(p_j {\pi_j}^{(s)})^{{y_j}^{(s)}}}{{y_j}^{(s)}!}][\prod_{j=1}^{8} \frac{\prod_{s=1}^{8} ({\pi_j}^{(s)})^{\mu^{(s)}\tau - 1}}{D(\mu\tau)}]$$

4.4 Inference about p and π

The inference about the parameter p can be obtained as a conditional probability:

$$p|y \sim Dirichlet(\sum_{s=1}^{8} y_1{}^{(s)} + 1, ..., \sum_{s=1}^{8} y_8{}^{(s)} + 1)$$

Similarly, the inference about the π_j is given by,

$$\pi|\mu, \tau, y \sim Dirichlet(y_j{}^{(1)} + \mu^{(1)}\tau, ..., y_j{}^{(8)} + \mu^{(8)}\tau)$$

with independent over $j = 1..8$.

4.5 Inference about Missing Data in Tables

Like the ignorable nonresponse model, the missing data for nonignorable nonresponse model follows exactly same kind of constraints. We can easily infer about these missing data using the appropriate Binomial or Multinomial model. The only thing to keep in mind that the π_j for a table is dependent on the cell status j.

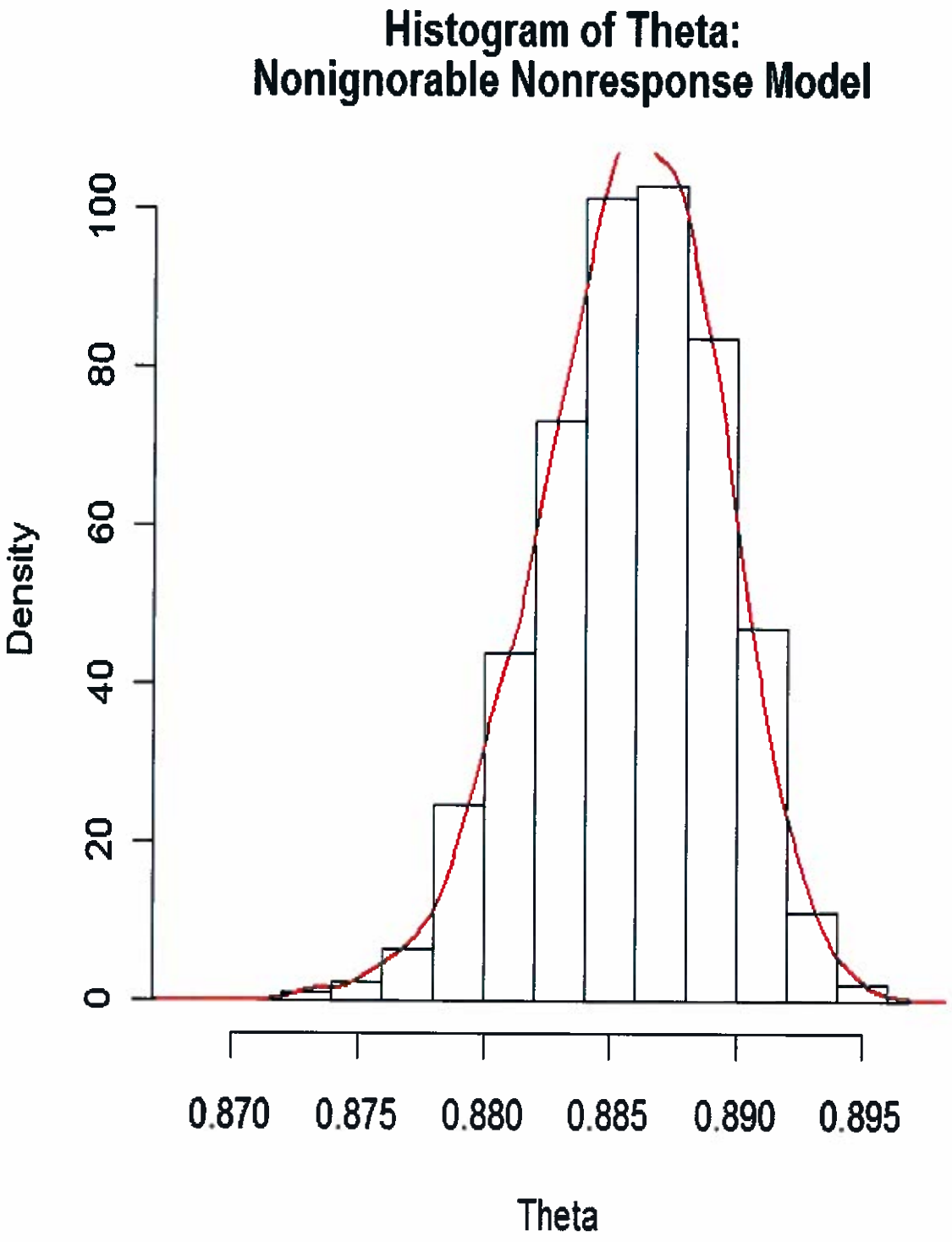
4.6 Simulation and Gibbs Sampling

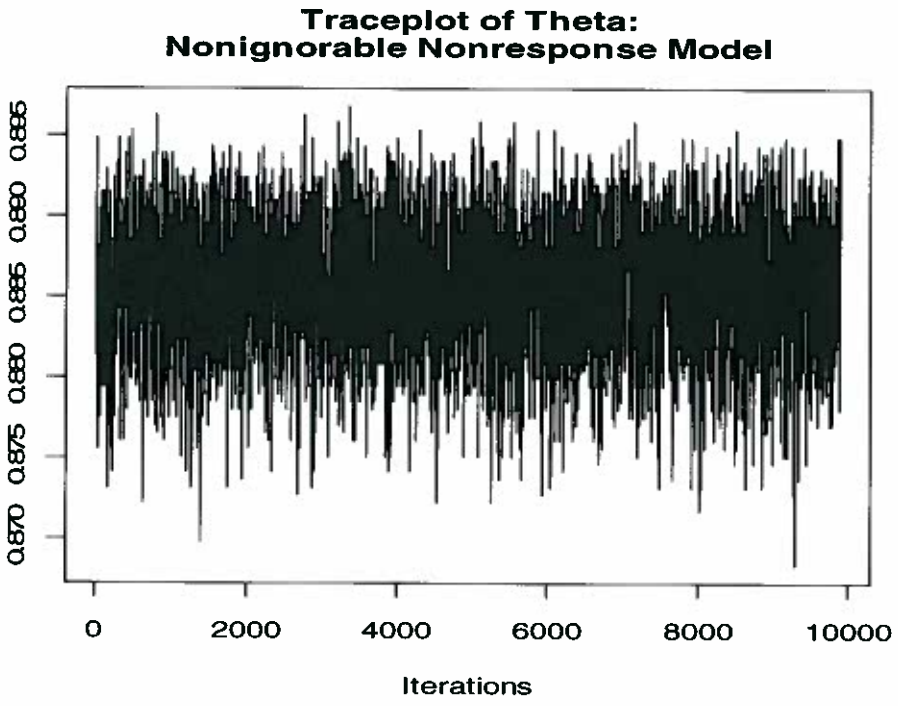
We use three-stage Gibbs sampler that creates a Markov Chain from a joint distribution. Initially the parameters π (probability for each table) and p (probability for each cell) were populated using Dirichlet distributions. Once inside the simulation loop, these probabilities were used to compute the missing data inside the tables with their individual constraints. These missing data in conjunction with drawing from posterior Dirichlet distributions gives the next level of cell probabilities. This process is continued until the specified number of simulation ends. The whole simulation results were stored in a vector for further processing.

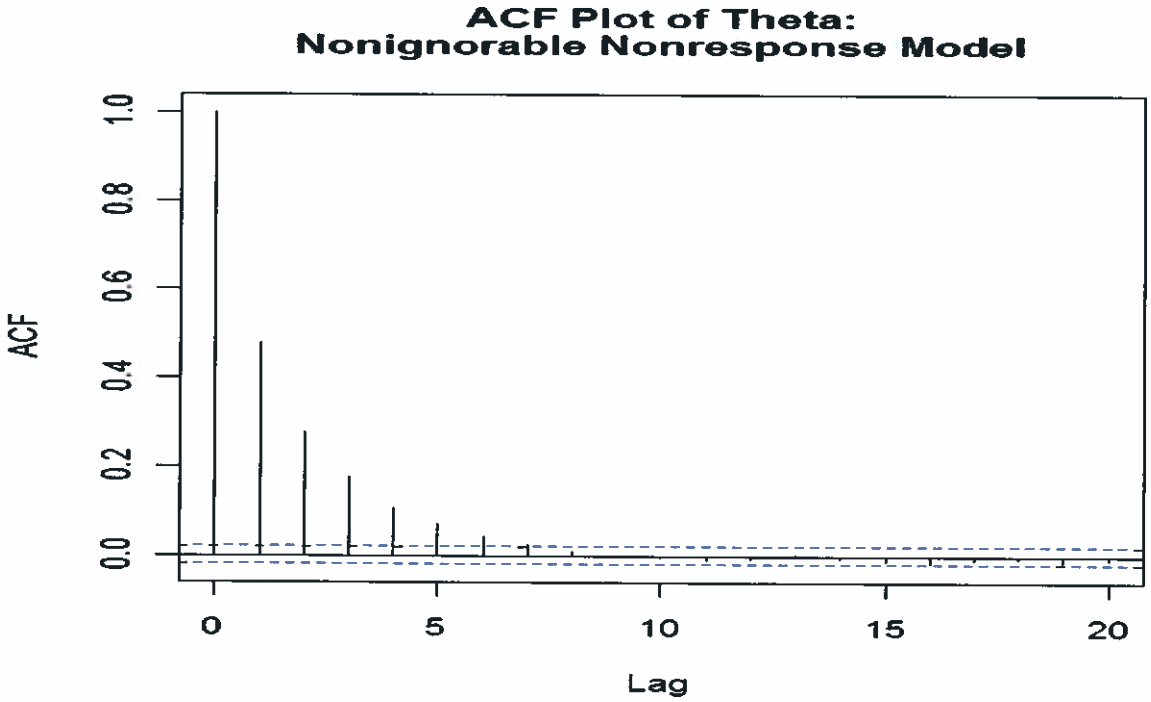
4.7 Computation of θ

Once we get the vector containing the simulation results, we compute θ, the proportion of voters who are planning to attend and vote in favor of independence. After 10000 simulations we get the value as Theta for nonignorable Model = 0.885 (which of course changes from simulations to simulations). But this is very close (0.889) to that obtained using the non-parametric approach of imputing the missing data inside the tables, and close to the ignorable nonresponse model.

A histogram of theta along with its kernel density is shown. Also shown is the trace plot of MCMC object for theta. Finally, the autocorrelation diagram of theta shows a rapid convergence during the lags 1 to 20. The **90%** credible interval is found to be $\mathbf{(0.879, 0.891)}$.


Histogram of Theta:
Nonignorable Nonresponse Model
Density
0
20
40
60
80
100
0.870
0.875
0.880
0.885
0.890
0.895
Theta


Traceplot of Theta:
Nonignorable Nonresponse Model
0.870
0.875
0.880
0.885
0.890
0.895
0
2000
4000
6000
8000
10000
Iterations


ACF Plot of Theta:
Nonignorable Nonresponse Model
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
Lag

5 Sensitivity Analysis

As described in the above section, for the $\pi_j{}^{(s)}$ we center the nonignorable nonresponse model on the ignorable nonresponse model:

$$\pi_j|\mu,\tau \sim Dirichlet(\mu_1\tau,\mu_2\tau,...,\mu_8\tau)$$

where the parameter τ tells us about the closeness of non-ignorable nonresponse model to the ignorable nonresponse model. For example, if τ is large, the π_j will be similar and converge to the π of a ignorable nonresponse model, and if τ is small, the π_js will be different.

In this section we will take a uniform prior on μ and vary τ to study sensitive analysis. The joint conditional posterior density of μ and τ, $p(\mu^{(s)},\tau|\pi_j, j=1...8)$ is

$$p(\mu^{(s)},\tau|\pi_j, j=1...8) \propto \frac{\prod_{s=1}^{8}(\delta_s)^{\mu^{(s)}\tau}}{\{D(\mu\tau)\}^8},$$

$\sum_1^8 \mu^{(s)} = 1, \mu^{(s)} \geq 0, s = 1..8, \tau > 0$ *where*

$$\delta_s = \prod_{j=1}^{8}\prod_{s=1}^{8}\pi_j^{(s)} \text{ and } D(\mu\tau) = \frac{\prod_{s=1}^{8}\Gamma(\mu^{(s)}\tau)}{\Gamma(\tau)}$$

is the Dirichlet function.

By letting $\mu^{[s]}$ denote the vector of all components of μ except $\mu^{(s)}$, we have the posterior density

$$p(\mu^{(s)}|\mu^{[s]},\tau,\pi_j, j=1...8) \propto \frac{\delta_s^{\mu^{(s)}\tau}}{\{\Gamma(\mu^{(s)}\tau)\}^8}\frac{\delta_8^{\mu^{(8)}\tau}}{\{\Gamma(\mu^{(8)}\tau)\}^8}$$

where $0 \leq \mu^{(s)} \leq 1 - \sum_{s'=1,s'\neq s}^{7}\mu^{(s')}, s = 1..7.$

A grid was used to draw a sample from $p(\mu^{(s)}|\mu^{[s]},\tau,\pi_j, j=1...8)$. Dividing the range of $\mu^{(s)}, (0, 1-\sum_{s'=1,s'\neq s}^{7}\mu^{(s')})$, into 100 interval of equal width we form pmf of $\mu^{(s)}, s = 1..7$. Next, a cdf is computed, from which we draw a uniform deviate in these intervals. Finally, $\mu^{(8)}$ is computed by taking $\mu^{(8)} = 1 - \sum_{j=1}^{7}\mu^{(s)}$.

5.1 Simulation and Gibbs Sampling

We extend the three-stage nonignorable nonresponse Gibbs sampler by including a function that computes the μ for a given τ using the joint conditional distribution of $\mu^{(s)}$ given τ. All other functions of the nonignorable non-response remains same.

5.2 Computation of θ

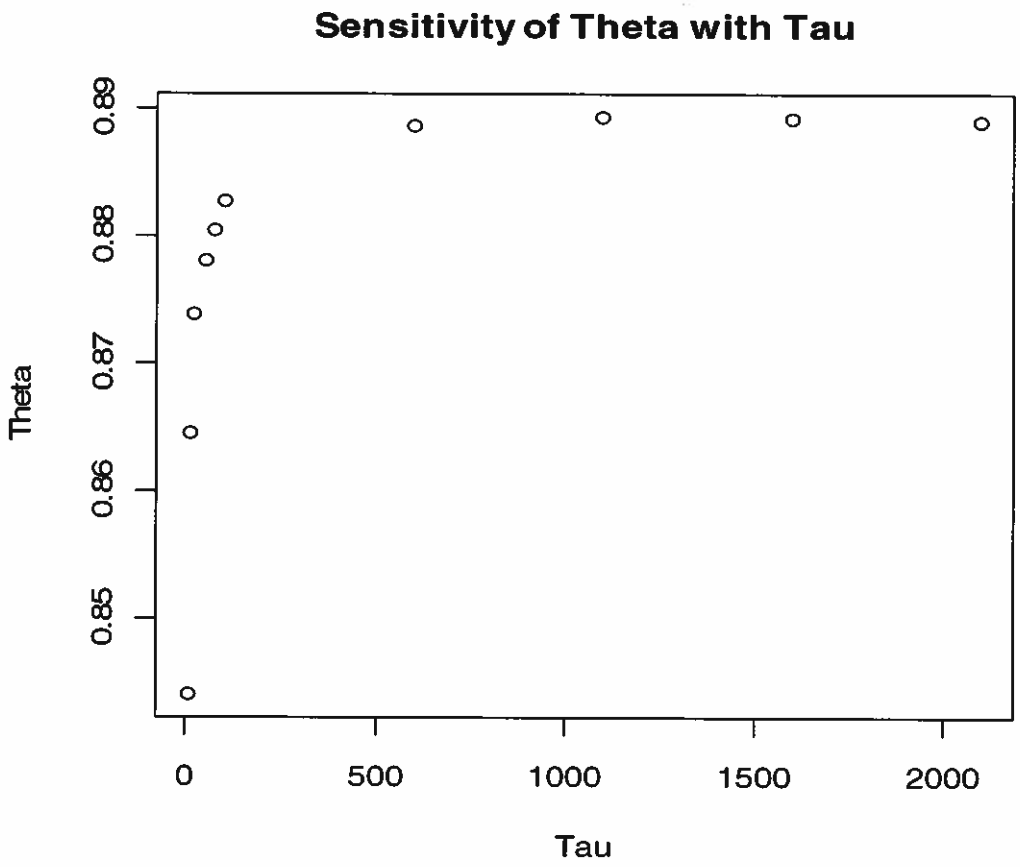
Once we get the vector containing the simulation results, we compute θ, the proportion of voters who are planning to attend and vote in favor of independence. We have taken τ to vary from as low as 5 to as high as 2100. After 10000 simulations, we get the value as 0.889, for a τ value of 2100. Note that this is exactly the same value (0.889), that obtained using the non-parametric approach of imputing the missing data inside the tables, close to the ignorable non-response model as well as close to nonignorable non-response model.

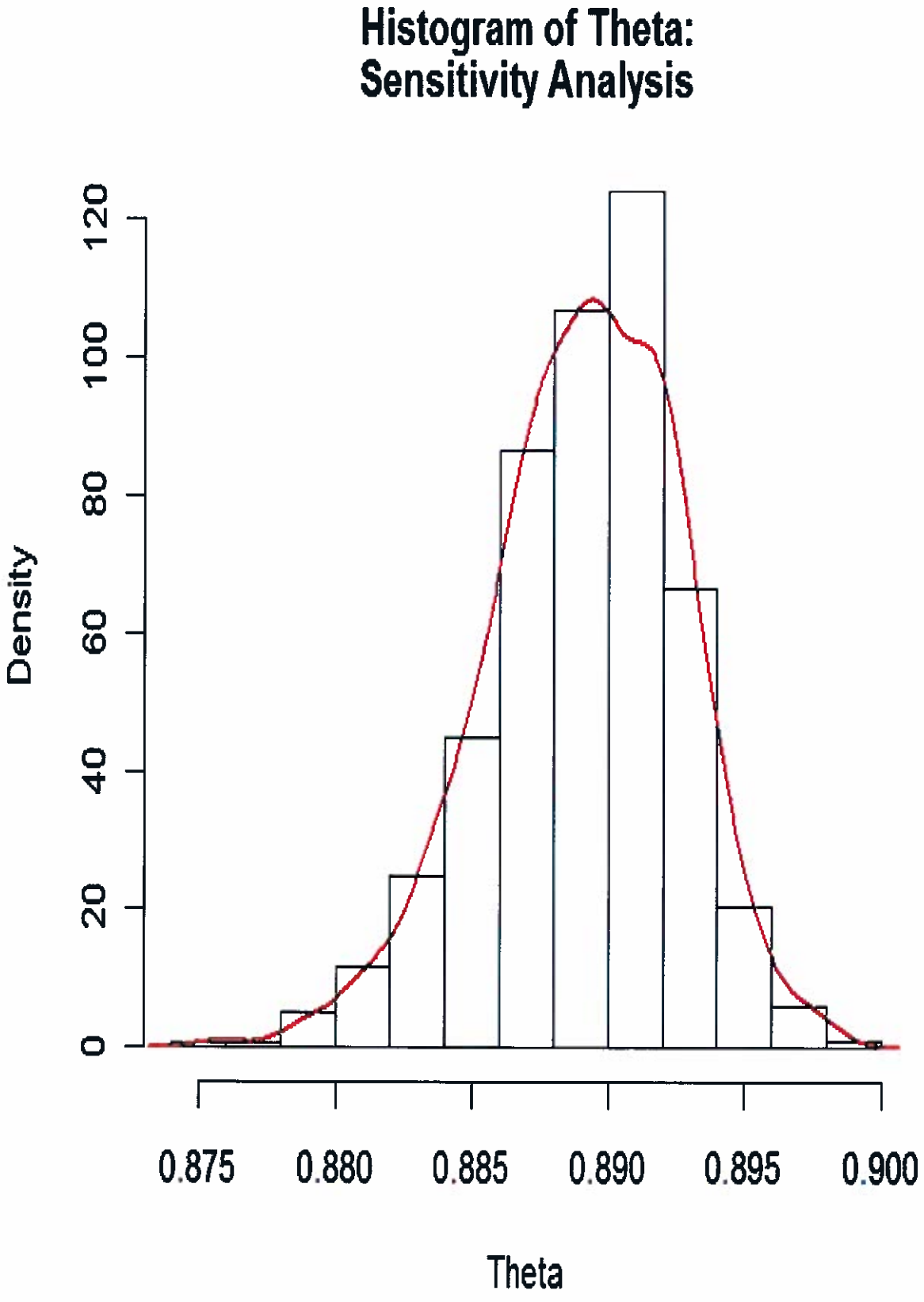
A histogram of theta along with its kernel density, a trace plot of the MCMC object for theta and the autocorrelation diagram of theta showing a rapid convergence during the lags 1 to 20 are included. When $\tau = 2100$, the 90% credible interval of θ computed to be **(0.883, 0.894).**

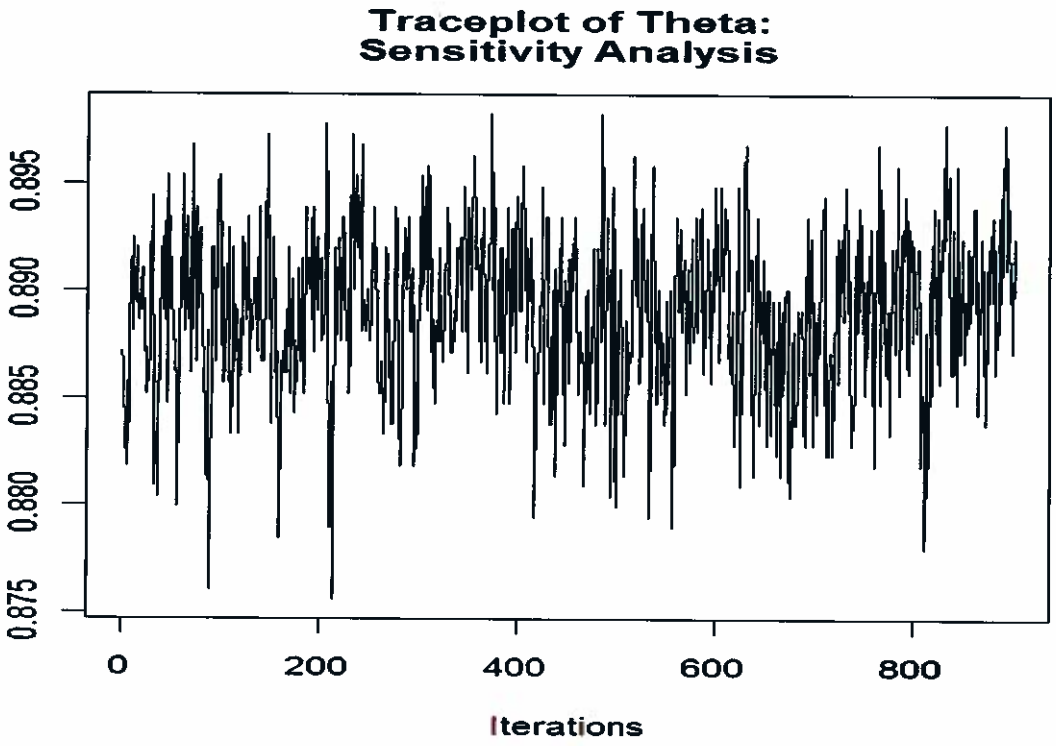
By varying τ from very low values of 5 to high values of 2100, we see that the θ is increasing and eventually stabilizes after $\tau > 600$. The following table of values of θ vs. τ as well as the attached diagram shows this feature implying that our model is robust with the choice of the prior μ and choice of τ.

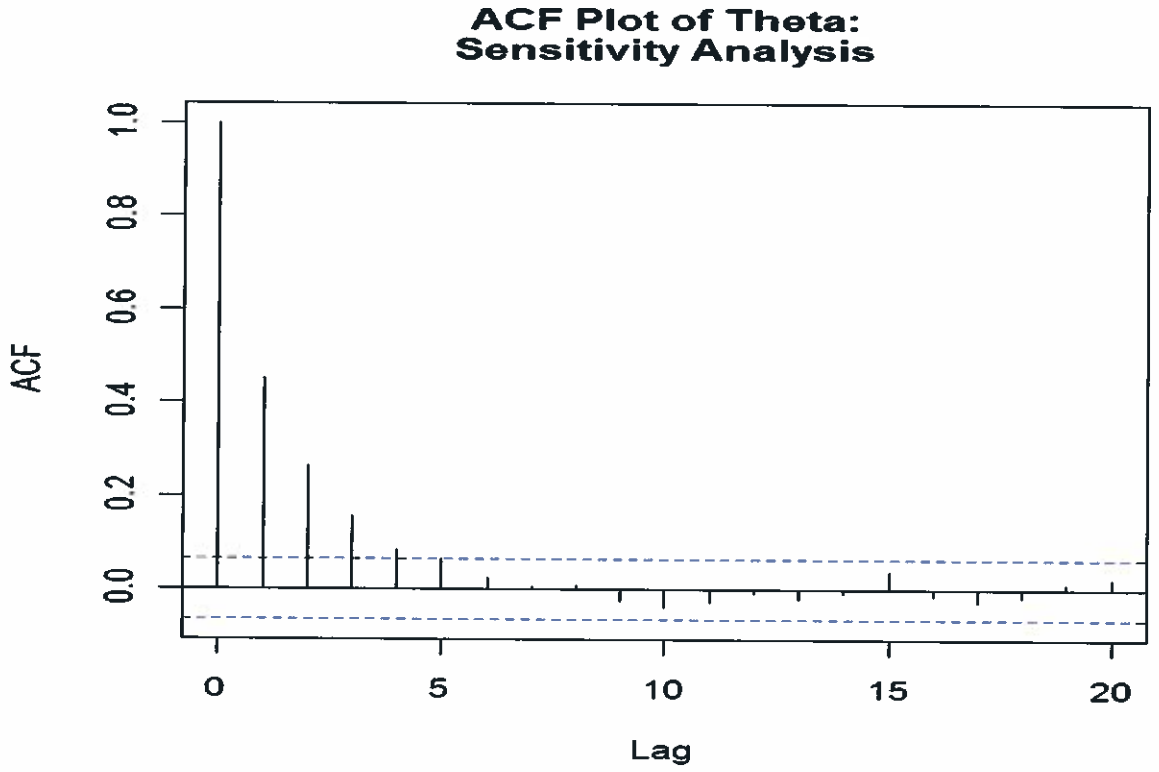
Table 20: Sensitivity of θ with τ

τ	θ	SD	90% credible Interval
5	0.844	0.016	(0.814, 0.870)
10	0.865	0.012	(0.844, 0.880)
20	0.879	0.008	(0.860, 0.883)
50	0.878	0.010	(0.865, 0.886)
75	0.880	0.006	(0.870, 0.887)
100	0.883	0.005	(0.874, 0.889)
600	0.889	0.009	(0.882, 0.894)
1100	0.889	0.005	(0.883, 0.895)
1600	0.889	0.005	(0.882, 0.895)
2100	0.889	0.005	(0.883, 0.894)


Sensitivity of Theta with Tau
Theta
0.85
0.86
0.87
0.88
0.89
0
500
1000
1500
2000
Tau


Histogram of Theta:
Sensitivity Analysis
Density
0
20
40
60
80
100
120
0.875
0.880
0.885
0.890
0.895
0.900
Theta


Traceplot of Theta:
Sensitivity Analysis
0.875
0.880
0.885
0.890
0.895
0
200
400
600
800
Iterations


ACF Plot of Theta:
Sensitivity Analysis
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
Lag

6 Concluding Remarks

We have shown how to estimate the proportion of voters who attended and voted for independence in the Slovenia Public Opinion survey by treating the 'Don't Knows' response as Missing data. Primarily we have used three separate models and compared the estimated proportion θ for each case. First we have used non-parametric MAR model and found the θ to be 0.889. Secondly, we have used the Bayesian ignorable nonresponse model to estimate θ which comes to be 0.882. We have used Markov Chain multi-stage Gibbs sampler for this simulation. The output data from the results of the ignorable model have been used to estimate prior parameters τ and μ vector of the model. This in turn is fed into the nonignorable model to estimate the proportion θ which comes to be 0.885. Histogram, Autocorrelation Function, the MCMC trace plots and the 90% credible interval computation have been made in each case. Finally, the sensitivity analysis shows that the proportion θ stabilizes after τ is greater than 600. The close results from all three models as well as the stabilization of the θ by the varying τ suggest we have a robust underlying model.

References

1. Rubin, D.B., Stern, H.S., and Vehovar V. (1995), Handling 'Don't Know' Survey Responses: The Case of the Slovenian Plebiscite, *Journal of American Statistical Association* **90**, 822-828.

2. Nandram, B., Cox, L.H., and Choi, J.W. (2005), Bayesian Analysis of Nonignorable Missing Categorical Data: An application to Bone Minearal Density and Family Income, *Statistics Canada* **31**, No. 2., pp. 213-225.

3. Albert, Jim (2010), *Bayesian Computation with R*, Second Edition, Springer Science+Busisness Media, New-York.

4. Casella, G. and Robert, C.P. (2010), *Introducing Monte Carlo Methods with R*, Springer Science+Busisness Media, New-York.

APPENDIX - R Code

```
rm(list=ls(all=TRUE))
#library(LearnBayes)
library(MCMCpack)

###############   Define the arrays ####################
comp_arr = c(1191, 8, 8, 0, 158, 68, 7, 14)
i_dk_arr = c(21, 4, 29, 3)
a_dk_arr = c(107, 3, 18, 43)
s_dk_arr = c(90, 2, 1, 2)
ai_dk_arr = c(9,31)
si_dk_arr = c(109, 25)
sa_dk_arr = c(19, 8)
sai_dk_arr = c(96)

n1 = comp_arr[1] + comp_arr[5]
N1 = sum(comp_arr)
N2=(sum(i_dk_arr) + sum(a_dk_arr) + sum(s_dk_arr) + sum(sa_dk_arr)
     + sum(si_dk_arr) + sum(ai_dk_arr) + sum(sai_dk_arr))
w = N1/(N1+N2)

#################################################################
#   (1) Non-Parametric Method Functions                         #
################ Calculate the arrays ######################
calc_arr = function (a,s,p)
{
  #cat("a=[",a,"],s=[",s,"]\n")
  len = length(s)
  temp_arr = c(0,0,0,0,0,0,0,0)
  for(i in 1:len)
  {
      s1=toString(s[i])
#cat("s1=",s1,"\n")
len1 = nchar(s1)
total=0
      for (j in 1:len1)
      {
    k = as.numeric(substr(s1,j,j))

          ## sum up the individual needed elements of the
          ##complete array
```

```
            total = total+p[k]
        }

#cat("total=",total,"\n")

        for (j in 1:len1)
        {
      k = as.numeric(substr(s1,j,j))
            # prorate the don't care elements according to the Y/N
            # weights in the complete array

            temp_arr[k] = (p[k]/total)*a[i]
        }
    }
    return (temp_arr)
}

#########################################################################
doNonParametric=function()
{
     ## In each of these routines, the elements of the second
      ## vector are the elements of the complete_array that need
      ## summed up to get the individual totals. The individual
      ## elements when divided by the totals will give the
     ## probability. The probability is multiplied by the don't
      ## care arrays to get their prorated values.
     ##
    sat_arr = calc_arr(i_dk_arr, c(12,34,56,78), comp_arr)
    sti_arr = calc_arr(a_dk_arr, c(13,24,57,68), comp_arr)
    tai_arr = calc_arr(s_dk_arr, c(15,26,37,48), comp_arr)
    stt_arr = calc_arr(ai_dk_arr, c(1234,5678), comp_arr)
    tat_arr = calc_arr(si_dk_arr, c(1256,3478), comp_arr)
    tti_arr = calc_arr(sa_dk_arr, c(1357,2468), comp_arr)
    ttt_arr = calc_arr(sai_dk_arr, c(12345678), comp_arr)

########### Print the arrays  ######################

    #cat("comp_arr=",comp_arr,",sum=",sum(comp_arr),"\n")
    #cat("sat_arr=",sat_arr,",sum=",sum(sat_arr),"\n")
    #cat("sti_arr=",sti_arr,",sum=",sum(sti_arr),"\n")
    #cat("tai_arr=",tai_arr,",sum=",sum(tai_arr),"\n")
    #cat("stt_arr=",stt_arr,",sum=",sum(stt_arr),"\n")
    #cat("tat_arr=",tat_arr,",sum=",sum(tat_arr),"\n")
```

```
    #cat("tti_arr=",tti_arr,",sum=",sum(tti_arr),"\n")
    #cat("ttt_arr=",ttt_arr,",sum=",sum(ttt_arr),"\n")

    #rounded values
    #cat("comp_arr=",comp_arr,",sum=",sum(comp_arr),"\n")
    #cat("r_sat_arr=",round(sat_arr),",sum=",sum(round(sat_arr)),"\n")
    #cat("r_sti_arr=",round(sti_arr),",sum=",sum(round(sti_arr)),"\n")
    #cat("r_tai_arr=",round(tai_arr),",sum=",sum(round(tai_arr)),"\n")
    #cat("r_stt_arr=",round(stt_arr),",sum=",sum(round(stt_arr)),"\n")
    #cat("r_tat_arr=",round(tat_arr),",sum=",sum(round(tat_arr)),"\n")
    #cat("tti_arr=",round(tti_arr),",sum=",sum(round(tti_arr)),"\n")
    #cat("ttt_arr=",round(ttt_arr),",sum=",sum(round(ttt_arr)),"\n")


############ Calculate theta ##################################################
##### theta = (n1+n2)/(N1+N2) = w*(n1/N1) + (1-w)*(n2/N2)
##### where w = N1/(N1+N2)

     oth_arr = sat_arr + sti_arr + tai_arr + stt_arr +
               tat_arr + tti_arr + ttt_arr

     #cat("oth_arr=",round(oth_arr),",sum=",sum(round(oth_arr)),"\n")
#cat("all_arr=",round(comp_arr+oth_arr),"sum=",sum(round(oth_arr+comp_arr)),"\n")
     #n1 = comp_arr[1] + comp_arr[5]
     n2 = oth_arr[1] + oth_arr[5]
     #N1 = sum(comp_arr)
     #N2 = sum(oth_arr)
     #w = N1/(N1+N2)

     cat("Percent explained by complete cases=",w*(n1/N1),
         "and by missing data =",(1-w)*(n2/N2),"\n")
     theta = w*(n1/N1) + (1-w)*(n2/N2)
     #cat("Non-Parametric theta=",theta,"\n")
     return (theta)
}
###############################################################################
#### (2) Ignorable Model Functions                                         ###
###############################################################################

################ Calculate the arrays ##################
calc_arr2 = function (a,s,p)
{
  #cat("In calc_arr: p=",p,"\n")
  #cat("a=[",a,"],s=[",s,"]\n")
```

```
  len = length(s)
  temp_arr = c(0,0,0,0,0,0,0,0)
  for(i in 1:len)
  {
      s1=toString(s[i])
#cat("s1=",s1,"\n")
len1 = nchar(s1)
total=0
      for (j in 1:len1)
      {
        k = as.numeric(substr(s1,j,j))

          ## sum up the individual needed elements of the
          ##complete array

          total = total+p[k]
      }

#cat("total=",total,"\n")
      temp2_arr = c(rep(0,len1))
      k_arr=c(rep(0,len1))
      for (j in 1:len1)
      {
        k = as.numeric(substr(s1,j,j))
        k_arr[j]=k
        # prorate the don't care elements according to the Y/N
          # weights in the complete array

          #temp_arr[k] = (p[k]/total)*a[i]
          temp2_arr[j] = (p[k]/total)
          #cat("temp_arr=",temp_arr,"\n")
      }
      #cat("k_arr,temp2=",k_arr,temp2_arr,"\n")
      w=rmultinom(1,a[i],temp2_arr)
      #cat("w=", w,"\n")
      temp_arr[k_arr]=w
  }
  return (temp_arr)
}

##############################################################################
#### Get the Y data values using the Probability values
##############################################################################
getY=function(PI,p_vec)
```

```
{
    # PIA is the array of Prob. for the tables.
    #cat("PI=",PI,"\n")

    #cat("p_vec=",p_vec,"\n")
    comp_arr2=rep(0,8)
    #m=matrix(p_vec, nrow=8, ncol=8, byrow=T)   #matrix of probabilities

    #cat("in getY p_mat[2,]=",p_vec[9:16],"\n")
    #cat("BP-start\n")
    sat_arr2 = calc_arr2(i_dk_arr, c(12,34,56,78),PI[2]*p_vec)
    sti_arr2 = calc_arr2(a_dk_arr, c(13,24,57,68),PI[3]*p_vec)
    tai_arr2 = calc_arr2(s_dk_arr, c(15,26,37,48),PI[4]*p_vec)
    stt_arr2 = calc_arr2(ai_dk_arr, c(1234,5678),PI[5]*p_vec)
    tat_arr2 = calc_arr2(si_dk_arr, c(1256,3478),PI[6]*p_vec)
    tti_arr2 = calc_arr2(sa_dk_arr, c(1357,2468),PI[7]*p_vec)
    ttt_arr2 = calc_arr2(sai_dk_arr, c(12345678),PI[8]*p_vec)

    #cat("sat_arr2: 12,34,56,78=[",sat_arr2,"]",i_dk_arr,"\n")
    #cat("sti_arr2: 13,24,57,68=[",sti_arr2,"]",a_dk_arr,"\n")
    #cat("tai_arr2: 15,26,37,48=[",tai_arr2,"]",s_dk_arr,"\n")
    #cat("stt_arr2: 1234,5678=[",stt_arr2,"]",ai_dk_arr,"\n")
    #cat("tat_arr2: 1256,3478=[",tat_arr2,"]",si_dk_arr,"\n")
    #cat("tti_arr2: 1357,2468=[",tti_arr2,"]",sa_dk_arr,"\n")
    #cat("ttt_arr2: 12345678=[",ttt_arr2,"]",sai_dk_arr,"\n")
    #cat("BP-end\n")

    return(c(comp_arr, sat_arr2, sti_arr2, tai_arr2,
             stt_arr2,tat_arr2, tti_arr2, ttt_arr2))
}
##############################################################################
#### Get the Probability P values using the datavalues
##############################################################################
getP=function(y_vec)
{
     #cat("y_vec=",y_vec,"\n")
     m=matrix(y_vec, nrow=8, ncol=8, byrow=T)       #matrix of data
     temp=NULL
     for (i in 1:8)
     {
         temp=c(temp, sum(m[,i]) + 1)
         #temp=c(temp, sum(m[i,1]) + 1)
     }
       #temp=c(temp, rdirichlet(1, sum(m[,i]) + 1))
```

```
     #cat("col_sum+1=",temp,"\n")
     return(rdirichlet(1,temp))
}

#############################################################################
getVi=function(yy)
{

       s1=s5=NULL
        for (j in 1:8)
        {
            s1=c(s1,yy[8*(j-1)+1])
            #cat("s1=",sum(s1),"\n")
            s5=c(s5,yy[8*(j-1)+5])
            #cat("s5=",sum(s5),"\n")
        }
        Vi=(sum(s1)+sum(s5))

    return(Vi)
}
#############################################################################
getPI=function()
{
    alpha=c(1454+1, 57+1, 171+1, 95+1, 40+1, 134+1, 27+1, 96+1)
    return(rdirichlet(1, alpha))
}

#############################################################################

getMU=function(P)
{
### Estimate mu's from the previously run data from Probability tables.
    mu=NULL;
    for (s in 1:8)
    {
       mu[s] = mean(P[,s])
    }
    #cat("MU=",mu,"\n")
    return (mu)
}

#############################################################################
getTAU=function(P, mu)
{
```

```
# Get the variance covariance matrix for P
# For dirichlet distribution, if
# (X1,X2,X3,...,X8) ~ Dirichlet(mu[1]*tau, mu[2]*tau,...,mu[8]*tau)
# and when mu[1]+mu[2]+...+mu[8]=1, then
# E[Xi] = mu[i]/tau, var(X[i]) = (mu[i]*(1-mu[i]))/(1+tau),
# cov(Xi,Xj)=-mu[i]*mu[j]/(1+tau)
#

T=NULL    # T=tau
k=0
vcvm=cov(P)
vcvm
for (i in 1:8)
{
   for (j in 1:8)
   {
       k=k+1
       if (i==j)
           T[k]=(mu[i]*(1-mu[i]))/vcvm[i,i] -1
       else
           T[k]=(-1)*(mu[i]*mu[j])/vcvm[i,j] -1
   }
}
## Take the geometric mean of T to get tau
#cat("vcvm=",vcvm,"\n")
#cat("T=",T,"\n")
tau=(prod(T))^(1/64)
#cat("TAU=",tau, "\n")
return (tau)
}

#########################################################################
doIgnorable=function(Nsim)
{
#########################################################################
## Use GIBBS Sampler with dirichlet as prior and multinomial as likelihood
#########################################################################
#Nsim=10000
unitv8=c(rep(1,8))
PI=rdirichlet(1,unitv8)
#unitv8
#Y=matrix(0,nrow=Nsim,ncol=64,byrow=T)
V=rep(0,Nsim)
P=matrix(0,nrow=Nsim,ncol=8,byrow=T)
```

```
#P[1,]=rdirichlet(1,unitv8)
pp=rdirichlet(1,unitv8)
#set.seed(100)
for (i in 1:Nsim)
{
    yy = getY(PI,pp)
    #cat("yy=",yy,"\n")
    pp = getP(yy)       ## posterior pp
    PI = getPI()        ## posterior PI
    ## save the data
    P[i,] = pp
    #cat("pp=",pp,"\n")
    #Y[i,] = yy
    V[i] =getVi(yy)
    #cat("V[",i,"]=",V[i],"\n")

}

############ Calculate theta For Ignorable Model ##############################
##### theta = (n1+n2)/(N1+N2) = w*(n1/N1) + (1-w)*(n2/N2)
##### where w = N1/(N1+N2)

 #oth_arr = sat_arr + sti_arr + tai_arr + stt_arr +  tat_arr + tti_arr + ttt_arr

     #n1 = comp_arr[1] + comp_arr[5]
     #N1 = sum(comp_arr)
     #N2=(sum(i_dk_arr) + sum(a_dk_arr) + sum(s_dk_arr) + sum(sa_dk_arr)
     #+ sum(si_dk_arr) + sum(ai_dk_arr) + sum(sai_dk_arr))
     #w = N1/(N1+N2)
     # n2 = oth_arr[1] + oth_arr[5]
     #V=getV(Y)
     n3=mean(V)     ## sum of n1 and n2
     #cat("Percent explained by complete cases=",w*(n1/N1),
     # "and by missing data =",(1-w)*(n2/N2),"\n")
     #theta = w*(n1/N1) + (1-w)*(n2/N2)
     theta=n3/(N1+N2)
     #cat("Theta for Ignorable Model =",theta,"\n")

     muVec = getMU(P)
     tau = getTAU(P,muVec)
     #cat("MuVec =",muVec,"tau=",tau,"\n")
     #list (muVec, tau, V)
     #c(theta, muVec, tau, V)
     c(muVec, tau, V)
```

```
}


########## Histogram and Kernel density ####################
plotHTA=function(st,V, what, tstr)
{
    Nsim=length(V)
    V1=V[st:Nsim]
    x <- V1
    if (what=='H')
    {
        header = c("Histogram of Theta:", tstr)
        hist(x, main=header, xlab="Theta", freq=FALSE)
        #hist(x, main="Histogram of Theta: ", tstr, freq=FALSE)
        lines(density(x), col = "red", lwd = 2)
    }
    else if (what=='T')
    {
        header = c("Traceplot of Theta:", tstr)
        plot(mcmc(x))
        traceplot(mcmc(x), main=header)
    }
    else if (what=='A')
    {
        header = c("ACF Plot of Theta:", tstr)
        plot(mcmc(x))
        acf(x, 20, main=header)    # for 20 lags
     }
}


#########################################################################
#  (3) NonIgnorable Method Functions                                    #
#########################################################################

#############################################################################
#### Get the Y data values using the Probability values
#############################################################################
getYNI=function(PIM,p_vec)
{
    # PIA is the array of Prob. for the tables.
    #cat("PI=",PI,"\n")

    #cat("p_vec=",p_vec,"\n")
```

```
    comp_arr2=rep(0,8)
    m=matrix(PIM, nrow=8, ncol=8, byrow=T)   #matrix of probabilities

    #cat("in getYNI m[,2]=",m[,2],"\n")
    #cat("BP-start\n")
    sat_arr2 = calc_arr2(i_dk_arr, c(12,34,56,78),m[,2]*p_vec)
    sti_arr2 = calc_arr2(a_dk_arr, c(13,24,57,68),m[,3]*p_vec)
    tai_arr2 = calc_arr2(s_dk_arr, c(15,26,37,48),m[,4]*p_vec)
    stt_arr2 = calc_arr2(ai_dk_arr, c(1234,5678),m[,5]*p_vec)
    tat_arr2 = calc_arr2(si_dk_arr, c(1256,3478),m[,6]*p_vec)
    tti_arr2 = calc_arr2(sa_dk_arr, c(1357,2468),m[,7]*p_vec)
    ttt_arr2 = calc_arr2(sai_dk_arr, c(12345678),m[,8]*p_vec)

    #cat("sat_arr2: 12,34,56,78=[",sat_arr2,"]",i_dk_arr,"\n")
    #cat("sti_arr2: 13,24,57,68=[",sti_arr2,"]",a_dk_arr,"\n")
    #cat("tai_arr2: 15,26,37,48=[",tai_arr2,"]",s_dk_arr,"\n")
    #cat("stt_arr2: 1234,5678=[",stt_arr2,"]",ai_dk_arr,"\n")
    #cat("tat_arr2: 1256,3478=[",tat_arr2,"]",si_dk_arr,"\n")
    #cat("tti_arr2: 1357,2468=[",tti_arr2,"]",sa_dk_arr,"\n")
    #cat("ttt_arr2: 12345678=[",ttt_arr2,"]",sai_dk_arr,"\n")
    #cat("BP-end\n")

    return(c(comp_arr, sat_arr2, sti_arr2, tai_arr2,
             stt_arr2,tat_arr2, tti_arr2, ttt_arr2))
}

#########################################################################
getPIM=function(mu, tau, yy)
{

    m=matrix(yy, nrow=8, ncol=8, byrow=T)       #matrix of data
    PIM=matrix(0,nrow=8,ncol=8,byrow=T)
    for (i in 1:8)                              ## i is the cell index
    {
        #cat("In getPIM m[",j,"]=",m[j,])
        temp=NULL
        for (j in 1:8)                          ## j is the table index
        {
            temp=c(temp, m[j,i] + mu[j]*tau)
        }
        #cat("temp=", temp)
        PIM[i,] = rdirichlet(1, temp)
    }
    return (PIM)
```

```
}


#############################################################################
doNonIgnorable=function(Nsim, mu, tau)
{
#mu = getMU(P)
#tau = getTAU(P,mu)
#mu=c( 0.7589292, 0.006613945, 0.01036109, 0.001731849, 0.1208561, 0.06826119,
#0.008097546, 0.025149 )
#tau=730

#Nsim=10000
unitv8=c(rep(1,8))
PIM=matrix(0,nrow=8,ncol=8,byrow=T)
for (i in 1:8)
{
    PIM[i,]=rdirichlet(1,mu*tau)   # Different for different cells, i=cell id
    cat("bpPIM[",i,"]=",PIM[i,], "\n")
}
#cat("PIM=",PIM, "\n")
#unitv8
#Y=matrix(0,nrow=Nsim,ncol=64,byrow=T)
V=rep(0,Nsim)
P=matrix(0,nrow=Nsim,ncol=8,byrow=T)
#P[1,]=rdirichlet(1,unitv8)
pp=rdirichlet(1,unitv8)      # same for both Ignorable and Non-ignorable
#set.seed(100)
for (i in 1:Nsim)
{

    yy = getYNI(t(PIM),pp)
    #cat("yy=",yy,"\n")
    pp = getP(yy)                    ## posterior pp
    PIM = getPIM(mu, tau, yy)        ## posterior PI
    ## save the data
    P[i,] = pp
    #cat("pp=",pp,"\n")
    #Y[i,] = yy
    V[i] =getVi(yy)
    #cat("V[",i,"]=",V[i],"\n")

 }
```

```
    ###### Theta for the non-ignorable model ##############

     #n3=mean(V)     ## sum of n1 and n2
     #theta=n3/(N1+N2)
     #cat("Theta for Non-Ignorable Model =",theta,"\n")
     #V1=V[500:Nsim]/2074
     # c(theta,V)
     return(V)
}

#######################################################################
logGamma=function(r)
{
    if (r < 100)
       val = log(gamma(r))
    else
    {
        sum = 0
        while (r > 100)
        {
           r = r-1
           sum = log(r) + sum
        }
        val = sum + log(gamma(r))
    }
    return (val)
}

#####################################################################
getMuFromPMF=function(ngrid, bottom, ungg, wing)
{

    ## Since wing is a logarithm value, first find the max out of it.
    ## This will act as a proportional constant for the posterior density


    term0 = wing[1]
    if (term0 !=-Inf)
    {
        for (it1 in 1:ngrid)
        {
        term0 = max(term0,wing[it1])
        }
```

```
    ## Sum the expoential of these to get the real value except by a const.
    asum = 0
    for (it1 in 1:ngrid)
    {
        asum = asum + exp(wing[it1]-term0)
    }
    #cat ("asum=",asum, "\n")

    probg=c(rep(0,ngrid))
    ##  Then find the prob density by dividing by the sum
    for (it1 in 1:ngrid)
    {
        probg[it1] = exp(wing[it1]-term0)/asum
    }
}
else
   probg=c(rep(0,ngrid))

## Next the cumulative pdf
probgc=c(rep(0,ngrid))
for (it1 in 1:ngrid)
{
    if (it1==1)
        probgc[it1] = probg[it1]
    else
        probgc[it1] = probgc[it1-1] + probg[it1]
}
#cat("probgc=",probgc,"\n")

## Next use a random number to get the interval in which it belongs to
uu = runif(1)
#cat("uu=",uu,"\n")
ipick = 1
for (it1 in 1:ngrid)
{
    if (it1 > 1)
    {
        if (( uu > probgc[it1-1]) & (uu <= probgc[it1])) ipick = it1
    }
    #else cat("Unknown value", uu,"\n")
}
if (ipick == 1)
     rval = bottom + (ungg[ipick]-bottom)*runif(1)
else
```

```
        rval = ungg[ipick-1] + (ungg[ipick]-ungg[ipick-1])*runif(1)
    #cat ("rval=",rval,"\n")
    return(rval)
}


#####################################################################
getMuVecSA=function(PIM, tau, muVec)
{
    ngrid=100
    NewMuVec=c(rep(0,8))

    m=matrix(PIM, nrow=8, ncol=8, byrow=T)   #matrix of probabilities

    sum7 = sum(muVec) - muVec[8]

    mt8 = (1 - sum7)*tau  # shall we take -1?
    g8 = (prod(m[,8]))^(1/8)
    lnh8= mt8*log(g8) - logGamma(mt8)

    for (j in 1:7)
    {

        #cat("bp2PIM[column",j,"]=",m[,j], "\n")
        g = (prod(m[,j]))^(1/8)   # 8th root of (delta = prod of cell prob.)

        mu= 1 - (sum7 - muVec[j])

        muGrid=c(rep(0,ngrid))
        mt=c(rep(0,ngrid))
        lnpmf=c(rep(0,ngrid))

        for (i in 1:ngrid)
        {
            muGrid[i] = i*mu/ngrid
            mt[i] = muGrid[i]*tau # shall we take -1 ?
            lnpmf[i] = 8*(mt[i]*log(g) - logGamma(mt[i]) + lnh8)
     #cat("muGrid[i]=",muGrid[i],"mt[",i,"]=",mt[i], "lnpmf=", lnpmf[i], "\n")
        }

        NewMuVec[j]=getMuFromPMF(ngrid, 0, muGrid, lnpmf)

    }
    NewMuVec[8] = 1 - (sum(NewMuVec))   ## only 1-7 are filled up
```

```
    c(NewMuVec)
}


#######################################################################
#   (4) Sensitivity Analysis With Grid Functions                      #
#######################################################################
doSensitivityAWithGrid=function(Nsim, tau)
{
muVec=c(rep(0.125,8))

unitv8=c(rep(1,8))
PIM=matrix(0,nrow=8,ncol=8,byrow=T)
for (i in 1:8)
{
    PIM[i,]=rdirichlet(1,muVec*tau)   # Different for different cells, i=cell id
    cat("bpPIM1[",i,"]=",PIM[i,], "\n")
}
#cat("PIM=",PIM, "\n")
V=rep(0,Nsim)
P=matrix(0,nrow=Nsim,ncol=8,byrow=T)
pp=rdirichlet(1,unitv8)      # same for both Ignorable and Non-ignorable
#set.seed(100)
for (i in 1:Nsim)
{
    muVec=getMuVecSA(t(PIM), tau, muVec)
    #cat("muVecAfter=",muVec,"\n")
    yy = getYNI(t(PIM),pp)
    #cat("yy=",yy,"\n")
    pp = getP(yy)                        ## posterior pp
    PIM = getPIM(muVec, tau, yy)         ## posterior PI
    ## save the data
    P[i,] = pp
    #cat("pp=",pp,"\n")
    V[i] =getVi(yy)
    #cat("V[",i,"]=",V[i],"\n")

 }
    ###### Theta for the SA model ##############
    #n3=mean(V)
    #theta=n3/(N1+N2)
    #c(theta, V)
    return(V)
}
```

```
#########################################################################
###########           MAIN ROUTINES       ##########################
#########################################################################

######################
# (1)Non-Parametric
######################
#theta=doNonParametric()
#cat("Theta for Non-Parametric Model =",theta,"\n")

####################
# (2)Ignorbale Model
####################
Nsim1=10000
z=doIgnorable(Nsim1)
muVec=z[1:8]
tau=z[9]
V=z[10:(Nsim1+9)]/(N1+N2)
theta=mean(V)
cat("Theta for Ignorable Model =",theta,"\n")

plotHTA(100,V, "H", "Ignorable Non-response Model")
plotHTA(100,V, "T", "Ignorable Non-response Model")
#plotHTA(100,V, "T", "Ignorable Non-response Model")

## 90% credible interval
#quantile(V,c(.05,.95))/(N1+N2)

# muVec = 0.758852619 0.006591329 0.010346730 0.001628363 0.120930036
# 0.068257136 0.008085182 0.025308606
# tau = 894.547

#########################
# (3) Non-Ignorable Model
#########################
#for (i in 1:5)
#{
Nsim2=10000

#use the muVec and tau obtained from Ignorable model
z=doNonIgnorable(Nsim2,muVec, tau)
V=z/(N1+N2)
theta=mean(V)
```

```
cat("Theta for Non-Ignorable Model =",theta,"\n")

plotHTA(100,V, "H", "Non-ignorable Non-response Model")
#plotHTA(100,V, "T", "Non-ignorable Non-response Model")
#plotHTA(100,V, "A", "Non-ignorable Non-response Model")

## 90% credible interval
#quantile(V,c(.05,.95))/(N1+N2)
# }


################################
#(4)doSensitivityAWithGrid()
################################

#tau=100
thetaVec=rep(0,10)
sdVec=rep(0,10)
credVec=matrix(0, nrow=10, ncol=2,byrow=T)
#tauVec=seq(100, 2100, by=500)
tauVec=c(5,10,20,50,75,100,600,1100,1600,2100)
i=0
for (tau in tauVec)
{
i=i+1
Nsim2=1000
#muVec=c(0.7589668, 0.006723437, 0.01017159, 0.001670488, 0.1205841, 0.06831049,
# 0.00810527, 0.02546785)
#tau=674.5044
z=doSensitivityAWithGrid(Nsim2,tau)
V=z/(N1+N2)
theta=mean(V)
thetaVec[i]= theta
sdVec[i]=sd(V)
credVec[i,]=quantile(V,c(.05,.95))
cat("Tau=",tau, "Theta for Sensitivity Analysis Model", theta,"\n")
cat("Tau=",tau, "sd=",sdVec[i],"credInt=", credVec[i,],"\n")

plotHTA(100,V, "H", "Sensitivity Analysis")
plotHTA(100,V, "T", "Sensitivity Analysis")
#plotHTA(100,V, "A", "Sensitivity Analysis")

## 90% credible interval
#quantile(z,c(.05,.95))/(N1+N2)
```

```
}

thetaVec=rep(0,5)
tauVec=tauVec=seq(100, 2100, by=500)
#thetaVec=c(0.8857639, 0.8854624 , 0.8856851,   0.8855564,  0.8855910)
plot(tauVec, thetaVec, xlab="Tau", ylab="Theta",
     main="Sensitivity of Theta with Tau")
w2=cbind(tauVec,thetaVec,sdVec,credVec)
```