Food Security Analysis with Lasso and Ridge Regression

2024-12-13

Clearing the Environment and Loading Libraries

```
# Removing all objects from the environment to ensure a clean workspace
# rm(list=ls())

# Loading necessary libraries
library(ggplot2)          # For creating various plots
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
library(RColorBrewer)     # For color palettes
library(tidyverse)        # For data manipulation and visualization
```

```
## Warning: package 'forcats' was built under R version 4.2.3
```

```
## Warning: package 'lubridate' was built under R version 4.2.3
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.0     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ lubridate 1.9.2     ✓ tibble    3.2.0
## ✓ purrr     1.0.1     ✓ tidyr     1.3.0
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
## ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conflicts t
o become errors
```

```
library(pROC)             # For ROC curve analysis
```

```
## Warning: package 'pROC' was built under R version 4.2.3
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(glmnet)            # For lasso and ridge regression models
```

```
## Warning: package 'glmnet' was built under R version 4.2.3
```

```
## Loading required package: Matrix
```

```
## Warning: package 'Matrix' was built under R version 4.2.3
```

```
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```
library(lubridate)         # For date/time manipulation
library(sf)                # For spatial data handling (shapefiles)
```

```
## Warning: package 'sf' was built under R version 4.2.3
```

```
## Linking to GEOS 3.9.3, GDAL 3.5.2, PROJ 8.2.1; sf_use_s2() is TRUE
```

```
library(dplyr)             # For data manipulation
library(tigris)            # For US Census shapefiles
```

```
## Warning: package 'tigris' was built under R version 4.2.3
```

```
## To enable caching of data, set `options(tigris_use_cache = TRUE)`
## in your R script or .Rprofile.
```

```
library(ggthemes)   # Themes for ggplot2
```

```
## Warning: package 'ggthemes' was built under R version 4.2.3
```

```r
library(logistf)    # Logistic regression with Firth correction
```

```
## Warning: package 'logistf' was built under R version 4.2.3
```

```r
library(haven)      # Reading and writing SPSS, Stata, and SAS files
```

```
## Warning: package 'haven' was built under R version 4.2.3
```

```r
library(knitr)      # Dynamic report generation
library(readr)
```

Loading the CPS Data

```r
# Clear the environment
# Uncomment the following line only if you are testing the code
# rm(list = ls())


# Load the data
cps <- read.csv("C:/Users/sophi/OneDrive/Documents/STAT 172/STAT_172_Final_Project/data/cps_0000
6.csv")

# Check the distribution of the target variable
table(cps$FSFOODS)
```

```
##
##     1     2     3     4    96    97    99
## 11870  2905   474    90     2     8  4098
```

```
##### CLEAN CPS DATA #####

# Create derived variables
cps <- cps %>%
  mutate(
    SEX = SEX - 1,  # Convert SEX to a dummy variable (0 = Male, 1 = Female)
    CHILD = ifelse(AGE < 18, 1, 0),  # Identify children under 18
    ELDERLY = ifelse(AGE > 60, 1, 0),  # Elderly defined as 60+
    BLACK = ifelse(RACE == 200, 1, 0),  # Dummy variable for Black race
    HISPANIC = ifelse(HISPAN > 0, 1, 0),  # Dummy variable for Hispanic ethnicity
    EDUC = as.integer(EDUC %in% c(91, 92, 111, 123, 124, 125)),  # High school or higher educati
on
    EMP = as.integer(EMPSTAT %in% c(1, 10, 12)),  # Employment status
    MARRIED = as.integer(MARST %in% c(1, 2)),  # Married or partnered
    DIFF = ifelse(DIFFANY == 2, 1, 0),  # Difficulty variable
    COUNTY = as.factor(COUNTY)  # Convert COUNTY to a factor
  )

# Categorize family income into meaningful groups
cps <- cps %>%
  mutate(
    FAMINC_category = case_when(
      FAMINC == 100 ~ "Under $5,000",
      FAMINC == 210 ~ "$5,000 - $7,499",
      FAMINC == 300 ~ "$7,500 - $9,999",
      FAMINC == 430 ~ "$10,000 - $12,499",
      FAMINC == 470 ~ "$12,500 - $14,999",
      FAMINC == 500 ~ "$15,000 - $19,999",
      FAMINC == 600 ~ "$20,000 - $24,999",
      FAMINC == 710 ~ "$25,000 - $29,999",
      FAMINC == 720 ~ "$30,000 - $34,999",
      FAMINC == 730 ~ "$35,000 - $39,999",
      FAMINC == 740 ~ "$40,000 - $49,999",
      FAMINC == 820 ~ "$50,000 - $59,999",
      FAMINC == 830 ~ "$60,000 - $74,999",
      FAMINC == 841 ~ "$75,000 - $99,999",
      FAMINC == 842 ~ "$100,000 - $149,999",
      FAMINC == 843 ~ "$150,000 and over",
      FAMINC %in% c(995, 996, 997, 999) ~ "Missing/Refused/Don't know",
      TRUE ~ "Unknown"
    )
  )

# Aggregate data to the household level
cps_data <- cps %>%
  group_by(CPSID = as.factor(CPSID)) %>%
  summarise(
    COUNTY = first(COUNTY),
    weight = first(HWTFINL),  # Family-level weight
    hhsize = n(),  # Household size
    FSTOTXPNC_perpers = FSTOTXPNC / hhsize,  # Expenditures per person
    FSSTATUS = first(FSSTATUS),  # Food security status
```

```
    FSSTATUSMD = first(FSSTATUSMD),  # Food security moderate status
    FSFOODS = first(FSFOODS),  # Food insecurity indicator
    FSWROUTY = first(FSWROUTY),  # Food worry indicator
    FSBAL = first(FSBAL),  # Food balance indicator
    FSRAWSCRA = first(FSRAWSCRA),  # Raw food score
    FSTOTXPNC = first(FSTOTXPNC),  # Total food expenditure
    female = sum(SEX),  # Count of females in the household
    hispanic = sum(HISPANIC),  # Count of Hispanic individuals
    black = sum(BLACK),  # Count of Black individuals
    kids = sum(CHILD),  # Count of children
    elderly = sum(ELDERLY),  # Count of elderly individuals
    education = sum(EDUC),  # Count of educated individuals
    married = sum(MARRIED),  # Count of married individuals
    income = first(FAMINC_category)  # Family income category
  ) %>%
  ungroup()
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## ℹ Please use `reframe()` instead.
## ℹ When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `summarise()` has grouped output by 'CPSID'. You can override using the
## `.groups` argument.
```

```
# Handle missing values in food security variables
cps_data <- cps_data %>%
  mutate(
    FSSTATUS = ifelse(FSSTATUS %in% c(98, 99), NA, FSSTATUS),
    FSSTATUSMD = ifelse(FSSTATUSMD %in% c(98, 99), NA, FSSTATUSMD),
    FSFOODS = ifelse(FSFOODS %in% c(98, 99), NA, FSFOODS),
    FSWROUTY = ifelse(FSWROUTY %in% c(96, 97, 98, 99), NA, FSWROUTY),
    FSBAL = ifelse(FSBAL %in% c(96, 97, 98, 99), NA, FSBAL),
    FSRAWSCRA = ifelse(FSRAWSCRA %in% c(98, 99), NA, FSRAWSCRA),
    FSTOTXPNC = ifelse(FSTOTXPNC %in% c(999), NA, FSTOTXPNC)
  ) %>%
  mutate(
    FSSTATUS = ifelse(FSSTATUS > 1, 1, 0),
    FSSTATUSMD = ifelse(FSSTATUSMD > 1, 1, 0),
    FSFOODS = ifelse(FSFOODS > 1, 1, 0),
    FSWROUTY = ifelse(FSWROUTY > 1, 1, 0),
    FSBAL = ifelse(FSBAL > 1, 1, 0),
    FSRAWSCRA = ifelse(FSRAWSCRA > 1, 1, 0)
  )
```

Loading in the ACS data

```r
# Load the ACS data (Assuming the SAS data file is in "data" directory)
acs <- read_sas("C:/Users/sophi/OneDrive/Documents/STAT 172/STAT_172_Final_Project/data/spm_pu_2
022.sas7bdat")

# Filter for a specific state (e.g., Iowa, "19") and calculate weights
acs <- acs %>%
  filter(st == "19") %>%  # Filter for state code
  group_by(serialno = as.factor(serialno)) %>%
  arrange(desc(Sex), desc(Age)) %>%
  mutate(weight = first(wt)) %>%  # Assign household weight
  select(-wt) %>%  # Drop individual weight column
  ungroup()

# Create derived variables matching the CPS definitions
acs <- acs %>%
  mutate(
    SEX = Sex - 1,  # Convert SEX to dummy variable (0 = Male, 1 = Female)
    CHILD = ifelse(Age < 18, 1, 0),  # Children under 18
    ELDERLY = ifelse(Age > 60, 1, 0),  # Elderly defined as age > 64
    BLACK = ifelse(Race == 2, 1, 0),  # Black race dummy variable
    HISPANIC = ifelse(Hispanic > 0, 1, 0),  # Hispanic ethnicity dummy variable
    EDUC = as.integer(Education %in% c(3, 4)),  # Education level (e.g., high school or higher)
    MARRIED = as.integer(Mar %in% c(1)),  # Married or partnered
    PUMA = as.factor(PUMA)  # Convert PUMA to a factor
  )

# Aggregate data to the family level
acs_data <- acs %>%
  group_by(serialno = as.factor(serialno)) %>%
  summarise(
    PUMA = first(PUMA),  # Retain PUMA for household
    hhsize = length(serialno),  # Household size
    female = sum(SEX),  # Number of females
    hispanic = sum(HISPANIC),  # Number of Hispanic individuals
    black = sum(BLACK),  # Number of Black individuals
    kids = sum(CHILD),  # Number of children
    elderly = sum(ELDERLY),  # Number of elderly individuals
    education = sum(EDUC),  # Number of educated individuals
    married = sum(MARRIED),  # Number of married individuals
    AGI = first(AGI),  # Adjusted Gross Income
    weight = weight[1]  # Household weight
  )

# Create income categories
acs_data <- acs_data %>%
  mutate(
    income = case_when(
      AGI < 5000 ~ "Under $5,000",
      AGI >= 5000 & AGI <= 7499 ~ "$5,000 - $7,499",
      AGI >= 7500 & AGI <= 9999 ~ "$7,500 - $9,999",
      AGI >= 10000 & AGI <= 12499 ~ "$10,000 - $12,499",
      AGI >= 12500 & AGI <= 14999 ~ "$12,500 - $14,999",
```

```
      AGI >= 15000 & AGI <= 19999 ~ "$15,000 - $19,999",
      AGI >= 20000 & AGI <= 24999 ~ "$20,000 - $24,999",
      AGI >= 25000 & AGI <= 29999 ~ "$25,000 - $29,999",
      AGI >= 30000 & AGI <= 34999 ~ "$30,000 - $34,999",
      AGI >= 35000 & AGI <= 39999 ~ "$35,000 - $39,999",
      AGI >= 40000 & AGI <= 49999 ~ "$40,000 - $49,999",
      AGI >= 50000 & AGI <= 59999 ~ "$50,000 - $59,999",
      AGI >= 60000 & AGI <= 74999 ~ "$60,000 - $74,999",
      AGI >= 75000 & AGI <= 99999 ~ "$75,000 - $99,999",
      AGI >= 100000 & AGI <= 149999 ~ "$100,000 - $149,999",
      AGI >= 150000 ~ "$150,000 and over",
      TRUE ~ "Unknown"  # Catch undefined or NA cases
    )
  )
```

Data Preparation

```
# Cleaning CPS data: removing unnecessary columns and handling missing values
cps_data = subset(cps_data, select = -c(FSTOTXPNC_perpers, FSSTATUSMD, FSSTATUS, FSWROUTY, FSBA
L, FSRAWSCRA, FSTOTXPNC))
cps_data = cps_data[complete.cases(cps_data), ]
```

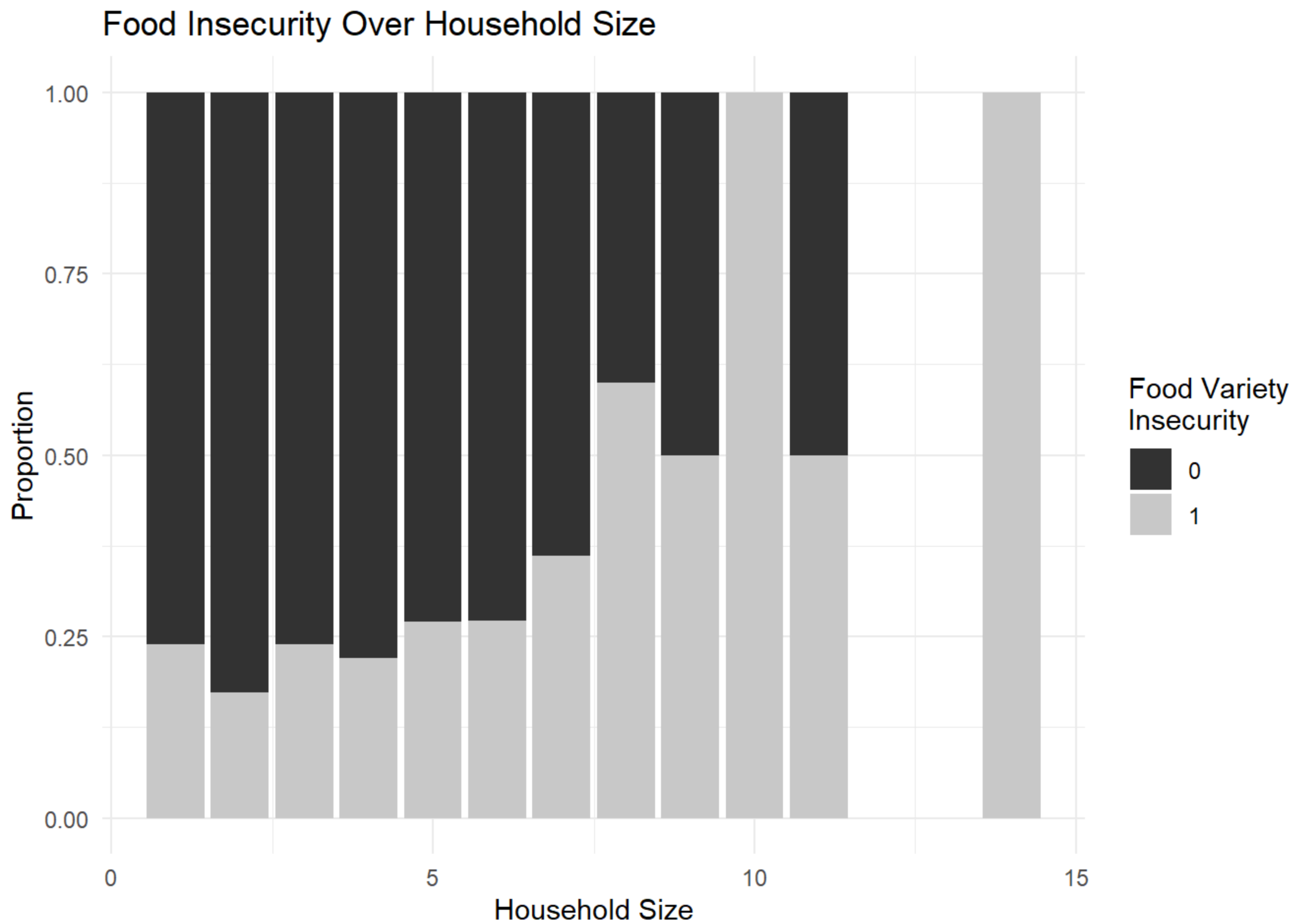
Exploratory Analysis

Data Preparation for Visualization

```
# Adding a factor column for FSFOODS
cps_data = cps_data %>%
  mutate(FSFOODS_factor = as.factor(FSFOODS))
```

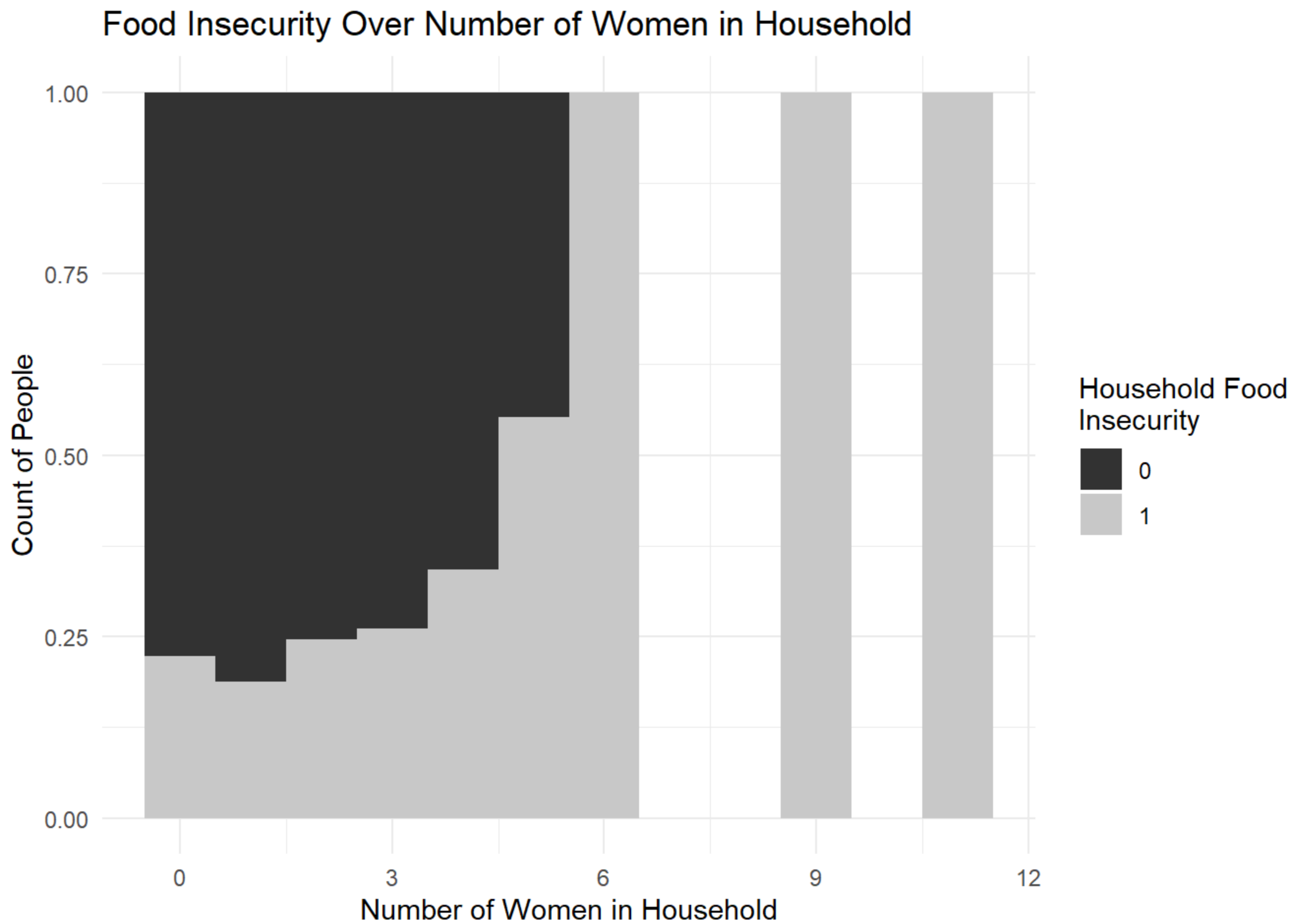
Food Insecurity Visualizations

```
# 1. Food Insecurity Over Household Size
ggplot(data=cps_data) +
  geom_bar(aes(x=hhsize, fill=FSFOODS_factor), position="fill") +
  labs(x="Household Size", y="Proportion") +
  ggtitle("Food Insecurity Over Household Size") +
  scale_fill_grey("Food Variety\nInsecurity") +
  theme_minimal()
```

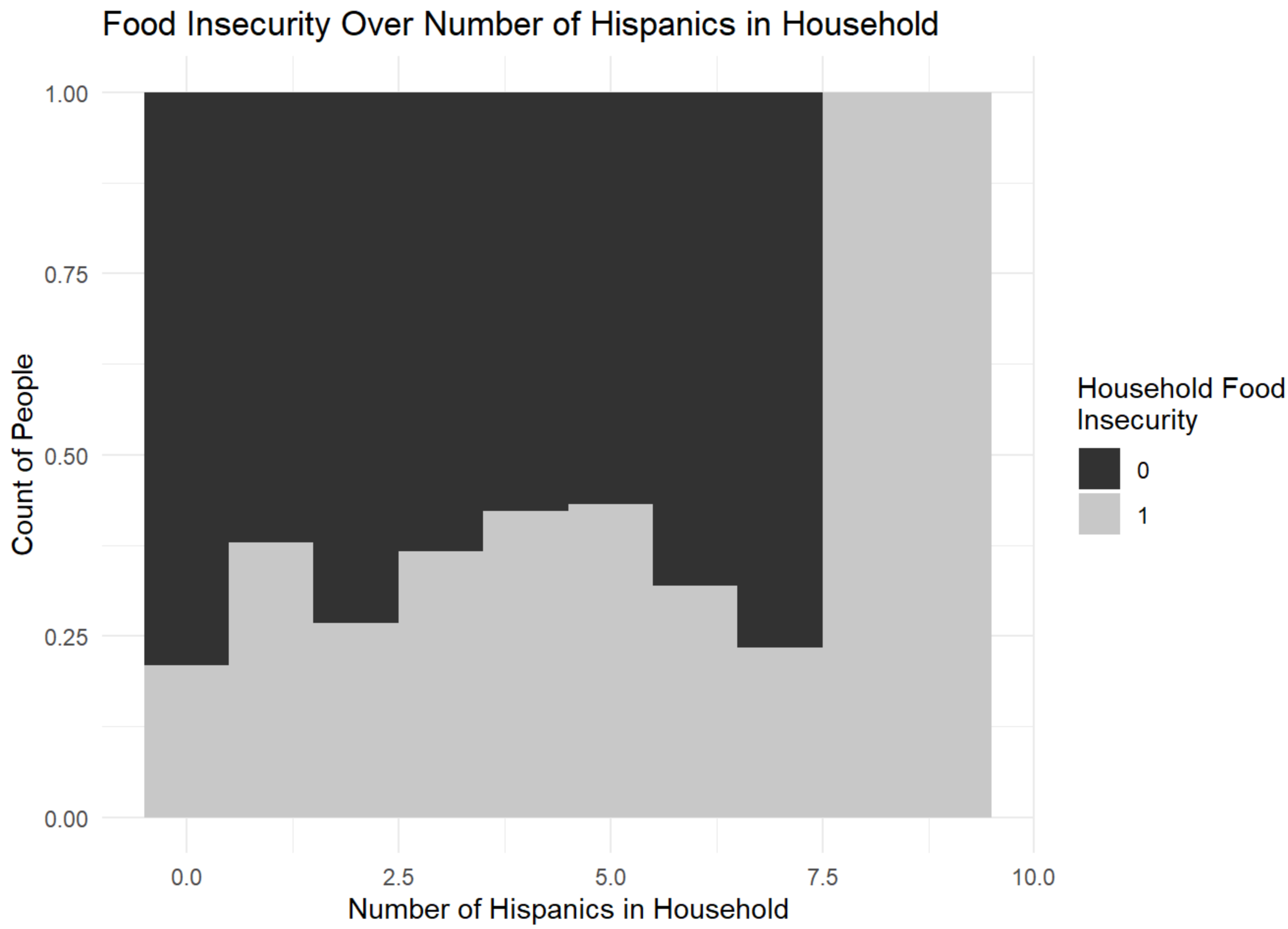



```
# 2. Food Insecurity Over the Number of Women in the Household:
ggplot(data=cps_data) +
  geom_histogram(aes(x=female, fill = FSFOODS_factor), binwidth = 1, position = "fill") +
  labs(x="Number of Women in Household", y="Count of People") +
  ggtitle("Food Insecurity Over Number of Women in Household") +
  scale_fill_grey("Household Food\nInsecurity") +
  theme_minimal()
```

```
## Warning: Removed 6 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```

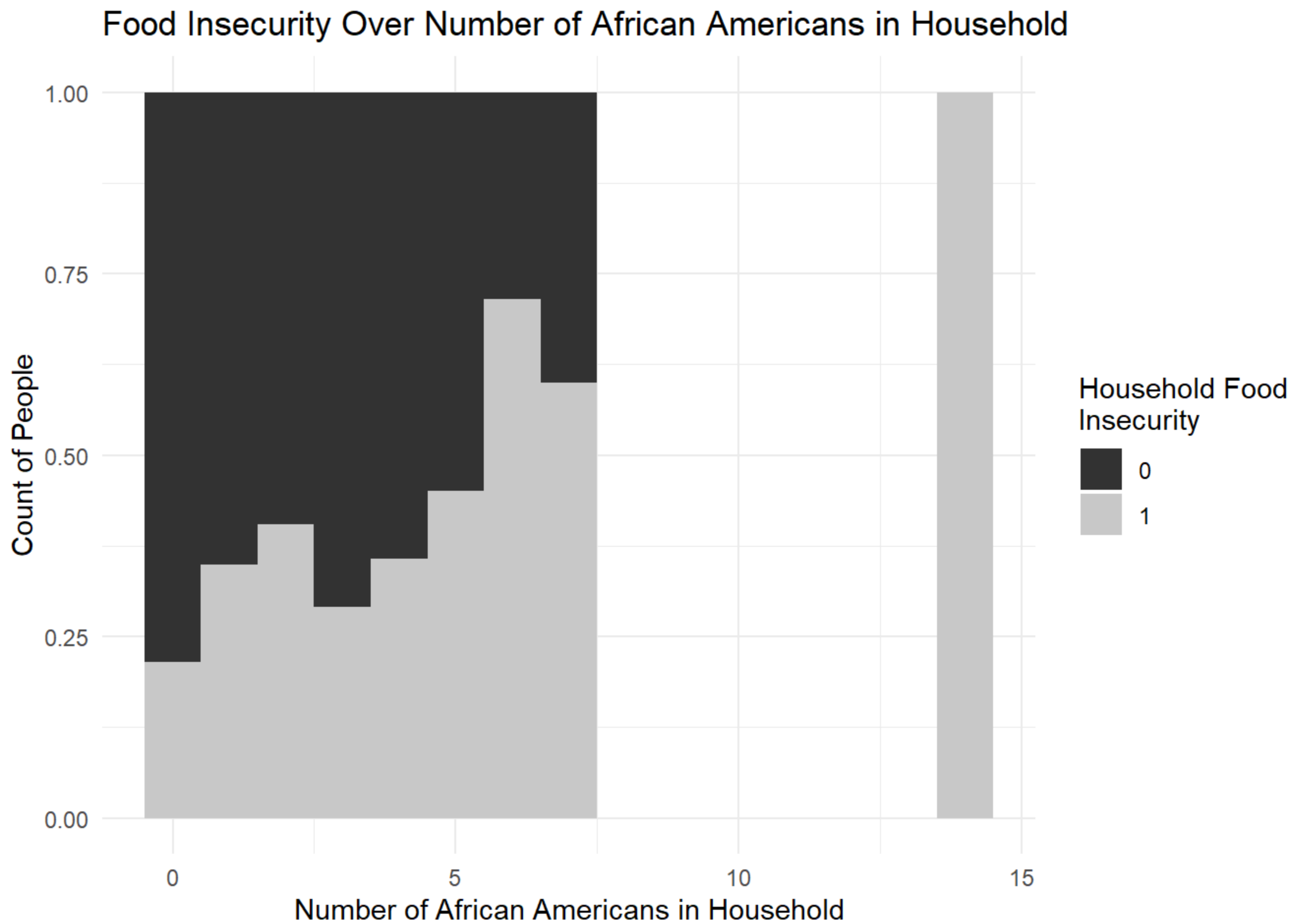



```
# 3. Food Insecurity Over the Number of Hispanics in the Household:
ggplot(data=cps_data) +
  geom_histogram(aes(x=hispanic, fill = FSFOODS_factor), binwidth = 1, position = "fill") +
  labs(x="Number of Hispanics in Household", y="Count of People") +
  ggtitle("Food Insecurity Over Number of Hispanics in Household") +
  scale_fill_grey("Household Food\nInsecurity") +
  theme_minimal()
```



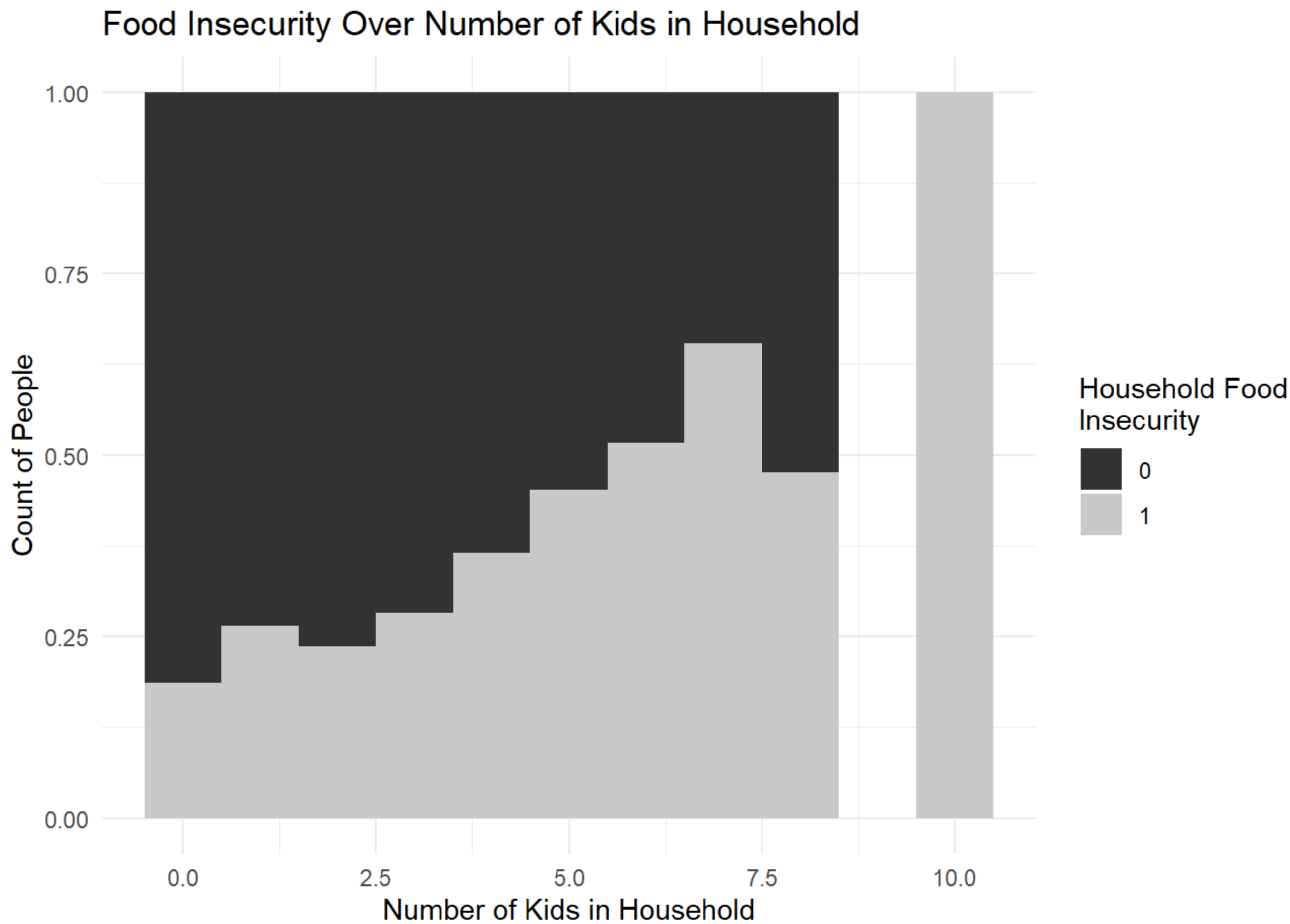

```
# 4. Food Insecurity Over the Number of African Americans in the Household:
ggplot(data=cps_data) +
  geom_histogram(aes(x=black, fill = FSFOODS_factor), binwidth = 1, position = "fill") +
  labs(x="Number of African Americans in Household", y="Count of People") +
  ggtitle("Food Insecurity Over Number of African Americans in Household") +
  scale_fill_grey("Household Food\nInsecurity") +
  theme_minimal()
```

```
## Warning: Removed 12 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```

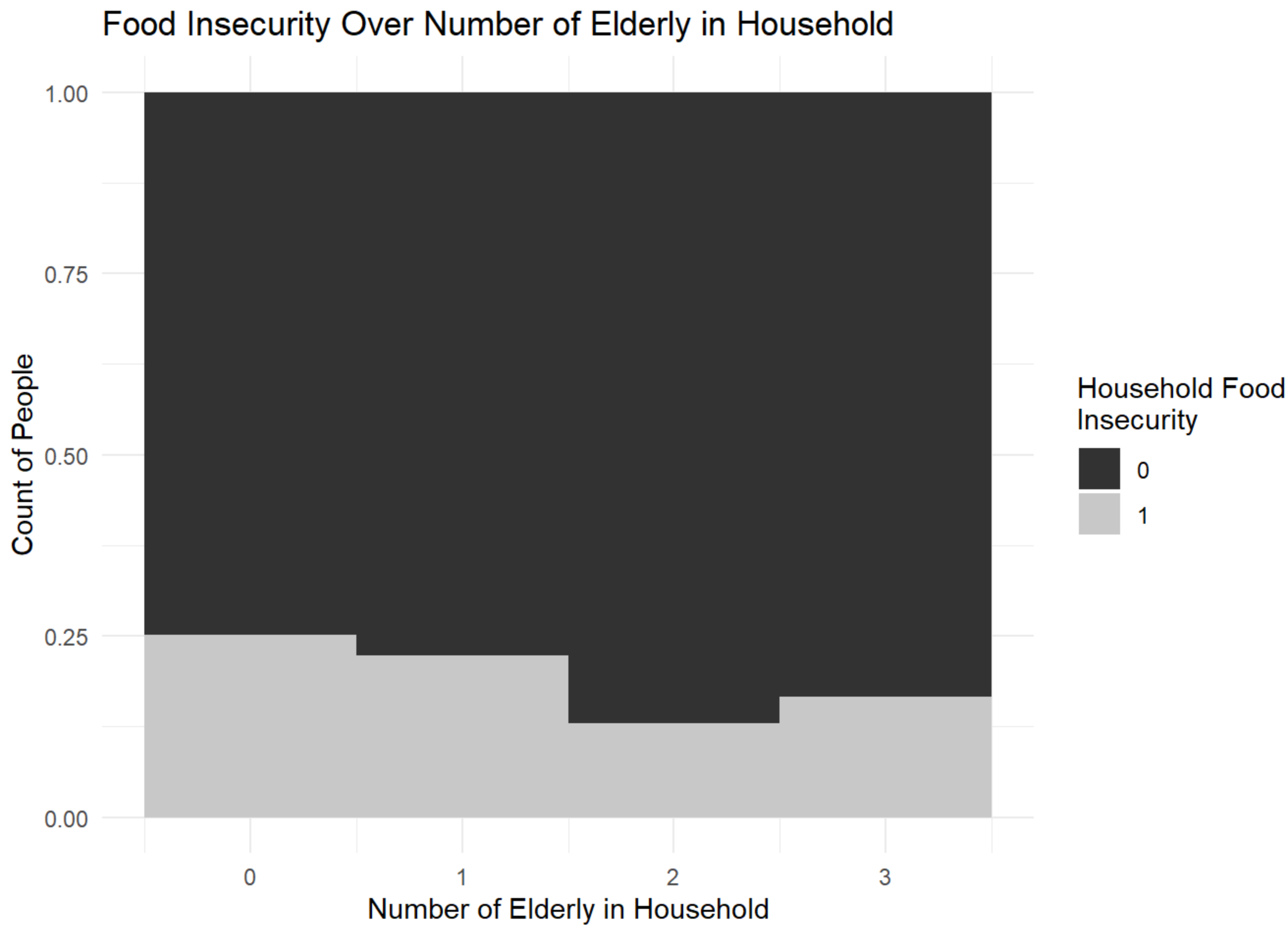



```
# 5. Food Insecurity Over the Number of Kids in the Household:
ggplot(data=cps_data) +
  geom_histogram(aes(x=kids, fill = FSFOODS_factor), binwidth = 1, position = "fill") +
  labs(x="Number of Kids in Household", y="Count of People") +
  ggtitle("Food Insecurity Over Number of Kids in Household") +
  scale_fill_grey("Household Food\nInsecurity") +
  theme_minimal()
```

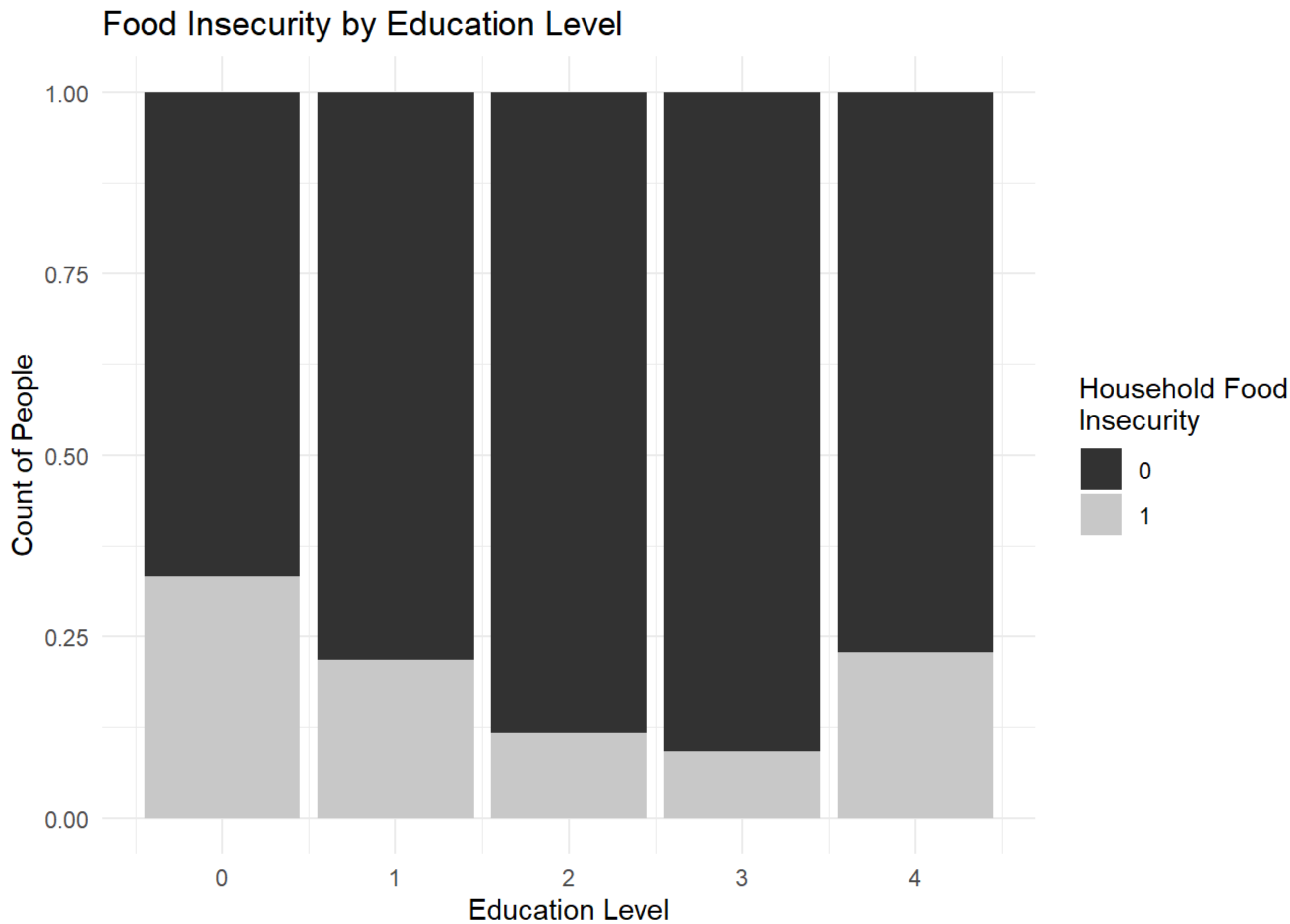
```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```



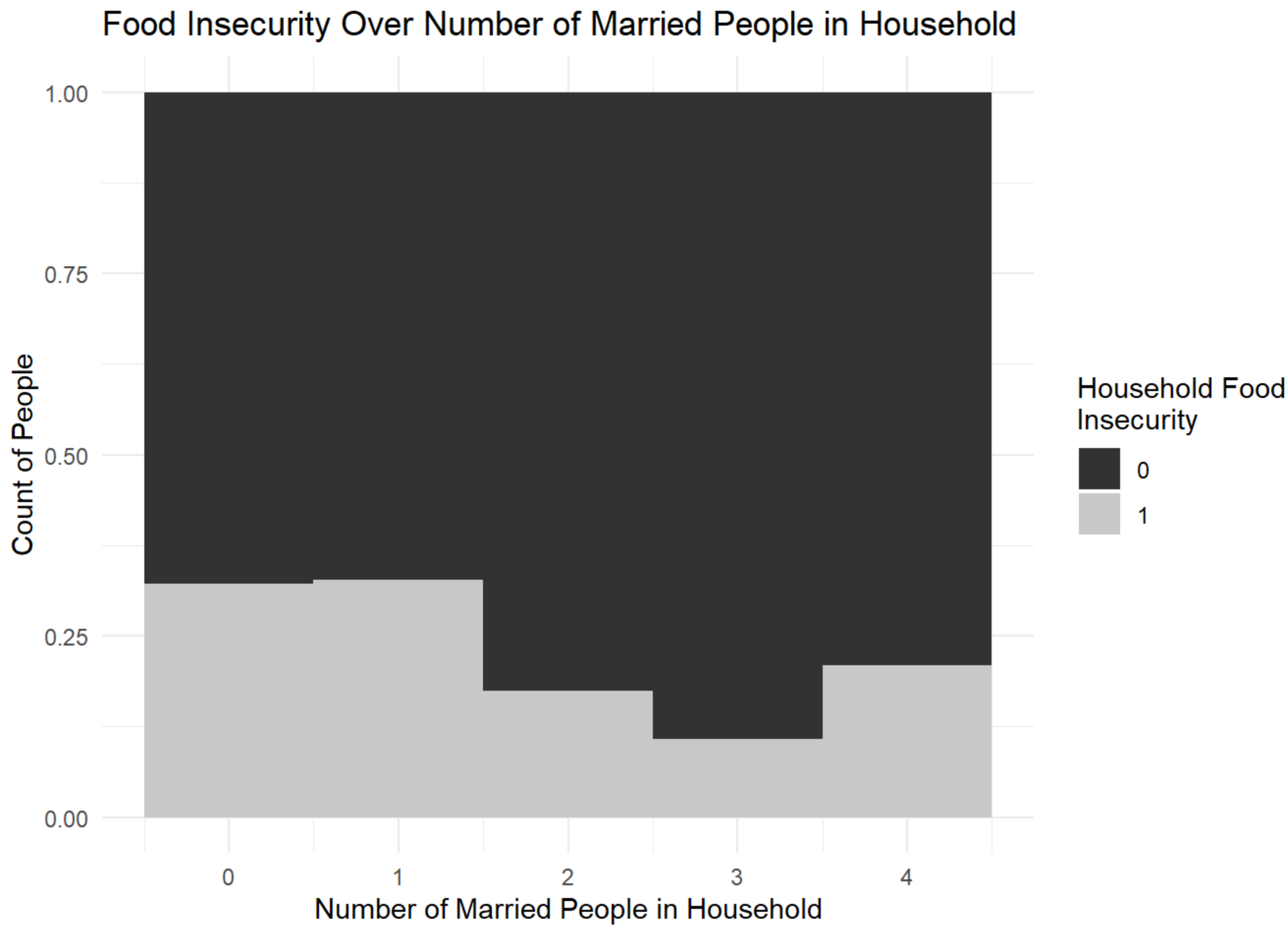

```
# 6. Food Insecurity Over the Number of Elderly in the Household:
ggplot(data=cps_data) +
  geom_histogram(aes(x=elderly, fill = FSFOODS_factor), binwidth = 1, position = "fill") +
  labs(x="Number of Elderly in Household", y="Count of People") +
  ggtitle("Food Insecurity Over Number of Elderly in Household") +
  scale_fill_grey("Household Food\nInsecurity") +
  theme_minimal()
```




```
# 7. Food Insecurity by Education Level:
ggplot(data=cps_data) +
  geom_bar(aes(x=education, fill = FSFOODS_factor), position = "fill") +
  labs(x="Education Level", y="Count of People") +
  ggtitle("Food Insecurity by Education Level") +
  scale_fill_grey("Household Food\nInsecurity") +
  theme_minimal()
```




```
# 8. Food Insecurity Over the Number of Married People in the Household:
ggplot(data=cps_data) +
  geom_histogram(aes(x=married, fill = FSFOODS_factor), binwidth = 1, position = "fill") +
  labs(x="Number of Married People in Household", y="Count of People") +
  ggtitle("Food Insecurity Over Number of Married People in Household") +
  scale_fill_grey("Household Food\nInsecurity") +
  theme_minimal()
```



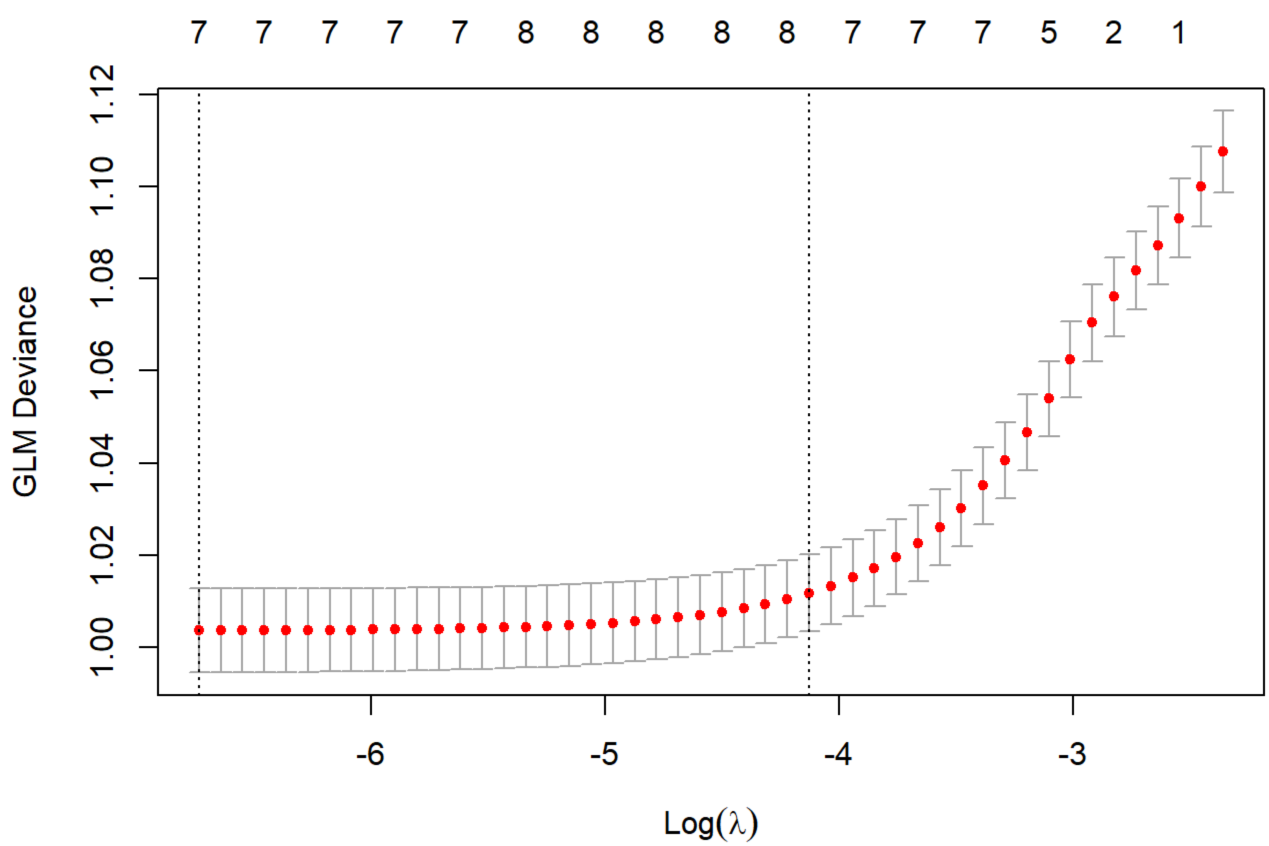
Splitting the Data into Training and Testing Sets

```
# Setting a random seed for reproducibility
RNGkind(sample.kind = "default")
set.seed(122111598)

# Splitting the data into training (70%) and testing (30%) sets
train.idx = sample(x=1:nrow(cps_data), size = floor(.7*nrow(cps_data)))

# Creating training and testing datasets
train.df = cps_data[train.idx,]
test.df = cps_data[-train.idx,]
```

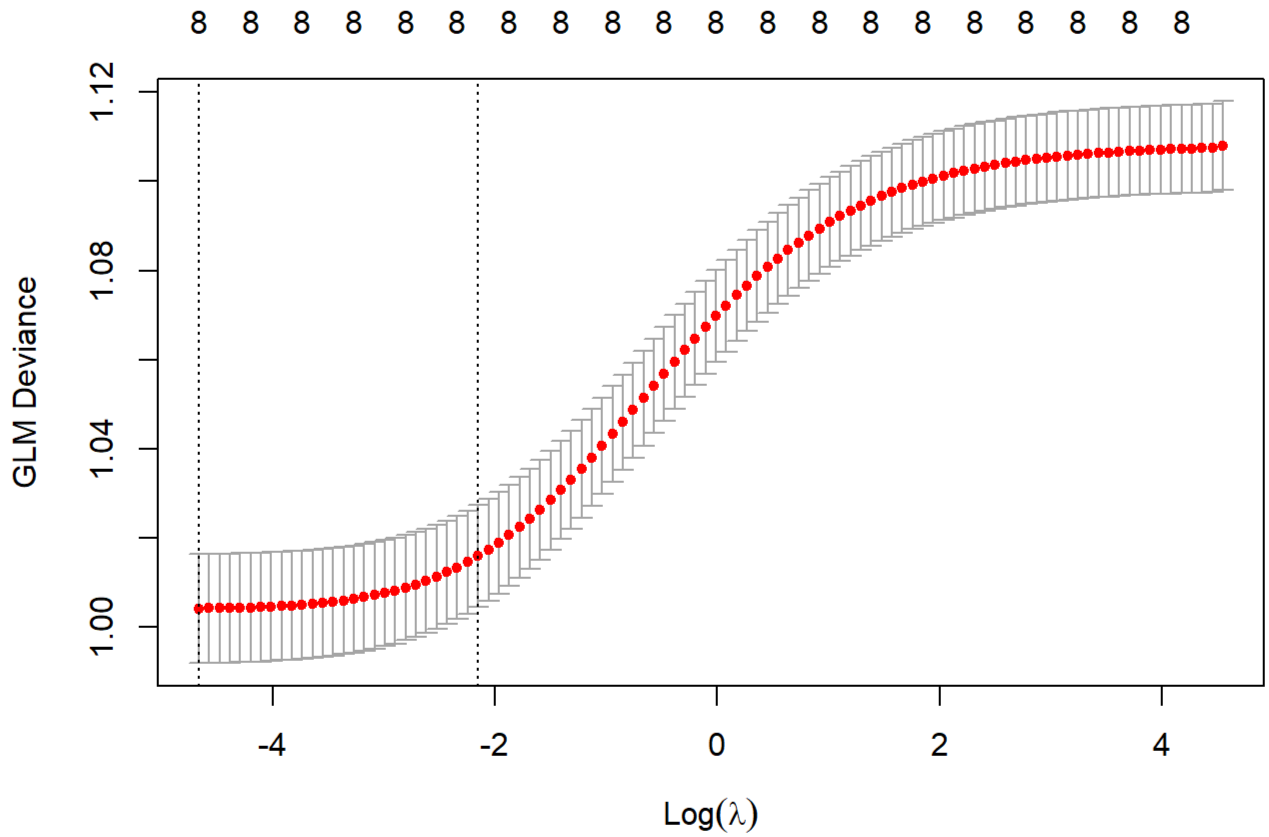
Lasso and Ridge Regression

Cross-Validation for Lambda

```
# Creating design matrices
x.train = model.matrix(FSFOODS ~ hhsize + female + hispanic + black + kids + elderly + education
+ married, data = train.df)[, -1]
x.test = model.matrix(FSFOODS ~ hhsize + female + hispanic + black + kids + elderly + education
+ married, data = test.df)[, -1]

# Creating response vectors
y.train = as.vector(train.df$FSFOODS)
y.test = as.vector(test.df$FSFOODS)

# Cross-validation
lr_lasso_cv = cv.glmnet(x.train, y.train, family = binomial(link = logit), weights = as.integer
(train.df$weight), alpha = 1)
lr_ridge_cv = cv.glmnet(x.train, y.train, family = binomial(link = logit), weights = as.integer
(train.df$weight), alpha = 0)

# Plotting cross-validation results
plot(lr_lasso_cv)
```



```
plot(lr_ridge_cv)
```



Fitting Final Models

```
# Extracting best lambda values
best_lasso_lambda = lr_lasso_cv$lambda.min
best_ridge_lambda = lr_ridge_cv$lambda.min

# Fitting models
final_lasso = glmnet(x.train, y.train, family = binomial(link = "logit"), weights = as.integer(t
rain.df$weight), alpha = 1, lambda = best_lasso_lambda)
final_ridge = glmnet(x.train, y.train, family = binomial(link = "logit"), weights = as.integer(t
rain.df$weight), alpha = 0, lambda = best_ridge_lambda)
```

Model Performance and Predictions

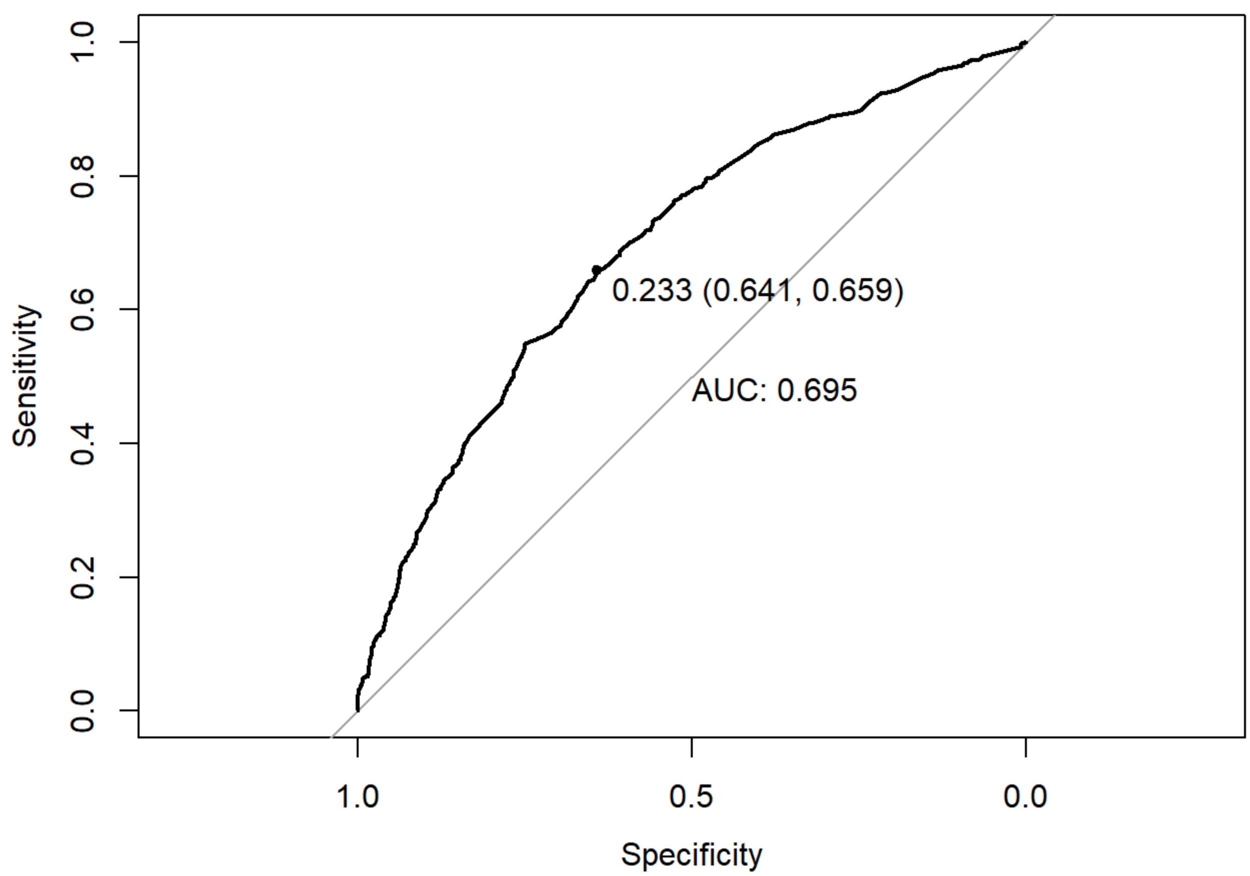
```
# Adding predictions to test dataset
test.df.preds = test.df %>%
  mutate(
    lasso_pred = predict(final_lasso, x.test, type = "response")[,1],
    ridge_pred = predict(final_ridge, x.test, type = "response")[,1]
  )

# Generating ROC curves
lasso_rocCurve = roc(response = as.factor(test.df.preds$FSFOODS), predictor = test.df.preds$lass
o_pred, levels = c("0", "1"))
```

```
## Setting direction: controls < cases
```

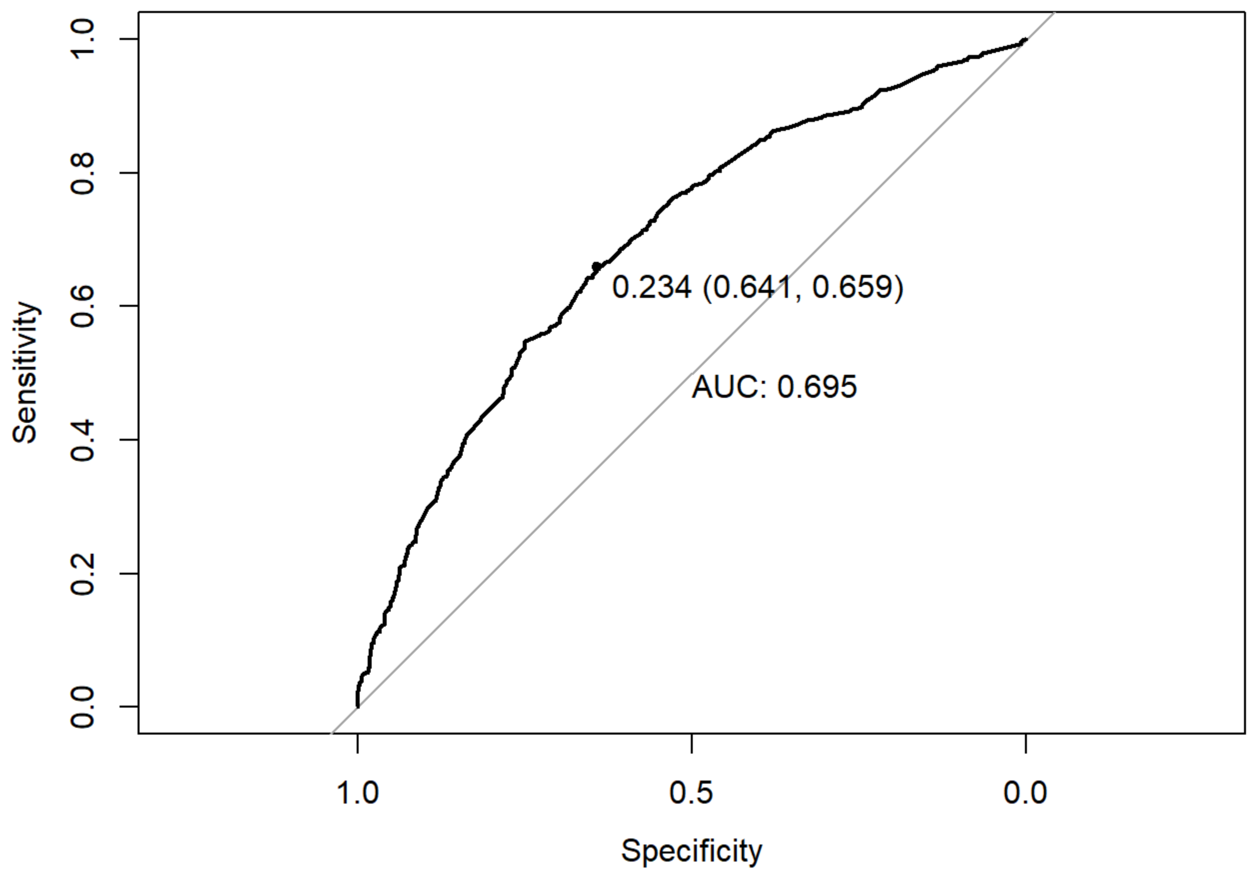
```
ridge_rocCurve = roc(response = as.factor(test.df.preds$FSFOODS), predictor = test.df.preds$ridg
e_pred, levels = c("0", "1"))
```

```
## Setting direction: controls < cases
```

```
# Plotting ROC curves
plot(lasso_rocCurve, print.thres = TRUE, print.auc = TRUE)
```



```
plot(ridge_rocCurve, print.thres = TRUE, print.auc = TRUE)
```



Coefficients and Interpretation

```
# Extracting coefficients
lr_lasso_coeff = coef(lr_lasso_cv, s = "lambda.min") %>% as.matrix()
lr_ridge_coeff = coef(lr_ridge_cv, s = "lambda.min") %>% as.matrix()

# Displaying coefficients
lr_lasso_coeff
```

```
##                      s1
## (Intercept) -0.82904528
## hhsize       0.08816995
## female       0.17770841
## hispanic     0.14421753
## black        0.13396168
## kids         0.00000000
## elderly     -0.19555147
## education   -0.52913233
## married     -0.38062406
```

```
lr_ridge_coeff
```

```
##                      s1
## (Intercept) -0.83079698
## hhsize       0.07671354
## female       0.16946228
## hispanic     0.14549445
## black        0.13548186
## kids         0.01470601
## elderly     -0.19167190
## education   -0.50530959
## married     -0.36706903
```

Predicting on ACS Data

```
# Preparing ACS data
acs_test = subset(acs_data, select = c(hhsize, female, hispanic, black, kids, elderly, educatio
n, married))
acs_test = as.matrix(acs_test)

# Adding predictions to ACS data
acs_data = acs_data %>%
  mutate(
    lasso_pred_prob = predict(final_lasso, acs_test, type = "response")[,1],
    ridge_pred_prob = predict(final_ridge, acs_test, type = "response")[,1]
  )

# Aggregating predictions by PUMA
puma_acs = acs_data %>%
  filter(elderly > 0) %>%
  group_by(PUMA = as.factor(PUMA)) %>%
  summarise(
    mean_lasso_prob = weighted.mean(lasso_pred_prob, weights = weight),
    mean_ridge_prob = weighted.mean(ridge_pred_prob, weights = weight),
    senior_count = sum(elderly)
  ) %>% ungroup()

# Finding PUMAs with highest probabilities
puma_acs[max(puma_acs$mean_lasso_prob) == puma_acs$mean_lasso_prob,]
```

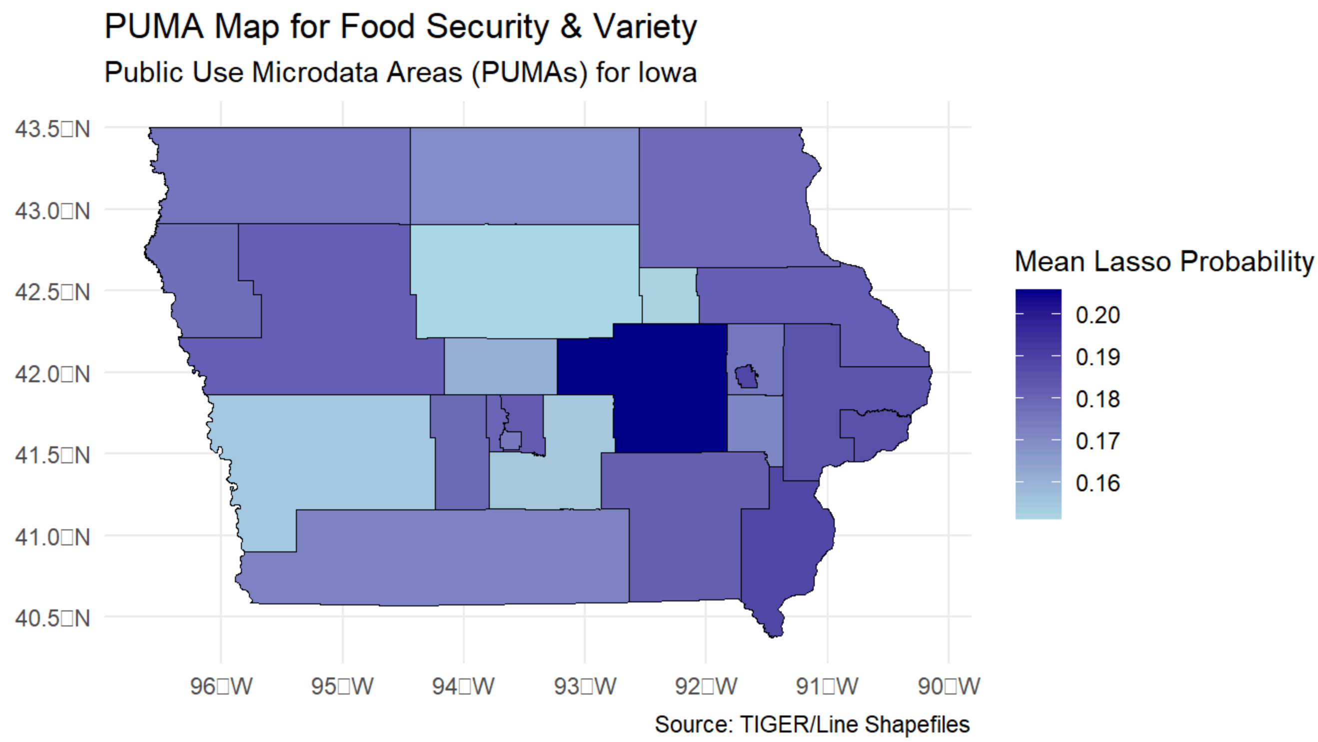
```
## # A tibble: 1 × 4
##   PUMA    mean_lasso_prob mean_ridge_prob senior_count
##   <fct>             <dbl>           <dbl>        <dbl>
## 1 1901503           0.206           0.206          240
```

```
puma_acs[max(puma_acs$mean_ridge_prob) == puma_acs$mean_ridge_prob,]
```

```
## # A tibble: 1 × 4
##   PUMA    mean_lasso_prob mean_ridge_prob senior_count
##   <fct>             <dbl>           <dbl>        <dbl>
## 1 1901503           0.206           0.206          240
```

Mapping PUMA Areas

```
# Fetching PUMA shapefile data
options(tigris_class = "sf")
options(tigris_use_cache = TRUE)

pumas <- pumas(state = "IA", year = 2022)

# Creating a map of PUMAs colored by Lasso probabilities
ggplot(data = pumas) +
  geom_sf(aes(fill = puma_acs$mean_lasso_prob), color = "black") +
  scale_fill_gradient(
    low = "lightblue", high = "darkblue",  # Color gradient
    name = "Mean Lasso Probability"
  ) +
  labs(
    title = "PUMA Map for Food Security & Variety",
    subtitle = "Public Use Microdata Areas (PUMAs) for Iowa",
    caption = "Source: TIGER/Line Shapefiles"
  ) +
  theme_minimal()
```


PUMA Map for Food Security & Variety
Public Use Microdata Areas (PUMAs) for Iowa
43.5□N
43.0□N
42.5□N
42.0□N
41.5□N
41.0□N
40.5□N
96□W
95□W
94□W
93□W
92□W
91□W
90□W
Mean Lasso Probability
0.20
0.19
0.18
0.17
0.16
Source: TIGER/Line Shapefiles