ARE 212 Problem Set 4

Aline Abayo, Eleanor Adachi, Anna Cheyette, Karla Neri, Stephen Stack

2024-02-29

```
options(scipen = 999)

# Load packages
library(pacman)
p_load(tidyverse,haven, readr, knitr, readxl, psych,
       stats4, qwraps2, car, lmtest, stargazer, margins,
       sandwich)

# get directory of current file
current_directory <-
  dirname(dirname(rstudioapi::getSourceEditorContext()$path))

# read in data
my_data <- read_dta(file.path(current_directory, "data",
                              "pset4_2024.dta"))
```

Exercise 1

1.

Estimate the model: $logquantity_i = \beta_1 + fuel_i\beta_2 + logprice_i\beta_3 + weight_i\beta_4 + \epsilon_i$.

```
my_data <-
  my_data %>%
  mutate(log_quantity = log(qu),
         log_price = log(price))

# 1. Estimate the model via OLS
y1 <- my_data$log_quantity
X1 <- cbind(1, my_data$fuel, my_data$log_price, my_data$weight)
(b1 <- solve(t(X1) %*% X1) %*% t(X1) %*% y1)
```

```
##              [,1]
## [1,] 13.021293997
## [2,] -0.091952918
## [3,] -0.797538193
## [4,] -0.001074254
```

2.

Conduct a Breusch Pagan Test for heteroskedastic errors using the canned reg12<-lm(lqu~ etc) and bptest(reg13). Do we have a problem?

```r
reg12 <- lm(log_quantity ~ fuel + log_price + weight, data = my_data)
summary(reg12)
```

```
##
## Call:
## lm(formula = log_quantity ~ fuel + log_price + weight, data = my_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3033 -0.9477  0.1108  1.0524  3.1891
##
## Coefficients:
##              Estimate Std. Error t value             Pr(>|t|)
## (Intercept) 13.021294   0.223353  58.299 < 0.0000000000000002 ***
## fuel        -0.091953   0.025616  -3.590              0.00034 ***
## log_price   -0.797538   0.148890  -5.357         0.0000000953 ***
## weight      -0.001074   0.000240  -4.475         0.0000080823 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.404 on 1874 degrees of freedom
## Multiple R-squared:  0.1787, Adjusted R-squared:  0.1774
## F-statistic: 135.9 on 3 and 1874 DF,  p-value: < 0.00000000000000022
```

```r
# conduct breusch-pagan test
(bp1 <- bptest(reg12))
```

```
##
##  studentized Breusch-Pagan test
##
## data:  reg12
## BP = 56.873, df = 3, p-value = 0.000000000002735
```

The very low p-value from the Breusch Pagan test of 0 indicates the presence of heteroskedasticity in the residuals of our regression model. This violates the assumption of homoskedasticity, indicating that we do have a problem.

3.

Calculate the White robust standard errors. Comment on how they compare to the traditional OLS standard errors. Is this the right way to go about dealing with potential heterogeneity problems?

```r
# projection matrix
P1 <- X1 %*% solve(t(X1) %*% X1) %*% t(X1)

# residual maker of reg y1 on x: M = I - P
M1 <- diag(length(y1)) - P1

# residuals
e1 <- M1 %*% y1

# white robust std errors of OLS estimates
# inv(X'X) (\sum Xi' ei ei Xi) inv(X'X)

X1e <- cbind(e1, my_data$fuel*e1, my_data$log_price * e1, my_data$weight * e1)
Vb1_whiteRobust <- solve(t(X1) %*% X1) %*% t(X1e) %*% X1e %*% solve(t(X1) %*% X1)
```

```r
# get standard errors
(seb1_whiteRobust<-sqrt(diag(Vb1_whiteRobust)))
```

```
## [1] 0.2238751058 0.0232433645 0.1485702322 0.0002495783
```

- Intercept: The robust standard error is slightly higher than the traditional OLS standard error.
- fuel: The robust standard error is slightly lower than the traditional OLS standard error.
- log_price: The robust standard error is slightly lower than the traditional OLS standard error, albeit very similar.
- weight: The robust standard error is slightly higher than the traditional OLS standard error.

The differences in standard errors suggest that the assumption of homoscedasticity may not hold for this model, as the Breusch-Pagan test also indicated. When the robust standard errors differ significantly from the traditional OLS standard errors, it implies that the variance of the residuals is not constant, and the OLS standard errors may not be fully reliable for inference.

Is this the right way to go about dealing with potential heterogeneity problems? - Not sure??

4.

Suppose that there is a model where a structural parameter of interest γ is defined as $\gamma = log(\beta_2+2)(\beta_3+3\beta_4)$. Using the OLS estimation results from eq. 1, calculate $\hat{\gamma}$ and its white standard error.

```r
beta_2 <- b1[2]
beta_3 <- b1[3]
beta_4 <- b1[4]

# Calculate gamma_hat
(gamma_hat <- log(beta_2 + 2) * (beta_3 + 3 * beta_4))
```

```
## [1] -0.5173558
```

```r
# Gradient of gamma_hat w.r.t. beta_2, beta_3, and beta_4
gradient_gamma <- c(
  1 / (beta_2 + 2) * (beta_3 + 3 * beta_4),
  log(beta_2 + 2),
  3 * log(beta_2 + 2)
)

# Remove the first row and column for intercept
Vb_submatrix <- Vb1_whiteRobust[-1, -1]

# Calculate the variance of gamma_hat using the Delta Method
var_gamma_hat <- t(gradient_gamma) %*% Vb_submatrix %*% gradient_gamma

# Standard error of gamma_hat
(se_gamma_hat <- sqrt(var_gamma_hat))
```

```
##             [,1]
## [1,] 0.1008321
```

$\hat{\gamma}$ is -0.5173558 and the white standard error is 0.1008321.

Exercise 2

eq. 2: $log(quantity) = beta_0 + fuel_i\beta_1 + log(price)_i\beta_2 + weight_i\beta_4 + luxury\beta_5 + \epsilon_i$

```r
# estimate the model via OLS

X2 <- cbind(1, my_data$fuel, my_data$log_price, my_data$year,
            my_data$weight, my_data$luxury)

(b2 <- solve(t(X2) %*% X2) %*% t(X2) %*% y1)
```

```
##               [,1]
## [1,] 68.7789505837
## [2,] -0.1524851356
## [3,] -1.4418206274
## [4,] -0.0274178787
## [5,] -0.0002186054
## [6,]  0.9853449176
```

1.

Please interpret your results in terms of the log price variable OLS coefficient.

The OLS coefficient for log price is -1.4418206. The negative value represents the inverse relationship between price and quantity demanded. A a 1% increase in price is associated with a -1.4418206% decrease in quantity demanded, holding other factors constant.

2.

- Fuel efficiency: more fuel efficient cars may be priced higher due to technology needed to achieve better fuel economy. Consumers looking to reduce long term fuel costs might prefer these vehicles, affecting their quantity demanded.
- Temporal changes: prices (due to inflation, taxation, etc.) change over time for a variety of reasons.

The omission of "year" from equation 1 indicates that bias arises because "year" is likely correlated with both the dependent variable (log quantity) and independent variables such as log price.

3.

if we omit fuel from equation (eq.2) how does your OLS estimate of the log price change? What does this imply about the covariance between log price and fuel?

```r
# omit fuel and re-run
X2_2 <- cbind(1, my_data$log_price, my_data$year, my_data$weight,
              my_data$luxury)

(b2_2 <- solve(t(X2_2) %*% X2_2) %*% t(X2_2) %*% y1)
```

```
##               [,1]
## [1,] 23.1508249820
## [2,] -1.3808902626
## [3,] -0.0046579168
## [4,] -0.0009874881
## [5,]  0.9964318660
```

Omitting fuel from the equation and re-running the regression changes the OLS estimate for the log price variable from -1.4418206 to -1.3808903. This change in the coefficient for log price indicates that there is covariance between log price and fuel that was influencing the original estimate.

Exercise 3

eq. 3: $logprice_i = \alpha_0 + fuel_i\alpha_1 + year\alpha_2 + weight_i\alpha_3 + luxury_i\alpha_4 + average\_o1_i\alpha_5 + v_i$

```
# add log average other
my_data <-
  my_data %>%
  mutate(laverage_o1 = log(average_o1))

X3 <- cbind(1, my_data$fuel, my_data$year, my_data$weight,
            my_data$luxury, my_data$laverage_o1)
y3 <- my_data$log_price

(b3 <- solve(t(X3) %*% X3) %*% t(X3) %*% y3)
```

```
##                [,1]
## [1,] 25.5281059245
## [2,] -0.0026079937
## [3,] -0.0126962776
## [4,]  0.0004580575
## [5,]  0.0053356366
## [6,]  0.7720019382
```

The coefficient of average_o1 is 0.7720019.

Exercise 4

eq. 4: $logquantity_i = \alpha_0 + fuel_i\alpha_1 + year\alpha_2 + weight_i\alpha_3 + luxury_i\alpha_4 + average\_o1_i\alpha_5 + v_i$

```
X3_2 <- cbind(1, my_data$fuel, my_data$year, my_data$weight,
            my_data$luxury, my_data$laverage_o1)
y3_2 <- my_data$log_quantity

(b3_2 <- solve(t(X3_2) %*% X3_2) %*% t(X3_2) %*% y3_2)
```

```
##               [,1]
## [1,] 26.176493455
## [2,] -0.146165397
## [3,] -0.006309850
## [4,] -0.001169946
## [5,]  0.936857361
## [6,] -0.927842632
```

The coefficient on average_o1 is -0.9278426. It suggests that an increase in the average log prices of the same car type in other European countries is associated with a decrease in the log of quantity sold in the dataset's country, holding other factors constant.

Exercise 5

1.

```
# unclear what to do here if not the same as the next
```

2.

```
# first stage regression including log average other as a regressor
y_price <- my_data$log_price
X5_s1 <- cbind(1, my_data$laverage_o1, my_data$fuel, my_data$year,
               my_data$weight, my_data$luxury)

# First stage regression with log price
b5_s1 <- solve(t(X5_s1) %*% X5_s1) %*% t(X5_s1) %*% y_price

# Predicted logprice from the first stage
logprice_hat <- X5_s1 %*% b5_s1

# Dependent variable for the second stage, log(quantity)
y_quantity <- my_data$log_quantity

# Independent variables for the second stage, replacing logprice with logprice_hat
X5_s2 <- cbind(1, logprice_hat, my_data$fuel, my_data$year, my_data$weight,
              my_data$luxury)  # Use the predicted logprice from the first stage

# Second stage regression
(b5_s2 <- solve(t(X5_s2) %*% X5_s2) %*% t(X5_s2) %*% y_quantity)
```

```
##               [,1]
## [1,] 56.8578477547
## [2,] -1.2018656767
## [3,] -0.1492998550
## [4,] -0.0215690704
## [5,] -0.0006194225
## [6,]  0.9432700790
```

The estimate of the log price coefficient from the 2SLS regression is -1.2018657. This represents the estimated effect of a 1% increase in the log price on the log quantity of cars sold, using the average log prices in other European countries (average_o1) as an instrument to address potential endogeneity between log price and log quantity. The coefficient suggests negative relationship between price and quantity demanded - it implies that a 1% increase in price is associated with a -1.2018657% decrease in quantity of cars sold.

3.

Estimate the first-stage regression and, in the second stage, use logprice and also include the residuals from the first stage in the second stage, following a control function approach.

```
# Calculate residuals from the first stage
residuals1 <- y_price - X5_s1 %*% b5_s1

# Independent variables for the second stage, including logprice and
# first-stage residuals
X5_3_s2 <- cbind(1, my_data$log_price, my_data$fuel, my_data$year,
                 my_data$weight, my_data$luxury, residuals1)

# Second-stage regression
(b5_3_s2 <- solve(t(X5_3_s2) %*% X5_3_s2) %*% t(X5_3_s2) %*% y_quantity)
```

```
##               [,1]
## [1,] 56.8578477602
## [2,] -1.2018656767
```

```
## [3,] -0.1492998550
## [4,] -0.0215690704
## [5,] -0.0006194225
## [6,]  0.9432700790
## [7,] -0.4543582724
```

4.

```r
# extract coefficient on log price from first stage regression
lprice_coef_s1 <- b5_s1[2]

# Reduced-form regression: Regress log_quantity directly on all exogenous variables
# use X5_s1 = constant + log(average_o1) + fuel + year + weight + luxury
b5_rf <- solve(t(X5_s1) %*% X5_s1) %*% t(X5_s1) %*% y_quantity

# extract coefficient for instrument laverage_o1
rf_coef <- b5_rf[2]

# Compute the 2SLS estimate for the log_price coefficient
# This is done by dividing the reduced-form coefficient by the first-stage coefficient
(iv_estimate <- rf_coef / lprice_coef_s1)
```

```
## [1] -1.201866
```

5.

We get -1.2 for all.

6.

The OLS estimate for log price is -0.8 while the 2SLS estimate is -1.2 a difference of -0.4. This difference in estimates suggests that the OLS regression might have been biased due to omitted variable bias or simultaneous causality, where log_price could be correlated with the error term ϵ_i.

```r
# calculate standard errors from 2SLS

# Calculate residuals
residuals_2sls <- y_quantity - X5_s2 %*% b5_s2

# Calculate variance of residuals (S2)
n <- nrow(X5_s2)  # Number of observations
k <- ncol(X5_s2)  # Number of parameters estimated (including intercept)
S2 <- as.numeric((t(residuals_2sls) %*% residuals_2sls) / (n - k))

# Calculate the variance-covariance matrix (V) of the 2SLS regression coefficients
V <- solve(t(X5_s2) %*% X5_s2) * S2

# Calculate standard errors (se) of the coefficients
(se <- sqrt(diag(V)))
```

```
## [1] 18.4368653963  0.2772925535  0.0348213699  0.0091371697  0.0005145661
## [6]  0.1404462376
```

```r
#TODO: comparison of standard errors
```

7.

```
# add log average_o2
my_data <-
  my_data %>%
  mutate(laverage_o2 = log(average_o2))

Z <- cbind(1, my_data$laverage_o1, my_data$laverage_o2)
residuals_regression <- solve(t(Z) %*% Z) %*% t(Z) %*% residuals_2sls

# Calculate R-squared from the regression of 2SLS residuals on instruments
residuals_predicted <- Z %*% residuals_regression
residuals_variance <- t(residuals_2sls) %*% residuals_2sls
explained_variance <- t(residuals_predicted) %*% residuals_predicted
R_squared <- as.numeric(explained_variance / residuals_variance)

N <- nrow(Z)  # Number of observations
q1 <- ncol(Z) - 1  # Number of instruments minus the number of endogenous variables
test_statistic <- N * R_squared

# Step 4: Compare the test statistic to the chi-squared distribution
p_value <- 1 - pchisq(test_statistic, df = q1)

# Output the test statistic and p-value
list(test_statistic = test_statistic, p_value = p_value)
```

```
## $test_statistic
## [1] 43.24473
##
## $p_value
## [1] 0.000000000406936
```

```
# is this correct??
```

Given the very low p-value, this result suggests that we can reject the null hypothesis of the validity of the overidentifying restrictions. This indicates that at least one of the instruments is likely correlated with the error term ϵ_i, violating the assumption that instruments must be exogenous.

Exercise 6

1.

```
# read in PBM data
pbm <- read_dta(file.path(current_directory, "data", "pset4_PBM_2024.dta"))


pbm <-
  pbm %>%
  # filter to only those who chose noChoice == 0
  filter(noChoice == 0,
         # filter out NA values
         !is.na(chosePBM),
         !is.na(yourage),
         !is.na(treat),
```

```
         !is.na(relprice))
```

2.

```
lpm_model <- lm(chosePBM ~ yourage + treat + relprice, data = pbm)

# Calculate robust standard errors
(coeftest(lpm_model, vcov = vcovHC(lpm_model, type = "HC1")))
```

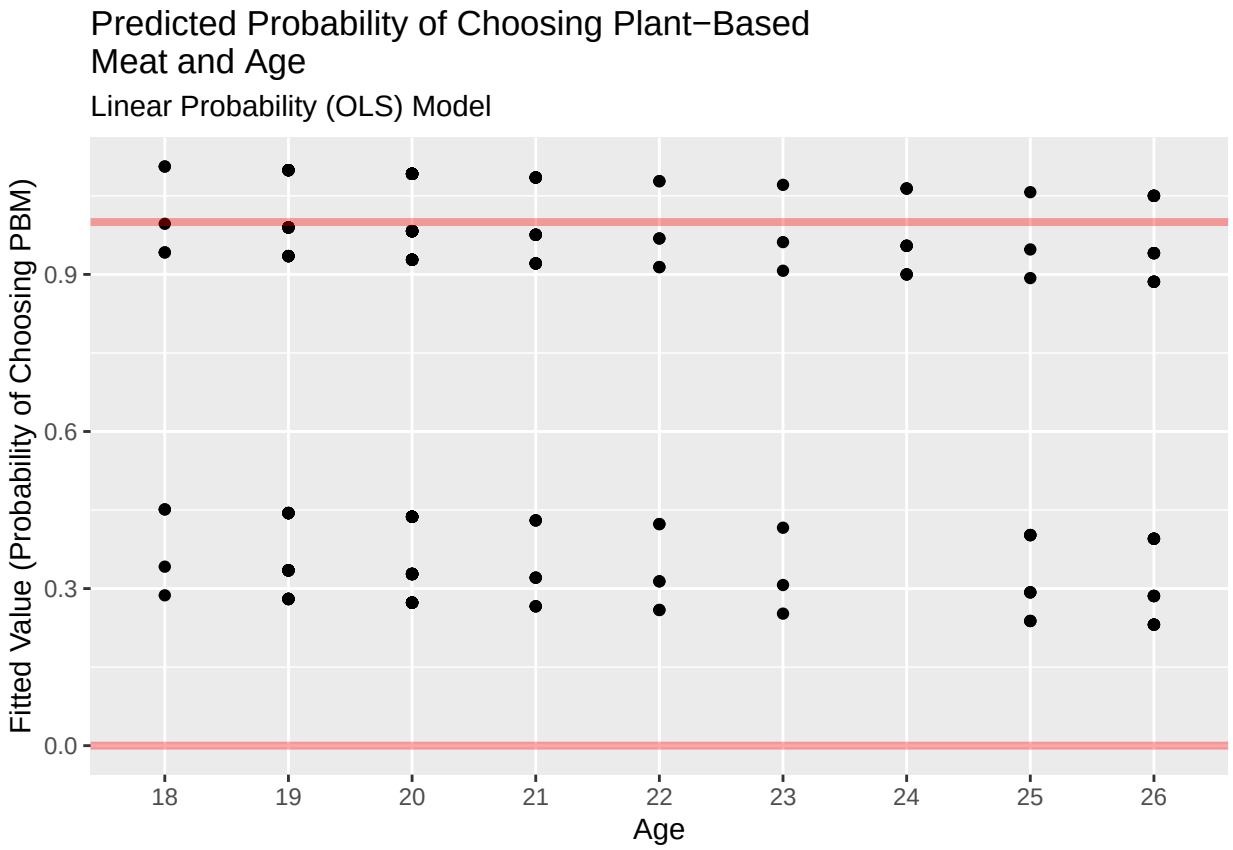
```
##
## t test of coefficients:
##
##               Estimate Std. Error t value              Pr(>|t|)
## (Intercept)  0.7955118  0.2220245  3.5830             0.0004060 ***
## yourage     -0.0070039  0.0093038 -0.7528             0.4522565
## treat        0.6548708  0.0417453 15.6873 < 0.00000000000000022 ***
## relprice    -0.2729136  0.0810310 -3.3680             0.0008727 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.

```
# add fitted values to dataframe
pbm <-
  pbm %>%
  mutate(reg_fit = lpm_model$fitted.values)

# plot predicted values and education
ggplot(pbm, aes(x = factor(yourage), y = reg_fit)) +
  geom_point() + # add points
  # add horizontal line at y=0
  geom_hline(yintercept=0, size = 1.4, alpha = 0.35, color = "red") +
  # add horizontal line at y=1
  geom_hline(yintercept=1, size = 1.4, alpha = 0.35, color = "red") +
  # generate labels
  labs(title = str_wrap("Predicted Probability of Choosing Plant-Based Meat and Age",
                        45),
       subtitle = "Linear Probability (OLS) Model",
       x = "Age",
       y = "Fitted Value (Probability of Choosing PBM)")
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



The maximum predicted value is 1.1060906, which is greater than 1. This value is meaningless for a binary variable. The linear probability model does not inherently constrain the predicted values to fall within the 0 to 1 range, so it can be problematic for binary dependent variables.

4.

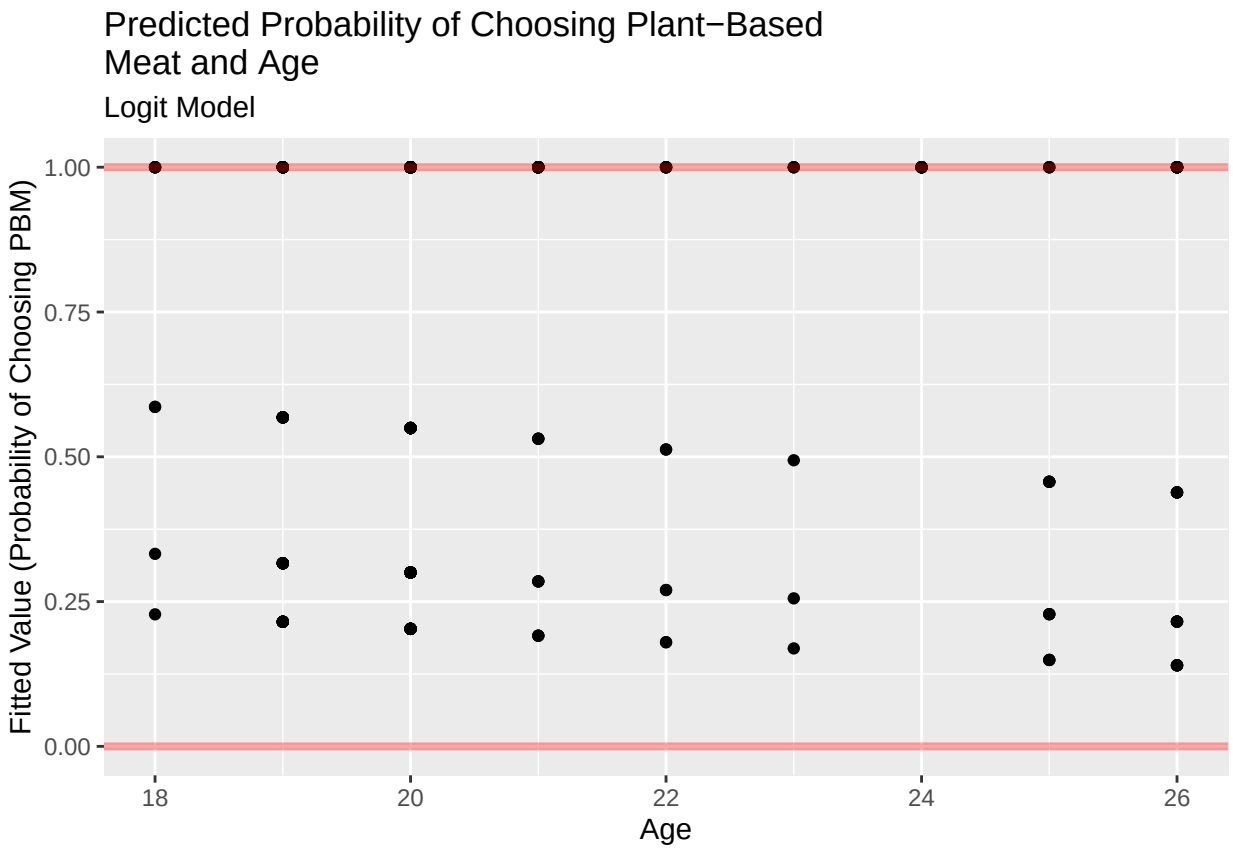
```
logit1 <- glm(chosePBM ~ yourage + treat + relprice, pbm,
              family = binomial(link = "logit"))
summary(logit1)
```

```
##
## Call:
## glm(formula = chosePBM ~ yourage + treat + relprice, family = binomial(link = "logit"),
##     data = pbm)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    3.77357    1.98666   1.899 0.057505 .
## yourage       -0.07444    0.08353  -0.891 0.372805
## treat         21.40387 1471.09693   0.015 0.988392
## relprice      -2.60801    0.78828  -3.308 0.000938 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 327.20  on 261  degrees of freedom
```

```
## Residual deviance: 149.36  on 258  degrees of freedom
## AIC: 157.36
##
## Number of Fisher Scoring iterations: 19
```

```
pbm <-
  pbm %>%
  mutate(log1_fit = logit1$fitted.values)


# produce figure for logit model
ggplot(pbm, aes(x = yourage, y = log1_fit)) +
  # First add points, color determined by whether in or out of [0,1]
  geom_point() + # add points
  geom_hline(yintercept=0, size = 1.4, alpha = 0.35, color = "red") +
  geom_hline(yintercept=1, size = 1.4, alpha = 0.35, color = "red") +
  labs(title = str_wrap("Predicted Probability of Choosing Plant-Based Meat and Age",
                        45),
       subtitle = "Logit Model",
       x = "Age",
       y = "Fitted Value (Probability of Choosing PBM)")
```



Did the logit specification fix the problem in 3?

Yes, the logit model does not predict any values above 1.

5.

```
# marginal effect for the logit model
summary(margins(logit1))
```

```
##    factor     AME       SE       z      p     lower    upper
##  relprice -0.2544   0.0638 -3.9873 0.0001   -0.3795  -0.1294
##     treat  2.0879 143.5001  0.0145 0.9884 -279.1672 283.3429
##   yourage -0.0073   0.0081 -0.9007 0.3677   -0.0231   0.0085
```

```
# create dataframe of mean data
meandata <-
  pbm %>%
  select(yourage, treat, relprice) %>%
  summarise_all(mean)

# marginal effect at the mean
summary(margins(logit1, data = meandata))
```

```
##    factor     AME     SE       z      p   lower  upper
##  relprice -0.0001 0.0602 -0.0013 0.9990 -0.1180 0.1179
##     treat  0.0006 0.4493  0.0014 0.9988 -0.8800 0.8813
##   yourage -0.0000 0.0017 -0.0013 0.9989 -0.0034 0.0034
```

Yes, the logit marginal effect for the means and the logit marginal effect using the original data are different.

should they be this different?? did I do something wrong here?

6.

```
# add being a vegetarian dummy and having had PBM before as covariates
logit2 <- glm(chosePBM ~ yourage + treat + relprice + youvegetarian + pbm,
              pbm, family = binomial(link = "logit"))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
(summary_logit2 <- summary(logit2))
```

```
##
## Call:
## glm(formula = chosePBM ~ yourage + treat + relprice + youvegetarian +
##     pbm, family = binomial(link = "logit"), data = pbm)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)      6.8871     3.5441   1.943 0.051985 .
## yourage         -0.2319     0.1538  -1.508 0.131545
## treat           23.7815  2128.9824   0.011 0.991088
## relprice        -4.6199     1.2811  -3.606 0.000311 ***
## youvegetarian   22.3130  3196.5662   0.007 0.994431
## pbm              1.9099     0.7324   2.608 0.009120 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 327.201  on 261  degrees of freedom
```

```
## Residual deviance:  72.831  on 256  degrees of freedom
## AIC: 84.831
##
## Number of Fisher Scoring iterations: 20
```

```
# Perform Wald test for 'treat' coefficient
wald_test_treat <- (summary_logit2$coefficients["treat", "Estimate"] /
                      summary_logit2$coefficients["treat", "Std. Error"])^2
p_value_treat <- 1 - pchisq(wald_test_treat, df = 1)

# Perform Wald test for 'youvegetarian' and 'pbm' coefficients
wald_test_vegetarian <- (summary_logit2$coefficients["youvegetarian", "Estimate"] /
                           summary_logit2$coefficients["youvegetarian", "Std. Error"])^2
p_value_vegetarian <- 1 - pchisq(wald_test_vegetarian, df = 1)

wald_test_pbm <- (summary_logit2$coefficients["pbm", "Estimate"] /
                    summary_logit2$coefficients["pbm", "Std. Error"])^2
p_value_pbm <- 1 - pchisq(wald_test_pbm, df = 1)


# Compile the results into a list
list(
  wald_test_treat = wald_test_treat,
  p_value_treat = p_value_treat,
  wald_test_vegetarian = wald_test_vegetarian,
  p_value_vegetarian = p_value_vegetarian,
  wald_test_pbm = wald_test_pbm,
  p_value_pbm = p_value_pbm
)
```

```
## $wald_test_treat
## [1] 0.0001247768
##
## $p_value_treat
## [1] 0.9910875
##
## $wald_test_vegetarian
## [1] 0.00004872449
##
## $p_value_vegetarian
## [1] 0.9944306
##
## $wald_test_pbm
## [1] 6.799167
##
## $p_value_pbm
## [1] 0.009120041
```

The treat variable has a p-value of 0.9910875, which is not statistically significant, so we fail to reject the null hypothesis that the treatment does not have an effect on the outcome of choosing plant-based meat. The vegetarian coeeficient has a p-value of 0.9944306, which is also not statistically significant. The pbm variable has a p-value of 0.00912, which is significant at the 1% confidence level. ## 7.

```
# define the negative log likelihood function
logl.logit <- function(theta, x, y){
```

```
  n <- nrow(x)
  x = as.matrix(x)

  beta <- theta[1:ncol(x)]

  loglik <-
    sum(y * log(exp(x %*% beta) / (1 + exp(x %*% beta))) +
    (1 - y) * log(1 - exp(x %*% beta) / (1 + exp(x %*% beta))))

  return(-loglik)

}

yW <- pbm$chosePBM
xW <- cbind(1, pbm$relprice, pbm$pbm)

resultsML <- optim(c(0,0,0), logl.logit, method = "BFGS", hessian = T,
                   x = xW, y = yW)

out <- list(beta = resultsML$par,
            vcov = solve(resultsML$hessian),
            ll = 2*resultsML$value)
out
```

```
## $beta
## [1]  1.700680 -1.614858  1.326138
##
## $vcov
##             [,1]        [,2]        [,3]
## [1,]  0.47946944 -0.37786377 -0.03845285
## [2,] -0.37786377  0.33752221 -0.01606893
## [3,] -0.03845285 -0.01606893  0.08769343
##
## $ll
## [1] 298.8559
```

```
(se <- sqrt(diag(out$vcov)))
```

```
## [1] 0.6924373 0.5809666 0.2961308
```

```
# compare with canned functions
logit_ML <- glm(chosePBM ~ relprice + pbm, pbm,
                    family = binomial(link = "logit"))
summary(logit_ML)
```

```
##
## Call:
## glm(formula = chosePBM ~ relprice + pbm, family = binomial(link = "logit"),
##     data = pbm)
##
## Coefficients:
##             Estimate Std. Error z value   Pr(>|z|)
## (Intercept)   1.7007     0.6924   2.456    0.01404 *
## relprice     -1.6149     0.5810  -2.780    0.00544 **
## pbm           1.3261     0.2961   4.478 0.00000753 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 327.20  on 261  degrees of freedom
## Residual deviance: 298.86  on 259  degrees of freedom
## AIC: 304.86
##
## Number of Fisher Scoring iterations: 4
```

The results are the same between the canned and the custom log-likelihood functions.

8.

```
# marginal effect
(marginsML <- margins(logit_ML))
```

```
## Average marginal effects
```

```
## glm(formula = chosePBM ~ relprice + pbm, family = binomial(link = "logit"),     data = pbm)
```

```
##  relprice    pbm
##   -0.3114 0.2557
```

```
# sample average of choosing PBM
(avg_pbm <- mean(pbm$chosePBM))
```

```
## [1] 0.6832061
```

```
# What percentage of the mean is the estimated marginal effect?
(percent_me <- summary(marginsML)$AME[["pbm"]]/avg_pbm * 100)
```

```
## [1] 37.42683
```

Having had PBM before increases the probability of choosing PBM in the survey by 37.4268315% among those who choose one of the two alternatives.

Exercise 7

```
 # A function to run the simulation
BiasSimulator <- function(simulationSize, sampleSize, trueBeta) {

    OLSBiasGenerator <- function(sampleSize, trueBeta) {
      # First generate x from N(0,1)
      x <- rnorm(n = sampleSize)
      # Now the error from N(0,1)
      e <- rnorm(n = sampleSize)
      # Now combine trueBeta, x, and e to get y
      y <- trueBeta[1] + trueBeta[2] * x + e
      # Define the data matrix of independent vars.
      X <- cbind(1, x)
      # Force y to be a matrix
      y <- matrix(y, ncol = 1)
      # Calculate the OLS estimates
      b.ols <- solve(t(X) %*% X) %*% t(X) %*% y
```

```r
  # Convert b_ols to vector
  b.ols %<>% as.vector()
  # Calculate bias, force to 2x1 data.frame()
  #in pset
  #sigma2bias<-true sigma-sigmahat2
  #then define sigam2 bias and have that as only output
  biasBeta <- (trueBeta- b.ols) %>%
    matrix(ncol = 2) %>%
    data.frame()
  # Set names
  names(biasBeta) <- c("interceptBias", "regressorBias")
  # Return the bias
  return(biasBeta)
}

# Run OLSBiasGenerator simulationSize times with given parameters
simulation.dt <- lapply(
  X = 1:simulationSize,
  FUN = function(i) OLSBiasGenerator(sampleSize, trueBeta)) %>%
  # Bind the rows together to output a nice data.frame
  bind_rows()

# Return simulation.dt
return(simulation.dt)
#simulation dt is a Simulationsize by number of true parametres matrix
}
```