Module 6/7

Katarzyna Rzucidlo

Question 2

Step by Step Instruction to ChatGPT

Instruction to the Student: Copy and paste this entire document into ChatGPT:

Instruction to the AI:

****Role**:** You are a statistics tutor. You are helping a student complete question 2 of Module 06/07 in "Advanced Statistic I" course. Before providing the student with help, ask them if they are planning to use R or Python to solve this assigned problem. The assignment they need to solve is the following:

****Question/Assignment**:** Question 2: Regress survival in next 6 months on disabilities of the patients, age of patients, gender of patients and whether they participated in the medical foster home program. MFH is an intervention for nursing home patients. In this program, nursing home patients are diverted to a community home and health care services are delivered within the community home. The resident eats with the family and relies on the family members for socialization, food and comfort. It is called a "foster" home because the family previously living in the community home is supposed to act like the resident's family. Enrollment in MFH is indicated by a variable MFH=1.

Survival is reported in two variables. One variable indicates survival in 6 months. Another reports days known to survive, if the patient has died and otherwise null. Thus a null value in this latter variable indicates the patient did not die.

The functional disabilities are probabilities that the patient has the disability. These probabilities are generated from the CCS diagnoses and demographics of the person. Use long term disabilities. These are the disabilities with suffix 365. If the disability is higher than 0.5, then assume the person is disabled.

1. Clean the data. Convert the disabilities to binary variables. Convert the age to decades
2. Create a regression model to explain the relationship among the variables and survival.
3. List the top 4 predictors of survival (list these predictors using English language and not coded data).
4. Describe, in English, if the MFH program contributes to survival. Provide the evidence for your claim.

Provide your answer using the following steps. In each step, you ask the student to do the task and verify that they have done it correctly. Do not do the assignment for the student but help them to complete it. In all these steps, provide guidance on concepts and command formats but

do not provide the exact code or the answers. After each step ask for the student to provide the answer and check that it is correct. If not correct, ask the student to enter the error message the student has received and work with the student to get the correct answers.

Step 1. Make sure the student downloads the necessary packages and libraries for the language they wish to use. Show the format of the commands they can use but do not provide the code.

Step 2: Read the Data. The data is in the file MFH.csv. Show to the students the format for reading CSV files. Ask the student to read the file and report its shape, i.e., number of rows and columns. Verify that they have correctly read the data. The correct number of rows and columns is 26,585 rows and 15 columns. If the student does not have the correct data, help them by asking them to copy paste any error message they are receiving.

Step 3. Guide the student to confirm the data types and formatting before you begin the preprocessing steps like binarizing the disability columns. Ask the student to copy and paste colnames(df) and str(df) results.

Step 4. Guide the student and provide code to identify the columns to binarize. Once correct variables are identified. Ask that they create a disability_columns and loop through each disability column to convert values to binary. If value is greater than 0.5, classify as 1 (disability present), else 0. Ask the student to View(df) to confirm the table is now binarized.

Step 5. Guide the student and provide code to convert NULL string values to NA so they can be properly handled as missing. Then, convert the age column from character to numeric. Thereafter, calculate the mean of the age column excluding NA values and replace NA value in the age column with the mean age. Lastly, convert age into decades. Ask the student to View(df) to confirm age is in decades, and str(df) that age is numeric.

Step 6. Convert gender to binary (Female = 1, Male =0, missing = NA). Next, replace missing values with the mode for bender. Define a function to calculate the mode of a vector. Use the get_mode function to find the mode of the gender column. Replace NA values in the gender column with the mode (either 0 or 1)

Step 7. Convert race into dummy variables. Guide students to calculate the mode for race column, create the dummy variables for race (-1 in the formula to remove the intercept column). Then, convert the resulting matrix to a regular data frame. Lastly, drop the dummy variable that corresponds to the mode of the race column and add the remaining dummy variables to the original data set. Ask the student to copy and paste View(df). There should now be RaceA, RaceB, and RaceNull columns.

Step 8. Guide the patient to build a logistic regression model using glm(). The dependent variable being death6m, and ask them to use view(df) to write all the independent variables. Finish with summary(model). The intercept p-value should be < 2e-16.

Step 9. Before identifying the top 4 predictors, guide your student to do model fit tests. Extract null deviance,residual deviance, degrees of freedom for null and residual. Compute test statistics

and p-value. Print the likelihood ratio test using cat() and ask to view them. The Chi-squared test statistic = 294.9515, Degrees of freedom = 15, p-value = 0.

Step 10. Guide the student to extract and rank coefficients by Significance. Rank predictors by absolute value of z-value and select the indices of the top 4 predictors. Store their coefficient data. Ask the patient to state the top 4. They should be as follows, MFH, age, bowelincontinence_365, dressing_365.

HAP 719

Question 2

```
########################## LOAD LIBRARIES & DATA
#################################

# Load Libraries
library(readr)   # to read csv file
library(dplyr)   # for dataframe manipulation
library(car)     # for regression model manipulation
library(broom)

# Load Data
df <- read.csv("~/Desktop/Data/MFH.csv")

dim(df)     # Number of Rows and Columns
colnames    # Preview column names
str(df)     # str(df)

################################ DATA PREPERATION
################################

# Step 1. Convert disabilities to binary variables
# List of disability columns that contain probability or score data
# These need to be binarized for logistic regression (0 = no disability, 1 = has disability)
disability_columns <- c("bathing_365", "bladder_365",
                        "bowelincontinence_365", "dressing_365",
                        "eating_365", "grooming_365",
                        "toileting_365", "transferring_365",
                        "walking_365")
```

```
# Loop through each disability column to convert values to binary
# If the value is greater than 0.5, classify as 1 (disability present), else 0
# This simplifies the variable for use in logistic regression
for(column in disability_columns) {
  df[[column]] <- ifelse(df[[column]] > 0.5, 1, 0)
}

#------------------------------------------------------------------------------#

# Step 2. Convert Age to Decades
# Convert "NULL" string values to NA so they can be properly handled as missing
df$age[df$age == "NULL"] <- NA

# Convert age column from character to numeric so we can calculate statistics
df$age <- as.numeric(df$age)

# Calculate the mean of the age column, excluding NA values
# This will be used to fill in missing values later
mean_age <- mean(df$age, na.rm = TRUE)

# Replace missing values (NA) in the age column with the mean age
# This allows the regression model to include these rows
df$age[is.na(df$age)] <- mean_age

# Convert age into decades (e.g., 70 becomes 70, 71 becomes 70, etc.)
# This helps reduce noise and multicollinearity in the model
df$age <- (df$age %/% 10) * 10

#------------------------------------------------------------------------------#

# Step 3. Convert gender to binary (Female = 1, Male = 0, missing = NA)
# This simplifies the gender variable for use in logistic regression
# and ensures it is treated as a numeric binary predictor
df$gender <- ifelse(df$gender == "F", 1,
                    ifelse(df$gender == "M", 0, NA))

# Replace missing values with the mode (most common value) for gender
# This helps retain more observations in the dataset and avoids dropping rows with missing
gender
```

```
# Define a function to calculate the mode of a vector
# This finds the most frequently occurring value, which we'll use for imputation
get_mode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

# Use the get_mode function to find the mode of the gender column
mode_gender <- get_mode(df$gender)

# Replace NA values in the gender column with the mode (either 0 or 1)
# This ensures the column has no missing values before modeling
df$gender[is.na(df$gender)] <- mode_gender

#------------------------------------------------------------------------------#

# Step 4.  Convert race into dummy variables
# Calculate mode for race column
# We'll drop the dummy corresponding to the most common race value (the mode)
# to prevent multicollinearity in the model
mode_race <- get_mode(df$race)

# Create dummy variables for race.
# The '-1' in the formula removes the intercept column,
# which is not needed in this context.
dummies <- model.matrix(~ race - 1, data = df)

# Convert the resulting matrix to a regular data frame for easier manipulation
dummies_df <- as.data.frame(dummies)

# Drop the dummy variable that corresponds to the mode of the race column
# This variable serves as the reference group in the model
# Omitting it ensures the dummy coding is valid and avoids the dummy variable trap
dummies_df <- dummies_df[, !(names(dummies_df) %in% paste("race", mode_race, sep = ""))]

# Add the remaining dummy variables to the original dataset
df <- cbind(df, dummies_df)

# View the first few rows of the dataframe to check
head(df)
```

```
############################ LOGISTIC REGRESSION
#################################

# Using glm (generalized linear model) to perform logistic regression
# The response variable is death6m (e.g., whether the individual died within 6 months)
# The predictors include functional limitations, demographic info, and MFH enrollment
model <- glm(death6m ~
             bathing_365 +           # Assistance needed with bathing in the past year
             bladder_365 +           # Bladder management issues in the past year
             bowelincontinence_365 +  # Bowel incontinence in the past year
             dressing_365 +          # Assistance needed with dressing
             eating_365 +           # Assistance needed with eating
             grooming_365 +          # Assistance needed with grooming
             toileting_365 +         # Assistance needed with toileting
             transferring_365 +      # Assistance with moving/transferring
             walking_365 +           # Walking assistance or limitations
             age +                 # Age in decades
             gender +              # Binary gender variable (e.g., 1 = Female, 0 = Male)
             raceB +               # Dummy variable for Race category B
             raceA +               # Dummy variable for Race category A
             raceNULL +            # Dummy variable for Race missing/null
             MFH,                  # Enrollment in Medical Foster Home (1 = Yes, 0 = No)
           family = binomial(link = "logit"), # Specify logistic regression model
           data = df             # Use the cleaned dataset df
)

summary(model)

################################### PART C
#########################################
# Step 1: Likelihood Ratio Test (Model Fit)
# Extract Deviance details
# Null deviance: deviance of a model with only the intercept (no predictors)
# Residual deviance: deviance of the full model with predictors
# These are used to evaluate model improvement
null_deviance     <- summary(model)$null.deviance
residual_deviance <- summary(model)$deviance
df_null           <- summary(model)$df.null    # Degrees of freedom for null model
df_residual       <- summary(model)$df.residual     # Degrees of freedom for fitted model
```

```
# Compute test statistic
# The difference in deviance follows a chi-squared distribution under the null hypothesis
chi_sq_statistic <- null_deviance - residual_deviance
df_fit <- df_null - df_residual    # Degrees of freedom = number of predictors

# Compute p-value
# The p-value tells us if the full model provides a significantly better fit than the null
p_value <- 1 - pchisq(chi_sq_statistic, df_fit)

# Print results of the Likelihood Ratio Test
cat("Likelihood Ratio Test:\n")
cat("----------------------\n")
cat("Chi-squared Test Statistic:", chi_sq_statistic, "\n")
cat("Degrees of Freedom:", df_fit, "\n")
cat("p-value:", p_value, "\n")

#-------------------------------------------------------------------------------#

# Step 2. Extract and Rank Coefficients by Significance
# Extracting coefficients (excluding intercept)
# This is used for further analysis (e.g., identifying top predictors)
coefficients <- summary(model)$coefficients[-1, ]    # Removes the intercept row

# Rank predictors by absolute value of z-value (most significant first)
top_predictors_ranked <- order(abs(coefficients[, "z value"]), decreasing = TRUE)

# Select the indices of the top 4 predictors
top_4_indices <- top_predictors_ranked[1:4]

#Store their coefficient data
top_4_data <- coefficients[top_4_indices, ]

#-------------------------------------------------------------------------------#

# Step 3: Display Top 4 Predictors with Pretty Labels

# Create a named vector to map technical names to human-readable labels
pretty_names <- c(
  "MFH" = "Medical Foster Care",
```

```
  "age" = "Age",
  "bowelincontinence_365" = "Bowel Incontinence",
  "dressing_365" = "Assistance to Dress"
)

# Updated loop using pretty labels
cat("\nTop 4 Predictors of Survival:\n")
cat("-----------------------------\n")

# Loop through the top 4 predictor indices
for (index in top_4_indices) {
  var_name <- rownames(coefficients)[index]    # get variable name
  # Match pretty label if available; otherwise use original name
  display_name <- ifelse(var_name %in% names(pretty_names), pretty_names[var_name],
var_name)
  # Print predictor statistics
  cat("Predictor:", display_name, "\n",
      "Coefficient:", coefficients[index, "Estimate"], "\n",
      "Standard Error:", coefficients[index, "Std. Error"], "\n",
      "z-value:", coefficients[index, "z value"], "\n",
      "p-value:", coefficients[index, "Pr(>|z|)"], "\n\n")
}
```