HW1

2023-10-03

```r
# Load packages
pacman::p_load(magrittr, here, tidyverse, janitor, pracma, rmatio)

# Load data
air_df = read_csv(here("Data", "airline.txt"))
```

```
## Rows: 20412 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (4): ARR_DELAY, DAY_OF_WEEK, DEP_DELAY, DISTANCE
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
air_df %<>% clean_names()
```

Question 1

```r
# Delay = b0 + b1distance + b2departure delay + b3-8 day of week fixed effects

# Generate dummy variable for day fixed effects
air_df %<>% mutate(monday = if_else(day_of_week == 1, 1,0),
                 tuesday = if_else(day_of_week == 2, 1,0),
                 wednesday = if_else(day_of_week == 3, 1,0),
                 thursday = if_else(day_of_week == 4, 1,0),
                 friday = if_else(day_of_week == 5, 1,0),
                 saturday = if_else(day_of_week == 6, 1,0),
                 sunday = if_else(day_of_week == 17, 1,0)) %>%
  select(-day_of_week)

# First we'll make a matrix of the x values for ease of use
dependent_vars = air_df |> select(-c(arr_delay, sunday)) |>  # We have to drop sunday because of multic
  mutate(constant = 1) |>
  select(constant, everything()) |> # Move the constant column to the start to match the output of lm
  as.matrix()

# Function that calculates sum of squared errors
sum_squares = function(beta){
  squared_error = (air_df$arr_delay - dependent_vars %*% beta)^2
  sum_squares = sum(squared_error)
}
```

```r
# We'll use the fminsearch from the pracma package which is equivilant to its matlab version to find th
ols = fminsearch(sum_squares, rep(0,9))

# Print coefficients
names(ols$xmin) = colnames(dependent_vars)
ols$xmin
```

```
##     constant    dep_delay     distance       monday      tuesday    wednesday
## -1.267098776  1.016565016 -0.003133055  0.843533706 -0.074605986  1.012883212
##     thursday       friday     saturday
##  0.509084412 -0.872909482 -0.697221162
```

```r
# Calculate coefficients using matrix algebra
solve(t(dependent_vars)%*%dependent_vars)%*%t(dependent_vars)%*%air_df$arr_delay
```

```
##                   [,1]
## constant  -1.267055552
## dep_delay  1.016565637
## distance  -0.003133065
## monday     0.843525547
## tuesday   -0.074683873
## wednesday  1.012869061
## thursday   0.509066346
## friday    -0.872985813
## saturday  -0.697300656
```

```r
# Exactly the same result
```

Question 2

```r
# Generate dummy variable for a flight being over 15 minutes late
air_df %<>% mutate(late = if_else(arr_delay > 15, 1, 0))

# Get dependent variables for this model
dependent_vars = air_df |> select(c(distance, dep_delay)) |>
  mutate(constant = 1L) |>
  select(constant, everything()) |>
  as.matrix()

# Function to calculate log liklihood
probability = function(beta){
  prob = as.matrix(air_df$late) * log(1/(1+exp(-dependent_vars%*%beta))) + (1 - as.matrix(air_df$late))
  sum_prob = sum(prob)
  return(sum_prob)
}

# Search over betas. We're using a new function here because fminsearch didn't work here. Fminsearch on
logit = optim(c(1,0,0), probability, control = list(fnscale = -1))

# Print coefficients
```

```r
names(logit$par) = colnames(dependent_vars)
logit$par
```

```
##      constant      distance     dep_delay
## -1.740651e+00 -7.059086e-05  4.049501e-02
```

```r
# Compare to logit regression
logit_real = glm(late ~ distance + dep_delay, data = air_df,family = binomial(link = "logit"))
```

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

```r
# Its fairly close
logit_real$coefficients
```

```
##   (Intercept)      distance     dep_delay
## -2.6130481999 -0.0001367126  0.1294956106
```

Question 3

```r
# Load matlab file
iv_df = read.mat(here("Data", "IV.mat"))

# Extract x,y,z
X = iv_df[[1]]
Y = iv_df[[2]]
Z = iv_df[[3]]

# Define initial weight matrix
W = diag(4)

# Define g function
g = function(beta){
  1/length(Z) * t(Z)%*%(Y - X%*%beta)
}

# Define objective function
Q = function(beta, W){
  t(g(beta)) %*% W %*% g(beta)
}

# Minimize objective function
first_stage = fminsearch(Q, rep(0,3), W = W, method = "Hooke-Jeeves")

# Print coefficients
names(first_stage$xmin) = c("B1", "B2", "B3")
first_stage$xmin
```

```
##      B1      B2      B3
## 1.91869 1.10360 3.65262
```

```r
# Calculate residuals
beta_hat = first_stage$xmin
e = Y - X%*%beta_hat

# Calculate variance
G_hat = t(Z) %*% X
omega = sapply(1:4, function(i){
  (t(Z)%*%Z)[i,]*(e^2)[i,]
  }
)

variance_first = solve(t(G_hat)%*%W%*%G_hat)%*%t(G_hat)%*%W%*%omega%*%W%*%G_hat%*%solve(t(G_hat)%*%W%*%G

# Calculate SE
se_first = variance_first |> diag() |> sqrt()
names(se_first) = c("SE Beta1", "SE Beta2", "SE Beta3")
se_first
```

```
##  SE Beta1  SE Beta2  SE Beta3
## 0.1654447 0.2841884 0.2414821
```

```r
# Second stage
# Update w
W_hat = solve(sapply(1:nrow(Z), function(i){(e^2)[i]*Z[i,]}) %*% Z)

# Minimize objective function again
second_stage = fminsearch(Q, beta_hat, W = W_hat, method = "Hooke-Jeeves")

# Print coefficients
names(second_stage$xmin) = c("B1", "B2", "B3")
second_stage$xmin
```

```
##       B1       B2       B3
## 1.921046 1.093994 3.661452
```

```r
# Calculate residuals
beta_hat_second = second_stage$xmin
e_second = Y - X%*%beta_hat

# Calculate variance
omega_second = sapply(1:4, function(i){
  (t(Z)%*%Z)[i,]*(e_second^2)[i,]
  }
)

variance_second = solve(t(G_hat)%*%W_hat%*%G_hat)%*%t(G_hat)%*%W_hat%*%omega_second%*%W_hat%*%G_hat%*%so

# Calculate SE
se_second = variance_second |> diag() |> sqrt()
names(se_second) = c("SE Beta1", "SE Beta2", "SE Beta3")
se_second
```

```
##  SE Beta1  SE Beta2  SE Beta3
## 0.1654953 0.2981169 0.2222664
```