Real Estate Value Prediction

Machine Learning Project

SHIH-YUAN WANG

Instructor: Alfonso Berumen

Outline

PART ONE
Introduction

PART TWO
Data Description

PART THREE
Exploratory Data Analysis

PART FOUR
Data Transformation

PART FIVE
Data Splitting

PART SIX
Modelling

PART SEVEN
Evaluation & Conclusion

Introduction

PART ONE

Problem Introduction

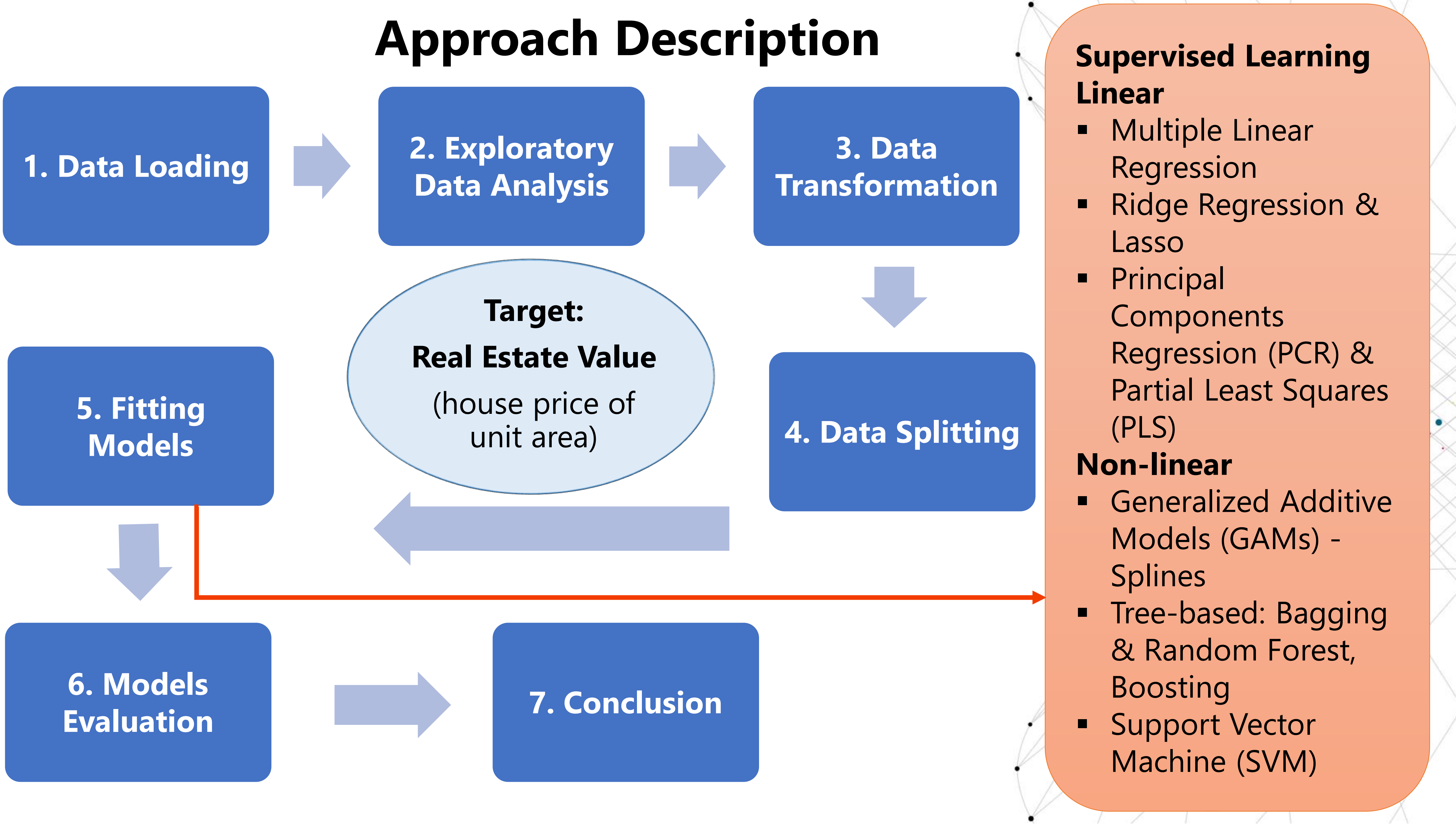
Motivation

- Understand where people tend to live in a city
  → identify what factors have more impact on property prices
  → help urban design and policies

Purpose

- Predict **real estate value** based on several features
- e.g. transaction date, house age, transportation, and other amenities

Approach Description



Data Description

PART TWO

Data

- **Real estate valuation data set:**

  Source: UCI Machine Learning Repository

  https://archive.ics.uci.edu/ml/datasets/Real+estate+valuation+data+set

- **Description:** The market historical data set of real estate valuation is collected from Xindian Dist., New Taipei City, Taiwan from August 2012 to July 2013.

- **Citation:** Yeh, I. C., & Hsu, T. K. (2018). Building real estate valuation models with comparative approach through case-based reasoning. Applied Soft Computing, 65, 260-271.

Data Description

Number of Instances:	414
Number of Attributes:	7

Predictors

Designation	Attribute	Unit	Value for example
X1	Transaction date	year and month	2013.250=2013 March, 2013.500=2013 June, etc.
X2	House age	year	32
X3	Distance to the nearest MRT station	meter	84.87882
X4	Number of convenience stores in the living circle on foot	Integer	10
X5	Geographic coordinate: latitude	degree	24.98298
X6	Geographic coordinate: longitude	degree	121.54024

Response

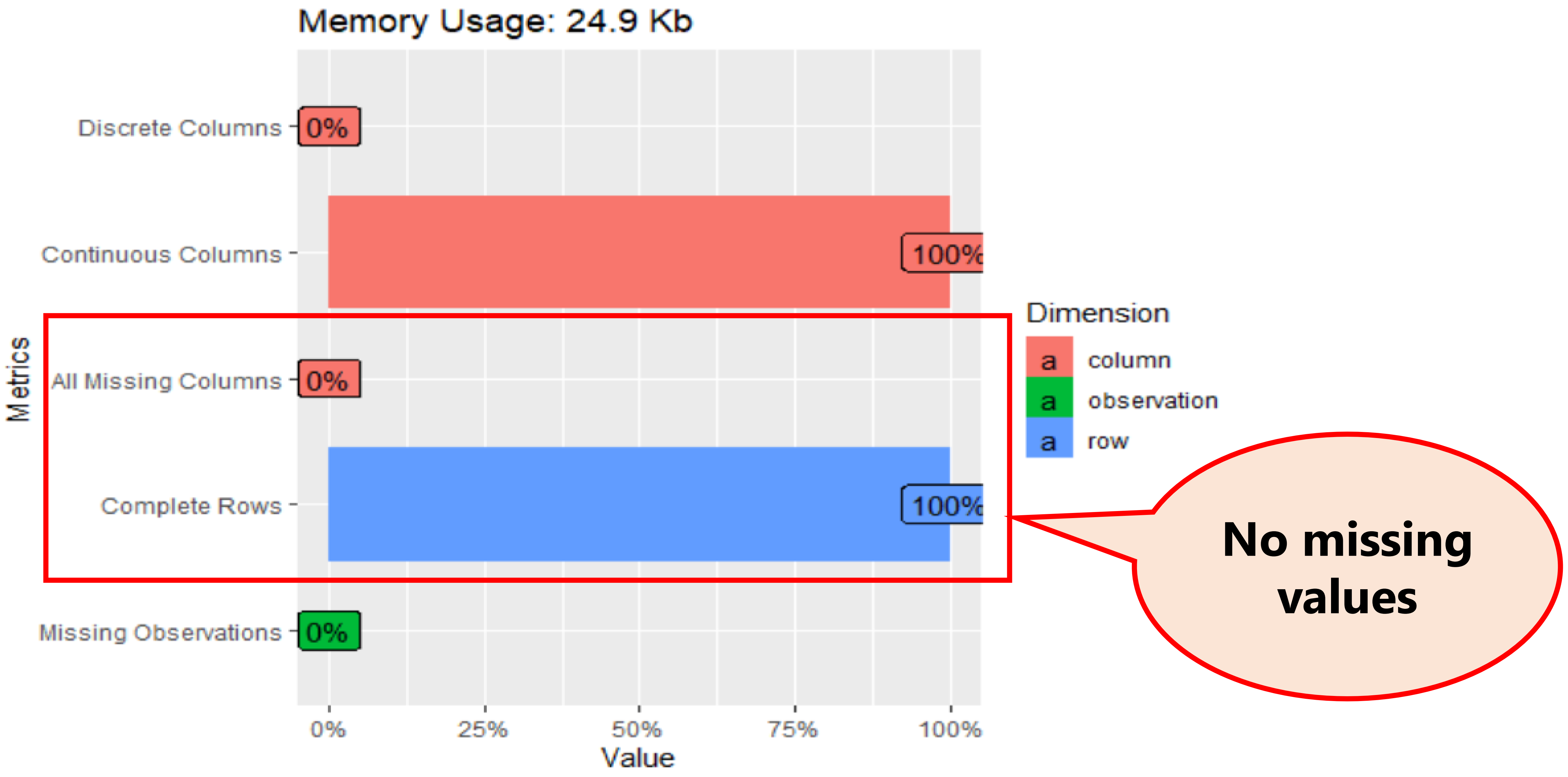
Y	house price of unit area	10,000 New Taiwan Dollar/Ping, where Ping is a local unit, 1 Ping = 3.3 meter squared	37.9

```
head(housing)
#   x1.tranDate x2.age  x3.disMRT x4.numConv   x5.lat  x6.long y.price
# 1    2012.917   32.0   84.87882         10 24.98298 121.5402    37.9
# 2    2012.917   19.5  306.59470          9 24.98034 121.5395    42.2
# 3    2013.583   13.3  561.98450          5 24.98746 121.5439    47.3
# 4    2013.500   13.3  561.98450          5 24.98746 121.5439    54.8
# 5    2012.833    5.0  390.56840          5 24.97937 121.5425    43.1
# 6    2012.667    7.1 2175.03000          3 24.96305 121.5125    32.1

# Descriptive statistics
summary(housing)
#  x1.tranDate      x2.age         x3.disMRT        x4.numConv        x5.lat         x6.long          y.price
# Min.   :2013   Min.   : 0.000   Min.   :  23.38   Min.   : 0.000   Min.   :24.93   Min.   :121.5   Min.   :  7.60
# 1st Qu.:2013   1st Qu.: 9.025   1st Qu.: 289.32   1st Qu.: 1.000   1st Qu.:24.96   1st Qu.:121.5   1st Qu.: 27.70
# Median :2013   Median :16.100   Median : 492.23   Median : 4.000   Median :24.97   Median :121.5   Median : 38.45
# Mean   :2013   Mean   :17.713   Mean   :1083.89   Mean   : 4.094   Mean   :24.97   Mean   :121.5   Mean   : 37.98
# 3rd Qu.:2013   3rd Qu.:28.150   3rd Qu.:1454.28   3rd Qu.: 6.000   3rd Qu.:24.98   3rd Qu.:121.5   3rd Qu.: 46.60
# Max.   :2014   Max.   :43.800   Max.   :6488.02   Max.   :10.000   Max.   :25.01   Max.   :121.6   Max.   :117.50
```

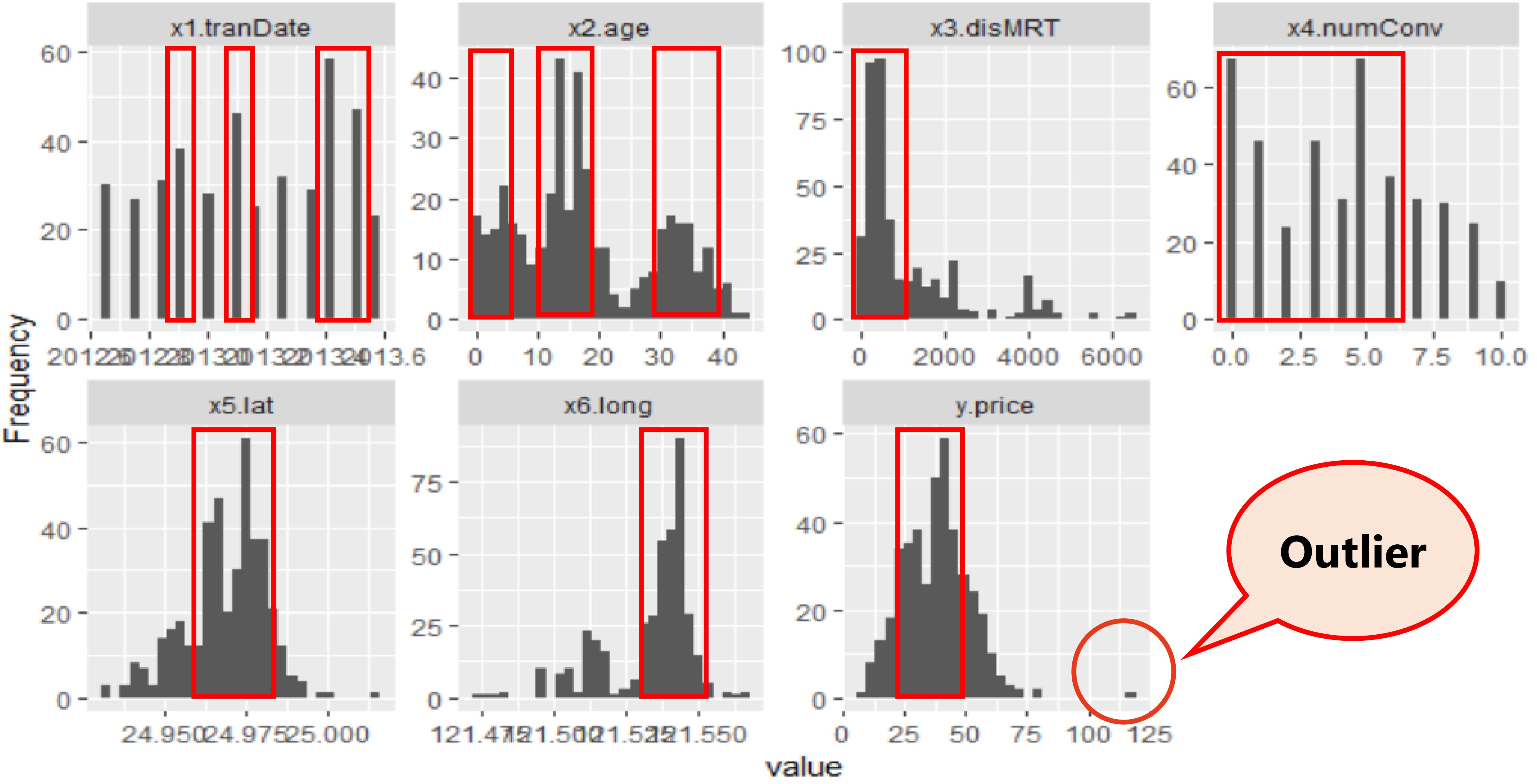
Exploratory Data Analysis

PART THREE

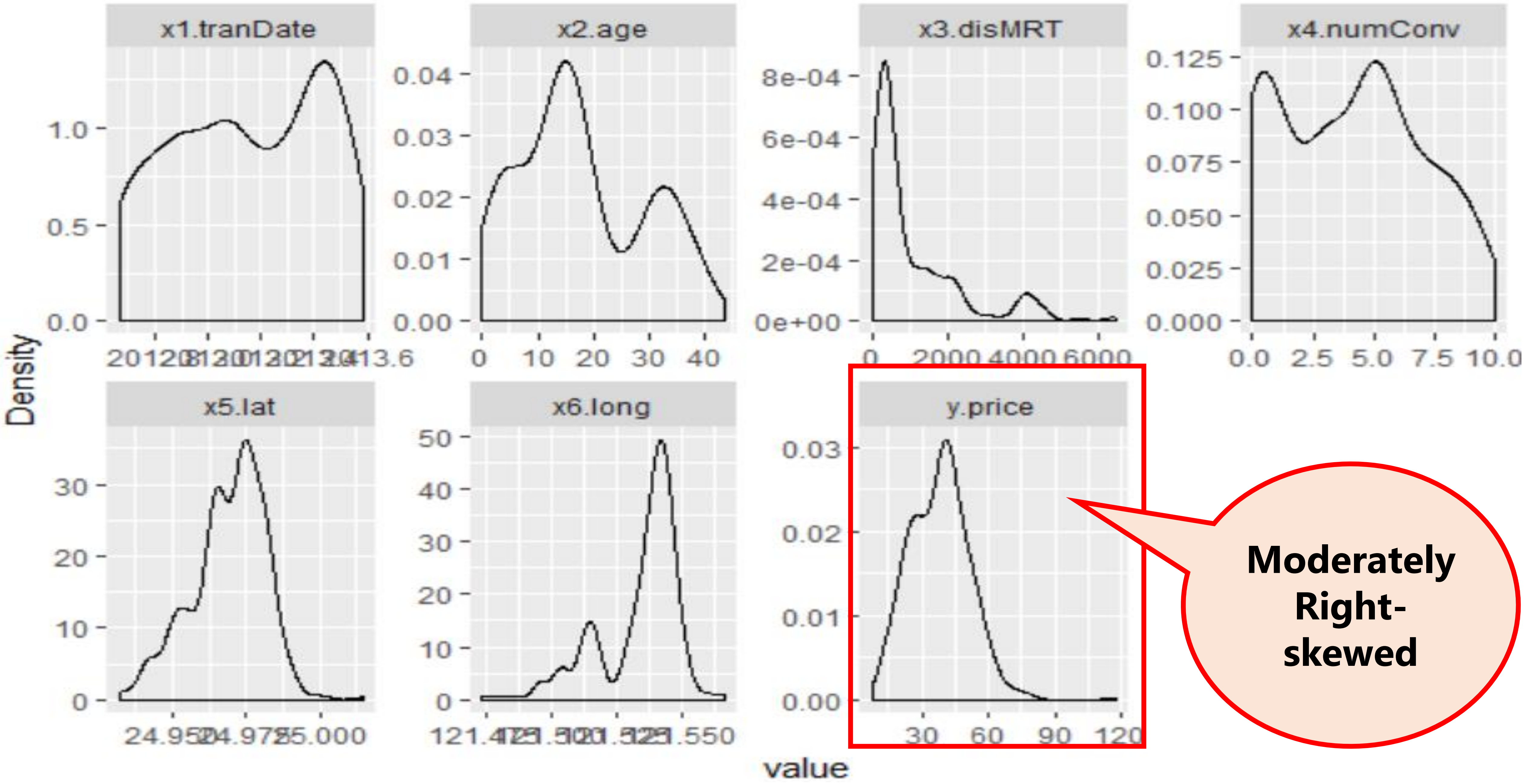
Data Overview



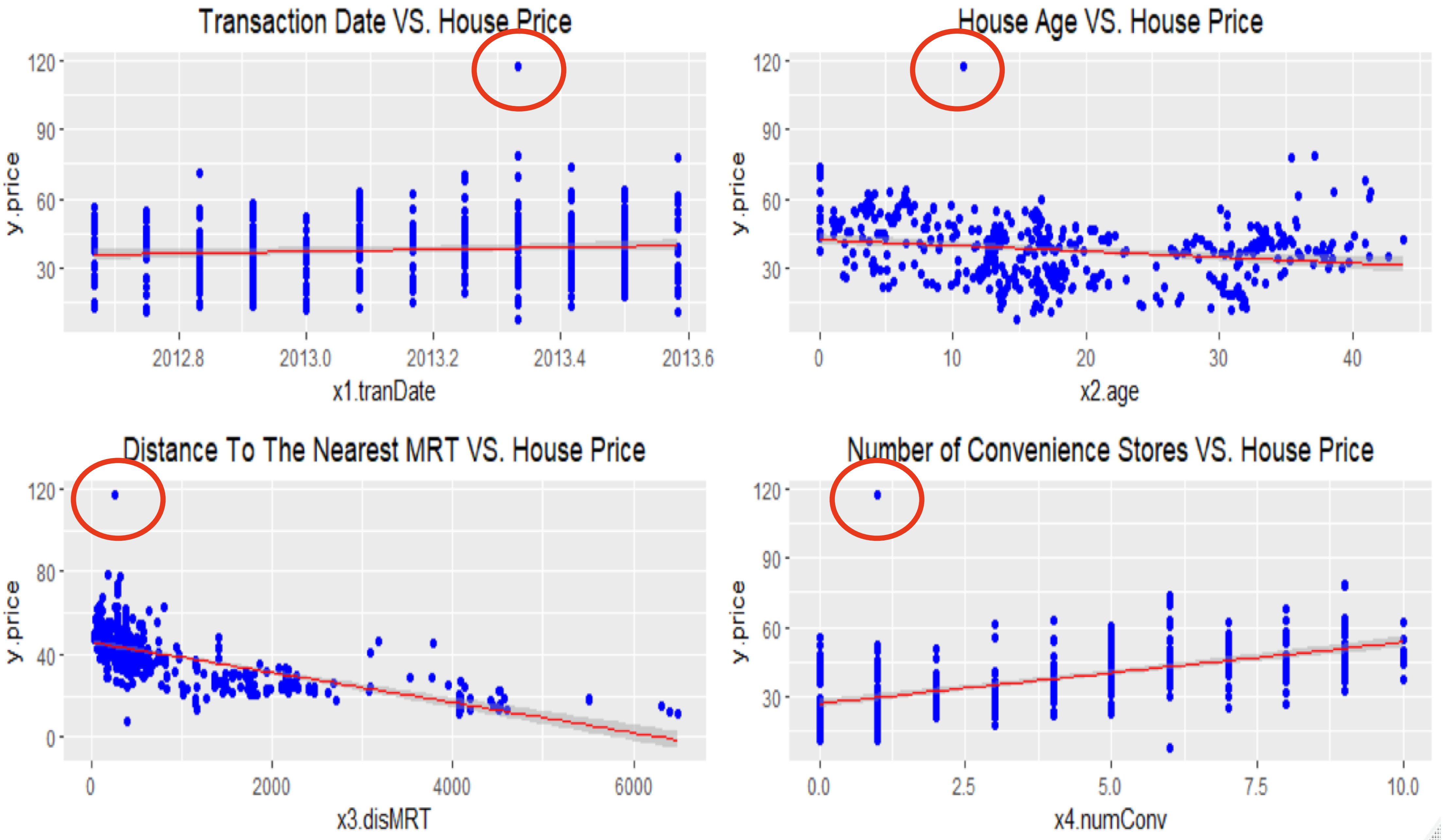
The Distribution of Each Variable - Histogram



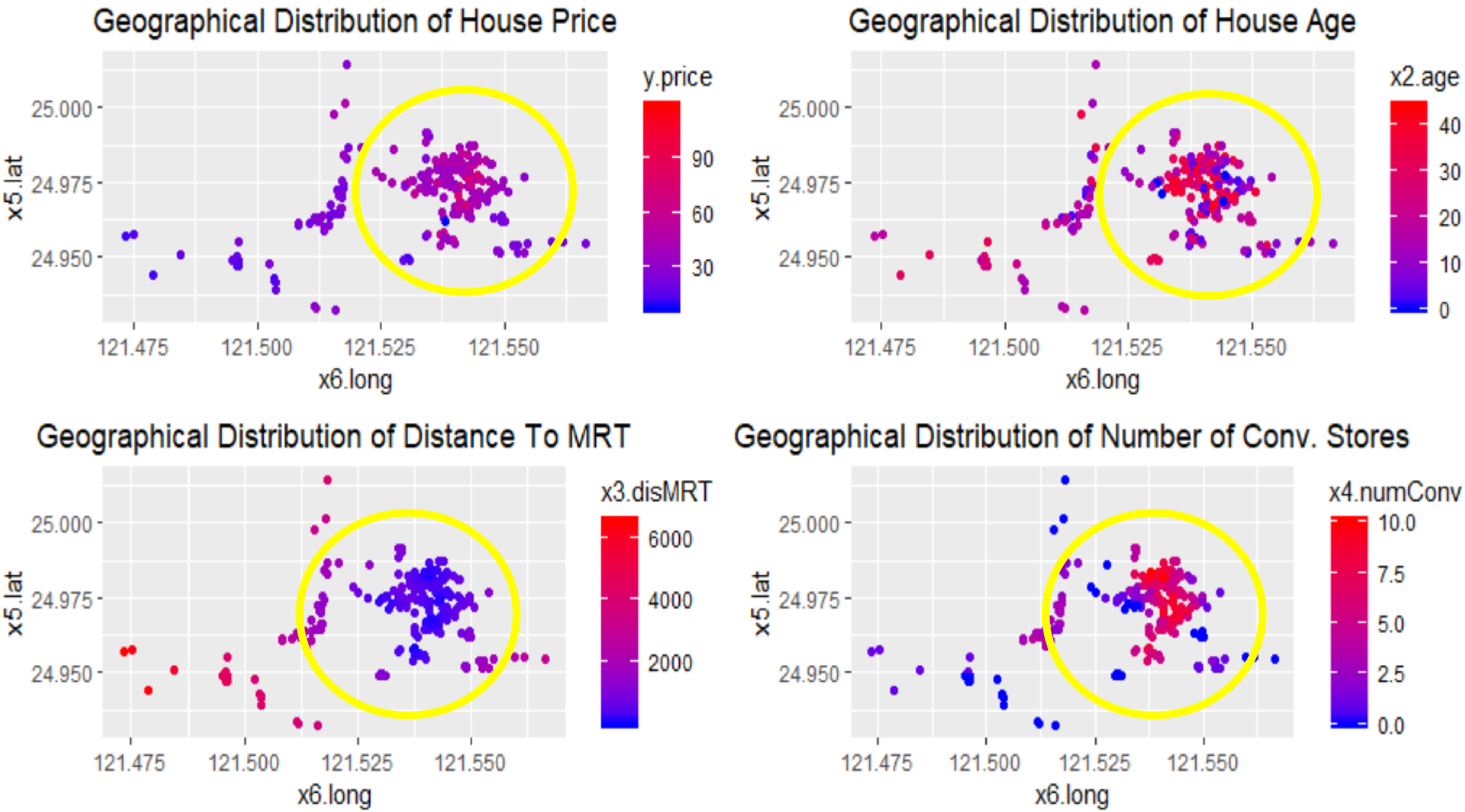
Density Distribution



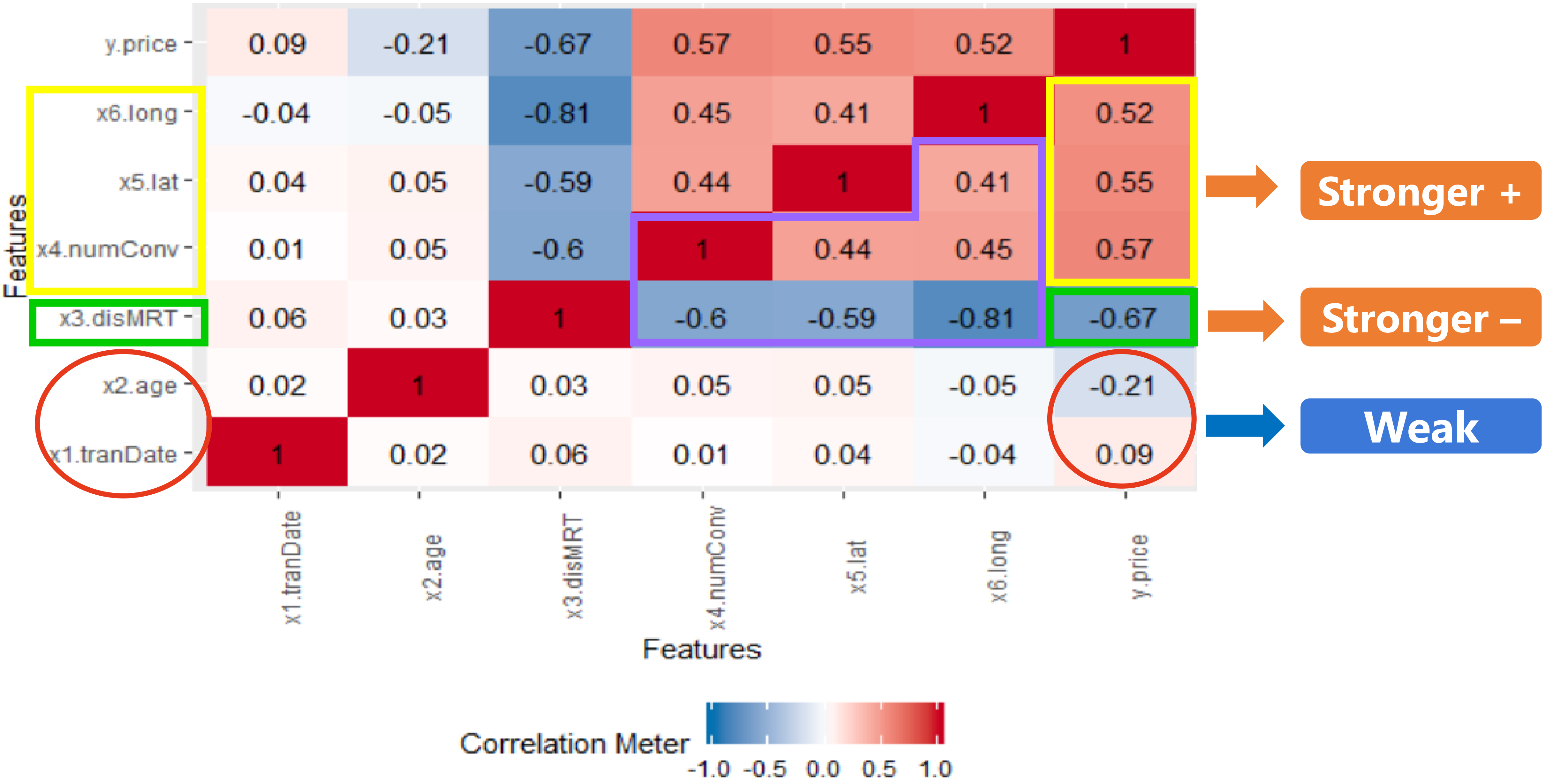
Relationships Between Variables



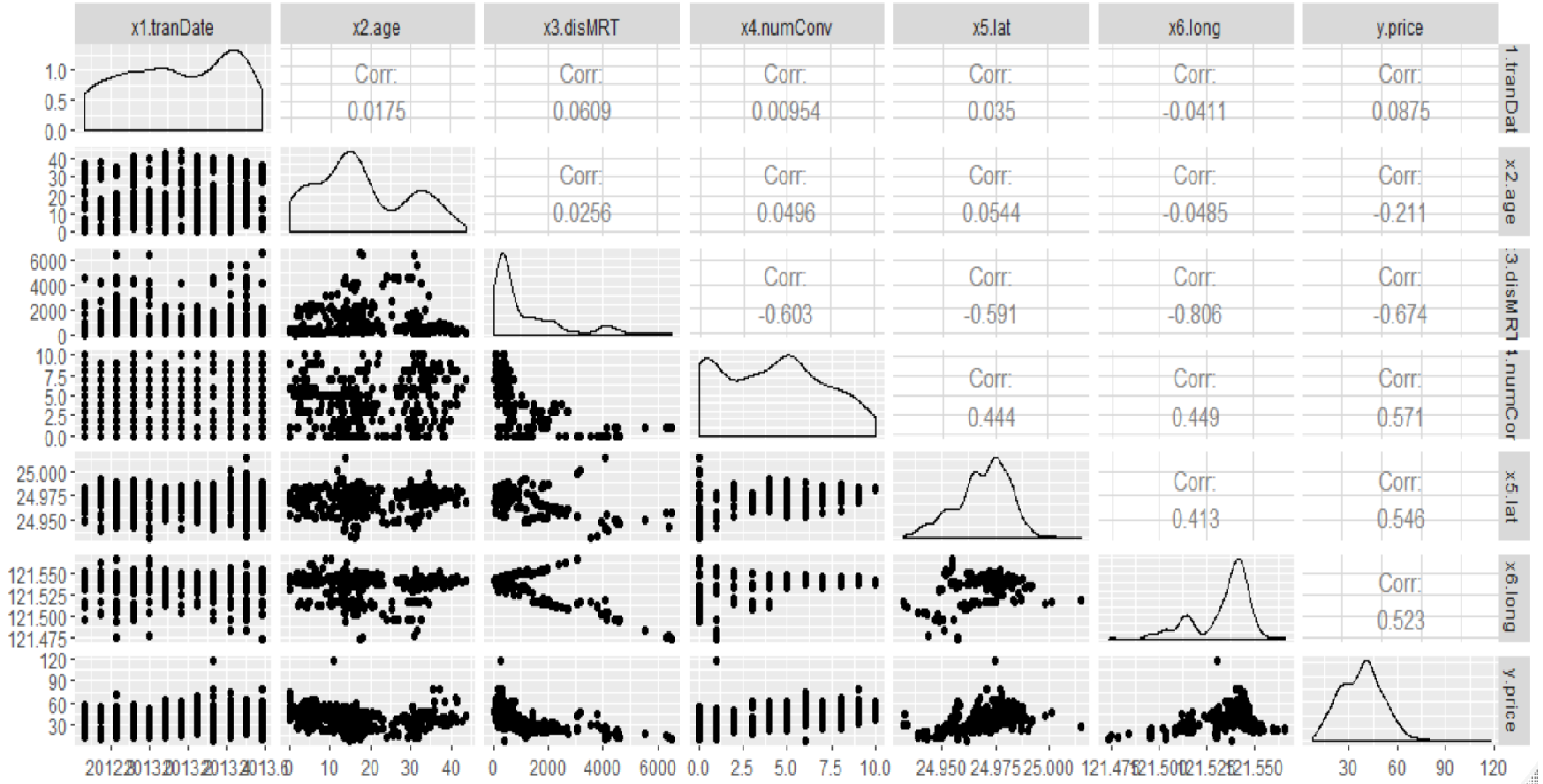
Geographical Distribution of Variables



Correlation Heatmap



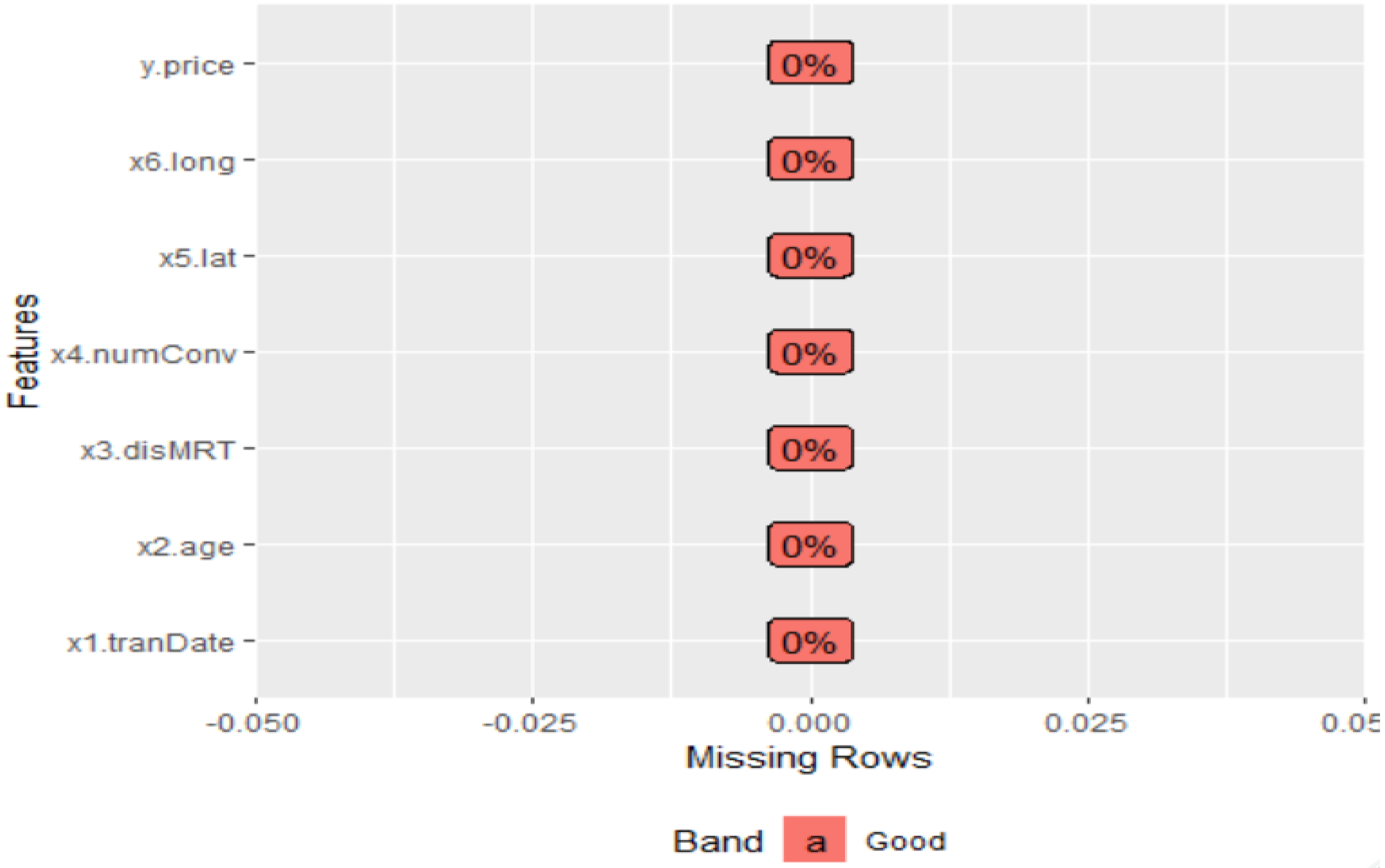
Overview of the Scatter Plot, Density Distribution, and Correlation



Data Transformation

PART FOUR

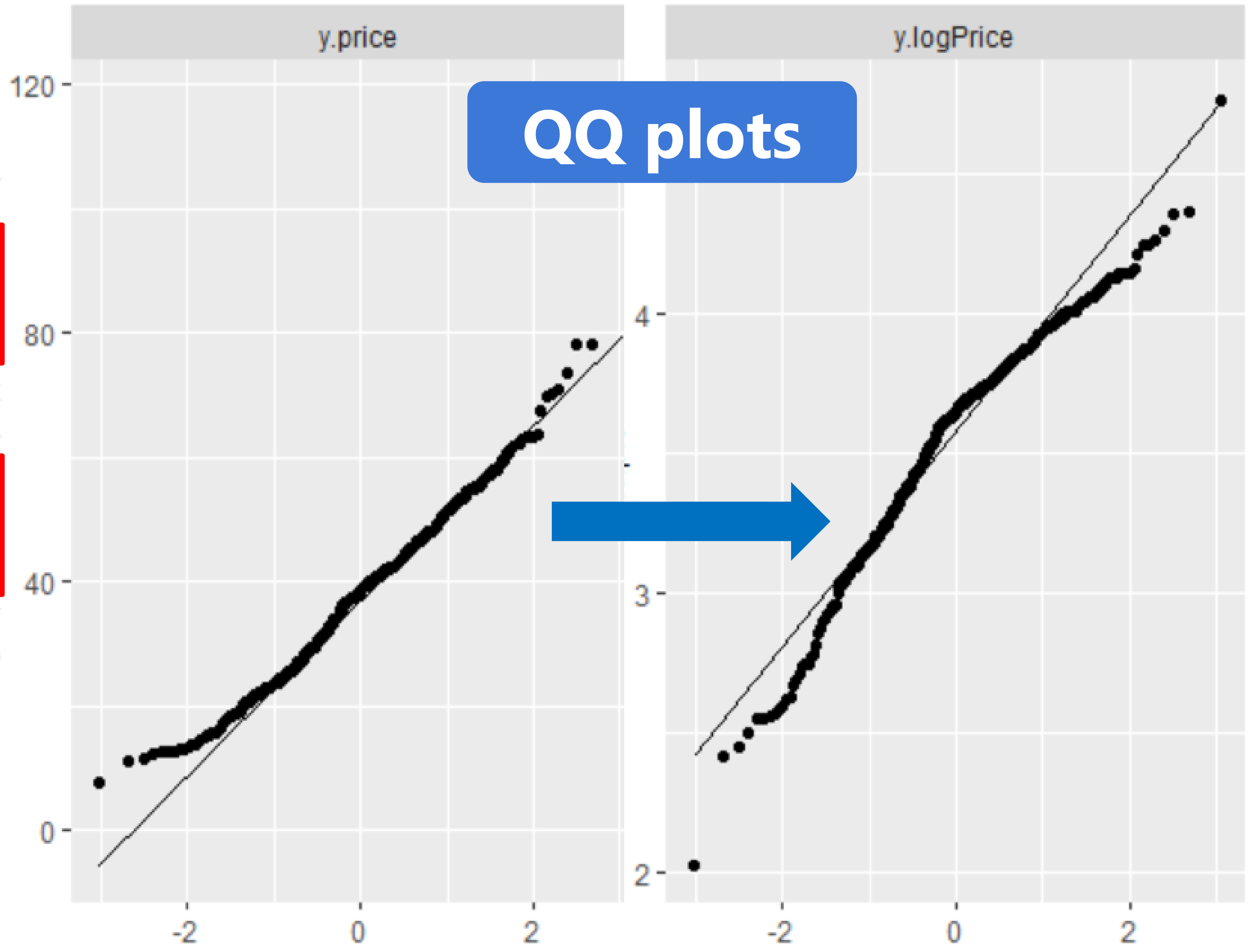
No Missing Values



Check Skewness – Transforming the Response Variable?

skewness(housing$**y.price**)
 # 0.5955128

skewness(housing$**y.logPrice**)
 # -0.7012889

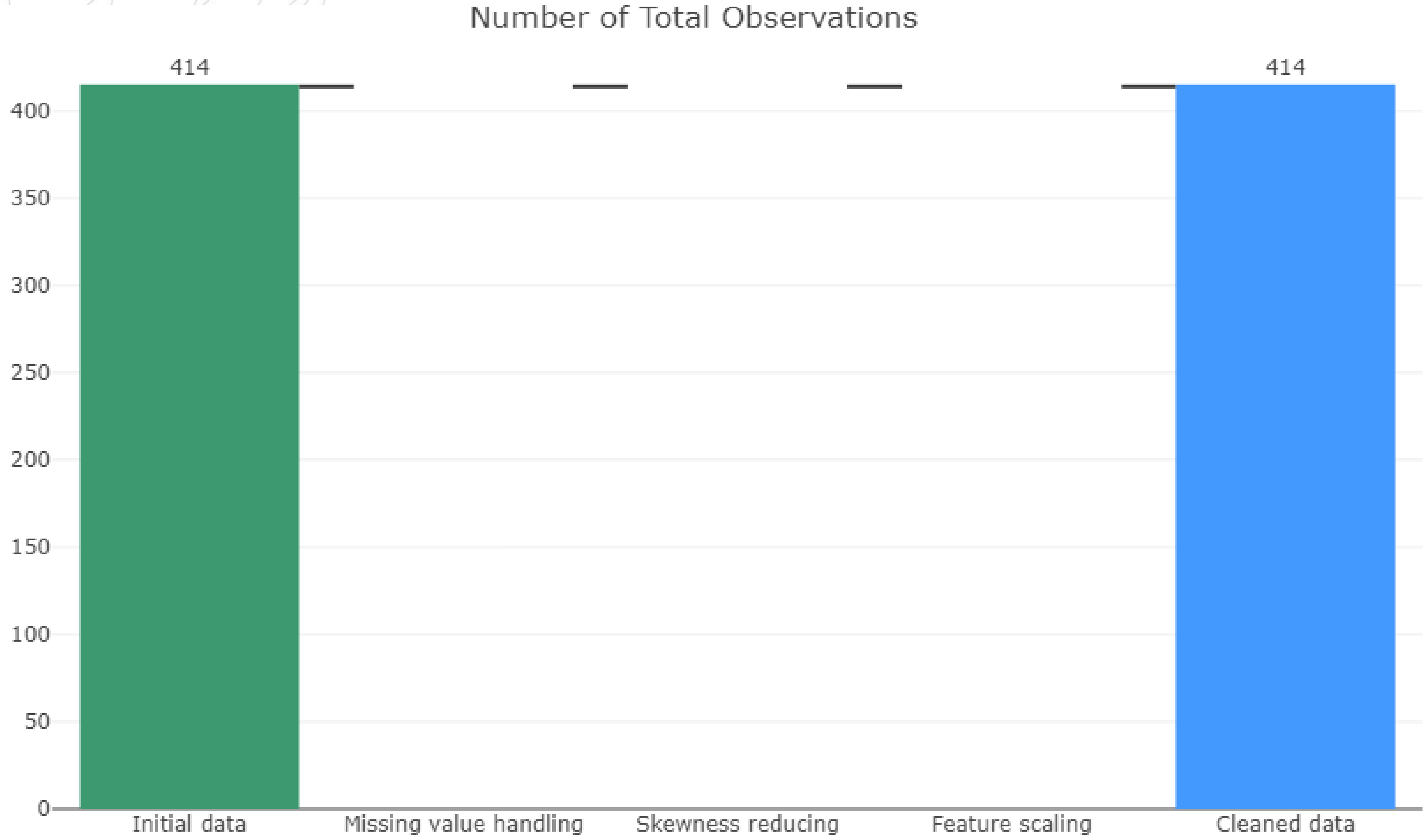


Feature Scaling - Standardization

```
head(housing)
#    x1.tranDate x2.age   x3.disMRT x4.numConv    x5.lat  x6.long y.price
# 1     2012.917   32.0   84.87882          10 24.98298 121.5402    37.9
# 2     2012.917   19.5  306.59470           9 24.98034 121.5395    42.2
# 3     2013.583   13.3  561.98450           5 24.98746 121.5439    47.3
# 4     2013.500   13.3  561.98450           5 24.98746 121.5439    54.8
# 5     2012.833    5.0  390.56840           5 24.97937 121.5425    43.1
# 6     2012.667    7.1 2175.03000           3 24.96305 121.5125    32.1
```

```
# After scaling: a new data frame named cleaned_housing
cleaned_housing <- as.data.frame(cbind(scalehousing, y.price=housing[,7]))
head(cleaned_housing)
#   x1.tranDate     x2.age  x3.disMRT x4.numConv     x5.lat    x6.long y.price
# 1   -0.823725  1.2541110 -0.7915373  2.0049816  1.1240698  0.4482199    37.9
# 2   -0.823725  0.1568964 -0.6158665  1.6654877  0.9113415  0.4006542    42.2
# 3    1.540380 -0.3873220 -0.4135150  0.3075125  1.4850633  0.6873517    47.3
# 4    1.244867 -0.3873220 -0.4135150  0.3075125  1.4850633  0.6873517    54.8
# 5   -1.119238 -1.1158725 -0.5493321  0.3075125  0.8331800  0.5922203    43.1
# 6   -1.710265 -0.9315405  0.8645401 -0.3714751 -0.4818677 -1.3566716    32.1
```

Number of Observations Changed or Dropped



Data Splitting

PART FIVE

Data Splitting

Training / Test split: 80% Training / 20% Test

```
> glimpse(housing.train)
Observations: 331
Variables: 7
$ x1.tranDate <dbl> 1.2448671, -0.2326989, -0.2326989, 0.9493540, -0.5282120, 0.9493540, 0.6538406, 1...
$ x2.age      <dbl> -0.09765740, 1.65788591, 0.86789142, -0.05376881, 0.05156378, -0.07132425, 0.1305...
$ x3.disMRT   <dbl> -0.62954984, -0.47148551, -0.45065301, -0.09229443, -0.79311596, -0.56710558, -0....
$ x4.numConv  <dbl> 0.3075125, 1.3259939, 0.3075125, -0.0319813, 2.0049816, 1.3259939, 1.3259939, 2.0...
$ x5.lat      <dbl> 1.04751987, 0.09024253, -0.48670240, 1.58659272, 1.12568141, -0.12329156, 0.23931...
$ x6.long     <dbl> 0.65933358, 0.75446504, 0.64108918, 0.04684332, 0.44952308, 0.72644687, 0.7433880...
$ y.price     <dbl> 59.6, 38.1, 37.4, 40.0, 46.6, 42.3, 48.1, 48.0, 42.3, 21.8, 55.3, 23.6, 40.5, 19....
> glimpse(housing.test)
Observations: 83
Variables: 7
$ x1.tranDate <dbl> -1.11923841, 1.24486707, 1.54038012, -1.41475147, -1.71026452, 0.94935402, -0.823...
$ x2.age      <dbl> -1.115872517, -0.396099760, 1.578886463, -0.001102515, -1.423092596, -0.633098107...
$ x3.disMRT   <dbl> -0.54933208, 0.06414048, -0.39986812, -0.58080074, -0.84026209, -0.63759367, 0.21...
$ x4.numConv  <dbl> 0.3075125, -0.0319813, -0.7109689, -1.0504627, 0.9865001, 0.9865001, -1.0504627, ...
$ x5.lat      <dbl> 0.8331800, 1.8154368, 1.0773341, 0.5165049, -0.1055642, 0.5036123, -1.3690414, 0....
$ x6.long     <dbl> 0.59222028, 0.04554015, 0.83591321, -0.14146485, 0.49904357, 0.78508955, 0.981216...
$ y.price     <dbl> 43.1, 34.3, 50.5, 37.4, 47.7, 51.6, 24.6, 38.8, 27.0, 22.1, 25.0, 55.1, 47.7, 46....
```

Modelling

PART SIX

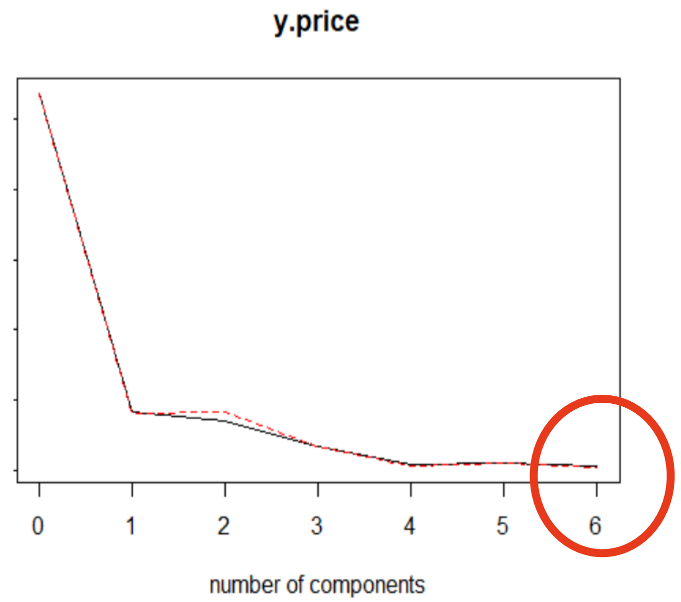
```
### Model 1.1 (lm1.1): use all of the predictors

lr1.1 <- lm(y.price~., data=housing.train)
summary(lr1.1)
# Residual standard error: 8.864 on 324 degrees of freedom
# Multiple R-squared:  0.5867,  Adjusted R-squared:  0.579
# F-statistic: 76.66 on 6 and 324 DF,  p-value: < 2.2e-16
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   38.0601     0.4878  78.019  < 2e-16 ***
x1.tranDate    1.2722     0.4994   2.548   0.0113 *
x2.age        -3.1560     0.4903  -6.437 4.39e-10 ***
x3.disMRT     -5.9223     0.9831  -6.024 4.62e-09 ***
x4.numConv     3.0211     0.6077   4.971 1.08e-06 ***
x5.lat         3.7945     0.6117   6.203 1.69e-09 ***
x6.long       -0.4261     0.8084  -0.527   0.5985
```

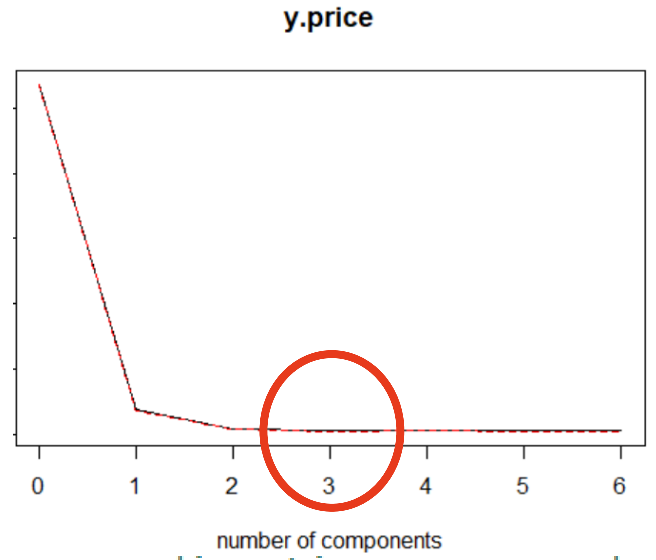
```
### Model 1.2 (lm1.2): use all predictors except for x6.long (the least significant variable)

lr1.2 <- lm(y.price~.-x6.long, data=housing.train)
summary(lr1.2)
# Residual standard error: 8.854 on 325 degrees of freedom
# Multiple R-squared:  0.5863,  Adjusted R-squared:   0.58
# F-statistic: 92.14 on 5 and 325 DF,  p-value: < 2.2e-16

# Check collinearity using vif(): variance inflation factors
vif(lr1.1)
# x1.tranDate      x2.age   x3.disMRT  x4.numConv      x5.lat     x6.long
#    1.012077    1.011098    3.701476    1.564287    1.485543    2.597717
vif(lr1.2)
# x1.tranDate      x2.age   x3.disMRT  x4.numConv      x5.lat
#    1.011745    1.009241    1.829119    1.561999    1.461529
```

```r
### Model 2.1 (ridge2.1): Ridge Regression

ridge2.1 <- glmnet(x.trainMatrix, y.train, alpha=0, lambda=grid, thresh=1e-12)
plot(ridge2.1)

# Use cross-validation to choose the tuning parameter λ
# cv.glmnet(): cross-validation function (nfolds: default is 10)
set.seed(10)
cv.ridge <- cv.glmnet(x.trainMatrix, y.train, alpha=0)
plot(cv.ridge)
bestlam_ridge2.1 <- cv.ridge$lambda.min
bestlam_ridge2.1  # 0.8983818

### Model 2.2 (lasso2.2): Lasso

lasso2.2 <- glmnet(x.trainMatrix, y.train, lambda=grid)
plot(lasso2.2)

# Use cross-validation to choose the tuning parameter λ
set.seed(10)
cv.lasso <- cv.glmnet(x.trainMatrix, y.train, alpha=1)
plot(cv.lasso)
bestlam_lasso2.2 <- cv.lasso$lambda.min
bestlam_lasso2.2  # 0.1498591
```

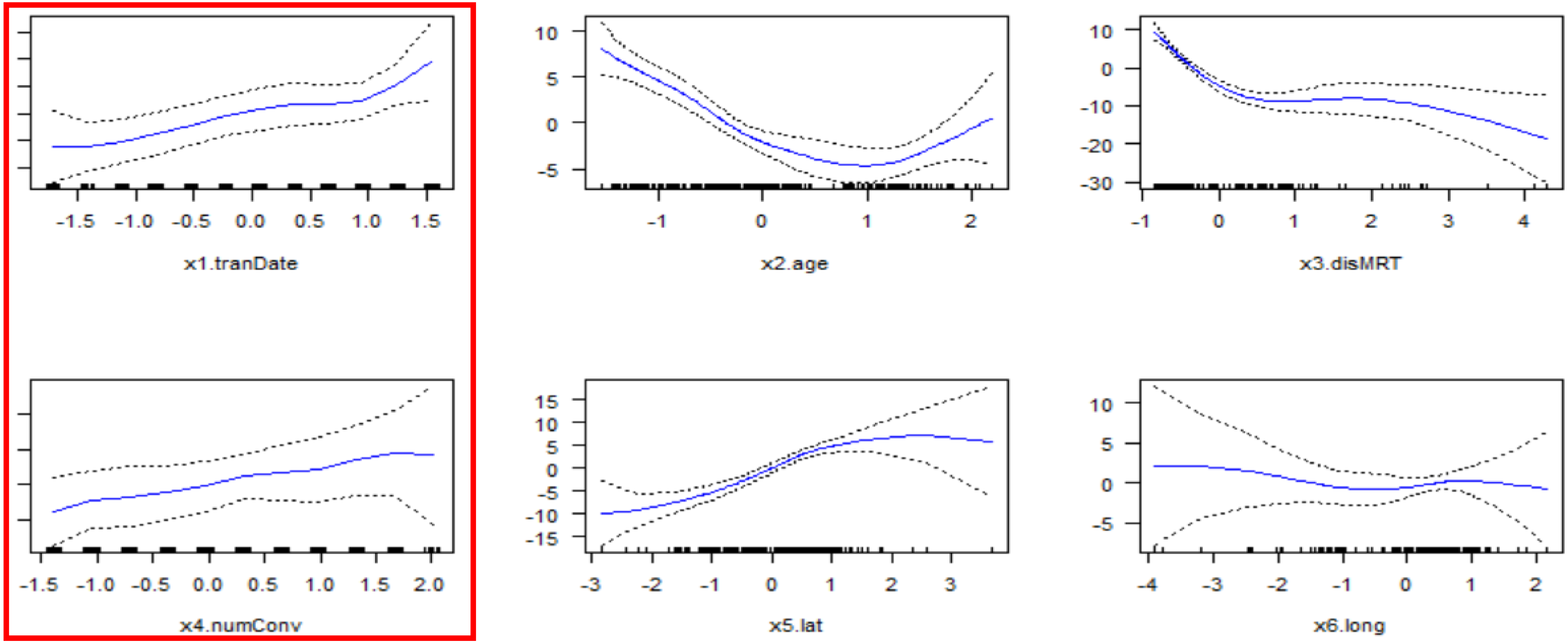
```r
### Model 3.1 (pcr3.1): Principal Components Regression

set.seed(10)
pcr3.1 <- pcr(y.price~., data=housing.train, scale=TRUE, validation="CV")
summary(pcr3.1)
validationplot(pcr3.1, val.type="MSEP")  # plot the cross-validation score
# The lowest cross-validation error occurs when M = 6 components are used.
```



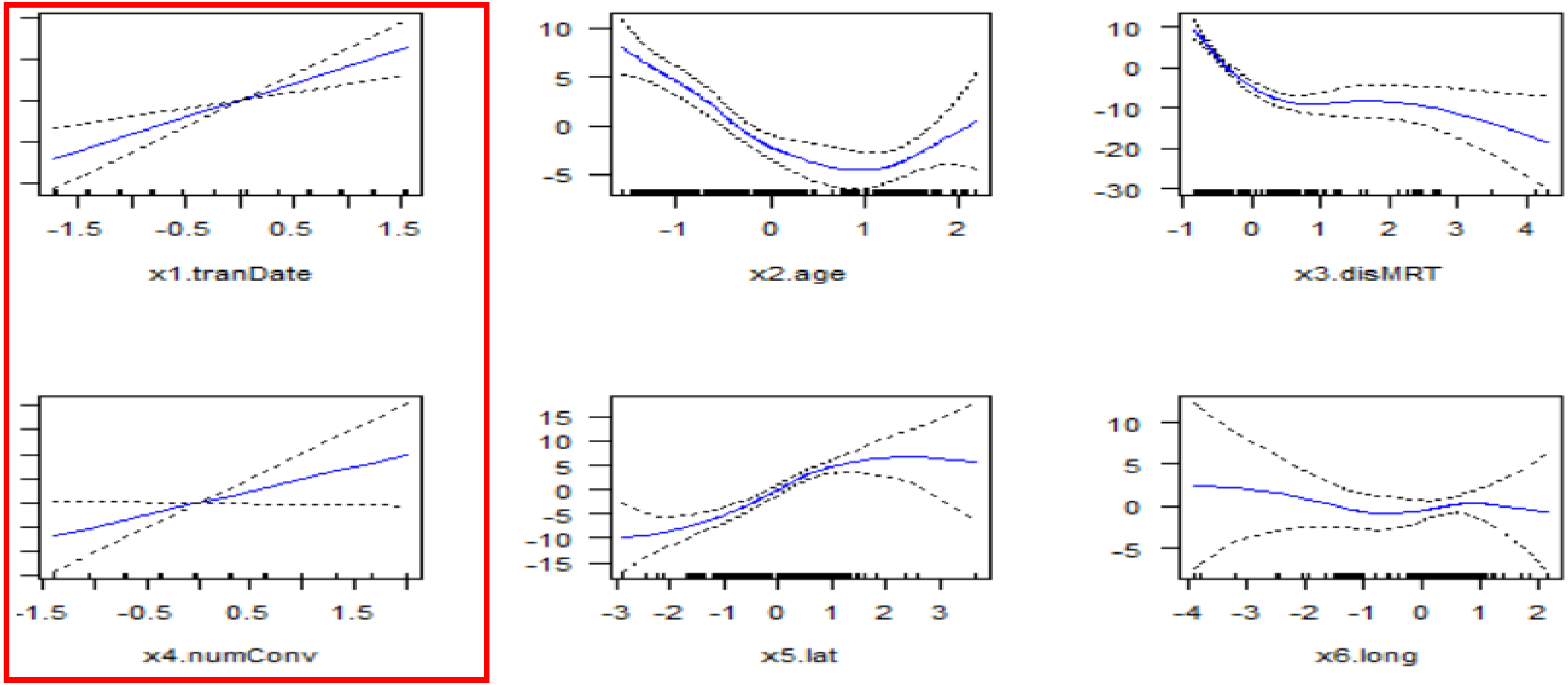

```r
### Model 3.2 (pls3.2): Partial Least Squares

set.seed(10)
pls3.2 <- plsr(y.price~., data=housing.train, scale=TRUE, validation="CV")
summary(pls3.2)
validationplot(pls3.2, val.type="MSEP")
# The lowest cross-validation error occurs when only M = 3 partial least squares directions are used.
```



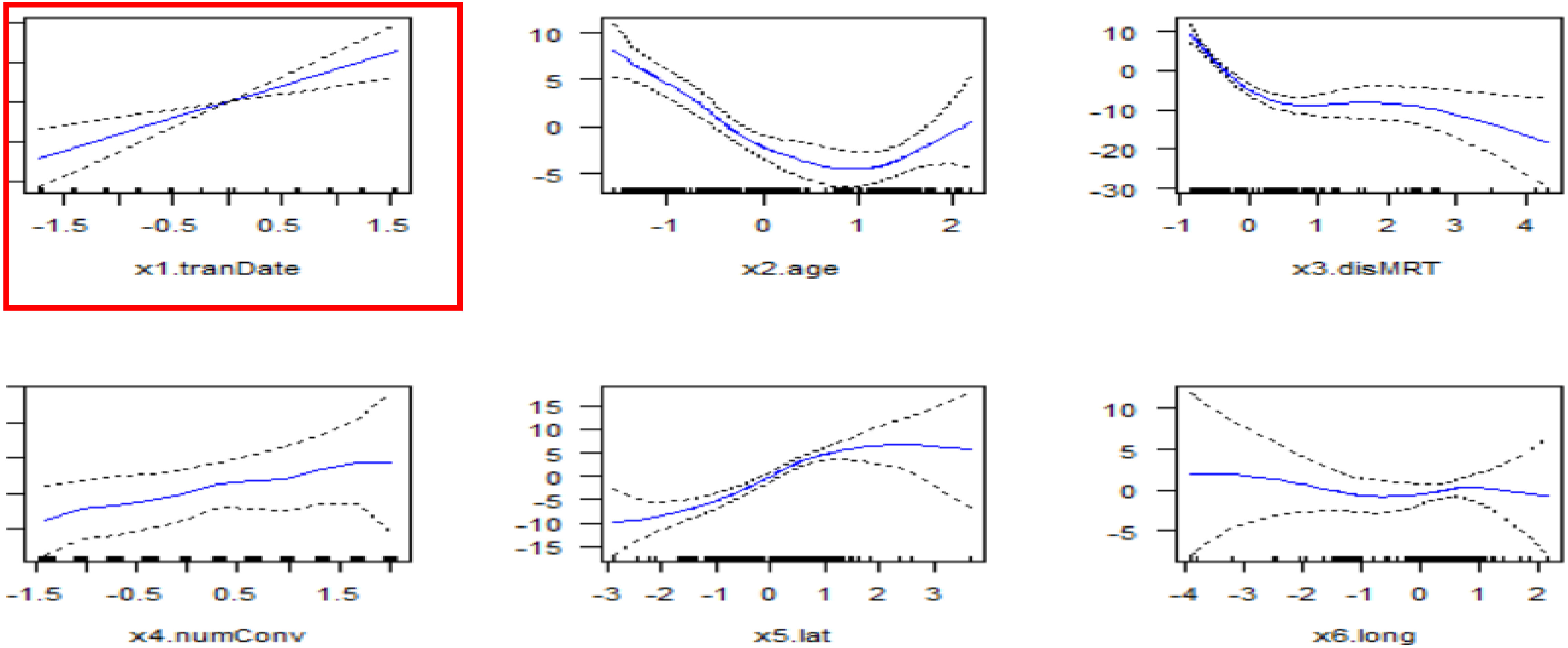
```
### Model 4.1 (gam4.1): fit a GAM using "smoothing spline" functions of all predictors

gam4.1 <- gam(y.price~s(x1.tranDate)+s(x2.age)+s(x3.disMRT)+s(x4.numConv)+
             s(x5.lat)+s(x6.long), data=housing.train)
summary(gam4.1)
plot(gam4.1, se=TRUE, col="blue")
# the function of "x1.tranDate" and "x4.numConv" looks rather linear
```



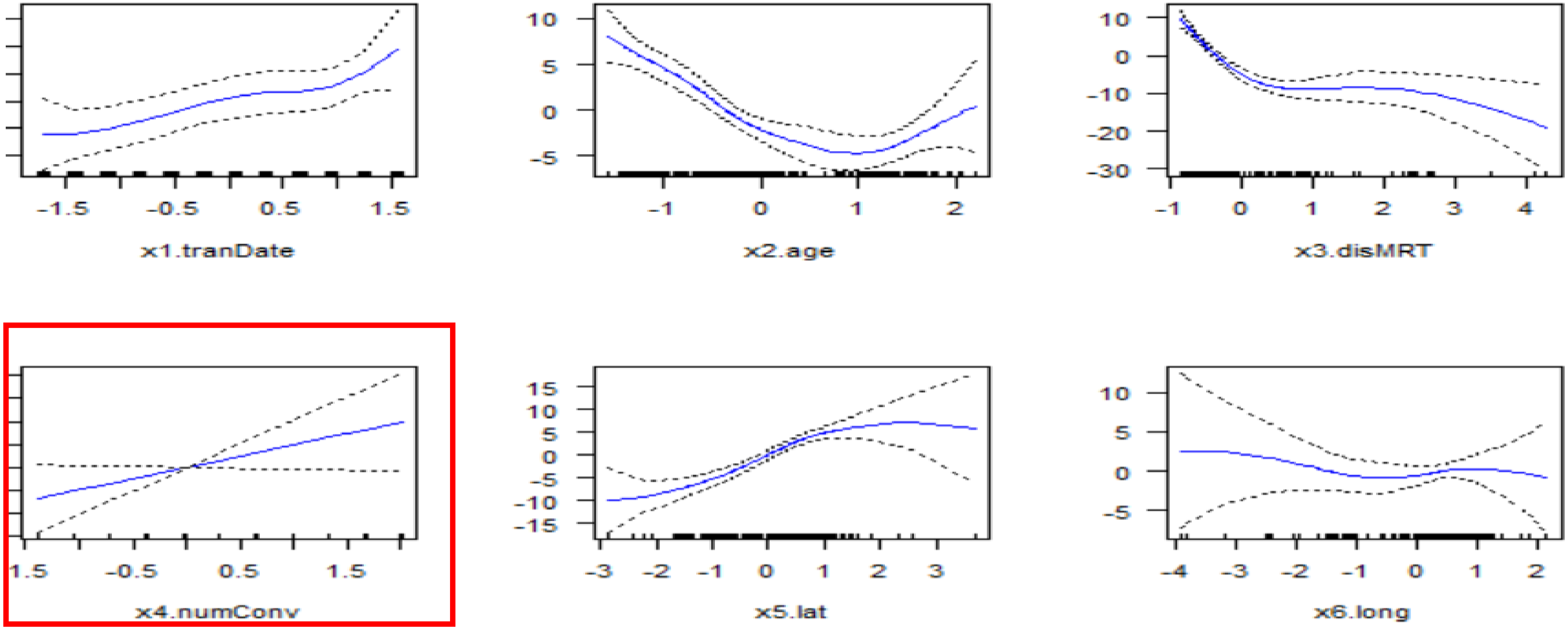
```
### Model 4.2 (gam4.2): fit a GAM using linear functions of "x1.tranDate" and "x4.numConv"
#                       and smoothing spline functions of the other predictors

gam4.2 <- gam(y.price~x1.tranDate+s(x2.age)+s(x3.disMRT)+x4.numConv+
                s(x5.lat)+s(x6.long), data=housing.train)
summary(gam4.2)
plot(gam4.2, se=TRUE, col="blue")
```



```
### Model 4.3 (gam4.3): fit a GAM using a linear function of "x1.tranDate"
#                       and smoothing spline functions of the other predictors

gam4.3 <- gam(y.price~x1.tranDate+s(x2.age)+s(x3.disMRT)+s(x4.numConv)+
                s(x5.lat)+s(x6.long), data=housing.train)
summary(gam4.3)
plot(gam4.3, se=TRUE, col="blue")
```



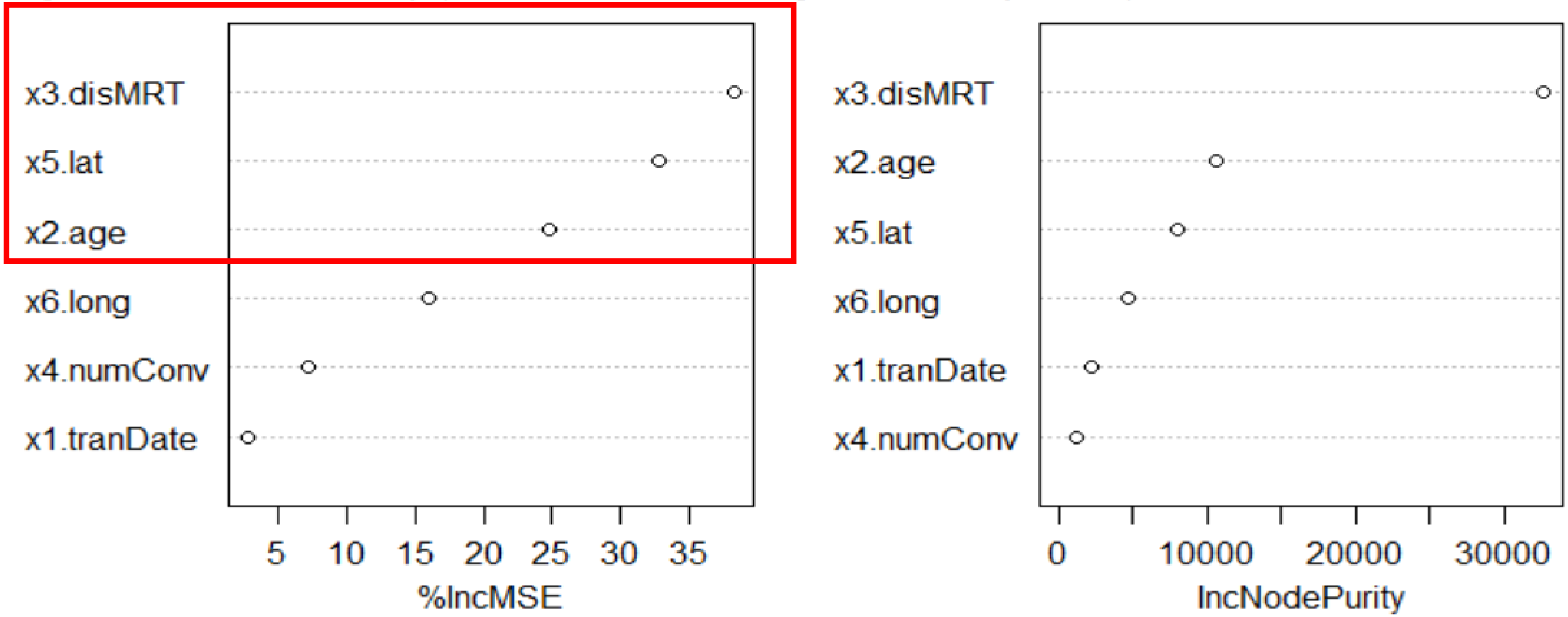
```
### Model 4.4 (gam4.4): fit a GAM using a linear function of "x4.numConv"
#                       and smoothing spline functions of the other predictors

gam4.4 <- gam(y.price~s(x1.tranDate)+s(x2.age)+s(x3.disMRT)+x4.numConv+
              s(x5.lat)+s(x6.long), data=housing.train)
summary(gam4.4)
plot(gam4.4, se=TRUE, col="blue")
```



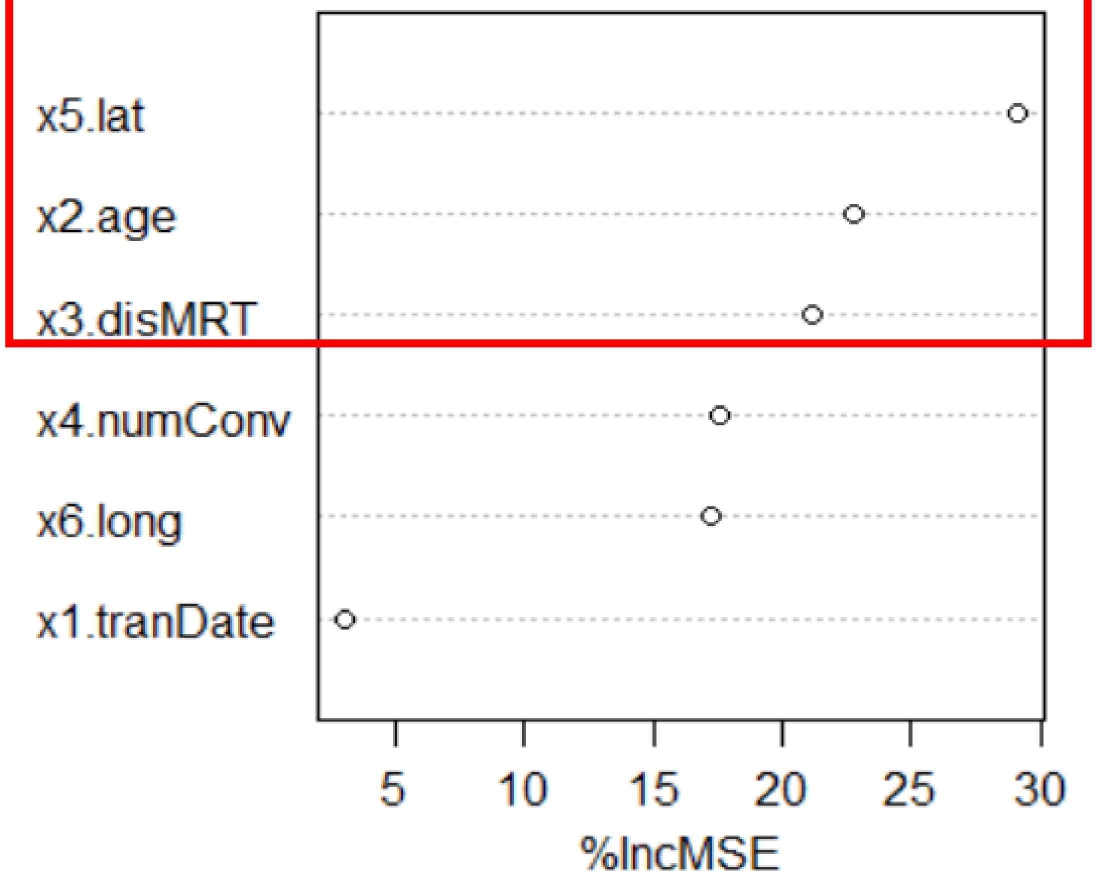
ANOVA Test: Which model is the best?

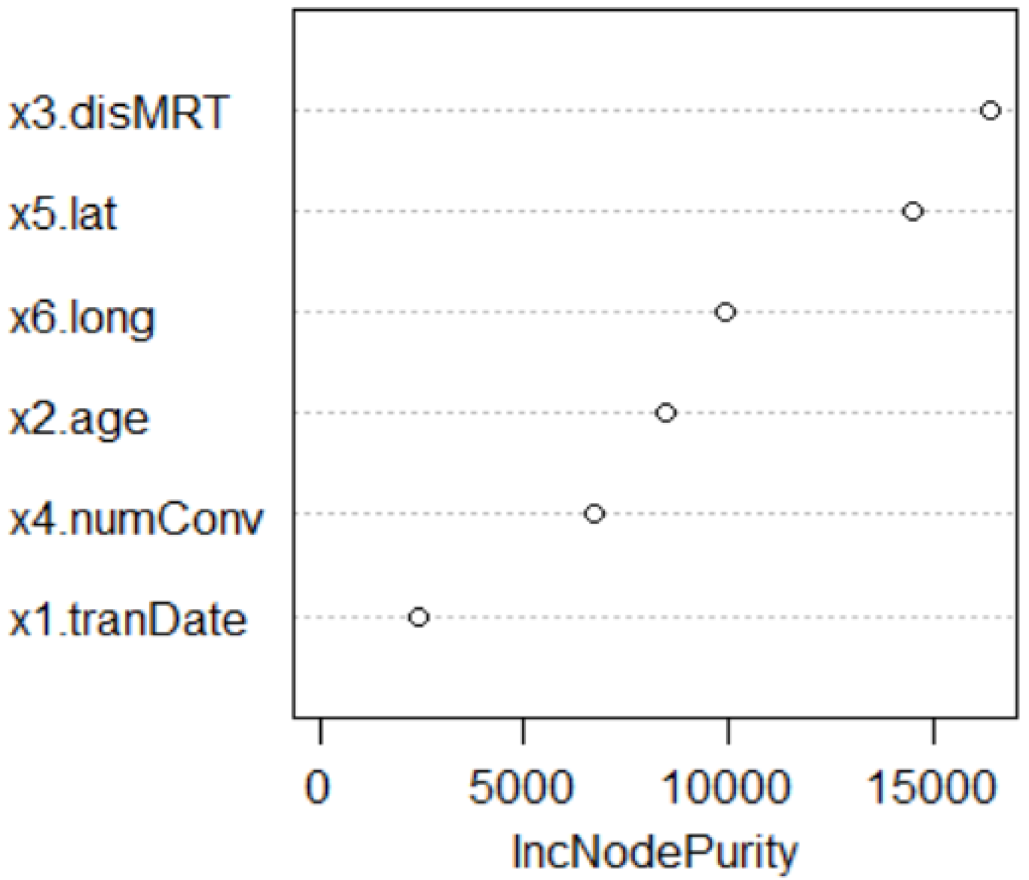
```
> anova(gam4.1, gam4.2, gam4.3, gam4.4, test="F")
Analysis of Deviance Table

Model 1: y.price ~ s(x1.tranDate) + s(x2.age) + s(x3.disMRT) + s(x4.numConv) +
    s(x5.lat) + s(x6.long)
Model 2: y.price ~ x1.tranDate + s(x2.age) + s(x3.disMRT) + x4.numConv +
    s(x5.lat) + s(x6.long)
Model 3: y.price ~ x1.tranDate + s(x2.age) + s(x3.disMRT) + s(x4.numConv) +
    s(x5.lat) + s(x6.long)
Model 4: y.price ~ s(x1.tranDate) + s(x2.age) + s(x3.disMRT) + x4.numConv +
    s(x5.lat) + s(x6.long)
  Resid. Df Resid. Dev          Df Deviance          F    Pr(>F)
1       306      18276
2       312      18491 -6.0000e+00 -215.345     0.6009    0.7296
3       309      18441  3.0000e+00   49.827     0.2781    0.8412
4       309      18335  2.9689e-05  106.650 60147.1809 3.948e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```r
### Model 5.1 (bag5.1): Bagging - A special case of a random forest with m = p

set.seed(10)
# mtry=6: all 6 predictors should be considered for each split of the tree (bagging)
bag5.1 <- randomForest(y.price~., data=housing.train, mtry=6, importance=TRUE) # ntree=500
```



```
### Model 5.2 (rf5.2): Random Forests

set.seed(10)
rf5.2 <- randomForest(y.price~., data=housing.train, importance=TRUE)
# Default: "p/3" variables (mtry=2, ntree=500)
```





```r
### Model 5.3 (bst5.3): Boosting

set.seed(10)
bst5.3 <- gbm(y.price~., data=housing.train, distribution="gaussian", n.trees=5000,
              interaction.depth=4, cv.folds = 10) # The default shrinkage parameter λ is 0.001
summary(bst5.3)  # the relative influence statistics
# x3.disMRT, x2.age, and x5.lat are the most important variables in this model

# Tuning a gbm model: early stopping
# Avoid overfitting: stop the training procedure once the performance stops improving
# beyond a certain number of iterations.
# Determine the optimum number of iterations
ntree_opt_cv <- gbm.perf(bst5.3, method="cv")
ntree_opt_cv  # 86
```

```
> summary(bst5.3)  # the relative influence statistics
                    var   rel.inf
x3.disMRT     x3.disMRT 35.735538
x2.age           x2.age 21.495335
x5.lat           x5.lat 20.568601
x6.long         x6.long  9.740266
x1.tranDate x1.tranDate  7.910174
x4.numConv   x4.numConv  4.550086
```

```
### Model 6.1 (svm6.1): Support Vector Machine - Regression

# Tuning the model by varying values of maximum allowable error and cost parameter
set.seed(10)
svm6.1 <- tune(svm, y.price~., data=housing.train, kernel = "radial",
               ranges = list(epsilon = seq(0,1,0.1), cost = 1:50))
svm6.1
plot(svm6.1) # Map tuning results
svm6.1_best <- svm6.1$best.model  # Find out the best model
```

```
> svm6.1_best

Call:
best.tune(method = svm, train.x = y.price ~ ., data = housing.train,
 ranges = list(epsilon = seq(0,
    1, 0.1), cost = 1:50), kernel = "radial")


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  5
      gamma:  0.1666667
    epsilon:  0.1


Number of Support Vectors:  258
```

Evaluation & Conclusion

PART SEVEN

Actual VS. Predicted

```
> head(housing.test)[7:18]
   y.price lr1.1_pred lr1.2_pred ridge2.1_pred lasso2.2_pred pcr3.1_pred
5     43.1   47.24933   47.32899      46.99748      47.18470    47.24933
15    34.3   47.28675   47.39237      47.10749      47.00660    47.28675
16    50.5   38.98889   39.24219      39.36362      39.22052    38.98889
18    37.4   38.55001   38.28957      38.09747      38.51641    38.55001
20    47.7   47.71888   47.62086      47.18693      47.50773    47.71888
22    51.6   49.59867   49.71828      49.32855      49.30233    49.59867
   pls3.2_pred gam4.4_pred bag5.1_pred rf5.2_pred bst5.3_pred svm6.1_pred
5     47.69216    49.89302    50.08994   50.35590    48.87671    54.61754
15    48.19191    41.24944    33.29943   35.79252    37.35752    33.78017
16    39.05754    44.60781    48.07467   45.81815    43.06665    36.00115
18    37.82729    39.28048    36.51833   36.05265    36.27422    29.63891
20    47.55934    52.81183    48.11222   48.70960    49.80282    51.52078
22    49.75387    51.11380    56.17460   55.74879    61.11446    52.48325
```

Error Measures

```r
# MSE: Mean Squared Error
mse <- function(predictions, actual) {
  mean((predictions - actual) ^ 2 )
}
# RMSE: Root Mean Squared Error
rmse <- function(predictions, actual) {
  sqrt(mean((predictions - actual) ^ 2 ))
}
# Mean Absolute Error
mae <- function(predictions, actual) {
  mean(abs(predictions - actual))
}
# MAPE: Mean Absolute Percentage Error
mape <- function(predictions, actual) {
  mean(abs((predictions - actual)/actual))
}
# For models error summary
modelsSummary <- data.frame(Model = character(), MSE = double(), RMSE = double(),
                            MAE = double(), MAPE = double(), Description = character())
```

Summary of Test Error

Linear

	Model	MSE	RMSE	MAE	MAPE	Description
1	lr1.1	82.71355	9.094699	6.833200	0.2191385	all predictors
2	lr1.2	82.46512	9.081031	6.842128	0.2194620	all predictors - x6.long
3	ridge2.1	80.40068	8.966642	6.745362	0.2117971	
4	lasso2.2	81.53137	9.029472	6.803126	0.2158941	
5	pcr3.1	82.71355	9.094699	6.833200	0.2191385	6 components
6	pls3.2	84.22834	9.177600	6.883452	0.2170109	3 pls directions

Non-linear

7	gam4.4	51.65687	7.187271	5.191150	0.1578802	linear: x4.numConv, smoothing: others
8	bag5.1	46.21509	6.798168	4.896822	0.1472833	mtry=6, ntree=500
9	rf5.2	39.12631	6.255103	4.406327	0.1360753	mtry=2, ntree=500
10	bst5.3	42.52357	6.521010	4.678538	0.1376725	interaction.depth=4, optimal ntree=86
11	svm6.1	44.40091	6.663401	4.825837	0.1442710	radial kernel, cost=5, epsilon=0.1

THANK YOU