Paige Lee, P2 code

Original P2 assignment document: https://gsd-ses-5394-sp2025.github.io/examples/P2/P2.html (https://gsd-ses-5394-sp2025.github.io/examples/P2/P2.html)

1. Setup

Loading libraries and helper functions

```r
# Loading libraries
library(tidyverse)
library(here)
library(mlogit)
library(knitr)
library(caret)
library(dplyr)
library(dfidx)
library(MASS)

# Loading mlogit helper functions
source(here::here("P2_vehicle_availability/mlogit_helpers.R"))
```

Loading data

```r
# Loading household-level data from the 2017 National Household Travel Survey (NHTS)
hh_data <- here("P2_vehicle_availability", "data", "hhpub.csv") |> read_csv(show_col_types = FALSE)

# Loading person-level data
person_data <- here("P2_vehicle_availability", "data", "perpub.csv") |> read_csv(show_col_types = FALSE)
```

2. Feature selection and engineering

Selecting variables

```r
# Select desired variables from the household data
hh_data <- hh_data |> dplyr::select(WRKCOUNT,DRVRCNT, HHVEHCNT, HHSIZE, NUMADLT, HHFAMINC, HBPPOPDN, HOUSEID)

# Select desired variables from the person data
person_data <- person_data |> dplyr::select(HOUSEID, R_AGE, WORKER, DRIVER)
```

Mutating and constructing variables

Outcome variable

The categorical vehicle availability outcome variable has the following three categories:

- Zero vehicles
- Insufficient vehicles (fewer vehicles than drivers)
- Sufficient vehicles (at least as many vehicles as drivers)

```r
# Categorical outcome variable
hh_data <- hh_data |>
  mutate(veh_avail = case_when(HHVEHCNT == 0 ~ "Zero",
                               DRVRCNT > HHVEHCNT ~ "Insuff.",
                               TRUE ~ "Suff."))
```

Number of children

Number of children = number of people - number of adults in each household

```r
hh_data <- hh_data |>
  mutate(n_child = HHSIZE - NUMADLT)
```

Number of seniors

Using the person-level data, we can select those who are older than 64, group by household, and join that data with the household data

```
n_seniors <- person_data |>
  mutate(is_senior = R_AGE > 64) |>
  group_by(HOUSEID) |>
  summarise(n_seniors = sum(is_senior))

hh_data <- hh_data |>
  left_join(n_seniors)
```

Presence of a 3rd driver

Binary variable for whether or not each household has more than two drivers

```
hh_data <- hh_data |>
  mutate(three_drivers = DRVRCNT > 2)
```

Number of drivers beyond two

For households with more than two drivers, how many additional drivers do they have?

```
hh_data <- hh_data |>
  mutate(n_extra_drivers = ifelse(three_drivers, DRVRCNT - 2, 0))
```

Income

The low-income designation depends on both income and household size. Any household with income greater than $125,000, regardless of size, are designated as high income.

```
hh_data <- hh_data |>
  mutate(HHFAMINC = as.numeric(HHFAMINC)) |>
  filter(HHFAMINC > 0) |>
  mutate(income = case_when(HHFAMINC < 4 ~ "low",
                             HHFAMINC < 5 & HHSIZE > 1 ~ "low",
                             HHFAMINC < 6 & HHSIZE > 3 ~ "low",
                             HHFAMINC < 7 & HHSIZE > 5 ~ "low",
                             HHFAMINC < 8 & HHSIZE > 7 ~ "low",
                             HHFAMINC > 8 ~ "high",
                            TRUE ~ "medium")) |>
    mutate(income = factor(income, levels = c("medium", "low", "high")))
```

Non-worker driver

Binary variable for whether there is anyone in a given household who is a driver but not a worker

```
non_work_driver <- person_data |>
  mutate(non_work_driver = WORKER == "02" & DRIVER == "01") |>
  group_by(HOUSEID) |>
  summarise(non_work_driver = max(non_work_driver))

hh_data <- hh_data |>
  left_join(non_work_driver)
```

Density

The density of the household's census block group can be used to classify a household's neighborhoods as high, medium, or low density

```
hh_data <- hh_data |>
  filter(HBPPOPDN > 0) |>
  mutate(density = case_when(HBPPOPDN < 7000 ~ "Low",
                             HBPPOPDN < 10000 ~ "High",
                             TRUE ~ "Medium"))
```

Dropping variables we won't be using

```
hh_data <- hh_data |> dplyr::select(HOUSEID, veh_avail, WRKCOUNT, n_child, n_seniors, n_extra_drivers, three_driv
ers, non_work_driver, income, density)
```

Splitting data into training and testing

We're splitting the data into 50% training to use to train the model and 50% testing to test the model on new, unseen data

```
set.seed(1)

# Take a random sample of 50% of the IDs in the data
hh_data_train_ids <- sample(hh_data$HOUSEID,
                        size = ceiling(nrow(hh_data)/2))

# Assign these IDs to constitute the training set
hh_data_train <- hh_data |>
  filter(HOUSEID %in% hh_data_train_ids)

# Assign the remaining unused IDs to constitute the testing set
hh_data_test <- hh_data |>
  filter(!(HOUSEID %in% hh_data_train_ids))
```

Creating dfidx data

The mlogit package for multinomial logistic regression requires the data to be in the dfidx format

```
# Create the dfidx datasets
veh_dfidx_train <- fn_make_dfidx(hh_data_train,
                                "HOUSEID",
                                "veh_avail")

veh_dfidx_test <- fn_make_dfidx(hh_data_test,
                                "HOUSEID",
                                "veh_avail")
```

```
# Convert the appropriate categorical variables to factors
veh_dfidx_train$income <- factor(veh_dfidx_train$income)
veh_dfidx_test$income <- factor(veh_dfidx_test$income)

veh_dfidx_train$density <- factor(veh_dfidx_train$density)
veh_dfidx_test$density <- factor(veh_dfidx_test$density)

# Convert the appropriate binary variables to binary
veh_dfidx_train$three_drivers <- as.logical(veh_dfidx_train$three_drivers)
veh_dfidx_test$three_drivers <- as.logical(veh_dfidx_test$three_drivers)

veh_dfidx_train$non_work_driver <- as.logical(veh_dfidx_train$non_work_driver)
veh_dfidx_test$non_work_driver <- as.logical(veh_dfidx_test$non_work_driver)
```

3. Modeling

We will fit a multinomial logistic regression model

```
model_veh <- mlogit(choice ~ 0 | WRKCOUNT + n_child + n_seniors + n_extra_drivers + three_drivers + non_work_driv
er + income + density | 0, veh_dfidx_train, reflevel = "Suff.")

summary(model_veh)
```

```
##
## Call:
## mlogit(formula = choice ~ 0 | WRKCOUNT + n_child + n_seniors +
##     n_extra_drivers + three_drivers + non_work_driver + income +
##     density | 0, data = veh_dfidx_train, reflevel = "Suff.",
##     method = "nr")
##
## Frequencies of alternatives:choice
##    Suff.  Insuff.     Zero
## 0.883323 0.068361 0.048316
##
## nr method
## 9 iterations, 0h:0m:4s
## g'(-H)^-1g = 0.000126
## successive function values within tolerance limits
##
## Coefficients :
##                            Estimate Std. Error  z-value  Pr(>|z|)
## (Intercept):Insuff.       -4.287954   0.072690 -58.9899 < 2.2e-16 ***
## (Intercept):Zero           0.179721   0.088389   2.0333 0.0420221 *
## WRKCOUNT:Insuff.           0.423791   0.031699  13.3694 < 2.2e-16 ***
## WRKCOUNT:Zero             -3.162741   0.067690 -46.7238 < 2.2e-16 ***
## n_child:Insuff.            0.211287   0.018114  11.6642 < 2.2e-16 ***
## n_child:Zero              -0.155181   0.042966  -3.6117 0.0003042 ***
## n_seniors:Insuff.          0.340944   0.024345  14.0046 < 2.2e-16 ***
## n_seniors:Zero            -0.513402   0.047115 -10.8967 < 2.2e-16 ***
## n_extra_drivers:Insuff.    0.241909   0.059482   4.0669 4.764e-05 ***
## n_extra_drivers:Zero       0.462414   0.441284   1.0479 0.2946923
## three_driversTRUE:Insuff.  0.746038   0.084065   8.8745 < 2.2e-16 ***
## three_driversTRUE:Zero     0.433414   0.596741   0.7263 0.4676533
## non_work_driverTRUE:Insuff.  1.234070   0.055258  22.3329 < 2.2e-16 ***
## non_work_driverTRUE:Zero    -4.185006   0.072671 -57.5886 < 2.2e-16 ***
## incomelow:Insuff.          0.574217   0.037487  15.3176 < 2.2e-16 ***
## incomelow:Zero             1.949541   0.057127  34.1266 < 2.2e-16 ***
## incomehigh:Insuff.        -0.351304   0.050216  -6.9958 2.637e-12 ***
## incomehigh:Zero            0.130857   0.117096   1.1175 0.2637735
## densityLow:Insuff.        -0.310028   0.039573  -7.8344 4.663e-15 ***
## densityLow:Zero           -0.477784   0.059697  -8.0035 1.110e-15 ***
## densityMedium:Insuff.      0.834553   0.059858  13.9423 < 2.2e-16 ***
## densityMedium:Zero         1.549666   0.070877  21.8643 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log-Likelihood: -19986
## McFadden R^2:  0.27352
## Likelihood ratio test : chisq = 15049 (p.value = < 2.22e-16)
```

The regression coefficients represent the "utility of an alternative." For example, if the coefficient `n_child:Insuff.` is 0.2 and the coefficient `n_child:Zero` is -0.13, that means relative to having sufficient vehicles (the intercept), each additional child in a household increases the utility of having insufficient vehicles (positive coefficient) and decreases the utility of having zero vehicles (negative coefficient).

4. Making predictions

Making predictions using the test set (new/unseen). The output contains the head (first five rows) of the predictions, showing the predicted probabilities that each household has sufficient, insufficient, or zero vehicles (the response variable).

```
predicts_test <- predict(model_veh, veh_dfidx_test) |>
  as.data.frame() |>
  rownames_to_column("HOUSEID") |>
  mutate(HOUSEID = as.numeric(HOUSEID)) |>
  left_join(hh_data_test)

head(predicts_test) |>
  kable()
```

HOUSEID	Suff.	Insuff.	Zero	veh_avail	WRKCOUNT	n_child	n_seniors	n_extra_drivers	three_drivers	non_work_driver	income
30000008	0.9757649	0.0229386	0.0012965	Suff.	2	0	0	0	FALSE	0	medium
30000012	0.7657739	0.0260488	0.2081772	Suff.	1	0	0	0	FALSE	0	high
30000019	0.8696155	0.1056618	0.0247227	Suff.	0	0	2	0	FALSE	1	low
30000029	0.9324260	0.0638008	0.0037732	Suff.	0	0	2	0	FALSE	1	medium

HOUSEID	Suff.	Insuff.	Zero	veh_avail	WRKCOUNT	n_child	n_seniors	n_extra_drivers	three_drivers	non_work_driver	income
30000039	0.9313035	0.0685147	0.0001817	Suff.	1	0	2	0	FALSE	1	high
30000082	0.9644641	0.0345963	0.0009396	Suff.	2	2	0	0	FALSE	0	medium

5. Evaluating the model

```
# Designate the alternative with the highest predictive probability as the most likely choice
predicts_test <- predicts_test |>
  mutate(most_likely = case_when((Suff. > Insuff.) & (Suff. > Zero) ~ "Suff.",
                                 (Zero > Insuff.) & (Zero > Suff.) ~ "Zero",
                                 TRUE ~ "Insuff."))

# Convert the most_likely and veh_avail variables to factors
predicts_test <- predicts_test |>
  mutate(most_likely = factor(most_likely,
                              levels = c("Suff.", "Insuff.", "Zero"))) |>
  mutate(veh_avail = factor(veh_avail,
                            levels = c("Suff.", "Insuff.", "Zero"))) |>
  mutate(correct = veh_avail == most_likely)

# Calculate a confusion matrix to generate accuracy and reliability statistics
confusionMatrix(data = predicts_test$most_likely,
                reference = predicts_test$veh_avail)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Suff. Insuff.  Zero
##    Suff.   54838    4243  1417
##    Insuff.   173     139     0
##    Zero      330       0  1468
##
## Overall Statistics
##
##                Accuracy : 0.9016
##                  95% CI : (0.8992, 0.9039)
##     No Information Rate : 0.8839
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3173
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Suff. Class: Insuff. Class: Zero
## Sensitivity                0.9909       0.031721     0.50884
## Specificity                0.2211       0.997029     0.99447
## Pos Pred Value             0.9064       0.445513     0.81646
## Neg Pred Value             0.7616       0.931890     0.97670
## Prevalence                 0.8839       0.069991     0.04608
## Detection Rate             0.8759       0.002220     0.02345
## Detection Prevalence       0.9663       0.004983     0.02872
## Balanced Accuracy          0.6060       0.514375     0.75166
```

Definitions

- **No information rate:** the accuracy you would achieve if you had no model and just classified every household as the most common value among Suff, Insuff, and Zero.
- **Sensitivity:** the percent of true positives that were correctly identified → true positive rate
  - A highly sensitive test will have few false negatives
  - Ex. high sensitivity → model misses fewer disease cases
- **Specificity:** the percent of true negatives that were correctly identified → true negative rate
  - A highly specific test will have few false positives
  - Ex. high specificity → model correctly identifies more people who don't have the disease
- **Positive predictive value:** the probability that a positive prediction is correct → measures accuracy
  - $PPV = \frac{TP}{TP+FP}$
- **Negative predictive value:** the probability that a negative prediction is correct → measures accuracy
  - $NPV = \frac{TN}{TN+FN}$
- **Prevalence:** the percent of observations in each category

- **Detection rate:** the proportion of true positive cases that are correctly identified by the test
  - true positives / total positives
  - Represents the sensitivity of a test → how well can the test identify TPs when they're present
- **Detection prevalence:** the proportion of all predicted positive cases (TPs and FPs) within a population
  - predicted positives / total predictions
  - Represents the proportion of cases flagged as positive by the test → includes TPs and FPs
- **Balanced accuracy:** the average of a model's sensitivity and specificity - (sensitivity + specificity) / 2

6. Building a new model

Can we estimate a vehicle availability model that performs better than this one?

Data dictionary: https://nhts.ornl.gov/tables09/CodebookBrowser.aspx (https://nhts.ornl.gov/tables09/CodebookBrowser.aspx)

Description of the steps I took to get to my final model

- Variables used in the current model: WRKCOUNT, n_child, n_seniors, n_extra_drivers, three_drivers, non_work_driver, income, density
- Model 1
  - Variables I added: TRAVDAY (travel day of the week), URBAN (household's urban area classification), HH_RACE (race of household respondent), WEBUSE17 (frequency of internet use)
  - The model's performance metrics didn't improve much, especially considering many more predictors were added due to the additional categorical variables with multiple levels.
- Model 2
  - Drop WEBUSE17 since none of the predictors were significant
  - Transform TRAVDAY to just have two levels: weekday or weekend
    - Sunday is coded as 01, and Saturday is coded as 07
- Model 3
  - Transform the HH_RACE variable into "white" or "non-white" since whites make up the vast majority of observations. Drop "refused" and "don't know" observations (missing).
- Model 4
  - Transform URBAN into "urban" for 01, 02, 03 and "non-urban" for 04
  - 01 is urban area, 02 is urban cluster, and 03 is surrounding urban
  - 04 is not urban
  - Goal is to reduce the number of predictors especially since only urban04 is significant, but urban 01-03 are not
- Model 5
  - Remove TRAVDAY_transformed from the model → model worsens a bit → add TRAVDAY_transformed back
- Model 6
  - Add EDUC to the model, dropping missing values
  - EDUC is from the person-level data, so we must join it with the household level data
  - Select the highest education level in each household and assign that as the household's education level since EDUC is an ordinal variable
- Model 7
  - Adding education was good, but we want to reduce the number of predictors since not all were significant
  - After taking the maximum education level of a household, convert it to low or high education (categorical)
  - Group 01, 02, and 03 together → less than high schoo, high school, and some college or associates degree → low_educ
  - Group 04 and 04 together → bachelor's degree, graduate or professional degree → high_educ
- Model 8
  - Add CNTTDHH to the model → count of household trips on travel day (numeric)
- Model 9
  - Add the medical condition variables → CONDNIGH, CONDPUB, CONDRIDE, CONDRIVE, CONDSPEC, CONTAX, CONTRAV
  - These are person-level variables → merge by household, and take the lowest value (yes is coded as 01, no is coded as 02), and drop NAs
  - This addition lowered the accuracy and sensitivity, but it increased the specificity
- Model 10
  - Add bike, bus, car, rail, taxi, train, walk (all household variables)
  - Convert 1, 2, 3 to "regularly" and 4, 5 to "not_regularly"
  - Drop NAs (coded as negative values)
  - Removed RAIL_transformed since all values got converted to "regularly"
  - This addition increased the specificity even more
- Model 11
  - Add HOMEOWN and HH_HISP
  - Convert HH_HISP to binary and remove NAs
  - Drop NAs from HOMEOWN
- Model 12
  - Now that we've included many variables, it's time to use stepwise methods for variable selection

Performing variable selection, data cleaning, and feature engineering again

```r
# Loading household-level data from the 2017 National Household Travel Survey (NHTS)
hh_data <- here("P2_vehicle_availability", "data", "hhpub.csv") |> read_csv(show_col_types = FALSE)

# Loading person-level data
person_data <- here("P2_vehicle_availability", "data", "perpub.csv") |> read_csv(show_col_types = FALSE)

# Selecting desired variables from the NHTS data
hh_data <- hh_data |> dplyr::select(WRKCOUNT,DRVRCNT, HHVEHCNT, HHSIZE, NUMADLT, HHFAMINC, HBPPOPDN, HOUSEID, TRA
VDAY, URBAN, HH_RACE, CNTTDHH, BIKE, BUS, CAR, RAIL, TAXI, TRAIN, WALK, HOMEOWN, HH_HISP)

# Select desired variables from the person data
person_data <- person_data |> dplyr::select(HOUSEID, R_AGE, WORKER, DRIVER, EDUC, CONDNIGH, CONDPUB, CONDRIDE, CO
NDRIVE, CONDSPEC, CONDTAX, CONDTRAV)

# Categorical outcome variable
hh_data <- hh_data |>
  mutate(veh_avail = case_when(HHVEHCNT == 0 ~ "Zero",
                               DRVRCNT > HHVEHCNT ~ "Insuff.",
                               TRUE ~ "Suff."))

# Number of children
hh_data <- hh_data |>
  mutate(n_child = HHSIZE - NUMADLT)

# Number of seniors
n_seniors <- person_data |>
  mutate(is_senior = R_AGE > 64) |>
  group_by(HOUSEID) |>
  summarise(n_seniors = sum(is_senior))

hh_data <- hh_data |>
  left_join(n_seniors)

# Presence of a 3rd driver
hh_data <- hh_data |>
  mutate(three_drivers = DRVRCNT > 2)

# Number of drivers beyond two
hh_data <- hh_data |>
  mutate(n_extra_drivers = ifelse(three_drivers, DRVRCNT - 2, 0))

# Income
# NOTE: missing values (coded as negative) are dropped!
hh_data <- hh_data |>
  mutate(HHFAMINC = as.numeric(HHFAMINC)) |>
  filter(HHFAMINC > 0) |>
  mutate(income = case_when(HHFAMINC < 4 ~ "low",
                             HHFAMINC < 5 & HHSIZE > 1 ~ "low",
                             HHFAMINC < 6 & HHSIZE > 3 ~ "low",
                             HHFAMINC < 7 & HHSIZE > 5 ~ "low",
                             HHFAMINC < 8 & HHSIZE > 7 ~ "low",
                             HHFAMINC > 8 ~ "high",
                            TRUE ~ "medium")) |>
    mutate(income = factor(income, levels = c("medium", "low", "high")))

# Non-worker driver
non_work_driver <- person_data |>
  mutate(non_work_driver = WORKER == "02" & DRIVER == "01") |>
  group_by(HOUSEID) |>
  summarise(non_work_driver = max(non_work_driver))

hh_data <- hh_data |>
  left_join(non_work_driver)

# Density
hh_data <- hh_data |>
  filter(HBPPOPDN > 0) |>
  mutate(density = case_when(HBPPOPDN < 7000 ~ "Low",
                             HBPPOPDN < 10000 ~ "High",
                             TRUE ~ "Medium"))

# Travel day transformed
hh_data$TRAVDAY_transformed <- ifelse(as.numeric(hh_data$TRAVDAY) %in% c(1, 7), "weekend", "weekday")
```

```
# Race transformed
# Drop the missing values (negative values)
hh_data <- hh_data |>
  mutate(HH_RACE = as.numeric(HH_RACE)) |>
  filter(HH_RACE > 0) |>
  mutate(HH_RACE_transformed = case_when(
    HH_RACE == 1 ~ "white",
    HH_RACE > 1 ~ "non-white"
  ))

# Urban transformed
hh_data <- hh_data |>
  mutate(URBAN = as.numeric(URBAN)) |>
  mutate(URBAN_transformed = case_when(
    URBAN < 4 ~ "urban",
    URBAN == 4 ~ "non-urban"
  ))

# Educ transformed, filtered out NAs
# First take the max education of each household, then assign low or high educ labels
EDUC_transformed <- person_data |>
  mutate(EDUC = as.numeric(EDUC)) |>
  filter(EDUC > 0) |>
  group_by(HOUSEID) |>
  summarise(EDUC_transformed = case_when(
    max(EDUC, na.rm = TRUE) <= 3 ~ "low_educ",
    max(EDUC, na.rm = TRUE) > 3 ~ "high_educ"
  ), .groups = "drop")

hh_data <- hh_data |>
  left_join(EDUC_transformed, by = "HOUSEID")

# Convert CNTTDHH to numeric
hh_data$CNTTDHH <- as.numeric(hh_data$CNTTDHH)

# CONDNIGH, CONDPUB, CONDRIDE, CONDRIVE, CONDSPEC, CONDTAX, CONDTRAV
# Join person-level data with household-level data and drop NAs
CONDNIGH <- person_data |>
  mutate(CONDNIGH = as.numeric(CONDNIGH)) |>
  filter(CONDNIGH > 0) |>
  group_by(HOUSEID) |>
  summarise(CONDNIGH = as.logical(if_else(max(CONDNIGH, na.rm = TRUE) == 2, 0, 1)), .groups = "drop")
hh_data <- hh_data |>
  left_join(CONDNIGH, by = "HOUSEID")

CONDPUB <- person_data |>
  mutate(CONDPUB = as.numeric(CONDPUB)) |>
  filter(CONDPUB > 0) |>
  group_by(HOUSEID) |>
  summarise(CONDPUB = as.logical(if_else(max(CONDPUB, na.rm = TRUE) == 2, 0, 1)), .groups = "drop")
hh_data <- hh_data |>
  left_join(CONDPUB, by = "HOUSEID")

CONDRIDE <- person_data |>
  mutate(CONDRIDE = as.numeric(CONDRIDE)) |>
  filter(CONDRIDE > 0) |>
  group_by(HOUSEID) |>
  summarise(CONDRIDE = as.logical(if_else(max(CONDRIDE, na.rm = TRUE) == 2, 0, 1)), .groups = "drop")
hh_data <- hh_data |>
  left_join(CONDRIDE, by = "HOUSEID")

CONDRIVE <- person_data |>
  mutate(CONDRIVE = as.numeric(CONDRIVE)) |>
  filter(CONDRIVE > 0) |>
  group_by(HOUSEID) |>
  summarise(CONDRIVE = as.logical(if_else(max(CONDRIVE, na.rm = TRUE) == 2, 0, 1)), .groups = "drop")
hh_data <- hh_data |>
  left_join(CONDRIVE, by = "HOUSEID")

CONDSPEC <- person_data |>
  mutate(CONDSPEC = as.numeric(CONDSPEC)) |>
  filter(CONDSPEC > 0) |>
  group_by(HOUSEID) |>
  summarise(CONDSPEC = as.logical(if_else(max(CONDSPEC, na.rm = TRUE) == 2, 0, 1)), .groups = "drop")
hh_data <- hh_data |>
```

```
  left_join(CONDSPEC, by = "HOUSEID")

CONDTAX <- person_data |>
  mutate(CONDTAX = as.numeric(CONDTAX)) |>
  filter(CONDTAX > 0) |>
  group_by(HOUSEID) |>
  summarise(CONDTAX = as.logical(if_else(max(CONDTAX, na.rm = TRUE) == 2, 0, 1)), .groups = "drop")
hh_data <- hh_data |>
  left_join(CONDTAX, by = "HOUSEID")

CONDTRAV <- person_data |>
  mutate(CONDTRAV = as.numeric(CONDTRAV)) |>
  filter(CONDTRAV > 0) |>
  group_by(HOUSEID) |>
  summarise(CONDTRAV = as.logical(if_else(max(CONDTRAV, na.rm = TRUE) == 2, 0, 1)), .groups = "drop")
hh_data <- hh_data |>
  left_join(CONDTRAV, by = "HOUSEID")

# Transforming BIKE, BUS, CAR, RAIL, TAXI, TRAIN, WALK
hh_data <- hh_data |>
  mutate(BIKE = as.numeric(BIKE)) |>
  mutate(BIKE_transformed = case_when(
    BIKE < 4 ~ "regularly",
    BIKE >=4 ~ "non_regularly"
  ) |> as.factor())

hh_data <- hh_data |>
  mutate(BUS = as.numeric(BUS)) |>
  mutate(BUS_transformed = case_when(
    BUS < 4 ~ "regularly",
    BUS >=4 ~ "non_regularly"
  ) |> as.factor())

hh_data <- hh_data |>
  mutate(CAR = as.numeric(CAR)) |>
  mutate(CAR_transformed = case_when(
    CAR < 4 ~ "regularly",
    CAR >=4 ~ "non_regularly"
  ) |> as.factor())

hh_data <- hh_data |>
  mutate(RAIL = as.numeric(RAIL)) |>
  mutate(RAIL_transformed = case_when(
    RAIL < 4 ~ "regularly",
    RAIL >=4 ~ "non_regularly"
  ) |> as.factor())

hh_data <- hh_data |>
  mutate(TAXI = as.numeric(TAXI)) |>
  mutate(TAXI_transformed = case_when(
    TAXI < 4 ~ "regularly",
    TAXI >=4 ~ "non_regularly"
  ) |> as.factor())

hh_data <- hh_data |>
  mutate(TRAIN = as.numeric(TRAIN)) |>
  mutate(TRAIN_transformed = case_when(
    TRAIN < 4 ~ "regularly",
    TRAIN >=4 ~ "non_regularly"
  ) |> as.factor())

hh_data <- hh_data |>
  mutate(WALK = as.numeric(WALK)) |>
  mutate(WALK_transformed = case_when(
    WALK < 4 ~ "regularly",
    WALK >=4 ~ "non_regularly"
  ) |> as.factor())

# HOMEOWN drop NAs
hh_data <- hh_data |>
  mutate(HOMEOWN = as.numeric(HOMEOWN)) |>
  filter(HOMEOWN > 0)

# HH_HISP drop NAs and convert to binary
hh_data <- hh_data |>
  mutate(HH_HISP = as.numeric(HH_HISP)) |>
  filter(HH_HISP > 0) |>
  mutate(HH_HISP = as.logical(if_else(HH_HISP == 2, 0, 1)))

# Selecting the variables we want
hh_data <- hh_data |> dplyr::select(HOUSEID, veh_avail, WRKCOUNT, n_child, n_seniors, n_extra_drivers, three_driv
ers, non_work_driver, income, density, TRAVDAY_transformed, URBAN_transformed, HH_RACE_transformed, EDUC_transfor
med, CNTTDHH, CONDNIGH, CONDPUB, CONDRIDE, CONDRIVE, CONDSPEC, CONDTAX, CONDTRAV, BIKE_transformed, BUS_transform
ed, CAR_transformed, RAIL_transformed, TAXI_transformed, TRAIN_transformed, WALK_transformed, HOMEOWN, HH_HISP)

# Convert categorical variables to factors
hh_data$veh_avail <- factor(hh_data$veh_avail)
hh_data$density <- factor(hh_data$density)
hh_data$income <- factor(hh_data$income)
hh_data$TRAVDAY_transformed <- factor(hh_data$TRAVDAY_transformed)
hh_data$URBAN_transformed <- factor(hh_data$URBAN_transformed)
hh_data$HH_RACE_transformed <- factor(hh_data$HH_RACE_transformed)
hh_data$EDUC_transformed <- factor(hh_data$EDUC_transformed)

# Convert the appropriate binary variables to binary
hh_data$three_drivers <- as.logical(hh_data$three_drivers)
hh_data$non_work_driver <- as.logical(hh_data$non_work_driver)
```

Train test split and convert to dfidx format

```
set.seed(1)

# Take a random sample of 50% of the IDs in the data
hh_data_train_ids <- sample(hh_data$HOUSEID,
                        size = ceiling(nrow(hh_data)/2))

# Assign these IDs to constitute the training set
hh_data_train <- hh_data |>
  filter(HOUSEID %in% hh_data_train_ids)

# Assign the remaining unused IDs to constitute the testing set
hh_data_test <- hh_data |>
  filter(!(HOUSEID %in% hh_data_train_ids))

# Create the dfidx datasets
veh_dfidx_train <- fn_make_dfidx(hh_data_train,
                                "HOUSEID",
                                "veh_avail")

veh_dfidx_test <- fn_make_dfidx(hh_data_test,
                                "HOUSEID",
                                "veh_avail")
```

Modeling

```
# Full model, eval = FALSE (before stepwise variable selection)
model_veh <- mlogit(choice ~ 0 | WRKCOUNT + n_child + n_seniors + n_extra_drivers + three_drivers + non_work_driv
er + income + density + TRAVDAY_transformed + URBAN_transformed + HH_RACE_transformed + EDUC_transformed + CNTTDH
H + CONDNIGH + CONDPUB + CONDRIDE + CONDRIVE + CONDSPEC + CONDTAX + CONDTRAV + BIKE_transformed + BUS_transformed
+ CAR_transformed + TAXI_transformed + TRAIN_transformed + WALK_transformed + HOMEOWN + HH_HISP | 0, veh_dfidx_tr
ain, reflevel = "Suff.")

summary(model_veh)
```

Stepwise variable selection

```r
# Set to eval = FALSE because I don't need to perform stepwise variable selection everytime; just once to find th
e best subset of predictors
# Define a stepwise variable selection function for mlogit since it's not compatible with the built in AIC functi
ons
# This function is from ChatGPT
stepwise_mlogit <- function(data, full_formula) {
  # Fit the full model
  current_model <- mlogit(full_formula, data, reflevel = "Suff.")
  best_aic <- AIC(current_model)

  # Extract predictor variable names
  predictors <- all.vars(update(full_formula, . ~ . - 0))[!all.vars(update(full_formula, . ~ . - 0)) %in% c("choi
ce")]

  total_vars <- length(predictors)  # Count total variables
  step <- 1  # Track step number

  for (var in predictors) {
    new_predictors <- setdiff(predictors, var)  # Remove one variable
    new_formula <- as.formula(paste("choice ~ 0 |", paste(new_predictors, collapse = " + "), "| 0"))

    message(paste("Step", step, "of", total_vars, "- Removing:", var))

    # Fit new model
    new_model <- mlogit(new_formula, data, reflevel = "Suff.")
    new_aic <- AIC(new_model)

    # Keep the model if it improves AIC
    if (new_aic < best_aic) {
      best_aic <- new_aic
      current_model <- new_model
      predictors <- new_predictors  # Update remaining variables
    }

    step <- step + 1  # Increment step count
  }

  message("Stepwise selection complete. Best AIC:", best_aic)
  return(current_model)
}

# Define the full model formula
full_formula <- choice ~ 0 | WRKCOUNT + n_child + n_seniors + n_extra_drivers + three_drivers + non_work_driver +
income + density + TRAVDAY_transformed + URBAN_transformed + HH_RACE_transformed + EDUC_transformed + CNTTDHH + C
ONDNIGH + CONDPUB + CONDRIDE + CONDRIVE + CONDSPEC + CONDTAX + CONDTRAV + BIKE_transformed + BUS_transformed + CA
R_transformed + TAXI_transformed + TRAIN_transformed + WALK_transformed | 0

# Run stepwise selection
best_model <- stepwise_mlogit(veh_dfidx_train, full_formula)

# View the final selected model
summary(best_model)
```

```r
# Final model after stepwise variable selection
model_veh <- mlogit(choice ~ 0 | WRKCOUNT + n_child + n_seniors + n_extra_drivers + three_drivers + non_work_driv
er + income + density + URBAN_transformed + HH_RACE_transformed + CNTTDHH + CONDNIGH + CONDPUB + CONDRIDE + CONDR
IVE + CONDSPEC + CONDTAX + CONDTRAV + BUS_transformed + CAR_transformed + TAXI_transformed + TRAIN_transformed +
WALK_transformed | 0, veh_dfidx_train, reflevel = "Suff.")

summary(model_veh)
```

```
##
## Call:
## mlogit(formula = choice ~ 0 | WRKCOUNT + n_child + n_seniors +
##     n_extra_drivers + three_drivers + non_work_driver + income +
##     density + URBAN_transformed + HH_RACE_transformed + CNTTDHH +
##     CONDNIGH + CONDPUB + CONDRIDE + CONDRIVE + CONDSPEC + CONDTAX +
##     CONDTRAV + BUS_transformed + CAR_transformed + TAXI_transformed +
##     TRAIN_transformed + WALK_transformed | 0, data = veh_dfidx_train,
##     reflevel = "Suff.", method = "nr")
##
## Frequencies of alternatives:choice
##    Suff.  Insuff.     Zero
## 0.801393 0.095314 0.103293
##
## nr method
## 9 iterations, 0h:0m:2s
## g'(-H)^-1g = 3.6E-06
## successive function values within tolerance limits
##
## Coefficients :
##                                      Estimate Std. Error  z-value  Pr(>|z|)
## (Intercept):Insuff.                -4.6941326  0.3552496 -13.2136 < 2.2e-16 ***
## (Intercept):Zero                    1.7336125  0.3352380   5.1713 2.325e-07 ***
## WRKCOUNT:Insuff.                    0.3123005  0.0601242   5.1943 2.055e-07 ***
## WRKCOUNT:Zero                      -2.8755573  0.1489194 -19.3095 < 2.2e-16 ***
## n_child:Insuff.                     0.2271301  0.0485547   4.6778 2.899e-06 ***
## n_child:Zero                       -0.1950307  0.1123843  -1.7354 0.0826715 .
## n_seniors:Insuff.                   0.4582923  0.0439432  10.4292 < 2.2e-16 ***
## n_seniors:Zero                     -0.6799440  0.0916950  -7.4153 1.215e-13 ***
## n_extra_drivers:Insuff.             0.2596335  0.1217158   2.1331 0.0329156 *
## n_extra_drivers:Zero                2.0561934  0.6527013   3.1503 0.0016311 **
## three_driversTRUE:Insuff.           0.9564769  0.1720882   5.5581 2.728e-08 ***
## three_driversTRUE:Zero             -1.7006101  1.0768996  -1.5792 0.1142965
## non_work_driverTRUE:Insuff.         1.2486129  0.1450438   8.6085 < 2.2e-16 ***
## non_work_driverTRUE:Zero           -3.5327716  0.1461093 -24.1790 < 2.2e-16 ***
## incomelow:Insuff.                   0.5325811  0.0659922   8.0704 6.661e-16 ***
## incomelow:Zero                      1.3263354  0.1358369   9.7642 < 2.2e-16 ***
## incomehigh:Insuff.                 -0.5000855  0.1385884  -3.6084 0.0003081 ***
## incomehigh:Zero                    -0.8103450  0.6485872  -1.2494 0.2115187
## densityLow:Insuff.                 -0.0277451  0.0773471  -0.3587 0.7198128
## densityLow:Zero                    -0.2121704  0.1339624  -1.5838 0.1132381
## densityMedium:Insuff.               0.0417462  0.1388995   0.3005 0.7637581
## densityMedium:Zero                  0.5286740  0.1868949   2.8287 0.0046734 **
## URBAN_transformedurban:Insuff.      0.4683106  0.0809519   5.7850 7.249e-09 ***
## URBAN_transformedurban:Zero         0.6275634  0.1669141   3.7598 0.0001701 ***
## HH_RACE_transformedwhite:Insuff.   -0.1002347  0.0865234  -1.1585 0.2466725
## HH_RACE_transformedwhite:Zero      -0.5976660  0.1250256  -4.7803 1.750e-06 ***
## CNTTDHH:Insuff.                     0.0123327  0.0060052   2.0537 0.0400096 *
## CNTTDHH:Zero                       -0.0168737  0.0144216  -1.1700 0.2419892
## CONDNIGHTRUE:Insuff.               -0.1117210  0.0729631  -1.5312 0.1257204
## CONDNIGHTRUE:Zero                  -1.1429978  0.1554510  -7.3528 1.941e-13 ***
## CONDPUBTRUE:Insuff.                 0.3611441  0.1580223   2.2854 0.0222894 *
## CONDPUBTRUE:Zero                    0.8537292  0.1633906   5.2251 1.741e-07 ***
## CONDRIDETRUE:Insuff.               -0.2487314  0.0759480  -3.2750 0.0010565 **
## CONDRIDETRUE:Zero                   0.9441004  0.1179023   8.0075 1.110e-15 ***
## CONDRIVETRUE:Insuff.               -0.4177726  0.1112884  -3.7540 0.0001741 ***
## CONDRIVETRUE:Zero                   0.5162857  0.1312441   3.9338 8.362e-05 ***
## CONDSPECTRUE:Insuff.               -0.3642978  0.2160525  -1.6862 0.0917661 .
## CONDSPECTRUE:Zero                   0.4967109  0.1639330   3.0300 0.0024458 **
## CONDTAXTRUE:Insuff.                -0.1642017  0.3394078  -0.4838 0.6285357
## CONDTAXTRUE:Zero                    0.8206421  0.2603653   3.1519 0.0016222 **
## CONDTRAVTRUE:Insuff.                0.1436028  0.0663678   2.1637 0.0304842 *
## CONDTRAVTRUE:Zero                  -0.2944897  0.1212208  -2.4294 0.0151253 *
## BUS_transformedregularly:Insuff.    0.5215804  0.1553875   3.3566 0.0007889 ***
## BUS_transformedregularly:Zero       1.3853517  0.1756467   7.8872 3.109e-15 ***
## CAR_transformedregularly:Insuff.   -0.0320920  0.2867695  -0.1119 0.9108958
## CAR_transformedregularly:Zero      -3.0181448  0.1581379 -19.0855 < 2.2e-16 ***
## TAXI_transformedregularly:Insuff.  -0.4880370  0.1632592  -2.9893 0.0027958 **
## TAXI_transformedregularly:Zero      0.6936025  0.1702400   4.0743 4.616e-05 ***
## TRAIN_transformedregularly:Insuff. -0.0173478  0.1805795  -0.0961 0.9234670
## TRAIN_transformedregularly:Zero    -0.4430133  0.1987085  -2.2295 0.0257830 *
## WALK_transformedregularly:Insuff.   0.1069320  0.0675564   1.5829 0.1134545
## WALK_transformedregularly:Zero      0.4909410  0.1439793   3.4098 0.0006501 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Log-Likelihood: -5006.2
## McFadden R^2:  0.42901
## Likelihood ratio test : chisq = 7522.6 (p.value = < 2.22e-16)
```

Model evaluation

```
predicts_test <- predict(model_veh, veh_dfidx_test) |>
  as.data.frame() |>
  rownames_to_column("HOUSEID") |>
  mutate(HOUSEID = as.numeric(HOUSEID)) |>
  left_join(hh_data_test)

# Designate the alternative with the highest predictive probability as the most likely choice
predicts_test <- predicts_test |>
  mutate(most_likely = case_when((Suff. > Insuff.) & (Suff. > Zero) ~ "Suff.",
                                 (Zero > Insuff.) & (Zero > Suff.) ~ "Zero",
                                 TRUE ~ "Insuff."))

# Convert the most_likely and veh_avail variables to factors
predicts_test <- predicts_test |>
  mutate(most_likely = factor(most_likely,
                              levels = c("Suff.", "Insuff.", "Zero"))) |>
  mutate(veh_avail = factor(veh_avail,
                            levels = c("Suff.", "Insuff.", "Zero"))) |>
  mutate(correct = veh_avail == most_likely)

# Calculate a confusion matrix to generate accuracy and reliability statistics
confusionMatrix(data = predicts_test$most_likely,
                reference = predicts_test$veh_avail)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Suff. Insuff.  Zero
##    Suff.   10861    1200   283
##    Insuff.    96      60     1
##    Zero      166       1  1111
##
## Overall Statistics
##
##                Accuracy : 0.8732
##                  95% CI : (0.8675, 0.8787)
##     No Information Rate : 0.8072
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5241
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: Suff. Class: Insuff. Class: Zero
## Sensitivity                0.9764       0.047581     0.79642
## Specificity                0.4416       0.992251     0.98651
## Pos Pred Value             0.8799       0.382166     0.86933
## Neg Pred Value             0.8174       0.911834     0.97728
## Prevalence                 0.8072       0.091516     0.10124
## Detection Rate             0.7882       0.004354     0.08063
## Detection Prevalence       0.8959       0.011394     0.09275
## Balanced Accuracy          0.7090       0.519916     0.89147
```