STATS101C Regression Project

Jolina Hor

2025-12-01

Libraries

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyr)
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(janitor)
```

```
##
## Attaching package: 'janitor'

## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```r
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(mlr3)
library(mlr3learners)
library(mlr3viz)
library(mlr3pipelines)
library(ranger)
library(xgboost)
```

```
##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-10
```

READ IN DATA

```r
purchases <- read.csv("amazon-purchases.csv")
survey <- read.csv("survey_train_test.csv")
```

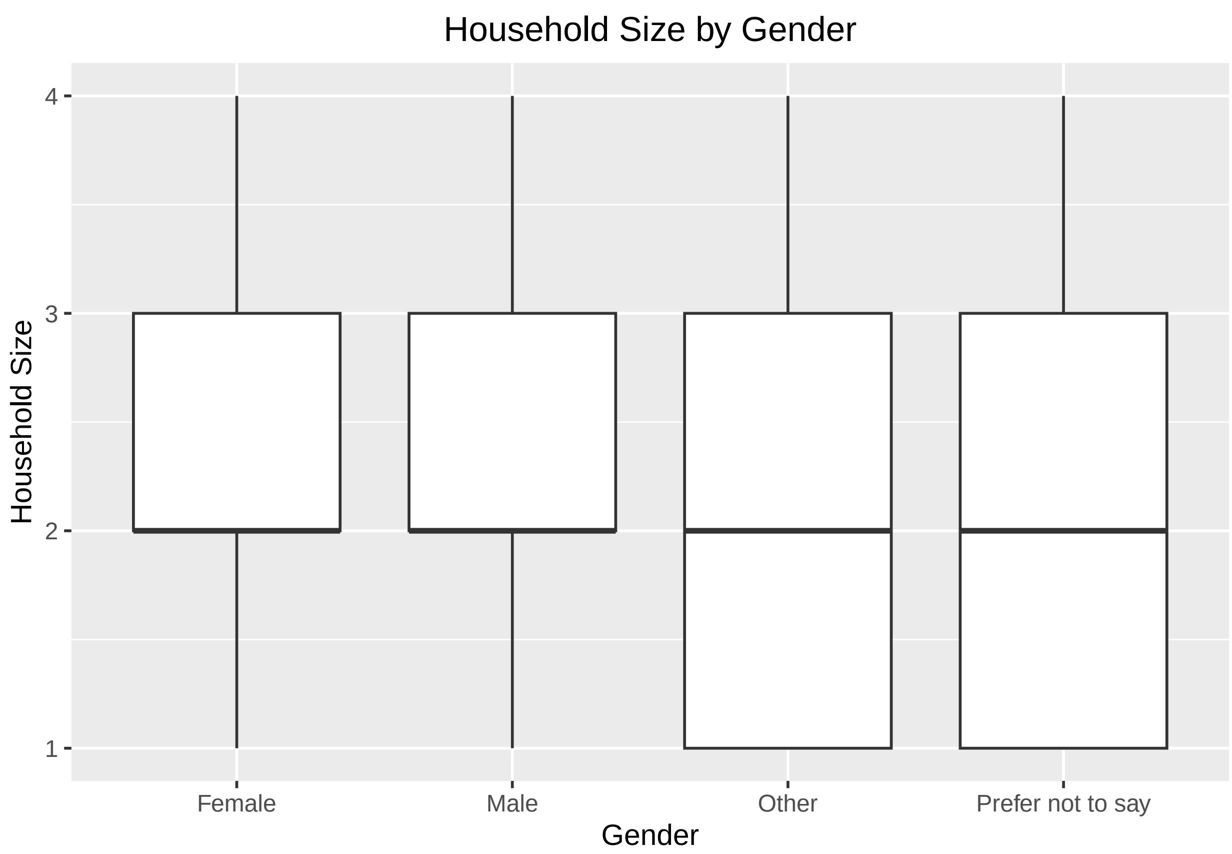
CLEAN SURVEY

```r
survey_clean <- survey %>%
  select(
    response_id = Survey.ResponseID,
    household_size_num = Q.amazon.use.hh.size.num,
    gender = Q.demos.gender,
    state = Q.demos.state,
    is_test = test
  )
```

SURVEY DATA EDA

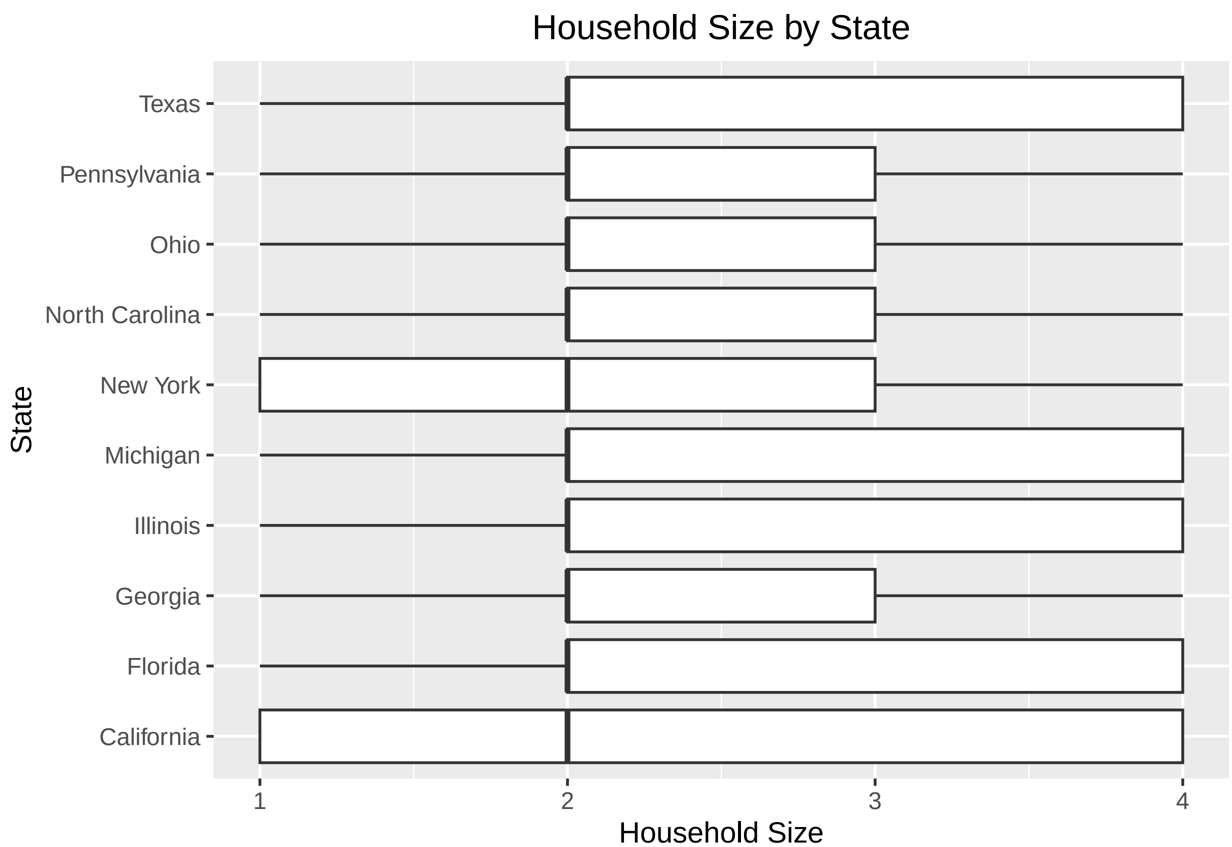
```r
#1st plot - household size by gender
survey_clean %>%
  ggplot(aes(x = gender, y =  household_size_num)) +
  geom_boxplot() +
  labs(title = "Household Size by Gender",
       x = "Gender",
       y = "Household Size") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning: Removed 2000 rows containing non-finite outside the scale range
## (‘stat_boxplot()‘).
```



```r
#2nd plot - household size by state
#grab the top 10 states for easier visuals
top_states <- survey_clean %>%
  count(state, sort = TRUE) %>%
  slice_head(n = 10) %>%
  pull(state)

#household size by state
survey_clean %>%
  filter(state %in% top_states) %>%
  ggplot(aes(x = reorder(state, household_size_num),
             y = household_size_num)) +
  geom_boxplot() +
  coord_flip() +
  labs(title="Household Size by State",
       x= "State", y= "Household Size") +
    theme(plot.title = element_text(hjust = 0.5))
```

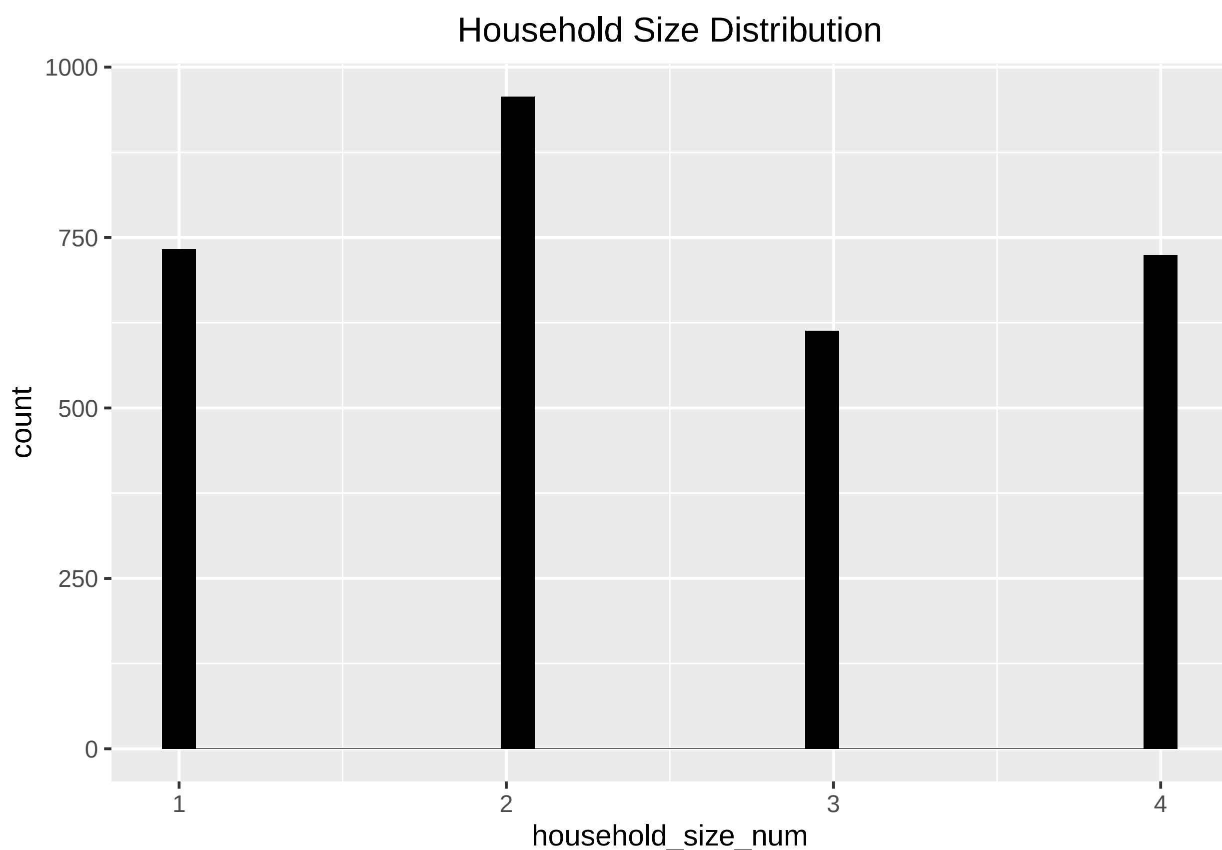
```
## Warning: Removed 1097 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```




```r
#3rd plot - household size distribution
survey_clean %>%
  ggplot(aes(household_size_num)) +
  geom_histogram(fill = "black") +
```

```r
  labs(title = "Household Size Distribution") +
    theme(plot.title = element_text(hjust = 0.5))
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.

## Warning: Removed 2000 rows containing non-finite outside the scale range
## (`stat_bin()`).
```



CLEAN PURCHASES

```r
## CLEAN PURCHASES
# Top 1000 Categories
top_cats <- purchases %>%
  filter(Category != "") %>%
  count(Category, sort = TRUE) %>%
  slice_head(n = 1000) %>%
  pull(Category)

purchases_clean <- purchases %>%
  rename(
    order_date_raw = Order.Date,
```

```r
    price_per_unit = Purchase.Price.Per.Unit,
    quantity = Quantity,
    category = Category,
    response_id = Survey.ResponseID
  ) %>%
  mutate(
    #order_date = as.Date(order_date_raw, format = "%m/%d/%Y"),
    order_date = as.Date(order_date_raw, format = "%Y-%m-%d"),

    # buckets
    top_categories = ifelse(category %in% top_cats, category, "Other"),
    total_item_price = price_per_unit * quantity
  ) %>%
  filter(!is.na(price_per_unit), !is.na(quantity))
```

FEATURE ENGINEERING

```r
# numeric aggregate statistics
user_general_stats <- purchases_clean %>%
  group_by(response_id) %>%
  summarise(
    # we discussed as a team what kind of aggregate data would be helpful for predicting household size
    total_spend = sum(total_item_price),
    total_items = sum(quantity),
    total_orders = n_distinct(order_date),
    unique_categories = n_distinct(category),
    avg_order_value = mean(total_item_price),
    avg_items_per_order = sum(quantity) / n_distinct(order_date),
    avg_unit_price = mean(price_per_unit),

    #time based features
    avg_days_between_orders = as.numeric(max(order_date) - min(order_date)) / n()
  ) %>%
  mutate(
    # spend, item, price data was heavily left skewed
    log_total_spend = log1p(total_spend),
    log_total_items = log1p(total_items),
    log_avg_unit_price = log1p(avg_unit_price)
  )

# categorical aggregate statistics
user_category_stats <- purchases_clean %>%
  group_by(response_id, top_categories) %>%
  summarise(
    spend = sum(total_item_price),
    items = sum(quantity),
    .groups = "drop"
  ) %>%
  pivot_wider(
    names_from = top_categories,
    values_from = c(spend, items),
```

```
    values_fill = 0
  ) %>%
  clean_names()
```

PCA

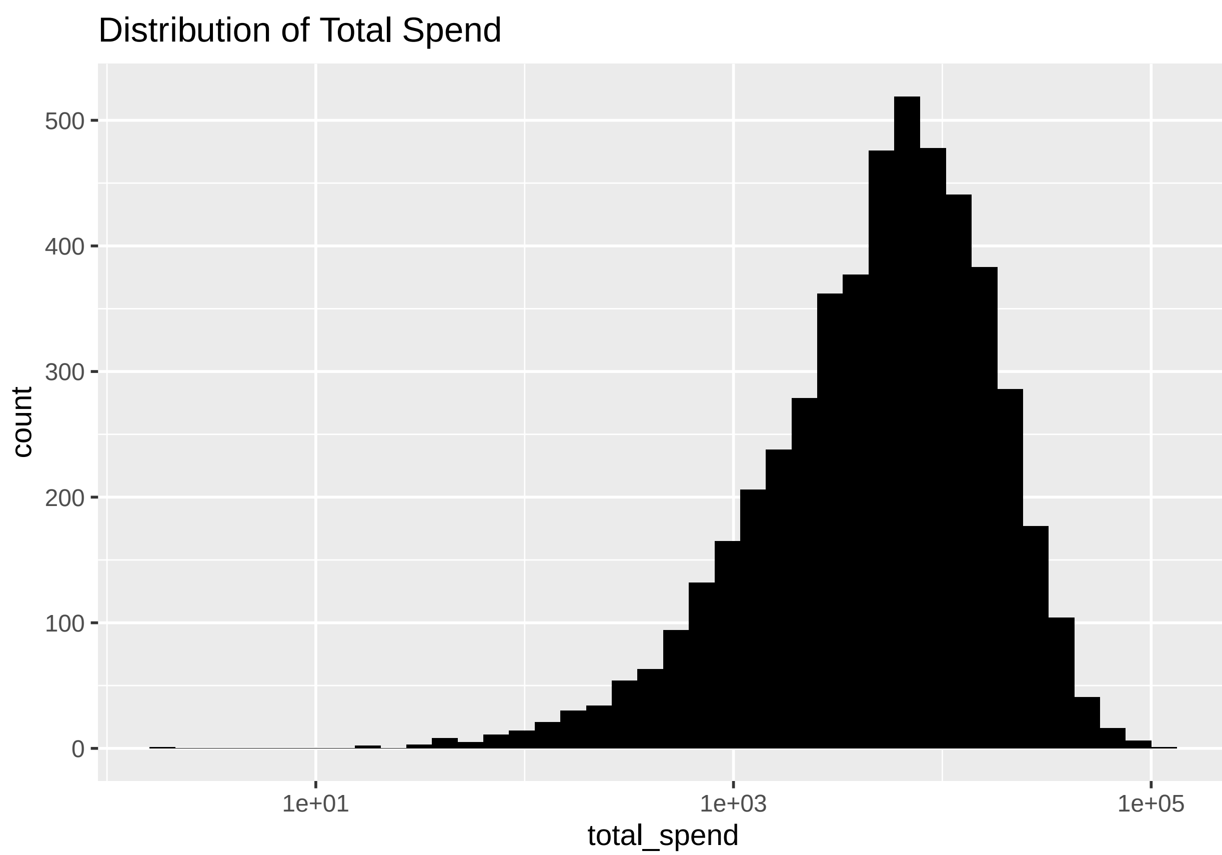
```
pca_data <- user_category_stats %>%
  select(starts_with("items_"))

# remove columns with zero variance
pca_data <- pca_data[, apply(pca_data, 2, var) != 0]

# PCA
pca_result <- prcomp(pca_data, center = TRUE, scale. = TRUE)

# top 10 PCs
pca_features <- as.data.frame(pca_result$x[, 1:10])
colnames(pca_features) <- paste0("PC", 1:10)

# response_id directly from the original table
pca_df <- cbind(response_id = user_category_stats$response_id, pca_features)
```

JOIN TABLES & SPLIT DATASET

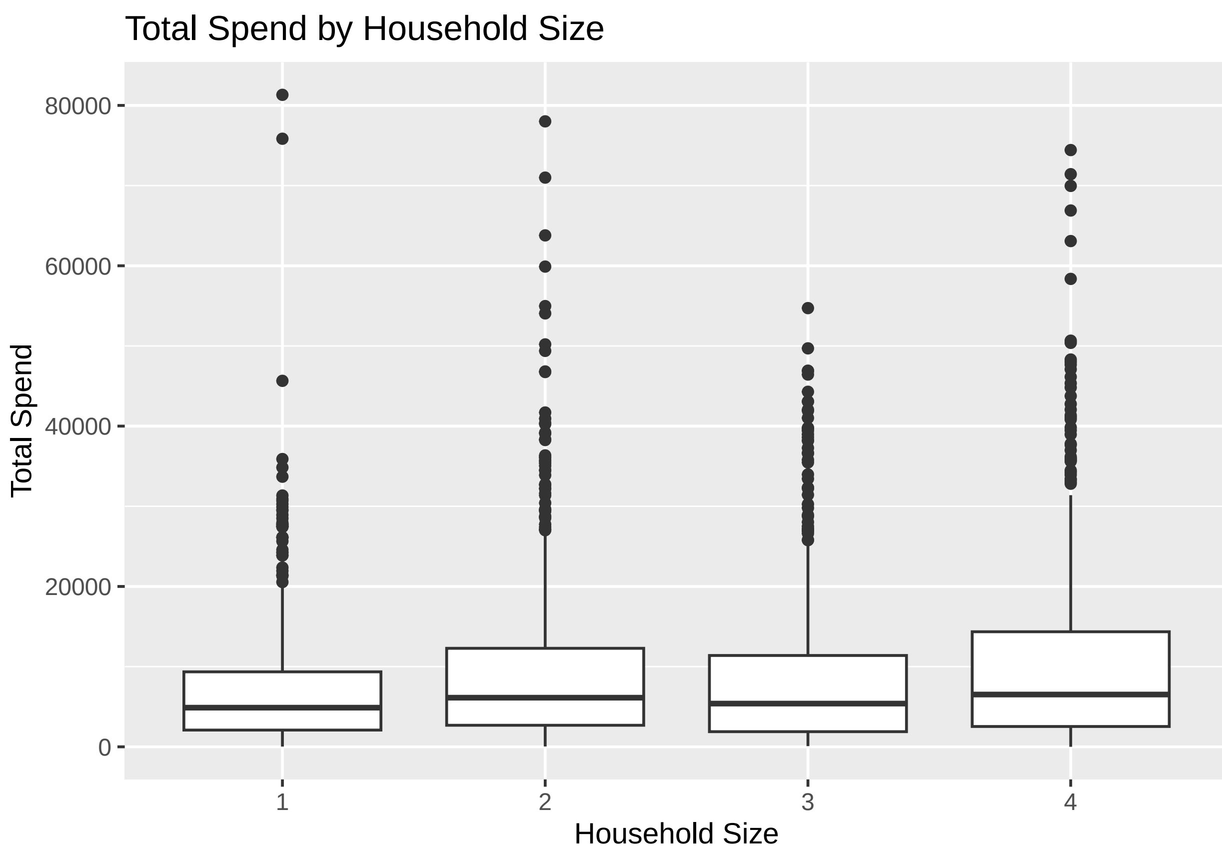
```
# combine features
all_features <- user_general_stats %>%
  left_join(user_category_stats, by = "response_id") %>%
  left_join(pca_df, by = "response_id") %>%
  mutate(
    # use ratios since maybe high-spending families will contribute more to groceries, toys, and baby,
    pct_grocery = spend_grocery / ifelse(total_spend == 0, 1, total_spend),
    pct_toys = spend_toys_and_games / ifelse(total_spend == 0, 1, total_spend),
    pct_baby = spend_baby_product / ifelse(total_spend == 0, 1, total_spend),

    # increases household size
    has_baby = ifelse(items_baby_product > 0 | items_baby_formula > 0 | items_infant_toddler_car_seat >
    has_school_kid = ifelse(items_backpack > 0 | items_writing_instrument > 0, 1, 0),
  )

# join with survey data
final_data <- survey_clean %>%
  left_join(all_features, by = "response_id") %>%
  # NA for $0 purchases
  mutate(across(where(is.numeric) & !household_size_num, ~replace_na(., 0)))

# split to train/test
train_df <- final_data %>% filter(is_test == FALSE, !is.na(household_size_num))
test_df <- final_data %>% filter(is_test == TRUE)
```

AMAZON PURCHASES DATA W/HOUSEHOLD SIZE INFO EDA

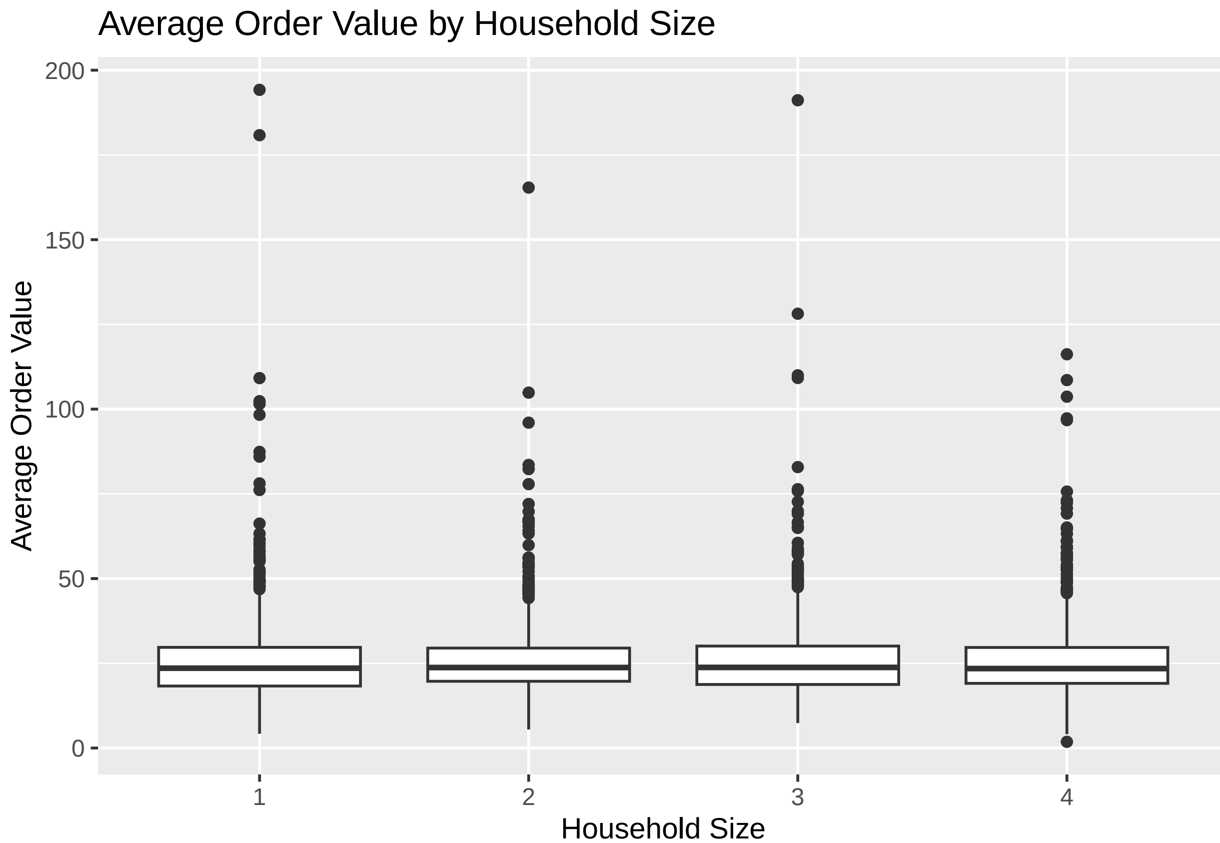
```r
#1st plot - total spend by consumers
ggplot(user_general_stats, aes(total_spend)) +
  geom_histogram(bins = 40, fill = "black") +
  scale_x_log10() +
  labs(title = "Distribution of Total Spend")
```



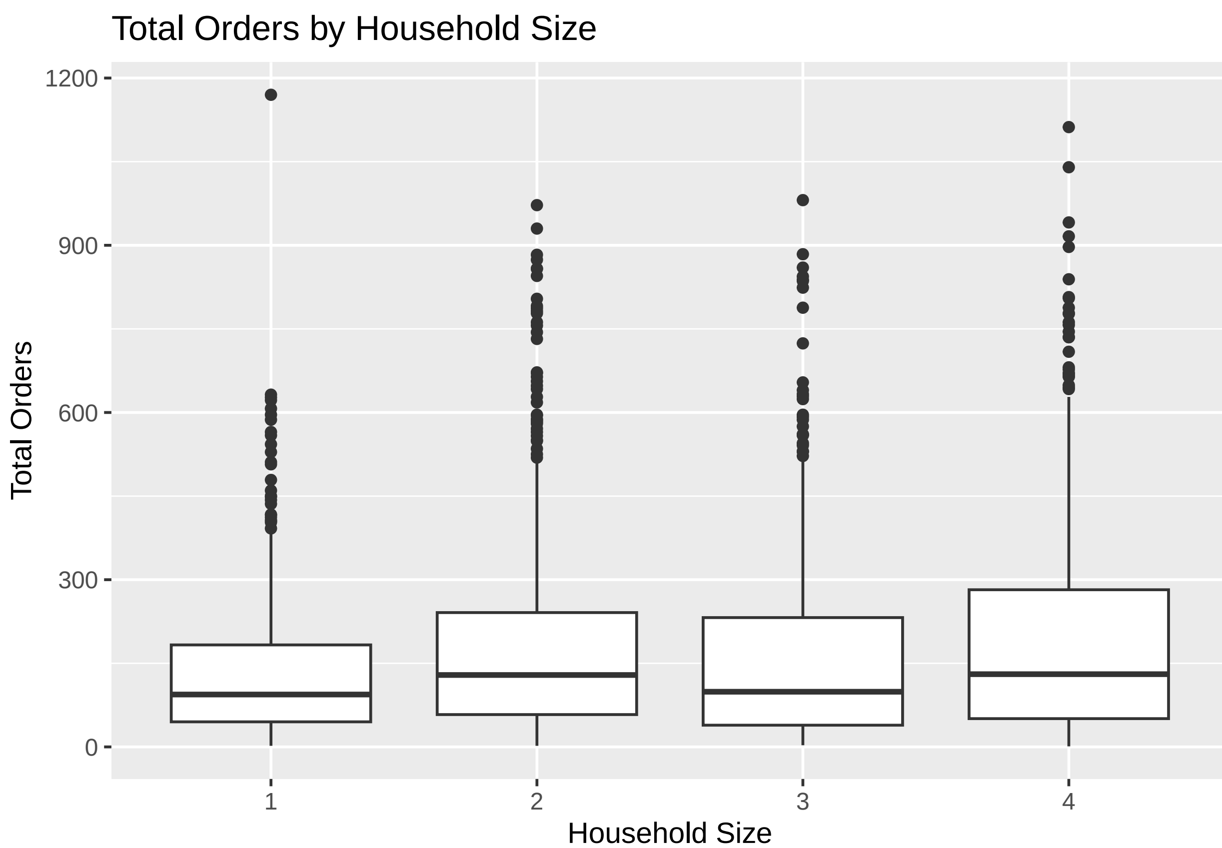

```r
#2nd plot - household size and total spend
train_df %>%
  ggplot(aes(x = factor(household_size_num), y = total_spend)) +
  geom_boxplot() +
  labs(title = "Total Spend by Household Size",
       x = "Household Size",
       y = "Total Spend")
```



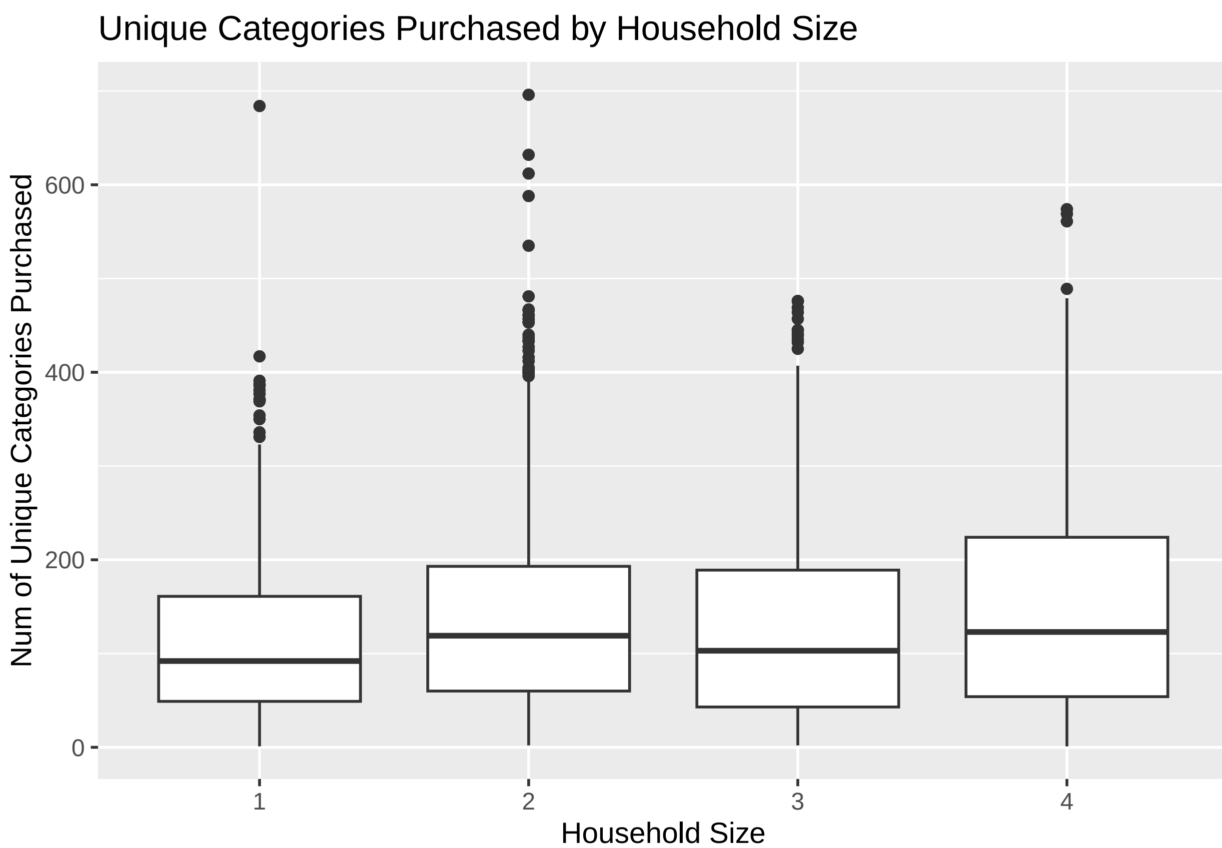

```r
#3rd plot - Average order value by household size
train_df %>%
  ggplot(aes(x = factor(household_size_num), y = avg_order_value)) +
  geom_boxplot() +
  labs(title = "Average Order Value by Household Size",
       x = "Household Size",
       y = "Average Order Value")
```



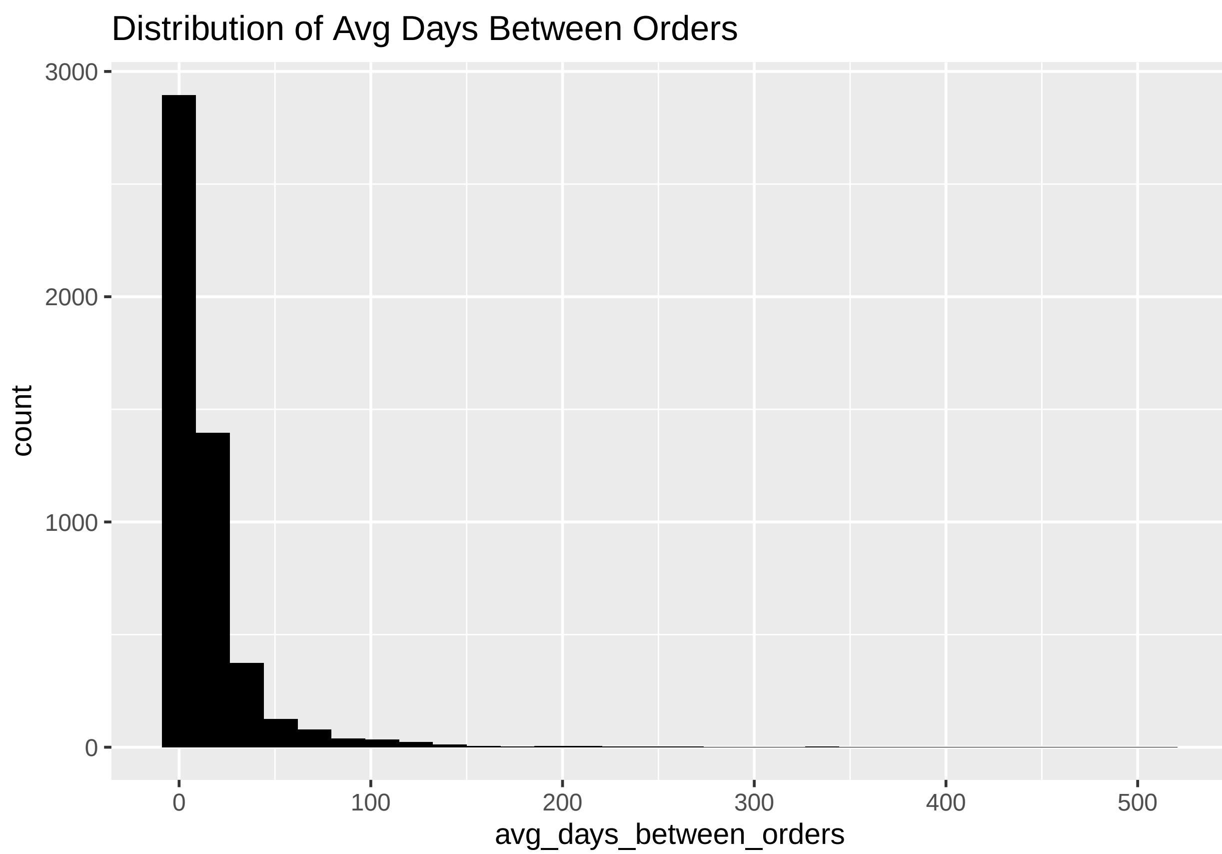

```
#4th plot - household size and total orders
train_df %>%
  ggplot(aes(x = factor(household_size_num), y = total_orders)) +
  geom_boxplot() +
  labs(title = "Total Orders by Household Size",
       x = "Household Size",
       y = "Total Orders")
```



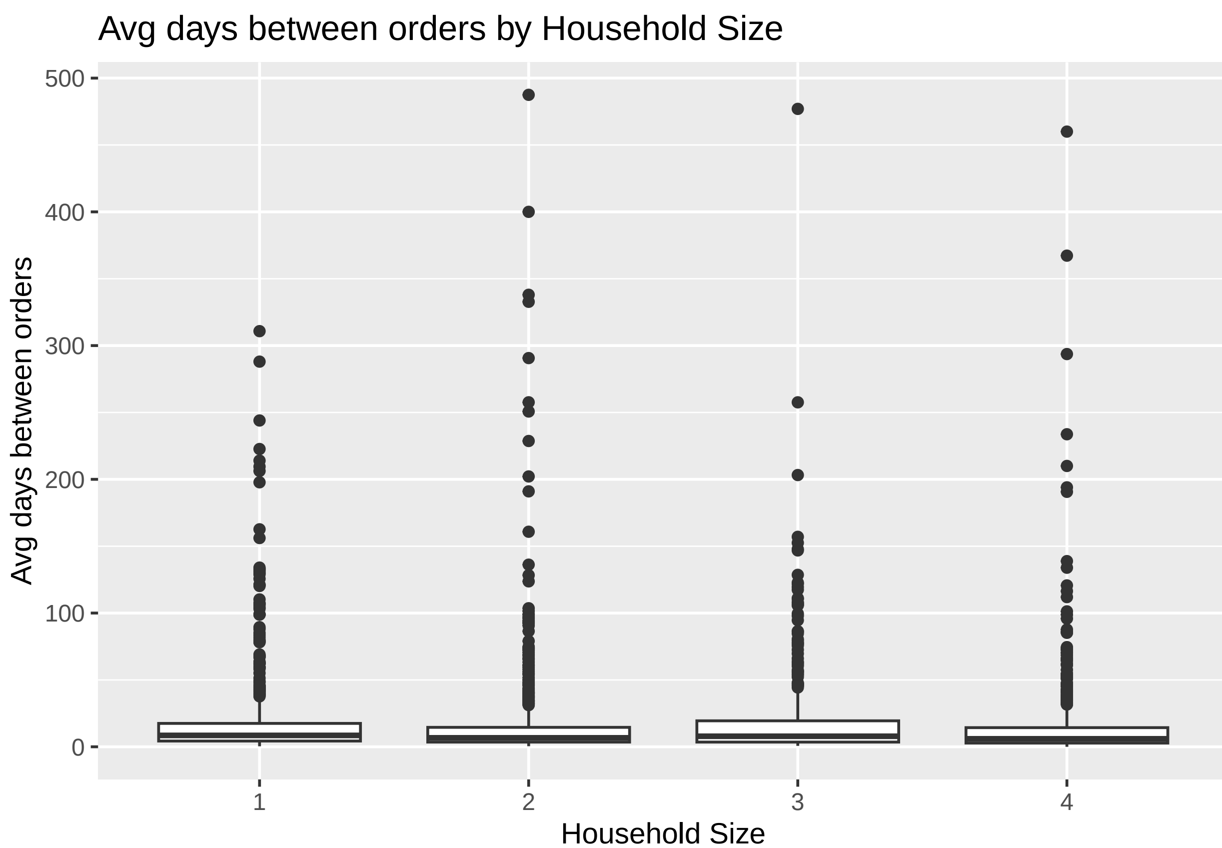

```r
#5th plot - household size and unique categories
train_df %>%
  ggplot(aes(x = factor(household_size_num), y = unique_categories)) +
  geom_boxplot() +
  labs(title = "Unique Categories Purchased by Household Size",
       x = "Household Size",
       y = "Num of Unique Categories Purchased")
```



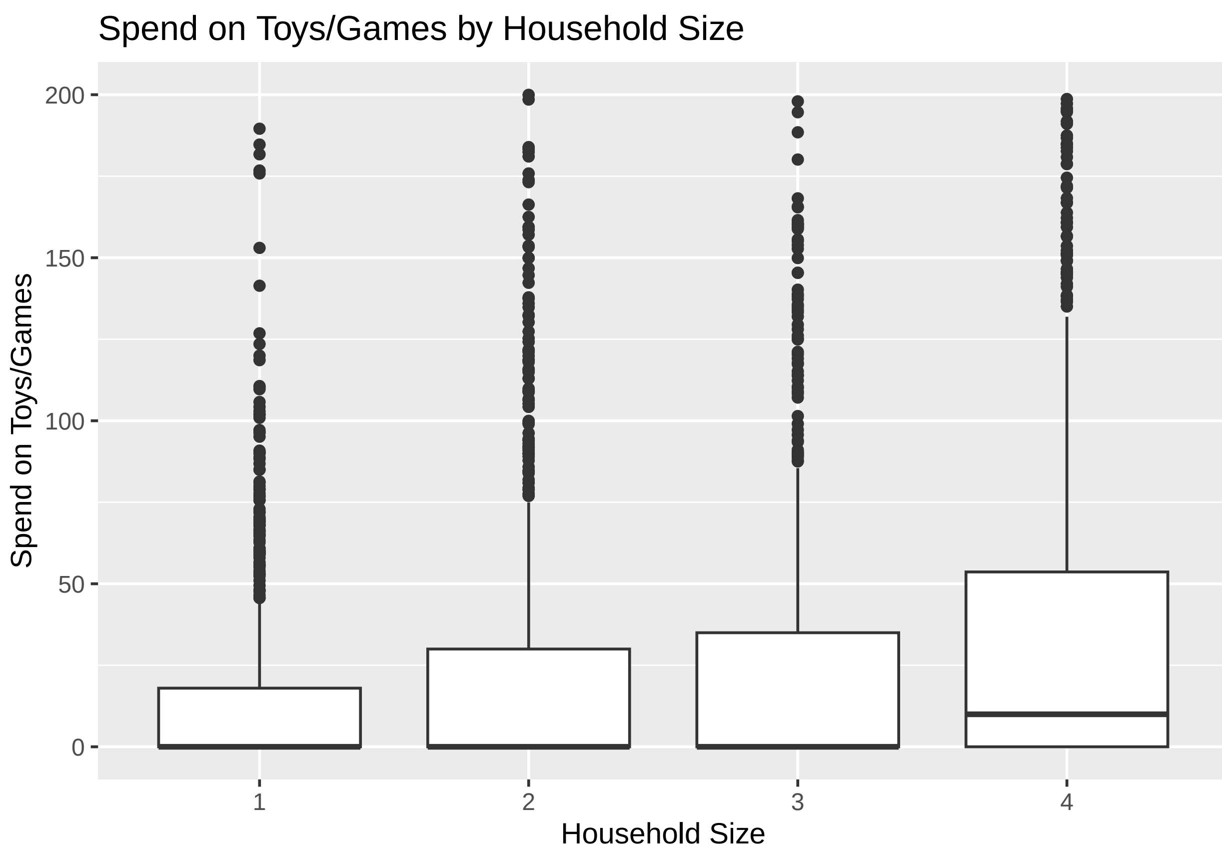

```r
#6th plot - household size and avg number of days between orders
ggplot(user_general_stats, aes(avg_days_between_orders)) +
  geom_histogram(bins = 30, fill = "black") +
  labs(title = "Distribution of Avg Days Between Orders")
```



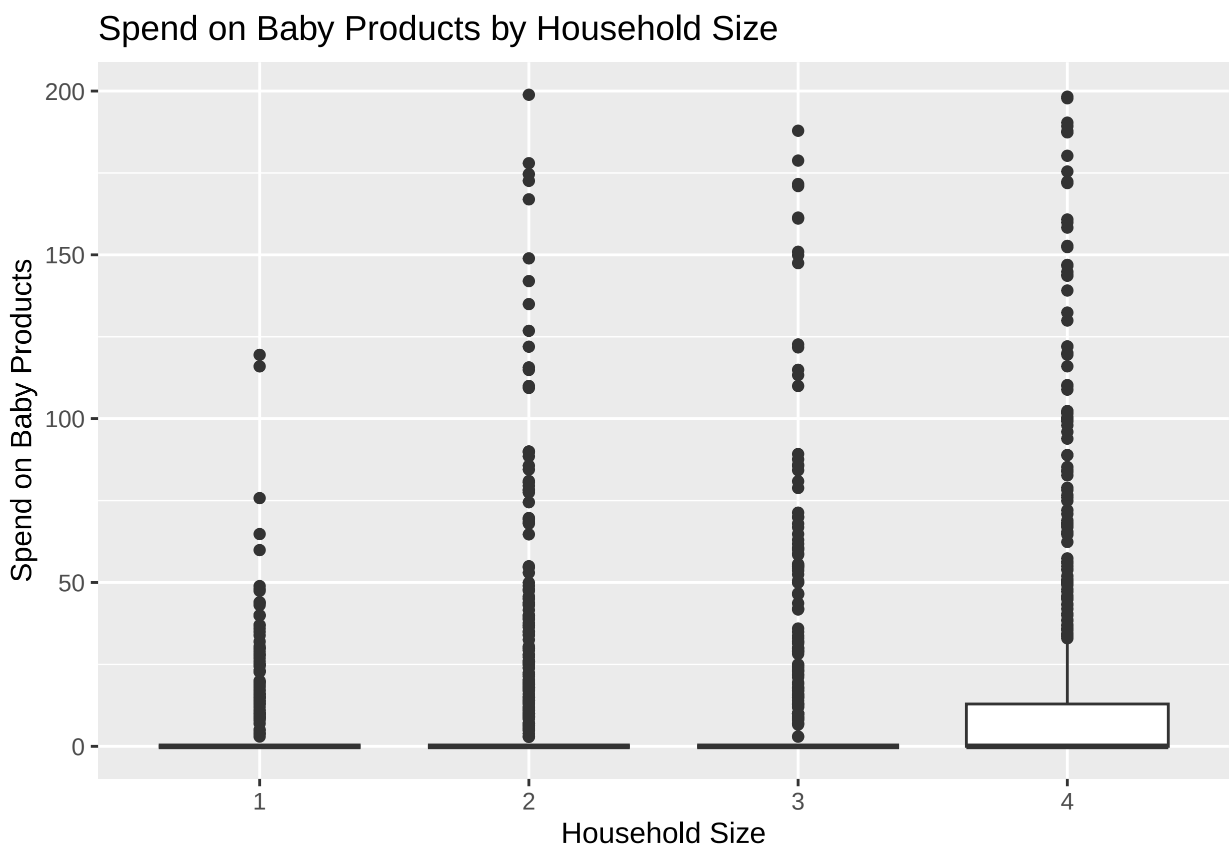

```r
#7th plot - household size and avg days between orders
train_df %>%
  ggplot(aes(x = factor(household_size_num), y = avg_days_between_orders)) +
  geom_boxplot() +
  labs(title = "Avg days between orders by Household Size",
       x = "Household Size",
       y = "Avg days between orders")
```




```
#8th plot - household size and toys/games spend
train_df %>%
  filter(spend_toys_and_games <= 200) %>% #to see the distribution easier
  ggplot(aes(x = factor(household_size_num), y = spend_toys_and_games)) +
  geom_boxplot() +
  labs(title = "Spend on Toys/Games by Household Size",
       x = "Household Size",
       y = "Spend on Toys/Games")
```




```r
#9th plot - household size and baby product spend
train_df %>%
  filter(spend_baby_product <= 200) %>% #to see the distribution easier
  ggplot(aes(x = factor(household_size_num), y = spend_baby_product)) +
  geom_boxplot() +
  labs(title = "Spend on Baby Products by Household Size",
       x = "Household Size",
       y = "Spend on Baby Products")
```



MODEL-PREPARATION

```
predictors <- c(
  "log_total_spend", "log_total_items", "log_avg_unit_price",
  "total_orders", "unique_categories",
  "avg_items_per_order", "avg_days_between_orders",

  "PC1", "PC2", "PC3", "PC4", "PC5",
  "PC6", "PC7", "PC8", "PC9", "PC10",

  "has_school_kid", "has_baby",
  "pct_grocery", "pct_toys",

  "items_toys_and_games",
  "items_costume_outfit",
  "items_toy_building_block",
  "items_backpack",
  "items_baby_product",
  "items_grocery",
  "items_paper_towel",
  "items_toilet_paper"
)
```

```r
# Ensure columns exist
valid_predictors <- predictors[predictors %in% names(train_df)]

# since household sizes of 1 and 4 don't appear as often, we make the dataset more balanced
train_df_balanced <- bind_rows(
  train_df,
  train_df %>% filter(household_size_num == 1),
  train_df %>% filter(household_size_num >= 4),
  train_df %>% filter(household_size_num >= 4),
  train_df %>% filter(household_size_num >= 4)
)

tsk_house <- as_task_regr(train_df_balanced[, c("household_size_num", valid_predictors)], target = "hous
```

MODEL-TRAIN: LINEAR REGRESSION

```r
# Linear Regression
lrn_lm <- lrn("regr.lm", id = "regr.lm")
```

MODEL-TRAIN: LASSO REGRESSION

```r
# Lasso Regression
lrn_lasso <- lrn("regr.cv_glmnet",
  id = "regr.lasso",
  alpha = 1,
  s = 0.00658601781237308, # from the analysis
  nfolds = 10,
  standardize = TRUE
)
```

MODEL-TRAIN: RIDGE REGRESSION

```r
# Ridge Regression
lrn_ridge <- lrn("regr.cv_glmnet",
  id = "regr.ridge",
  alpha = 0,
  s = "lambda.min",
  nfolds = 10,
  standardize = TRUE
)
```

MODEL-TRAIN: RANDOM FOREST

```r
# Random Forest
lrn_rf <- lrn("regr.ranger",
  importance = "impurity",
  num.trees = 100,
  mtry = 5,
  min.node.size = 10
)
```

MODEL-TRAIN: XGBOOST

```r
# XGBoost
# default parameters
lrn_xgb <- as_learner(po("encode", method = "one-hot") %>>% lrn("regr.xgboost"))
```

MODEL-TRAIN: XGBOOST TUNED

```r
#the tuning made the XGboost model even worse and had a very long run time
lrn_xgboost <- as_learner(
  po("encode", method = "one-hot") %>>%
    lrn("regr.xgboost",
        eta = to_tune(0.05, 0.15),
        max_depth = to_tune(3, 4),
        subsample = to_tune(0.8, 1)
    ))

at_xgb = auto_tuner(
  tuner = tnr("random_search"),
  term_evals = 50,
  learner = lrn_xgboost,
  resampling = rsmp("holdout"),
  measure = msr("regr.mse")
)
```

CROSS VALIDATION

```r
set.seed(10)
# learners for benchmarks
learners <- list(lrn_lm, lrn_lasso, lrn_ridge, lrn_rf, lrn_xgb)

# Cross Validation with 5-folds
rsmp_cv5 <- rsmp("cv", folds = 5)

# results
bmr_design <- benchmark_grid(
  tasks = tsk_house,
  learners = learners,
  resamplings = rsmp_cv5
)

set.seed(10)
bmr <- benchmark(bmr_design)
```

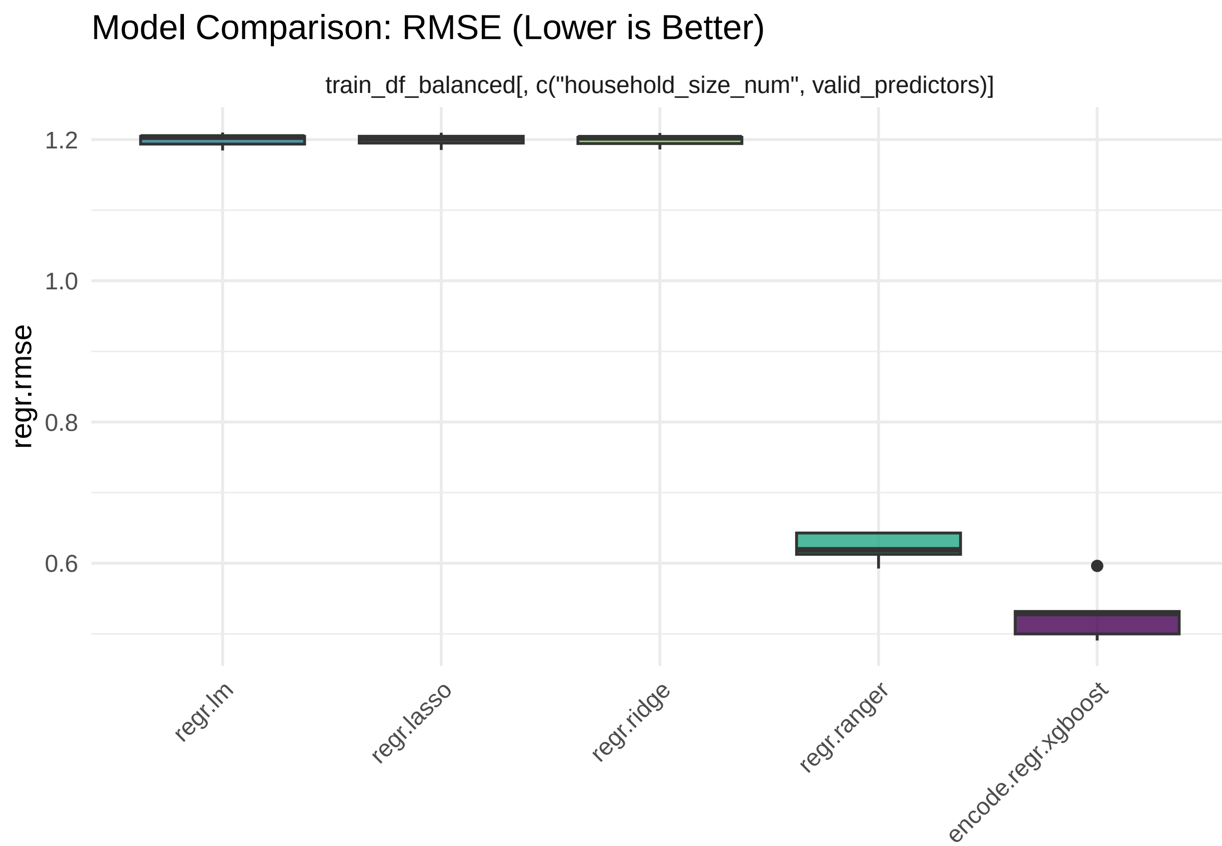
MODEL-ANALYSIS: COMPARISON

```r
# aggregates
results <- bmr$aggregate(measures = msrs(c("regr.rmse", "regr.rsq")))
print(results[, c("learner_id", "regr.rmse", "regr.rsq")])
```

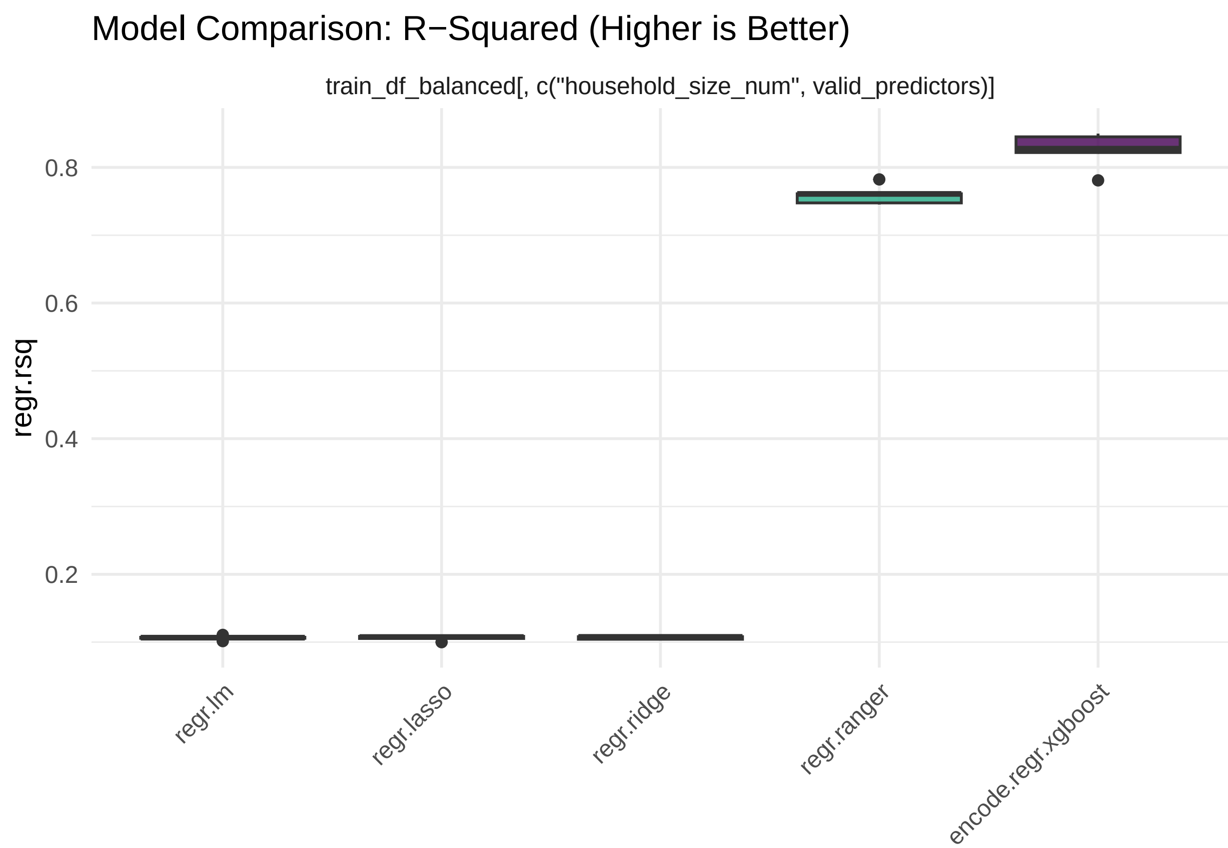
```
##              learner_id regr.rmse  regr.rsq
##                  <char>     <num>     <num>
## 1:              regr.lm 1.1992512 0.1062974
## 2:           regr.lasso 1.1992459 0.1062964
```

```
## 3:          regr.ridge 1.1991053 0.1065059
## 4:         regr.ranger 0.6219874 0.7594492
## 5: encode.regr.xgboost 0.5294093 0.8250556
```

```r
# Plot RMSE
autoplot(bmr, measure = msr("regr.rmse")) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Model Comparison: RMSE (Lower is Better)")
```

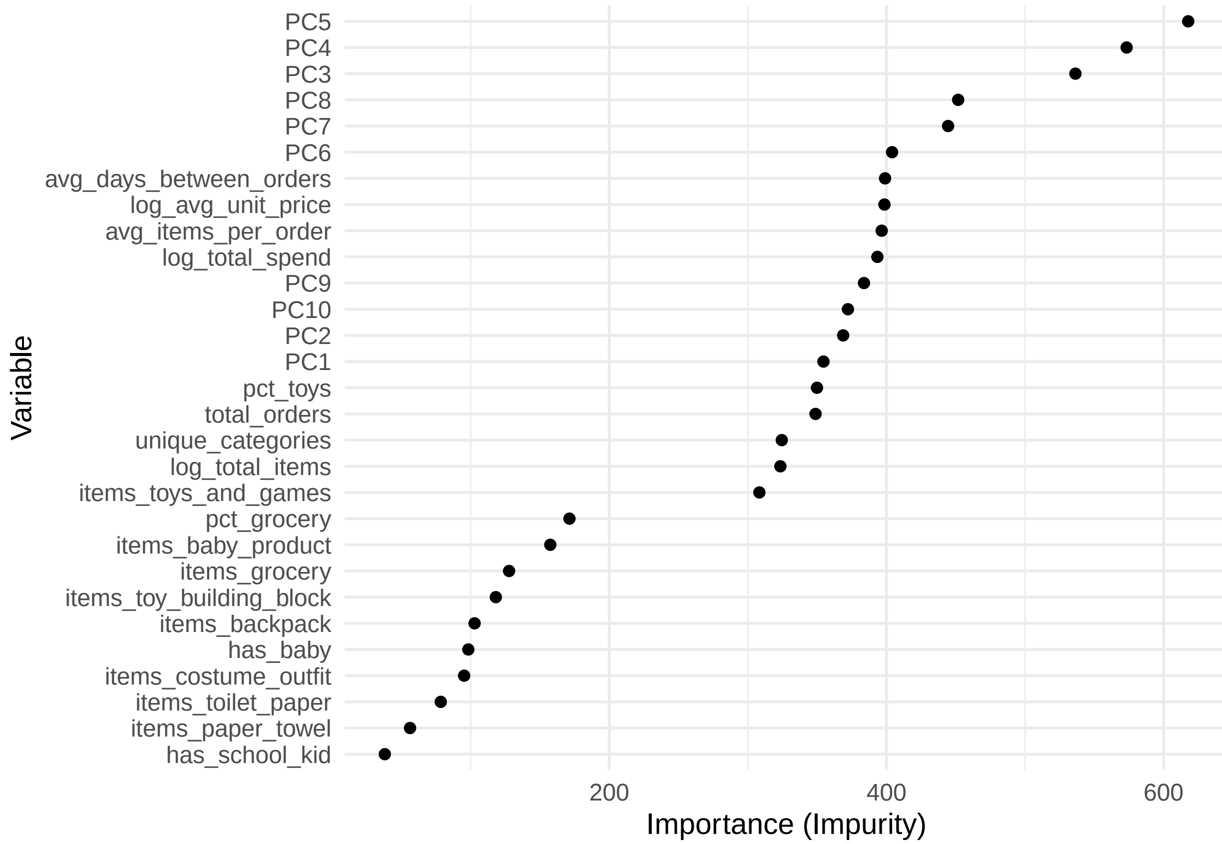



```r
# Plot R^2
autoplot(bmr, measure = msr("regr.rsq")) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Model Comparison: R-Squared (Higher is Better)")
```



MODEL-ANALYSIS: VARIABLE IMPORTANCE (RF)

```
# Train RF on the full task to get importance
lrn_rf$train(tsk_house)
importance_scores <- lrn_rf$model$model$variable.importance

# Plot VI
ggplot(data = data.frame(var = names(importance_scores), value = importance_scores),
       aes(x = value, y = reorder(var, value))) +
  geom_point() +
  ylab("Variable") + xlab("Importance (Impurity)") +
  theme_minimal()
```

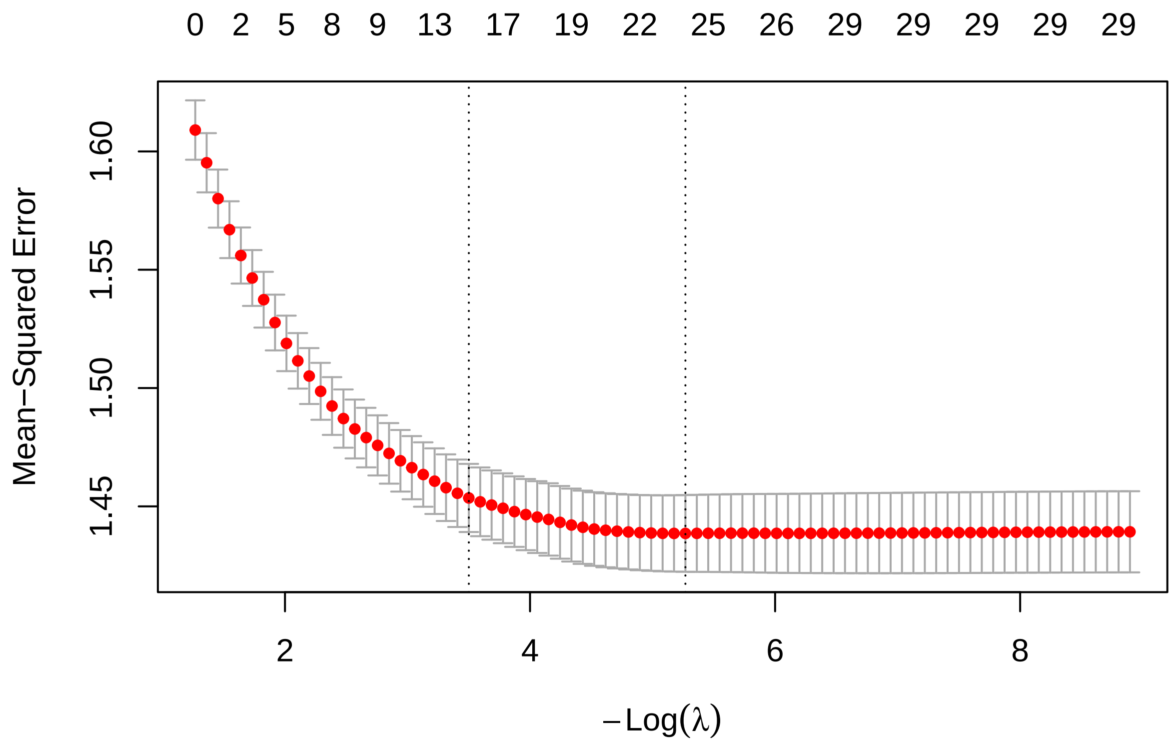


MODEL-ANALYSIS: LASSO LAMBDA SELECTION

```
# training lasso for lambda selection
lrn_lasso$train(tsk_house)
print(paste("Min Lambda:", lrn_lasso$model$lambda.min))
```

```
## [1] "Min Lambda: 0.00515343204288818"
```

```
plot(lrn_lasso$model)
```



EXPORT CSV

```
# Final Model (better accuracy with submission)
final_learner <- lrn_rf

# train on full balanced data
final_learner$train(tsk_house)

# prepare test data
test_data_subset <- test_df[, valid_predictors]

# predict
preds <- predict(final_learner, newdata = test_data_subset)

# create submission csv
submission <- data.frame(
  Survey.ResponseID = test_df$response_id,
  Q.amazon.use.hh.size.num = round(preds)
)

# handle NAs
submission$Q.amazon.use.hh.size.num <- pmin(pmax(submission$Q.amazon.use.hh.size.num, 1), 5)
submission$Q.amazon.use.hh.size.num[is.na(submission$Q.amazon.use.hh.size.num)] <- 2

write.csv(submission, "submission.csv", row.names = FALSE)
head(submission)
```

```
##    Survey.ResponseID Q.amazon.use.hh.size.num
## 1 R_2aP0GyIR66gSTiR                        3
## 2 R_11Y3ZtLQrhccXC7                        2
## 3 R_3papb8tA6MhY9Ga                        2
## 4 R_2PdHkMSzW7totCp                        3
## 5 R_1g0gCzG1DmCEKob                        3
## 6 R_279w6VtL6RMPoAy                        2
```