LASSO

Loading Date and Data Exploratory Analysis

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# Read in dataset
data = read.csv("Team_Bits_Data.csv")

# Remove price=0 entries
data = data[data$PRICE != 0, ]

# Remove rows with NA in all columns except 'YR_RMDL'
data = data %>% filter(!if_any(-YR_RMDL, is.na))

# Remove not useful columns
data = data %>% select(-SSL, -OBJECTID, -GIS_LAST_MOD_DTTM,
                       -QUALIFIED, -SALE_NUM, -BLDG_NUM,
                       -STYLE_D, -STRUCT_D, -GRADE_D,
                       -CNDTN_D, -EXTWALL_D, -ROOF_D,
                       -INTWALL_D, -USECODE, -HEAT_D,
                       -NUM_UNITS, -STRUCT)

head(data)
```

```
##   BATHRM HF_BATHRM HEAT AC ROOMS BEDRM  AYB YR_RMDL  EYB STORIES
## 1      4         1    8  Y    12     6 1911    2021 1989    3.75
## 2      3         1    1  Y    13     5 1912    2009 1978    3.00
## 3      3         1    7  Y     6     4 1910    2022 1993    3.00
## 4      3         1    7  Y    11     4 1912    2000 1978    3.00
## 5      4         1    1  Y    11     5 1912    2007 1993    3.00
## 6      7         1    8  Y    16     7 1895    2014 1993    3.00
##                 SALEDATE   PRICE  GBA STYLE GRADE CNDTN EXTWALL ROOF INTWALL
## 1 2019/08/19 04:00:00+00 3275000 6765    10     8     4      20   11       6
## 2 1999/08/04 04:00:00+00  550000 2282     7     6     4      14    2       6
## 3 2019/07/22 04:00:00+00 1700000 2016     7     6     4      14    6       6
## 4 2021/10/27 04:00:00+00 1500000 2034     7     6     4      14    6       6
## 5 2023/04/18 04:00:00+00 2232500 2655     7     6     5      14    2       6
## 6 2013/12/30 05:00:00+00 1320000 2894     7     6     5      14    6       6
##   KITCHENS FIREPLACES LANDAREA
## 1        1          6     2104
## 2        2          3      936
## 3        2          2      936
## 4        2          2      988
## 5        3          4     1674
## 6        4          1     1674
```

Variable Explanation

We are dealing with housing data in this report, let me go over through the meanings behind each predictor:

1. PRICE: response
2. BATHRM: # bathrooms
3. HF_BATHRM: # half bathrooms
4. HEAT: heating
5. AC: air conditioning
6. ROOMS: # rooms
7. BEDRM: # bedrooms
8. AYB: The earliest time the main portion of the building was built
9. YR_RMDL: Year structure was remodelled
10. EYB: The year an improvement was built
11. STORIES: # stories in primary dwelling
12. SALEDATE: Date of sale
13. GBA: Gross building area in square feet
14. STYLE: House style
15. GRADE: House grade
16. CNDTN: House condition
17. EXTWALL: Exterior wall tyle
18. ROOF: Roof type
19. INTWALL: Interior wall type
20. KITCHENS: # kitchens
21. FIREPLACES: # fireplaces
22. LANDAREA: Land area of property in square feet

NA Data

Now let us explore the percentage of missing data for each predictor:

```r
missing_data = round(sapply(data, function(x) mean(is.na(x) * 100)), 3)

missing_data
```

```
##     BATHRM  HF_BATHRM       HEAT         AC      ROOMS      BEDRM        AYB
##      0.000      0.000      0.000      0.000      0.000      0.000      0.000
##    YR_RMDL        EYB    STORIES   SALEDATE      PRICE        GBA      STYLE
##     36.432      0.000      0.000      0.000      0.000      0.000      0.000
##      GRADE      CNDTN    EXTWALL       ROOF    INTWALL   KITCHENS FIREPLACES
##      0.000      0.000      0.000      0.000      0.000      0.000      0.000
##   LANDAREA
##      0.000
```

From the R output above, observe that "YR_RMDL: Year structure was remodeled" has around 36% missing data. A possible explanation for this could be: not all buildings were remodeled.

Preprocessing

- Converted some predictors to numerical values:
  - AC: "Y" and "N" corresponds to "1" and "0".
  - SALEDATE: Transform calendar format values in SALEDATE to numerical values using as.Date().
- Created dummy variables for categorical predictors:

    - These categorical variables include: "HEAT", "STYLE", "GRADE", "CNDTN", "EXTWALL", "ROOF" and "INTWALL".

- Introduced a few new variables:
    - `SALE_YEAR`: The year that the house was sold, it is derived from `SALEDATE`.
    - `SALE_AYB_DIFF`: The difference between the year sold and the year built.
    - `SALE_EYB_DIFF`: The difference between the year sold and the year an improvement was applied.
    - `SALE_RMDL_DIFF`: The difference between the year sold and the year structure was remodeled.'

```r
library(lubridate)
```

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

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

```r
library(fastDummies)

# Transform Yes/No for having AC to numerical values
data$AC = ifelse(data$AC == 'Y', 1, 0)

# Create dummy variables for categorical predictors
data = dummy_cols(
  data,
  select_columns = c("HEAT", "STYLE", "GRADE", "CNDTN",
                     "EXTWALL", "ROOF", "INTWALL", "AC"),
  remove_selected_columns = TRUE,
  remove_first_dummy = TRUE
)

# Add SALEYEAR
data$SALE_YEAR = year(ymd_hms(data$SALEDATE))

# Add SALEYEAR and AYB diff
data$SALE_AYB_DIFF = data$SALE_YEAR - data$AYB

# Add SALEYEAR and EYB diff
data$SALE_EYB_DIFF = data$SALE_YEAR - data$EYB

# Add SALEYEAR and YR_RMDL diff
data$SALE_RMDL_DIFF = data$SALE_YEAR - data$YR_RMDL

# Convert SALEDATE column to numeric values
data$SALEDATE = as.numeric(as.Date(data$SALEDATE))
```

```r
# Replace NA with column median
data = data.frame(lapply(data, function(column) {
  column_median = median(column, na.rm = TRUE)
  column[is.na(column)] = column_median
  column
}))
```

```r
# Define box-cox and inverse box-cox transformation
powerfun = function(y, lambda) {
  if (lambda == 0) {
    return(log(y))
  } else {
    return((y^lambda - 1) / lambda)
  }
}

inv_powerfun = function(y_transformed, lambda) {
  if (lambda == 0) {
    return(exp(y_transformed))
  } else {
    return((lambda * y_transformed + 1)^(1/lambda))
  }
}
```

Data for Training and Validating

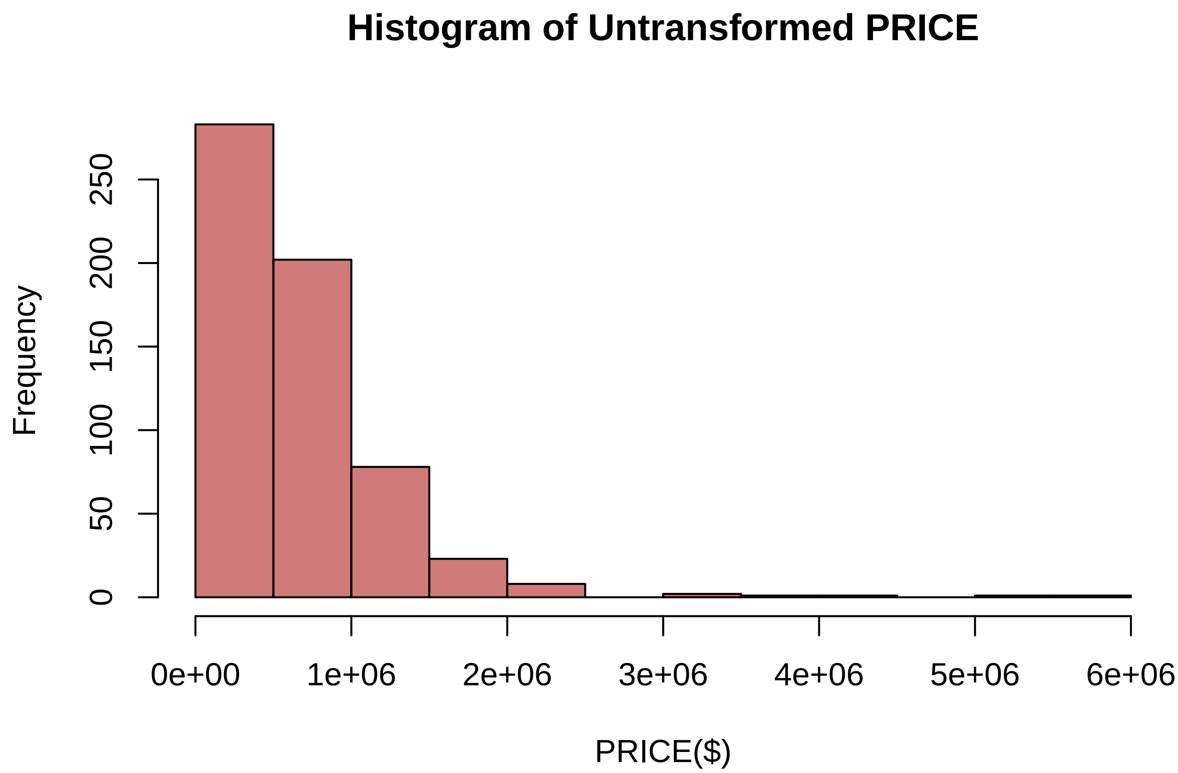
```r
set.seed(9159)

# Randomly sample 600 data entries for our project
clean_data = data[sample(nrow(data), 600),]

data_train = clean_data[1:500, ]  # First 500 rows for training
data_valid = clean_data[501:600, ]  # Last 100 rows for validation
```

Data Visualization

```r
library(lmtest)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

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

```r
# Plot histogram for untransformed response variable `PRICE`
hist(clean_data$PRICE, col=adjustcolor('firebrick',alpha=0.6),
     xlab='PRICE($)', ylab='Frequency',
     main='Histogram of Untransformed PRICE')
```

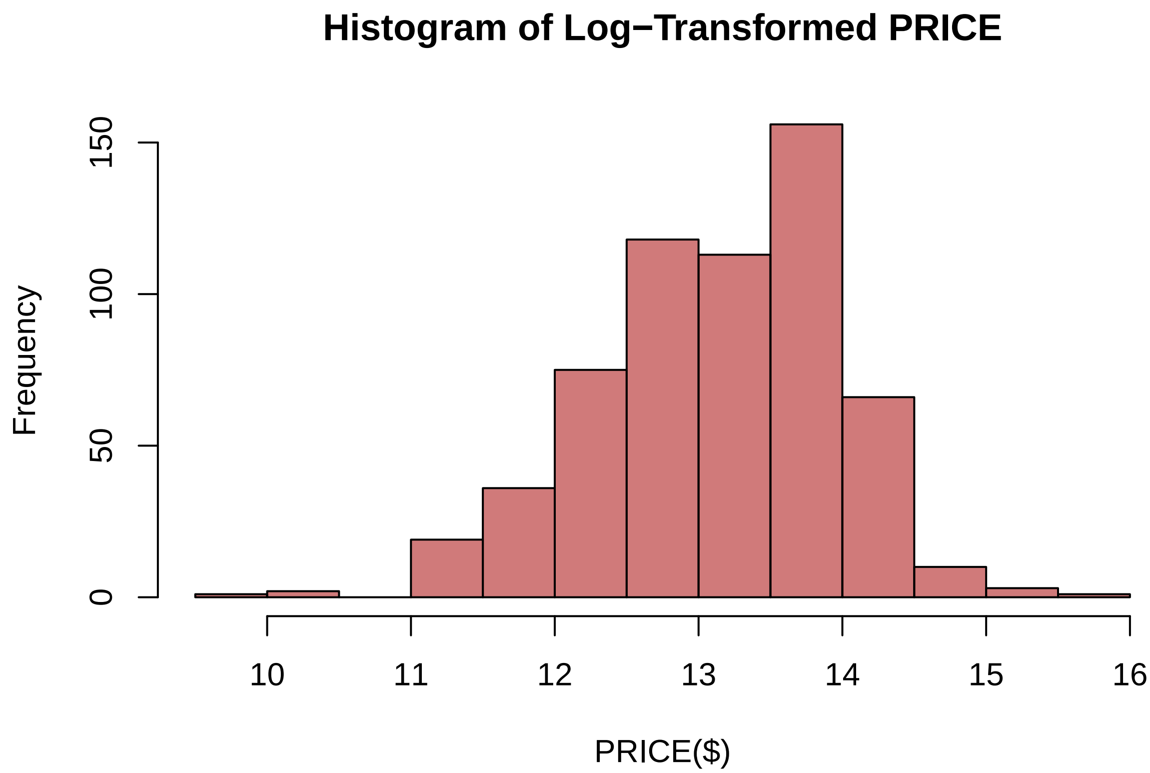



```r
shapiro.test(clean_data$PRICE)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  clean_data$PRICE
## W = 0.76073, p-value < 2.2e-16
```

```r
lambda = 0.0
```

```r
# Plot histogram for log-transformed response variable `PRICE`
hist(powerfun(clean_data$PRICE, lambda), col=adjustcolor('firebrick',alpha=0.6),
     xlab='PRICE($)', ylab='Frequency',
     main='Histogram of Log-Transformed PRICE')
```



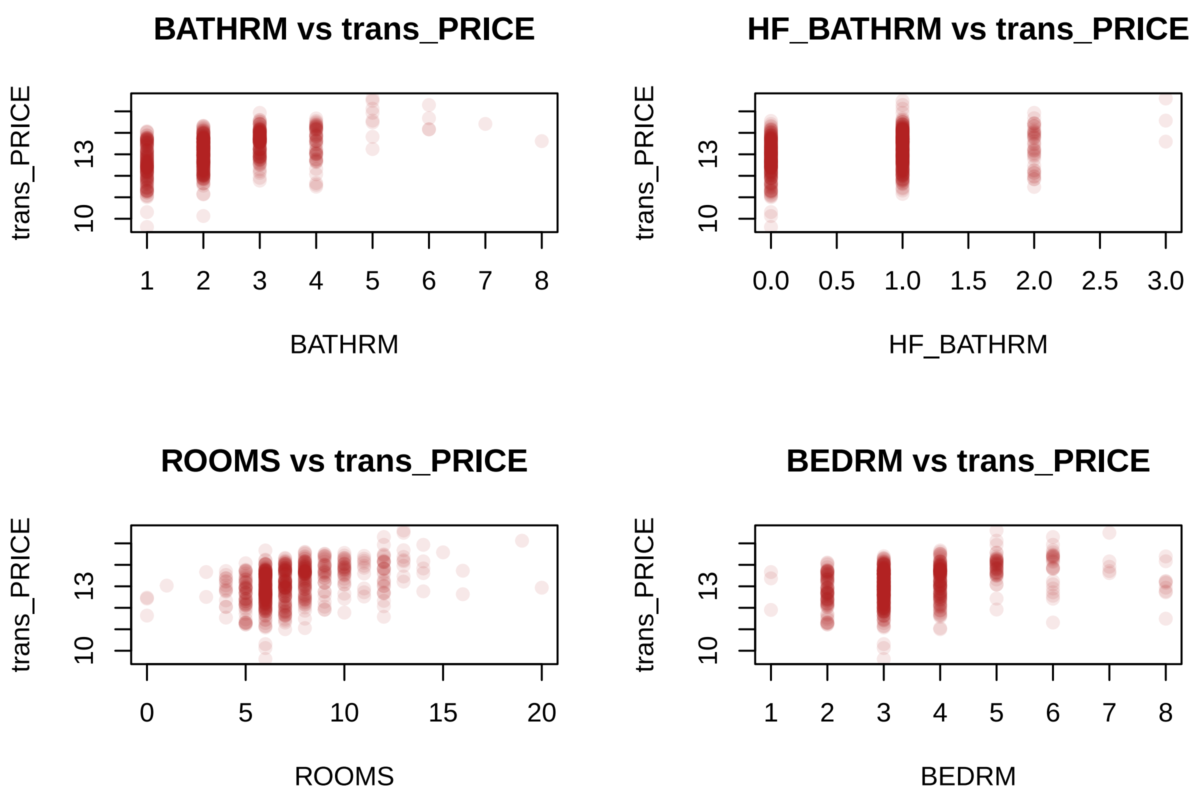

```r
shapiro.test(powerfun(clean_data$PRICE, lambda))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  powerfun(clean_data$PRICE, lambda)
## W = 0.98444, p-value = 5.178e-06
```

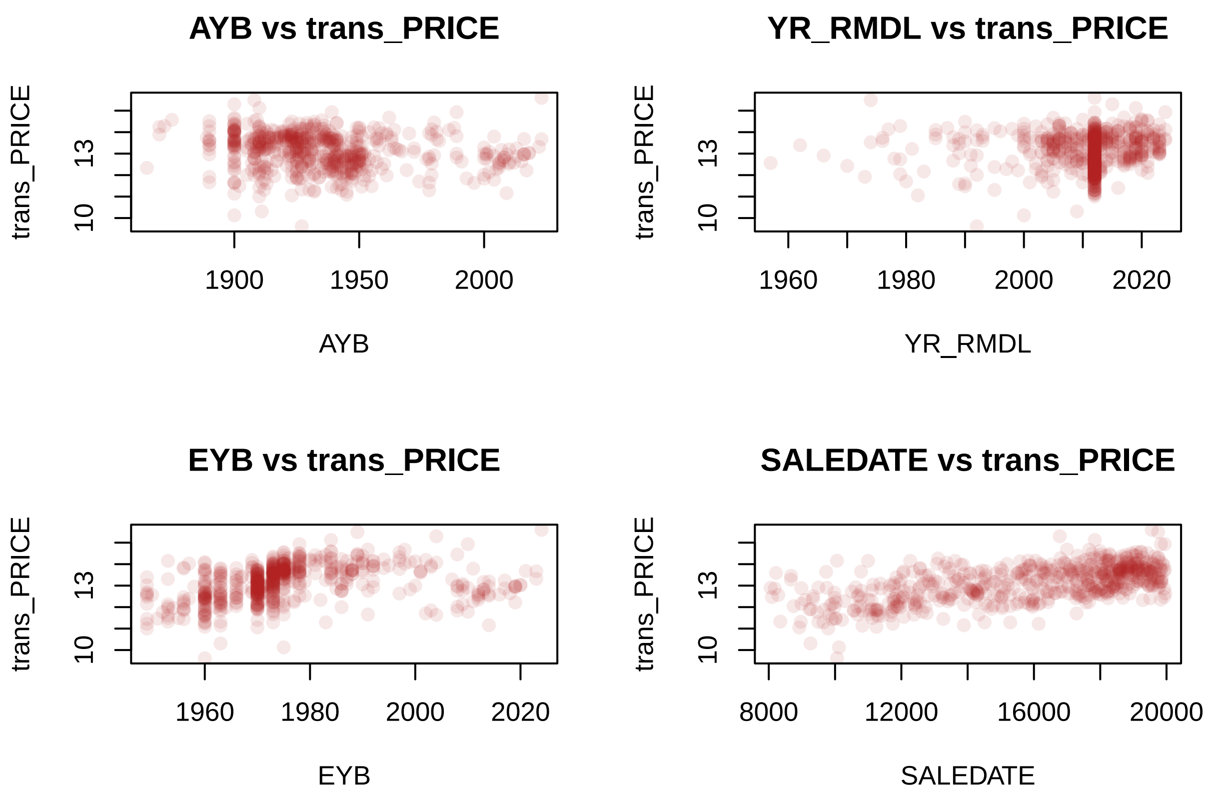
```r
trans_PRICE = powerfun(clean_data$PRICE, lambda)
```

```r
par(mfrow = c(2, 2))

plot(clean_data$BATHRM, trans_PRICE,
     pch = 19, col = adjustcolor('firebrick', alpha = 0.1),
     xlab = "BATHRM", ylab = "trans_PRICE",
     main = "BATHRM vs trans_PRICE")

plot(clean_data$HF_BATHRM, trans_PRICE,
     pch = 19, col = adjustcolor('firebrick', alpha = 0.1),
     xlab = "HF_BATHRM", ylab = "trans_PRICE",
     main = "HF_BATHRM vs trans_PRICE")

plot(clean_data$ROOMS, trans_PRICE,
     pch = 19, col = adjustcolor('firebrick', alpha = 0.1),
     xlab = "ROOMS", ylab = "trans_PRICE",
     main = "ROOMS vs trans_PRICE")
```

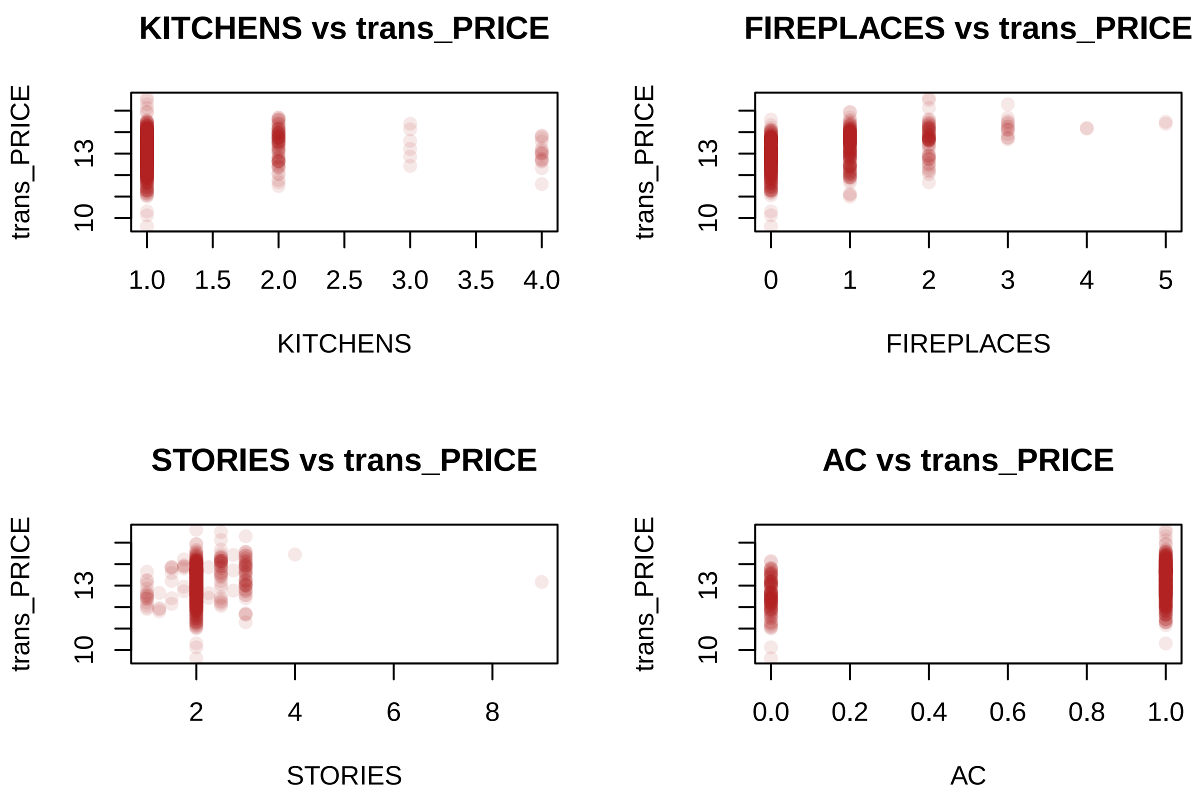
```r
plot(clean_data$BEDRM, trans_PRICE,
     pch = 19, col = adjustcolor('firebrick', alpha = 0.1),
     xlab = "BEDRM", ylab = "trans_PRICE",
     main = "BEDRM vs trans_PRICE")
```



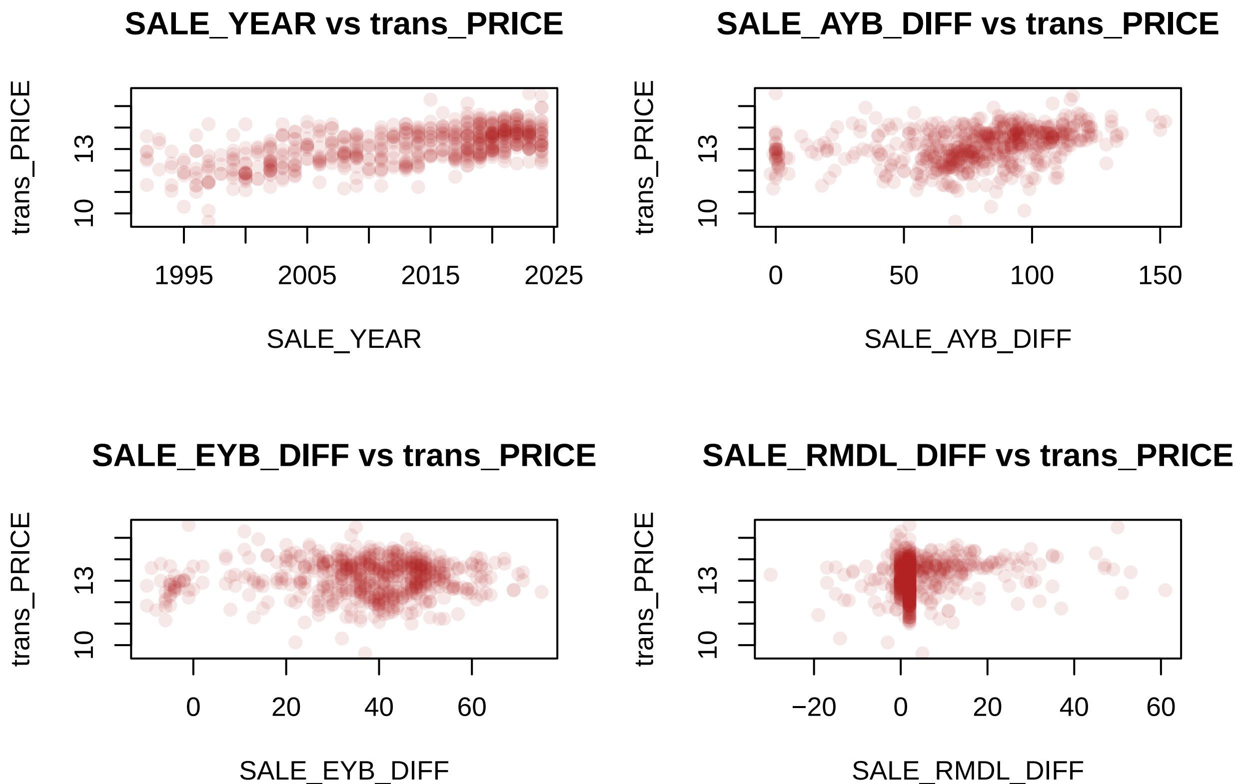

```r
par(mfrow = c(2, 2))

plot(clean_data$AYB, trans_PRICE,
     pch = 19, col = adjustcolor('firebrick', alpha = 0.1),
     xlab = "AYB", ylab = "trans_PRICE",
     main = "AYB vs trans_PRICE")

plot(clean_data$YR_RMDL, trans_PRICE,
     pch = 19, col = adjustcolor('firebrick', alpha = 0.1),
     xlab = "YR_RMDL", ylab = "trans_PRICE",
     main = "YR_RMDL vs trans_PRICE")

plot(clean_data$EYB, trans_PRICE,
     pch = 19, col = adjustcolor('firebrick', alpha = 0.1),
     xlab = "EYB", ylab = "trans_PRICE",
     main = "EYB vs trans_PRICE")

plot(clean_data$SALEDATE, trans_PRICE,
     pch = 19, col = adjustcolor('firebrick', alpha = 0.1),
     xlab = "SALEDATE", ylab = "trans_PRICE",
     main = "SALEDATE vs trans_PRICE")
```



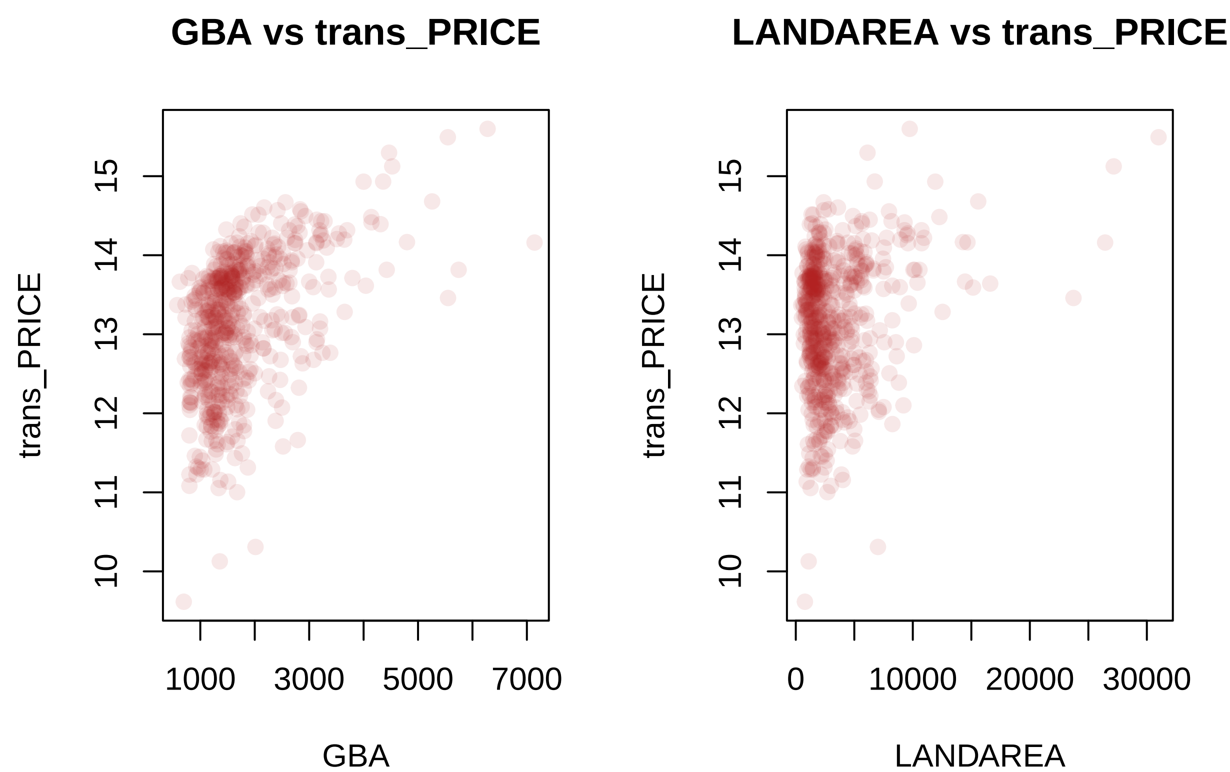

```r
par(mfrow = c(2, 2))

plot(clean_data$KITCHENS, trans_PRICE,
     pch = 19, col = adjustcolor('firebrick', alpha = 0.1),
     xlab = "KITCHENS", ylab = "trans_PRICE",
     main = "KITCHENS vs trans_PRICE")

plot(clean_data$FIREPLACES, trans_PRICE,
     pch = 19, col = adjustcolor('firebrick', alpha = 0.1),
     xlab = "FIREPLACES", ylab = "trans_PRICE",
     main = "FIREPLACES vs trans_PRICE")

plot(clean_data$STORIES, trans_PRICE,
     pch = 19, col = adjustcolor('firebrick', alpha = 0.1),
     xlab = "STORIES", ylab = "trans_PRICE",
     main = "STORIES vs trans_PRICE")

plot(clean_data$AC, trans_PRICE,
     pch = 19, col = adjustcolor('firebrick', alpha = 0.1),
     xlab = "AC", ylab = "trans_PRICE",
     main = "AC vs trans_PRICE")
```




```r
par(mfrow = c(2, 2))

plot(clean_data$SALE_YEAR, trans_PRICE,
     pch = 19, col = adjustcolor('firebrick', alpha = 0.1),
     xlab = "SALE_YEAR", ylab = "trans_PRICE",
     main = "SALE_YEAR vs trans_PRICE")

plot(clean_data$SALE_AYB_DIFF, trans_PRICE,
     pch = 19, col = adjustcolor('firebrick', alpha = 0.1),
     xlab = "SALE_AYB_DIFF", ylab = "trans_PRICE",
     main = "SALE_AYB_DIFF vs trans_PRICE")

plot(clean_data$SALE_EYB_DIFF, trans_PRICE,
     pch = 19, col = adjustcolor('firebrick', alpha = 0.1),
     xlab = "SALE_EYB_DIFF", ylab = "trans_PRICE",
     main = "SALE_EYB_DIFF vs trans_PRICE")

plot(clean_data$SALE_RMDL_DIFF, trans_PRICE,
     pch = 19, col = adjustcolor('firebrick', alpha = 0.1),
     xlab = "SALE_RMDL_DIFF", ylab = "trans_PRICE",
     main = "SALE_RMDL_DIFF vs trans_PRICE")
```




```r
par(mfrow = c(1, 2))

plot(clean_data$GBA, trans_PRICE,
     pch = 19, col = adjustcolor('firebrick', alpha = 0.1),
     xlab = "GBA", ylab = "trans_PRICE",
     main = "GBA vs trans_PRICE")

plot(clean_data$LANDAREA, trans_PRICE,
     pch = 19, col = adjustcolor('firebrick', alpha = 0.1),
     xlab = "LANDAREA", ylab = "trans_PRICE",
     main = "LANDAREA vs trans_PRICE")
```



Model Building and Analysis

Model Training

```r
library(glmnet)
```

```
## Loading required package: Matrix
## Loaded glmnet 4.1-8
```

```r
X_train = model.matrix(powerfun(PRICE, lambda)~., data=data_train)[,-1]
X_test = model.matrix(powerfun(PRICE, lambda)~., data=data_valid)[,-1]
y_train = powerfun(data_train$PRICE, lambda)
y_test = powerfun(data_valid$PRICE, lambda)
```

```r
cv_lasso = cv.glmnet(X_train, y_train, alpha=1)
best_lasso_lambda = cv_lasso$lambda.min
```

```r
lasso_model = glmnet(X_train, y_train, alpha=1, lambda=best_lasso_lambda)
```

```r
# Check non-zero LASSO coefficients
predict(lasso_model, type="coefficients", s=best_lasso_lambda)
```

```
## 116 x 1 sparse Matrix of class "dgCMatrix"
##                             s1
## (Intercept)      7.516945e+00
## BATHRM           5.731575e-02
```

```
## HF_BATHRM       4.134033e-02
## ROOMS           1.256720e-02
## BEDRM           .
## AYB             .
## YR_RMDL         .
## EYB             1.351532e-03
## STORIES         .
## SALEDATE        1.279783e-04
## GBA             1.474965e-04
## KITCHENS        .
## FIREPLACES      1.626841e-01
## LANDAREA        4.875322e-06
## HEAT_1         -7.447608e-02
## HEAT_2          .
## HEAT_3          .
## HEAT_4          .
## HEAT_5          .
## HEAT_6          .
## HEAT_7          .
## HEAT_8          1.923311e-01
## HEAT_9          .
## HEAT_10         .
## HEAT_11         .
## HEAT_12         .
## HEAT_13         1.329453e-02
## STYLE_1         .
## STYLE_2         .
## STYLE_3         .
## STYLE_4         .
## STYLE_5         .
## STYLE_6         .
## STYLE_7         .
## STYLE_8         .
## STYLE_9         .
## STYLE_10        .
## STYLE_11        .
## STYLE_12        .
## STYLE_13        .
## STYLE_14        .
## STYLE_15        .
## STYLE_94        .
## STYLE_99        .
## GRADE_1         .
## GRADE_2         .
## GRADE_3        -3.014377e-01
## GRADE_4        -8.476860e-02
## GRADE_5         .
## GRADE_6         1.679326e-01
## GRADE_7         1.441810e-01
## GRADE_8         4.255033e-02
## GRADE_9         1.636605e-01
## GRADE_10        .
## GRADE_11        .
## GRADE_12        .
```

```
## CNDTN_1             .
## CNDTN_2          -1.267885e-01
## CNDTN_3          -1.427508e-01
## CNDTN_4             .
## CNDTN_5           1.235548e-01
## CNDTN_6             .
## EXTWALL_1           .
## EXTWALL_2           .
## EXTWALL_3           .
## EXTWALL_4           .
## EXTWALL_5         1.430457e-02
## EXTWALL_6           .
## EXTWALL_7           .
## EXTWALL_8           .
## EXTWALL_10          .
## EXTWALL_11          .
## EXTWALL_12          .
## EXTWALL_13          .
## EXTWALL_14          .
## EXTWALL_15        5.388650e-02
## EXTWALL_16          .
## EXTWALL_17          .
## EXTWALL_18        1.517712e-01
## EXTWALL_19       -3.552675e-01
## EXTWALL_20          .
## EXTWALL_21          .
## EXTWALL_22       -4.554345e-03
## EXTWALL_23          .
## EXTWALL_24          .
## ROOF_1           -6.948948e-03
## ROOF_2              .
## ROOF_3              .
## ROOF_4              .
## ROOF_5              .
## ROOF_6            6.314744e-02
## ROOF_7              .
## ROOF_8              .
## ROOF_9              .
## ROOF_10             .
## ROOF_11           1.087782e-02
## ROOF_12             .
## ROOF_13             .
## ROOF_14             .
## ROOF_15             .
## INTWALL_1           .
## INTWALL_2        -1.530926e-01
## INTWALL_3           .
## INTWALL_4           .
## INTWALL_5           .
## INTWALL_6           .
## INTWALL_7           .
## INTWALL_8         1.037291e-01
## INTWALL_9           .
## INTWALL_10          .
```

```
## INTWALL_11      .
## AC_1            1.222395e-01
## SALE_YEAR       .
## SALE_AYB_DIFF   4.156588e-03
## SALE_EYB_DIFF   .
## SALE_RMDL_DIFF  .
```

Compute Loss Metrics

```r
# Predicted values are transformed by powerfun(PRICE, lambda)
valid_pred = predict(lasso_model, s=best_lasso_lambda, newx=X_test)

# Using inv_powerfun to convert back to original scale
inv_valid_pred = inv_powerfun(valid_pred, lambda)
```

Compute MSE, RMSE and RMSLE

```r
# Compute metrics on validation dataset
mse = mean((data_valid$PRICE - inv_valid_pred)^2)
rmse = sqrt(mse)
rmsle = sqrt(mean((log(data_valid$PRICE) - log(inv_valid_pred))^2))
```

```r
cat("MSE:", mse, "\n")
```

```
## MSE: 165035403551
```

```r
cat("RMSE:", rmse, "\n")
```

```
## RMSE: 406245.5
```

```r
cat("RMSLE:", rmsle, "\n")
```

```
## RMSLE: 0.434073
```