IST772 Summary Template: Chapter 10 – Logistic Regression
Originality Assertion: By submitting this file you affirm that this writing is your own.

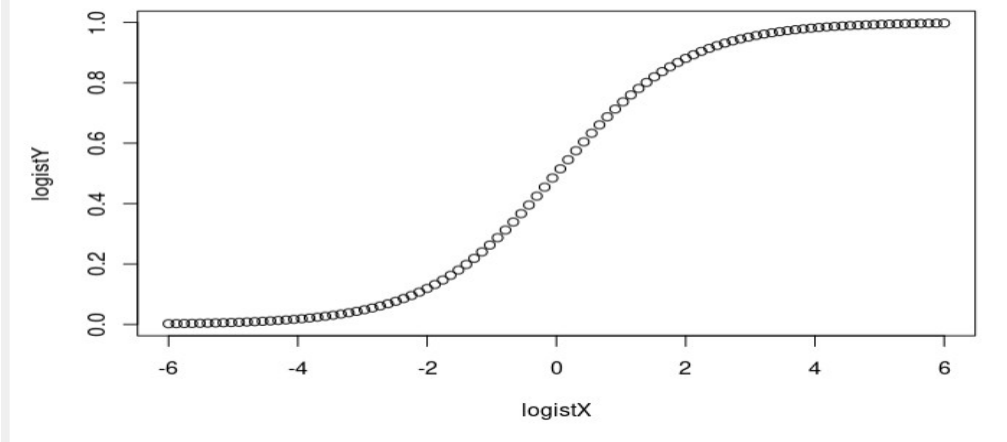
Name: Hendi Kushta
Date: 04/02/2023

****Important** Copying and/or pasting anything from the textbook will not be acceptable for your chapter notes submissions. You must write your notes in your own words and generate your own code, results, and graphs in R. This is what forces your brain to process the material that you read.**

INTRODUCTION

In the previous chapters, we learned about methods for predicting or modeling numerical data. However, these methods cannot be used for predicting categorical data, which is data that is divided into two or more groups. Logistic regression is a statistical technique that can be used for predicting binary outcomes, such as whether a customer will buy a product or whether a patient will recover from a disease. The technique works by creating a model that can predict the probability of a particular outcome using a combination of categorical and numerical data. By using logistic regression, we can gain insights into complex phenomena with binary outcomes. The technique can be used in various fields such as marketing, healthcare, and education, and is a powerful tool for making informed decisions.

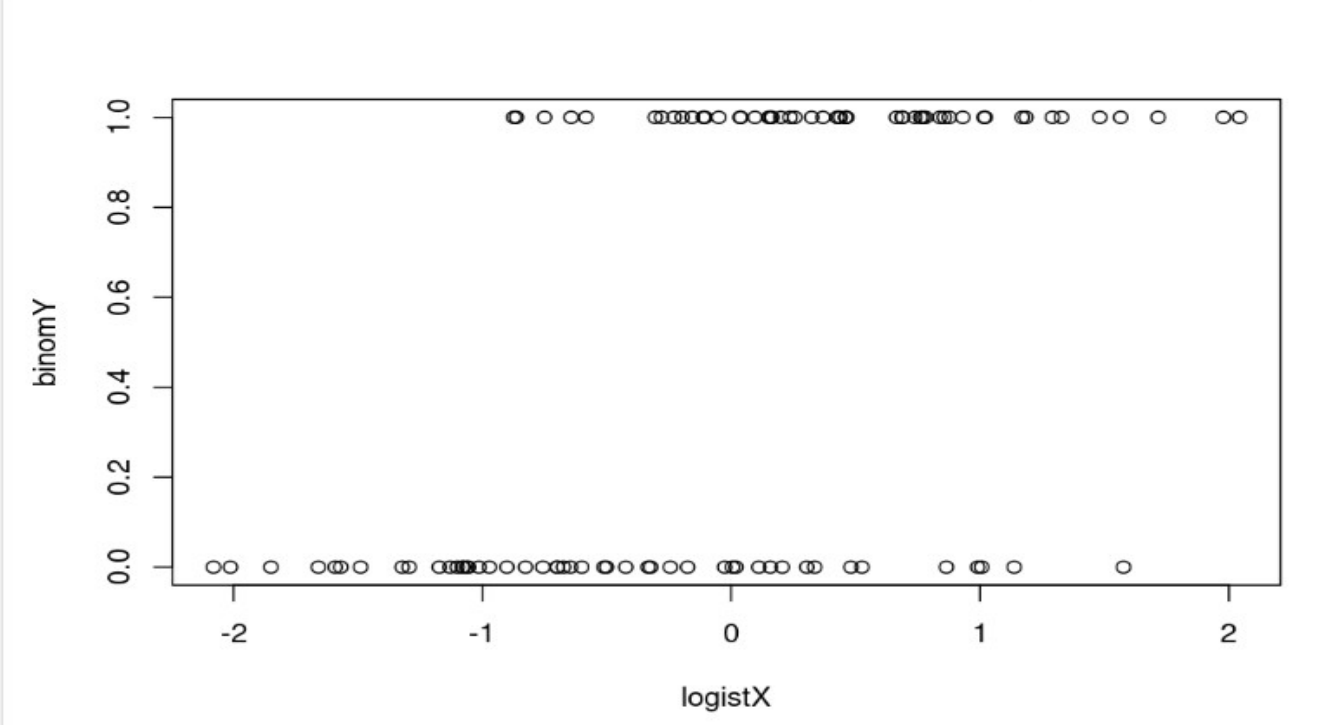
```{r}
# Create a sequence of 100 numbers, ranging
# from -6 to 6 to serve as the X variable
logistX <- seq(from=-6, to=6, length.out=100)
# Compute the logit function using exp(), the inverse of log()
logistY <- exp(logistX)/(exp(logistX)+1)
# Now review the beautiful S curve
plot(logistX,logistY)
```



IST772 Summary Template: Chapter 10 – Logistic Regression
Originality Assertion: By submitting this file you affirm that this writing is your own.

The logistic curve is a mathematical function that transforms a continuous range of values into probabilities that range between 0 and 1. It is often used to model the probability of a binary outcome, such as whether a person is telling the truth or not. The logistic regression model uses the logistic curve to estimate the probability of a binary outcome based on one or more predictor variables. In the article, an example is given of using a GSR sensor to measure palm sweat and detect lies. The logistic curve is used to estimate the probability of a person lying based on their GSR readings. Finally, the article provides some code examples in R for creating predictor variables and fitting a logistic regression model.

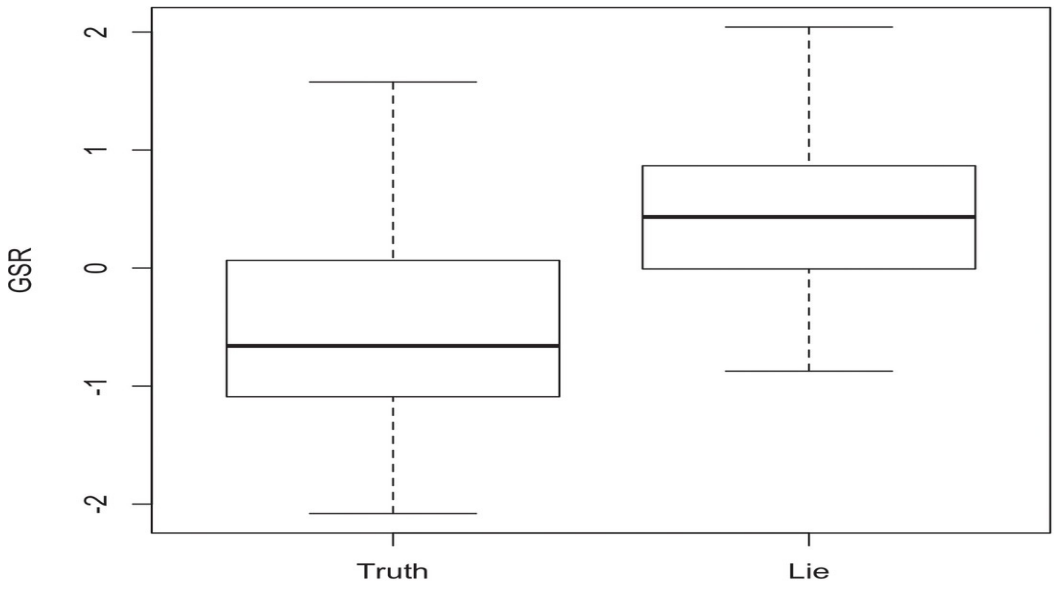
```
# Create a random, standard-normal predictor variable
set.seed(123)
logistX <- rnorm(n=100,mean=0,sd=1)
# Create an outcome variable as a logit function of the predictor
logistY <- exp(logistX)/(exp(logistX)+1)
# Make the dichotomous/binomial version of the outcome variable
binomY <- round(logistY)
# Add noise to the predictor so that it does not perfectly predict the outcome
logistX <- logistX/1.41 + rnorm(n=100,mean=0,sd=1)/1.41
plot(logistX, binomY)
```



The author used the "set.seed()" function to ensure that the random numbers generated will be the same for everyone who tries to replicate the results. They created a variable called "X" which has a normal distribution with a mean of 0 and a standard deviation of 1. The author then recreated the logistic curve based on the new values of "X". They transformed the "Y" variable into a binary outcome

using the round() function to split the responses at the inflection point of the probability curve (0.5). The author added some noise to the predictor variable to make the relationship between "X" and "Y" imperfect. The scatterplot of the two variables is difficult to interpret, but by reversing the view of "X" and "Y" and using a box plot, it becomes easier to visualize. This was done to prepare for a logistic regression analysis.

The graph in Figure 10.3 presents a clearer way of interpreting the data, despite having the outcome variable on the x-axis and the predictor on the y-axis, which is the opposite of what is typically done in statistics. The vertical axis shows the distribution of the predictor variable, with a separate boxplot of the predictor across the two categories of the outcome variable. The graph shows that for low GSR values, which indicate dry palms, most cases are categorized as "Truth", whereas for high GSR values, which indicate sweaty palms, most cases are categorized as "Lie". However, there is still some overlap between the categories due to some people having naturally dry or sweaty palms, indicating that GSR is not a perfect predictor of truth or lies. To determine how well GSR predicts the outcome variable, we will perform a logistic analysis.



The code for fitting a logistic regression model in R is similar to that for linear regression, except that we specify the "binomial()" family as the link function. The "Deviance Residuals" show information about the distribution of the model residuals. The coefficients are the most interesting part of the output, and the coefficient on the predictor variable (GSR) is significantly different from 0, indicating that it is a strong predictor of truth or lie. However, the coefficient is not easy to interpret as a simple slope as in linear regression, but instead

represents the logarithm of the odds of the dependent variable. Therefore, we can transform the coefficient to regular odds for better interpretation.

```{r}
glmOut <- glm(binomY ~ logistX, data=logistDF, family=binomial())
summary(glmOut)
```

The author explains that the coefficient for logistX can be used to calculate the change in odds of binomY for a one unit change in logistX. For example, if the GSR variable changes from 0 to 1, it is almost five times more likely that the person is lying. However, it is important to note that the point estimate from the model may not be the population value, and the confidence interval should also be considered. The confidence interval for logistX ranges from 2.65:1 to 10.19:1, indicating a fair amount of uncertainty around the point estimate of 4.9:1. The author also explains the concept of null deviance, which represents the error in the model if there was no connection between the X and Y variables. The residual deviance shows how much the error in the null model is reduced by introducing the X variable, and can be tested using a chi-square distribution.

A LOGISTIC REGRESSION MODEL WITH REAL DATA

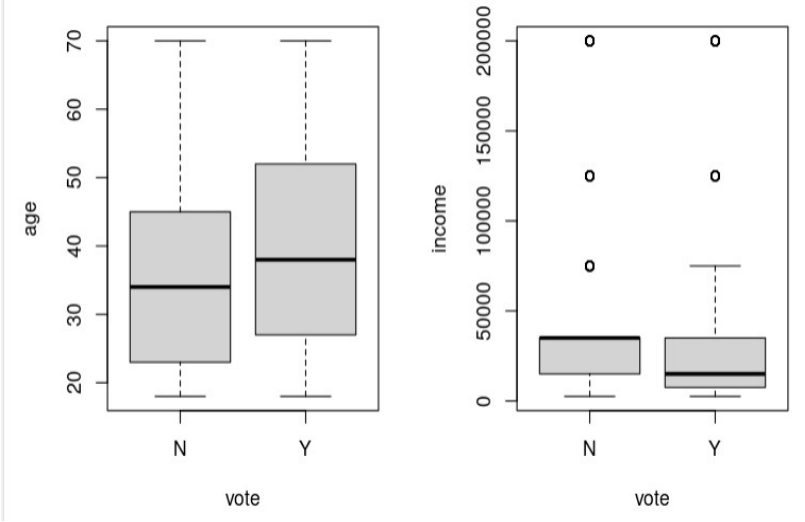
In this section, the author provides an example of using real data to apply the concepts learned in logistic regression. The data used in this example is the Chile data set from the "car" package, which contains responses from 2,700 Chilean citizens on their views of the 1988 plebiscite on President Pinochet. The author will use two metric variables (age and income) to predict whether the individual would vote "Y" in favor of keeping Pinochet in office or "N" against keeping him in office. Before proceeding, the author filters out missing values and extracts only the "Y" and "N" responses.

```{r}
#install.packages("car")
library(car)
ChileY <- Chile[Chile$vote == "Y",]        # Grab the Yes votes
ChileN <- Chile[Chile$vote == "N",]        # Grab the No votes
ChileYN <- rbind(ChileY,ChileN)          # Make a new dataset with those
ChileYN <- ChileYN[complete.cases(ChileYN),]     # Get rid of missing data
ChileYN$vote <- factor(ChileYN$vote,levels=c("N","Y")) # Simplify the factor
```

IST772 Summary Template: Chapter 10 – Logistic Regression
Originality Assertion: By submitting this file you affirm that this writing is your own.

The author presents a sequence of R code to select and combine specific rows from a data set named "Chile" based on the poll results of Chilean citizens regarding the 1988 plebiscite on president Pinochet. The code selects only those rows with "Y" and "N" votes, binds them together into a new data frame, and removes any rows with missing values. The resulting data set is converted into a factor with two levels, "Y" and "N", and contains 1,703 rows of data. The author also provides code to generate box plots for the predictors of age and income, divided by Yes and No votes. The No side won the plebiscite with 56% of the vote, while the data set contains 51% No votes.

```r
par(mfrow=c(1,2))
boxplot(age ~ vote, data=ChileYN)
boxplot(income ~ vote, data=ChileYN)
```



The figureindicate that younger voters may have a higher tendency to vote "no," while wealthier voters may also be more likely to vote "no." However, there is significant overlap in the distributions of the predictors for both "Yes" and "No" votes, making it difficult to determine if these differences are simply due to sampling errors. To explore further, the author will now use logistic regression to predict whether a person will vote "Yes" or "No" based on their age and income level.

IST772 Summary Template: Chapter 10 – Logistic Regression
Originality Assertion: By submitting this file you affirm that this writing is your own.

```
{r}
chOut <- glm(formula = vote ~ age + income, family = binomial(),
data = ChileYN)
summary(chOut)
```

```
Call:
glm(formula = vote ~ age + income, family = binomial(), data = ChileYN)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.435  -1.126  -1.004   1.191   1.373

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -7.581e-01  1.418e-01  -5.346 9.01e-08 ***
age          1.924e-02  3.324e-03   5.788 7.11e-09 ***
income      -2.846e-07  1.142e-06  -0.249    0.803
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2360.3  on 1702  degrees of freedom
Residual deviance: 2326.0  on 1700  degrees of freedom
AIC: 2332

Number of Fisher Scoring iterations: 4
```

The logistic regression output indicates that the intercept has a significant difference from zero. The intercept in this type of regression represents the log-odds of a "Yes" vote when both age and income are equal to zero. However, since it is unlikely that the research included polling newborns with no income, the intercept value does not have much meaning in this context. The coefficient for age is statistically significant, meaning that we can reject the null hypothesis that the log-odds of age is 0 in the population. The coefficient for income is not significantly different from zero, meaning that we fail to reject the null hypothesis that the log-odds of income is equal to 0 in the population. To convert the log-odds coefficients to regular odds, we can use the "exp(coef(chOut))" command.

```
{r}
exp(coef(chOut))
```

```
(Intercept)         age      income
  0.4685354   1.0194293   0.9999997
```

The intercept odds of 0.46:1 represent the log-odds of a Yes vote for a newborn with no money. For each additional year of age, the log-odds increase by about 1.9%, resulting in odds of 1.019:1. The income predictor does not have a significant effect on the log-odds of a Yes vote, with the odds remaining almost

exactly the same regardless of income. The confidence intervals for these odds can be checked using exp(confint(chOut)).

```{r}
exp(confint(chOut))
```

```
Waiting for profiling to be done...
                2.5 %    97.5 %
(Intercept) 0.3544246 0.6180933
age         1.0128365 1.0261271
income      0.9999975 1.0000020
```

The results from the hypothesis tests match the confidence intervals, with income coefficient being nonsignificant as the confidence interval includes 1:1. The confidence interval for the intercept ranges from 0.35:1 to 0.61:1. The confidence interval for the age coefficient ranges from 1.0128:1 to 1.0261:1. To understand the confidence interval for age better, you can scale it to decades instead of years, which reveals that for a 10-year change in age, the confidence interval ranges from a 12.8% to 26.1% increase in the likelihood of voting Yes. The chi-square test shows that the model with only age as a predictor is statistically significant, while adding income does not improve the model's fit. Finally, effect size can be calculated using pseudo-R-squared measures.

```{r}
table(round(predict(chOut, type="response")), ChileYN$vote)
```

```
    N   Y
0 565 449
1 302 387
```

The table() function is used to create a two-by-two contingency table that compares the predicted outcomes from a model to the actual outcomes. The table allows for easy identification of correct and incorrect predictions, which are shown on the main diagonal and off-diagonal, respectively. From this table, the overall accuracy of the model can be calculated, which is found to be quite low in this case. This reinforces the earlier finding that although age is a significant predictor of voting behavior, including income and age in a predictive model does not result in a very good model. Additionally, the number of false positives and

false negatives can be identified from the table, and it is asked which type of error is more common in this model.

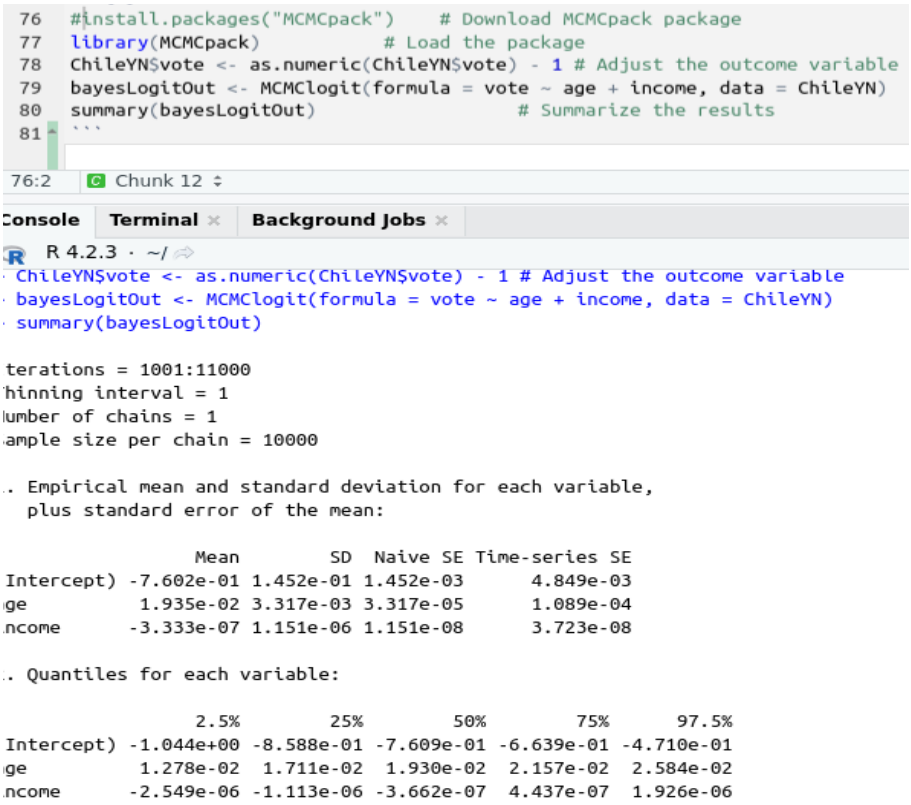
BOX ON P.222: MULTINOMIAL LOGISTIC REGRESSION

Multinomial logistic regression is a statistical method designed for categorical outcome variables that have more than two categories. Unlike binomial logistic regression, which only deals with two categories, multinomial logistic regression requires us to designate one category as the baseline and compare the other categories to that baseline. For instance, when classifying colleges based on their characteristics, we could use a multinomial outcome variable with three categories: community college, liberal arts college, and comprehensive college. By setting community college as the baseline, we can model the comparisons between community versus liberal arts and community versus comprehensive. Multinomial logistic regression can be broken down into k-1 binomial models, where k is the number of categories in the outcome variable. Instead of modeling the outcome variable as a single entity, we treat it as a set of binary models, with each model comparing one category to the baseline category. To perform a multinomial logistic regression analysis, we can use a range of packages available in R, such as mlogit, nnet, MCMCpack, and BayesLogit. Each package offers various functions to estimate multinomial logistic models using different techniques, including maximum likelihood estimation, neural networks, and Bayesian methods.
When dealing with a multinomial outcome variable that is ordered, it's possible to simplify the analysis by creating a model where the coefficients describe the log-odds of a transition between any two adjacent categories. An example of such an outcome variable is the highest educational attainment, which could have four ordered categories: primary school-only, secondary school, college, and graduate school. To learn more about this topic, interested readers can consult a white paper by Starkweather and Moske or search for "multinomial logistic regression CRAN" to find helpful information online.

BAYESIAN ESTIMATION OF LOGISTIC REGRESSION

Bayesian methods can also be used for logistic regression, similar to ANOVA and linear regression. The goal of the Bayesian analysis is to use weakly informative priors on the coefficients, and generate posterior distributions using the Markov chain Monte Carlo technique. The posterior distributions provide us with point estimates of the coefficients and the highest density interval around the mean, indicating where the population value of the coefficient is likely to lie. Although the BayesFactor package does not provide a logistic regression procedure, we

can use MCMCpack, an R package created by Martin, Quinn, and Park (2011), to achieve similar results. MCMCpack uses some of the same underlying packages as BayesFactor, so the output will be similar.



```
#install.packages("MCMCpack")    # Download MCMCpack package
library(MCMCpack)           # Load the package
ChileYN$vote <- as.numeric(ChileYN$vote) - 1 # Adjust the outcome variable
bayesLogitOut <- MCMClogit(formula = vote ~ age + income, data = ChileYN)
summary(bayesLogitOut)                  # Summarize the results
```

76:2 Chunk 12

Console | Terminal | Background Jobs

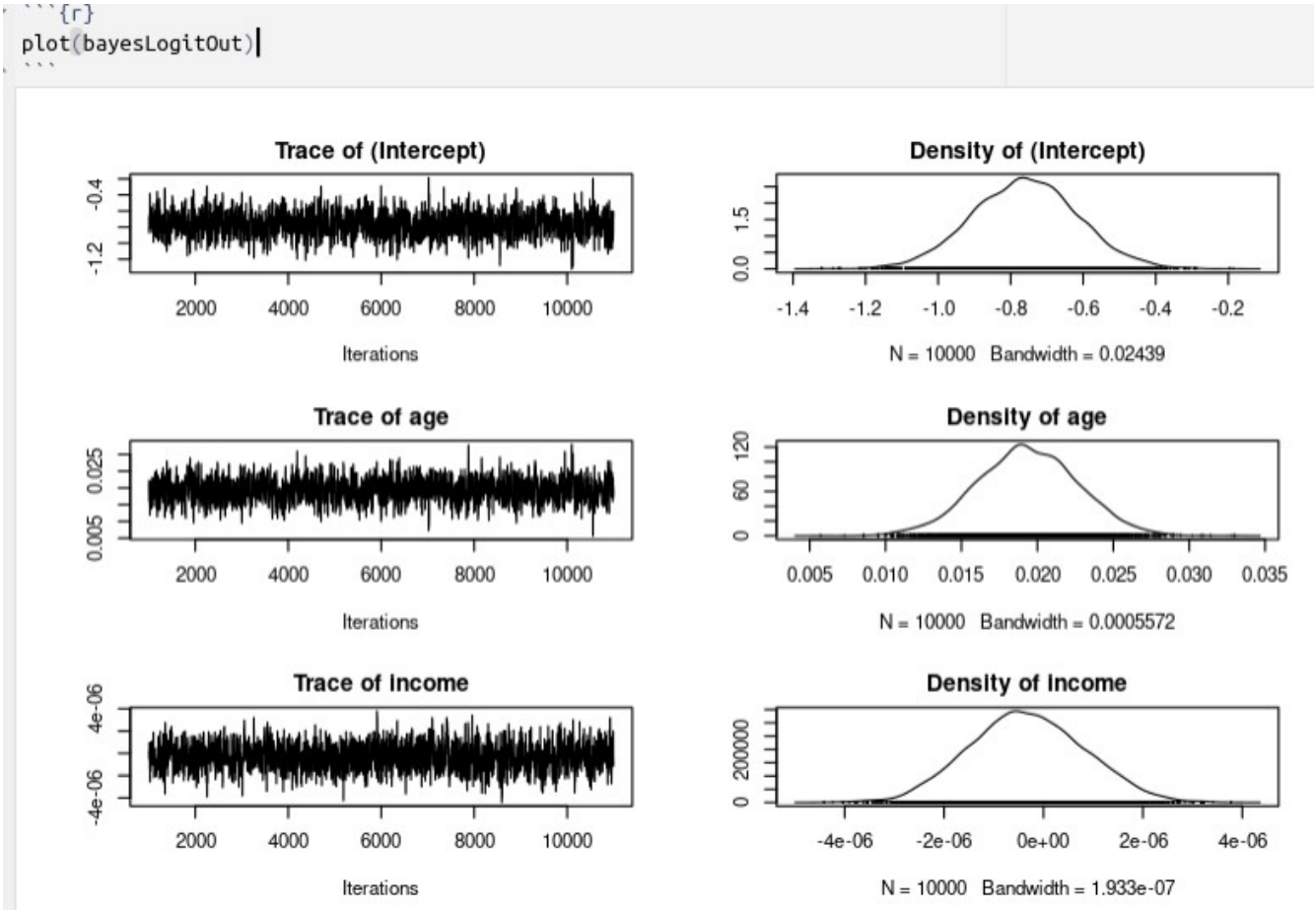
R 4.2.3 · ~/

```
ChileYN$vote <- as.numeric(ChileYN$vote) - 1 # Adjust the outcome variable
bayesLogitOut <- MCMClogit(formula = vote ~ age + income, data = ChileYN)
summary(bayesLogitOut)

terations = 1001:11000
hinning interval = 1
umber of chains = 1
ample size per chain = 10000

. Empirical mean and standard deviation for each variable,
  plus standard error of the mean:

                Mean        SD  Naive SE Time-series SE
Intercept) -7.602e-01 1.452e-01 1.452e-03      4.849e-03
ge          1.935e-02 3.317e-03 3.317e-05      1.089e-04
ncome      -3.333e-07 1.151e-06 1.151e-08      3.723e-08

. Quantiles for each variable:

                2.5%        25%        50%        75%      97.5%
Intercept) -1.044e+00 -8.588e-01 -7.609e-01 -6.639e-01 -4.710e-01
ge          1.278e-02  1.711e-02  1.930e-02  2.157e-02  2.584e-02
ncome      -2.549e-06 -1.113e-06 -3.662e-07  4.437e-07  1.926e-06
```

In the given code snippet, the MCMCpack package was downloaded and loaded into memory using the library() command. Then, the outcome variable was transformed using as.numeric() to convert the factor coding into a number and adjust the values. The MCMClogit() function was used with the same model specification as glm() to obtain the posterior distributions of the coefficients. The summary() function was run on the result to obtain the point estimates, standard deviations, and quantiles of each coefficient, including the highest density interval (HDI). A plot function was used to visualize the HDIs. The output showed that the point estimates for the intercept and coefficients were similar to traditional logistic regression, and the HDIs provided a range in which the population values of the coefficients were likely to lie.

IST772 Summary Template: Chapter 10 – Logistic Regression
Originality Assertion: By submitting this file you affirm that this writing is your own.

```{r}
plot(bayesLogitOut)
```



The figure in this section shows the output of a Markov chain Monte Carlo analysis conducted with the MCMClogit() function. The left column shows the trace of the MCMC algorithm for each coefficient, while the right column displays the density plots for each coefficient. The density plots show the likely position of each coefficient, with the true population value being closer to the middle of each distribution. It is possible to graphically review the posterior distribution and the highest density interval (HDI) for each coefficient, which provides extensive information about the "alternative hypothesis" being tested for each coefficient. The section also includes some code that allows us to examine the posterior distribution of age in terms of regular odds, rather than log-odds.

The author explains how they converted the posterior distribution of the age coefficient from log-odds to regular odds using the apply() function. They created a histogram of the resulting distribution and marked off the edges of the 95% HDI. The histogram shows a symmetric distribution with a center point of about 1.02, indicating an increase of about 2% in the likelihood of a Yes vote for every increase in age of 1 year. The 95% HDI spans a range of values starting at 1.013 and ending at 1.026, which is similar, but not identical, to the confidence interval

obtained from the glm() analysis. The histogram provides valuable information about the most likely range of coefficient values in the population.

```{r}
ageLogOdds <- as.matrix(bayesLogitOut[,"age"])  # Create a matrix for apply()
ageOdds <- apply(ageLogOdds,1,exp)     # apply() runs exp() for each one
hist(ageOdds)                          # Show a histogram
abline(v=quantile(ageOdds,c(0.025)),col="black") # Left edge of 95% HDI
abline(v=quantile(ageOdds,c(0.975)),col="black") # Right edge of 95% HDI
```