Project1

Chengpeng Dai

2025-06-13

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

Load common packages

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(lubridate)
library(naniar)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
library(sf)
```

```
## Linking to GEOS 3.13.0, GDAL 3.8.5, PROJ 9.5.1; sf_use_s2() is TRUE
```

```
library(tmap)
library(spdep)
```

```
## Loading required package: spData
## To access larger datasets in this package, install the spDataLarge
## package with: `install.packages('spDataLarge',
```

```
## repos='https://nowosad.github.io/drat/', type='source')`
```

```
library(lme4)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
library(feasts)
```

```
## Loading required package: fabletools
## Registered S3 method overwritten by 'tsibble':
##   method               from
##   as_tibble.grouped_df dplyr
##
## Attaching package: 'fabletools'
##
## The following object is masked from 'package:lme4':
##
##     refit
```

```
library(zoo)
```

```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(INLA)
```

```
## Loading required package: sp
## This is INLA_24.06.27 built 2024-06-27 03:00:51 UTC.
##  - See www.r-inla.org/contact-us for how to get help.
##  - List available models/likelihoods/etc with inla.list.models()
##  - Use inla.doc(<NAME>) to access documentation
```

```
library(car)
```

```
## Loading required package: carData
##
## Attaching package: 'car'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
```

```
library(glmnet)
```

```
## Loaded glmnet 4.1-9
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following objects are masked from 'package:fabletools':
##
##     MAE, RMSE
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(e1071)
```

```
##
## Attaching package: 'e1071'
##
## The following object is masked from 'package:fabletools':
##
##     interpolate
```

```
library(dplyr)
library(tibble)
library(rsample)
```

```
##
## Attaching package: 'rsample'
##
## The following object is masked from 'package:e1071':
##
##     permutations
```

```
library(rjags)
```

```
## Loading required package: coda
## Linked to JAGS 4.3.2
## Loaded modules: basemod,bugs
```

```
library(patchwork)
library(rnaturalearth)
library(rnaturalearthdata)
```

```
##
## Attaching package: 'rnaturalearthdata'
##
## The following object is masked from 'package:rnaturalearth':
##
##     countries110
```

```
library(ggplot2)
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(knitr)
```

EDA

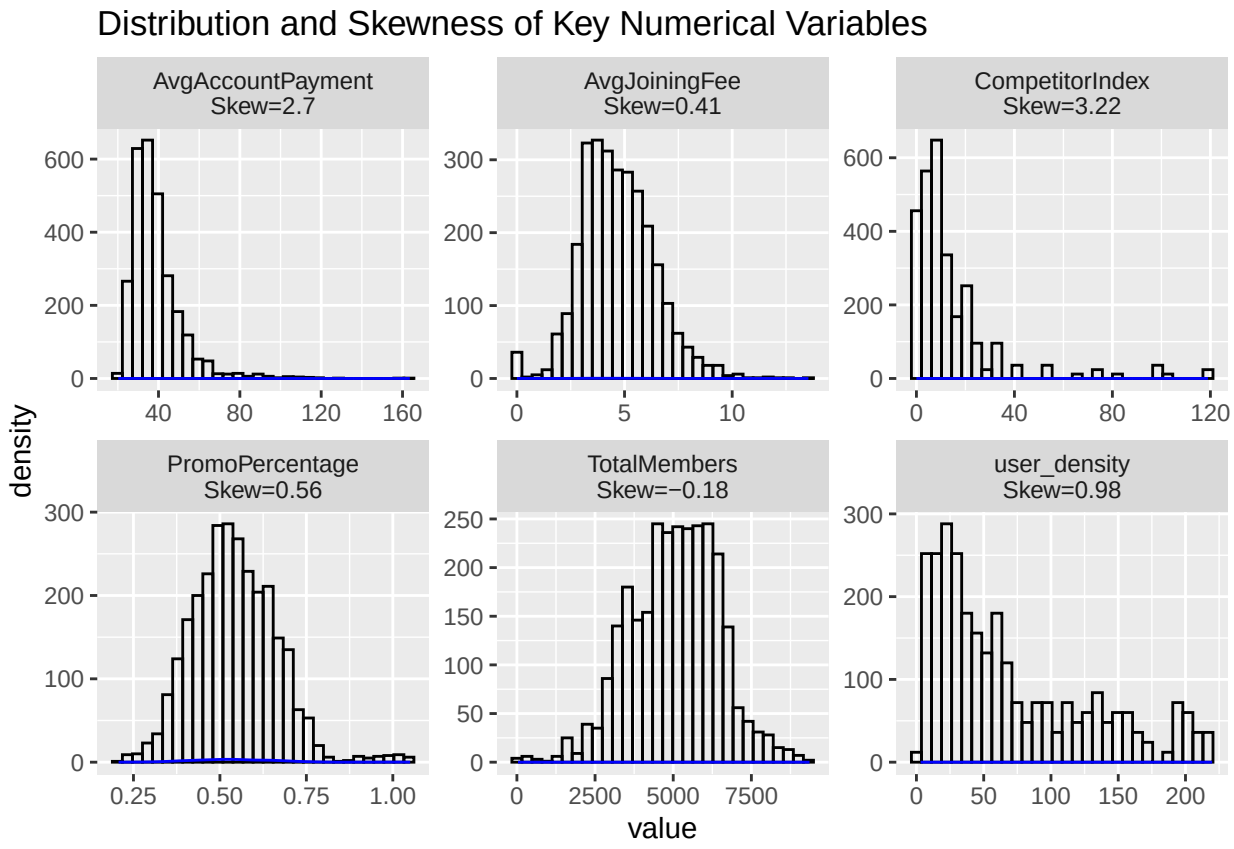
```
df <- read_csv("/Users/Devin/Library/CloudStorage/OneDrive-UniversityofEdinburgh/24Fall-3/Project_1/Uoe
```

```
## Rows: 2832 Columns: 52
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (4): StartMonth, GymSiteType, GymParking, HashedGymPublicName
## dbl (48): TotalMembers, GymUsableSqFt, GymMaxOccupancy, unis_within_0_0.5_mi...
##
## 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.
```

```
df$GymUsableSqFt <- NULL
df$GymMaxOccupancy <- NULL

df <- df %>%
  mutate(
    StartMonth = as.Date(as.yearmon(StartMonth, "%b-%y"))
  )
```

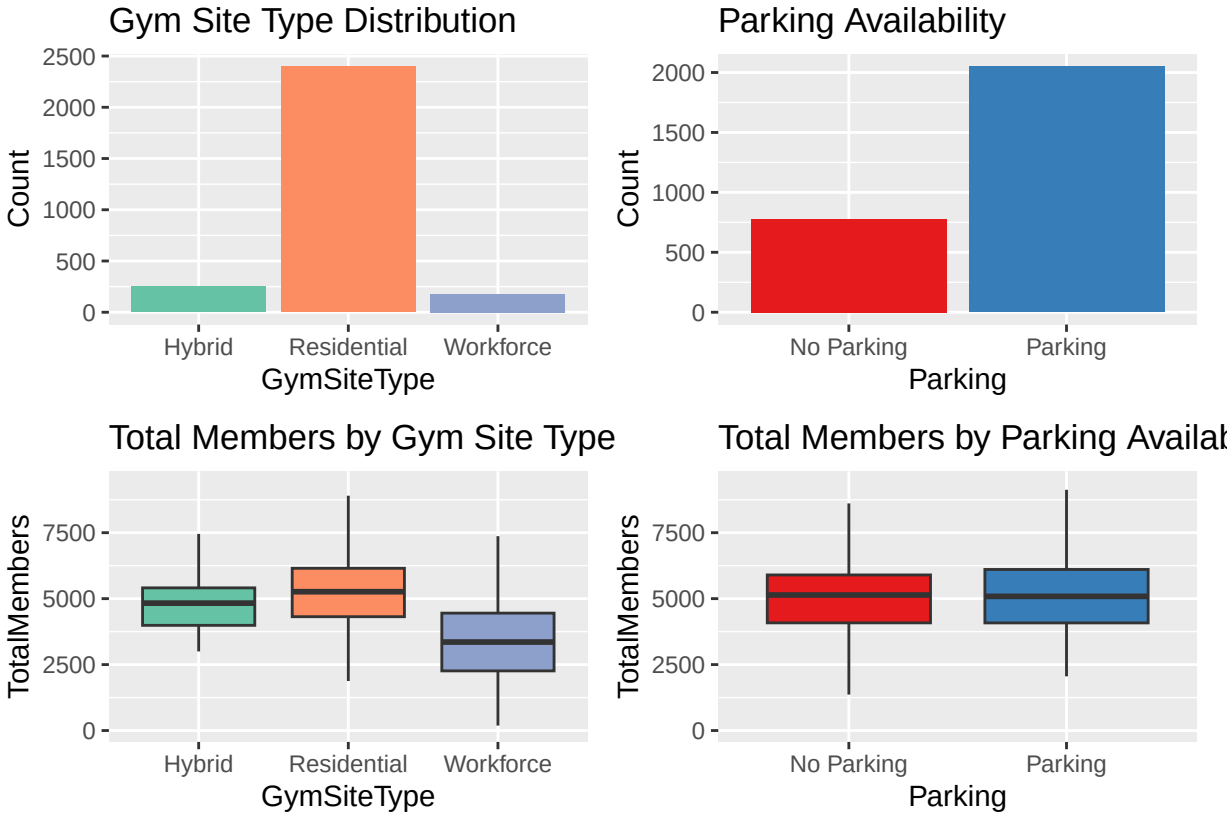
```
# Univariate exploration
# Distribution of numerical variables: Histogram + Density plot
key_numeric_vars <- c("TotalMembers",
                      "AvgAccountPayment",
                      "PromoPercentage",
                      "AvgJoiningFee",
                      "user_density",
                      "CompetitorIndex")

df %>%
  select(all_of(key_numeric_vars)) %>%
  mutate(across(everything(), as.numeric)) %>%
  pivot_longer(
    cols = everything(),
    names_to = "var",
    values_to = "value"
  ) %>%
  group_by(var) %>%
  mutate(skewness = e1071::skewness(value, na.rm = TRUE)) %>%
  ggplot(aes(x = value)) +
  geom_histogram(bins = 30, fill = NA, color = "black") +
  geom_density(color = "blue") +
```

```
  facet_wrap(~ paste0(var, '\nSkew=', round(skewness,2)), scales = "free", ncol = 3) +
  labs(title = "Distribution and Skewness of Key Numerical Variables")
```



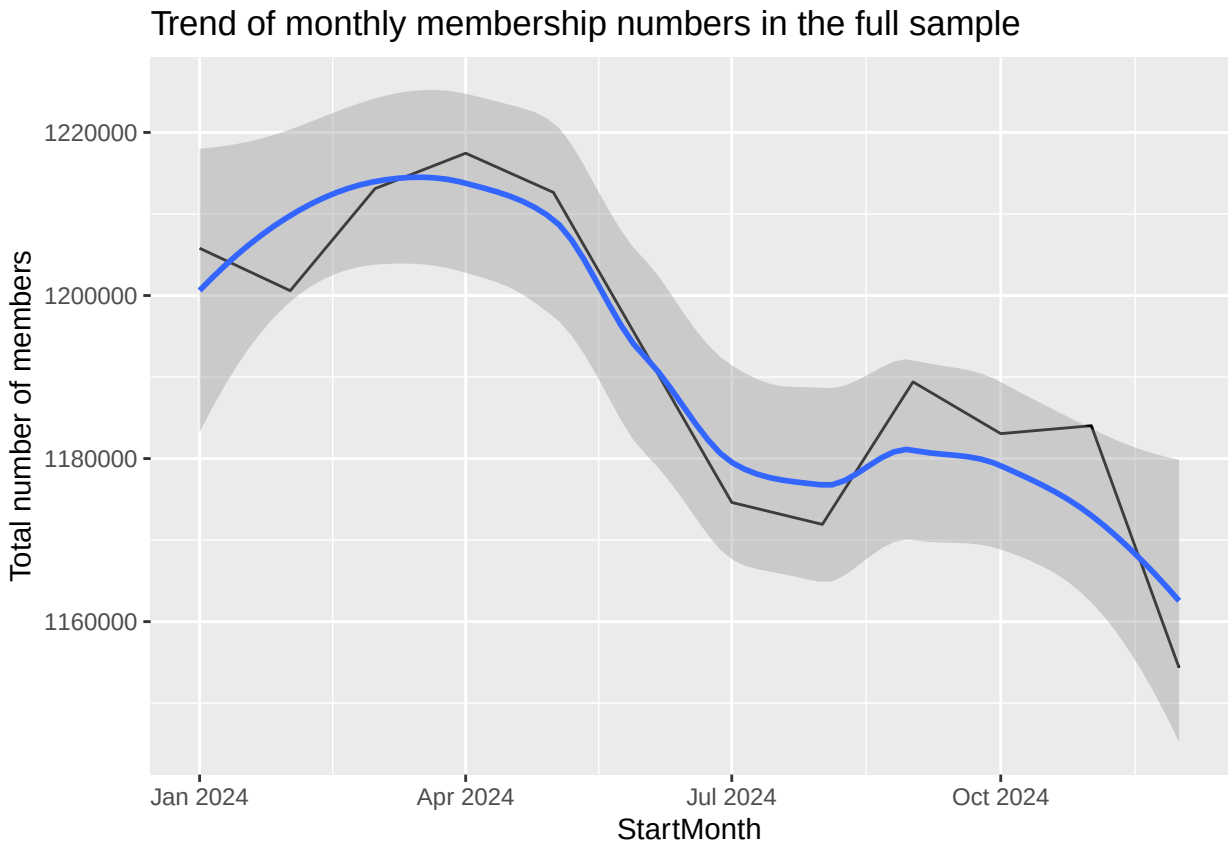

```
# Distribution of categorical variables
df$GymSiteType <- factor(df$GymSiteType, levels = c("Hybrid", "Residential", "Workforce"))
df$GymParking <- factor(df$GymParking, levels = c("No Parking", "Parking"))
# 1. GymSiteType
p1 <- df %>%
  count(GymSiteType) %>%
  ggplot(aes(x = n, y = GymSiteType, fill = GymSiteType)) +
  geom_col(show.legend = FALSE) +
  scale_fill_brewer(palette = "Set2") +
  coord_flip() +
  labs(title = "Gym Site Type Distribution", x = "Count", y = "GymSiteType")

# 2. GymParking
p2 <- df %>%
  count(GymParking) %>%
  ggplot(aes(x = GymParking, y = n, fill = GymParking)) +
  geom_col(show.legend = FALSE) +
  scale_fill_brewer(palette = "Set1") +
  labs(title = "Parking Availability", x = "Parking", y = "Count")

# 3. GymSiteType and TotalMembers
p3 <- ggplot(df, aes(x = GymSiteType, y = TotalMembers, fill = GymSiteType)) +
  geom_boxplot(outlier.shape = NA) +
  scale_fill_brewer(palette = "Set2") +
```

```
  labs(title = "Total Members by Gym Site Type", x = "GymSiteType", y = "TotalMembers") +
  theme(legend.position = "none")

# 4. GymParking  TotalMembers
p4 <- ggplot(df, aes(x = GymParking, y = TotalMembers, fill = GymParking)) +
  geom_boxplot(outlier.shape = NA) +
  scale_fill_brewer(palette = "Set1") +
  labs(title = "Total Members by Parking Availability", x = "Parking", y = "TotalMembers") +
  theme(legend.position = "none")

(p1 | p2) / (p3 | p4)
```



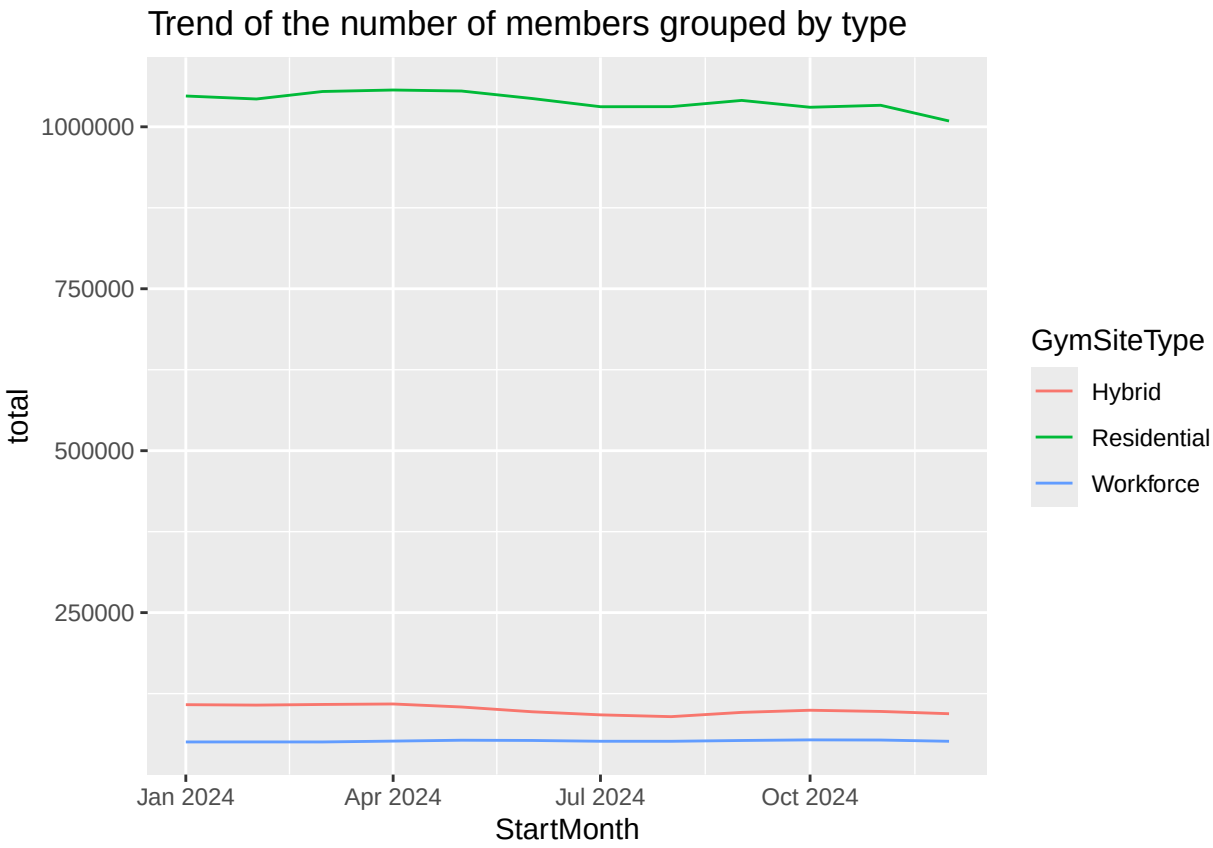

```
# Time series
df %>%
  group_by(StartMonth) %>%
  summarize(total_members = sum(TotalMembers, na.rm = TRUE)) %>%
  ggplot(aes(x = StartMonth, y = total_members)) +
    geom_line() +
    geom_smooth(method = "loess") +
    labs(title = "Trend of monthly membership numbers in the full sample",
         y = "Total number of members")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



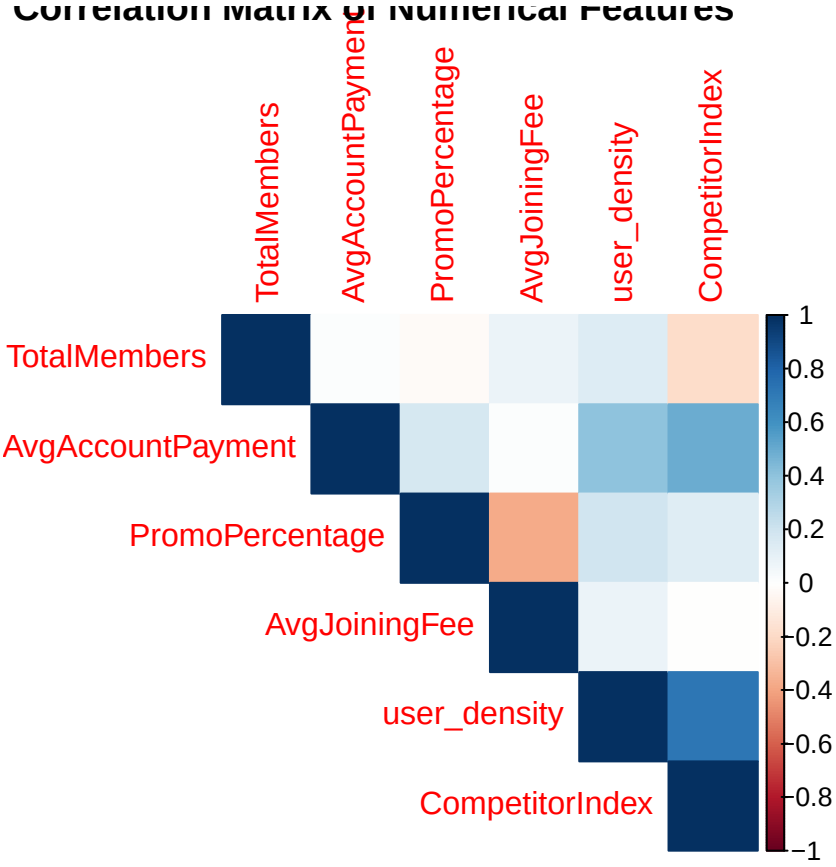

```
# Examples of categorical trend
df %>%
  group_by(StartMonth, GymSiteType) %>%
  summarize(total = sum(TotalMembers, na.rm = TRUE)) %>%
  ggplot(aes(x = StartMonth, y = total, color = GymSiteType)) +
    geom_line() +
    labs(title = "Trend of the number of members grouped by type")
```

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



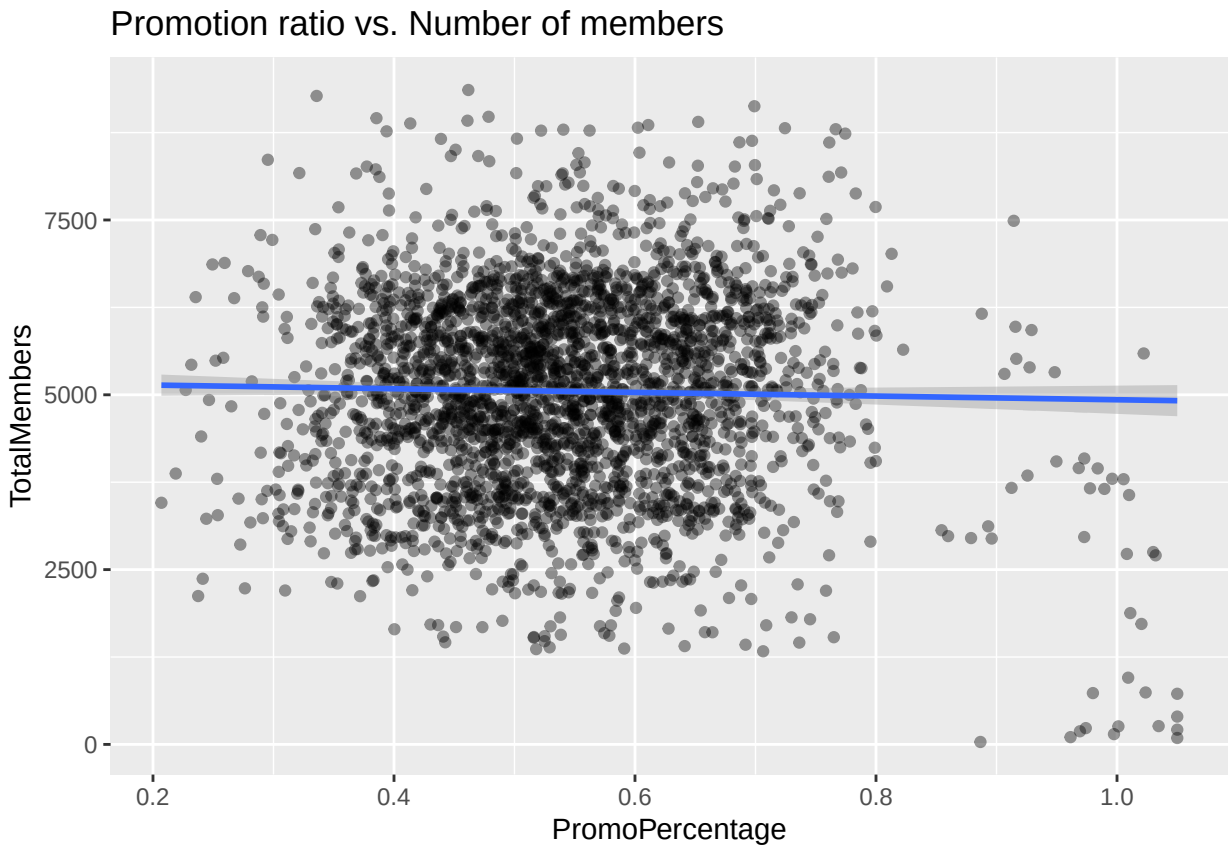

```
# # Spatial EDA
# gyms_sf <- df %>%
#   distinct(HashedGymPublicName, .keep_all = TRUE) %>%
#   st_as_sf(coords = c("Longitude", "Latitude"), crs = 4326)
# tmap_mode("view")
# tm_shape(gyms_sf) +
#   tm_dots(size = 0.1, col = "TotalMembers",
#           palette = "Blues", title = "Number of members")
#
# # Examples of categorical trend
# nb <- spdep::knearneigh(st_coordinates(gyms_sf), k = 8) %>% spdep::knn2nb()
# lw <- spdep::nb2listw(nb, style = "W")
# spdep::moran.test(gyms_sf$TotalMembers, lw)
```

```
# Bivariate
cor_df <- df %>% select(all_of(key_numeric_vars)) %>% na.omit()
corrplot(cor(cor_df), method = "color", type = "upper",
         title = "Correlation Matrix of Numerical Features")
```



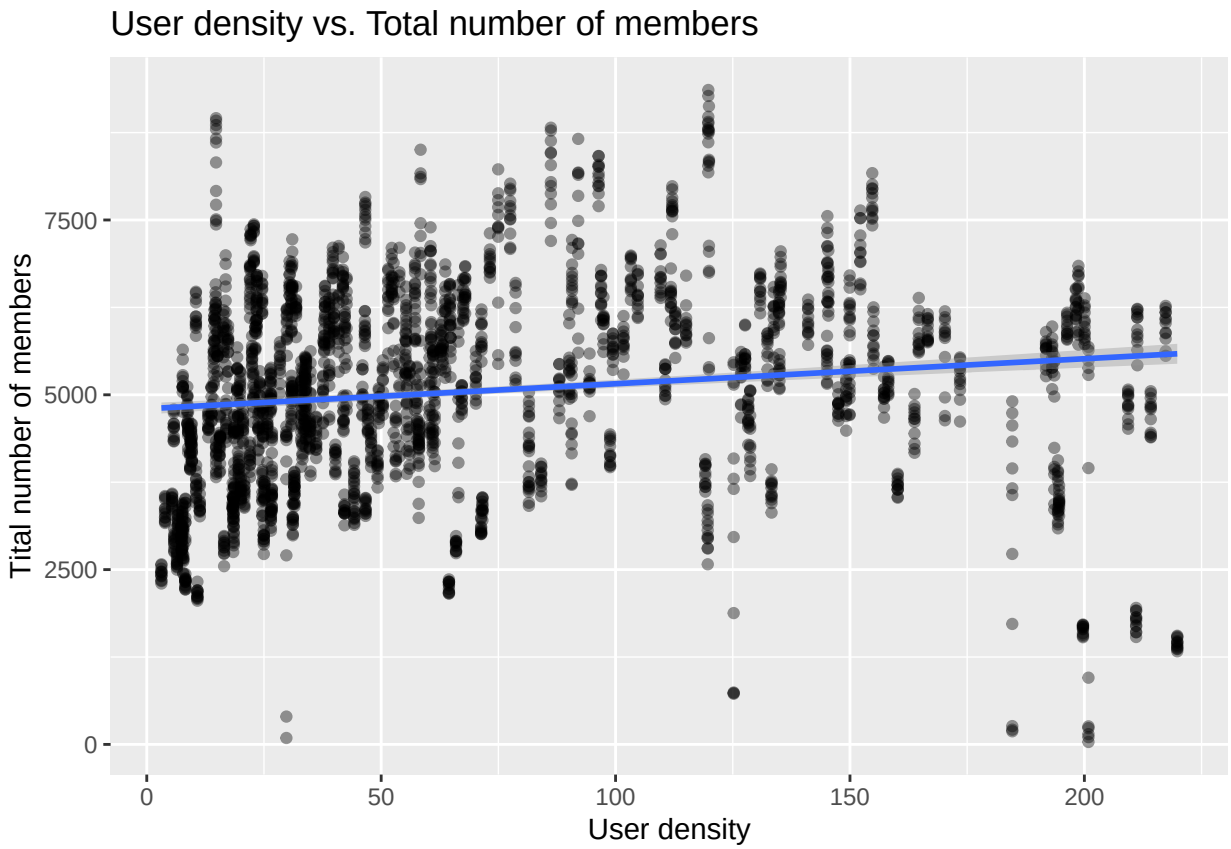

```
# Scatter points Total Members vs Promo Percentage
df %>%
  ggplot(aes(x = PromoPercentage, y = TotalMembers)) +
    geom_point(alpha = 0.4) +
    geom_smooth(method = "lm") +
    labs(title = "Promotion ratio vs. Number of members")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



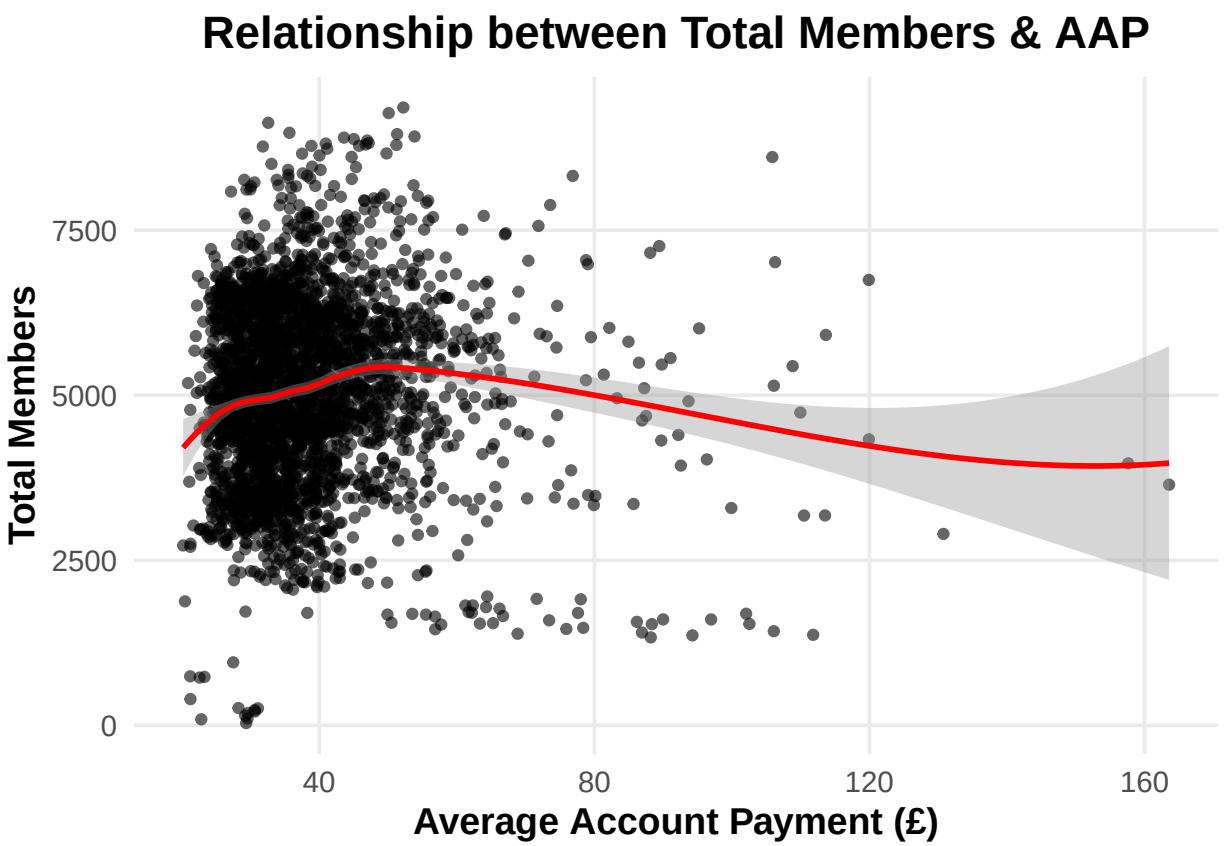

```
# User density vs. number of members
df %>%
  ggplot(aes(x = user_density, y = TotalMembers)) +
    geom_point(alpha = 0.4) + geom_smooth(method = "lm") +
    labs(title = "User density vs. Total number of members",
         x = "User density", y = "Tital number of members")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
ggplot(df, aes(x = AvgAccountPayment, y = TotalMembers)) +
  geom_point(alpha = 0.6, size = 1.5, color ="black") +
  geom_smooth(method = "loess", se = TRUE, color = "red", linetype = "solid") +
  labs(
    title = "Relationship between Total Members & AAP",
    x = "Average Account Payment (£)",
    y = "Total Members"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5),
    axis.title = element_text(face = "bold"),
    panel.grid.minor = element_blank()
  )
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
cor(df$AvgAccountPayment, df$TotalMembers, use = "complete.obs")
```

```
## [1] 0.01133169
```

Select variables

```
# Candidate variable definition

exclude_vars <- c("TotalMembers", "StartMonth", "HashedGymPublicName", "Longitude", "Latitude")
cand_vars <- setdiff(names(df), exclude_vars)
safe_vars <- paste0("`", cand_vars, "`")
formula_all <- reformulate(termlabels = safe_vars, response = "logTM")

# Preliminary OLS
# Fit an OLS model with all candidate variables
df_mod <- df %>% mutate(logTM = log(TotalMembers + 1))
ols1 <- lm(formula_all, data = df_mod)
print(summary(ols1))
```

```
##
## Call:
## lm(formula = formula_all, data = df_mod)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7389 -0.1399  0.0271  0.1726  0.8044
##
```

```
## Coefficients: (13 not defined because of singularities)
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              1.005e+01  3.591e-01  27.980  < 2e-16 ***
## GymSiteTypeResidential   5.788e-02  2.504e-02   2.311 0.020900 *
## GymSiteTypeWorkforce    -3.101e-02  3.896e-02  -0.796 0.426180
## GymParkingParking        1.273e-01  1.876e-02   6.783 1.43e-11 ***
## unis_within_0_0.5_mile   3.187e-02  5.757e-03   5.536 3.37e-08 ***
## unis_within_0.5_1_mile  -4.376e-03  3.985e-03  -1.098 0.272235
## unis_within_1_2_mile     2.587e-03  2.810e-03   0.920 0.357407
## PromoPercentage         -4.332e-01  5.444e-02  -7.957 2.53e-15 ***
## AvgJoiningFee            2.514e-03  3.956e-03   0.636 0.525146
## AvgAccountPayment        2.849e-03  5.877e-04   4.848 1.31e-06 ***
## BASEADL_0.5              1.489e-05  2.391e-06   6.229 5.41e-10 ***
## UMAP2D_1                -5.498e-03  2.125e-03  -2.587 0.009726 **
## UMAP2D_2                -2.504e-03  3.463e-03  -0.723 0.469729
## POP_0.5_1                4.367e-06  1.299e-06   3.361 0.000786 ***
## POP_1_2                 -2.190e-06  4.550e-07  -4.812 1.57e-06 ***
## POP_2_3                  4.416e-07  3.102e-07   1.424 0.154649
## POP_3_4                  2.473e-07  2.335e-07   1.059 0.289673
## DENS_0_0.5                      NA         NA      NA       NA
## DENS_0.5_1                      NA         NA      NA       NA
## DENS_1_2                        NA         NA      NA       NA
## DENS_2_3                        NA         NA      NA       NA
## DENS_3_4                        NA         NA      NA       NA
## DENSITY_DROP_2                  NA         NA      NA       NA
## DENSITY_DROP_3                  NA         NA      NA       NA
## DENSITY_DROP_4                  NA         NA      NA       NA
## NEAR_POP_SHARE           8.368e-01  8.109e-01   1.032 0.302163
## INNER_RING_SHARE        -7.077e-01  2.867e-01  -2.468 0.013639 *
## RATIO_0_5_TO_1          -3.472e-01  9.426e-02  -3.683 0.000235 ***
## RATIO_1_TO_2            -1.903e-01  7.945e-02  -2.395 0.016685 *
## `0-0.5_mile_comp`       -5.428e-02  9.640e-03  -5.631 1.97e-08 ***
## `0.5-1_mile_comp`       -2.741e-02  7.456e-03  -3.676 0.000241 ***
## `1-2_mile_comp`         -3.072e-02  6.422e-03  -4.784 1.80e-06 ***
## CompetitorIndex          2.667e-03  2.392e-03   1.115 0.264922
## transaction_density_4    2.478e-04  2.559e-04   0.968 0.332953
## user_density            -6.034e-03  1.574e-03  -3.834 0.000129 ***
## avg_spend_per_user      -1.335e-03  3.297e-04  -4.047 5.32e-05 ***
## user_repeat_rate        -2.133e+00  3.283e-01  -6.496 9.74e-11 ***
## distance_weighted_spend  9.935e-03  2.530e-03   3.927 8.79e-05 ***
## avg_merchant_per_user    2.581e-01  1.595e-01   1.618 0.105759
## txn_count_0_1_mi         2.754e-05  5.992e-06   4.597 4.47e-06 ***
## txn_density_0_1_mi              NA         NA      NA       NA
## txn_count_1_2_mi         2.800e-05  4.830e-06   5.798 7.48e-09 ***
## txn_density_1_2_mi              NA         NA      NA       NA
## txn_count_2_3_mi         2.273e-05  4.484e-06   5.069 4.26e-07 ***
## txn_density_2_3_mi              NA         NA      NA       NA
## txn_count_3_4_mi                NA         NA      NA       NA
## txn_density_3_4_mi              NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3043 on 2798 degrees of freedom
## Multiple R-squared:  0.3382, Adjusted R-squared:  0.3304
```

```
## F-statistic: 43.34 on 33 and 2798 DF,  p-value: < 2.2e-16
```

```
# Check for variables that are completely collinear
aliases <- alias(ols1)
if(length(aliases$Complete) > 0) {
  cat("Completely collinear variables (linear combinations) detected:", paste(names(aliases$Complete), c
  cat("Please use alias(ols1) to view the specific linear relationship, and retry after manually removi
} else {
  # VIF calculation, capturing aliasing coefficient errors
  vif_vals <- tryCatch(car::vif(ols1), error = function(e) {
    message("Unable to calculate VIF: The model has aliased coefficients. Please use alias(ols1) to vie
    NULL
  })
  print(vif_vals)
}
```

```
## Completely collinear variables (linear combinations) detected:  Please use alias(ols1) to view the sp
```

```
# stepwise regression guided by BIC
step_bic <- step(ols1, direction = "both", k = log(nrow(df)))
```

```
## Start:  AIC=-6502.84
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
##     unis_within_1_2_mile + PromoPercentage + AvgJoiningFee +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + UMAP2D_2 + POP_0.5_1 +
##     POP_1_2 + POP_2_3 + POP_3_4 + DENS_0_0.5 + DENS_0.5_1 + DENS_1_2 +
##     DENS_2_3 + DENS_3_4 + DENSITY_DROP_2 + DENSITY_DROP_3 + DENSITY_DROP_4 +
##     NEAR_POP_SHARE + INNER_RING_SHARE + RATIO_0_5_TO_1 + RATIO_1_TO_2 +
##     `0-0.5_mile_comp` + `0.5-1_mile_comp` + `1-2_mile_comp` +
##     CompetitorIndex + transaction_density_4 + user_density +
##     avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_density_0_1_mi +
##     txn_count_1_2_mi + txn_density_1_2_mi + txn_count_2_3_mi +
##     txn_density_2_3_mi + txn_count_3_4_mi + txn_density_3_4_mi
##
##
## Step:  AIC=-6502.84
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
##     unis_within_1_2_mile + PromoPercentage + AvgJoiningFee +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + UMAP2D_2 + POP_0.5_1 +
##     POP_1_2 + POP_2_3 + POP_3_4 + DENS_0_0.5 + DENS_0.5_1 + DENS_1_2 +
##     DENS_2_3 + DENS_3_4 + DENSITY_DROP_2 + DENSITY_DROP_3 + DENSITY_DROP_4 +
##     NEAR_POP_SHARE + INNER_RING_SHARE + RATIO_0_5_TO_1 + RATIO_1_TO_2 +
##     `0-0.5_mile_comp` + `0.5-1_mile_comp` + `1-2_mile_comp` +
##     CompetitorIndex + transaction_density_4 + user_density +
##     avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_density_0_1_mi +
##     txn_count_1_2_mi + txn_density_1_2_mi + txn_count_2_3_mi +
##     txn_density_2_3_mi + txn_count_3_4_mi
##
##
## Step:  AIC=-6502.84
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
##     unis_within_1_2_mile + PromoPercentage + AvgJoiningFee +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + UMAP2D_2 + POP_0.5_1 +
##     POP_1_2 + POP_2_3 + POP_3_4 + DENS_0_0.5 + DENS_0.5_1 + DENS_1_2 +
```

```
##     DENS_2_3 + DENS_3_4 + DENSITY_DROP_2 + DENSITY_DROP_3 + DENSITY_DROP_4 +
##     NEAR_POP_SHARE + INNER_RING_SHARE + RATIO_0_5_TO_1 + RATIO_1_TO_2 +
##     `0-0.5_mile_comp` + `0.5-1_mile_comp` + `1-2_mile_comp` +
##     CompetitorIndex + transaction_density_4 + user_density +
##     avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_density_0_1_mi +
##     txn_count_1_2_mi + txn_density_1_2_mi + txn_count_2_3_mi +
##     txn_density_2_3_mi
##
##
## Step:  AIC=-6502.84
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
##     unis_within_1_2_mile + PromoPercentage + AvgJoiningFee +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + UMAP2D_2 + POP_0.5_1 +
##     POP_1_2 + POP_2_3 + POP_3_4 + DENS_0_0.5 + DENS_0.5_1 + DENS_1_2 +
##     DENS_2_3 + DENS_3_4 + DENSITY_DROP_2 + DENSITY_DROP_3 + DENSITY_DROP_4 +
##     NEAR_POP_SHARE + INNER_RING_SHARE + RATIO_0_5_TO_1 + RATIO_1_TO_2 +
##     `0-0.5_mile_comp` + `0.5-1_mile_comp` + `1-2_mile_comp` +
##     CompetitorIndex + transaction_density_4 + user_density +
##     avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_density_0_1_mi +
##     txn_count_1_2_mi + txn_density_1_2_mi + txn_count_2_3_mi
##
##
## Step:  AIC=-6502.84
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
##     unis_within_1_2_mile + PromoPercentage + AvgJoiningFee +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + UMAP2D_2 + POP_0.5_1 +
##     POP_1_2 + POP_2_3 + POP_3_4 + DENS_0_0.5 + DENS_0.5_1 + DENS_1_2 +
##     DENS_2_3 + DENS_3_4 + DENSITY_DROP_2 + DENSITY_DROP_3 + DENSITY_DROP_4 +
##     NEAR_POP_SHARE + INNER_RING_SHARE + RATIO_0_5_TO_1 + RATIO_1_TO_2 +
##     `0-0.5_mile_comp` + `0.5-1_mile_comp` + `1-2_mile_comp` +
##     CompetitorIndex + transaction_density_4 + user_density +
##     avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_density_0_1_mi +
##     txn_count_1_2_mi + txn_count_2_3_mi
##
##
## Step:  AIC=-6502.84
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
##     unis_within_1_2_mile + PromoPercentage + AvgJoiningFee +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + UMAP2D_2 + POP_0.5_1 +
##     POP_1_2 + POP_2_3 + POP_3_4 + DENS_0_0.5 + DENS_0.5_1 + DENS_1_2 +
##     DENS_2_3 + DENS_3_4 + DENSITY_DROP_2 + DENSITY_DROP_3 + DENSITY_DROP_4 +
##     NEAR_POP_SHARE + INNER_RING_SHARE + RATIO_0_5_TO_1 + RATIO_1_TO_2 +
##     `0-0.5_mile_comp` + `0.5-1_mile_comp` + `1-2_mile_comp` +
##     CompetitorIndex + transaction_density_4 + user_density +
##     avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_count_1_2_mi +
##     txn_count_2_3_mi
##
##
## Step:  AIC=-6502.84
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
```

```
##     unis_within_1_2_mile + PromoPercentage + AvgJoiningFee +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + UMAP2D_2 + POP_0.5_1 +
##     POP_1_2 + POP_2_3 + POP_3_4 + DENS_0_0.5 + DENS_0.5_1 + DENS_1_2 +
##     DENS_2_3 + DENS_3_4 + DENSITY_DROP_2 + DENSITY_DROP_3 + NEAR_POP_SHARE +
##     INNER_RING_SHARE + RATIO_0_5_TO_1 + RATIO_1_TO_2 + `0-0.5_mile_comp` +
##     `0.5-1_mile_comp` + `1-2_mile_comp` + CompetitorIndex + transaction_density_4 +
##     user_density + avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_count_1_2_mi +
##     txn_count_2_3_mi
##
##
## Step:  AIC=-6502.84
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
##     unis_within_1_2_mile + PromoPercentage + AvgJoiningFee +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + UMAP2D_2 + POP_0.5_1 +
##     POP_1_2 + POP_2_3 + POP_3_4 + DENS_0_0.5 + DENS_0.5_1 + DENS_1_2 +
##     DENS_2_3 + DENS_3_4 + DENSITY_DROP_2 + NEAR_POP_SHARE + INNER_RING_SHARE +
##     RATIO_0_5_TO_1 + RATIO_1_TO_2 + `0-0.5_mile_comp` + `0.5-1_mile_comp` +
##     `1-2_mile_comp` + CompetitorIndex + transaction_density_4 +
##     user_density + avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_count_1_2_mi +
##     txn_count_2_3_mi
##
##
## Step:  AIC=-6502.84
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
##     unis_within_1_2_mile + PromoPercentage + AvgJoiningFee +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + UMAP2D_2 + POP_0.5_1 +
##     POP_1_2 + POP_2_3 + POP_3_4 + DENS_0_0.5 + DENS_0.5_1 + DENS_1_2 +
##     DENS_2_3 + DENS_3_4 + NEAR_POP_SHARE + INNER_RING_SHARE +
##     RATIO_0_5_TO_1 + RATIO_1_TO_2 + `0-0.5_mile_comp` + `0.5-1_mile_comp` +
##     `1-2_mile_comp` + CompetitorIndex + transaction_density_4 +
##     user_density + avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_count_1_2_mi +
##     txn_count_2_3_mi
##
##
## Step:  AIC=-6502.84
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
##     unis_within_1_2_mile + PromoPercentage + AvgJoiningFee +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + UMAP2D_2 + POP_0.5_1 +
##     POP_1_2 + POP_2_3 + POP_3_4 + DENS_0_0.5 + DENS_0.5_1 + DENS_1_2 +
##     DENS_2_3 + NEAR_POP_SHARE + INNER_RING_SHARE + RATIO_0_5_TO_1 +
##     RATIO_1_TO_2 + `0-0.5_mile_comp` + `0.5-1_mile_comp` + `1-2_mile_comp` +
##     CompetitorIndex + transaction_density_4 + user_density +
##     avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_count_1_2_mi +
##     txn_count_2_3_mi
##
##
## Step:  AIC=-6502.84
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
##     unis_within_1_2_mile + PromoPercentage + AvgJoiningFee +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + UMAP2D_2 + POP_0.5_1 +
```

```
##     POP_1_2 + POP_2_3 + POP_3_4 + DENS_0_0.5 + DENS_0.5_1 + DENS_1_2 +
##     NEAR_POP_SHARE + INNER_RING_SHARE + RATIO_0_5_TO_1 + RATIO_1_TO_2 +
##     `0-0.5_mile_comp` + `0.5-1_mile_comp` + `1-2_mile_comp` +
##     CompetitorIndex + transaction_density_4 + user_density +
##     avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_count_1_2_mi +
##     txn_count_2_3_mi
##
##
## Step:  AIC=-6502.84
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
##     unis_within_1_2_mile + PromoPercentage + AvgJoiningFee +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + UMAP2D_2 + POP_0.5_1 +
##     POP_1_2 + POP_2_3 + POP_3_4 + DENS_0_0.5 + DENS_0.5_1 + NEAR_POP_SHARE +
##     INNER_RING_SHARE + RATIO_0_5_TO_1 + RATIO_1_TO_2 + `0-0.5_mile_comp` +
##     `0.5-1_mile_comp` + `1-2_mile_comp` + CompetitorIndex + transaction_density_4 +
##     user_density + avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_count_1_2_mi +
##     txn_count_2_3_mi
##
##
## Step:  AIC=-6502.84
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
##     unis_within_1_2_mile + PromoPercentage + AvgJoiningFee +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + UMAP2D_2 + POP_0.5_1 +
##     POP_1_2 + POP_2_3 + POP_3_4 + DENS_0_0.5 + NEAR_POP_SHARE +
##     INNER_RING_SHARE + RATIO_0_5_TO_1 + RATIO_1_TO_2 + `0-0.5_mile_comp` +
##     `0.5-1_mile_comp` + `1-2_mile_comp` + CompetitorIndex + transaction_density_4 +
##     user_density + avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_count_1_2_mi +
##     txn_count_2_3_mi
##
##
## Step:  AIC=-6502.84
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
##     unis_within_1_2_mile + PromoPercentage + AvgJoiningFee +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + UMAP2D_2 + POP_0.5_1 +
##     POP_1_2 + POP_2_3 + POP_3_4 + NEAR_POP_SHARE + INNER_RING_SHARE +
##     RATIO_0_5_TO_1 + RATIO_1_TO_2 + `0-0.5_mile_comp` + `0.5-1_mile_comp` +
##     `1-2_mile_comp` + CompetitorIndex + transaction_density_4 +
##     user_density + avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_count_1_2_mi +
##     txn_count_2_3_mi
##
##                           Df Sum of Sq    RSS     AIC
## - AvgJoiningFee            1    0.0374 259.11 -6510.4
## - UMAP2D_2                 1    0.0484 259.12 -6510.3
## - unis_within_1_2_mile     1    0.0784 259.15 -6509.9
## - transaction_density_4    1    0.0868 259.16 -6509.8
## - NEAR_POP_SHARE           1    0.0986 259.17 -6509.7
## - POP_3_4                  1    0.1038 259.18 -6509.7
## - unis_within_0.5_1_mile   1    0.1117 259.18 -6509.6
## - CompetitorIndex          1    0.1151 259.19 -6509.5
## - GymSiteType              2    0.9060 259.98 -6508.9
```

```
## - POP_2_3                  1    0.1877 259.26 -6508.7
## - avg_merchant_per_user    1    0.2424 259.31 -6508.1
## - RATIO_1_TO_2             1    0.5311 259.60 -6505.0
## - INNER_RING_SHARE         1    0.5641 259.64 -6504.6
## - UMAP2D_1                 1    0.6198 259.69 -6504.0
## <none>                                 259.07 -6502.8
## - POP_0.5_1                1    1.0461 260.12 -6499.4
## - `0.5-1_mile_comp`        1    1.2512 260.32 -6497.1
## - RATIO_0_5_TO_1           1    1.2560 260.33 -6497.1
## - user_density             1    1.3613 260.43 -6495.9
## - distance_weighted_spend  1    1.4282 260.50 -6495.2
## - avg_spend_per_user       1    1.5166 260.59 -6494.3
## - txn_count_0_1_mi         1    1.9568 261.03 -6489.5
## - `1-2_mile_comp`          1    2.1195 261.19 -6487.7
## - POP_1_2                  1    2.1441 261.22 -6487.4
## - AvgAccountPayment        1    2.1764 261.25 -6487.1
## - txn_count_2_3_mi         1    2.3792 261.45 -6484.9
## - unis_within_0_0.5_mile   1    2.8382 261.91 -6479.9
## - `0-0.5_mile_comp`        1    2.9358 262.01 -6478.9
## - txn_count_1_2_mi         1    3.1123 262.19 -6477.0
## - BASEADL_0.5              1    3.5922 262.67 -6471.8
## - user_repeat_rate         1    3.9070 262.98 -6468.4
## - GymParking               1    4.2603 263.33 -6464.6
## - PromoPercentage          1    5.8625 264.94 -6447.4
##
## Step:  AIC=-6510.38
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
##     unis_within_1_2_mile + PromoPercentage + AvgAccountPayment +
##     BASEADL_0.5 + UMAP2D_1 + UMAP2D_2 + POP_0.5_1 + POP_1_2 +
##     POP_2_3 + POP_3_4 + NEAR_POP_SHARE + INNER_RING_SHARE + RATIO_0_5_TO_1 +
##     RATIO_1_TO_2 + `0-0.5_mile_comp` + `0.5-1_mile_comp` + `1-2_mile_comp` +
##     CompetitorIndex + transaction_density_4 + user_density +
##     avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_count_1_2_mi +
##     txn_count_2_3_mi
##
##                            Df Sum of Sq    RSS     AIC
## - UMAP2D_2                  1    0.0567 259.17 -6517.7
## - unis_within_1_2_mile      1    0.0757 259.19 -6517.5
## - transaction_density_4     1    0.0829 259.19 -6517.4
## - NEAR_POP_SHARE            1    0.0976 259.21 -6517.3
## - unis_within_0.5_1_mile    1    0.1058 259.22 -6517.2
## - POP_3_4                   1    0.1064 259.22 -6517.2
## - CompetitorIndex           1    0.1066 259.22 -6517.2
## - GymSiteType               2    0.9068 260.02 -6516.4
## - POP_2_3                   1    0.1923 259.30 -6516.2
## - avg_merchant_per_user     1    0.2337 259.34 -6515.8
## - RATIO_1_TO_2              1    0.5194 259.63 -6512.7
## - INNER_RING_SHARE          1    0.5558 259.67 -6512.3
## - UMAP2D_1                  1    0.6285 259.74 -6511.5
## <none>                                 259.11 -6510.4
## - POP_0.5_1                 1    1.0269 260.14 -6507.1
## - `0.5-1_mile_comp`         1    1.2254 260.33 -6505.0
## - RATIO_0_5_TO_1            1    1.2636 260.37 -6504.6
```

```
## - user_density             1    1.3413 260.45 -6503.7
## - distance_weighted_spend  1    1.4077 260.52 -6503.0
## + AvgJoiningFee            1    0.0374 259.07 -6502.8
## - avg_spend_per_user       1    1.4841 260.59 -6502.2
## - txn_count_0_1_mi         1    1.9404 261.05 -6497.2
## - `1-2_mile_comp`          1    2.0927 261.20 -6495.5
## - POP_1_2                  1    2.1577 261.27 -6494.8
## - AvgAccountPayment        1    2.1896 261.30 -6494.5
## - txn_count_2_3_mi         1    2.3743 261.48 -6492.5
## - unis_within_0_0.5_mile   1    2.8038 261.91 -6487.8
## - `0-0.5_mile_comp`        1    3.0120 262.12 -6485.6
## - txn_count_1_2_mi         1    3.1110 262.22 -6484.5
## - BASEADL_0.5              1    3.6625 262.77 -6478.6
## - user_repeat_rate         1    3.9300 263.04 -6475.7
## - GymParking               1    4.2233 263.33 -6472.5
## - PromoPercentage          1    7.6108 266.72 -6436.3
##
## Step:  AIC=-6517.71
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
##     unis_within_1_2_mile + PromoPercentage + AvgAccountPayment +
##     BASEADL_0.5 + UMAP2D_1 + POP_0.5_1 + POP_1_2 + POP_2_3 +
##     POP_3_4 + NEAR_POP_SHARE + INNER_RING_SHARE + RATIO_0_5_TO_1 +
##     RATIO_1_TO_2 + `0-0.5_mile_comp` + `0.5-1_mile_comp` + `1-2_mile_comp` +
##     CompetitorIndex + transaction_density_4 + user_density +
##     avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_count_1_2_mi +
##     txn_count_2_3_mi
##
##                            Df Sum of Sq    RSS     AIC
## - unis_within_1_2_mile      1    0.0547 259.22 -6525.1
## - transaction_density_4     1    0.0625 259.23 -6525.0
## - CompetitorIndex           1    0.0793 259.25 -6524.8
## - NEAR_POP_SHARE            1    0.1199 259.29 -6524.3
## - POP_3_4                   1    0.1305 259.30 -6524.2
## - unis_within_0.5_1_mile    1    0.1538 259.32 -6524.0
## - POP_2_3                   1    0.1790 259.35 -6523.7
## - avg_merchant_per_user     1    0.2316 259.40 -6523.1
## - GymSiteType               2    1.0170 260.18 -6522.5
## - RATIO_1_TO_2              1    0.4988 259.67 -6520.2
## - UMAP2D_1                  1    0.5729 259.74 -6519.4
## - INNER_RING_SHARE          1    0.6062 259.77 -6519.0
## <none>                                  259.17 -6517.7
## - POP_0.5_1                 1    0.9841 260.15 -6514.9
## - `0.5-1_mile_comp`         1    1.1701 260.34 -6512.9
## - user_density              1    1.2866 260.45 -6511.6
## - distance_weighted_spend   1    1.3680 260.54 -6510.7
## + UMAP2D_2                  1    0.0567 259.11 -6510.4
## + AvgJoiningFee             1    0.0457 259.12 -6510.3
## - RATIO_0_5_TO_1            1    1.4165 260.58 -6510.2
## - avg_spend_per_user        1    1.4276 260.59 -6510.1
## - txn_count_0_1_mi          1    2.0597 261.23 -6503.2
## - `1-2_mile_comp`           1    2.0662 261.23 -6503.2
## - AvgAccountPayment         1    2.1349 261.30 -6502.4
## - POP_1_2                   1    2.2870 261.45 -6500.8
```

```
## - txn_count_2_3_mi         1    2.3505 261.52 -6500.1
## - unis_within_0_0.5_mile   1    2.7953 261.96 -6495.3
## - `0-0.5_mile_comp`        1    2.9557 262.12 -6493.5
## - txn_count_1_2_mi         1    3.1448 262.31 -6491.5
## - BASEADL_0.5              1    3.6405 262.81 -6486.2
## - user_repeat_rate         1    3.9567 263.12 -6482.7
## - GymParking               1    4.8459 264.01 -6473.2
## - PromoPercentage          1    7.6211 266.79 -6443.6
##
## Step:  AIC=-6525.06
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
##     PromoPercentage + AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 +
##     POP_0.5_1 + POP_1_2 + POP_2_3 + POP_3_4 + NEAR_POP_SHARE +
##     INNER_RING_SHARE + RATIO_0_5_TO_1 + RATIO_1_TO_2 + `0-0.5_mile_comp` +
##     `0.5-1_mile_comp` + `1-2_mile_comp` + CompetitorIndex + transaction_density_4 +
##     user_density + avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_count_1_2_mi +
##     txn_count_2_3_mi
##
##                           Df Sum of Sq    RSS     AIC
## - transaction_density_4    1    0.0653 259.29 -6532.3
## - CompetitorIndex          1    0.0825 259.30 -6532.1
## - POP_3_4                  1    0.1083 259.33 -6531.8
## - NEAR_POP_SHARE           1    0.1221 259.34 -6531.7
## - unis_within_0.5_1_mile   1    0.1557 259.38 -6531.3
## - POP_2_3                  1    0.2076 259.43 -6530.7
## - avg_merchant_per_user    1    0.2161 259.44 -6530.6
## - GymSiteType              2    0.9985 260.22 -6530.1
## - RATIO_1_TO_2             1    0.4947 259.72 -6527.6
## - INNER_RING_SHARE         1    0.6061 259.83 -6526.4
## - UMAP2D_1                 1    0.6127 259.83 -6526.3
## <none>                                 259.22 -6525.1
## - POP_0.5_1                1    0.9804 260.20 -6522.3
## - `0.5-1_mile_comp`        1    1.1518 260.37 -6520.5
## - user_density             1    1.3060 260.53 -6518.8
## - distance_weighted_spend  1    1.3274 260.55 -6518.5
## + unis_within_1_2_mile     1    0.0547 259.17 -6517.7
## - avg_spend_per_user       1    1.4086 260.63 -6517.7
## + AvgJoiningFee            1    0.0414 259.18 -6517.6
## + UMAP2D_2                 1    0.0357 259.19 -6517.5
## - RATIO_0_5_TO_1           1    1.4542 260.68 -6517.2
## - txn_count_0_1_mi         1    2.0289 261.25 -6510.9
## - `1-2_mile_comp`          1    2.0595 261.28 -6510.6
## - AvgAccountPayment        1    2.1671 261.39 -6509.4
## - POP_1_2                  1    2.2498 261.47 -6508.5
## - txn_count_2_3_mi         1    2.3765 261.60 -6507.2
## - unis_within_0_0.5_mile   1    2.7450 261.97 -6503.2
## - `0-0.5_mile_comp`        1    2.9487 262.17 -6501.0
## - txn_count_1_2_mi         1    3.0914 262.31 -6499.4
## - BASEADL_0.5              1    3.6732 262.89 -6493.2
## - user_repeat_rate         1    3.9550 263.18 -6490.1
## - GymParking               1    4.8300 264.05 -6480.7
## - PromoPercentage          1    7.7408 266.96 -6449.7
##
```

```
## Step:  AIC=-6532.3
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
##     PromoPercentage + AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 +
##     POP_0.5_1 + POP_1_2 + POP_2_3 + POP_3_4 + NEAR_POP_SHARE +
##     INNER_RING_SHARE + RATIO_0_5_TO_1 + RATIO_1_TO_2 + `0-0.5_mile_comp` +
##     `0.5-1_mile_comp` + `1-2_mile_comp` + CompetitorIndex + user_density +
##     avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_count_1_2_mi +
##     txn_count_2_3_mi
##
##                           Df Sum of Sq    RSS     AIC
## - CompetitorIndex          1    0.1041 259.39 -6539.1
## - unis_within_0.5_1_mile   1    0.1230 259.41 -6538.9
## - NEAR_POP_SHARE           1    0.1238 259.41 -6538.9
## - POP_2_3                  1    0.1999 259.49 -6538.1
## - avg_merchant_per_user    1    0.2585 259.55 -6537.4
## - POP_3_4                  1    0.2598 259.55 -6537.4
## - GymSiteType              2    1.0333 260.32 -6536.9
## - RATIO_1_TO_2             1    0.4896 259.78 -6534.9
## - INNER_RING_SHARE         1    0.6051 259.89 -6533.6
## - UMAP2D_1                 1    0.6515 259.94 -6533.1
## <none>                                 259.29 -6532.3
## - POP_0.5_1                1    0.9225 260.21 -6530.2
## - `0.5-1_mile_comp`        1    1.2046 260.49 -6527.1
## - distance_weighted_spend  1    1.2801 260.57 -6526.3
## + transaction_density_4    1    0.0653 259.22 -6525.1
## + txn_count_3_4_mi         1    0.0653 259.22 -6525.1
## + txn_density_3_4_mi       1    0.0653 259.22 -6525.1
## + unis_within_1_2_mile     1    0.0576 259.23 -6525.0
## + AvgJoiningFee            1    0.0360 259.25 -6524.7
## + UMAP2D_2                 1    0.0198 259.27 -6524.6
## - avg_spend_per_user       1    1.5279 260.81 -6523.6
## - RATIO_0_5_TO_1           1    1.5814 260.87 -6523.0
## - `1-2_mile_comp`          1    2.1633 261.45 -6516.7
## - AvgAccountPayment        1    2.1860 261.47 -6516.5
## - POP_1_2                  1    2.3115 261.60 -6515.1
## - user_density             1    2.4308 261.72 -6513.8
## - txn_count_0_1_mi         1    2.6156 261.90 -6511.8
## - unis_within_0_0.5_mile   1    2.7127 262.00 -6510.8
## - txn_count_2_3_mi         1    3.1661 262.45 -6505.9
## - `0-0.5_mile_comp`        1    3.2298 262.52 -6505.2
## - txn_count_1_2_mi         1    3.7882 263.07 -6499.2
## - user_repeat_rate         1    3.8912 263.18 -6498.1
## - BASEADL_0.5              1    3.9161 263.20 -6497.8
## - GymParking               1    4.7847 264.07 -6488.5
## - PromoPercentage          1    7.8251 267.11 -6456.0
##
## Step:  AIC=-6539.11
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + unis_within_0.5_1_mile +
##     PromoPercentage + AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 +
##     POP_0.5_1 + POP_1_2 + POP_2_3 + POP_3_4 + NEAR_POP_SHARE +
##     INNER_RING_SHARE + RATIO_0_5_TO_1 + RATIO_1_TO_2 + `0-0.5_mile_comp` +
##     `0.5-1_mile_comp` + `1-2_mile_comp` + user_density + avg_spend_per_user +
##     user_repeat_rate + distance_weighted_spend + avg_merchant_per_user +
```

```
##     txn_count_0_1_mi + txn_count_1_2_mi + txn_count_2_3_mi
##
##                            Df Sum of Sq    RSS     AIC
## - unis_within_0.5_1_mile    1    0.1371 259.53 -6545.6
## - NEAR_POP_SHARE            1    0.1530 259.54 -6545.4
## - POP_2_3                   1    0.2105 259.60 -6544.8
## - POP_3_4                   1    0.2217 259.61 -6544.6
## - avg_merchant_per_user     1    0.2620 259.65 -6544.2
## - GymSiteType               2    1.0160 260.41 -6543.9
## - RATIO_1_TO_2              1    0.5285 259.92 -6541.3
## - UMAP2D_1                  1    0.6213 260.01 -6540.3
## - INNER_RING_SHARE          1    0.6885 260.08 -6539.5
## <none>                                  259.39 -6539.1
## - POP_0.5_1                 1    1.0680 260.46 -6535.4
## - distance_weighted_spend   1    1.2731 260.66 -6533.2
## + CompetitorIndex           1    0.1041 259.29 -6532.3
## + transaction_density_4     1    0.0869 259.30 -6532.1
## + txn_density_3_4_mi        1    0.0869 259.30 -6532.1
## + txn_count_3_4_mi          1    0.0869 259.30 -6532.1
## + unis_within_1_2_mile      1    0.0618 259.33 -6531.8
## + AvgJoiningFee             1    0.0245 259.37 -6531.4
## + UMAP2D_2                  1    0.0036 259.39 -6531.2
## - avg_spend_per_user        1    1.5177 260.91 -6530.5
## - RATIO_0_5_TO_1            1    1.6578 261.05 -6529.0
## - `0.5-1_mile_comp`         1    1.6628 261.05 -6529.0
## - AvgAccountPayment         1    2.2025 261.59 -6523.1
## - user_density              1    2.3884 261.78 -6521.1
## - POP_1_2                   1    2.5259 261.92 -6519.6
## - txn_count_0_1_mi          1    2.6013 261.99 -6518.8
## - unis_within_0_0.5_mile    1    2.6515 262.04 -6518.3
## - txn_count_2_3_mi          1    3.1571 262.55 -6512.8
## - txn_count_1_2_mi          1    3.7147 263.11 -6506.8
## - user_repeat_rate          1    3.8323 263.22 -6505.5
## - BASEADL_0.5               1    3.9234 263.31 -6504.5
## - `0-0.5_mile_comp`         1    4.3776 263.77 -6499.7
## - GymParking                1    4.9165 264.31 -6493.9
## - PromoPercentage           1    7.8034 267.19 -6463.1
## - `1-2_mile_comp`           1   11.5520 270.94 -6423.7
##
## Step:  AIC=-6545.56
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + PromoPercentage +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + POP_0.5_1 +
##     POP_1_2 + POP_2_3 + POP_3_4 + NEAR_POP_SHARE + INNER_RING_SHARE +
##     RATIO_0_5_TO_1 + RATIO_1_TO_2 + `0-0.5_mile_comp` + `0.5-1_mile_comp` +
##     `1-2_mile_comp` + user_density + avg_spend_per_user + user_repeat_rate +
##     distance_weighted_spend + avg_merchant_per_user + txn_count_0_1_mi +
##     txn_count_1_2_mi + txn_count_2_3_mi
##
##                            Df Sum of Sq    RSS     AIC
## - POP_2_3                   1    0.1594 259.69 -6551.8
## - NEAR_POP_SHARE            1    0.1823 259.71 -6551.5
## - POP_3_4                   1    0.2211 259.75 -6551.1
## - avg_merchant_per_user     1    0.2643 259.79 -6550.6
## - RATIO_1_TO_2              1    0.5688 260.10 -6547.3
```

```
## - GymSiteType              2    1.3063 260.83 -6547.2
## - UMAP2D_1                 1    0.6836 260.21 -6546.1
## <none>                                  259.53 -6545.6
## - INNER_RING_SHARE         1    0.7458 260.27 -6545.4
## - POP_0.5_1                1    1.2070 260.74 -6540.4
## - distance_weighted_spend  1    1.2918 260.82 -6539.4
## + unis_within_0.5_1_mile   1    0.1371 259.39 -6539.1
## + CompetitorIndex          1    0.1183 259.41 -6538.9
## + unis_within_1_2_mile     1    0.0630 259.46 -6538.3
## + txn_count_3_4_mi         1    0.0479 259.48 -6538.1
## + txn_density_3_4_mi       1    0.0479 259.48 -6538.1
## + transaction_density_4    1    0.0479 259.48 -6538.1
## + UMAP2D_2                 1    0.0221 259.51 -6537.9
## + AvgJoiningFee            1    0.0218 259.51 -6537.8
## - avg_spend_per_user       1    1.5250 261.05 -6536.9
## - RATIO_0_5_TO_1           1    1.7086 261.24 -6534.9
## - `0.5-1_mile_comp`        1    2.0590 261.59 -6531.1
## - AvgAccountPayment        1    2.2206 261.75 -6529.4
## - user_density             1    2.3576 261.89 -6527.9
## - unis_within_0_0.5_mile   1    2.5398 262.07 -6525.9
## - POP_1_2                  1    2.5688 262.10 -6525.6
## - txn_count_0_1_mi         1    2.6679 262.20 -6524.5
## - txn_count_2_3_mi         1    3.2767 262.81 -6518.0
## - user_repeat_rate         1    3.7228 263.25 -6513.2
## - txn_count_1_2_mi         1    3.7809 263.31 -6512.5
## - BASEADL_0.5              1    3.8878 263.42 -6511.4
## - `0-0.5_mile_comp`        1    4.6926 264.22 -6502.8
## - GymParking               1    4.9760 264.50 -6499.7
## - PromoPercentage          1    8.1581 267.69 -6465.9
## - `1-2_mile_comp`          1   12.1449 271.67 -6424.0
##
## Step:  AIC=-6551.77
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + PromoPercentage +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + POP_0.5_1 +
##     POP_1_2 + POP_3_4 + NEAR_POP_SHARE + INNER_RING_SHARE + RATIO_0_5_TO_1 +
##     RATIO_1_TO_2 + `0-0.5_mile_comp` + `0.5-1_mile_comp` + `1-2_mile_comp` +
##     user_density + avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_count_1_2_mi +
##     txn_count_2_3_mi
##
##                           Df Sum of Sq    RSS     AIC
## - NEAR_POP_SHARE           1    0.2087 259.90 -6557.4
## - avg_merchant_per_user    1    0.2165 259.90 -6557.4
## - POP_3_4                  1    0.4709 260.16 -6554.6
## - RATIO_1_TO_2             1    0.5086 260.20 -6554.2
## - GymSiteType              2    1.3071 261.00 -6553.4
## <none>                                  259.69 -6551.8
## - UMAP2D_1                 1    0.7677 260.45 -6551.4
## - INNER_RING_SHARE         1    0.9318 260.62 -6549.6
## - POP_0.5_1                1    1.1995 260.89 -6546.7
## + DENSITY_DROP_3           1    0.1594 259.53 -6545.6
## + POP_2_3                  1    0.1594 259.53 -6545.6
## + DENS_2_3                 1    0.1594 259.53 -6545.6
## + DENSITY_DROP_4           1    0.1594 259.53 -6545.6
```

```
## + CompetitorIndex          1    0.1252 259.56 -6545.2
## - distance_weighted_spend  1    1.3651 261.05 -6544.9
## + unis_within_1_2_mile     1    0.0894 259.60 -6544.8
## + unis_within_0.5_1_mile   1    0.0861 259.60 -6544.8
## + txn_count_3_4_mi         1    0.0488 259.64 -6544.4
## + txn_density_3_4_mi       1    0.0488 259.64 -6544.4
## + transaction_density_4    1    0.0488 259.64 -6544.4
## + AvgJoiningFee            1    0.0240 259.66 -6544.1
## + UMAP2D_2                 1    0.0089 259.68 -6543.9
## - avg_spend_per_user       1    1.5931 261.28 -6542.4
## - RATIO_0_5_TO_1           1    1.7589 261.45 -6540.6
## - `0.5-1_mile_comp`        1    1.9301 261.62 -6538.7
## - AvgAccountPayment        1    2.2796 261.97 -6535.0
## - user_density             1    2.3787 262.07 -6533.9
## - POP_1_2                  1    2.4257 262.11 -6533.4
## - txn_count_0_1_mi         1    2.6152 262.30 -6531.3
## - unis_within_0_0.5_mile   1    2.7760 262.46 -6529.6
## - user_repeat_rate         1    3.7230 263.41 -6519.4
## - txn_count_1_2_mi         1    3.7727 263.46 -6518.9
## - BASEADL_0.5              1    3.9211 263.61 -6517.3
## - `0-0.5_mile_comp`        1    4.5602 264.25 -6510.4
## - txn_count_2_3_mi         1    4.7402 264.43 -6508.5
## - GymParking               1    5.1003 264.79 -6504.6
## - PromoPercentage          1    8.1520 267.84 -6472.2
## - `1-2_mile_comp`          1   11.9868 271.67 -6431.9
##
## Step:  AIC=-6557.44
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + PromoPercentage +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + POP_0.5_1 +
##     POP_1_2 + POP_3_4 + INNER_RING_SHARE + RATIO_0_5_TO_1 + RATIO_1_TO_2 +
##     `0-0.5_mile_comp` + `0.5-1_mile_comp` + `1-2_mile_comp` +
##     user_density + avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     avg_merchant_per_user + txn_count_0_1_mi + txn_count_1_2_mi +
##     txn_count_2_3_mi
##
##                           Df Sum of Sq    RSS     AIC
## - avg_merchant_per_user    1    0.1940 260.09 -6563.3
## - RATIO_1_TO_2             1    0.3528 260.25 -6561.5
## - POP_3_4                  1    0.4502 260.35 -6560.5
## - GymSiteType              2    1.2215 261.12 -6560.1
## <none>                                 259.90 -6557.4
## - UMAP2D_1                 1    0.8113 260.71 -6556.6
## - POP_0.5_1                1    1.0562 260.95 -6553.9
## - INNER_RING_SHARE         1    1.1638 261.06 -6552.7
## + NEAR_POP_SHARE           1    0.2087 259.69 -6551.8
## + DENSITY_DROP_3           1    0.1858 259.71 -6551.5
## + POP_2_3                  1    0.1858 259.71 -6551.5
## + DENS_2_3                 1    0.1858 259.71 -6551.5
## + DENSITY_DROP_4           1    0.1858 259.71 -6551.5
## + CompetitorIndex          1    0.1654 259.73 -6551.3
## + unis_within_0.5_1_mile   1    0.1077 259.79 -6550.7
## + unis_within_1_2_mile     1    0.0968 259.80 -6550.5
## + txn_count_3_4_mi         1    0.0501 259.85 -6550.0
## + txn_density_3_4_mi       1    0.0501 259.85 -6550.0
```

```
## + transaction_density_4    1    0.0501 259.85 -6550.0
## + AvgJoiningFee            1    0.0225 259.87 -6549.7
## + UMAP2D_2                 1    0.0197 259.88 -6549.7
## - distance_weighted_spend  1    1.4992 261.39 -6549.1
## - avg_spend_per_user       1    1.7027 261.60 -6546.9
## - `0.5-1_mile_comp`        1    1.8509 261.75 -6545.3
## - RATIO_0_5_TO_1           1    1.9597 261.86 -6544.1
## - POP_1_2                  1    2.2198 262.12 -6541.3
## - AvgAccountPayment        1    2.2275 262.12 -6541.2
## - user_density             1    2.4321 262.33 -6539.0
## - txn_count_0_1_mi         1    2.6108 262.51 -6537.1
## - unis_within_0_0.5_mile   1    2.6561 262.55 -6536.6
## - BASEADL_0.5              1    3.7847 263.68 -6524.4
## - txn_count_1_2_mi         1    3.8570 263.75 -6523.7
## - user_repeat_rate         1    3.8922 263.79 -6523.3
## - `0-0.5_mile_comp`        1    4.6427 264.54 -6515.2
## - GymParking               1    4.9094 264.81 -6512.4
## - txn_count_2_3_mi         1    4.9640 264.86 -6511.8
## - PromoPercentage          1    8.1657 268.06 -6477.8
## - `1-2_mile_comp`          1   12.3246 272.22 -6434.2
##
## Step:  AIC=-6563.28
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + PromoPercentage +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + POP_0.5_1 +
##     POP_1_2 + POP_3_4 + INNER_RING_SHARE + RATIO_0_5_TO_1 + RATIO_1_TO_2 +
##     `0-0.5_mile_comp` + `0.5-1_mile_comp` + `1-2_mile_comp` +
##     user_density + avg_spend_per_user + user_repeat_rate + distance_weighted_spend +
##     txn_count_0_1_mi + txn_count_1_2_mi + txn_count_2_3_mi
##
##                           Df Sum of Sq    RSS     AIC
## - RATIO_1_TO_2             1    0.4278 260.52 -6566.6
## - POP_3_4                  1    0.4529 260.54 -6566.3
## - GymSiteType              2    1.2760 261.37 -6565.3
## <none>                                 260.09 -6563.3
## - INNER_RING_SHARE         1    1.0315 261.12 -6560.0
## - POP_0.5_1                1    1.0387 261.13 -6559.9
## - UMAP2D_1                 1    1.0589 261.15 -6559.7
## + avg_merchant_per_user    1    0.1940 259.90 -6557.4
## + NEAR_POP_SHARE           1    0.1862 259.90 -6557.4
## + CompetitorIndex          1    0.1656 259.92 -6557.1
## + DENSITY_DROP_3           1    0.1347 259.95 -6556.8
## + DENS_2_3                 1    0.1347 259.95 -6556.8
## + POP_2_3                  1    0.1347 259.95 -6556.8
## + DENSITY_DROP_4           1    0.1347 259.95 -6556.8
## + unis_within_0.5_1_mile   1    0.1151 259.98 -6556.6
## + transaction_density_4    1    0.0815 260.01 -6556.2
## + txn_count_3_4_mi         1    0.0815 260.01 -6556.2
## + txn_density_3_4_mi       1    0.0815 260.01 -6556.2
## + unis_within_1_2_mile     1    0.0728 260.02 -6556.1
## + UMAP2D_2                 1    0.0186 260.07 -6555.5
## + AvgJoiningFee            1    0.0154 260.07 -6555.5
## - distance_weighted_spend  1    1.5441 261.63 -6554.5
## - avg_spend_per_user       1    1.6002 261.69 -6553.9
## - `0.5-1_mile_comp`        1    1.7583 261.85 -6552.1
```

```
## - RATIO_0_5_TO_1           1    1.9378 262.03 -6550.2
## - AvgAccountPayment        1    2.2685 262.36 -6546.6
## - user_density             1    2.4041 262.49 -6545.2
## - POP_1_2                  1    2.4403 262.53 -6544.8
## - unis_within_0_0.5_mile   1    2.5696 262.66 -6543.4
## - txn_count_0_1_mi         1    2.9425 263.03 -6539.4
## - BASEADL_0.5              1    3.7170 263.81 -6531.0
## - txn_count_1_2_mi         1    4.1199 264.21 -6526.7
## - user_repeat_rate         1    4.5221 264.61 -6522.4
## - `0-0.5_mile_comp`        1    4.6433 264.73 -6521.1
## - GymParking               1    4.7404 264.83 -6520.1
## - txn_count_2_3_mi         1    5.1318 265.22 -6515.9
## - PromoPercentage          1    8.1214 268.21 -6484.1
## - `1-2_mile_comp`          1   12.1313 272.22 -6442.1
##
## Step:  AIC=-6566.57
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + PromoPercentage +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + POP_0.5_1 +
##     POP_1_2 + POP_3_4 + INNER_RING_SHARE + RATIO_0_5_TO_1 + `0-0.5_mile_comp` +
##     `0.5-1_mile_comp` + `1-2_mile_comp` + user_density + avg_spend_per_user +
##     user_repeat_rate + distance_weighted_spend + txn_count_0_1_mi +
##     txn_count_1_2_mi + txn_count_2_3_mi
##
##                           Df Sum of Sq    RSS     AIC
## - POP_3_4                  1    0.3092 260.83 -6571.2
## - GymSiteType              2    1.1980 261.72 -6569.5
## - POP_0.5_1                1    0.6825 261.20 -6567.1
## <none>                                  260.52 -6566.6
## - UMAP2D_1                 1    0.9337 261.45 -6564.4
## + RATIO_1_TO_2             1    0.4278 260.09 -6563.3
## + avg_merchant_per_user    1    0.2691 260.25 -6561.5
## + CompetitorIndex          1    0.1879 260.33 -6560.7
## + unis_within_0.5_1_mile   1    0.1474 260.37 -6560.2
## + transaction_density_4    1    0.0802 260.44 -6559.5
## + txn_count_3_4_mi         1    0.0802 260.44 -6559.5
## + txn_density_3_4_mi       1    0.0802 260.44 -6559.5
## + DENSITY_DROP_3           1    0.0654 260.45 -6559.3
## + POP_2_3                  1    0.0654 260.45 -6559.3
## + DENS_2_3                 1    0.0654 260.45 -6559.3
## + DENSITY_DROP_4           1    0.0654 260.45 -6559.3
## + unis_within_1_2_mile     1    0.0569 260.46 -6559.2
## + NEAR_POP_SHARE           1    0.0322 260.49 -6559.0
## + UMAP2D_2                 1    0.0067 260.51 -6558.7
## + AvgJoiningFee            1    0.0061 260.51 -6558.7
## - avg_spend_per_user       1    1.4730 261.99 -6558.6
## - distance_weighted_spend  1    1.5770 262.10 -6557.4
## - `0.5-1_mile_comp`        1    1.6258 262.14 -6556.9
## - POP_1_2                  1    2.0211 262.54 -6552.6
## - RATIO_0_5_TO_1           1    2.0807 262.60 -6552.0
## - AvgAccountPayment        1    2.3752 262.89 -6548.8
## - user_density             1    2.4780 263.00 -6547.7
## - unis_within_0_0.5_mile   1    2.5752 263.09 -6546.7
## - txn_count_0_1_mi         1    3.0062 263.52 -6542.0
## - BASEADL_0.5              1    4.1192 264.64 -6530.1
```

```
## - txn_count_1_2_mi         1    4.1498 264.67 -6529.8
## - `0-0.5_mile_comp`        1    4.7510 265.27 -6523.3
## - INNER_RING_SHARE         1    4.8401 265.36 -6522.4
## - GymParking               1    4.9042 265.42 -6521.7
## - txn_count_2_3_mi         1    5.1171 265.63 -6519.4
## - user_repeat_rate         1    5.2931 265.81 -6517.6
## - PromoPercentage          1    8.2929 268.81 -6485.8
## - `1-2_mile_comp`          1   12.3006 272.82 -6443.9
##
## Step:  AIC=-6571.16
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + PromoPercentage +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + POP_0.5_1 +
##     POP_1_2 + INNER_RING_SHARE + RATIO_0_5_TO_1 + `0-0.5_mile_comp` +
##     `0.5-1_mile_comp` + `1-2_mile_comp` + user_density + avg_spend_per_user +
##     user_repeat_rate + distance_weighted_spend + txn_count_0_1_mi +
##     txn_count_1_2_mi + txn_count_2_3_mi
##
##                            Df Sum of Sq    RSS     AIC
## - POP_0.5_1                 1    0.6504 261.48 -6572.1
## <none>                                  260.83 -6571.2
## - GymSiteType               2    1.4767 262.30 -6571.1
## - UMAP2D_1                  1    0.9932 261.82 -6568.3
## + POP_3_4                   1    0.3092 260.52 -6566.6
## + DENS_3_4                  1    0.3092 260.52 -6566.6
## + RATIO_1_TO_2              1    0.2842 260.54 -6566.3
## - avg_spend_per_user        1    1.1883 262.01 -6566.2
## + txn_count_3_4_mi          1    0.2698 260.56 -6566.1
## + txn_density_3_4_mi        1    0.2698 260.56 -6566.1
## + transaction_density_4     1    0.2698 260.56 -6566.1
## + avg_merchant_per_user     1    0.2573 260.57 -6566.0
## + DENSITY_DROP_3            1    0.2164 260.61 -6565.6
## + POP_2_3                   1    0.2164 260.61 -6565.6
## + DENS_2_3                  1    0.2164 260.61 -6565.6
## + CompetitorIndex           1    0.1200 260.71 -6564.5
## + unis_within_0.5_1_mile    1    0.1080 260.72 -6564.4
## + NEAR_POP_SHARE            1    0.0397 260.79 -6563.6
## + unis_within_1_2_mile      1    0.0285 260.80 -6563.5
## + UMAP2D_2                  1    0.0157 260.81 -6563.4
## + DENSITY_DROP_4            1    0.0127 260.81 -6563.4
## + AvgJoiningFee             1    0.0118 260.81 -6563.3
## - distance_weighted_spend   1    1.5418 262.37 -6562.4
## - `0.5-1_mile_comp`         1    1.5924 262.42 -6561.9
## - POP_1_2                   1    1.7924 262.62 -6559.7
## - RATIO_0_5_TO_1            1    2.1145 262.94 -6556.2
## - AvgAccountPayment         1    2.3136 263.14 -6554.1
## - txn_count_0_1_mi          1    2.7090 263.54 -6549.8
## - unis_within_0_0.5_mile    1    2.7373 263.56 -6549.5
## - user_density              1    3.0408 263.87 -6546.3
## - BASEADL_0.5               1    4.0014 264.83 -6536.0
## - txn_count_1_2_mi          1    4.1061 264.93 -6534.9
## - `0-0.5_mile_comp`         1    4.6384 265.46 -6529.2
## - GymParking                1    4.6744 265.50 -6528.8
## - INNER_RING_SHARE          1    4.7593 265.59 -6527.9
## - txn_count_2_3_mi          1    4.8255 265.65 -6527.2
```

```
## - user_repeat_rate         1    5.2431 266.07 -6522.7
## - PromoPercentage          1    8.4804 269.31 -6488.5
## - `1-2_mile_comp`          1   12.1849 273.01 -6449.8
##
## Step:  AIC=-6572.06
## logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile + PromoPercentage +
##     AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 + POP_1_2 + INNER_RING_SHARE +
##     RATIO_0_5_TO_1 + `0-0.5_mile_comp` + `0.5-1_mile_comp` +
##     `1-2_mile_comp` + user_density + avg_spend_per_user + user_repeat_rate +
##     distance_weighted_spend + txn_count_0_1_mi + txn_count_1_2_mi +
##     txn_count_2_3_mi
##
##                            Df Sum of Sq    RSS     AIC
## <none>                                  261.48 -6572.1
## + DENSITY_DROP_2            1    0.6504 260.83 -6571.2
## + DENS_0.5_1                1    0.6504 260.83 -6571.2
## + POP_0.5_1                 1    0.6504 260.83 -6571.2
## - GymSiteType               2    1.5825 263.06 -6570.9
## - UMAP2D_1                  1    1.0087 262.49 -6569.1
## - avg_spend_per_user        1    1.1414 262.62 -6567.7
## + POP_3_4                   1    0.2771 261.20 -6567.1
## + DENS_3_4                  1    0.2771 261.20 -6567.1
## + DENSITY_DROP_3            1    0.2395 261.24 -6566.7
## + POP_2_3                   1    0.2395 261.24 -6566.7
## + DENS_2_3                  1    0.2395 261.24 -6566.7
## + CompetitorIndex           1    0.2262 261.25 -6566.6
## + avg_merchant_per_user     1    0.1981 261.28 -6566.3
## + unis_within_0.5_1_mile    1    0.1763 261.30 -6566.0
## - POP_1_2                   1    1.3176 262.80 -6565.8
## + txn_count_3_4_mi          1    0.1275 261.35 -6565.5
## + txn_density_3_4_mi        1    0.1275 261.35 -6565.5
## + transaction_density_4     1    0.1275 261.35 -6565.5
## - `0.5-1_mile_comp`         1    1.3870 262.87 -6565.0
## + unis_within_1_2_mile      1    0.0322 261.44 -6564.5
## + RATIO_1_TO_2              1    0.0313 261.45 -6564.4
## + NEAR_POP_SHARE            1    0.0266 261.45 -6564.4
## + DENSITY_DROP_4            1    0.0039 261.47 -6564.2
## + AvgJoiningFee             1    0.0037 261.47 -6564.1
## + UMAP2D_2                  1    0.0034 261.47 -6564.1
## - distance_weighted_spend   1    1.4850 262.96 -6564.0
## - unis_within_0_0.5_mile    1    2.4830 263.96 -6553.2
## - AvgAccountPayment         1    2.5513 264.03 -6552.5
## - user_density              1    3.0565 264.53 -6547.1
## - txn_count_0_1_mi          1    3.0937 264.57 -6546.7
## - INNER_RING_SHARE          1    4.1431 265.62 -6535.5
## - GymParking                1    4.2686 265.75 -6534.1
## - txn_count_1_2_mi          1    4.3533 265.83 -6533.2
## - `0-0.5_mile_comp`         1    4.6793 266.16 -6529.8
## - txn_count_2_3_mi          1    4.8758 266.35 -6527.7
## - user_repeat_rate          1    5.1416 266.62 -6524.9
## - RATIO_0_5_TO_1            1    6.6552 268.13 -6508.8
## - PromoPercentage           1    8.6566 270.13 -6487.8
## - `1-2_mile_comp`           1   12.2954 273.77 -6449.9
## - BASEADL_0.5               1   12.3841 273.86 -6449.0
```

```r
print(summary(step_bic))
```

```
##
## Call:
## lm(formula = logTM ~ GymSiteType + GymParking + unis_within_0_0.5_mile +
##     PromoPercentage + AvgAccountPayment + BASEADL_0.5 + UMAP2D_1 +
##     POP_1_2 + INNER_RING_SHARE + RATIO_0_5_TO_1 + `0-0.5_mile_comp` +
##     `0.5-1_mile_comp` + `1-2_mile_comp` + user_density + avg_spend_per_user +
##     user_repeat_rate + distance_weighted_spend + txn_count_0_1_mi +
##     txn_count_1_2_mi + txn_count_2_3_mi, data = df_mod)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8003 -0.1411  0.0243  0.1730  0.7428
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               1.043e+01  2.524e-01  41.314  < 2e-16 ***
## GymSiteTypeResidential    6.867e-02  2.319e-02   2.961 0.003092 **
## GymSiteTypeWorkforce     -4.281e-02  3.662e-02  -1.169 0.242483
## GymParkingParking         1.168e-01  1.725e-02   6.773 1.53e-11 ***
## unis_within_0_0.5_mile    2.813e-02  5.446e-03   5.166 2.56e-07 ***
## PromoPercentage          -4.712e-01  4.885e-02  -9.645  < 2e-16 ***
## AvgAccountPayment         3.029e-03  5.784e-04   5.236 1.76e-07 ***
## BASEADL_0.5               1.945e-05  1.686e-06  11.536  < 2e-16 ***
## UMAP2D_1                 -6.405e-03  1.945e-03  -3.292 0.001006 **
## POP_1_2                  -1.248e-06  3.316e-07  -3.763 0.000171 ***
## INNER_RING_SHARE         -6.329e-01  9.486e-02  -6.673 3.01e-11 ***
## RATIO_0_5_TO_1           -4.241e-01  5.014e-02  -8.457  < 2e-16 ***
## `0-0.5_mile_comp`        -4.953e-02  6.984e-03  -7.091 1.67e-12 ***
## `0.5-1_mile_comp`        -1.754e-02  4.544e-03  -3.861 0.000116 ***
## `1-2_mile_comp`          -2.151e-02  1.872e-03 -11.495  < 2e-16 ***
## user_density             -3.490e-03  6.090e-04  -5.731 1.10e-08 ***
## avg_spend_per_user       -8.940e-04  2.553e-04  -3.502 0.000469 ***
## user_repeat_rate         -2.296e+00  3.089e-01  -7.433 1.40e-13 ***
## distance_weighted_spend   9.360e-03  2.343e-03   3.995 6.64e-05 ***
## txn_count_0_1_mi          3.022e-05  5.242e-06   5.766 9.00e-09 ***
## txn_count_1_2_mi          2.654e-05  3.881e-06   6.840 9.69e-12 ***
## txn_count_2_3_mi          2.647e-05  3.657e-06   7.239 5.82e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.305 on 2810 degrees of freedom
## Multiple R-squared:  0.3321, Adjusted R-squared:  0.3271
## F-statistic: 66.53 on 21 and 2810 DF,  p-value: < 2.2e-16
```

```r
# LASSO variable screening
x <- model.matrix(formula_all, data = df_mod)[,-1]
y <- df_mod$logTM
cv_lasso <- cv.glmnet(x, y, alpha = 1)
best_lambda <- cv_lasso$lambda.min
lasso_coef <- coef(cv_lasso, s = best_lambda)
print(lasso_coef)
```

```
## 47 x 1 sparse Matrix of class "dgCMatrix"
##                                      s0
## (Intercept)               9.990694e+00
## GymSiteTypeResidential    5.709849e-02
## GymSiteTypeWorkforce     -4.069281e-02
## GymParkingParking         1.248506e-01
## unis_within_0_0.5_mile    3.035035e-02
## unis_within_0.5_1_mile   -4.638511e-03
## unis_within_1_2_mile      1.679518e-03
## PromoPercentage          -4.424078e-01
## AvgJoiningFee             1.746392e-03
## AvgAccountPayment         2.762549e-03
## BASEADL_0.5               1.488663e-05
## UMAP2D_1                 -5.186150e-03
## UMAP2D_2                 -5.313798e-04
## POP_0.5_1                 .
## POP_1_2                   .
## POP_2_3                   .
## POP_3_4                   .
## DENS_0_0.5                2.147734e-09
## DENS_0.5_1                .
## DENS_1_2                  .
## DENS_2_3                  .
## DENS_3_4                  .
## DENSITY_DROP_2            1.154245e-05
## DENSITY_DROP_3           -1.318910e-05
## DENSITY_DROP_4           -4.128263e-06
## NEAR_POP_SHARE            5.462119e-01
## INNER_RING_SHARE         -6.161918e-01
## RATIO_0_5_TO_1           -3.407086e-01
## RATIO_1_TO_2             -1.728327e-01
## `0-0.5_mile_comp`        -4.991296e-02
## `0.5-1_mile_comp`        -2.238679e-02
## `1-2_mile_comp`          -2.438250e-02
## CompetitorIndex           6.028805e-04
## transaction_density_4     .
## user_density             -3.654298e-03
## avg_spend_per_user       -9.528662e-04
## user_repeat_rate         -2.108133e+00
## distance_weighted_spend   8.182795e-03
## avg_merchant_per_user     2.654335e-01
## txn_count_0_1_mi          2.664708e-05
## txn_density_0_1_mi        1.378600e-12
## txn_count_1_2_mi          2.532602e-05
## txn_density_1_2_mi        .
## txn_count_2_3_mi          2.097394e-05
## txn_density_2_3_mi        4.419955e-13
## txn_count_3_4_mi          .
## txn_density_3_4_mi        .
```

```r
# Cross - validation model comparison
set.seed(123)
train_ctrl <- trainControl(method = "cv", number = 5)
```

```r
# OLS CV (Variables after step_bic model selection, which can be replaced with formula_all)
cv_ols <- train(
  formula_all,
  data = df_mod,
  method = "lm",
  trControl = train_ctrl
)
print(cv_ols)
```

```
## Linear Regression
##
## 2832 samples
##   45 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2266, 2265, 2266, 2267, 2264
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.3061412  0.3262578  0.2015713
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
# LASSO CV
cv_glmnet <- train(
  x = x, y = y,
  method = "glmnet",
  tuneGrid = expand.grid(alpha = 1, lambda = cv_lasso$lambda),
  trControl = train_ctrl
)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

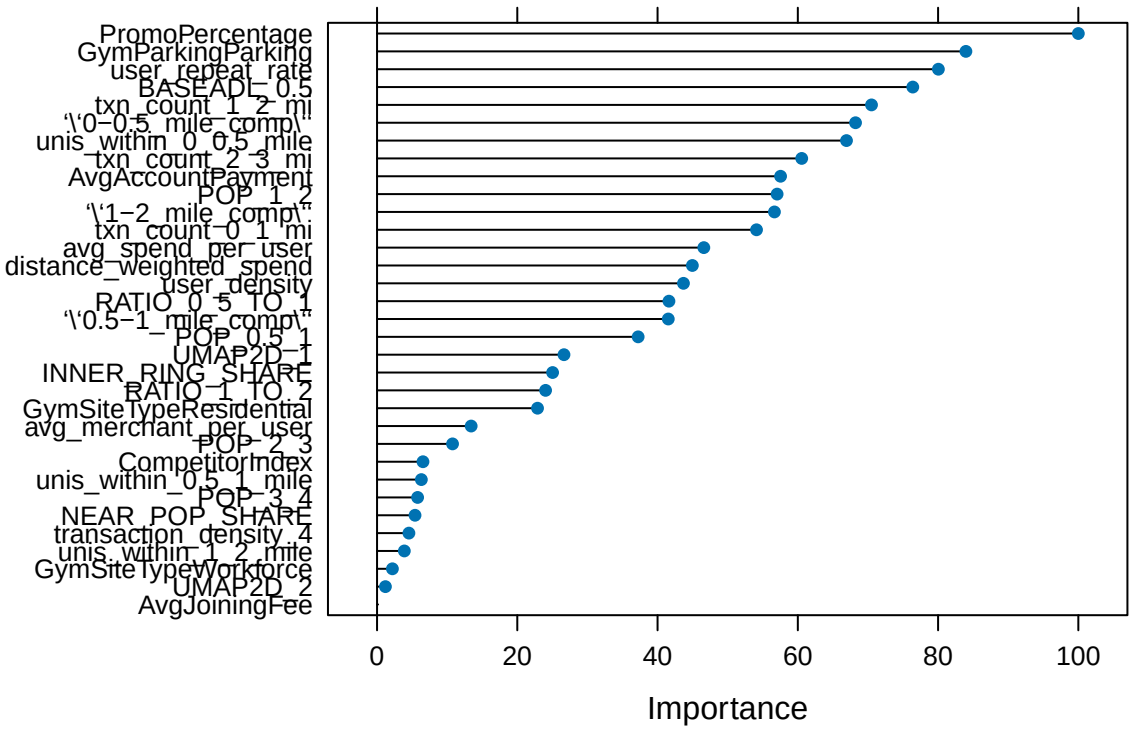
```r
print(cv_glmnet)
```

```
## glmnet
##
## 2832 samples
##   46 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2266, 2264, 2266, 2265, 2267
## Resampling results across tuning parameters:
##
##   lambda        RMSE       Rsquared    MAE
##   2.045720e-05  0.3017140  0.34149257  0.2017783
##   2.245175e-05  0.3017114  0.34150242  0.2017731
##   2.464076e-05  0.3017068  0.34151942  0.2017644
##   2.704321e-05  0.3017018  0.34153798  0.2017550
##   2.967989e-05  0.3016962  0.34155882  0.2017447
##   3.257364e-05  0.3016901  0.34158197  0.2017333
##   3.574953e-05  0.3016835  0.34160633  0.2017212
```

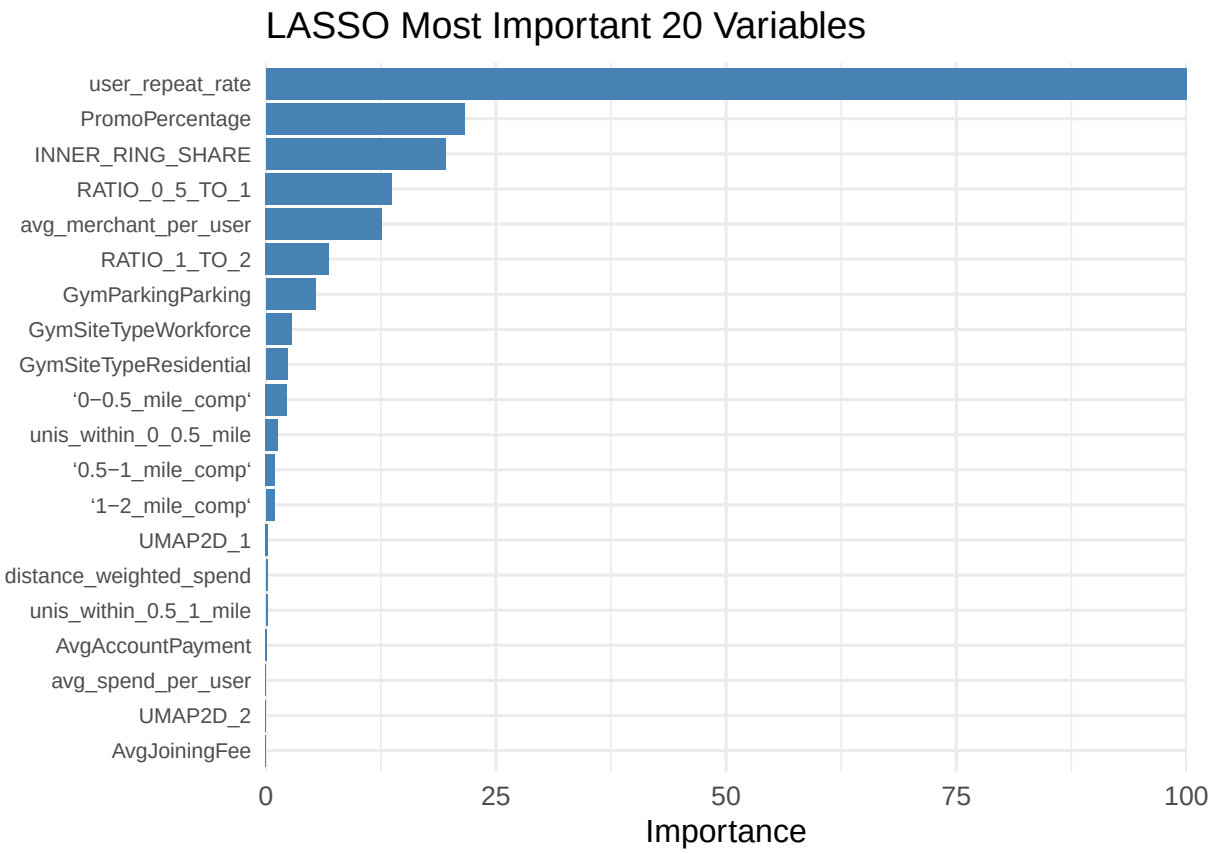
```
##   3.923506e-05  0.3016764  0.34163267  0.2017080
##   4.306043e-05  0.3016689  0.34166032  0.2016940
##   4.725877e-05  0.3016605  0.34169147  0.2016782
##   5.186645e-05  0.3016517  0.34172428  0.2016616
##   5.692336e-05  0.3016416  0.34176147  0.2016427
##   6.247332e-05  0.3016306  0.34180216  0.2016220
##   6.856439e-05  0.3016186  0.34184681  0.2015989
##   7.524933e-05  0.3016050  0.34189733  0.2015727
##   8.258605e-05  0.3015894  0.34195538  0.2015433
##   9.063809e-05  0.3015720  0.34202097  0.2015108
##   9.947519e-05  0.3015531  0.34209292  0.2014782
##   1.091739e-04  0.3015328  0.34216998  0.2014456
##   1.198182e-04  0.3015120  0.34224924  0.2014124
##   1.315003e-04  0.3014895  0.34233436  0.2013762
##   1.443215e-04  0.3014661  0.34242328  0.2013370
##   1.583926e-04  0.3014420  0.34251457  0.2012956
##   1.738357e-04  0.3014162  0.34261206  0.2012503
##   1.907845e-04  0.3013993  0.34267285  0.2011930
##   2.093857e-04  0.3013963  0.34267781  0.2011282
##   2.298006e-04  0.3013827  0.34272823  0.2010781
##   2.522059e-04  0.3013612  0.34280459  0.2010130
##   2.767956e-04  0.3013434  0.34286224  0.2009512
##   3.037829e-04  0.3013352  0.34288351  0.2009037
##   3.334013e-04  0.3013275  0.34290147  0.2008723
##   3.659075e-04  0.3013387  0.34283612  0.2008035
##   4.015831e-04  0.3013396  0.34281525  0.2007496
##   4.407369e-04  0.3013370  0.34281909  0.2007133
##   4.837082e-04  0.3013353  0.34279670  0.2006675
##   5.308692e-04  0.3013224  0.34282802  0.2006410
##   5.826283e-04  0.3013147  0.34283269  0.2006047
##   6.394338e-04  0.3012866  0.34290894  0.2005629
##   7.017778e-04  0.3012640  0.34296248  0.2004944
##   7.702003e-04  0.3012417  0.34301449  0.2004558
##   8.452938e-04  0.3012187  0.34305546  0.2003844
##   9.277089e-04  0.3012227  0.34298316  0.2003491
##   1.018159e-03  0.3012165  0.34296554  0.2002877
##   1.117429e-03  0.3011998  0.34299713  0.2002295
##   1.226377e-03  0.3011951  0.34297433  0.2001860
##   1.345947e-03  0.3011993  0.34291861  0.2001544
##   1.477175e-03  0.3012090  0.34284379  0.2001257
##   1.621198e-03  0.3012265  0.34273595  0.2000978
##   1.779263e-03  0.3012609  0.34255527  0.2000794
##   1.952738e-03  0.3013203  0.34226957  0.2000739
##   2.143128e-03  0.3013953  0.34192178  0.2000898
##   2.352080e-03  0.3014848  0.34152061  0.2001170
##   2.581405e-03  0.3015922  0.34105009  0.2001421
##   2.833089e-03  0.3017255  0.34046838  0.2001846
##   3.109312e-03  0.3018802  0.33980861  0.2002367
##   3.412466e-03  0.3020448  0.33913705  0.2002687
##   3.745177e-03  0.3021989  0.33855516  0.2002890
##   4.110327e-03  0.3023683  0.33796757  0.2003161
##   4.511079e-03  0.3025602  0.33734631  0.2003478
##   4.950904e-03  0.3028095  0.33653597  0.2004179
##   5.433611e-03  0.3031231  0.33550625  0.2005308
```

```
##   5.963381e-03  0.3035148  0.33419903  0.2006903
##   6.544803e-03  0.3039916  0.33258112  0.2008985
##   7.182913e-03  0.3045600  0.33063089  0.2011885
##   7.883239e-03  0.3052258  0.32833563  0.2015692
##   8.651845e-03  0.3059770  0.32577010  0.2020470
##   9.495389e-03  0.3068694  0.32263918  0.2026757
##   1.042118e-02  0.3078966  0.31898113  0.2034802
##   1.143723e-02  0.3090292  0.31495501  0.2043770
##   1.255235e-02  0.3102542  0.31063709  0.2054067
##   1.377618e-02  0.3114876  0.30656062  0.2065030
##   1.511934e-02  0.3127148  0.30294570  0.2076079
##   1.659346e-02  0.3141138  0.29879273  0.2089070
##   1.821130e-02  0.3157229  0.29375769  0.2104330
##   1.998688e-02  0.3174357  0.28846920  0.2120434
##   2.193558e-02  0.3191164  0.28397032  0.2135997
##   2.407427e-02  0.3208434  0.27980245  0.2151942
##   2.642148e-02  0.3226963  0.27539544  0.2168378
##   2.899755e-02  0.3248213  0.26975021  0.2187262
##   3.182477e-02  0.3273639  0.26177876  0.2209210
##   3.492765e-02  0.3303989  0.25022079  0.2234775
##   3.833305e-02  0.3338928  0.23439563  0.2263948
##   4.207047e-02  0.3376490  0.21505019  0.2295157
##   4.617229e-02  0.3415511  0.19248516  0.2327899
##   5.067403e-02  0.3450462  0.17279302  0.2357229
##   5.561469e-02  0.3479977  0.15767225  0.2382702
##   6.103705e-02  0.3508790  0.14153153  0.2407584
##   6.698809e-02  0.3535986  0.12562448  0.2430112
##   7.351935e-02  0.3555455  0.12010175  0.2444195
##   8.068739e-02  0.3574338  0.11796574  0.2455248
##   8.855431e-02  0.3595801  0.11569631  0.2467159
##   9.718825e-02  0.3619810  0.11461235  0.2481192
##   1.066640e-01  0.3647695  0.11453610  0.2498739
##   1.170636e-01  0.3671931  0.08279812  0.2515002
##
## Tuning parameter 'alpha' was held constant at a value of 1
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 1 and lambda = 0.001226377.
```

```r
# Visualize CV results
plot(caret::varImp(cv_ols), main = "OLS Variable Importance")
```




```r
imp <- caret::varImp(cv_glmnet)$importance
imp$Var <- rownames(imp)
top20 <- imp %>%
  arrange(desc(Overall)) %>%
  slice_head(n = 20)
ggplot(top20, aes(x = reorder(Var, Overall), y = Overall)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  scale_y_continuous(expand = c(0, 0)) +
  labs(
    title = "LASSO Most Important 20 Variables",
    x = NULL,
    y = "Importance"
  ) +
  theme_minimal(base_size = 12) +
  theme(
    axis.text.y = element_text(size = 8),
    plot.margin = margin(t = 5, r = 20, b = 5, l = 5)
  )
```



Variable processing

```
selected_vars <- c(
  "BASEADL_0.5", "POP_0.5_1", "POP_1_2", "POP_2_3", "POP_3_4",
  "DENS_0_0.5", "DENS_0.5_1", "DENS_1_2", "DENS_2_3", "DENS_3_4",
  "DENSITY_DROP_2", "DENSITY_DROP_3", "DENSITY_DROP_4",
  "txn_count_0_1_mi", "txn_density_0_1_mi",
  "txn_count_1_2_mi", "txn_density_1_2_mi",
  "txn_count_2_3_mi", "txn_density_2_3_mi",
  "txn_count_3_4_mi", "txn_density_3_4_mi"
)

id_cols   <- c("HashedGymPublicName", "Longitude", "Latitude", "StartMonth")
# Numerical predictor variables (excluding ID, latitude and longitude, time, and response)
num_cols <- df %>%
  select(-all_of(id_cols), -TotalMembers) %>%
  select(where(is.numeric)) %>%
  names()
# Categorical (non-numerical) predictor variables
cat_cols <- df %>%
  select(-all_of(id_cols), -TotalMembers) %>%
  select(where(~ !is.numeric(.))) %>%
  names()

# Calculate skewness
```

```r
skew_df <- tibble(
  var = selected_vars,
  skew = map_dbl(selected_vars, ~ skewness(df[[.x]], na.rm = TRUE))
)

# Classification processing methods
skewed_vars <- skew_df %>% filter(abs(skew) > 1) %>% pull(var)
scale_vars  <- setdiff(selected_vars, skewed_vars)

# • For variables with small skewness (scale_vars), it is quite reasonable to directly perform standard
# • For variables with high skewness (skewed_vars), standardization still retains the influence of extr

df_trans <- df %>%
  mutate(across(all_of(skewed_vars), ~ log(.x + 1))) %>%
  mutate(across(all_of(scale_vars), ~ as.numeric(scale(.x)))) %>%
  mutate(across(all_of(cat_cols), as.factor))

# Perform preprocessing (only on selected columns)
df_final <- df %>%
  mutate(across(all_of(skewed_vars), ~ log(.x + 1))) %>%
  mutate(across(all_of(scale_vars),  ~ as.numeric(scale(.x)))) %>%
  mutate(across(all_of(cat_cols), as.factor))  #


# Check which variables are logged and which are standardized.
skew_df %>% mutate(transformed = case_when(
  var %in% skewed_vars ~ "log1p",
  var %in% scale_vars  ~ "standardize",
  TRUE                 ~ "-"
)) %>% print(n = Inf)
```

```
## # A tibble: 21 x 3
##    var                   skew transformed
##    <chr>                <dbl> <chr>
##  1 BASEADL_0.5         0.854  standardize
##  2 POP_0.5_1           1.07   log1p
##  3 POP_1_2             1.13   log1p
##  4 POP_2_3             1.23   log1p
##  5 POP_3_4             1.26   log1p
##  6 DENS_0_0.5          0.854  standardize
##  7 DENS_0.5_1          1.07   log1p
##  8 DENS_1_2            1.13   log1p
##  9 DENS_2_3            1.23   log1p
## 10 DENS_3_4            1.26   log1p
## 11 DENSITY_DROP_2      0.463  standardize
## 12 DENSITY_DROP_3     -0.405  standardize
## 13 DENSITY_DROP_4      0.0719 standardize
## 14 txn_count_0_1_mi    0.549  standardize
## 15 txn_density_0_1_mi  0.549  standardize
## 16 txn_count_1_2_mi    1.03   log1p
## 17 txn_density_1_2_mi  1.03   log1p
## 18 txn_count_2_3_mi    0.939  standardize
## 19 txn_density_2_3_mi  0.939  standardize
## 20 txn_count_3_4_mi    1.19   log1p
```

```
## 21 txn_density_3_4_mi  1.19   log1p
```

```
glimpse(df_final)
```

```
## Rows: 2,832
## Columns: 50
## $ StartMonth              <date> 2024-01-01, 2024-02-01, 2024-03-01, 2024-04-0~
## $ TotalMembers            <dbl> 3373, 3439, 3413, 3547, 3389, 3402, 3651, 3377~
## $ GymSiteType             <fct> Residential, Residential, Residential, Residen~
## $ GymParking              <fct> Parking, Parking, Parking, Parking, Parking, P~
## $ unis_within_0_0.5_mile  <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0~
## $ unis_within_0.5_1_mile  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1~
## $ unis_within_1_2_mile    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ PromoPercentage         <dbl> 0.6442813, 0.3344648, 0.4310850, 0.4503497, 0.~
## $ AvgJoiningFee           <dbl> 2.105873, 3.818538, 3.207478, 3.001166, 3.6444~
## $ AvgAccountPayment       <dbl> 25.98586, 31.23805, 25.15532, 27.32462, 30.932~
## $ BASEADL_0.5             <dbl> -0.3737683, -0.3737683, -0.3737683, -0.3737683~
## $ UMAP2D_1                <dbl> 15.117895, 15.117895, 15.117895, 15.117895, 15~
## $ UMAP2D_2                <dbl> 0.9827856, 0.9827856, 0.9827856, 0.9827856, 0.~
## $ POP_0.5_1               <dbl> 9.674816, 9.674816, 9.674816, 9.674816, 9.6748~
## $ POP_1_2                 <dbl> 10.20665, 10.20665, 10.20665, 10.20665, 10.206~
## $ POP_2_3                 <dbl> 9.983878, 9.983878, 9.983878, 9.983878, 9.9838~
## $ POP_3_4                 <dbl> 10.699909, 10.699909, 10.699909, 10.699909, 10~
## $ DENS_0_0.5              <dbl> -0.3737683, -0.3737683, -0.3737683, -0.3737683~
## $ DENS_0.5_1              <dbl> 8.826151, 8.826151, 8.826151, 8.826151, 8.8261~
## $ DENS_1_2                <dbl> 7.971912, 7.971912, 7.971912, 7.971912, 7.9719~
## $ DENS_2_3                <dbl> 7.238681, 7.238681, 7.238681, 7.238681, 7.2386~
## $ DENS_3_4                <dbl> 7.522775, 7.522775, 7.522775, 7.522775, 7.5227~
## $ DENSITY_DROP_2          <dbl> 0.2006485, 0.2006485, 0.2006485, 0.2006485, 0.~
## $ DENSITY_DROP_3          <dbl> -0.17487873, -0.17487873, -0.17487873, -0.1748~
## $ DENSITY_DROP_4          <dbl> -0.9381247, -0.9381247, -0.9381247, -0.9381247~
## $ NEAR_POP_SHARE          <dbl> 0.08579795, 0.08579795, 0.08579795, 0.08579795~
## $ INNER_RING_SHARE        <dbl> 0.2192245, 0.2192245, 0.2192245, 0.2192245, 0.~
## $ RATIO_0_5_TO_1          <dbl> 0.6430349, 0.6430349, 0.6430349, 0.6430349, 0.~
## $ RATIO_1_TO_2            <dbl> 0.5875133, 0.5875133, 0.5875133, 0.5875133, 0.~
## $ `0-0.5_mile_comp`       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ `0.5-1_mile_comp`       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1~
## $ `1-2_mile_comp`         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ CompetitorIndex         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 4, 4, 4~
## $ transaction_density_4   <dbl> 124.6979, 124.6979, 124.6979, 124.6979, 124.69~
## $ user_density            <dbl> 18.34261, 18.34261, 18.34261, 18.34261, 18.342~
## $ avg_spend_per_user      <dbl> 144.3027, 144.3027, 144.3027, 144.3027, 144.30~
## $ user_repeat_rate        <dbl> 0.8373102, 0.8373102, 0.8373102, 0.8373102, 0.~
## $ distance_weighted_spend <dbl> 22.12553, 22.12553, 22.12553, 22.12553, 22.125~
## $ avg_merchant_per_user   <dbl> 1.138829, 1.138829, 1.138829, 1.138829, 1.1388~
## $ txn_count_0_1_mi        <dbl> 0.0573393, 0.0573393, 0.0573393, 0.0573393, 0.~
## $ txn_density_0_1_mi      <dbl> 0.0573393, 0.0573393, 0.0573393, 0.0573393, 0.~
## $ txn_count_1_2_mi        <dbl> 6.242223, 6.242223, 6.242223, 6.242223, 6.2422~
## $ txn_density_1_2_mi      <dbl> 4.015139, 4.015139, 4.015139, 4.015139, 4.0151~
## $ txn_count_2_3_mi        <dbl> -0.9720249, -0.9720249, -0.9720249, -0.9720249~
## $ txn_density_2_3_mi      <dbl> -0.9720249, -0.9720249, -0.9720249, -0.9720249~
## $ txn_count_3_4_mi        <dbl> 7.495542, 7.495542, 7.495542, 7.495542, 7.4955~
## $ txn_density_3_4_mi      <dbl> 4.416496, 4.416496, 4.416496, 4.416496, 4.4164~
## $ Latitude                <dbl> 53.75214, 53.75214, 53.75214, 53.75214, 53.752~
## $ Longitude               <dbl> -2.3624993, -2.3624993, -2.3624993, -2.3624993~
```

```
## $ HashedGymPublicName     <chr> "303d35421b8d224ed997d4af7f0a0dfd189ac35c84b1d~
```

```
df_tmp <- df %>%
  mutate(
    logTM = log(TotalMembers)
  )

# 1. First, define a set of possible  values;
# 2. For each , perform a simple model cross - validation (here only weighted_pop is used for demonstra
# 3. Record the  corresponding to the minimum RMSE.
# 4. Finally, assign this best_beta to your mutate to generate the formal weighted features.

betas <- seq(0.1, 2, by = 0.1)
results <- tibble(beta = betas, RMSE = NA_real_)

set.seed(42)
for(i in seq_along(betas)) {
  b <- betas[i]
  df_w <- df_tmp %>%
    mutate(
      weighted_pop =
        POP_0.5_1 * exp(-b * 0.75) +
        POP_1_2   * exp(-b * 1.5)  +
        POP_2_3   * exp(-b * 2.5)  +
        POP_3_4   * exp(-b * 3.5)
    )
  cv <- train(
    logTM ~ weighted_pop,
    data      = df_w,
    method    = "lm",
    trControl = trainControl(method = "cv", number = 5)
  )
  results$RMSE[i] <- cv$results$RMSE
}

best <- results %>% slice_min(RMSE, n = 1)
best_beta <- best$beta
cat("best  =", best_beta, "Corresponding to RMSE =", best$RMSE, "\n")
```

```
## best  = 1.3 Corresponding to RMSE = 0.3686686
```

```
beta <- best_beta

df_time <- df %>%
  # Exponential decay weighted features
    mutate(
    weighted_pop =
      POP_0.5_1 * exp(-beta * 0.75) +
      POP_1_2 * exp(-beta * 1.5)  +
      POP_2_3 * exp(-beta * 2.5)  +
      POP_3_4 * exp(-beta * 3.5),

    weighted_den =
      DENS_0_0.5 * exp(-beta * 0.25) +
      DENS_0.5_1 * exp(-beta * 0.75) +
```

```
      DENS_1_2 * exp(-beta * 1.5)  +
      DENS_2_3 * exp(-beta * 2.5)  +
      DENS_3_4 * exp(-beta * 3.5),

    weighted_txn =
      txn_density_0_1_mi * exp(-beta * 0.5)  +
      txn_density_1_2_mi * exp(-beta * 1.5)  +
      txn_density_2_3_mi * exp(-beta * 2.5)  +
      txn_density_3_4_mi * exp(-beta * 3.5),

    weighted_comp =
      `0-0.5_mile_comp` * exp(-beta * 0.25) +
      `0.5-1_mile_comp` * exp(-beta * 0.75) +
      `1-2_mile_comp`   * exp(-beta * 1.5),

    weighted_unis =
      unis_within_0_0.5_mile * exp(-beta * 0.25) +
      unis_within_0.5_1_mile * exp(-beta * 0.75) +
      unis_within_1_2_mile   * exp(-beta * 1.5),

    weighted_density_drop =
      DENSITY_DROP_2 * exp(-beta * 0.75) +
      DENSITY_DROP_3 * exp(-beta * 1.5)  +
      DENSITY_DROP_4 * exp(-beta * 2.5),

    weighted_ring_share =
      NEAR_POP_SHARE * exp(-beta * 0.25) +
      INNER_RING_SHARE * exp(-beta * 0.75),

    weighted_ratio =
      RATIO_0_5_TO_1 * exp(-beta * 0.75) +
      RATIO_1_TO_2 * exp(-beta * 1.5)
  )%>%
  mutate(
    price_share = AvgAccountPayment / avg_spend_per_user
  )



m_wpop <- mean(df_time$weighted_pop, na.rm = TRUE)
sd_wpop   <- sd(  df_time$weighted_pop, na.rm = TRUE)

m_wden <- mean(df_time$weighted_den, na.rm = TRUE)
sd_wden   <- sd(  df_time$weighted_den, na.rm = TRUE)

m_wtxn <- mean(df_time$weighted_txn, na.rm = TRUE)
sd_wtxn   <- sd(  df_time$weighted_txn, na.rm = TRUE)

m_wcomp <- mean(df_time$weighted_comp, na.rm = TRUE)
sd_wcomp   <- sd(  df_time$weighted_comp, na.rm = TRUE)

m_unis_std <- mean(df_time$weighted_unis, na.rm = TRUE)
sd_unis_std   <- sd(  df_time$weighted_unis, na.rm = TRUE)
```

```r
m_ddrop_std   <- mean(df_time$weighted_density_drop, na.rm=TRUE)
sd_ddrop_std    <-  sd(df_time$weighted_density_drop, na.rm=TRUE)

m_ringshare_std    <- mean(df_time$weighted_ring_share, na.rm=TRUE)
sd_ringshare_std      <-sd(df_time$weighted_ring_share, na.rm=TRUE)

m_ratio_std    <- mean(df_time$weighted_ratio, na.rm=TRUE)
sd_ratio_std    <- sd(df_time$weighted_ratio, na.rm=TRUE)


df_time <- df_time %>%
  mutate(
    w_pop = (weighted_pop - m_wpop) / sd_wpop,
    w_den = (weighted_den - m_wden) / sd_wden,
    w_txn = (weighted_txn - m_wtxn) / sd_wtxn,
    w_comp = (weighted_comp - m_wcomp) / sd_wcomp,
    w_unis = (weighted_unis - m_unis_std) / sd_unis_std,
    w_ddrop_std = (weighted_density_drop - m_ddrop_std) / sd_ddrop_std,
    w_ringshare_std = (weighted_ring_share - m_ringshare_std) / sd_ringshare_std,
    w_ratio_std =(weighted_ratio - m_ratio_std) / sd_ratio_std
  )
```

```r
# Define the variables to be excluded
drop_vars <- c(
  "POP_0.5_1", "POP_1_2", "POP_2_3", "POP_3_4",
  "DENS_0_0.5", "DENS_0.5_1", "DENS_1_2", "DENS_2_3", "DENS_3_4",
  "txn_density_0_1_mi", "txn_density_1_2_mi",
  "txn_density_2_3_mi", "txn_density_3_4_mi",
  "txn_count_0_1_mi", "txn_count_1_2_mi",
  "txn_count_2_3_mi", "txn_count_3_4_mi",
  "0-0.5_mile_comp", "0.5-1_mile_comp", "1-2_mile_comp",
  "unis_within_0_0.5_mile", "unis_within_0.5_1_mile", "unis_within_1_2_mile",
  "DENSITY_DROP_2", "DENSITY_DROP_3", "DENSITY_DROP_4",
  "NEAR_POP_SHARE", "INNER_RING_SHARE",
  "RATIO_0_5_TO_1", "RATIO_1_TO_2",
  "weighted_pop", "weighted_den", "weighted_txn", "weighted_comp","weighted_unis",
  "weighted_density_drop", "weighted_ring_share", "weighted_ratio"
)

# Select numerical variables and exclude the above-mentioned variables
numeric_data <- df_time %>%
  select(where(is.numeric)) %>%
  select(-any_of(drop_vars))

# Calculate the correlation coefficient matrix
cor_mat <- cor(numeric_data, use = "complete.obs")
cor_vec <- cor_mat["TotalMembers", ]
cor_vec <- cor_vec[names(cor_vec) != "TotalMembers"]
# Take the top 10 with the highest absolute values
top10 <- sort(abs(cor_vec), decreasing = TRUE)[1:10]

top10_df <- tibble(
  Variable    = names(top10),
```

```
  Correlation = cor_vec[names(top10)]
)

print(top10_df)
```

```
## # A tibble: 10 x 2
##    Variable         Correlation
##    <chr>                  <dbl>
##  1 BASEADL_0.5            0.235
##  2 w_comp                -0.221
##  3 w_den                  0.219
##  4 w_txn                  0.209
##  5 w_unis                -0.193
##  6 CompetitorIndex       -0.189
##  7 w_ringshare_std       -0.185
##  8 user_repeat_rate      -0.181
##  9 w_ddrop_std            0.180
## 10 Latitude              -0.149
```

Main Indicators and Model Summary

$$RMSE = \sqrt{\frac{1}{n}\sum(\hat{y}_i - y_i)^2}, \quad MAE = \frac{1}{n}\sum|\hat{y}_i - y_i|$$

OLS

```
# Divide the training set and the test set (3/4; 1/4) by each row.
df_time$GymSiteType <- factor(df_time$GymSiteType,levels = c("Hybrid", "Residential", "Workforce"))
sp <- initial_split(df_time, prop = .75)
set_train <- training(sp)
set_test <- testing(sp)

set_train_ols <- set_train %>% select(-any_of(c(drop_vars, "HashedGymPublicName", "StartMonth")))
set_test_ols <- set_test %>% select(-any_of(c(drop_vars, "HashedGymPublicName", "StartMonth")))

# OLS
m_ols <- lm(TotalMembers ~ ., data = set_train_ols)
pred_ols <- predict(m_ols, newdata = set_test_ols)

# RMSE   MAE
rmse <- sqrt(mean((pred_ols - set_test_ols$TotalMembers)^2))
mae  <- mean(abs(pred_ols - set_test_ols$TotalMembers))

summary(m_ols)
```

```
##
## Call:
## lm(formula = TotalMembers ~ ., data = set_train_ols)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -6294.8  -724.9     4.3   725.5  3152.6
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               1.967e+04  6.850e+03   2.872  0.00412 **
## GymSiteTypeResidential    2.526e+02  1.037e+02   2.437  0.01490 *
## GymSiteTypeWorkforce     -3.411e+02  1.558e+02  -2.189  0.02870 *
## GymParkingParking         3.612e+02  7.677e+01   4.705 2.70e-06 ***
## PromoPercentage          -7.046e+02  2.277e+02  -3.095  0.00199 **
## AvgJoiningFee             3.301e+01  1.667e+01   1.980  0.04783 *
## AvgAccountPayment         2.134e+01  9.230e+00   2.312  0.02089 *
## BASEADL_0.5              -2.560e-01  5.095e-01  -0.502  0.61545
## UMAP2D_1                 -1.543e+01  8.889e+00  -1.735  0.08281 .
## UMAP2D_2                 -1.310e+01  1.435e+01  -0.913  0.36137
## CompetitorIndex          -2.818e+01  4.646e+00  -6.066 1.55e-09 ***
## transaction_density_4     1.303e+00  9.750e-01   1.336  0.18161
## user_density             -1.257e+00  6.571e+00  -0.191  0.84832
## avg_spend_per_user       -4.646e+00  2.120e+00  -2.191  0.02854 *
## user_repeat_rate         -1.037e+04  1.475e+03  -7.031 2.76e-12 ***
## distance_weighted_spend   2.257e+01  1.048e+01   2.154  0.03135 *
## avg_merchant_per_user     5.403e+02  6.982e+02   0.774  0.43907
## Latitude                 -6.351e+01  2.477e+01  -2.564  0.01042 *
## Longitude                -1.186e+01  2.551e+01  -0.465  0.64188
## price_share              -3.512e+03  1.966e+03  -1.787  0.07415 .
## w_pop                    -1.515e+03  2.687e+03  -0.564  0.57282
## w_den                     3.764e+03  6.468e+03   0.582  0.56063
## w_txn                     9.883e+00  4.765e+01   0.207  0.83572
## w_comp                   -5.239e+02  9.337e+01  -5.612 2.27e-08 ***
## w_unis                    3.669e+02  5.166e+01   7.103 1.67e-12 ***
## w_ddrop_std               1.723e+01  5.101e+02   0.034  0.97307
## w_ringshare_std          -3.070e+02  4.698e+01  -6.534 8.01e-11 ***
## w_ratio_std              -6.899e+01  5.345e+01  -1.291  0.19690
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1109 on 2096 degrees of freedom
## Multiple R-squared:  0.3877, Adjusted R-squared:  0.3798
## F-statistic: 49.15 on 27 and 2096 DF,  p-value: < 2.2e-16
```

```
cat(glue::glue("
RMSE = {round(rmse,2)}
MAE  = {round(mae,2)}
R2_ols = {round(summary(m_ols)$r.squared,2)}
adj_R2_ols = {round(summary(m_ols)$adj.r.squared,2)}
"))
```

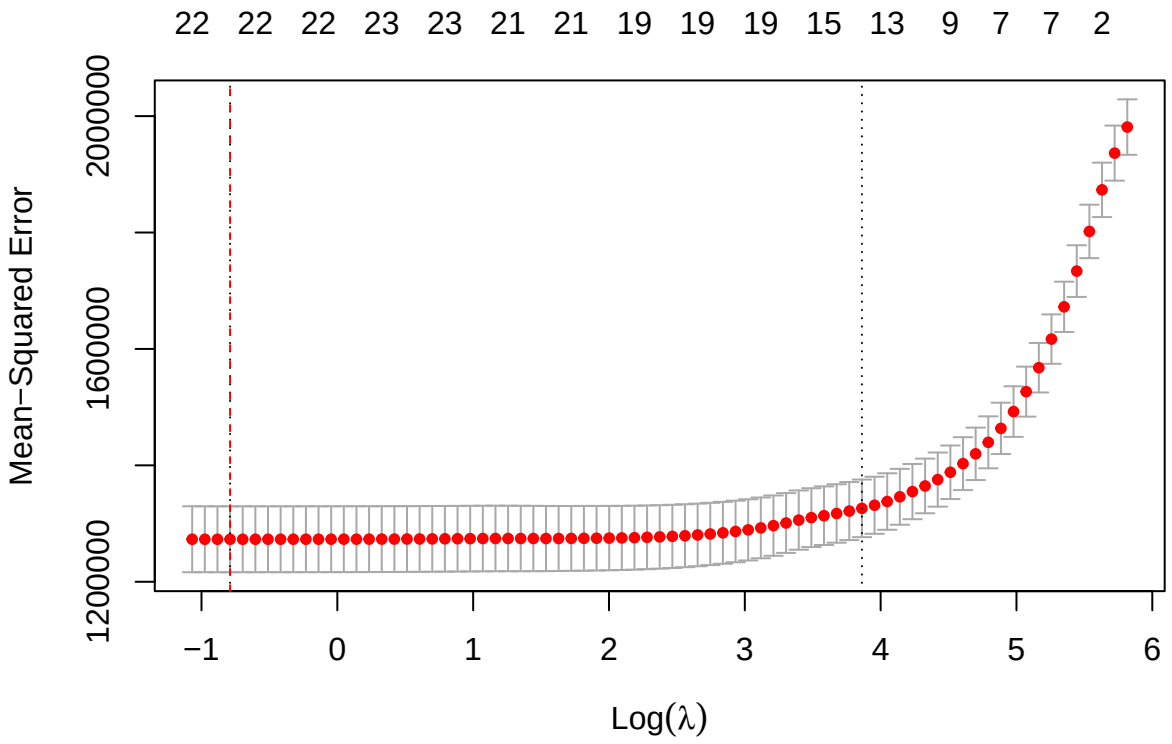
```
## RMSE = 1168.97
## MAE  = 886.41
## R2_ols = 0.39
## adj_R2_ols = 0.38
```

Lasso

```
set_train_lasso <- set_train_ols
set_test_lasso <- set_test_ols

X_train <- set_train_lasso %>%
  select(where(is.numeric)) %>%
  select(-TotalMembers) %>%
  as.matrix()
y_train <- set_train_lasso$TotalMembers

X_test <- set_test_lasso %>%
  select(where(is.numeric)) %>%
  select(-TotalMembers) %>%
  as.matrix()
y_test  <- set_test_lasso$TotalMembers

# Select  by cross - validation
set.seed(2025)
cv_lasso <- cv.glmnet(
  x = X_train,
  y = y_train,
  alpha = 1,
  nfolds = 10,
  type.measure = "mse"
)
# View the optimal lambda
best_lambda <- cv_lasso$lambda.min
cat("The best  = ", best_lambda, "\n")
```

```
## The best  =  0.454046
```

```
# Visualize the CV curve
plot(cv_lasso)
abline(v = log(best_lambda), col = "red", lty = 2)
```




```r
# Fit Lasso with the optimal
m_lasso <- glmnet(
  x = X_train,
  y = y_train,
  alpha  = 1,
  lambda = best_lambda
)

# Predict and evaluate on the test set
pred_lasso <- predict(m_lasso, newx = X_test, s = best_lambda)
rmse_lasso <- sqrt(mean((pred_lasso - y_test)^2))
mae_lasso  <- mean(abs(pred_lasso - y_test))
sst_lasso <- sum((y_test - mean(y_test))^2)
sse_lasso <- sum((y_test - pred_lasso)^2)
R2_lasso <- 1 - sse_lasso / sst_lasso

n <- length(y_test)
p <- length(coef(m_lasso)) - 1
adjR2_lasso <- 1 - (1 - R2_lasso) * (n - 1) / (n - p - 1)

cat(sprintf("
Lasso Regression ( =%.5f) Test Set Evaluation
  - RMSE = %.2f
  - MAE  = %.2f
  - R2   = %.4f
  - Adjusted R2 = %.4f
", best_lambda, rmse_lasso, mae_lasso, R2_lasso, adjR2_lasso))
```

```
##
## Lasso Regression ( =0.45405) Test Set Evaluation
##   - RMSE = 1182.54
```

```
##   - MAE  = 895.83
##   - R2   = 0.3061
##   - Adjusted R2 = 0.2817
```

JAGS

```
# AvgAccountPayment
mean_AAP <- mean(set_train$AvgAccountPayment, na.rm = TRUE)
sd_AAP <- sd(set_train$AvgAccountPayment, na.rm = TRUE)

# AvgJoiningFee
mean_AJF <- mean(set_train$AvgJoiningFee, na.rm = TRUE)
sd_AJF <- sd(set_train$AvgJoiningFee, na.rm = TRUE)

# distance_weighted_spend
mean_dws <- mean(set_train$distance_weighted_spend, na.rm = TRUE)
sd_dws <- sd(set_train$distance_weighted_spend, na.rm = TRUE)

# avg_spend_per_user
mean_aspu <- mean(set_train$avg_spend_per_user, na.rm = TRUE)
sd_aspu <- sd(set_train$avg_spend_per_user, na.rm = TRUE)

df_jags <- set_train %>%
  mutate(
    y = log(TotalMembers),
    AAP = (AvgAccountPayment - mean_AAP) / sd_AAP,
    AJF = (AvgJoiningFee - mean_AJF) / sd_AJF,
    dws = (distance_weighted_spend - mean_dws) / sd_dws,
    urepeat = user_repeat_rate,
    u1 = UMAP2D_1,
    u2 = UMAP2D_2,
    competitor = CompetitorIndex,
    aspu = (avg_spend_per_user - mean_aspu) / sd_aspu,
    ampu = avg_merchant_per_user,
    parking_num = as.integer(GymParking == "Parking"),
    lat = Latitude,
    lon = Longitude,
    userd = user_density,
    w_ddrop = w_ddrop_std,
    w_unis = w_unis,
    promo = PromoPercentage,
    w_ringshare = w_ringshare_std,
    w_ratio = w_ratio_std,
    w_den = w_den,
    w_txn = w_txn,
    w_comp = w_comp,

    group = as.integer(GymSiteType)
  )


# data for jags
data_jags <- list(
```

```
  N = nrow(df_jags),
  J = nlevels(df_jags$GymSiteType),
  y = df_jags$y,
  AAP = df_jags$AAP,
  AJF = df_jags$AJF,
  dws = df_jags$dws,
  urepeat = df_jags$urepeat,
  u1 = df_jags$u1,
  u2 = df_jags$u2,
  competitor = df_jags$competitor,
  aspu = df_jags$aspu,
  ampu = df_jags$ampu,
  parking = df_jags$parking_num,
  lat = df_jags$lat,
  lon = df_jags$lon,
  userd = df_jags$userd,
  w_ddrop = df_jags$w_ddrop,
  w_unis = df_jags$w_unis,
  promo = df_jags$promo,
  w_ringshare = df_jags$w_ringshare,
  w_ratio = df_jags$w_ratio,
  w_den = df_jags$w_den,
  w_txn = df_jags$w_txn,
  w_comp = df_jags$w_comp,

  group      = df_jags$group,
  tau_mu     = 1,
  tau_beta   = 1,
  sigma_max  = 1000
)


m_hier <- "
model {
  for(i in 1:N) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha[group[i]]
             + beta_AAP * AAP[i]
             + beta_AJF * AJF[i]
             + beta_dws * dws[i]
             + beta_urepeat * urepeat[i]
             + beta_u1 * u1[i]
             + beta_u2 * u2[i]
             + beta_comp * competitor[i]
             + beta_aspu * aspu[i]
             + beta_ampu * ampu[i]
             + beta_parking * parking[i]
             + beta_lat * lat[i]
             + beta_lon * lon[i]
             + beta_userd * userd[i]
             + beta_w_ddrop * w_ddrop[i]
             + beta_w_unis * w_unis[i]
             + beta_promo * promo[i]
```

```
            + beta_w_ringshare * w_ringshare[i]
            + beta_w_ratio * w_ratio[i]
            + beta_w_den * w_den[i]
            + beta_w_txn * w_txn[i]
            + beta_w_comp * w_comp[i]
  }

  #
  for(j in 1:J) {
    alpha[j] ~ dnorm(mu_alpha, tau_mu)
  }
  #
  mu_alpha ~ dnorm(0, tau_mu)

  beta_AAP ~ dnorm(0, tau_beta)
  beta_AJF ~ dnorm(0, tau_beta)
  beta_dws ~ dnorm(0, tau_beta)
  beta_urepeat ~ dnorm(0, tau_beta)
  beta_u1 ~ dnorm(0, tau_beta)
  beta_u2 ~ dnorm(0, tau_beta)
  beta_comp ~ dnorm(0, tau_beta)
  beta_aspu ~ dnorm(0, tau_beta)
  beta_ampu ~ dnorm(0, tau_beta)
  beta_parking ~ dnorm(0, tau_beta)
  beta_lat ~ dnorm(0, tau_beta)
  beta_lon ~ dnorm(0, tau_beta)
  beta_userd ~ dnorm(0, tau_beta)
  beta_w_ddrop ~ dnorm(0, tau_beta)
  beta_w_unis ~ dnorm(0, tau_beta)
  beta_promo ~ dnorm(0, tau_beta)
  beta_w_ringshare ~ dnorm(0, tau_beta)
  beta_w_ratio ~ dnorm(0, tau_beta)
  beta_w_den ~ dnorm(0, tau_beta)
  beta_w_txn ~ dnorm(0, tau_beta)
  beta_w_comp ~ dnorm(0, tau_beta)

  tau_alpha <- pow(sigma_alpha, -2)
  sigma_alpha ~ dunif(0, sigma_max)
  tau <- pow(sigma, -2)
  sigma ~ dunif(0, sigma_max)
}
"

# Initial value
ini_value <- function(){
  list(
    mu_alpha = rnorm(1,0,1),
    alpha = rnorm(data_jags$J,0,1),

    beta_AAP = dnorm(0, 1),
    beta_AJF = dnorm(0, 1),
    beta_dws = dnorm(0, 1),
    beta_urepeat = rnorm(1),
```

```
    beta_u1 = rnorm(1),
    beta_u2 = rnorm(1),
    beta_comp = rnorm(1),
    beta_aspu = rnorm(1),
    beta_ampu = rnorm(1),
    beta_parking = rnorm(1),
    beta_lat = rnorm(1),
    beta_lon = rnorm(1),
    beta_userd = rnorm(1),
    beta_w_ddrop = rnorm(1),
    beta_w_unis = rnorm(1),
    beta_promo = rnorm(1),
    beta_w_ringshare = rnorm(1),
    beta_w_ratio = rnorm(1),
    beta_w_den  = rnorm(1),
    beta_w_txn  = rnorm(1),
    beta_w_comp = rnorm(1),

    sigma = runif(1,0,10),
    sigma_alpha = runif(1,0,10)
  )
}


# sample
set.seed(2025)
m_jags <- jags.model(textConnection(m_hier),
                  data = data_jags,
                  inits = ini_value,
                  n.chains = 3, quiet=TRUE)
update(m_jags, 4000)

pars_jags <- c("mu_alpha","alpha",
               "beta_AAP","beta_AJF", "beta_dws",
               "beta_urepeat","beta_u1","beta_u2","beta_comp",
               "beta_aspu","beta_ampu","beta_parking","beta_lat",
               "beta_lon", "beta_userd", "beta_w_ddrop","beta_w_unis",
               "beta_promo","beta_w_ringshare", "beta_w_ratio", "beta_w_den",
               "beta_w_txn", "beta_w_comp", "sigma","sigma_alpha")
mcmc_jags <- coda.samples(m_jags, pars_jags, n.iter=10000)

# 5.1 Gelman-Rubin R-hat & ESS
gel_full <- gelman.diag(mcmc_jags, multivariate = FALSE)
print(gel_full)
```

```
## Potential scale reduction factors:
##
##                  Point est. Upper C.I.
## alpha[1]               1.04       1.08
## alpha[2]               1.04       1.08
## alpha[3]               1.04       1.08
## beta_AAP               1.00       1.00
## beta_AJF               1.00       1.00
## beta_ampu              1.04       1.08
```

```
## beta_aspu              1.01       1.01
## beta_comp              1.01       1.02
## beta_dws               1.00       1.00
## beta_lat               1.27       2.00
## beta_lon               1.02       1.06
## beta_parking           1.00       1.00
## beta_promo             1.01       1.01
## beta_u1                1.02       1.06
## beta_u2                1.04       1.11
## beta_urepeat           1.16       1.52
## beta_userd             1.02       1.06
## beta_w_comp            1.01       1.02
## beta_w_ddrop           1.00       1.02
## beta_w_den             1.01       1.02
## beta_w_ratio           1.01       1.05
## beta_w_ringshare       1.00       1.01
## beta_w_txn             1.00       1.00
## beta_w_unis            1.00       1.00
## mu_alpha               1.02       1.04
## sigma                  1.01       1.02
## sigma_alpha            1.00       1.00
```

```
# If all R-hats are  1.05 and the ESS is large enough, it is considered to have good convergence.
# DIC
dic <- dic.samples(m_jags, type="pD", n.iter=4000)
cat("DIC =", sum(dic$deviance)+sum(dic$penalty), "\n")
```

```
## DIC = 858.2024
```

```
# Posterior mean prediction, RMSE, MAE
invisible({
# Extract the posterior mean
poster <- suppressWarnings(summary(mcmc_jags)$statistics)


a <- poster[grep("^alpha\\[", rownames(poster)), "Mean"]
b <- function(x) poster[x, "Mean"]

# Build the linear predicted value mu
mu_hat <- a[data_jags$group] +
  b("beta_AAP") * data_jags$AAP +
  b("beta_AJF") * data_jags$AJF +
  b("beta_dws") * data_jags$dws +
  b("beta_urepeat") * data_jags$urepeat +
  b("beta_u1") * data_jags$u1 +
  b("beta_u2") * data_jags$u2 +
  b("beta_comp") * data_jags$competitor +
  b("beta_aspu") * data_jags$aspu +
  b("beta_ampu") * data_jags$ampu +
  b("beta_parking") * data_jags$parking +
  b("beta_lat") * data_jags$lat +
  b("beta_lon") * data_jags$lon +
  b("beta_userd") * data_jags$userd +
  b("beta_w_ddrop") * data_jags$w_ddrop +
  b("beta_w_unis") * data_jags$w_unis +
```

```
  b("beta_promo") * data_jags$promo +
  b("beta_w_ringshare") * data_jags$w_ringshare +
  b("beta_w_ratio") * data_jags$w_ratio +
  b("beta_w_den") * data_jags$w_den +
  b("beta_w_txn") * data_jags$w_txn +
  b("beta_w_comp") * data_jags$w_comp

# Restore the predicted values to the original space
TotalMembers_hat <- exp(mu_hat)
y_true <- exp(data_jags$y)

# Evaluation indicators
rmse_jags <- sqrt(mean((TotalMembers_hat - y_true)^2))
mae_jags  <- mean(abs(TotalMembers_hat - y_true))
})

cat(sprintf("
Model Metrics:
RMSE_jags = %.2f
MAE_jags  = %.2f
", rmse_jags, mae_jags))
```

```
##
## Model Metrics:
## RMSE_jags = 1166.40
## MAE_jags  = 921.83
```

```
# R2
# True values & JAGS predicted values
pred_jags <- TotalMembers_hat

# Sample size \(n\) and the number of fixed effects \(p\) (excluding the random intercept and the inter
n <- length(y_true)
p <- 21

# SSE & SST
SSE_jags <- sum((pred_jags - y_true)^2)
SST      <- sum((y_true - mean(y_true))^2)

R2_jags <- 1 - SSE_jags / SST
adjR2_jags <- 1 - (1 - R2_jags) * (n - 1) / (n - p - 1)

cat(sprintf("JAGS model R2v= %.4f\n", R2_jags))
```

```
## JAGS model R2v= 0.3136
```

```
cat(sprintf("JAGS model Adjusted R2 = %.4f\n", adjR2_jags))
```

```
## JAGS model Adjusted R2 = 0.3067
```

jags Research on Variable Influence

```
poster <- summary(mcmc_jags)$statistics
```

```r
# Only take the main slope variables (starting with beta_)
sel <- grep("^beta_", rownames(poster))
coefs <- poster[sel, c("Mean", "SD")]
coefs <- as.data.frame(coefs)
coefs$Var <- rownames(coefs)
coefs$AbsMean <- abs(coefs$Mean)

# Sort and display
coefs_sorted <- coefs[order(-coefs$AbsMean), ]
print(coefs_sorted[, c("Var", "Mean", "SD", "AbsMean")], digits = 3)
```

```
##                              Var      Mean       SD  AbsMean
## beta_ampu               beta_ampu  1.182417 0.470565 1.182417
## beta_urepeat         beta_urepeat -0.494445 1.011179 0.494445
## beta_promo             beta_promo -0.406947 0.062339 0.406947
## beta_w_den             beta_w_den  0.145432 0.025996 0.145432
## beta_w_comp           beta_w_comp -0.127641 0.024923 0.127641
## beta_parking         beta_parking  0.116760 0.022712 0.116760
## beta_w_ringshare beta_w_ringshare -0.077898 0.013076 0.077898
## beta_w_unis           beta_w_unis  0.063174 0.013512 0.063174
## beta_w_ddrop         beta_w_ddrop  0.046175 0.012952 0.046175
## beta_aspu               beta_aspu -0.042313 0.019561 0.042313
## beta_AAP                 beta_AAP  0.031864 0.008191 0.031864
## beta_dws                 beta_dws  0.030572 0.015416 0.030572
## beta_w_txn             beta_w_txn  0.022545 0.012508 0.022545
## beta_w_ratio         beta_w_ratio -0.021750 0.011488 0.021750
## beta_lat                 beta_lat  0.014101 0.017799 0.014101
## beta_AJF                 beta_AJF  0.007298 0.007839 0.007298
## beta_lon                 beta_lon  0.006692 0.007900 0.006692
## beta_u2                   beta_u2 -0.005970 0.004671 0.005970
## beta_comp               beta_comp -0.005724 0.001267 0.005724
## beta_u1                   beta_u1 -0.005543 0.002726 0.005543
## beta_userd             beta_userd -0.000147 0.000489 0.000147
```

SPDE

```r
df_spde <- df_jags

# Extract the latitude and longitude matrix
coords <- as.matrix(df_spde[, c("lon", "lat")])

# Build a triangular mesh
mesh <- inla.mesh.2d(
  loc = coords,
  max.edge = c(0.1, 0.5),
  cutoff = 0.01,
  offset = c(0.1, 0.2)
)

# Define the SPDE Matern prior
spde <- inla.spde2.pcmatern(
  mesh = mesh,
  prior.range = c(0.1, 0.5),
```

```
  prior.sigma = c(1, 0.01)
)

# Projection matrix A
A <- inla.spde.make.A(mesh, loc = coords)

# Construct INLA stack (fixed effects + spatial random field)
df_spde$GymSiteType <- factor(df_spde$GymSiteType,
                              levels = c("Hybrid", "Residential", "Workforce"))
stk <- inla.stack(
  data  = list(y = df_spde$y),
  A  = list(1, A),
  effects = list(
    data.frame(
      intercept = 1,
      GymSiteType = df_spde$GymSiteType,
      AAP = df_spde$AAP,
      AJF = df_spde$AJF,
      dws = df_spde$dws,
      urepeat = df_spde$urepeat,
      u1 = df_spde$u1,
      u2 = df_spde$u2,
      competitor = df_spde$competitor,
      aspu = df_spde$aspu,
      ampu = df_spde$ampu,
      parking_num = df_spde$parking_num,
      lat = df_spde$lat,
      lon = df_spde$lon,
      userd = df_spde$userd,
      w_ddrop = df_spde$w_ddrop,
      w_unis = df_spde$w_unis,
      promo = df_spde$promo,
      w_ringshare = df_spde$w_ringshare,
      w_ratio = df_spde$w_ratio,
      w_den = df_spde$w_den,
      w_txn = df_spde$w_txn,
      w_comp = df_spde$w_comp
    ),
    field = 1:spde$n.spde
  ),
  tag = "est"
)

# Spatial Model Formula
formula_spde <- y ~ -1 + intercept +
  GymSiteType * AAP +
  AJF + dws + urepeat + u1 + u2 + competitor +
  aspu + ampu + parking_num + lat + lon + userd +
  w_ddrop + w_unis + promo + w_ringshare + w_ratio +
  w_den + w_txn + w_comp + f(field, model = spde)

# Spatial Model Formula
res_spde <- inla(
```

```
  formula = formula_spde,
  data = inla.stack.data(stk),
  family = "gaussian",
  control.predictor= list(
    A = inla.stack.A(stk),
    compute = TRUE
  ),
  control.compute  = list(
    dic  = TRUE,
    waic = TRUE,
    cpo  = TRUE
  )
)

cat(sprintf("DIC=%.2f  WAIC=%.2f\n",
            res_spde$dic$dic, res_spde$waic$waic))
```

```
## DIC=-3194.83  WAIC=6055.02
```

```
# Compare prediction performance RMSE/MAE
# True TotalMembers
y_true <- exp(df_spde$y)

# Take out the predicted indices of the "est" part in the stack
idx_est <- inla.stack.index(stk, "est")$data

# Extract the corresponding predictions and restore them to the original space
pred_spde_obs <- exp(res_spde$summary.fitted.values$mean[idx_est])

rmse_spde <- sqrt(mean((pred_spde_obs - y_true)^2))
mae_spde  <- mean(abs(pred_spde_obs - y_true))

cat(sprintf("Spatial model RMSE_spde = %.2f, MAE_spde = %.2f\n", rmse_spde, mae_spde))
```

```
## Spatial model RMSE_spde = 425.31, MAE_spde = 273.53
```

```
# R2
p <- length(res_spde$summary.fixed$mean) - 1
n <- length(y_true)

#  SSE & SST
SSE <- sum((y_true - pred_spde_obs)^2)
SST <- sum((y_true - mean(y_true))^2)

R2 <- 1 - SSE / SST
adjR2 <- 1 - (1 - R2) * (n - 1) / (n - p - 1)

cat(sprintf("R2 = %.4f\n", R2))
```

```
## R2 = 0.9087
```

```
cat(sprintf("adjR2 = %.4f\n", adjR2))
```

```
## adjR2 = 0.9076
```

Verify spde on the test set

```
df_spde_test <- set_test %>%
  mutate(
    y = log(TotalMembers),
    AAP = (AvgAccountPayment - mean_AAP) / sd_AAP,
    AJF = (AvgJoiningFee - mean_AJF) / sd_AJF,
    dws = (distance_weighted_spend - mean_dws) / sd_dws,
    urepeat = user_repeat_rate,
    u1 = UMAP2D_1,
    u2 = UMAP2D_2,
    competitor = CompetitorIndex,
    aspu = (avg_spend_per_user - mean_aspu) / sd_aspu,
    ampu = avg_merchant_per_user,
    parking_num = as.integer(GymParking == "Parking"),
    lat = Latitude,
    lon = Longitude,
    userd = user_density,
    w_ddrop = w_ddrop_std,
    w_unis = w_unis,
    promo = PromoPercentage,
    w_ringshare = w_ringshare_std,
    w_ratio = w_ratio_std,
    w_den = w_den,
    w_txn = w_txn,
    w_comp = w_comp,
    GymSiteType = factor(GymSiteType,levels = c("Hybrid", "Residential", "Workforce")),
    group = as.integer(GymSiteType)
  )
df_spde_test$intercept <- 1

# Prepare test set coordinates & covariates
coords_test <- as.matrix(df_spde_test[, c("lon","lat")])

# Construct the projection matrix A_test
A_test <- inla.spde.make.A(mesh, loc = coords_test)

# Extract the posterior means of the fixed effects and the spatial random fields
# Fixed effect mean
beta_mean <- res_spde$summary.fixed$mean

# Mean of the spatial random field (field on nodes)
field_mean <- res_spde$summary.random$field$mean

# Build a linear prediction mu_test
# Fixed effects part
Xtest <- model.matrix(
  ~ -1 + intercept + GymSiteType * AAP + AJF + dws + urepeat + u1 + u2 + competitor +
    aspu + ampu + parking_num + lat + lon + userd +
    w_ddrop + w_unis + promo + w_ringshare + w_ratio +
    w_den + w_txn + w_comp,
  data = df_spde_test
)
```

```r
mu_fixed <- as.vector(Xtest %*% beta_mean)

# Spatial random field part: Interpolation at test points
mu_spatial <- as.vector(A_test %*% field_mean)

# Synthesize log predicted values
mu_test <- mu_fixed + mu_spatial

# Restore to the original TotalMembers space
pred_test <- exp(mu_test)
actual_test <- df_spde_test$TotalMembers

# RMSE & MAE
rmse_test <- sqrt(mean((pred_test - actual_test)^2))
mae_test  <- mean(abs(pred_test - actual_test))

# R2
n_test <- length(actual_test)
p      <- length(beta_mean) - 1

SSE_test <- sum((actual_test - pred_test)^2)
SST_test <- sum((actual_test - mean(actual_test))^2)

R2_test    <- 1 - SSE_test / SST_test
adjR2_test <- 1 - (1 - R2_test) * (n_test - 1) / (n_test - p - 1)

cat(sprintf("Test set performance:\n"))
```

```
## Test set performance:
```

```r
cat(sprintf("RMSE_spde_test = %.2f\n", rmse_test))
```

```
## RMSE_spde_test = 491.72
```

```r
cat(sprintf("MAE_spde_test = %.2f\n", mae_test))
```

```
## MAE_spde_test = 306.41
```

```r
cat(sprintf("R2_spde_test = %.4f\n", R2_test))
```

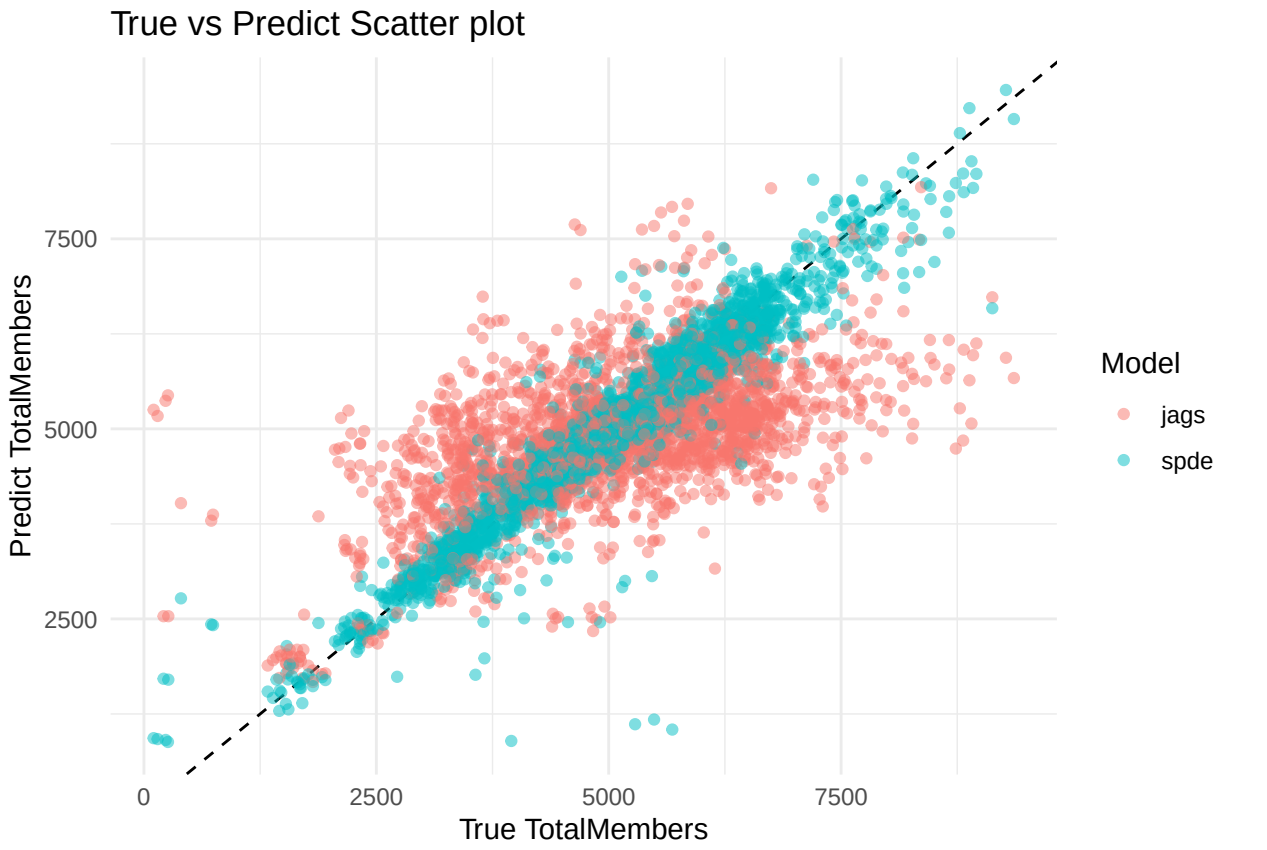
```
## R2_spde_test = 0.8800
```

```r
cat(sprintf("Adjusted R2_spde_test= %.4f\n", adjR2_test))
```

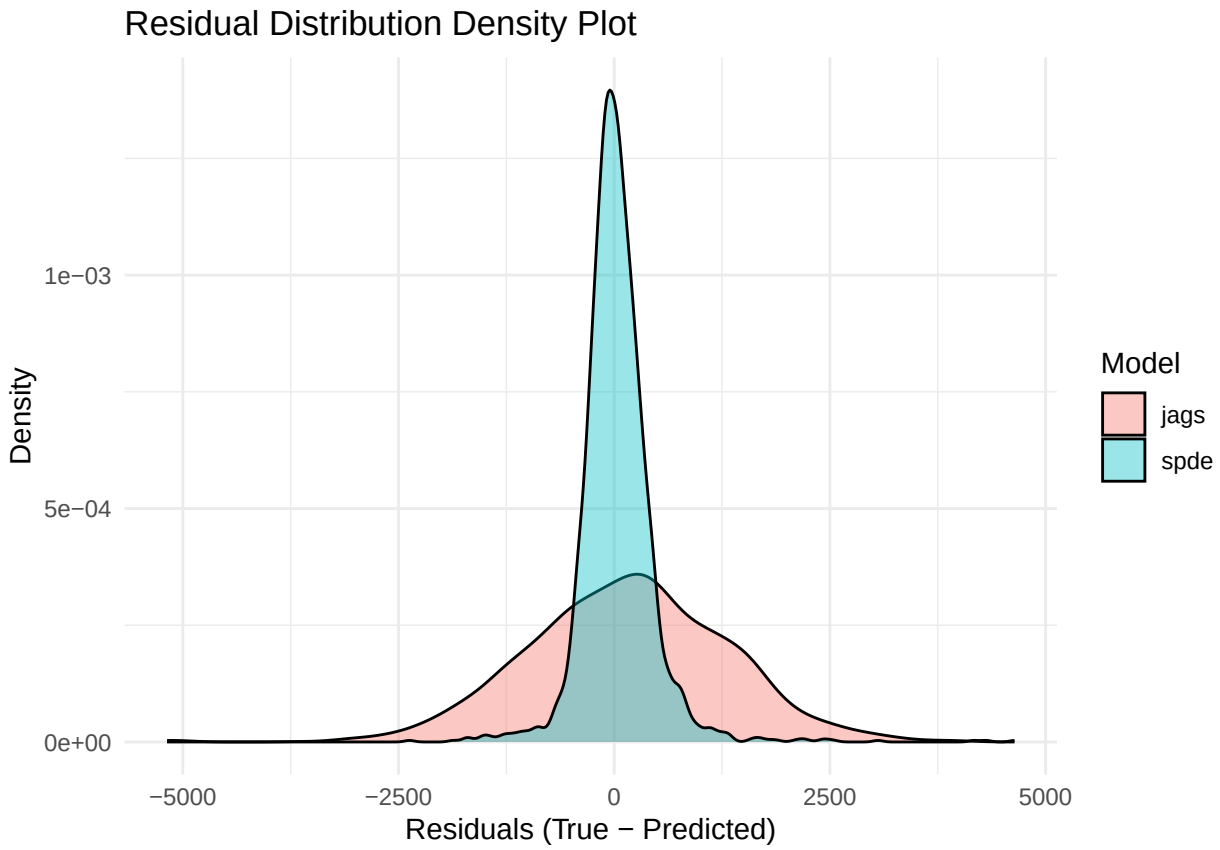
```
## Adjusted R2_spde_test= 0.8754
```

```r
# True values of the training set
y_true      <- exp(df_jags$y)
# JAGS predicted value
pred_jags   <- TotalMembers_hat
# SPDE predicted value
pred_spde   <- pred_spde_obs
```

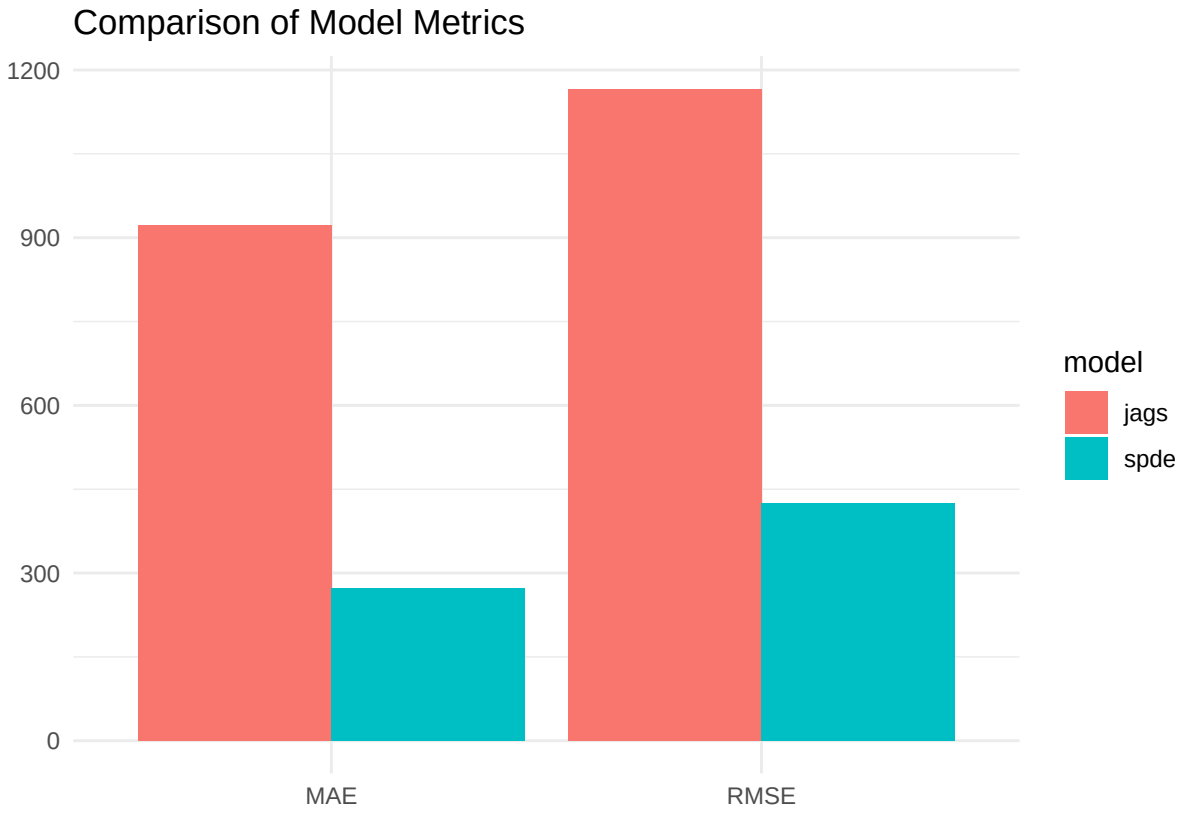
```r
df_compare <- tibble(
  actual    = y_true,
  jags      = pred_jags,
  spde      = pred_spde
) %>%
  pivot_longer(cols = c(jags, spde),
               names_to  = "model",
               values_to = "predicted") %>%
  mutate(
    residual = actual - predicted
  )

metrics <- df_compare %>%
  group_by(model) %>%
  summarize(
    RMSE = sqrt(mean((predicted - actual)^2)),
    MAE  = mean(abs(predicted - actual)),
    R2   = 1 - sum((actual - predicted)^2) / sum((actual - mean(actual))^2),
    .groups = "drop"
  ) %>%
  pivot_longer(
    cols = -model,
    names_to = "metric",
    values_to = "value"
  )

p_scatter <- ggplot(df_compare, aes(x = actual, y = predicted, color = model)) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed") +
  geom_point(alpha = 0.5) +
  labs(x = "True TotalMembers", y = "Predict TotalMembers",
       title = "True vs Predict Scatter plot",
       color = "Model") +
  theme_minimal()

p_resid <- ggplot(df_compare, aes(x = residual, fill = model)) +
  geom_density(alpha = 0.4) +
  labs(x = "Residuals (True - Predicted)", y = "Density",
       title = "Residual Distribution Density Plot",
       fill = "Model") +
  theme_minimal()

metrics_plot <- metrics %>%
  filter(metric %in% c("RMSE", "MAE"))

p_metrics <- ggplot(metrics_plot, aes(x = metric, y = value, fill = model)) +
  geom_col(position = "dodge") +
  labs(x = "", y = "", title = "Comparison of Model Metrics") +
  theme_minimal()

p_scatter
```


True vs Predict Scatter plot
Predict TotalMembers
7500
5000
2500
0
2500
5000
7500
True TotalMembers
Model
jags
spde

```
p_resid
```


Residual Distribution Density Plot
Density
1e−03
5e−04
0e+00
−5000
−2500
0
2500
5000
Residuals (True − Predicted)
Model
jags
spde

```
p_metrics
```


Comparison of Model Metrics
1200
900
600
300
0
MAE
RMSE
model
jags
spde

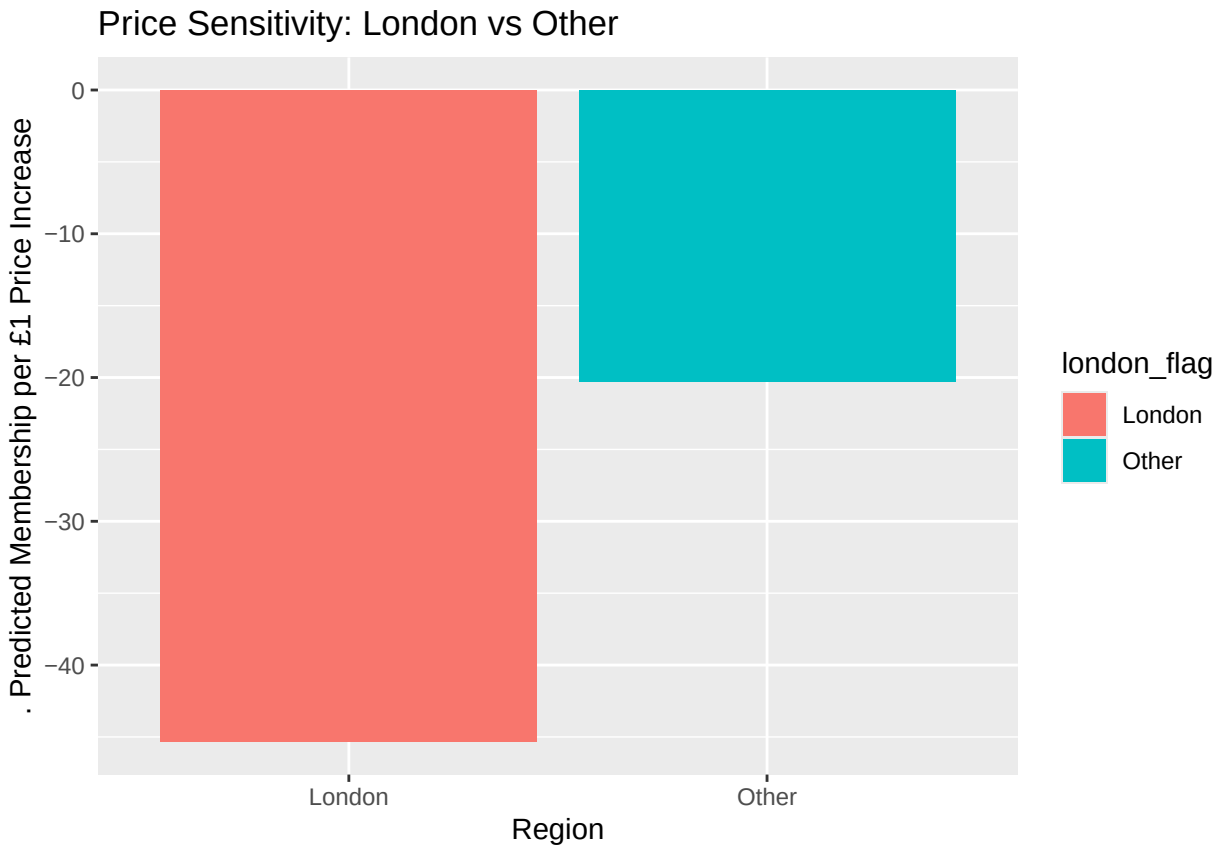
How sensitive is membership demand to price changes in each location?

```
delta_AAP <- 1 / sd_AAP
london_center_lat <- 51.509865
london_center_lon <- -0.118092
london_radius_km <- 25

haversine <- function(lat1, lon1, lat2, lon2) {
  R <- 6371
  delta_lat <- (lat2 - lat1) * pi / 180
  delta_lon <- (lon2 - lon1) * pi / 180
  a <- sin(delta_lat / 2)^2 + cos(lat1 * pi / 180) * cos(lat2 * pi / 180) * sin(delta_lon / 2)^2
  c <- 2 * atan2(sqrt(a), sqrt(1 - a))
  R * c
}

# Construct a new price variable (AAP + delta_AAP)
df_spde$intercept <- 1
df_spde$AAP_new <- df_spde$AAP + delta_AAP

# Use the new AAP to predict the number of members after "the price is increased by 1 pound"
X_new <- model.matrix(
  ~ -1 + intercept + GymSiteType * AAP_new + AJF + dws + urepeat + u1 + u2 + competitor +
    aspu + ampu + parking_num + lat + lon + userd +
    w_ddrop + w_unis + promo + w_ringshare + w_ratio +
    w_den + w_txn + w_comp,
  data = df_spde
)

coords <- as.matrix(df_spde[, c("lon", "lat")])
A <- inla.spde.make.A(mesh, loc = coords)
beta_mean <- res_spde$summary.fixed$mean
field_mean <- res_spde$summary.random$field$mean

mu_fixed_new <- as.vector(X_new %*% beta_mean)
mu_spatial <- as.vector(A %*% field_mean)
mu_pred_new <- mu_fixed_new + mu_spatial
pred_new <- exp(mu_pred_new)

# Merge into data
df_sens <- df_spde %>%
  mutate(
    price_effect = pred_new - TotalMembers,
    price_elasticity = 100 * (pred_new - TotalMembers) / TotalMembers,
    GymSiteType = GymSiteType,
    london_flag = ifelse(
      haversine(lat, lon, london_center_lat, london_center_lon) <= london_radius_km,
      "London", "Other"
    )
  )

# Grouping and Visualization
```

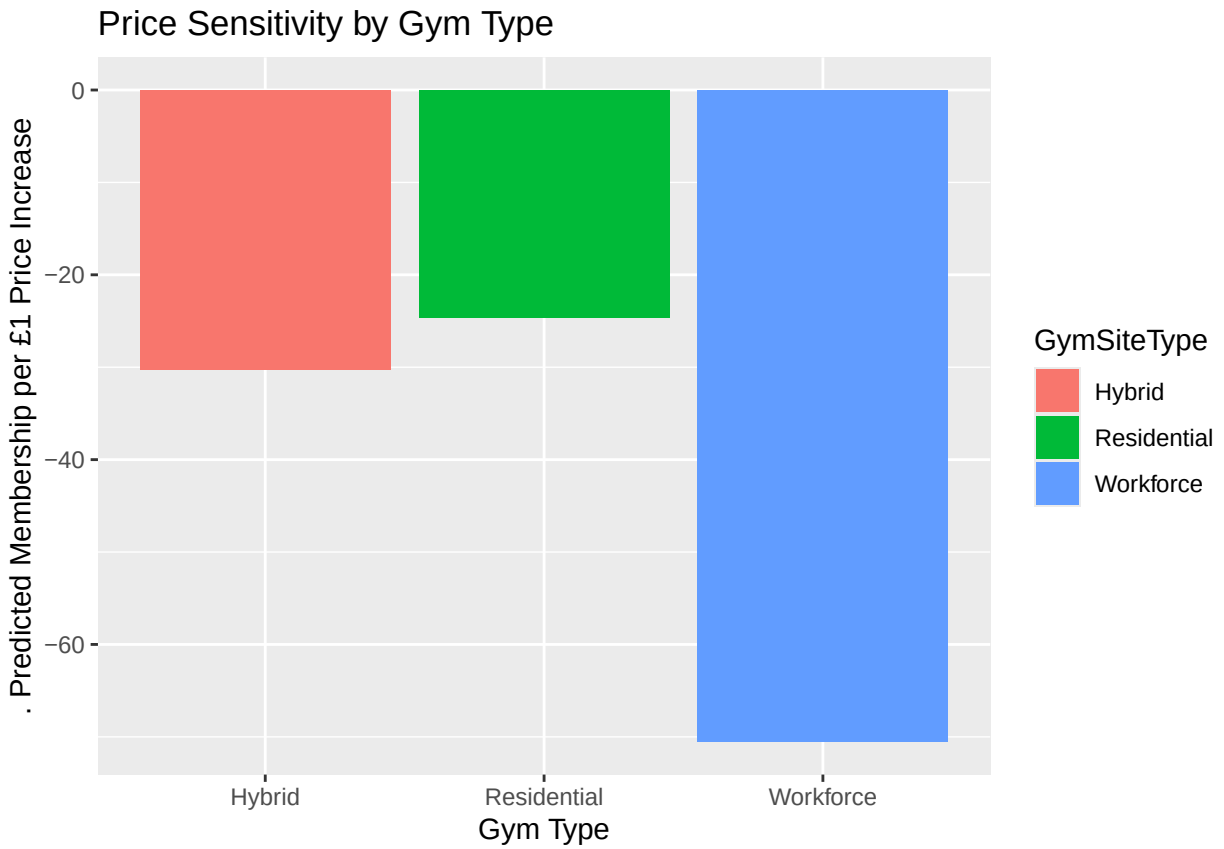
```r
summary_region <- df_sens %>%
  group_by(london_flag) %>%
  summarize(
    avg_effect = mean(price_effect),
    avg_elasticity = mean(price_elasticity),
    weighted_elasticity = sum(price_effect) / sum(TotalMembers) * 100,
    n = n()
  )

summary_type <- df_sens %>%
  group_by(GymSiteType) %>%
  summarize(
    avg_effect = mean(price_effect),
    avg_elasticity = mean(price_elasticity),
    weighted_elasticity = sum(price_effect) / sum(TotalMembers) * 100,
    n = n()
  )

summary_type_region <- df_sens %>%
  group_by(GymSiteType, london_flag) %>%
  summarize(
    avg_effect = mean(price_effect),
    avg_elasticity = mean(price_elasticity),
    weighted_elasticity = sum(price_effect) / sum(TotalMembers) * 100,
    n = n()
  )
```

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

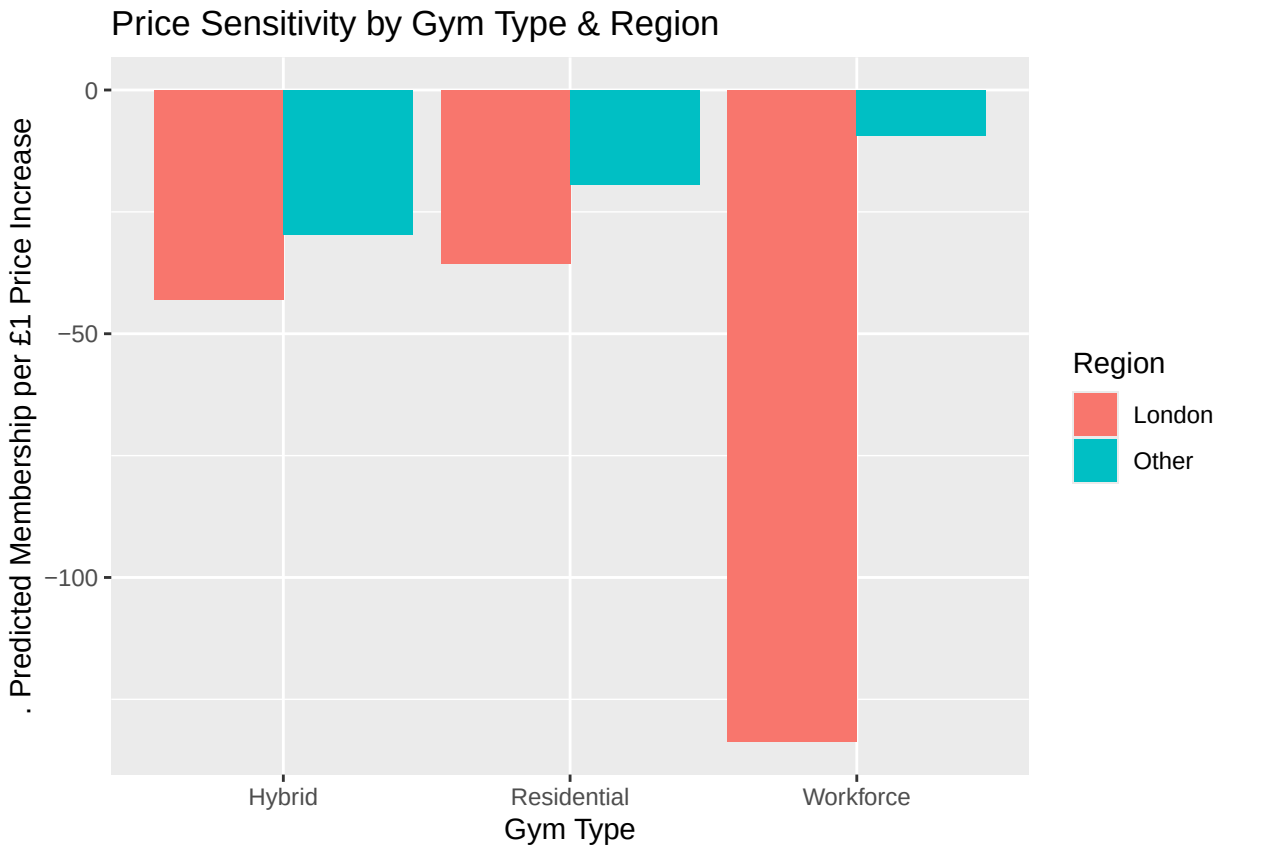
```r
# Visualization
ggplot(summary_region, aes(x = london_flag, y = avg_effect, fill = london_flag)) +
  geom_col() +
  labs(title = "Price Sensitivity: London vs Other",
       x = "Region", y = "Δ Predicted Membership per £1 Price Increase")
```




```
ggplot(summary_type, aes(x = GymSiteType, y = avg_effect, fill = GymSiteType)) +
  geom_col() +
  labs(title = "Price Sensitivity by Gym Type",
       x = "Gym Type", y = "Δ Predicted Membership per £1 Price Increase")
```




```
ggplot(summary_type_region, aes(x = GymSiteType, y = avg_effect, fill = london_flag)) +
  geom_col(position = "dodge") +
  labs(title = "Price Sensitivity by Gym Type & Region",
       x = "Gym Type", y = "Δ Predicted Membership per £1 Price Increase", fill = "Region")
```



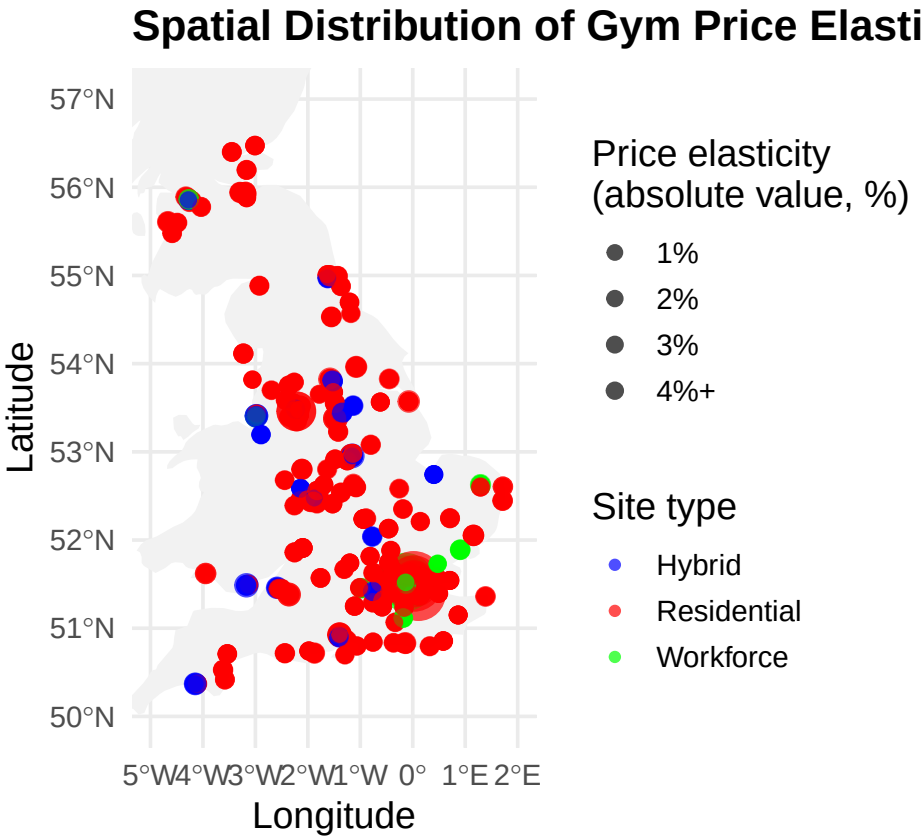

```
kable(
  summary_type_region,
  digits = 2,
  caption = "Average and Weighted Price Elasticity by Gym Type and Region",
  col.names = c("GymSiteType", "London/Other",
                "Avg ΔMembers", "Avg Elasticity (%)",
                "Weighted Elasticity (%)", "n")
)
```

Table 1: Average and Weighted Price Elasticity by Gym Type and Region

GymSiteType	London/Other	Avg ΔMembers	Avg Elasticity (%)	Weighted Elasticity (%)	n
Hybrid	London	-43.03	1.61	-0.92	9
Hybrid	Other	-29.59	0.16	-0.62	179
Residential	London	-35.58	3.70	-0.62	577
Residential	Other	-19.49	0.49	-0.39	1229
Workforce	London	-133.75	16.29	-4.72	64
Workforce	Other	-9.28	0.63	-0.24	66

```
uk <- ne_countries(scale = "medium", country = "United Kingdom", returnclass = "sf")

df_sens$GymSiteType <- factor(df_sens$GymSiteType, levels = c("Hybrid", "Residential", "Workforce"))

GymSiteType_colors <- c(
```

```
  "Hybrid" = "blue",
  "Residential" = "red",
  "Workforce" = "green"
)

ggplot() +
  geom_sf(data = uk, fill = "grey95", color = NA) +
  geom_point(
    data = df_sens,
    aes(
      x = lon, y = lat,
      color = GymSiteType,
      size = abs(price_elasticity)
    ),
    alpha = 0.7
  ) +
  scale_color_manual(
    name = "Site type",
    values = GymSiteType_colors
  ) +
  scale_size_continuous(
    name = "Price elasticity\n(absolute value, %)",
    breaks = c(1, 2, 3, 4),
    labels = c("1%", "2%", "3%", "4%+"),
    range = c(2, 10)
  ) +
  labs(
    x = "Longitude",
    y = "Latitude",
    title = "Spatial Distribution of Gym Price Elasticity"
  ) +
  coord_sf(xlim = c(-5, 2), ylim = c(50, 57)) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "right",
    plot.title = element_text(size = 16, face = "bold")
  )
```



Visualization

```
# Extract all fixed effects
coef_tab <- res_spde$summary.fixed

# Sort in descending order by absolute mean value
coef_tab$abs_mean <- abs(coef_tab$mean)
coef_tab <- coef_tab[order(-coef_tab$abs_mean), ]

# Show the top few
print(coef_tab[, c("mean", "0.025quant", "0.975quant", "abs_mean")], digits=3)
```

```
##                                mean 0.025quant 0.975quant abs_mean
## intercept                  8.195813  -22.86109   39.25273 8.195813
## GymSiteTypeWorkforce       2.803514  -28.20670   33.81373 2.803514
## GymSiteTypeResidential     2.742157  -28.26823   33.75255 2.742157
## GymSiteTypeHybrid          2.650141  -28.36015   33.66043 2.650141
## urepeat                   -2.472003   -4.23683   -0.71246 2.472003
## promo                     -0.344412   -0.42959   -0.25921 0.344412
## ampu                       0.114682   -0.78127    1.00990 0.114682
## GymSiteTypeWorkforce:AAP   0.083526    0.04837    0.11868 0.083526
## w_ddrop                    0.074269    0.00578    0.14338 0.074269
## w_den                      0.066360   -0.06559    0.19668 0.066360
## w_unis                     0.059862   -0.02200    0.14204 0.059862
## w_ratio                   -0.042780   -0.09552    0.00995 0.042780
## w_ringshare               -0.036772   -0.09563    0.02228 0.036772
```

```
## GymSiteTypeResidential:AAP  0.029689    0.00189    0.05749 0.029689
## w_txn                      -0.026872   -0.08935    0.03489 0.026872
## AAP                        -0.018932   -0.04396    0.00610 0.018932
## dws                         0.017902   -0.05982    0.09569 0.017902
## competitor                 -0.016441   -0.02283   -0.01013 0.016441
## lon                        -0.013814   -0.04711    0.01955 0.013814
## parking_num                -0.010627   -0.09580    0.07352 0.010627
## lat                        -0.007073   -0.04041    0.02628 0.007073
## w_comp                     -0.006555   -0.13963    0.12781 0.006555
## AJF                        -0.006549   -0.01819    0.00509 0.006549
## u2                          0.003985   -0.01543    0.02345 0.003985
## aspu                        0.003711   -0.07117    0.07876 0.003711
## userd                       0.003356    0.00122    0.00553 0.003356
## u1                          0.000724   -0.01227    0.01380 0.000724
```

```
coef_tab$significant <- with(coef_tab,
    ( `0.025quant` > 0 & `0.975quant` > 0 ) |
    ( `0.025quant` < 0 & `0.975quant` < 0 )
)

sig_vars <- coef_tab[coef_tab$significant, ]
print(sig_vars[, c("mean", "0.025quant", "0.975quant", "significant")], digits=3)
```

```
##                               mean 0.025quant 0.975quant significant
## urepeat                    -2.47200   -4.23683   -0.71246        TRUE
## promo                      -0.34441   -0.42959   -0.25921        TRUE
## GymSiteTypeWorkforce:AAP    0.08353    0.04837    0.11868        TRUE
## w_ddrop                     0.07427    0.00578    0.14338        TRUE
## GymSiteTypeResidential:AAP  0.02969    0.00189    0.05749        TRUE
## competitor                 -0.01644   -0.02283   -0.01013        TRUE
## userd                       0.00336    0.00122    0.00553        TRUE
```

What's the ideal price point for maximizing revenue in a specific location?

```
# Set price range
price_grid <- seq(10, 60, by = 0.5)

# Take the first 100 points
n_sites <- min(100, nrow(df_spde))

# Spatial field average value
field_mean <- res_spde$summary.random$field$mean

# Record the results
results <- data.frame(
  site_id = 1:n_sites,
  orig_price = NA_real_,
  GymSiteType = NA_character_,
  orig_members = NA_real_,
  best_price = NA_real_,
  max_revenue = NA_real_,
  demand_at_best = NA_real_
```

```
)

for (i in 1:n_sites) {
  this_point <- df_spde[i, ]
  spatial_offset <- as.vector(A[i, ] %*% field_mean)

  orig_price <- this_point$AvgAccountPayment
  orig_members <- this_point$TotalMembers
  site_type <- as.character(this_point$GymSiteType)

  rev_grid <- numeric(length(price_grid))
  demand_grid <- numeric(length(price_grid))

  for (j in seq_along(price_grid)) {
    this_price <- price_grid[j]
    this_AAP <- (this_price - mean_AAP) / sd_AAP

    covar_row <- data.frame(
      intercept = 1,
      GymSiteType = factor(this_point$GymSiteType, levels = c("Hybrid", "Residential", "Workforce")),
      AAP = this_AAP,
      AJF = this_point$AJF,
      dws = this_point$dws,
      urepeat = this_point$urepeat,
      u1 = this_point$u1,
      u2 = this_point$u2,
      competitor = this_point$competitor,
      aspu = this_point$aspu,
      ampu = this_point$ampu,
      parking_num = this_point$parking_num,
      lat = this_point$lat,
      lon = this_point$lon,
      userd = this_point$userd,
      w_ddrop = this_point$w_ddrop,
      w_unis = this_point$w_unis,
      promo = this_point$promo,
      w_ringshare = this_point$w_ringshare,
      w_ratio = this_point$w_ratio,
      w_den = this_point$w_den,
      w_txn = this_point$w_txn,
      w_comp = this_point$w_comp
    )
    X_new <- model.matrix(
      ~ -1 + intercept + GymSiteType * AAP + AJF + dws + urepeat + u1 + u2 + competitor +
        aspu + ampu + parking_num + lat + lon + userd +
        w_ddrop + w_unis + promo + w_ringshare + w_ratio +
        w_den + w_txn + w_comp,
      data = covar_row
    )
    mu_fixed <- as.vector(X_new %*% res_spde$summary.fixed$mean)
    mu_pred <- mu_fixed + spatial_offset
    demand <- exp(mu_pred)
    revenue <- this_price * demand
```

```
    demand_grid[j] <- demand
    rev_grid[j] <- revenue
  }

  max_idx <- which.max(rev_grid)

  results$orig_price[i]   <- orig_price
  results$GymSiteType[i]  <- site_type
  results$orig_members[i] <- orig_members
  results$best_price[i]   <- price_grid[max_idx]
  results$max_revenue[i]  <- rev_grid[max_idx]
  results$demand_at_best[i] <- demand_grid[max_idx]
}

head(results, 100)
```

```
##     site_id orig_price GymSiteType orig_members best_price max_revenue
## 1         1   49.76229 Residential         6553         60    403664.4
## 2         2   31.86496 Residential         6546         60    383285.6
## 3         3   53.85660 Residential         8920         60    492817.6
## 4         4   21.27039 Residential          398         60    172011.4
## 5         5  130.76375 Residential         2900         60    167324.2
## 6         6   45.74215 Residential         7515         60    428558.0
## 7         7   37.38008 Residential         4863         60    311870.7
## 8         8   63.57703 Residential         5802         60    362020.1
## 9         9   35.53741 Residential         6311         60    382902.9
## 10       10   33.09622 Residential         5364         60    302884.6
## 11       11   42.84683 Residential         5977         60    320256.9
## 12       12   30.79135 Residential         5813         60    367033.3
## 13       13   34.56794 Residential         4441         60    259340.1
## 14       14   39.29420      Hybrid         4434         60    243030.2
## 15       15   41.69905 Residential         6764         60    400066.4
## 16       16   37.59785 Residential         6068         60    357639.6
## 17       17   29.40561 Residential         3335         60    212144.6
## 18       18   35.43446 Residential         6008         60    368261.0
## 19       19   30.17003 Residential         3787         60    227664.4
## 20       20   41.80710 Residential         4237         60    271264.6
## 21       21   56.88668 Residential         5871         60    370786.7
## 22       22   37.34240 Residential         7052         60    424543.5
## 23       23   30.25512 Residential         5363         60    300905.6
## 24       24   35.25028 Residential         3437         60    227633.0
## 25       25   28.75936      Hybrid         6462         60    351409.6
## 26       26   55.45999 Residential         2327         60    176764.6
## 27       27   32.23027 Residential         4911         60    286530.6
## 28       28   28.32629 Residential         5805         60    352902.5
## 29       29   35.48694      Hybrid         3979         60    277858.3
## 30       30   30.53194 Residential         4066         60    268160.7
## 31       31   35.50755 Residential         7526         60    415775.6
## 32       32   53.52255 Residential         6319         60    365381.1
## 33       33   35.82275   Workforce         3684         60    259594.3
## 34       34   47.89189 Residential         4854         60    321114.7
## 35       35   27.78948 Residential         6529         60    419414.8
## 36       36   40.72382 Residential         2102         60    136032.8
```

```
## 37       37   28.77936 Residential        3917         60    236929.8
## 38       38   47.22740 Residential        5329         60    377945.8
## 39       39   29.51025 Residential        5095         60    299557.2
## 40       40   21.10362 Residential        3690         60    235526.1
## 41       41   45.36810 Residential        5176         60    182345.7
## 42       42   38.24015 Residential        3356         60    183718.0
## 43       43   26.61749 Residential        6466         60    369843.4
## 44       44   26.00489 Residential        5704         60    364549.9
## 45       45   50.32992 Residential        5804         60    428055.6
## 46       46   31.04419 Residential        6414         60    399657.9
## 47       47   28.94087 Residential        6239         60    379266.4
## 48       48   28.75553 Residential        2965         60    182283.6
## 49       49   32.75674   Workforce        3697         60    245396.0
## 50       50   32.83326 Residential        5845         60    347534.4
## 51       51   28.97305 Residential        5725         60    353681.3
## 52       52   43.67383 Residential        4907         60    320110.0
## 53       53   22.19042 Residential        5036         60    321335.5
## 54       54   32.07840 Residential        6360         60    397075.3
## 55       55   38.02813 Residential        4639         60    289862.2
## 56       56   29.63155      Hybrid        4946         60    271283.6
## 57       57   32.72520 Residential        6089         60    378961.7
## 58       58   30.07553 Residential        5908         60    341671.5
## 59       59   36.66992      Hybrid        3991         60    245582.4
## 60       60   36.51983 Residential        2350         60    136572.5
## 61       61   26.78690 Residential        6093         60    361254.0
## 62       62   44.59809 Residential        6651         60    360070.7
## 63       63   30.32761   Workforce        7284         60    492126.6
## 64       64   92.58029      Hybrid        3935         60    274307.9
## 65       65   31.09515 Residential        3488         60    236904.6
## 66       66   40.19714 Residential        6248         60    363903.5
## 67       67   33.19212 Residential        6193         60    411185.9
## 68       68   24.39020 Residential        6256         60    396336.4
## 69       69   38.61919 Residential        3870         60    268031.9
## 70       70   37.84290      Hybrid        4559         60    254693.4
## 71       71   37.28882 Residential        5192         60    316571.3
## 72       72   37.40582 Residential        2369         60    154012.6
## 73       73   35.05313 Residential        2120         60    145379.5
## 74       74   35.79660 Residential        5087         60    311246.1
## 75       75   44.28016 Residential        6065         60    377013.0
## 76       76   33.01591 Residential        5822         60    326600.4
## 77       77   29.32046      Hybrid        4966         60    280381.0
## 78       78   40.71417 Residential        4746         60    288739.1
## 79       79   41.84017   Workforce        2909         60    197217.3
## 80       80   79.93288   Workforce        3337         60    208640.6
## 81       81   28.67948 Residential        4709         60    272912.7
## 82       82   40.68745 Residential        6359         60    378911.5
## 83       83   47.38377 Residential        5286         60    319892.0
## 84       84   37.43408 Residential        6500         60    394137.8
## 85       85   43.12559 Residential        6380         60    388094.0
## 86       86   39.40252 Residential        6264         60    382183.6
## 87       87   30.46058 Residential        3381         60    216535.2
## 88       88   31.96740 Residential        3668         60    223793.6
## 89       89   30.58539 Residential        4413         60    265971.9
## 90       90   39.12087 Residential        6187         60    348018.5
```

```
## 91       91   37.36705 Residential         7385         60    390682.7
## 92       92   31.14501 Residential         5271         60    305945.8
## 93       93   33.50361 Residential         6340         60    371954.1
## 94       94   31.76081 Residential         2766         60    181458.1
## 95       95   26.64654 Residential         2733         60    185489.4
## 96       96   89.80123 Residential         5465         60    179016.7
## 97       97   61.72654   Workforce         1715         60    101206.8
## 98       98   24.30259 Residential         6559         60    383069.3
## 99       99   57.55606 Residential         4560         60    289511.2
## 100     100   52.81152 Residential         5036         60    314168.4
##     demand_at_best
## 1         6727.741
## 2         6388.093
## 3         8213.626
## 4         2866.857
## 5         2788.737
## 6         7142.633
## 7         5197.845
## 8         6033.668
## 9         6381.715
## 10        5048.076
## 11        5337.616
## 12        6117.221
## 13        4322.335
## 14        4050.504
## 15        6667.774
## 16        5960.659
## 17        3535.744
## 18        6137.684
## 19        3794.406
## 20        4521.076
## 21        6179.778
## 22        7075.724
## 23        5015.094
## 24        3793.884
## 25        5856.827
## 26        2946.077
## 27        4775.510
## 28        5881.708
## 29        4630.972
## 30        4469.345
## 31        6929.593
## 32        6089.685
## 33        4326.571
## 34        5351.912
## 35        6990.247
## 36        2267.214
## 37        3948.829
## 38        6299.096
## 39        4992.620
## 40        3925.435
## 41        3039.095
## 42        3061.967
## 43        6164.057
```

```
## 44       6075.832
## 45       7134.259
## 46       6660.966
## 47       6321.106
## 48       3038.060
## 49       4089.934
## 50       5792.240
## 51       5894.688
## 52       5335.167
## 53       5355.591
## 54       6617.922
## 55       4831.037
## 56       4521.393
## 57       6316.029
## 58       5694.525
## 59       4093.040
## 60       2276.209
## 61       6020.901
## 62       6001.178
## 63       8202.110
## 64       4571.798
## 65       3948.411
## 66       6065.058
## 67       6853.099
## 68       6605.606
## 69       4467.199
## 70       4244.890
## 71       5276.188
## 72       2566.877
## 73       2422.991
## 74       5187.434
## 75       6283.550
## 76       5443.339
## 77       4673.017
## 78       4812.318
## 79       3286.955
## 80       3477.343
## 81       4548.546
## 82       6315.191
## 83       5331.533
## 84       6568.963
## 85       6468.234
## 86       6369.726
## 87       3608.920
## 88       3729.893
## 89       4432.866
## 90       5800.309
## 91       6511.378
## 92       5099.097
## 93       6199.235
## 94       3024.302
## 95       3091.490
## 96       2983.612
## 97       1686.780
```

```
## 98        6384.489
## 99        4825.187
## 100       5236.139
```