Problem Set 3

Rajvi Jasani

GitHub Repository

This is the link to my GitHub repository https://github.com/rajvijasani/STATS506-Problem-Set-3.git

Problem 1 - Vision

a.

```r
library(knitr)
library(haven)
vision_data <- read_xpt("data/VIX_D.XPT")
demo_data <- read_xpt("data/DEMO_D.XPT")
# merge() by default performs an inner join,
# i.e. joins data which matches both tables
merged_data <- merge(vision_data, demo_data, by = "SEQN")
print(c(
  "rows" = nrow(merged_data),
  "cols" = ncol(merged_data)
))
```

```
rows cols
6980   99
```

Attribution of source: Asked ChatGPT for a function to export .xpt file. Referred to R Documentation for specifications of merge().

b.

```r
# according to the data documentation, ages above 85 have been topcoded at 85
age_brackets <- c("0-9",
                  "10-19",
                  "20-29",
                  "30-39",
                  "40-49",
                  "50-59",
                  "60-69",
                  "70-79",
                  "80-89")
# adding the age range variable to the data
merged_data$AGERANGE <- cut(
  merged_data$RIDAGEYR,
  breaks = seq(0, 90, by = 10),
  right = FALSE,
  labels = age_brackets
)
# table of number of observations for each age range; total 6980
all_age <- table(merged_data$AGERANGE)
# data frame consisting of respondents that wear
# glasses/contact lenses; VIQ220=1; total 2765
all_glasses <- merged_data[!is.na(merged_data$VIQ220) &
                             merged_data$VIQ220 == 1, ]
# table of number of observations of respondents
# wearing glasses/contact lenses for each age range; total 2765
glasses_age <- table(all_glasses$AGERANGE)
proportion <- glasses_age / all_age
# data frame combining age bracket and corresponding proportion
proportion_age <- data.frame(Age = names(proportion),
                             Proportion = as.numeric(proportion))
# rounding off to 5 edcimals points
proportion_age$Proportion <- round(proportion_age$Proportion, 5)
# replacing NaN with zero
proportion_age$Proportion[is.nan(proportion_age$Proportion)] <- 0
# nice table
kable(
  proportion_age,
  col.names = c(
    "Age bracket",
    "Proportion of glasses/contact lens users for distant vision"
  ),
  format = "html"
)
```

Age bracket	Proportion of glasses/contact lens users for distant vision
0-9	0.00000
10-19	0.30358
20-29	0.29971
30-39	0.32885
40-49	0.35092
50-59	0.53090
60-69	0.59304
70-79	0.63753
80-89	0.58101

Attribution of source: Asked ChatGPT for a function to get age values as a range. Asked ChatGPT on how to use the kable function for creating a clean table.

c.

```
# VIQ220 = Glasses/Contact lenses worn for
# distance vision - 433 missing, 2 'don't know'
# RIDAGEYR = Age in years - 0 missing
# RIDRETH1 = Race/Ethnicity - 0 missing
# RIAGENDR = Gender - 0 missing
# INDFMPIR = Poverty income ratio - 342 missing

# cleaning data before modeling
# recoding VIQ220=9 ('don't know') responses to NA
merged_data$VIQ220[merged_data$VIQ220 == 9] <- NA
# removing observations where response or predictors have NA values
cleaned_data <- na.omit(merged_data[, c("VIQ220", "RIDAGEYR", "RIDRETH1", "RIAGENDR", "INDFMI
# recoding VIQ220 'No' values from 2 to 0
cleaned_data$VIQ220[cleaned_data$VIQ220 == 2] <- 0
# recoding RIAGENDR 'female' values from 2 to 0
cleaned_data$RIAGENDR[cleaned_data$RIAGENDR == 2] <- 0

# model 1
model_1 <- glm(VIQ220 ~ RIDAGEYR, data = cleaned_data, family = "binomial")
odds_ratio_1 <- round(exp(coefficients(model_1)), 5)
s_size_1 <- nobs(model_1)
psuedo_r2_1 <- round(1 - (model_1$deviance / model_1$null.deviance), 5)
```

```r
aic_1 <- round(AIC(model_1), 5)

# model 2
model_2 <- glm(VIQ220 ~ RIDAGEYR + RIDRETH1 + RIAGENDR,
               data = cleaned_data,
               family = "binomial")
odds_ratio_2 <- round(exp(coefficients(model_2)), 5)
s_size_2 <- nobs(model_2)
psuedo_r2_2 <- round(1 - (model_2$deviance / model_2$null.deviance), 5)
aic_2 <- round(AIC(model_2), 5)

# model 3
model_3 <- glm(VIQ220 ~ RIDAGEYR + RIDRETH1 + RIAGENDR + INDFMPIR,
               data = cleaned_data,
               family = "binomial")
odds_ratio_3 <- round(exp(coefficients(model_3)), 5)
s_size_3 <- nobs(model_3)
psuedo_r2_3 <- round(1 - (model_3$deviance / model_3$null.deviance), 5)
aic_3 <- round(AIC(model_3), 5)

# data frame with all summaries
model_summaries <- data.frame(
  Model = c("Model 1", "Model 2", "Model 3"),
  Sample_Size = c(s_size_1, s_size_2, s_size_3),
  Psuedo_R2 = c(psuedo_r2_1, psuedo_r2_2, psuedo_r2_3),
  AIC = c(aic_1, aic_2, aic_3),
  OR_Intercept = c(odds_ratio_1["(Intercept)"], odds_ratio_2["(Intercept)"], odds_ratio_3["(I
  OR_Age = c(odds_ratio_1["RIDAGEYR"], odds_ratio_2["RIDAGEYR"], odds_ratio_3["RIDAGEYR"]),
  OR_Race = c(NA, odds_ratio_2["RIDRETH1"], odds_ratio_3["RIDRETH1"]),
  OR_Gender = c(NA, odds_ratio_2["RIAGENDR"], odds_ratio_3["RIAGENDR"]),
  OR_PIR = c(NA, NA, odds_ratio_3["INDFMPIR"])
)
# nice table
kable(
  model_summaries,
  caption = "Logistic Regression Model Comparison",
  col.names = c(
    "Model",
    "Sample Size",
    "Psuedo R2",
    "AIC",
    "OR Intercept",
    "OR Age",
    "OR Race",
    "OD Gender",
    "OD PIR"
  ),
  format = "html"
)
```

Table 2: Logistic Regression Model Comparison

Model	Sample Size	Psuedo R2	AIC	OR Intercept	OR Age	OR Race	OD Gender	OD PIR
Model 1	6247	0.04733	8119.871	0.29265	1.02452	NA	NA	NA
Model 2	6247	0.06075	8009.571	0.25997	1.02491	1.12684	0.60693	NA
Model 3	6247	0.06906	7940.790	0.20251	1.02405	1.09722	0.59536	1.15327

Attribution of source: Referred to R Documentation for glm(). Used ChatGPT to understand how to get odds ratios, McFadden's pseudo R^2 and AIC values.

d.

```
print(coefficients(model_3)["RIAGENDR"])
```

```
  RIAGENDR
-0.5185954
```

```
print(odds_ratio_3["RIAGENDR"])
```

```
RIAGENDR
 0.59536
```

The coefficient has a negative value (approx -0.5186) which indicates that a man (1) is less likely to wear glasses or contact lenses. The odds ratio (approx 0.59) indicates that the odds of men wearing glasses or contact lenses is 59% lower than odds of women wearing glasses or contact lenses.

```
# rows=gender, cols=glasses/contact lenses
gender_glasses<-table(cleaned_data$RIAGENDR,cleaned_data$VIQ220)
print(gender_glasses)
```

```
       0    1
  0 1673 1521
  1 1919 1134
```

```r
print(chisq.test(gender_glasses))
```

```

	Pearson's Chi-squared test with Yates' continuity correction

data:  gender_glasses
X-squared = 69.683, df = 1, p-value < 2.2e-16
```

The Chi-square test with a Chi-square statistic of 69.683 and p value of near zero indicates that proportion of glasses/contact lenses significantly differs based on the gender.

Attribution of source: Used ChatGPT to get the function for Chi-squared test.

Problem 2 - Sakila

```r
library(DBI)
library(RSQLite)
sakila <- dbConnect(SQLite(), "data/sakila_master.db")
```

a.

```r
dbGetQuery(
  sakila,
  "SELECT release_year, COUNT(film_id)
  FROM film
  GROUP BY release_year
  ORDER BY release_year
  LIMIT 1"
)
```

```
  release_year COUNT(film_id)
1         2006           1000
```

b.

SQL + R:

```r
film <- dbGetQuery(sakila, "SELECT * FROM film_category")
category <- dbGetQuery(sakila, "SELECT * FROM category")
merged_fc <- merge(film, category, by = "category_id")
m_count <- table(merged_fc$name)
print(m_count[which.min(m_count)])
```

```
Music
   51
```

SQL only:

```r
dbGetQuery(
  sakila,
  "SELECT c.name AS genre, COUNT(fc.film_id) as m_count
  FROM category as c
  INNER JOIN film_category as fc ON c.category_id = fc.category_id
  GROUP BY c.name
  ORDER BY m_count
  LIMIT 1"
)
```

```
  genre m_count
1 Music      51
```

c.

SQL + R:

```r
customer <- dbGetQuery(sakila, "SELECT customer_id, address_id FROM customer")
address <- dbGetQuery(sakila, "SELECT address_id, city_id FROM address")
city <- dbGetQuery(sakila, "SELECT city_id, country_id FROM city")
country <- dbGetQuery(sakila, "SELECT country_id, country FROM country")

merge_ca <- merge(customer, address, by = "address_id")
merge_cac <- merge(merge_ca, city, by = "city_id")
merge_all <- merge(merge_cac, country, by = "country_id")
c_count <- table(merge_all$country)
print(c_count[c_count == 13])
```

```
Argentina   Nigeria
       13        13
```

SQL only:

```
dbGetQuery(
  sakila,
  "SELECT country, COUNT(country) AS c_count
  FROM country as co
  RIGHT JOIN
    (SELECT country_id
    FROM city as ci
    RIGHT JOIN
      (SELECT city_id
      FROM address as a
      RIGHT JOIN customer AS c ON a.address_id=c.address_id
      ) AS ac ON ci.city_id=ac.city_id
    ) AS acc ON co.country_id=acc.country_id
  GROUP BY country
  HAVING c_count==13"
)
```

```
    country c_count
1 Argentina      13
2   Nigeria      13
```

Problem 3 - US Record

```
us500 <- read.csv("data/us-500.csv")
```

a.

```
print(sum(grepl(".com$", us500$email)) / nrow(us500))
```

```
[1] 0.732
```

b.

```
emails <- us500$email
# removing the TLD
emails_no_tld <- sub("\\.[a-zA-Z]{2,}$", "", emails)
print(sum(grepl("[^a-zA-Z0-9@]", emails_no_tld)) / nrow(us500))
```

```
[1] 0.506
```

c.

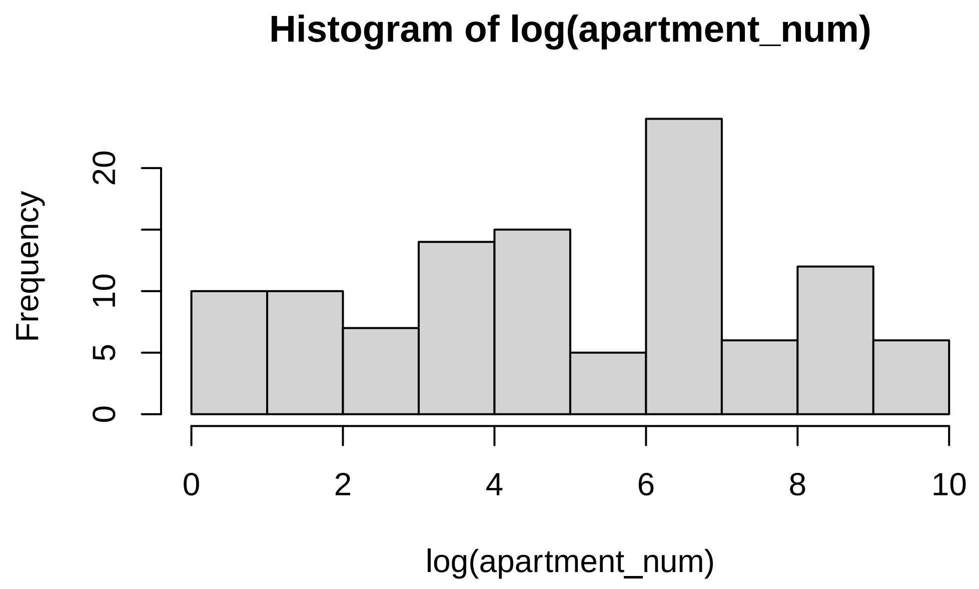
```
area_p1 <- substr(us500$phone1, 1, 3)
area_p2 <- substr(us500$phone2, 1, 3)
all_area <- c(area_p1, area_p2)
print(sort(table(all_area), decreasing = TRUE)[1:5])
```

```
all_area
973 212 215 410 201 
 36  28  28  28  24 
```

d.

For this question, I quickly browsed through the data to notice that there were 2 types of observations that has numbers at the end, one where the number was preceded by a '#' (e.g. '#5435') and others that did not (e.g. 'Highway 67'). The other case clearly does not indicate an apartment number so I will only be extracting numbers at the end that are preceded by a '#'.

```
# checking if an address ends in # followed by numbers
# if yes, we substitute the address
# with the apartment number captured in the capture group
# else, we substitute NA
apartment_num <- ifelse(grepl("#[0-9]+$", us500$address),
                        sub(".*#([0-9]+)$", "\\1", us500$address),
                        NA)
# cleaning data by removing NA and converting to numeric values
apartment_num <- na.omit(apartment_num)
apartment_num <- as.numeric(apartment_num)
#
hist(log(apartment_num))
```



Attribution of source: Asked ChatGPT for idea to handle address that do not end in '#' followed by number.

e.

```
# frequency (in %) of occurrence of leading digits in our data
print(table(substr(apartment_num, 1, 1)) / length(apartment_num) * 100)
```

```

        1         2         3         4         5         6         7         8 
10.091743 10.091743 11.009174 10.091743 12.844037  9.174312 11.009174 10.091743 
        9 
15.596330 
```

```
# frequency (in %) of occurrence of leading digits according to Benford's law
print((log10((1:9) + 1) - log10(1:9)) * 100)
```

```
[1] 30.103000 17.609126 12.493874  9.691001  7.918125  6.694679  5.799195
[8]  5.115252  4.575749
```

Looking at the frequencies of occurrence of leading digits in our data and the frequencies that each digit should follow according to Benford's law, we can clearly see that Benford's law is not followed, rather they have almost equal frequencies. And that's why the apartment numbers can not pass as real data.

Attribution of source: Wikipedia page on Benford's law for frequency distribution